Doctoral theses at NTNU, 2022:134

Muhammad Mudassar Yamin

Modelling and Analyzing Attack-Defense Scenarios for Cyber-Ranges

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of Science and Technology

Muhammad Mudassar Yamin

Modelling and Analyzing Attack-Defense Scenarios for Cyber-Ranges

Thesis for the Degree of Philosophiae Doctor

Gjøvik, May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

ISBN 978-82-326-5470-3 (printed ver.)
ISBN 978-82-326-6809-0 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2022:134

Printed by NTNU Grafisk senter

Contents

To my family, friends and well wishers

Declaration of Authorship

I, Muhammad Mudassar Yamin, hereby declare that this thesis and the work presented in it are entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed:

(Muhammad Mudassar Yamin)

Date:

Abstract

Rome was not built in a day, but it was burnt to the ground in only six. Wood naturally catches fire, and without adequate engineering, fireproof houses and training for firefighters, destruction caused by fire is inevitable. In the 21st century, our modern world is built not on wood but on a digital infrastructure that was proposed in the 20th century with very little thought to security. This has resulted in a countless number of incidents in which that infrastructure has been compromised, from hospitals serving critically ill patients to gas pipelines providing necessary heating to people living in adverse climate conditions.

The current state of affairs is unacceptable, and serious efforts are needed to design and build a secure digital world and train individuals to use and operate it securely. Engineers and scientists design road infrastructure with great safety measures, but traffic accidents still happen. Indeed, they remain one of the leading causes of death in the world, and most traffic accidents are caused by human error or negligence. Similarly, the digital infrastructure can be designed and deployed securely, but its overall security and safety depend upon the humans who are operating and using it. Therefore, there is a great need to train individuals to operate the digital infrastructure in a secure manner.

Multiple efforts are being made to provide this training. These efforts include cybersecurity education and training based on different pedagogical methods involving classroom teaching, workshops, seminars, conferences and hands-on training. However, the effects of these efforts are not yet visible, as we experience ever-increasing damage caused by cyber-attacks. Traditionally, most cybersecurity awareness and training has been achieved through classrooms and workshops. Little focus has been on hands-on cybersecurity exercises. This is because designing and deploying infrastructure to deliver realistic hands-on exercises is a resource-

intensive, complex and difficult task that requires considerable manual technical expertise. This makes the training very expensive and the process error-prone and difficult to standardize.

In order to solve these issues, different researchers have tried to remove inefficiencies in cybersecurity exercises by automating different phases of the exercises with limited success. Some efforts yielded very specific testbed-related artifacts, which were only applicable to that specific testbed, while other efforts lacked the complexity required for realistic cybersecurity exercises. Moreover, there is a lack of consensus among the community on defining the training scenarios that can be used in such exercises. Therefore, standard specifications of scenarios that can be executed in a cybersecurity exercise environment are needed. In this work, I attempt to overcome and address these issues by enhancing efficiency, realism and standardization with a novel method of modeling and executing cybersecurity exercise scenarios in a cybersecurity exercise environment, or *a cyber range*.

This is achieved through the development of a domain-specific language that is used to model and specify the technical requirements for cybersecurity exercises at an abstract level. The model of the exercise scenario is formalized and verified through logic programming, and then the technical requirements are translated into operational artifacts through an orchestrator. The operational artifacts contain an exercise infrastructure with vulnerabilities, traffic generators and attack/defense agents that can exploit or defend those vulnerabilities at an operational level in a cyber range. The proposed system goes beyond the state of the art by overcoming many inefficiencies in cybersecurity exercise scenario modeling and deployment, making their execution efficient, realistic and computationally repeatable. The proposed artifacts and solutions were tested in Norway's national cybersecurity competitions, university classrooms and other cybersecurity exercises with positive results.

Acknowledgement

Many people helped and supported me in conducting this research. First, I would like to thank my parents and siblings for their support and understanding as I pursued this endeavor. Second, I would like to thank my principal supervisor, Professor Basel Katt, who leads the research activities in the Norwegian Cyber Range, and the Co-supervisor, Slobodan Petrovic, for their guidance and support. Furthermore, I would like to express my gratitude to Espen Torseth, who is responsible for technical and administrative tasks at the Norwegian Cyber range and helped me greatly in this research work by sharing his experience and valuable suggestions.

Additionally, I would like to thank the Norwegian Cybersecurity challenge for providing access to the Norwegian national team to conduct this research activity. Furthermore, I am grateful to the COINS Research School for Computer Information Security for providing funding for Ph.D. courses and granting access to a valuable research network that helped me make useful connections to share the research with a broader audience.

I would like to thank my friends and colleagues at NTNU Gjovik, who provided support and made beautiful memories while pursuing the Ph.D. research. Finally, I would like to thank the European Information Security Agency for providing me with the opportunity to conduct research on their behalf and to work with experts from 20 European countries to develop a common roadmap for the European Cybersecurity Challenge.

Chapter 1

Introduction

1.1 Synopsis

Our world is rapidly changing into a digital-first society, and the ongoing pandemic has increased the pace of this digital transformation even more. In a hyper-connected world, this digital society brings opportunities and challenges. These opportunities can help humanity through better connectivity, idea sharing and a better understanding of each other. However, the same technologies empowering the digital world can be misused by different state and non-state actors to cause an unimaginable amount of harm to society, from the manipulation of elections to attacks on a country's critical infrastructure.

To avoid this danger to society, we need awareness of and training on such attacks or threats at the individual and organizational levels. Such awareness could be provided through cybersecurity skills improvement programs. Just as individuals need gyms where they can exercise and maintain their physical fitness, for a cyber-safe and secure society we need platforms for individuals and organizations to practice their cybersecurity skills. Such platforms may be cyber ranges that can provide continuous training and a learning environment for teaching cybersecurity skills. However cyberthreats are evolving constantly, and there is a need to model and analyze those threats and provide realistic scenarios that can be practiced in cyber ranges to address those threats.

Different threat actors can have different objectives that can affect the security of a country at multiple levels. Some threat actors target the society with propaganda and social media manipulation rather than targeting the operational infrastructure. Meanwhile, some threat actors exploit technical vulnerabilities and

target the national infrastructure. Analyzing and modeling their attacks is becoming a necessity for training individuals and organizations to respond appropriately in a professional manner, with the goal of avoiding disasters and making society more resilient against cyberattacks.

1.2 Research Context

Cybersecurity has become one of the fundamental pillars of countries' national security. Indeed, the security of the societal functions of a country rests on this pillar. Due to the changing nature of modern warfare, cybermethods of waging war are being increasingly utilized compared to kinetic methods. This fundamental shift in the nature of warfare has led to the development of increased cyber capabilities in many countries. Realizing the evolving nature of warfare, the government of Norway was one of the first governments to develop a national cybersecurity policy back in 2003 [1]. This led to the development of different entities that can respond to attacks on Norwegian digital infrastructure. However, considering the significant increase of cyberattacks in modern times, in its 2018 national cybersecurity policy [2] the government of Norway included the cyber range as a key measure for training individuals and organizations to handle evolving cyberattacks.

Cyber ranges can provide the environment for the execution of different types of cybersecurity exercises that may or may not need computer-added features. For example, cyber ranges can be used to execute discussion-based exercises, which involve seminars, workshops and tabletop exercise scenarios. These kinds of scenarios do not require computer infrastructure for execution. In such exercises, scenarios are discussed by experts and mangers, and their inputs are used for solving a cybersecurity incidents. Cyber ranges can also be used for executing operational exercises, including drills, functional exercises and full-scale exercises [3]. Drills are used for regular testing and verification of different components present within an organization, such as testing an intrusion detection system with malware samples. Meanwhile, functional exercises include CTF *Capture The Flag*, *Attack Defense* and *Red versus blue* scenarios. These kinds of scenarios are used for training and evaluating offensive and defensive skills within an organization. On the other hand, *full-scale* exercises combine concepts from tabletop exercises and functional exercises to create a coherent exercise environment for both managers and operational-level analysts.

The Norwegian Cyber Range (NCR) project was launched in 2018 [4] with the main aim to establish a center of excellence in Norway for organizing and providing cybersecurity training and education to different public and private actors. One of the goals of the project was to combine strategic, tactical and operational levels of cybersecurity and execute full-scale cybersecurity exercises. This research work

focused on the operational layer of cybersecurity exercises within NCR, and the next section will highlight the motivation for this endeavor.

1.3 Motivation

The shortage of cybersecurity skills is a well-known and well-documented problem [5]. Consequently, a considerable amount of work is being carried out to address this problem. Most of this work is related to developing exercises and training materials to educate as many individuals as possible. However, these are mostly discussion-based scenario exercises [6]. While they have the potential to increase cybersecurity awareness and address the current challenges that different stakeholders face, there is a lack of technical cybersecurity training and education, as highlighted by ENISA *(European Network and Information Security Agency)* in its 2015 report on the status of cybersecurity exercises [6]. Similarly, in its 2021 report, ENISA indicated that there are some Europeans countries that are lacking the capability to organize operation-based cybersecurity exercises [7].

Such operation-based exercises can be provided in environments that represent actual IT systems, which are vulnerable to cyberattacks. These IT systems can be represented in a digital emulated environment, such as a cyber range. They provide an actual representation of an organization to execute realistic cybersecurity incidents in a safe manner and provide people the opportunity to learn hands-on skills to handle such incidents. Organizing such exercises takes months to years [6, 8], as it involves a complex cybersecurity exercise life-cycle. Existing solutions [9, 10] are addressing these inefficiencies by automating the preparation, deployment and dry run of cybersecurity exercises to a certain extent. However, most of the existing solutions are currently focused on certain phases of the cybersecurity exercise life-cycle. Little attention has been given to the dry run and execution phases, while most of the focus is on environment creation [11, 12].

In terms of environment creation, the main challenge is that there is a significant lack of methods for representing complex IT systems in a realistic manner. Most of the existing solutions focus on small-scale CTF competitions [12, 13] with very little network complexity. The second problem related to environment creation is that the environments created are highly inflexible after deployment [8, 14]. Cybersecurity exercise scenarios need to be flexible and adaptable so they are readily available in an efficient manner with changing scenario requirements. Regarding exercise execution, there is a lack of automated adversaries [15] that can add friction to the exercise to provide realistic cybersecurity training in the event human adversaries are not present.

Finally, very little effort has been made to formally model and analyze the exer-

cise scenarios that integrate both the static environment as well as the dynamic behavior of attackers and defenders when running within the environment. Using formal methods for modeling and analyses enables the verification and validation of exercise scenarios in various phases of the exercise life-cycle. For example, it can help to design and build a cybersecurity exercise environment with few errors [11] and make it possible for the exercise designer to verify and proof various scenario properties during execution.

There is a need to develop algorithms and tools to overcome the aforementioned problems in the operation-based cybersecurity exercise life-cycle. Such solutions should enable the design, creation and execution of cybersecurity exercise scenarios in an efficient, repeatable, realistic and effective manner.

1.4 Aim and Scope

As practical hands-on cybersecurity skills are in high demand [5, 16], operation-based cybersecurity exercises were the focus in this research. The 2015 Report on "National and international cyber-security exercises" [6] by ENISA suggested that the number of cybersecurity exercises conducted is far below the required number, and due to the increased awareness of cybersecurity problems, policymakers are demanding more cybersecurity exercises. From 2015 to 2020, the number of CTFs organized on CTFtime doubled [17], which indicates the growing demand for operation-based cybersecurity exercises, and the number is expected to grow even further in the near future.

The same report by ENISA also indicated that 81 percent of the exercises that are conducted are simulations, table-tops and workshops, compared to 11 percent of operation-based red/blue team exercises. This indicates significant shortcomings in operation-based cybersecurity exercises. Therefore, the research focused efforts on identifying inefficiencies in conducting operation-based cybersecurity exercises and developing tools and techniques to tackle those inefficiencies, as indicated in Figure 1.1. Discussion-based exercises are out of the scope of this study. In our experiments and case studies, we focused on university-organized exercises and labs. Hence, multinational and very large-scale exercises, e.g., the scale of lockedshield [6], are beyond our current scope.

1.5 Research Questions

To understand the objectives and goals of this research, four research questions were formulated.

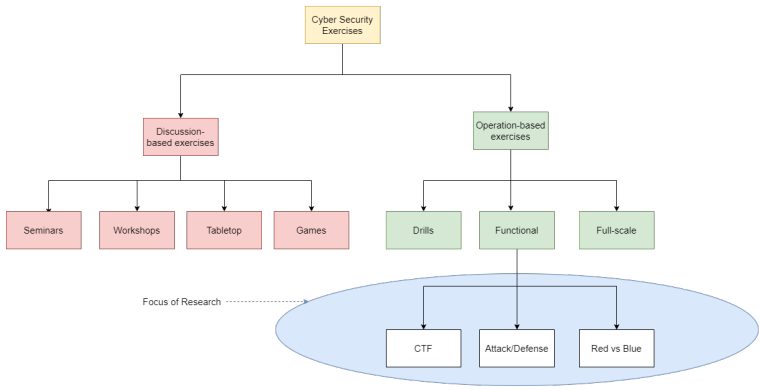


Figure 1.1: Research focus

Research Question 1

First, it is important to identify the challenges, problems and open questions associated with cybersecurity exercises. Therefore the first research question was formulated as follows:

RQ1 What are the current challenges involved in conducting cybersecurity exercises efficiently in terms of time, computational resources and learning outcomes?

Inefficiencies in conducting cybersecurity exercises were identified through a literature review and an experimental observation. A taxonomy and functional architecture of the cyber range was developed for further investigation.

Research Question 2

Based on the findings of RQ1, in-depth research and analysis on adversarial attack and defense scenario modeling and execution in cybersecurity exercises is needed. Therefore, RQ2(a) and RQ2(b) were formulated as follows:

RQ2(a) How can an efficient and adaptable *active offensive opposition* process execution be modeled against a given cybersecurity exercise defense scenario?

RQ2(b) How can an efficient and adaptable *active defensive opposition* process execution be modeled against a given cybersecurity exercise attack scenario?

The findings of RQ2(a) and RQ2(b) were used as a basis for modeling the exercise environment, which is the focus of RQ2(c):

RQ2(c) How can an efficient and adaptable cybersecurity exercise environment be modeled with respect to attack and defense scenarios?

A study on current cybersecurity curricula was performed to identify the skills that need to be taught in a cybersecurity educational program. The identified skills were used to develop a serious game for teaching attack and defense tactics, tools and procedures in a cyber range. The study's findings revealed that such platforms could be useful in teaching cybersecurity skills. A platform was developed to transform the simulated exercise environment into emulated infrastructures to conduct operation-based exercises.

Research Question 3

Cybersecurity exercises can be conducted in a manually created network environment. However, due to the dynamic nature of cybersecurity attack and defense models, the exercise environment also needs to be dynamic. RQ3(a) investigates how a dynamic exercise environment can be generated. Therefore RQ3(a) was formulated as follows:

RQ3(a) How can dynamic cybersecurity exercise environment be generated autonomously with respect to a given cybersecurity exercise model?

RQ3(b) considers how cyberattack and defense scenario models can be executed in the generated exercise environment of RQ3(a). Therefore RQ3(b) was formulated as:

RQ3(b) How can cybersecurity attack and defense scenario models be executed autonomously in a cybersecurity exercise?

A DSL (domain-specific language) was developed to emulate the role of the scenario designers, attackers and defenders present in the cybersecurity exercise. The DSL constructs were used to emulate the attacker in a semi-autonomous manner. Moreover, the DSL also contained constructs for benign users that can emulate normal user behavior to conduct realistic cybersecurity exercises. Furthermore, a special exploit chain detection algorithm was developed to emulate defender behavior in an exercise environment that utilized security event correlation and process monitoring to identify and autonomously prevent attacks. The DSL was used in a case study scenario of a Norwegian cybersecurity challenge in which the internal working of an organization network topolgy was emulated, and the participants

of the case studies were asked to perform penetration testing, incident response and forensic analysis. The results indicate that such a platform could be useful in conducting cybersecurity exercises on a large scale realistically, efficiently and effectively.

Research Question 4

RQ4 evaluates the proposed solutions in RQ3. Therefore, RQ4 was formulated as follows:

RQ4 How can the developed solutions be evaluated in terms of time, computational resource and learning outcome requirements with respect to existing solutions?

A case study was conducted during an ethical hacking course at the Norwegian University of Science and Technology in which the developed DSL was used to test and verify the proposed concepts in a classroom setting. In the case study, the developed DSL was used to orchestrate an exercise infrastructure. The DSL attack constructs created forensic attack traces through an agent that human participants verified. Meanwhile, a defender DSL construct was used to emulate defender behavior in the exercise infrastructure through defender agents. The human participants were tasked to exploit the system on which these agents were running. The case study indicated that such agents could be useful in conducting the exercises in a realistic manner.

1.6 Background

1.6.1 Cyber Ranges

With the rise of cyberthreats and the growing acceptance of cyberwarfare, cyber testbeds have become an essential capability for the military and governments to evaluate their capability to combat possible cybersecurity threats [18]. Such testbeds provide the ability to test and validate the strategies, tactics, interoperability, functionality and performance of cyberwarfare (defense and offensive) solutions. They also provide opportunities to conduct experiments to test new technologies and mitigate cyberattacks.

The term *cyber range* first appeared in the literature around 1977, where it was used to describe a family of supercomputers developed by *Control Data Corporation* for scientific and industrial usage [19]. In the context of security testing and training, the use of a cyber range can be traced back to the MIT LARIAT project, in which MIT collaborated with the US Department of Defense (DoD) to construct testbeds to test systems and detect cyberattacks in the early 2000s [20].

A study from the Australian military in 2013 [21] revitalized the subject and attracted the attention of the scientific community. Cybersecurity researchers have been burdened with the time and cost needed to establish secure test environments capable of analyzing new threats and evaluating new technologies [22]. In order to reduce these costs, DARPA *Defense Advanced Research Projects Agency* launched the National Cyber Range (NCR) program to develop the architecture, software and tools required for secure, self-contained cybertesting sites [23, 24]. NCR was developed to test network attack and defense strategies by providing businesses and research organizations with a real-world simulation environment to verify advanced concepts and capabilities for defending US communications networks against cyberthreats. Multiple countries are now in the process of establishing their own cyber ranges for different testing, training, and experimentation purposes. Some of the famous publicly known cyber ranges are the Michigan cyber range, which was established in 2012 [25], and the NATO cyber range, which was established in Estonia around 2014 [26]. Other European projects, such as KYPO [27], Cyber Wiser [28], ECHO [29] and SPARTA [30], are ongoing.

Using cyber ranges is one of the most effective ways to learn new cybersecurity skills by practicing realistic cyberexercises and techniques in a controlled and safe environment [31]. Cyber ranges facilitate training of on-demand threat scenarios to improve people's ability to identify vulnerabilities and respond to cyberthreats [32]. Cyber ranges are modeled similarly to the physical shooting ranges used by the police and military. Training arenas are thus created that simulate a wide range of security incidents so that cybersecurity experts can practice how to respond [33].

Attackers, defenders and the general systems with which users interact shape cyberspace as it exists today. Modeling the attack and defense scenarios in cyberdomains is an important line of research. Modeling and presenting the full spectrum of cyberattacks and defense behavior is a complex and daunting task given the sheer number of options the attacker and defender have [34]. Live simulations and penetration tests provide a limited and detailed set of data on cyberattack scenarios for network-specific behavior. Synthesis of this data with alternative network configurations and attack types and behavior can provide a more accurate and robust security assessment, as it helps to better understand how vulnerabilities are realized. Cyber range training is useful for people and organizations that want to experiment with new cyberdefense technologies. They can test new ideas and see how teams interact with emerging cybersecurity solutions. Additionally, it can help them to create tailor-made cybersecurity for their solutions to test in an actual environment.

1.6.2 Cybersecurity Exercises

There are multiple ways in which cybersecurity exercises can be conducted. Two of the most prominent types are discussion-based exercises and operation-based cybersecurity exercises [3]. In discussion-based exercises, a cybersecurity scenario is given, which mostly deals with an incident that is happening and evolving. Higher-level managers strategize how to respond to the incident based on their experience and decision-making. Such exercises involve hypothetical scenarios and address them in a theoretical way. Although they are very good at devising strategies for discussing and addressing different cybersecurity incidents, they are not well suited for teaching practical cybersecurity skills. Such practical skills involve exploiting known and unknown vulnerabilities within an IT infrastructure, defending against such attacks and performing forensics analysis after an incident has happened. For teaching such skills, operation-based cybersecurity exercises are used. In an operation-based exercise environment, a scenario is given on an actual IT system that has different tasks associated with attackers and defenders. In most cases, attackers have to identify some information on one of the IT systems, and such information is known as a flag to gain points, while defenders have to stop such actions and prevent attackers.

Conducting discussion-based and operation-based cybersecurity exercises requires different skill sets. For discussion-based exercises, the designer needs to have vast experience in cybersecurity incidents at a higher management level to design cybersecurity scenarios. Meanwhile, for an operation-based exercise, the exercise designer needs to have technical know-how about cybersecurity vulnerabilities and their defenses to set up IT infrastructure to conduct practical hands-on cybersecurity exercises. When organizing cybersecurity exercises, many tabletop-based exercises require little technical infrastructure and operational know-how on cybersecurity incidents. Thus, they can be conducted with very limited resources like pen and paper. Conversely, operation-based exercises require IT infrastructure with specific software, services and networking facilities, making them very difficult and expensive in terms of time and resources to organize.

1.6.3 Operation-Based Cybersecurity Exercises

Operation-based cybersecurity exercises basically involve attack and defense execution in an exercise environment. In this environment, a team of attackers (i.e., red team) tries to compromise the network and application layer vulnerabilities present in the environment while a team of defenders (i.e., blue team) tries to defend and prevent the attacks. In a recent study [35], the researchers found that such exercises are beneficial for cybersecurity skill development. The researchers conducted knowledge surveys on the participants before and after a cybersecur-

ity exercise, and they found significant improvement in network security skills, including ARP poisoning, duplication in DNS entries and firewall/router assessment.

In another study, Mirkovic et al. [36] achieved similar results to those observed in pre- and post-exercise surveys. This clearly suggests that operation-based cybersecurity exercises can help in increasing the skill set of participants in an effective manner. However, conducting an operation-based cybersecurity exercise is a costly process [37]; even at a small scale, the cost of the software and hardware required for the exercise is high. This is due to the fact that realistic cybersecurity exercises require licensed software and extensive redesign of applications according to the scenario requirements. This makes it difficult to conduct operation-based cybersecurity exercises in a regular cost-efficient manner.

The purpose of the exercises is mostly to locate a certain piece of text, a so-called flag, on a server or website. Such exercises simulate scenarios from the real world, such as hacking remote sites or exploiting vulnerabilities in certain applications. They can be individual and team-based, attracting a wide range of participants, including students, enthusiasts and professionals. The exercises can last from a few hours to a full day or even several days. They have grown from their humble roots to a sports level, with thousands of individual games and leagues taking place around the world. There are mainly three types of scenarios that are utilized in such exercises:

1. Jeopardy-style
   In Jeopardy-style scenarios, the participants can see a list of challenges and are free to choose which challenge they would like to solve. The challenges can be from different categories (e.g.., web, reverse, forensics, etc.) and are not linked with each other. However a single challenge can have multiple flags based upon the challenge difficulty level.

2. Attack/Defense
   In an attack/defense scenarios, participants are assigned vulnerable systems while attempting to attack the opponent's systems. Participants start with the time allotted to them to patch and secure their own systems and try to detect as many vulnerabilities as possible before their opponents attack the participant's infrastructure.

3. Red Team vs. Blue Team
   In a red team against blue team scenario, one team plays as an attacker while the other team plays as a defender. The complex infrastructure of an organization is emulated, and attackers learn vital techniques in compromising

the infrastructure, while defenders learn how to defend their systems against active attacks.

1.6.4 Cyber Security Exercise life-Cycle

Designing and building cybersecurity exercise scenrios is a complex and challenging task. There is a whole life-cycle involved in conducting cybersecurity exercises. Vykopal et al. [8] presented the lessons learned from an operation-based cybersecurity exercise in a cyber range. After analyzing multiple cybersecurity exercises, the researchers shared their *cybersecurity exercise life-cycle*. The life-cycle has five phases, as follows:

- Preparation
  In this phase, the exercise objectives are defined, the scenario for the exercise is developed and the necessary infrastructure for the scenarios is created. This phase takes from weeks to months in the cybersecurity exercise life-cycle because it involves considerable planning and development.

- Dry run
  In this phase, the developed scenario and deployed infrastructure are tested by a team of experts. Changes are made to ensure that everything is working as planned. This phase also takes a few weeks because it involves debugging the exercise scenario and infrastructure for any error.

- Execution
  In this phase, the cybersecurity exercise is executed by different teams to try to achieve the objectives defined in the exercise scenario. This phase usually takes from a few days to few weeks, depending on the nature of the exercise.

- Evaluation
  In this phase, the different participating teams' performance in the cybersecurity exercise is evaluated based on the achieved objectives. This phase usually takes a few days to evaluate team performance.

- Repetition
  In this phase, the overall exercise is analyzed to identify any technical and nontechnical problems that need to be addressed before rerunning the exercise. This phase usually takes a few days to fix newly identified problems.

1.7 Related Work

Cyberattack Scenario Modeling In Cybersecurity Exercises

In the literature, multiple cyberattack modeling languages are used, including Correlated Attack Modeling Language (CAML) [38], which uses different modules. The modules represent a step in a cybersecurity attack scenario. These modules can be linked to form a multi-step cybersecurity attack scenarios. The modules contain attack patterns for the purpose of attack modeling process. However, this language models cyberattacks using attack trees and graphs that are at an abstract level. Thus, these models can only be used for the identification of probable weaknesses in the system defenses. [39] modeled and simulated cyberattacks by analyzing attacker behavior and mapping the steps in that behavior, such as reconnaissance, intrusion and escalation. Then, they generated attack steps for launching an attack for a given scenario. The present research will focus on emulating such attack steps in a cybersecurity exercise.

In terms of attack execution methods and solutions, such as the Scanning, Vulnerabilities, Exploits and Detection tool (SVED) [40], Armitage and Simulated Cognitive Attacker Modeling (SC2RAM) [41] exists. SVED uses tools like OpenVas for vulnerability scanning and metasploit for vulnerability exploitation. SVED is also integrated within the CRATE [42] cyber range for some attack team automation tasks. Armitage is an another free penetration testing automation tool, which uses the nmap scanning engine for vulnerability scanning and metasploit to perform exploitation. There is a commercial version of armitage, which is called cobalt strike, which has more penetration testing exploits and features but the same core functionality. SC2RAM is utilized to mimic human cognitive behavior in a cybersecurity exercise. It uses freely available exploits to attack targets based upon cognitive decision-making. In the Cyris [43] cyber range, an automatic attack emulation tool is implemented to generate attacks like brute-force and distributed denial-of-service attacks (DDOS), but it is static in nature and used for traffic log generation. These tools follow a single attack model for multiple attack scenarios, seeking to identify and exploit the vulnerabilities through multiple attempts. This creates a great deal of network traffic noise due to their brute-force approach to vulnerability scanning and exploitation. Therefore, they are not ideal for cybersecurity exercise execution. Researchers also used crude and basic scripts for the automation of attack team tasks in the security test bed Power Cyber [44]. The researchers created separate machines with pre-defined bash scripts to automate specific attack scenarios in a cybersecurity exercise. The bash scripts execute freely available exploits to generate a network attack. However, using this approach has disadvantages as well. The scenario builders need to be familiar with many ex-

ploit automation techniques, and new exploit automation will be required for new scenarios. The present research will utilize the previously developed techniques for the creation of a new cyberattack modeling language. In comparison to those mentioned above, our cyberattack modeling language will be used in the creation of a dynamic attack scenario that will be executed autonomously. Our language will provide a high level of abstraction for the exercise designer and can be used with very little technical knowledge of vulnerabilities. Our modeling language will be integrated with the whole cybersecurity exercise life-cycle and will aid in performing dry runs on injected vulnerabilities in the exercise infrastructure as well as adversary emulation.

Cyber-Defense Scenario Modeling In Cybersecurity Exercises

A network attack description and response architecture based on the multi-level rule expression language [45] is described in the literature. It is used to classify a set of attack trees based upon a predefined rule set then uses the rule set to model an appropriate defense to address the attacks. Similarly, [46] presented the Predictive, Probabilistic Cybersecurity Modeling Language (P2CySeMoL), which is an attack graph tool that is designed to estimate the cybersecurity of enterprise architectures. It provides a theoretical approach on how cyberattacks and defenses are related quantitatively. Hence, its users only model their assets and how they are connected in order to enable calculations. These models are validated using the literature, domain experts, surveys, observations, experiments and case studies. Most of the models as well as most of their work are created on an abstract level using attack trees and attack graphs [47]. These models are suitable for running simulations, but they cannot be used in operation-based cybersecurity exercises for the emulation of defenders. In [48], the researchers proposed a human immune system-inspired autonomic system for cyberdefense. The researchers argued that it is impractical to patch every vulnerability present in an environment, and a multi-layered autonomous cyberdefense system similar to the human immune system is required. The present research will utilize the previously developed techniques for the creation of a new cyberdefense modeling language. However, the cyberdefense modeling language will be used in the creation of dynamic defense scenarios that are autonomously executed. The language will be used to specify agent behavior, which will be injected into the exercise environment and act as an active defender.

Cybersecurity Exercise Scenario Environment Modeling

Multiple cyber ranges have already implemented scenario description languages in their cybersecurity exercises. One early example of this is TELELAB [49], in which XML-based scenario description is used to generate new scenarios from a template of cybersecurity exercise resources. These generated scenarios need to

be manually modeled in XML to generate new cybersecurity exercise scenarios. Similar to TELELAB, Cyris [43] uses a YAML-based cyber range modeling language in which the cyber range host, guest and clones can be modeled and generated. Like Telelab, Cyris also includes reconfigured templates for cyber range host and guest systems, and the clone systems are manually modified from the guest systems according to the given cybersecurity exercise scenario. In comparison, Cyrtone [50] takes the process a bit further by introducing the concept of a training description in the cybersecurity environment modeling process. Cyrtone uses Cyris for environment creation; however, the clone systems are generated based upon the input in the training description. CyberVan [51] is another example similar to Cyris in which simulation models from NS2 and Opnet can be used in the creation of the emulated exercise environment. An implementation that automates the process of cybersecurity exercise scenario generation is SECGEN [13] (Security Scenario Generation). It is a framework for generating randomly vulnerable rich-scenario VMs for learning computer security and hosting CTF Events. SECGEN has a definition of multiple vulnerabilities, which can be inserted in VMs that are randomly generated by an XML-based scenario configuration input. In the scenario configuration, a scenario description defines the number and type of operating systems and services that are involved as well as the number of vulnerabilities that are going to exist in the scenario. Then, the scenario is generated automatically by incorporating the setting defined in the configuration file. TELELAB and SECGEN only create the environment for a cybersecurity exercise. The environment still requires teams of attackers and defenders simultaneously to mimic a real-world cybersecurity environment.

1.7.1 Cyber Range State of The Art

A considerable amount of research work has been carried out on cybersecurity and technologies that are enabling the execution of cybersecurity exercises efficiently and autonomously. In 2020, a study presented the CRACK cyber range [11], which used logical programming to formally model cybersecurity exercise scenarios and cloud orchestration technologies to deploy the modeled scenarios. The researchers emulated the network of an organization to present the work. They automated the roles of white team members in the cybersecurity exercise life-cycle to automatically deploy the exercise infrastructure. Further, they formally validated the deployed infrastructure through logical verification.

In 2019, researchers presented a model-driven approach for modeling cybersecurity scenarios [52]. They used two different approaches. In the first approach, they modeled the security assurance requirement for an organization, and in the second they modeled the specification for creating a security testbed to check and verify the security requirements. They presented an emulation framework model

for meeting the operational cybersecurity requirements. Their work can be used to model an organization as a whole; however, no artifacts were presented in their work to emulate the organization.

In 2021, researchers presented a model-driven cyber range assurance platform [53] in which they used the models developed in [52] for cyberthreat and training preparation. The researchers combined different simulation, emulation and serious game techniques to provide training related to the security assurance of an organization. Then, they used feedback from the training participants to adapt their models to provide training according to their skill set needs. They used an Internet of Things (IoT) smart home scenario to verify their developed tools and techniques.

Similarly, in [12] a new method of conducting cybersecuirty exercises was introduced. The researchers presented Nautilus which is a tool for the automatic deployment and sharing of cybersecurity exercise scenarios. The researchers developed a scenario description language to specify the scenario requirements and used virtualization technologies to deploy the scenario. The researchers argued that configuring and deploying scenarios for a dynamic threat environment is a difficult task, and specifying scenario requirements in a scenario language enables sharing the scenario with multiple partners for standardized training.

Likewise, in 2021 a method was presented for generating models of IT systems automatically [54]. The researcher used expert-defined roles that can take very simple inputs about an organization and transform them into operational IT infrastructure models that can be used to verify different properties of the IT infrastructure. The researchers made the system flexible enough to accept new rules to model new scenarios for different organizations. They used linear programming to implement the solution and verify its properties. While they did not actually deploy the cybersecurity exercise environment, the study made a valuable contribution in terms of modeling the networked topology of different IT infrastructures that can be used in operational cybersecurity exercises.

In parallel, a 2020 study focused on virtualization and automation of cybersecurity training and experimentation [55]. The researchers developed an orchestration interface for automatically deploying cybersecurity exercise infrastructure. A scenario description language was used to specify the scenario requirements formally and automatically deploy them using different orchestration technologies. A python-based orchestrator was used that can take the scenario inputs and transform them into cyber range artifacts and deploy the cybersecurity exercise environment.

Additionally, in 2021 a hybrid cyber range called PAIDEUSIS [56] was presen-

ted. The researchers combined technologies from CRACK with actual hardware representing IoT, SCADA and other devices to organize hybrid capture the flag competitions. The researchers used different freely available tools to accomplish this task and proposed a system for the hybrid cyber range. The range consisted of multiple cyber range environments, including SEcube, Chip whisperer, hardware and network emulation environment. By combining all these technologies and ranges, the researchers were able to integrate hardware devices with virtualized networks. In their hybrid cyber range, there were multiple subnets that had different components, some of which were virtualized while others were interconnected to develop the hybrid cyberenvironment.

1.8 Methodology

As the research moved toward the development of artifacts for conducting cybersecurity exercises, the DSR (design science research) methodology was the method of choice. DSR is a results-oriented research method in information technology that provides specific guidelines for the evaluation and iteration of research projects. It is largely applied in engineering and computer science, but it is not limited to these disciplines and can be found in many disciplines and areas. It is used to apply knowledge to produce effective artifacts [57]. In general, the DSR methodology contains five steps [58], which are depicted in Figure 1.2. These steps are as follows:

- **Awareness of the problem** can come from a review of the literature and through observation and experiments to identify problems and inefficiencies. The output of this phase is a proposal for the new research effort.
- **The suggestion** phase follows after the awareness of the problem. Solutions are suggested for the identified problem. The output of this phase is the tentative design of the solution for the identified problem.
- **Development** is the phase in which the tentative design of the solution, which is suggested in the suggestion phase, is implemented. The output of this phases is an artifact that can be used in solving the identified problem.
- **Evaluation** is the phase in which the developed artifact is evaluated based upon defined quantitative or qualitative matrices.
- **The conclusion** is the final step of the research, where the results of the evaluation are used to identify how well the artifact was able to solve the identified problem.

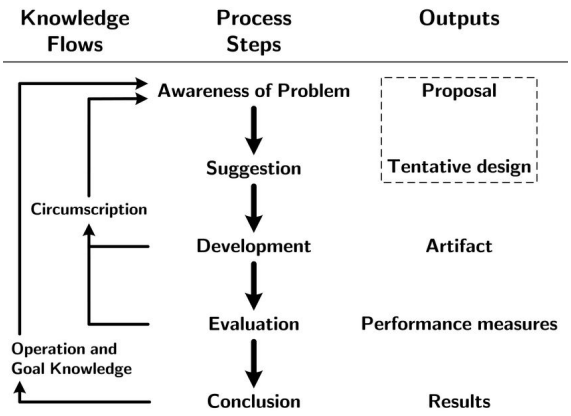


Figure 1.2: General methodology of design science research [58]

The research questions tackled in this thesis were addressed in multiple research papers, which employed a multitude of research methodologies. Four primary research methodologies were used: SLR *(systematic literature review)*, *surveys*, *case studies* and *experiments*. These methods were applied in eight research articles during different DSR activity phases. For the awareness of the problem, SLR, surveys and case studies were employed, which provided an overview of the current status, opportunities and challenges in cyber range technologies. Based on the findings, the hypothesis for the study was formalized, and the next phase of DSR was investigated. In the *suggestions* phase, surveys, case studies and experiments were conducted to propose the tentative design for addressing the research questions. In the *development* phase, *case studies* and *experiments* were conducted on the developed artifacts to check their performance against defined test cases, and the results were presented in relevant publications. Finally in the *evaluation* phase, *case studies* and *experiments* were conducted on the developed artifact to check its usability in an academic setting. The different research methods that were used during the different DSR activities with respect to the research questions and research papers are presented in Table 1.1.

DSR is an iterative process, and the output of different research steps is connected with those of previous research steps. Some of the methods that were used to carry out different research steps along with their methods and how they were connected with other steps include the following:

- Problem identification: Systematic literature review methods [59] were used to identify the current challenges of unclassified cybersecurity exercise platforms. Additionally, an experimental observation of cybersecurity exercises

at NTNU was conducted. In the observatory study, the set of problems related to conducting cybersecurity exercises were analyzed and discussed. This study was used to identify the inefficiencies in conducting cybersecurity exercises in terms of time, computational resources and learning outcome requirements. This step was mapped with the first step of the DSR methodology and used to answer the first research question *RQ1*.

- Requirements gathering: In this step, we gathered the requirements and proposed the tentative design of the modeling artifacts. Literature reviews and surveys were used to gather information from cybersecurity exercise stakeholders on proposed research. Requirements for the adversarial environment were established based on the information gathered from the cybersecurity exercise stakeholders. Furthermore, an extensive analysis of current cybersecurity exercise adversarial environments was performed by conducting a literature review on current adversarial tools and techniques. This covered the first part of *RQ2*.

- Tentative design: This step focused on establishing the tentative design for attack and defense scenario and environment modeling in cybersecurity exercises. Based on the design requirements gathered in the previous step, an efficient and adaptable attack and defense scenario and environment models for cybersecurity exercises were proposed. For the attack and defense scenario and exercise environment modeling step, ***model-driven engineering*** (MDE) [60] methods were used, which allowed us to formalize the structure, behavior and requirements of the attack and defense scenario and environment within the domain of cybersecurity education [60]. This helped the researcher to focus more on capturing the domain knowledge in formal models, thus reducing the need for domain experts in conducting cybersecurity exercises. In terms of concrete MDE techniques, the researcher used ***domain-specific modeling language*** [60](DSML) tools. DSML allowed the researcher to create domain-specific languages for the attack and defense scenarios and exercise environment. This helped the researcher to segregate the roles of attack and defense and environment models that enable them to be used independently with respect to each other.

- Artifact development: This step focused on the development of three artifacts: autonomous cybersecurity exercise environment generation, autonomous cybersecurity attack scenario execution and autonomous cybersecurity defense scenario execution. The scenario environment and the attack and defense execution behavior were formally modeled and analyzed.The formal model was developed in Datalog, which is a programming language based

on a declarative logic [61]. The formal model analysis helped to identify errors and verify different scenario properties of the exercise scenario models and execution behavior before their actual execution. In terms of cybersecurity exercise execution, the researcher developed an agent-based system [62]. The agent works similar to the multi-agent cyberattack and defense simulation framework [63], but the developed cyberattack and defense agents are **emulation**-based artifacts with an additional environment creation platform. The key contribution in this phase was the integration of the dynamic cyberattack and defense scenario and security exercise environment models developed in RQ2 with an agent-based emulation platform. For exercise environment emulation, OpenStack [1] was used because it is being used in NTNU for research and development. For attacker emulation, a Kali linux [2]-based agent was implemented due to its penetration testing support. For the defender agent, a C-Sharp based agent was developed to run in a Windows-based exercise environment. This step mapped to the third step of the DSR methodology and answered the third research question *RQ3*.

- Evaluation: This step focused on the evaluation of the developed artifacts. Multiple studies were conducted on cybersecurity exercises, which generated research data. According to researchers [64], the gathered data can be used in both quantitative and qualitative evaluations. In quantitative evaluation, research deals with the collection and analysis of numerical data. It enables quantification and statistical analysis of the data to extract important information. In qualitative evaluation, research deals with the collection and analysis of descriptive data. This makes it possible to conduct research involving humans. That often includes information about characteristics that cannot be measured in quantitative research, such as emotional state and social characteristics. We performed verification and validation of the developed artifacts based upon quantitative and qualitative evaluation metrics. This step mapped to the fourth step of the DSR methodology and answered the fourth research four *RQ4*. More details about this step are provided in the next section (Section 1.8.1).

1.8.1 Verification and Validation

In qualitative research, the researchers observe the study participants and draw conclusions about their behavior, motivations and struggles. This type

[1] https://docs.openstack.org/openstack-ansible/latest/
[2] https://www.kali.org/

Research Questions	Research Papers	DSR Activity	Research Methods			
			SLR	Survey	Case Study	Experiment
RQ1	1,2,3	Awareness	✓	✓	✓	
RQ2	4	Suggestions		✓	✓	✓
RQ3	5,6	Development			✓	✓
RQ4	7	Evaluation		✓	✓	✓

Table 1.1: Mapping research methods used for addressing different RQs with DSR methodology

of data can be collected from diaries, accounts and in-depth interviews and analyzed based on sound theory and thematic analysis. Data-gathering methods can help predict potential reactions, and large ethnographies allow data to be collected in natural and uncontrolled environments. This means that qualitative researchers study things in their natural environment and try to understand and interpret phenomena in terms of the meanings assigned to them by humans. The aim of qualitative research is to understand and live the social realities of individuals, groups and cultures as far as possible from the perspective of the participants.

Quantitative research is used to quantify opinions, attitudes, behaviors and other defined variables, with the aim of supporting or refuting hypotheses about certain phenomena or contextualizing the results of a sample of a broad population or group. Quantitative techniques include various forms of questionnaires, surveys, structured interviews and behavioral observations based on explicit coding and categorization schemes. Because quantitative research determines what is measured and what is measured to detect patterns (e.g., behavior, motivation, emotions and cognition), quantitative data collection is considered a structured qualitative method.

Two case studies were conducted in which two experiments were performed to verify and validate the artifact outcomes in a cybersecuirty exercise environment. The first experiment dealt with quantitative evaluation of the developed artifact with respect to metrics like time, computational resources and exercise environment creation accuracy. This helped the researcher in the artifact verification step. The second experiment dealt with qualitative evaluation of the developed artifact, and the skill level improvement of the exercise participants was analyzed. This helped the researcher to validate the developed artifact.

Case Study 1

The first case study was conducted at the *Norwegian Cybersecurity Challenge*. An experiment was performed in which the autonomous environment generation capability of the developed artifacts was validated. The same exercise scenario was assigned to two human white teams. The first team was aided with the autonomous environment generation artifact, while the second team was only allowed to use traditional environment creation tools and techniques. The results of this experiment were validated based upon the following metrics.

- **Time** This is the most important metric for our evaluation purpose. As we stated in the research background, preparation of the cybersecurity

exercise environment is the longest phase in the cybersecurity exercise life-cycle. The researcher measured the time difference in environment creation between the teams that were aided by the autonomous environment generation artifact and those that were not.

- **Computational resource** Comparative analysis of the computational resources required for preparing the cybersecurity exercise environment was performed. Metrics like memory, processing power and space requirements were compared to existing cybersecurity environment creation tools and techniques, including SECGEN, Cyrtone and CyberVan.
- **Environment creation accuracy** A dry run was performed on the exercise environment created by the autonomous environment creation artifact. The aim of the dry run was to identify how much the exercise scenario in terms of exercise objectives was represented in the created environments.

The result of this experiment helped us to establish the following:

1. The autonomous environment creation artifact is more time-efficient and cost-effective compared to manual environment creation.
2. The manual environment creation process is more error prone compared to the autonomous environment creation process.

Case Study 2

The second case study was conducted during a lab exercise on information security courses taught at NTNU. In the case study, an experiment was performed on two groups of students. The first group of students was the control group, who performed the cyberattack and defense exercise against human adversaries. The second group of students was the experimental group. This group performed the cyberattack and defense exercise against autonomous attack and defense adversaries. The results of this experiment were analyzed with pre- and post-exercise surveys, which helped us to establish the following points:

1. Autonomous adversaries can provide realistic cybersecurity training.
2. Autonomous adversaries are more accurate in terms of attack and defense scenario execution repetition compared to human adversaries.

1.9 Summary of Contributions

1.9.1 List of Publications

1. First Research Question

   (a) Yamin, M. M., & Katt, B. (2018). Inefficiencies in Cyber-Security Exercises Life-Cycle: A Position Paper. In AAAI Fall Symposium: ALEC (pp. 41-43).

   (b) Yamin, M. M., Katt, B., Torseth, E., Gkioulos, V., & Kowalski, S. J. (2018, September). Make it and break it: An IoT smart home testbed case study. In Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control (pp. 1-6).

   (c) Yamin, M. M., Katt, B., & Gkioulos, V. (2020). Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. Computers & Security, 88, 101636.

2. Second Research Question

   (a) Yamin, M. M., Katt, B., & Nowostawski, M. (2021). Serious Games as a Tool to Model Attack and Defense Scenarios for Cyber-Security Exercises. Computers & Security, 110, 102450.

3. Third Research Question

   (a) Yamin, M. M., Katt, B., & Gkioulos, V. (2019, March). Detecting windows based exploit chains by means of event correlation and process monitoring. In Future of Information and Communication Conference (pp. 1079-1094). Springer, Cham.

   (b) Yamin, M. M. & Katt, B.(2022). Modeling and Executing Cyber Security Exercise Scenarios in Cyber Ranges. Computers & Security, 116, 102635.

4. 4th Research Question

   (a) Yamin, M. M. & Katt, B. Use of Cyber Attack and defense agents in Cyber Ranges: A Case Study. (Accepted in Computers & Security undergoing minor revision)

Additional Publications

1. Hannay, J. E., Stolpe, A., & Yamin, M. M. (2021, July). Toward AI-Based Scenario Management for Cyber Range Training. In International Conference on Human-Computer Interaction (pp. 423-436). Springer, Cham.

2. Wen, S. F., Yamin, M. M., & Katt, B. (2021, September). Ontology-Based Scenario Modeling for Cyber Security Exercise. In 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (pp. 249-258). IEEE Computer Society.

3. De Zan, T., & Yamin, M. M. . (2021). Towards a Common ECSC roadmap. European Information Security Agency, ISSN 978-92-9204-464-0.

4. Yamin, M. M., Ullah, M., Ullah, H., & Katt, B. (2021). Weaponized AI for cyber attacks. Journal of Information Security and Applications, 57, 102722.

5. Yamin, M. M., & Katt, B. (2019, June). Modeling attack and defense scenarios for cyber security exercises. In 5th interdisciPlinary cyber research conference (p. 7).

6. Yamin, M. M., & Katt, B. (2019, August). Cyber security skill set analysis for common curricula development. In Proceedings of the 14th International Conference on Availability, Reliability and Security (pp. 1-8).

1.9.2 List of Major Contributions

First Research Question

1. *Inefficiencies in Cyber-Security Exercises Life-Cycle: A Position Paper*
   In the first Paper, the researcher identified the problem that there are inefficiencies in conducting cybersecurity exercises. The current way of conducting such exercises is error-prone, and it takes a long time and considerable resources to configure the exercise infrastructure manually. Automation was suggested as a way to help and reduce the cost of conducting the cybersecurity exercises.

2. *Make It and Break It: An IoT Smart Home Testbed Case Study*
   The researcher conducted an experimental case study to observe the whole cybersecurity exercise life-cycle. In the case study, two teams were tasked to develop a testbed, and then the teams were asked to switch sides and attack each other's testbed. The exercise was called *make it and break it*. The case study was conducted to identify all the inefficiencies present in the whole cybersecurity exercise life-cycle. The results indicated that there are many inefficiencies in the cybersecurity exercise life-cycle, and they can be removed using automation technologies. This conclusion was drawn from a survey conducted with the exercise participants before and after the exercise.

3. *Cyber Ranges and Security Testbeds: Scenarios, Functions, Tools and Architecture*
   In the third paper, the researcher conducted a detailed systematic literature review of articles related to cyber ranges and cybersecurity testbeds. The current state of cyber ranges and the technologies available in automating exercise setups in cyber ranges were identified. A taxonomy of cyber ranges was developed, and a functional architecture of cyber ranges was proposed.

Second Research Question

1. *Serious Games as a Tool to Model Attack and Defense Scenarios for Cyber-Security Exercises*
   In this work, the researcher developed a serious game to test different cybersecurity attack and defense skills before the actual deployment of the exercise infrastructure. The researcher conducted different field studies and conducted surveys. The survey identified that conducting cybersecurity exercises in such a way is useful and teaches different skills without actually deploying the infrastructure beforehand. The research proposed the first version of a DSL for infrastructure deployment to conduct operational cybersecurity exercises.

Third Research Question

1. *Detecting Windows-Based Exploit Chains by Means of Event Correlation and Process Monitoring*
   In this work, the researcher developed the core algorithms required for the defender agent. In the proposed algorithm, security event information in a Windows-based environment is collected and analyzed based upon given patterns. When the patterns are identified, specific actions are executed to stop the attacker. An experiment was performed on the algorithm to identify its effectiveness against different one-day vulnerabilities. The performance of the algorithm was found to be satisfactory in this case.

2. *Modeling and Executing Cyber Security Exercise Scenarios in Cyber Ranges*
   In this work, the whole process of the cybersecurity exercise life-cycle was modeled and executed computationally using a DSL. The models were formally analyzed and verified using logic programming. The DSL contains multiple parts that are used to emulate multiple team roles, including white team, red team, blue team and traffic generators. This made the cybersecurity excise execution efficient, computationally repeatable and cost-effective. Multiple experiments were performed in this work to identify the effectiveness

of the proposed solution. It was found that the proposed solution is very cost-effective and efficient in conducting cybersecurity exercises.

Fourth Research Question

1. *Use of Cyber Attack and Defense Agents in Cyber Ranges: A Case Study*
   In this work, the researcher conducted a case study and utilized the attacker and defender agents developed in the previous research question in a computer versus human exercise. The case study highlighted the working of the attacker and defender agents, and a new concept of *execution plans* was developed to formally model their behavior in an exercise environment. The case study had two parts. First, the attacker created forensic traces, which were then used to evaluate the performance of the human participants. In the second part, an active defender agent was placed in an exercise environment that defended attacks against human participants. The results indicated that both the attacker and defender agents performed as expected. The participants identified attack activity by the attacker agent, while the defender agent created necessary friction to prevent human attackers from achieving their objective.

1.10 Limitations

Although this research work removed many inefficiencies from the cybersecurity exercise life-cycle, there are a few limitations that are highlighted below based on each research question:

First Research Question

The first research question identified the current state of the art, and the challenges, opportunities and technologies related to the cyber range were identified using a systematic literature review and experimental observation. Although the results were promising at the time, general cyber range technologies were identified and not specific technologies related to SCADA, IoT, etc. The field is evolving so rapidly that the results quickly become outdated. Moreover, the experimental observation of more human participants could yield better results.

Second Research Question

A serious game was developed to test and verify attack and defense modeling scenarios in a simulated environment. Although the study results were promising, continuous experimental validation of the developed game could have led to more systematic results. Additionally, the game was designed to be used for helping

people to understand cybersecurity scenarios from attacker and defender perspectives. However, most of the people played it from the attacker prospective due to insufficient knowledge on defense tactics.

Third Research Question

For the third research question, the lessons learned from the first and second questions were used to develop a DSL to orchestrate the life-cycle of the cybersecurity exercise. This resulted in improving the execution of the cybersecurity exercise life-cycle and removing many inefficiencies. The DSL was used in multiple exercises and was validated through multiple experiments. Despite the fact that defender agents are able to monitor and react to run time events, there has been a limitation in addressing the monitoring part of the whole cybersecurity exercise infrastructure. Moreover, the researcher used specifications related to vulnerabilities based on strict operational requirements of software, service and configurations, which is a very unorthodox approach compared to traditionally accepted vulnerability specifications like CVE and CWE. Integrating other forms of vulnerability classification in the developed DSL is one of the future work directions.

Fourth Research Question

To address the last research question, experimental case studies were conducted to validate the effectiveness of the agents developed during this research. The results were positive and indicated their effectiveness in cybersecurity exercises; however, further experimental validation could improve the quality of the results. Moreover, the agents currently utilize the predefined intelligence of an expert to launch attacks and defenses autonomously, but an intelligent logic-based solution for their run time decision-making would make them more useful.

1.11 Conclusion and Future Work

Although cybersecurity exercises have been conducted since the 1990s, the field of systematic training through cyber range technology is still emerging. When this research work started, inefficiencies in the cybersecurity exercise life-cycle were a major challenge in conducting cybersecurity exercises, as the life-cycle took too much time and resources. This research work effectively removed the inefficiencies from the cybersecurity exercise life-cycle and made the execution of cybersecurity exercises very efficient and scalable in terms of time and resources. We conducted systematic literature reviews, experimental observations, case studies and artifact development to reduce the inefficiencies in the cybersecurity exercise life-cycle.

Additionally, We employed formal scenario modeling and analyzed cybersecurity

attack and defense scenarios before their actual deployment. This removed many errors in the cybersecurity exercise planning and design phase. We developed an orchestrator that can take a scenario and deploy it automatically, which optimized the cybersecurity exercise deployment process and removed many inefficiencies. We represented the scenario models as easily changeable DSL specifications, which make them flexible and adaptable based on changing scenario requirements. We integrated an attack and defense agent in the exercise environment to perform dry runs and support the execution of exercises to make the whole cybersecurity exercise life-cycle more efficient and realistic. The proposed solution was the first functional prototype of the Norwegian cyber range and was extensively used in multiple cybersecurity exercises involving hundreds of people.

Although the present research removed many inefficiencies from the cybersecurity exercise life-cycle, there is always room for improvement. One of the requirements from various exercise organizers has been to create new unique scenarios for each exercise. While the developed solution supports various phases of the cybersecurity exercise life-cycle, it requires input from a human expert. That expert needs to design and use the developed DSL to specify a scenario for each exercise. Therefore, in the future, the researcher will be working on an AI-based exercise scenario planner that will assist in the planning of cybersecurity exercise scenarios based on the given objectives. The scenario planner will combine machine reasoning with scenario formalism, which will be used to make the cybersecurity exercise requirements more abstract. This will help to generate new and unique scenarios that can be deployed using the developed orchestrator.

Bibliography

[1] cyber_security_strategy_norway.pdf, (Accessed on 12/13/2021).
URL https://www.regjeringen.no/globalassets/upload/fad/vedlegg/ikt-politikk/cyber_security_strategy_norway.pdf

[2] list-of-measures–national-cyber-security-strategy-for-norway.pdf, (Accessed on 12/13/2021).
URL https://tinyurl.com/2p93hysw

[3] R. Gurnani, K. Pandey, S. K. Rai, A scalable model for implementing cyber security exercises, in: 2014 International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, 2014, pp. 680–684.

[4] About norwegian cyber range - ntnu, (Accessed on 12/13/2021).
URL https://www.ntnu.no/ncr

[5] Isc2-cybersecurity-workforce-study-2021.ashx, (Accessed on 12/19/2021).
URL https://www.isc2.org//-/media/ISC2/Research/2021/ISC2-Cybersecurity-Workforce-Study-2021.ashx

[6] B. Uckan Färnman, M. Koraeus, S. Backman, The 2015 report on national and international cyber security exercises: Survey, analysis and recommendations (2015).

[7] Towards a common ecsc roadmap — enisa, (Accessed on 12/14/2021).
URL https://tinyurl.com/5xrfjaxf

[8] J. Vykopal, M. Vizváry, R. Oslejsek, P. Celeda, D. Tovarnak, Lessons learned from complex hands-on defence exercises in a cyber range, in: 2017 IEEE Frontiers in Education Conference (FIE), IEEE, 2017, pp. 1–8.

[9] I. Priyadarshini, Features and architecture of the modern cyber range: a qualitative analysis and survey, University of Delaware, 2018.

[10] T. Gustafsson, J. Almroth, Cyber range automation overview with a case study of crate, in: Nordic Conference on Secure IT Systems, Springer, 2020, pp. 192–209.

[11] E. Russo, G. Costa, A. Armando, Building next generation cyber ranges with crack, Computers & Security 95 (2020) 101837.

[12] G. Bernardinetti, S. Iafrate, G. Bianchi, Nautilus: A tool for automated deployment and sharing of cyber range scenarios, in: The 16th International Conference on Availability, Reliability and Security, 2021, pp. 1–7.

[13] Z. C. Schreuders, T. Shaw, G. Ravichandran, J. Keighley, M. Ordean, et al., Security scenario generator (secgen): A framework for generating randomly vulnerable rich-scenario vms for learning computer security and hosting ctf events, in: USENIX, USENIX Association, 2017.

[14] O. Darwish, C. M. Stone, O. Karajeh, B. Alsinglawi, Survey of educational cyber ranges, in: Workshops of the International Conference on Advanced Information Networking and Applications, Springer, 2020, pp. 1037–1045.

[15] F. Maymí, R. Bixler, R. Jones, S. Lathrop, Towards a definition of cyberspace tactics, techniques and procedures, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 4674–4679.

[16] B. E. Endicott-Popovsky, V. M. Popovsky, Application of pedagogical fundamentals for the holistic development of cybersecurity professionals, ACM Inroads 5 (1) (2014) 57–68.

[17] Ctf events — enisa, (Accessed on 12/19/2021).
URL https://www.enisa.europa.eu/publications/ctf-events

[18] Militaries developing cyber testbeds to test and validate strategies, tactics and security measures for cyber warfare | international defense security & technology inc., (Accessed on 08/06/2021).
URL https://tinyurl.com/msw5c3ka

[19] A. H. J. Sale, Primitive data types, Australian Compuer Journal 9 (2) (1977) 63–71.

[20] D. J. Pack, W. Streilein, S. Webster, R. Cunningham, Detecting http tunneling activities, Tech. rep., MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB (2002).

[21] J. Davis, S. Magrath, A survey of cyber ranges and testbeds (2013).

[22] Darpa builds cyber range to test security measures – gcn, (Accessed on 08/06/2021).
URL https://gcn.com/articles/2010/06/07/defense-it-1-cyber-range.aspx

[23] A secure architecture for the range-level command and control system of a national cyber range testbed, in: 5th Workshop on Cyber Security Experimentation and Test (CSET 12), USENIX Association, Bellevue, WA, 2012.
URL https://www.usenix.org/conference/cset12/workshop-program/presentation/rosenstein

[24] M. Rosenstein, F. Corvese, A secure architecture for the range-level command and control system of a national cyber range testbed., in: CSET, 2012.

[25] M. Turčaník, A cyber range for armed forces education, Information & Security 46 (3) (2020) 304–310.

[26] Nato investing in the development of estonian cyber range | kaitseministeerium, (Accessed on 12/13/2021).
URL https://kaitseministeerium.ee/en/news/nato-investing-development-estonian-cyber-range

[27] J. Vykopal, R. Ošlejšek, P. Čeleda, M. Vizvary, D. Tovarňák, Kypo cyber range: Design and use cases (2017).

[28] L. A. Org, A. Refsdal, G. Erdogan, R. Manella, R. Cascella, P. Lombardi, M. Nannipieri, D3. 3-cyber risk modelling tool (2016).

[29] Echo federated cyber range – echo network, (Accessed on 12/20/2021).
URL https://echonetwork.eu/echo-federated-cyber-range/

[30] J. Hajny, S. Ricci, E. Piesarskas, M. Sikora, Cybersecurity curricula designer, in: The 16th International Conference on Availability, Reliability and Security, 2021, pp. 1–7.

[31] Cyber range - test bed for cybersecurity | rise, (Accessed on 08/06/2021).
URL https://www.ri.se/en/test-demo/cyber-range

[32] C. Braghin, S. Cimato, E. Damiani, F. Frati, L. Mauri, E. Riccobene, A model driven approach for cyber security scenarios deployment, in: Computer Security, Springer, 2019, pp. 107–122.

[33] What is a cyber range? | cybersecurity guide, (Accessed on 08/06/2021). URL https://cybersecurityguide.org/resources/cyber-ranges/

[34] Cyber threat assessment via attack scenario simulation using an integrated adversary and network modeling approach | the society for modeling & simulation international, https://tinyurl.com/2bzab73r, (Accessed on 08/06/2021).

[35] E. Moore, S. Fulton, D. Likarish, Evaluating a multi agency cyber security training program using pre-post event assessment and longitudinal analysis, in: IFIP World Conference on Information Security Education, Springer, 2017, pp. 147–156.

[36] J. Mirkovic, A. Tabor, S. Woo, P. Pusey, Engaging novices in cybersecurity competitions: A vision and lessons learned at acm tapia 2015, in: 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15), 2015.

[37] N. Childers, B. Boe, L. Cavallaro, L. Cavedon, M. Cova, M. Egele, G. Vigna, Organizing large scale hacking competitions, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2010, pp. 132–152.

[38] S. Cheung, U. Lindqvist, M. W. Fong, Modeling multistep cyber attacks for scenario recognition, in: DARPA information survivability conference and exposition, 2003. Proceedings, Vol. 1, IEEE, 2003, pp. 284–292.

[39] M. E. Kuhl, J. Kistner, K. Costantini, M. Sudit, Cyber attack modeling and simulation for network security analysis, in: Proceedings of the 39th Conference on Winter Simulation: 40 years! The best is yet to come, IEEE Press, 2007, pp. 1180–1188.

[40] H. Holm, T. Sommestad, Sved: Scanning, vulnerabilities, exploits and detection, in: Military Communications Conference, MILCOM 2016-2016 IEEE, IEEE, 2016, pp. 976–981.

[41] R. M. Jones, R. O'Grady, D. Nicholson, R. Hoffman, L. Bunch, J. Bradshaw, A. Bolton, Modeling and integrating cognitive agents within the emerging cyber domain, in: Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Vol. 20, Citeseer, 2015.

[42] T. Sommestad, Experimentation on operational cyber security in crate, NATO STO-MP-IST-133 Specialist Meeting, Copenhagen, Denmark, 2015.

[43] C. Pham, D. Tang, K.-i. Chinen, R. Beuran, Cyris: A cyber range instantiation system for facilitating security training, in: Proceedings of the Seventh Symposium on Information and Communication Technology, ACM, 2016, pp. 251–258.

[44] A. Ashok, S. Krishnaswamy, M. Govindarasu, Powercyber: A remotely accessible testbed for cyber physical security of the smart grid, in: Innovative Smart Grid Technologies Conference (ISGT), 2016 IEEE Power & Energy Society, IEEE, 2016, pp. 1–5.

[45] S. Souissi, L. Sliman, B. Charroux, An attack description and response architecture based on multi-level rule expression language, Journal of information assurance and security (JIAS) (2016).

[46] H. Holm, K. Shahzad, M. Buschle, M. Ekstedt, Cysemol: Predictive, probabilistic cyber security modeling language, IEEE Transactions on Dependable and Secure Computing 12 (6) (2015) 626–639.

[47] B. Kordy, L. Piètre-Cambacédès, P. Schweitzer, Dag-based attack and defense modeling: Don't miss the forest for the attack trees, Computer science review 13 (2014) 1–38.

[48] D. Dasgupta, Immuno-inspired autonomic system for cyber defense, information security technical report 12 (4) (2007) 235–241.

[49] M. Casini, D. Prattichizzo, A. Vicino, The automatic control telelab: A user-friendly interface for distance learning, IEEE Transactions on Education 46 (2) (2003) 252–257.

[50] R. Beuran, D. Tang, C. Pham, K.-i. Chinen, Y. Tan, Y. Shinoda, Integrated framework for hands-on cybersecurity training: Cytrone, Computers & Security (2018).

[51] R. Chadha, T. Bowen, C.-Y. J. Chiang, Y. M. Gottlieb, A. Poylisher, A. Sapello, C. Serban, S. Sugrim, G. Walther, L. M. Marvel, et al., Cybervan: A cyber security virtual assured network testbed, in: Military Communications Conference, MILCOM 2016-2016 IEEE, IEEE, 2016, pp. 1125–1130.

[52] I. Somarakis, M. Smyrlis, K. Fysarakis, G. Spanoudakis, Model-driven cyber range training: a cyber security assurance perspective, in: Computer Security, Springer, 2019, pp. 172–184.

[53] M. Smyrlis, I. Somarakis, G. Spanoudakis, G. Hatzivasilis, S. Ioannidis, Cyra: A model-driven cyber range assurance platform, Applied Sciences 11 (11) (2021) 5165.

[54] I. Kovačević, S. Groš, A. Đerek, B. Bijelić, Automatically generating models of it systems, arXiv preprint arXiv:2107.11102 (2021).

[55] I. Bica, R. L. Unc, S. Țurcanu, Virtualization and automation for cybersecurity training and experimentation, in: International Conference on Information Technology and Communications Security, Springer, 2020, pp. 227–241.

[56] G. Berra, G. Ferraro, M. Fornero, N. Maunero, P. Prinetto, G. Roascio, Paideusis: A remote hybrid cyber range for hardware, network, and iot security training.

[57] S. T. March, G. F. Smith, Design and natural science research on information technology, Decision support systems 15 (4) (1995) 251–266.

[58] V. K. Vaishnavi, W. Kuechler, Design science research methods and patterns: innovating information and communication technology, Crc Press, 2015.

[59] D. Budgen, P. Brereton, Performing systematic literature reviews in software engineering, in: Proceedings of the 28th international conference on Software engineering, ACM, 2006, pp. 1051–1052.

[60] D. C. Schmidt, Model-driven engineering, COMPUTER-IEEE COMPUTER SOCIETY- 39 (2) (2006) 25.

[61] J. W. Lloyd, Foundations of logic programming, Springer Science & Business Media, 2012.

[62] D. Helbing, Agent-based modeling, in: Social self-organization, Springer, 2012, pp. 25–70.

[63] I. Kotenko, Multi-agent modelling and simulation of cyber-attacks and cyber-defense for homeland security, in: Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. IDAACS 2007. 4th IEEE Workshop on, IEEE, 2007, pp. 614–619.

[64] T. W. Edgar, D. O. Manz, Research Methods for Cyber Security, Syngress, 2017.

Chapter 2

Research Articles

2.1 Inefficiencies in Cyber-Security Exercises Life-Cycle: A Position Paper

Inefficiencies in Cyber-Security Exercises Life-Cycle: A Position Paper

Muhammad Mudassar Yamin and Basel Katt
Department of Information Security and Communication Technology
Norwegian University of Science and Technology (NTNU), Norway
muhammad.m.yamin@ntnu.no, basel.katt@ntnu.no

Abstract

Our world is becoming digitalized day by day, this leads to an increase amount of cyber-attacks by cyber-criminals. To tackle the increasing amount of cyber-attacks, cyber-security professionals are required in a high number. However, the required number of cyber-security professionals is not present. Despite the fact that academia and industry are trying to increase the number of cyber-security professionals, however, the tools and techniques used for cyber-security professional development are ineffective, as the gap between required and available cyber-security professionals is still increasing. One of the primary tools that is used in cyber-security professional development is hands-on cyber-security exercises. In this position paper, we will analyze the inefficiencies present in conducting hands-on cyber-security exercises and what can be done to reduce and eliminate those inefficiencies.

INTRODUCTION

Cyber-security exercises run attack and defense scenarios on a virtual and physical environment. A team of individuals, known as white time, creates the environment. In the environment, a team of attackers, known as red team, tries to exploit vulnerabilities present in the environment while a team of defenders, known as a blue team, tries to defend and prevent the attacks. In a recent study (Moore, Fulton, and Likarish 2017) researchers find out that such an exercise is very beneficial in cyber-security skill development. The researcher conducted knowledge surveys on participants before and after a cyber-security exercise and they found significant improvement in network security skills like ARP-Posioning, duplication in DNS entries and firewall/routers assessment as seen in figure 1.

These cyber-security exercises are usually conducted within hours and days but the time required to prepare these cyber-security exercises often spans up to months (Vykopal et al. 2017). This makes cyber-security exercises very costly and time consuming to be used in large scale to help reducing the growing cyber-security skills gap (Furnell, Fischer, and Finch 2017). Researchers divided cyber-security exercise in five phases to get the clear picture of cyber-security exercise development and execution steps. These five phases make the cyber-security exercise development and execution life cycle (Vykopal et al. 2017):

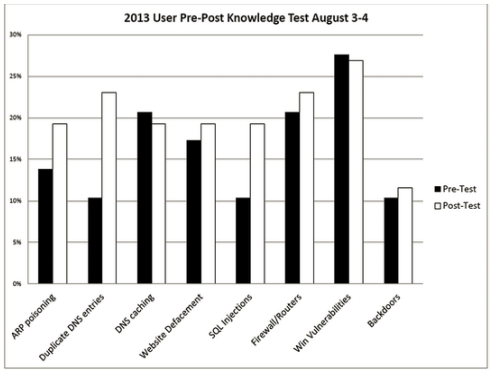


Figure 1: Participants knowledge test prior and after cyber-security exercise (Moore, Fulton, and Likarish 2017)

- **Preparation** It is the lengthiest part of cyber-security exercise development and execution. It involves setting up exercise objectives, defining a story, establishing points weight-age and creating a virtual environment for the cyber security exercise.
- **Dry run** The dry run is the testing of the developed virtual environment according to exercise objective by cyber-security experts. This process also takes long time due to the changes and adjustments required for the cyber-security exercise.
- **Execution** This is the phase where the actual cyber-security exercise takes place. Teams of attackers and defenders try to achieve the set of defined objectives.

Based upon the complexity of cyber-security exercise it can take hours to days.

- **Evaluation** At this phase teams performance is assessed according to the level of exercise objective completion. Feedback from participants is collected for future exercises. This phase usually takes few hours for its completion.
- **Repetition** The whole process is repeated for a set of new teams utilizing the lessons learned from the previous exercises and making necessary changes.

Conducting cyber security exercises in the described manner is a length, tedious and error-prone process (Beuran et al. 2018). Therefore, it is the position put forward that *cyber-security exercises are a good tool for cyber-security skill development but the inefficiencies in cyber-security exercise development and execution life cycle limits its ability to be widely used for cyber-security skill development.*

SURVEY OF THE LITERATURE

Researchers have been trying to reduce the inefficiencies present in conducting cyber-security exercise. In term of environment preparation phase of cyber-security exercise life cycle most of the current research is focused on reducing the time required for the preparation of virtual environment. Researchers developed multiple solutions for this problem two of them are Telelab (Willems and Meinel 2012) and SecGen (Schreuders et al. 2017) (security scenario generator). In TeleLab the researchers created multiple templates of virtual environment and developed an environment definition language, through which existing template are modified automatically to create new virtual environments for cyber security exercises. In SecGen researchers take the approach of TeleLab a bit further. Instead of defining the detailed environment schema using an environment definition language, SecGen takes the environment requirement i.e. number of machine, number of vulnerabilities, type of vulnerabilities etc. as input and randomly generate a virtual environment through the combination of existing virtual environment templates.

