

Jardar Hollås  
Petter Jørgensen  
Charlotte Larsen  
Tryggvi T. Zabelberg

# Automated Security Testing with MITRE Caldera and Azure Pipelines

Graduate thesis in Digital Infrastructure and Cyber Security  
Supervisor: Jia-Chun Lin

May 2023



Norwegian University of  
Science and Technology



Jardar Hollås  
Petter Jørgensen  
Charlotte Larsen  
Tryggvi T. Zabelberg

# **Automated Security Testing with MITRE Caldera and Azure Pipelines**

Graduate thesis in Digital Infrastructure and Cyber Security  
Supervisor: Jia-Chun Lin  
May 2023

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





# Preface

We would like to thank the SOC team at Sopra Steria, comprising Kristoffer, Truls, Joakim, and Mikael, for entrusting us with the project task and providing us with the opportunity to work on this project. We are grateful for their collaboration, expertise, and continuous support throughout the entire process. Additionally, we would like to express our appreciation to our supervisor, Jia-Chun Lin, for her guidance, encouragement, and valuable insights that greatly contributed to the success of our project. We are honored to have had the privilege to work with such a dedicated team and are grateful for the knowledge and experience gained through this endeavor.

## SAMMENDRAG

Tittel :	Automated Security Testing with MITRE Caldera and Azure Pipelines	Dato: 21.05.2023
Deltaker(e) :	Jardar Hollås Petter Jørgensen Charlotte Larsen Tryggvi T. Zabelberg	
Veileder(e) :	Jia-Chun Lin	
Evt. oppdragsgiver :	Sopra Steria	
Stikkord/nøkkel ord (3-5 stk) :	Cybersikkerhet, Pipelines, MITRE Caldera, Azure DevOps Pipelines, Sikkerhetstesting	
Antall sider/ord :	79/15127	Antall vedlegg : 6   Publiseringsavtale inngått : Åpen
<b>Sammendrag</b> Sopra Steria er et ledende teknologi- og konsulentfirma, kjent for sine digitale tjenester. SOC-teamet til Sopra Steria ønsket å automatisere sikkerhetstesting for å forbedre og effektivisere testprosessene deres. Den foreslåtte løsningen måtte bruke et rammeverk for cybersikkerhet med åpen kildekode, som også kunne integreres i deres deteksjonslab. Etter en grundig undersøkelse bestemte vi oss for det rammeverket som stemte best med våre krav. Dette var MITRE Caldera, designet for automatisk emulering av trusselaktører. Vi konfigurerte agenter og brukte APIer levert av Caldera for å automatisere sikkerhetstesting. Sikkerhetstestmiljøet som tester utføres i, implementeres med IaC-verktøyet Terraform på skyplattformen Microsoft Azure. Løsningen vi utviklet inkluderer to Pipelines, en som automatisk oppretter et testmiljø og en som gir muligheten til å kjøre ulike sikkerhetstester innenfor det. Dette dokumentet dekker først bakgrunnsteori rundt sikkerhetstesting og skytjenester. Vi fortsetter med en gjennomgang av våre satte krav, deretter design og implementering av systemet vårt. Etterfulgt er en detaljert evaluering av systemet vårt og en diskusjon av utviklingsprosessen, potensielle alternativer og til slutt videre arbeid for å gi en helhetlig løsning på problemstillingen.		

## ABSTRACT

Title :	Automated Security Testing with MITRE Caldera and Azure Pipelines	Date: 21.05.2023
Participants :	Jardar Hollås Petter Jørgensen Charlotte Larsen Tryggvi T. Zabelberg	
Supervisor(s) :	Jia-Chun Lin	
Employer :	Sopra Steria	
Keywords :	Cybersecurity, Pipelines, MITRE Caldera, Azure DevOps Pipelines, Security Testing	
(3-5)		
Number of pages/words : 79/15127	Number of appendix : 6	Availability : Open
<b>Abstract</b> Sopra Steria is a prominent technology and consulting firm, renowned for its digital services. The SOC team of Sopra Steria requested an automated security testing solution to enhance and streamline their testing processes. The proposed solution had to use an open-source cybersecurity framework, which also could be integrated into their detection lab environment. After a thorough investigation, we settled on the framework that aligned best with our requirements. That was MITRE Caldera, designed for automatic adversary emulation. We configured agents and utilized APIs provided by Caldera to create a solution for automated security testing. The Security Testing Environment in which tests are executed, is implemented with the IaC tool Terraform on the cloud platform Microsoft Azure. The system we developed includes two Pipelines, which provide the automatic construction of a testing environment and the ability to run various security tests within it. This thesis initially covers background theory around security testing and cloud computing. We continue with our specific requirements, design, and implementation. Following is a detailed evaluation of our system and a discussion of the development process, potential alternatives, and further work to provide a comprehensive solution to the problem statement.		

# Contents

<b>Abstract</b> . . . . .	<b>i</b>
<b>Sammendrag</b> . . . . .	<b>ii</b>
<b>Preface</b> . . . . .	<b>iii</b>
<b>Contents</b> . . . . .	<b>iv</b>
<b>Figures</b> . . . . .	<b>vi</b>
<b>Tables</b> . . . . .	<b>vii</b>
<b>Code Listings</b> . . . . .	<b>viii</b>
<b>Glossary</b> . . . . .	<b>ix</b>
<b>1 Introduction</b> . . . . .	<b>1</b>
1.1 Background . . . . .	1
1.2 Project Goals . . . . .	2
1.3 Limitations . . . . .	2
1.4 Project Group . . . . .	3
1.5 Thesis Structure . . . . .	4
<b>2 Background</b> . . . . .	<b>5</b>
2.1 Cloud Computing . . . . .	5
2.2 Security and Penetration Testing . . . . .	7
2.3 MITRE ATT&CK . . . . .	9
2.4 Cybersecurity Frameworks . . . . .	10
2.5 Automation tools . . . . .	12
<b>3 Related Work</b> . . . . .	<b>14</b>
<b>4 System Design</b> . . . . .	<b>16</b>
4.1 Requirements . . . . .	16
4.2 System Architecture . . . . .	19
4.2.1 Security Testing Environment Deployment Pipeline . . . . .	20
4.2.2 Adversary Operations Pipeline . . . . .	22
4.3 Use Cases . . . . .	24
<b>5 Development Process</b> . . . . .	<b>27</b>
5.1 Development Model . . . . .	27
5.2 Phases and Timeline . . . . .	28
5.3 Documentation . . . . .	30
5.4 Routines . . . . .	31
<b>6 Implementation</b> . . . . .	<b>33</b>



6.1	Automation . . . . .	33
6.1.1	Defining Infrastructure as Code with Terraform . . . . .	34
6.1.2	Integrating IaC model into an Azure Pipeline . . . . .	41
6.2	Security Testing . . . . .	50
6.2.1	Security Testing Script . . . . .	50
<b>7</b>	<b>Evaluation . . . . .</b>	<b>55</b>
7.1	Test Case . . . . .	55
7.2	Requirements and Confidence Level . . . . .	63
<b>8</b>	<b>Discussion . . . . .</b>	<b>70</b>
8.1	The Project Task . . . . .	70
8.2	Evaluating Security Frameworks . . . . .	72
<b>9</b>	<b>Closing Remarks . . . . .</b>	<b>76</b>
9.1	Learning Outcome . . . . .	76
9.2	Conclusion . . . . .	77
9.3	Further work . . . . .	78
9.3.1	Overall System . . . . .	78
9.3.2	MITRE Caldera . . . . .	78
9.3.3	Infrastructure Deployment . . . . .	79
	<b>Bibliography . . . . .</b>	<b>80</b>
<b>A</b>	<b>Contract . . . . .</b>	<b>86</b>
<b>B</b>	<b>Project Wiki . . . . .</b>	<b>93</b>
<b>C</b>	<b>Project Owner Evaluation - E-mail . . . . .</b>	<b>98</b>
<b>D</b>	<b>All Meeting Notes . . . . .</b>	<b>102</b>
<b>E</b>	<b>Project Plan . . . . .</b>	<b>141</b>
<b>F</b>	<b>Additional Listings . . . . .</b>	<b>160</b>

# Figures

1.1	Internal Penetration Testing vs. External Penetration Testing . . . . .	3
2.1	The infrastructure of server virtualization . . . . .	6
2.2	Illustration of Penetration Testing . . . . .	8
2.3	Example of MITRE ATT&CK Matrix . . . . .	10
2.4	API Facilitating Communication . . . . .	13
4.1	The System Architecture on a high-level . . . . .	19
4.2	Security Testing Environment Deployment Pipeline diagram . . . . .	21
4.3	Adversary Operations Pipeline diagram . . . . .	22
4.4	Example of structure in the result report . . . . .	23
4.5	Use Cases . . . . .	24
5.1	Project phases . . . . .	28
5.2	Azure DevOps Kanban . . . . .	30
6.1	IaC Model . . . . .	34
6.2	Data Flow from Terraform Perspective . . . . .	40
6.3	Data Flow from Pipeline Perspective . . . . .	42
6.4	Illustration of the implemented Security Testing Environment . . . . .	50
6.5	Sequence diagram of the security testing script . . . . .	51
7.1	Azure DevOps Web GUI . . . . .	56
7.2	Navigation to pipelines . . . . .	56
7.3	Running the STEDP . . . . .	57
7.4	Pipeline stage status . . . . .	58
7.5	Deployed infrastructure . . . . .	58
7.6	Caldera agent status in web interface . . . . .	59
7.7	Configuring the adversary profile ID value for the AOP . . . . .	60
7.8	Currently running operation . . . . .	61
7.9	Sample output, from ability "Current user" . . . . .	62
7.10	Solution Stored in Azure DevOps Repository . . . . .	66

# Tables

4.1	Functional Requirements . . . . .	17
4.2	Non-Functional Requirements . . . . .	18
4.3	Use Case 1 . . . . .	25
4.4	Use Case 2 . . . . .	25
4.5	Use Case 3 . . . . .	26
7.1	The definition of confidence level in this thesis . . . . .	63
7.2	Evaluation Results for the Functional Requirements . . . . .	68
7.3	Evaluation Results for the Non-Functional Requirements . . . . .	68

# Code Listings

1	A look at the providers.tf file . . . . .	35
2	The first part of the main.tf file is dedicated to referencing existing resources . . . . .	36
3	main.tf part 2: specifies configurations for the Caldera container . .	38
4	main.tf part 3: A module is included . . . . .	39
5	Preamble for the STEDP file . . . . .	41
6	Stage 1: DOWNLOAD . . . . .	43
7	Stage 2: VALIDATE . . . . .	44
8	Stage 3: Plan . . . . .	45
9	Stage 4: Deploy . . . . .	47
10	Stage 5: Agent Installation . . . . .	48
11	AOP implementation . . . . .	49
12	API call to initiate and run an operation . . . . .	52
13	API call to get the report when the operation is finished . . . . .	53
14	Do-while-loop to make sure operation is finished . . . . .	54
15	Filtering and saving the operation results into a file . . . . .	54
16	Sample report-snippet after successful operation . . . . .	62
17	Agent-Configuration.ps1 . . . . .	161
18	CalderaAgentSetup.ps1 . . . . .	162
19	1-network-interface.tf . . . . .	162
20	2-virtual-machine.tf . . . . .	163
21	Module outputs.tf . . . . .	164
22	Module variables.tf . . . . .	164

# Glossary

- AOP** Adversary Operations Pipeline responsible for executing security tests in form of a pre-existing Adversary Profile selected by the user. 19, 22, 23, 49, 50, 60, 61, 66
- API** Application Programming Interface. A set of rules and protocols that enables software applications to communicate and share data with each other. i, ii, 13, 14, 22, 34, 50–53, 65, 76
- Atomic** Atomic, in the context of security testing is a style of testing that focuses on the smallest unit of software, as in atoms. Rather than complex composed tests. 11
- Attack payload** In cybersecurity, a payload is the malicious code or instructions delivered to a target system to exploit a vulnerability or achieve a specific goal. 15
- Blue Teaming** A defensive cybersecurity group responsible for identifying and mitigating threats, as well as enhancing an organization’s overall security measures. 8
- Caldera Web Interface** The included graphical user interface of the MITRE Caldera framework, used to configure and manage tests graphically. 59, 61, 78
- CLI** Command Line Interface, such as PowerShell or the Windows CMD. 41, 43, 76
- cmdlet** A cmdlet is a lightweight, single-function command in PowerShell that performs a specific action or task. 52, 53
- container** Often known as a Docker container. A lightweight, standalone executable package that includes everything needed to run an application, including code, runtime, system tools, libraries, and settings. Can be thought of as a lightweight virtual machine. viii, 12, 38, 40, 64
- CPU** Central Processing Unit. The main processor of a computer, that runs the machine’s Operating system and applications [1]. 5

- GUI** Graphical User Interface. An interface that allows users to interact with electronic devices through graphical icons and audio indicators. 14, 25, 26, 42, 55, 76
- IaC** Infrastructure as Code. i, ii, 12, 33, 34
- image** A configuration file that includes all the dependencies, configurations, and instructions needed to create a (docker) container based on that image. 64, 78
- Kanban** Framework used to implement agile and DevOps software development. Work items are represented visually on a kanban board, allowing team members to see the state of every piece of work at any time [2]. 6, 27, 28, 30
- MSSP** Managed Security Service Provider. An organization that provides outsourced monitoring and management of security devices and systems [3]. 1
- NSM** Nasjonal Sikkerhetsmyndighet / Norwegian National Security Authority. The Norwegian government agency responsible for overseeing security services within the country. 1
- OT** Operational Technology. Hardware and software that detects or causes a change, through direct monitoring and/or control of industrial equipment, assets, processes, and events [4]. 1, 72
- PowerShell** A command-line shell and scripting language developed by Microsoft for automating tasks and managing Windows-based systems. 11, 13, 20, 22, 23, 48, 50, 52
- Purple Teaming** A collaborative approach that combines both red and blue team efforts, focusing on improving an organization's cybersecurity through continuous testing and defense enhancements. 8
- RAT** Remote Access Tool is a type of software that enables a local user to connect to and access a remote computer, server or network [5]. 11, 73
- Red Teaming** A simulated cyber attack conducted by an authorized group to evaluate and improve an organization's security posture. 8
- Scrum** An agile project management framework that helps teams structure and manage their work through a set of values, principles, and practices [6]. 6, 27, 28

- SG** Security groups delineate areas where different security measures can be applied [7]. 15
- SOC** Security Operations Center. An in-house or outsourced team of IT security professionals that monitors an organization's entire IT infrastructure in order to detect and address cybersecurity events [8]. i, ii, 1
- STEDP** Security Testing Environment Deployment Pipeline responsible for setting up and configuring the Security Testing Environment. 19, 20, 40–42, 45, 48, 49, 55, 57–59, 64
- Terraform** A popular open-source Infrastructure-as-Code tool used for automating and managing IT infrastructure. i, ii, 2, 12, 20, 28, 34, 35, 37–41, 43, 44, 46, 76, 79
- TTPs** Tactics Techniques and Procedures. The specific methods, techniques, and strategies used by cybersecurity adversaries during an attack. A term coined by MITRE. 9, 66, 74, 76
- vanilla** A product or service that is basic and has no special features [9]. 65
- Virtual Private Network** A virtual private network or VPN is a secure tunnel that allows remote users to access a private network over the internet. 78
- VM** Virtual Machine. A software-based emulation of a physical computer, allowing multiple operating systems to run concurrently on a single host machine. 5, 12, 15, 17, 19, 20, 22, 23, 39, 42, 48–50, 52, 59, 62, 76, 78

# Chapter 1

## Introduction

This chapter introduces the background of the project. It also describes the goals, limitations, and stakeholders involved in the project.

### 1.1 Background

Our client Sopra Steria is a prominent technology and consulting firm and is renowned for its digital services, software development, and consulting expertise [10]. They offer a wide range of security services through their Security Operations Center (SOC), which operates as a Managed Security Service Provider (MSSP). In the Nordic cybersecurity market, they offer consultancy, operational technology (OT), and application security services through their Cybersecurity Centre [11]. Sopra Steria is one of the qualified organizations in the Norwegian National Security Authority's (NSM) quality assurance program for incident response [12].

One of the challenges the SOC team of Sopra Steria faces is the need for efficient testing and simulation of threats to increase security. Currently, all testing in their detection lab is done manually, which can be time-consuming and prone to human error. To address this issue, Sopra Steria wants to automate parts of the testing process by implementing a solution that enables the simulation and emulation of attacks on test clients in their lab environment. The goal is to build a comprehensive solution that can be installed in their current detection lab, with documentation for installation and configuration of the chosen framework. The solution should also allow for automatic installation and testing through Azure DevOps Pipelines. Azure DevOps Pipelines is a tool integrated into the Azure Cloud platform made to assist in the automatic building and installation of software [13]. The project aims to evaluate possible open-source frameworks for the automated emulation of adversaries and provide a summary of the pros and cons of each.



## 1.2 Project Goals

The objective of this project is to develop a technical solution that can simulate and emulate attacks against one or multiple test clients in a network and provide a comprehensive report of the process. The solution is intended to streamline the security testing process of the SOC team of Sopra Steria and will be installed in their detection lab environment. It is designed for employees who have a technical background and are responsible for security testing. Our project will be made with the intention of laying the groundwork for the project owners to be able to understand how to use, integrate and potentially build upon our solution in the future.

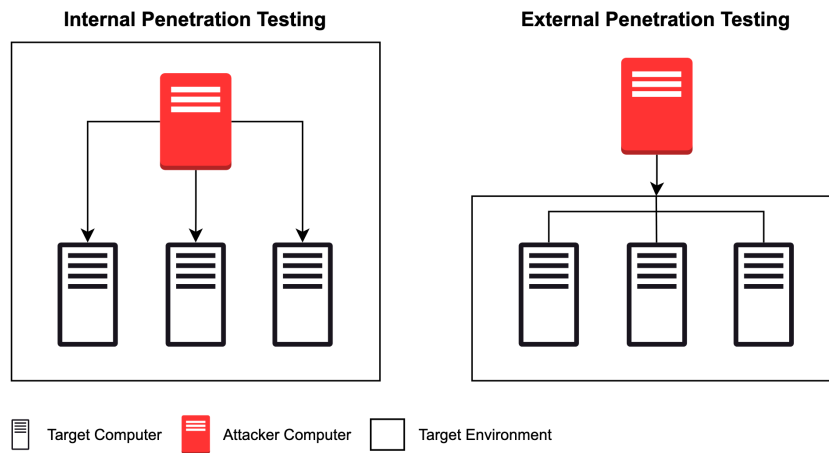
The main goal is to automate the testing process and to evaluate the most appropriate open-source security frameworks for this purpose. Additionally, the project will provide a collection of tests mapped directly to the MITRE ATT&CK security framework matrix for categorizing cybersecurity incidents [14]. Automatic deployment and testing using Azure DevOps Pipelines is a key element as well. Detailed documentation on the installation and configuration of the solution will also be provided.

## 1.3 Limitations

The final product of this project will be an Azure Git Repository, stored in Azure DevOps. It consists of the necessary elements needed to run simulation tests with MITRE Caldera, including the setup and configuration of the test environment using Terraform, the deployment of a Caldera server, and its agents onto its target virtual machine for testing. When the tests are finished, a report detailing the operation will be generated.

When executing various simulation tests, it is presumed that we possess internal access to the target environment. Therefore, the acquisition of access from outside the network or environment is not within the scope of our project.

Figure 1.1 exemplifies the main difference between internal and external penetration testing. Internal penetration testing is when the penetration tester has access to the network or environment of the target. When doing an internal penetration test, the goal is to determine how much damage an attacker can cause behind the firewall and outer defenses of an organization. In comparison, the penetration tester does not have access to the target's network or environment when performing external penetration testing. The goal of external penetration testing is to measure the security of the organization's firewall and external security measures [15].



**Figure 1.1:** Internal Penetration Testing vs. External Penetration Testing

The solution runs in the detection lab of Sopra Steria. Their detection lab is set up with proper rules and configuration for the security solutions Defender for Cloud and Sentinel. These are highly configurable security tools integrated into the Azure Cloud platforms, including the detection lab. Setting up and configuring these security solutions on a higher level than the base configuration will not be a priority during our project.

When starting the project work, we were uncertain if we would be capable of making our own program and security tests. Thus, we initially assumed that we would focus on using and integrating tests included in recognized cybersecurity frameworks such as MITRE Caldera and Metasploit. We consider this to be the most appropriate way of meeting the requirements of our project while minimizing the risk associated with developing our own tests. Our goal is therefore to integrate and efficiently use these tests and frameworks in the detection lab for automation.

Chapter 4.1 describes the specific requirements for this project. Requirement NF7 specifically, requires us to evaluate different open-source cybersecurity frameworks. However, there are numerous frameworks available online that offer a wide range of options within cybersecurity. Considering this, we decided to focus our research on well-known open-source frameworks that strictly fit our requirements. This is discussed further in Chapter 8.2.

## 1.4 Project Group

Our clients, also referred to as the project owners, are Truls Dahlsveen, Joakim Fauskrud, Kristoffer Pettersen, and Mikael Vagnes from Sopra Steria. Jia-Chun Lin, associate professor at NTNU in Gjøvik, is the supervisor for the project group.

The project group, consists of four members: Petter Jørgensen, Jardar Hollås, Trygvi T. Zabelberg, and Charlotte Larsen. We are all students in the bachelor's program Digital Infrastructure and Cyber Security [16] at NTNU in Gjøvik. We have taken relevant courses in computer science fields about infrastructure, networking, cybersecurity, penetration testing, scripting, automation, and more. Some of us have also worked as software engineers and security analysts. The diverse backgrounds and skill sets of the members in the group will provide an enriched development environment and enhance the overall quality of the end product.

Throughout the project we have made use of experience obtained through various subjects provided by our study program, an online course recommended by the project owners, and experience several of the group members have attained through part-time- or summer jobs. 'Intermediate Purple Teaming' is an online learning path provided by AttackIQ Academy [17]. This course was recommended by the project owners and equipped us with an introduction to the concepts of red, blue, and purple teaming, attack simulation, and more. The NTNU course 'INFT2504 - Cloud services as a business' provided an introduction to Microsoft Azure and how to navigate it and its many sub-services and features.

## 1.5 Thesis Structure

- **Chapter 1 - Introduction:** Description of the background, project goals, limitations, and structure of the thesis.
- **Chapter 2 - Background:** Explanation of concepts relevant to our project.
- **Chapter 3 - Related Work:** Discussion of research or academic work we have taken inspiration from or otherwise made use of.
- **Chapter 4 - System Design:** Description of the system architecture and design, including requirements and use cases.
- **Chapter 5 - Development Process:** Explanation of how the development process took place, including routines, work methods and meetings.
- **Chapter 6 - Implementation:** Covers the specific implementation of our project and its code.
- **Chapter 7 - Evaluation:** Evaluation of each requirement against our actual solution, including a test case and feedback from our client.
- **Chapter 8 - Discussion:** Discussion about various decisions made throughout the project, and a thorough review of cybersecurity frameworks.
- **Chapter 9 - Closing Remarks:** Conclusion, learning outcome and suggested further work.

## Chapter 2

# Background

In this chapter, we will explore fundamental concepts of cloud computing, penetration testing, and automation. Furthermore, we will present an overview of open-source cybersecurity frameworks relevant to our work.

### 2.1 Cloud Computing

Cloud computing is the delivery of different computing services which can be accessed on-demand over the Internet. These include services such as servers, databases, software, analytics, and more [18]. The foundation of cloud computing is a process called virtualization; a process that utilizes the hardware of a physical computer together with software, to create multiple virtual computers, commonly known as virtual machines (VM). Each of these VMs run their own Operating System (OS) and have the same functions as a normal computer [19].

Figure 2.1<sup>1</sup> is an example of how the virtualization infrastructure for a server can look like. The hardware is the physical server components, such as the Central Processing Unit (CPU), memory, network, and disk drives [20]. The Hypervisor is the software used to allocate resources between the VMs and the physical computer. It ensures that the different VMs have access to the physical resources they need and that they do not interfere with each other's memory space [19]. Each VM is made up of virtual hardware, a guest OS, and application. Where virtual hardware is the allocated resources for the VM and guest OS is the OS that the VM is running [21].

---

<sup>1</sup>An overview of types of virtualization in cloud computing, <https://www.znetlive.com/blog/virtualization-in-cloud-computing/>

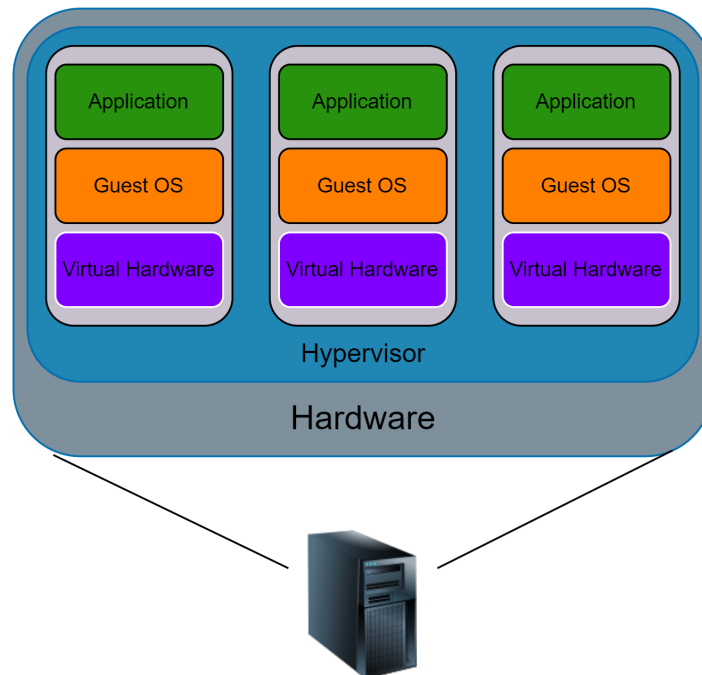


Figure 2.1: The infrastructure of server virtualization

### Microsoft Azure

In this project, we have used Microsoft Azure to deploy and run the necessary infrastructure needed to perform automated security testing. Azure is a cloud computing platform and service offered by Microsoft. It provides a wide range of cloud-based services and allows businesses to build, deploy, and manage applications and services through Microsoft-managed data centers. Azure supports various programming languages and provides monitoring of applications, as well as security, to ensure data protection [22].

One of the more important Azure components that we used during the project is Azure DevOps. It is a cloud-based platform within Microsoft Azure that offers a variety of features for the development, storage, deployment, and maintenance of software. In our project we primarily use Azure DevOps for storage of project files, in 'Repos', and to host our Pipelines. The DevOps environment of an organization is commonly split into multiple projects, with each project having access to the features Azure DevOps offers. Within Azure DevOps is the feature **Boards**, which enables the project group to follow popular development models like Scrum or Kanban; **Repos** (short for Repositories), a Git-style tool for storing and version-controlling code and files; **Pipelines**, allowing for deployment to various production environments such as Azure Cloud or other Cloud service providers [23].

## 2.2 Security and Penetration Testing

Security testing is a process that assists organizations in identifying vulnerabilities and flaws in their security systems. It entails a variety of techniques such as vulnerability scanning, code review, and penetration testing [24]. Among these, penetration testing is a commonly used approach to simulate or emulate an attack on a system and find flaws in a safe and controlled manner [15].

Penetration testing is conducted by ethical hackers who attempt to breach an organization's security safeguards in a controlled setting. This can assist enterprises in detecting vulnerabilities before bad actors exploit them. Furthermore, the insights gathered through penetration testing can give useful information about an organization's overall security posture. This may be used to patch vulnerabilities and increase security defenses, therefore protecting sensitive data and preventing data breaches [15].

Figure 2.2 illustrates this process. An ethical hacker executes an attack on a network or computer and tries to avoid or suspend its defenses or antivirus. If succeeded, the ethical hacker can potentially insert a harmless virus and extract information from the target. This information can then be documented and used to improve the defenses. If the attack fails, it could indicate that the defenses are effective against that type of attack. The attacker can then change their methods and try other attack vectors. Using logs, and monitoring both the attacking and defending systems, an organization can learn about and improve its defenses in collaboration with the ethical hacker.

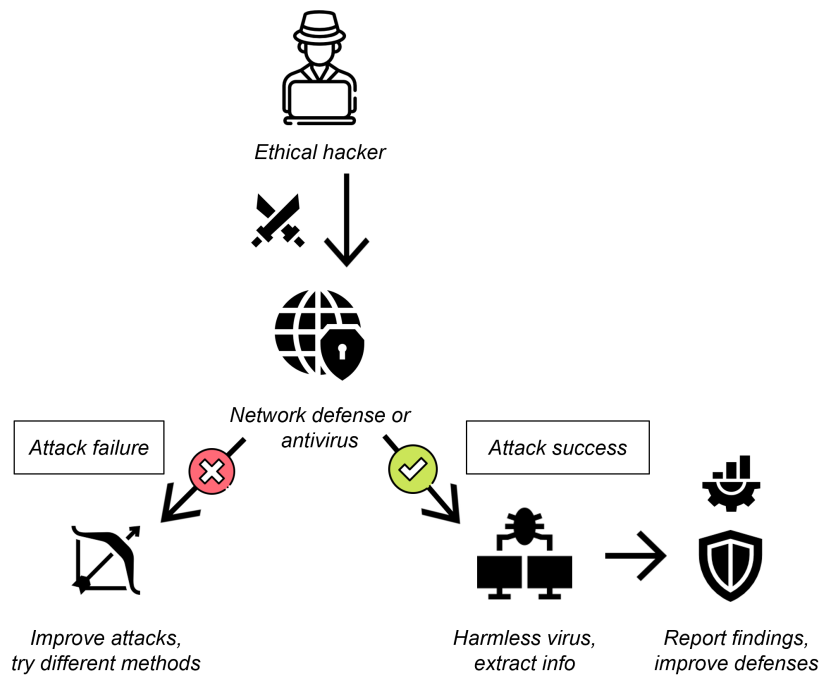


Figure 2.2: Illustration of Penetration Testing

### Red Teaming and Blue Teaming

Red teaming and blue teaming, are terms originating from military training operations where reds represent attackers and blues represent defenders [25]. Combining red teaming and blue teaming is a collaborative and observational cybersecurity practice that simulates targeted network and system attacks. The attackers on the red team use a variety of tactics to hack the system, while the defenders on the blue team focus on detecting and defending against attacks. The process of using both red and blue teaming is called Purple Teaming, which is any method, process, or activity that leverages collaboration between the red and blue aspects of organizational security [26]. It can be an efficient method of completing a comprehensive security posture assessment, allowing the organization to identify and fix vulnerabilities. Red and blue teaming should be used by organizations that want to always maintain a consistent cyber threat landscape. This approach can be used by organizations to proactively identify and deal with weaknesses, as well as identify the overall security posture, thereby protecting their valuable assets from cyber threats and adversaries.