In the dry-run phase of cyber-security exercise recent studies (Ošlejšek et al. 2018) has shown that this phase has a lot of room for improvement. It is identified that team of human attacker and defenders does manual verification of the developed environment. That makes the process quite inefficient.

In execution phase multiple solutions in the literature are available for the execution of cyber-attacks and defense in cyber-security exercise execution. Two of the attack execution tools are Simulated Cognitive Cyber Red-team Attack Agent (SC2RAM) (Jones et al. 2015) and Scanning, Vulnerabilities, Exploits and Detection tool (SVED) (Holm and Sommestad 2016). SC2RAM is developed to mimic the red team execution steps in a cyber security-exercise. It can perform basic DoS (Denialof-Service) attack on a given network. It is still at prototyping stage and is being tested at Michigan cyber-range (Jones et al. 2015). SVED on the other hand utilizes freely available exploit tools such as metasploit and nmap and automate their operations to execute red team activities in a cyber security exercise. SVED is deployed at CRATE cyber range (Sommestad 2015). In term of cyber-defense process execution it is identified that in a cyber-security exercise skilled human professionals are required to conduct the exercise. Most of the cyber-defense research is focused on antivirus, antimalware, firewall and SIEM development, which left a lot of room for improvement in cyber-defense process execution without human involvement in a cyber-security exercise.

In term of evaluation phasen in most cyber-security exercises the theme of the exercise is CTF (Capture the flag competition). As the name suggests flags are used for point scoring and evaluation purposes. Flags contain some value of a random length when submitted to exercise or competition management systems points will be awarded. Based upon the number of points at the end of cyber security exercise or CTF competition, teams are evaluated. But the flag based evaluation mechanism is not ideal for overall performance analysis of individuals and teams. Flags only indicate that they either successful or not in completing a task, flags dont indicate at which approach they use or at which stage they feel difficult in completing the task. To tackle this problem KYPO (Čeleda et al. 2015) cyber range implemented an evaluation mechanism that is dependent upon event log monitoring. Event logs contains specific information about the activities that are being performed on a system. Based upon this information automatic evaluation is performed.

ANALYSIS AND DISCUSSION

The literature contains interesting solutions for the reduction of inefficiencies in cyber-security exercise development and execution life cycle. But these solutions have their cons as well. If we consider the autonomous generation of experimental environment by TeleLab and SecGen, we will notice that the environment which is generated is based upon an environment that is already available and if a participant already participated in an environment that is used for the creation of the environment then the participants will have unfair advantage.

The autonomous attack execution in the cyber-security exercise by SC2RAM and SVED gives a capability to the team of defenders to practice their skills without the availability of an actual attacker. But these tools are currently at an initial phase of their testing and have only basic capabilities. That makes them unsuitable for realistic training.

The scoring mechanism in KYPO cyber range is a very good approach for automatic evaluation of a participants performances in a cyber security-exercsie by monitoring the event logs created by the participants activity. However, this approach can only give a holistic view of participant performance, which is only good for calculating the overall performance of a participant,

not the performance of a participant at specific phase of the cyber-security exercise.

POTENTIAL SOLUTION

Research is being carried out to address the issues present in conducting operation based cyber-security exercises. Researchers in (Jones et al. 2015) presented a novel technique to model and execute an active opposition in a cyber-security exercise. The researchers discussed the missing element in the exercise environment that is active opposition. The researchers argued that: *The environment may have static defenses, such as access control or firewalls, or a fixed set of intrusion methods to defend against, but it typically lacks any active opposition that might adapt defensive or offensive actions (e.g., monitor logs, blocked connections, exploit switching or information gathering)*

The researchers presented techniques to model cyber-attack/defense adversaries and highlighted possible approaches that can be used in the implementation of such adversaries. Based upon this research, a tool is developed for autonomous execution of highly skilled red-team attackers SC2RAM: A Deployable Cognitive Model of a Cyber Attacker (Jones et al. 2015). This tool can train blue teamers to tackle cyber-security challenges and can configure and test defensive systems. SC2RAM is deployed at Michigan cyber-range to perform basic cyber-attack simulation, as it is still at prototype stage. On the other hand tools that mimic blue teams actions in a security exercise is still need to be implemented (Jones et al. 2015). We are planning to model the roles of white, blue and red teamers with respect to each other for the development of a cyber-security exercise platform that can assist execution of cyber-security exercises in a autonomous manner by autonomously preparing the exercise environment and generating autonomous adversaries according to the exercise environment. This will effectively remove the need of human adversaries and support staff required for conducting a cyber-security exercise. By reducing these inefficiencies cyber-security exercises can be conducted regularly at a wider scale, which will help in reducing the cyber-security skill gap currently present in industry.

CONCLUSIONS

From the above discussion it can be observed that multiple phases involved in cyber-security exercise development and execution can be automated to reduce cost and time required for conducting cyber-security exercises in an efficient manner. As it was suggested earlier *inefficiencies in cyber-security exercise development and execution life cycle limit its ability to be widely used for cyber-security skill development.* We can conclude that the roles of white, blue and red teamer in a cyber-security exercise need to be executed autonomously, which will increase the efficiency of preparation, execution and evaluation phases in cyber-security exercise life cycle .This will (1) reduce the cost and time require for conducting cyber-security exercise, (2) provide better training by always-available autonomous adversaries, and (3) make cyber-exercises computationally repeatable for conducting systematic training.

References

Beuran, R.; Tang, D.; Pham, C.; Chinen, K.-i.; Tan, Y.; and Shinoda, Y. 2018. Integrated framework for hands-on cybersecurity training: Cytrone. *Computers & Security*.

Čeleda, P.; Čegan, J.; Vykopal, J.; and Tovarňák, D. 2015. Kypo–a platform for cyber defence exercises. *M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence. NATO Science and Technology Organization*.

Furnell, S.; Fischer, P.; and Finch, A. 2017. Can't get the staff? the growing need for cyber-security skills. *Computer Fraud & Security* 2017(2):5–10.

Holm, H., and Sommestad, T. 2016. Sved: Scanning, vulnerabilities, exploits and detection. In *Military Communications Conference, MILCOM 2016-2016 IEEE*, 976–981. IEEE.

Jones, R. M.; OGrady, R.; Nicholson, D.; Hoffman, R.; Bunch, L.; Bradshaw, J.; and Bolton, A. 2015. Modeling and integrating cognitive agents within the emerging cyber domain. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, volume 20. Citeseer.

Moore, E.; Fulton, S.; and Likarish, D. 2017. Evaluating a multi agency cyber security training program using pre-post event assessment and longitudinal analysis. In *IFIP World Conference on Information Security Education*, 147–156. Springer.

Ošlejšek, R.; Vykopal, J.; Burská, K.; and Rusňák, V. 2018. Evaluation of cyber defense exercises using visual analytics process.

Schreuders, Z. C.; Shaw, T.; Ravichandran, G.; Keighley, J.; Ordean, M.; et al. 2017. Security scenario generator (secgen): A framework for generating randomly vulnerable rich-scenario vms for learning computer security and hosting ctf events. In *USENIX*. USENIX Association.

Sommestad, T. 2015. Experimentation on operational cyber security in crate. NATO STO-MP-IST-133 Specialist Meeting, Copenhagen, Denmark.

Vykopal, J.; Vizváry, M.; Oslejsek, R.; Celeda, P.; and Tovarnak, D. 2017. Lessons learned from complex hands-on defence exercises in a cyber range. In *Frontiers in Education Conference (FIE)*, 1–8. IEEE.

Willems, C., and Meinel, C. 2012. Online assessment for hands-on cyber security training in a virtual lab. In *Global Engineering Education Conference (EDUCON), 2012 IEEE*, 1–10. IEEE.

2.2 Make it and Break it - An IoT Smart Home Testbed Case Study

2.3 Cyber ranges and security testbeds: Scenarios, functions, tools and architecture

Computers & Security 88 (2020) 101636

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Cyber ranges and security testbeds: Scenarios, functions, tools and architecture

Muhammad Mudassar Yamin*, Basel Katt, Vasileios Gkioulos

Norwegian University of Science and Technology, Department of Information Security and Communication Technology, Teknologivegen 22, Gjøvik 2815, Oppland, Norway

ARTICLE INFO

Article history:
Received 21 March 2019
Revised 8 July 2019
Accepted 6 October 2019
Available online 7 October 2019

Keywords:
Cyber range
Security testbed
Scenarios
Cyber security
Security exercise

ABSTRACT

The first line of defense against cyber threats and cyber crimes is to be aware and get ready, e.g., through cyber security training. Training can have two forms, the first is directed towards security professionals and aims at improving understanding of the latest threats and increasing skill levels in defending and mitigating against them. The second form of training, which used to attract less attention, aims at increasing cyber security awareness among non-security professionals and the general public. Conducting such training programs requires dedicated testbeds and infrastructures that help realizing and executing the training scenarios and provide a playground for the trainees. A *cyber range* is an environment that aims at providing such testbeds. The purpose of this paper is to study the concept of a cyber range, and provide a systematic literature review that covers unclassified cyber ranges and security testbeds. In this study we develop a taxonomy for cyber range systems and evaluate the current literature focusing on architecture and scenarios, but including also capabilities, roles, tools and evaluation criteria. The results of this study can be used as a baseline for future initiatives towards the development and evaluation of cyber ranges in accordance with existing best practices and lessons learned from contemporary research and developments.

Contents

* Corresponding author.
E-mail address: muhammad.m.yamin@ntnu.no (M.M. Yamin).

https://doi.org/10.1016/j.cose.2019.101636
0167-4048/

1. Introduction

The recent security incidents worldwide have shown that there is an increase in the complexity and severity of cyber security threats. The attackers become more organized and the attack vectors are using more advanced and automated techniques and tools. The first line of defense against such attacks is increasing cyber security awareness in the public and security skills among the security professionals, in order to be ready and aware of the latest threat techniques and tools. These training programs include the execution of cyber security labs and exercises. In general terms, we define a cyber security exercise as a training exercise that runs attack and/or defense scenarios on virtual and/or physical environments with the aim of improving the attack and/or defence understandings and skills of the participants. Different groups of people are involved in preparing and executing such exercises. A group of individuals, known as *white team*, creates the training environment. Another group, known as *red team*, tries to exploit vulnerabilities present in the environment, while a third group, known as *blue team*, tries to defend the environment and prevent attacks. These are the main basic roles for those who are involved in an exercise. More comprehensive list of all roles within an exercises is discussed later. Please note that we use the term *security exercise* for any practical training or awareness activity.

Researchers divided a security exercise life cycle in five phases (Vykopal et al., 2017a), which are *preparation, dry run, execution, evaluation*, and *repetition*. In the first phase the exercise objectives, scenario story, scoring method, and the environment will be set up. In the dry run phase, the developed environment will be tested according to the exercise objectives. The execution phase involves running the exercise, in which the participants in the attacking and/or defending side will try to achieve their objectives. In the evaluation phase, the performance of the participants will be assessed based on the scoring method and learning objectives. Finally, in the last phase, the environment is cleaned and the whole process is repeated for a new exercise. It has been observed (Vykopal et al., 2017a) that security exercises are usually conducted and evaluated (execution and evaluation phases) in few hours up to a few days, while the preparation and dry run often take up to months for completion. This makes security exercises very costly and time consuming to be used in large-scale to help reducing the growing cyber security skills gap (Furnell et al., 2017).

In order to maintain and manage security exercises and their environment, a cyber range concept has been proposed. Recently, the concept and the term has attracted a great attention, but has been used differently in different contexts. Some use it to refer to a virtual environment, and others include other physical elements to a cyber range. It can refer to a university lab environment, or it can refer to a classified security exercise environment. There has been some attempts to study and classify the concept of a cyber range, e.g., the survey conducted by the Australian defense in 2013 (Davis and Magrath, 2013). Such studies provide a general background and classification of the term, though, (1) they do not cover all aspects of a cyber range system, e.g., architecture, management or scenarios, (2) they are outdated when it comes to cyber range technologies and tools, and (3) they do not discuss research trends and directions. Others, like Holm et al. (2015) and Qassim et al. (2017) are not generic enough and focus on specific exercise domains, like smart grids. To cover the gap in the literature, we conducted a systematic literature review on the topic of cyber range systems. The goals is to analyze the current state of the art within the topic of unclassified cyber ranges and security testbeds, and make recommendations regarding the architecture, capabilities, tools, the testing and training process, scenarios, and evaluation. The result can be used as a baseline for future initiatives towards the development, standardization and evaluation of cyber ranges in accordance with existing best practices and lessons learned from contemporary implementations.

The rest of this paper is structured as follows. In this next section, we present the related work covering the similar surveys and reviews conducted on this topics. In Section 3, we present the methodology and in Section 4 we discuss the results. In Section 5, we synthesize the result and present a general purpose architecture for a cyber range and summarize the research trends and directions. Finally, in Section 6 we discuss and conclude the paper.

2. Related work

During planning and writing this article, no other systematic literature review was found by the authors on the topic of cyber ranges and security testbeds. Yet, a multitude of survey articles has been identified with focus on specific application domains such as industrial control systems, mobile ad-hoc networks and cyber physical systems. Leblanc et al. (2011a) in 2011 presented an overview of cyber attack and computer network operations

simulation and modeling approaches. The discussed approaches have been identified within the open literature, and originate from governmental and academic efforts as well as from the private sector. These include, but are not limited to, ARENA, RINSE (Real-Time Immersive Network Simulation Environment), SECUSIM, and NetENGINE. In respect to research activities driven by the private sector and academia, the authors found that there are substantial efforts focused on cyber attack modeling, with constructive automated simulations. The results enabled the discovery of cyber attack patterns, with accuracy that is primarily dependent on the utilized models. Yet, the authors noticed that the governing parameters for most of these models are not validated against real world scenarios. Therefore, they mostly focused on specific artificial educational scenarios, rather than analysis of realistic cyber attacks in general. Furthermore, they overlooked also cascading effects on organizational or national scale.

Siaterlis and Masera (2009) in 2009 investigated available software for the creation of testbeds for Internet security research. The authors identified that numerous publications refer to prototypes rather than to software that is ready to be used for the creation of testbeds. Accordingly, they proposed a framework for feature based evaluation of the available software, as well as, they provided a literature review and comparison of state-of-the-art tools. This study excluded platforms that (i) share computational resources, (ii) focus only on simulation, (iii) are specific to wireless or sensor networks, (iv) run on a single computer, and (v) use custom hardware. The proposed framework consists of 13 basic and 6 compound features, including (i) distinction of roles, (ii) remote access, (iii) virtualization, and (iv) clean reconfiguration. The authors categorized their findings to overlay testbeds, including Planetlab and X-Bone, and cluster testbeds, including Grid'5000, Emulab, and ModelNet. They concluded that Emulab and Planetlab provide the most mature solutions for each testbed type and sufficient documentation for the development of dedicated testbeds, while Flexlab seeks to combine the best characteristics of the two approaches.

Davis and Magrath (2013) provided a survey of unclassified cyber ranges and testbeds, in a study completed in October 2013. The article provides an overview of background information in terms of supported functionalities and terminology, and also covers specific implementations originating from the military, public governments, and academia. SECUSIM, RINSE, ARENA, and LARIAT are some of the testbeds covered. The authors promoted hardware emulation as the most realistic approach, with simulations, on the other hand, providing increased flexibility and scalability advantages. Yet, as the study suggests, the middle ground providing parameterized support for emulation, simulation, and virtualization is increasingly explored, highlighting again Emulab and DETER as the most mature solutions.

Holm et al. (2015), Sun et al. (2018), Qassim et al. (2017), and Cintuglu et al. (2017) focused on testbeds dedicated to cyber physical systems, such as industrial control systems, SCADA, and the power grid. The articles investigated testbeds that have been proposed for scientific research and educational activities in aspects related to objectives, capabilities, architectural designs, integrated components, as well as implementation techniques for satisfying requirements. The authors also referred to these articles with explicit design and integration recommendations. Specifically, although the examined testbeds seem to target objectives such as vulnerability analysis, education, and tests of defensive mechanisms, these are not thoroughly described. In order for them to relate to specific architectural decisions, they must be refined and aligned with specific target vulnerabilities.

Balenson et al. (2015) focused on cyber security experimentation for the future. They worked on devising fundamental and new experimentation techniques for cyber security research. They concluded that new methods of research is required in cyber security focusing on just hardware and software is not enough. A community driven approach is required to constantly train the workforce in a dynamic cyber security enviroment. Carnegie Mellon University has developed a LMS (Learning Management System) which is called StepForward. It provides the opportunity to teach students both theoretical and practical cyber security skill set in a realistic environment by combining multiple choice questions with emulated labs. In term of cyber security competitions that use different cyber ranges and security testebeds, a comprehensive list is maintained at cybersecuritydegrees. Cyber security competitions are a good way to measure the effectiveness of cyber security training.

3. Methodology

The systematic literature review is a research review that aims at identifying, evaluating and synthesizing the existing literature of scientific work regarding a particular research question or topic. We decided to follow this method because it results in a credible, objective and unbiased evaluation of the current literature. This study has been conducted in accordance with the protocol described by Chitu and Kira (2010) in their article *"A Guide to Conducting a Systematic Literature Review of Information Systems Research"*. The protocol consists of eight consecutive steps, namely: (1) Define the purpose of the literature review, (2) establish a protocol among the participants, (3) search the literature, (4) perform practical literature screening, (5) perform quality appraisal, (6) perform data extraction, (7) synthesize the results, and (8) write the review. Three researchers participated in the literature review. In the following paragraphs, we provide the required insights of the adopted methodology in order to enhance the readability of the following sections and support future derivative or continuation studies.

3.1. Purpose of the literature review

The main purpose of this literature review is to study the concept of a cyber range system. Various aspects of a cyber range will be considered and a taxonomy will be created. Specifically, the objectives of this systematic literature review can be summarized as follows:

1. To identify and classify the capabilities and functionalities deployed within contemporary cyber ranges and security testbeds.
2. To collect and critically evaluate existing cyber ranges and security testbeds' architectural models.
3. To identify and classify scenarios, for training or testing, applied in cyber ranges and security testbeds.
4. To identify the different roles and teams associated with the execution of an exercise in a cyber range.
5. To identify and classify hardware and software tools utilized within contemporary cyber ranges and security testbeds.
6. To identify methods to evaluate different cyber ranges against a standard.
7. To study the research trends and directions on the topic of cyber ranges and security testbeds.

3.2. Establishing the review protocol

Three researchers participated in this systematic literature review from the period between March 2018 until January 2019. At the beginning, a discussion round resulted in the selection of the concrete methodology. The methodology was shared and studied by all members. After the selection and the study of the methodology, a concrete protocol for the execution of the review was established and a cloud based repository was created to maintain temporary files and document the conducted steps. Templates for

documentations, data extraction, and storing the results according to the established protocol were created as well.

3.3. Searching the literature

We followed the established protocol for systematic literature review in order to help the reproducibility of the study (Chitu and Kira, 2010) and provided the details in comprehensive methodology. We employed keywords based search technique in order to identify relevant literature. The keywords were selected very carefully in order to fulfill the purpose of the review described in 3.1. We performed a preliminary search using only the term"cyber range" and the results were not comprehensive. We noticed that there are some work that uses the name security testbed and security exercise when talking about a "cyber range" system. So, we decided to use the words "testbed" and "exercise". The collection of the literature was undertaken in accordance with the following parameters:

- Examined scientific databases: ACM digital library, IEEE Xplore, ScienceDirect, Springer Link, and Wiley online library.
- Utilized keywords (advanced search): "Cyber Range", "Security"+"Testbed", "Security"+"Test-bed", "Security Exercise".
- Publication period: 15 years (2002–2018).
- The total period of the literature review: March 2018–January 2019.

3.3.1. Search criteria

The search for security testbed results in a large amount of work, in which researchers conducted an experiment and they used a specific testbed for that purpose. These works were not of an interest for this review, and accordingly, we developed the list of rigorous inclusion and exclusion criteria. Thus, we listed the topics in which security testbeds were only mentioned to describe an experiment that was conducted in a particular domain, e.g., robots, UAV, and RFID testbeds. The application domains that can be included in the survey are vast, ranging from chemical-focused laboratories, to environmental systems. Covering all possible domains in one survey is not feasible and not possible. Therefore, we had to exclude some of the application domains to make it feasible, taking the maturity of the domain and the security relevance as two factors in this decision. Based on an internal discussion among the researchers, we decided on the list of inclusion and exclusion criteria that cover most important domains (not all), but make the survey feasible. For example, we cover the smart grid and industrial SCADA systems, but at the same time, we excluded transport systems, UAV, and robotics. The same applies for mobile infrastructure. In this case, we focused on application layer in the mobile testbeds, e.g., BYOD testbed scenarios, but we excluded infrastructure focused testbeds, like 4G/5G/GSM, and WIMAX testbeds. Thus, the identified literature was based on the following inclusion and exclusion criteria:

1. Inclusion criteria: The following inclusion criteria were applied in the review.
   - Articles written in English.
   - Security relevant testbed and exercises. Either presenting a whole cyber-range or a section/component of a cyber-range.
   - IoT (Internet of Things) related testbeds.
   - CPS (Cyber Physical Systems) and SCADA related testbeds.
   - Articles related to cyber-range federation.
   - Articles related to mobile applications testbeds.
2. Exclusion criteria: Based on the aforementioned discussion, in the following is the list of criteria we developed to filter out papers that are not within the scope of this review.
   - Articles that mention testbeds in the context of other work. The focus must be on the testbed.
   - Testbeds for UAV (Unmanned Aerial Vehicle).
   - Testbeds for RFID, NFC, and WIMAX.
   - Testbeds for cryptographic protocols.
   - Testbeds for robots.
   - Testbeds for trust related issues.
   - Testbeds focusing on security of structures, transportation, and security/safety of persons.
   - Testbeds focusing on climate change and the environment.
   - Testbeds for simulation of underwater sensor.
   - Conference abstracts, book reviews, conference info, discussion, editorials, mini reviews, news, short communications.
3. Quality appraisal: The focus of this paper is to study cyber ranges and security testbeds as a whole, in order to give insights to those who are designing, building, researching, standardizing, or operating a cyber ranges and security testbeds. For this reason, a relevant quality appraisal criteria is defined to cover and study cyber ranges and security testbeds as a whole. This survey can be followed by other surveys that focuse on a particular aspect of cyber ranges and security testbeds like scenarios, teaming, scoring etc. To ensure significant and quality contributions, we established an additional filtering step. We decided on the following list of topics related to general cyber range investigation, which are part of the taxonomy that we propose later in the paper. We noticed in the initial screening, that papers that use testbeds in the context of another research that is not related to the testbed itself, mentioned the scenario and an additional aspect, like scoring, monitoring, or management, depending on the research conducted. This means that papers that mentioned only one or two of the topics we specify, are not relevant. Therefore, significance and relevance were decided if articles include in their investigation at least three of the following five areas or topics of investigation:
   (a) Scenarios (architecture and story/behavior)
   (b) Monitoring and logging
   (c) Teaming
   (d) Scoring
   (e) Management (Id management, resource management, cyber range management, life cycle management)

   Additionally, the following quality assurance criteria were taken into consideration.
   (a) Originality of the work.
   (b) Quality of presentation.
   (c) Scientific soundness and method.
   (d) Papers that have been cited should be included in the survey. This rule is exempted from papers that were published recent, i.e., less or equal then two years. The citation data as of August 10th 2018 is parented in appendix Table 13.

3.4. Practical literature screening

Based on the aforementioned steps and criteria, we conducted the practical literature screening. The following rounds were resulted.

1. Round 1: Collection of the literature was conducted in March 30th. It resulted in a total entries of **385**.
2. Round 2: Elimination of duplicates was conducted in April 25th, and resulted in a total entries of **310**.
3. Round 3: Back tracing additional entries from the citations of the current articles was conducted in June 20th. It resulted in a total number of entries **341**.
4. Round 4: Quality appraisal was conducted on August 10th, and resulted in the total number of articles **100**.

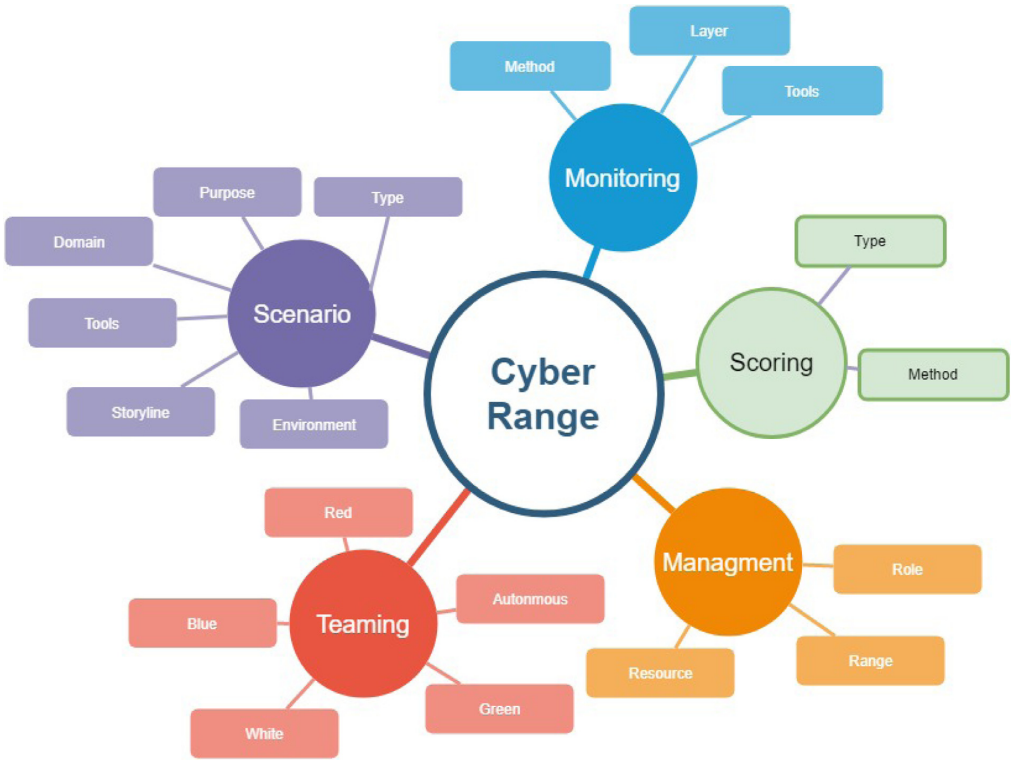


Fig. 1. Cyber Range taxonomy.

3.5. Classification and data extraction

Based on the work we have done in developing a cyber range and after the first screening of the literature, we propose an initial taxonomy to classify cyber ranges as shown in Fig. 1. A new updated taxonomy is developed after the survey was conducted and will be presented in Section 4.2. In the following is a short description of each concept.

1. **Scenarios**
   A scenario defines the execution environment as well as the storyline that indicates the execution steps of a test or a training exercise. It accurately represents the operational environment and training requirements, and drives training execution to ensure the achievement of training objectives. The scenario describes and provides documentation, summaries, action orders, etc., to ensure the representative operational context supports testing and training objectives (Staff, 2012). We classify a scenario to extract information about what is the purpose of the exercise, or test? Where an exercise, or a test, is executed? How an exercise, or a test, is executed? And which tools are used in the execution of a scenario? answers to these questions are given below.
   (a) **Purpose**
   The purpose explains what are the objectives of the scenario, i.e. the execution of a cyber security training exercise or the experimentation validation of new cyber security tools and techniques. Based upon the scenario objectives, scenario environment is developed, details of which are given below:
   (b) **Environment**
   The scenario environment is the topology where the scenario is executed. The scenario depends upon the exercise and experiment objectives. If the exercise is an operation-based, then the environment will be a technical infrastructure, i.e., computer based, physical, virtualized or hybrid. If the exercise is a table-top or discussion based the environment can be non computer based (Gurnani et al., 2014). In a table/top based cyber security exercise, a cyber scenario is discussed and the decision making ability of the exercise participants is evaluated. It can be computer aided or can be executed without the use of any digital equipment.
   (c) **Storyline**
   A storyline of a scenario tells a single or multiple stories about how the exercise will be executed. It includes the development of relevant actions and events that constitute the scenario and how these are connected to generate the whole narrative of a scenario. This allows the overall understating and controlling of a big technical scenario, and gives the ability to critically evaluate the exercise, or test, outcome (Staff, 2012). In term of experimental validation of new technologies, single or multiple test cases can be executed for research are investigation.
   (d) **Type**
   The type of the scenario indicates whether the scenario is static or dynamic. We define a scenario to be static, if it includes a static environment, and no changes are applied during the execution of the exercise. This means that the storyline does not include any dynamic components that change over time. A dynamic scenarios are scenarios that in-

clude, besides the static environment, a dynamic component that will make changes during the execution of the scenario. For example, a simulator, or a traffic generator that can be injected, or executed, during the exercise.

(e) **Domain**
The domain indicates the application domain of the scenario, e.g., IoT, network, cloud etc.

(f) **Tools**
The tools which are used in the development of a scenario. This includes the tools which are needed for the creation of the environment of the scenario, or the tools which are used in the development of a storyline.

2. **Monitoring**
Monitoring includes the methods, the tools and the layers at which real time monitoring of cyber security exercises and tests are performed (Staff, 2012). Monitoring of cyber security exercise participants is performed by designated observers (Kick, 2014). The methods that the observers employ, the tools that they use and the layers at which they perform monitoring are further classified:

(a) **Methods**
This classifies methods employed to monitor the cyber security exercise and tests, i.e., how the cyber security exercise, or the test, is monitored. Either automatically with the use of tools that gather data for analysis, or manually by human observers.

(b) **Tools**
This classifies the software and hardware tools that can be used for monitoring of cyber security exercises and tests. The software and hardware tools may include security information and event management (SIEM) solutions and intrusion detection systems etc.

(c) **Layers**
This classifies the layer at which monitoring is being performed. Depending on the type of an exercise, monitoring can be performed at multiple TCP/IP layers, in case of an operation-based exercise; or at an abstract social layer, in case of a table-top exercise.

3. **Teaming**
In a cyber security exercise, teaming includes an individual and a group of individuals that design, develop, manage and participate in a cyber security exercise or a test (Schepens et al., 2002). Based upon a team's role in a cyber security exercise different colors are assigned to them to identify their role (Vykopal et al., 2017b). Details of which are given below:

(a) **Red team**
Red teaming is a form of information security assessment in which cyber- security adversaries are modeled to identify vulnerability present in a system during an exercise or a test (Wood and Duggan, 2000). The red team is responsible to identify and exploit potential vulnerabilities that are present in the exercise environment.

(b) **Blue team**
Blue teaming is a form of active defense against an active attack on a cyber security exercise and test environment (White and Williams, 2005). The blue team is responsible to identify and patch potential vulnerabilities that can be exploited by a red team.

(c) **White team**
A white team designs the exercise and experiment scenario, objectives, rules and evaluation criteria. They set a set of rules of engagement between red and blue team, inject the vulnerabilities in the environment for patching and exploitation; and sometimes they act as instructors to give hints to the participating teams (Vykopal et al., 2017b).

(d) **Green team**
A green team is responsible for the development, monitoring and maintenance of the exercise infrastructure designed by the white team. They are also responsible for fixing bugs and crashes in the infrastructure occurred during an exercise execution (Vykopal et al., 2017b).

(e) **Autonomous teams**
Team roles that are being automated by different tools and techniques are considered as autonomous teams. For example, Secgen (Schreuders et al., 2017) is used for the automation of scenario environment development which is the role of a green team, and SVED (Holm and Sommestad, 2016) is used for the role automation of a red team.

In some cyber security exercises, additional teams are included, which are exercise/specific and not present in cyber security exercise life cycle (Vykopal et al., 2017b). Details of which are given below:

(a) **Orange Team**
Orange team members assign different technical tasks to blue team members during the exercise. Blue team members can earn points if they are able to successfully complete the tasks.

(b) **Purple Team**
Purple teams perform the communication role between multiple exercises teams. They do information sharing to increase the exercise effectiveness. This enhances the effectiveness of a red team in attacking the exercises environment and increases the capability a blue team in defending the network.

(c) **Yellow Team**
Yellow team members simulate the behavior of normal users that are using the infrastructure created by the green team. They perform tasks like generating legitimate network traffic which can be used by red and blue teams in attack and defense.

4. **Scoring**
Scoring uses data from monitoring systems in order to give performance related semantics to the low level technical events observed during monitoring of cyber security exercises and tests. Some scoring indicators might not depend on technical monitoring events, like flags or over-the shoulder evaluation mechanisms. The scoring mechanism is also used to measure the teams and test progress during an exercise, or a test (Vykopal et al., 2017b). The methods and tools used in the scoring mechanism are further classified:

(a) **Methods**
This classifies whether the scoring is done based upon achieving a specific objective, i.e flags, or it is done by analyzing logs that are generated during cyber security exercises or tests.

(b) **Tools**
This classifies the software and hardware tools that are used for scoring of cyber security exercises or tests. The tools may include flags submission dashboards, log analyzers, etc.

5. **Management**
Management involves the assignment of roles and duties to individuals and teams. Allocation of computational and other resources required for conducting a cyber security exercise, or a test, and the overall management of the cyber range.

(a) **Role management**
Role management classifies the methods, tools and techniques with which the identities and roles of individuals and teams involved in a cyber security exercise, or a test, are managed.

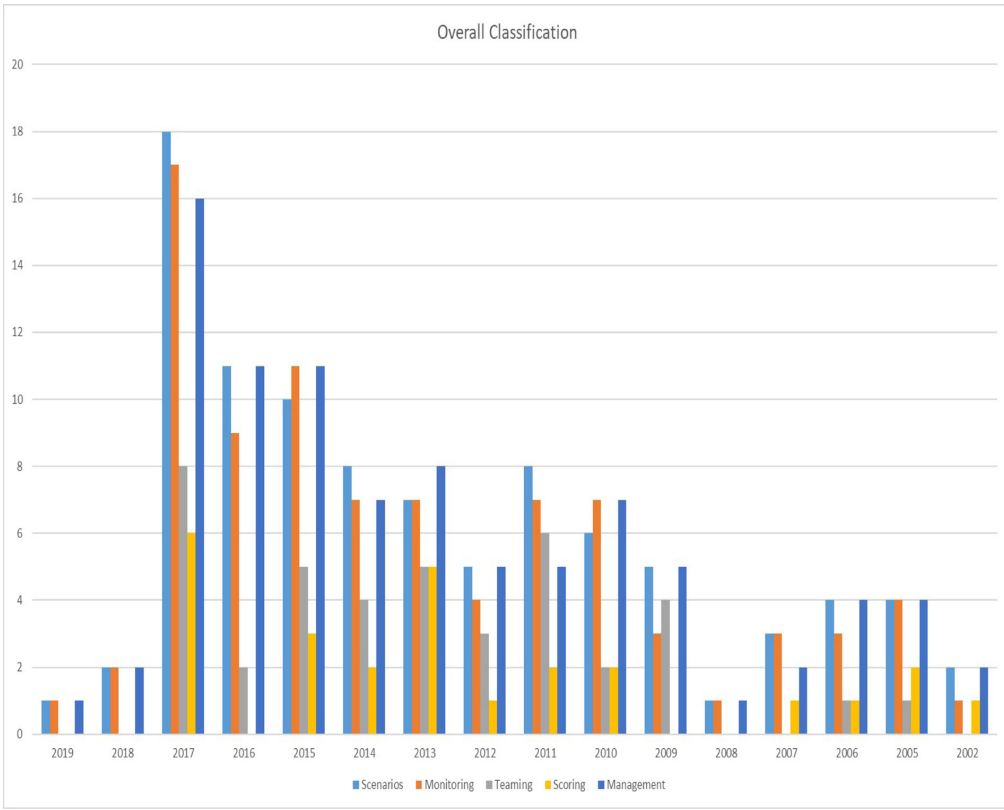


Fig. 2. Overall classification of cyber ranges and security testbeds capabilities with respect to years.

(b) **Resource management**
Resource management classifies the computational resources like processing frequency, memory and disk space required for conducting cyber security exercise, or a test.

(c) **Range management**
Range management classifies the methods, tools and techniques with which the holistic view of overall cyber security exercise, or a test, is presented in portals and dashboards.

4. Analysis of results

In this section we present and discuss the results of the literature review. First, we discuss how the main capabilities identified in the taxonomy, presented in Section 3.5, have been investigated, or considered in the literature. Then, we discuss, in more details, the architecture of contemporary cyber ranges, scenarios, teaming, evaluation criteria, tools used, and future directions presented in major work.

4.1. General capabilities

As per our selection strategy presented in 3.3, a classification of the capabilities and functionalities deployed within contemporary cyber ranges and security testbeds is presented in Fig. 2 and Table 1. We identified that the capability that was mostly investigated in the literature is *scenarios* with 94 papers that include details about scenarios. The second most prominent capability is management with 91 papers. Then there were 86 papers that have details about the monitoring infrastructure, 41 papers contain details about teams, and only 26 papers have details about the scoring mechanism.

In order to analyze the evolution of these different capabilities over time, Fig. 2 depicts how the interest of different capabilities has increased steadily, with few exceptions, since 2002. It can be noticed that in the period between 2007 and 2008 the number of publications dropped, and then continued increasing in 2009 until 2017. This is correlated with the fact the major cyber ranges, like the US National Cyber Range have started development in the period between 2008 and 2009. Before that date, most of the work was conducted in terms of general purpose security testbeds. Around the time the US National Cyber Range (Palleschi, 2010), among FIRE(Future Internet Research and Experimentation) (Gavras et al., 2007) in Europe started which aimed to interconnect existing security testbeds. Due to which many researchers started looking at the new “Cyber Range” concept, which explains the dip in publication around 2008. It is worth mentioning that due to the fact that the screening happened in the second quarter in 2018, the figures related to 2018 is not complete. Also, there were few papers that were found during the search with publication date scheduled in 2019.

4.2. New taxonomy

The taxonomy presented in Section 3.5 is good for identifying the general capabilities of cyber ranges and security testbeds. However, after reviewing the selected papers and analyzing the col-

Table 1
Capabilities and functionalities deployed with in contemporary cyber ranges and security testbeds.

Paper	Scenarios	Monitoring	Teaming	Scoring	Mng.
Čeleda et al. (2015); Childers et al. (2010); Ernits et al. (2015); Maennel et al. (2017); Ošlejšek et al. (2017); Vykopal et al. (2017a,b) Davis and Magrath (2013); Leblanc et al. (2011b)	✓	✓	✓	✓	✓
Border (2007); Chadha et al. (2016); Domínguez et al. (2017); Farooqui et al. (2014); Flauzac et al. (2016); Sun et al. (2018) Chandra and Mishra (2019); Jung et al. (2008) Richmond (2005) Alves et al. (2016) Edgar et al. (2011); Furfaro et al. (2017); Xypolytou et al. (2017) Gao et al. (2013); Gunathilaka et al. (2016); Louthan et al. (2010); Mirkovic et al. (2010); Rubio-Hernan et al. (2016); Shumba (2006); Tsai and Yang (2018) Al-Ayyoub et al. (2015); Almalawi et al. (2013); Alvarenga and Duarte (2016); Ashok et al. (2016); Liljenstam et al. (2005); Rahman et al. (2009); Subaşu et al. (2017) Edgar and Manz (2017); Lee et al. (2017); Moraes et al. (2014); Pfrang et al. (2016); Siaterlis and Genge (2014); Soupionis and Benoist (2015); Volynkin and Skormin (2007) Bergin (2015); Gao et al. (2015); Hahn et al. (2013); Herold et al. (2017); Jirsik et al. (2014); Kouril et al. (2014); Miciolino et al. (2015); Tsai et al. (2017) Caliskan et al. (2017); Chow et al. (2010); Fovino et al. (2010); Genge et al. (2012); Siboni et al. (2016); White et al. (2002) Cintuglu et al. (2017); Koutsandria et al. (2015); Mallouhi et al. (2011) Barcellos et al. (2012); Hu et al. (2006)	✓	✓			✓
Line and Moe (2015); Patriciu and Furtuna (2009); Urias et al. (2012); Willems and Meinel (2011, 2012) Benzel et al. (2009, 2006); Damodaran and Smith (2015); Edgar and Rice (2017); Morris et al. (2011)	✓	✓	✓		✓
Braidley (2016); Ferguson et al. (2014); Li et al. (2009); Marshall (2009); Murphy et al. (2014); Pham et al. (2016); Yasuda et al. (2016)	✓		✓		✓
Siaterlis and Masera (2010); Stites et al. (2013)	✓	✓		✓	✓
Kuhl et al. (2007)	✓	✓		✓	
Glumich and Kropa (2011); Siaterlis et al. (2011)	✓	✓	✓		
Chiang et al. (2013); Gephart and Kuperman (2010)		✓	✓		✓
Rossey et al. (2002); Snyder (2006)	✓			✓	✓
Sommestad (2015)		✓	✓	✓	✓
Hoffman et al. (2005); Rursch and Jacobson (2013b); Sommestad and Hallberg (2012)	✓		✓	✓	✓
Antonioli et al. (2017); Labuschagne and Grobler (2017); Silva et al. (2014)	✓	✓	✓	✓	
Doupé et al. (2011); Reed et al. (2013)	✓		✓	✓	
Vigna et al. (2014)		✓	✓	✓	
Alfieri et al. (2005)		✓		✓	✓
Rursch and Jacobson (2013a)		✓	✓	✓	✓

lected data, we identified that the taxonomy that we used to identify the general capabilities was not sufficient in presenting cyber ranges and security testbeds functionalities in depth. Therefore, we are proposing a new updated taxonomy for presenting the functionality of cyber ranges and security testbeds based upon the collected data. The developed taxonomy is parented in Fig. 3. In this section, we will focus our discussion on the new elements that were added to the new taxonomy. We will refer to the papers that included information about these new concepts. In general, it is worth mentioning the following two main changes compared to the initial taxonomy. First, due to its importance and being related to different other concepts, environment is presented on its own, separately from scenarios. Second, we added the learning concept, as we noticed that learning modules were mentioned repeatedly in cyber ranges. Scoring is considered as a sub-element of the learning module, and thus added as a sub-concept to the learning concept. Apart from that, we expanded the scenario concept with the scenario lifecycle, the management with command&control, and data storage concepts.

4.2.1. Scenarios

In this section, first, we discuss the cyber security scenario lifecycle management. It involves creating, generating, editing, deploying and executing a cyber security scenario. The following work (Alvarenga and Duarte, 2016; Furfaro et al., 2017; Gephart and Kuperman, 2010; Hu et al., 2006; Marshall, 2009; Ošlejšek et al., 2017; Tsai and Yang, 2018; Yasuda et al., 2016) have specialized components in their architecture to create and edit cyber security scenarios. They mostly have a designer dashboard in which different components of a scenario are presented, and can be used to develop new scenarios. The works in Lee et al. (2017); White et al. (2002) have components to generate cyber security scenarios using different automation techniques. The scenarios are created mostly in a human and machine readable language like XML and JSON, which is then executed on a compiler to deploy the scenario. The works presented in the these papers (Furfaro et al., 2017; Herold et al., 2017; Hu et al., 2006; Marshall, 2009; Tsai and Yang, 2018; Vigna et al., 2014) included special scenario deployment component which is responsible for deploying network resources, like routers and firewalls, and relevant applications, like vulnerable software. For scenario execution, (Alvarenga and Duarte, 2016; Ernits et al., 2015; Gao et al., 2013; Moraes et al., 2014; Rubio-Hernan et al., 2016) have module that can control the scenario flow, like start, stop and pause scenario execution. Works in Alvarenga and Duarte (2016); Willems and Meinel (2011) have orchestration modules that combine multiple components to execute a scenario. Finally, (Alvarenga and Duarte, 2016; Barcellos et al., 2012; Chandra and Mishra, 2019; Ernits et al., 2015; Furfaro et al., 2017; Gao et al., 2013; Jung et al., 2008; Labuschagne and Grobler, 2017; Rursch and Jacobson, 2013b; Siboni et al., 2016) have components that are used to generate different events within the scenario execution to make the scenario more dynamic and realistic. These events can be the launch of automatic attacks, like in Chandra and Mishra (2019); Ernits et al. (2015); Furfaro et al. (2017), or can represent traffic generation, like in Labuschagne and Grobler (2017); Rursch and Jacobson (2013b).

Fig. 4 shows the evolution of the different purposes of scenarios, i.e., testing, education, and experiment. It can be seen that testing and education are gaining a lot of attention in the last few years, particularly testing. With respect to the scenario type, we can distinguish between both static and dynamic scenarios (cf. Section 3). Fig. 5 shows the evolution of scenario types discussed in the reviewed papers. It can be seen that before 2011 static scenarios, in which the scenario story was not discussed but included only the static topology, was dominant. Since 2011, cyber range scenarios started to add the dynamic component, in which the sto-

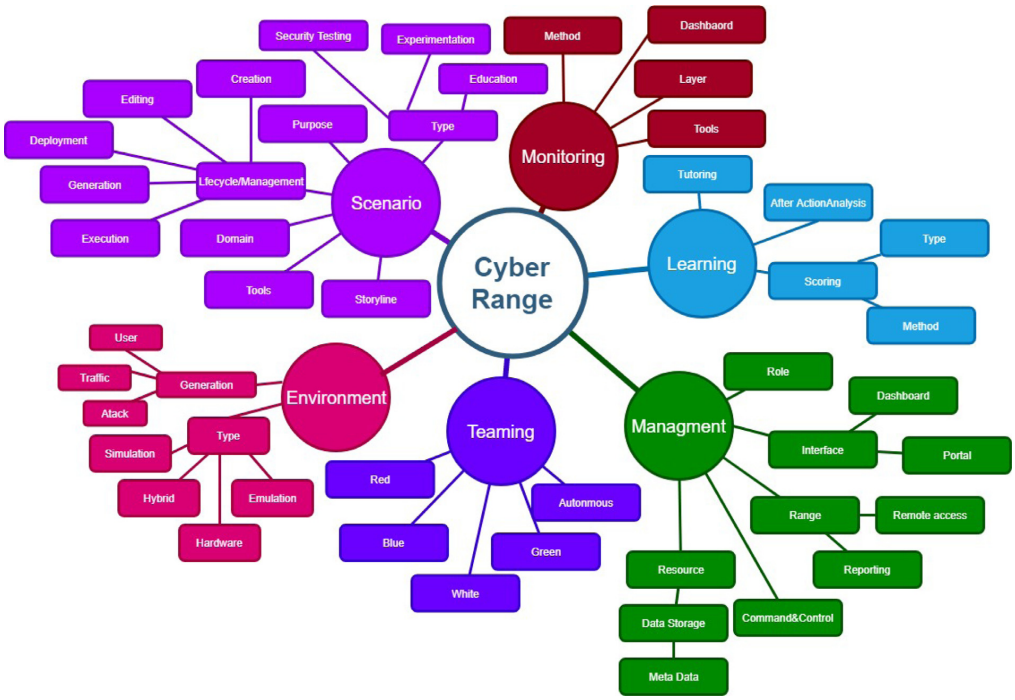


Fig. 3. Updated taxonomy of a cyber range.

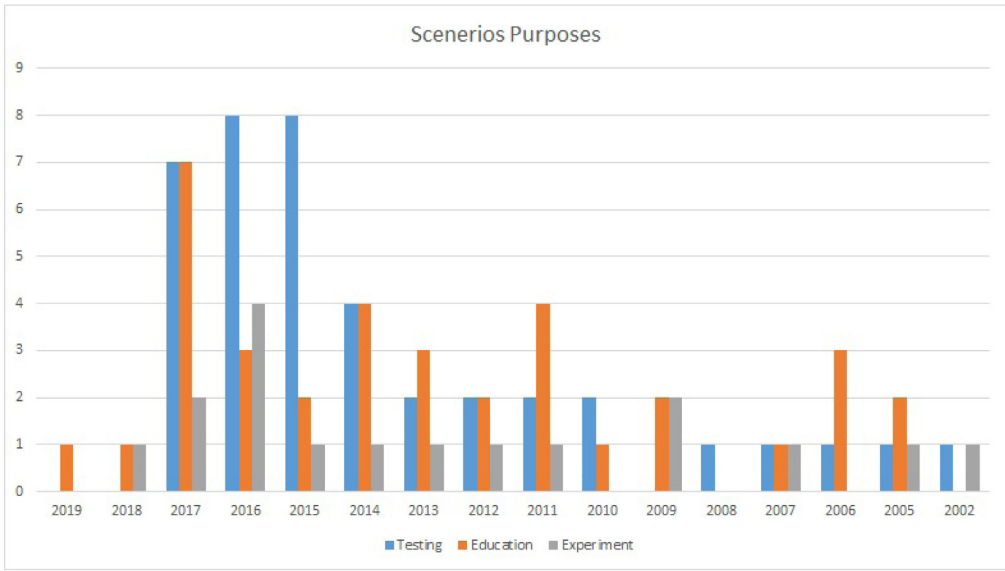


Fig. 4. Classification of cyber-ranges and security testbeds based upon the scenarios purpose.

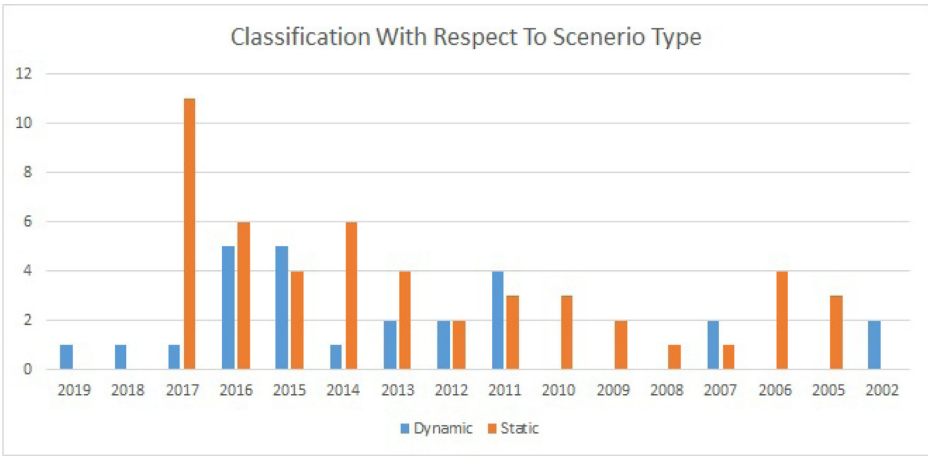


Fig. 5. Classification of cyber-ranges and security testbeds based upon the scenario type which they support.

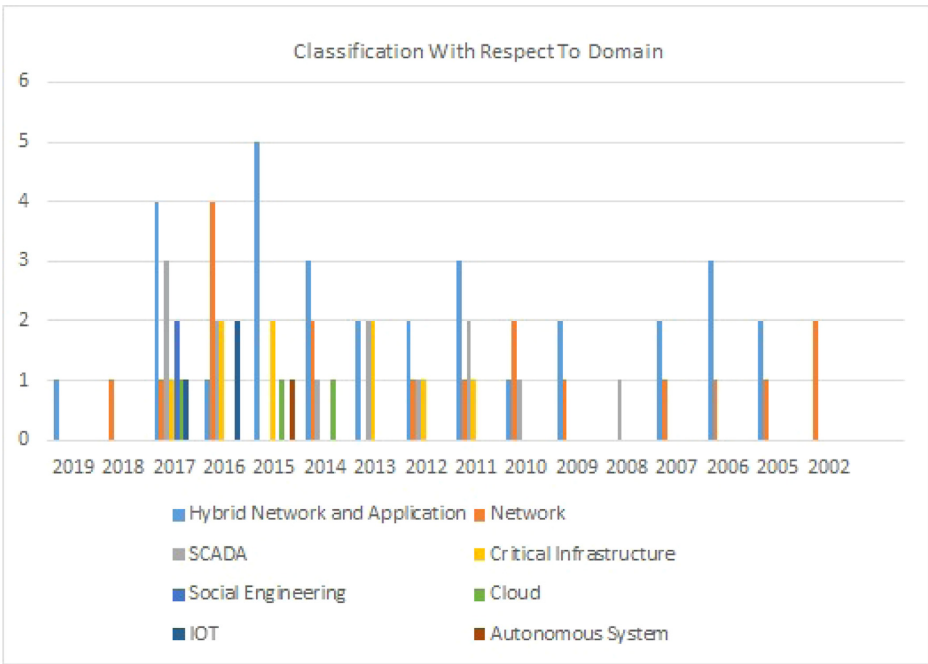


Fig. 6. Classification of cyber-ranges and security testbeds based upon the scenarios domains.

ryline and the behavior are specified. This shows an advancement in the specification and execution of scenarios in cyber ranges and security testbeds.

Finally, when it comes to the domains of the scenarios, Fig. 6 shows the different application domains, in which scenarios are specified. Those domains are (1) hybrid network applications, (2) Networking, (3) SCADA systems, (4) social engineering, (5) IoT systems, (6) critical infrastructure, (7) Cloud based systems, and (8) autonomous systems. The figure indicates that networking systems were the main application domain for cyber ranges and security testbeds, SCADA system started to gain attention from 2010, and in recent year cyber ranges and security testbeds have covered most application domain aforementioned. In Table 2 we present scenario samples from each application domain, including the purpose, the environment, the storyline topic, and tools used.

4.2.2. Monitoring

In this section, we will talk about the methods, dashboard, layer and tools that are used for monitoring of cyber ranges and security test beds. Works in Alvarenga and Duarte (2016); Čeleda et al. (2015); Herold et al. (2017); Siboni et al. (2016); White et al. (2002) use different data collection and analysis modules for monitoring purposes. While (Alfieri et al., 2005; Chandra and Mishra, 2019; Furfaro et al., 2017; Labuschagne and Grobler, 2017;

Table 2
Scenarios and their purpose in different domains.

Id	Domain	Paper	Purpose	Environment	Storyline	Tools
1	Hybrid Network and Application	Herold et al. (2017)	Education	Hybrid	Network topology configuration for students	XEN, CISCO routers
2	Networks	Benzel et al. (2006)	Experiment	Emulation	DDoS, Worm Behavior, Early Routing Security experiments	Emulab
3	IOT	Siboni et al. (2016)	Testing	Hardware	Bring your own device scenario testing for enterprises	Smart Wacthes, google glass, printers
4	Critical Infrastructure	Genge et al. (2012)	Testing	Emulation	DoS attack on a powergrid	Emulab
5	SCADA	Fovino et al. (2010)	Experiment	Hardware	DoS, ICT worm, Phishing, DNS poisoning experiments	ABB 800F, OpenPMC (PLC), Emerson MD, Turbogas Subsystem, Turbogas Control Subsystem, Steam cycle Subsystem Plant Control subsystem
6	Social Engineering	Braidley (2016)	Testing	Simulation	Social engineering testing for enterprises using employee online data	Netkit
7	Cloud	Jirsik et al. (2014)	Experiment	Emulation	DDoS attack testing on different network toplogies	OPENNEBULA, Netflow, Low Orbit Ion Canon
8	Autonomous System	Bergin (2015)	Testing	Simulation	Military autonomous vehicle DDoS attack testing	JAUS messages, JSONS, NOSQL, PYTHON, RUBY, NODE.JS, JAVASCRIPT,XML, REST FULL WEBAPI

Lee et al., 2017; Mallouhi et al., 2011; Tsai and Yang, 2018) use event logging mechanism and analysis techniques for monitoring purposes. Alfieri et al. (2005); Chandra and Mishra (2019); Herold et al. (2017); Lee et al. (2017); White et al. (2002) have specialized dashboards preset in the architecture to present the monitored information. Alfieri et al. (2005); Chandra and Mishra (2019); Furfaro et al. (2017); Labuschagne and Grobler (2017); Lee et al. (2017); Mallouhi et al. (2011); Tsai and Yang (2018) use mainly application layer protocols for data collection, while in Alvarenga and Duarte (2016); Čeleda et al. (2015); Herold et al. (2017); Siboni et al. (2016); White et al. (2002), authors use network layer protocols for monitoring purposes. In term of tools, these cyber ranges and security testbeds uses multitude of different tools, a detailed list of those tools is provided in Section 4.4.5.

4.2.3. Learning

In this section, we will discuss the learning and tutoring component, the after action analysis mechanism and scoring techniques present in different cyber ranges and security testbeds. Authors in Ernits et al. (2015); Hu et al. (2006); Subaşu et al. (2017); Willems and Meinel (2012) have a tutoring or learning management system present in their functional architecture. These tutoring systems mainly consists of text, images and multimedia clips. Authors in Alvarenga and Duarte (2016) have an after action analysis module that operates over the complete experimental data set. Its main attribution is data pre-processing and calculation of a supplemental set of metrics derived from experimental bulk data. In term of scoring mechanisms, the work in Vigna et al. (2014) uses a score bot that is responsible for monitoring the status of the services and calculates the score for each team. (Ernits et al., 2015) use a scoreboard in which progress of participants is presented based upon the task they completed. Details of scoring mechanisms and tools are presented in Section 4.4.8.

4.2.4. Management

In this section we present the roles, interfaces, range management, command and control, and resource management within the reviewed cyber ranges. Different teams perform different roles within the cyber range and security testbeds, we shared the details of different teams in Section 4.2.5. In term of interfaces, (Alves et al., 2016; Čeleda et al., 2015; Tsai and Yang, 2018) have dashboards that graphically presents the current state of cyber range and security test beds; while (Alfieri et al., 2005; Liljenstam et al., 2005) have special portals for communication. For interfaces, the work in Siboni et al. (2016) has a reporting module that is responsible for starting, enrolling devices and simulating. Authors in Willems and Meinel (2012) have a remote desktop component that is used to initialize, start, monitor, and terminate remote desktop connections to machines. The work in Čeleda et al. (2015) uses an API to manage remote access between different components of a cyber range, and authors in Genge et al. (2012) use a proxy that enables running remote code and integrate different physical components. Mallouhi et al. (2011) have a control component that represents the main command and control for all the resources and services present within the security testbed. The works in Bergin (2015); Furfaro et al. (2017); Ošlejšek et al. (2017); Rursch and Jacobson (2013b); Vigna et al. (2014); White et al. (2002) have data storage modules that store elements like scenario models, attack tools, exercise and experiment rules and results; while authors in Alfieri et al. (2005) have a module for cataloging different attack and defense scenarios.

4.2.5. Teaming

Fig. 7 presents the different types of teams that participate in activities conducted at cyber ranges and security testbeds. The main types of teams are, red, blue, white, green, and autonomous teams. Red and blue teams correspond to red and/or blue exercise types. Autonomous teams, in which some activities of a team is performed by an autonomous system, or agent, have gained an attention since 2014. Autonomous teams are added as a separate type to study the status of using automation of different team roles in cyber security exercises.

4.2.6. Environment

In this section, we discuss the concept *environment*. This include the scenario execution environment type and different event generation tools that are used in scenario environments. Works in Alvarenga and Duarte (2016); Border (2007); Childers et al. (2010); Ernits et al. (2015); Flauzac et al. (2016); Snyder (2006); Subaşu et al. (2017); Vigna et al. (2014); Willems and Meinel (2011, 2012) use an emulated environment for scenario execution. Their scenarios usually contain virtualized nodes running different services. Authors in Antonioli et al. (2017); Bergin (2015); Čeleda et al. (2015); Chandra and Mishra (2019); Gephart and Kuperman (2010); Jung et al. (2008); Labuschagne and Grobler (2017); Mallouhi et al. (2011); Moraes et al. (2014); Pfrang et al. (2016); Sommestad (2015); Soupionis and Benoist (2015); Tsai and Yang (2018) use Hybrid environment for the execution of cyber secu-

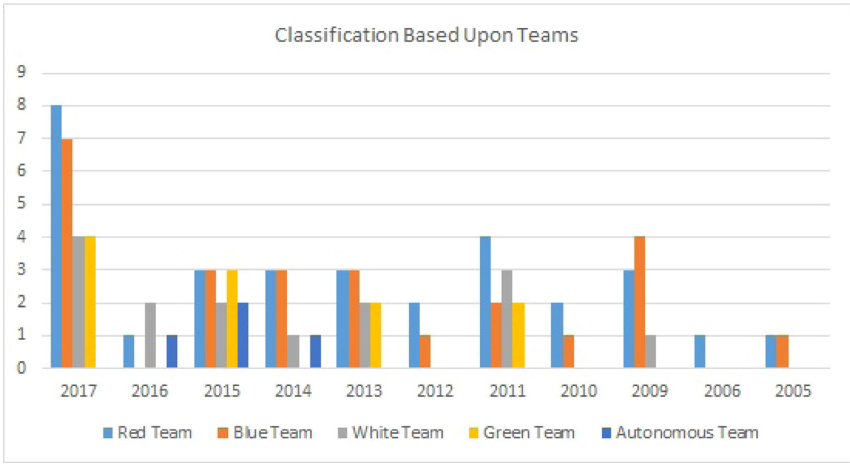


Fig. 7. Classification of cyber-ranges and security testbeds based upon the teams.

rity scenarios. The environment contains both hardware, virtualized and simulated elements. The hardware usually contain specialized equipment, like PLCs that are difficult to emulate. In term of hardware based environments, works in Alves et al. (2016); Ashok et al. (2016); Furfaro et al. (2017); Gao et al. (2013); Lee et al. (2017); Louthan et al. (2010); Miciolino et al. (2015); Rubio-Hernan et al. (2016); Rursch and Jacobson (2013b) use actual hardware cyber security scenario execution. These scenarios are mostly relate to IoT, SCADA and critical infrastructure. Works in Al-Ayyoub et al. (2015); Barcellos et al. (2012); Genge et al. (2012); Jung et al. (2008); Li et al. (2009); Liljenstam et al. (2005); Subaşu et al. (2017); White et al. (2002) use different simulation and modeling techniques for cyber security scenario execution. Details of different event generation tools, like traffic and user behavior, are presented in Section 4.4.6 and 4.4.7.

Fig. 8 indicates the type of the runtime environment that are used in cyber ranges and security testbeds in the last 15 years. It can be sees that HW-only equipment has not been used widely. From 2002 until 2015, there has been only one paper presented a pure HW run time environment. Emulation has been, and still, used widely in cyber ranges and security testbeds. Since 2016, hybrid approaches have also become widely used.

4.3. Evaluation

In this section we discuss the different methods that have been used in order to evaluate cyber ranges and security testbeds. Out of 100 papers, 8 have details about the evaluation techniques employed in the cyber ranges and security testbed. Four papers used quantitative evaluation methods to evaluate the cyber ranges and security testbeds as a whole. The other four used qualitative methods to evaluate the functionality of cyber ranges and security testbed by executing specific tests on them.

4.3.1. Overall and performance evaluation

The following papers applied quantitative evaluation methods to evaluate the cyber ranged and security testbeds as a whole, especially the performance.

1. Researchers in Herold et al. (2017) based their evaluation on the time for testbed generation. They measured the time required for generating an infrastructure of 3 router, 1 switch and 4 PCs' for an educational scenario. The total time required for generating the testbed was 42 min 32 s.
2. Researchers in Yasuda et al. (2016) applied similar method and found out that the network environment generation tool took about 1624s to construct an environment consisting of three segments, i.e., the client, internal-server, and DMZ segments. For a single team in the cyber security exercise, there were five instances in total for each segment: the firewall, Windows 7 client, file server, database, and DNS/mail instances. It took about 6754s to finish the construction of identical segments for four teams for the conducted cyber security exercise.
3. In a distributed system scenario in White et al. (2002), the researchers used Netbed's batch system to evaluate every possible combination of 7 bandwidths, 5 latencies, and 3 application parameter settings on four different configurations on a set of 20 nodes. The result was performing a total of 420 different tests in 30 h, averaging 4.3 min each.
4. In simulation environment for validating protocols for distributed applications, researchers in Barcellos et al. (2012) employed similar quantitative evaluation methods, which is also based upon time requirements.

4.3.2. Functional evaluation

The following papers applied qualitative evaluation methods to evaluate the functionality of cyber ranges and security testbeds.

1. In a scenario of critical infrastructure protection (Morris et al., 2011), researchers employed CSET (cyber security evaluation tool) [1]. CSET is a qualitative evaluation method in which multiple security standards are integrated like NIST, Transportation Security Administration (TSA), North American Electric Reliability Corporation (NERC), U.S. Department of Defense (DoD), and others. When a security level is selected for evaluation, the CSET present a questionnaire based upon the above standards and measure the security level based upon the answers from security experts.
2. In another scenario of SCADA testbed and security device (Jung et al., 2008), researchers developed there own evaluation matrices for evaluating the security of SCADA testbed. Their evaluation matrices consist of.

[1] https://ics-cert.us-cert.gov/Assessments.

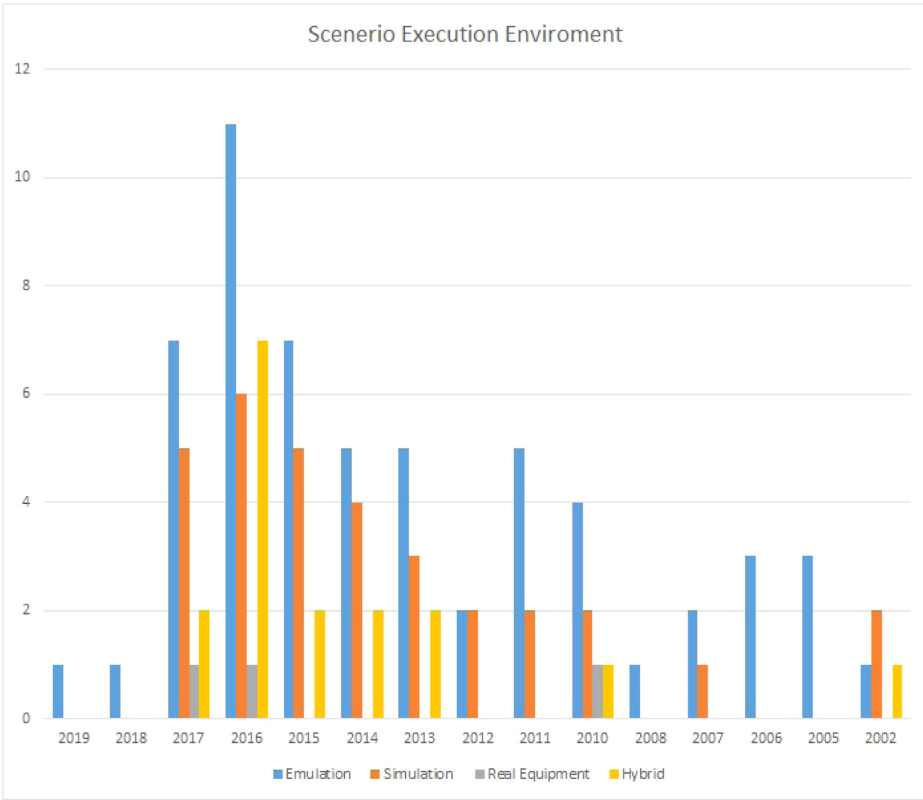


Fig. 8. Classification of cyber-ranges and security testbeds based upon the scenario execution environment.

- The level of exposure of SCADA systems.
- Ports of which the access is available (such as TCP/IP, MODBUS).
- Access to websites connected with the SCADA system.
- Vulnerabilities of websites connected with the SCADA system.
- Vulnerabilities of Remote Terminal Unit(RTU) and Master Terminal Unit(MTU).
- The status of common firewalls.

3. Researchers in a testbed of wearable IoT devices (Siboni et al., 2016) employed a scenario based evaluation in which they determined what type of scenario capabilities their testbed supports. Scenario based evaluation takes into account the following capabilities in a scenario.
   - Scanning (e.g., IP and port scanning)
   - Fingerprinting
   - Process enumeration
   - Data leakage
   - Side-channel attacks
   - Data collection
   - Management access
   - Breaking encrypted traffic
   - Spoofing/masquerade attack
   - Communication delay attacks
   - Communication tampering
   - List known vulnerabilities
   - Vulnerability scan
4. In a cloud-based testbed for simulation of cyber attacks (Kouril et al., 2014), researcher used two experiments to evaluate the testbed in a qualitative manner, in which they used slowHttptest to validate the effectiveness of a security module on a web server. In the first experiment, a web server is equipped with a security module to mitigate a cyber attack, while in the second experiment a web server is targeted without the security module. During the first experiment the server became unavailable after 14 seconds of the attack. However, as soon as the duration of the connection reached the timeout set by the mitigation module, the connection was terminated and the server returned HTTP code 400. In the second experiment, the server became unavailable after 14 seconds and remained in this state for next 586 seconds until the attack ended, as no mitigation module was activated.

4.4. Tools

In this section we identify and classify hardware and software tools utilized within contemporary cyber ranges and security test beds. Details of the tools with respect to year and domain of application as indicated in Section 3.5 will be presented.

4.4.1. Emulation tools

Table 3
Emulation tools used in cyber ranges and security test beds.

ID	Tool Name	Year	Paper	Domain
1	LAAS Cloud infrastructure	2014	Kouril et al. (2014)	Cloud
2	Openstack	2017	Edgar and Manz (2017)	Cloud
3	EMULAB	2012	Genge et al. (2012)	Critical Infrastructure
4	Unity Pro-XL v7.0 suite	2015	Miciolino et al. (2015)	Critical Infrastructure
5	EMULAB	2014	Siaterlis and Genge (2014)	Critical Infrastructure
6	Virtual Box	2013	Stites et al. (2013)	Critical Infrastructure
7	NetEm	2017	Xypolytou et al. (2017)	Critical Infrastructure
8	User-Mode Linux (UML)	2006	Hu et al. (2006)	Hybrid Network and Application
9	Vmware Vsphere	2017	Caliskan et al. (2017)	Hybrid Network and Application
10	Emulab	2015	Soupionis and Benoist (2015)	Hybrid Network and Application
11	KVM	2016	Pham et al. (2016)	Hybrid Network and Application
12	XEN Worlds	2010	Gephart and Kuperman (2010)	Hybrid Network and Application
13	CITRIX XEN	2019	Chandra and Mishra (2019)	Hybrid Network and Application
14	Virtual Box	2015	Sommestad (2015)	Hybrid Network and Application
15	Vmware	2005	Hoffman et al. (2005)	Hybrid Network and Application
16	Vmware	2011	Doupé et al. (2011)	Hybrid Network and Application
17	OPENNEBULA	2015	Čeleda et al. (2015)	Hybrid Network and Application
18	OPENNEBULA	2015	Vykopal et al. (2017a)	Hybrid Network and Application
19	Qemu	2012	Willems and Meinel (2012)	Hybrid Network and Application
20	KVM	2012	Willems and Meinel (2012)	Hybrid Network and Application
21	XEN	2010	Childers et al. (2010)	Hybrid Network and Application
22	OPEN VZ	2010	Childers et al. (2010)	Hybrid Network and Application
23	Qemu	2011	Willems and Meinel (2011)	Hybrid Network and Application
24	KVM	2011	Willems and Meinel (2011)	Hybrid Network and Application
25	Mininet	2015	Al-Ayyoub et al. (2015)	Hybrid Network and Application
26	Virtualbox	2014	Vigna et al. (2014)	Hybrid Network and Application
27	Virtual Machine	2010	Louthan et al. (2010)	Hybrid Network and Application
28	Cyber Smart	2009	Marshall (2009)	Hybrid Network and Application
29	Vmware	2007	Border (2007)	Hybrid Network and Application
30	Vmware ESXI	2013	Rursch and Jacobson (2013b)	Hybrid Network and Application
31	Vmware	2005	Richmond (2005)	Hybrid Network and Application
32	Vmware ESXI	2013	Rursch and Jacobson (2013a)	Hybrid Network and Application
33	OpenFlow switches (OVS)	2016	Flauzac et al. (2016)	IOT
34	Vmware Vsphere	2016	Flauzac et al. (2016)	IOT
35	Qemu system	2016	Flauzac et al. (2016)	IOT
36	XEN with the xapi toolstack	2017	Herold et al. (2017)	Network
37	KVM	2016	Yasuda et al. (2016)	Network
38	Vmware ESXI	2016	Yasuda et al. (2016)	Network
39	OPENNEBULA	2014	Jirsik et al. (2014)	Network
40	Xen-VM	2016	Chadha et al. (2016)	Network
41	Fluxbox desktop through Guacamole	2016	Chadha et al. (2016)	Network
42	Emulab	2006	Benzel et al. (2006)	Network
43	XEN	2014	Moraes et al. (2014)	Network
44	XORP Router	2009	Li et al. (2009)	Network
45	Open VZ	2009	Li et al. (2009)	Network
46	Future internet test bed FITS	2016	Alvarenga and Duarte (2016)	Network
47	Emulab	2018	Tsai and Yang (2018)	Network
48	Emulab	2011	Siaterlis et al. (2011)	Network
49	Proxmox VE	2016	Pfrang et al. (2016)	SCADA
50	Mininet	2017	Antonioli et al. (2017)	SCADA
51	CORE emulator	2013	Almalawi et al. (2013)	SCADA
52	Vmware Esxi	2012	Urias et al. (2012)	SCADA
53	Vyatta software routers	2012	Urias et al. (2012)	SCADA

4.4.2. Simulation tools

Table 4
Simulation tools used in cyber ranges and security test beds.

ID	Tool name	Year	Paper	Domain
1	QualNet	2015	Bergin (2015)	Autonmous Systems
2	Simulink	2015	Koutsandria et al. (2015)	Critical Infrastructure
3	Digsilent Powerfactory	2013	Hahn et al. (2013)	Critical Infrastructure
4	Real-time digital simulator	2013	Hahn et al. (2013)	Critical Infrastructure
5	Simulink	2014	Siaterlis and Genge (2014)	Critical Infrastructure
6	SCADASim	2013	Stites et al. (2013)	Critical Infrastructure
7	ModelNet	2002	White et al. (2002)	Network
8	Network Simulator	2002	White et al. (2002)	Network
9	Arena	2007	Kuhl et al. (2007)	Network
10	Opnet	2016	Chadha et al. (2016)	Network
11	QualNet	2016	Chadha et al. (2016)	Network
12	ns2	2016	Chadha et al. (2016)	Network
13	ns3	2016	Chadha et al. (2016)	Network
14	PRIME (Parallel Real-time Immersive network Modeling Environment)	2009	Li et al. (2009)	Network
15	iSSFNet	2005	Liljenstam et al. (2005)	Network
16	Opnet	2011	Mallouhi et al. (2011)	SCADA
17	PowerWorld	2011	Mallouhi et al. (2011)	SCADA
18	Matlab	2014	Farooqui et al. (2014)	SCADA
19	Simulink	2014	Farooqui et al. (2014)	SCADA
20	Truetime	2014	Farooqui et al. (2014)	SCADA
21	CIROS 6.0	2016	Pfrang et al. (2016)	SCADA
22	Digital I/O, Analog I/O	2008	Jung et al. (2008)	SCADA
23	MODBUS IO	2013	Almalawi et al. (2013)	SCADA
24	Opnet	2012	Urias et al. (2012)	SCADA
25	Matlab	2013	Gao et al. (2013)	SCADA
26	Smulink	2013	Gao et al. (2013)	SCADA
27	Simulink	2016	Alves et al. (2016)	SCADA
28	Matlab	2016	Alves et al. (2016)	SCADA
29	SimHydraulics	2016	Alves et al. (2016)	SCADA
30	OpenPlc	2016	Alves et al. (2016)	SCADA

4.4.3. Hardware

Table 5
Hardware devices used in cyber ranges and security test beds.

ID	Tool name	Year	Paper	Domain
1	Allen Bradley RSLogix 5000	2011	Morris et al. (2011)	Critical Infrastructure
2	L35E PLCs.	2011	Morris et al. (2011)	Critical Infrastructure
3	Factory Talk View 5.0 HMI screens	2011	Morris et al. (2011)	Critical Infrastructure
4	Phasor measurement units	2011	Morris et al. (2011)	Critical Infrastructure
5	Phasor data concentrator	2011	Morris et al. (2011)	Critical Infrastructure

(*continued on next page*)

Table 5 (*continued*)

ID	Tool name	Year	Paper	Domain
6	Synchrophasor vector processor	2011	Morris et al. (2011)	Critical Infrastructure
7	protection relays controllers	2011	Morris et al. (2011)	Critical Infrastructure
8	substation GPS clock	2011	Morris et al. (2011)	Critical Infrastructure
9	Omicron relay test	2011	Morris et al. (2011)	Critical Infrastructure
10	calibration device	2011	Morris et al. (2011)	Critical Infrastructure
11	Real Time Digital Simulator (RTDS)	2011	Morris et al. (2011)	Critical Infrastructure
12	amplifiers	2011	Morris et al. (2011)	Critical Infrastructure
13	PMUs	2011	Morris et al. (2011)	Critical Infrastructure
14	Cisco 5510	2011	Morris et al. (2011)	Critical Infrastructure
15	MU Dynamics MU-4000 Analyzer	2011	Morris et al. (2011)	Critical Infrastructure
16	IEEE C37.118,	2011	Morris et al. (2011)	Critical Infrastructure
17	PLC	2015	Koutsandria et al. (2015)	Critical Infrastructure
18	Intelligebt End Device	2013	Hahn et al. (2013)	Critical Infrastructure
19	PLC	2013	Hahn et al. (2013)	Critical Infrastructure
20	PLC	2015	Gao et al. (2015)	Critical Infrastructure
21	Remote Terminal Unit	2015	Gao et al. (2015)	Critical Infrastructure
22	Smart Transmitter	2015	Gao et al. (2015)	Critical Infrastructure
23	Cisco 6503	2014	Siaterlis and Genge (2014)	Critical Infrastructure
24	IEC 60870-5-104	2016	Gunathilaka et al. (2016)	Critical Infrastructure
25	IEC 61,850 MMS	2016	Gunathilaka et al. (2016)	Critical Infrastructure
26	HP ProLiant DL380 G7	2015	Ernits et al. (2015)	Hybrid Network and Application
27	Google Glass	2016	Siboni et al. (2016)	IOT
28	Sony Smart watches	2016	Siboni et al. (2016)	IOT
29	Energy Management System	2018	Lee et al. (2017)	IOT
30	Remote Terminal Unit	2018	Lee et al. (2017)	IOT
31	Smart surveillance camera	2017	Furfaro et al. (2017)	IOT
32	Android Smart Phone	2017	Furfaro et al. (2017)	IOT
33	Cisco routers	2017	Herold et al. (2017)	Network
34	Cisco routers	2010	Chow et al. (2010)	Network
35	Siemens Devices	2010	Fovino et al. (2010)	SCADA
36	Emerson Devices	2010	Fovino et al. (2010)	SCADA
37	ABB Devices	2010	Fovino et al. (2010)	SCADA
38	Filed Dev	2010	Fovino et al. (2010)	SCADA

Table 5 (*continued*)

ID	Tool name	Year	Paper	Domain
39	PLC	2017	Domínguez et al. (2017)	SCADA
40	PLC	2016	Pfrang et al. (2016)	SCADA
41	SIEMENS S7-300	2016	Pfrang et al. (2016)	SCADA
42	Cisco ASA	2016	Pfrang et al. (2016)	SCADA
43	RS485 Multiport	2008	Jung et al. (2008)	SCADA
44	Phasor Data Concentrator	2016	Ashok et al. (2016)	SCADA
45	Phasor Measurement Units	2016	Ashok et al. (2016)	SCADA
46	SEL 421	2016	Ashok et al. (2016)	SCADA
47	Multifunction protection relays (7SJ610, 7SJ82)	2016	Ashok et al. (2016)	SCADA
48	SICAM PAS	2016	Ashok et al. (2016)	SCADA
49	Power TG	2016	Ashok et al. (2016)	SCADA
50	PLC	2013	Almalawi et al. (2013)	SCADA
51	PLC	2016	Rubio-Hernan et al. (2016)	SCADA
52	Raspbery PI	2016	Rubio-Hernan et al. (2016)	SCADA
53	Cisco 2600 router	2012	Urias et al. (2012)	SCADA
54	Juniper M61	2012	Urias et al. (2012)	SCADA
55	PLC	2013	Gao et al. (2013)	SCADA
56	Remote Teminal Unit	2013	Gao et al. (2013)	SCADA
57	Rasbery PI	2016	Alves et al. (2016)	SCADA

4.4.4. Management tools

Table 6
Management tools used in cyber ranges and security test beds.

1	Energy Management System	2013	Hahn et al. (2013)	Critical Infrastructure
2	Energy Management System	2014	Siaterlis and Genge (2014)	Critical Infrastructure
3	Energy Management System	2016	Gunathilaka et al. (2016)	Critical Infrastructure
4	ISEAGE	2013	Rursch and Jacobson (2013a)	Hybrid Network and Application
5	SIGAR API	2019	Chandra and Mishra (2019)	Hybrid Network and Application
6	3vilSh3llfor backdoor	2011	Doupé et al. (2011)	Hybrid Network and Application
7	vmService	2012	Willems and Meinel (2012)	Hybrid Network and Application
8	vmService	2011	Willems and Meinel (2011)	Hybrid Network and Application
9	HAMIDS	2017	Antonioli et al. (2017)	SCADA
10	Xentop	2014	Moraes et al. (2014)	Network

4.4.5. Monitoring tools

Table 7
Monitoring tools used in cyber ranges and security test beds.

1	Netflow	2014	Kouril et al. (2014)	Cloud
2	IPFIX	2014	Kouril et al. (2014)	Cloud
3	IPFIX	2017	Edgar and Manz (2017)	Cloud
4	OSISoft PI Historian	2011	Morris et al. (2011)	Critical Infrastructure
5	Zabbix	2012	Genge et al. (2012)	Critical Infrastructure
6	Libpcap	2015	Koutsandria et al. (2015)	Critical Infrastructure
7	OSIsoft	2015	Koutsandria et al. (2015)	Critical Infrastructure
8	Wireshark	2015	Miciolino et al. (2015)	Critical Infrastructure
9	Energy Management System	2013	Hahn et al. (2013)	Critical Infrastructure
10	Open V Switch	2015	Gao et al. (2015)	Critical Infrastructure
11	Energy Management System	2014	Siaterlis and Genge (2014)	Critical Infrastructure
12	Energy Management System	2016	Gunathilaka et al. (2016)	Critical Infrastructure
13	Tcpdump	2017	Xypolytou et al. (2017)	Critical Infrastructure
14	Security Onion Linux	2017	Caliskan et al. (2017)	Hybrid Network and Application
15	OSSEC	2017	Caliskan et al. (2017)	Hybrid Network and Application
16	Tcpdump	2016	Pham et al. (2016)	Hybrid Network and Application
17	Wireshark	2016	Pham et al. (2016)	Hybrid Network and Application
18	SIGAR API	2019	Chandra and Mishra (2019)	Hybrid Network and Application
19	3vilSh3llfor backdoor	2011	Doupé et al. (2011)	Hybrid Network and Application
20	Nagios	2015	Čeleda et al. (2015)	Hybrid Network and Application
21	Nagios	2015	Vykopal et al. (2017a)	Hybrid Network and Application
22	vmService	2012	Willems and Meinel (2012)	Hybrid Network and Application
23	vmService	2011	Willems and Meinel (2011)	Hybrid Network and Application
24	Catbird	2015	Al-Ayyoub et al. (2015)	Hybrid Network and Application
25	ISEAGE	2013	Rursch and Jacobson (2013b)	Hybrid Network and Application
26	Snort	2005	Richmond (2005)	Hybrid Network and Application
27	SyscallAnomaly	2005	Richmond (2005)	Hybrid Network and Application
28	ISEAGE	2013	Rursch and Jacobson (2013a)	Hybrid Network and Application
29	Wireshark	2016	Siboni et al. (2016)	IOT
30	ADB	2016	Siboni et al. (2016)	IOT
31	Open V Switch	2016	Flauzac et al. (2016)	IOT
32	Opendaylight controller	2016	Flauzac et al. (2016)	IOT
33	Tcpdump	2017	Herold et al. (2017)	Network
34	Tcpdump	2002	White et al. (2002)	Network
35	Traceroute	2002	White et al. (2002)	Network
36	FRONTIER	2010	Chow et al. (2010)	Network
37	SHINE	2010	Chow et al. (2010)	Network
38	Netflow	2014	Jirsik et al. (2014)	Network
39	IPFIX	2014	Jirsik et al. (2014)	Network
40	Emulab	2006	Benzel et al. (2006)	Network
41	Network Flight Recorder (NFR) Sentivist	2006	Benzel et al. (2006)	Network
42	FloodWatch	2006	Benzel et al. (2006)	Network
43	OPENFLOW	2014	Moraes et al. (2014)	Network
44	Xentop	2014	Moraes et al. (2014)	Network
45	Tcpdump	2009	Li et al. (2009)	Network
46	Testbed@TWISC Monitor	2018	Tsai and Yang (2018)	Network
47	NAGIOS	2005	Alfieri et al. (2005)	Network
48	Zabbix	2011	Siaterlis et al. (2011)	Network
49	NetDecoder	2017	Domínguez et al. (2017)	SCADA
50	CanAnalyzer	2017	Domínguez et al. (2017)	SCADA
51	Open V Switch	2016	Pfrang et al. (2016)	SCADA
52	Pf sense	2016	Pfrang et al. (2016)	SCADA
53	SNORT	2016	Pfrang et al. (2016)	SCADA
54	OSSEC	2016	Pfrang et al. (2016)	SCADA
55	HAMIDS	2017	Antonioli et al. (2017)	SCADA
56	Wireshark	2012	Urias et al. (2012)	SCADA
57	Tcpdump	2012	Urias et al. (2012)	SCADA

4.4.6. Traffic generation tools

Table 8
Traffic generation tools used in cyber ranges and security test beds.

ID	Tool Name	Year	Paper	Domain
1	Low Orbit Ion Canon	2014	Kouril et al. (2014)	Cloud
2	Modbus	2011	Morris et al. (2011)	Critical Infrastructure
3	Events (GOOSE)	2011	Morris et al. (2011)	Critical Infrastructure
4	Generic Object Oriented Substation	2011	Morris et al. (2011)	Critical Infrastructure
5	DNP3	2011	Morris et al. (2011)	Critical Infrastructure
6	EtherNet/IP	2011	Morris et al. (2011)	Critical Infrastructure
7	ISAGE	2013	Hahn et al. (2013)	Critical Infrastructure
8	Open flow	2015	Gao et al. (2015)	Critical Infrastructure
9	Modbus	2014	Siaterlis and Genge (2014)	Critical Infrastructure
10	DNP3	2014	Siaterlis and Genge (2014)	Critical Infrastructure
11	Modbus	2016	Gunathilaka et al. (2016)	Critical Infrastructure
12	ISEAGE	2013	Rursch and Jacobson (2013b)	Hybrid Network and Application
13	Traffic Collector/Replayer	2013	Rursch and Jacobson (2013a)	Hybrid Network and Application
14	Printer	2016	Siboni et al. (2016)	IOT
15	SSH	2016	Siboni et al. (2016)	IOT
16	SNMP	2016	Siboni et al. (2016)	IOT
17	MicroWorks	2018	Lee et al. (2017)	IOT
18	SSH	2017	Herold et al. (2017)	Network
19	SNMP	2017	Herold et al. (2017)	Network
20	Policy Enabled Agent	2010	Chow et al. (2010)	Network
21	Low Orbit Ion Canon	2014	Jirsik et al. (2014)	Network
22	Emulab	2006	Benzel et al. (2006)	Network
23	hydra	2018	Tsai and Yang (2018)	Network
24	tfn2k	2018	Tsai and Yang (2018)	Network
25	Modbus Rsim	2011	Mallouhi et al. (2011)	SCADA
26	MODBUS	2008	Jung et al. (2008)	SCADA
27	DNP3	2016	Rubio-Hernan et al. (2016)	SCADA
28	Modbus	2016	Rubio-Hernan et al. (2016)	SCADA
29	Virtual Control System Environment	2012	Urias et al. (2012)	SCADA

4.4.7. User behavior generation tools

Table 9
Use behavior generation tools used in cyber ranges and security test beds.

ID	Tool name	Year	Paper	Domain
1	AMICI	2015	Soupionis and Benoist (2015)	Hybrid Network and Application
2	ConsoleUser	2015	Sommestad (2015)	Hybrid Network and Application
3	AutoIT	2016	Chadha et al. (2016)	Network
4	Netkit	2017	Braidley (2016)	Social Engineering

4.4.8. Scoring tools and mechanisms

Table 10
Scoring mechanisms and tools used in cyber ranges and security test beds.

ID	Tool name	Year	Paper	Domain
1	Task Based	2013	Stites et al. (2013)	Critical Infrastructure
2	Score Bot	2005	Hoffman et al. (2005)	Hybrid Network and Application
3	Jeopardy board	2014	Silva et al. (2014)	Hybrid Network and Application
4	ICTF score board, Flags	2011	Doupé et al. (2011)	Hybrid Network and Application
5	ICTF score board, Flags	2010	Childers et al. (2010)	Hybrid Network and Application
6	Score Bot	2014	Vigna et al. (2014)	Hybrid Network and Application
7	Flags	2006	Snyder (2006)	Hybrid Network and Application

4.4.9. Scenario definition

Table 11
Scenario definition mechanisms in cyber ranges and security test beds.

ID	Tool name	Year	Paper	Domain
1	XML	2015	Bergin (2015)	Autonmous Systems
2	JSON	2015	Bergin (2015)	Autonmous Systems
3	XML	2012	Genge et al. (2012)	Critical Infrastructure
4	YAML	2016	Pham et al. (2016)	Hybrid Network and Application
5	XML	2013	Reed et al. (2013)	Hybrid Network and Application
6	XML	2012	Willems and Meinel (2012)	Hybrid Network and Application
7	XML	2011	Willems and Meinel (2011)	Hybrid Network and Application
8	XML	2017	Herold et al. (2017)	Network
9	XML	2010	Chow et al. (2010)	Network
10	Integration Markup Language (IML)	2010	Chow et al. (2010)	Network
11	Policy Editor Tools	2010	Chow et al. (2010)	Network
12	Policy negotiation tool	2010	Chow et al. (2010)	Network
13	XML	2007	Kuhl et al. (2007)	Network
14	XML	2016	Chadha et al. (2016)	Network
15	XML	2002	Rossey et al. (2002)	Network
16	JSON	2016	Alvarenga and Duarte (2016)	Network
17	Offense and Defense Toolbox	2018	Tsai and Yang (2018)	Network

4.4.10. Security testing tools

Table 12
Security Testing tools used in cyber ranges and security test beds.

ID	Tool name	Year	Paper	Domain
1	Juas Messages	2015	Bergin (2015)	Autonmous Systems
2	Low Orbit Ion Canon	2014	Kouril et al. (2014)	Cloud
3	Ettercap	2011	Morris et al. (2011)	Critical Infrastructure
4	Ettercap	2015	Miciolino et al. (2015)	Critical Infrastructure
5	GunPG1	2006	Hu et al. (2006)	Hybrid Network and Application
6	John-the-Ripper	2006	Hu et al. (2006)	Hybrid Network and Application
7	Bit torrent	2012	Barcellos et al. (2012)	Hybrid Network and Application
8	Kali Linux	2017	Caliskan et al. (2017)	Hybrid Network and Application
9	PathTest	2015	Soupionis and Benoist (2015)	Hybrid Network and Application
10	Iperf	2015	Soupionis and Benoist (2015)	Hybrid Network and Application
11	FTK Imager	2011	Glumich and Kropa (2011)	Hybrid Network and Application
12	Zora	2011	Glumich and Kropa (2011)	Hybrid Network and Application
13	netcat	2011	Glumich and Kropa (2011)	Hybrid Network and Application
14	cron	2011	Glumich and Kropa (2011)	Hybrid Network and Application
15	hex editor	2011	Glumich and Kropa (2011)	Hybrid Network and Application
16	offensive computing.net	2011	Glumich and Kropa (2011)	Hybrid Network and Application
17	Helix Forensics Live Linux CD	2011	Glumich and Kropa (2011)	Hybrid Network and Application
18	WinHex	2011	Glumich and Kropa (2011)	Hybrid Network and Application
19	md5sum	2011	Glumich and Kropa (2011)	Hybrid Network and Application
20	FTK Imager	2011	Glumich and Kropa (2011)	Hybrid Network and Application
21	vxheaven.org	2019	Chandra and Mishra (2019)	Hybrid Network and Application
22	SlowHTTPTest	2019	Chandra and Mishra (2019)	Hybrid Network and Application
23	LOIC	2019	Chandra and Mishra (2019)	Hybrid Network and Application
24	John the ripper	2006	Snyder (2006)	Hybrid Network and Application
25	SVED	2015	Sommestad (2015)	Hybrid Network and Application
26	ENCASE Enterprise	2014	Silva et al. (2014)	Hybrid Network and Application
27	WireShark	2014	Silva et al. (2014)	Hybrid Network and Application
28	IDA Pro-	2014	Silva et al. (2014)	Hybrid Network and Application
29	Volatility	2014	Silva et al. (2014)	Hybrid Network and Application
30	Hex Workshop	2014	Silva et al. (2014)	Hybrid Network and Application
31	PDF Dissector	2014	Silva et al. (2014)	Hybrid Network and Application
32	One-class support vector machine (OCSVM)	2018	Lee et al. (2017)	IOT
33	Low Orbit Ion Canon	2014	Jirsik et al. (2014)	Network
34	Crimeware toolkits	2016	Chadha et al. (2016)	Network
35	Metasploit	2016	Chadha et al. (2016)	Network
36	Nmap	2016	Chadha et al. (2016)	Network
37	Symantec ManHunt	2006	Benzel et al. (2006)	Network
38	Nmap	2011	Mallouhi et al. (2011)	SCADA
39	Nmap	2008	Jung et al. (2008)	SCADA
40	Nessus	2008	Jung et al. (2008)	SCADA
41	Wireshark	2008	Jung et al. (2008)	SCADA
42	WinHTTrack	2008	Jung et al. (2008)	SCADA
43	Netcraft	2008	Jung et al. (2008)	SCADA
44	Kartoo	2008	Jung et al. (2008)	SCADA

4.5. Future research trends and directions

In order to analyze the future research trends and directions, we looked closely to all papers since 2016 and we briefly present their future work in this section, and discuss and summarize them in Section 5.2.

1. Design of cyber warfare testbed (Chandra and Mishra, 2019). Tow main future direction were proposed, the first is using *OS container*, as they are lightweight and support a wide range of OSs. The second direction is focusing on *simulating human behavior* using agent based simulation toolkit.
2. Testbed@ TWISC: A network security experiment platform (Tsai and Yang, 2018).
   The authors of this work foresee threefold future development. The first is using virtualization and *SDN (SW Defined Networks)* due to its high programmability capability. The second is *federation*, which is required to support large scale exercises. Particularly they planned to use VPLS(Virtual Private LAN Service). Finally, they planned to work on what they call *Software Defined Security* that aims at tackling the additional attack vector on virtualization.
3. Achieving reproducible network environments with INSALATA (Herold et al., 2017).
   Few Future directions were proposed by the authors. They mainly focus on extending the current capability, e.g., (1)

better *monitoring and event collection*, and (2) more *realistic* network environment reproducibility. Furthermore, *efficient deployment* is another goal for the future.

4. Capability Detection and Evaluation Matrics for Cyber Security lab Exercises (Caliskan et al., 2017).
   The authors planned to extend the experiment setting and invite different students to take part for the sake of *cross validation*. *Stability* to support large scale exercises were also planned.
5. Control frameworks in network emulation testbeds: A survey (Tsai et al., 2017).
   Two main directions can be identified in this paper, which are (1) supporting more *realistic* scenarios, and (2) *visualization and analytics*.
6. Cybersecurity training in control systems using real equipment (Domínguez et al., 2017).
   Further work of this work includes the *educational* evaluation of the laboratory.
7. Design and implementation of cybersecurity testbed for industrial IoT systems (Lee et al., 2017).
   The main future direction of this work is to use the testbed to *test and evaluate new security technologies* to various critical infrastructure systems, e.g., next generation intelligent power control system.
8. Developing a capability to classify technical skill levels within a Cyber Range (Labuschagne and Grobler, 2017).
   One idea that were discussed is the development of an *intent capability* whereby the intent of the user can be predicted.
9. Experiment as a service (Edgar and Rice, 2017).
   The main future direction discussed in this paper is the development of *sharable and validated models (scenarios)* of *realistic* environments to support *federation*.
10. Extending Our Cyber-Range CYRAN with Social Engineering Capabilities (Braidley, 2016).
    The social media profiles didn't use any real employee photo due to privacy concerns this can be improved in future using alternate images of employees. The content posted on social media is only text based in future other media formats like videos and images can be integrated for better representation of real social media.
11. Gamifying ICS security training and research: Design, implementation, and results of S3 (Antonioli et al., 2017).
    Future work discussed was to use the method applied in the paper as a foundation to enable others to run similar security *educational* experiments. This implies also the possibility to *share* the experiment models among different parties.
12. Improving and Measuring Learning Effectiveness at Cyber Defense Exercises (Maennel et al., 2017).
    Future work was planned to develop a *learning* metrics and trends *benchmark*, which will provide a baseline to evaluate *learning* improvement in cybersecurity exercises.
13. KYPO Cyber Range: Design and Use Cases (Vykopal et al., 2017a).
    The future direction for KYPO is to use the current developed infrastructure to *test and experiment* with recent complex cyber attacks in order to evaluate and study *detection and mitigation* control against cyber threats to the critical infrastructure.
14. Modeling and simulation architecture for training in cyber defence education (Subaşu et al., 2017).
    There are several courses of future development arising from the ideas presented above. A further direction is to make a comparison between our proposed architecture and existing military or commercial training solutions.
15. The FUSE testbed: Establishing a microgrid for smart grid security experiments (Xypolytou et al., 2017).
    Similar to the previous future work, FUSE testbed was planned to be used to study methods and techniques to *detect* anomalies against critical infrastructure. *Security, availability and reliability* will be evaluated in the testbed to enhance *situational-awareness*.
16. Advanced security testbed framework for wearable IoT devices (Siboni et al., 2016).
    Afte completing the development of the testbed, the main future work discussed for this paper is to use the testbed in *testing* smart city IoT devices. The development of a lightweight anitmalware is also planned.
17. Alfons: A Mimetic Network Environment Construction System (Yasuda et al., 2016).
    Optimizing and enhancing efficiency of the system are the main future work planned for the Alfons system.
18. Cybervan: A cyber security virtual assured network testbed (Chadha et al., 2016).
    In the following are the future work directions discussed for Cybervan: (1) *Scalability*, (2) *portability* to various virtualization and container technologies, (3) supporting more *realistic* scenarios (4) introducing *cognitive factors in simulation* of user/attacker behaviors, (5) enhancing *testing and validation* procedures of new technologies by developing an automated state space *exploration* mechanisms, and finally (6) enhancing *automation* capabilities in order to increase resource and research productivity.
19. CyRIS: A cyber range instantiation system for facilitating security training (Pham et al., 2016).
    Two main issues were planned for future work of CyRIS system, the first is *scalability* and the second is *automation* of network configuration capabilities.
20. Design and architecture of an industrial IT security lab (Pfrang et al., 2016).
    The two main directions planned for this work are to (1) apply the infrastructure for *education and awareness* training, and (2) perform advanced security *monitoring* by including remote production sites.
21. Developing a distributed software defined networking testbed for IoT (Flauzac et al., 2016).
    The main future work discussed in this paper is to expand *simulation* capabilities to include IPv6 and evaluate performance evaluation.
22. PowerCyber: A remotely accessible testbed for Cyber Physical security of the Smart Grid (Ashok et al., 2016).
    The activities planned as future work include (1) developing *library of models* and datasets, (2) increasing the *user community*, and (3) developing advanced *realistic* use cases.
23. RIO: A denial of service experimentation platform in a Future Internet Testbed (Alvarenga and Duarte, 2016).
    The main future work is to work on *efficiency* by studying the impact of each step on the experimentation overall time. Furthermore, the authors were planning to investigate possible *automation* of the platform.
24. Softgrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions (Gunathilaka et al., 2016).
    Future directions discussed for this work are multifold. (1) Supporting distributed setups and *emulation*, (2) *testing and evaluation* of different security solution and attack vectors, and (3) supporting other SCADA *protocols*, are the main directions discussed.
25. Virtualization of industrial control system testbeds for cybersecurity (Alves et al., 2016).
    The future work presented focused on improving the studied *emulated* and virtual testbeds. Regarding virtualization, it

was proposed to compare the system characteristics of both the virtual and the physical controller. Finally, *scalability* is the last issue the authors were planning to investigate.

5. Synthesis

The analysis of data related to tool yielded some interesting results. In term of scenario definition, XML is predominately used as indicated in Table 11. XML provide a self descriptive way for designing and storing a scenario definition. The developed scenario definition can then be used in scenario simulation and emulation. It is used in autonomous systems, critical infrastructure, network and hybrid network and application scenarios. For monitoring, Tcpdump, IPFIX, and Wireshark were the most widely used tools in cyber ranges and security test-beds. They are used for monitoring traffic in cloud, network, critical infrastructure, and SCADA domains. Details of all the monitoring tools used in cyber ranges and security testbeds are presented in Table 7. Multitude of different hardware devices were used in construction of different cyber ranges and security testbeds. However CISCO based devices are most widely used from the domain of critical infrastructure to networks and SCADA. Different PLC devices were also used in the construction of SCADA and critical infrastructure testbeds. Details of hardware devices used in construction of different cyber ranges and security testbeds are presented in Table 5. For emulation, Vmware based tools and Emulab were mostly used for critical infrastructure, hybrid network and application and networks domain. Vmware was also used in IoT and SCADA domains as well. Details of emulation tools used in cyber ranges and security testbeds is presented in Table 3. In term of scenario simulation, Quanet, Simulink, Network Simulator and Matlab were widely used as indicated in Table 4. Qualnet was used for both autonomous systems and critical infrastructure. Simulink was used for Critical infrastructure and SCADA. While Network Simulator and Matlab were used specifically for networks and SCADA respectively.

Different tools were used for traffic generation purposes in different domains. Modbus traffic is mostly used for SCADA and critical infrastructure while Low Orbit Ion Canon is used for TCP/UDP traffic generation. Details of traffic generation tools are presented in Table 8. Different tools were used for security testing, user behavior generation, and scoring purposes in different domains, details of which are presented in Tables 9, 10 and 12, respectively.

In term of the scenario types static and dynamic, a significant shift towards dynamic scenarios is witnessed in 2011 as indicated in Fig. 5. We believe that this shift happened due to identification of famous Stuxnet (Langner, 2011) worm in 2010 which created a lot of tidal waves in the cyber security research community. This observation is further backed by the data presented in Fig. 6, in which the rise of critical infrastructure and SCADA related testbeds can be observed. With the rise of cyber threats from nation state actors, investment in cyber security research increased with the aim to develop cyber resilience. This included development of new cyber security tools and methods as well as educating a workforce to handle cyber security crisis. This shift of sudden rise of education related scenarios in 2011 can be observed in Fig. 4. In the future, we believe that with abundant availability of computational resources, more and more tasks within the cyber ranges and security testbeds will get automated. From scenario creation to scenario execution and analysis, human role will become limited. This trend has started from 2014 with the appearance of autonomous teams in cyber ranges and security tetsbeds as indicated in Fig. 7.

5.1. Architecture and capabilities

In Section 4.2 and 4.1, we presented a new taxonomy that included general capabilities of cyber ranges and security testbeds. We also looked into the details of the architectural model of each cyber range reviewed in this paper. Analyzing the various architectural models of different systems, we can see that the same components are named differently in different systems. For example, scenario execution element, orchestration module, controlling component could indicate the same functional component. We highlighted in Section 4.2 the main concepts and used unified terminology. In this section, we aim at developing a unified functional architecture for cyber ranges based on the knowledge we gained from analyzing the architectures of cyber ranges and security testbeds, aforementioned.

Fig. 9 shows a unified functional architecture that is developed from studying the literature. The architecture is divided into main components and within each component we define a set of subcomponents.

- Portal.
  Portal provide the interface for communication between acyber range, or a security testbed, and multiple users. The users can be cyber range admins, white team users to create and edit cyber security scenarios and other clients who use the cyber ranges for various tests and experiments. The cyber range and security testbed admin user performs over management activities related to the cyber range or the testbed, which includes resource management and access management to other users like instructor, testers, trainee, or a white team member scenario creator. The scenario creator creates scenarios which can be deployed for cyber security exercise and experiments. The clients can use cyber range and testbed resources for testing and experimentation according to their requirements.
- Management.
  In management functions, resources and roles are managed. Resources includes the memory, processing and storage capabilities, while roles management include the assignment of duties for the cyber security exercises and experiment. A cyber range and a security testbed management is related to overall range management. It deals with assigning roles to exercise and experiment managers, as well as necessary computational resources to conduct the exercise and run the experiment. Multiple exercises and experiments can be conducted at a same time on cyber ranges and security testbeds. Exercise management deals with the segregation of roles and resource of an exercise or an experiment participant. In an exercise or experiment, multiple scenarios can be conducted, scenario management deals with the management of multiple exercises or experiment scenarios on the environment. Extensive collection of log information and analysis is performed from the cyber range and security testbed infrastructure for managing the cyber range and security testbed infrastructure in optimal manner.
- Training and education.
  A training and education module provides tutoring system for cyber range and security testbed. The tutoring system consist of cyber security concepts and their practical exercises for cyber security education purposes. The training outcome is evaluated using a scoring mechanism. Multiple scoring mechanisms can be used like flag-based scoring, task-based scoring and scoring with the help of event log information. After action analysis using training participant feedback and event information is performed to remove inefficiencies in conducting cyber security exercises and improve their qualities.
- Testing.
  As mentioned before, besides training, the second main objective of a cyber range is testing and security assessment. We noticed two main types of tests that can be conducted in a cyber range. The first is to test the security of a system or a product, and the second is to test a new defence or attack method or technique. A testing module aims at defining the test cases,

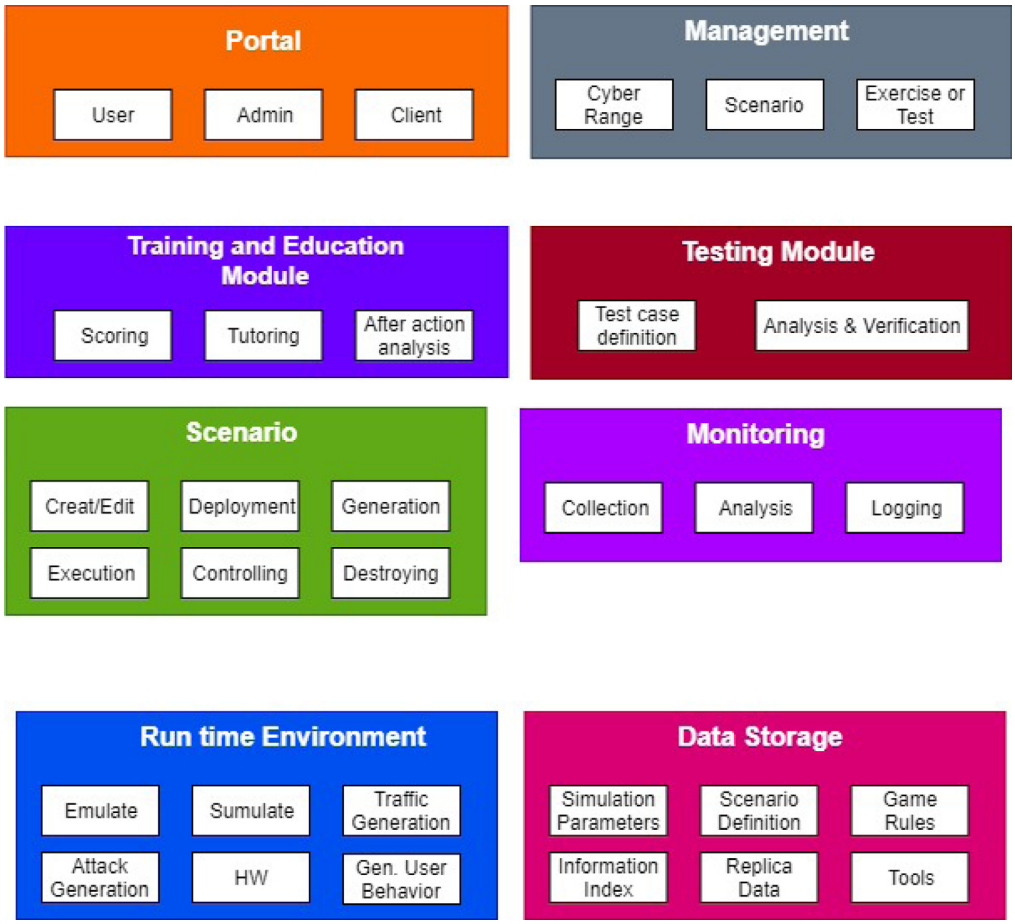


Fig. 9. Cyber range and security testbed functional architecture.

which will be turned into a scenario that will be deployed and executed on the run time environment. After executing the scenario, through the scenario module and the run time environment, the result will be sent back to the testing module to conduct the final analysis and evaluation of the system under test.

- Scenario.
  White team members have access to scenario creator interface. The scenario creator interface is used to create, edit, deploy, generate, execute, control and destroy cyber security scenarios. The scenario creator gives capability to design and deploy new cyber security scenarios and save the scenario configuration in a file. The scenario editor allows to edit predefined scenarios for modification. The scenario deployer reads the saved scenario configuration file and deploys the scenario on emulated, simulated, or hybrid environment. The scenario generator is used to generate new cyber security scenarios using minimum scenario configurations. The scenario executor executes the scenario and performs different actions during different phases of the scenario, like injecting network traffic or initiating a user behavior at different stages to make the scenario more realistic. The scenario controller gives the functionality of modifying the scenario during execution. The scenario destroyer is used to remove obsolete scenario from cyber security exercises to be ready for the next exercise.
- Monitoring.
  Monitoring provides the capability to monitor cyber security exercise and experiment execution. It includes collection of logs from multiple sources and analysis on those logs. The log sources contain different network and operating system interfaces. The logs are mostly in different formats, so their format needs to be unified using some pre-processing techniques. Analysis is then performed on the unified logs to identify different activities being performed by cyber security exercise and experiment participants at different stages of an exercise and an experiment scenario.
- Run time environment.
  The run time environment represents the infrastructure layer that contains physical, virtual, hybrid and cloud platforms, on which the scenario is deployed. Red team attacks the infrastructure and blue team defends the infrastructure. The activities of both teams create events that are used for monitoring and scoring purposes. To make the cyber security exercise and experiment environment more realistic, user behavior and random network traffic is generated.
- Data storage.
  Data storage aims at storing various artifacts needed for executing the training, or testing, scenarios. It includes scenario defi-

nition files, information about the rules that need to be implemented in the scenario, and tools required for the scenario execution. The data storage act as a library for the scenarios with relevant meta/data related to scenario difficulty and complexity. This assists in designing cyber security exercise and experiment according to the skill set of participants.

- Teaming.
  Although not presented in functional architecture of cyber range and security testbeds, teaming roles can't be ignored. A white team is responsible for scenario creation and setting the learning objectives for the scenario. Green team in involved in the monitoring of the scenario. While red and blue teams have access to run time environment for scenario execution. Autonomous teams can be used to emulate or simulate any role of red, blue, white and green teams.

5.1.1. Ideal methods and tools

In this section we will discuss about the ideal methods and tools for cyber range and security testbed development. First, we would argue that there is a lack in standards for cyber ranges and security testbed development. Their is a need to standardize this field, we found cyber range interoperability standards (Damodaran and Smith, 2015) that governs the federation principles for cyber ranges and security testbeds. So, we suggest that any future development of cyber ranges and security should be governed by accepted standards. Secondly, from the results indicated in Fig. 6, it can be augured that hybrid network and application domain over emulation is most popular for cyber range and security testbed development. Therefore, we expect to see more research in the field. We would like to suggest the use of open source or publicly available tools for their development. For hybrid network and application domains, we would suggest the use of cloud infrastructure like Opennebula or Openstack for emulation due to their standardize work environment. With cloud, we would also suggest the use of standard APIs for communication with specific hardware which can't be emulated like PLCs. APIs should also be used for management, monitoring and giving access to teams on the cyber range and security testbed.

5.2. Future research trends and directions

In Section 4.5 we presented the main future plans for all recent work related to cyber range and security testbeds. In this section we compile these plans and provide the main directions for future work. We categorize the future direction into the following categories

1. Efficiency.
   One of the main topics for future work that were discussed by reviewed papers is enhancing the efficiency of exercise lifecycle. To do that, automation is mentioned as a possible technique to make the deployment and execution of exercises more efficient (Beuran et al., 2018; Herold et al., 2017; Pham et al., 2016).
2. Scalability, realism and virtualization.
   To achieve the best result from a training exercise or a testing process, the run time environment should be as close as possible to the real world. While developing small scale and class-room oriented testbeds is feasible and easy to achieve, scaling the testbed to provide as realistic scenarios as possible is a challenging task. Scalability is mentioned by many papers as one of enhancement plans for their cyber ranges (Beuran et al., 2018; Chadha et al., 2016; Pham et al., 2016). Using the new virtualization and emulation techniques, e.g., SDN, is put as an option. Particularly, SDN provides a high degree of programmability that is desired in such settings. Container technology and its support lightweight nature was another scalability enabled future technology. Regarding the issue of realism, one paper proposed to provide the support for a larger number of protocols, e.g., SCADA protocols, in future design of security testbeds (Alves et al., 2016).
3. Federation.
   Another related topic is federation. Federation is also mentioned by couple of papers as one of the main future direction. Activities and issues related to federation include "sharability", potability, support of multiple locations, developing standard way to describe scenarios, defining a library for models and data, and expanding the user community (Edgar and Rice, 2017; Tsai and Yang, 2018).
4. User behavior simulation.
   Current work identified that techniques used today for user behavior simulation has its limitation. To overcome its limitation, advances in user behavior simulation is proposed as one potential future work (Braidley, 2016; Chandra and Mishra, 2019). Examples of the proposed enhancements in the future are to use agent based simulations and introducing cognitive factors.
5. Monitoring.
   Monitoring capabilities are essential for any cyber range or security testbed installations. However, the degree of monitoring and the way it can be used vary from one solution to another. Future work related to monitoring is to use advanced security monitoring and data collection techniques (Herold et al., 2017; Pfrang et al., 2016).
6. Testing and evaluation.
   Few papers proposed, as future work, to extend the current cyber ranges and security testbeds with new testing and evaluation capabilities in order to (1) test new security solutions and technologies (Lee et al., 2017), (2) testing new attack vectors and attack techniques (Vykopal et al., 2017a), (3) testing for some security features that were not considered before in the testbeds like reliability and availability (Gunathilaka et al., 2016), and (4) enhancing the testing techniques (Alvarenga and Duarte, 2016).
7. Education and learning.
   One of the issues that are missing in many current cyber ranges and training testbeds is considering learning and educational aspects (Pfrang et al., 2016). Thus, future work was proposed to support techniques and methods to evaluate learning effectiveness and improvements, e.g., by developing learning metrics (Caliskan et al., 2017).
8. Benchmarking.
   The final aspect that we identified as future work is the plans to conduct comparisons between the developed cyber ranges and security testbeds and others. In order to support this activities, we believe that developing a cyber range benchmark is essential for the future (Subaşu et al., 2017).

6. Discussion and conclusion

From the systematic literature review we confirmed our observations that the interest in cyber range and security testbeds has increased in the last few years as indicated in Fig. 2. We identified that scenarios play a major role in cyber range and security testbed development as indicated in Fig. 2. These scenarios focuse on cyber security testing, experimental and educational purposes as indicated in Fig. 4. These scenarios are executed on emulated, simulated, hybrid, or real equipment environment as indicated in Fig. 8. The execution of theses scenarios is either static or dynamic as indicated in Fig. 5. Static scenarios have a linear execution and they execute according to predefined process. Dynamic scenarios have a non linear execution and their execution depends upon the dynamic changes that are introduced in to the environment. These Dynamic changes are introduced by the teams involved in the scenario.

Most of the uses cases of cyber ranges and security testbeds are centered around the needs for red and blue team training as indicated in Fig. 7. The role of white and green teams need to be focused for cyber security scenario development and cyber security scenario management. A new trend of autonomous teams is starting to appear in cyber ranges and security testbed. These teams automate the role of red, blue and white teams, to reduce the time required in conducting cyber security exercises, tests and experiments. However, concrete methods to model the behavior of these teams are missing and *modeling of attack and defense scenarios for cyber ranges and security testbeds* are required for systemic execution and evaluation of a cyber security scenario.

The interest to use cyber ranges in testing, besides education, has increased in the last few years. This indicates that cyber ranges are not exclusively educational platforms, but can be used in other purposes, like testing. Most of the security test beds and cyber ranges are focusing on either quantitative evaluation methods or qualitative evaluation methods. Evaluation criteria which focuses on both quantitative and qualitative analysis on the security testbed and cyber ranges is missing. New evaluation metrics which focus on evaluating a single scenario on multiple test beds on both qualitative and quantitative manner will assist the evaluation of the security testbeds and cyber ranges in a systemic comparative analysis.

The Fig. 6 indicates that networking systems were the main application domain for cyber ranges and security testbeds, SCADA system started to gain attention from 2010, and in recent year cyber ranges and security testbeds have covered most application domain aforementioned. IOT, social engineering and testbeds for autonomous system are being developed. However, most of these testbeds use a hybrid environment in which they combine emulation, simulation and real equipment to produce most realistic cyber security environment for cyber security exercises, training, education and experiments.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Appendix: Citation Data

Table 13
Reviewed paper citation data as of Augusts 10 2018.

No.	Paper Title	Citation Count	Year Published
1	Design of Cyber Warfare Testbed	2	2019
2	Cyber security of a power grid: State-of-the-art	10	2018
3	Testbed@ TWISC: A network security experiment platform	0	2018
4	Achieving reproducible network environments with INSALATA	1	2017
5	A Survey on Smart Grid Cyber-Physical System Testbeds.	52	2017
6	Capability Detection and Evaluation Metrics for Cyber Security lab Exercises	0	2017
7	Control frameworks in network emulation testbeds: A survey	1	2017
8	Cybersecurity training in control systems using real equipment	1	2017
9	Design and implementation of cybersecurity testbed for industrial IoT systems	2	2017
10	Developing a capability to classify technical skill levels within a Cyber Range	0	2017
11	Experiment as a service	1	2017
12	Extending Our Cyber-Range CYRAN with Social Engineering Capabilities	0	2017
13	Gamifying ICS security training and research: Design, implementation, and results of S3	2	2017
14	Improving and Measuring Learning Effectiveness at Cyber Defense Exercises	1	2017
15	Instrumentation Research Methods for Cyber Security	7	2017
16	KYPO Cyber Range: Design and Use Cases	9	2017
17	Lessons learned from complex hands-on defence exercises in a cyber range	2	2017
18	Modeling and simulation architecture for training in cyber defence education	0	2017
19	The FUSE testbed: establishing a microgrid for smart grid security experiments	1	2017
20	Towards a Unified Data Storage and Generic Visualizations in Cyber Ranges	1	2017
21	Using virtual environments for the assessment of cybersecurity issues in IoT scenarios	14	2017
22	Advanced security testbed framework for wearable IoT devices	18	2016
23	Alfons: A Mimetic Network Environment Construction System	4	2016
24	Cybervan: A cyber security virtual assured network testbed	11	2016
25	CyRIS: A cyber range instantiation system for facilitating security training	12	2016
26	Design and architecture of an industrial IT security lab	4	2016
27	Developing a distributed software defined networking testbed for IoT	7	2016
28	PowerCyber: A remotely accessible testbed for Cyber Physical security of the Smart Grid	5	2016
29	RIO: A denial of service experimentation platform in a Future Internet Testbed	0	2016
30	Security of Cyber-Physical Systems	1	2016
31	Softgrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions	4	2016
32	Virtualization of industrial control system testbeds for cybersecurity	10	2016
33	A real-time testbed environment for cyber-physical security on the power grid	12	2015
34	Communications network analysis in a SCADA system testbed under cyber-attacks	9	2015
35	Cyber-attack and defense simulation framework	4	2015
36	Cyber modeling & simulation for cyber-range events	9	2015
37	Cyber-physical systems testbed based on cloud computing and software defined network	6	2015
38	Cyber-physical testbed The impact of cyber attacks and the human factor	3	2015
39	Experimentation on operational cyber security in CRATE	2	2015
40	i-tee: A fully automated Cyber Defense Competition for Students	7	2015
41	KYPO A Platform for Cyber Defence Exercises	7	2015
42	Sdsecurity: A software defined security experimental framework	44	2015
43	Understanding collaborative challenges in it security preparedness exercises	10	2015
44	Building a Virtual Cybersecurity Collaborative Learning Laboratory (VCCLL)	1	2014
45	Cloud-based security research testbed: A DDoS use case	10	2014
46	Cloud-based testbed for simulation of cyber attacks	23	2014
47	Cyber-physical testbeds	17	2014

(*continued on next page*)

Table 13 (*continued*)

No.	Paper Title	Citation Count	Year Published
48	Cyber security backdrop: A scada testbed	13	2014
49	Factors impacting performance in competitive cyber exercises	19	2014
50	FITS: A flexible virtual network testbed architecture	42	2014
51	National cyber range overview	14	2014
52	Ten Years of iCTF: The Good, The Bad, and The Ugly.	30	2014
53	The design of ics testbed based on emulation, physical, and simulation (eps-ics testbed)	13	2014
54	A survey of cyber ranges and testbeds	15	2013
55	Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid	197	2013
56	Instrumenting competition-based exercises to evaluate cyber defender situation awareness	13	2013
57	SCADAVT-A framework for SCADA security testbed based on virtualization technology	25	2013
58	Smart Grid Security Educational Training with ThunderCloud: A Virtual Security Test Bed	1	2013
59	This IS Child's Play	2	2013
60	When a testbed does more than testing: The Internet-Scale Event Attack and Generation Environment (ISEAGE)-providing learning and synthesizing experiences for â	5	2013
61	Amici: An assessment platform for multi-domain security experimentation on critical infrastructures	24	2012
62	Beyond network simulators: Fostering novel distributed applications and protocols through extendible design	6	2012
63	Cyber security exercises and competitions as a platform for cyber security experiments	20	2012
64	Cyber Security Assessment Tools and Methodologies	5	2012
65	Online assessment for hands-on cyber security training in a virtual lab	30	2012
66	Supervisory Command and Data Acquisition (SCADA) system cyber security analysis using a live, virtual, and constructive (LVC) testbed	36	2012
67	Towards an experimental testbed facility for cyber-physical security research	10	2012
68	A control system testbed to validate critical infrastructure protection concepts	69	2011
69	An overview of cyber attack and computer network operations simulation	28	2011
70	A testbed for analyzing security of SCADA control systems (TASSCS)	95	2011
71	DefEX: Hands-On Cyber Defense Exercise for Undergraduate Students	7	2011
72	Hit'em where it hurts: a live security exercise on cyber situational awareness	34	2011
73	Practical network security teaching in an online virtual laboratory	16	2011
74	Using an Emulation Testbed for Operational Cyber Security Exercises	2	2011
75	An experimental platform for assessing SCADA vulnerabilities and countermeasures in power plants	61	2010
76	An Intelligent network for federated testing of NetCentric systems	4	2010
77	A survey of software tools for the creation of networked testbeds	14	2010
78	Design of a virtual computer lab environment for hands-on information security exercises	13	2010
79	Organizing large scale hacking competitions	44	2010
80	The Blunderdome: An Offensive Exercise for Building Network, Systems, and Web Security Awareness.	6	2010
81	The DETER project: Advancing the science of cyber security experimentation and test	78	2010
82	Current developments in DETER cybersecurity testbed technology	26	2009
83	Guide for designing cyber security exercises	30	2009
84	Real-time security exercises on a realistic interdomain routing experiment platform	7	2009
85	The Cyber Scenario Modeling and Reporting Tool (CyberSMART)	3	2009
86	Network modelling and simulation tools	63	2009
87	Design on SCADA test-bed and security device	29	2008
88	Cyber attack modeling and simulation for network security analysis	82	2007
89	Large-scale reconfigurable virtual testbed for information security experiments	18	2007
90	The development and deployment of a multi-user, remote access virtualization system for networking, security, and system administration classes	84	2007
91	A virtual machine architecture for creating IT-security laboratories	13	2006
92	Ethical hacking and password cracking: a pattern for individualized security exercises	5	2006
93	Experience with deter: a testbed for security research	195	2006
94	Teaching hands-on Linux host computer security	3	2006
95	Exploring a national cybersecurity exercise for universities	78	2005
96	Rinse: The real-time immersive network simulation environment for network security exercises	82	2005
97	The INFN-grid testbed	9	2005
98	ViSe: A virtual security testbed	14	2005
99	An integrated experimental environment for distributed systems and networks	1667	2002
100	Lariat: Lincoln adaptable real-time information assurance testbed	112	2002

References

Al-Ayyoub, M., Jararweh, Y., Benkhelifa, E., Vouk, M., Rindos, A., et al., 2015. Sdsecurity: a software defined security experimental framework. In: Communication Workshop (ICCW), 2015 IEEE International Conference on. IEEE, pp. 1871–1876.

Alfieri, R., Barbera, R., Belluomo, P., Cavalli, A., Cecchini, R., Chierici, A., Ciaschini, V., DellAgnello, L., Donno, F., Ferro, E., et al., 2005. The infn-grid testbed. Future Gener. Comput. Syst. 21 (2), 249–258.

Almalawi, A., Tari, Z., Khalil, I., Fahad, A., 2013. Scadavt-a framework for scada security testbed based on virtualization technology. In: Local Computer Networks (LCN), 2013 IEEE 38th Conference on. IEEE, pp. 639–646.

Alvarenga, I.D., Duarte, O.C.M., 2016. Rio: a denial of service experimentation platform in a future internet testbed. In: Network of the Future (NOF), 2016 7th International Conference on the. IEEE, pp. 1–5.

Alves, T., Das, R., Morris, T., 2016. Virtualization of industrial control system testbeds for cybersecurity. In: Proceedings of the 2nd Annual Industrial Control System Security Workshop. ACM, pp. 10–14.

Antonioli, D., Ghaeini, H.R., Adepu, S., Ochoa, M., Tippenhauer, N.O., 2017. Gamifying ics security training and research: design, implementation, and results of s3. In: Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy. ACM, pp. 93–102.

Ashok, A., Krishnaswamy, S., Govindarasu, M., 2016. Powercyber: a remotely accessible testbed for cyber physical security of the smart grid. In: Innovative Smart Grid Technologies Conference (ISGT), 2016 IEEE Power & Energy Society. IEEE, pp. 1–5.

Balenson, D., Tinnel, L., Benzel, T., 2015. Cybersecurity experimentation of the future (cef): catalyzing a new generation of experimental cybersecurity research. Tech. Rep.. SRI International.

Barcellos, M.P., Antunes, R.S., Muhammad, H.H., Munaretti, R.S., 2012. Beyond network simulators: fostering novel distributed applications and protocols through extendible design. J. Netw. Comput. Appl. 35 (1), 328–339.

Benzel, T., Braden, B., Faber, T., Mirkovic, J., Schwab, S., Sollins, K., Wroclawski, J., 2009. Current developments in deter cybersecurity testbed technology. In: Conference for Homeland security, 2009. CATCH'09. Cybersecurity applications & technology. IEEE, pp. 57–70.

Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A., Sklower, K., Ostrenga, R., Schwab, S., 2006. Experience with deter: a testbed for security research. In: Testbeds and Research Infrastructures for the Development of Networks and

Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on. IEEE. 10–pp.
Bergin, D.L., 2015. Cyber-attack and defense simulation framework. J. Defense Model. Simul. 12 (4), 383–392.
Beuran, R., Tang, D., Pham, C., Chinen, K.-i., Tan, Y., Shinoda, Y., 2018. Integrated framework for hands-on cybersecurity training: cytrone. Comput. Secur..
Border, C., 2007. The development and deployment of a multi-user, remote access virtualization system for networking, security, and system administration classes. ACM SIGCSE Bull. 39 (1), 576–580.
Braidley, S., 2016. Extending our cyber-range cyran with social engineering capabilities. De Montfort University MSc Thesis Report.
Caliskan, E., Tatar, U., Bahsi, H., Ottis, R., Vaarandi, R., 2017. Capability detection and evaluation metrics for cyber security lab exercises. In: ICMLG2017 5th International Conference on Management Leadership and Governance. Academic Conferences and publishing limited, p. 407.
Čeleda, P., Čegan, J., Vykopal, J., Tovarňák, D., 2015. Kypo–a platform for cyber defence exercises. M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence. NATO Science and Technology Organization.
Chadha, R., Bowen, T., Chiang, C.-Y.J., Gottlieb, Y.M., Poylisher, A., Sapello, A., Serban, C., Sugrim, S., Walther, G., Marvel, L.M., et al., 2016. Cybervan: A cyber security virtual assured network testbed. In: Military Communications Conference, MILCOM 2016-2016 IEEE. IEEE, pp. 1125–1130.
Chandra, Y., Mishra, P.K., 2019. Design of cyber warfare testbed. In: Software Engineering. Springer, pp. 249–256.
Chiang, C.J., Poylisher, A., Gottlieb, Y., Serban, C., 2013. Cyber testing tools and methodologies. Presentation at ITEA, November.
Childers, N., Boe, B., Cavallaro, L., Cavedon, L., Cova, M., Egele, M., Vigna, G., 2010. Organizing large scale hacking competitions. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, pp. 132–152.
Chitu, O., Kira, S., 2010. A guide to conducting a systematic literature review of information systems research. Sprouts: Working Pap. Inf. Syst. 26 (10).
Chow, E., James, M., Chang, H.-P., Vatan, F., Sudhir, G., 2010. An intelligent network for federated testing of netcentric systems. In: Policies for Distributed Systems and Networks (POLICY), 2010 IEEE International Symposium on. IEEE, pp. 44–52.
Cintuglu, M.H., Mohammed, O.A., Akkaya, K., Uluagac, A.S., 2017. A survey on smart grid cyber-physical system testbeds.. IEEE Commun. Surv. Tutor. 19 (1), 446–464.
cybersecuritydegrees, A comprehensive list of cyber security competitions.
Damodaran, S.K., Smith, K., 2015. CRIS Cyber Range Lexicon, Version 1.0. Technical Report. Massachusetts Institute of Technology Lexington Lincoln Laboratory.
Davis, J., Magrath, S., 2013. A survey of cyber ranges and testbeds. Technical Report. Defence Science and Technology Organization Edinburgh (Australia) Cyber and Electronic Warfare Division.
Domínguez, M., Prada, M.A., Reguera, P., Fuertes, J.J., Alonso, S., Morán, A., 2017. Cybersecurity training in control systems using real equipment. IFAC-PapersOnLine 50 (1), 12179–12184.
Doupé, A., Egele, M., Caillat, B., Stringhini, G., Yakin, G., Zand, A., Cavedon, L., Vigna, G., 2011. Hit'em where it hurts: a live security exercise on cyber situational awareness. In: Proceedings of the 27th Annual Computer Security Applications Conference. ACM, pp. 51–61.
Edgar, T., Manz, D., Carroll, T., 2011. Towards an experimental testbed facility for cyber-physical security research. In: Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research. ACM, p. 53.
Edgar, T.W., Manz, D.O., 2017. Research Methods for Cyber Security. Syngress.
Edgar, T.W., Rice, T.R., 2017. Experiment as a service. In: Technologies for Homeland Security (HST), 2017 IEEE International Symposium on. IEEE, pp. 1–6.
Ernits, M., Tammekänd, J., Maennel, O., 2015. i-tee: a fully automated cyber defense competition for students. In: ACM SIGCOMM Computer Communication Review, 45. ACM, pp. 113–114.
Farooqui, A.A., Zaidi, S.S.H., Memon, A.Y., Qazi, S., 2014. Cyber security backdrop: a scada testbed. In: Computing, Communications and IT Applications Conference (ComComAp), 2014 IEEE. IEEE, pp. 98–103.
Ferguson, B., Tall, A., Olsen, D., 2014. National cyber range overview. In: Military Communications Conference (MILCOM), 2014 IEEE. IEEE, pp. 123–128.
Flauzac, O., Gonzalez, C., Nolot, F., 2016. Developing a distributed software defined networking testbed for iot. Procedia Comput. Sci. 83, 680–684.
Fovino, I.N., Masera, M., Guidi, L., Carpi, G., 2010. An experimental platform for assessing scada vulnerabilities and countermeasures in power plants. In: Human System Interactions (HSI), 2010 3rd Conference on. IEEE, pp. 679–686.
Furfaro, A., Argento, L., Parise, A., Piccolo, A., 2017. Using virtual environments for the assessment of cybersecurity issues in iot scenarios. Simul. Model. Pract. Theory 73, 43–54.
Furnell, S., Fischer, P., Finch, A., 2017. Can't get the staff? the growing need for cyber-security skills. Comput. Fraud Secur. 2017 (2), 5–10.
Gao, H., Peng, Y., Dai, Z., Wang, T., Jia, K., 2013. The design of ics testbed based on emulation, physical, and simulation (eps-ics testbed). In: Intelligent Information Hiding and Multimedia Signal Processing, 2013 Ninth International Conference on. IEEE, pp. 420–423.
Gao, H., Peng, Y., Jia, K., Wen, Z., Li, H., 2015. Cyber-physical systems testbed based on cloud computing and software defined network. In: Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2015 International Conference on. IEEE, pp. 337–340.
Gavras, A., Karila, A., Fdida, S., May, M., Potts, M., 2007. Future internet research and experimentation: the fire initiative. ACM SIGCOMM Comput. Commun. Rev. 37 (3), 89–92.
Genge, B., Siaterlis, C., Hohenadel, M., 2012. Amici: an assessment platform for multi-domain security experimentation on critical infrastructures. In: International Workshop on Critical Information Infrastructures Security. Springer, pp. 228–239.
Gephart, N., Kuperman, B.A., 2010. Design of a virtual computer lab environment for hands-on information security exercises. J. Comput. Sci. Colleges 26 (1), 32–39.
Glumich, S.M., Kropa, B.A., 2011. DefEX: Hands-On Cyber Defense Exercise for Undergraduate Students. Technical Report. Air Force Research Lab Rome NY Information Directorate.
Gunathilaka, P., Mashima, D., Chen, B., 2016. Softgrid: a software-based smart grid testbed for evaluating substation cybersecurity solutions. In: Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy. ACM, pp. 113–124.
Gurnani, R., Pandey, K., Rai, S.K., 2014. A scalable model for implementing cyber security exercises. In: Computing for Sustainable Global Development (INDIACom), 2014 International Conference on. IEEE, pp. 680–684.
Hahn, A., Ashok, A., Sridhar, S., Govindarasu, M., 2013. Cyber-physical security testbeds: architecture, application, and evaluation for smart grid. IEEE Trans. Smart Grid 4 (2), 847–855.
Herold, N., Wachs, M., Dorfhuber, M., Rudolf, C., Liebald, S., Carle, G., 2017. Achieving reproducible network environments with insalata. In: IFIP International Conference on Autonomous Infrastructure, Management and Security. Springer, pp. 30–44.
Hoffman, L.J., Rosenberg, T., Dodge, R., Ragsdale, D., 2005. Exploring a national cybersecurity exercise for universities. IEEE Secur. Privacy 3 (5), 27–33.
Holm, H., Karresand, M., Vidström, A., Westring, E., 2015. A survey of industrial control system testbeds. In: Buchegger, S., Dam, M. (Eds.), Secure IT Systems. Springer International Publishing, Cham, pp. 11–26.
Holm, H., Sommestad, T., 2016. Sved: scanning, vulnerabilities, exploits and detection. In: Military Communications Conference, MILCOM 2016-2016 IEEE. IEEE, pp. 976–981.
Hu, J., Cordel, D., Meinel, C., 2006. A virtual machine architecture for creating it-security laboratories.
Jirsik, T., Husak, M., Celeda, P., Eichler, Z., 2014. Cloud-based security research testbed: a ddos use case. In: Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, pp. 1–2.
Jung, S., Song, J.-g., Kim, S., 2008. Design on scada test-bed and security device. Int. J. Multim. Ubiq.Eng. 3 (4), 75–86.
Kick, J., 2014. Cyber exercise playbook. Technical Report. MITRE CORP BEDFORD MA.
Kouril, D., Rebok, T., Jirsik, T., Cegan, J., Drasar, M., Vizváry, M., Vykopal, J., 2014. Cloud-based testbed for simulation of cyber attacks. In: Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, pp. 1–6.
Koutsandria, G., Gentz, R., Jamei, M., Scaglione, A., Peisert, S., McParland, C., 2015. A real-time testbed environment for cyber-physical security on the power grid. In: Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy. ACM, pp. 67–78.
Kuhl, M.E., Kistner, J., Costantini, K., Sudit, M., 2007. Cyber attack modeling and simulation for network security analysis. In: Proceedings of the 39th Conference on Winter Simulation: 40 years! The best is yet to come. IEEE Press, pp. 1180–1188.
Labuschagne, W.A., Grobler, M., 2017. Developing a capability to classify technical skill levels within a cyber range. In: ECCWS 2017 16th European Conference on Cyber Warfare and Security. Academic Conferences and publishing limited, p. 224.
Langner, R., 2011. Stuxnet: dissecting a cyberwarfare weapon. IEEE Secur. Privacy 9 (3), 49–51.
Leblanc, S.P., Partington, A., Chapman, I., Bernier, M., 2011a. An overview of cyber attack and computer network operations simulation. In: Proceedings of the 2011 Military Modeling & Simulation Symposium. Society for Computer Simulation International, San Diego, CA, USA, pp. 92–100.
Leblanc, S.P., Partington, A., Chapman, I., Bernier, M., 2011b. An overview of cyber attack and computer network operations simulation. In: Proceedings of the 2011 Military Modeling & Simulation Symposium. Society for Computer Simulation International, pp. 92–100.
Lee, S., Lee, S., Yoo, H., Kwon, S., Shon, T., 2017. Design and implementation of cybersecurity testbed for industrial iot systems. J.Supercomput. 1–15.
Li, Y., Liljenstam, M., Liu, J., 2009. Real-time security exercises on a realistic interdomain routing experiment platform. In: Principles of Advanced and Distributed Simulation, 2009. PADS'09. ACM/IEEE/SCS 23rd Workshop on. IEEE, pp. 54–63.
Liljenstam, M., Liu, J., Nicol, D., Yuan, Y., Yan, G., Grier, C., 2005. Rinse: the real-time immersive network simulation environment for network security exercises. In: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation. IEEE Computer Society, pp. 119–128.
Line, M.B., Moe, N.B., 2015. Understanding collaborative challenges in it security preparedness exercises. In: IFIP International Information Security Conference. Springer, pp. 311–324.
Louthan, G., Roberts, W., Butler, M., Hale, J., 2010. The blunderdome: an offensive exercise for building network, systems, and web security awareness.. CSET.
Maennel, K., Ottis, R., Maennel, O., 2017. Improving and measuring learning effectiveness at cyber defense exercises. In: Nordic Conference on Secure IT Systems. Springer, pp. 123–138.
Mallouhi, M., Al-Nashif, Y., Cox, D., Chadaga, T., Hariri, S., 2011. A testbed for analyzing security of scada control systems (tasscs). In: Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES. IEEE, pp. 1–7.
Marshall, J., 2009. The cyber scenario modeling and reporting tool (cybersmart). In: Cybersecurity Applications & Technology Conference For Homeland Security. IEEE, pp. 305–309.