## 2.3 MITRE ATT&CK

MITRE ATT&CK [14] is a security framework for recognizing and categorizing attackers and the Tactics, Techniques, and Procedures (TTPs) they use in cyberattacks. The framework consists of a matrix that details numerous strategies and approaches utilized by attackers at each stage of an attack. Each tactic and technique is assigned a unique identification to facilitate tracking and categorization. Specific tools and methodology, including solutions or countermeasures, are also provided here, which has made it a widely adopted tool in the cybersecurity industry [27].

The MITRE ATT&CK framework is maintained by the MITRE Corporation [28], a not-for-profit organization dedicated to advancing scientific and technological knowledge in a variety of domains, including cybersecurity [29].

### MITRE ATT&CK Matrix

The MITRE ATT&CK Enterprise Matrix, part of the ATT&CK framework, maps the TTPs of adversaries onto an interactive matrix [30] as shown in Figure 2.3<sup>2</sup>. The first row contains tactics such as Reconnaissance, Resource Development, and Initial Access. The columns beneath these highlight some common techniques within these tactics. Inside these techniques, we can also find specific procedures which describe a practical way of carrying out that technique, including relevant tools and code.

---

<sup>2</sup>An overview of the MITRE ATT&CK Matrix, <https://attack.mitre.org/matrices/enterprise/>



Reconnaissance	Resource Development	Initial Access	Execution	Persistence
10 techniques	7 techniques	9 techniques	13 techniques	19 techniques
Active Scanning (3)	Acquire Infrastructure (7)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (5)
Gather Victim Host Information (4)	Compromise Accounts (3)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs
Gather Victim Identity Information (3)	Compromise Infrastructure (7)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (14)
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)
Gather Victim Org Information (4)	Establish Accounts (3)	Phishing (3)	Inter-Process Communication (3)	Browser Extensions
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary
Search Closed Sources (2)	Stage Capabilities (6)	Supply Chain Compromise (3)	Scheduled Task/Job (5)	Create Account (3)
Search Open Technical Databases (5)		Trusted Relationship	Serverless Execution	Create or Modify System Process (4)
Search Open Websites/Domains (3)		Valid Accounts (4)	Shared Modules	Event Triggered Execution (16)
Search Victim-Owned Websites			Software Deployment Tools	External Remote Services
			System Services (2)	
			User Execution (3)	

Figure 2.3: Example of MITRE ATT&CK Matrix

## 2.4 Cybersecurity Frameworks

Cybersecurity frameworks are structured approaches used to identify, evaluate and manage cybersecurity risks [31]. They provide a common language for discussing and analyzing cyber threats and are especially useful when comparing adversary behavior across the cybersecurity landscape. Frameworks such as MITRE ATT&CK help us identify and make use of recognized approaches to test and defend against a multitude of known threats. Others provide specific, easy-to-use software, step-by-step methods, or code examples which assist in the automation of security testing.

Open-source refers to the characteristic that a framework or software is free to use, modify, and distribute, and its source code is publicly available [32]. This provides transparency and enables the community to collaborate on development, making it a valuable resource for building custom security automation workflows and improving cybersecurity defenses. Because of the above benefits and the requirement from our client mentioned in Chapter 4.1, our final solution will incorporate open-source frameworks and software.

There is a vast amount of options when considering open-source cybersecurity frameworks. As mentioned in Chapter 1.3, we will not be able to cover all frameworks in the field. A selection of relevant frameworks and their background can be seen below. Further frameworks and a detailed discussion of each are provided in Chapter 8.2.

### **Atomic Red Team**

Atomic Red Team is an open-source framework that contains a collection of tests aimed to imitate real-world attacks and detect security weaknesses. Atomic refers to atoms, which in this context means to focus on the smallest unit of software tests rather than complex, composed tests. These tests are mapped directly to the MITRE ATT&CK framework and enable the user to assess the security posture of an organization and find vulnerabilities that need to be fixed [33]. Specifically, it provides us with useful PowerShell code and scripts we can run to test security solutions on different platforms.

### **MITRE Caldera**

The Atomic Red Team tests can be run on their own but are also easily integrated into other tools such as MITRE Caldera. MITRE Caldera is an open-source, highly customizable security framework designed to emulate the behaviors of threat actors and simulate cyberattacks to test the defenses of an organization [34]. Caldera emulates various adversarial activities with remote access tools (RAT) installed on infected machines that are controlled by a master [35]. It is built with connections to the MITRE ATT&CK framework and offers a suite of tools to create and execute cyberattacks, gather and analyze data, and generate reports. The Atomic Red Team tests can be implemented and used as part of our solution through plugin functionality provided by MITRE Caldera [36]. A Caldera operation is a function in which a collection of Atomic Red Team tests or other tests can be run in sequential order, and where a report of the results is generated afterward [37]. These operations can be custom-built, or based on built-in adversary profiles which mimic real attacks from known adversaries, which is where our focus is. The official documentation of MITRE Caldera defines adversary profiles as "groups of abilities, representing the tactics, techniques, and procedures (TTPs) available to a threat actor. Adversary profiles are used when running an operation to determine which abilities will be executed" [38].

### **Metasploit**

Further relevant frameworks include Metasploit, which is a penetration testing framework developed by Rapid7 [39]. It is designed to help security professionals and ethical hackers discover and exploit vulnerabilities in target systems, to identify and patch security flaws before attackers can exploit them. Metasploit can

also be integrated with MITRE Caldera to expand the number of available security tests significantly [40].

## 2.5 Automation tools

Automating cybersecurity tasks can significantly enhance the security posture of an organization by reducing workload, improving accuracy, and allowing teams to focus on more strategic tasks. Automation tools are software programs that facilitate the efficient and streamlined execution of tasks. From routine tasks such as identifying threats to complex operations such as incident response and vulnerability management, automation tools come in various forms. These range from basic scripts to advanced systems that employ machine learning and artificial intelligence [41].

### Terraform

As part of our efforts to automate our cybersecurity processes, we have utilized Terraform, an Infrastructure as Code (IaC) tool developed by HashiCorp [42]. Terraform is used for managing cloud infrastructure resources as it is compatible with all major cloud service providers, including Amazon Web Services, Microsoft Azure, and Google Cloud Platform. The declarative syntax language of Terraform allows us to specify the desired end state for infrastructure resources, while the platform-specific methods such as Azure handle the steps required to achieve that desired state.

### Pipeline

A Pipeline [13] is a set of automated processes organized into steps or stages. Pipelines are used for continuous integration (CI) and continuous delivery (CD) of software. It is commonly used to allow developers to verify the compatibility of new code after pushing changes to a repository and then to deploy these changes to the production environment. A typical Pipeline may have the following stages:

1. **Build:** During this stage the Pipeline environment will be prepared and deployed onto a 'Build Agent'. A Build Agent is a VM or container specialized for executing the Pipeline operation. The agent will download and compile the source repository before continuing to the next defined phase.
2. **Test:** A Pipeline will commonly have a phase where it uses various testing techniques to verify that the newly pushed code does not cause unforeseen issues. In other words, to make sure that the new code is properly integrated into the existing solution.
3. **Release:** After verifying the new code in a testing environment, the Pipeline may also automatically deploy the new code to production.

4. **Monitor:** Some advanced Pipelines may continuously monitor the functionality of the Production environment. In case of a sudden failure, the Pipeline might attempt to restart the production environment automatically, or even revert to a previous safe version if necessary.

### Coding and Scripting

To clarify any discrepancies, we distinguish between the two traditionally similar terms coding and scripting. Coding is the process of writing instructions in a programming language to build software from scratch [43]. In contrast, scripting involves writing sequences of instructions or commands that are executed by a computer program to automate particular processes or tasks [44]. While coding involves creating complete software applications, scripting is focused on automating repetitive tasks such as installing or configuring software (or executing security tests) and is a key element of our project. A popular cross-platform scripting language used for automation is PowerShell [45]. It has many use cases, with installation and configuration of software being one which is relevant for us.

### API

An Application Programming Interface (API), is a set of defined rules that enable different applications to communicate with each other. IBM defines it as an intermediary layer that processes data transfers between systems, letting companies open their application data and functionality to external third-party developers, business partners, and internal departments within their companies [46]. It can be thought of as a translator that takes in code from the external client in a language the client understands and translates it into a language the server application understands. As illustrated in Figure 2.4, the API acts as an intermediary between the client and server applications. In our solution, we use REST APIs. A REST API is an API that conforms to the design principles of the REST, or "representational state transfer" architectural style [47]. This is an industry-acknowledged standard for designing APIs.

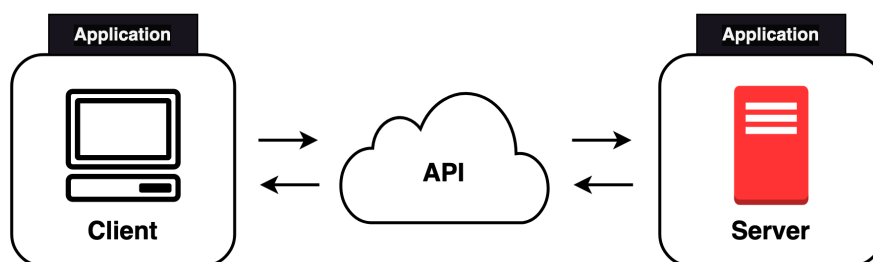


Figure 2.4: API Facilitating Communication

## Chapter 3

# Related Work

In this chapter, we will examine existing literature and research that is relevant to our project. These works include topics related to cybersecurity, automated red teaming, and the usage of MITRE Caldera, among others. By analyzing and understanding the findings and approaches of these works, we aim to contribute to the development and improvement of our own project.

In an NTNU master thesis, "Automated Red Teams in Maritime Cybersecurity Exercises" [48], Solli A.L. applied the Atomic Red Team framework to run automated security tests against a maritime testbed environment. The author describes testbeds as any system that is designed to replicate the essence of a real system. This was used to replicate a maritime environment for testing purposes for the project. The tests were created with the intention of being used in a red team versus blue team cybersecurity exercise between students. Applying MITRE Caldera for security testing was explored in the thesis, however, it was decided against due to it being regarded as too complex for their use case. Detailed attack scenarios made up of tests from the Atomic Red Team framework were used in the master thesis to present a realistic attack scenario to the blue team in the cybersecurity exercises. The blue team were the exercise participants with the responsibility of protecting the aforementioned testbeds. In the thesis, the author chose to only use the Atomic Red Team library when conducting tests.

In contrast, we decided to use the cybersecurity framework, MITRE Caldera. The Caldera framework integrates tests from the Atomic Red Team library, however, it encompasses further functionality beyond these tests. It enables agent-server functionality which allows us to execute tests from a server against a target host. When completed, results from these tests are gathered on the server. This is all accessible through MITRE Caldera's web Graphical User Interface (GUI) or API.

In M. Nachaat's research article [49] MITRE Caldera was used to showcase methods of bypassing the built-in Windows security measures, such as Windows Defender Antivirus and Firewall, using the initial access tactic [50]. This tactic, as

defined in MITRE ATT&CK refers to the method used by hackers to gain access to a system or network by exploiting a vulnerability or misconfiguration. The Attack payload remained undetected and showed how hackers can exploit vulnerabilities to evade security measures. However, the study had some limitations. First off, it only explored the initial access tactic for bypassing security. No further tactics were tested, and their relevancy for us is therefore lower. Another limitation of the article was that the reverse shell back from the victim device was only possible through the command used if the CALDERA platform was deployed. Despite this, the study provided valuable insights into how hackers can bypass Microsoft Windows Security using MITRE Caldera.

In a research paper written by Seongmo An et al. [51], the application of automated cloud security is explored with the proposed tool CloudSafe. The solution is described by the authors as a combination of various tools and frameworks intended to automate the security assessment process in the cloud. The research paper highlights the separation of privilege [52] between different stakeholders as a significant challenge when developing security solutions in the cloud. The issue was cited as being related to the difficulty in implementing the tools due to them not having enough privileges to access necessary information. In response to this, the use of Security Groups (SGs) was proposed to solve the problem. In short, while it may vary between different cloud providers, Security Groups are a mechanism used to segment access and rights to different resources in a cloud environment [7]. Using this, the authors delegated the necessary access to the various tools and frameworks to ensure they worked as intended. Furthermore, the paper mentions that the overall goal of CloudSafe is to generate reports which could be used to understand the security posture of the cloud infrastructure.

CloudSafe, the proposed solution put forward in the aforementioned research paper is similar to the solution we are developing in that they both work on securing assets in the cloud and both generate reports with the results of their operations. The key difference between CloudSafe and our solution is the focus area, where CloudSafe measures the overall security posture of the cloud infrastructure, while our solution measures the security of individual VMs in a cloud environment. Another difference worth mentioning regards access to the tools and frameworks in the cloud. Although both our solution and CloudSafe use Security Groups to function correctly, it caused less of an issue for our project due to the number and scale of tools and frameworks being smaller in our project.

## Chapter 4

# System Design

This chapter introduces the infrastructure, design, and requirements for the main components of our solution. As specified below in Requirement NF6 from Table 4.2, our solution must be developed, and all resources be hosted, in the Azure Environment of Sopra Steria. In the following sections, we will go over the requirements, the system architecture, and the use cases for our project.

### 4.1 Requirements

The requirements are divided into functional and non-functional sections describing how each part relates to each aspect of our solution. Functional requirements outline the system's core capabilities and processes, indicating what the technical system specifically is meant to do when used. Non-functional requirements focus on quality, usability, and other aspects outside of purely technical abilities. This ensures that it operates efficiently according to our client's needs and that the additional external requirements are met, such as documentation and evaluation [53]. Tables 4.1 and 4.2 below go into the details of the functional and non-functional requirements for the project.

Table 4.1: Functional Requirements

No.	Requirement Description
F1	<p data-bbox="391 842 1295 875"><b>Automatic Deployment of Infrastructure via Azure DevOps Pipelines</b></p> <p data-bbox="391 880 1295 1012">One of the main requirements of the project was the automatic deployment of infrastructure using Azure DevOps pipelines. This implies the creation of VMs and the installation and configuration of necessary software should be performed using Azure DevOps pipelines.</p>
F2	<p data-bbox="391 1025 1295 1093"><b>The system should be able to autonomously perform security tests within the detection lab</b></p> <p data-bbox="391 1097 1295 1232">The solution should be able to perform security tests within Sopra Steria's detection lab independently. When configured, the solution has to be able to run security tests against the project owner's selected targets without any manual interference by the user.</p>
F3	<p data-bbox="391 1245 1295 1312"><b>Users should be able to define the tests or adversary profiles that the system executes</b></p> <p data-bbox="391 1317 1295 1444">When completed the users of the solution, the security testers, should be able to define and select what tests they want to execute. Furthermore, they need to have the ability to select what adversary profile the system should execute.</p>



Table 4.2: Non-Functional Requirements

No.	Requirement Description
NF1	<p data-bbox="421 658 1174 692"><b>Open-Source solution; non-copyrighted and customizable</b></p> <p data-bbox="421 696 1235 763">The solution needs to be open-source in order to make sure that it can easily be modified, adapted and shared by its users.</p>
NF2	<p data-bbox="421 770 1225 837"><b>Documentation for security tests should be mapped to MITRE ATT&amp;CK</b></p> <p data-bbox="421 842 1155 943">The security tests should be mapped to the MITRE ATT&amp;CK cybersecurity framework ensuring that test results are easy to share, understand and compare.</p>
NF3	<p data-bbox="421 949 826 983"><b>Documentation for installation</b></p> <p data-bbox="421 987 1244 1055">A short and concise description explaining how to use the solution. Needs to be written in the Azure DevOps Wiki.</p>
NF4	<p data-bbox="421 1061 1219 1128"><b>The code for the solution must be stored in an Azure DevOps Repository made for the purpose</b></p> <p data-bbox="421 1133 1230 1200">Code and other related files used for the project need to be stored in the projects Azure DevOps repository.</p>
NF5	<p data-bbox="421 1207 1145 1240"><b>The Pipeline must be hosted in Azure DevOps Pipelines</b></p> <p data-bbox="421 1245 1244 1346">The Pipelines, which are responsible for executing tests, deploying, and configuring infrastructure, have to be hosted and created using the Pipelines module of Azure DevOps.</p>
NF6	<p data-bbox="421 1352 1193 1420"><b>The Infrastructure for the solution must be deployed to the Sopra Steria Azure Cloud Environment</b></p> <p data-bbox="421 1424 1251 1491">When the solutions infrastructure is deployed, it has to be deployed in Sopra Steria's Azure Cloud development environment.</p>
NF7	<p data-bbox="421 1498 1222 1532"><b>Evaluation of frameworks for automatic cybersecurity testing</b></p> <p data-bbox="421 1536 1182 1637">The most compatible cybersecurity framework should be used for developing the solution. Therefore it is important that an evaluation of different frameworks takes place.</p>

## 4.2 System Architecture

This section covers the system architecture for our solution. The system is made up of three main components, two of them being Azure DevOps Pipelines and the last one being the Security Testing Environment. The first pipeline, the Security Testing Environment Deployment Pipeline (STEDP) is responsible for orchestrating and configuring the last component, the Security Testing Environment. Which means it creates and configures the testing environment. Lastly, there is the Adversary Operations Pipeline (AOP). It is used for running security tests on the VM inside the created environment. We will go over the operations inside the two pipelines in the following subsections.

Figure 4.1 illustrates the high-level system architecture of our solution. The numbered arrows in the figure indicate the sequence in which the actions must be executed. Detailed descriptions of these actions can be found in the list below:

### 1. Executing the STEDP

The user starts the STEDP from inside the Azure environment of Sopra Steria.

### 2. STEDP sets up the Security Testing Environment

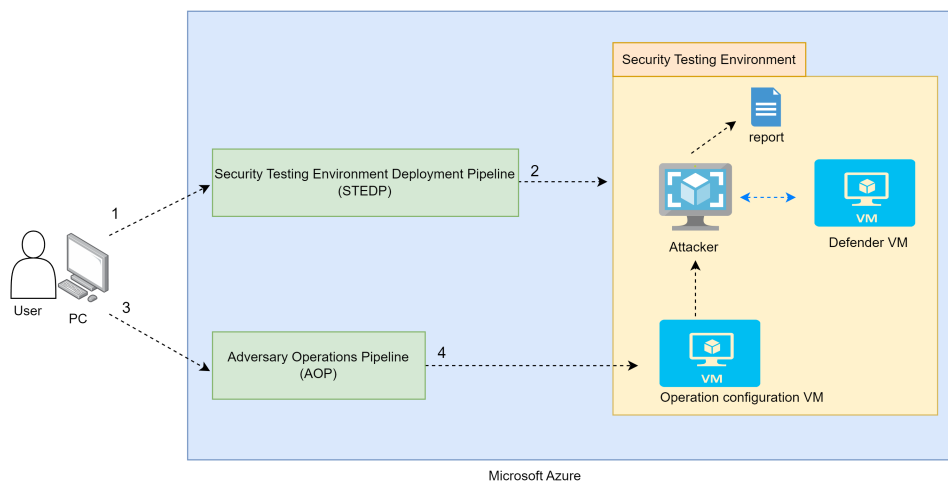
STEDP configures and initiates the Security Testing Environment.

### 3. Executing the AOP

As soon as the Security Testing Environment is operational, the user can start the AOP from inside the Azure environment of Sopra Steria.

### 4. The AOP operates inside the Security Testing Environment

AOP runs a collection of security tests defined by the user, on the VM inside the Security Testing Environment.



**Figure 4.1:** The System Architecture on a high-level

### 4.2.1 Security Testing Environment Deployment Pipeline

The STEDP is the first pipeline the user executes and is made up of five stages: Download, Validate, Plan, Deploy, and Agent Installation. The pipeline is in charge of configuring and initiating the Security Testing Environment. It also configures communication between the components deployed inside the environment. Figure 4.2 illustrates how the STEDP operates. The description for the actions can be seen in the list below:

1. **Executing the STEDP**

The user starts the STEDP from inside the Azure environment of Sopra Steria.

2. **Download stage**

STEDP downloads configuration files from an Azure DevOps repository.

3. **Validate, Plan, and Deploy stage**

STEDP starts by initializing and validating Terraform. Dependencies needed for executing the code are installed, then validated in order to make sure that everything will work properly. The currently running environment is then compared against the configuration files to identify any disparities. This guarantees that the environment receives the same components as specified in the configuration file during the deployment to the environment.

4. **Setting up the Security Testing Environment**

STEDP uses Terraform to deploy the Security Testing Environment.

5. **Agent Installation stage**

After the Security Testing Environment is operational, the STEDP runs a PowerShell script on the VMs inside the environment.

6. **Communication between VMs**

The VMs become Caldera agents that communicate with the Attacker, which is also the Caldera Server that monitors the security tests.

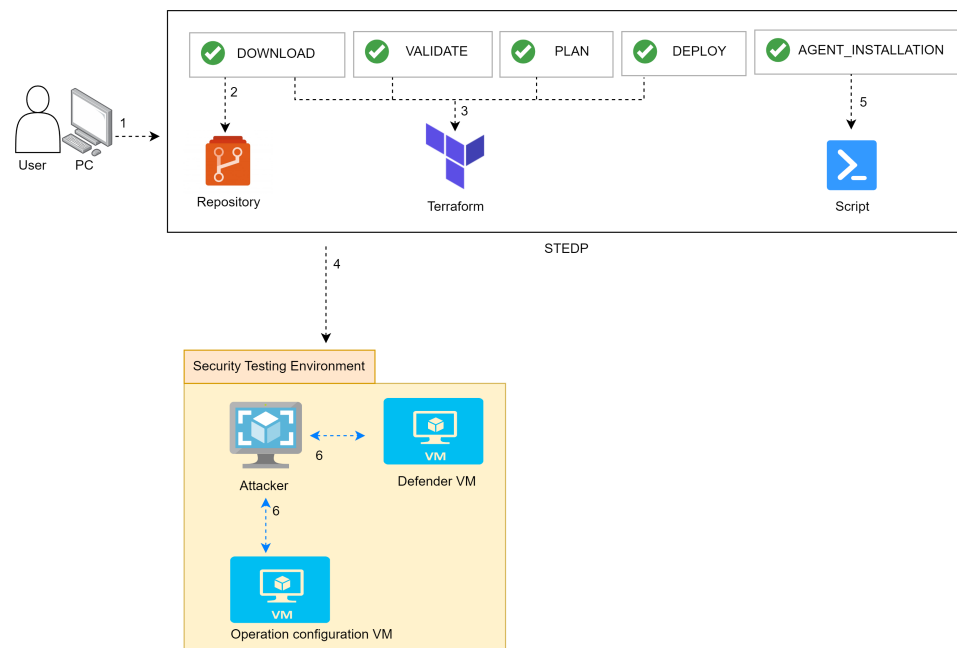


Figure 4.2: Security Testing Environment Deployment Pipeline diagram

## 4.2.2 Adversary Operations Pipeline

The AOP is relatively uncomplicated with only one stage, Adversary Operation Execution. The AOP is used to run the different security tests on the VMs inside the Security Testing Environment. It executes a PowerShell script that utilizes MITRE Caldera APIs to perform the security tests and collect the result data. Figure 4.3 shows how the AOP operates. It is worth noting that the Caldera Server mentioned in the descriptions is the Attacker in the figure, the descriptions of the actions can be seen in the list below:

1. **Executing the AOP**  
The user starts the AOP from inside the Azure environment of Sopra Steria.
2. **Adversary Operation Execution stage**  
AOP runs a PowerShell script on the Operation configuration VM inside the Security Testing Environment.
3. **Defining the security tests being run**  
The Operation configuration VM updates the Caldera Server with which adversary operation it should be running.
4. **Executing security tests**  
The Caldera Server executes the security tests on the Defender VM inside the Security Testing Environment.
5. **Generate a test report**  
The Caldera Server generates a report containing information about the security tests and saves it on the Operation configuration VM.

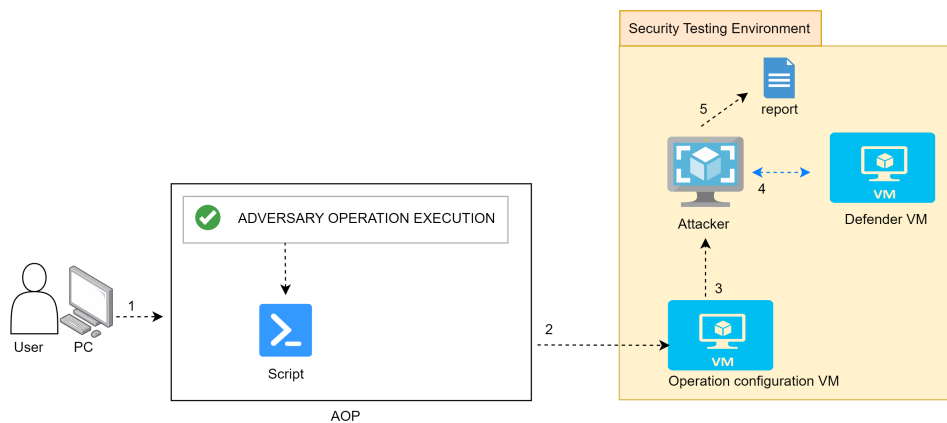


Figure 4.3: Adversary Operations Pipeline diagram

## Security test report

The AOP saves a report containing the security test results on the VM inside the Security Testing Environment. This report contains all the information from the different tests and is written in JSON format [54]. The PowerShell script in the AOP writes out the report two times, first a full report and then a filtered report. This was to check the success of filtering the results and to make the report easier to read. We designed the filtered report to contain most of the important results of the tests themselves. In the end, the report includes the name of the operation, when it started, when it finished, the adversary profile it ran, any skipped abilities, and the security test results. Figure 4.4<sup>1</sup> illustrates an example of how the structure of the data can look like in the result report.

```
{
  "adversary": {
    "adversary_id": "1a98b8e6-18ce-4617-8cc5-e65a1a9d490e",
    "atomic_ordering": [
      "6469befa-748a-4b9c-a96d-f191fde47d89",
      "90c2efaa-8205-480d-8bb6-61d90dbaf81b",
      "4e97e699-93d7-4040-b5a3-2e906a58199e",
      "300157e5-f4ad-4569-b533-9d1fa0e74d74",
      "ea713bc4-63f0-491c-9a6f-0b01d560b87e"
    ],
    "description": "An adversary to steal sensitive files",
    "has_repeatable_abilities": false,
    "name": "Thief",
    "objective": "495a9828-cab1-44dd-a0ca-66e58177d8cc",
    "plugin": "stockpile",
    "tags": []
  },
  "facts": [
    {
      "collected_by": [],
      "created": "2022-05-11T22:07:07Z",
      "limit_count": -1,
      "links": [
        "fa7ac865-004d-4296-9d68-fd425a481b5e"
      ]
    }
  ]
}
```

Figure 4.4: Example of structure in the result report

---

<sup>1</sup>Sample Operation Report, <https://caldera.readthedocs.io/en/latest/Operation-Results.html?#sample-operation-report>

### 4.3 Use Cases

Figure 4.5 is a visual representation of the use cases related to the execution of the build pipeline for deploying infrastructure and the test pipeline for running security tests. The diagram illustrates how a user can initiate both processes while having the flexibility to define specific tests to be run, thereby providing a comprehensive overview of the seamless interaction between the user, the deployment of infrastructure, and the execution of customized security tests. The user is the security tester who will be utilizing our solution.

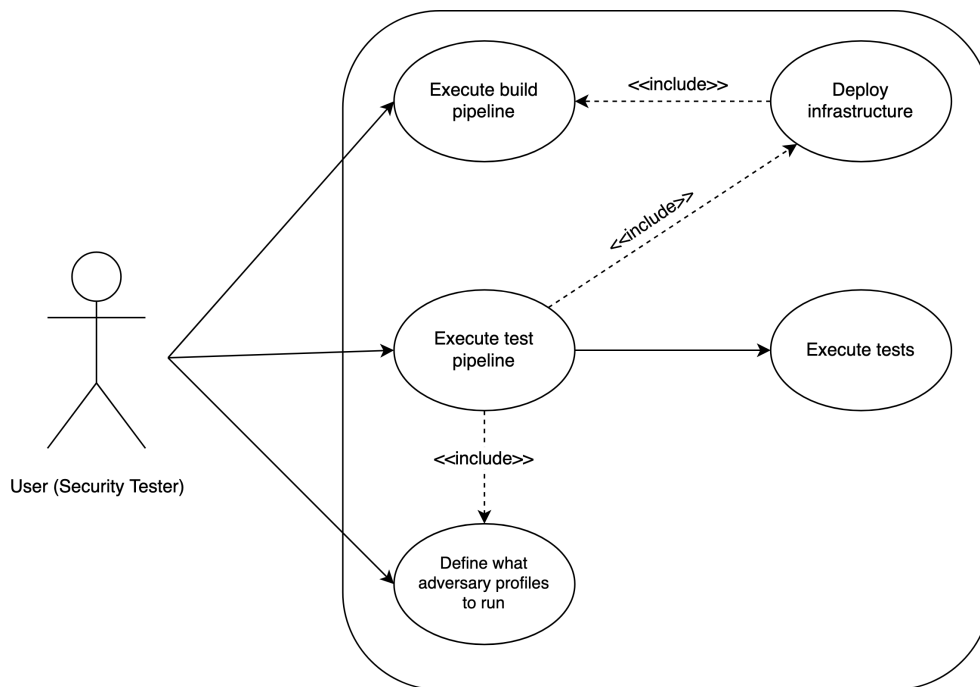


Figure 4.5: Use Cases

In Table 4.3, we outline Use Case 1, which focuses on the execution of the build pipeline. This highlights the steps and properties related to the pipeline for deploying the necessary infrastructure for security testing.