Miciolino, E.E., Bernieri, G., Pascucci, F., Setola, R., 2015. Communications network analysis in a scada system testbed under cyber-attacks. In: Telecommunications Forum Telfor (TELFOR), 2015 23rd. IEEE, pp. 341–344.
Mirkovic, J., Benzel, T.V., Faber, T., Braden, R., Wroclawski, J.T., Schwab, S., 2010. The deter project: advancing the science of cyber security experimentation and test. In: Technologies for Homeland Security (HST), 2010 IEEE International Conference on. IEEE, pp. 1–7.
Moraes, I.M., Mattos, D.M., Ferraz, L.H.G., Campista, M.E.M., Rubinstein, M.G., Costa, L.H.M., de Amorim, M.D., Velloso, P.B., Duarte, O.C.M., Pujolle, G., 2014. Fits: a flexible virtual network testbed architecture. Comput. Netw. 63, 221–237.
Morris, T., Srivastava, A., Reaves, B., Gao, W., Pavurapu, K., Reddi, R., 2011. A control system testbed to validate critical infrastructure protection concepts. Int. J. Crit. Infrastruct.Protect. 4 (2), 88–103.
Murphy, J., Sihler, E., Ebben, M., Lovewell, L., Wilson, G., 2014. Building a virtual cybersecurity collaborative learning laboratory (vccll). In: Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 1.
Ošlejšek, R., Toth, D., Eichler, Z., Burská, K., 2017. Towards a unified data storage and generic visualizations in cyber ranges. In: ECCWS 2017 16th European Conference on Cyber Warfare and Security. Academic Conferences and publishing limited, p. 298.
Palleschi, A., 2010. Pentagon fought proposal: congress adopts provision to halt funding for national cyber range. Inside the Air Force 21 (51), 10–10.
Patriciu, V.-V., Furtuna, A.C., 2009. Guide for designing cyber security exercises. In: Proceedings of the 8th WSEAS International Conference on E-Activities and information security and privacy. World Scientific and Engineering Academy and Society (WSEAS), pp. 172–177.
Pfrang, S., Kippe, J., Meier, D., Haas, C., 2016. Design and architecture of an industrial it security lab. In: International Conference on Testbeds and Research Infrastructures. Springer, pp. 114–123.
Pham, C., Tang, D., Chinen, K.-i., Beuran, R., 2016. Cyris: A cyber range instantiation system for facilitating security training. In: Proceedings of the Seventh Symposium on Information and Communication Technology. ACM, pp. 251–258.
Qassim, Q., Jamil, N., Abidin, I.Z., Rusli, M.E., Yussof, S., Ismail, R., Abdullah, F., Ja'afar, N., Hasan, H.C., Daud, M., 2017. A survey of scada testbed implementation approaches. Indian J. Sci. Technol. 10 (26).
Rahman, M.A., Pakštas, A., Wang, F.Z., 2009. Network modelling and simulation tools. Simul. Model. Pract. Theory 17 (6), 1011–1031.
Reed, T., Nauer, K., Silva, A., 2013. Instrumenting competition-based exercises to evaluate cyber defender situation awareness. In: International Conference on Augmented Cognition. Springer, pp. 80–89.
Richmond, M., 2005. Vise: a virtual security testbed. Tech. Rep. University of California, Santa Barbara.
Rossey, L.M., Cunningham, R.K., Fried, D.J., Rabek, J.C., Lippmann, R.P., Haines, J.W., Zissman, M.A., 2002. Lariat: Lincoln adaptable real-time information assurance testbed. In: Aerospace Conference Proceedings, 2002. IEEE, 6. IEEE. 6–6.
Rubio-Hernan, J., Rodolfo-Mejias, J., Garcia-Alfaro, J., 2016. Security of cyber-physical systems. In: Conference on Security of Industrial-Control-and Cyber-Physical Systems. Springer, pp. 3–18.
Rursch, J.A., Jacobson, D., 2013a. This is child's play creating a "playground"(computer network testbed) for high school students to learn, practice, and compete in cyber defense competitions. In: Frontiers in Education Conference, 2013 IEEE. IEEE, pp. 1776–1778.
Rursch, J.A., Jacobson, D., 2013b. When a testbed does more than testing: the internet-scale event attack and generation environment (iseage)-providing learning and synthesizing experiences for cyber security students.. In: Frontiers in Education Conference, 2013 IEEE. IEEE, pp. 1267–1272.
Schepens, W.J., Ragsdale, D.J., Surdu, J.R., Schafer, J., New Port, R., 2002. The cyber defense exercise: an evaluation of the effectiveness of information assurance education. J. Inf. Secur. 1 (2), 1–14.
Schreuders, Z.C., Shaw, T., Ravichandran, G., Keighley, J., Ordean, M., et al., 2017. Security scenario generator (secgen): a framework for generating randomly vulnerable rich-scenario vms for learning computer security and hosting ctf events. USENIX. USENIX Association.
Shumba, R., 2006. Teaching hands-on linux host computer security. J. Educ. Resour. Comput.(JERIC) 6 (3), 5.
Siaterlis, C., Genge, B., 2014. Cyber-physical testbeds. Commun. ACM 57 (6), 64–73.
Siaterlis, C., Masera, M., 2009. A review of available software for the creation of testbeds for internet security research. In: 2009 First International Conference on Advances in System Simulation, pp. 79–87. doi:10.1109/SIMUL.2009.33.
Siaterlis, C., Masera, M., 2010. A survey of software tools for the creation of networked testbeds. Int. J. Adv. Secur. 3 (2), 1–12.
Siaterlis, C., Perez-Garcia, A., Masera, M., 2011. Using an emulation testbed for operational cyber security exercises. In: International Conference on Critical Infrastructure Protection. Springer, pp. 185–199.
Siboni, S., Shabtai, A., Tippenhauer, N.O., Lee, J., Elovici, Y., 2016. Advanced security testbed framework for wearable iot devices. ACM Trans. Internet Technol. (TOIT) 16 (4), 26.
Silva, A., McClain, J., Reed, T., Anderson, B., Nauer, K., Abbott, R., Forsythe, C., 2014. Factors impacting performance in competitive cyber exercises. In: Proceedings of the Interservice/Interagency Training, Simulation and Education Conference, Orlando, FL.
Snyder, R., 2006. Ethical hacking and password cracking: a pattern for individualized security exercises. In: Proceedings of the 3rd annual conference on Information security curriculum development. ACM, pp. 13–18.
Sommestad, T., 2015. Experimentation on operational cyber security in crate. NATO STO-MP-IST-133 Specialist Meeting, Copenhagen, Denmark.
Sommestad, T., Hallberg, J., 2012. Cyber security exercises and competitions as a platform for cyber security experiments. In: Nordic Conference on Secure IT Systems. Springer, pp. 47–60.
Soupionis, Y., Benoist, T., 2015. Cyber-physical testbedthe impact of cyber attacks and the human factor. In: Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for. IEEE, pp. 326–331.
Staff, U.J., 2012. Joint training manual for the armed forces of the united states (cjcsm 3500.03 d). Washington, DC: Joint Chiefs of Staff.
StepForward, Carnegie mellon university - software engineering institute.
Stites, J., Siraj, A., Brown, E.L., 2013. Smart grid security educational training with thundercloud: a virtual security test bed. In: Proceedings of the 2013 on InfoSecCD'13: Information Security Curriculum Development Conference. ACM, p. 105.
Subaşu, G., Roşu, L., Bădoi, I., 2017. Modeling and simulation architecture for training in cyber defence education. In: Electronics, Computers and Artificial Intelligence (ECAI), 2017 9th International Conference on. IEEE, pp. 1–4.
Sun, C.-C., Hahn, A., Liu, C.-C., 2018. Cyber security of a power grid: state-of-the-art. Int. J. Electr. Power Energy Syst. 99, 45–56.
Tsai, P.-W., Piccialli, F., Tsai, C.-W., Luo, M.-Y., Yang, C.-S., 2017. Control frameworks in network emulation testbeds: a survey. J. Comput. Sci. 22, 148–161.
Tsai, P.-W., Yang, C.-S., 2018. Testbed@ twisc: a network security experiment platform. Int. J. Commun. Syst. 31 (2), e3446.
Urias, V., Van Leeuwen, B., Richardson, B., 2012. Supervisory command and data acquisition (scada) system cyber security analysis using a live, virtual, and constructive (lvc) testbed. In: Military Communications Conference, 2012-MILCOM 2012. IEEE, pp. 1–8.
Vigna, G., Borgolte, K., Corbetta, J., Doupe, A., Fratantonio, Y., Invernizzi, L., Kirat, D., Shoshitaishvili, Y., 2014. Ten years of ictf: the good, the bad, and the ugly.. 3GSE.
Volynkin, A., Skormin, V., 2007. Large-scale reconfigurable virtual testbed for information security experiments. In: Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on. IEEE, pp. 1–9.
Vykopal, J., Ošlejšek, R., Čeleda, P., Vizvary, M., Tovarňák, D., 2017a. Kypo cyber range: design and use cases.
Vykopal, J., Vizváry, M., Oslejsek, R., Celeda, P., Tovarnak, D., 2017b. Lessons learned from complex hands-on defence exercises in a cyber range. In: Frontiers in Education Conference (FIE). IEEE, pp. 1–8.
White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A., 2002. An integrated experimental environment for distributed systems and networks. ACM SIGOPS Oper. Syst. Rev. 36 (SI), 255–270.
White, G.B., Williams, D., 2005. The collegiate cyber defense competition. In: Proceedings of the 9th Colloquium for Information Systems Security Education.
Willems, C., Meinel, C., 2011. Practical network security teaching in an online virtual laboratory. In: Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 1.
Willems, C., Meinel, C., 2012. Online assessment for hands-on cyber security training in a virtual lab. In: Global Engineering Education Conference (EDUCON), 2012 IEEE. IEEE, pp. 1–10.
Wood, B.J., Duggan, R.A., 2000. Red teaming of advanced information assurance concepts. In: DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings, 2. IEEE, pp. 112–118.
Xypolytou, E., Fabini, J., Gawlik, W., Zseby, T., 2017. The fuse testbed: establishing a microgrid for smart grid security experiments. e & i Elektrotechnik und Informationstechnik 134 (1), 30–35.
Yasuda, S., Miura, R., Ohta, S., Takano, Y., Miyachi, T., 2016. Alfons: a mimetic network environment construction system. In: International Conference on Testbeds and Research Infrastructures. Springer, pp. 59–69.

Muhammad Mudassar Yamin Mudassar is currently a PhD candidate at Norwegian University of Science and Technology, his focus of research is systems security. He holds multiple cyber security certifications like OSCP, LPT-MASTER, CEH, CHFI, CPTE, CISSO, CBP. A list of his publication can be found here: https://scholar.google.no/citations?user=Do_xVQMAAAAJ&hl=en.

Basel Katt is currently working as an Associate Professor in Norwegian University of Science and Technology. His focus of research is:

- Software security and security testing
- Software vulnerability analysis
- Model driven software development and model driven security
- Access control, usage control and privacy protection
- Security monitoring, policies, languages, models and enforcement

A list of his publication can be found here: https://wo.cristin.no/as/WebObjects/cristin.woa/wa/fres?sort=ar&pnr=811108&action=sok.

Vasileios Gkioulos is currently working as a post-doctoral research fellow. His area of research is security of cyber physical systems. A list of his publication can be found here: https://scholar.google.no/citations?user=Jgo7_q4AAAAJ&hl=en.

2.4 Serious games as a tool to model attack and defense scenarios for cyber-security exercises

COMPUTERS & SECURITY 110 (2021) 102450

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Serious games as a tool to model attack and defense scenarios for cyber-security exercises

Muhammad Mudassar Yamin, Basel Katt, Mariusz Nowostawski*

Norwegian University of Science and Technology, Gjøvik 2815, Norway

ARTICLE INFO

Article history:
Received 30 March 2020
Revised 8 May 2021
Accepted 16 August 2021
Available online 20 August 2021

Keywords:
Cyber range
Cyber-security
Exercises
Scenarios
Attack
Defense

ABSTRACT

Technology is evolving rapidly; this poses a problem for security specialists and average citizens as their technological skill sets are quickly made obsolete. This makes the knowledge and understanding of cyber-security in a technologically evolving world difficult. Global IT infrastructure and individuals' privacy are constantly under threat. One way to tackle this problem is by providing continuous training and self-learning platforms. Cyber-security exercises can provide a necessary platform for training people's cyber-security skills. However, conducting cyber-security exercises with new and unique scenarios requires comprehensive planning and commitment to the preparation time and resources. In this work, we propose a serious game for the development of cyber-security exercise scenarios. The game provides a platform to model simulated cyber-security exercise scenarios, transforming them into an emulated cyber-security exercise environment using domain-specific language (DSL) and infrastructure orchestration. In this game, players can play as cyber attackers or defenders in a multiplayer environment to make operational cyber-security decisions in real-time. The decisions are evaluated for the development of operational cyber-attack and defense strategies.

1. Introduction

During the European cyber-security challenge (ecs, 2019), we found that the teams involved were facing problems in strategizing their approach for solving cyber-security exercise scenarios. The team members had a sufficient level of technical skills to tackle the technical problems present in the challenge, but their decision-making skills in prioritizing the best moves were lagging. One way to overcome this issue would be by conducting many operational cyber-security exercises with unique scenarios, such that the exercise participants could get the right level of experience in decision making. However, conducting such exercises is resource intensive and time-consuming (Yamin and Katt, 2018). Therefore, in the current research, we investigate an efficient way to conduct cyber-security exercises that can help exercise the participants' skill improvement.

Through a review of the literature (Amorim et al., 2013; Hendrix et al., 2016; Schreuders and Butterfield, 2016), we identified that serious games are actively used in cyber-security skill development. However, one problem with such games is related to the static level design, which makes the integration of new and unique scenarios difficult. Another problem is that many are turned-based games as opposed to real-time strategy games. Real-time strategy games enhance players' cognitive flexibility (Glass et al., 2013) and help in training for the dynamic nature of cyber-security scenarios. To address

* Corresponding author.
E-mail addresses: muhammad.m.yamin@ntnu.no (M.M. Yamin), basel.katt@ntnu.no (B. Katt), mariusz.nowostawski@ntnu.no (M. Nowostawski).
https://doi.org/10.1016/j.cose.2021.102450
0167-4048/

these issues and challenges, we ask the following research questions:

1. How can serious games be used to model dynamic cyber-security exercise scenarios in a realistic manner?
2. How can modeled cyber-security exercises be used in devising cyber attack and defense strategies in a realistic manner?
3. Is it efficient to conduct cyber-security exercises in a simulated modeled environment for exercise participants' skill improvement?
4. Is it efficient to transform simulated cyber-security exercise scenarios in a game to an emulated infrastructure, and how usable, flexible, and complete is it?

To answer the research questions, we developed (1) a real-time cyber-security strategy game that enhances players' cognitive flexibility and (2) a cyber-security exercise scenario domain-specific language (DSL) to model the scenario and generate the emulated infrastructure. The game is thoroughly assessed using surveys given to a large group of participants during the Norwegian cyber-security challenge 2019 (sta, 2019); the findings of this survey are presented in this article.

The rest of the current article is organized as follows: First, we share the research background and key concepts of cyber-security exercises. Then, we proceed with sharing a brief related work on serious games in cyber-security. Continuing this, we will state our research methodology, present our cyber-security strategy game with the developed DSL, and their assessment and evaluation. Finally, we conclude the article with a discussion and conclusion.

2. Research background

The importance of cyber warfare training is critical when considered in light of contemporary examples, such as the cyber-attacks on Estonia and the crippling of Georgia's government websites using advanced hacking methods. Cyber-warfare was first deployed during *Operation Desert Storm* against Iraq. The communication networks of the Iraqi forces were crippled, so they were forced to use less-secure microwave communication, which was easily intercepted and led to their eventual defeat. The Bosnia–Herzegovina war saw the use of cyber warfare to cripple governmental infrastructure to such an extent that the paramilitary force was turned against the actual military. In the 1990s, the U.S. (United States) government realized that they were vulnerable to cyber-attacks; they had been using offensive cyber capabilities to achieve their tactical and strategic objectives, which resulted in a similar response from other state actors.

Norway is one of the leading digital nations in the world. The government encourages public and private sectors to take part in digital innovations for the country's progress. Digital infrastructure is challenged by many factors, including the existence of complex vulnerabilities that can be exploited by advanced cyber-security attacks, along with the need to secure the provision of successful digitalization solutions. Norway was the first country to make a cyber-security strategy in 2003 and then revised it in 2007 and 2012 (cyb, 2012). A follow-up report was also published, emphasizing responsibility for securing digital assets. The present strategy is Norway's fourth cyber-security strategy, and its purpose is to address the challenges of digitalization faced by Norwegian society (Nat, 2020). While designing strategies, stakeholders from the public and private sectors are taken into account.

More than 300 delegates took part in the formulation of Norway's present strategy. The introduction deals with the challenges, strategy, vision, and strategic goals. In the introduction, it is noted that Norwegian society has become immensely digitalized to benefit private individuals, companies, and authorities. Here, the challenge of digitalization is cyber threats and the thorough dependency of society on digitalization. Digital infrastructures and systems are growing more complex, global, and integrated. All kinds of devices are being connected to the Internet. The fast speed of digitalization makes it challenging to forecast the resulting threats. Some threats include ransomware, industrial espionage, sabotage, blackmail, cyber-bullying, and identity theft (Nat, 2020). The strategy aims to address cyber-security issues, but to do this, the appropriate authorities must give access to a broad range of tools, along with the development of regulations and knowledge of supervisory activities. A follow-up report (lis, 2021) was released, highlighting the 50 steps taken by the government to implement the national cyber-security strategy. Step 27, 41, and 42 are as follows:

- Development of the Norwegian Cyber Range (NCR), which will be the first national test arena for cyber-security.
- Conducting *National cyber-security exercise*.
- Participation in international exercises such as NATO coalition, Locked Shields, Cyber Europe, and NATOs CMX.

2.1. *Cyber-security training and exercises*

There is a constant need for training and self-learning platforms to achieve the stated objectives for cyber-security education. Cyber-security threats are on the rise, so the field of cyber-security education is emerging to train the next generation of cyber-security professionals (Ford et al., 2017). However, the cyber-security field faces a skills gap problem because of the nature of the rapidly changing cyber-security environment (Endicott-Popovsky and Popovsky, 2014). This situation makes it difficult to train and educate the next generation of cyber-security professionals. There are two main types of cyber-security exercises. The first is table-top discussions, and the second is practical hands-on operation-based exercises (Gurnani et al., 2014). Table-top exercises are discussion based and conducted in the form of seminars, workshops, and idea exchanges mostly related to policy-oriented issues. In comparison, most operation-based cyber-security education and training programs employ hands-on activities aiming to improve the exercise participants' technical skills and abilities. These cyber-security exercises are executed in simulated, emulated, physical, or hybrid practice environments. Recent studies have identified that the required practice environments are being developed via manual setup and configuration, a methodology that is ineffective, tedious, and error

prone (Beuran et al., 2018). These practice environments are known as cyber-ranges.

According to Pham et al. (2016), cyber-ranges are well-defined controlled (virtual) exercise environments that are used in cyber-security training to efficiently help trainees gain practical knowledge through hands-on activities. Creating these exercise environments that contain the necessary features such as network topology, virtual machines, and security-related content is not an easy task (Beuran et al., 2018; Yamin and Katt, 2018). Many cyber-ranges try to automate the creation of these exercise environments, such as with Cytrone (Beuran et al., 2018), CyberVAN (Chadha et al., 2016), Cyris (Pham et al., 2016), Telelab (Casini et al., 2003), and Secgen (Schreuders et al., 2017). Like the natural environment in which animals and plants interact with the environment to utilize its resources, these exercise environments need to be interacted with to utilize the resources in them. These interactions can be done in the form of cyber-security exercises, an assessment of new technologies, a vulnerability assessment, malicious activity profiling, security data generation, and so forth. Individuals and teams on a cyber-range perform these interactions. In terms of operation-based cyber-security exercises, these teams include the following:

1. White team: A team that creates or generates a cyber-security exercise environment.
2. Red team: A team that attacks the cyber-security exercise environment.
3. Blue team: A team that defends the cyber-security exercise environment.

Multiple additional teams are also part of cyber-security exercises and can include Green, Orange, Yellow, and Purple teams, which we have explained in our previous work (Yamin et al., 2019). Their involvement solely depends on the scale and objectives of an exercise. However, in the current work, we are only focusing on the White, Red, and Blue teams. These teams are primarily involved in three main types of cyber-security exercises:

1. Cyber-attack exercise: theses exercises are conducted to train, assess, and evaluate the performance of red teams. An environment is created by a white team in which red teams need to achieve specific objectives to compromise the exercise environment in a particular time period.
2. Cyber-defense exercise: these exercises are conducted to train, assess, and evaluate the blue team's performance. A white team creates an exercise environment. A blue team needs to investigate and prevent cyber-attacks by red teams and to prevent these attacks within a particular time period.
3. Cyber-attack/defense exercise: these exercises are conducted to assess and evaluate red and blue teams' performance at the same time. A white team creates an exercise environment in which active engagement between red and blue teams occurs to simultaneously attack and defend an exercise environment.

Based on our research findings (Yamin et al., 2018), we have identified that automation can reduce the time requirements for cyber-security exercises. For this, serious games could help (Hendrix et al., 2016)*to overcome the inefficiencies in cyber-security exercises*. The gamification of cyber-security exercises is a recent trend in which participants are divided into teams for achieving a specific objective like finding flags. The participants' strategies to solve the problems like *Capture The Flag (CTF)* in a cyber-security exercise scenario are very difficult to model because of the real-time decision-making of exercise participants. This makes the decision tree that is involved in such problem solving very complex. To address this, we propose a real-time cyber-security strategy game. Players will have the ability to play as an attacker or defender in a real-time multiplayer environment. Resources are assigned to attackers and defenders based on the scenario requirement, and their actions are recorded and observed by an observer. A detailed scenario creator is developed in which experts model the scenario in the game that can be transformed into an emulated environment. This results in a dynamic generation of attack and defense trees generated during the real-time cyber-security strategy game.

3. Related work

In the related work section, we provide a brief overview of serious games developed and used for cyber-security exercises and the methods for cyber-security scenario modeling.

3.1. *Games for cyber-security exercises*

In 2016, Hendrix et al. (2016) conducted a detailed survey of serious games for cyber-security education. They identified 15 games from industry and 14 games from academia that are actively being used for this purpose. Next, they categorized the games by their types like 2D point and clicked turn-based scenarios, 3D virtual world (sims style), and enterprise contingency planning. The games' target audiences comprises science curriculum students, children, and teenagers. The researchers stated that these games are used for training and education for short-term purposes only. For long-term training and education, scenario-based games are required. These scenario-based games represent unique case studies that can help in case-based learning.

In 2016, Alotaibi et al. (2016) conducted a review of serious games for cyber-security awareness. They identified 12 academic research articles and nine serious games being used for cyber-security education and awareness. These games had shown positive results in the evaluation of their effectiveness in cyber-security education and awareness. However, a large population set is needed in future research to better understand their impact. Moreover, the games currently being used deal with general cyber issues; there is a need to develop games that can be used in training specific scenarios.

In 2016, Schreuders and Butterfield (2016) conducted a two-year study on gamification for teaching and learning computer security in higher education. The study aimed to improve student engagement, increase student experience, and the content coverage of education material. They used freely available security educational games with in-house developed solutions for measuring students' progress with semi-

autonomous evaluations. The authors identified that games could be useful for initial motivation and student engagement; however, with time, students' motivation and engagement levels tend to decrease. In terms of increasing positive student experiences and content coverage, the study yielded positive results. The authors stated that gamification approaches work well when no extensive task-based assessment is involved in the education and training process.

In 2013, Amorim et al. (2013) proposed gamification as a new cyber-security education and training approach. They stated that the new approach should be a model-driven approach for the agile development of cyber-security exercises. The authors further stated that in terms of simulation and emulation, exercise execution depends on the training needs. The effectiveness of training exercises can be assessed with performance support systems. Their research was concluded by stating that cyber-security training requirements change with the technology. Therefore, new content and material for training exercises are continuously required, and model-driven agile development techniques can achieve this.

Adam Shostack (Ada, 2020) maintained an online list of table-top cyber-security games used for educational purposes. These games were mostly board and card games played between multiple players to learn different cyber-security concepts in a fun and engaging manner. As of June 2020, the list contained 28 games for security educational purposes, one game for teaching privacy principles, three non-game decks, and table-top games with some additional resources. These games did not require any software to play.

3.2. *Methods for cyber-security scenario modeling*

Cheung et al. (2003) presented CAML *(correlated attack modeling language)*, which uses a module of small attack steps to create a cyber-security attack scenario. The modules were designed to be very generic so that they could be used to model different cyber-attack scenarios. The researchers divided an attack model scenario into four parts: *vulnerability*, *exploit, attack step*, and *composite attack*. *Vulnerability* is the condition in the system or procedure that enables an adversary to perform actions that violate the security of the system and procedure, while *exploitation* is the process of exploiting a single vulnerability. *Attack steps* are the actions of the adversary for achieving specific goals, while *composite attack* combines multiple attack steps. The researchers' attack modeling methodology considered the attacker's goal and sub-goals, developing a relationship between attacker goals and the coresponding system events that can be observed to detect an attack.

Liu et al. (2005) presented AIOS *incentive-based modeling and inference of attacker intent, objectives, and strategies*. They integrated attacker intent regarding the cost of action to model the attackers' objectives. They also developed a game-theoretic AIOS formalization to capture the inter-dependencies between attacker intent, objective, strategies, and defender objective, along with the strategies to deduce AIOS automatically. They applied the developed AIOS on a real-world DDoS scenario to validate AIOS effectiveness in modeling attack and defense scenarios.

Marshall (2009) presented CyberSMART i.e.(*cyber scenario modeling and reporting tool*). They divided the cyber-security exercise into three tracks: *description and objectives*, *gamespace*, and *scenario*. *Description and objectives* define the scope of the exercise and what learning outcome is expected to be achieved. *Gamespace* defines the exercise environment and networking topology on which the exercises are planned to be executed. In comparison, the *scenario* defines the set of events expected to happen to achieve an objective. The researchers argued that there might be multiple objectives and sub-objectives in a cyber-security exercise, so there would be multiple scenarios to achieve those objectives. The authors proposed an event-based pyramid model for the representation of exercise objective and the corresponding scenarios.

Shiva et al. (2010) applied game theory concepts to dynamic cyber-security scenarios and considered the interaction between attackers and defenders in the cyber-security scenario as a game. The researchers suggested a model with rewards and punishments for the adversaries' actions. The model works by considering the *Nash equilibrium* as a key defining point for defender strategies. The defenders try to reach the *Nash equilibrium* to win against the attacker, while the attacker tries to avoid the *zero-sum* state in the game. The attacker receives a payoff if they can avoid a zero-sum state, and the whole game continues.

Russo et al. (2018) presented *scenario design and validation for next generation cyber ranges*, in which they proposed a model to design, validate, automatically generate, and test cyber-security scenarios. The researchers introduced scenario description language *SDL*, which is used to model the scenarios. The SDL has 10 elements: *system, firewall, policy, software, user, principal, vulnerability, file, invariant*, and *goal*. In the scenario, *principals* represent the subjects operating in the system while a *goal* is the objective of the *principals*. The researchers executed the SDL on a cloud orchestration platform for scenario deployment and validation.

4. Research methodology

Numerous different research methods were employed in the current work, including serious game development methodology, ontology development methods, and model-driven engineering methodology. The last one includes the development of the DSL and its compiler. Furthermore, various quantitative and qualitative assessment methods for evaluation were used. First, for the development of serious games, we used the framework in cyber-security proposed by Le Compte et al. (2015). The framework provides a precise methodology for conducting research related to serious games in cyber-security. The framework has six steps for the development life cycle and evaluation of serious cyber-security games: (1) Preliminary analysis in which the available resources for game development are evaluated, pedagogical objectives are defined, the target audience is identified, and the game mechanics are defined based on the pedagogical objectives. (2) The design phase is responsible for the game's conceptual modeling, ensuring that the game objectives are well conveyed to the players. (3) The development phase aims at developing the game based on the resources and objectives identified before. (4) Game assessment evaluates the game, in which a test group of a target audience can be used in the

game assessment process, and the feedback will then be used to improve the game mechanics. (5) The deployment phase is the one responsible for deploying the game for real-world assessment and training. The final phase is the (6) the player assessment phase, in which the game's effectiveness in achieving its pedagogic objectives is measured. This can be achieved through tests, surveys, and questionnaires given to game players.

Second, to model the various concepts present in the cyber-security exercise scenarios, we carefully analyzed the cyber-security exercise domain and developed an ontology (Maines et al., 2015). This ontology highlighted various abstract concepts related to cyber-security exercises that must be incorporated in the DSL; these are presented in Fig. 10. For the development of the DSL, we employed model-driven engineering (Schmidt, 2006) techniques. These techniques are used to develop the scenario language and its syntax and then to develop a compiler for the language that will process an instance of the scenario language and generate various usable artifacts, such as *HEAT* (hea, 2019) and *Puppet* (Pup, 2020) templates, which can be used to generate the exercise infrastructure.

For the verification and assessment of our developed artifacts, we employed both quantitative and qualitative evaluation methods. We created a cyber-security exercise scenario based on a real penetration testing activity, along with using pre- and post-exercise survey methods (Yamin et al., 2018) to quantify and measure the skill improvement of the exercise participants. For the qualitative evaluation, we used expert feedback against a set of four predefined evaluation matrices: *efficiency*, *usability*, *completeness*, and *flexibility* that we identified from the literature (Yamin et al., 2019).

5. Proposed system

As we have argued, the current way of conducting cyber-security exercises is not efficient; therefore, we are proposing a system that addresses one of the most time-consuming parts of the cyber-security exercise life cycle (Yamin and Katt, 2018): the preparation of an exercise scenario. Furthermore, a dry run is partially covered as well. In the preparation of a scenario, a ***White Team*** creates the environment in which the ***Red Team*** and the ***Blue Team*** practice their attack and defense skills. We identified the major cyber-security scenario definition techniques (Yamin et al., 2019) in which a scenario definition language is used for the orchestration of the cyber-security exercises infrastructure. Our proposed system utilizes the concept of *SDL*; however, we are proposing a fundamentally new way of creating and deploying a scenario. Our proposed system has three primary parts:

1. *Cyber security strategy game*
   The game is used to model the network topology for a cyber-security exercise scenario. The games provide an interface for presenting high-level scenario requirements and transforming them into low-level technical requirements. The game is basically designed to facilitate the process of cyber-security exercise scenario modeling and validation in a simulated environment by incorporating the roles of the Red and Blue Teams. The game provides an opportunity for the scenario designer to develop and test hundreds of cyber-security scenarios before deploying them in a realistic, emulated environment.
2. *Domain specific language*
   The DSL is used to represent the low-level technical details present in the cyber-security exercise scenario. The *cyber security strategy game* saves an exercise scenario as an instance of the DSL in the form of a YAML Ben-Kiki et al. (2005) file. The DSL is designed to accommodate 11 key concepts related to cyber-security exercises. It can be used to implement three types of cyber-security exercises, which are presented in Section 7.
3. *Infrastructure orchestration module*
   The *infrastructure orchestration module* is a compiler that takes the DSL and performs syntax validation. If the code has no errors, then it generates the infrastructure, as described in the DSL. The infrastructures generated in the form of **HEAT** (hea, 2019) and **Puppet** (Pup, 2020) stack and deploy them on the open stack cloud environment. The technical details of the *infrastructure orchestration module* are presented in the corresponding section.

A schematic representation of the proposed system and its layers of abstraction are presented in Fig. 1:

6. Cyber-security strategy game

As discussed in Section 2, we identified the inefficiencies in cyber-security exercise development (Yamin and Katt, 2018). We also identified that automation could help in reducing these inefficiencies (Yamin et al., 2018). As a first step toward this automation, we hypothesized that serious gamification would help (Yamin and Katt, 2019b) in removing the identified inefficiencies. To validate our hypothesis, we conducted a survey during NCSC (Norwegian Cyber-Security Challenge) 2019 (sta, 2019). The test subjects consisted of 25 participants who qualified for the initial CTF round at the NCSC, in which around 150 people participated. In the survey, we asked questions about serious games for cyber-security education, evaluated our developed game, and assessed players' skill sets, the details of which are given in subsequent sections.

6.1. *Preliminary analysis*

The game was developed as a proof of concept by three bachelor students during their final year project at NTNU (Norwegian University of Science and Technology) (git, 2020). The game pedagogic objectives are to achieve the following:

1. Increase player awareness of how cyber-attacks and defenses are conducted.
2. Provide an understating for strategizing cyber-attacks and defenses.
3. Provide an understating for decision making at the operational cyber-security level.

This was achieved by incorporating the concepts of penetration testing methodology (Allen et al., 2014) and the cyber

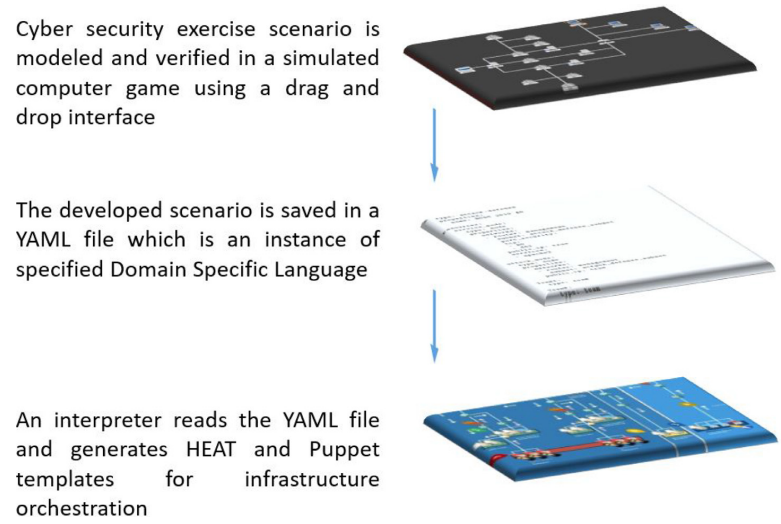


Fig. 1 – Proposed system parts and corresponding layers of abstraction.

kill chain (Hutchins et al., 2011; Yadav and Rao, 2015). These concepts included a total of 16 skills developed after an analysis of a common curriculum being taught at US-DODapproved certification programs (Yamin and Katt, 2019a). The skill set concepts included in the game are as follows:

- Network and system security
- Information security and management
- Cyber-security incidents and response
- Risk analysis and management
- Forensics and cryptography
- Windows and Cisco device security
- Application and web security
- Security concepts and controls

To apply the cyber-security skill set in a realistic environment, a methodological approach of the cyber kill chain was used, incorporating both the attackers' and defenders' actions; the details are specifically given in Section 6.2.1. Currently, the game integrates most of the stated skill set; however, a very specific skill set related to Windows and Cisco device security still requires additional work. We planned to use the game for cyber-security education; therefore, we set the target audience age group between 16 and 25 years old. A sample of 25 top-ranking individuals selected out of 150 participants of NCSC 2019 qualifiers participated in the survey; this target audience group was selected based on the target audience of the European cyber-security challenge (ecs, 2019). We considered the sample group as a reliable indicator for such research in the Norwegian context. The survey questions were straightforward, neutral, and easy to understand. Besides *Yes* and *No* answers, the participants were given the option *Maybe* if they were not sure. One of the questions related to computer games in general, and the other two tackled attack and defense scenarios separately. The wording was carefully chosen based on the background of the participant (highly technical). Finally, the survey was administered, and the participants responded to the questions in a relaxed environment to avoid any biases. Below are the findings of our survey about serious games in cyber-security exercises.

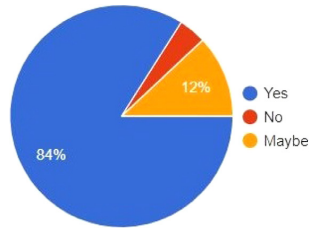


Fig. 2 – Percentage of the participants who thought computer games could help in cyber-security education.

1. Do you think computer games can help in cyber-security education?
   The first question was a general question about the role of computer games in cyber-security exercises. Here, 84% of participants considered that they could play an important role, 12% were not sure about the role of computer games in cyber-security exercises, and only 4% did not consider them useful. The survey findings are presented in Fig. 2.
2. Do you think practicing attack strategies in games is useful before launching a real attack?
   The second question was related to cyber-attack strategies. The purpose was to identify whether it is a good approach to practice a simulated attack strategy before launching a

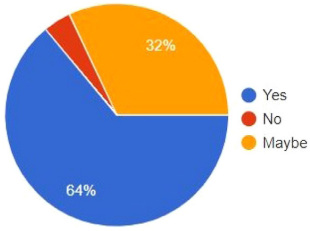


Fig. 3 – Percentage of the participants who thought practicing attack strategies in games is useful before launching a real attack.

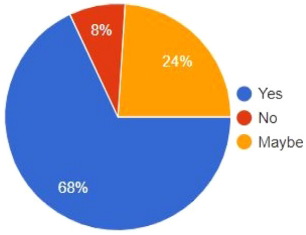


Fig. 4 – Percentage of the participants who thought practicing defense strategies in games is useful before defending against a real attack.

real attack on actual infrastructure. Here, 64% of the survey participants considered this a useful approach, 32% were not sure, and 4% did not find this approach useful. The survey findings are shown in Fig. 3.

3. Do you think practicing defense strategies in games is useful before defending against a real attack?
   The third question was related to cyber defense strategies. The purpose was to identify whether it is a good approach to practice a simulated defense strategy before defending against a real attack on actual infrastructure. Here, 68% of the survey participants considered this a useful approach, 24% were not sure, and 8% did not find this approach useful. The survey findings are shown in Fig. 4.

In *Question 1*, 84% of the survey participants stated that the game could help in cyber-security education, while in *Question 2* and *Question 3*, 64% and 68% stated that it could help in devising attack and defense strategies, respectively. Here, we observed a slight deviation of the survey participants' feedback. We believe that the majority of the survey participants considered that such games are good for cyber-security education in general. However, the survey participants had their own experiences and skill sets regarding operational strategies, which we believe caused the deviation. For instance, most of the survey participants were good at devising attack strategies; therefore, they believed the game would help improve their competence in devising defense strategies and rated it a bit higher.

Regarding the game mechanics, multiple cyber-security strategy games already exist (Hendrix et al., 2016); they are mostly turn-based strategy games. However, because of the dynamic and complex nature of cyber-security exercises, a turn-based strategy is not beneficial for developing cognitive flexibility. Therefore, we decided to develop a real-time strategy game (Glass et al., 2013) to accommodate cyber-security concepts such as penetration testing methodology and cyber kill chain in a multiplayer environment.

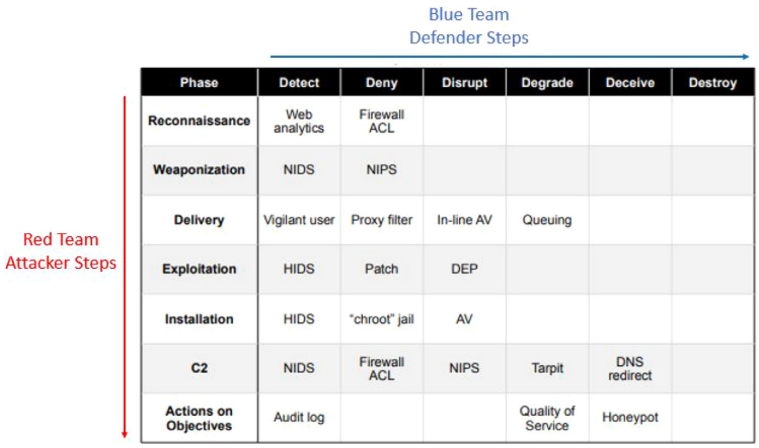
6.2. *Design*

6.2.1. *Integrating cyber-security in a serious game*

- Cyber-security exercise scenario modeling
  Cyber-security exercise scenarios are quite dynamic, and modeling the scenarios based on specific events (Marshall, 2009) is not useful in a multiplayer environment. Adversary player actions can change planned scenario events. Therefore, we opted for a no-win condition in the cyber-security strategy game model. The game players are given the objective to attack or defend a system within a specific time interval. The penetration level assesses players' performance during the attack or the number of attacks stopped during the defense. For the attack and defense steps, we used *Lockheed Martin's* course of action matrix (Hutchins et al., 2011), which is widely accepted in the academic and industrial communities; this matrix is presented in Fig. 5.
- Penetration testing methodology
  We incorporated concepts from penetration testing execution standards in the game design (PTE, 2020) for the attackers. These concepts deal with reconnaissance and information gathering about systems by using active and passive measures. Then, the gathered information is used for the identification and discovery of vulnerabilities. After this, the discovered vulnerabilities are used for the exploitation and post-exploitation of the systems. Additionally, performing an analysis of the exploited vulnerabilities and sharing the findings in a report is also incorporated.
- Cyber kill chain
  For the defenders, we incorporated the concepts from the cyber kill chain (Yadav and Rao, 2015). Cyber kill chain concepts are used to stop the attacker during different phases of attacks, such as during discovery, weaponization, exploitation, and so forth. The defenders have to prioritize the security of the assets at risk and assets protected by other security controls like firewalls, IPS (intrusion prevention systems), and so forth.

6.2.2. *Actors and functionalities*

- White Team
  For White Team members, a scenario modeling interface is proposed in which a White Team member can create a complete network topology. The network topology can contain interconnected components such as APIs, web servers, computers, firewalls, IPS, and so forth. These components have a security level that can be defined as affecting the attack and defense cost within the game. New se-



Phase	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Web analytics	Firewall ACL				
Weaponization	NIDS	NIPS				
Delivery	Vigilant user	Proxy filter	In-line AV	Queuing		
Exploitation	HIDS	Patch	DEP			
Installation	HIDS	"chroot" jail	AV			
C2	NIDS	Firewall ACL	NIPS	Tarpit	DNS redirect	
Actions on Objectives	Audit log			Quality of Service	Honeypot	

Fig. 5 – Cyber kill chain course of action matrix for attackers and defenders (Hutchins et al., 2011).

curity vulnerabilities (owa, 2020) can be injected in these components according to the need of the scenario. White Team members can also introduce cyber asymmetry by setting system vulnerabilities and exploitation levels from low to high, depending on the exercise objectives. For observing the game, an interface for White Team members is also proposed to observe the Red and Blue Teams' gameplay and progress in real time.

- Red Team
  For Red Team members, an interface is proposed to access different penetration testing methods, such as discovery, probing, and exploitation. They also have a list of known exploits that can be used if they found a vulnerable system. However, all the systems are not vulnerable to known exploits, so they have a panel for researching new exploits related to those systems.
- Blue Team
  For Blue Team members, an interface is proposed to have full visibility of the network topology in the scenario. They have to identify whether the systems are up to date with no vulnerabilities; if they find a vulnerability, they can patch it. The defenders have the option to place security controls like firewalls and IPS within the topology to secure the systems further.
- Green Team
  For Green Team members, an interface is proposed to have full visibility of the network topology in the scenario where there are live-action representations of Red and Blue Teams at the same time. This interface provides the capability to monitor the team's performance and engage spectators in the game.
- Game economy
  Every action in the game has a cost; the cost is determined by the White Team members who planned the scenario. Red and Blue Team members have to make operational cyber-security strategy decisions to achieve their objectives while keeping the cost of their actions in mind. Moreover, time plays an important role during the gameplay because the game is intended to be completed in a specific amount of time, so the game players must make quick decisions.

6.3. *Development*

The game took nearly five months from its initial planning to complete development. The game was developed using Unity 3D (Unity Technologies, 2020), a standard game development engine. The game is called *Red vs Blue, Cyber-security Simulator*. We made the game open source so that anybody can make changes to the in-game functionality per their requirements; the game source code is available at GitLab (git, 2020). The game's most important component is a dynamic cyber-security exercise scenario creator, which provides drag-and-drop functionality of different IT infrastructure objects to create a network topology. The developed topology is saved in a YAML file in the form of a scenario model. We used the developed models to deploy an emulated cyber-security exercise infrastructure. The developed real-time cyber-security strategy game is presented in Fig. 6. The technical details of the game functionality are discussed next.

6.3.1. *Program flow*

The game has two main parts: first is a scenario creator, which was developed to help *White Team* members in designing cyber-security exercise scenarios. A new scenario can be created, or old scenarios can be edited from a YAML file in the scenario creator. The second part involves the gameplay in which *Red* and *White* Team members play the developed scenario. The Red Team members can attack, exploit, probe, and analyze the system's components present within the scenario environment, while Blue Team members can probe, analyze, and defend the system components. There is a third part of the game in which the whole gameplay of Red and Blue Team members can be monitored; this is for the *Green Team* mem-

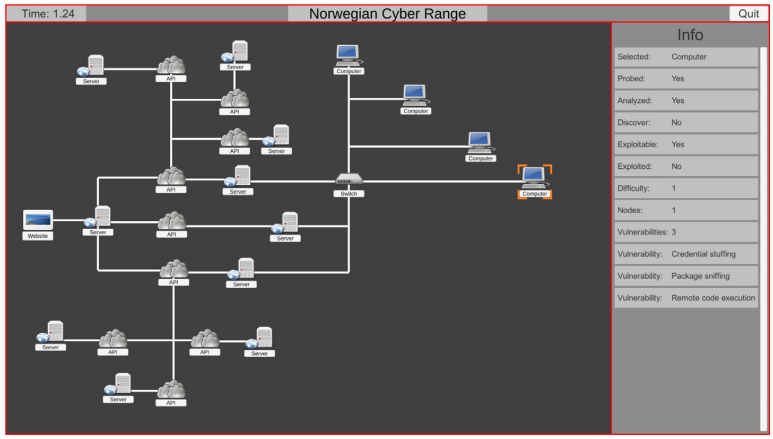


Fig. 6 – Developed real-time cyber-security strategy game.

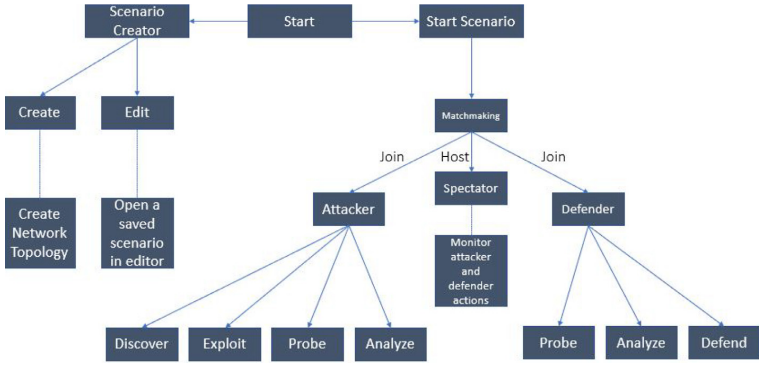


Fig. 7 – Game program flow.

bers. The game program flow with its major components, is presented in Fig. 7.

6.3.2. *Scenario creator*

The scenario creators provide the *White Team* members with three functionalities:

- *System components*
  System components comprise computers, router, switches, APIs, and other infrastructure-related components that can be dragged and dropped on a 2D plane. The components are configurable in such a way that the vulnerabilities or defenses associated with them can be defined.
- *Component connections*
  The component connection allows the *White Team* members to define the inter-connectivity between the different system components. This inter-connectivity helps design wide ranges of cyber-security scenarios because system components can be the same for multiple scenarios. However, the network topology can change the way the attackers and defenders play the scenario.
- *Component menus*
  Component menus allow the *White Team* members to configure the component with vulnerabilities and defenses. Then, they can configure the component level of exploitation by adding high-risk vulnerabilities in it, or they can set the component with no vulnerabilities at all. It all depends on the scenario requirement and complexity. Fig. 8 represents the component menu, as seen by White Team members.

6.3.3. *Attack and defense game play*

The game offers simulated attack and defense gameplay in which attackers can discover, exploit, probe, and analyze different system components, while defenders can probe, analyze, and defend different system components. Attackers and defenders have realistic options available at their disposal to make choices like scanning a network for an attacker or patching the system and placing a firewall in front of a vulnerable

Fig. 8 – Component configuration menu.

component for a defender. These attack and defense choices have a cost that is pre-assigned by the *White Team* members. There is a reward system for successful exploitation and defenses, which opens other options to attackers and defenders, like scanning for 0day vulnerabilities for the attacker and setting up SIEM (security information and event management) solutions for defenders during the gameplay. For a specific example, consider a machine present in a network that has an RCE (remote code execution vulnerability). If the attacker can identify the vulnerability, then the attacker can exploit the vulnerability with a cost of 5. The defender has two options here: (1) patch the vulnerability, which may have a cost of 2 or (2) to place a firewall in front of the vulnerable machine, which may have a cost of 10. There may be multiple machines with the same vulnerabilities to make things complex, and patching all those machines might not be the ideal solution. So the defender has to identify the network paths from where an attacker can exploit such vulnerabilities and place the appropriate defenses.

6.3.4. *Networking and logging*

The game is implemented as a client-server architecture, in which one instance can host a game, while multiple instances of attackers and defenders can join the game. When a game is hosted at an instance, it acts as a server and starts to listen for a TCP connection. When a client wants to join the server, it needs to send a request to a server, and the server assigns the client a game lobby in which the game is hosted. When a client joins the lobby, it obtains access to a messaging server in which different team members can communicate in a textual format. The actions performed by the attackers and defenders and their communications are logged for a post-exercise evaluation about what can be done or what went wrong for a team. The logs are visible to cyber-security exercise observers and are presented in Fig. 9.

Fig. 9 – Event logs collected during a cyber-security exercise.

7. Domain-specific language

In connection with the strategic game, we developed a DSL for specifying and generating cyber-security scenarios as a part of *Norwegian Cyber Range* research activities; this was done with the collaboration of one of our master's students, Dunfjeld (2019). The scenario language, together with its interpreter, are publicly available on Github (Git, 2020). DSLs are programming languages used to solve problems in a very specific domain compared with *general purpose programming languages*, which are used to address problems in a wide area of domains. A DSL provides a layer of abstraction to the user that closely matches the domain-specific problem description and removes the unnecessary overheads of setting up frameworks and writing application-specific technical code. In our proposed DSL, we have identified 11 key concepts required to model a cyber-security exercise scenario, which are presented in the DSL ontology in Fig. 10. A *scenario* has *objectives*, such as capture or defend a flag. To achieve the *objectives*, a *scenario* includes *teams*, *challenges* and *phases*, all of

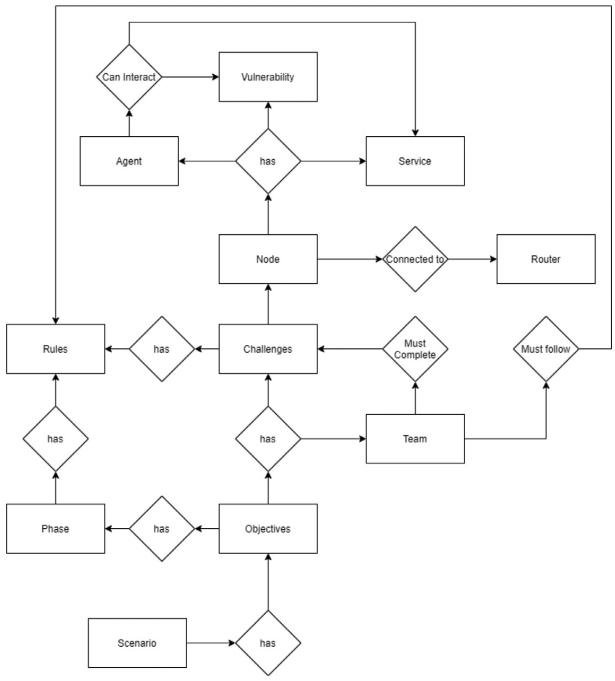


Fig. 10 – An ontology of the main concepts of a cyber-security exercise scenario.

which have *rules*. Teams can be attackers or defenders trying to pass the challenges presented in the scenario. Challenges includes attacking a vulnerable system or patching a vulnerability. These actions are performed during different phases of the exercise, such as the start or middle phase. The challenges are presented on the *node*, which has *vulnerabilities*, *services* and *agents*. The nodes are connected to the *router*, which is connected to the internet for providing access to the exercise platform. For the language, we defined both the abstract syntax and concrete syntax. Additionally, the language compiler/interpreter has been defined and developed (Voelter et al., 2013). The details are presented below.

7.1. *Abstract syntax*

The abstract syntax of the DSL is used to represent the different concepts present within the domain; the identified concepts are as follows:

1. *Scenario properties*
   There are multiple types of cyber-security exercises scenarios; the scenario concept in the DSL is used to define the main properties of a scenario, which are as follows:
   - *Name:* A string value that is used to define the name of the competition or the event within which the scenario will be executed, such as *Defcon CTF*.
   - *Type:* A string value used to define scenario types, such as *jeopardy* and *attack-defense*.
   - *Start date:* A date value that indicates the scenario start date in the format of dd.mm.yyyy.
   - *End date:* A date value that indicates the scenario end date in the format of dd.mm.yyyy.
   - *Start time:* A time value that indicates the start time of the scenario in the 24 h format of hh:mm.
   - *End time:* A time value that indicates the end time of the scenario in the 24 h format of hh:mm.
   - *Docker hosts:* An integer value that indicates the number of docker hosts that are going to run virtual machines in the scenario.
   - *Objectives:* A list of all objectives of the scenario, which will be explained later in more detail.
   - *Agents:* A list of all agents that are active during the scenario, which will be explained later in more detail.
   - *Rules:* A list of all rules that need to be followed in the scenario, which will be explained later in more detail.
   - *Teams:* A list of all teams participating in the scenario, which will be explained later in more detail.

   Scenario properties that have list values like *objectives*, *agents*, *rules*, and *teams* do not contain the definition of the concepts; they just refer to the objects that have the concept definition.
2. *Node*
   The concept of a *node* is used to define a virtual machine that is present in the scenario. It has the following properties and sub-properties:

- *Type:* A string value that is used to define the type of the node.
- *Flavor:* Flavor is used to allocate the amount of RAM, CPU, and storage in the cloud. It is a string value, and if it is not implemented, the default settings are used.
- *OS:* A string value that is used to specify the operating system for a virtual machine.
- *Public IP:* A Boolean value that is used to assign a public IP address to the VM. By default, a VM is not publicly accessible, so this property is used to assign a floating IP address to the VM.
- *Networks:* A list value that is used to represent the connection of nodes in a network topology. It should at least have two of the following sub-proprieties:
  - *Router:* It is the name of the router with which nodes are connected.
  - *Subnet:* A string value is used to indicate the subnets in which nodes are connected.
  - *Port security:* One or many TCP or UDP property values used to represent the open ports on the virtual machine and respective services. By default, only SSH and ICMP ports and services are open for management and diagnostic purposes.
- *Vulnerabilities:* A list value that indicates the vulnerable application and services that need to be installed on a node. The detailed properties and sub-concepts of the vulnerabilities will be explained later.
- *Services:* A list value that indicates the services that needs to be installed on a node.
- *User accounts* A list value that contains the user account details that are present on a node. It has the following sub-properties:
  - *Username:* A string value that indicates the username of the user.
  - *Name:* A string value that indicates the user's full name.
  - *Password:* A string value that indicates the user password in a hash form.
  - *Uid:* A string value that indicates the user's identifier.
  - *Gid:* A string value that indicates the user's primary group ID.
  - *Group:* A string value that can be used to override the user's primary group value.
  - *Groups:* A list value that contains the groups' names in which the user is present.
  - *Home:* A string value that indicates the user's home folder.
  - *ssh key:* A list value that contains the ssh key, which is used to access the user's account.
  - *Shell:* A string value that indicates the user shell's address.

3. *Router*
   A router is used to provide the necessary networking functionality to different nodes. It has the following properties:
   - *Type:* A string value used to define the type of the router.
   - *Network:* The network property contains the information of all the subnets connected to the router. The subnets have their own properties, which are as follows:
     - *CIDR:* A string that indicates the IP range of the subnet.
     - *Gateway IP:* A string that indicates the gateway IP address of the subnet.
     - *Routes:* A list of strings that contains the information of routes between different subnets.
4. *Service:* Services are used to define the applications that are running on the nodes and that are not vulnerable and are used to make the scenario more realistic.
   - *Type:* A string value used to define a service that contains the information about the service that is needed to be connected to a node.
5. *Vulnerability*
   A vulnerability is a component of a node, for example, an application or a service installed in a node, that is intentionally vulnerable or contains an implementation bug or design flaw.
   - *Type:* A string value used to define the vulnerability type such as *DoS*, *RCE*, *XSS*, and so forth.
6. *Challenge*
   A challenge concept is used to represent an exercise or a task that needs to be completed to earn points in a cyber-security exercise. It has the following properties:
   - *Type:* A string value used to define the type of a challenge.
   - *Points:* An integer value that represents the maximum number of points awarded after completing a challenge.
   - *Port:* An integer value that indicates the port number through which the challenge is accessed.
   - *Prerequisites:* A list of strings indicates some other challenges that need to be completed before accessing the current challenge.
7. *Team*
   The team concept is used to identify the participants' role in a cyber-security exercise. It is also used for point allocation. There can be multiple Red or Blue Teams present in a cyber-security exercise.
   - *Type:* A string value used to define the team type, that is, Red Team or Blue Team.
   - *Members:* A list value that contains the contact information of each member of the team.
8. *Agent*
   Agents are used for performing specific tasks in a cyber-security exercise in an automatic manner. They can be used to generate traffic or launch autonomous attacks.
   - *Type:* A string value that is used to define the type of the agent like *Traffic generator*, *Attacker*, and so forth.
   - *Sub type:* A property is used to define the sub-category of an agent. The agents can be used for traffic generation, user behavior simulation, and so forth.
9. *Phase*
   A scenario can be broken down into multiple phases, for example, vulnerability discovery, vulnerability exploitation, and so forth. Transitioning from one phase to another results in a possible change of objectives and rules defined in this concept.
   - *Type:* A string value used to define the type of a phase like *start*, *middle*, and *final*.
   - *Objectives:* A list value that contains scenario objectives in textual format with respect to a particular phase.
   - *Rules:* A list value that contains scenario rules in textual format with respect to a particular phase.

```
scenario:
type: jeopardy
properties:
    name: Fancy-name-of-CTF
    start_date: 01.01.1970
    end_date: 07.01.1970
    start_time: 12:00
    end_time: 23:59
    docker_hosts: 3

resources:
    shellshock:
      type: challenge
      properties:
        port: 1338
        points: 50

    heartbleed:
      type: challenge
      properties:
        port: 1337
        points: 30
```

Fig. 11 – Jeopardy-style CTF generation sample code.

- *Agent*:: A list of agents that are phase specific

10. *Objectives*
    A description of scenario objectives that must be completed to successfully complete a scenario. A scenario may have single or multiple objectives depending on the complexity of the scenario.
    - *Text:* A string value that indicates scenario objectives in textual format.
11. *Rules*
    Rules contain information for teams in the scenario, such as *"DOS on the nodes is not allowed"*.
    - *Type:* A string value used to define the type of a rule like *allowed* or *not allowed*.
    - *Text:* A list value that contains scenario rules in textual format.

7.2. Concrete syntax

The concrete syntax is used to create a scenario instance. It can be generated by the real-time cyber-security strategy game and presented in the previous sections, or a user can write it directly with the help of an interactive interface. YAML specification is used for the specification of the concrete syntax of our scenario language. It provides the necessary indentation, helping in creating hierarchical structures and representation of the data in lists, keys, or a combination thereof. All concepts of our language are defined in the form of objects, and the structure for representing any concept in object format is identical. Below is an example of how a concept is represented in an object form in YAML.

```
identifier:
type: concept-type
properties:
    some property: some value
```

The concepts are identified by the object identifier. The object identifier is used as a reference to that object. Each object needs to have a mandatory property *type*, which is used to specify the object type. Each object has its properties assigned by a property identifier. Fig. 11 presents a sample of the concrete syntax used to generate a simple jeopardy-style cyber-security exercise containing three docker hosts and two vulnerabilities. Here, not all elements need to be present in the scenario. For those elements that are not mentioned, default values will be assigned, for example, routers and networks.

7.3. Compiler/Interpreter

Five steps are involved in the compilation of a DSL scenario instance:

1. *Loading*
   The scenario file is loaded into the compiler using the python library oyaml. *oyaml* preserves the dictionary ordering of the file when loading the scenario YAML file.
2. *Syntax validation*
   Syntax validation is performed, and whether the loaded files contain the scenario information according to YAML specification or not is checked. If the file does not follow the YAML specification, the syntax validation process fails, and compilations stop.

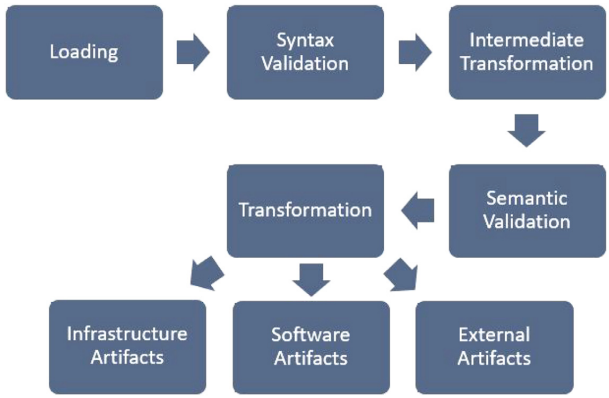


Fig. 12 – Compilation process of DSL.

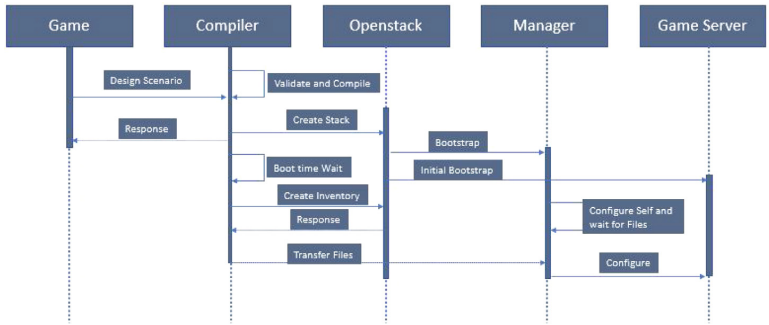


Fig. 13 – Infrastructure orchestration process from a simulated game to an emulated environment.

3. *Intermediate transformation*
   The YAML file data are transformed into a Python dictionarytype data structure. This helps access data and apply compilation logic in Python. One reason to choose YAML as a scenario specification language is its ability to be easily transformed into a Python data structure.
4. *Semantic validation*
   Although syntax validation can ensure that the scenario syntax is correct, semantic errors can still exist. To avoid semantic errors, multiple semantic validation types were applied, as follows:
   - Verification of the input flow structure.
   - Verification of objects that are associated with a particular scenario type.
   - Verification of mandatory properties in objects and their values according to the required data formats.
   - Verification of optional properties in objects and their values according to required data formats.
   - Verification of assigned OS/services/vulnerabilities in the objects present in the compiler database.
   - Verification of assigned IP addresses and that they are in the correct subnet.
5. *Transformation*
   After semantic validation, the data structures are transformed into three artifacts that are usable for low-level platform-specific technology. In our case, we use OpenStack as the cloud platform technology, which will host the final exercise infrastructure. Thus, the end result of the transformation is a set of HEAT templates that can be used to deploy the network and the virtual machines on our OpenStack-based private cloud. The details of the three artifacts are as follows:
   - *Infrastructure artifacts*
     These are artifacts that are used to build a networking component and the virtual machines present in a cyber-security exercise scenario; they are HEAT templates that are used to deploy the exercise infrastructure on OpenStack.
   - *Software artifacts*
     These are artifacts that are used for the installation and configuration of operating systems and services. They are transformed into Ansible (Ope, 2019) templates. Virtual machines have their separate configuration templates that define their settings according to the scenario definition.

- *External artifacts*
  The artifacts that are not related to software and infrastructure artifacts are presented as external artifacts. They contain rules and objectives of the scenario, which are merely textual information related to the cyber-security exercise participants' scenarios.

The compilation process of our DSL is presented in Fig. 12.

8. Infrastructure orchestration module

After a successful compilation of the scenario, the compiler also performs the provisioning process for deploying the cyber-security exercise infrastructure. Because of this, compilation and provisioning become a one-click process. Fig. 13 shows how different components of the proposed systems interact with each other for the full orchestration of the deployment of the cyber-security exercise infrastructure. This is a multi-step process in which, first, a *White Team* member creates the scenario topology within the cyber-security strategy game. According to the DSL specification, the scenario is saved in a YAML file, which is then validated and compiled by the compiler. If the compilation process is successful, the aforementioned artifacts, including the HEAT templates, are generated, and a request is generated to the OpenStack orchestration API to create a stack based on the generated HEAT templates. For the configuration of VMs, the Cloud-init option of OpenStack is used, which initiates basic bootstrapping of the VMs, such as installing Ansible and transferring SSH keys.

When nodes are created that do not have an IP address, DHCP is used to allocate the IP addresses to the nodes. The compiler makes a query to OpenStack and requests the IP addresses of the nodes that were created. A waiting period is added into the compiler to ensure all the nodes are set up and have acquired an IP address. The compiler then updates the list of IP addresses and uses SSH to transfer the required configuration of the nodes, that is, the compiler-generated Ansible templates.

Ansible is a push-based configuration setup utility, meanings that for configuring a VM of the cyber-security exercise scenario, an additional VM is needed to push the scenario configuration from within the scenario network. A manager node is created, which receives the configuration from the compiler. After the Ansible files are received from the compiler, a manager node starts pushing the configuration of all the network-related functions and VM-related services and vulnerabilities based on the scenario requirements specified by the scenario's DSL instance.

8.1. *Nature of emulation*

The toolset produces emulation for a cloud native environment, which is currently OpenStack based. The emulation supports *network*, *transport*, *session*, *presentation*, and *application* layer protocols. However, the *datalink* and *physical* layer protocols were not supported because of the inherent limitations of cloud-native software networking (Ope, 2021). The toolset can create and deploy small and large exercise environments based on the scenario requirements. These exercise environments can be configured to be vulnerable using Ansible. These environments can support a variety of exercises such as jeopardy-style CTF, attack/defense, Red/Blue teaming, and so forth. A generated exercise environment using the developed toolset is presented in Fig. 14.

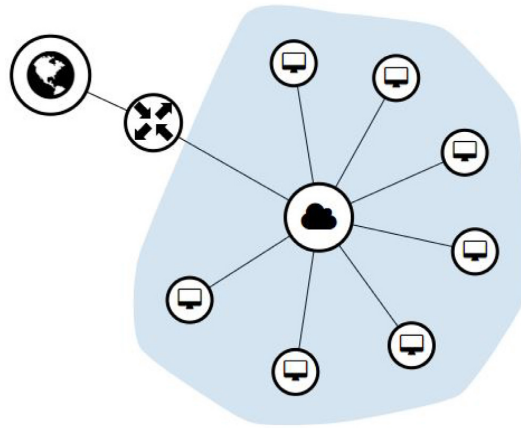


Fig. 14 – Emulated environment produced using the developed toolset.

9. Game and players assessment

To evaluate the whole toolset developed in this research work, we conducted a case study in the context of the NCSC in 2019; a summary of the results of the study are presented in Section 9.5. The NCSC is used for selecting, evaluating, and training the Norwegian team that will participate in the European Cyber-Security Challenge (ECSC). Our research team was part of this process, and the field study was conducted in this context. The goal was to evaluate both the developed game and the scenario language toolset during one of the two qualification rounds of NCSC 2019.

9.1. *Number of participants and demographic data*

The test subjects consisted of 25 participants, 20 male and five females, who qualified for the initial CTF round at NCSC 2019, in which more than 150 people participated from all over Norway. All the survey participants were ethnic Norwegians between the ages of 16 and 25.

9.2. *Task performed by the participants*

The participants were given a brief tutorial about the game and how it works. The participants were seasoned CTF players and had expertise in offense techniques. Therefore, a Red Team game was chosen. Each participant was tasked to play the game individually as an attacker for 20 min in an isolated environment without any external interference. They were

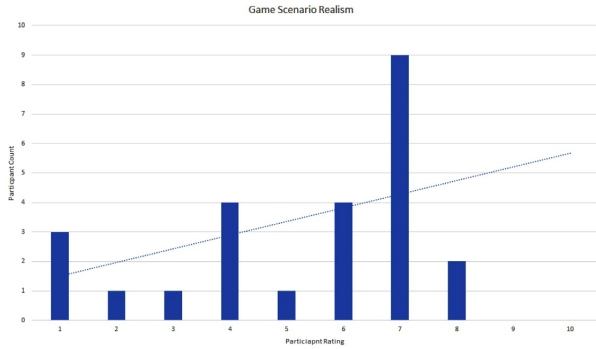


Fig. 15 – Game scenario realism rating.

asked to critically analyze the game because they were expected to provide feedback after the game session. The participants attacked a network with no active opposition but with limited resources. Their decision-making process for attacking the network was evaluated.

9.3. *Data collection*

The data for the study were collected in three ways:

- Post-game session survey
  A survey was conducted after the game session in which the participants were asked six questions, of which four questions were related to game realism and usability and two to the players' assessment of the improvement of their skills, which are reported in the summary of results.
- Game recordings
  The game can act as an observer, so the gameplay can be remotely observed. This functionality was used to record the participant gameplay for participant evaluation in making real-time strategy decisions.
- Post-game session interview
  An expert from CYFOR (Norwegian Cyber Force) (Cyb, 2020) conducted post-game session interviews with the participants. The interviews were used for the psychometric analysis of the participants to assess their cognitive abilities.

9.4. *Data analysis*

Data from the surveys were analyzed using a simple statistical method of *trend line* (Tre, 2020), which is a line that can be drawn on a scatter diagram to represent a trend in the data. In our study, the trend line is presented in histogram charts in the summary of the results. The game recordings and interviews were used for the evaluation of individual player performance during NCSC 2019. In the interviews, the participants were asked to self-reflect on their experience. The details of the individual participants' cognitive performance evaluation processes are out of the scope of this work and will be presented in a study dedicated to this topic. The cyber defense retrospective timeline analysis (Knox et al., 2019) was used for the qualitative evaluation of the individual participants.

9.5. *Summary of the results*

9.5.1. *Game assessment:*

In our field study, we developed a scenario related to an organization's internal network exploitation. The scenario was based on real cyber-security incidents that involved a private organization. The organization had an internet-facing website that was connected to multiple APIs. The website itself was not vulnerable, but one of the deployed APIs was vulnerable to RCE (remote code execution vulnerability). The attacker could exploit the vulnerability and ingress into an internal network with multiple subnets. After that, the attacker had to identify important subnet and resources based on the retrieved information from the network interfaces before penetrating into the important subnet to achieve full network exploitation. On the defender side, the defenders had to patch the vulnerable systems and make strategies to secure the important network subnets with limited resources. We developed questions related to scenario realism in the game and the overall game usability, asking the participants to rate the game from 1 to 10, where 1 was the lowest and 10 the highest value. To ensure correct and sound answers by all participants, they received a short training session on the questionnaires included in the study and the meaning of the scales used before the study. The findings of the survey are as follows:

1. How realistic is the current game in representing cyber-security exercise scenarios?
   Most of the survey participants considered that the representation of cyber-security exercise scenarios was realistic in the game. Here, 15 out of the 25 participants rated the game realism as more than 5, out of which two rated it 8, nine rated it 7, and four rated it 4, as shown in Fig. 15.
2. How realistic is the current game in devising cyber attack strategies?
   The majority of the survey participants considered that the game was realistically devising cyber-attack strategies. Here, 14 out of the 25 participants rated the game realism at more than 5, out of which one rated it 9, two rated it 8, six rated it 7, and five rated it 6, as shown in Fig. 16.
3. How realistic is the current game in devising cyber defense strategies?

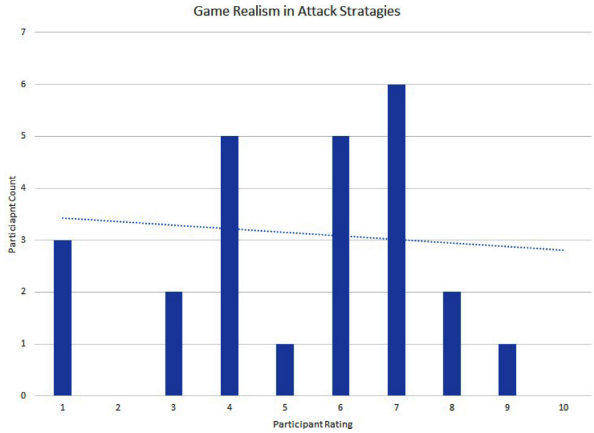


Fig. 16 – Cyber attack strategies realism.

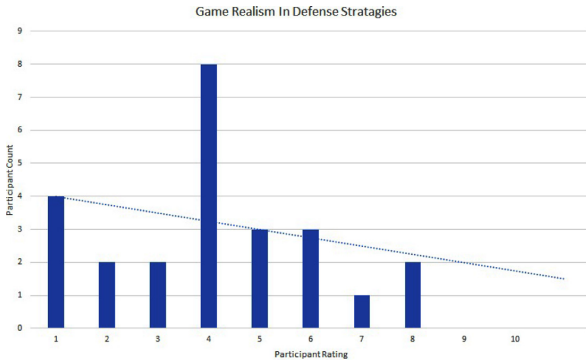


Fig. 17 – Cyber defense strategies realism.

Most of the survey participants considered that the current game was not suitable for realistically devising cyber defense strategies. Here, 19 out of the 25 participants rated the game realism as less than 6, out of which four rated it 1, two rated it 2, two rated it 3, eight rated it 4, and three rated it 5, as shown in Fig. 17.

4. Do you think that the current game can be useful for cyber-security education?
   Here, 44% of the survey participants considered that the game could be useful for cyber-security education. In addition, 36% of the participants were not sure about the game's usability, while 20% of the participants did not consider the game useful in cyber-security education, as shown in Fig. 18.

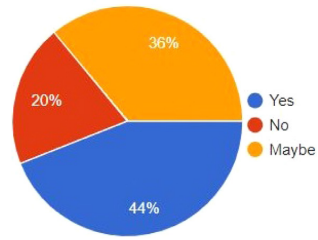


Fig. 18 – Percentage of the participants who thought the developed game is useful for cyber-security education.

9.5.2. *Player assessment:*
The game was successfully deployed during the NCSC 2019 (sta, 2019), in which it was used for players' assessment. We asked the participants to self-assess the way in which the cyber-security exercise was conducted and if their skills had improved. The findings of the survey are as follows:

1. Do you think playing/practicing the cyber-security exercise scenario in a simulated/modeled game is an efficient way to conduct cyber-security exercises?
   Here, 64% of the survey participant considered that playing and practicing cyber-security exercises in a simulated/modeled environment is an efficient way of conduct-

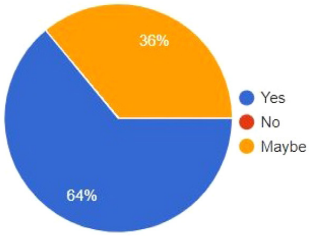


Fig. 19 – Percentage of the participants who thought it is efficient to conduct cyber-security exercises scenarios in a simulated modeled environment.

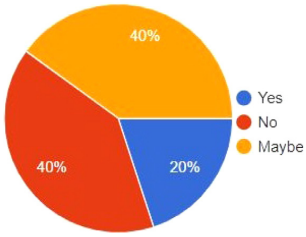


Fig. 20 – Percentage of operational strategy decision-making skill improvement in cyber-security exercises.

ing the cyber-security exercise. In comparison, 36% of the participants were not sure about it, as indicated in Fig. 19.

2. Do you think that your cyber-security exercise operational strategy decision-making skills have improved after playing this game?
   Here, 20% of the survey participants considered that the game helped them in developing their operational cyber-security strategy skills, while 40% stated they do not see any skill improvement and 40% were not sure, as shown in Fig. 20.

During the interview, one observation was that when the expert from CYFOR asked two participants *"Why did you choose the selected strategy?"*. The first replied that she randomly selected the strategy, while the second participant replied that she critically evaluated all possible strategies and then selected the optimum strategy. The decision making helped the second participant secure a place on the national team.

9.6. *Threats to validity*

We tried to quantify the findings of the field study by using statistical methods on a small data set of 25 participants. Although the data set is comparable to similar studies (Abbott et al., 2015), a larger data set would have provided us with more insights. Moreover, we did not take notes of the participants' prior experiences in cyber-security exercises and only tested one scenario during the evaluation process. This is because of the limited time available for the exercise participants. Conducting the experiment with different scenarios and adapting (Pusey et al., 2016), the scenario according to the participants' prior experiences could have yielded more accurate results, which will be taken into account in future experiments. Additionally, the post-survey questionnaire was only tested by the research team, which caused threats to its validity. This is because similar instruments were not identified in the literature. In this exploratory study, we wanted to test the survey, and we plan to validate our instrument in future work.

10. Scenario language evaluation

To evaluate the developed scenario language and its related toolset, including the compilation, deployment, and orchestration, two field studies were conducted in the context of the NCSC 2019. Below, we discuss the conducted studies and their results.

10.1. *Number of participants and demographic data*

Two technical experts were used for the assessment of the developed system. One expert from the Netherlands was actively involved in creating and deploying cyber-security exercise scenarios for NCSC 2019. The other expert from Norway has expertise in infrastructure orchestration on OpenStack using HEAT and Puppet templates.

10.2. *Task performed by the participants*

To evaluate the developed DSL and compiler's performance in terms of the cyber-security exercise scenario infrastructure and its provision, we conducted two case studies. These case studies involved replicating two infrastructures used in NCSC 2019. Two independent experts conducted the replication. For the first time, they did not use our language toolset, and the second time, they used our language toolset. The experts were tasked with generating the two scenario infrastructures: (1) a jeopardy-style CTF and (2) an attack/defense style cyber-security exercise. The exact examples of both the CTF and attack/defense scenarios can be found and accessed in the Github project of the language toolset (Dunfjeld, 2019) in the *examples* folder.

10.3. *Data collection*

After replicating the infrastructure, the experts were interviewed about their experience using the developed artifacts. Interviews were conducted in a semi-structured form to collect their qualitative feedback. The interviews contained questions about a set of four metrics used to evaluate the performance of the DSL toolset qualitatively. The four metrics are as follows:

- *Efficiency:* In this metric, we measured the time required by manual labor compared with the proposed system to deploy and generate the same infrastructure. Time is one observation data point; however, the experts' opinion was also used for making the assessment.
- *Usability:* In this metric, we tried to identify how useful the proposed system was in generating cyber-security exercise infrastructure. Expert observation and feedback were used to assess this metric.

Table 1 – Result of case studies.

Case study	Efficiency	Usability	Completeness	Flexibility
Replicating jeopardy NCSC	5 min with the developed tool compare to 20+ minutes by 2 experts without the developed tool	Bare-bones structure of scenario only	Limited to container-based challenges	Not flexible after deployment
Attack and Defense Exercises	5 min with the developed tool compare to 60+ minutes by 2 experts without the developed tool	Bare-bones structure of scenario only	Limited to container-based challenges	Not flexible after deployment

- *Completeness:* In this metric, we measured the capability of the proposed system of fulfilling the infrastructure requirements for a given cyber-security exercise scenario. The data source for this measurement was the observation made during the replication of the given infrastructures and experts' feedback.
- *Flexibility:* In this metric, we tried to identify the post-deployment modification capability of a cyber-security exercise scenario generated by the proposed system.

The list of questions asked during the interviews is presented in Appendix A.

10.4. *Data analysis*

The expert feedback was analyzed using a comparative analysis (Berg-Schlosser et al., 2009). Their feedback was compared to establish a common understanding of the performance of the system. The common understating was then used to evaluate the overall system using the pre-defined qualitative metrics.

10.5. *Summary of the results*

Both experts agreed that the developed tool was efficient in deploying cyber-security exercise infrastructure when it came to time. For example, it took only five minutes to deploy a replica of the NCSC jeopardy scenario using the developed tools; in contrast, the two experts took more than 20 min for the same task using general purpose infrastructure orchestration technologies like OpenStack HEAT. The attack/defense scenario took the experts more than an hour to deploy, while with the developed tools, they were able to replicate it in five minutes. However, in terms of usability, completeness, and flexibility, there is a room for improvement because our method only provides a bare-bones infrastructure that only supports container-based challenges. These challenges are suitable for application layer security exercises but do not provide much of an attack surface for network layer attacks. The summarized results are presented in Table 1.

10.6. *Threats to validity*

We used only two experts for the assessment of the proposed system. This is because of the lack of such experts in the field. In the future, we will try to get the feedback of as many experts as possible to obtain more feedback of the system. Moreover, the proposed system was only tested in NTNU's highly customized cloud infrastructure. There may be operational and technical difficulties in other deployment environments. We made the proposed DSL toolset open source and hopefully will receive feedback from other researchers about the operational and technical issues and testing to further enhance its performance and functionality.

11. Discussion and conclusion

In the present study, we developed a multi-layer system (toolset) to support the planning and execution of cybersecurity exercises. The developed system bridges the gap between two different perspectives: a strategic simulation-based serious game and a low-level technical cybersecurity exercise infrastructure. The glue that connects both of these perspectives is a DSL and its corresponding ontology. The language was used to (1) define the input needed to configure the simulation game, (2) transform the game specification into an intermediate scenario format, and (3) use the concrete intermediate scenario format to generate low-level infrastructure artifacts. Additionally, we conducted a case study to evaluate realism and efficiency.

We developed a serious game that provides a drag-and-dropbased graphical user interface to configure the exercise scenario based on the scenario language. This helped model and test cybersecurity exercises scenario in a simulated environment before actual deployment in an emulated environment. The game provides a layer of abstraction to model cyber-security exercise scenarios and test different attack and defense strategies. In terms of cyber-security exercise scenario modeling, we developed a DSL that enables modeling *White Team* of the members' role. The developed language allowed for efficiently translating the cyber-security exercise scenario developed in the game's simulated environment to an emulated environment of an actual infrastructure.

We conducted a case study in which we identified that the game achieved its desired objectives for strategizing cyber attack and defense. The results from the case study indicate that the game realistically represents the cyber-security exercises scenario. The toolset developed during the present research produced an emulation for a cloud-native environment, which is OpenStack based. The emulation supports most of the application and network layer protocols, making it useful in conducting cybersecurity exercises in a university setting. We suggest that such a toolset is also useful for cyber-security ed-

ucation, which is key because practicing cybersecurity strategies in a simulated environment can result in skill improvements.

Currently, the scenarios developed by our toolset offer low fidelity and are not suitable for use in a military cyber operations center and for exercises conducted to train cyber mission planners. For such scenarios, complex, multi-sector, and evolving organizational infrastructures are needed, for which the developed game is not yet flexible enough. Moreover, in terms of devising cyber defense strategies, the results are not positive. This could be because of the participants' profiles with experience in attacking techniques. The game's target audience played the game from the attacker's perspective, which, according to our assessment, did not give them full insights into a defender's actions and strategies.

Serious games can be a viable tool to model new and unique scenarios for cyber-security exercises. The modeled scenarios can be realistic and can be used to realistically devise cyber-attack strategies. In terms of cyber defense strategies, the research results are inconclusive and require further research. The developed game has been identified as a useful tool for conducting cyber-security exercises in an efficient manner, which helps in operational cyber-security skill set improvement. The target audience for the game was individuals between the ages of 16 and 25, and the game can be useful for their skill improvement. However, the developed toolset is not suitable for complex military-grade cybersecurity exercises yet.

In the future, we plan to use the data generated from the game from devising attack and defense strategies to develop autonomous attack and defense agents. These agents can emulate Red and Blue Teams' actions in an actual cyber-security exercise. It would be interesting to assign different levels of capabilities to these agents and test different cyber warfare concepts such as cyber asymmetry. Moreover, we plan to conduct a longitudinal study during the *Ethical Hacking* course taught at NTNU, which will help us identify the usability of the game in providing continuous training and self-learning, hence providing new and unique scenarios. Moreover, to improve other factors such as usability, completeness, and flexibility, we are conducting further research.

Declaration of Competing Interest

The authors declare no conflict of interest in publishing the article "Serious Games as a Tool to Model Attack and Defense Scenarios for Cyber-Security Exercises"

CRediT authorship contribution statement

Muhammad Mudassar Yamin: Conceptualization, Investigation, Methodology, Software, Validation, Writing – original draft. **Basel Katt:** Supervision, Writing – review & editing, Project administration. **Mariusz Nowostawski:** Supervision, Writing – review & editing, Project administration.

Acknowledgment

We would like to acknowledge the valuable contributions of the three undergrad students Christian Bråthen Tverberg, Maarten Dijkstra, and Nataniel Gåsøy, and one master's student, Mihkal Dunfjeld, all of whom took part in ongoing research activities at the Norwegian Cyber Range and assisted us in developing the necessary artifacts for this research.

Appendix A. Interview questioner

1. How much time did it take to deploy a specific cyber-security exercise scenario manually?
2. How much time did it take to deploy a specific cyber-security exercise scenario with the developed tool?
3. Is the deployed scenario usable for cyber-security exercise?
4. Does the deployed scenario provide the required functionality?
5. Is the deployed scenario flexible for changes?
6. What do you think can be improved in the developed tool?

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cose.2021.102450.

REFERENCES

Adam shostack: Tabletop infosec games. 2020. https://adam.shostack.org/games.html. (Accessed on 06/20/2020).

Abbott RG, McClain JT, Anderson BR, Nauer KS, Silva AR, Forsythe JC. In: Technical Report. Automated Performance Assessment in Cyber Training Exercises. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); 2015.

Allen L, Heriyanto T, Ali S. Kali Linux–Assuring Security by Penetration Testing. Packt Publishing Ltd; 2014.

Alotaibi F, Furnell S, Stengel I, Papadaki M. A review of using gaming technology for cyber-security awareness. Int. J. Inf. Secur. Res.(IJISR) 2016;6(2):660–6.

Amorim JA, Hendrix M, Andler SF, Gustavsson PM. Gamified training for cyber defence: methods and automated tools for situation and threat assessment. Proceedings of the NATO Modelling and Simulation Group (MSG) Annual Conference 2013 (MSG-111), 2013.

Ben-Kiki O, Evans C, Ingerson B, 2005. Yaml ain't markup language (yaml version 1.1. Technical Report, yaml.org, 23.

Berg-Schlosser D, De Meur G, Rihoux B, Ragin CC. Qualitative comparative analysis (QCA) as an approach, 1; 2009. p. 18.

Beuran R, Tang D, Pham C, Chinen K-i, Tan Y, Shinoda Y. Integrated framework for hands-on cybersecurity training: cytrone. Comput. Secur. 2018;78:43–59.

Casini M, Prattichizzo D, Vicino A. The automatic control telelab: a user-friendly interface for distance learning. IEEE Trans. Educ. 2003;46(2):252–7.

Chadha R, Bowen T, Chiang C-YJ, Gottlieb YM, Poylisher A, Sapello A, Serban C, Sugrim S, Walther G, Marvel LM, et al. Cybervan: a cyber security virtual assured network testbed. In: Proceedings of the MILCOM 2016-2016 IEEE Military Communications Conference. IEEE; 2016. p. 1125–30.

Cheung S, Lindqvist U, Fong MW. Modeling multistep cyber attacks for scenario recognition, Vol. 1. IEEE; 2003. p. 284–92.
Cyberforsvaret - forsvaret.no. 2020. https://forsvaret.no/cyberforsvaret. (Accessed on 06/20/2020).
Cyber security strategy Norway 2012, 2012. https://tinyurl.com/rxmw9t9m. (Accessed on 04/27/2021).
Dunfjeld M. Cyber security testbed provisioning using a domain specific language. NTNU; 2019. Master's thesis.
Endicott-Popovsky BE, Popovsky VM. Application of pedagogical fundamentals for the holistic development of cybersecurity professionals. ACM Inroads 2014;5(1):57–68.
Ecsc2019. 2019. www.europeancybersecuritychallenge.eu.
Github - mdunfjeld/ctfgen. 2020. https://github.com/mdunfjeld/ctfgen (Accessed on 03/30/2020).
Ford V, Siraj A, Haynes A, Brown E. Capture the flag unplugged: an offline cyber competition. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. ACM; 2017. p. 225–30.
Glass BD, Maddox WT, Love BC. Real-time strategy game training: emergence of a cognitive flexibility trait. PLoS One 2013;8(8):e70350.
Gurnani R, Pandey K, Rai SK. A scalable model for implementing cyber security exercises. In: Proceedings of the International Conference on Computing for Sustainable Global Development (INDIACom). IEEE; 2014. p. 680–4.
Maarten dijkstra / cyber security simulator. 2020, https://ntnu.box.com/s/1ysjltlu025h0w0383gmgziqqsu0vp85.
Heat openstack - orchestration. 2019. https://wiki.openstack.org/wiki/Heat (Accessed on 12/01/2019).
Hendrix M, Al-Sherbaz A, Victoria B. Game based cyber security training: are serious games suitable for cyber security training? Int. J. Serious Games 2016;3(1):53–61.
Hutchins EM, Cloppert MJ, Amin RM, et al. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Lead. Issues Inf. Warf. Secur. Res. 2011;1(1):80.
Knox BJ, Lugo RG, Sütterlin S. Cognisance as a human factor in military cyber defence education. IFAC-PapersOnLine 2019;52(19):163–8.
Le Compte A, Elizondo D, Watson T. A renewed approach to serious games for cyber security. In: Proceedings of the 7th International Conference on Cyber Conflict: Architectures in Cyberspace. IEEE; 2015. p. 203–16.
Liu P, Zang W, Yu M. Incentive-based modeling and inference of attacker intent, objectives, and strategies. ACM Trans. Inf. Syst. Secur. (TISSEC) 2005;8(1):78–118.
list-of-measures–national-cyber-security-strategy-for-norway.pdf. 2021. https://www.regjeringen.no/contentassets/c57a0733652f47688294934ffd93fc53/list-of-measures--national-cyber-security-strategy-for-norway.pdf. (Accessed on 04/26/2021).
Maines CL, Llewellyn-Jones D, Tang S, Zhou B. A cyber security ontology for BPMN-security extensions. In: Proceedings of the IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. IEEE; 2015. p. 1756–63.
Marshall J. The cyber scenario modeling and reporting tool (cybersmart). In: Proceedings of the Cybersecurity Applications & Technology Conference for Homeland Security. IEEE; 2009. p. 305–9.
National cyber security strategy for norway - regjeringen.no. 2020. https://www.regjeringen.no/en/dokumenter/national-cyber-security-strategy-for-norway/id2627177/. (Accessed on 06/19/2020).
Norwegian cyber security challenge 2019 (ncsc19) finale. 2019. https://www.ntnu.no/ncsc/ncsc19-finale.
Openstack docs: Openstack-ansible documentation. 2019. https://docs.openstack.org/openstack-ansible/latest/. (Accessed on 12/01/2019).
Openstack docs: Openstack networking. 2021. https://docs.openstack.org/neutron/train/admin/intro-os-networking.html. (Accessed on 04/27/2021).
Owasp top ten vulnerabilities. 2020https://www.owasp.org/index.php/.
Pham C, Tang D, Chinen K-i, Beuran R. Cyris: A cyber range instantiation system for facilitating security training. In: Proceedings of the Seventh Symposium on Information and Communication Technology. ACM; 2016. p. 251–8.
Puppet - openstack. 2020. https://wiki.openstack.org/wiki/Puppet.
Pusey P, Gondree M, Peterson Z. The outcomes of cybersecurity competitions and implications for underrepresented populations. IEEE Secur. Priv. 2016;14(6):90–5.
Russo E, Costa G, Armando A. Scenario design and validation for next generation cyber ranges. In: Proceedings of the IEEE 17th International Symposium on Network Computing and Applications (NCA). IEEE; 2018. p. 1–4.
Schmidt DC. Model-driven engineering. Comput. IEEE Comput. Soc. 2006;39(2):25.
Schreuders ZC, Butterfield E. Gamification for teaching and learning computer security in higher education. Proceedings of the {USENIX} Workshop on Advances in Security Education ({ASE} 16), 2016.
Schreuders ZC, Shaw T, Shan-A-Khuda M, Ravichandran G, Keighley J, Ordean M. Security scenario generator (secgen): a framework for generating randomly vulnerable rich-scenario VMS for learning computer security and hosting {CTF} events. Proceedings of the {USENIX} Workshop on Advances in Security Education ({ASE} 17), 2017.
Shiva S, Roy S, Dasgupta D. Game theory for cyber security. In: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. ACM; 2010. p. 34.
The penetration testing execution standard. 2020. http://www.pentest-standard.org.
Trendline analysis in excel. 2020. https://www.uwyo.edu/ceas/resources/current-students/classes/esighelp/windows help files/microsoft office/excel-trendline analysis.pdf.
Unity Technologies. https://unity.com/.
Voelter M, Benz S, Dietrich C, Engelmann B, Helander M, Kats LC, Visser E, Wachsmuth G. DSL engineering: designing, implementing and using domain-specific languages. dslbook.org; 2013.
Yadav T, Rao AM. Technical aspects of cyber kill chain. In: Proceedings of the International Symposium on Security in Computing and Communication. Springer; 2015. p. 438–52.
Yamin MM, Katt B. Inefficiencies in cyber-security exercises life-cycle: a position paper. Proceedings of the AAAI Symposium on Adversary-Aware Learning Techniques and Trends in Cybersecurity (ALEC 2018), 2018.
Yamin MM, Katt B. Cyber security skill set analysis for common curricula development. In: Proceedings of the 14th International Conference on Availability, Reliability and Security; 2019. p. 1–8.
Yamin MM, Katt B. Modeling attack and defense scenarios for cyber security exercises. In: Proceedings of the 5th Interdisciplinary Cyber Research Conference 2019; 2019. p. 7.
Yamin MM, Katt B, Gkioulos V. Cyber ranges and security testbeds: scenarios, functions, tools and architecture. Comput. Secur. 2019:101636.
Yamin MM, Katt B, Torseth E, Gkioulos V, Kowalski SJ. Make it and break it: an IoT smart home testbed case study. In: Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control. ACM; 2018. p. 26.

Muhammad Mudassar Yamin is currently doing his Ph.D. at the Department of Information and Communication Technology at the Norwegian University of Science and Technology. He is the member of the system security research group and the focus of his research is system security, penetration testing, security assessment, intrusion detection. Before joining NTNU, Mudassar was an Information Security consultant and served multiple government and private clients. He holds multiple cyber security certifications like OSCE, OSCP, LPT-MASTER, CEH, CHFI, CPTE, CISSO, CBP.

Basel Katt is currently working as an Associate Professor at the Department of Information and Communication Technology at the Norwegian University of Science and Technology. He is the technical project leader of Norwegian cyber range. Focus of his research areas are: Software security and security testing Software vulnerability analysis Model driven software development and model driven security Access control, usage control and privacy protection Security monitoring, policies, languages, models and enforcement

Mariusz Nowostawski is an Associate Professor at Norwegian University of Science and Technology. Previously, an academic lecturer at University of Otago, New Zealand. His MSc studies were focused on AI and machine learning, and his Ph.D. on autonomous systems and computational modelling of the biological process of life. Mariusz has worked on high-end networking applications on GPUs and multicore systems with Sun Microsystems and Oracle. He is currently involved in forensics research with Europol. Bitcoin anonymity. Cryptocurrencies.

2.5 Modeling and Executing Cyber Security Exercise Scenarios in Cyber Ranges

Computers & Security 116 (2022) 102635

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

TC 11 Briefing Papers

Modeling and executing cyber security exercise scenarios in cyber ranges

Muhammad Mudassar Yamin*, Basel Katt

Department of Information Security and Communication Technology, Norwegian University of Science and Technology, Teknologivegen 22, Gjøvik 2815, Innlandet, Norway

ARTICLE INFO

Article history:
Received 15 April 2021
Revised 13 January 2022
Accepted 31 January 2022
Available online 9 February 2022

Keywords:
Cyber range
Security
Exercises
Scenario
Modeling

ABSTRACT

The skill shortage in global cybersecurity is a well-known problem; to overcome this issue, cyber ranges have been developed. These ranges provide a platform for conducting cybersecurity exercises; however, conducting such exercises is a complex process because they involve people with different skill sets for the scenario modeling, infrastructure preparation, dry run, execution, and evaluation. This process is very complex and inefficient in terms of time and resources. Moreover, the exercise infrastructure created in current cyber ranges does not reflect the dynamic environment of real-world systems and does not provide adaptability for changing requirements. To tackle these issues, we developed a system that can automate many tasks of the cybersecurity exercise life cycle. We used model-driven approaches to (1) model the roles of the different teams present in the cybersecurity exercises and (2) generate automation artifacts to execute their functions efficiently in an autonomous manner. By executing different team roles such as attackers and defenders, we can add friction in the environment, making it dynamic and realistic. We conducted case studies in the form of operational cybersecurity exercises involving national-level cybersecurity competitions and a university class setting in Norway to evaluate our developed system for its efficiency, adaptability, autonomy, and skill improvement of the exercise participants. In the right conditions, our proposed system could create a complex cybersecurity exercise infrastructure involving 400 nodes with customized vulnerabilities, emulated attackers, defenders, and traffic generators under 40 minutes. It provided a realistic environment for cybersecurity exercises and positively affected the exercise participants' skill sets.

1. Introduction

Conducting operational cybersecurity exercises is a difficult and challenging task (Pham et al., 2016), and creating the environment for such exercises is error-prone and mostly manual (Beuran et al., 2018). Creating such cybersecurity exercise environments and executing exercise scenarios can be done using cyber ranges (Yamin et al., 2020). Exercise scenarios can help with conducting hands-on operational cybersecurity exercises, as well as discussion-based or table-top exercises for educational purposes. Although both types of cybersecurity exercises are very important, we identified that there were inefficiencies in operational-based exercises (Yamin and Katt, 2018b), hence hindering their capabilities to be widely used for cybersecurity education, as well as for other public or private institutions.

Multiple researchers have been trying to make the process of executing operational cybersecurity exercises more efficient and less manual labour intensive (Pham et al., 2016; Russo et al., 2020; Schreuders et al., 2017; Yamin and Katt, 2019). These researchers were successful in dealing with the inefficiencies in cybersecurity exercises. However, most of the proposed solutions are just limited to deploying the exercise infrastructure only. Moreover, because of the dynamic nature of evolving cybersecurity threats, there is a need to model scenarios so that they are adaptable enough to accommodate changes in scenario requirements before and after scenario deployment. Additionally, for a realistic cybersecurity exercise environment, there is a need to autonomously execute cybersecurity exercise operations (Jones et al., 2015; Yamin et al., 2020). These operations range from emulating virtual users and generating network traffic to executing offensive and defensive operations within the exercise environment. Traditionally, these tasks have been the responsibility of human teams; therefore, there is the need to increase the automation level to make the exercise life

* Corresponding author.
E-mail address: muhammad.m.yamin@ntnu.no (M.M. Yamin).

https://doi.org/10.1016/j.cose.2022.102635
0167-4048/

cycle more efficient. So in the current work, we investigate four research questions:

1. How can we model and execute realistic cybersecurity exercise scenarios more efficiently?
2. Is it possible to make cybersecurity exercise models adaptable to changing requirements?
3. What operations in cybersecurity exercises can be executed autonomously to reduce dependability on human teams?
4. How much do such exercise scenarios improve the skills of cybersecurity exercise participants?

In the present work, we developed a software-based solution that addresses these questions. We used our solution to model, verify, deploy, test, and execute cybersecurity exercise scenarios in a controlled and safe environment. We performed a case study with top cybersecurity talents in Norway to evaluate our solution's performance on a set of defined matrices. Our developed solution is now being actively used in research and educational activities at the *Norwegian University of Science and Technology* (NTNU), the details of which are presented in the current paper.

To demonstrate the capabilities of the proposed system, two case studies were conducted during *Norwegian Cyber Security Challenge*(NCSC) 2020. NCSC is a national competition in Norway in which individuals ranging from 16–25 years old participate. NCSC has multiple rounds, and our developed solution was used in its final rounds for two case studies. The first case study involves a penetration testing scenario in which a small organization network was orchestrated. The network had three subnets: public, demilitarized zone, and internal network. Red team members had direct access to the public network, but they had to exploit vulnerabilities in the public network to pivot into the demilitarized zone and internal network. The participants were asked to find vulnerabilities in the network and achieve a very specific objective: updating the content of a file in a specific machine present in the internal network.

The second case study was also conducted during the NCSC 2020 finals, in which an attack/defense scenario was created. The scenario topology comprised five identical isolated networks and the execution was divided into two parts. In the first part, teams were tasked with patching the vulnerabilities present in their corresponding networks in a particular amount of time. In the second part, the isolated networks were interconnected, and the teams were tasked with attacking each other's network.

The paper is structured as follows: In Section 2, we provide the necessary background and related work for this research. Continuing that, in Section 3, we present the methodology that we employed for this research. Following that, in Section 4, we present the requirement and our design for modeling and executing cybersecurity exercises. After that, in Section 6, we discuss the implementation details of our solution and its evaluation through a case study. Finally, in Section 7 and 8, we conclude the article with a discussion and conclusion respectively.

2. Background and related work

2.1. Background

2.1.1. Cyber ranges

The word "Cyber Range" was first used during the 1970s and 1980s to describe a class of mainframe super computers developed by Control Data Corporation (CDC) (Lord, 1985; Weeden and Cefola, 2010). These computers were used in mathematically intensive tasks and the modeling of complex natural phenomena. Moving forward, in 2004, the US Congress directed the Center for Technology and National Security Policy (CTNSP) to develop a program *"to find practical ways in which the defense IT community can gain a mutual understanding of defense needs and industry capabilities and identify opportunities to integrate IT innovations in the U.S. military strategy."* (Kramer et al., 2006). In the report findings, the US Northern Command suggested the creation of a "Cyber range" for homeland security and homeland defense. Cyber ranges provide an interactive representation of an organizational environment, including tools, application, network architecture, and people functions; they are used for cybersecurity training, testing, and educational scenarios in a controlled and safe environment for providing hands-on cybersecurity experiences (NIST, 2020).

We conducted a detailed study on unclassified cyber ranges (Yamin et al., 2020) in which we discussed the scenarios, functions, tools, and architecture of such platforms. We developed a taxonomy of cyber ranges and proposed a functional architecture for building future cyber ranges. In the proposed architecture, there are six modules for the cyber range, which are presented in Fig. 1. One of the modules that deal with the run-time environment has different functions such as running the exercise infrastructure on emulated, simulated, or hardware infrastructure, generating attacks, user behavior, and traffic to add necessary realism into the environment.

This run-time environment is the prime necessity for conducting operational cybersecurity exercises. Other modules of cyber ranges also depend on this run-time environment for the scenario management and the exercise monitoring. This run-time environment also facilitates the operations of training, testing, and educational modules. Because of their central role, a lot of research has been conducted on cyber ranges (Pham et al., 2016; Russo et al., 2020; Schreuders et al., 2017; Yamin and Katt, 2019), and creating such a dynamic environment is usually the responsibility of the different teams present in cybersecurity exercises, which we discuss in Section 2.1.2.

2.1.2. Cyber security exercises

There are multiple teams involved in the cybersecurity exercise life cycle that perform different roles, as follows:

1. White Team
   White team members are subject matter experts who define the cybersecurity exercises objective and plan the scenario. They can assist the other participating teams in understating the scenario and can provide hints.
2. Green Team
   Green team members are responsible for deploying the cybersecurity exercises infrastructure as per the specification of the white team members' developed scenario. They are also responsible for the monitoring and maintenance of exercise infrastructure during the cybersecurity exercises.
3. Red Team
   Red team members are the attackers in the cybersecurity exercises. They attack the cybersecurity infrastructure developed by the green team members for achieving the objectives defined by the white team members.
4. Blue Team
   Blue team members are the defenders in the cybersecurity exercises. They defend the cybersecurity infrastructure developed by the green team members for achieving the objectives defined by the white team members.

2.1.3. Cyber security exercises scenarios

There are different types of operational cybersecurity exercise scenarios. We categorized them based on the network topology that is employed to execute the scenarios.

1. Jeopardy Style CTF
   The jeopardy-style capture-the-flag (CTF) competition uses the simplest network topology in which individuals and teams of

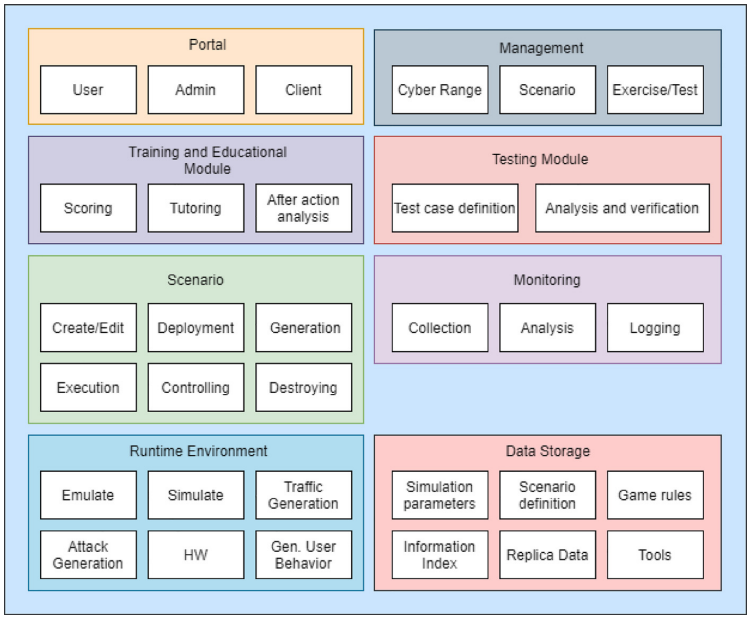


Fig. 1. Cyber range functional architecture Yamin et al. (2020).

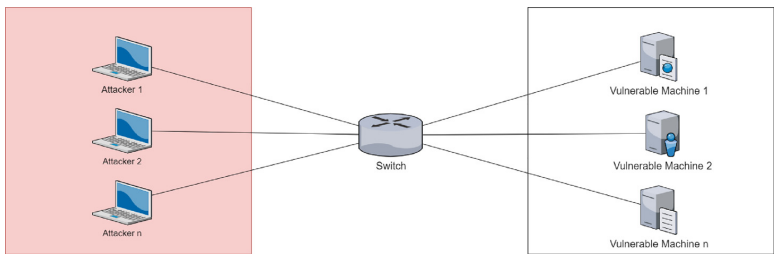


Fig. 2. Simple jeopardy-style CTF scenario network topology.

attackers have access to the machine over a network that has vulnerabilities. Attackers must exploit those vulnerabilities and retrieve a unique string, which is called the flag. The flag is then used for scoring purposes. The exercises are time-bound, so whoever scores the most will be declared the winner at the end. A simple representation of a jeopardy-style CTF competition network topology is presented in Fig. 2.

2. Attack/Defense
   Attack/defense cybersecurity exercises are team-based exercises in which each team has access to a network for which they are responsible for maintaining/defending the services of the different machines present in the network. The teams have the capability to launch attacks on the other team networks and disrupt the running of services. Flags and service availability statistics are mostly used for scoring purposes. A simple representation of an attack/defense cybersecurity exercise scenario network topology is presented in Fig. 3.
3. Red Team/Blue Team
   Red team/blue team exercises are objective-oriented, in which a team of attackers/red team is assigned an objective to penetrate into an organization and perform specific tasks such as data exfiltration and manipulation. The blue team performs actions related to incident response and forensics to figure out the red team's objectives. The exercise is conducted for skill improvement for the both red and blue teams and is mostly evaluated based on which team clearly achieved its objectives. A simple representation of the red team/blue team cybersecurity exercise scenario network topology is presented in Fig. 4.

2.1.4. Cyber security exercises life cycle

Developing, verifying, testing, and evaluating cybersecurity exercise scenarios is a challenge in and of itself. There is a whole life cycle involved in conducting cybersecurity exercises, Vykopal et al. (2017b) presented the lessons learned from an operation-based cybersecurity exercise in a cyber range. After analyzing multiple cybersecurity exercises, the researchers shared their *cybersecurity exercise life cycle*. The life cycle has five phases, as follows:

- Preparation
  In this phase, the exercises' objectives are defined, the scenario for the exercise is developed, and the necessary infrastructure for the scenarios is deployed. This phase takes from weeks to months in the cybersecurity exercise life cycle because it involves a lot of planning and development.

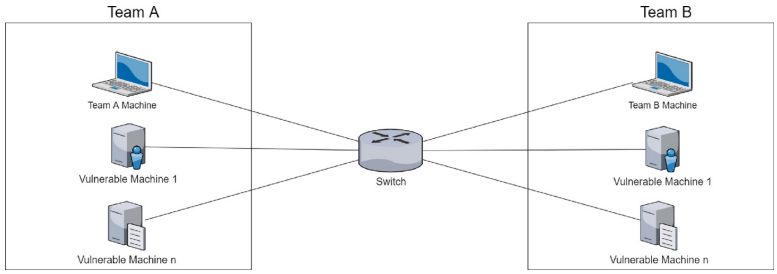


Fig. 3. Simple attack/defense scenario network topology.

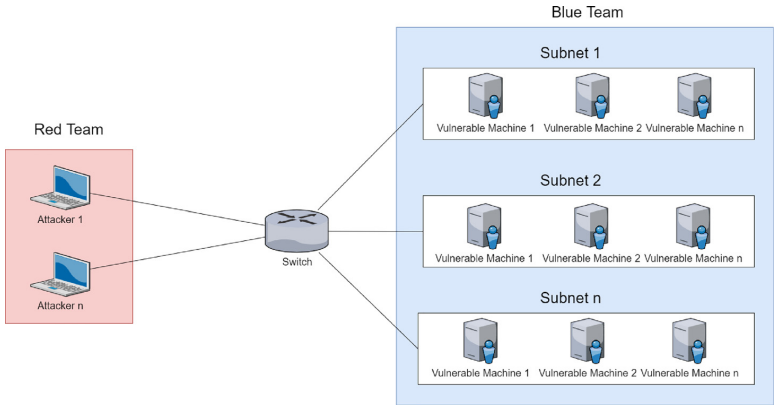


Fig. 4. Simple red/blue scenario network topology. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- Dry run
  In this phase, the developed scenario and deployed infrastructure are tested by a team of experts. Changes are made to ensure that everything is working as planned. This phase also takes a few weeks because it involves debugging the exercises scenario and infrastructure for any error.
- Execution
  In this phase, the cybersecurity exercise is executed by different teams to try to achieve the objectives defined in the exercise scenario. This phase usually takes from a few days to few weeks, depending on the nature of the exercise.
- Evaluation
  In this phase, the different participating teams' performance in the cybersecurity exercise is evaluated based on the achieved objectives. This phase usually takes a few days to evaluate team performance.
- Repetition
  In this phase, the overall exercise is analyzed to identify any technical and nontechnical problems that need to be addressed before rerunning the exercise. This phase usually takes a few days to fix newly identified issues.

From our previous research, we have identified that the current way of conducting cybersecurity exercises is not efficient (Yamin and Katt, 2018b). Here, introducing automation at different phases of the cybersecurity exercise life cycle can greatly reduce the time required for such exercises (Yamin et al., 2018). Another problem with the current way of conducting cybersecurity exercises is their static nature because they do not offer active opposition to attackers and defenders (Jones et al., 2015) and are not adaptable to real-life environments. Hence, we propose a modern cybersecurity operation triad that can be applied to the cybersecurity exercise life cycle to make them more efficient and address these shortcomings.

2.2. Related work

A lot of research has been carried out in the development of security testbeds and cyber ranges; we will highlight some of the related works in the field. In 2000, the development of EMULAB/Netbed began (White et al., 2002), which is a combination of software and specialized hardware facilities. It has two parts: end nodes and core nodes. The end nodes host the experimental artifacts, while the core nodes are designed to be used as end nodes, simulated routers, traffic shaping nodes, and traffic generators. The core nodes allow the EMULAB/Netbed infrastructure to be reconfigured, making it useful for a variety of networking and cybersecurity experiments.

In 2004, the cyber-Defense Technology Experimental Research laboratory (DETER) (Mirkovic et al., 2010) started an over 300-node facility. DETER uses physical nodes and provides access to students over SSH DETER uses Emulab for the programming of routers present in the network to create new network topologies. At its inception, most of the work in DETER lab was done manually, with a later edition of the automation of traffic generation.

In 2008, the first phase of the Cyber Range and Training Environment (CRATE) (Almroth and Gustafsson, 2020) started. CRATE provides a hybrid cyber range environment; that is, it runs on both emulated virtual machines and dedicated hardware. CRATE supports the design, deployment, and execution of cybersecurity scenarios through a set of dedicated APIs that use JSON as the input.

CRATE is deployed over local servers, and access to users is provided through VPN. CRATE uses a high level of automation and divides the cyber range network into parts, first *event plane* and then *control plane*. In the event plane, internet traffic and attacks are emulated to add realism in the environment, while in the control plane, the cybersecurity exercise is conducted.

In 2012, researchers presented Telelab (Willems and Meinel, 2012), which was used to develop a single virtual machine-based lab environment for cybersecurity exercises. The interesting thing about this is that it uses an XML-based lab requirement specification to inject vulnerabilities in a virtual machine through a local agent running on the machine. Similar to Telelab, researchers created *SecGen* (Schreuders et al., 2017), which also uses XML configuration language for creating a virtual machine with vulnerabilities randomly selected from a catalog. They used Puppet and Vagrant for automating the process of vulnerability injection. Similar to Telelab and Secgen, in 2019, researchers developed *Alpaca* (Eckroth et al., 2019), which creates single virtual machines for cybersecurity exercises using Ansible. However, it has a prolog-based exercise planner, which is used to develop multi-step attack scenarios for complex training.

In 2016, researchers presented *Cyber Range Instantiation System for Facilitating Security Training* (CyRIS) (Pham et al., 2016). They automated the process of designing and deploying infrastructure for cybersecurity exercises using a YAML-based language. The researchers used Libvirt (lib, 2021) virtualization API for deploying exercises infrastructure on local servers. Continuing this, in 2017, they developed CyTrONE (Beuran et al., 2017), an integrated cybersecurity training framework. CyTrONE uses YAML-based language to do two things: first, create content for a learning management system (LMS), which is a mobile-based application and second is to design, deploy, and emulate attacks on a virtual environment. For the infrastructure orchestration, CyTrONE uses CyRIS.

In 2016, researchers presented (Yasuda et al., 2016) *a mimetic network environment construction system* Alfons. It was developed to mimic a realistic environment for malware execution and dynamic analysis in a local environment. It was written in Ruby using an XML-based environment composition file with SpringOS API calls to deploy the infrastructure on StarBed virtualized nodes. Alfons was not designed for conducting cybersecurity exercises, but it was developed to provide a platform to conduct forensic analyses for blue team members.

In 2017, researchers presented the KYPO Cyber Range (Vykopal et al., 2017a), which uses a JSON-based *scenario definition language*, which is used for designing and deploying the exercise infrastructure on an Openstack-based cloud environment. The scenario specification is translated into Ansible and Puppet scripts, which are then used for infrastructure orchestration. It supports the execution of attacks on infrastructures such as for phishing and DoS through SDL-defined templates. It is useful for conducting CTF-like competitions and has a physical facility for conducting cybersecurity exercises as well; however, it can also be accessed remotely.

In 2020, researchers presented the *Cyber Range Automated Construction Kit* (CRACK) (Russo et al., 2020), which supports the design, automated verification, deployment, and automated testing of complex cyber range scenarios. The CRACK framework is built on the extended version of a *scenario definition language*(SDL) (Russo et al., 2018), which is YAML based. SDL was developed to be compatible with open TOSCA (Ope, 2021) standards and is suitable for deploying infrastructure on the cloud in an agnostic manner. The researchers combined their SDL with Datalog logic programming for verification of the scenario properties, and they performed a case study for conducting a cybersecurity exercise on the infrastructure of a small organization. In the case study, three networks were behind firewall protection. Also, in 2020 researchers presented the AIT cyber range (Leitner et al., 2020), which uses Ansible and Terraform infrastructures for creating a cybersecurity exercise environment. The conducted exercises involved nearly 350 students.

We consider EMULAB, DETER, and CRATE as hardware-specific reliant testbeds that support cybersecurity exercises. We believe they can support large-scale cybersecurity exercises; however, this will require a lot of human effort for setting up the exercise environment. Telelab, Secgen, and Alpaca can be used for small-scale training exercises but are not suitable for large-scale exercise. The CYRIS, KYPO, CRACK, and AIT cyber ranges can support large-scale exercises using infrastructure as code and cloud technologies; however, they do not provide enough flexibility to change the exercise infrastructure after deployment. Moreover, the above solutions do not provide the necessary friction (Jones et al., 2015) for conducting realistic cybersecurity exercises.

In our solution, we utilized previous suitable techniques and added new elements to make the cybersecurity exercises more realistic. We utilized similar techniques from TELELAB to make virtual machines vulnerable in an agentless manner. We developed a JSON-based SDL similar to KYPO for infrastructure provisioning. We implemented a similar solution from CRACK for scenario infrastructure verification. We added similar capabilities in the CRATE event plane for emulating attacks, presenting user behavior, and generating network traffic. Finally, we introduced emulated defenders in the cyber range environment to make the exercise more realistic.

If we compare the proposed solution with other systems identified in the literature we can see that most systems are focusing on preparation of cybersecurity exercise environment. While *CRACK* is using formal verification for *Dry Run* purposes. *KYPO*, *CRACK* and our proposed system are using widely adapted cloud technologies that make them more computationally repeatable in terms of infrastructure deployment compare to other systems that are using very specific hardware and virtualization technologies. For dry run we used formal modeling and analysis for verification of different scenario properties before the scenario actual operational deployment. In terms of execution, the proposed system is introducing new capabilities like attacker and defender agents for conducting cybersecurity exercises in a more efficient manner. The process of evaluation can be automated in different ways, like flags style scoring. However, for complex exercises, detailed root cause analysis of system compromise is required which needs further investigation. A comparison between our proposed solution and other systems present in the literature based upon cybersecurity exercise life cycle is presented in Fig. 5.

3. Methodology

The overall research methodology that we used is **DSR** (*design science research*) (Vaishnavi and Kuechler, 2015). DSR focuses on the development and performance improvement of artifacts for increasing the functional performance of the artifact. These artifacts are usually algorithms and systems that involve human-computer interactions. DSR has three parts 1) knowledge flows, 2) process steps, and 3) output. Knowledge flows recursively integrate the information identified in the previous process steps into the next process steps. The process steps involve six processes: 1) awareness of problem, 2) suggestions, 3) development, 4) evaluation, and 5) conclusion. The execution of these processes results in the output in the form of 1) proposal, 2) tentative design, 3) artifacts, 4) performance measurement, and 5) results. We published our findings related to the proposal and tentative design in the following research articles (Yamin and Katt, 2018b; 2019; Yamin et al., 2020; 2018), in which we identified that the current way of executing cy-

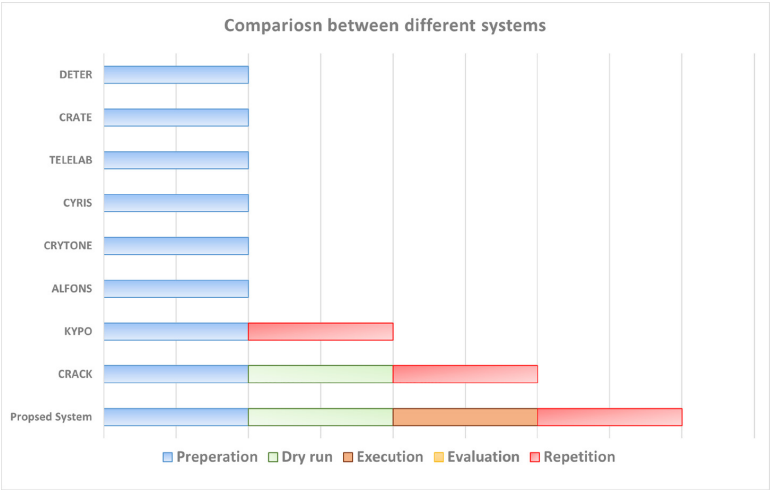


Fig. 5. Comparison between different cyber range systems with respect to cybersecurity exercise life cycle.

bersecurity exercises is not efficient and that automation can help to reduce this inefficiency.

For the artifact's development and to reduce these inefficiencies, we used **MDE** (*model-driven engineering*) (Schmidt, 2006). Model-driven approaches provide the ability to express complex domain-specific concepts in an abstract manner, which was very difficult in third-generation programming languages. This helps in increasing the productivity by a factor of 10 and improves consistency, traceability, and maintainability in the software development process (Chang et al., 2019). MDE combines two important technologies:

1. **DSML** (*Domain specific modeling languages*)
   In DSML, domain experts specify the domain knowledge in the form of a model. The model contains the concepts from the domain, their key semantics, relationship, and constraints associated with combining different concepts. DSML is used to specify the specific problem related to a specific domain in a DSML's instance, which is the abstract and human-readable representation of the problem.
2. Transformation engines and generators
   Transformation engines and generators take the DSML's instance and transform them into concrete software artifacts in an automated manner. This automated process of software artifact generation involves multiple steps, which include, but are not limited to, the following:
   - Text-to-model transformation
     In this step, the human-readable DSML instance is transformed into a computer-readable model. The DSML instance is usually the source code of a program that is transformed into a computer model using different parsing rules.
   - Model validation
     In this step, the model is verified using a set of defined semantic rules to ensure it is correct, which is done by construction implementation of the model.
   - Model to model, or model to code, transformation
     In this step, the model is transformed into another model, or a concrete code, that represents the problem specified in the DSML's instance.

The proposed system has an abstract scenario modeler that can generate two artifacts: one is a concrete DSL *domain specific language* instance that can be deployed to execute the cybersecurity exercise scenario, and the other is as a formal model of the scenario in Datalog, which can be used to analyze the different properties of the scenario before deployment. A model-to-model translation methodology was used in scenario modeler, which gave us the ability to logically verify some of the scenario properties. We developed a JSON-based **DSL** and used *text-to-model transformation* for orchestrating the exercise infrastructure and executing different cybersecurity operations. We used logic programming (Lloyd, 2012) to formally verify and analyze and verify different properties of the scenario for *model validation* before actual deployment. We used *model-to-model transformation* to combine the DSL model with a formal model, hence providing *meta-model conformance*. For the practical implementation of the artifact, we used different programming languages and operating system automation techniques, ranging from Python, Bash, Power shell, HEAT templates, and many more. The programming languages and techniques were selected with no particular preferences and were employed as the functionality's need arose. The artifact was developed in a very modular way. Each module can work independently from each other, providing us with a lot of flexibility in executing different cybersecurity operations.

For performance measurement of the developed artifact, we employed **applied experimentation** in operational cybersecurity exercises (Edgar and Manz, 2017). In *applied experimentation*, the performance of the developed artifact is measured against a set of predefined test cases and benchmarks. These were used to evaluate the overall performance of the artifact. Because the developed artifact involved system performance for deploying the exercise infrastructure and skill improvement of exercise participants, we used a mixed methods approach to gather quantitative and qualitative research data. We conducted a case study in which a cybersecurity exercise scenario was deployed and executed. We measured different quantitative matrices such as the scenario deployment efficiency, its usability in cybersecurity exercises, its flexibility to accommodate new changes, its adaptability to be used in contexts other than the defined context, and its scalability. We conducted pre and post-exercise surveys to measure the qualitative matrices,

for example, realism and skill improvement from the cybersecurity exercise scenario execution.

4. Design

4.1. Requirements for modern cyber security operations

With the rapid advancement of technologies such as *IaC* infrastructure as code (Wha, 2020) and *SOAR* security orchestration, automation, and response (Wha, 2020), it has become increasingly apparent that cyber operations are evolving. This advancement was made to execute such operations in an efficient, adaptable, and autonomous manner. This enabled us to reduce the cybersecurity exercise life cycle's inefficiencies and execute them in an efficient, adaptable, and autonomous manner.

4.1.1. Efficient

As stated earlier, preparing the environment for cybersecurity exercises takes anywhere from weeks to months. Efficiency in this context implies that the process of creating the exercise scenario and infrastructure should not take more than a couple of hours to a few days and should be applicable and accurate, as per the specified requirements. Efficiency means reducing the time required for creating the scenario, which does not affect the scenario's applicability and accuracy. ***Applicability*** measures whether the deployed scenario is employable for conducting practical cybersecurity exercises. In comparison, ***accuracy*** measures whether the deployed scenario fulfills the specified requirements in the scenario model. Multiple factors can affect the timeline, which may include the size and complexity of the exercise, but these factors need to be addressed in a systemic manner to remove inefficiency.

4.1.2. Adaptable

Here, adaptability implies that the deployed cybersecurity exercise infrastructure is flexible and scalable enough to adapt to new changes per the chaining scenario requirements. ***Flexibility*** measures the deployed scenario's capability to accept changes after deployment; in comparison, ***Scalability*** measures whether the deployed scenario is expandable enough to accommodate additional teams in the scenario. The adaptability will enable the cybersecurity exercises scenario developers to adapt the scenario for participants with different skill levels, creating a balanced environment for different participants.

4.1.3. Autonomous

In this context, autonomous implies that most of the cybersecurity operations are executed with minimum or no human interference. If we consider the example of autonomous cars, we can identify six levels of autonomy: *0: no automation, 1: driver assistance, 2: partial automation, 3: conditional automation, 4: high automation*, and *5: full automation* (Taxonomy, 2020). Cyber operations such as an attack and defense scenario can be autonomous in a cybersecurity exercise environment. Human intervention is still possible in monitoring the situation; however, this intervention must be conditional, which would be in contrast to the second level, where human monitoring is required.

4.2. Integrating modern operations with cyber security exercise life cycle

After an in-depth analysis of the five phases of the cybersecurity exercise life cycle, we divided these phases in to eight independent modular activities, and those eight modular process have 11 technical functions. The updated cybersecurity exercise life cycle is presented in Fig. 6, and details of the new activities and functions of the cybersecurity exercise life cycle are given below.

4.2.1. Preparation

Scenario Modeling In this phase of the scenario, a logical network topology with vulnerabilities was modeled, attacker and defender capabilities to exploit or defend those vulnerabilities were defined, and probable attack and defense strategies were analyzed and logically verified. We developed a DSL to specify different cybersecurity operational requirements during the exercise. A DSL provides a layer of abstraction to solve domain-specific problems without dealing with the necessary overhead of general purpose programming language. We simplified the cybersecurity operation in an exercise into five general operations: *infrastructure orchestrator, vulnerability injector, attacker agent, defender agent*, and *traffic generator*. These operations have their specific properties in a cybersecurity exercise scenario; to simplify things, the properties of our scenario modeling language are presented in BNF (Backus-Naur form). BNF is a notation technique for context-free grammar and is also used to describe the syntax of languages used in computing, such as computer programming languages.

1. **Scenario Language Design**
   The scenario modeling was performed through our developed scenario language, which was verified logically by Datalog. Our scenario language has multiple parts whose requirements are presented in the coming sections. Before defining the actual scenario modeling language, we first define some of the basic variables that were used in the language:
   These basic variables are used to define the characters, strings, integers, IP addresses, ranges, and CIDR used in rest of the language.
   - **Infrastructure Orchestrator**
     The infrastructure orchestrator has two parts *subnet* and *machine*. Subnet is used to represent the network with which machines are connected. It requires three things to be specified in the language:
     (a) CIDR (CIDR value like *10.10.0.10/24*)
     (b) Name (String value that indicates the name of subnet, like *Public*)
     (c) A network interface (String value of *Network ID* from Openstack)
     The second part *machine* is a host; hosts are virtual machines that are connected to the specified *subnet* and in which vulnerabilities are injected. It requires r things to be specified in the language:
     (a) Name (String value that indicates the name of a machine like *Machine1*)
     (b) Operating system (String value *operating system id* that is already uploaded over Openstack)
     (c) Key (String value *SSH key* that can be used for maintenance and monitoring)
     (d) Depends (String value name of *subnet* with whine *Machine* is connected)
     The BNF representation for specifying the requirements of infrastructure orchestration is as follows:
   - **Vulnerability Injector**
     When the *machines* are deployed, then the vulnerability injection can be done. The process *vulnerability injection* requires the following properties:
     (a) MachineIP (IP address value of a deployed machine like *10.10.1.10*)
     (b) MachineUserID (String value that indicates admin user account on a deployed machine like *root*)
     (c) MachineUserPassword (String value that indicates the admin user password of the deployed machine like *toor*)
     (d) OS (String value that indicates the name of a machine like *Machine1*)

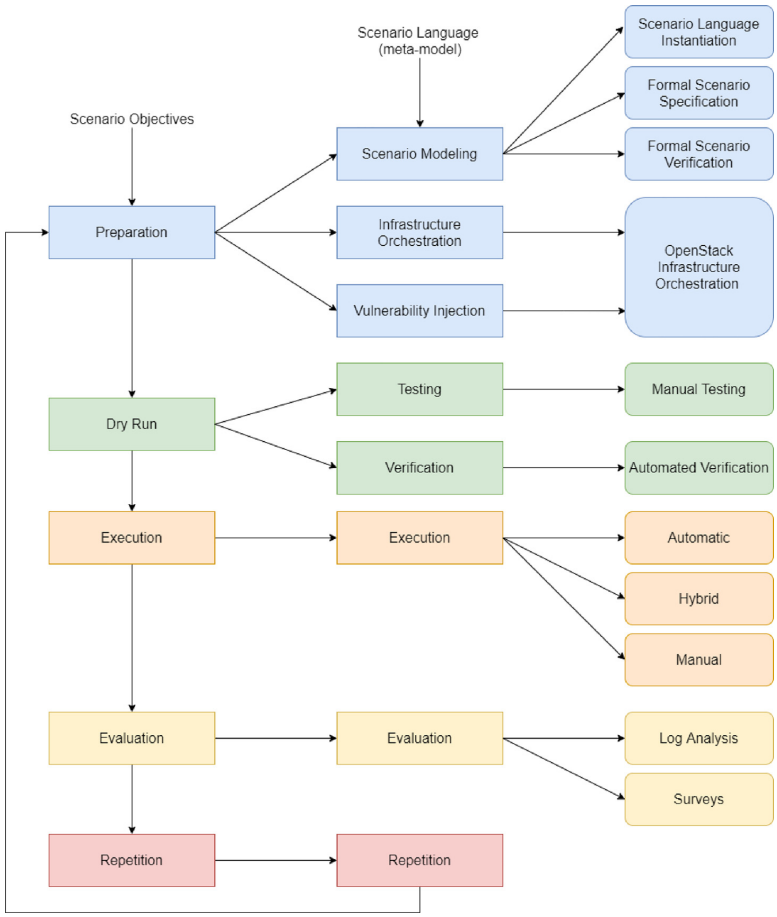


Fig. 6. Modern cybersecurity operations integrated with cybersecurity exercise life cycle.

(e) Vulnerability (String value that indicates the type of vulnerability that needs to be injected in the machine like *WeakPassword*)
(f) Parameter (String value that indicates the vulnerability-specific parameter like "passwd|apollo")

The BNF representation for specifying the requirements of the vulnerability injection is as follows:

- Attacker Agent After the vulnerabilities are injected, they can be verified by an *attack agent*. It can also be used to emulate attacker behavior during a cybersecurity exercise. The attacker agent requires six properties to be specified before its execution.
(a) ToolName (String value that indicates the tool name to be used by the agent like *nmap*)
(b) AgentIP (IP address value of a *Kali Linux*-based machine present in the network topology like *10.10.1.5*)
(c) AgentUserID (String value that indicates admin user account of the *Kali Linux* machine like *root*)
(d) AgentUserPassword (String value that indicates admin user password of the *Kali Linux* machine like *toor*)
(e) Argument ((String value that indicates attacker agent action-specific parameter like *-sS -sV*)
(f) Target (IP address value of a deployed machine like *10.10.1.10*)

The BNF representation for specifying the requirements of attacker agent behavior is as follows:

- **Defender Agent**
To add friction and realism in cybersecurity exercises, a host-based defender agent is developed that can be injected into the deployed machines. It has the following five properties.
(a) MachineIP (IP address value of a deployed machine like *10.10.1.10*)
(b) MachineUserID (String value that indicates the admin user account on deployed machine like *root*)
(c) MachineUserPassword (String value that indicates the admin user password of the deployed machine like *toor*)
(d) OS (String operating system name of the deployed machine like *Windows*)
(e) Parameter (String value that indicates the action-specific parameters of the defender agent in a CSV file like *netstat -ano| taskkill /F /PID ??*)

The BNF representation for specifying the requirements of defender agent behavior is as follows:

- Traffic Generator

To make the scenario execution dynamic with realistic traffic, two modules have been developed. First, there is TcpRelay, which can replay traffic from prerecorded network traffic using PCAP files. Second, there is VncBot, which can emulate user behavior from prerecorded VDO files. These modules have similar six requirements as those of the attack agent:

(a) ToolName (String value that indicates the tool name to be used by the agent like *VncBot*)
(b) AgentIP (IP address value of a *Kali Linux*-based machine present in network topology like *10.10.1.5*)
(c) AgentUserID (String value that indicates the admin user account of the *Kali Linux* machine like *root*)
(d) AgentUserPassword (String value that indicates the admin user password of the *Kali Linux* machine like *toor*)
(e) Argument ((String value that indicates an attacker agent action-specific parameter like *Test.vdo|toor* where *toor* is the password of the VNC-enabled machine deployed in the exercise infrastructure)
(f) Target (IP address value of a deployed machine like *10.10.1.10*)

The BNF representation for specifying the requirements of the traffic generator is as follows:

2. **Formal Scenario Specification**

Multiple models have been proposed for modeling the attackers' and defenders' behavior during a cyber engagement. These models focus on the chain of events that lead up to compromising the computer systems. Lockheed Martin Hutchins et al. (2011) put forward the *cyber kill chain* methodology to protect computer network damage espionage. The *cyber kill chain* consists of the following steps and stages:

(a) Reconnaissance: Looking out for intrusions, via email, conferences, and so forth.
(b) Weaponization: Malicious intent realized through PDF and word files for intrusion purposes.
(c) Delivery: Sending the intrusion payload 2004–2010 according to Lockheed Martin mostly sent through email attachments, USB, and websites.
(d) Exploitation: Intrusion is in and focuses on its target
(e) Installation: Providing a means of access to the compromised system to the adversary
(f) Command and control: After getting access, obtaining all the controls of the compromised, intruded system, and controlling it, done manually, not automatically via an internet controller server.
(g) Actions on objectives: To get access to the information and resources for which the last six phases took place.

They (Hutchins et al., 2011) also proposed the defender's course of action against the attacker in *cyber kill chain*, including what type of visibility the defender has and what tools and techniques a defender can use to stop the attacker. The course of action matrix is presented in Fig. 7.

Other models like *MITRE* (MIT, 2020) and the *unified kill chain* (Pau, 2020) provide more technical details of the attacker's and defender's steps, but at this stage, we chose the *cyber kill chain* for two main reasons: First, it is very well established and well-known modeling technique, and second, it offers a layer of simplicity and abstraction compared with other models, which focus more on core technical steps.

(a) **Scenario Formalization Background**

We used Datalog (Dat, 2020) for formal modeling of the scenario and to verify the different scenario properties. Datalog is a programming language based on a declarative logic (Lloyd, 2012). It is employed by researchers for large-scale software analyses (Naik, 2020), automatic evaluations of cybersecurity matrices (Zaber and Nair, 2020), and the verification of cybersecurity exercise scenarios (Russo et al., 2020), making it suitable as a formal model for cybersecurity exercise scenarios. It consists of two parts: facts and clauses. A fact conforms to the parts of the elements of the predicated phenomenon. A clause refers to information deriving from other subsets of information. Clauses rely on terms, which can contain variables; however, facts cannot. It adjudicates whether the specific term is adherent to the specified facts and clauses. If it happens to be so, the specific query is validated via a query engine, providing the prerequisite facts and clauses.

When running a Datalog operation, the specified conditions include a combination of two facts along with a singular clause. We assign a condition that if the query is valid, a specific response is to be expected at the end. The conclusion of the said experiment is that the specific response is received and that the query is satisfied. By utilizing the clauses via their variables, the engine can pinpoint and find the result. For a concrete example (Ceri et al., 1989), consider the facts *"John is the father of Harry"* and *"Harry is the father of Larry"*. A clause will allow us to deduce facts from other facts. In this example, consider we want to know *"Who is the grandfather of Larry?"*. We can use three variables *X,Y* and *Z* and make a deductive clause: If *X* is the parent of *Y* and *Y* is the father of *Z*, then *X* will be the grandfather of *Z*. To represent facts and clauses, Datalog uses *horn clauses* in a general shape:

$$L_0 : -L_1 ..., Ln$$

Each instance of L represents a *literal* in the form of a *predicate* symbol that contains one or multiple *terms*. A *term* can have a constant or variable value. A Datalog clause has two parts: the left hand side part is called the head, while the right hand side part is called the body. The body of the clause can be empty, which makes the clause a fact. A body that contains at least literal represents the rules in the clause. Lets us represent the above mentioned facts that *"John is the father of Harry"* and *"Harry is the father of Larry"* as follows:

$$Father(John, Harry)$$

$$Father(Harry, Larry)$$

The clause if *X* is the father of *Y* and *Y* is the father of *Z*, then *X* will be the grandfather of *Z* can be represented as follows:

$$GrandFather(Z, X) : -Father(Y, X), Father(Z, Y)$$

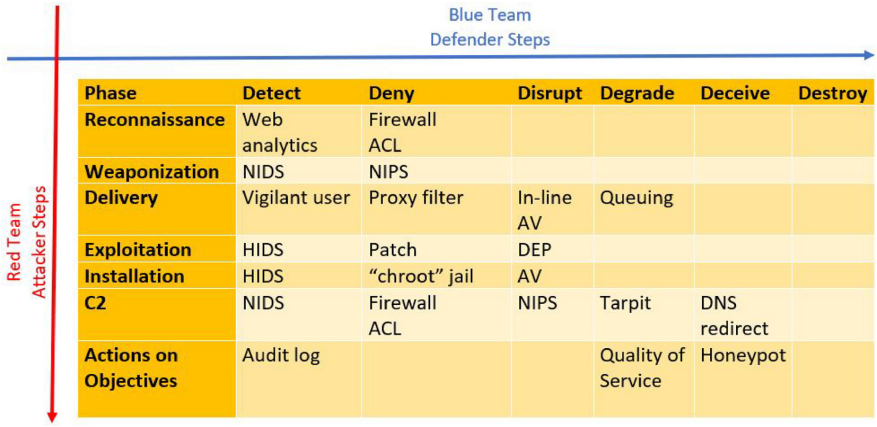
(b) **Scenario Formalization**

We have defined four basic predicates for our scenario modeling, which are *1) link, 2) vulnerable, 3) capability,* and *4) killchain*. The facts for the scenario model are presented as follows:

The *Link* predicate is logically represented as *Link(H,N)*, and it has the two variables of host *H* and network *N*. *H* is a string value that indicates the machine name (virtual machine name), while *N* is a string value that indicates the name of the network with which it is connected. For a concrete example, say a *Host* name *'Machine1'* connected with network name *'Public'* can be represented as follows:

$$Link('Machine1', 'Public')$$

The *Vulnerable* predicate is logically represented as *vulnerable(H,V)*, it has two variables host *H*, which is the specified machine name and *V*, a string value that indicates the presence of a particular vulnerability in *H*. A concrete example



Phase	Detect	Deny	Disrupt	Degrade	Deceive	Destroy
Reconnaissance	Web analytics	Firewall ACL				
Weaponization	NIDS	NIPS				
Delivery	Vigilant user	Proxy filter	In-line AV	Queuing		
Exploitation	HIDS	Patch	DEP			
Installation	HIDS	"chroot" jail	AV			
C2	NIDS	Firewall ACL	NIPS	Tarpit	DNS redirect	
Actions on Objectives	Audit log			Quality of Service	Honeypot	

Fig. 7. Cyber kill chain course of action matrix for attacker and defender Hutchins et al. (2011).

could be that *'Machine1'* is vulnerable to a *'SSHBruteforce'* attack and can be represented as follows:

$$Vulnerable('Machine1', 'SSHBruteForce')$$

The *capability* predicate is logically represented as *capability('V','A','DE')*, and it has three variables *V*, which is the vulnerability present in *H*, *A*, which is a *Bool* value that indicates whether a particular vulnerability *V* is exploitable by the attacker and *DE* that indicates whether a particular vulnerability *V* is defendable by the defender. A concrete example of a *'SSHBruteforce'* vulnerability that can be exploited by an attacker but cannot be defended by the defender is represented as follows:

$$Capability('SSHBruteForce', 'YES', 'NO')$$

The *KillChain* predicate is logically represented as *KillChain(H,R,W,D,E,C,O)*. It has seven variables host *H* and raw *cyber kill chain* process of reconnaissance *R*, weaponization *W*, delivery *D*, exploitation *E*, command and control *C*, and actions and objectives *O*. The *cyber kill chain* process variables have *Bool* values that were assigned based on *V* present in *H*. A concrete example for a *host 'Machine1'* that is completely exploitable as per the *cyber kill chain* can be represented as follows:

$$KillChain('Machine1', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')$$

A detailed scenario model with all its facts and clauses for the scenario presented in Fig. 10 is given in Appendix B. We used the clauses for logical verification of the scenario, which are presented in next section.

3. **Formal Scenario Verification**

We developed a tool for scenario modeling and verification. The tool integrates the concepts of our scenario language with Datalog modeling and provided us with the capability to model and verify cybersecurity exercise scenarios at the same time. In the scenario modeler, a user can specify the following:

(a) The networking topology that is required for the scenario
(b) The type of machines that are present in the network
(c) The type of vulnerabilities present in the machine
(d) The capabilities of the attacker and defender who can exploit or defend those vulnerabilities

After the specification is given to the modeler, the modeler can generate the instance required for the orchestration of cybersecurity exercise operations and a formal model in Datalog. Different type sof logical analyses can be performed before the actual deployment; some examples of security properties and questions that can be verified include the following:

(a) Which machines are reachable from a specific point in the network?
(b) Which machines are vulnerable to attack and can be reached by an attacker?
(c) Which machines are vulnerable to an attack but can be defended by the defender to limit attacker ingress into the network?

This is achieved by defining clauses that contain specific rules related to the scenario. First, we need to logically inter *link* different *hosts*. This can be done by creating a rule for a direct bidirectional link connection between the hosts using variables *X* and *Y*, as follows:

$$CanReach(X, Y) \leq Link(Y, X)$$

Second, similar to the grandfather and grandchildren case, to identify which *hosts* are indirectly connected in the network, we can create a new *CanReach* with variable *Z*. This can be used to find a direct link between *X* and *Y*, as well as an indirect link between *X* and *Z* through *Y*:

$$CanReach(X, Y) \leq Link(X, Y)$$
$$CanReach(X, Y) \leq Link(X, Z)$$

To check which machines are connected to a machine, for example, *Machine1*, with a specific vulnerability, for example, *BufferOveflow*, we can verify this by the following clause:

$$CanReach('Mahine1', Y) \,\&\, Vulnerable(Y, 'BufferOverFlow')$$

To check which *hosts* are connected to a vulnerable *host*, for example, Machine1, which is not defendable by a defender, we can verify it with the following clause:

$$Capability(V, 'YES', 'NO') \,\&\, CanReach('Mahine1', Y) \,\&\, Vulnerable(Y, V)$$

To integrate *cyber kill chain* concepts into the model, we can specify the impact of a *vulnerability* injected in the *host* regarding whether it allows the attacker to perform steps like *reconnaissance*, *exploitation*, and so forth. This impact is a bool value that suggests the *cyber kill chain* stage the attacker can theoretically reach. We can create the following clause:

$$Capability(V, 'YES', 'NO') \,\&\, CanReach('Mahine1', Y)$$

$$\&Vulnerable(Y, V)\,\&KillChain(Y, 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')$$

The verification of the scenario can be done on run time before the actual deployment of the exercise infrastructure. The verification process used the facts and clauses generated in Datalog syntax through our developed tool. With the help of mathematical logical operation, specially transitive relation, the Datalog engine can return properties verification results based upon given quires. The queries are similar to SQL quires and can use logical operators like AND or OR for verification of different properties. If the property is verified then the Datalog engine will return the output containing the elements that were verified by the query. If the output is empty then it is considered that the property is not verified. A complete scenario model with its properties verification steps are presented in Appendix B.

Infrastructure Orchestration In this phase, the modeled scenario that has been formally verified is transformed into an emulated network topology. This transformation is achieved by utilizing infrastructure as code technologies in which the template for the infrastructure orchestration is generated; this is deployed over a cloud instance. Multiple cloud providers like *Microsoft, Google, and Amazon* provide infrastructure orchestration technologies, but they are a closed source and paid solution. We opted for an open source solution called *Openstack*, which provides a functionality similar to *Microsoft, Google, and Amazon*; however, it also provides the capability to set up local cloud infrastructures without relying on third-party infrastructure. Openstack provides infrastructure orchestration to *HEAT* templates. HEAT templates provide an interface to specify the requirement for the network topology and the type of system present in the network. *Vulnerability Injection* Vulnerability injection in this type of network topology is a difficult process (Russo et al., 2018). Researchers have used different infrastructure configuration technologies like Ansible and Puppet for vulnerability injection (Leitner et al., 2020). However, we opted for a fundamentally different technique for vulnerability injection and developed our own custom vulnerability injector. Vulnerabilities are injected per the scenario model requirement using different operating system automation techniques. These OS automation techniques basically open an SSH connection in the machines deployed in the emulated network and manipulate software, services, and configuration using *Bash*, *Powershell*, and *Python* scripts. This enabled us to modify the scenario after infrastructure deployment and inject new vulnerabilities to make the scenario more flexible and balanced, if required.

4.2.2. Dry run

In the dry run phase, different scenario properties are checked to identify whether the deployed scenario fulfills the specified requirements in the scenario model. We divided this phase into two parts. *Manual Testing* Manual testing is performed as a quality assurance process for verifying different scenario requirements. In this phase, the deployed infrastructure is manually checked for any abnormalities. This is done by manually executing a dry run, which involves checking the network topology and exploiting the injected vulnerabilities. *Automated Verification* A manual dry run of the exercises usually takes a lot of time as well. To address this issue, we used the attacker agent to verify different scenario properties automatically. The attacker agent is a Kali Linux-based host machine present in the deployed network infrastructure. It receives the instruction of what actions to take from the scenario language. When a vulnerability is modeled to be injected into a host, a model for the attacker agent action is also generated to verify the vulnerability properties. This automatic verification includes different network link connections and vulnerability presence, along with exploitability.