**Table 4.3:** Use Case 1

Use Case	Execute build pipeline
Goal in Context	Deploy infrastructure
Requirements	F1
Actors	Security Tester
Precondition	Actor has access to Azure DevOps
Trigger	User clicks the appropriate 'run pipeline' GUI button
Description	<ol style="list-style-type: none"> <li>1. User signs into Azure DevOps</li> <li>2. User clicks appropriate button</li> </ol>

Table 4.4 presents Use Case 2, which defines the specific tests to be run during the testing process. This use case demonstrates how users can customize the security tests or adversary profiles to be executed.

**Table 4.4:** Use Case 2

Use Case	Define what adversary profiles to run
Goal in Context	Manually set tests to be run
Requirements	F3
Actors	Security Tester
Precondition	Actor has access to Azure DevOps
Trigger	User defines what adversary profiles or tests are run in specified file
Description	<ol style="list-style-type: none"> <li>1. User signs into Azure DevOps</li> <li>2. User edits relevant files, adding or removing IDs for desired tests or adversary profiles</li> </ol>



In Table 4.5, we describe Use Case 3, which deals with the execution of the test pipeline. This use case showcases the procedure a user follows to initiate the defined set of tests on the deployed environment.

**Table 4.5:** Use Case 3

Use Case	Execute test pipeline
Goal in Context	Execute defined set of tests on deployed environment
Requirements	F2
Actors	Security Tester
Precondition	Actor has access to Azure DevOps, infrastructure has been deployed and tests to run have been defined
Trigger	User clicks the appropriate 'run pipeline' GUI button
Description	<ol style="list-style-type: none"><li>1. User signs into Azure DevOps</li><li>2. User clicks appropriate button</li></ol>

## Chapter 5

# Development Process

This chapter covers the development process for our project. Including the choices made for work methods, documentation, and routines. It is based on the initial planning documented in our Project Plan which is found in Appendix E, but has been subject to change throughout the process to ensure better adaptability.

### 5.1 Development Model

The choice of a development model for a software project like ours is crucial to meet our requirements and ensure an efficient workflow. When deciding which development model to use, we took several factors into consideration. The scope and topic of the project meant that we would have to spend a lot of time on research and learning to use new systems. Therefore we wanted to use a model that was quick and easy to use. Furthermore, it was important that it could easily be implemented to form a desired workflow suited for our team.

We mainly considered the Scrum, Waterfall, and Kanban development models. First, we considered the Scrum development model. This is a model that some of our team members are familiar with. Qualities of this model include working close together in teams, a process for gathering feedback, and setting goals with sprint planning. Scrum is a popular model which allows for flexibility and continuous improvement [6]. The second option was the waterfall model, which is sequential, meaning that when one phase ends the next one starts. The model does not allow you to go back to the previous phase after ending it [55]. This lowers the flexibility to a degree, and we decided that this would be too restrictive, due to the need for overlapping phases where we work on multiple phases at one time in our project. Lastly, the Kanban development model. All team members are familiar with and had used this model before. In this model, larger tasks are divided into smaller subtasks and placed on a Kanban board to provide a clear overview [2]. Tasks are allocated to distinct segments on the board, based on their status, the task owner,

and whether they require attention, are in progress, or have been completed. A significant advantage of using the Kanban model was its seamless integration into the Azure DevOps development platform [56]. In the end, the choice was between the Scrum and Kanban models, where we decided to go with the Kanban model due to its ease of implementation into our development workflow.

## 5.2 Phases and Timeline

The team started out our work by dividing the project into five distinct phases, to help organize our process. 1. Planning, 2. Familiarization and Research, 3. Project Implementation and First Draft, 4. Writing and Finalizing, and 5. Presentation. Figure 5.1 displays a simplified diagram of how many weeks each phase was estimated to take. These phases allowed us to effectively structure our work, ensuring a methodical and comprehensive approach to meeting the project goals of our project. We generally followed and stuck to these phases but made room for flexibility and overlapping when required. The following sections will provide an overview of each phase and its key focus elements.

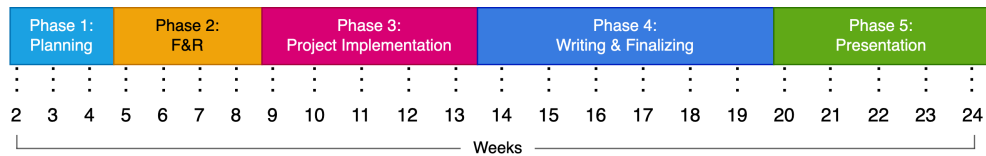


Figure 5.1: Project phases

### Phase 1: Planning

In the planning phase, our main goal was to establish a strong foundation for the project by developing a project plan and contract, ensuring that all stakeholders were aligned on objectives, roles, and expectations. This phase also involved completing relevant courses and familiarizing ourselves with key tools and platforms for our process. The finished Project Plan can be found in Appendix E.

### Phase 2: Familiarization and Research

During the familiarization and research phase, we aimed to gain a deep understanding of the tools, technologies, and methodologies relevant to our project. We familiarized ourselves with the Azure DevOps environment and conducted an initial proof-of-concept (PoC) security test. This phase also included research into automation strategies, security testing frameworks, and assessing the current threat environment relevant to our project. We also received some sample code from our clients that showed an example of how to deploy infrastructure resources to their Azure environment using Pipelines and Terraform, which helped us get an idea of how we might do the same for our solution.

**Phase 3: Project Implementation and First Draft**

In the project implementation phase, we focused on developing a working solution by integrating selected frameworks, automating deployment and execution of security tests, and thus creating a Minimum Viable Product (MVP). Azure DevOps and Pipelines are keywords here. We also ran a multitude of security tests using MITRE Caldera and Atomic Red Team, and worked on implementing the Metasploit framework into our product. Overall, this phase involved extensive testing, coding, and documentation of our progress and findings.

**Phase 4: Writing and Finalizing**

During the writing and finalizing phase, our primary objective was to write our thesis and thoroughly document our work, including the methodologies, tools, and frameworks used in the project. We focused on quality control to ensure that our writing and documentation were accurate, comprehensive, and well-organized, allowing for easy review and understanding for any stakeholders.

**Phase 5: Presentation**

The presentation phase is the final phase, where we will prepare a thorough presentation that summarizes the objectives, methods, findings, and outcomes of our project.

## 5.3 Documentation

Working on a project spanning over a longer period of time meant that it was important to choose the right tools for documentation. Since the project was quite technical, it was important for us to use tools that had solid version control and online accessibility. If an error occurred it would be useful to be able to revert to an older working version of the code to figure out what caused the error. From the start of the project, we knew that we would both work together in-person and online, which meant that having a stable online collaboration platform was necessary. In the sections below we discuss some of these tools and methods utilized throughout our process.

### Kanban boards

We used the digital Kanban board Azure Boards, which is provided as a feature of Azure DevOps to manage our project tasks and keep track of progress [56]. This allowed us to easily visualize our workflow, prioritize tasks, and quickly identify and resolve any bottlenecks.

Figure 5.2<sup>1</sup> illustrates an Azure DevOps kanban board where users can collaborate and manage tasks together.

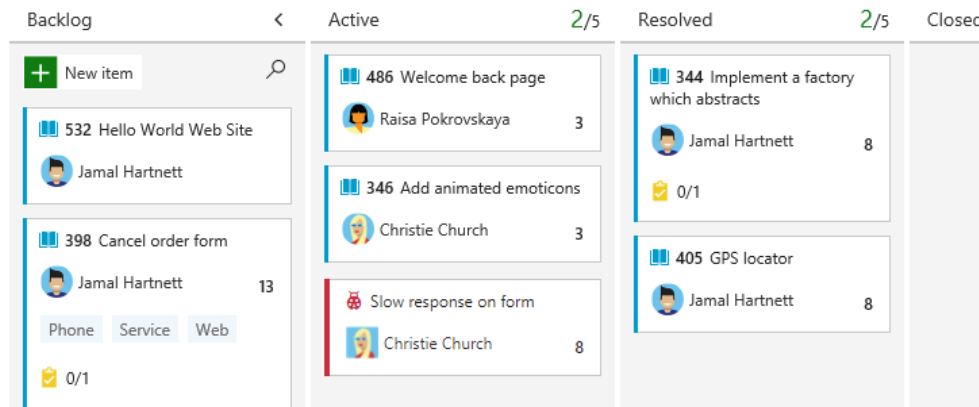


Figure 5.2: Azure DevOps Kanban

### Gantt chart

A Gantt chart was made in order to map out the project timeline and ensure that all tasks were completed on schedule. This chart followed our phases and estimated development process, including tasks and milestones for each phase. This helped us to stay organized and plan our workload effectively and also allowed us to

<sup>1</sup>Start using your Kanban board, <https://learn.microsoft.com/en-us/azure/devops/boards/boards/kanban-quickstart?view=azure-devops>

identify any potential scheduling conflicts in advance. The actual Gantt chart used can be found as part of our Project Plan in Appendix E.

### **Collaborative tools**

The use of collaborative tools like Google Docs, Azure Git Repositories, and Overleaf Latex assisted in facilitating communication and collaboration among team members. An internal Discord [57] channel was used for team-only communication and digital work sessions. Microsoft Teams [58] assisted in communication between the project group, owners, and supervisor. These tools allowed us to work more efficiently and effectively and ensured that everyone had access to the most up-to-date project information over the Internet.

## **5.4 Routines**

Establishing set routines enables us to attain well-defined milestones, accomplish goals effectively, and establish a structured approach for conducting meetings. It is crucial to foster a shared comprehension of these aspects among the team members, our supervisor, and the project owners to ensure seamless coordination throughout the project. The following sections provide an overview of the routines we implemented to guide our project's progress.

### **Milestones and Goals**

We set clear milestones and goals at the beginning of the project and regularly checked our progress against these markers. This helped us to stay focused and motivated and ensured that we were always working towards specific objectives. These were specifically tracked in our Gantt chart.

### **Physical and Digital meetings**

We used a combination of physical and digital meetings to communicate and work with team members, supervisors, and clients. This allowed us to maintain a strong sense of collaboration and ensure that everyone was on the same page, regardless of their location. Internal work meetings were held multiple times every week within the project group, to ensure continuous progress.

### **Supervisor meetings**

We scheduled regular weekly meetings with our supervisor to discuss project progress and receive feedback. This allowed us to stay on track and make any necessary adjustments to our approach and also helped us to build a stronger working relationship with our supervisor. The full informal notes from these meetings can be found in Appendix D.

### **Client meetings**

Bi-weekly meetings were held between the project group and our client to ensure that their needs and requirements were being met throughout the development process. This helped us to maintain a strong sense of alignment with the goals of our client and allowed us to make any necessary adjustments to our approach in real-time.

## Chapter 6

# Implementation

The objective of this project, as stated in Chapter 1.2, is to develop an automated system for conducting security tests. In this chapter, we will elaborate on the implementation of the two key aspects of this goal: **security testing** and **automation**, ensuring compliance with the requirements outlined in Chapter 4.1. Additionally, we will look into the details of the solution's components, discussing the different technologies, tools, and frameworks used.

### 6.1 Automation

The issue of automation in our project is a problem in two parts. The first part of this problem is to automate the process that deploys the testing environment, while the second problem is to automate the security testing process. In our solution, each of these problems is solved by integrating the tasks that make up each process into pipelines.

Before the deployment process could be integrated into a pipeline, the process itself had to be designed as robustly as possible. We achieved this by designing the process according to the concept of Infrastructure as Code (IaC). According to official Microsoft documentation [59], the core idea of IaC is to define Infrastructure using an 'IaC model'. An IaC model is a collection of human-readable files, preferably using a declarative definition syntax, to collectively define deployable infrastructure.



### 6.1.1 Defining Infrastructure as Code with Terraform

Our IaC model is designed using Terraform. In Figure 6.1 we illustrate the hierarchical relationship of the files that make up our IaC model. The model is centered around the 'main.tf' file, which then references data or imports module configurations from other files when necessary. Note however that there is no relation between the 'providers.tf' file and other files. This is because the 'providers.tf' file is not part of the IaC model. Instead, it declares which plugins are required for the model in order for it to interact with the APIs and services that are necessary to apply the model to the target environment. This file is read during the initialization process of Terraform, and the specified dependencies are then installed on the host that is applying the Terraform model.

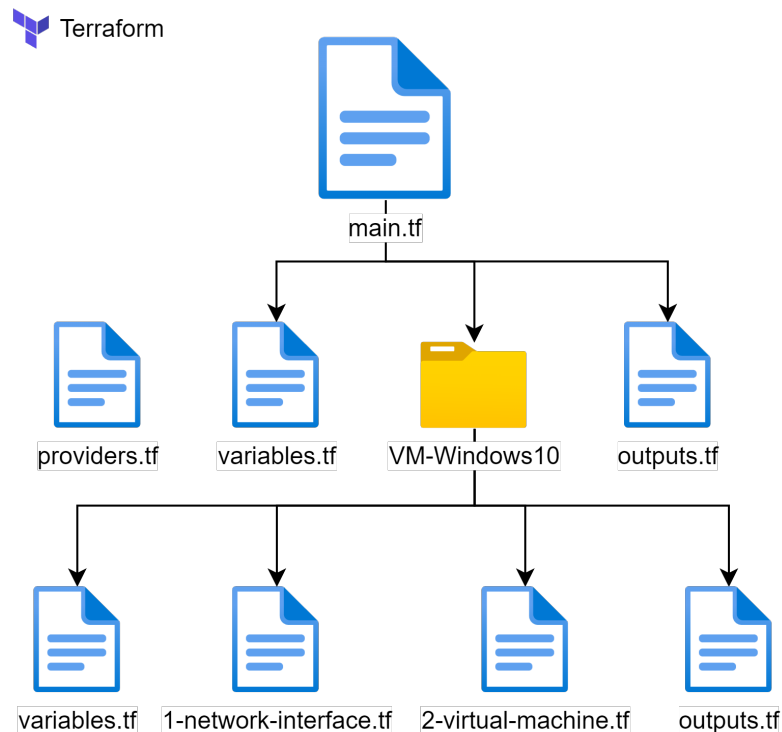


Figure 6.1: IaC Model

```
1 terraform {
2   required_version = ">=0.12"
3
4   required_providers {
5     azurerm = {
6       source = "hashicorp/azurerm"
7       version = "~>3.0"
8     }
9     docker = {
10      source = "kreuzwerker/docker"
11      version = "3.0.1"
12    }
13  }
14 }
15 provider "azurerm" {
16   features {}
17 }
```

**Listing 1:** A look at the providers.tf file

The providers file shown above in Listing 1 specifies the required plugins using the **required\_providers** block. The providers block indicates that the AzureRM, short for Azure Resource Manager plugin provided by Hashicorp, and the 'Docker' plugin provided by Kreuzwerker are required for the Terraform model to be applied correctly. The **providers** block is used to create a provider instance that contains the options and credentials required to manage resources in the target environment. This block can be used to override the configurations inherited from the host that applies the model, but this is not needed for our solution, so the configuration block is empty.

## Infrastructure as Code, model implementation

```

1 #####
2 ## Reference existing Resources
3 #####
4
5 # Resource Groups
6 data "azurerm_resource_group" "rg_main" {
7     name      = var.resource_group_name
8 }
9 data "azurerm_resource_group" "rg_network_spoke" {
10     name      = var.resource_group_spoke_name
11 }
12
13 # Networking
14 data "azurerm_virtual_network" "devops_spoke_vnet" {
15     name = var.network_name
16     resource_group_name = data.azurerm_resource_group.rg_network_spoke.name
17 }
18 data "azurerm_subnet" "spoke-snet-managed-env" {
19     virtual_network_name = data.azurerm_virtual_network.devops_spoke_vnet.name
20     name = var.subnet_managed_env
21     resource_group_name = data.azurerm_resource_group.rg_network_spoke.name
22 }
23 data "azurerm_subnet" "spoke-snet-container-services" {
24     virtual_network_name = data.azurerm_virtual_network.devops_spoke_vnet.name
25     name = var.subnet_container_services
26     resource_group_name = data.azurerm_resource_group.rg_network_spoke.name
27 }
28 data "azurerm_network_security_group" "nsg" {
29     name = var.network_security_group_name
30     resource_group_name = data.azurerm_resource_group.rg_network_spoke.name
31 }
32
33 # Application security group
34 data "azurerm_application_security_group" "asg" {
35     name = var.asg_name
36     resource_group_name = data.azurerm_resource_group.rg_main.name
37 }
38
39 # Container image registry
40 data "azurerm_container_registry" "reg" {
41     name = var.registry_name
42     resource_group_name = data.azurerm_resource_group.rg_main.name
43 }

```

**Listing 2:** The first part of the main.tf file is dedicated to referencing existing resources

Listing 2 includes a section of code from the 'main.tf' file, which is the centerpiece of our IaC model. Non-Functional Requirement 6 listed in Table 4.2 stipulates that our solution must deploy any necessary infrastructure to an already existing cloud environment owned by Sopra Steria. This environment already has a basic architecture containing Resource Groups, a Virtual Network, and various security groups and configurations.

Since we are deploying to an already existing environment, the first section of the **main.tf** file is dedicated to referencing existing resources and declaring them as data sources. Data associated with these resources can be used later in the model by referencing the data sources directly. A data source is declared using the syntax:

```
data "<resource_type>" "<data_source_name>" {
  <attribute_name> = <attribute_value>
}
```

Where **<resource\_type>** is the type of infrastructure resource that the data source represents, **<data\_source\_name>** is a unique identifier used to reference the data source later. Finally **<attribute\_name>** and **<attribute\_value>** specify the name and value for the attribute that is used to identify the target resource. There can be several attributes used to identify a target resource, in which case Terraform will search for the resource that matches all of the specified values.

Lines 13-17 in Listing 2, is an example of a data source declaration of a **azurerm\_virtual\_network** resource. This data source represents an existing Azure virtual network in the cloud environment our solution deploys to. The specific virtual network we are referencing is identified by finding the target resource that has matching values for the two specified attributes 'name' and **resource\_group\_name**. The name attribute must be equal to the variable **network\_name** from the **variable.tf** file. The second desired value for the attribute **resource\_group\_name** is set equal to the 'name' attribute of another data source, namely the Azure Resource Group **rg\_network\_spoke**.

```
1 # Create container group
2 resource "azurerm_container_group" "terraform_caldera_group" {
3     name                = "${var.prefix}-Caldera"
4     location            = var.location
5     resource_group_name = data.azure_terraform_resource_group.rg_main.name
6     ip_address_type     = "Private"
7     subnet_ids         = [data.azure_terraform_subnet.spoke-snet-container-services.id]
8     os_type             = "Linux"
9
10    image_registry_credential{
11        username = var.registry_username
12        password = var.registry_secret
13        server   = data.azure_terraform_container_registry.reg.login_server
14    }
15
16    container{
17        name = "caldera-container"
18        image = "${data.azure_terraform_container_registry.reg.login_server}/caldera"
19        cpu   = "1.0"
20        memory = "1.5"
21        ports {
22            port      = 80
23            protocol = "TCP"
24        }
25        ports {
26            port      = 8888
27            protocol = "TCP"
28        }
29    }
30
31    depends_on = [
32        data.azure_terraform_container_registry.reg
33    ]
34 }
```

**Listing 3:** main.tf part 2: specifies configurations for the Caldera container

Listing 3 shows the code that defines an Azure Container Group. The basic configurations for the group are set in the initial resource block (lines 3-8). Most of these configurations are either passed to Terraform as environment variables, or the required data is fetched from a previously declared data source. Also note the **image\_registry\_credential** block, which defines the credentials required to authenticate with an Azure Container Registry. This is the location in our project's Azure Environment that stores our container image. If connection or authentication to this registry fails, the creation process will abort. The 'container' block defines the single container that operates in this Container Group. This is the 'Attacker' shown in various figures from Chapter 4. The container runs MITRE Caldera as a service accessible on a locally hosted web server open to web traffic on port 8888. The image for this container is fetched from the aforementioned Azure Container Registry.

```

1  # Create VM
2  module "VM-Windows10" {
3      source                      = "./modules/VM-Windows10"
4
5      resource_group_name        = data.azure_terraform_resource_group.rg_main.name
6      location                    = var.location
7      vm_image                    = {
8          publisher = "MicrosoftWindowsDesktop"
9          offer     = "windows-10"
10         sku       = var.m_vmw10_vm_image_sku
11         version  = "latest"
12     }
13     vm_size                = var.m_vmw10_vm_size
14     prefix                  = var.prefix
15     subnet_id              = data.azure_terraform_subnet.spoke-snet-managed-env.id
16     nsg_id                  = data.azure_terraform_network_security_group.nsg.id
17     asg_id                  = data.azure_terraform_application_security_group.asg.id
18     admin_username         = var.m_vmw10_admin_username
19     admin_password         = var.m_vmw10_admin_password
20 }

```

**Listing 4:** main.tf part 3: A module is included

The final part of the 'main.tf' file can be seen above in Listing 4. This file defines a VM by importing resources from a Terraform module named 'VM-Windows10'. A Terraform module is a self-contained package of configuration files along with any dependent resources [60]. In this case, the VM-Windows 10 module contains definitions for a VM, and an associated Network Interface, as well as default configurations for the two. The code within this module defines resources to be created in a similar manner to how the container is defined in Listing 3. The exact code can be viewed in Appendix F in Listings 19-22. Implementation-specific data, such as the login credentials for the Admin account on the VM, are passed to this module using environment variables.

### Environment data, Ingress- and Egress points

In our solution, a large portion of configuration values are specified using variables. This can be seen in the provided code examples wherever an attribute value is set to `var.<variable_name>`. These variables are declared in the `variables.tf` files, according to the syntax shown below:

```

variable "<variable_name>" {
    description = "<description>"
    sensitive   = <true/false>
    default     = <variable_value>
}

```

The 'default' attribute is the content of the variable, while 'sensitive' determines whether the variable can be output by Terraform. The 'sensitive' option allows

us to secure passwords or other sensitive information from being outputted as plain text. Note that the 'default' attributes are not set for any of our variables. These values are instead passed to Terraform from Azure Pipelines as environment variables, Figure 6.2 illustrates how this works in general terms, while the exact details will be explained in further detail in Section 6.1.2.

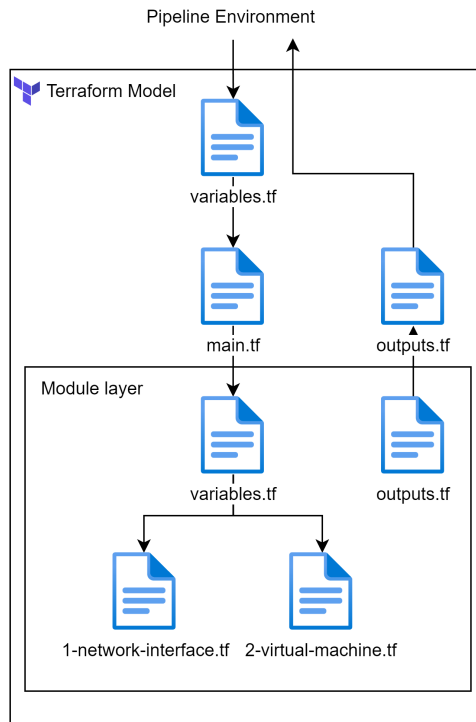


Figure 6.2: Data Flow from Terraform Perspective

As illustrated in the figure above, the 'variable.tf' and 'output.tf' files are also used to send and receive data between layers internal to the Terraform model. The VM-Windows10 module has a file of its own where variables are declared. Data for these variables are passed from 'main.tf' when the module is first included in the Terraform model. This is shown in Listing 4, where the module variables are set in lines 5-19.

Also, note the **outputs.tf** files. These files are responsible for passing data from Terraform to the runtime environment. In our solution, the STEDP needs to know the IP address of the Caldera container after it has been deployed. This IP address is allocated dynamically and can change between deployments. Therefore, Terraform outputs this data after the creation process is complete, as shown below.

```

1  output "container_instance_ip" {
2    value = azurerm_container_group.terraform_caldera_group.ip_address
3  }

```

## 6.1.2 Integrating IaC model into an Azure Pipeline

The Terraform model that has been discussed so far can be thought of as the 'blueprint' for the environment to be deployed. According to our requirements as mentioned in Chapter 4.1, this blueprint must now be integrated into Azure Pipelines. Terraform alone is not fully 'automatic'. The user has to manage authentication to the target environment, and several commands must be run manually through a CLI to apply the model. Finally, Terraform cannot make all the necessary post-deployment configurations that our solution requires. By integrating our Terraform model into a pipeline, all of these tasks are handled as a single process.

### STEDP Implementation, Preamble and data flow

The Security Testing Environment Deployment Pipeline (STEDP) is the centerpiece of our solution to automated infrastructure deployment. The pipeline works by encapsulating all tasks that make up the deployment process as individual stages within the pipeline. The pipeline allows for simple logic such as establishing dependencies between stages. For example, if one stage is a producer of data that is then consumed in a later stage, pipelines allow this dependent relationship to be specified. If the first stage fails, any dependent stages will be aborted. This way we can ensure the integrity of the deployment process.

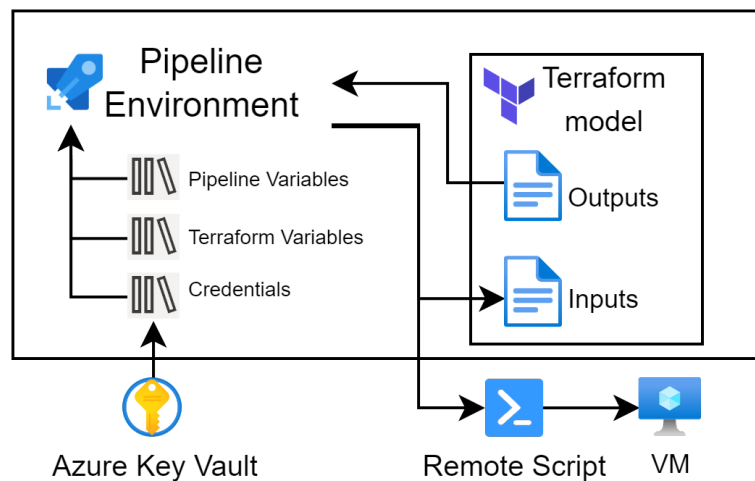
```
1  trigger: none
2
3  name: Infrastructure-Deployment
4  resources:
5    repositories:
6      - repository: Bachelor
7        type: git
8        name: Bachelor Project
9        ref: main
10 pool:
11   vmImage: ubuntu-latest
12
13 variables:
14   - group: 'Pipeline Environment Variables'
15   - group: 'Terraform Environment Variables'
16   - group: 'Pipeline Credentials'
17   - name: environment
18     value: 'Bachelor'
19   - name: LocalRepo
20     value: '$(Build.Repository.LocalPath)'
```

Listing 5: Preamble for the STEDP file

The first part of Listing 5 is a preamble that specifies various configurations, and defines and imports variables. The **resources** block (lines 5-9) defines our project repository as an available resource for the build agent which is configured



to run on a VM with the 'ubuntu-latest' image. The **variables** block (lines 13-20) defines variables and imports variable groups from our repository's pipeline library. Variable groups are imported when the line corresponding to a given variable group is read during run-time. The two primary variable groups are the **Pipeline Environment Variables** and the **Terraform Environment Variables** groups. It is not strictly necessary for these two groups to be separate, however, for the sake of good order, the variables have been separated into these two groups based on where the data they represent is consumed. The variables contained in these variable groups are generally defined in plain text. They can be edited in the Azure DevOps GUI to change certain configurations of the infrastructure to be deployed, tasks to be executed, or to match changes made to the target environment. The final group that can be seen above, the **Credentials** variable group, is a notable exception to this. This group imports its values as encrypted secrets from an Azure Key Vault located in our Azure Cloud Environment.



**Figure 6.3:** Data Flow from Pipeline Perspective

The data flow shown in Figure 6.2 can be expanded to also illustrate how data flows from the perspective of the pipeline environment at large, which results in Figure 6.3. As seen in the figure the pipeline also passes data on to a **Remote Script**. This script is stored on an Azure Storage Account located within our project environment.

### STEDP implementation, stages

The general purpose of each stage in the STEDP is discussed briefly in Chapter 4.2.1. In this section, the actual code behind these stages will be shown and explained.

```

1 - stage: 'DOWNLOAD'
2   jobs:
3     - deployment: 'DOWNLOAD_FILES'
4       displayName: 'DOWNLOAD FILES'
5       environment: $(environment)
6       strategy:
7         runOnce:
8           deploy:
9             steps:
10            - checkout: Bachelor
11              - task: PublishBuildArtifacts@1
12                inputs:
13                  PathtoPublish: '$(LocalRepo)/Terraform'
14                  ArtifactName: 'drop'
15                  publishLocation: 'Container'

```

Listing 6: Stage 1: DOWNLOAD

The code for the **DOWNLOAD** stage can be seen above in Listing 6. A stage divide further into **jobs**, which again divide into **steps**. This stage contains only one 'deployment' type job named **DOWNLOAD\_FILES** that executes with a 'runOnce' strategy. The first step **checkout: Bachelor** is a built-in task in Azure Pipelines that moves the process into our Repository. The second step is a 'task' named **Publish-BuildArtifacts@1**. This task publishes the Terraform folder from our repository as an artifact, allowing it to be shared in-between stages. 'Tasks' are named functions provided by Azure Pipelines to perform a wide array of operations. Official documentation for every task is accessible on Microsoft's Websites [61]. Also, note the number behind the '@'. This number represents the version of the task being used, which ensures that newly released versions of a task that could potentially be incompatible are not automatically used.