4.2.3. Execution

Preparation and the resulting dry run take the most time in the cybersecurity exercise life cycle. When these parts are completed, the exercise can be executed; however, finding the right people for the exercise was a challenge because if you want to conduct a blue team exercise, you need a red team or vice versa. To address this issue, we added automation in the execution part as well, so our proposed exercises could be executed in the below ways. *Automatic* In automatic execution, attacker and defender actions can be specified for testing different cybersecurity scenarios in an automated manner. We used agent-based techniques for this purpose, in which we can inject attacker and defender agents within the exercise infrastructure. This agent follows the requirements specified in the DSL to execute attacker and defender actions. *Hybrid* In hybrid execution, an automated agent can emulate an attacker or defender against a human team in a cybersecurity exercise. In a hybrid execution, an adversary team's requirement is removed, making the cybersecurity exercise life cycle less reliant on human input and reducing the inefficiencies related to finding human teams. *Manual* In manual execution, a normal cybersecurity exercise is conducted in which all participants are human. Depending on the training requirements, the proposed system supports the manual execution of cybersecurity exercises. In manual execution, both red and blue teams consist of human participants who perform a cyber-attack and defense within the exercise infrastructure.

4.2.4. Evaluation

Most operational cybersecurity exercises are evaluated using flags. Flags are a textual string that the participants must capture from a system to receive a score. A different variation of this method is called dynamic scoring. Time is taken to capture the flag, and the number of times the flags are captured are also taken into account for awarding higher and lower scores. We could easily integrate such a scoring mechanism into our system. However, we opted for more systemic evaluation methods, which are given below. *Log Analysis* We have collected the command line history of exercise participants, which can be used to analyze the participants' capability and what type of skills they have. We plan to train an AI model for this purpose and use it to classify exercise participants' skill sets based on the *cyber kill chain*. Because we are still collecting data from the exercises and working on this part, this is not included in this paper. *Surveys* We used pre and post-exercise surveys to identify any skill improvements of the exercise participants and get qualitative data about the exercises. Researchers have previously used these methods (Moore et al., 2017) for such purposes.

4.2.5. Repetition

In this phase, the feedback from the surveys is analyzed, and problems are identified in the scenarios, including whose solutions were incorporated in the next iterations of the exercise.

4.3. Full system workflow

We have presented the whole system workflow in Fig. 8 which we discussed in Section 4.2. In the workflow, different parts of the cybersecurity exercise life cycle are presented in different colors. The system uses our scenario language to model the scenario in a coherent, logical model that is platform independent. The logical model is used to verify different scenario properties, and if they fulfill the scenario requirement, then the platform-independent model is transformed into platform-dependent artifacts. These artifacts create the network topology, inject vulnerabilities, generate traffic, and emulate various exercise teams. After this, the scenario is manually tested and automatically verified from the attacker

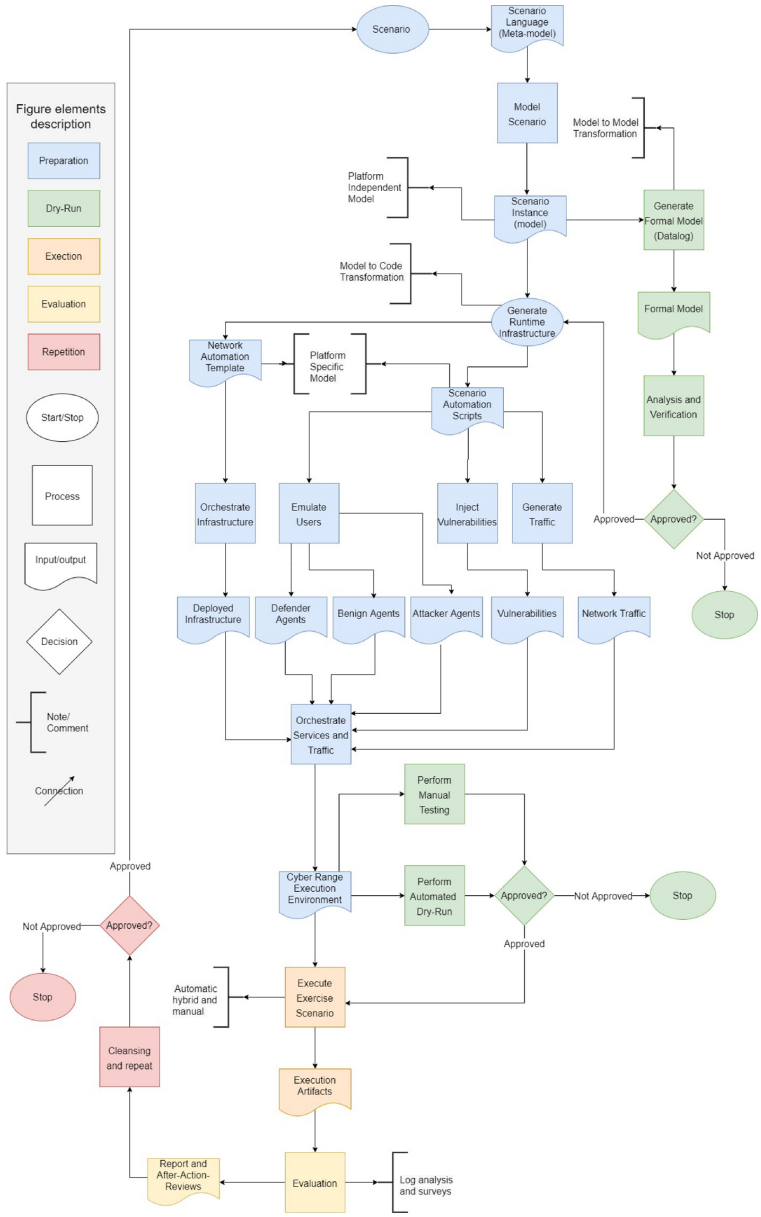


Fig. 8. Cyber security exercise operation workflow.

agent, and if it satisfies the scenario specification, then the exercise is executed. After exercise execution, relevant data are collected for exercise experience improvement in the next exercises.

The concept defined in the our scenario language instance is presented in concrete syntax for the orchestrator to understand and generate the necessary artifacts. The concrete syntax is JSON *Java script object notation* representations, which is specifically chosen because of its excellent capability to represent different models into objects with minimum or no changes in the code data structure. The details of the concrete syntax with respect to the specific concept is presented below:

Our scenario language input is transformed into emulated artifacts, and these artifacts can be used together or independently in five different processes for the execution of the cybersecurity exercise life cycle. The orchestrator is developed using the.Net framework, in which C# played a major role; secondary components of the orchestrator were developed using python scripts, whose calls were controlled by the C# base program. This modular design approach allowed us to include many existing system automa-

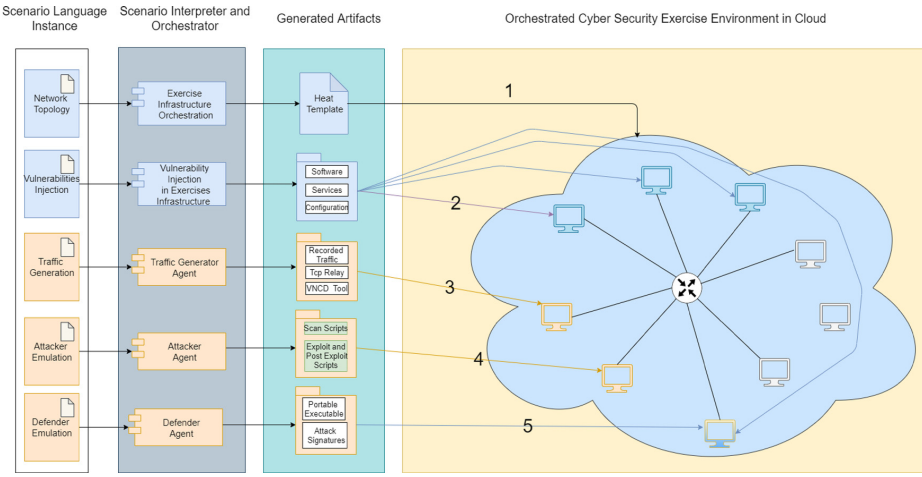


Fig. 9. Cyber security exercise operation orchestrator.

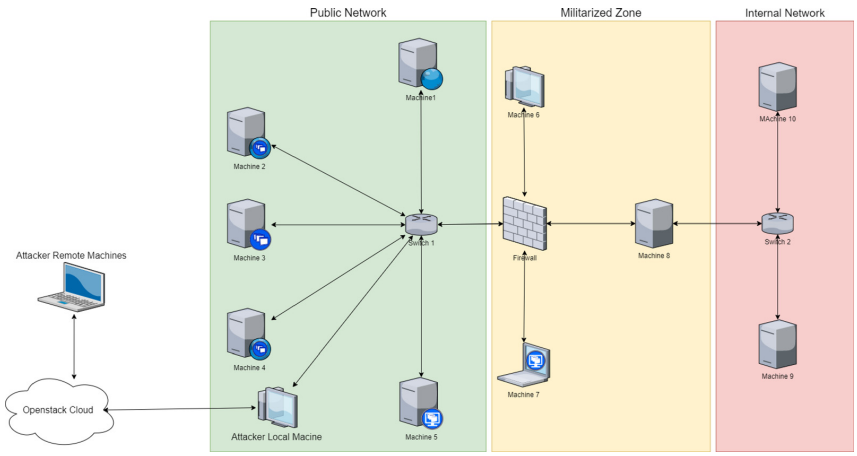


Fig. 10. Case study scenario.

tion techniques inside a single platform that provided much flexibility from scenario generation to scenario execution in a semiautonomous manner. The logical operation of the scenario orchestrator is presented in Fig. 9. The details of each component are given below.

Although some technical functions in the cybersecurity exercise life cycle are still manual, such as manual testing and survey, they are included to increase the exercise quality from a human perspective.

5. System implementation

The system is developed in a highly modular way, and different modules are used to automate different phases of the cybersecurity exercise lifecycle. Each module can run independently, or all the modules can run as a whole through a common API that uses our developed DSL for the exercise environment orchestration. The system is implemented in a web application that provides the interface to the developed API. It can take input from the developed DSL for orchestrating different cybersecurity operations roles. The website can be deployed in a local environment or on the cloud. The developed solutions require Openstack base cloud for performing infrastructure provisioning. While Vulnerability injection, attacker and defender emulation, and traffic generation can be done on a system supporting standard SSH protocol.

A variety of programming languages were used for the development of the application. The front end of the application was developed in Asp.net. Similarly, the API was developed in C#; however, the API runs multiple Python, Bash, and HEAT scripts in the back end. Package managers were used to installing vulnerabilities in Linux-based systems, while a silent install technique was used to install vulnerable software on Windows-based machines. In the case of the configuration and services, SSH-based automation techniques were used to make the system vulnerable. The source code of the developed orchestrator application can be found here NCR (2021). The repository contains pre-requisite information and detailed installation instruction for its easy deployment.

The system has three interfaces that are used to provide input and platform access. The first interface is for the administrator; the administrator can design and deploy customized cyber ranges hav-

ing access to detailed network configuration and vulnerability injection modules. The second interface is for a teacher or a coach who need to deploy a scenario in a very short amount of time, so the teacher or a coach can specify the type of vulnerabilities and the number of machines that are needed to be deployed for the scenario and the things deployed automatically. The final interface is for students who need to access the platform for practicing cybersecurity skills. They can access the platform and have access to multiple predefined scenarios that they can deploy by themselves and practice in the environment.

If we want to start the exercise from scratch, we can use all the modules that are presented in Fig. 9. We can perform (1) scenario modeling and orchestration for deploying exercise infrastructure, (2) vulnerabilities' injection and verification, (3) traffic generation, and (4) attacker and (5) defender agents for executing the exercise scenarios. If the infrastructure is already present, the implemented system can be used to inject vulnerabilities based upon the given requirement. Additionally, if the infrastructure is vulnerable, the developed solution can generate an attack model to verify those vulnerabilities. This modular system gives us the flexibility to adapt to various cybersecurity scenarios for dynamic cybersecurity exercises. The system module's usage for modeling an exercise scenario is presented in the form of a case study. It highlights the different modules developed during the research and their usage for conducting this research work.

6. Case studies

The competition organizers gave the scenario requirements, in which they requested two scenarios: a penetration testing scenario and an attack and defenses scenario. For the penetration testing scenario, the competition organizers gave the following scenario requirements:

1. The scenario should represent the IT infrastructure of a small- or medium-sized organization.
2. The scenario should have vulnerabilities that are exploitable in a particular amount of time
3. The scenario should be suitable for participants with various skill set levels.

We created a sample scenario description, which is provided in Appendix A and a rough network topology presented in Fig. 10 to translate the high-level requirements in to low-level technical artifacts based on our experience for conducting such exercises (Yamin and Katt, 2019; Yamin et al., 2018). We presented these ideas to the competition organizers and after their feedback and approval, we used it for the penetration testing scenario. We will used this network topology, as presented in Fig. 10, to showcase that a scenario can be modeled and orchestrated on a virtualized environment in the cloud. The scenario was logically verified and tested by an attacker agent, and the scenario also incorporated a defender agent in one of the scenario machines, which was included to add the friction and make the scenario more realistic. This case study was used to evaluate the *efficiency* and *autonomy* offered by our proposed solution.

In the second case study, the competition organizers provided the machines required for an attack/defense scenario with the diagram of the network topology. The requirement from the organizers was the following:

1. Deploy the machines in identical isolated networks for the first phase of the attack/defense scenario.
2. Update the network topology in the second phase of the attack/defense scenario so that the deployed networks can be interconnected.

```
<digit>          ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<letter>         ::= a | b | c | ... | y | z
<integer>        ::= <digit> { <digit> }
<string>         ::= " <char> { <char> } "
<char>           ::= <letter>  |  <digit>
<IP-Address>     ::= <Class-Address>"."<VLAN-Location>"."
                     <Device-Code>"."<Node>
<Class-Address>  ::= 001 to 255
<VLAN-Location>  ::= 001 to 255
<Device-Code>    ::= 001 to 255
<Node>           ::= 001 to 255
<CIDR>           ::= <Class-Address>"."<VLAN-Location>"."
                     <Device-Code>"."<Node>"/"<Range>
<Range>          ::= 0 to 24
```

Listing 1. Defining the basic variables.

```
<Subnet>       ::= <SubnetName> <CIDR> <NetworkID>
<SubnetName>   ::= <string>
<CIDR>         ::= <CIDR>
<NetworkID>    ::= <string>

<Machine>      ::= <MachineName> <OS> <Key> <Depends>
<MachineName>  ::= <SubnetName>
<OS>           ::= <string>
<Key>          ::= <string>
<Depends>      ::= <string>
```

Listing 2. Defining infrastructure.

```
<Vuln>                 ::= <MachineIP> <MachineUserID>
                           <MachineUserPassword> <OS>
                           <Vulnerability> <Parameter>
<MachineIP>            ::= <IP-Address>
<MachineUserID>        ::= <string>
<MachineUserPassword>  ::= <string>
<OS>                   ::= <string>
<Vulnerability>        ::= <string>
<Parameter>            ::= <string>
```

Listing 3. Defining vulnerabilities.

Because this scenario involved integrating external machines and updating network topology at run time, we used the *adaptability* to evaluate our proposed solution.

6.1. Preparation

6.1.1. Formal scenario modeling and analysis

We used our developed scenario modeler and verifier tool for the preparation of the scenario presented in Fig. 10. The scenario description is presented in Appendix A. The formal model generated by our tool is presented in Appendix B. The formal model of the scenario allowed us to verify different scenario properties. A Datalog analysis engine execution allowed us to logically verify different scenario properties, such as which machines present in the network were directly or indirectly accessible to the attackers. This allowed us to identify different edge cases like attacker accessibility without machine exploitation so that we could update the scenario before emulated network deployment. In Appendix B of the formal scenario model, we present the process of verifying a condition. An example of Datalog[1] query execution for identification of an attacker accessibility to different subnets is presented in Listing 7:

Similarly, in another example presented in Appendix B of the formal scenario model, we used the Datalog analysis engine exe-

[1] Definition in pydatalog ref:https://sites.google.com/site/pydatalog/home.

```
<Agent>                 ::= <ToolName> <AgentIP>
                            <AgentUserID>
                            <AgentUserPassword>
                            <Target> <Argument>
<ToolName>              ::= <string>
<AgentIP>               ::= <IP-Address>
<AgentUserID>           ::= <string>
<AgentUserPassword>     ::= <string>
<Argument>              ::= <string>
<Target>                ::= <IP-Address>
```

Listing 4. Defining attacker agent behavior .

```
    <Agent>::= <AgentIP> <AgentUserID>
    <AgentUserPassword> <OS>
    <Parameter>
    <AgentIP>               ::= <IP-Address>
    <AgentUserID>           ::= <string>
    <AgentUserPassword>     ::= <string>
    <OS>                    ::= <string>
    <Parameter>             ::= <string>
```

Listing 5. Defining defender agent behavior.

```
<Agent>                 ::= <AgentIP> <AgentUserID>
                            <AgentUserPassword> <Argument>
                            <Target>
<AgentIP>               ::= <IP-Address>
<AgentUserID>           ::= <string>
<AgentUserPassword>     ::= <string>
<Argument>              ::= <string>
<Target>                ::= <IP-Address>
```

Listing 6. Defining traffic generator behavior.

```
Query
---------------------
CanReach(Attacker1,Y)

Output
---------------------
***Public***
Machine1
Machine2
Machine3
Machine4
Machine5
***Demilitarized Zone***
Machine6
Machine7
Machine8
***Internal***
Machine9
Machine10
```

Listing 7. Analysis of an attacker that can reach other machines through direct and indirect links in different subnets like public, demilitarized zone, and internal.

cution for identifying which hosts were vulnerable to specific vulnerabilities and were reachable by attacker machines. This allowed us to determine the probable attacks' paths based on attacker capabilities and remove any edge cases where an attacker could not exploit the machines present in the logical representation of the exercise environment. A sample execution of the machines that were exploitable by a particular attacker capability is presented in Listing 8:

When different scenario properties are verified then the infrastructure is orchestrated.

6.1.2. Infrastructure orchestrator

The infrastructure orchestrator takes the JSON input from our scenario language and transforms it into HEAT templates. The

```
Query
---------------------
CanReach(Attacker1,Y) &Vulnerable(Y,BufferOverFlow)

Output
---------------------
Machine4
Machine7
```

Listing 8. Analysis of a host *Machine1* that can reach machines vulnerable to *BufferOverflow* vulnerability.

```
[
  {
    "Subnet 1": {
      "Name": "Public",
      "CIDR": "10.10.0.0/24",
      "NetworkID": "e18b412c-75c0-44a3-a326-708659d04152"
    },
    "Machine 0": {
      "Name": "Linux1",
      "OS": "721b1bc5-430e-44b9-89e3-45c92f3617fb",
      "key": "test",
      "Depends": "Public"
    }
  }
]
```

Listing 9. Concrete syntax for infrastructure generation.

HEAT templates were used to deploy the infrastructure using Openstack orchestration API. It should be noted that the required operating system images for the exercise are needed to be uploaded on Openstack before running the HEAT template. The orchestrator is currently only generating the exercise infrastructure in Openstack, but it is possible to transform our scenario language instance to other cloud orchestration technologies.

An example of the infrastructure is presented in Listing 9:

6.1.3. Vulnerability injector

The vulnerability injector can inject three types of vulnerabilities into the deployed infrastructure: software, services, and configuration. For software, the vulnerability injector reads the executable of the vulnerable program from the local drive and uses SSH to move it to a specified remote machine and then install it using different OS automation techniques. For services, the vulnerability injector reads a docker container file containing vulnerable services and moves it through SSH to the specified remote machine and deploys it automatically using different OS automation techniques. For configuration, a set of predefined bad configurations are integrated on the orchestrator, which can be specified in our scenario language and employed on remote machines using SSH. The configuration ranges from setting a user with a weak password to open directory shares for exploiting different vulnerabilities.

An example of our developed scenario language instance for vulnerability injection is presented in Listing 10:

In "Vuln 1," the vulnerability is "VulnerableProgram," and the parameter is "BufferOverflow.exe." This code will take the buffer overflow vulnerable program of SDL orchestrator machine and deploy it over a remote machine with the specified IP address using SSH and OS automation techniques.

In "Vuln 2," the vulnerability is "WeakPassword," and the parameter is "root2,toor." This code will create a new user account "root2" with "toor" as a password on a remote machine using an SSH connection and OS automation techniques.

In "Vuln 3," the vulnerability is "DockerInject," and the parameter is ""docker run -d -p 80:80 -p 3306:3306 -e MYSQL_Pass=mypass\vulnerables/."" It will take "web-dvwa.tar" from the orchestrator machine and automatically deploy the docker on the remote machine using a SSH connection and OS automation techniques.

Table 1
Vulnerability type, property and parameter mapping.

Vulnerability Type	Vulnerability Property	Parameter Property	Description
Software	VulnerableProgram	IntgratedHomePro.exe	Name of executable
Software	VulnerableProgram	Icecast.exe	Name of executable
Software	VulnerableProgram	WingFTP.exe	Name of executable
Service	DockerInject	docker run -d -p 80:80 -p 3306:3306 -e MYSQL_Pass=\"mypass\" vulnerables/	Service command to be run or deploy and its parameters
Service	EnableTelnet	Install-WindowsFeature -name Telnet-Client	Service command to be run or deploy and its parameters
Service	EnableRDP	Invoke–Command –Computername “server1”, “Server2” –ScriptBlock {Set–ItemProperty –Path “HKLM:\System \CurrentControlSet \Control \Terminal Server” –Name “fDenyTSConnections” –Value 1}	Service command to be run or deploy and its parameters
Configuration	WeakPassword	root,toor	Usename and password
Configuration	DisableFirewall	NetSh Advfirewall set allprofiles state off	Disabling security service
Configuration	EnableLocalShare	net share Docs=E:\Documents /grant:everyone,FULL	Changing local drive access settings

```
[
  {
   "Vuln 1": {
      "MachineIP": "192.168.81.128",
      "MachineUserID": "root",
      "MachineUserPassword": "toor",
      "OS": "Windows",
      "Vulnerability": "VulnerableProgram",
      "Parameter": "BufferOverflow.exe"
    },
    "Vuln 2": {
      "MachineIP": "192.168.81.130",
      "MachineUserID": "root",
      "MachineUserPassword": "toor",
      "OS": "Linux",
      "Vulnerability": "WeakPassword",
      "Parameter": "root2,toor"
    },
    "Vuln 3": {
      "MachineIP": "192.168.81.128",
      "MachineUserID": "root",
      "MachineUserPassword": "toor",
      "OS": "web-dvwa.tar",
      "Vulnerability": "DockerInject",
      "Parameter": "docker run -d -p 80:80 -p 3306:3306 -e
      ↪  MYSQL_Pass=\"mypass\" vulnerables/"
    }
  }
]
```

Listing 10. Concrete syntax for vulnerability injection.

The orchestrator contains a mapping list between potential values of the *vulnerability* property and the vulnerability type such that the orchestrator understands the type of the vulnerability that needs to be injected by reading that property value. Consequently, the orchestrator expects a specific information in the parameter property. Table 1 shows a sample keys of these parameters and the vulnerability types they represent, including *software, services* and *configuration* vulnerability types. The complete list of the orchetrator include over 800 vulnerabilities related to both Windows and Linux environments. This list is constantly expanding with new vulnerabilities. For example, the *vulnerability* property `VulnerableProgram` is mapped to the *software vulnerability* type, and indicates a program that contains a software problem (c.f. Table 1). In this case, the *parameter* property indicates the name of the vulnerable executable.

If we want to allow multiple vulnerabilities of the same type, then we add two vulnerabilities with the same *vulnerability* property and different *parameter* properties. For example, if we want to allow two “bufferoverflow” software vulnerabilities in same machine, (1) we define two vulnerabilities, whose *vulnerability* property is `VulnerableProgram`, and (2) each one refer to different buffer overflow executable to be deployed, e.g., `bufferoverflow1.exe` and `bufferoverflow2.exe`. The rest of the values like *MachineIP*, *MachineUserID*., will remain same in case both vulnerabilities are being deployed on the same machine. It should be noted that this simple classification of vulnerabilities can be mapped to other types of classifications, like CWE and CVE, however this is out of scope of this work.

6.1.4. Attacker agent

The attacker agent is a Kali Linux machine deployed within the exercise infrastructure. The Kali Linux machine is controlled through SSH by the orchestrator and performs the steps specified in our scenario language. These steps include performing network scanning, launching actual exploits, and post-exploitation. Different automation techniques have been used to achieve this process. One such technique for automating Metasploit is presented in Fig. 11. Here, we used Metasploit resource scripts to specify and pre and post-attacker steps during exploitation. In Fig. 11 vulnserver.rb is the Metasploit exploit, which is then automated in step 1 and transformed into vul.rc, which is the Metasploit resource script. The resource script is sufficient for launching the attack, but we integrated post-exploitation steps in it to mimic a real attacker. In Step 2, gather.rc is integrated, which emulates post-exploitation steps such as capturing network information, as indicated in Fig. 11. It should be worth mentioning that there can be multiple attacker agents in the scenario, with each one performing different types of attacks independent of each other.

An example of our scenario language for attacker behavior emulation in the scenario is presented in Listing 11:

6.1.5. Defender agent

The defender agent is a portable executable that can be injected into the machine that is present in the scenario environment. A configuration file is also injected with the agent, and the CSV file contains a list of actions that an attacker can perform and a list of reactions against those actions that the defender agent can take. For a Windows-based environment, the defender agent analyzes Windows security event logs to identify the attacker actions specified in the configuration file and execute the appropriate reaction for stopping the attacker. The defender agent’s internal working is based on our exploit chain detection algorithm (Yamin et al., 2019). Here, we looked for a particular Windows security event ID 4688 and analyzed the command line argument for the identification of malicious behavior, as we presented in (Yamin and Katt, 2018a). It should be worth mentioning that there can be multiple defender agents running with different configurations within the scenario, reacting to different types of attacks independent of each other.

An example of our scenario language for defender behavior emulation in the scenario is presented in Listing 12:

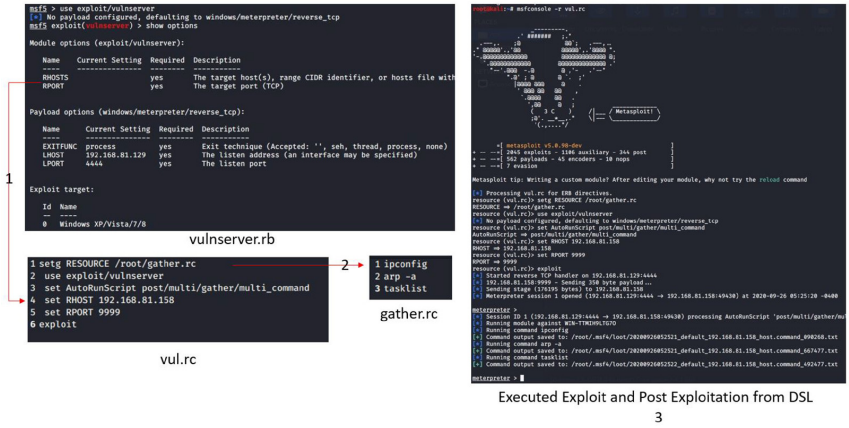


Fig. 11. Automated exploitation and post exploitation using metasploit.

```
[
  {
    "ActiveScan": {
      "AgentIP": "192.168.81.128",
      "AgentUserID": "root",
      "AgentUserPassword": "toor",
      "Argument": "SV",
      "Target": "192.168.81.130"
    },
    "BruteForce": {
      "AgentIP": "192.168.81.128",
      "AgentUserID": "root",
      "AgentUserPassword": "toor",
      "Argument": "SSH",
      "Target": "192.168.81.130"
    },
    "MetaSploit": {
      "AgentIP": "192.168.81.128",
      "AgentUserID": "root",
      "AgentUserPassword": "toor",
      "Argument": "Vulnserver",
      "Target": "192.168.81.130"
    }
  }
]
```

Listing 11. Concrete syntax for attacker behavior emulation.

```
[
  {
   "Defender 1": {
      "MachineIP": "192.168.81.132",
      "MachineUserID": "root",
      "MachineUserPassword": "toor",
      "OS": "Windows",
      "Parameter": "Actions1.csv"
    },
    "Defender 2": {
      "MachineIP": "192.168.81.134",
      "MachineUserID": "root",
      "MachineUserPassword": "toor",
      "OS": "Windows",
      "Parameter": "Actions2.csv"
    }
  }
]
```

Listing 12. Concrete syntax for defender behavior emulation.

6.1.6. Traffic generator

The traffic generator is a Kali Linux machine deployed within the exercise infrastructure. The Kali Linux machine is controlled through SSH by the orchestrator and performs the steps specified in our scenario language. These steps included replaying already captured network traffic and exercising specific email generation

```
[
  {
    "TcpRelay": {
      "AgentIP": "192.168.81.128",
      "AgentUserID": "root",
      "AgentUserPassword": "toor",
      "Argument": "Traffic.pcap",
      "Target": "Null"
    },
    "VncBot": {
      "AgentIP": "192.168.81.130",
      "AgentUserID": "root",
      "AgentUserPassword": "toor",
      "Argument": "Userbehavior.vdo|toor",
      "Target": "192.168.81.133",
    }
  }
]
```

Listing 13. Concrete syntax for traffic generation.

to emulate benign user behavior. For emulating benign users, Vncdotool (vnc, 2020) is used, enabling us to mimic user behavior in the GUI over VNC-enabled remote machines using a prerecorded VNC session. This added an extra layer of realism within the cybersecurity exercise environment. It should be noted that there can be multiple traffic generators in the scenario, generating different types of traffic independent from each other.

An example of our scenario language for the traffic generation and user behavior emulation in presented in Listing 13.

6.2. Dry run

The attacker agent developed during the research has two uses. Besides its role during the execution of the exercise, it can also perform a dry run on the developed infrastructure to test and verify that everything is working as expected. Some of the attacker agent's functionality tested during the dry run is presented in the form of the logs at Appendix C. Because our attacker agent is a Kali Linux machine, it can perform a network scan both actively and passively to check and verify that the machines in the scenario are up and running. Listing 14 shows the log of a passive scan that identifies the running machine in the scenario. Listing 15 shows the log generated by launching a brute-force attack on one of the deployed machines in the scenario. Similarly, Listing 16 shows the log of an automated exploit execution using Metasploit.

Additionally, the attacker agent can play the role of traffic generator using different Kali Linux functionalities. An example of an agent generating and replaying random network traffic is pre-

sented in Listing 13. As stated earlier, for emulating user behavior, VNCDOTOOL was used. A sample script for emulating a user to an open notepad and writing a few lines is presented in Listing 18. Moreover, a sample of the successful and unsuccessful email generation from our scenario language is presented in Listing 19. These functionalities can be used in different ways, depending on the scenario requirements.

6.3. Execution and evaluation

The platform was used for multiple qualifying rounds at NCSC 2020. The challenge had multiple rounds of qualifications, and in the first round, which was online, 150 people participated from all around Norway. Twenty-five individuals were invited for the second and third rounds, which were deployed by the proposed system. In the second round, 17 individuals participated in a penetration testing scenario. In the third and final round, the participants took part in an attack and defense scenario. The platform was evaluated for its efficiency in creating a realistic cybersecurity exercise infrastructure. Its capability to execute operations that are traditionally performed by human participants, along with its capability to adapt changes based on changing requirements. Moreover, it was also evaluated for improving the skill set of the cybersecurity exercise participants.

6.3.1. Platform deployment evaluation

The platform was evaluated by deploying the scenario presented in Fig. 10. The scenario had nine exploitable machines, one machine with no known vulnerability, and one machine running the defender agent. The machine running the defender agent had a similar vulnerability present in one of the exploitable machines. The machines were divided into three subnets: public, demilitarized zone, and internal network. Each team had five Kali machines present in the public network, which they could access over the internet using SSH. The scenario was deployed within five minutes using the orchestrator, but this does not accurately reflect the complexities involved in deploying such a scenario. For deploying the scenario, we needed to (1) collect the required operating system, vulnerable programs, services, and configuration details, which took a few hours to days, depending on the scenario requirement; (2) upload the required operating system on the cloud in the form of RAW images, which took a few minutes to hours depending on the internet speed; (3) specify the scenario according to the scenario language, which took a few minutes to hours depending on scenario complexity; (4) deploy the scenario on the cloud using the proposed system, which also took a few minutes to hours depending on scenario complexity; and (5) verify scenario properties using the emulated dry run, which took a few minutes to hours depending on scenario complexity. When all the perquisites for deploying and testing the scenario were fulfilled, our scenario language was able to provide the functionality and was very efficient in deploying the scenario. We then evaluated the proposed solution based on a set of five qualitative matrices: efficiency, usability, completeness, flexibility, scalability, and adaptability.

6.3.2. Efficiency

As stated earlier, multiple factors are involved in the efficiency of deploying cybersecurity scenarios. Consider the case of cooking a dish as an example of those factors. First, you must gather all the ingredients and then follow a recipe to make a dish. Here, we can measure the efficiency with respect to time in three ways: (1) cooking time, (2) the time required to gather the ingredients, and (3) the time required to grow the ingredients. The standard way of measuring how fast a dish is made is based on the cooking time, so we are ignoring the time required to gather the necessary components and artifacts for deploying the scenario and are only measuring the time our scenario language took to deploy the scenario, which was approximately five minutes. This figure can vary greatly because we deployed the scenario on a highly customized cloud infrastructure (Ope, 2021) that is specifically optimized for such infrastructure orchestration. Technically, the cloud infrastructure comprised 608 CPUs, 5.5 TB RAM for general purpose and 84 CPUs, 1792 GB RAM, 2 Tesla v100, 2 Tesla a100 for GPU-accelerated workloads with 133 TiB of total SSID storage. We suspect that the deployment efficiency result will be different in different cloud infrastructures, but we will investigate this in future research. *Applicability* Applicability measures whether the deployed scenario is employable for conducting operational cybersecurity exercises without manual tuning of the infrastructure. We tested the developed orchestrator in two different cybersecurity exercises that involved the top cybersecurity talent present in Norway, finding that the deployed scenario's performance was up to par with similar systems (Leitner et al., 2020; Russo et al., 2020; Vykopal et al., 2017a). The deployed infrastructure faced some failures during the exercises like some services failing to respond after continuing attacks; however, these issues were mitigated on the spot to ensure smooth running of the exercises.

Accuracy Accuracy measures whether the deployed scenario fulfilled the requirements specified in our scenario language accurately, such as the network topology and vulnerabilities present in the machines. Our developed scenario language fulfilled the technical requirement for deploying the infrastructure, injecting vulnerabilities, and executing a dry run as specified; the experimentally validated details of which are presented in Section 6.2 and Section 6.3.5.

6.3.3. Adaptability

Adaptability refers to the capability of developed solutions to accept and accommodate changes from different sources and implement the exercise scenario. We used another qualifying round for NCSC to check this capability. This qualifying round was an attack/defense scenario in which the vulnerable machine was developed by one of our colleagues using another source code vulnerability injector solution. Our colleague shared the four vulnerable machines with us, and using our scenario language, we deployed the scenario in an attack/defense network topology. Five teams participated in the attack/defense scenario, and they were assigned five identical isolated networks with four vulnerable machines each. The attack/defense scenario had two parts: in the first part, the teams had to patch the vulnerabilities in their assigned network. In the second part, the teams' networks were interconnected and were tasked to exploit the vulnerabilities in the other teams' machines. This changing network topology, while accommodating unknown machines, was used to verify the adaptability of our developed system. The deployed attack/defense scenario on Openstack is presented in Fig. 12.

Flexibility Flexibility measures the capability of the deployed scenario to accept changes after deployment. We developed the scenario orchestrator in a modular way, in which the infrastructure was independent of the vulnerability injection steps. This enabled us to inject new vulnerabilities after the scenario had been deployed. We consider this a highly useful feature because it enabled us to use the same network topology with different types of vulnerabilities for individuals with different skill sets. Moreover we could change the network topology, add additional machines with new vulnerabilities, and launch new attacks to make the scenario more dynamic based on the given requirements. *Scalability* Scalability measures whether the deployed scenario is expandable and can accommodate additional teams in the scenario. There were a total of 15 machines that were allocated to one team, but there were a total of five teams involved in NCSC, so the scenario was

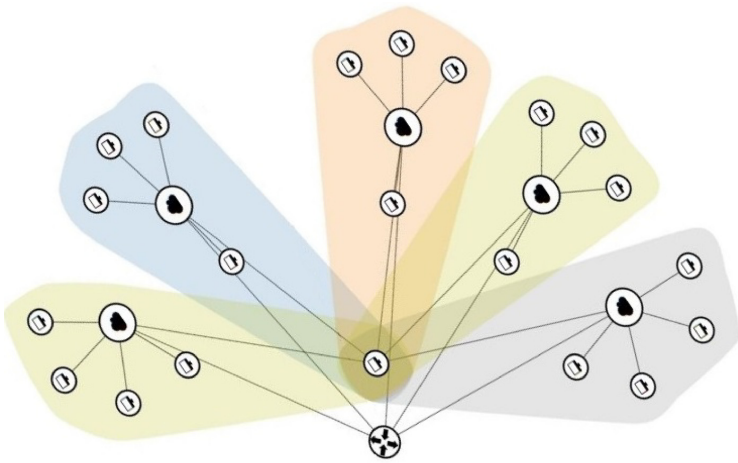


Fig. 12. Deployed Attack/defense scenario.

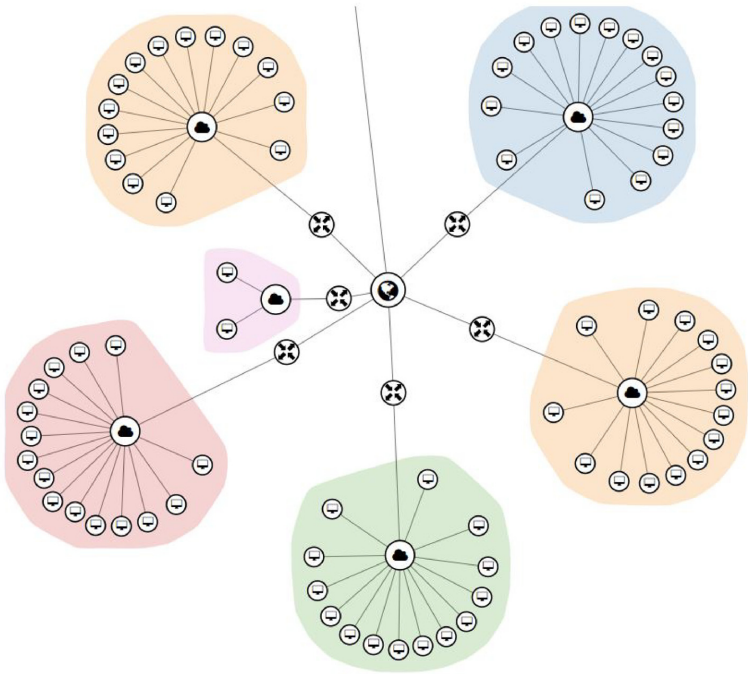


Fig. 13. Deployed penetration testing scenario.

replicated within five minutes to instantiate 75 virtual machines, which enabled us to assign a completely segregated network to each team. The fully deployed five networks comprising 75 machines that implemented the case study scenario are presented in Fig. 10 and can be seen in Fig. 13:

Because we conducted this research to perform cybersecurity exercises that are scalable for a large number of participants, we used the developed tool to create and deploy the exercise infrastructure for one of the cybersecurity courses taught at NTNU. The course had 84 students, and the infrastructure was required for a four-week-long red and blue team exercise. The infrastructure included nearly 400 machines with 19 teams and one research network and was deployed in approximately 37 minutes. The deployment time was noted form the Openstack *SystemUsageData* which includes the timestamps for when the *Stack CREATE started* and when the *Stack CREATE is completed* Sys (2021). It should be noted that each network was deployed as a separate HEAT template, and we added a one-minute break after each network deployment to manage Openstack API calls. The deployed network topology is presented in Fig. 14.

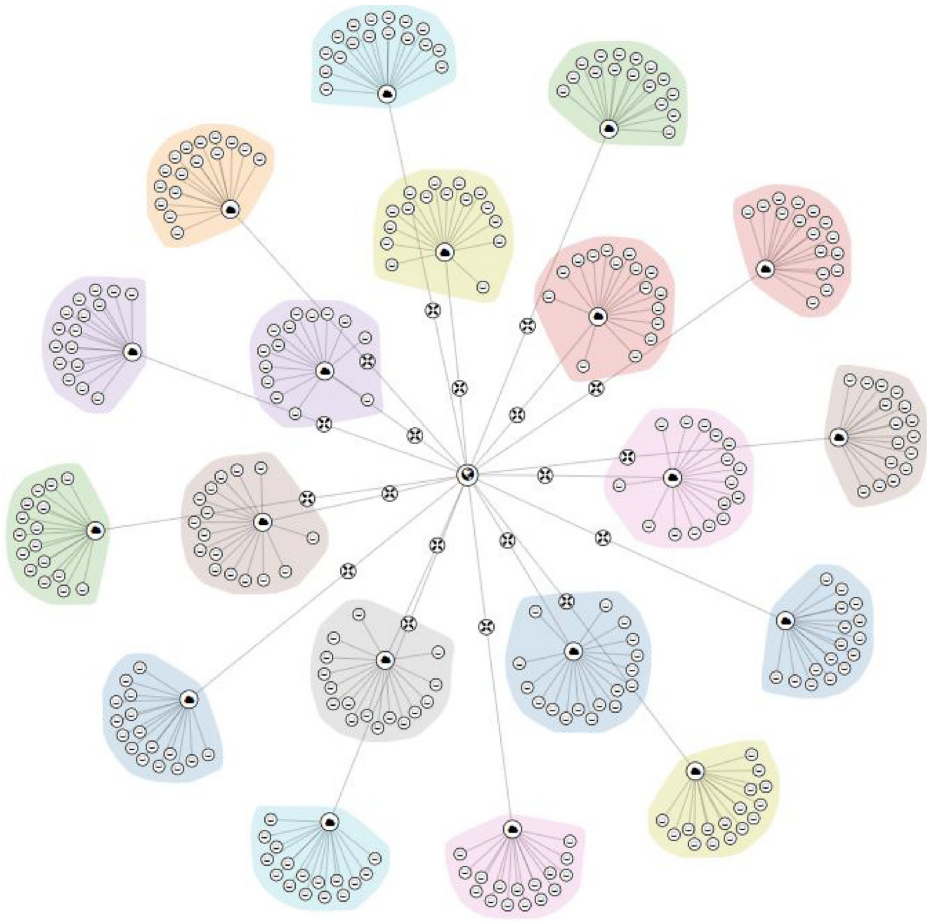


Fig. 14. Deployed red and blue team cybersecurity exercise scenario. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

6.3.4. Autonomy

Autonomy refers to the capability of a developed solution to perform tasks that have been traditionally performed by a human. We used the developed solution to perform an emulated dry run automatically on the scenario infrastructure, as discussed in Section 6.2. This provided us with the capability to replace humans in the dry run phase of cybersecurity exercises. Second, two machines in the scenario had the same vulnerability, but one machine was running the proposed defender agent. Five teams played the scenario, and two teams were able to exploit the vulnerability and get the flag on the machine that was not running the defender agent. The teams did not have any knowledge about a defender agent running on the system and were incentivized to exploit the machine to get additional points, as stated in Appendix A. With this experiment, we concluded that the defender agent was working as expected and could be a useful addition to cybersecurity exercises to add friction for increased realism.

6.3.5. Preliminary skill improvement evaluation

The skill improvement was measured in the final round of NCSC, in which 17 individuals, who qualified from the nationwide CTF of Norway, participated. The participants aged between 16–25, more than half of the participants were high school and university students, while others were employed in different public and private sectors. They participated in the form of randomly assigned groups in the CTF consisting of 3–4 individuals each. We employed a multitude of quantitative and qualitative methods for identifying skill improvements in exercise participants that were highlighted in Maennel (2020). First, we conducted pre and post-exercise surveys to identify any skill improvement in the individuals. This methodology was employed in a case study where individuals measured specific skill set levels over the passage of time (Moore et al., 2017). In summary a total of 7 technical skills were measured that were required to solve the challenges present in the exercise. The pre and post self-classification of skill level were presented in Fig. 15 and details of the results are provided in Section 6.3.5.1 and 6.3.5.2.

Secondly, we used CTF instrumentation suggested in Chothia and Novakovic (2015); however, each group was assigned their own separate exercise infrastructure and unique flags were injected manually to avoid *Flag Plagiarism*. Additionally, we asked the exercise participants to provide a *Penetration Test Report* to evaluate their performance in a qualitative manner. The

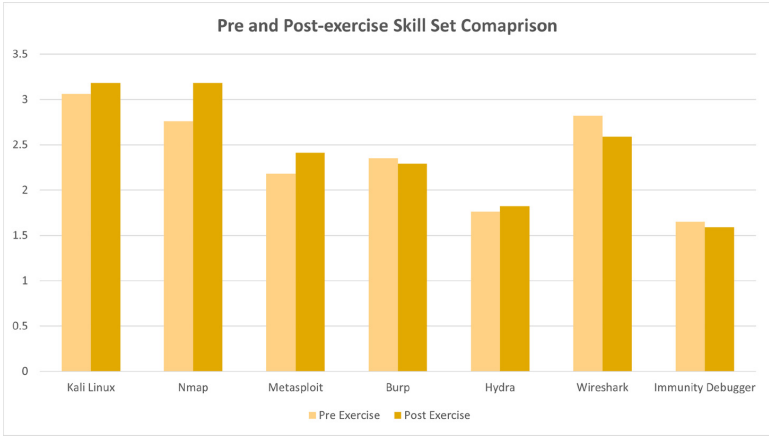


Fig. 15. Self-classification of skill level Pre and Post Exercise.

You classify yourself as a low/medium/high skilled individual in cyber security?

	Response Total	Response Percent
low	5	29%
medium	6	35%
high	6	35%
Total Respondents	17	100%

Fig. 16. Self-classification of skill level.

How many CTF you previously played

	Response Total	Response Percent
1-5	2	12%
5-10	8	47%
>10	7	41%
Total Respondents	17	100%

Fig. 17. Experience in CTF competitions.

survey questions are presented in Appendix D, and the results of the survey and CTF instrumentation are given below:

Pre-exercise Survey For the first question, we asked the participants about their self-classification of skill sets. The majority of the participants had medium and high cybersecurity skills, 35% each, while 29% reported that they had low cybersecurity skills. Details of the participants' answers about their self-classification of skill level are presented in Fig. 16:

The second question we asked was about the participants' experiences in CTF competitions. The question was intended to get information that could be used to measure the CTF quality in post-exercises surveys. Most participants (47%) replied that they had participated between in five and 10 CTFs before, 41% participated in more than 10 CTFs, while only 12% participated between one and five CTFs. Experience in CTF competitions is presented in Fig. 17:

The third question we asked was rating their skill-specific cybersecurity tools from 1 to 5, where 1 is the lowest and 5 is the highest value. These tools included Kali Linux, Nmap, Meatsploit, Burp, hydra, Wireshark, and Immunity Debugger. Most of the participants were familiar with Kali Linux, with a mean score of 3.06 out of 5, while most were not familiar with Immunity Debugger, with a mean score of 1.65 out of 5. Details of their pre–exercise skill set response related to specific tools are presented in Fig. 18.

Post-Exercise Survey After the pentest scenario, we conducted a post-exercise survey to measure the skill improvement of the scenario participants. The first question we asked was related to skill improvement with respect to specific tools. The participants reported that they saw skill improvements in Kali Linux, Nmap, Metasploit, Burp, and Hydra but did not see any improvements in their Wireshark and Immunity Debugger skills. Details of their post–skill set responses related to specific tools are presented in Fig. 19.

The second question we asked was related to their overall skill improvement, in which 24% reported that they had a definite skill improvement, while 41% reported that they saw some skill improvement after playing the scenario. Here, 35% reported that they did not have any skill improvement, which was expected because 35% represented the six seniors who were already highly skilled. Details of their post–exercise overall skill improvement are presented in Fig. 20.

Rate your skills in different cyber security tools from 1 to 5, where 1 is the lowest value and 5 is the highest value:

	1	2	3	4	5	Response Total	Response Average
Kali linux	5,88% (1)	23,53% (4)	29,41% (5)	41,18% (7)	0% (0)	17	3,06
Nmap	23,53% (4)	17,65% (3)	23,53% (4)	29,41% (5)	5,88% (1)	17	2,76
Metasploit	41,18% (7)	23,53% (4)	11,76% (2)	23,53% (4)	0% (0)	17	2,18
Burp	41,18% (7)	11,76% (2)	23,53% (4)	17,65% (3)	5,88% (1)	17	2,35
hydra	58,82% (10)	17,65% (3)	17,65% (3)	0% (0)	5,88% (1)	17	1,76
Wireshark	11,76% (2)	41,18% (7)	17,65% (3)	11,76% (2)	17,65% (3)	17	2,82
Immunity Debugger	76,47% (13)	5,88% (1)	5,88% (1)	0% (0)	11,76% (2)	17	1,65
					Total Respondents		17

Fig. 18. Pre-exercise skill set specific to tools.

Rate your skills in different cyber security tools from 1 to 5, where 1 is the lowest value and 5 is the highest value:

	1	2	3	4	5	Response Total	Response Average
Kali linux	0% (0)	23,53% (4)	41,18% (7)	29,41% (5)	5,88% (1)	17	3,18
Nmap	0% (0)	35,29% (6)	29,41% (5)	17,65% (3)	17,65% (3)	17	3,18
Metasploit	23,53% (4)	41,18% (7)	17,65% (3)	5,88% (1)	11,76% (2)	17	2,41
Burp	47,06% (8)	17,65% (3)	0% (0)	29,41% (5)	5,88% (1)	17	2,29
hydra	52,94% (9)	29,41% (5)	5,88% (1)	5,88% (1)	5,88% (1)	17	1,82
Wireshark	29,41% (5)	23,53% (4)	23,53% (4)	5,88% (1)	17,65% (3)	17	2,59
Immunity Debugger	76,47% (13)	5,88% (1)	5,88% (1)	5,88% (1)	5,88% (1)	17	1,59
					Total Respondents		17

Fig. 19. Post-exercise skill set.

Did you havve any skill improvement after playing the CTF?

	Response Total	Response Percent
Yes	4	24%
No	6	35%
Somewhat	7	41%
Total Respondents	17	100%

Fig. 20. Post-exercise skill improvement.

How realistic was the CTF compare to other CTF you played before? Give it rating from 1 to 5, where 1 indicates the lowest value and 5 indicates the highest value.

	1	2	3	4	5	Response Total	Response Average
Realistic level:	11,76% (2)	41,18% (7)	41,18% (7)	0% (0)	5,88% (1)	17	2,47
					Total Respondents		17

Fig. 21. Scenario realism.

The third question was related to the realistic level of the scenario; here, we asked the participants to rate the scenario's realism from 1 to 5. The scenario received a total rating of 2.47 out of 5. We consider this sufficient because most of the scenario was generated by our scenario language. Details of scenario's realism rating are presented in Fig. 21:

The last question we asked was related to scenario difficulty level, in which 59% of the participants considered the scenario to be difficult, 24% considered the difficulty level as medium, while 18% considered the difficulty level as easy. Details of the scenario difficulty level are presented in Fig. 22:

CTF Instrumentation We used CTFd CTF (2021) for scoring and instrumentation purposes. CTFd is widely used for scoring purposes for such exercises and is very user friendly and easy to deploy. The CTFd score board indicated how each group is performing in solving different challenges with respect to time as indicated in Fig. 23. This helped us to compare the skill set required to solve a challenge with the skill set stated by the participants in pre and post-exercises surveys.

For instance, if we analyze the task complemented by the one of the leading group we can see that it completed a task related to debugging of an application and brute force attack on another application as indicated in Fig. 24 where *NewBie* is for brute-force and *Debugger1* is for debugging. Other groups were struggling with those tasks, which was reflected in post exercise survey where participant self assessed their skills lower then compare to pre-exercise survey. Similarly majority of exercise participants were able to complete tasks that involved *Metasploit*, *Nmap* and *Kali Linux* which was positively reflected in post exercise survey. This helped us to measure the actual skill set compare to perceived skill set of the exercise participants. Finally a thematic analysis on the submitted penetration test reports were performed. Individual feedback for each participating group was given for focusing on particular skill set in future.

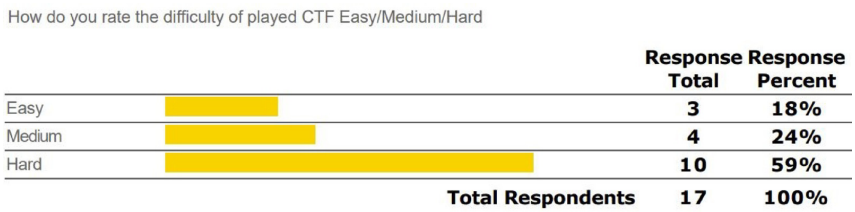


Fig. 22. Scenario difficulty level.

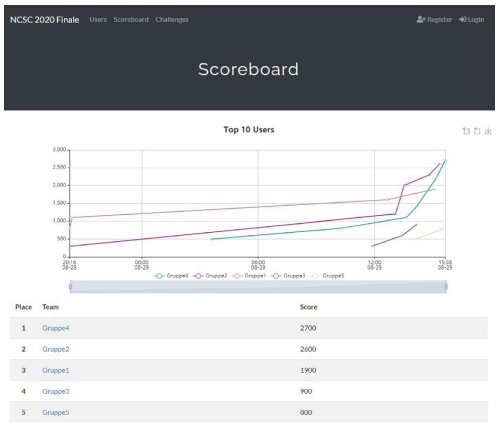


Fig. 23. Final Score Board of CTF.

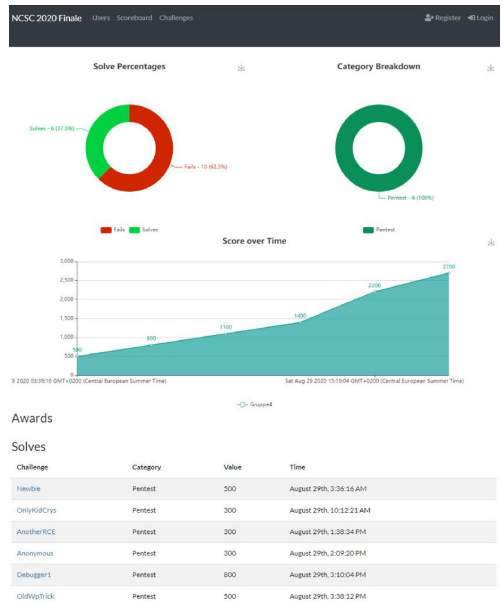


Fig. 24. Challenge Completion Analysis.

7. Discussion

In the course of this research, we investigated inefficiencies in the cybersecurity exercise lifecycle. Conducting cybersecurity exercises is a complex and challenging task involving a whole cycle which includes phases of *Preparation*, *Dry run*, *Execution*, *Evaluation* and *Repetition*. Each phase brings a new set of challenges and inefficiencies for cybersecurity exercise organizers. The challenges range from preparing operational exercise infrastructure to finding skilled individuals to perform dry run on it. Moreover, identifying skilled individuals to play the role of Red or Blue team members is also challenging during the exercise execution phase which adds additional inefficiencies. We proposed a system to address those inefficiencies and make the execution more efficient. The results and limitations of the prospered research work are presented below:

7.1. Results

We investigated 4 *Research Questions* in this work, *Question 1* was to identify systems and methods with which we can model and execute realistic cybersecurity exercise scenarios more efficiently. We proposed our developed orchestrator for addressing this question, we used model-driven engineering to develop a DSL that can use to specify cybersecurity exercise models involving realistic adversaries and traffic generation. Using our orchestrator, the DSL can computationally execute various functions of different teams present in the cybersecurity exercise environment to conduct exercises with minimal human effort, which makes the exercise execution efficient.

In *Research Question 2* we investigated the following query: *Is it possible to make cybersecurity exercise models adaptable to changing requirements?*. We addressed this question by conducting three cyber security exercises involving around 100 people. The exercises were conducted over the period of 6 months with changing requirements. The first exercise *Penetration testing* exercise was conducted to test the developed platform. The second *Attack and Defense* exercise was conducted to check the adaptability and flexibility of the developed platform, in which the developed platform incorporated machines from other sources. The third exercise which was a large *Red VS Blue* team exercise was conducted to test and validate the scale-ability and performance of developed autonomous agents.

In *Research Question 3* we investigated *What operations in cybersecurity exercises can be executed autonomously to reduce dependability on human teams?*. Our work indicated that in terms of automation, White, Green, and partially Red teams that are involved in the *Preparation* and *Dry run* phases are automated completely. For exercise execution, forensic traces for a “Red vs Blue” team exercise can be created, which removes the need for a human Red team in such scenarios. Similarly, autonomous defense agents can be injected into the machines present in the network to prevent the human attacker to achieve their objectives in a penetration testing scenario. A separate case study highlighting the internal working of attack and defense agents will be published soon. We are currently working on an AI model to analyze cybersecurity

skills based on the participants' bash history. We are in the advanced stages of its development but lack sufficient data for proper training of the AI model. To address this, we are conducting cybersecurity exercises, collecting relevant data from them, and planning to publish a separate study.

In *Research Question 4* we investigated *How much do such exercise scenarios improve the skills of cybersecurity exercise participants?*. Our work indicated that the developed cybersecurity exercise execution platform has a positive impact on exercise participants. The participants reported skill improvement on 5 out 7 tested technical skills which was co-related with the type of challenges completed in the exercises. The results are positive, yet not definitive, and further studies are required for better assessment of the skill improvement.

7.2. Limitations

Although the proposed system addresses the problems in conducting cybersecurity exercises efficiently and realistically, nevertheless it has few limitations. One of the first limitations of the proposed system is in vulnerability injection. The proposed system uses its own classification of vulnerabilities based upon operational requirements of the software, services, and configuration, which isn't commonly used in standards like CVE and CWE. We will address this in our future work by integrating other classification standards into the vulnerability classification of the vulnerability injection function. Secondly, for the evaluation, only 17 people participated and provided their feedback on the system for qualitative evaluation. However, we would like to emphasize that the 17 survey participants were selected from the pool of 150 people who competed in a nationwide CTF in Norway. The 17 participants were actively participating in the study and the exercises. On the other hand, one of ENISA report from 2021 Tow (2022) indicated that while the number of participants in cyber security competitions can be high, the active participants are a small portion of the total number (ca. 11%). We are targeting very specific skill sets and incorporating a large group is very challenging, especially during the pandemic. Similarly, from the same report, it was also indicated that the average training group for different European countries is around 20 people. Based upon this information we believe that the data presented in the qualitative evaluation surveys might not be conclusive, but it has meaningful insights. We would further like to highlight similar studies Abbott et al. (2015); Moore et al. (2017); Peker et al. (2016); Yamin et al. (2021) that involve 26, 30, 28, and 25 people, respectively.

8. Conclusion and future work

In the current work, we presented a cybersecurity exercise modeling and execution tool for cyber ranges. At the core of the tool, we have proposed a scenario language that unifies concepts of different operations present in cybersecurity exercises. This enabled us to computationally orchestrate their functionality. We used our tool to conduct multiple cybersecurity exercises in an efficient, adaptable, and autonomous manner, which resulted in decreasing the inefficiencies in the cybersecurity exercise life cycle. This will enable the utilization of operational cybersecurity exercises on a wide scale for education and training purposes, helping overcome the ever-growing cybersecurity skill shortage.

This research work resulted in the development of a DSL that can be used to model cybersecurity exercises scenarios. The modeled scenarios can be executed in an efficient and realistic manner using the orchestrator developed during this research. The proposed models are adaptable based upon changing cybersecurity exercise scenario requirements. This is achieved through the automation of various roles in cybersecurity exercises. The white team role is automated to specify the scenario requirements and automatic deployment of exercise infrastructure. The red team role is automated to perform dry runs and act as an automated adversary during exercise execution. While blue team role was automated to provide active defense during cybersecurity exercise execution. We evaluated the developed system by conducting multiple cybersecurity exercises and measured exercises participant skill improvement.

In the future, we are planning to expand the current version of our scenario language and add a new concept called "module." The "module" concept will specify a template of an organizational infrastructure, for example, an *bank* or an *internet service provider* ISP. When the "module" specification is given to our scenario language, the infrastructure will be provisioned for automatic deployment. One key feature that we are planning to integrate into such infrastructure provisioning is multi-cloud orchestration because we are going to emulate the whole infrastructure of an organization, which will be quite resource-demanding for single cloud deployment. This will also help us realistically model different segregated organizations on the internet for cybersecurity scenarios. Finally, we are planning to use TLA+ to model the attacker and defender steps during a cybersecurity exercise. This will enable us to perform a more formal analysis on different attack and defense strategies. We will conduct more in-depth case studies for autonomous cybersecurity exercise execution and then analyze such technology's effectiveness in training human attackers and defenders.

Declaration of Competing Interest

The authors declare no conflict of interest in publishing the article "Modeling and Executing Cyber Security Exercise Scenarios in Cyber Ranges".

Appendix A. Scenario Description

NCSC 2020 Penetration Test Scenario

A1. Good corp LLC

Good Corp LLC is a SME (Small and Medium Enterprise) working in the Management Consultancy area. It provides assistance to various government and non-government entities in making decisions that are economically feasible, environmentally sustainable and socially acceptable. Good Corp LLC is currently conducting a study that is focusing on economic benefits of increased Baltic sea area by forecasting success of recent DS (Delete Sweden) movement.

While the DS movement got allot of support from Nordic regions and is economically feasible and environmentally sustainable. However, it is not socially acceptable by some people, so they launched an anti-DS movement Ref: https://www.change.org/p/sweden-anti-delete-sweden.

But the anti-DS movement didn't get any significant public support, so a group of 5 social activist hackers decided to take things in their own hands. They planned to launch a cyber-attack on Good Corp LLC and tamper with the data on its servers in order to change its report that it is going to present to the government.

A2. Good corp LLC employees

Good Corp LLC is an equal opportunity organization and provide people from diverse backgrounds and environments to grow and achieve success. New employees come and go. Some learn new things while other earn. Due to current COVID situation Good Corp LLC is struggling to provide orientation sessions to new employees

which includes overview of company values and basic cybersecurity awareness. Mike is a young and passionate researcher who recently joined Good Corp LLC but missed the orientation sessions. Good Corp LLC is facing a lot of cybersecurity challenges as it is involved in very high-profile research, so it invited a cybersecurity firm to pentest their employees and infrastructure. The firm identified a lot of security issues in Good Corp LLC infrastructure. Some of them were related to very famous ransomware vulnerabilities while others were related system configuration and bad password policies. One shocking finding of the pentest was a data breach that was identified but thankfully it didn't contain some important information. Just some old file and custom software programs that Good Corp LLC uses in its day to day operations. The good thing that was identified in the pertest was that the CEO Jason was found to be very well secured while one of the mangers has no zero day vulnerabilities. The CEO decided to continue the operations of the company while patches were implemented and issues with data breach were resolved. Link to data breach:

A3. Rules

1. You are tasked to discover, exploit, analyze and report vulnerabilities in Good Corp LLC infrastructure.
2. Every system has a file flag.txt. Locate it and post the content of it in the scoreboard.
3. Take a screenshot of the flag content with IP address of exploited machine and use it as evidence in the report.
4. Bonus point for putting a file in CEO computer with content "DS is not recommended"
5. Not allowed to use any sort of DoS Attacks
6. Not allowed to share any information about the scenarios with anyone other than your teammates

Appendix B. Formal Scenario Model and Verification for the Use Case

Lets us consider this scenario in which 10 machines are connected to 3 sub-nets. In the scenario two subnets are directly connected *PUblic* and *MilitrizedZone* while a machine present in the network have dual network interface which provide access to subnet *Internal*. We can define the scenario model the following way.

1. **Defining Links**

Link Facts

$\%Specifying\ facts\ which\ Hosts\ are\ connected\ to\ which\ Network$
$Link('Machine1', 'Public')$
$Link('Machine2', 'Public')$
$Link('Machine3', 'Public')$
$Link('Machine4', 'Public')$
$Link('Machine5', 'Public')$
$Link('Public', 'DemilitarizedZone')$
$\%Specifying\ two\ Networks\ are\ directly\ connected$
$Link('Machine6', 'DemilitarizedZone')$
$Link('Machine7', 'DemilitarizedZone')$
$Link('Machine8', 'DemilitarizedZone')$
$Link('Machine8', 'Internal')$
$\%Specifying\ a\ Host\ is\ connected\ with\ multiple\ network\ interface$
$Link('Machine9', 'Internal')$
$Link('Machine10', 'Internal')$

To check which machines are connected to which network and can reach which machines we can define a new *Term CanReach* with variable *X*, *Y* and *Z*.

Link Clauses

$\%A\ clause\ which\ creates\ a\ bidirectional\ link\ between\ two\ Hosts\ X\ and\ Y$
$Link(X, Y) \leq Link(Y, X)$
$\%A\ clause\ to\ check\ direct\ link\ between\ two\ Hosts\ X\ and\ Y$
$CanReach(X, Y) \leq Link(X, Y)$
$\%A\ clause\ to\ check\ link\ between\ two\ Hosts\ X\ and\ Z\ via\ Host\ Y$
$CanReach(X, Y) \leq Link(X, Z)$

2. **Defining Vulnerabilities**

Vulnerability Facts

$\%Specifying\ facts\ which\ Hosts\ are\ vulnerable\ to\ which\ vulnerability$
$Vulnerable('Machine1', 'SSHBruteForce')$
$Vulnerable('Machine2', 'EasyFTPExploit't)$
$Vulnerable('Machine3', 'MS17-010')$
$Vulnerable('Machine4', 'BufferOverFlow')$
$\%A\ Host\ can\ be\ vulnerable\ to\ multiple\ vulnerabilities$
$Vulnerable('Machine4', 'MS17-010')$
$Vulnerable('Machine5', 'XXE')$
$Vulnerable('Machine6', 'AppacheExploit')$
$Vulnerable('Machine7', 'MS14-068')$
$Vulnerable('Machine7', 'BufferOverFlow')$
$Vulnerable('Machine8', 'RDPBrutforce')$
$\%A\ Host\ can\ have\ no\ known\ vulnerability\ as\ well$
$Vulnerable('Machine9', 'NoVulnerability')$
$Vulnerable('Machine10', 'EasyFTPExploit')$

To check which machine is connected to machine with a specific vulnerability e.g. *BufferOveflow* we can verify it by the following clause:

Vulnerability Clause

$CanReach('Attacker1', Y)\ \&\ Vulnerable(Y, 'BufferOverFlow')$

3. **Defining Capabilities**

Capabilities Facts

$\%Specifying\ which\ vulnerabilities\ are\ exploitable\ and\ which\ are\ defendable$
$Capability('SSHBruteForce', 'YES', 'NO')$
$Capability('WebExploit', 'YES', 'NO')$
$Capability('MS17-010', 'YES', 'YES')$
$Capability('BufferOverFlow', 'YES', 'NO')$
$Capability('MS14-068', 'YES', 'YES')$
$Capability('XXE', 'YES', 'NO')$
$Capability('AppacheExploit', 'YES', 'NO')$
$Capability('RDPBrutforce', 'YES', 'NO')$
$Capability('EasyFTPExploit', 'YES', 'YES')$
$Capability('NoVulnerability', 'NO', 'YES')$

To check which machine is connected to vulnerable machines which are not defendable by defender we can create the following clause:

Capabilities Clause

$Capability(V, 'YES', 'NO')\ \&\ CanReach('Attacker1', Y)\ \&\ Vulnerable(Y, V)$

4. Defining KillChain

KillChain Facts

%Specifying facts that which host is exploitable to different stages of CKC.
KillChain('Machine1', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')
KillChain('Machine2', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')
KillChain('Machine3', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')
KillChain('Machine4', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')
KillChain('Machine5', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')
KillChain('Machine6', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')
KillChain('Machine7', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')
KillChain('Machine8', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')
KillChain('Machine9', 'YES', 'YES', 'YES', 'YES', 'YES', 'NO', 'NO')
KillChain('Machine10', 'YES', 'NO', 'NO', 'NO', 'NO', 'NO', 'NO')

To integrate Cyber Kill Chain concepts to the model we can create the following clause:

KillChainClause

%A clause to check which specific Host is connected to Hosts
%in a network that are vulnerable and are not defendable
%and are exploititable to different stages of Cyber Kill Chain .
$Capability(V, 'YES', 'NO') \,\&\, CanReach('Mahine1', V) \,\&\, Vulnerable(Y, V) \,\&$
$KillChain(Y, 'YES', 'YES', 'YES', 'YES', 'YES', 'YES', 'YES')$

Appendix C. Dry Run Logs

```
Date and Time:12:20 PM
 17 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 1020
 ------------------------------------------------------------
   IP            At MAC Address     Count     Len  MAC Vendor
 ------------------------------------------------------------
 192.168.81.2    00:50:56:fc:37:72      3     180  VMware, Inc.
 192.168.81.130 00:50:56:29:2a:2e      1      60  VMware, Inc.
 192.168.81.1    00:50:56:c0:00:08     12     720  VMware, Inc.
 192.168.81.254 00:50:56:e0:48:87      1      60  VMware, Inc.
```

Listing 14. Sample log for passively checking network connection.

```
Date and Time:12:26 PM
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in
↪  military or secret service organizations, or for illegal
↪  purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2019-02-28
↪  15:25:16
[DATA] max 64 tasks per 1 server, overall 64 tasks, 5084 login
↪  tries (l:1/p:5084), ~80 tries per task
[DATA] attacking ssh://192.168.243.130:22/
[STATUS] 1022.00 tries/min, 1022 tries in 00:01h, 4191 to do in
↪  00:05h, 64 active
[22][ssh] host: 192.168.243.130   login: root   password: jason1
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 62 final worker threads did
↪  not complete until end.
Hydra (http://www.thc.org/thc-hydra) finished at 2019-02-28
↪  15:26:33
```

Listing 15. Sample log generated by launching a brute-force attack during dry run.

```
      .:okOOOkdc'           'cdkOOOko:.
    .xOOOOOOOOOOOc       cOOOOOOOOOOx.
   :OOOOOOOOOOOOOk,   ,kOOOOOOOOOOOOO:
  'OOOOOOOOkkkkOOOO: :OOOOOOOOOOOOOOO'
  oOOOOOOO.MMMM.oOOOoOOOOl.MMMM,OOOOOOOo
  dOOOOOOO.MMMMMM.cOOOOOc.MMMMMM,OOOOOOOx
  lOOOOOOO.MMMMMMMMM;d;MMMMMMMMM,OOOOOOOl
  .OOOOOOO.MMM.;MMMMMMMMMM;MMMM,OOOOOOO.
   cOOOOOO.MMM.OOc.MMMMM'oOO.MMM,OOOOOOc
    oOOOOO.MMM.OOOO.MMM:OOOO.MMM,OOOOOo
     lOOOO.MMM.OOOO.MMM:OOOO.MMM,OOOOl
      ;OOOO'MMM.OOOO.MMM:OOOO.MMM;OOOO;
       .dOOo'WM.OOOOocccxOOOO.MX'xOOd.
         ,kOl'M.OOOOOOOOOOOO.M'dOk,
           :kk;.OOOOOOOOOOOO.;Ok:
             ;kOOOOOOOOOOOOOOk:
               ,xOOOOOOOOOOx,
                 .lOOOOOOOl.
                    ,dOd,
                      .

       =[ metasploit v5.0.98-dev                          ]
+ -- --=[ 2045 exploits - 1106 auxiliary - 344 post       ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

Metasploit tip: View advanced module options with advanced

[*] Processing vul.rc for ERB directives.
resource (vul.rc)> setg RESOURCE /root/gather.rc
RESOURCE => /root/gather.rc
resource (vul.rc)> use exploit/vulnserver
[*] No payload configured, defaulting to
↪  windows/meterpreter/reverse_tcp
resource (vul.rc)> set AutoRunScript
↪  post/multi/gather/multi_command
AutoRunScript => post/multi/gather/multi_command
resource (vul.rc)> set RHOST 192.168.81.158
RHOST => 192.168.81.158
resource (vul.rc)> set RPORT 9999
RPORT => 9999
resource (vul.rc)> exploit
[*] Started reverse TCP handler on 192.168.81.129:4444
[*] 192.168.81.158:9999 - Sending 350 byte payload...
[*] Sending stage (176195 bytes) to 192.168.81.158
[*] Meterpreter session 1 opened (192.168.81.129:4444 ->
↪  192.168.81.158:49172) at 2020-10-09 04:05:26 -0400

meterpreter >
[*] Session ID 1 (192.168.81.129:4444 -> 192.168.81.158:49172)
↪  processing AutoRunScript 'post/multi/gather/multi_command'
[*] Running module against WIN-TTMIH9LTG7O
[*] Running command ipconfig
[+] Command output saved to: /root/.msf4/loot/
↪  20201009040527_default_192.168.81.158_host.command_553149.txt
[*] Running command arp -a
[+] Command output saved to: /root/.msf4/loot/
↪  20201009040527_default_192.168.81.158_host.command_236368.txt
[*] Running command tasklist
[+] Command output saved to: /root/.msf4/loot/
↪  20201009040527_default_192.168.81.158_host.command_168109.txt
```

Listing 16. Sample log generated from launching a Metasploit exploit from our scenario language.

```
Date and Time:5:10 AM
Actual: 136045 packets (13973936 bytes) sent in 3.39 seconds
Rated: 4114044.4 Bps, 32.91 Mbps, 40052.79 pps
Flows: 0 flows, 0.00 fps, 136045 flow packets, 0 non-flow
Statistics for network device: eth0
        Successful packets:        136045
        Failed packets:            0
        Truncated packets:         0
        Retried packets (ENOBUFS): 0
        Retried packets (EAGAIN):  0
```

Listing 17. Sample log for network traffic generation.

```
key tab
pause 1
key enter
pause 1
type notepad
pause 1
key enter
pause 1
type "Testing Automation"
pause 1
key enter
type "which is very basic proof of concpet"
pause 1
key enter
type "showing a user behavior on notepad"
pause 1
key enter
type "in a very limited manner"
pause 1
key enter
key alt
key enter
```

Listing 18. Sample VDO configuration for emulating a user behavior.

```
Date and Time:8:22 AM
Jan 22 11:20:39 kali sendemail[3707]: Email was sent successfully!


Date and Time:2:42 AM
Jan 23 05:40:29 kali sendemail[1722]: ERROR => Received:
↪          554 5.7.1 [2] Message rejected under suspicion of SPAM;
↪  https://ya.cc/1IrBc 1579776029-NTJI8wwzTr-eSWu5qsR
```

Listing 19. Sample log for successful and unsuccessful phishing email generation.

Appendix D. Survey Questions

D1. Pre-Exercise

1. You classify yourself as a Senior or Junior?*
   - Senior
   - Junior
2. You classify yourself as a low/medium/high skilled individual in cybersecurity?*
   - Low
   - Medium
   - High
3. How many CTF you previously played*
   - 1 to 5
   - 5 to 10
   - More then 10
4. Rate your skills in different cybersecurity tools from 1 to 5, where 1 is the lowest value and 5 is the highest value:*
   - Kali linux
   - Nmap
   - Metasploit
   - Burp
   - hydra
   - Wireshark
   - Immunity Debugger

D2. Post-Exercise

1. Rate your skills in different cybersecurity tools from 1 to 5, where 1 is the lowest value and 5 is the highest value:*
   - Kali linux
   - Nmap
   - Metasploit
   - Burp
   - hydra
   - Wireshark
   - Immunity Debugger
2. How do you rate the difficulty of played CTF Easy/Medium/Hard*
   - Low
   - Medium
   - High
3. How realistic was the CTF compare to other CTF you played before? Give it rating from 1 to 5, where 1 indicates the lowest value and 5 indicates the highest value.
4. Did you havve any skill improvement after playing the CTF?*
   - Yes
   - No
   - Somewhat

CRediT authorship contribution statement

Muhammad Mudassar Yamin: Conceptualization, Resources, Investigation, Writing – original draft. **Basel Katt:** Supervision, Writing – review & editing.

References

Abbott, R.G., McClain, J.T., Anderson, B.R., Nauer, K.S., Silva, A.R., Forsythe, J.C., 2015. Automated Performance Assessment in Cyber Training Exercises. Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Almroth, J., Gustafsson, T., 2020. Crate exercise control–a cyber defense exercise management and support tool. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, pp. 37–45.

Beuran, R., Pham, C., Tang, D., Chinen, K.-i., Tan, Y., Shinoda, Y., 2017. Cytrone: an integrated cybersecurity training framework. SCITEPRESS–Science and Technology Publications 157–166.

Beuran, R., Tang, D., Pham, C., Chinen, K.-i., Tan, Y., Shinoda, Y., 2018. Integrated framework for hands-on cybersecurity training: cytrone. Computers & Security 78, 43–59.

Ceri, S., Gottlob, G., Tanca, L., et al., 1989. What you always wanted to know about datalog(and never dared to ask). IEEE Trans Knowl Data Eng 1 (1), 146–166.

Chang, W., Zhao, S., Wei, R., Wellings, A., Burns, A., 2019. From java to real-time java: a model-driven methodology with automated toolchain. In: Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems, pp. 123–134.

Chothia, T., Novakovic, C., 2015. An offline capture the flag-style virtual machine and an assessment of its value for cybersecurity education. 2015 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 15).

Ctfd : The easiest capture the flag platform. 2021. (Accessed on 10/13/2021) https://ctfd.io/.

Datalog: Deductive database programming. 2020. (Accessed on 09/30/2020) https://docs.racket-lang.org/datalog/index.html.

Eckroth, J., Chen, K., Gatewood, H., Belna, B., 2019. Alpaca: Building dynamic cyber ranges with procedurally-generated vulnerability lattices. In: Proceedings of the 2019 ACM Southeast Conference, pp. 78–85.

Edgar, T.W., Manz, D.O., 2017. Research methods for cyber security. Syngress.

Hutchins, E.M., Cloppert, M.J., Amin, R.M., 2011. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Leading Issues in Information Warfare & Security Research 1 (1), 80.

Jones, R.M., O'Grady, R., Nicholson, D., Hoffman, R., Bunch, L., Bradshaw, J., Bolton, A., 2015. Modeling and integrating cognitive agents within the emerging cyber domain. In: Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Vol. 20. Citeseer.

Kramer, F., Starr, S., Wentz, L., 2006. Actions to enhance the use of commercial information technology (it) in department of defense (dod) systems. In: Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'05). IEEE, pp. 8–pp.

Leitner, M., Frank, M., Hotwagner, W., Langner, G., Maurhart, O., Pahi, T., Reuter, L., Skopik, F., Smith, P., Warum, M., 2020. Ait cyber range: Flexible cyber security environment for exercises, training and research. In: Proceedings of the European Interdisciplinary Cybersecurity Conference, pp. 1–6.

Lloyd, J.W., 2012. Foundations of logic programming. Springer Science & Business Media.

Libvirt: The virtualization api. 2021. (Accessed on 02/25/2021)https://libvirt.org/.

Lord, D., 1985. Worldwide networking for academics. Data Processing 27 (5), 27–31.

Maennel, K., 2020. Learning analytics perspective: Evidencing learning from digital datasets in cybersecurity exercises. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, pp. 27–36.

Mirkovic, J., Benzel, T.V., Faber, T., Braden, R., Wroclawski, J.T., Schwab, S., 2010. The deter project: Advancing the science of cyber security experimentation and test. In: 2010 IEEE International Conference on Technologies for Homeland Security (HST). IEEE, pp. 1–7.

Moore, E., Fulton, S., Likarish, D., 2017. Evaluating a multi agency cyber security training program using pre-post event assessment and longitudinal analysis. In: IFIP World Conference on Information Security Education. Springer, pp. 147–156.
Mitre att&ck®. 2020. (Accessed on 09/30/2020) https://attack.mitre.org/.
Ncr orchestrator application. 2021. (Accessed on 10/15/2021) https://tinyurl.com/4y57t3vb.
Naik, M., 2020. Petablox: Large-Scale Software Analysis and Analytics Using Datalog. Technical Report. GEORGIA TECH RESEARCH INST ATLANTA ATLANTA United States.
NIST, 2020. Cyber ranges. (Accessed on 11/09/2020) https://www.nist.gov/system/files/documents/2018/02/13/cyber_ranges.pdf.
Openstack at ntnu - skyhigh - ntnu wiki. 2021. (Accessed on 03/24/2021) https://www.ntnu.no/wiki/display/skyhigh/Openstack+at+NTNU.
Opentosca. 2021. (Accessed on 02/25/2021) https://www.opentosca.org/.
Paul-pols—the-unified-kill-chain.pdf. 2020. (Accessed on 09/30/2020) https://www.csacademy.nl/images/scripties/2018/Paul-Pols---The-Unified-Kill-Chain.pdf.
Peker, Y.K., Ray, L., Da Silva, S., Gibson, N., Lamberson, C., 2016. Raising cybersecurity awareness among college students. Journal of The Colloquium for Information Systems Security Education, Vol. 4. 17–17
Pham, C., Tang, D., Chinen, K.-i., Beuran, R., 2016. Cyris: A cyber range instantiation system for facilitating security training. In: Proceedings of the Seventh Symposium on Information and Communication Technology, pp. 251–258.
Russo, E., Costa, G., Armando, A., 2018. Scenario design and validation for next generation cyber ranges. In: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA). IEEE, pp. 1–4.
Russo, E., Costa, G., Armando, A., 2020. Building next generation cyber ranges with crack. Computers & Security 101837.
Schmidt, D.C., 2006. Model-driven engineering. Computer-IEEE Computer Society-39 (2), 25.
Schreuders, Z.C., Shaw, T., Shan-A-Khuda, M., Ravichandran, G., Keighley, J., Ordean, M., 2017. Security scenario generator (SecGen): A framework for generating randomly vulnerable rich-scenario vms for learning computer security and hosting {CTF} events. 2017 {USENIX} Workshop on Advances in Security Education ({ASE} 17).
Systemusagedata - openstack. 2021. (Accessed on 10/12/2021) https://tinyurl.com/a8y3bed.
Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles - sae international. 2020. (Accessed on 09/25/2020) https://www.sae.org/standards/content/j3016_201806/.
Towards a common ecsc roadmap – enisa. 2022. https://www.enisa.europa.eu/publications/towards-a-common-ecsc-roadmap. (Accessed on 01/11/2022).
Vaishnavi, V.K., Kuechler, W., 2015. Design science research methods and patterns: innovating information and communication technology. Crc Press.
Vykopal, J., Ošlejšek, R., Čeleda, P., Vizvary, M., Tovarňák, D., 2017. Kypo cyber range: design and use cases. SciTePress.
Vykopal, J., Vizváry, M., Oslejsek, R., Celeda, P., Tovarnak, D., 2017. Lessons learned from complex hands-on defence exercises in a cyber range. In: 2017 IEEE Frontiers in Education Conference (FIE). IEEE, pp. 1–8.
vncdotool. pypi. 2020. (Accessed on 10/08/2020) https://pypi.org/project/vncdotool/.
What is infrastructure as code (iac)? | ibm. 2020. (Accessed on 09/25/2020) https://www.ibm.com/cloud/learn/infrastructure-as-code.
What is soar? security definition | fireeye. 2020. (Accessed on 09/25/2020) https://www.fireeye.com/products/helix/what-is-soar.html.
Weeden, B.C., Cefola, P.J., 2010. Computer systems and algorithms for space situational awareness: history and future development. Advances in the Astronautical Sciences 138 (25), 2010.
White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A., 2002. An integrated experimental environment for distributed systems and networks. ACM SIGOPS Operating Systems Review 36 (SI), 255–270.
Willems, C., Meinel, C., 2012. Online assessment for hands-on cyber security training in a virtual lab. In: Proceedings of the 2012 IEEE Global Engineering Education Conference (EDUCON). IEEE, pp. 1–10.
Yamin, M.M., Katt, B., 2018. Detecting malicious windows commands using natural language processing techniques. In: International Conference on Security for Information Technology and Communications. Springer, pp. 157–169.
Yamin, M.M., Katt, B., 2018. Inefficiencies in cyber-security exercises life-cycle: A position paper. CEUR workshop proceedings.
Yamin, M.M., Katt, B., 2019. Modeling attack and defense scenarios for cyber security exercises. In: 5th interdisciPlinary cyber research conference 2019, p. 7.
Yamin, M.M., Katt, B., Gkioulos, V., 2019. Detecting windows based exploit chains by means of event correlation and process monitoring. In: Future of Information and Communication Conference. Springer, pp. 1079–1094.
Yamin, M.M., Katt, B., Gkioulos, V., 2020. Cyber ranges and security testbeds: scenarios, functions, tools and architecture. Computers & Security 88, 101636.
Yamin, M.M., Katt, B., Nowostawski, M., 2021. Serious games as a tool to model attack and defense scenarios for cyber-security exercises. Computers & Security 110, 102450.
Yamin, M.M., Katt, B., Torseth, E., Gkioulos, V., Kowalski, S.J., 2018. Make it and break it: An IoT smart home testbed case study. In: Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control, pp. 1–6.
Yasuda, S., Miura, R., Ohta, S., Takano, Y., Miyachi, T., 2016. Alfons: A mimetic network environment construction system. In: International Conference on Testbeds and Research Infrastructures. Springer, pp. 59–69.
Zaber, M., Nair, S., 2020. A framework for automated evaluation of security metrics. In: Proceedings of the 15th International Conference on Availability, Reliability and Security, pp. 1–11.

Muhammad Mudassar Yamin is currently doing his Ph.D. at the Department of Information and Communication Technology at the Norwegian University of Science and Technology. He is the member of the system security research group and the focus of his research is system security, penetration testing, security assessment, intrusion detection. Before joining NTNU, Mudassar was an Information Security consultant and served multiple government and private clients. He holds multiple cyber security certifications like OSCE, OSCP, LPT-MASTER, CEH, CHFI, CPTE, CISSO, CBP.

Basel Katt is currently working as an Associate Professor at the Department of Information and Communication Technology at the Norwegian University of Science and Technology. He is the technical project leader of Norwegian cyber range. Focus of his research areas are: • Software security and security testing • Software vulnerability analysis • Model driven software development and model driven security • Access control, usage control and privacy protection • Security monitoring, policies, languages, models and enforcement

2.6 Detecting Windows Based Exploit Chains by Means of Event Correlation and Process Monitoring

Future of Information and Communications Conference (FICC) 2019
14-15 March 2019 | San Francisco

Detecting Windows Based Exploit Chains by Means of Event Correlation and Process Monitoring

Muhammad Mudassar Yamin
Norwegian University of Science and Technology,
Department of Information Security and Communication Technology
Gjøvik, Norway
muhammad.m.yamin@ntnu.no

Basel Katt
Norwegian University of Science and Technology,
Department of Information Security and Communication Technology
Gjøvik, Norway
basel.katt@ntnu.no

Vasileios Gkioulos
Norwegian University of Science and Technology,
Department of Information Security and Communication Technology
Gjøvik, Norway
vasileios.gkioulos@ntnu.no

This article presents a novel algorithm for the detection of exploit chains in a Windows based environment. An exploit chain is a group of exploits that executes synchronously, in order to achieve the system exploitation. Unlike high-risk vulnerabilities that allow system exploitation using only one execution step, an exploit chain takes advantage of multiple medium and low risk vulnerabilities. These are grouped, in order to form a chain of exploits that when executed achieve the exploitation of the system. Experiments were performed to check the effectiveness of developed algorithm against multiple anti-virus/anti-malware solutions available in the market.

Keywords—Exploit Chain, Event Correlation, Process Monitoring, Windows, process correlation

I. INTRODUCTION

Recently, the Pwn2own 2018 researchers introduced multiple Zero Day exploits, which were primarily based on a chain of multiple exploits for the exploitation of systems and services [1]. Traditional anti-virus and anti-malware software uses process monitoring and process isolation techniques for detection, according to suspicious process behavior pattern [2]. Yet, as we see in the Pwn2Own 2018 results, the researchers were able to break such process isolation and sandboxing process protection techniques. Examples of exploits that cannot be detected using traditional techniques are the *guest-to-host* exploits [3], and the macro-less DDE (dynamic data execution) in an MS office application [4]. In this article, we present a novel technique for the detection of such exploits using process execution monitoring my means of event correlation. The technique performs detection in a signature free and fully autonomous manner, using only the process names for monitoring and detection of exploitations. We use event correlation with respect to events extracted from process monitoring logs to create a chain of suspicious processes generated by the application to identify a detection. This article is organized in six sections. The first section introduces the problem, while in the second section we discuss related work and provide additional information about the problem background. The following sections present the proposed algorithm and initial experimentation results, while in the last sections we provide a discussion, future work and conclude the article.

II. RELATED WORK

The authors were not able to identify in the literature any viable existing technique for the detection of complex exploit chains such as *guest-to-host* exploits, while most cloud security vendors use defense in depth architectures to avoid security incidents involving *guest-to-host* exploits [5]. Existing techniques for securing a host from *guest-to-host* exploits use a multistep approach. Initially, an external process hook or agent is added in each virtual machine, which is then updated for malware and virus definitions from an external source [6]. Another technique used for securing virtual machines, relies on VMI (Virtual Machine Introspection) based process monitoring, for malware detection on a virtual machine from an external source [7]. Graph based event correlation (on the virtual machine) for anomaly detection using machine learning techniques [8]. The problem faced by existing techniques, is that they mostly focus on the protection of the virtual machine, without taking into account the new *guest-to-host* exploits, which exploit guest isolation using an exploit chain and allow the guest virtual machine to access the host operating system. Furthermore, in respect to the macro-less DDE in MS office applications [4], the research focus is on using malicious PowerShell commands for exploiting the system. For the detection of malicious PowerShell commands, researchers are currently using machine-learning techniques [9]. Yet, such existing detection techniques are vulnerable to the use of command line obfuscation for avoiding detection [10].

III. PROBLEM BACKGROUND

To further explain the problem a brief technical background is given.

A. Exploit Chains

In a normal IT security environment one vulnerability is enough to compromise the security of a system. However, due to continue system security improvements finding such vulnerabilities is becoming harder day by day. On the other hand low impact vulnerabilities are usually easy to find, researcher demonstrated multiple exploits which use these low impact vulnerabilities [4][5], and chain them together to compromise system security. To further explain the flow of a single exploit and an exploit chain we created a simple flow

chart for easy understating. Flow chart showing comparison of traditional exploits and an exploit chain flow is given in figure 1:

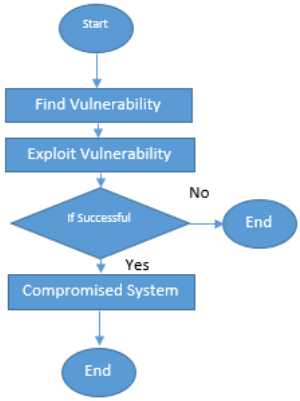


Fig (1) Example of traditional exploit with a single vulnerability

In comparison to a vulnerability that is exploited by a single exploit, in an exploit chain multiple vulnerabilities are involved. Each exploit uses the output of the previous to accomplish the objectives. A flow chart representation of exploit chain is seen in the figure 2:

Fig (2) Example of exploit chain with multiple vulnerability

Similar concepts exists in literature such as attack chain or attack paths which is set of possible steps that an attacker could take to compromise a system, involving multiple nodes on which exploitation is performed. In contrast, exploit chaining is the process of linking multiple vulnerabilities of one node which are present in a system and executing them in a specific order to compromise security.

B. Window Event logging Mechanism

The Microsoft Security Event logging mechanism is present in every new release of Windows since Windows XP. This event logging mechanism allows the identification of the type of computer events happening in Windows based systems when an exploit is executed. Researchers at JPCERT [15] provided details of such security events in there technical report. In this research paper we focus on Event ID 4688 [11]. Which is a Windows new process creation event. Each 4688 event contains the following fields

- **SubjectUserSid :** Security id of account from where the process is executed
- **SubjectUserName :** Account name from where the process is executed
- **SubjectDomainName :** Domain Name
- **SubjectLogonId :** Logon id of account from where the process is executed
- **NewProcessId :** Unique hexadecimal new process identifier
- **NewProcessName :** New process name executed by parent process
- **ProcessId :** Unique hexadecimal process identifier
- **CommandLine :** Command which is executed
- **TargetUserSid :** Security id of account on which process executed
- **TargetUserName :** User name
- **TargetDomainName :** Computer name
- **TargetLogonId :** Login id of account on which process executed
- **ParentProcessName :** Name of process which executes new process
- **MandatoryLabel :** Secure object control integrity label assigned to new process

From the information present in the fields of 4688 event we used *NewProcessId, ProcessId, TargetDomainName* in our detection algorithm. The *ProcessId* is a unique identifier issued by computer operating system to a running process. *NewProcessId* is a unique identifier issued by computer operating system to a process that is executed by another running process.TargetDomainName is the unique name of the computer on the domain.

C. Guest-to-host exploit

A recent report from SpiceWork [13] shows that server virtualization adoption reached 85% in comparison to 15% of physical IT infrastructure in 2017, as seen in figure 3.

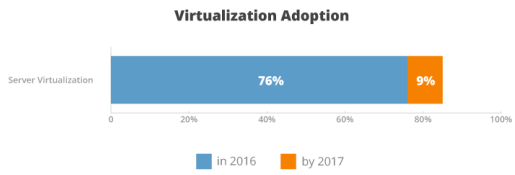


Fig (3) [13] Server virtualization trends

This trend leads security researchers to develop exploits that can break guest isolation and compromise the host machine. A list of few vulnerabilities is given below

- CVE-2017-4924: An out of bound memory corruption vulnerability in Vmware 12.x to 12.5.7 Implementation of SVGA(Virtual graphic card) allows attackers to execute code host system
- CVE-2017-4934: An heap buffer overflow vulnerability in Vmware 12.x to 12.5.8 Implementation of VMNET (virtual machine network) allows attackers to execute code host system
- CVE-2017-4936: An out-of-bounds read vulnerability in Vmware 12.x to 12.5.8 JPEG2000 parser in the TPView.dll allows guest to execute code or perform a DOS (Denial of Service) on the Windows OS.

A detailed list of *guest-to-host* escape vulnerabilities can be found online [14]. An example of these vulnerabilities is CVE-2017-4924 in which an out of bound memory corruption in vmwar-vmx.exe with incorrect memory mapping exists. This allows Data Execution Prevention bypass which leads to code execution on host from virtual machine.

Exploit writers were able to exploit this vulnerability and they created a POC (Proof of Concept) [17] for its exploitation. In the POC first the guest isolation is escaped by out of bound memory corruption and then CMD is executed by exploiting host Windows task registry. From CMD, PowerShell is executed to achieve remote shell level access on host. Schematically the exploit chain presented in figure 4.

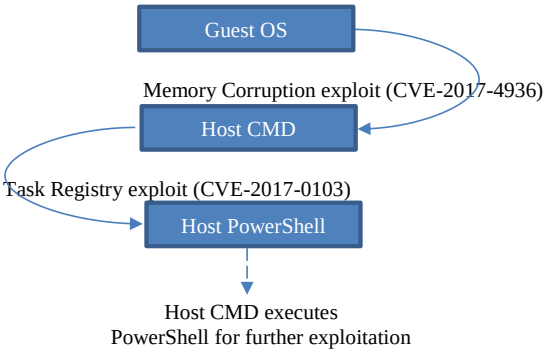


Fig (4) Guest to host escape exploit chain

Ideally the virtualization provide isolation between Guest OS and Host OS, where only the relevant services are shared as seen in the figure 5:

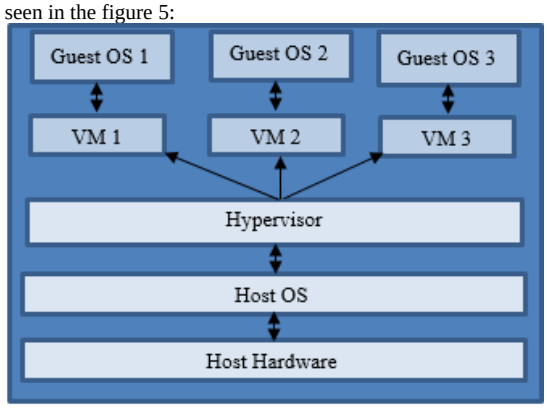


Fig (5) Isolated guest and host in virtualized environment

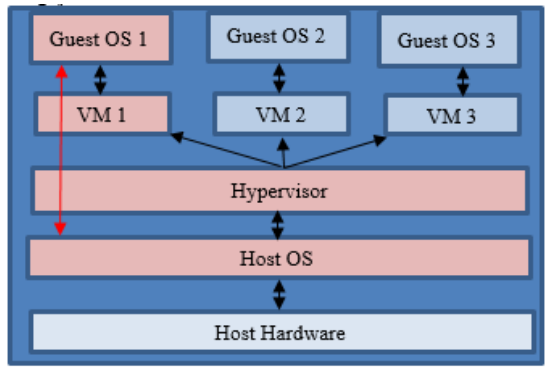


Fig (6) Broken isolation between guest and host

D. Macro-less DDE Attacks

To transfer data between different applications Windows provides the functionality of *Dynamic Data Execution*. The communication or COM Objects of Microsoft word and Microsoft excel, have public access to this DDE functionality. The functionality allows Microsoft Word and Excel to execute system commands legitimately. Exploit writers misused this functionality and were able to develop complex exploits such as macro-less DDE code execution [4]. It is also very difficult to detect with traditional detection techniques since the functionality is legitimate feature and is not blocked and patched by Microsoft [4]. Anti-virus and anti-malware solutions are using signature-based detection mechanism for the detection of macro-less DDE but the signature-based detection was also easily bypass able using command obfuscation techniques [9]. The exploit execution of macro-less DDE is similar to guest-to-host escape but in this case Microsoft Word or Excel is used to create exploit chain. First DDE on Microsoft Word or Excel is exploited which allows

the exploited process to use COM object in Windows and pass data related to secondary logon elevation vulnerability in windows through which CMD is executed. Now when the CMD process is started a command line argument containing malicious PowerShell script is passed to obtain a remote shell of the host a schematic repression of the explication chain is seen in figure 7:

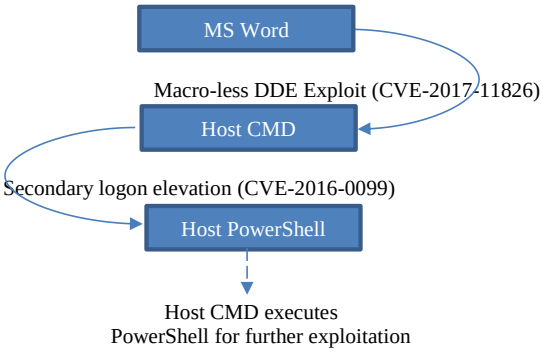


Fig (7) Guest to host escape exploit chain

IV. DETECTION METHODOLGH

The detection algorithm is developed by analyzing Windows security logs. Consider the following Windows security logs of Vmware *guest-to-host* Escape exploit. It breaks the Guest isolation, executes a CMD command on the host to run a PowerShell Exploit. The logs generated by the exploit can be seen in the figure 8:

Fig (8) Windows event logs generated from a *guest to host* exploit

After analyzing the logs a clear link is established between the processes generated by the exploit, as the ProcessID of a process is the NewProcessID of previous process involved in the exploit chain. We identified that by co-relating multiple events based upon the relation of ProcessID and NewProcessID we can create a process execution chain of the exploit. Accordingly a detection algorithm has been developed based on this finding. The proposed algorithm works in the following manner:

Exploit Chain Detector (ECD) Algorithm

Input: a list of ordered Windows event logs A; a list of process names to be monitored B
/* an event logs has the following attributes: *NewProcessId, ProcessId, ProcessName, TargetDomainName**/
/* B contains a list of process names that are executed after a vulnerability is exploited retrieved from report[1] [15] */
Output: a list of string stacks D, a Boolean represents if exploit chains are detected c
/* D will contain all exploit chains detected by the algorithm, and c is true if one chain is found*/
Initialization: create an empty event log a ; initialize c with the value false ; create integer m with initial value 0

```
1  for (i=0; i<Size(A); i++) do
2      if (A_i.ProcessId ∈ B) then
3          a=A_i
4          for (j=i; i<Size(A); j++) do
5              if (a. ProcessId == A_j.NewProcessId && a.TargetDomainName == A_j.TargetDomainName) then
6                  D_m.Push(a.ProcessName)
7                  a=A_j
8                  if(A_(j+m).NewProcessId==Null) then
9                      c=true
10                     m=m+1
11                 end if
12             end if
13         end for
14     end if
15 end for
```


1 (The list is created according to the JPCERT report Detecting Lateral Movement through Tracking Event Logs, which suggest the following processes for active tracking: cmd, powershell, regsvr32, rundll32, mshta)

The ECD (Exploit Chain Detector) algorithm requires only two inputs for execution. First is the security monitoring logs on which detection is performed *A*, and second is the list of process names that need to be monitored *B*. *A* is directly retrieved from host which contains individual events with multiple fields like ProcessId, NewProcessId ,ProcessName, TargetDomainName etc. *B* is the list of process names given by JPCERT[15] that are executed after a vulnerability is exploited. In output the algorithm returns whether an exploit chain is detected or not in a boolean variable c. If detected then it also shows the exploit chains in a stack *D*. For initializing the algorithm we need an empty event log *a*, an integer *m* with value 0 and c will be initialized with the value false..

When the algorithm starts processing it reads all the event logs available in *A*, then it start checking one by one if the ProcessName of an event in *A* is present in *B*. If a match is found the single event of *A* is stored in *a* and ProccessName is pushed to the stack D. Now a second comparison is performed on those events of *A* which are present after *a*, in the comparison ProcessId of *a* and ComputerName of *a* is compared with the ProcessId of the next event of *A* and ComputerName of that event in the coming logs. If a match is found ProccessName is pushed to a stack D and value of *a* is updated with current value of event at *A* .This process is performed until there is no NewProcessID in *A*. When this happens true value is assigned to *c* while the stack *D* contains the whole exploit chain. We calculated the algorithm complexity and it was

$$O(n \log n)$$

The algorithm complexity is good for detection of exploit chain in environment with small or medium amount of security logs data but in an environment with large amount of event log data the algorithm will take considerable amount of time for detection of exploit chains.

V. Implementation

We developed our proposed algorithm on a simple python based Windows logging mechanism. It is based on the standard pywin32 library presented at python library blog post [12], while the detection algorithm is built around this logging mechanism. The logs come in a recursive manner, as post exploitation is done after the initial exploitation with respect to time. We developed our detection algorithm POC on Microsoft Visual Studio 2017[2] on python environment 3.6. Our implementation contains the following primary functions.

1) Get-All-System-Events

This function takes all event logs from system which include application events, security events, setup events, system events and forwarder events and write them to separate files on disk.

2) Event-Parser

Event log parser read the event from the disk and parse them to individual readable events and forward it to next function *Get All Event Logs*

3) Get-All-Event Logs

Get-All-Event-Logs is the core function of our detection algorithm. It takes parsed events from *Event-parser*, then performs event process comparison and event corelation for the detetction of exploit chains.

To test the algorithm we run the developed tool on a Core i5 3320M 2.60 ghz system with 16 gb of RAM against 17098 Windows security events and two executed exploits *guest-to-host, macro-less DDE*. The Execution took 7.3s for the detection of the exploit chains, which can be seen in the figure 9:

Fig (9) Implemented algorithm process execution time and function calls

The implementation works without any malware, virus or malicious command signature for the detection of the exploit chain. We performed detailed experimentation on the developed algorithm to check the effeteness of our algorithm. The following section elaborates the experimental details and results:

VI. Experimentataion And Results

Two experiments were performed to check the effectiveness of the developed algorithm one is a *guest-to-host* exploit the other is a macro-less DDE exploit details of which are given below:

A. *Guest-to-host* exploit

1. Experimental Setup

We created our experimental setup on a 64-bit Windows 10 machine running on a VMware Workstation 12.5.5. A Guest Windows 10 operating system is installed on the Vmware. For detection comparison analysis we installed Bit Defender Home, Avira Home, Kasper Sky Home, Avast Home and Panda Security Suite on the Host OS and deactivated them.

2. Controlled Exploit Execution

We executed a *guest-to-host* proof of concept for CVE-2017-4924 [16], [17] on Guest windows 10 operating system. The exploit breaks the Guest isolation and executed an CMD on host machine and then executed PowerShell. Execution of exploit on Process Hacker can be seen in the figure 10:

2 https://www.visualstudio.com/

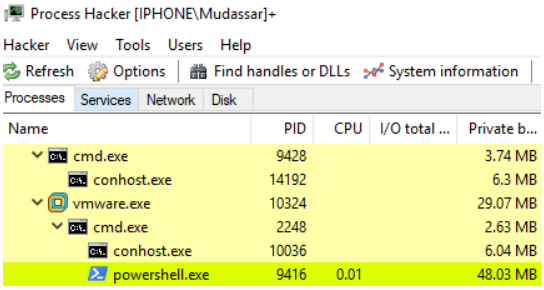


Fig (10) *Guest to host* exploit execution

3. Scenarios

We created two scenarios for comparison of our developed algorithm with different anti-virus/anti-malware solution available in the market. In the first scenario we executed the exploit only against our detection algorithm. In the second scenario we executed the exploit against different anti-virus/anti-malware solution available in the market. Details of which is given below:

a) Detection against developed algorithm

We executed our detection algorithm on the Windows 10 Host OS and executed the exploit on Windows 10 Guest OS and we were able to detect the exploit chain in the first run successfully.

Vmware → *CMD* → *PowerShell*

The detection of the exploit chain by the developed algorithm can be seen in the figure 11:

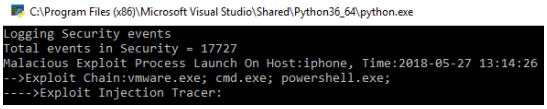


Fig (11) *Guest-to-host* exploit detection

The exploit chain detected by our algorithm is according to the process execution tree shown at process hacker. However, due to the event correlation capabilities of the developed algorithm with respect to malicious process monitoring, we are able to mark the chain as being malicious.

b) Detection against anti-virus/anti-malware solutions

We ran Bit Defender Home, Avira Home, Kasper Sky Home, Avast Home and Panda Security Suite on the Windows 10 Host OS one by one while executing the guest-to-host exploit on a Window 10 Guest OS for the possible detection of exploit chain we weren't able to identify any malicious activity.

4. Experimental Results

We ran a comparative analysis of our detection techniques with different anti-virus and anti-malware solution available in the market. The table 1 shows the result of detection by different security software.

Solution	Detection Yes/No
Proposed Algorithm	Yes
Windows Defender	No
Bit Defender Home	No
Avira Home	No
Kasper Sky Home	No
Avast Home	No
Panda Security Suite	No

Table (1) Result of Comparative Detection Analysis of Developed algorithm and Different Software Security Software

B. Macro-less DDE exploit

1. Experimental Setup

We used Microsoft Office 2013 running on 64-bit Window 10 for the experimentation purpose.

2. Controlled Exploit Execution

For macro-less DDE Exploit we developed an obfuscated DDE Exploit for CVE-2017-11826.The exploit first executes CMD from MS Word then from CMD it executes PowerShell for further exploitation. The exploit execution on Process Hacker can be seen in the figure 12:

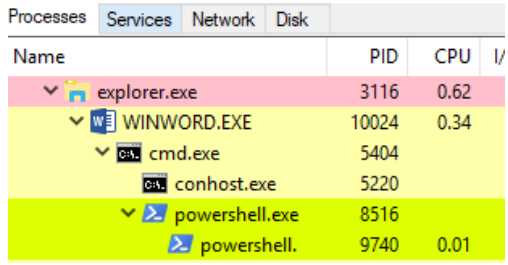


Fig (12) Macro-less DDE exploit execution

3. Scenarios

We created two scenarios for the evaluation of our developed algorithm in the first scenario we executed the exploit on the Experimental setup to check the detection against our developed algorithm. In the second scenario we used online service VIrustototal[3] which performed detection analysis against 59 anti-virus/anti-malware solution details of the scenarios is given below:

a) Detection against developed algorithm

We executed our detection algorithm on the Windows 10 running MS Word and we were able to detect the exploit chain in the first run successfully.

Word → *CMD* → *PowerShell*

The detection of the exploit chain by the developed algorithm can be seen in the figure 13:

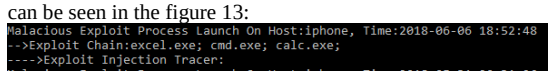


Fig (13) Macro-less DDE exploit detection

3 https://www.virustotal.com/#/file/27c058180a47a5f73ac013e908dde0ec823a28a561408749872e54e6944a4c3f/detection

The exploit chain detected by our algorithm is according to the process execution tree shown at Process Hacker. However, due to the event correlation capabilities of the developed algorithm with respect to malicious process monitoring, we are able to mark the chain as being malicious.

b) Detection against anti-virus/anti-malware solutions

As stated earlier we developed an obfuscated macro-less DDE exploit which have zero detection signature against 59 anti-virus and anti-malware solution on Virus Total[3] as seen in the figure 14:

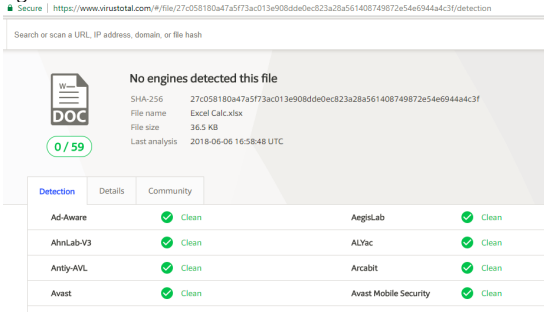


Fig (14) Obfuscated macro-less DDE exploit

4. Experimental Results

Our analysis is being performed on 59 anti-virus and anti-malware solution for saving space few results are omitted but details of analysis can be found online*. Table 2 presenting the detection result compare to different anti-virus and anti-malware solution is given below:

Solution	Detection Yes/No
Proposed Algorithm	Yes
Ad-Aware	No
AegisLab	No
AhnLab-V3	No
ALYac	No
Antiy-AVL	No
Arcabit	No
Avast	No
Avast Mobile Security	No
AVG	No

Table (2) Result of comparative detection analysis of developed algorithm and different software security software

VII. Discussion

The key factor of failure of other detection techniques compare to our techniques is that other detection techniques focus on signature and illegitimate behavior of processes that are being executed. As shown above legitimate behavior of an application can be used for malicious purposes. Similarly signatures of malicious code can be obfuscated as well to avoid detection. Our detection technique works completely different in comparison to other techniques it tries to identify the chain of processes that are being executed by a process and then co-relate them for the identification of malicious processes in the chain. Therefore it has the capability to detect those exploits which are not detected by other available solutions.

We believe that the algorithm complexity is not ideal and there is a lot of room for improvement. But the approach which the algorithm use is quite unique for detection of malicious exploit chains. We intend to further refine the technique for other detection like the detection malicious activates of user by means of event correlation.

VIII. Conclusion and Future Wrok

With the proposed detection technique we are able to identify complex exploit chains. We assume that some complex user administration automation scripts may cause false positives due to their complex execution nature, but overall the detection technique is satisfactory in detecting complex exploit chains. A significant benefit of this technique is that it works completely blindly, without any signature and behavior metrics. In the future, we intent to further refine our technique to trace the exploit chain when the process migrates to another process. Furthermore we will perform our experiments in a large network for identification of false positives in our detection algorithm.

References

[1] Pwn2own 2018 – Day Two Results and Master Of Pwn https://www.zerodayinitiative.com/blog/2018/3/15/pwn2own-2018-day-two-results-and-master-of-pwn Accessed 17 May 2018

[2] Srinivasan, Deepa, Zhi Wang, Xuxian Jiang, and Dongyan Xu. "Process out-grafting: an efficient out-of-vm approach for fine-grained process execution monitoring." In Proceedings of the 18th ACM conference on Computer and communications security, pp. 363-374. ACM, 2011.

[3] Mandal, Debasish, and Yakun Zhang. THE GREAT ESCAPES OF VMWARE: A RETROSPECTIVE CASE STUDY OF VMWARE GUEST-TO-HOST ESCAPE VULNERABILITIES. PDF. London: Blackhat, December, 2017.

[4] Sensepost https://sensepost.com/blog/2017/macro-less-code-exec-in-msword/ Accessed 17 May 2018

[5] Neumann, William C., Thomas E. Corby, and Gerald Allen Epps. "System for secure computing using defense-in-depth architecture." U.S. Patent 7,428,754, issued September 23, 2008.

[6] Win, Thu Yein, Huaglory Tianfield, and Quentin Mair. "Big data based security analytics for protecting virtualized infrastructures in cloud computing." IEEE Transactions on Big Data 4, no. 1 (2018): 11-25.

[7] Wang, Xiaoguang, Yong Qi, Zhi Wang, Yue Chen, and Yajin Zhou. "Design and Implementation of SecPod, A Framework for Virtualization-based Security Systems." IEEE Transactions on Dependable and Secure Computing (2017).

[8] Ucci, Daniele, Leonardo Aniello, and Roberto Baldoni. "Survey on the Usage of Machine Learning Techniques for Malware Analysis." arXiv preprint arXiv:1710.08189 (2017).

[9] Hendler, Danny, Shay Kels, and Amir Rubin. "Detecting Malicious PowerShell Commands using Deep Neural Networks." arXiv preprint arXiv:1804.04177 (2018).

[10] Dosfuscation: Exploring the Depths Of Cmd.exe Obfuscation and Detection Techniques « Dosfuscation: Exploring the Depths Of Cmd.exe Obfuscation and Detection Techniques

Daniel Bohannon - https://www.fireeye.com/blog/threat-research/2018/03/dosfuscation-exploring-obfuscation-and-detection-techniques.html Accessed 19 May 2018

[11] 4688(s) A New Process Has Been Created. (windows 10)

Mir0sh - https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4688 Accessed 19 May 2018

[12] URLhttps://www.blog.pythonlibrary.org/2010/07/27/pywin32-getting-windows-event-logs/ Website TitleThe Mouse Vs. The Python Date Accessed May 27, 2018

[13] Server Virtualization and Os Trends

Spiceworks, Inc - https://community.spiceworks.com/networking/articles/2462-server-virtualization-and-os-trends Accessed 24 May 2018

[14] Virtual Machine Escape

https://en.wikipedia.org/wiki/Virtual_machine_escape Accessed 17 May 2018

[15] Research Report Released: Detecting Lateral Movement Through Tracking Event Logs (version 2)

https://blog.jpcert.or.jp/2017/12/research-report-released-detecting-lateral-movement-through-tracking-event-logs-version-2.htm Accessed 17 May 2018

[16] Comsecuris/vgpu_shader_pocs

Comsecuris - https://github.com/Comsecuris/vgpu_shader_pocs Accessed 18 May 2018

[17] 0patch Blog Luka Treiber - http://blog.0patch.com/2017/10/micropatching-hypervisor-with-running.html Accessed 18 May 2018

2.7 Use of Cyber Attack and defense agents in Cyber Ranges: A Case Study

Use of Cyber Attack and defense agents in Cyber Ranges: A Case Study

Muhammad Mudassar Yamin and Basel Katt

Norwegian University of Science and Technology, Gjøvik 2815, Norway
{muhammad.m.yamin,basel.katt}@ntnu.no

Abstract. With the ever-changing cybersecurity landscape, the need for continuous training for a new cybersecurity skill set is a requirement. This training can be delivered on platforms like cyber ranges. Cyber ranges support training by providing a simulated or emulated representation of computer network infrastructure besides additional training and testing services. Cyber attack and defense skills can be gained by attacking and defending the infrastructure; however, to provide more realistic training, there is a need for necessary friction in the environment, which can be related to both the attacker's and defender's actions. The actions of human teams—both attackers and defenders—provide this friction. Involving human teams in large-scale cybersecurity exercises is relatively inefficient and not feasible for standardizing training because different teams apply different tactics. Currently, the proposed solutions for cyber range training platforms focus on automating the deployment of the cybersecurity exercise infrastructure but not on the execution part. This leaves room for improving exercise execution by adding realism and efficiency. This research presents an agent-based system that emulates cyber attack and defense actions during cybersecurity exercise execution; this helps provide realistic and efficient cybersecurity training. To specify agents' behavior and decision making, a new model, called the execution plan (EP), was developed and utilized in this work.

Keywords: Cyber attack agent · Cyber defense agent · Cyber Range · Security Exercise.

1 Introduction

Conducting operational cybersecurity exercises is a difficult and inefficient task [YK18a]. We have found that automating the different roles involved in cybersecurity exercises can reduce these inefficiencies [YKT+18]. These roles primarily involve a human team required to set up the exercise technical network infrastructure. Additionally, there is a team that attacks the deployed infrastructure as an attacker and a team that defends it as a defender [YKG20]. There can be multiple ways with which a cybersecurity exercise can be executed that may or may not involve both attackers and defenders at the same time. However, in a realistic environment, to train attackers, the systems being attacked are expected to be defended by somebody. Because of shortage in the cybersecurity skills, it is very difficult to

find people with the relevant skill set [YK19a] to conduct continuous cybersecurity exercises. Moreover, different people have different tactics and techniques in cybersecurity operations, making a standardized assessment of cybersecurity exercises difficult [HWD+17]. Therefore, there is a need for automating attack and defense roles in cybersecurity exercises. Despite its importance, there is a lack of research dealing with realism and efficiency in cybersecurity exercise execution in cyber ranges. Most of the related work deals with automating the creation and deployment of the exercise infrastructure. This leaves room for researchers to improve the realism and efficiency of cybersecurity exercise execution. We tackle this issue by proposing an agent-based system, one in which we model the attacker and defender roles and automate their execution as required. In particular, we developed a new modeling technique: execution plan (EP). EP is a multi-level model for specifying behavior and decision-making process for attacking and defending agents. We argue that such agents will add the necessary friction in the cybersecurity exercise environment to make them realistic and reduce the human input of attackers and defenders to make exercise execution more efficient. Therefore, in the current research, we aim to answer the following research question (RQ):

***RQ**: How can cybersecurity attack and defense scenario models be executed autonomously in a cybersecurity exercise to make cybersecurity exercise execution realistic and efficient?*

We present our experience in developing and using cyber-attack and defense agents during cybersecurity exercises against human adversaries. The current paper focuses on the conceptual design, agent decision modeling, practical implementation and user experience with cyber-attack and defense agents. The system is evaluated using a case study against defined benchmarks. The case study involved an operational cybersecurity exercise in which the attack and defense agents were deployed along with human participants. The attack agents were used to create forensic traces for blue team members, which were verified in their forensic reports. At the same time, the defense agents were used to add friction or realism in the exercise environment and were evaluated based on the compromised status of the systems on which they were deployed. The paper is structured as follows: first, we share the research background and related work. After that, we state our research methodology. We then present the conceptual design and technical implementation of cyber attack and defense agents. Finally, we present the experimental results and conclude the article.

2 Research Background and Related Work

Cybersecurity is important both against individual inexperienced hackers and against coordinated teams of hackers that might or might not have governmental support. The conventional methods of teaching cybersecurity include lectures, assignments, seminars, and hands-on labs. Hands-on methods include competitions, challenges, and exercises such as the following:

1. Capture the flag (CTF), which focus on attacking, defending, or attack and defense at the same time.
2. Cyber defense exercises (CDX), which focus primarily on defending.

These competitions, challenges, and exercises are conducted in isolated safe environments, which are called *cyber ranges*. Cyber ranges can host single standalone challenges for CTF competitions or represent a sector/organization's complex computer network infrastructure for CDX exercises [YKG20].

An important element missing from virtual environments is an element of active opposition. A training simulation environment has static defenses (access control, firewalls, fixed set of intrusion methods, etc.) but lacks active opposition (e.g., monitor logs, blocked connections, exploit switching, or information gathering). This results in the cyber operators behaving as though opponents do not have a tangible existenceand higher-level goals. Second, it ignores an opportunity to tailor the student's learning experience through adjustable adversary behavior [JON+15].

Cyberwarfare is an imminent threat to military and civilian systems; it could damage the economy and national security. Cyber aggressors are guided by cognitive behavior (script-kiddies, ideological activists, investigators, financial criminals, intelligence agents, or cyber warfighters). Building an effective training system for cyberwarfare currently faces many barriers. Current training environments are unable to capture the purpose, creativity, and adaptability of cyber warfighters, and cyber warfighters need to be effectively trained against a cunning and adaptive opponents.

We conducted a detailed study on cyber ranges [YKG20] and identified that after 2014, different operations in cybersecurity exercises have become automated

2.1 Related Work

Multiple researchers have worked in the development of cyber attack and defense agents. Here, we discuss some of the work relevant to our research. For emulating attacker steps, a lot of research work has been carried out, resulting in open source, free, and commercially available solutions. Some of them are the following:

Splunk attack range [spl] is a limited cyber range deployment tool in which a small infrastructure can be deployed on cloud and local machines. The infrastructure is monitored by various Splunk attack monitoring and detection engines. Different attacks of the infrastructures are simulated using ART *(Atomic Red Team)* [red]. ART follows the MITRE attack chain model [MIT] and can simulate an attack on Windows, Linux, and Mac OS systems. It uses YAML-based inputs to execute atomic tests for adversary actions on the target systems.

APTSimulator [APT] is an open source advanced persistence threat simulation tool. The tool uses batch scripts with various hacking utilities to create system compromise traces like command and control, defense evasion, lateral movement, and so forth. It is used for endpoint detection agent assessment, testing security monitoring and detection capabilities, and preparing the digital forensic class environment. It roughly follows the MITRE attack chain model and is also limited to emulating attacks on Windows-based host machines.

Metta [ube] is an open source information security preparedness tool. The tool uses Virtualbox, with different development tools like Redis/Celery, python, and vagrant, to simulate adversarial actions. Input is given to the tool in the form of a YAML file, which is parsed to execute the attacker's action on the host- and network-based systems. Metta follows the MITRE attack chain model and is limited to emulating attacks on Windows-based systems.

In the case of cybersecurity exercises, there is a need for the repeatable [HS16] execution of attacker steps for standardized training. Moreover, there are legal and ethical concerns in developing autonomous cyber-attack agents [YB18], so for an attack agent, the attack execution steps need to be planned while keeping a human in the loop before executing them in a cybersecurity exercise environment.

For the defense agent, most related work has focused on network- and host-based detection systems, while some have looked into introducing active attack-repellent features in defense agents. Randolph el al. [JON+15] conducted research about modeling and integrating cognitive agents in the cyber domain. The purpose was to develop agents that can produce the necessary friction during cybersecurity exercises to create realism. They developed a novel model for cyber offense and defense; they used the model to create software adapters that translate from task-level actions to network-level events to support agent-network interoperability for cybersecurity operations. They presented a high-level defender goal hierarchy in which the defender has to (1) establish a baseline, (2) detect an ongoing attack, (3) stop an ongoing attack, and (4) prevent future attacks.

Kott et al. [KTD+18b] put forward the idea of the development of a *"Hello world"* program for the intelligent autonomous defense agent. The researchers stated that the autonomous agent, should have the following six capabilities to be considered intelligent:

1. The agent should be strictly associated with its environment and should be useless outside its designed environment.
2. The agent should interact with its environment using appropriate sensors.
3. The agent should act to achieve its stated goals.
4. The agent activities should be sustained overtime to be autonomous.
5. The agent should have an internal world model that can express its states with performance measures.
6. The agent should learn new knowledge and modify its model and goals over time based on new knowledge.

Paul et al. [TKD+18] proposed the reference architecture for autonomous intelligent cyber defense agents (AICA). In their proposed architecture, the researchers stated that AICA has five main high-level functions:

1. Sensing world state information
2. Planning and action selection
3. Collaboration and negotiation
4. Action execution
5. Learning and knowledge improvement

The researchers also suggested a functional architecture for AICA and stated that according to their assessment, the development of such agents is not beyond the current technical capabilities and can be developed within ten years.

Although other researchers are also investigating autonomous intelligent cyber defense agents, the work is still in the design stage [KTD+18b,TKD+20]. One implementation of such an agent was proposed by Randolph et al. [JON+15], in which the researchers suggested the idea of adding friction in cybersecurity exercises. Their approach utilized cognitive modeling of human cybersecurity experts to model the behavior of the agent based on human expertise. We consider such an approach not suitable for agent development because human experts have different subjective experiences, which can result in unintentional bias in their behavior, as observed in data-driven AI algorithms [YUUK21]. Kott et al. [KTD+18b] and Paul et al. [TKD+18] provided the baseline requirement and functional needs of the cyber defense agent, which we considered suitable to fulfill our requirements. For the attack agent, we find the ART, APTSimulator, and META approaches suitable for usage in cybersecurity to conduct cybersecurity exercises. We integrated multiple ideas and approaches suggested by various researchers for the development of our attack and defense agents. Our attack agent follows a systemic step-by-step execution of attacks similar to ART, APTSimulator, and META, but it is integrated with a cybersecurity exercise orchestrator, making it suitable for computationally repeatable cybersecurity exercises and experiments. Our defense agent is also integrated with our cybersecurity exercise orchestrator and provides example implementation of the design presented by Kott et al. [KTD+18b].

3 Methodology

We employed numerous research methodologies during this research activity. We used the **DSR** *design science research methodology* [HC10] for the overall development of the necessary artifacts for this research. DSR is a very well-known research methodology that has five phases *1) awareness of the problem, 2) suggestion, 3) development, 4) evaluation,* and *5)conclusion.* These phases are iterative in nature, and the results of these phases are used to improve the overall design to produce a research artifact that addresses the research problem [KV08].

In awareness of the problem, the research problem can be identified, so we leaned on certain studies for this phase [YK18a,YKT+18,YKG20]. In the suggestion phase, the solution's designs are proposed to address the research problem, and we conducted this step using certain studies [YKG20,YK19b].We are currently developing and evaluating the artifacts, and the present paper is presenting the results of this phase. For the development of such an artifact, model-driven approaches are widely employed [BCD+19]. In such an approach, a complex problem is abstractly defined in the form of a model, and automation techniques are used to generate low-level artifacts from the abstract model. In our previous work, we developed a **DSL** *domain specific language* to formally specify the cybersecurity exercise environment [YK22]. The environment contains the exercise infrastructure and agents

running within the environment. In this work, we used the DSL developed for automating the creation and deployment of the exercise environment. Additionally, we augmented the DSL with a new modeling technique based on attack/defense trees that we call execution plans (EPs). EPs enable a designer to model the behavior and decision making of attack and defense agents. Finally, in this work, we use **applied experimentation** in operational cybersecurity exercises against defined benchmarks to gather the analytical data for evaluating the performance of developed cybersecurity defense agents [EM17].

4 Conceptual Design

This research work is a part of a larger initiative in which the whole process of the cybersecurity exercise life cycle is automated. To achieve this, a DSL is developed to transform abstract concepts related to the cybersecurity exercise life cycle [YK18a] into concrete artifacts. These abstract concepts include defining the network topology, defining the vulnerabilities in the nodes connected to the network, defining benign network traffic, and defining attacker and defender behavior. In this work, our scope is limited to attacker and defender behavior, so we only focus on the concepts involved in attack and defense agent development. Figure 1 illustrates the DSL meta-model for defining attacker and defender properties used to generate agent artifacts. Later in section 5, a concrete syntax will be presented, which will give an example of an instantiation of this DSL meta-model.

The attack agent comprises of a total six concepts. The first one is the *attack-action-id*, which is an identifier of a potential attack action. In case the attack action causes a particular tool to be executed, the tool name can be used as an identifier. The second, third, and fourth concepts are *Agent IP*, *Agent User ID*, and *Agent User Password*. These concepts provide the information about the agent from which the attacker action is going to be performed. The fifth concept is *Argument*, which represents one of the properties that are needed for the attacker's action to be executed. In terms of an a tool that represents an attacker's action, this concept represents the tool's specific arguments. The sixth and final concept for the attack agent is *Target*, which is the IP address of a machine on which the attack agent performs its actions. Similarly, the defense agent has five properties. The first three are *Agent IP*, *Agent User ID*, and *Agent User Password*, which provide the information necessary to install the defense agent on a system. The fourth property is the *Operating System*, which provides the information of the operating system the defense agent is working on. The final property is the *Parameter*, which provides the defender with a specific knowledge base. The parameter contains a list of pairs, each of which consists of an attacker action and the defender reaction to it.

The DSL uses an orchestrator that implements the abstract concepts defined in it to concretely create operational artifacts for establishing the necessary exercise infrastructures, generating network traffic, emulating benign users, launching cyber attacks, and defending against these attacks. The orchestrator has a master control unit connected to the attack agent and used to control them in a semi-autonomous

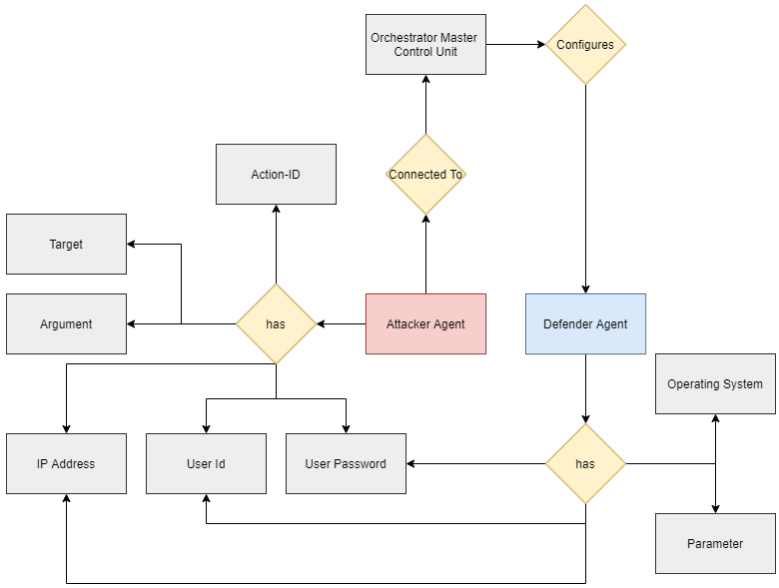


Fig. 1: A meta-model of the attack and defense agents

manner, while the orchestrator configures the defense agents before deployment so that they can work in an autonomous mode. The operational artifacts contain network topology templates for cloud deployment, specification of the vulnerabilities in the form of software, service and configuration, and the specification of the benign, attacker, and defense agents' behaviors present in the deployed network. The specification is given to the orchestrator in JSON, and it starts generating the necessary artifacts in five steps, as shown in Figure 2. First, the exercises network infrastructure is generated through a HEAT template for infrastructure deployment in Opestack-based cloud. In the second step, software, service, and configuration are invoked in the deployed infrastructure using a custom SSH-based installation and configuration module to make the infrastructure vulnerable. In the third step, part of the deployed infrastructure is used to generate benign traffic using various automated tools like TCP relay and VNCD tool. In the fourth step, an attack agent is used to test and verify the vulnerabilities present in the exercise infrastructure. In the fifth and final step, a defense agent is deployed in part of the infrastructure to add the necessary friction in the cybersecurity exercise. The DSL implementation related to exercise network infrastructure generation and generating benign user

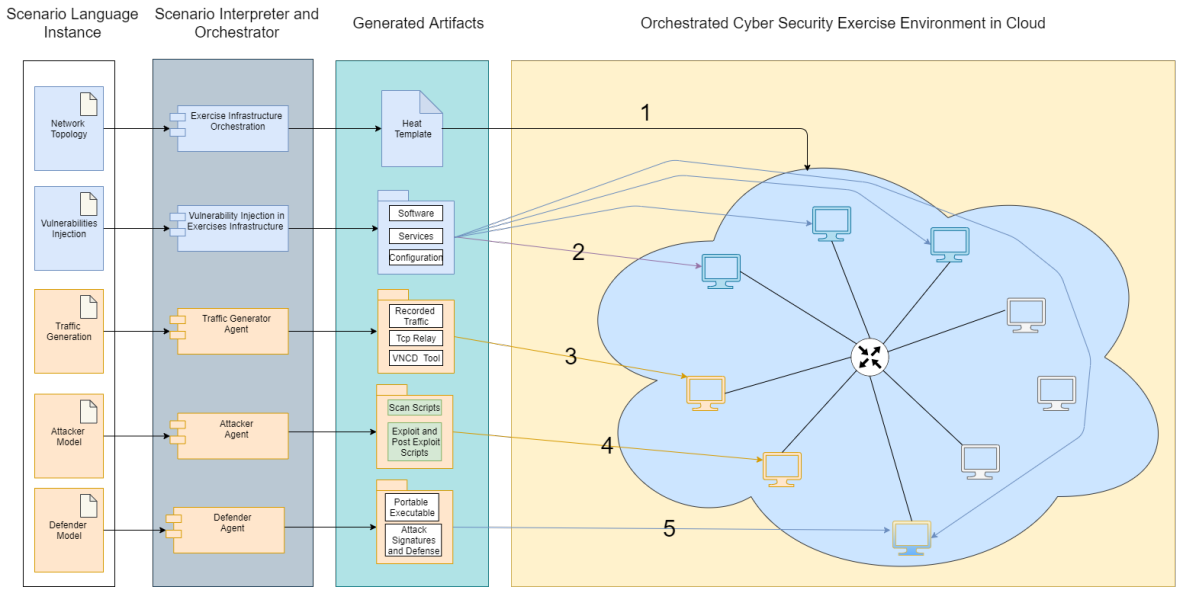


Fig. 2: Cyber security exercise operation orchestrator

behavior and traffic is part of another research work. In the current work, we are only focusing on the DSL instance of attackers and defenders.

There can be multiple ways attack and defense agents can be designed and deployed. This depends on the eventual goal of the agent, that is, what is expected from the agent. We can model the goals of the attack and defense agents based on the responsibilities of the red and blue teams. Lockheed Martin's *cyber kill chain* course of action matrix [HCA+11] provides a simplified way to model the attack and defense phases. For the attacker, there are seven phases *reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objective*. These attack phases utilize a set to tools and techniques to achieve the attacker's eventual objectives and goals, which could be the disruption of services or extraction of data. On the defense side, there are six phases to stop the attacker: *detect, deny, disrupt, degrade, deceive, and destroy*. The defender uses different network/host intrusion detection and prevention systems, firewalls, antivirus software, and honeypots to achieve the objective of stopping the attacker.

There are other models like MITRE ATT&CK that can be applied for modeling attacker actions and relevant defender reactions. However, MITRE ATT&CK strictly focuses on concrete actions, tactics, and techniques that are specific to an

operating system. The cyber kill chain is very general and can easily model attack and defense in a different operating environment. We consider the generality of the cyber kill chain suitable for modeling attack and defense scenarios for cybersecurity exercises.

Theses attack and defense phases are executed by utilizing different tools and techniques. These tools and techniques provide relevant information to the agents so that they can make intelligent decisions. However, it should be noted that this intelligent decision making is strictly dependent on the amount of information being shared, which is related to the agent's goals.

If we look carefully at the information sources, there are primarily two types: external and internal. For an attack agent, external information can be gathered from scanning networks and identifying software, services, and configurations, while its internal information can be the knowledge about the exploits on the identified software, services, and configurations. A defense agent's external information sources contain information from the environment such as network activity, while internal information sources contain information about the system's internal activities such as event logs, which are widely used to detect system exploitation and lateral movement [Cen17]. These information sources are combined to provide *security information and event management* capabilities for defending against the attacks by correlating information from multiple sources. However, information correlation requires manually defining security events to look for and manually take actions against them to stop the attacker in its tracks. A defense agent can assess the type of traffic to identify whether it is benign or malicious. An attacker can overwhelm the defender by launching multiple attacks at the same time, which could make intelligent decision making very difficult. It will also create a new threat vector for the defender because decision making depends on external sources that can be manipulated.

Let us analyze the course of action matrix [HCA+11] for attackers and defenders. Here, the attacker's reconnaissance and weaponization goals can be detected by external information sources like web analytics and NIDS (network intrusion detection systems). In contrast, exploitation and installation can be detected by HIDS (host intrusion detection system). Although detecting an attack is desirable at an early stage, a host-based system can be better suited to respond directly to the attack; it can detect a security event and automatically respond to it by making operational changes such as applying local firewall rules and installing patches through its knowledge base without relying on an external input to deny the attacker actions. The knowledge base can contain information about the expected attacker's actions and the appropriate defender response. This knowledge can be useful for known attack tactics and techniques; however, it needs to be updated for new attack detection and responses, which require some intelligence. This intelligent behavior can be learned by analyzing the attack vectors and implementing security actions against them, manually first and automatically later. The attack vectors can be learned by constantly monitoring the system state and detecting changes. When a change is detected, the events that lead to that change can be fetched for identifying the malicious actions. A set of predefined reactions can be

specified for implementation against a particular set of actions to deny the attacker from using them for further exploitation.

All components and parts of a cybersecurity exercise environment are considered a system, and each system is running software and services with system-specific configurations. The system for the attack agent is a Kali Linux machine controlled by another system running our developed orchestrator software and using SSH as a service for communication with the Kali machine. The orchestrator can control multiple Kali machines to lunch multiple attacks at the same time. Similarly, for the defender part, the orchestrator can inject a defense agent with its knowledge base as software that can independently run on the injected system to protect its software, service, and configuration. Additionally, there are traffic generators that are present in the cybersecurity exercise environment, which are basically attack agents performing benign activities such as replying PCAP files and automating GUI user behavior using VNCDtool. The agents and their interactions are presented in 3, which is mapped with the third, fourth, and fifth steps of the orchestrator, as presented in Figure 2.

The developed agents operate in a cloud-based cybersecurity exercise environment. The environment has attack and vulnerable machines on which the attack and defense agents are operating. The vulnerable machines have vulnerabilities related to software, services, and configurations that an attack agent can remotely sense. The behavior of the agents in the environment is governed by the world state. The *world state* is the software's, services', and configurations' specific information provided to the agents. New information about the world state is gathered from the environment in which the agents are operating by using their sensing capability. The sensing capability indicates which type of systems the agent is interacting with and triggers an action when it finds some specific information about the world state [KTD+18b,KTD+18a]. For the attack agent, active and passive reconnaissance tools like arp-scan and Nmap are used to gather information about the services running in the network. This information is used to create the world model for the attack agent, and changes in these services will update the world state for the attacker. When the attack agent senses a vulnerable service, it triggers a change in its world state, upon which an attack action is selected and executed on the vulnerable machine. Similarly for the defense agent, Windows security logs provides an active sensing capability to create a model of the world state, which includes the type of software and services running on the system and any changes in their configuration. When an attacker interacts with the system defended by the defense agent, it creates logs that are then used to update the world state and trigger a reaction from the defense agent. The decision-making process of the proposed agents are discussed in section 4.1.

When the attacker's action is executed, an event is created in the vulnerable machine environment. The defense agent has a list of malicious events, the current world state, and responses to those events. The defense agent can sense the known events generated by attacker actions, which can trigger a change in its world state. The defense agent then selects an appropriate response and executes it to counter the attack agent's actions. The interactions between the attack and defense agents

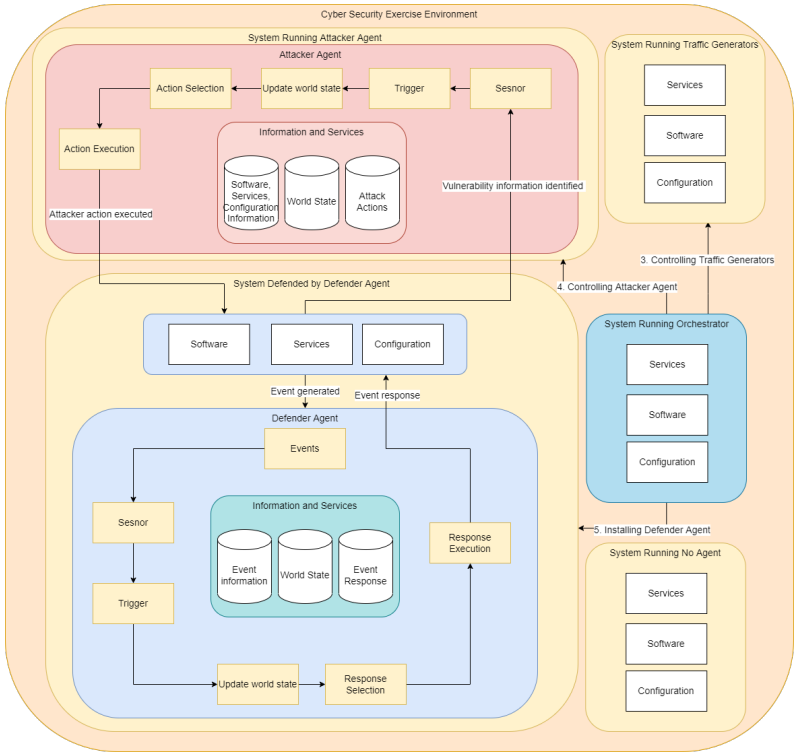


Fig. 3: Attack and defense agent environment and interactions

in the cybersecurity exercise environment is presented in Figure 3, and details of the attack and defense agent work flows are presented in Figure 6 and Figure7 respectively:

In terms of deploying the agents in the cybersecurity exercise environment, there are certain considerations. Some researchers have implemented script execution techniques on the machine [Gen] to generate attack logs; however, we deployed the attack agents on a remote machine to emulate realistic adversary behavior. On the other hand, the defense agents were deployed on the machines because a central command and control unit could have been compromised to disable the defenders' functionality.

4.1 Agent Decision Modeling

On an abstract level, our agents have five properties— *Knowledge*, *Goals*, *Conditions*, *Actions*, and *Commands*, where the knowledge is provided through the DSL. The *Knowledge* of an agent contains information about the world state, like the software, services, and configuration running in the environment. An agent's *Condition* is used to perform condition-specific actions on the software, services, and configuration. These conditions are triggered based on events that change the world state. An *Action* is executed using a set of *Commands*. A set of successful action executions result in the achievement of a *Goal*, which is modeled based on the *CKC* course of action matrix. For the attacker, these goals are *Reconnaissance, Delivery, Exploitation, Installation, Command and Control* and *Actions on Objective*. Similarly, for the defender, these goals are *Detect* and *Deny*.

The DSL instance is translated into EPs (Execution Plans) for the achievement of specific *Goals*. We adapted the concepts from the attack and defense trees [KMRS14], as well as the hierarchy of action plans [Kot05], to develop the EP models. The EP models consist of three levels of decisions *high, medium,* and *low*.