Listing 7 shows the **VALIDATE** stage contains one job and four steps. The first three steps, **TerraformInstaller@0** (line 12-14), **CopyFiles@2** (line 15-19), and **TerraformTaskV2@2** (line 19-29), are necessary for ensuring that the Terraform software, the model configuration files, and any dependencies are present on the build agent and correctly configured. As will be seen later, these steps are present at the beginning of every stage that uses Terraform. The **TerraformInstaller@0** task, checks and installs the latest version of Terraform if needed. **CopyFiles@2** fetches a fresh copy of the Terraform build artifact created in the 'DOWNLOAD' stage and places it in the current stage's working directory. Lastly, the **TerraformTaskV2@2** allows the pipeline to execute Terraform CLI commands using the 'command' input to determine which command to run. In this case, 'init' is provided as the input, meaning that the command 'Terraform init' will be executed. This command sets up the necessary back-end configuration explained at the beginning of Section 6.1.1.

```

1  - stage: 'VALIDATE'
2    dependsOn:
3      - 'DOWNLOAD'
4    jobs:
5      - deployment: 'Terraform_VALIDATE'
6        displayName: 'Terraform VALIDATE'
7        environment: $(environment)
8        strategy:
9          runOnce:
10         deploy:
11           steps:
12             - task: TerraformInstaller@0
13               inputs:
14                 terraformVersion: '0.14.10'
15             - task: CopyFiles@2
16               inputs:
17                 SourceFolder: '$(Agent.BuildDirectory)/drop'
18                 Contents: '**'
19                 TargetFolder: '$(System.DefaultWorkingDirectory)/terraform/'
20             - task: TerraformTaskV2@2
21               inputs:
22                 provider: 'azurerms'
23                 command: 'init'
24                 backendServiceArm: $(backendServiceArm)
25                 backendAzureRmResourceGroupName: $(ResourceGroupName)
26                 backendAzureRmStorageAccount: $(backendAzureRmStorageAccount)
27                 backendAzureRmContainerName: $(backendAzureRmContainerName)
28                 backendAzureRmKey: $(backendAzureRmKey)
29                 workingDirectory: '$(System.DefaultWorkingDirectory)/terraform/'
30             - task: TerraformTaskV2@2
31               inputs:
32                 provider: 'azurerms'
33                 command: 'validate'

```

Listing 7: Stage 2: VALIDATE

The three tasks discussed in the previous paragraph prepare the run-time environment for the real purpose of this stage, which is to validate the Terraform model. This is done using another **TerraformTaskV2** task, this time with **validate** as the command input (lines 30-33). This command ensures that configurations are syntactically correct and that all required variables are properly defined. If there are any issues, the validation process will output the error message to the pipeline, which will then abort. In lines 22-29, we also see several examples of how environment variables that have been imported from the aforementioned variable groups are referenced. The syntax of such a variable is simply `'$(<variable_name>')`. These statements are then expanded into the value of the corresponding variable when read during run-time.

```

1  - stage: 'PLAN'
2  dependsOn:
3  - 'DOWNLOAD'
4  - 'VALIDATE'
5  jobs:
6  - deployment: 'Terraform_PLAN'
7    displayName: 'Terraform PLAN'
8    environment: $(environment)
9    strategy:
10   runOnce:
11     deploy:
12       steps:
13         # - task: TerraformInstaller@0
14         # - task: CopyFiles@2
15         # - task: TerraformTaskV2@2 (init)
16         - task: TerraformTaskV2@2
17           inputs:
18             provider: 'azurerms'
19             command: 'plan'
20             commandOptions: >-
21               -var="resource_group_name=$(ResourceGroupName)"
22               -var="resource_group_spoke_name=$(ResourceGroupSpokeName)"
23               -var="location=$(infrastructureLocation)"
24               -var="network_name=$(virtualNetworkName)"
25               -var="subnet_managed_env=$(SubNetworkNameVM)"
26               -var="subnet_container_services=$(SubNetworkNameContainer)"
27               -var="network_security_group_name=$(nsgName)"
28               -var="asg_name=$(asgName)"
29               -var="prefix=$(name_prefix)"
30               -var="registry_name=$(registryName)"
31               -var="registry_username=$(registryName)"
32               -var="registry_secret=$(caldera-secret)"
33               -var="m_vmw10_vm_image_sku=$(module_vm_image_sku)"
34               -var="m_vmw10_vm_size=$(module_vm_size)"
35               -var="m_vmw10_admin_username=$(module_admin_username)"
36               -var="m_vmw10_admin_password=$(module-admin-secret)"
37               -lock=false
38             environmentServiceNameAzureRM: $(backendServiceArm)
39             workingDirectory: '$(System.DefaultWorkingDirectory)/terraform/'

```

Listing 8: Stage 3: Plan

Listing 8 shows code for the **PLAN** stage of the STEDP. First note the three steps for the tasks **TerraformInstaller@0**, **CopyFiles@2**, and **TerraformTaskV2@2** using command-input 'init' in Lines 13-15. In the code in our repository, these tasks are present and implemented identically to the ones seen in Listing 7, where they also serve the same purpose. In Listing 8 these tasks are represented as comments using only their names, only to save space in this listing. These tasks must be executed in this stage as well because configurations made to the working directory by these processes do not carry over between stages. The purpose of the **PLAN** stage is to generate a plan for what changes must be made to the target environ-

ment in order for it to match the configurations specified in the Terraform model. This is done using the **TerraformTaskV2@2** task with 'plan' as the command input (lines 16-39). The desired changes are output to the pipeline, which can help with auditing in the event of an error. This stage also provides another layer of validation by checking the target environment for conflicts such as missing or duplicate resources. Also note the **commandOptions**, where values for environment variables are passed to Terraform as illustrated in Figure 6.2 and Figure 6.3.

The primary functions of the **DEPLOY** stage seen in Listing 9 are defined similarly to what is done in the **PLAN** stage. As can be seen above the **DEPLOY** stage is identical until line 25, the only exception being that the command input to **TerraformTaskV2@2** is 'apply' instead of 'plan'. The difference is that the 'apply' command implements required changes instead of just outputting a readable plan. From line 25 onwards, there are several tasks dedicated to 'catching' data that have been outputted by Terraform (lines 25-31). This is a part of the relationship between sending and receiving data illustrated in Figure 6.2 and 6.3.

```

1 - stage: 'DEPLOY'
2   dependsOn:
3     - 'DOWNLOAD'
4     - 'VALIDATE'
5     - 'PLAN'
6   jobs:
7     - deployment: 'Terraform_DEPLOY'
8       displayName: 'Terraform DEPLOY'
9       environment: $(environment)
10      strategy:
11        runOnce:
12          deploy:
13            steps:
14              # - task: TerraformInstaller@0
15              # - task: CopyFiles@2
16              # - task: TerraformTaskV2@2 (init)
17              - task: TerraformTaskV2@2
18                inputs:
19                  provider: 'azurerm'
20                  command: 'apply'
21                  commandOptions: >-
22                    # same variables as 'PLAN' stage
23                  environmentServiceNameAzureRM: $(backendServiceArm)
24                  workingDirectory: '$(System.DefaultWorkingDirectory)/terraform/'
25              - powershell: |
26                $terraformOutput = Get-Content `
27                  "$(terraformApply.jsonOutputVariablesPath)" | ConvertFrom-Json
28                Write-Host "##vso[task.setvariable variable=`
29                  container_ip_address;isoutput=true]" `
30                  $terraformOutput.container_instance_ip.value
31                name: setOutputVariables
32      - job: 'PassOutputToStage'
33        dependsOn: 'Terraform_DEPLOY'
34        variables:
35          - name: out_container_ip_address
36            value: $[dependencies.Terraform_DEPLOY.outputs`
37              ['Terraform_DEPLOY.setOutputVariables.container_ip_address']]
38        steps:
39          - powershell: |
40            Write-Host "##vso[task.setvariable variable=`
41              container_ip_address;isoutput=true] $(out_container_ip_address)"
42            name: terraformOutput

```

Listing 9: Stage 4: Deploy

Finally, **AGENT INSTALLATION** shown in Listing 10 is the last stage of the STEDP. As mentioned this stage receives data from the **DEPLOY** stage, which is done in lines 8-10. From here, the stage checks out our project repository before executing the script named **Agent-Configuration.ps1** that contains a single PowerShell command, *Set-AzVMExtension*. The *Set-AzVMExtension* command allows us to make configurations directly onto a VM. In this case by executing the **CalderaAgentSetup.ps1** script, which is stored as a remote resource in our Azure environment. Both of these scripts can be viewed in Appendix F. The **CalderaAgentSetup.ps1** script installs a Caldera agent which runs as a process on the VM and handles the execution of security tests. This is explained in further detail in Section 6.2.

```

1  - stage: 'AGENT_INSTALLATION'
2    dependsOn: 'DEPLOY'
3    jobs:
4      - deployment: 'Azure_AGENT_INSTALLATION'
5        displayName: 'Azure AGENT INSTALLATION'
6        environment: $(environment)
7        variables:
8          - name: tf_out_container_ip_address
9            value: $[stageDependencies.DEPLOY.PassOutputToStage.outputs`
10              ['terraformOutput.container_ip_address']]
11          - name: tf_out_vm_name
12            value: "$(name_prefix)-w10"
13        strategy:
14          runOnce:
15            deploy:
16              steps:
17                - checkout: Bachelor
18                  task: AzurePowerShell@5
19              inputs:
20                azureSubscription: '$(backendServiceArm)'
21                ScriptType: 'FilePath'
22                ScriptPath: '$(LocalRepo)/Scripts/Agent-Configuration.ps1'
23                azurePowerShellVersion: 'LatestVersion'
24                pwsh: true
25            env:
26              DEPLOY_RG: $(ResourceGroupName)
27              CONTAINERIP: $(tf_out_container_ip_address)
28              VM_NAME: $(tf_out_vm_name)

```

**Listing 10:** Stage 5: Agent Installation

## AOP Implementation

The AOP is as described in Chapter 4.2.2, responsible for configuring and executing security tests using the infrastructure resources deployed by the STEDP. Here we will briefly describe how the AOP integrates the process of security testing. The in-depth details surrounding the implementation of the security testing process itself, as well as the environment they are performed in will be explained in Section 6.2.

The AOP starts with a preamble mostly identical to the STEDP which was shown in Listing 5. The only difference in the preamble is that the AOP imports an additional variable group named **Adversary Operation Variables**, which contains a variable determining what sort of security tests are executed. Line 21 of Listing 11 shows this variable being referenced and passed on to the Security Testing Script, which is discussed in Section 6.2.1. The AOP as seen below only has a single stage. This is because the sole purpose of this pipeline is to initiate and pass data to a remote script executed on the VM that is the target for security tests. This is done using the same *Set-AzVMExtension* method as for the **AGENT INSTALLATION** stage of the STEDP shown in Listing 10.

```

1  #
2  #  Preamble
3  #
4  - stage: 'ADVERSARY_OPERATION'
5    jobs:
6      - deployment: 'ADVERSARY_OPERATION_EXEC'
7        displayName: 'ADVERSARY OPERATION EXECUTION'
8        environment: $(environment)
9        strategy:
10         runOnce:
11           deploy:
12             steps:
13               - checkout: Bachelor
14                 task: AzurePowerShell@5
15                 inputs:
16                   azureSubscription: '$(backendServiceArm)'
17                   scriptType: 'FilePath'
18                   scriptPath: '$(LocalRepo)/Scripts/Execute-AdversaryOperation.ps1'
19                   azurePowerShellVersion: 'LatestVersion'
20                   pwsh: true
21             env:
22               DEPLOY_RG: $(ResourceGroupName)
23               VM_NAME: "$(name_prefix)-w10"
24               ADVERSARY_PROFILE: $(AdversaryProfileID)

```

Listing 11: AOP implementation



## 6.2 Security Testing

A significant aspect of our project revolves around automating security tests to the greatest extent possible. This was solved by implementing MITRE Caldera and using the AOP. One difference between the system design and our implementation is the number of VMs inside the Security Testing Environment. We decided to combine the Operation configuration VM and the Defender VM mentioned in Chapter 4.2.2. Implementing one VM made testing simpler and made the deployment of resources more cost-efficient.

Figure 6.4 illustrates the implemented Security Testing Environment. The VM will be both the defender and the Caldera agent, while the Attacker is still the Caldera server. In the following subsection, we will go over our implementation involving the security testing script used in the AOP.

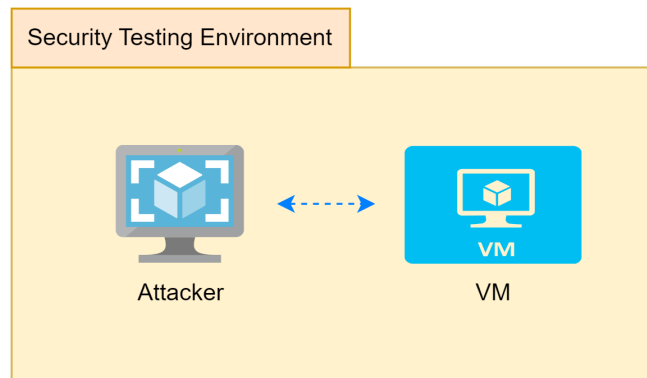


Figure 6.4: Illustration of the implemented Security Testing Environment

### 6.2.1 Security Testing Script

The security testing script is a PowerShell script that has been designed to automate the process of security testing. The script utilizes two APIs: one API to initiate and commence a security operation, and another API to retrieve a report detailing the results of the operation. Once the operation has been completed and the report has been generated, the script applies filters to the report and saves the filtered report onto the desktop of the VM.

Figure 6.5 provides a graphical representation of the security script. The sequence diagram demonstrates how the script utilizes the two APIs: first to initiate and run a security operation, and second to retrieve a report detailing the operation results. The "makeOperation" and "getReport" lines going from the VM to the Caldera Server represent the APIs. The script continues to call the "getReport" API until the security operation has finished. Finally, once the operation is complete, the script saves the report onto the VM.

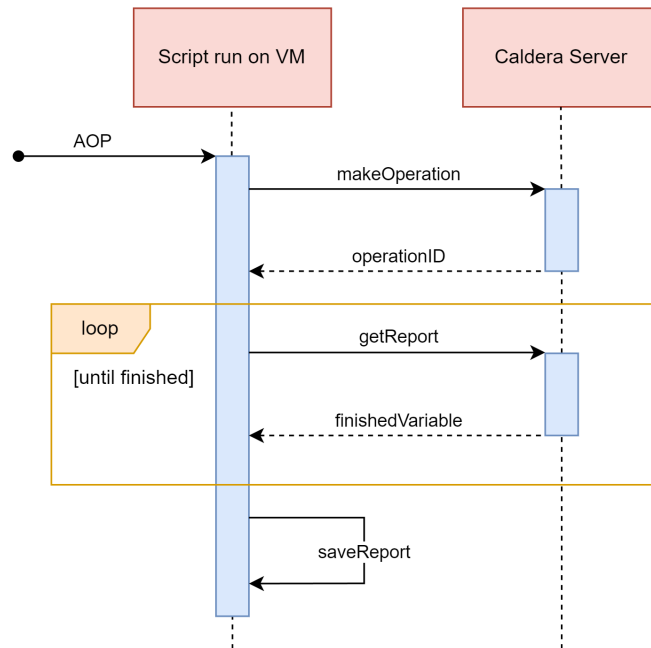


Figure 6.5: Sequence diagram of the security testing script

Listing 12 shows the first API call of the security testing script. The first step is to construct the request body for the API. This is accomplished by creating a variable that holds a hash table, which is subsequently converted to a JSON-formatted string. The hash table contains four keys: "index", "name", "adversary\_id" and "auto\_close". The MITRE Caldera `/api/rest` is the API endpoint used and is a REST API that allows for interaction with the core components of Caldera. Since this API can interact with every part of Caldera, the "index" key is used to direct requests to their corresponding object types [62]. We specify that "index" = "operations" to initiate a new operation, which we name "testoperation". The "adversary\_id" specifies which adversary profile we intend to execute, and it is a variable containing identification towards one of Caldera's pre-made adversary profiles. Lastly, the "auto\_close" is an operation value that makes the operation close automatically when it has no further tasks to perform.

```
1 # Create request body
2 $body = @{
3     index = "operations"
4     name = "testoperation"
5     adversary_id = $adversaryid
6     auto_close = 1
7 } | ConvertTo-Json
8
9 # Make PUT request
10 $Response = Invoke-RestMethod `
11     -Method Put `
12     -Uri "http://$containerip:8888/api/rest" `
13     -Headers @{"accept"="application/json"; "key"="ADMIN123"} `
14     -Body $body
```

**Listing 12:** API call to initiate and run an operation

After constructing the request body, the script proceeds with the API request using PowerShell's `Invoke-RestMethod` cmdlet. This is a cmdlet designed to send HTTP requests to REST APIs [63]. In this particular instance, the cmdlet is used with four parameters: "Method", "Uri", "Headers", and "Body". By specifying the "Method" parameter as PUT, we inform the Caldera Server that a new operation is being initiated. The "Uri" parameter represents the URL, which consists of the IP address to the Caldera Server and the specific API that is being used. Within the "Uri", we have a `$containerip` variable, which contains the IP address of the Caldera Server. This variable is defined at the start of the script by reading the IP from a file located on the VM, where the IP is output during the infrastructure creation process. The "headers" parameter takes a hash table that contains information regarding the accepted objects and the key header. Caldera requires a key header to be passed along with the request whenever you send requests to any IP address [62]. The default value of this key for Caldera is "ADMIN123" [64]. Lastly, we have the "Body" parameter, which simply takes the `$body` variable mentioned previously and sends it along with the request.

Listing 13 shows the second API used in the security testing script. The first step is to retrieve the operation ID. This is accomplished by assigning the value of the 'id' property from the `$Response` variable to a new variable named `$id`. Next, a "headers" body is created as a hash table containing three keys: "accept", "Content-Type", and "key". The "accept" and "Content-Type" are both set to "application/json", which makes it possible to accept and send JSON-formatted data. Lastly, the "key" is the default key header for MITRE Caldera. This "headers" body is then assigned to a variable named `$headers`. The script then makes use of the `Invoke-RestMethod` cmdlet to send a POST request to the REST API. The "Method" parameter is set to POST. The "Uri" parameter is combined of the `$containerip` variable, which holds the IP address of the Caldera server, and with the API `/api/v2/operations/$id/report`. The "Headers" parameter is set to the previously created `$head-`

ers variable. Lastly, the "Body" parameter is set to "enable\_agent\_output" with a value of true. This informs the Caldera Server to include agent output in the operation report. Lastly, we assigned the entire Invoke-RestMethod to a \$result variable.

```
1 # Get operation ID
2 $id = $Response.id
3
4 # Create headers body
5 $headers = @{
6     "accept" = "application/json"
7     "Content-Type" = "application/json"
8     "key" = "ADMIN123"
9 }
10
11 # Make initial POST request
12 $result = Invoke-RestMethod `
13     -Method Post `
14     -Uri "http://$containerip:8888/api/v2/operations/$id/report" `
15     -Headers $headers `
16     -Body '{
17         "enable_agent_output": true
18     }'
19 $result
```

**Listing 13:** API call to get the report when the operation is finished

A small problem with the report API, is that it can be called even if the operation is not yet finished, which can lead to a report with no information. In order to ensure that the operation report retrieved contains complete information, a loop was implemented in the script. The loop continually checks for the "finish" variable inside the report API until it is not null, indicating that the operation is complete. The loop is shown in Listing 14, and it utilizes the do-while structure. Within the loop, the Invoke-RestMethod cmdlet is called together with the `/api/v2/operations/$id/report`, using the same parameters as previously. Following the API call, the script is put to sleep for 60 seconds using the Start-Sleep cmdlet. This prevents the script from making unnecessary requests for the operation report while the operation is still in progress.

```

1  # Loop until $result.finish is not null
2  do {
3      $result = Invoke-RestMethod `
4          -Method Post `
5          -Uri "http://$containerip`:8888/api/v2/operations/$id/report" `
6          -Headers $headers `
7          -Body '{
8              "enable_agent_output": true
9          }'
10     Start-Sleep -Seconds 60
11 } while ($result.finish -eq $null)

```

**Listing 14:** Do-while-loop to make sure operation is finished

After ensuring we got a complete report containing all the security test information, we had to write out the report in a file. First, the "paw" for the Caldera agent must be retrieved, which is a unique identifier for a Caldera agent [65]. This identifier is stored in the variable \$paw. The \$paw variable is then used to access the different security tests run on the Caldera agent, and the relevant information is filtered and stored in the \$specific\_steps variable. Listing 15 shows the code for filtering the report. The information from the entire operation is also filtered and saved into the \$specific\_result\_json variable. Finally, the output from the filtering is written to a file located on the Caldera agent at the path specified by the variable \$path2.

```

1  # Get $paw id for the agent
2  $paw = $result.host_group.paw
3  $specific_steps = $result.steps.$paw.steps |
4      Select-Object name, description, ability_id, command, status, attack, output
5
6  # Convert the filtered result to JSON
7  $specific_result_json = $result |
8      Select-Object name, start, finish, adversary, skipped_abilities, @{
9      Name="steps";
10     Expression={ $specific_steps }
11 } | ConvertTo-Json -Depth 10
12
13 # Write the filtered result to a file
14 $path2 = "C:\Users\badmin\Desktop\filtered_report.txt"
15 $specific_result_json | Out-File -FilePath $path2

```

**Listing 15:** Filtering and saving the operation results into a file

## Chapter 7

# Evaluation

This chapter is devoted to evaluating our solution and determining whether it meets the requirements outlined in Chapter 4.1. We will examine how well our implementation matches the desired outcomes and discuss any discrepancies we have encountered. A full test case will be rehearsed thoroughly and evaluated alongside our subjective impressions. By doing this, we can determine the strengths and weaknesses of our implementation in a real scenario. Each functional and non-functional requirement will be discussed with an evaluation of how well our system satisfies them, and areas for improvement. This provides the reader with a better understanding of the effectiveness of our solution in regard to meeting the specified requirements. Lastly, feedback from the project owners evaluating the degree to which our solution met the project requirements will be showcased.

### 7.1 Test Case

The purpose of this test case is to demonstrate how our system performs when used in the intended way, in a real-world scenario, from start to end. We will set up the system to run with sample inputs and present the results along with relevant illustrations and observations. This walk-through is done according to the steps and documentation provided in our wiki. It will act as an example of how the system should be used, in its easiest form. The wiki is available for users with access to our Azure Git Repository but is also accessible in the Appendix B.

To get started, we open up the Azure DevOps web GUI and log in to an account with access to the SOC environment. As shown in Figure 7.1, we can find and enter the Git repository for our system, where we want to run the STEDP.

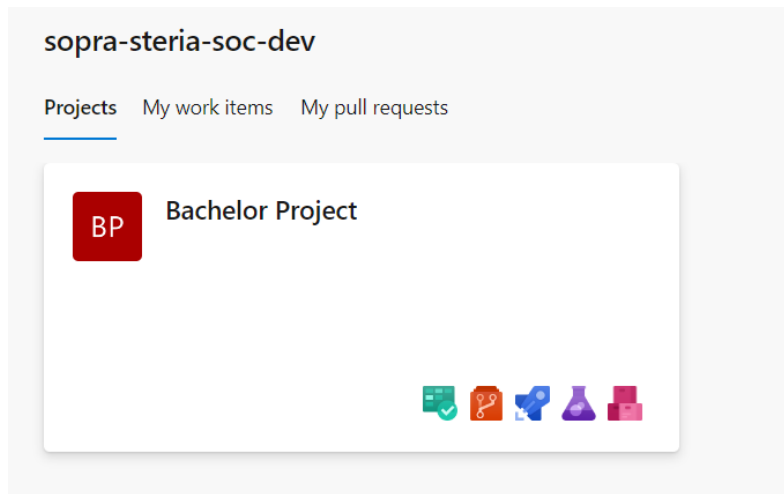


Figure 7.1: Azure DevOps Web GUI

After entering the repository, we can navigate to the pipelines in our system through the menu on the left. This is shown below, in Figure 7.2.

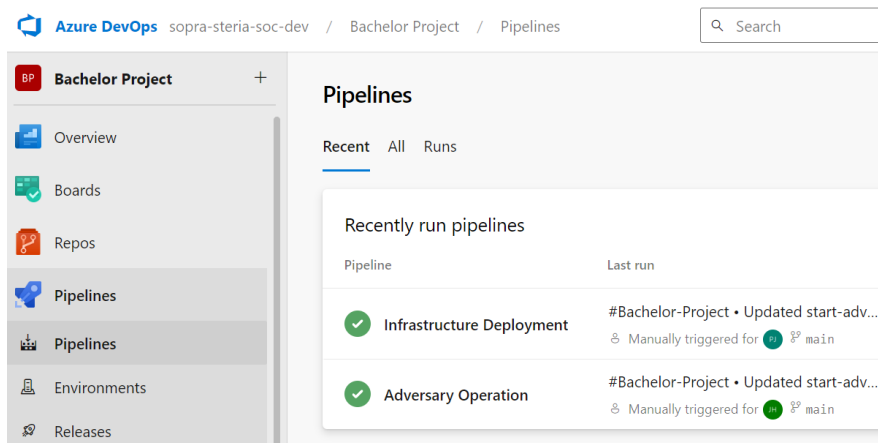


Figure 7.2: Navigation to pipelines

From here, we get an overview of the existing pipelines in our solution. It shows the status of the previous run and provides a simple clickable interface in which we can run the pipelines. We proceed by choosing the pipeline we want to run, which is the STEDP for infrastructure deployment since we are doing the initial setup. We do not have to change any variables or configure stages, as the default configuration should be sufficient in most cases.

**Run pipeline** ✕

Select parameters below and manually run the pipeline

Branch/tag

🔗 main ▾

Select the branch, commit, or tag

**Advanced options**

**Variables** ➤  
This pipeline has no defined variables

**Stages to run** ➤  
Run as configured

**Resources** ➤  
Use latest version of all resources

Enable system diagnostics

Cancel Run

**Figure 7.3:** Running the STEDP



We then run the pipeline and are prompted with a status page as shown in Figure 7.4 below, which displays the progress of each stage. Recall that these stages are explained in Chapter 4.2.1. The execution will usually take 10-20 minutes. This is mentioned in our Wiki, which is available in Appendix B.

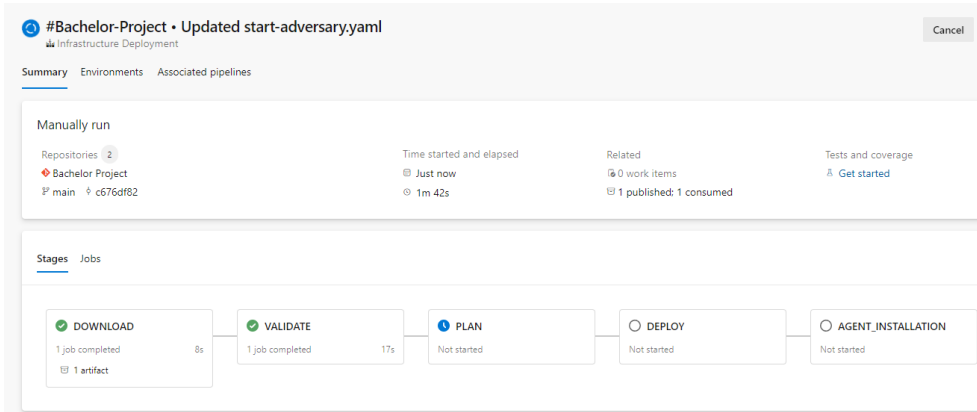


Figure 7.4: Pipeline stage status

Upon completion of the STEDP, the Security Testing Environment is deployed to the detection lab of Sopra Steria. We confirm this by checking the existing virtual machines and container instances in the Azure environment, as shown in Figure 7.5 below.

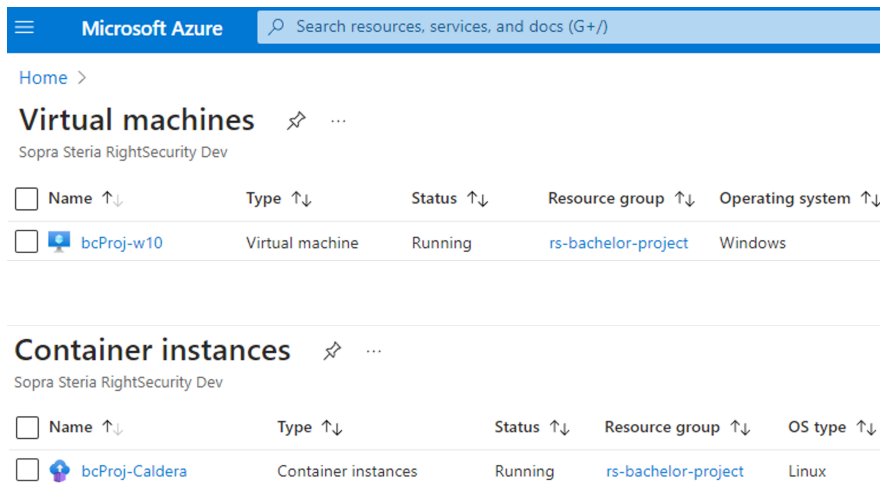
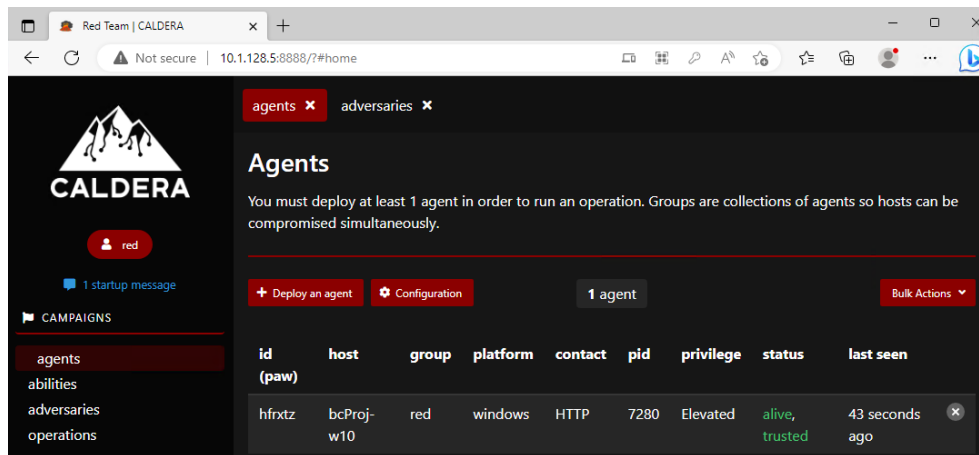


Figure 7.5: Deployed infrastructure

The last stage of the STEDP should set up a MITRE Caldera agent onto the virtual machine, which allows the Caldera container instance to execute security tests on said VM. We can confirm this is installed and functional by accessing the MITRE Caldera Web Interface. This is done by connecting to the VM through a remote desktop and opening a web browser with the URL to the Caldera container. This URL is by default the IP address of the container, followed by 8888. Accessing this successfully allows us to see the deployed agent with the id "hfrxtz". It is deployed to the virtual machine "bcProj-w10" with the status alive and trusted as seen in the following Figure, 7.6.