4.1.1 EP Model EPs are tree-structured models that represent the agent's decision-making process. An EP describes the goals, conditions, and commands of an agent, as well as showing the path that needs to be taken to reach the final conditions and fulfill the goals. These conditions result in one of the following EP outputs: plan fulfilled, plan not fulfilled, or plan maybe impractical.

The root of an EP tree is the goal of an agent, and an end-leaf of an EP tree represents the commands that will fulfill an agent's goal. An EP consists of three decision levels-Level 1, Level 2, and Level 3-and each level is represented as a sub-tree of the EP tree. The root of Level 1 of an EP tree is the root of the EP tree. The leaves of one sub-tree are the roots of the next level sub-tree. A parent node is connected with its children nodes using two possible operators, *AND* and *OR*, which are represented by $\wedge$ and $\vee$, respectively. The semantics of the nodes and operators in an EP tree depend on the level where the node exists.

Level 1 The Level 1 sub-tree of an EP tree is the first high level sub-tree of the EP tree. The root node of a Level 1 sub-tree of an EP tree represents the main goal of the EP tree, and the leaves represent a set of sub-goals. The operator $\wedge$ is used if all the sub-goals needs to be fulfilled for the parent goal to be achieved. On the other hand, $\vee$ can be used if the fulfillment of one sub-goal will result in the fulfillment of the parent goal.

Level 2 The Level 2 sub-tree of an EP tree is the second medium-level sub-tree of the EP tree. The root of the Level 2 sub-tree is a leaf in the Level 1 tree or the root of the EP tree where the Level 1 sub-tree consists of one node. Level 2 represents a sequence of conditions that need to be checked to decide which actions should be executed. The nodes of a Level 2 sub-tree are conditions with two possible outputs *Yes/No*, meaning that only $\vee$ operators are allowed in a Level 2 sub-tree. Each parent node is connected to, at most, two children

nodes, which represents the next conditions to be checked. A special type of condition is called "Not Fulfilled." It is a final condition with no children, and it is denoted by the symbol —; reaching this condition means that the plan is not or cannot be fulfilled. A leaf in a Level 2 sub-tree can be either an NF "Not Fulfilled" condition or an action.

Level 3 The Level 3 sub-tree of an EP tree is the third low-level sub-tree of the EP tree. Level 3 roots are actions represented in the leaf nodes of the Level 2 sub-tree, and the nodes represent concrete commands. Both $\wedge$ and $\vee$ are allowed in a Level 3 sub-tree. $\wedge$ means all the children's commands need to be executed to achieve the action, while $\vee$ means that the execution of any of the commands can achieve the action.

Output Plan output "Fulfilled" is reached when all the commands in the EP tree leaves are executed successfully and the goal is achieved. Plan output "Impractical" is reached when the result of the execution of one command leaf is not successful. Plan output "Not Fulfilled" is reached when the agent cannot reach an action leaf because of knowledge or resource limitations. The EP plan in this case will stop at the Level 2 sub-tree.

4.1.2 EP Formal Model We use Datalog [Dat] for formal modeling of the agents decisions and to verify the different decision properties like: *is the goal fulfilled or not?*. Datalog is a programming language based on a declarative logic [Llo12]. It is employed by researchers for large-scale software analyses [Nai20], automatic evaluations of cybersecurity matrices [ZN20], and the verification of cybersecurity exercise scenarios [RCA20], making it suitable as a formal model for cybersecurity exercise scenarios. It consists of two parts: facts and clauses. A fact conforms to the parts of the elements of the predicated phenomenon. A clause refers to information deriving from other subsets of information. Clauses rely on terms, which can contain variables; however, facts cannot. It adjudicates whether the specific term is adherent to the specified facts and clauses. If it happens to be so, the specific query is validated via a query engine, providing the prerequisite facts and clauses.

When running a Datalog operation, the specified conditions include a combination of two facts along with a singular clause. We assign a condition that if the query is valid, a specific response is to be expected at the end. The conclusion of the said experiment is that the specific response is received and that the query is satisfied. By utilizing the clauses via their variables, the engine can pinpoint and find the result. For a concrete example [CGT+89], consider the facts *"John is the father of Harry"* and *"Harry is the father of Larry"*. A clause will allow us to deduce facts from other facts. In this example, consider we want to know *"Who is the grandfather of Larry?"*. We can use three variables X, Y and Z and make a deductive clause: If X is the parent of Y and Y is the father of Z, then X will be the grandfather of Z. To represent facts and clauses, Datalog uses *horn clauses* in a general shape:

$$L_0 : -L_1 ..., Ln$$

Each instance of L represents a *literal* in the form of a *predicate* symbol that contains one or multiple *terms*. A *term* can have a constant or variable value. A

Datalog clause has two parts: the left hand side part is called the head, while the right hand side part is called the body. The body of the clause can be empty, which makes the clause a fact. A body that contains at least literal represents the rules in the clause. Lets us represent the above mentioned facts that *"John is the father of Harry"* and *"Harry is the father of Larry"* as follows:

$$Father(John, Harry)$$

$$Father(Harry, Larry)$$

The clause if X is the father of Y and Y is the father of Z, then X will be the grandfather of Z can be represented as follows:

$$GrandFather(Z, X) : -Father(Y, X), Father(Z, Y)$$

For our agents we define 4 basic predicates for decision modeling which are *1) Goal, 2) Condition, 3) Action* and *4) Fulfilled*. The facts for the decision model with their definitions are as follows:

Definition 1. *The Goals predicate is logically presented as Goals(Goal, SubGoal), and it has two variables Goal and SubGoal. The Goal is a string value which indicates attack and defense goals like 'Exploit System' for attack and 'Prevent Attacks' for defense. The SubGoal is also a string values which contains sub goals for achieving the Goal like 'Reconnaissance' and 'Exploitation' for attack and 'Detect' and 'Deny' for defense. A concrete example of Goals predicate for attack is presented as fallows:*

$$Goals('ExploitSystem','Reconnaissance')$$

$$Goals('ExploitSystem','Exploitation')$$

Similarly for defense Goals can be represented as fallow:

$$Goals('PreventAttacks','Detect')$$

$$Goals('PreventAttacks','Deny')$$

Definition 2. *The Condition predicate is logically presented as Condition(Goal, Parameter, Command) and it has three variables Goal, Parameter and Command. Goal is a string value which is the leaf of Level 1 tree consisting goals and sub goals like 'Reconnaissance', Parameter is a string value that is condition specif to achieve the goal like 'Active' for performing active reconnaissance, Command contain the parameter for performing a low level action like service and version scan using Nmap '-sS -sV'. A concrete example of Condition predicate is presented as fallows:*

$$Condition('Reconnaissance','Actvie','-sS - sV')$$

Similarly a defense Condition can be represented as fallow:

$$Condition('Deny','shell.exe','taskkill/IM"shell.exe"/F')$$

Definition 3. *The Action predicate is logically presented as Action(ActionName, ActionTarget), and it has two variables ActionName and ActionTarget. ActionName is a string value which contain low level goal action execution like Nmap for attacker and Deny for defender. ActionTarget is string value which contain the information of machine address on which the action is to be executed for the attacker and the pattern on which the action is triggered by the defender. A concrete example of Action predicate is presented as fallows:*
For attacker:

$$Action('Nmap','172.168.2,2')$$

For defense:

$$Action('Deny',' shell.exe')$$

Definition 4. *The Fullfilled predicate is logically presented as Fullfilled(Goal, ActionName) and it has two variables Goal and ActionName. Goal is a string value that indicates high level goal like 'Reconnaissance' and ActionNamel is a string value that indicates concrete tool or action to achieve the high level goal like 'Nmap'. A concrete example of Fullfilled predicate is presented as follows:*

$$FullFilled('Reconnaissance',' Nmap')$$

For defense: FullFilled('Deny','shell.exe')

4.1.3 EP Model Verification

The decision model presented in section 4.1 help us to verify various agent properties before their execution, like:

- *How high level goal can be translated into to low level actions*
- *Can the agent fulfill the given goal?*
- *What information is missing to achieve the goal?*

To verify the decision model we define a new predicate CheckGoals which takes two variables Goal and SubGoal and is logically presented as CheckGoals(Goal,SubGoal). A logical relationship is defined between the Goal and SubGoal so it can be established whether the Goal is the leaf for Level 2 conditions or the SubGoal.

$$CheckGoals(Goal,\ SubGoal) \leq CheckGoals(SubGoal,\ Goal)$$

FullFilled predicate is used to link the Goal and SubGoals which is presented as fallows:

$$FullFilled(Goal,\ SubGoal) \quad \leq CheckGoals(Goal,\ SubGoal)$$

Furthermore, it is defined whether a SubGoal needs to be completed in order to achieve the Goal. A relationship is established between Goal, SubGoal and Action using transitive property which is presented as fallows:

$$\begin{aligned} FullFilled(Goal,\ SubGoal) \quad &\leq CheckGoals(Goal,\ Action)) \\ \&\ FullFilled(SubGoal,\ Action) &\,\&\, (Goal\,! = SubGoal) \end{aligned}$$

To verify the attack decision for achieving high level goal using low level action based upon certain conditions following clause can be defined:

$$FullFilled('Reconnaissance',\ SubGoal)\ \&\ Condition(Goal,\ 'Active',\ Commands)\ \&\ Action(SubGoal,\ ActionTarget)$$

The clause will return the high level goal, and the low level specific action to be executed based upon the specific command. Similarly for the defense decision following clause can be defined:

$$FullFilled('Deny',\ SubGoal)\ \&\ Condition('Deny',\ SubGoal,\ Commands)$$

The clause will return the low level pattern through which the action is triggered and the low level command to deny that action.

4.1.4 EP Model Representation A schematic representation of the attack and defense agent EP models all three levels is presented in Figure 4 and Figure 5, respectively.

In Figure 4, a high-level *Goal* is given to an agent that has the aim of performing system exploitation. The sub-goals are *Reconnaissance* and *Exploitation*. The agent will check information in its knowledge base to make medium-level decisions, for example, whether information about the target is provided or not. If the target information is provided, it will check whether it is accessible or not; if it is accessible, then the agent will check if some specific argument is present to perform a specific kind of reconnaissance like *nbtscan*, which is a low-level decision. Otherwise, it will use a default reconnaissance technique like *nmap* or *netcat*. Similarly, if the target information is not provided, the agent will check whether there is a network interface and if on that network interface it can perform *arp-scan* a medium-level decision.If the condition returns true, then the agent can perform an arp-scan using default commands, which is a low-level decision.

In Figure 5, a high-level *Goal* is given to an agent to prevent the attackers from performing any actions. The sub-goals are to *Detect* and *Deny* attacker actions. Whenever an attacker performs an action, it creates an event. If the event is detected, then it is checked in the knowledge base of the defender; this is considered a medium-level decision. If the knowledge base contains information about the reaction to this specific action, then it will execute a specific command to the counter-attacker's action; this is considered a low-level decision, for example, killing a specific malicious process using *taskkill*. In another case, if there is no specific command to react to the attacker's action, then the agent will execute a general defense command to counter the action, such as closing the ports using *netstat* or *npx-kill-port*. If the action of the attacker is not detected, then the agent will fail to defend the system.

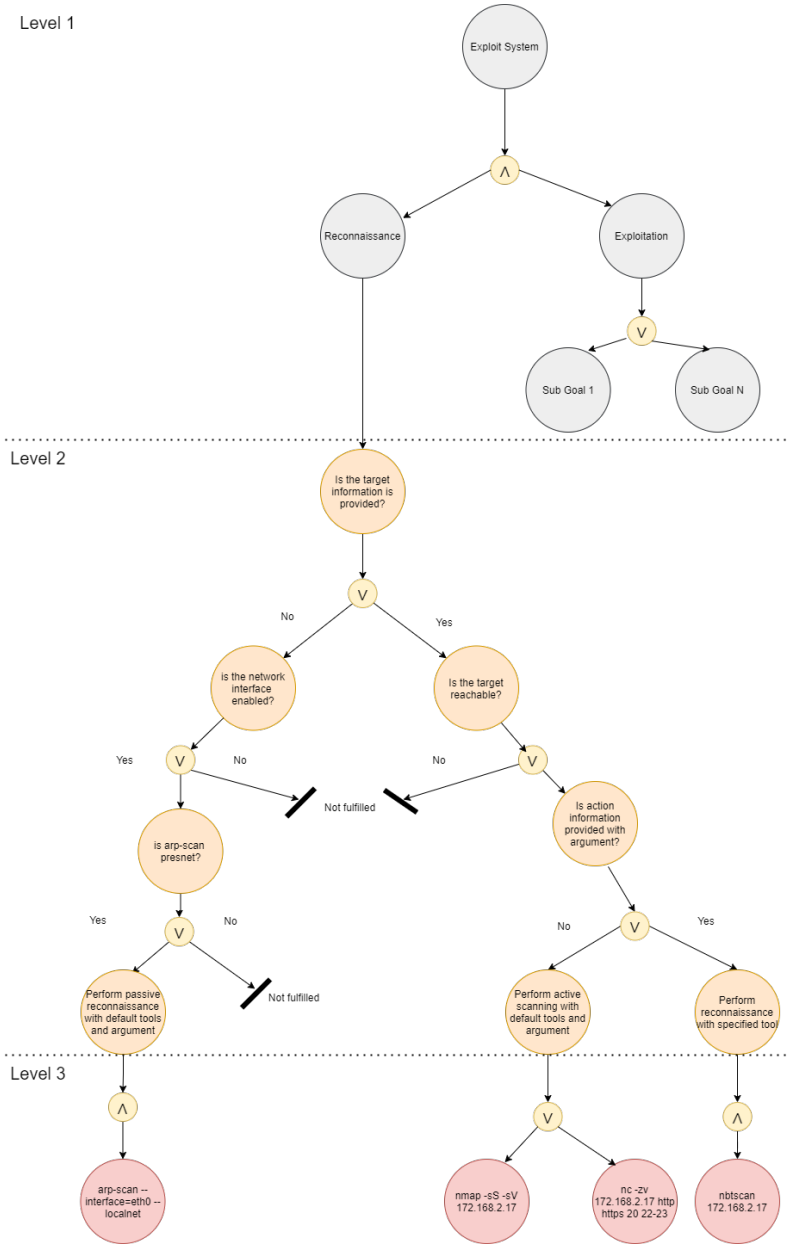


Fig. 4: Attack agent EP

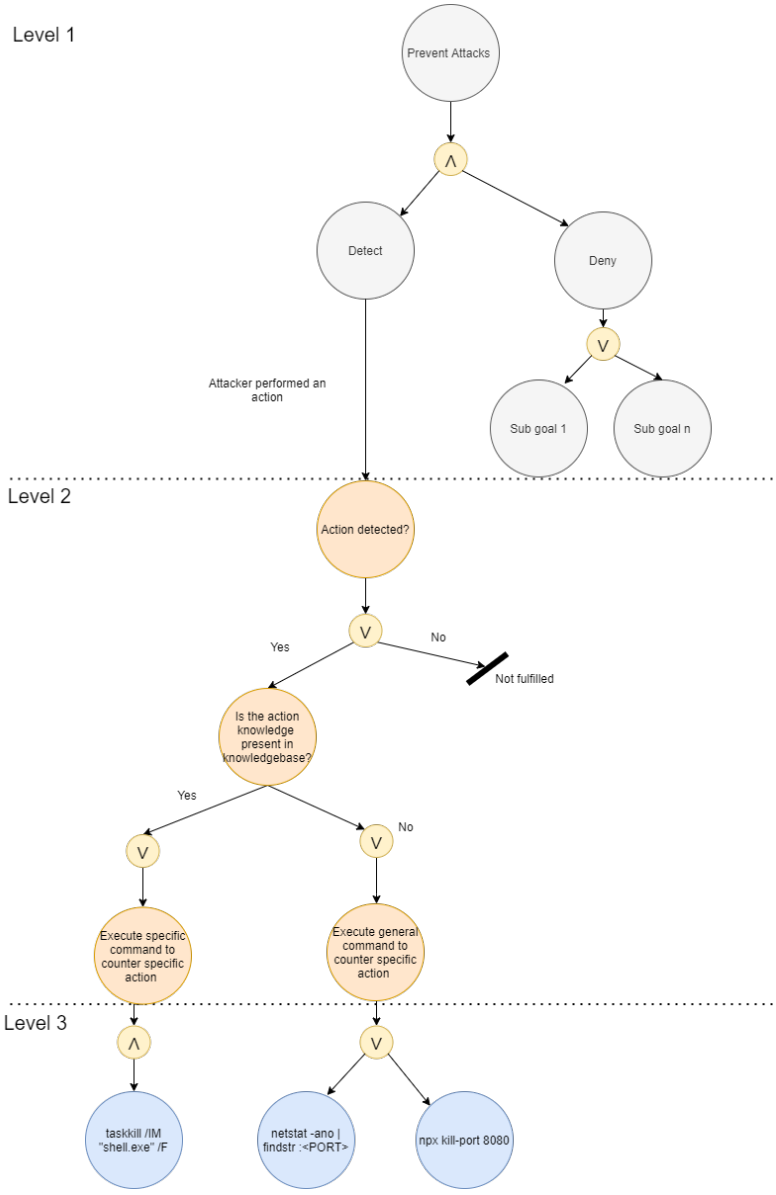


Fig. 5: Defense agent EP

4.1.5 Attack Agent Conceptually, the attack agent's goal is to perform the steps involved in exploiting vulnerable systems during a cybersecurity exercise. The steps involve performing scanning, identifying vulnerable services, and launching an attack on them. Multiple adversary emulation tools already exist in academia and industry, here using various techniques ranging from logical programming [Yue15] to AI [Sto18] for achieving this goal.

A model-driven approach for executing the attacks during a cybersecurity exercise can provide repeatable and standardized training. The model needs to follow penetration testing execution standards [PTE] to leave realistic attack and exploitation traces for the defender or blue team members to identify. This can be achieved by specifying the attacker's actions in a DSL, hence enabling the precise execution of attack steps and helping in the evaluation by the defenders in incident response and forensic analysis.

We combined complex attacker operations into six components of a DSL, whose concrete syntax instance is presented in Listing 1. These components specify the attack techniques that are going to be used by the attacker and on which target it needs to be performed. The DSL instance components are used to provide the necessary information to the EP model to specify the behavior of the attacker based on the *cyber kill chain*. Once the model has been created, it is verified by executing it in different operational cybersecurity exercises with the same network topology to check whether its execution is repeatable for standardized training.

Listing 1: Concrete syntax for attacker DSL instance

```
[
 {
  "nbtscan": {
   "AgentIP": "192.168.81.128",
   "AgentUserID": "root",
   "AgentUserPassword": "toor",
   "Argument": "",
   "Target": "172.168.2.17"
  }
 }
]
```

The attack agent DSL model has the following properties:

1. It can perform actions from the following list: *Reconnaissance, Delivery, Exploitation, Installation, Command and Control,* and *Actions on Objectives.*
2. It has six attributes: agent action name, agent to use, agent credential user ID and password, action-specific parameters and the target on which action is needed to performed.
3. It runs in a separate attack machine in the exercise network environment, where it has network-level access to vulnerable machines present in the exercise network.

4. It interacts with exercise networks using the specific actions, which can collect information about the software, services, and configurations present in exercise machines and then preform other actions to exploit those machines. Deciding on which actions and commands to execute is specified in the EP model.

Utilizing the above-mentioned properties, the attack agent launches attacks in a semi-autonomous manner, as defined in the EP models. These EPs consist of executing the attack phases presented in *CKC* by utilizing its attack agent properties. The attack agent's overall workflow is represented in Figure 6. Itrepresents how the

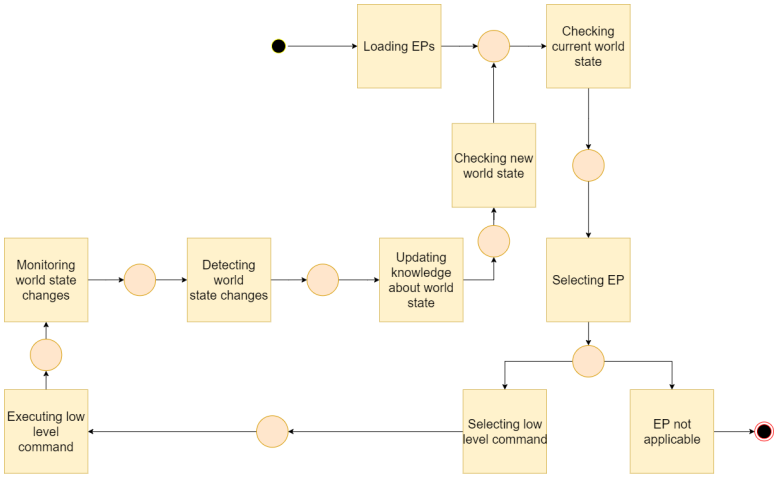


Fig. 6: Attack agent's work flow

agent functions. The agent then does the following: (1) First, the agents loads the goals and related EPs. (2) The agent checks the world state. Initially, the world state is empty for a newly deployed agent. (3) An EP model for the loaded goal is utilized by the agent to decide which actions to perform and fulfill the goal. (4) The agent decides which low-level commands to execute based on the EP model (4.1), and it will fail in case the goal was not fulfilled or was impractical (4.2). (5) The selected command(s) will be executed. (6) The agent (continuously) monitors the environment to detect any changes in the world state and to get the output and results of executing its commands. (7) The sensors detect new knowledge. (8) New knowledge updates the world state. (9) The agent checks the new world state.

4.1.6 Defense Agent The defense agent's primary goal is to defend its system from external and internal attackers. This primary goal has additional sub-goals

in which the defense agent has to *Detect, Deny, Disrupt, Degrade, Deceive, and Destroy*an attacker. The focus of this work is on *detect* and *deny.*These sub-goals are achieved by the usage of different tools and techniques at different stages of an attack. These tools and techniques include but are not limited to network and host intrusion detection and the prevention system, web analytics, security configuration, and system user training.

Our DSL instance is used to specify the defense agent's properties. Based on these properties, the EPs are executed by our orchestrator. The orchestrator inserts the agent in the machine present in the exercise network with the knowledge base of the events generated by the attacker's actions Conceptually, the defense agent has the following properties:

1. It can detect and deny the actions performed by the attacker.
2. It has a knowledge base that contains information about the attacker's actions and their countermeasures.
3. It runs on the exercise machines being attacked.
4. It interacts with the events generated by the attacks and implements specified countermeasures on the machine it is running.

Utilizing the above-mentioned properties, the defense agent can perform defense measures against launched attacks in a semi-autonomous manner, as defined in the EP models. These EPs consist of executing the defense phases presented in *CKC* by utilizing the defense agent properties. One key difference between the attack and defense agent is that the defense agent is not controlled by a Master and is independent in its execution. The Master only configures the knowledge base of the defense agent one time and uploads it on the machine that needs to be defended. The defense agent's overall workflow uploaded to a machine is represented in Figure 7 In the work flow, the agent does the following:(1) First, the agent loads the goals and related EPs.(2) The agent checks the world state based on the EP model of detecting the attacker. Initially, the world state is empty for a newly deployed agent. (3) The agent will use its sensors to detect an event generated by attacker actions based on its knowledge base. (4) In the detection phase, if the attacker's action event is detected, then the agent will check its knowledge base to counter the attacker's action (4.1). If the attacker's actions were not detected by the agent, then it will fail to deny the attacker (4.2). (5) The agent will update its world state and act on new EP models to deny the attacker. (6) To deny the attacker, the agent will check its knowledge base. (7) For denying the attacker, the agent will execute a low-level command that changes the world state. If the agent's knowledge base has information about the specific action, then it will execute a specific command (7.1). If the agent does not have specific countering information, then it will execute a general command (7.2). (8) New knowledge is updated in the world state of the agent. (9) The agent checks the new world state.

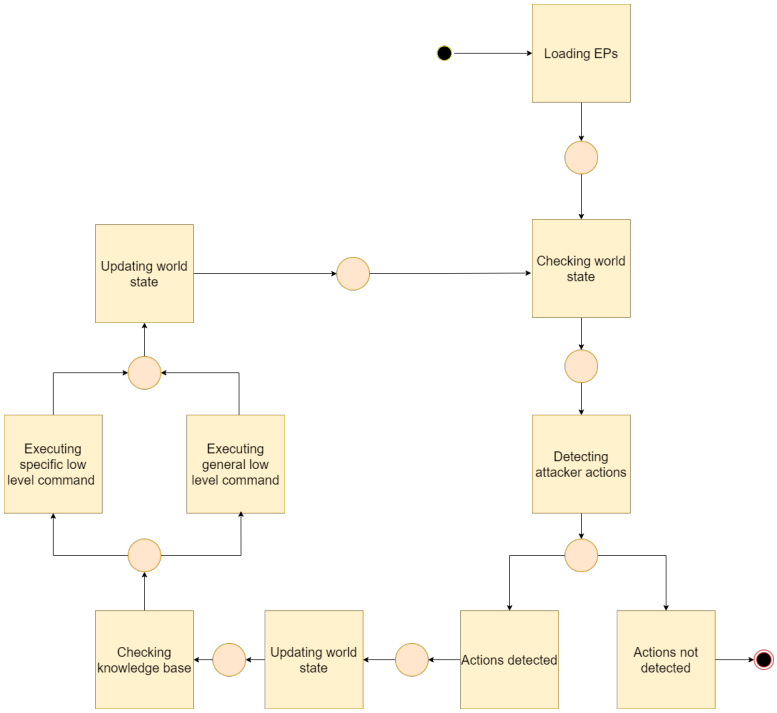


Fig. 7: Defense agent's work flow

5 Technical Implementation

5.1 Attack Agent

The attack agent is a Kali Linux automation utility that can automate most of the Kali Linux environment tools. These tools can perform red team operations such as reconnaissance, weaponization, delivery exploitation, installation, command and control, and actions on objectives. In a cybersecurity exercise environment, one or more Kali Linux machines can be deployed to perform the attacker's actions. The orchestrator has a remote Master control unit that controls these machines using a dedicated SSH connection. The Kali Linux agents and the Master control unit work in a *Botnet Command and Control* [ZM09,FSR09] manner. The attacker's actions are modeled in the DSL, which the Master control unit interprets and forwards to the Kali Linux machines as *EP*.

The Mastercontrol unit contains the *EP* of various attack stages, such as Nmap scripts for scanning and Metasploit scripts for exploitations. The resource script [Res] of Metasploit is used to perform post-exploitation on the exploited machines. An extensive logging mechanism is integrated into the Master control unit, which can collect logs from the Kali machines to confirm whether the launched attack steps were successful or not. A schematic diagram presenting the main components existing in an attack agent is shown in Figure 8. Each component represents a category of tools that can be used by the attack agent. For example, these components include scanners to collect information about the exercise environment and vulnerability executors to perform a particular attack or exploit. The results of the attacks are logged to update the world state and to inform the Master control unit about successful attacks.

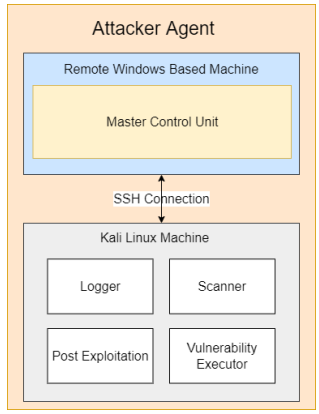


Fig. 8: Attack agent

5.1.1 Scanner The scanner can work both passively and actively. In passive mode, the scanner uses ARP resonance techniques for the identification of targets using Netdiscover [net]. When the targets are identified, it can switch into an active scanning mode and use Nmap for the identification of vulnerable services running on the system. The information of vulnerable services and their exploitation methods are provided in the DSL parameters.

5.1.2 Vulnerability Executor The vulnerability executor can take the information from the scanner to launch an attack based on predefined conditions, or it can follow the concrete action steps provided in DSL and the EP. DSL contains the

general static information for the attack agent, and the EP contains the execution plan to fulfill the attacker's goals. The conditions include finding a specific service or application signature and launching the relevant, very well-known approach. On the other hand, the concrete action steps from EP provides a repeatable execution of vulnerability exploits. The DSL constructs include (1) the tool or action name needed to be executed, (2) the agent's IP and credentials from which the action is to be executed, and (3) the specific vulnerability parameters and the target address on which the attack is executed.

5.1.3 Post Exploitation When a vulnerability is exploited, the post-exploitation module performs different tasks like credentials and memory dumps, backdoor injection, pivoting and lateral movement, and so forth. The post-exploitation steps are predefined, and because the cyber kill chain does not incorporate post-exploitation steps, concepts from MITRE Attack [MIT] are incorporated in it. For a Windows-based environment, most of the post-exploitation is performed through predefined Mimikatz commands with standard Meterpreter modules [Mim] and Powershell scripts [Pow]. For a Linux-based environment, a set of bash scripts [Lin] is used in an automated manner for post-exploitation.

5.1.4 Logger The logger logs all the different agents' activities with respect to time, the commands used, and their results in textual format. The logs are used to verify different attack agent success parameters in scanning, exploiting, and post-exploitation of the vulnerabilities in the cybersecurity exercise environment.

5.2 Defense Agent

The defense agent is a portable executable that can be deployed in a Windows-based machine. The defense agent's *EP* is generated based on the DSL instance, that is, on which system it is needed to be deployed on and what kind of action it needs to take, as presented in the Listing 2.

Listing 2: Concrete syntax for defender behavior emulation

```
[
 {
  "Defender 1": {
   "MachineIP": "192.168.81.132",
   "MachineUserID": "root",
   "MachineUserPassword": "toor",
   "OS": "Windows",
   "Parameter": "Actions1.csv"
  }
 }
]
```

The defense agent has multiple components, such as knowledge base, monitoring, analysis engine, event collector, and event responder. The knowledge base of the defender can be configured to adjust the agent behavior based on the scenario requirement, the details of which are given below and presented in Figure 9. The defense agent is deployed in a *Windows*-based environment. It uses a a custom monitor and analysis engine that collects security events from Windows event logs and that act as sensors to collect information about the environment. When an event is detected whose information is present in the defense agent knowledge base, a trigger will change its world state, resulting in the selection and execution of different responses against the attacks using the information present in the knowledge base. This defense step execution is mapped with different *CKC* phases and is conceptually represented in Figure 3.

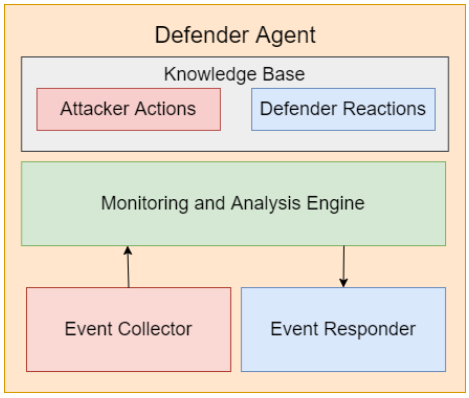


Fig. 9: Defense agent

5.2.1 Knowledge Base The knowledge base for the defense agent is a simple CSV file that contains the list of attacker actions and defender reactions. This approach of segregating the knowledge of the defense agent from the program provides the flexibility to use different levels of knowledge against the different skill set levels of attacker. For example, in a cybersecurity exercise for novice and expert hackers, the knowledge base can be adjusted to create a balanced environment [MTWP15]. We analyze some example attacker actions and defender reactions below.

5.2.2 Attacker Actions We can consider the example of *Pass the hash attack*on a remote Windows-based system. The attack will generate a 4688 Windows security event log that contains the following command line information:

Listing 3: Process command line information

```
C:\Windows\System32\wbem\WMIC.exe
```

This event and command line information can be inserted into the knowledge base to give the capability to the defender to detect such an attack signature. If the attacker is skilled enough, then the attacker can use various payload obfuscation techniques to bypass the defender's detection.

5.2.3 Defender Reaction The defender's reaction can be variable based on the scenario requirement, the knowledge for preventing the above attack can be added in the CSV file.

Listing 4: Defender reaction to the detected event

```
Reg add
↪  \HKEY_LOCAL_MACHINE\SYSTEM\CurentControlSet\Control\Terminal
↪  Server" /v fDenyTSConnections /t REG_DWORD /d 1 /f
```

The defense agent can disable the service being exploited to prevent the known attack; however, it will not be able to prevent the attacks that it has no information about. To address this, the defense agent has a monitoring and analysis engine to detect and respond to new attack patterns. Most of the attacks result in events that have similar patterns as the defined attack action; for example, opening a port from different exploits will have similar signature. The defense agent can utilize such information to prevent the attack.

5.2.4 Monitoring and Analysis Engine The monitoring and analysis engine provides the defense agent the capability to acquire and apply new knowledge. We can consider a case where the attacker was able to bypass the detection mechanism of the defender; then, the attacker will try to achieve its goals and objectives. For instance, an attacker can try to tamper with the content of the file, which was specified in the knowledge base to be monitored. This action will also generate an event log that can be traced back to the original process using our exploit chain detection algorithm [YKG19]. Using this information, the knowledge base of the defense agent can be updated automatically to kill the exploited process or close the vulnerable port using standard system commands.

Listing 5: Sample commands used for detecting file manipulation

```
type <filename.txt>
more <filename.txt>
```

Listing 6: Sample commands used for detecting user manipulation

```
whoami /all
net user
net user <user> <pass> /add
net localgroup administrators <user> /add
net user <pass> /del
```

Listing 7: Sample commands used for detecting host information gathering

```
systeminfo
driverquery
tasklist
fsutil fsinfo drives
net time
net file
net session
net use
```

5.2.5 Event Collector The event collector collects *Windows* security event logs. The event logs contain a lot of information that can be used in event correlation and for a process analysis. The event collectors parse the event logs, remove irrelevant information, and forward them to the monitoring and analysis engine for further processing. The event collector can be configured to collect the security logs from the active directory to perform network-centric cyber defense. However, currently, it is only working on Windows-based host systems.

Figure 10 represents sample attacker actions for the event collector in the defense agent in which attacker is trying to retrieve clear text WLAN credentials. One attacker action is to open a CMD shell on a compromised system; then, the attacker can fetch the WLAN profiles present on the system through a CMD command. The WLAN profiles contain information about the WIFI networks with which the system is or was connected. The attacker then fetches the clear text credentials of a WIFI network SSID through another command. The attack scenario involves the total execution of three commands. In Figure 10, it can be seen that the process ID for initiating the CMD console is 0x34f8, which then executed two child processes with the process ID 0x1a44 and 0x82c.

5.2.6 Event Responder The event responder is running on the host system with system-level privileges. It merely takes input from the monitoring and analysis engine and performs relevant operating system security tasks. These tasks involve executing *Windows command line* and *Powershell* for managing and implementing security setting changes on the system.

All these processes in Figure 10 create events in the Windows security logs and can be monitored by the monitoring and analysis engine. If the command entered

Fig. 10: Sample event collector

in the CMD console is being monitored and detected, then the parent process ID 0x34f8 will be used to kill the process. If the action of the attacker is not detected and a secure resource is retrieved, here being the credentials of the WLAN SSID, then the process is traced back, and the command parameters used for leading to the information retrieval are saved in the knowledge base to prevent future exploitation.

An attacker can bypass the defender command and event monitoring capability using different command-line obfuscation techniques [asi]. The techniques use special characters and encoding schemes to evade any pattern matching algorithms; an example of such technique is presented in Figure 11, where the command *whoami* is executed in a CMD shell of a Windows 10 machine using various obfuscation techniques. Although machine learning–based techniques are developed to detect such obfuscated commands [YK18b,HKR18], we did not integrate such a solution in the defense agent yet.

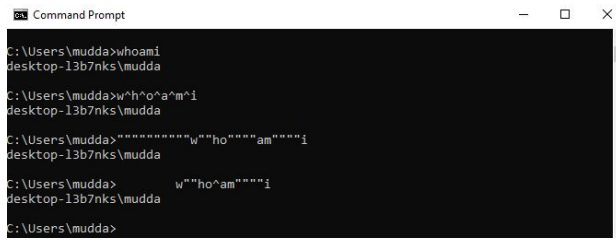


Fig. 11: Sample techniques for defense agent detection bypass

6 Experimentation

6.1 Experimental Setup of the Cyber Range

The experimental environment setup was created using our cybersecurity exercises scenario modeling language [YK19b]. The experimental setup was used to conduct three cybersecurity exercises, in which one was against the attack agent and two were against the defense agent. A total of 101 people who were from 20–25 years old and from Norway participated in the exercises; quantitative methods were used to evaluate the agents' performance.

The attack agents were used in a digital forensic and incident response cybersecurity exercise at the Norwegian University of Science and Technology [Cou], in which 84 people participated in 17 groups on a multi-subnet exercise network environment of 408 machines. Each group was provided with individual networks compromised by human attackers and attack agents. Each network contained 11 Windows- and Linux-based machines, and 2 out of 17 networks were compromised by the attack agent. The participants did not have any knowledge about the attacker and how they exploited the machine.

The defense agents were used in two cybersecurity exercises, which were conducted at the Norwegian Cyber Range [OmN]. The first exercise was a 48-hour long qualification round for the Norwegian national team for the European Cyber Security Challenge [Nor,Eur], in which 17 people participated in 5 groups on a multi-subnet exercise network environment of 75 machines. The second exercise was a 2-week-long exercise conducted during the Ethical Hacking course taught at the Norwegian University of Science and Technology, in which 84 people participated in 17 groups on a multi-subnet exercise network environment of 408 machines. Both exercises were focused on a penetration testscenario of a small organization.

The organization infrastructure has a multi-subnet network containing 11 Windows- and Linux-based machines. The participants had access to the public network through 5 dedicated Kali machines. For the attack agents, two Kali machines were used for launching attacks and creating forensic traces. In the organization's infrastructure, 2 out of those 11 machines had the same vulnerabilities, but a single

machine was running the proposed defense agent. Each group was assigned a segregated replica of the organization network and tasked with doing a pen-test. The scenario's ultimate goal was to tamper with the content of a file in the scenario machine running the defense agent, and the participants were incentivized with extra points to achieve the goal. However, they did not know the presence of the defense agent. A schematic diagram representing the experimental infrastructure is presented in the Figure 12.

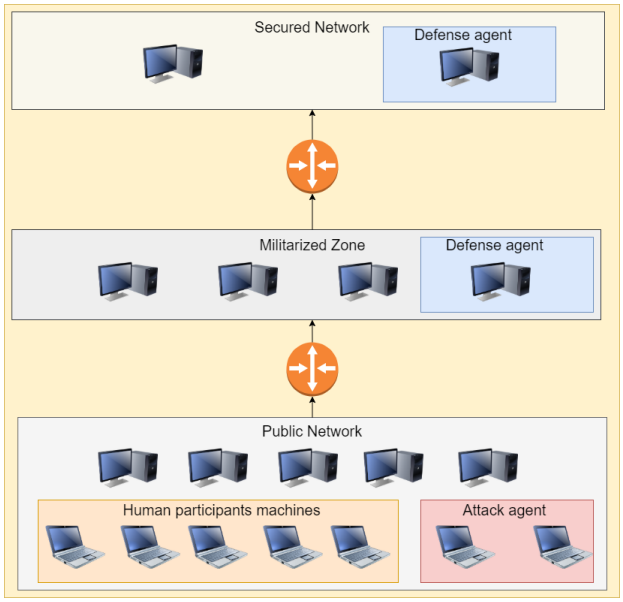


Fig. 12: Experimental setup of the cyber range

6.2 Test Cases

6.2.1 Attack Agent We defined four test cases to evaluate the attack agent performance. These test cases were selected based on the information we gathered from cybersecurity exercises [YKG20]. These include scanning the network, performing exploitation, post-exploitation, and launching attacks that were not successful. Details of the test cases are as follows:

1. Perform network scan on all the machines present in the exercise network.

2. Exploit n of the machines present in the exercise network.
3. Perform post-exploitation on n of the machines present in the exercise network.
4. Launch unsuccessful attacks on n of the machine present in the exercise network.

In Listing 8, a snippet of the test cases execution is provided. The agent's goal according to *EP* was to exploit system. The agent with the IP address of 10.10.4.96 first performed a full network scan using Nmap on sublet 10.10.1.1/24 to emulate a realistic adversary. Then, the second action was to launch a successful FTP exploit on 10.10.1.4; the FTP exploit was designed as Metasploit resource script, so it performed the post-exploitation steps automatically. After that, the agent launched an unsuccessful attack on 10.10.1.5 using the same exploit and a successful attack on 10.10.1.6 using a different exploit.

Listing 8: Concrete syntax for attacker behavior emulation

```
[
 {
  "ActiveScan": {
   "AgentIP": "10.10.4.96",
   "AgentUserID": "root",
   "AgentUserPassword": "toor",
   "Argument": "SV",
   "Target": "10.10.1.1/24"
  },
  "MetaSploit": {
   "AgentIP": "10.10.4.96",
   "AgentUserID": "root",
   "AgentUserPassword": "toor",
   "Argument": "FTPexploit",
   "Target": "10.10.1.4"
  },
  "MetaSploit": {
   "AgentIP": "10.10.4.96",
   "AgentUserID": "root",
   "AgentUserPassword": "toor",
   "Argument": "FTPexploit",
   "Target": "10.10.1.5"
  },
  "MetaSploit": {
   "AgentIP": "10.10.4.96",
   "AgentUserID": "root",
   "AgentUserPassword": "toor",
   "Argument": "Vulnserver",
   "Target": "10.10.1.6"
  }
```

```
 }
]
```

6.2.2 Defense Agent In case of the defense agent, we used three cases to evaluate their performance indicators:

1. Number of machines exploited not running the defense agent.
2. Number of machines exploited running the defense agent.
3. Files that are tampered with and that were monitored by the defense agent.

We used the knowledge base similar to Listing 4, 5, 7, and 6 in the case study for test case execution. The defender's goal according to *EP* was to detect and deny the attacker using its knowledge base. In the knowledge base, different attacker actions such as information gathering and user and file manipulation were presented, and the defender's actions against those activities were given.

6.3 Evaluation

We employed both quantitative and qualitative evaluation metrics to evaluate the agents. The quantitative metrics were used formally analyses the agent properties and to measure the efficiency of the developed artifacts in which the performance of humans was compared with the proposed agent in similar task with respect to time and resources; this is discussed in sections 6.3.2 and 6.3.3. The qualitative metrics used to measure the realism through a survey conducted on the participants who took part in the experimental scenario. The questions we asked are given in Appendix A, and its details are presented in section 6.3.4.

6.3.1 Agent Decision Modeling and Verification The EP model presented in section 4.1 helps us to analyze different test cases before their actual execution by the agents. This analysis helps us to verify different model properties like

- How high level goal can be translated into to low level actions
- Can the agent fulfill the given goal?
- What information is missing to achieve the goal?

This enables us to fine-tune agent decisions based upon model analysis for their precise execution. Listing 9 and 10 presents the implementation and logical verification conditions of the EP model for attack and defense decisions in a PyDatalog [pyD]. While listing 11 presents a sample analysis of the presented models. PyDatalog is an implementation of Datalog and in python and is used for its easy interoperability with rest of the technology stack used in this research for artifact development.

Listing 9: EP Decision Model implementation for attack

```
#Defining necessary term for the model
pyDatalog.create_terms('Goal','SubGoal','Condition','Action',
'CheckGoals','FullFilled','ActionName','ActionTarget','Commands')

#Defining root goal with sub goals of attack agent with an AND
↪  relation
+Goal('Exploit System','Reconnaissance')
+Goal('Exploit System','Exploitation')

#Defining sub goal of attack agent with an OR relation
+Goal('Exploitation','Service' or 'Configuration')

#Defining attack actions
+Action('Default', '172.168.2.1')
+Action('ping', '172.168.2.1')
+Action('Nmap', '172.168.2.1')

#Defining fulfillment requirements for attack agent
+FullFilled('Reconnaissance','ping')
+FullFilled('Reconnaissance','Nmap')
+FullFilled('Reconnaissance','WirShark')
+FullFilled('Reconnaissance','NetCat')
+FullFilled('Reconnaissance', 'Default')

#Defining conditions for attack agent

##Checking action target information is provided
FullFilled('Reconnaissance', SubGoal) &
↪  Action(SubGoal,ActionTarget) or print ('NotFullFilled')

##Condition to check is target accessible or not
+Condition('Reconnaissance', 'TargetAccess', '-i 4' )


##Validating target access
Condition(Goal, 'TargetAccess', Commands) &
↪  FullFilled('Reconnaissance', SubGoal) &
↪  Action(SubGoal,ActionTarget) or print ('NotFullFilled')


##Condition to check is network interface enabled
+Condition('Reconnaissance', 'NetworkInterface', 'netstat -i' )

##Validating network interface is enabled
```

```
Condition(Goal, 'NetworkInterface', Commands) &
↪  FullFilled('Reconnaissance', SubGoal) &
↪  Action(SubGoal,ActionTarget) or print ('NotFullFilled')


##Condition to check is arp-scan is present
+Condition('Reconnaissance', 'arp-scan-check', 'man arp-scan' )

##Validating network arp-scan is present
Condition(Goal, 'arp-scan-check', Commands) &
↪  FullFilled('Reconnaissance', SubGoal) &
↪  Action(SubGoal,ActionTarget) or print ('NotFullFilled')


##Checking action name information is provided
FullFilled('Reconnaissance', SubGoal) &
↪  Action(SubGoal,ActionTarget) or Action('Default',ActionTarget)

##Defining the attacker action to execute
+Condition('Reconnaissance', 'Active', '-sS -sV' )
+Condition('Reconnaissance', 'Passive', 'arp-scan -interface=eth0
↪  -localnet' )
+Condition('Reconnaissance', 'Default', 'nc -zv ' )
```

Listing 10: EP Decision Model implementation for defense

```
#Defining goals and sub goals of defense agent
+Goal('Prevent Attacks','Detect')
+Goal('Prevent Attacks','Deny')

#Defining defense actions
+Action('Deny', 'shell.exe')
+Action('Deny', 'port 8080')

#Defining fulfillment requirements for defense agent
+FullFilled('Detect','shell.exe')
+FullFilled('Detect','rootkit.exe')
+FullFilled('Detect','chroot.exe')
+FullFilled('Detect','port 8080')


#Check the attack action is detected or not
FullFilled('Detect', SubGoal) & Condition('Deny', SubGoal,
↪  Commands) or print ('NotFullFilled')
```

```
#Defining conditions for defense agent to prevent attack action
+Condition('Deny', 'shell.exe', 'taskkill /IM "shell.exe" /F' )
+Condition('Deny', 'Default', 'npx kill-port 8080' )
```

Listing 11: EP Decision Model analysis

```
#Establishing links between goals, sub goals
CheckGoals(Goal,SubGoal) <= CheckGoals(SubGoal,Goal)
FullFilled(Goal,SubGoal) <= CheckGoals(Goal,SubGoal)

#Establishing links between sub goals and conditions
CheckGoals(SubGoal,Condition) <= CheckGoals(SubGoal,Condition)
FullFilled(SubGoal,Condition) <= CheckGoals(SubGoal,Condition)

#Establishing links between conditions and action
CheckGoals(Condition, Action) <= CheckGoals(Action,Condition)
FullFilled(Condition, Action) <= CheckGoals(Action,Condition)

#Check a goal can be full filled to perform specific action
FullFilled(Goal,SubGoal) <= CheckGoals(Goal,Action) &
↪  FullFilled(SubGoal,Action) & (Goal != SubGoal)

#Sample analysis condition for attack agent EP decision
Condition(Goal, 'Actvie', Commands) & FullFilled('Reconnaissance',
↪  SubGoal) & Action(SubGoal,ActionTarget)

#Sample analysis condition for defense agent EP decision
FullFilled('Detect', SubGoal) & Condition('Deny', SubGoal,
↪  Commands)
```

While translating high-level goals to low-level action and task is a complex and challenging task, our model can perform it based upon the given conditions. Furthermore, the agents' different decisions were verified, highlighting the decisions that can result in goals not fulfilled or impractical. This allowed us to plan agent decisions based upon the scenario requirement. Like in some scenarios, the agents need to make wrong decisions against human adversaries to maintain realism. Similarly, in some scenarios, the agents were required to execute the actions as fast as possible, like performing dry runs on exercise infrastructure, so their decision model can be tuned to avoid unfulfilled and impractical decisions to save time. A sample decision model verification for attack and defense agent decisions highlighted in Figure 4 and Figure 5 is presented in Figure 13.

6.3.2 Attack agent Results

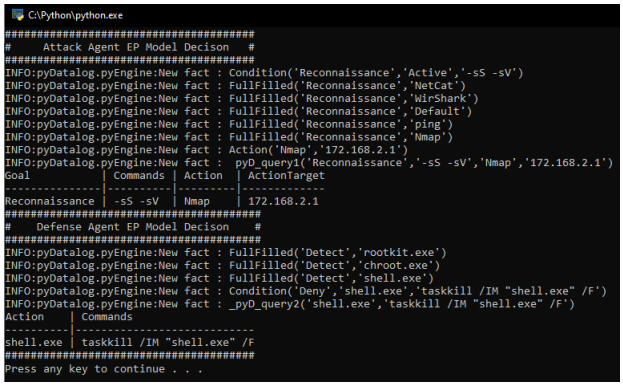


Fig. 13: Agent decision model verification

6.3.2.1 ***Task performed by humans:*** Human teams of attackers were given the task to perform penetration testing on the segregated exercise networks presented in Figure 12. They had to discover, exploit, analyze, and report the identified vulnerabilities in a penetration testing report. The penetration testing report was used for their evaluation and comparison with the attack agent's performance.

6.3.2.2 ***Task performed by agent:*** The attack agent was tasked with performing penetration testing on the similar segregated exercise network presented in Figure 12. First, the attack agent performed a full network scan of the network to emulate an adversary's scanning. Then, the attack agent created forensic evidence by launching attacks and performing post-exploitation. Out of the 11 machines, the attack agent was tasked to compromise 4 machines, perform post-exploitation, launch failed attacks on 3 machines, and launch no attack on 4 machines. The attack agent was programmed to emulate a human adversary, so it created successful and unsuccessful attack traces for forensic investigators.

6.3.2.3 ***Comparison of human and attack agent performance:*** The humans and attack agents were given same task, but the humans participated in teams of five, and most teams compromised a minimum of four machines, while one team compromised eight machines in the exercise network. The human teams took around 50 hours to complete the assigned task, while the attack agent were able to emulate the human performance in approximately 10 minutes.

6.3.2.4 ***Verification of performance:*** Human and attack agent performance was measured in the same way. The human participants in the digital forensic and incident response exercise were tasked with performing the forensic analysis of the compromised network by the human teams and the attack agent and to present

their findings in a digital forensic and incident response report. The report was used to assess the performance of the attack agent in the cybersecurity exercise, the summary of the findings are presented in Table 1:

Exercise 1			
Group task	Compromised machines identified	Post-exploitation identified	Attack attempts identified
Forensic analysis of machine compromised by humans	3	3	3
Forensic analysis of machine compromised attack agent	4	4	3

Table 1: Results of the cybersecurity exercise against the attack agent

6.3.2.5 ***Summary of the results:*** The human participants detected most of the successful attacks, post-exploitation, and unsuccessful attack attempts. The attack traces were identical to the attack traces generated by an human attacker, and the participants were not able to identify that they were generated by an attack agent. This indicates that the attack agent was providing the necessary realism and removing the need for a red team member for launching attacks, thus increasing the efficiency by automating different CKC phases like *Reconnaissance*, *Delivery*, *Exploitation*, *Installation*, *Command and Control*, and *Actions on Objectives*. It was identified that the proposed agent was working as expected and suitable in a cybersecurity exercise for creating digital forensic traces. We didn't include the factor of time in this work, which can be used by an experienced forensics analysts for distinction between human and attack agent generated forensic traces. However, over the passage of time we are incorporating the concepts of technical injects that are executed according to a scenario timeline to make the generation of forensic traces as realistic as possible.

6.3.3 Defense Agent Results

6.3.3.1 ***Tasks performed by humans:*** Teams of humans were tasked to compromise a vulnerable machine that was or was not running the defense agent. In the scenario, the human teams were incentivized with additional points to exploit a particular machine known as the *CEO Machine*, which had the same vulnerabilities present in another machine *Machine9* but was running the defense agent.

6.3.3.2 ***Tasks performed by agents:*** The defense agent was tasked to block or prevent the attacks launched by the human attackers. The defense agent had

a knowledge base to prevent particular attacks from a specific vulnerability also present in *Machine9*. Additionally, the defense agent was also monitoring a local file to prevent access to the attacker in case the attacker exploited a vulnerability not present in the defense agent's knowledge base.

6.3.3.3 ***Comparison of human and attack agent performance:*** In the exercise environment, there were no machines actively defended by human adversaries, so comparing the performance regarding efficiency of the human and defense agents is a bit difficult. However, the machines that were defended by the proposed agents were difficult to exploit compared with the undefended machines. According to IBM, the average detection time for a data breach is 206 days [Cos19]; we are not arguing that our proposed solution will drastically improve this, but we are arguing that having an active agent running in a system will restrict the attacker's actions and improve system security.

6.3.3.4 ***Verification of performance:*** In the second exercise, three out of five groups compromised the vulnerable machine not running the defense agent. In comparison, one group compromised the machine running the defense agent using a vulnerability that was not in the defense agent's knowledge base but could not tamper with the file because of the defense agent's actions. Similarly, in the third exercise, 8 out of 17 groups compromised the vulnerable machine, and 2 groups were able to compromise the machine running the defense agent using a vulnerability not present in the defense agent's knowledge base but were not able to tamper with the content of the file. The results of the experiments are presented in Table 2.

Exercise 2			
Number of Groups	Groups Exploited Vulnerable Machine	Groups Exploited Vulnerable Machine Running Defense Agent	Groups Tampered with the File
5	3	1	0
Exercise 3			
17	8	2	0

Table 2: Results of the cybersecurity exercise against the defense agent

6.3.3.5 ***Summary of the results:*** The results indicate that the defense agent created the necessary friction and added realism by preventing attacks present in its knowledge base during an operational cybersecurity exercise. The defense agent removed the need for an active blue team member during the exercise, thus increasing the efficiency by automating certain tasks in CKC like *Detect* and *Deny* for the defender. The inclusion of the defense agent made the exercise environment more dynamic and challenging because some groups were not able to compromise the machines running the defense agents.

6.3.4 Qualitative feedback from the exercise participants The qualitative feedback consists of survey responses from the 3 teams, which represent 15 students. The survey was conducted in a semi-informal way. The participants were asked to answer a list of pre-defined open-ended questions using digital communication apps, hence following COVID-19 restrictions. Fallow-up questions were asked if the answers needed further explanation. The survey was conducted in a relaxed and friendly environment, and the participants were given sufficient time to reflect on their experience and properly answer the question to avoid any biases. No personally indefinable information was collected during the survey to avoid any GDPR *(General Data Protection Regulation)*-related issues. During the survey, when we asked the participants if they found the scenario realistic, one survey participant stated the following:

Scenarios were pretty realistic for the hacking phase

This sentiment was shared by the majority of the survey participants. The exercise was conducted in two phases *Ethical Hacking* and *Incident response & Forensics*. The participants were very impressed by the complexity and dynamism of the scenario in the *Ethical Hacking* phase; this can be attributed to the presence of a dynamic defense agent in the exercise environment. However, for the *Incident response & Forensics*, they were not that impressed because the environment was static. The participants expected continuing attacks during this phase to make it more dynamic. The feedback of the participants was noted for implementing active attack execution in the *Incident response & Forensics* phase in the future. When we asked about the difficulty of the scenario, one participant stated the following:

I think it is good that the scenario is large and consist of both easy machines and more difficult ones. This allows weaker students to be able to get points and provides a challenge for stronger students with much experience. In my opinion, the project is good from a grading perspective

This sentiment was also shared by most of the survey respondents. The participants indicated that they found the machines to exploit having a variety of difficulty levels *easy, medium,* and *hard*, which allowed the participants with ranging skill sets to practice their skills. This indicates that the presence of the developed agents in the scenario provided balance in the lab, which made some machines difficult to exploit because of them having similar vulnerabilities to other easily exploitable machines. When we asked about the number of machines exploited by their teams, two teams stated that they exploited four machines, while one team exploited nine machine. Continuing from this, we asked a specific question about the machine that was not running the agent *Machine9* and the machine that was running the defense agent *CEO machine*. Two teams were not able to exploit both machines, while one team was able to exploit it *Machine9*. When we asked why they were not able to exploit, it they responded with the following:

No exactly each planned attack went through except for one where we were trying to do an smb exploitation but we couldn't figure out and came to the conclusion that it was rabbit hole and moved on

The rest of the questions were asked to improve the quality of the exercise scenario and are not relevant to this study. From the qualitative feedback, it can be concluded that the exercise scenario that incorporated our agents are quite realistic and offer the opportunity to exercise participants to practice their skills against realistic computational adversaries.

7 Discussion and Conclusion

In this work, we investigated the cyber-attack and defense agents' usage of cyber ranges for improving the realism and efficiency of cybersecurity exercise execution. We identified that such agents could provide the necessary level of friction during an exercise. For example, in a red team exercise, a defense agent will try to prevent attackers from achieving their objectives. On the other hand, in a blue team exercise, an attack agent will conduct various attacks and create forensics traces, such that the need for a human red team is reduced. This makes cybersecurity exercise execution more realistic and efficient.

We proposed EP models for specifying the agents' decision making. An EP model contains three levels of decisions: *high, medium,* and *low*. These decisions were translated using a DSL into goals, actions, and commands. We presented the workflows of the attack and defense agents to showcase how they made their decisions during the execution of a cybersecurity exercise. We employed the proposed agent-based system in cybersecurity exercises and presented their performance results in the form of a case study.

For the attack agent, we consider its performance satisfactory when applied in a semi-autonomous manner. The attack agents create realistic forensic traces during a cybersecurity exercise, which is verified by the human participants. On the other hand, our developed cyber defense agents currently have the six capabilities highlighted by Kott [KTD+18b]. However, we do not consider these agents suitable for deployment in the actual production environment for active cyber defense because they were tested in a controlled environment of cyber ranges. Yet, they can be considered a first step to achieve autonomous cyber defense. One of the limitation of the proposed defense agents is in the way they respond to attacks. If an agent detects a new attack on a specific port, it will update the knowledge base with some information about the attack's signature. Then, it will close the port or service for the next attack with a similar signature without analyzing the impact of its action, which is not suitable for real-world systems. We will address this issue in our future research, and we are planning to develop state machines that can help the defense agent take the optimal action to deal with the attacker.

In the future, we are also planning to integrate specific test cases with the attack agent to automatically perform the security assurance of systems and provide a quantitative score of the state of system security [KP18,KP19]. Finally, we plan to use our developed cyber-attack and defense agents in additional cybersecurity exercises with different network topology positions. This will help us analyze the impact of such agents at different stages of attacks and defenses, for example, an

exercise designer can use such agents in an outer network on dispensable systems to obtain the attacker's tactics and techniques for preventing attacks in the internal network.

References

[APT] *APTSimulator/test-sets at master · NextronSystems/APTSimulator*. https://github.com/NextronSystems/APTSimulator/tree/master/test-sets. – (Accessed on 01/27/2021)

[asi] *Dosfuscation*. https://i.blackhat.com/briefings/asia/2018/asia-18-bohannon-invoke_dosfuscation_techniques_for_fin_style_dos_level_cmd_obfuscation-wp.pdf - (Accessed on 01/28/2021)

[BCD+19] Braghin, Chiara ; Cimato, Stelvio ; Damiani, Ernesto ; Frati, Fulvio ; Mauri, Lara ; Riccobene, Elvinia: A model driven approach for cyber security scenarios deployment. In: *Computer Security*. Springer, 2019, S. 107–122

[Cen17] Center, JPCERT C.: *Detecting lateral movement through tracking event logs*. 2017

[CGT+89] Ceri, Stefano ; Gottlob, Georg ; Tanca, Letizia u. a.: What you always wanted to know about Datalog(and never dared to ask). In: *IEEE transactions on knowledge and data engineering* 1 (1989), Nr. 1, S. 146–166

[Cos19] *Cost of a Data Breach Study — IBM*. https://www.ibm.com/security/data-breach. Version: 2019. – (Accessed on 06/16/2021)

[Cou] *Course - Incident Response, Ethical Hacking and Forensics - IMT3004 - NTNU*. https://www.ntnu.edu/studies/courses/IMT3004#tab=omEmnet

[Dat] *Datalog: Deductive Database Programming*. https://docs.racket-lang.org/datalog/index.html. – (Accessed on 09/30/2020)

[EM17] Edgar, Thomas W. ; Manz, David O.: *Research methods for cyber security*. Syngress, 2017. – 271–297 S.

[Eur] *European Cyber Security Challenge — ECSC*. https://europeancybersecuritychallenge.eu/ - (Accessed on 01/28/2021)

[FSR09] Feily, Maryam ; Shahrestani, Alireza ; Ramadass, Sureswaran: A survey of botnet and botnet detection. In: *2009 Third International Conference on Emerging Security Information, Systems and Technologies* IEEE, 2009, S. 268–273

[Gen] *Generating Realistic Non-Player Characters for Training Cyberteams*. https://insights.sei.cmu.edu/blog/generating-realistic-non-player-characters-for-training-cyberteams/. – (Accessed on 04/19/2021)

[HC10] Hevner, Alan ; Chatterjee, Samir: Design science research in information systems. In: *Design research in information systems*. Springer, 2010, S. 9–22

[HCA+11] Hutchins, Eric M. ; Cloppert, Michael J. ; Amin, Rohan M. u. a.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. In: *Leading Issues in Information Warfare & Security Research* 1 (2011), Nr. 1, S. 80

[HKR18] Hendler, Danny ; Kels, Shay ; Rubin, Amir: Detecting malicious PowerShell commands using deep neural networks. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, S. 187–197

[HS16] Holm, Hannes ; Sommestad, Teodor: Sved: Scanning, vulnerabilities, exploits and detection. In: *MILCOM 2016-2016 IEEE Military Communications Conference* IEEE, 2016, S. 976–981

[HWD+17] Herold, Nadine ; Wachs, Matthias ; Dorfhuber, Marko ; Rudolf, Christoph ; Liebald, Stefan ; Carle, Georg: Achieving reproducible network environments with INSALATA. In: *IFIP International Conference on Autonomous Infrastructure, Management and Security* Springer, Cham, 2017, S. 30–44

[JON+15] Jones, Randolph M. ; O'Grady, Ryan ; Nicholson, Denise ; Hoffman, Robert ; Bunch, Larry ; Bradshaw, Jeffrey ; Bolton, Ami: Modeling and integrating cognitive agents within the emerging cyber domain. In: *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)* Bd. 20 Citeseer, 2015

[KMRS14] Kordy, Barbara ; Mauw, Sjouke ; Radomirović, Saša ; Schweitzer, Patrick: Attack–defense trees. In: *Journal of Logic and Computation* 24 (2014), Nr. 1, S. 55–87

[Kot05] Kotenko, Igor: Agent-based modeling and simulation of cyber-warfare between malefactors and security agents in Internet. In: *19th European Simulation Multiconference "Simulation in wider Europe*, 2005

[KP18] Katt, Basel ; Prasher, Nishu: Quantitative security assurance metrics: REST API case studies. In: *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, 2018, S. 1–7

[KP19] Katt, Basel ; Prasher, Nishu: Quantitative Security Assurance. In: *Exploring Security in Software Architecture and Design.* IGI Global, 2019, S. 15–46

[KTD+18a] Kott, Alexander ; Théron, Paul ; Drašar, Martin ; Dushku, Edlira ; LeBlanc, Benoît ; Losiewicz, Paul ; Guarino, Alessandro ; Mancini, Luigi ; Panico, Agostino ; Pihelgas, Mauno u. a.: Autonomous Intelligent Cyber-defense Agent (AICA) Reference Architecture. Release 2.0. In: *arXiv preprint arXiv:1803.10664* (2018)

[KTD+18b] Kott, Alexander ; Thomas, Ryan ; Drašar, Martin ; Kont, Markus ; Poylisher, Alex ; Blakely, Benjamin ; Theron, Paul ; Evans, Nathaniel ; Leslie, Nandi ; Singh, Rajdeep u. a.: Toward Intelligent Autonomous Agents for Cyber Defense: Report of the 2017 Workshop by the North Atlantic Treaty Organization (NATO) Research Group IST-152-RTG. In: *arXiv preprint arXiv:1804.07646* (2018)

[KV08] Kuechler, Bill ; Vaishnavi, Vijay: On theory development in design science research: anatomy of a research project. In: *European Journal of Information Systems* 17 (2008), Nr. 5, S. 489–504

[Lin] *Linux Post Exploitation Command List · mubix/post-exploitation Wiki.* https://github.com/mubix/post-exploitation/wiki/Linux-Post-Exploitation-Command-List. – (Accessed on 01/25/2021)

[Llo12] Lloyd, John W.: *Foundations of logic programming.* Springer Science & Business Media, 2012. – 1–31 S.

[Mim] *Mimikatz - Metasploit Unleashed.* https://www.offensive-security.com/metasploit-unleashed/mimikatz/. – (Accessed on 01/25/2021)

[MIT] *MITRE ATT&CK®.* https://attack.mitre.org/. – (Accessed on 01/25/2021)

[MTWP15] Mirkovic, Jelena ; Tabor, Aimee ; Woo, Simon ; Pusey, Portia: Engaging Novices in Cybersecurity Competitions: A Vision and Lessons Learned at {ACM} Tapia 2015. In: *2015 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, 2015

[Nai20] Naik, Mayur: Petablox: Large-Scale Software Analysis and Analytics Using Datalog / GEORGIA TECH RESEARCH INST ATLANTA ATLANTA United States. 2020. – Forschungsbericht

[net] *netdiscover*. https://manpages.debian.org/unstable/netdiscover/netdiscover.8.en.html. – (Accessed on 01/23/2021)

[Nor] *Norwegian Cyber Security Challenge - NCSC - NTNU*. https://www.ntnu.no/ncsc. – (Accessed on 01/28/2021)

[OmN] *Om Norwegian Cyber Range - NTNU*. https://www.ntnu.no/ncr. – (Accessed on 01/28/2021)

[Pow] *PowerShellMafia/PowerSploit: PowerSploit - A PowerShell Post-Exploitation Framework*. https://github.com/PowerShellMafia/PowerSploit. – (Accessed on 01/25/2021)

[PTE] *PTES Technical Guidelines - The Penetration Testing Execution Standard*. https://tinyurl.com/6cgn3cu. – (Accessed on 01/20/2021)

[pyD] *pyDatalog*. https://sites.google.com/site/pydatalog/home - (Accessed on 09/03/2021)

[RCA20] Russo, Enrico ; Costa, Gabriele ; Armando, Alessandro: Building Next Generation Cyber Ranges with CRACK. In: *Computers & Security* (2020), S. 101837

[red] *Aatomic-red-team: Small and highly portable detection tests based on MITRE's ATT&CK*. https://github.com/redcanaryco/atomic-red-team. – (Accessed on 01/27/2021)

[Res] *Resource Scripts — Metasploit Documentation*. https://docs.rapid7.com/metasploit/resource-scripts/. – (Accessed on 01/19/2021)

[spl] *splunk/attack_range: A tool that allows you to create vulnerable instrumented local or cloud environments to simulate attacks against and collect the data into Splunk*. https://github.com/splunk/attack_range. – (Accessed on 01/27/2021)

[Sto18] Stoecklin, Marc P.: Deeplocker: How AI can power a stealthy new breed of malware. In: *Security Intelligence, August* 8 (2018)

[TKD+18] Theron, Paul ; Kott, Alexander ; Drašar, Martin ; Rzadca, Krzysztof ; LeBlanc, Benoît ; Pihelgas, Mauno ; Mancini, Luigi ; Panico, Agostino: Towards an active, autonomous and intelligent cyber defense of military systems: The NATO AICA reference architecture. In: *2018 International conference on military communications and information systems (ICMCIS)* IEEE, 2018, S. 1–9

[TKD+20] Theron, Paul ; Kott, Alexander ; Drašar, Martin ; Rzadca, Krzysztof ; LeBlanc, Benoît ; Pihelgas, Mauno ; Mancini, Luigi ; De Gaspari, Fabio: Reference architecture of an autonomous agent for cyber defense of complex military systems. In: *Adaptive Autonomous Secure Cyber Systems*. Springer, 2020, S. 1–21

[ube] *Uber-Common:An information security preparedness tool to do adversarial simulation*. https://github.com/uber-common/metta. – (Accessed on 01/27/2021)

[YB18] Yamin, Muhammad M. ; Basel, KATT: Ethical Problems and Legal Issues in Development and Usage Autonomous Adversaries in Cyber Domain. (2018)

[YK18a] Yamin, Muhammad M. ; Katt, Basel: Inefficiencies in Cyber-Security Exercises Life-Cycle: A Position Paper. In: *AAAI Fall Symposium: ALEC*, 2018, S. 41–43

[YK18b] Yamin, Muhammd M. ; Katt, Basel: Detecting malicious windows commands using natural language processing techniques. In: *International Conference on Security for Information Technology and Communications* Springer, 2018, S. 157–169

[YK19a] Yamin, Muhammad M. ; Katt, Basel: Cyber Security Skill Set Analysis for Common Curricula Development. In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019, S. 1–8

[YK19b] Yamin, Muhammad M. ; Katt, Basel: Modeling attack and defense scenarios for cyber security exercises. In: *5th interdisciPlinary cyber research conference*, 2019, S. 7

[YK22] Yamin, Muhammad M. ; Katt, Basel: Modeling and executing cyber security exercise scenarios in cyber ranges. In: *Computers & Security* 116 (2022), S. 102635

[YKG19] Yamin, Muhammad M. ; Katt, Basel ; Gkioulos, Vasileios: Detecting Windows Based Exploit Chains by Means of Event Correlation and Process Monitoring. In: *Future of Information and Communication Conference* Springer, 2019, S. 1079–1094

[YKG20] Yamin, Muhammad M. ; Katt, Basel ; Gkioulos, Vasileios: Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. In: *Computers & Security* 88 (2020), S. 101636

[YKT+18] Yamin, Muhammad M. ; Katt, Basel ; Torseth, Espen ; Gkioulos, Vasileios ; Kowalski, Stewart J.: Make it and break it: An IoT smart home testbed case study. In: *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control*, 2018, S. 1–6

[Yue15] Yuen, Joseph: Automated cyber red teaming / DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION EDINBURGH (AUSTRALIA) CYBER AND .... 2015. – Forschungsbericht

[YUUK21] Yamin, Muhammad M. ; Ullah, Mohib ; Ullah, Habib ; Katt, Basel: Weaponized AI for cyber attacks. In: *Journal of Information Security and Applications* 57 (2021), S. 102722

[ZM09] Zeidanloo, Hossein R. ; Manaf, Azizah A.: Botnet command and control mechanisms. In: *2009 Second International Conference on Computer and Electrical Engineering* Bd. 1 IEEE, 2009, S. 564–568

[ZN20] Zaber, Matthew ; Nair, Suku: A framework for automated evaluation of security metrics. In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, S. 1–11

A Survey Question

1. Did you find the scenario realistic?
2. Did you find the scenario difficulty hard, medium, or easy?
3. How many machines did you exploited?
4. Did you find similarities between Machine9 and CEO machines?
5. Did you identify any of your attacks get blocked?
6. If yes, did you exploited both or only one and why?
7. What can be improved in the scenario?

ISBN 978-82-326-5470-3 (printed ver.)
ISBN 978-82-326-6809-0 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU

Norwegian University of
Science and Technology