The screenshot shows the MITRE Caldera web interface. The browser address bar displays "10.1.128.5:8888/?#home". The interface includes a sidebar with navigation options: agents, adversaries, abilities, operations, and CAMPAIGNS. The main content area is titled "Agents" and contains a table with the following data:

id	host	group	platform	contact	pid	privilege	status	last seen
hfrxtz	bcProj-w10	red	windows	HTTP	7280	Elevated	alive, trusted	43 seconds ago

Figure 7.6: Caldera agent status in web interface

The resulting infrastructure we just examined should allow us to run security tests and produce a report through the second pipeline, the Adversary Operation Pipeline (AOP). The AOP makes use of the deployed Caldera agent and runs a set of security tests based on the linked adversary ID. This ID can be configured in the Azure DevOps web interface by navigating to Pipelines > Library > Adversary Operation Variables, and then changing the value of **AdversaryProfileID** under Variables, as shown in red in Figure 7.7. The ID is by default set to a profile which is recommended by us as an initial collection of tests to confirm the security testing is functional. The profile is called "Check (discovery)" and runs a set of initial tests which extract basic system information.

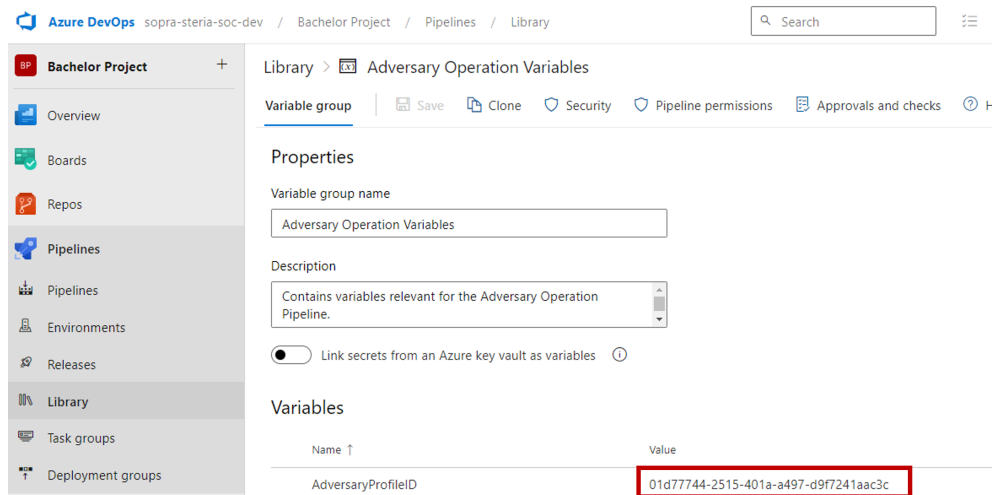


Figure 7.7: Configuring the adversary profile ID value for the AOP

We proceed by going back to the pipeline overview, choosing the AOP. As with the previous pipeline, we engage the pipeline and are provided with a status page. The duration of the run depends on the chosen adversary profile and the types of tests. From the Azure web interface we can see that the pipeline is running, but to access detailed progress information we need to enter the Caldera Web Interface again. Inside the web interface, under operations, we can choose the operation which by default is called "testoperation". We can then see each test being run as part of the operation, along with detailed information about them in Figure 7.8.

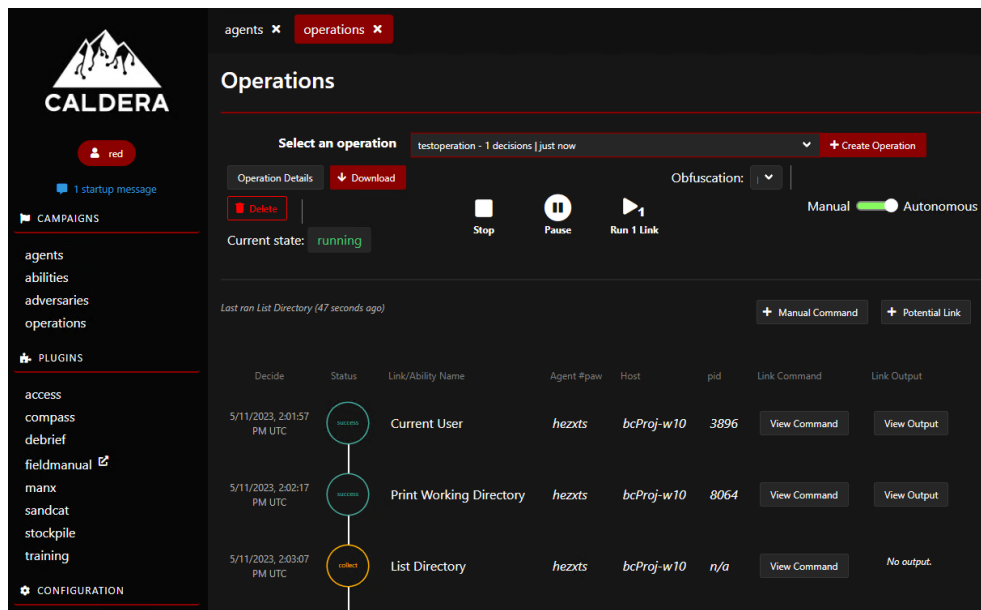


Figure 7.8: Currently running operation

For each completed test, we can click on "View Output" to get the actual result from running said test. An example output is provided in Figure 7.9, which confirms that the current user on this defending VM is "badadmin".

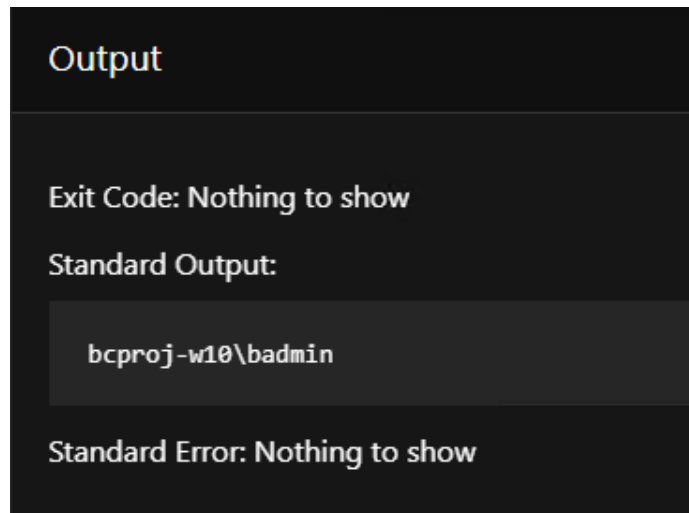


Figure 7.9: Sample output, from ability "Current user"

After successful execution, a text file report is produced and placed on the desktop of the defending VM. This file contains some of the same information accessible from the Caldera web interface, but in a complete form that can be filtered to our liking. The following is a snippet of this text report, to highlight some of the available information that can be worked with.

```
1  "ability_id": "bd527b63-9f9e-46e0-9816-b8434d2b8989",
2  "command": "whoami",
3  "plaintext_command": "whoami",
4  "delegated": "2023-05-11T14:01:57Z",
5  "run": "2023-05-11T14:02:16Z",
6  "status": 0,
7  "platform": "windows",
8  "executor": "psh",
9  "pid": 3896,
10 "description": "Obtain user from current session",
11 "name": "Current User",
12 "attack": {
13   "tactic": "discovery",
14   "technique_name": "System Owner/User Discovery",
15   "technique_id": "T1033"
16 },
17 "output": {
18   "stdout": "bcproj-w10\badmin",
19   "stderr": ""
```

Listing 16: Sample report-snippet after successful operation

## 7.2 Requirements and Confidence Level

In order to evaluate how well our solution meets the requirements of the project, we use confidence level as a measurement. This is a simplified, subjective way of evaluating how assured we are in our solution meeting a requirement. The table below defines what different confidence levels entail in the intervals of 0, 25, 50, 75, and 100 percent.

**Table 7.1:** The definition of confidence level in this thesis

	<b>Confidence Level</b>
100%	We are certain that the requirement has been met completely
75%	The main parts of the requirement have been met, but there is room for improvement
50%	Some of the requirement has been fulfilled, but there is a lot of room for improvement
25%	There is still a lot to do in order to fulfill the requirement
0%	No progression has been made to meet the requirement

## Functional Requirements

### **F1: The system should be able to automatically deploy infrastructure using Azure DevOps Pipelines.**

This implies that the deployment process should be streamlined, efficient, and require minimal manual intervention.

Our solution largely satisfies Requirement F1 of automatic deployment of the testing infrastructure via Azure DevOps Pipelines. It provides the capability to deploy a container instance based on a MITRE Caldera container image, a virtual machine intended to be used as the target for attacks, and a Caldera agent connected to this virtual machine. The infrastructure is deployed by running the STEDP, which is initiated by clicking a button in the Azure web interface. This functionality is the main and crucial part of this requirement and is fulfilled. However, we note that the current implementation only includes one container, one virtual machine, and an agent, all running in one specific version and a rather static state. The Caldera container image is based on a manually built image, instead of a continuously updated official one as that could not be found. The flexibility and robustness could therefore be determined as low, as some changes or updates could potentially break the solution. Since these aspects are considered of secondary importance, they do not compromise the ability of the system to fulfill the requirement currently. Therefore, we evaluate Requirement F1 as met with a confidence level of 90%.

### **F2: The system should be able to autonomously perform security tests within the detection lab.**

This requirement aims to eliminate the need for manual intervention in the testing process and ensure that the system can perform efficient cybersecurity evaluations.

We meet this requirement by providing a testing pipeline that automatically runs a predefined set of security tests within the detection lab which generates a report. The pipeline can be started with the click of a button, and once initiated, all necessary security tests are executed without requiring manual intervention. Additionally, our solution offers users the ability to define what collection of tests or adversary profiles are executed by simply changing the ID value for the adversary profile. While the default configuration runs a basic set of tests based on an adversary, the solution could be expanded upon to include custom tests outside of what is included in MITRE Caldera. After running the tests, a report outlining the results is generated. The report shows whether the security tests were successful, and the outputs or responses if applicable. It still needs manual review and does not suggest improvements or fixes on its own. Overall, the testing pipeline does autonomously perform security tests in the detection lab, but the set of tests and the report generated could be improved and expanded upon. We can conclude that our solution meets Requirement F2 with a confidence level of 70%.

**F3: Users should be able to define the tests or adversary profiles that the system executes.**

This requirement provides flexibility and customization options for users, allowing them to tailor the testing process according to their specific needs and requirements.

The solution aims to fulfill this requirement by allowing users to specify an adversary identity before running the security testing pipeline, which determines what tests are run. This is achieved through the use of an environment variable that can be easily modified in the Azure web interface. It is worth noting that the adversary identities used are based on MITRE ATT&CK adversary IDs, and limits the available options to those included in vanilla MITRE Caldera. The customization of the tests beyond the standard profiles provided requires additional knowledge and customization inside of MITRE Caldera, which can be done through its web interface or API but is not a formal component of our system. While this may present some limitations, the ability to specify which tests to run still fulfills the functional requirement with a confidence level of 80%.

## Non-Functional Requirements

**NF1: The solution should be open-source, non-copyrighted, and customizable.**

This requirement ensures that the solution can be easily modified, adapted, and shared by users without any legal or copyright issues.

The system meets Requirement NF1 entirely, as it uses only open-source software and frameworks that are entirely free to modify and use according to our needs. The MITRE Caldera and MITRE ATT&CK frameworks and tools for deployment such as Docker, and coding used directly in our solution all comply with this requirement. We have made use of non-open-source tools in our development process, but these tools are not directly part of our solution. Therefore, we consider this requirement to be completely fulfilled, with our solution meeting this requirement with a confidence level of 100%.

**NF2: Documentation for security tests should be mapped to MITRE ATT&CK.**

This requirement ensures that the testing process is based on a widely recognized and accepted framework, making it easier to compare and share results with others in the cybersecurity field.

All security tests in the solution are integrated parts of MITRE Caldera and are directly mapped to adversaries and tests in MITRE ATT&CK. This strong integration between the different components in the solution ensures that documentation for security tests is properly mapped to MITRE ATT&CK. Moreover, the fact that MITRE Caldera is developed and published by the MITRE Corporation adds



to the reliability of this fulfillment. We consider this requirement fulfilled with a confidence level of 100%.

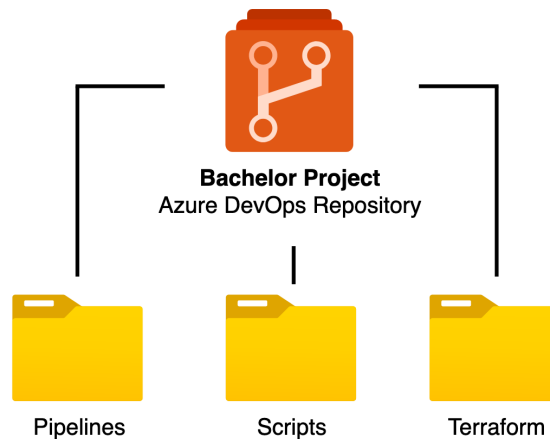
**NF3: The installation process for the solution should be well-documented, providing clear instructions and guidelines for users to follow.**

The documentation, which we refer to as the Project Wiki can be found as a part of our Azure Git Repository or in Appendix B. Our solution meets the Requirement NF3 by providing clear and concise documentation that includes step-by-step instructions for the installation process, as requested by our client. Users can easily follow these instructions to set up the system without any ambiguity. This is followed by an overview of some TTPs linked to their relevant MITRE ATT&CK pages, from a selection of the available adversary profiles. On the contrary, while the current documentation primarily covers the basic installation and usage, future improvements could expand the documentation to encompass additional usage scenarios and customization options within the Caldera platform. The resulting text file report from the AOP could also use more explanation for its use cases. We still consider this requirement fulfilled with a confidence level of 100%.

**NF4: The code for the solution must be stored in an Azure DevOps Repository created for the purpose.**

This requirement ensures that the codebase is managed and version-controlled effectively, enabling collaboration and easy tracking of changes.

The code for our solution is entirely stored within a dedicated Azure DevOps Git repository made specifically for our solution as illustrated in Figure 7.10. This ensures that the code and related materials are properly organized as planned. As such, we are confident that this requirement is fully satisfied with a level of 100%.



**Figure 7.10:** Solution Stored in Azure DevOps Repository

**NF5: The pipeline for the solution must be hosted in Azure DevOps Pipelines.**

This requirement ensures that the pipeline is securely hosted and easily accessible to authorized users, facilitating efficient deployment and testing processes.

The pipelines for our solution are entirely configured and run within Azure DevOps Pipelines, as planned. This requirement is considered fulfilled with a confidence level of 100%.

**NF6: The infrastructure for the solution must be deployed to the Sopra Steria Azure Cloud Environment.**

This requirement ensures that the solution is hosted on a secure and reliable cloud infrastructure of our client, providing a stable and scalable environment for testing and evaluation purposes.

The solution was continuously developed and tested within the detection lab of our client. This resulted in a successful system deployed and compatible with the Azure cloud environment of Sopra Steria, which meets the Requirement NF6. However, the compatibility and functionality of the solution in other environments is currently unknown, if our client were to need this. We consider this requirement to be fulfilled with a confidence level of 100%.

**NF7: Evaluation of frameworks for automatic cybersecurity testing.**

This is to ensure we are using a reasonable, acknowledged framework suited for our use. There is a multitude of opportunities here, and making a decent choice and attesting to it is therefore crucial.

The evaluation of cybersecurity frameworks for automatic security testing is a distinct part of the project and is accorded a dedicated section within the Discussion Chapter, under 8.2. Although not a direct practical component of the developed solution, it has been a significant focus of the research effort and our thesis. The team has thoroughly evaluated and assessed multiple frameworks for their compatibility with the proposed system. This aspect has been taken into consideration consistently throughout the development and research process. This requirement has thus been fulfilled and is documented in the relevant section of the thesis. There are some options that we did not cover, but for our purpose and scope, we consider this requirement to be fulfilled with a confidence level of 100%.

## Requirement Evaluation Results

Below are the summarized results of our subjective evaluation of the functional and non-functional requirements of the project.

**Table 7.2:** Evaluation Results for the Functional Requirements

No.	Requirement Description	Confidence Level
F1	Automatic deployment of infrastructure via Azure DevOps Pipeline	90%
F2	The ability of the system to autonomously perform security tests within the detection lab, eliminating manual intervention and promoting efficient cybersecurity evaluations	70%
F3	Users should be able to define what tests or adversary profiles are executed	80%

**Table 7.3:** Evaluation Results for the Non-Functional Requirements

No.	Requirement Description	Confidence Level
NF1	Open-Source solution; non-copyrighted and customizable	100%
NF2	Documentation for security tests should be mapped to MITRE ATT&CK	100%
NF3	Documentation for installation	100%
NF4	The code for the solution must be stored in an Azure DevOps repository made for the purpose	100%
NF5	The Pipeline must be hosted in Azure DevOps Pipelines	100%
NF6	The Infrastructure for the solution must be deployed to the Sopra Steria Azure Cloud Environment	100%
NF7	Evaluation of frameworks for automatic cybersecurity testing	100%

## **Project Owner Feedback**

We reached out to the project owner, Sopra Steria, to provide a counter to the possible biases we as a group had regarding the evaluation of our own project requirements. As a basis for evaluation, they were given a brief explanation of confidence levels and were provided access to our project files and thesis. The overall goal was to make them examine each functional and non-functional requirement, and then evaluate how well the project met those requirements with written feedback and confidence level scores.

Please see Appendix C for detailed feedback from the project owner. In summary, Sopra Steria thought their requirements might have been too simple, and that we may have been evaluating ourselves too harshly. They gave all but one requirement a confidence level of 100%. Requirement F2, concerning the ability of the system to autonomously perform security tests, got a confidence level of 95%. This was because they thought the solution could have been a little more user-friendly. Running security tests could have been more automated, and reports could have been easier to retrieve, but the feedback was otherwise positive. Overall, the project owners expressed satisfaction with our solution and its adherence to the set requirements.

## Chapter 8

# Discussion

This chapter serves as a comprehensive discussion of the work and decisions made throughout the project, including both successes and challenges encountered. There will also be a detailed account of mistakes made, along with an analysis of the unexpected changes in the plans that occurred during the project. Through this discussion, we aim to provide valuable insights, our subjective opinions, and lessons learned that can inform and improve future projects. It also aims to discuss alternatives and justify the choice of frameworks and tools. This discussion is of significance due to the research-heavy nature of our project, and Requirement NF7 in Chapter 4.2 being a formal requirement desired by the project owners.

### 8.1 The Project Task

At the start of our project, we had several discussions with the project owners to define the scope and objectives of our work. The main task was to evaluate different cybersecurity frameworks for security testing in a detection lab and propose a solution that met the specific requirements for the solution. The requirements stated that the solution should be open-source, easily customizable, and provide integration with the existing security solutions within Sopra Steria. We were given a high degree of freedom in our approach to the task, which allowed us to explore various options and make informed decisions based on our research and testing.

Said freedom allowed us to evaluate and make decisions that suited our use but also made it difficult to make conclusive decisions regarding the direction to take. This, combined with us being moderately unfamiliar with the options in the field, led to us spending a significant amount of time researching and discussing frameworks, tools, and solutions.

### **Direction and Consequences**

As we embarked on our project, intending to explore possibilities through an initial prototype, it gradually evolved into our primary focus and eventually became the ultimate solution. This final product effectively addresses the key requirements outlined in the original task, albeit with less emphasis on additional or innovative features. Depending on the perspective of the evaluator, one could perceive this approach as reliable and predictable, leveraging established methods and techniques, or perhaps unremarkable in its conventional implementation. However, we argue that by adhering to proven and recognized approaches, we mitigate the risks associated with our limited expertise and time constraints.

With a clear understanding of the project objectives and a set direction, our subsequent efforts were dedicated to identifying the most suitable tools and frameworks to meet the specified requirements.

### **Meeting the Requirements**

The requirements were primarily based on the guidelines outlined in our original project task. We also fine-tuned these requirements to align them with our desired outcome for the product. These changes reflect findings and decisions made in the research phase. The project owners were supportive of our efforts to make independent choices in this regard. The requirements, which have been discussed in Chapter 4.1, describe what our solution and to an extent thesis should do and cover in detail. These have also been evaluated against our system, in Chapter 7 under 7.2.

To meet these requirements, we engaged in the task of researching frameworks, as specified in Requirement NF7, to explore the available options and make better, informed decisions. The main objective of Requirement NF7 was to identify a cybersecurity framework that would be capable of conducting comprehensive testing of endpoint security products in a controlled environment. Further, the framework should easily integrate with other tools and allow for customization to meet specific project requirements. This would lay the foundation for a framework that would effectively meet our project objectives by providing the essential qualities of flexibility, reliability, and robustness. With this in mind, we then turned our attention to evaluating the available options, including specific security tools that were not considered frameworks.

### **Security Tools**

In our research phase, we discovered that there were few viable options available in the area that met our requirements from scratch. Our focus was on finding a framework for general automatic security testing against Windows machines, instead of specific targeting of a software program or web application. There are some key differences separating security frameworks developed for networking,

web, and operational technology (OT), or comprehensive ones for adversary emulation against operating systems, which we desired. While there is a multitude of tools available to assist in security testing within each of these fields, our requirements made that pool smaller.

In our work, we considered and explored some great security tools we already had experience with. Including BurpSuite for advanced web exploitation [66], Nessus for external vulnerability scanning [67], or many of the tools integrated into the Kali Linux distribution designed for digital forensics and penetration testing [68]. However, there were certain problems with most of these tools. Firstly, some of them are not open-source. Our requirements clearly state that all directly used frameworks in our solution have to be open-source, free of copyright, and highly customizable. Secondly, some use cases are too specific. BurpSuite, as mentioned, is great for web exploitation and penetration testing but is not a fully-fledged framework for adversary emulation. Lastly, the relevance of some of these tools was limited. Nessus completes a thorough external scan and provides recommended actions to fix vulnerabilities. This does not fit into our internal focus where we assume initial access to be already possessed, and external security to be out of scope. These tools and others could potentially provide valuable forensics information, but we were mainly in need of an automated adversary emulation framework.

## 8.2 Evaluating Security Frameworks

This section contributes to aligning with Requirement NF7 in 4.2 to evaluate security testing frameworks. When exploring potential frameworks for our project, we considered several options. Worth noting are MITRE Caldera, Stratus Red Team, Infection Monkey, Leonidas, and PurpleSharp. Each of these offers unique features and capabilities tailored to specific aspects of red team testing or adversary emulation.

Caldera, developed by MITRE, was an immediate candidate as an automated adversary emulation platform. It was mentioned in the project task and was assigned the role as the safe option to fall back on in case no better alternative was found. This comprehensive framework offers a wide range of features for conducting red team testing and adversary simulations [69]. Its maturity, support from the cybersecurity community, and alignment with industry standards combined with it being suggested by our client made it a strong contender.

Stratus Red Team stood out as a framework specifically designed for emulating offensive attack techniques in cloud environments. Its granular and self-contained approach allows for precise simulations in the cloud, providing valuable insights into potential vulnerabilities [70]. It is mapped to the MITRE ATT&CK framework, but its focus on cloud-native environments meant it did not align with the intended scope of our project.

Infection Monkey, on the other hand, focuses on automated penetration testing. This tool allows organizations to simulate various attack scenarios, helping assess the security of their networks and identify potential weaknesses. It is mainly used for automated detection of data center boundaries and internal server security [71]. Recall that in Chapter 4.1 on requirements, specifically Requirement F2, which states that "The system should be able to autonomously perform security tests within the detection lab". We determine "within the detection lab" to not align with the broad and technically different term "cloud environments", which therefore disqualifies Infection Monkey.

Leonidas is an automated attack simulation framework specifically designed for cloud environments [72]. Similar to our conclusion on the use of Infection Monkey, Leonidas' focus on broader cloud environments means it does not entirely align with our requirements.

PurpleSharp, an adversary simulation tool, executes adversary techniques in monitored Windows environments [73]. It is not widely used, only briefly mentioned in a single scholarly article that we could find which states that "PurpleSharp realizes emulation with two RATs installed on a tested system that is synchronized by an external agent. Some of these tools are dependant on agents and all of them emulate attacks incoherently, which makes the emulation less realistic and limited." [35]. While relevant and mostly fitting for our requirements, we did not consider prioritizing it over MITRE Caldera, due to its limited community support and acknowledgment.

### **Choosing Caldera**

Given the options and considerations, we also contemplated the possibility of developing our own testing program or tests, potentially based on the Atomic Red Team framework. This option would have provided us with greater flexibility and control over the testing process. However, after considering our expertise and available time, we decided to stick with the somewhat safe and fully-fledged solution provided by MITRE Caldera. The decision was based on several factors. Firstly, Caldera is a proven framework with a high level of maturity and support from the cybersecurity community. Its alignment with industry standards and its comprehensive feature set made it a reliable choice for our red team testing requirements. Additionally, Caldera offered the necessary customization options and integration capabilities we needed to meet our project objectives. Developing our own testing program or tests, although potentially innovative, would have required significant time and resources to ensure its effectiveness and reliability. By choosing Caldera, we were able to leverage a well-established and widely-used framework, providing us with a solid foundation for our security testing. This decision allowed us to focus on conducting deployment and comprehensive testing in an automated manner while minimizing the risks associated with developing our own solution.



### Limitations and Potential Extensions

While MITRE Caldera proved to be a reliable and robust framework for our needs, we acknowledge the potential limitations and future opportunities for our solution.

One limitation lies in the usefulness of the generated reports. Although Caldera provides comprehensive reports with detailed information about the tests conducted and the results obtained, interpreting and utilizing this information effectively requires a strong understanding of the underlying concepts and techniques involved. Without sufficient knowledge and expertise, the value of the report may be diminished, limiting its practicality for less experienced users or those lacking in-depth knowledge of red teaming and adversary emulation. The report does not provide a simple suggested plan of action, which would increase user-friendliness. It is important to consider the level of expertise required to effectively utilize the report and provide appropriate training or documentation to ensure its maximum utility.

Another limitation of our solution is its relatively inflexible nature. While it allows for easy switching between predefined adversary profiles, making changes to the target machine or incorporating it into an existing environment without running our deployment pipeline can be challenging. The current implementation assumes a specific environment and configuration, making it less adaptable to diverse setups. Enhancements in this area could involve developing more flexible deployment options that allow for seamless integration with existing environments, providing users with greater control and customization possibilities.

To overcome these limitations and further expand the capabilities of our solution, several potential extensions can be explored. One avenue for improvement is through the integration of plugins for MITRE Caldera, which it allows for and explains in its official documentation [40]. By creating plugins that integrate additional tools and functionalities, we can enhance the range of available and useful tests. For example, integrating Metasploit, a popular penetration testing framework [39] through Caldera's plugins [74] is an option we experimented with. It would significantly expand the number of attack vectors and techniques that can be simulated, providing a more comprehensive evaluation of the security posture of the target system. This integration would enable users to leverage the extensive capabilities of Metasploit alongside Caldera, increasing the effectiveness and coverage of our testing scenarios.

In addition to plugin development, future work could focus on expanding the repository of predefined adversary profiles within Caldera. By continuously researching and incorporating new TTPs, the framework can remain up to date with the evolving threat landscape. This could either be done by the developers of MITRE Caldera, and then by us through keeping the software up to date, or we could manually create adversary profiles or operations. This would ensure

that the security tests conducted using Caldera reflect the latest adversarial techniques, providing a more realistic and relevant assessment of the target system's defenses.

## Chapter 9

# Closing Remarks

In this chapter, we shall discuss the learning outcomes derived from the project. Subsequently, we will collectively arrive at a conclusion and explore ideas for further work that could improve the solution in the future.

### 9.1 Learning Outcome

Throughout the duration of this project, we dedicated a significant amount of effort to working with MITRE Caldera, as it was chosen as the framework for our solution. We learned how to configure the VMs as Caldera agents, by utilizing both the GUI and the CLI. Additionally, we thoroughly explored the use of various APIs within Caldera to facilitate the automation of the security testing processes.

Prior to undertaking this project, our team possessed limited experience in conducting security tests. Through our work with the project, we actively acquired knowledge regarding the diverse TTPs used by malicious actors. By utilizing the MITRE ATT&CK framework we gained firsthand experience in establishing connections between TTPs and the corresponding security tests.

In our solution, we incorporated the use of Terraform in conjunction with Microsoft Azure to establish and deploy the Security Testing Environment. Initially, we needed to acquire the knowledge of constructing various resources within Azure to successfully implement our solution. Fortunately, our client provided us with code examples that served as guidance throughout the learning process. With their support, we were able to learn how to effectively utilize Terraform and Azure, to set up the required infrastructure for our Security Testing Environment.

As a team, we have learned how important it is with good communication and open dialogue. We are a team of four, with a mix of prior working relationships. Two of us had previously collaborated, as had the other two. However, working together as a complete team was a new experience for all of us. This situation

provided us with an opportunity to learn how we worked as a team, discovering effective work methods and communication skills.

A few members of our team had little experience with writing in Overleaf LaTeX [75]. The thesis writing in itself required more time than first assumed. However, this experience taught us a valuable lesson in terms of the importance of not postponing thesis writing until the final stages, but rather engaging in continuous writing throughout the project.

## 9.2 Conclusion

The primary objectives of our project revolved around automating the detection lab testing process for Sopra Steria and evaluating open-source security frameworks. A considerable portion of our project time was dedicated to researching various frameworks suitable for our purposes. However, we soon discovered that the framework recommended by the project owners, MITRE Caldera, aligned exceptionally well with our requirements. This made us look deeper into understanding the implementation and utilization of MITRE Caldera, ensuring we knew how to utilize its capabilities.

In order to meet the project requirements, we made the decision to implement two distinct pipelines: one for constructing the testing environment and another for executing the security tests. The separation made for easier execution of different security tests without the need to set up a new environment for each test. However, this approach slightly reduced the level of automation, since it required manual utilization of two pipelines instead of one pipeline. Nonetheless, this decision was made with the project owners, so they could easier execute the different security tests.

To ensure usability for the project owners, we developed a wiki within the Azure repository. This documentation serves as a guide for utilizing our solution, with a focus on the usage of the two pipelines. The wiki provides step-by-step instructions, enabling effective utilization of the pipelines. Additionally, we included a few examples of various tests that could be executed.

In conclusion, our solution successfully meets the set project requirements, even though the requirements might have been too narrow in scope as suggested by our clients in their evaluation of our project. While alternative approaches for our solution could have been considered, time constraints, limited familiarity with MITRE Caldera, and a lack of extensive experience in security testing posed challenges. However, as a team, we are satisfied with the implementation of our solution. Despite the challenges encountered, we have managed to deliver a solution that addresses the project's objectives and provides a reliable framework for automating and conducting security tests.

## 9.3 Further work

Throughout our project, we have gained valuable insights and ideas that could not be fully realized within our given time frame. These ideas pave the way for potential further work, offering exciting opportunities for those interested in expanding our solution. In this section, we will briefly discuss some of these ideas, providing a glimpse into the possibilities for extending and improving them.

### 9.3.1 Overall System

Considering the overall system and our direction, there are some key aspects that could use more research and work to ensure a complete solution. For example, the interaction between the system and the detection lab environment, particularly how the Azure and Windows security measures respond to our system and tests. Understanding how the system integrates with these components and performs in the presence of various security measures, including antivirus software needs further exploration.

The actual security tests used in our solution seem comprehensive to the best of our knowledge but should be revised in detail and perhaps expanded upon. The information in the results gathered by the execution of these tests also requires further revision. It could be that some of the results provide information that is either incorrect or simply not useful enough. We have confirmed the results we could in our testing and laid our trust in the MITRE organization for the remaining parts, including mapping to the MITRE ATT&CK matrix. Researching and confirming details beyond what is provided in our thesis would be beneficial in order to validate the effectiveness of the solution.

### 9.3.2 MITRE Caldera

The possibilities for improvements with regard to our use of MITRE Caldera are many. One aspect is enabling access to the Caldera web server from outside of the relevant VM. The setup currently requires the user to access the Caldera Web Interface from within the VM, which is inconvenient. Finding a way to access this interface from any machine, possibly through a Virtual Private Network with appropriate network rules, could make this easier.

Another area that could benefit from improvement is the static nature of the Caldera image. The current image is built by us, remains static, and does not incorporate updates or new features. Updating the image periodically or finding a way to enable automatic updates would ensure that users have access to the latest features and security enhancements. This limitation exists due to us not finding a publicly hosted image that remains maintained. MITRE used to update an image on Docker Hub about two years ago which would simplify a fix, but it has since been abandoned [76].

As for extensions of the framework features and security tests, this is broadly covered in the Discussion Chapter, under 8.2. Elements to add to this could be: Exploring the additional existing features of Caldera, fetching and providing the report in a reasonable format and place, and providing a plan of action to improve found vulnerabilities.

### **9.3.3 Infrastructure Deployment**

In general, we believe that our solution effectively solves the problem of automatic deployment of infrastructure. However, one issue is the manual cleanup of infrastructure resources that no longer serve a purpose. The cleanup process could be automated by implementing a 'destruction pipeline.' This pipeline would reverse the implementation of our Terraform model by deleting any resources the model is configured to create. This would allow users to clean up unneeded resources with a single action instead of deleting each resource manually. It is worth mentioning that our solution currently only creates two resources, making a manual deletion relatively simple. Because of this, we were unable to prioritize the development of a destruction pipeline within the time limit of our project.

# Bibliography

- [1] arm, *Control processing unit (cpu)*. [Online]. Available: <https://www.arm.com/glossary/cpu>.
- [2] Atlassian, *What is kanban?* [Online]. Available: <https://www.atlassian.com/agile/kanban>.
- [3] Gartner, *Managed security service provider (mssp)*. [Online]. Available: <https://www.gartner.com/en/information-technology/glossary/mssp-managed-security-service-provider>.
- [4] Gartner, *Operational technology (ot)*. [Online]. Available: <https://www.gartner.com/en/information-technology/glossary/operational-technology-ot>.
- [5] Blumira, *Remote access tool (rat)*. [Online]. Available: <https://www.blumira.com/glossary/remote-access-tool-rat/>.
- [6] Atlassian, *What is scrum?* [Online]. Available: <https://www.atlassian.com/agile/scrum>.
- [7] Arcitura, *Cloud-based security groups*. [Online]. Available: [https://patterns.arcitura.com/cloud-computing-patterns/mechanisms/cloud\\_based\\_security\\_groups](https://patterns.arcitura.com/cloud-computing-patterns/mechanisms/cloud_based_security_groups).
- [8] IBM, *What is a security operations center (soc)*. [Online]. Available: <https://www.ibm.com/topics/security-operations-center>.
- [9] Cambridge Dictionary, *Vanilla - adjective*. [Online]. Available: <https://dictionary.cambridge.org/dictionary/english/vanilla>.
- [10] Sopra Steria, *About us*. [Online]. Available: <https://www.soprasteria.com/about-us>, (accessed: 24.03.2023).
- [11] Sopra Steria, *Sikkerhetstjenester*. [Online]. Available: <https://www.soprasteria.no/dette-kan-vi-fagomrader/sikkerhetstjenester>, (accessed: 24.03.2023).
- [12] NSM. 'Kvalitetsordning for leverandører som håndterer ikt-hendelser.' (Nov. 2022), [Online]. Available: <https://nsm.no/fagomrader/sikkerhetsstyring/leverandorforhold/kvalitetsordning-for-leverandorer-som-handterer-ikt-hendelser>.

- [13] Microsoft, *What is azure pipelines?* [Online]. Available: <https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops>.
- [14] MITRE, *Att&ck matrix for enterprise*. [Online]. Available: <https://attack.mitre.org/>, (accessed: 16.02.2023).
- [15] Cloudflare, *What is penetration testing?* [Online]. Available: <https://www.cloudflare.com/learning/security/glossary/what-is-penetration-testing/>.
- [16] NTNU, *Digital infrastruktur og cybersikkerhet*. [Online]. Available: <https://www.ntnu.no/studier/bdigsec>.
- [17] AttackIQ, *Intermediate purple teaming*. [Online]. Available: <https://www.academy.attackiq.com/learning-path/intermediate-purple-teaming>.
- [18] Microsoft, *What is cloud computing?* [Online]. Available: <https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-is-cloud-computing>.
- [19] IBM, *What is virtualization?* [Online]. Available: <https://www.ibm.com/topics/virtualization>.
- [20] Citrix, *What is hardware virtualization?* [Online]. Available: <https://www.citrix.com/solutions/vdi-and-daas/what-is-hardware-virtualization.html>.
- [21] IBM, *What are virtual machines (vms)?* [Online]. Available: <https://www.ibm.com/topics/virtual-machines>.
- [22] Microsoft. 'How does azure work?' (Feb. 2023), [Online]. Available: <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/get-started/what-is-azure>.
- [23] Microsoft, *Azure devops*. [Online]. Available: <https://azure.microsoft.com/en-us/products/devops>.
- [24] CrowdStrike. 'What is security testing?' (Dec. 2022), [Online]. Available: <https://www.crowdstrike.com/cybersecurity-101/security-testing/>.
- [25] R. Mejia, 'Red team versus blue team: How to run an effective simulation,' *CSO Online-Security and Risk*, 2008. [Online]. Available: <http://aldeilis.net/mumbai/0682.pdf>.
- [26] J. Oakley, 'Purple teaming. in: Professional red teaming,' *Apress, Berkeley, CA*, p. 105, 2019. [Online]. Available: [https://doi.org/10.1007/978-1-4842-4309-1\\_8](https://doi.org/10.1007/978-1-4842-4309-1_8).
- [27] E. P. Shanto Roy, C. Noakes, A. Laszka, S. Panda and G. Loukas, 'Sok: The mitre att&ck framework in research and practice,' *Apress, Berkeley, CA*, p. 8, 2023. [Online]. Available: <https://arxiv.org/abs/2304.07411>.



- [28] Okta. 'Defining & understanding the mitre att&ck framework.' (May 2022), [Online]. Available: <https://www.okta.com/identity-101/mitre-attack/>.
- [29] Mitre, *Who we are*. [Online]. Available: <https://www.mitre.org/who-we-are>.
- [30] MITRE Corporation, *Enterprise matrix*. [Online]. Available: <https://attack.mitre.org/matrices/enterprise/>.
- [31] N. Poggi. 'Cybersecurity frameworks 101 - the complete guide.' (Jun. 2022), [Online]. Available: <https://web.archive.org/web/20221204142314/https://preyproject.com/blog/cybersecurity-frameworks-101>.
- [32] RedHat. 'What is open source?' (Oct. 2019), [Online]. Available: <https://www.redhat.com/en/topics/open-source/what-is-open-source>.
- [33] *Faqs*. [Online]. Available: <https://github.com/redcanaryco/atomic-red-team/wiki/FAQs>.
- [34] *Caldera*. [Online]. Available: <https://github.com/mitre/caldera>.
- [35] J. Pružinec, Q. A. Nguyen, A. Baldwin, J. Griffin and Y. Liu, 'Kubo: A framework for automated efficacy testing of anti-virus behavioral detection with procedure-based malware emulation,' in *Proceedings of the 13th International Workshop on Automating Test Case Design, Selection and Evaluation*, New York, NY, USA: Association for Computing Machinery, 2022, p. 39, ISBN: 9781450394529. DOI: 10.1145/3548659.3561307. [Online]. Available: <https://doi.org/10.1145/3548659.3561307>.
- [36] W. Booth, *Caldera documentation - plugin library, atomic*, 2023. [Online]. Available: <https://caldera.readthedocs.io/en/latest/Plugin-library.html?highlight=atomic+red+team#atomic>.
- [37] W. Booth, *Caldera - learning the terminology*, 2023. [Online]. Available: <https://caldera.readthedocs.io/en/latest/Learning-the-terminology.html?highlight=adversary%5C%20profile#operations>.
- [38] W. Booth, *Caldera - learning the terminology*, 2023. [Online]. Available: <https://caldera.readthedocs.io/en/latest/Learning-the-terminology.html?highlight=adversary%5C%20profile#abilities-and-adversaries>.
- [39] Metasploit, *The world's most used penetration testing framework*. [Online]. Available: <https://www.metasploit.com/>.
- [40] W. Booth, *Caldera documentation - plugin library, atomic*, 2023. [Online]. Available: <https://caldera.readthedocs.io/en/latest/Plugin-library.html>.
- [41] Splunk, *What is security automation?* [Online]. Available: [https://www.splunk.com/en\\_us/data-insider/what-is-security-automation.html](https://www.splunk.com/en_us/data-insider/what-is-security-automation.html).

- [42] Hashicorp, *What is terraform?* [Online]. Available: <https://developer.hashicorp.com/terraform/intro>.
- [43] D. Lemonaki, *What is coding? computer coding definition.* [Online]. Available: <https://www.freecodecamp.org/news/what-is-coding/>.
- [44] Oxford English Dictionary, *Scripting n.* [Online]. Available: <https://www.oed.com/view/Entry/173567#eid23814948>.
- [45] Microsoft, *What is powershell?* [Online]. Available: <https://learn.microsoft.com/en-us/powershell/scripting/overview>.
- [46] IBM, *What is an api?* [Online]. Available: <https://www.ibm.com/topics/api>.
- [47] IBM, *What is a rest api?* [Online]. Available: <https://www.ibm.com/topics/rest-apis>.
- [48] A. L. Solli, 'Automated red teams in maritime cybersecurity exercises,' *NTNU Open*, 2022. [Online]. Available: <https://ntnuopen.ntnu.no/ntnu-xmli/handle/11250/3006990>.
- [49] M. Nachaat, 'Study of bypassing microsoft windows security using the mitre caldera framework,' *Rabdan Academy, Abu Dhabi*, 2022. [Online]. Available: <https://f1000research.com/articles/11-422/v3>.
- [50] MITRE, *Agents.* [Online]. Available: <https://attack.mitre.org/tactics/TA0001/>.
- [51] T. E. Seongmo An, J. S. Park, J. B. Hong, A. Nhlabatsi, N. Fetais, K. Khan and D. S. Kim, 'Cloudsafe: A tool for an automated security analysis for cloud computing,' *Institute of Electrical and Electronics Engineers (IEEE)*, 2014. [Online]. Available: <https://ieeexplore.ieee.org/document/8887392>.
- [52] A. Wolter, *Security: Separation of privilege.* [Online]. Available: <https://techcommunity.microsoft.com/t5/azure-sql-blog/security-separation-of-privilege/ba-p/2393637>.
- [53] M. Glinz, 'On non-functional requirements,' in *15th IEEE International Requirements Engineering Conference (RE 2007)*, 2007, pp. 21–26. DOI: 10.1109/RE.2007.45.
- [54] MITRE, *Operation results.* [Online]. Available: <https://caldera.readthedocs.io/en/latest/Operation-Results.html?#sample-operation-report>.
- [55] E. Conrad, 'Waterfall model - an overview,' 2011. [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/waterfall-model>.
- [56] Microsoft, *About boards and kanban.* [Online]. Available: <https://learn.microsoft.com/en-us/azure/devops/boards/boards/kanban-overview?view=azure-devops>.
- [57] D. Inc., *Discord.* [Online]. Available: <https://discord.com/>.

- [58] Microsoft, *Get more done with microsoft teams*. [Online]. Available: <https://www.microsoft.com/en-us/microsoft-teams/group-chat-software>.
- [59] Microsoft, *What is infrastructure as code (iac)*. [Online]. Available: <https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code>.
- [60] HashiCorp, *Modules*. [Online]. Available: <https://developer.hashicorp.com/terraform/language/modules>.
- [61] Microsoft, *Azure pipelines task reference*. [Online]. Available: <https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/reference/?view=azure-pipelines>.
- [62] MITRE, *The rest api*. [Online]. Available: <https://caldera.readthedocs.io/en/latest/The-REST-API.html>.
- [63] Microsoft, *Invoke-restmethod*. [Online]. Available: <https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invok-restmethod?view=powershell-7.3>.
- [64] MITRE, *Server configuration*. [Online]. Available: <https://caldera.readthedocs.io/en/latest/Server-Configuration.html?highlight=ADMIN123#configuration-file>.
- [65] MITRE, *Agents*. [Online]. Available: <https://caldera.readthedocs.io/en/latest/Learning-the-terminology.html?highlight=paw#agents>.
- [66] P Corporation, *Burp suite - application security testing software*, 2023. [Online]. Available: <https://portswigger.net/burp>.
- [67] T. Inc., *Nessus vulnerability assessment tool*, 2023. [Online]. Available: <https://www.tenable.com/products/nessus>.
- [68] K. Team, *Penetration testing and ethical hacking linux distribution*, May 2023. [Online]. Available: <https://www.kali.org/>.
- [69] *Caldera*. [Online]. Available: <https://caldera.mitre.org/>.
- [70] D. Inc., *Home*, 2023. [Online]. Available: <https://stratus-red-team.cloud/>.
- [71] L. Chen, A. Xu, X. Kuang, H. Lv, H. Yang, Y. Yang and B. Li, 'Detecting advanced attacks based on linux logs,' in *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, 2020, p. 61. DOI: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00022.
- [72] WithSecureLabs, *Github - withsecurelabs/leonidas*. [Online]. Available: <https://github.com/WithSecureLabs/leonidas>.
- [73] M. Velazco, *Purplesharp*. [Online]. Available: <https://www.purplesharp.com/en/latest/>.

- [74] Metasploit, *Integrating metasploit*. [Online]. Available: <https://caldera.readthedocs.io/en/latest/Plugin-library.html#metasploit-integration>.
- [75] *Overleaf latex*. [Online]. Available: <https://www.overleaf.com/>.
- [76] MITRE, *Mitre caldera on docker hub*. [Online]. Available: <https://hub.docker.com/r/mitre/caldera>.

## **Appendix A**

### **Contract**

Attached is our working contract, signed by the project group, project owners and the supervisor.

*Fastsatt av prorektor for utdanning 10.12.2020*

## **STANDARDAVTALE**

### **om utføring av studentoppgave i samarbeid med ekstern virksomhet**

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

#### **Forklaring av begrep**

##### **Opphavsrett**

Er den rett som den som skaper et åndsverk har til å fremstille eksemplarer av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

##### **Eiendomsrett til resultater**

Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

##### **Bruksrett til resultater**

Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

##### **Prosjektbakgrunn**

Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

##### **Utsatt offentliggjøring**

Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

## **1. Avtaleparter**

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Institutt for informasjonssikkerhet og kommunikasjonsteknologi
Veileder ved NTNU: Jia-Chun Lin e-post og tlf. <a href="mailto:jia-chun.lin@ntnu.no">jia-chun.lin@ntnu.no</a> / 45849287
Ekstern virksomhet: Sopra Steria Ekstern virksomhet sin kontaktperson: Truls Dahlsveen Kontaktperson e-post og tlf: <a href="mailto:truls.dahlsveen@soprasteria.com">truls.dahlsveen@soprasteria.com</a>
Student: Jardar Hollås Fødselsdato: 15.11.1998
Student: Petter Jørgensen Fødselsdato: 22.04.1999
Student: Charlotte Larsen Fødselsdato: 06.03.2000
Student: Tryggvi T. Zabelberg Fødselsdato: 07.08.2001

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

## 2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	x
Prosjektoppgave	
Annen oppgave	

Startdato: 09.01.2023
Sluttdato: 20.05.2023

Oppgavens arbeidstittel er: Sikkerhetsautomatisering
---------------------------------------------------------

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

## 3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototype, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet dekker følgende utgifter til utførelse av oppgaven:  
VM hosting kapasitet i Microsoft Azure inntil avtalt grense

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

#### **4. Studentens rettigheter**

Studenten har opphavsrett til oppgaven<sup>1</sup>. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

#### **5. Den eksterne virksomheten sine rettigheter**

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

#### **Alternativ a) (sett kryss) Hovedregel**

<input checked="" type="checkbox"/>	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven
-------------------------------------	------------------------------------------------------------------

---

<sup>1</sup> Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1



Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

#### Alternativ b) (sett kryss) Unntak

<input type="checkbox"/>	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt
--------------------------	-------------------------------------------------------------------------------------------------------------------------------

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

#### 6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

#### 7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

#### 8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

<input checked="" type="checkbox"/>	Opgaven skal være offentlig
-------------------------------------	-----------------------------

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Opgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss	Sett dato
<input type="checkbox"/>	ett år
<input type="checkbox"/>	to år
<input type="checkbox"/>	tre år

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

## 9. Generelt

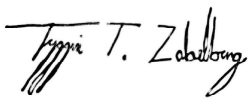
Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidsprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

**Signaturer:**

Instituttleder: Dato:
Veileder ved NTNU:  Jørund Løn
Dato: 24.01.2023
Student: Petter Fergerson Dato: 24.01.2023
Student: Jardar Hollås Dato: 24.01.2023
Student: Charlotte Jensen Dato: 24.01.2023
Student:  Dato: 24.01.2023
Ekstern virksomhet / dato: Truls Dahlsveen / 25.01.2023

## **Appendix B**

# **Project Wiki**

Attached is our Project Wiki, as seen in our Azure DevOps Repository when logged into the Azure Environment of our client.

# Project Wiki

Last updated by | Petter Jørgensen | May 20, 2023 at 1:12 PM GMT+2

---

## Step by step usage

### Prerequisites

1. Ensure that the [bcProj-w10 VM](#) ☐ and [bcProj-Caldera Container](#) ☐ does not already exist. If you wish to deploy multiple VM/Container pairs, this can be done by editing the 'name\_prefix' variable located in the [Terraform Environment](#) variable group. You may then change the Prefix to match the desired target before running the Adversary Operations Pipeline. In most scenarios however, the simplest solution is to delete existing infrastructure before deploying more.

### Infrastructure Deployment

1. Select the [Infrastructure Deployment Pipeline](#).
2. Click the 'Run Pipeline' button and then 'Run'. If you wish to run the pipeline on another branch than main, you can select another branch in the drop-down menu. The source branch for 'Resources' might also have to be changed.
3. Wait for the Pipeline to finish running. This will usually take 10-20 minutes.

Note: The username and password for the VM to be deployed can be changed by editing the 'module\_admin\_username' variable in the [Terraform Environment](#) variable group, and by replacing the 'module-admin-secret' secret located in Azure Key Vault [Caldera-Registry](#) ☐. If the password is changed, the new secret must be named the same, and it must then be re-imported into the [Pipeline-Credentials](#) variable group.

### Security Testing

1. Define which adversary profile to run. This is done by setting the value of the 'AdversaryProfileID' variable equal to the ID of the desired adversary profile. The 'AdversaryProfileID' variable is located in the [Adversary Operation Variables](#) variable group within the Pipeline Library. A sample of the available adversary profiles along with their documentation can be found at the bottom of this wiki.
2. Now select the [Adversary Operation Pipeline](#).
3. Click 'Run Pipeline'. The time it takes for the Pipeline to finish running varies depending on the Adversary Profile selected.
4. You can find the test results in a text file on the VM's desktop.

### Access VM or Caldera GUI

Caldera has a GUI accessible from a web-browser on the Virtual Machine. It can be accessed using the following steps:

1. RDP to the VM. IP can be found in [Azure](#) ☐. Default login credentials are: Username - 'badmin', Password - 'Administrator123'
2. Open a web-browser. Type '<container-ip>:8888' into the web-browser. You can find the IP for the container in [Azure](#) ☐.
3. Login using the credentials: Username - 'admin', Password - 'admin'.

## Sample Adversary Profiles

Below is a sample of adversary profiles that can be executed using the Adversary Operation Pipeline. Note that there are many more alternatives to choose from, all of which can be found by looking at the available adversaries in the Caldera GUI.









NB: The 'Windows Executor' attribute determines whether this procedure is capable of being executed on a Windows host. If 'no', then the Technique is skipped.

### Check

Caldera Adversary Profile ID: 01d77744-2515-401a-a497-d9f7241aac3c

Tactics: [Discovery](#) 

Techniques:

Tactic	Technique ID	Technique Name	Procedure	Windows Executor
Discovery	<a href="#">T1033</a> 	System Owner/User Discovery	Current User	yes
Discovery	<a href="#">T1083</a> 	File and Directory Discovery	Print Working Directory	yes
Discovery	<a href="#">T1083</a> 	File and Directory Discovery	List Directory	yes
Discovery	<a href="#">T1057</a> 	Process Discovery	View Processes	yes
Discovery	<a href="#">T1016</a> 	System Network Configuration Discovery	Network Interface Configuration	yes
Discovery	<a href="#">T1518</a> 	Software Discovery	Check Go	no
Discovery	<a href="#">T1518</a> 	Software Discovery	Check Chrome	no
Discovery	<a href="#">T1518</a> 	Software Discovery	Check Python	yes

### Discovery (discovery)

Caldera Adversary Profile ID: 0f4c3c67-845e-49a0-927e-90ed33c044e0

Tactics: [Discovery](#) 

Techniques:

Tactic	Technique ID	Technique Name	Procedure	Windows Executor
Discovery	<a href="#">T1033</a>	System Owner/User Discovery	Identify active user	yes
Discovery	<a href="#">T1087.001</a>	Account Discovery: Local Account	Find local users	no
Discovery	<a href="#">T1087.001</a>	Account Discovery: Local Account	Identify local users	yes
Discovery	<a href="#">T1016</a>	System Network Configuration Discovery	Snag broadcast IP	no
Discovery	<a href="#">T1057</a>	Process Discovery	Find user processes	yes
Discovery	<a href="#">T1135</a>	Network Share Discovery	View admin shares	yes
Discovery	<a href="#">T1018</a>	Remote System Discovery	Discover domain controller	yes
Discovery	<a href="#">T1069.001</a>	Permission Groups Discovery: Local Groups	Permission Groups Discovery	yes
Discovery	<a href="#">T1518.001</a>	Software Discovery: Security Software Discovery	Identify Firewalls	yes
Discovery	<a href="#">T1018</a>	Remote System Discovery	Discover Mail Server	yes
Discovery	<a href="#">T1217</a>	Browser Bookmark Discovery	Get Chrome Bookmarks	no

### Nosy Neighbor

Caldera Adversary Profile ID: 0b73bf34-fc5b-48f7-9194-dce993b915b1

Tactics: [Defense Evasion](#) , [Discovery](#) , [Impact](#)

Techniques:

Tactic	Technique ID	Technique Name	Procedure	Windows Executor
Defense Evasion	<a href="#">T1070.003</a>	Indicator Removal on Host: Clear Command History	Avoid logs	yes
Discovery	<a href="#">T1033</a>	System Owner/User Discovery	Identify active user	yes
Discovery	<a href="#">T1018</a>	Remote System Discovery	Collect ARP details	yes
Discovery	<a href="#">T1057</a>	Process Discovery	System processes	yes
Discovery	<a href="#">T1016</a>	System Network Configuration Discovery	Scan WIFI networks	yes
Discovery	<a href="#">T1016</a>	System Network Configuration Discovery	Preferred WIFI	yes
Impact	<a href="#">T1499</a>	Endpoint Denial of Service	Disrupt WIFI	yes

## Windows worm 2

Caldera Adversary Profile ID: 725226e0-45b8-4432-84ee-144d3f37ff8d

Tactics: [Discovery](#) , [Lateral Movement](#)

Techniques:

Tactic	Technique ID	Technique Name	Procedure	Windows Executor
Discovery	<a href="#">T1018</a>	Remote System Discovery	Collect ARP details	yes
Discovery	<a href="#">T1018</a>	Remote System Discovery	Reverse nslookup IP	yes
Lateral Movement	<a href="#">T1570</a>	Lateral Tool Transfer	Copy 54ndc47 (WinRM and SCP)	yes
Lateral Movement	<a href="#">T1021.006</a>	Remote Services: Windows Remote Management	Start Agent (WinRM)	yes



## **Appendix C**

# **Project Owner Evaluation - E-mail**

Attached are the e-mails sent between the project group and the project owners, with the client's evaluation of our solution. In Norwegian.

From: **Jardar Hollås** <jardarh@stud.ntnu.no>  
To: **Petter Jørgensen** <pettejor@stud.ntnu.no>; **Charlotte Larsen** <charllar@stud.ntnu.no>; **Tryggvi Thordarson Zabelberg** <tryggviz@stud.ntnu.no>  
Subject: FW: Evaluering av bacheloroppgave  
Date: 16.05.2023 14:12:43 (+02:00)

---

**From:** THORSTAD DAHLSVEEN Truls <truls.dahlsveen@soprasteria.com>  
**Sent:** Tuesday, May 16, 2023 4:10 PM  
**To:** Jardar Hollås <jardarh@stud.ntnu.no>  
**Cc:** FAUSKRUD Joakim <joakim.fauskrud@soprasteria.com>; PETERSEN Kristoffer <kristoffer.pettersen@soprasteria.com>  
**Subject:** Re: Evaluering av bacheloroppgave

Hei. Her kommer svar på krav fra alle tre:

Generelt sett har dere vært ganske strenge, og at kravene kanskje er litt for enkle (det kan vi ta kritikk for). Løsningen oppfyller slik vi ser generelt alle krav og selv om det er mye forbedringspotensialer føler vi det kommer utenfor de definerte kravene.

F1 - Basert på tabell 4.1 mener vi denne er 100% oppfylt. Det var ikke et av kravene at løsningen skal ta høyde for fremtidige oppdateringer, eller støtte mer infrastruktur enn nødvendig. Kravet er også bare at løsningen skal sette opp infrastruktur, ikke håndtere sletting og nyoppsett.

F2 - Kravet var at vi skulle kunne kjøre tester autonomt inne i deteksjonslabben. Testen må startes manuelt, men det er ikke hensiktsmessig å kjøre det automatisk slik det er tiltenkt. Det kunne vært mer brukervennlige måter å hente ut rapporten på, f.eks at den returneres i pipeline, men bortsett fra det leser vi kravet som oppfylt til 95%.

F3 - Brukere kan slik som systemet er satt opp i dag starte tester som kjøres. Dette kravet anser vi som oppfylt 100%.

NF1 - 100%

NF2 - 100%

NF3 - 100%, vi sa også at vi ville ha så kort og konsis dokumentasjon som mulig og dette er oppfylt på en god måte.

NF4 - 100%

NF5 - 100%

NF6 - 100%

NF7 - 100%, diskuterte alle løsninger vi foreslo og flere.

"Recondite" er vel eneste vi stusset litt på av ordbruk, ironisk nok 😊

- Joakim, Kristoffer og Truls

---

**From:** Jardar Hollås <[jardarh@stud.ntnu.no](mailto:jardarh@stud.ntnu.no)>  
**Sent:** Friday, May 12, 2023 10:35  
**To:** THORSTAD DAHLSVEEN Truls <[truls.dahlsveen@soprasteria.com](mailto:truls.dahlsveen@soprasteria.com)>  
**Subject:** FW: Evaluering av bacheloroppgave

---

**From:** Jardar Hollås  
**Sent:** Thursday, May 11, 2023 4:55 PM

To: Petter Jørgensen <[pettejor@stud.ntnu.no](mailto:pettejor@stud.ntnu.no)>

Subject: Evaluering av bacheloroppgave

Hei

Ref møte torsdag 11.05. Vi ønsker gjerne å motta noe evaluering av hvor godt vi har oppfylt kravene som er satt for prosjektet.

Om dere går gjennom hvert krav og vurderer individuelt, eller om dere skriver en generell helhetlig vurdering er opp til dere. Vi er fornøyde med alt vi får som kan gjøre vår vurdering litt mer objektiv :)

Håper at [Wikien](#) vår kan være til god hjelp i hvordan løsningen vår kan brukes. Det er også beskrevet et Test-Case i første del av [kapittel 8](#) hvor vi går gjennom et eksempelbruk av løsningen.

Dette er kravene som de er definert i kapittel 4 av [oppgaven vår](#). Merk at 'functional requirements' er det som er viktigst å få tilbakemelding på, da de fleste 'non-functional requirements' er relativt simple, og kan bare være enten er 100% eller 0%. F.eks:

Krav	Confidence Level	Beskrivelse
NF5	100%	Alle Pipelines er i sin helhet lokalisert i Azure Devops

No.	Functional Requirement Description
F1	<b>Automatic Deployment of Infrastructure via Azure DevOps Pipelines</b>
	One of the main requirements of the project was the automatic deployment of infrastructure using Azure DevOps pipelines. This implies the creation of VMs and the installation and configuration of necessary software should be performed using Azure DevOps pipelines.
F2	<b>The system should be able to autonomously perform security tests within the detection lab</b>
	The solution should be able to perform security tests within Sopra Steria's detection lab independently. Meaning that when configured, the solution has to be able to run security tests against the project owner's selected targets without any manual interference by the user.
F3	<b>Users should be able to define the tests or adversary profiles that the system executes</b>
	When completed, the users of the solution, the security testers, should be able to define and select what tests they want to execute. Furthermore, they will need to have the ability to select what adversary profile the system should execute.

No.	Non-functional Requirement Description
NF1	<b>Open-Source solution; non-copyrighted and customizable</b>
	The solution needs to be open-source in order to make sure that it can easily be modified, adapted and shared by its users.
NF2	<b>Documentation for security tests should be mapped to MITRE ATT&amp;CK</b>
	The security tests should be mapped to the well-known MITRE ATT&CK cybersecurity framework to make sure that test results are easy to share, understand and compare
NF3	<b>Documentation for installation</b>
	A short and concise description explaining how to use the solution. Needs to be written in the Azure DevOps Wiki
NF4	<b>The code for the solution must be stored in an Azure DevOps repository made for the purpose</b>
	Code and other related files used for the project need to be stored in the projects Azure DevOps repository
NF5	<b>The Pipeline must be hosted in Azure DevOps Pipelines</b>

	The Pipelines, which are responsible for executing tests, deploying, and configuring infrastructure, have to be hosted and created using the Pipelines module of Azure DevOps.
<b>NF6</b>	<b>The Infrastructure for the solution must be deployed to the Sopra Steria Azure Cloud Environment</b>
	When the solutions infrastructure is deployed, it has to be deployed in Sopra Steria's own Azure Cloud development environment.
<b>NF7</b>	<b>Evaluation of frameworks for automatic cybersecurity testing</b>
	The most compatible cybersecurity framework should be used for developing the solution. Therefore it is important that an evaluation of different frameworks takes place.

I vår egen evaluering gir vi en prosent-rangering til hvert krav iht. følgende definisjon, med en kort begrunnelse.

**Table 8.1: Confidence Level (Definition)**

<b>Confidence Level</b>	
100%	We are confident that the requirement has been met completely
75%	The main parts of the requirement have been met, but there is room for improvement
50%	Some of the requirement has been fulfilled, but there is a lot of room for improvement
25%	There is still a lot to do in order to fulfill the requirement
0%	No progression has been made to meet the requirement

Mvh  
Jardar Hollås

## **Appendix D**

# **All Meeting Notes**

Attached is all of our informal meeting notes and drafts, including from internal work sessions, weekly supervisor meetings and bi-weekly status meetings with our client. Most of them in Norwegian.

# Bi-weekly status meeting with client

07.12.22

Første møte.

Om oppdragsgiverne:

Truls Dahlsveen: Sec-engineering, driver med sikkerhetsovervåkning for mange kunder. Ruller ut deteksjon og sørger for at kundene er trygge. De sørger for å konfigurere at ting kan pushes ut automatisk. Designer og pusher ut løsninger - i tillegg et annet team som driver med sikkerhetspatching.

Kristoffer Pettersen: automatisering osv, fikk ikke med meg

Joakim Fauskrud: Jobber i socken, engineering, har jobbet med Truls og Kristoffer og Mikael. Konfigurerer og automatiserer i det daglige. Gikk på Gjøvik før.

Mikael Vagnes: Jobber som security engineer, ganske ny, automatisering. MS Cloud erfaring. Jobbet med Azure, migrering av ressurser og M365 og litt overalt.

---

Om oppgaven:

Utrulling via azure devops, eller github actions. +De sitter i en sock, bruker Microsoft Sentinel.

Ønsker å koble inn datakilder inn i "Data connectors i Microsoft Sentinel", ønsker å gjøre søk på loggene de tar inn. Søker på Indicators of Compromise IoC'er. Ønsker å gjøre deteksjon på det her. For emulering av trusselaktører har de ikke verifiseringsverktøy eller alarmer for, bacheloroppgaven skal: sette opp et miljø, sårbart eller et som ligner på et kunders miljø, så kjør automatiske tester mot den og se om det plukkes opp. Må lage deteksjoner som oppdager problemene. Flyten skal være: man har en pipeline, en yaml fil, når den kjører så kjører den tasks, skal kunne bygge inn et bibliotek av tasks, så kan man kjøre de sekvensielt i en pipeline, så skal det være lett å skrive nye tester og pakke inn i et repository som kjører mot det man vil.

Kunne kjøre tester mot et miljø, og plugge inn nye tester fort. Og dokumentasjon. Noe tverrfaglig, pipelines, devops, automatisering, de har skyplattform i Azure som vi kan låne for utvikling. Så er det Azure Devops eller AWS eller Github Actions.

Use Case Senter.

Alt av analyseregler ligger som kode. Deteksjonsregler, så har de pipelines som genererer dokumentasjon for hva den gjør. Lager fila så blir dokumentasjon automatisk laget. Blir mappet opp mot MITRE taktikker osv.

Har en development tenant som vi kan legges til i.

Kan kjøre på egne servere. Kan evt kjøre mot deteksjonslaben deres. Kan velge ferdige rammeverk, så kan man lage egne tester til de rammeverkene: atomic red team, MITRE Caldera.

Viktigst at vi får noen VM'er, start et miljø, se på rammeverk.  
De kan sette opp noen VM'er vi kan få tilgang til.

Om vi setter opp en Azure Organisasjon som er privat, kan vi få en Service Connection/principal, så har vi tilganger til å pushe kode til vm'er osv. Vanligvis vil teste mot Windows 10 og 11. Kan velge å kjøre Terraform via azure subscription som så kjører noen steg (kode/tester?), så kan vi prøve å kjøre kommandoer fra azure direkte mot vm'ene, eller fra en vm mot en annen vm.

Vi skal teste deres tjenester,

**academy.attackIQ.com** purple teaming learning path.

Socken er blått team som forsvarer (oppdragsgiverne), så skal vi (bachelorgruppa) være red team som skal angripe, så skal vi samarbeide som purple for å se hva de plukker opp, så skal det forbedres, så skal vi prøve igjen senere og se om det ble bedre. Vi, det røde teamet, skal kjøre tester som kjører ting kontinuerlig, i stedet for å kjøre en og en.

Vi skal drive med "foundations of breach and attack simulations". Plugg inn verktøyet mot et image til et firma (f.eks. Standard windows 10 maskin), og se hva man plukker opp og ikke. Kan implementere ganske fort, men å lage flere tester er viktig for at verktøyet skal bli bra. Vi velger litt hvor vi legger trykket.

Se på de som går på MITRE attack og MITRE attack simulation, slik at vi vet hvordan vi kan bruke disse til proaktivt sikkerhetsarbeid.

Beste tips om BA oppgave: hvis man skriver veldig mye, prøv å vær så nærme antall ord som mulig. Jo bedre man kan det, jo bedre kan man forklare det kortfattet.

Kan vise hvordan det fungerer under presentasjon. Dokumentasjon skal kokes ned til: "hvordan reproducerer vi dette?".

De kan lage eget prosjekt i devops hvor vi blir gjester, så er det scopet til kun et prosjekt, så kan vi invitere de inn som brukere i dev-tenanten også, så kan vi få en egen ressursgruppe der og.

<https://azure.microsoft.com/en-us/free/students/>

Helst skriv oppgaven på engelsk.

Anbefaler bruk av chatgpt og copilot.

Foreløpig plan: sette seg inn i AttackIQ kurs, ordne tilgang til azure og vm'er (kanskje free trial), finn ut og velg rammeverk.

# Weekly supervisor meeting with supervisor

10.01.23

## To-do list:

- Prosjektavtale. Mal på BB. Signert mellom oss og Sopra. Bare 1. Last opp til team-mappe. Frist 31. Januar
- Forbered en prosjektplan: Outline over hvordan vi skal gjøre ting, så får vi en template senere. Får tidligere prosjektplaner fra andre. Skal approves kun av supervisor Kelly. Last opp til team-mappe. Frist 31. Januar
- Første utkast av bachelor-thesis/prosjektrapport. Skal ha informasjon klart i hvert kapittel. Last opp til team-mappe. In English. Frist 31. mars
- Bacheloroppgave. Frist 12:00 lunsjtid 20. mai. Lastes opp på BB. Ikke last opp siste dag.
- 7-9. Juni, presentasjon: Bachelorpresentasjon på engelsk. Presenter resultat, metodikk osv.
- Avtale møte med oppdragsgiver. Diskuter arbeidsavtale og utforme problemstilling

## To know list:

- Vi må lede og jobbe på prosjektet.
- Aktivt hold kontakt med oppdragsgiver og etabler god kommunikasjon med dem
- MS Teams som kommunikasjonskanal mellom oss og Kelly. Skru på alle notifications/varsler
- 30-minutters ukentlig møte.
- E-mail til Kelly med ukentlig status-report dagen før:
  - Week of writing the report
  - What you have done in the previous week?
  - Do you encounter any problems? How did you solve them?
  - What do you want to discuss in this meeting?
  - What are you going to do in this week?
  - Is your group still on track according to your Gantt chart?
- Tildel roller, fordel arbeidsmengde mellom alle cirka likt.
- Begynn skriving så snart som mulig. Vi får karakter fra to sensorer, en intern og en ekstern. Kelly har ingen påvirkning på dette. Vi må jobbe "hardere" for å få bedre enn C, og bevise at vi har gjort mye research. Av Kellys grupper, har: 1 fått A, mange fått B, 1 fått C, og ingen dårligere - som er et godt tegn.
- Alltid sjekk BB
- Tidligere oppgaver: <https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/227496/browse?type=type&value=Bachelor+thesis>

Karakterer gis basert på "general impression". Ingeniør/profesjonell innsikt, teoretisk innsikt, implementering/prosess



# Weekly supervisor meeting with supervisor

17.01.23

Meeting time (after supervision meeting)

Evaluatoin criteria

- functional requirements
- non-functional requirements

--- Petter:

Bra at vi gjør kurs. Kelly støtter.

Første hovedkrav: Sett opp møter med Sopra.

Brukte en del tid på å forklare oppgaven til henne.

Spør Sopra hvordan de synes vi burde evaluere oss selv, hva er et velykket angrep eller ei? Hvor fort. Hvor mye. Prøv å tenke hva som kan brukes senere for å evaluere systemet.

Separer kravene inn i kategorier. Effektivitet, automatisering,

Hvor mange oppgaver er der? Dokumenter disse tingene. Spesielt i gantt-chartet, spør oppdragsgiver hvor lang tid hver del tar. Er estimatene realistiske, burde vi endre rekkefølge eller prioritet? Mest teknisk hjelp fra oppdragsgiver. Gantt-skjema til Kelly.

Hvor mye tid vi bruker på hver del må være realistisk.

Finn ut så mye som mulig på møtet med dem.

— Tryggvi:

- What kind of attacks should we implement? Who are their customers? What kind of threat agents go against them.

- How are we going to evaluate the performance? Automated reporting? How do we get the detection reporting and misses in one document.

- Issues with how realistic the outcomes of a lab would be? Would not be the same as in production.

- Is the goal to test different detection solutions?

- What is their criteria for detection. (How long should a tool take to detect an attack?)

- Documentation might be very important to them since they might have to maintain the project after we finish.

- What is the scope of their project?

- Risks?

- What they want us to test? (network)

- Do they want us to emulate, simulate e.g.?

- What kind of attacks do they want us to focus on.

Find what functional/non-functional requirements that we can use to measure success.

- Functional requirements?

- Non-functional requirements?

  - E.g. efficiency

- We should create a Gantt chart and base it on how we perceive it to be.

# Bi-weekly status meeting with client

19.01.23

Spørsmål til oppdragsgiver:

- Hva begynner vi med?
  - Tilgang til testingmiljø, kontoer osv.  
**Send e-postene vi vil bruke til innlogging, inn i Azure tenant hvor vi får rettigheter.**
  - Har de en test vi kan ta utgangspunkt i, for å fokusere på å få i gang testingen?
  
- Evalueringskriterier. Hvordan vurderer vi om testing/arbeidet vårt ble vellykket eller ei?  
**Kommer an på. Spesifikt på endepunkter/servere er enklest. Baser seg på MITRE. Ta i rekkefølge, få tilgang, execute osv. Assumed breach, start inne i nettverket på en VM (antar at trusselaktøren har fotfeste inne). Hva kan man få til med en vanlig bruker? Får i hovedsak 1 VM som vi skal teste mot. Så skal det (kanskje) testes mot deres deteksjonslab. Hovedsakelig anta at vi starter på et kompromittert endepunkt. Sentinel 1(?) eller Defender endpoint.**  
  
**Vi burde kanskje ta i bruk kontroll endepunkt som ikke har så mye sikkerhetsprogramvare på seg. Gjør det lettere å måle hvor effektiv testingen vi gjør er.**
  
- Hva tester vi mot egentlig? Kan de demonstrere testing som ligner på det vi skal gjøre?  
**Får intro med tilgang / etterpå?**
  
- Kan de dele opp oppgaven i noen hoved-deler, som de ser det?  
(Tenk; planlegging, oppsett av infrastruktur, design av tester, testing, evaluering/rapportskrivning)
  
- Hvor lang tid cirka tror de hver del av arbeidet tar

**Opp til oss. Kan begrense oss til spesifikke MITRE sub-teknikker, eller gå for en trusselaktør (en APT) og at vi skal emulere deres killchain. 7-8 tester f.eks.**

- Kontrakt
  - Punkt 5: bruksrett eller opphavsrett (**bruksrett**)
  - Punkt 8: oppgavens offentlighet (**offentlig**)
  - Eventuell taushetserklæring? (**ingen taushetserklæring**)
  
- (Bi)-ukentlige møter? Når? Frekvens? Hvor mye tid har de disponibel  
**Greit med bi-weekly, litt opp til oss.**

**En egen resource group som heter “bachelor-project”, er en egen service principal som kan brukes som har tilgang inn. I tillegg er det en gruppe som har tilgang og kan gjøre ting i ressursgruppa. Cost cap på ressurser på 2.5k i mnd. Scheduled stop som stopper 18:30, må si i fra om vi trenger det etter dette. Blir lagt bak en vpn. Veldig enkel windows vm som vi kan kjøre tester lokalt på. Og en azure devops som heter sopra steria soc dev, som vi kan lage pipelines og repo i og det vi vil. Og en service connection, en service principal i azure som gjør at vi kan kjøre/lage starter pipeline. Kjøre i context av service tenanten. Kan kjøre kommandoer på azure vm'er via remote. Skal ha tilgang til å sette opp VM'er. App som heter Azure VPN som vi laster ned og bruker. Send e-poster, så får vi teste om det funker.**

# Weekly supervisor meeting with supervisor

24.01.23

Introduction, talked about status report, showed off the parts of the project plan. Then gantt chart - she asked about phase 2 (why/what it was), we explained.

Kelly told us to start writing on the draft at least from the start of march. We will use the Gantt chart to assess our progress. Kelly asked about requirements; we should assess what requirements are suggested by us or the client, and decide what is **required** and what is optional.

- Separate into functional and non-functional requirements. Make a clear separation between core requirements and optional requirements. Bullet points
- She will sign the “standardavtale”, she got a link

Discussed further plans: risk analysis is fine, though maybe turn the risk table into a list/text format.

**Technical plan is important.** System plan with architecture, show the entire process like a flow chart for each component with each feature. What does it do? Take a look at 6.1 in the other group’s project plan. And figure 4.1, a user and how the user uses the system. How one component goes to another one. Make a flowchart/use case.

# Weekly supervisor meeting with supervisor

31.01.23

Project plan looks fine. **Approved by Kelly.** Only needed some context to the figures.

Write down what to test. Get some concrete ideas after the meeting with the client. Detail print, and present it to her.

Maybe come up with plans or things to do before the meeting with Sopra, to ask for suggestions or thoughts about it from sopra.

*Ble et veldig kort møte. Hun skal se over prosjektplanen og gi kommentarer.*

# Bi-weekly status meeting with client

02.02.23

- **Pipelines til bruk av automasjon / deployering?**
  - TERRAFORM: sett opp infra / ansible: ? / customscriptextension**
  - **set-azurevmcustomscriptextension**
- **Hvordan få rdp-tilgang til vm'ene?**  
**VM login info: badmin / Administrator123**
  
- **Showcase av en enkel test, fra deres side?**  
Forhold oss til

## Notater:

Én windows server, en domain controller, mange windows klienter og et par linux servere. VM'er i Azure, Defender i MS Defender, SentinelOne i M365 Defender. Sentinel -> Data connectors -> SentinelOne. To måter å hente security data, Azure Monitoring Agent og MS Defender. SentinelOne i Linux.

Vi skal kjøre tester som atomic red team fra en eller annen klient, f.eks. dl-w10..N. MS Sentinel -> Analytics, filtrer etter windows security events. Marker de og lag regel. Dev-tenant = sopra-steria-soc-dev.

Last ned RDP-fil, logg inn på "badmin" (?). får innlogg-info på teams?  
Bachelor project resource groupen. For å ha en side man sender data til setter man opp Log Analytics workspace [Home > Resource groups > rs-bachelor-project > Marketplace > Log Analytics Workspace >](#)

Gir oss et sted å sende logger til. MS Sentinel oppå dette igjen. Create data collection rules. Installerer en agent som skal hente logger. Extensions and applications. Defender for endpoints, **defender for cloud**, bør få recommendations derfra, settings -> defender plans.

**For oss:** I første omgang er det 1 VM. Den er kobla til Sentinel -> Logs. Der renner det inn i en egen tabell kalt WindowsEvent elns, så kan vi gå inn i data connectors og "connected".

Sentinel og defender for cloud kan kobles sammen.

Begynn med oppsett av VM'en, kjør tester,

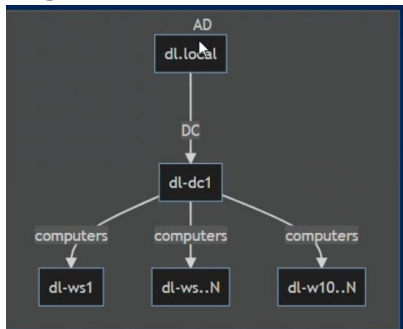
**VM login info:** badmin / Administrator123

Ansible, scriptextension i Azure,  
Bachelor project -> pipelines -> azure-pipelines.yml. Se høyre side "tasks".

De har laget to pipelines, en som lager og en som ødelegger detection laben.

Validerer, planlegger og deployer hvis det går gjennom.

**DetectionLab.yaml lager dette oppsettet, som er det som eksisterer i dag, altså dagens detectionlab som vi skal angripe/teste mot:**



Kan sette opp alt i azure i terraform. For å bruke ressurser i terraform må man referere til dem i .tf-koden. Har delt opp i moduler, en modul per kategori (DC, Linux, W10, WServ). Main.tf er den vi faktisk kjører (sier hvor vi henter mer kode fra), samler sammen ting.

I terraform, for å sette opp et miljø:

1. Setter opp nettverk, 2. setter opp VM, (masse in-depth greier som vi går gjennom senere).

Kjør et script i en fil, pek mot en vm, kjør via pipelines.

VM extensions, custom script extension azure. Kan kjøre script direkte på VM'en.

```
PowerShell
You can use the Set-AzVMCustomScriptExtension command to add the Custom Script Extension to an existing virtual machine. For more information, see Set-AzVMCustomScriptExtension.

Set-AzVMCustomScriptExtension -ResourceGroupName <resourceGroupName>
  -VMName <vmName>
  -Location myLocation
  -FileUri <fileUri>
  -Run 'myScript.ps1'
  -Name DemoScriptExtension
```

SPN appreg? Azure subscription er satt opp i pipeline som er der. Service principal, azure powershell, skal være i orden.



# Arbeidsøkt

03.02.23

Har fått RDP og SSH-tilgang til den ene vm'en. Fikk installert og kjørt Atomic Red Team (framework for enkel kjøring av tester som er mappet mot MITRE). Disse fungerte, og én av dem ble oppdaget og logget i Microsoft Defender for Cloud -> Security alerts i Azure. Resten kommer opp ved manuelt søk i loggene, under log monitoring, men gir ikke samme type varsel.

Kjørte cirka 5 tester, og kun den ene T1218.010 ga utslag under security alert. De andre er kanskje ikke kritiske nok, eller blir stoppet på et lavere nivå (lokalt av defender på vm'en, før den blir oppdaget av defender for cloud??)? Har heller ikke access til å "Take action" i azure.

# Arbeidsøkt

09.02.23

Alle har rdp-tilgang og får til å kjøre AtomicRedTeam-tester. Gjort research/testing, (vill klikking rundt omkring) i MS Sentinel og Defender for Cloud.

Brukte Atomic Red Team sin nettside, for å finne forskjellige tester som kunne bli kjørt:  
<https://atomicredteam.io/atomics/>

## Invoke-AtomicTest T1110.003 | Password Spraying

<https://atomicredteam.io/credential-access/T1110.003/>

For å se i logs:

- SecurityEvent | where CommandLine contains "T1110.003"

```
PS C:\Users\badmin> Invoke-AtomicTest T1110.003
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1110.003-1 Password Spray all Domain Users
The system cannot find the file C:\Users\badmin\AppData\Local\Temp\users.txt.
Done executing test: T1110.003-1 Password Spray all Domain Users
Executing test: T1110.003-2 Password Spray (DomainPasswordSpray)
IEX : At line:1 char:1
+ function Invoke-DomainPasswordSpray{
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
At line:2 char:1
```

## invoke-AtomicTest T1218.001

For å se i logs:

- SecurityEvent | where CommandLine contains "T1218.001"

## invoke-AtomicTest T1082 | System Information Discovery

Fikk ingen logs

- SecurityEvent | where CommandLine contains "T1547.010"

## Invoke-AtomicTest T1547.010 | Port Monitors

```
PS C:\Users\badmin> Invoke-AtomicTest T1547.010
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1547.010-1 Add Port Monitor persistence in Registry
ERROR: Access is denied.
Done executing test: T1547.010-1 Add Port Monitor persistence in Registry
PS C:\Users\badmin> █
```

## T1564.006 | Run Virtual Instance

Fikk ingen logs

- SecurityEvent | where CommandLine contains "T1564.006"

## T1204.002 | Malicious File

For å se i logs:

- SecurityEvent | where CommandLine contains "T1204.002"

CloseMe



Hi,

In case you executed this application without getting any alert, detection of PUA (Potentially Unwanted Applications) is not enabled.

For more security feature tests, please visit:  
<http://www.amtso.org/feature-settings-check.html>

Do you want to close this window?

Yes

No

Dokumentere Azure og Sentinel oppsett...

Microsoft Defender for Cloud: <https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-cloud-introduction>

Anti-Malware testfile EICAR was detected and stopped locally by MS Defender on the VM. It was detected as a security incident in Sentinel after ~20 minutes. Potentially unwanted program test was detected but failed to stop:

[https://portal.azure.com/#asset/Microsoft\\_Azure\\_Security\\_Insights/Incident/subscriptions/02d3f4e0-67c3-4fe4-a771-3cdc5685cca9/resourceGroups/rs-bachelor-project/providers/Microsoft.OperationInsights/workspaces/rs-bachelor-project-law/providers/Microsoft.SecurityInsights/Incidents/f076e336-3d06-4be6-8462-3a9a434ffbac](https://portal.azure.com/#asset/Microsoft_Azure_Security_Insights/Incident/subscriptions/02d3f4e0-67c3-4fe4-a771-3cdc5685cca9/resourceGroups/rs-bachelor-project/providers/Microsoft.OperationInsights/workspaces/rs-bachelor-project-law/providers/Microsoft.SecurityInsights/Incidents/f076e336-3d06-4be6-8462-3a9a434ffbac)

Near-Real-Time (NRT) regler (under Sentinel -> Analytics -> Create NRT) kan rapportere/varsle om incidents raskt, og hente dem ut fra loggene via en query.

# Arbeidsøkt

13.02.23

Undersøkt MITRE Caldera og fått til å kjøre dette lokalt. Forsøkte å få opp en web app via en container instance i Azure, men fikk ikke koblet til denne (hjelp), hadde ikke authorization til å starte en tilsvarende "Container App".

Det ble laget en "Research" mappe på Google docs. Startet med å finne linker og info om de forskjellige rammeverkene, og info om Azure. Må starte å tenke på dokumentasjon av disse rammeverkene, og fordeler/ulemper med dem. \o/

# Weekly supervisor meeting with supervisor

14.02.23

Kelly wants fewer meetings. We should only have meetings when we have something to show.

Need to decide on what we should do, define “work scope”.

- Tests against one VM or entire network.
- Framework decision and discussion, benefits/cons.

(Sopra Steria would like documentation about configuration and installation for the chosen framework)

# Bi-weekly status meeting with client

16.02.23

## Sentinel & MS Defender for Cloud

- Hvorfor samles ikke lignende alarmer inn i en incident i Sentinel?
- Forskjellen på de to (som de ser det); defender for cloud

Sentinel er opp til deg selv, du må lage alarmer selv og konfigurere det. Vi skal lage alarmer. Er ikke nødvendigvis en del av oppgaven å passe på at sentinel plukker opp alt (bestem en %, lag regler som plukker opp mer enn av standard). Vi skal vise at det fungerer fordi det plukkes opp(?).

Man kan legge til flere tester, se om de plukkes opp eller ikke, så kan vi endre reglene basert på det. Man skal enkelt kunne legge til flere tester osv. Forbedre deteksjon; kjør en del tester for å se hva man har deteksjon på og ikke, en del av det er å automatisk kjøre tester (flere og flere), så er det opp til oss hvilken del av økosystemet vi fokuserer på.

## Hub-and-spoke arkitektur, nettverket

- Application Security Group - legge den til begge enhetene så skal de kunne snakke med hverandre.

Har dere noen krav og spesifikasjoner når det kommer til hvor lang tid det skal gå før ondsinnet aktivitet skal kunne plukkes opp av deteksjonssystem?

Hvilken type tester bør vi kjøre?

MITRE Caldera, spør om deres forståelse av hvordan vi kan utnytte det?

Kjør Caldera på en container med public ip som er reachable fra utsiden (egen PC) men får connection refused på virtual machine som har privat ip. Får "no access" når vi prøver å endre/se på subnet som vm'en er på.

Sentinel bi-directional sync med Defender for Cloud. Er dette nødvendig eller ikke?

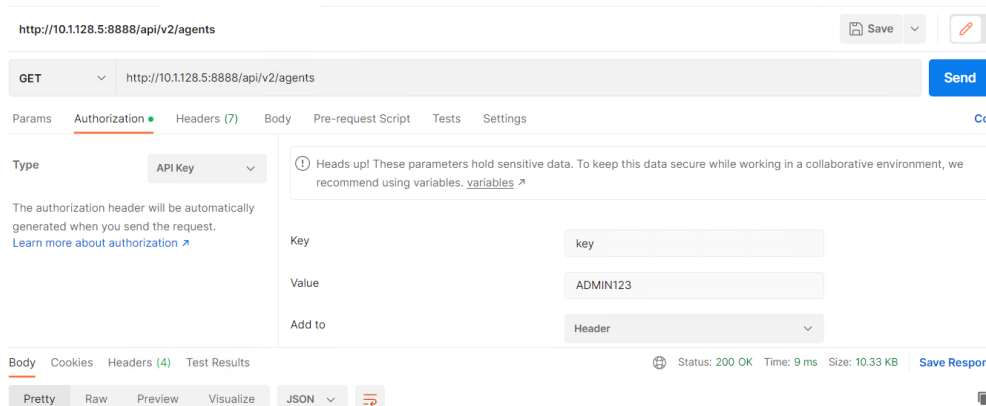
Informasjonsflyt? Defender for Cloud Agent -> Defender for Cloud -> Sentinel?

# Arbeidsøkt

22.02.23

Ting som må/kan automatiseres:

- Deployering av container med kjørende Caldera, og deployering av Caldera Agent på en valgt/ny VM, med rett IP til containeren
- Agent må ha [exception i Windows Defender](#) (splunkd.exe, evt. Mappa der den ligger C:\Users\Public). Gjøres manuelt under Virus & threat protection -> Virus & threat protection settings section -> Manage settings -> Scroll til Exclusions -> Add or remove exclusions
- Agent må kjøres på nytt ved restart av VM, eller legges til i on-startup
- Eksempel API-call, fra postman på VM til Caldera-container som henter aktive agents:



**GET** -> `http://10.1.128.5:8888/api/v2/agents`

Authorization -> API Key.

**key:** key

**Value:** ADMIN123

La body være tom. Bør få respons

- Defender på maskinen oppdager noen ting som kjøres pga en operation trigget av Caldera, selv om vi har gitt en exception for agenten

For rapporten:

- Dokumentere ulike rammeverk, og vurdere hvilken som burde bli brukt (Caldera?)
  - Vet ikke hvor god kilde men fant en liten liste med open source rammeverk:
    - <https://fourcore.io/blogs/top-10-open-source-adversary-emulation-tools>
    - <https://pentestit.com/adversary-emulation-tools-list/>
    - <https://www.csoonline.com/article/3268545/4-open-source-mitre-attandck-test-tools-compared.html>
    - <https://cybersecuritynews.com/cyber-attack-simulation-tools/>
    - <https://securitygladiators.com/security/software/best-cyber-attack-simulation-tools/>
- Vurdere:
  - Caldera

- Atomic Red Team
- Metasploit
- Metta?
- Red team Automation?
- Infection Monkey?

Krav var helst at de testene vi bruker innenfor rammeverket skal bli knyttet opp mot MITRE ATT&CK. Sikkert greit med et rammeverk hvor MITRE ATT&CK er integrert.

Caldera:

<https://caldera.mitre.org/>

<https://github.com/mitre/caldera>

<https://www.mitre.org/ourimpact/intellectual-property/caldera>

- Designed to easily automate adversary emulation, assist manual red-teams, and automate incident response.
  - Autonomous Adversary Emulation
  - Autonomous Incident Response
  - Manual Red-Team Engagements
- CALDERA leverages the ATT&CK model to identify and replicate adversary behaviors as if a real intrusion is occurring.

Atomic Red Team:

<https://atomicredteam.io/atomic-red-team/>

<https://github.com/redcanaryco/atomic-red-team>

- Open-source library of tests that security teams can use to simulate adversarial activity in their environments.
- Execute atomic tests directly from the command line.

Metasploit:

<https://www.metasploit.com/>

- “Metasploit helps security teams do more than just verify vulnerabilities, manage security assessments, and improve security awareness; it empowers and arms defenders to always stay one step (or two) ahead of the game.”

Metta:

<https://github.com/uber-common/metta>

<https://medium.com/uber-security-privacy/uber-security-metta-open-source-a8a49613b4a>

- Information security preparedness tool
- Uses Redis/Celery, python, and vagrant with virtualbox to do adversarial simulation.

Red Team Automation:

<https://github.com/endgameinc/RTA>

- Framework of scripts designed to allow blue teams to test their detection capabilities against malicious tradecraft, modeled after MITRE ATT&CK.

Infection Monkey:



<https://github.com/guardicore/monkey>

- The Infection Monkey is comprised of two parts:
  - Monkey - A tool which infects other machines and propagates to them.
  - Monkey Island - A dedicated server to control and visualize the Infection Monkey's progress inside the data center.

# Weekly supervisor meeting with supervisor

28.02.23

Skrivekurs 6. mars 10:00-11:00 med Kelly for alle hun er supervisor for.

Skal produktet vårt være fullstendig automatisert? Command-line basert? Skal brukeren kunne velge tester? En samling med tester?

Testene skal sannsynligvis bli kjørt sekvensielt.

Sjekk google scholar og finn litteratur om rammeverk for sikkerhetstesting (for skrivingen). Hvilket format og hvilken informasjon ønsker de (brukerne) å få ut av programmet vårt?

- Finn lignende rammeverk, sammenlign med Caldera. Nevn disse verktøyene og diskuter rundt dem.

Lage en enkel visualisering om nettverks-oppbygningen.

- Vise den til Kelly
- Kan ha den med i rapporten for å forklare oppgaven.

# Bi-weekly status meeting with client

02.03.23

App registration azure (legg inn i devops, så får pipeline tilgang).

Azure script extension for å kjøre powershell scripts. Når man kjører azure powershell module, sign in,

# Arbeidsøkt

08.03.23

Group meeting in Cisco-lab. Discussed and started writing an outline for the first draft. Inspected other Bachelor's theses for inspiration, and used supervisor's notes. Filled in keywords for each chapter, and started writing drafts for chapter 1 and 2. Research on how long each part should be, and planning the writing for the coming days/weeks.

Potential expansions of the program, or limitations, based on how long we get:

- Including metasploit framework testing. Preferably through Caldera
- Including custom scripts/ps1 files, maybe after deployment, like how we deploy the agents

# Arbeidsøkt

16.03.23

Continued writing on the first draft. Have a decent draft for the structure, and for most sub-sections under section 1. Discussed API usage,

# Weekly supervisor meeting with supervisor

21.03.23

## Spørsmål til Kelly

- **Hva skal være med i en “first draft”?**
- **Har hun krav til den?**
- **Må den godkjennes?**
- **Hvor lang bør den være?**

Ingen veldig spesifikke krav. Gjør det beste vi kan, jo mer vi leverer - jo mer kommentarer/tilbakemelding får vi

## Code Listing

- Should have source code that we are using for the project. Might be better to just refer to the Git repo here. It is OK to have the code listing chapter at its current place, and should NOT be placed at the end of the thesis.

## Background

- What should be put in this chapter? How much detail should be included here about the project. *Short introduction of 1-2 lines of Cloud computing. Explain how Azure relates to the project.*

Write the bachelor with the assumption that the readers have a technical background. We do not have to explain every basic concept.

Glossary should be right before introduction.

# Bi-weekly status meeting with client

22.03.23

Forrige ble utsatt pga. Sykdom.

- Spør om hvordan vi kan få tilgang til webservern (caldera gui) fra utsiden av nettverket.  
Er det vpn'en sin skyld?

Vi skal vise fram fullstendig automatisk deployering, så får de komme med input.

Også fått på plass Metasploit.

- Separat pipeline for testing, slik at vi kan deployere 1 gang, men kjøre testene flere ganger.
- Selv-lagde tester er utenfor "scope", holder oss til de som allerede er inkludert i caldera + (metasploit)

# Weekly supervisor meeting with supervisor

28.03.23

## Questions for Kelly about first draft:

- How should figures be numbered? After the exact section they're in, or just the chapter?
    - The way we are doing it now is OK.
  - Where do we get the standard frontpage?
    - New frontpage will be appended to our bachelor project when we upload.
  - How should illustrations look? Are colors allowed?
    - Important that figures are not decoration, they have to serve a purpose.
  - Should we use footers, for urls etc.? Frode said it would be a good idea, I think..
    - No more than 3 URLs in footnotes.
  - Should table captions be centered above the table?
    - Correct to have captions for tables above the table. For figures they should be under the figure.
  - Is the current thesis name too general? Should it be more specific?
    - Consider changing title
- 
- Glossary should have one descriptive sentence of the term, not just what it stands for.

## Extra notes:

- No supervisor meeting next week, easter break.
- We could send the first draft the 10th of April, since Kelly won't have the opportunity to review it before then.



# Bi-Weekly Meeting with client

13.04.23

## **Quick status update with Sopra Steria:**

1. Update to what we have been doing since the last meeting and what we are going to do forward.
  - Scripts that setup infrastructure are working, but tests are not currently running.
2. We were offered help on project writing by Sopra Steria.

# Arbeidsøkt

18.04.23

Endre tittel til å omfatte mer hva vi gjør (MITRE ATTACK / Caldera)

Introduction. Ellers lite.

# Weekly supervisor meeting with supervisor

25.04.23

Related work 3.1 is good. In 3.2 just introduce it with their last name.

# Bi-weekly status meeting with client

27/28.04.23

Prosjektoppgaver som gjenstår

- Hente ut rapport
- Log analytics?
- Destruct Pipeline
- Generell refinement
- Caldera image for øyeblikket statisk
- Mulig implementasjon av Metasploit med flere tester..?
  
- Fokus på skrijving fremover. Kan ta med et kapittel hvor vi skriver hva som skal gjøres videre. "Videre arbeid".
- Presentasjon av alt vi har gjort, sånn at de kan bruke det. Ta det etter vi er ferdige med skrijving.
- Skriv én liten wiki-side i repoet som forklarer hvordan det fungerer

Foreslått gjennomgang i morgen. Lavterskel. Opp mot en time gjennomgang.

Kommentarer: Ser bra ut. Hold wikien kort, skal helst ikke være mye. Fokuser på skrijving og gjør ferdig oppgaven.

Dele tilgang til bacheloroppgave med Sopra Steria? Overleaf lenke

<https://www.overleaf.com/read/mphnpyggnfvv>

^leselenke.

# Weekly supervisor meeting with supervisor

02.05.23

What goes into chapter 5 vs. what goes into chapter 7. We now have a system architecture illustration.

Kelly: implementation should be; how we did the solution, how the script works, how the versions/software works, how we do this automation. We can mention it in chapter 5 and explain it there, but in chapter 7 we explain **how** we implement/use it.

In chp 5, can we visualize: what are the components inside this and this part, inside e.g. the pipeline. Show how the pipeline looks. We need a specific/clear name to describe what "infrastructure" means. The purpose should be embedded into the name, security testing environment deployment pipeline. S.T.E.D.P. use an abbreviation or other name that accurately describes what it does.

Change the illustration to make it clear what order things run in. We need a better name for the container instance, maybe attacker.

Say in the implementation chapter, because of this and that we decided to do it into a single virtual machine - describe why it is like it is? Why is the attacker on a container? Can a vm be exchanged for another one? Give a number to each arrow in the figure. Keep the user and pc close.

Spent some time explaining our solution to Kelly, it was confusing for her.

Implementation details, and how we did this and that should be in chp 7. Features and logical design/functions, and overview of architecture should be in chp 5.

# Weekly supervisor meeting with supervisor

09.05.23

**Notes:** Include reference that explains declarative syntax? (Terraform chapt 2, 7)

Make the code in the thesis smaller? Keep our focus on actual writing. We cannot have too much code, there is no page limit. But we need an explanation of what it is for, why it is there. Describe from the user's perspective. The description in chapter 5 maybe has to be more clear? We need a leading paragraph before all the figures in chp 5 and expand what is going on, and explain for each step what the actions mean. Add a leading sentence before the action descriptions, that summarizes..- Move the actions above the figure, same for all figures.

She does not know if our evaluation is good enough. We have a confidence level, yes, but only from our perspective. Maybe get input from our client? Confidence level from ourselves, and from our client. Move the summarizing table of evaluation to the end of chp 8. Our own evaluation vs. our client's evaluation.

Mention all the requirements to the client, then they can provide an overall opinion/feedback on our tool. And we can have another section in chp 8 about that. Then specify that the first part is our own evaluation.

Subsection at the end of chp 8 with "evaluation from our client".

Do a live presentation of our solution in our presentation. We firstly describe the architecture, then each component, then the solution presentation, then the evaluation/confidence level tables etc.

Further work: list up all possibilities in bullet points, choose and suggest a few of them directed towards our client. Discuss them. Do not need to write a lot in closing remarks. 3 pages maybe.

For the implementation we can show the deep details, the corresponding interface (show mitre caldera gui!), show the report

**Test case** inside evaluation. Detailed showcase of our tool and how it can be used, including report generated, tests used, etc. Then detailed description of the requirements, maybe reference back to the test case. Discuss how the test case can be done, then do it, then show it (results).

Consider comparing to similar tools, not necessarily using just Caldera.

Keep the related work as one section, no title for each related work - but we can categorize into different sections with headlines if they are different enough.

Author's last name + "et al", and a paragraph describing an approach from a related work.

# Bi-weekly status meeting with client

11.05.23

Avtalte å sende en e-post med spørsmål om requirements, slik at de kan fylle ut, og vi kan bruke det i teksten. Ellers lite. Ser bra ut.

# Weekly supervisor meeting with supervisor

16.05.23

1. General status. **We did good.**
2. Scroll through the test case and get feedback
3. Do we have enough in chp. 5?
4. Check new entry in related work

**15 min presentasjon, 10 min spørsmål?**

**20 min presentasjon, QA 5 min?**

**Ble evaluation requirement om Wiki skrevet ferdig? I**

Check for capitalization errors, Chapter should be capitalized.

Should a reference for a figure have them same? For example -> Figure 3.1?

Table names chapter 8:

Evaluation Results (Non-functional Requirements) -> Evaluation Results for Non-functional Requirements

Combine chapter 4 & 5

First section about requirements, then show architecture, then show the use cases.

Keep more or less the same size for each chapter. Combine and be flexible.

Remove subsection title when we only discuss one work there.

Add citation for article

Remove bulletpoints in avoding ms defender part

Final look before submitting with Kell 19.May (friday) -> send a message reminding of this



# First draft feedback from supervisor

The draft is quite good, it can be better.

*Text marked in red is improvements that have been made already.*

*Text marked in green is general feedback that is not important to change anything about.*

1. The title can be better, too generic title. Add what the important features are. Add who the targets for the testing are.
2. Glossary is very good.
3. Introduction with leading paragraph is good.
4. OT != Operational Security
5. Sopra Steria's SOC Team -> Sopra Steria. The SOC team of Sopra Steria. More formal to type it like that. *but background info is good, about sopra etc.*
6. Multiple test clients in a network, providing -> Multiple test clients in a network and provide
7. Streamline Sopra Steria's security testing process, and it will be\*\*
8. It is designed for employees -> The solution is designed for employees
9. The solution is designed for employees who have the technical background and who are responsible for security testing.
10. We can't ensure that the users will have an understanding... -> less strong language here
11. Include a collection tests of tests -> provide a collection of tests.
12. automation using Azure DevOps pipelines, which is (explain what Azure DevOps pipelines are, since it is the first place where we use it.
13. (git) repository -> git repository. It is quite common and known.
14. Do not start the sentence with meaning. Run simulation tests with MITRE Caldera, including the setup and configuration ...
15. the deployment of a Caldera server -> the deployment of a Caldera server,
16. When the tests are finished -> When the tests are finished,
17. it is presumed that we possess initial access. What does this mean? It is unclear? When explaining the concepts in a high-level way, be specific
18. therefore, the acquisition of access -> therefore the acquisition of the access
19. Internal vs Network testing. Use rectangle straight lines for network lines instead of dashed lines.
20. It is unclear what the Caldera Server and Caldera Agent is. Explain what the Caldera Agent and Server is.
21. Rename it to Target Computer and Attacker/Pentester in the diagram.
22. When ready, the solution -> The solution will be executed in Sopra Steria's. This is unclear.
23. security frameworks like MITRE -> security frameworks such as MITRE. Like is informal, such as is more formal.
24. 'Intermediate Purple Teaming', an online learning path sentence is incomplete.
25. No bullet points are needed when we write about our experience, just include it in the text.
26. Thesis structure: Related work: Here we will mention "any". Any is too strong of a word here.
27. Thesis structure: Evaluation: and evaluates the feedback -> and evaluates our product/solution.
28. Furthermore, it will include -> Furthermore, we will present an overview of various open-source cybersecurity frameworks.

29. Delve is not the right word. Wrong use of this word.
30. We have to explain the virtualization figure better. Maybe improve it.
31. Remove the subheading 'Azure Cloud'.
32. In this subsection we will talk about ... (ref: Azure Cloud). It is important to guide the readers. We have to talk about what we are going through.
33. Azure was used as our network infrastructure -> explain why. It was chosen for us by our customer. This sentence needs to be fixed.
34. Interface of Azure (main layout of Azure) is too specific when we are explaining Azure on a high-level. It is not representative of what we were talking about.
35. Azure DevOps part. What features are we talking about. It is unclear when we are writing about it. Mention some of the relevant ones.
36. Some of the features Azure DevOps are ... -> Some of the features offered by Azure DevOps could be
37. Board, which enable a team to follow popular development models like Scrum or Kanban; Repos (short for Repositories).
38. The Pipelines feature allows -> Pipelines allowing for the deployment.
39. It is unclear how we are going to use Azure DevOps. We have to explain it. In addition we are going to use DevOps ...
40. We can combine Azure Cloud and Azure DevOps sections. Feels like reading a dictionary.
41. Security & Penetration Testing, lacks an original source. We have to cite where we took this from. It is not common sense.
42. Cite penetration testing.
43. No subsection about Red & Blue Team. Do not use subsections when there is only one of them.
44. Red & Blue Team -> Red Team & Blue Team. It is important for us to use the same terms all the time.
45. where reds are attackers -> where reds represent attackers.
46. simulates targeted network or system attacks -> simulates targeted network and system attacks
47. Missing reference for purple teaming. Cite a book or research paper.
48. allowing them to identify and fix vulnerabilities. Who is them?
49. Do not cite Wikipedia as it can be altered by editors.
50. Ethical hackers figure. Each figure and table cant stand alone. They have to have a corresponding description. The figure is too big. It is too wide.
51. MITRE ATT&CK no reference. We need to add a reference.
52. and categorizing attackers' or tactics.
53. details the numerous strategies -> details numerous strategies
54. are also provided here, which makes it an efficient tool. Is it true that it is an efficient tool. If we believe it is efficient, we have to say that we believe that.
55. No need to use subsection MITRE attack, just continue.
56. We do not have to write out tactics, techniques and procedures for the second time.
57. Explain the MITRE ATT&CK matrix illustration more.
58. Frameworks title is unclear. Cybersecurity Frameworks is better.
59. Frameworks we need a source here.
60. Frameworks like MITRE ATT&CK help us identify -> Frameworks like MITRE ATT&CK help identify
61. We will only be using open-source -> Because of these benefits, we adopted
62. We have to introduce the frameworks that we are going to talk about.
63. executions of tasks i\* -> executions of tasks. We need references on Automation tools.

64. Gaps between subsections and sections. We have to make it a better transition between them. Guide the readers.
65. More citation.
66. while the steps required for getting there. Getting there is unclear.
67. Coding & Scripting might be unnecessary
68. API figure is too big.
69. framework in order to run automated security -> to run
70. Not clear what not used in the exercises means.
71. In Audun Lundøy Solli -> In the thesis, the author chose to only use
72. Delete the first sentence of 3.2 -> just say the researchers.
73. Delete notes from 3.2.
74. Requirements needs citation.
75. Table 4.1 lists three functional requirements ...
76. 4.1 Requirement -> 4.1 Requirement Description in table
77. 4.1 instead of using no 1,2,3 use F1, F2, F3
78. 4.2 NF2, NF3, NF4
79. Explain a little bit about each requirement in the tables. Some of them are a little unclear. This can be done in the table itself.
80. 4.1 Use Cases needs a description / leading paragraph.
81. Who is the user? Table 4.3? It should be the company, Sopra Steria, security tester.
82. Label below the user 4.1 use cases.
83. System Design - Chapter 5 needs a lot of improvement.
84. Overall architecture is missing in System Design. Make a flowchart or something.
85. It is really important that we explain each part of the system.
86. Automated infrastructure deployment is not good. We should have a common name for the architecture.
87. We have to have an umbrella name for the whole tool.
88. Cite source for Kanban Boards.
89. client meetings. real-time -> real time.
90. In implementation add some screenshots to show it.

**Overall feedback:**

1. NB: Double-check every single sentence, before we submit.
2. Explain illustration, tables and figures more.
3. Add references to EVERYTHING that is not common sense.
4. More leading paragraphs.
5. Like is informal, use other words for this for example 'such as' in text.
6. Missing sources to definitions to a lot of the entries in the glossary.
7. When referencing Chapter or Figure it is important to use capital letter before. e.g. Chapter 4

## **Appendix E**

# **Project Plan**

Attached is the formal project plan, as written at the start of our project.

# **Bachelor's thesis Project Plan**

Group 111

Sopra Steria: Automated Security Testing

## **Team Members:**

Jardar Hollås

Petter Jørgensen

Charlotte Larsen

Tryggvi T. Zabelberg

## **Supervisor:**

Jia-Chun Lin

<b>1. OBJECTIVES AND SCOPE</b>	<b>3</b>
1.1. Background	3
1.2. Project goals	3
<b>2. SCOPE</b>	<b>4</b>
2.1 Subject area	4
2.2 Task description	4
2.3 Limitations	5
<b>3. PROJECT ORGANIZATION</b>	<b>6</b>
3.1. Introduction	6
3.2. Responsibilities and roles	6
3.3. Routines and team rules	7
<b>4. PLANNING, FOLLOW-UP AND REPORTING</b>	<b>8</b>
4.1. Process framework and methodology	8
4.2. Plan for status meetings and decision points during the period	9
<b>5. ORGANIZATION OF QUALITY ASSURANCE</b>	<b>10</b>
5.1. Documentation, standards, configuration management, tools...	10
5.2. Plan for Inspections and Testing	10
5.3. Project risk analysis	11
5.3.1 Explanation of evaluation	11
5.3.2 Risk evaluation	12
5.3.3 Risk Matrix	13
<b>6. IMPLEMENTATION PLAN</b>	<b>14</b>
6.1 Gantt Chart	14
6.2 Project breakdown	15
6.1.1 Security test development	15
6.1.2 Automation	16
<b>7. CONFIRMATION</b>	<b>17</b>
7.1 Signatures	17
<b>Bibliography</b>	<b>18</b>

# 1. OBJECTIVES AND SCOPE

---

## 1.1. Background

Sopra Steria is a leading consultant firm focusing on digitalization [1]. Their Security Operations Center (SOC) provides cybersecurity related services as a Managed Service Provider (MSP). This includes proactive simulation and emulation of threats, security testing, and automation of such processes.

## 1.2. Project goals

The goal for this project is to build a solution that allows for the simulation and/or emulation of attacks against one or several test clients in a network, resulting in a complete report of the process. The technical solution will be built in Azure and will be designed to be installed in a detection lab environment. The goal is to automate parts of the testing process and to assess which open-source frameworks are best suited for this purpose. A summary of the advantages and disadvantages of each solution will also be provided. Additionally, the project will include the creation and documentation of finished tests that can be mapped to MITRE's ATT&CK matrix [2]. The project will also include detailed documentation on the installation and configuration of the selected frameworks, as well as the option to explore the automation of the solution through Azure DevOps pipelines.

## 2. SCOPE

---

### 2.1 Subject area

In this project we are going to learn, use, and write about several different tools and technologies necessary for building our solution. Some of the main topics we will explore, research, and work with are:

- Creating and documenting tests mapped to open source security frameworks such as MITRE ATT&CK
- Assessment and implementation of open-source frameworks for (automatic) red-teaming
- Emulating TTPs (Tactics, Techniques and Procedures) of a threat actor against a test environment
- Building a solution in Azure DevOps for automation of the testing process
- Research and documentation of relevant literature

### 2.2 Task description

The task is to build a solution that enables the client to achieve sufficient alert detection through simulation and emulation of techniques, attacks and threat actors. This will be achieved using a detection lab that the client has set up. The solution will be running in Azure and deployed with Azure DevOps pipelines. Currently the client performs tests manually, but wants to do parts of the testing automatically.

The client therefore specifies the following for our solution:

#### **Core requirements:**

- Evaluate possible frameworks for automatic red-teaming
- The framework should be open source
- A justification of choice for any frameworks we choose to use should be included
  
- Create and document functional security tests which can be used with this framework
- There should be documentation provided for each individual test, and the test should be linked to MITRE ATT&CK
  
- Documentation for installation and configuration of any necessary framework or software



### **Optional, but desired requirements:**

- Automation of the solution
- Automatic installation / deployment of the solution via Azure DevOps pipelines
- Execution of security tests via Azure DevOps pipelines
- Documentation of automation, as well as an evaluation of the security of the implementation

## 2.3 Limitations

In order to ensure we stick to our project plan, reduce impact or possibility of scope creep and to reach our overall goals, it is important to establish clear and realistic limitations for the project.

The following should be considered or could have an impact:

- Setting a specific timeframe for specific tasks, phases and the project as a whole (ref. Fig 1. Gantt Chart), and following the schedule set beforehand
- Clearly defining the scope of the project and making sure that any new tasks or objectives align with the overall project goals, and are doable within a reasonable time frame
- Establishing a budget based on time spent and logging hours to ensure that the project stays within its time limits
- Limiting the number of frameworks and scenarios to be evaluated and implemented.
- Prioritizing and focusing on the most important tasks and objectives, rather than trying to do everything at once
- Regularly reviewing and assessing the progress of the project to ensure that it is on track and making necessary adjustments if needed
- By setting smart limitations and sticking to them, the team can ensure that the project stays focused and on track, and ultimately deliver a successful solution that meets the project's objectives.
- The team members' varying levels of experience might impact the time spent on learning about project-relevant topics. In turn this can have an effect on the timeline and capability of what we can complete during the project.

## 3. PROJECT ORGANIZATION

---

### 3.1. Introduction

This project agreement is based on joint objectives, role definitions, procedures and methods for working together within the team. This agreement has been prepared and agreed to by all team members. This is our common understanding of how the team should handle issues and achieve our objectives.

### 3.2. Responsibilities and roles

The responsibilities and roles are used as a guideline for the general tasks related to the project. These tasks include managing communication with the project supervisor Jia-Chun Lin [5], planning meetings and more. During the project we are open to adjusting and changing the roles when there is a need for it.

- **Project Lead:** Jardar
  - Meeting organizer
  - Point of Contact
  - Track project progress
  
- **Writing Coordinator:** Petter
  - Meeting minutes
  - Weekly summary for supervisor
  - Ensure writing consistency, quality and formatting
  
- **Archivist:** Tryggvi
  - Project documentation
  - Organize project documents and files
  
- **Research Lead:** Charlotte
  - Planning how the research will be done
  - Keep track of sources for later reference
  - Ensure compliance with methodology

### 3.3. Routines and team rules

- **Absence**

It is the responsibility of a team member to inform the team in case of any absence from planned meetings, or assigned work days. It's expected that the notification of absence is given ahead of time, so the team can adjust the workload.

- **Meetings**

Regular meetings with the team supervisor occur every Tuesday between 13:30 and 14:00. Unless otherwise specified these meetings take place digitally through Microsoft Teams. Regular meetings with the project contacts at SopraSteria occur bi-weekly at Thursdays 12:00.

Meetings will otherwise be organized by the team leader, who will also ensure that all relevant parties are informed of and agree to the time/place of the meeting.

- **Coordination and platforms**

Internal meetings and study sessions will be held on a regular basis, either digitally on the collaboration platform Discord or physically at the university campus. Discord will be used as the main platform internally for discussion and coordination (of meetings and otherwise). Meetings with the client and supervisor will be done through using Microsoft Teams.

Google Drive will be used as a digital archive for early stage collaborative writing, informal documents and drafts. Overleaf LaTeX is the planned platform for formal writing and reporting for the bachelor's thesis.

- **Time logging**

All members will log the time used on the project in a document dedicated to the purpose located on Google Drive. This will include a general explanation of what activities have been done, and approximately time used.

Time logging ensures that members contribute to the workload goal of approximately 30 hours per week.

- **Non-compliance**

Disagreements or non-compliance with the project agreement will be resolved internally through discussion if possible. If the issue persists the team's supervisor will be consulted for an alternative solution. Examples of non-compliance with the agreement would be:

- Failing to attend any agreed upon meetings without notice or reasoning
- Refusing to communicate or collaborate

## 4. PLANNING, FOLLOW-UP AND REPORTING

---

### 4.1. Process framework and methodology

The team has chosen to divide the project into a series of phases, milestones, tasks and sub-tasks to ensure efficiency and that a clear and organized work plan is established. To support these objectives, our team has decided to mostly stick to the agile Kanban project management model. Our system will be built in Azure, as specified in our requirements. We will be utilizing Azure DevOps' tools and features, including kanban boards for ease of access, seamless integration and efficiency.

Usage of Azure Boards and backlogs will allow the project team to easily view the status of any task, assign tasks to team members, and track the progress of the project as individual tasks, or as a whole. This is also designed to work directly with Git [3], so we can connect tasks directly to git code branches and commits, upon changes.

Azure (DevOps) is a cloud-based system that enables us to easily collaborate with each other, provide real-time updates, and access project analytics and reports. Finally, the system will be used to control access to resources, review project history and performance, and obtain an accurate and comprehensive view of the project's progress.

#### Specific advantages with the Kanban model

- Easy-to-use tools available to us already in Azure DevOps (Boards)
- The Kanban methodology provides a visual structure to help teams stay organized and on track.
- It offers flexibility and adaptability to quickly respond to changes in project scope and timelines.
- Kanban encourages teams to focus on the current tasks first, which can help to reduce scope creep.
- It's easy to implement and use, so teams can start using it quickly without a steep learning curve.
- Its visual nature makes it easier for teams to understand and discuss progress.
- It promotes collaboration and communication, creating a better team dynamic.
- It encourages continuous improvement and helps teams identify bottlenecks and inefficiencies.

## 4.2. Plan for status meetings and decision points during the period

Weekly meetings with our supervisor will be held each Tuesday 13:30. During these meetings we will go over the progress of the project, and discuss issues or complications that might have occurred.

The team decides work days on a weekly basis, with Tuesday-Thursday usually being the core work session days. The scheduling is flexible, to prevent clashes with work schedules and other commitments. Decision points are agreed upon during meetings and are chosen by the team collectively. When making decisions, we will discuss the progress of the project and any new ideas or direction that the project might take. This allows for the project to stay dynamic, agile and ensures everyone has a say.

## 5. ORGANIZATION OF QUALITY ASSURANCE

---

### 5.1. Documentation, standards, configuration management, tools...

Effective documentation, adherence to standards and proper management of source code are crucial for the success of the project. Our team is committed to ensuring that all work, including practical and technical work, is well-documented and that the necessary documentation is easily accessible to all team members (e.g. in shared locations such as on Google Drive). We will also maintain a time log to track progress after each work session, take meeting minutes and notes of important decisions, and regularly update our scheduling and Gantt charts.

In terms of standards, we will ensure that our code and scripts adhere to the agreed-upon languages and documentation standards, including short and concise commenting within the code itself. This will promote consistency, readability, and ease of maintenance - both for the team, and for potential future maintainers. We will ensure that our code is well-organized, commented, and accessible for the team and our client or supervisor when required.

### 5.2. Plan for Inspections and Testing

To ensure consistency of quality there has to be clear points that we can adhere to when we do inspection and testing throughout the project.

When inspecting test result data we have to be thorough on going over how the test was executed, if it was tested correctly and how the test results match up to the clients detection results.

#### **Inspections:**

- Check if tests were prevented, detected, or nothing happened
- Inspection of documentation and quality
  
- Was the test executed as planned?
  - Did the right tests get executed?
  - Were the tests ran against the right environment (e.g. operating system, hardware, endpoint)
  - Did we give the detection system enough time to respond to our tests?
  
- Validate test results. Did we receive correct testing results back? How do they match up to their detection system? Check for False Positives in detection system results.

**Testing:**

- Map testing stages onto MITRE's ATT&CK framework matrix.
- Follow MITRE's ATT&CK framework.
- Manual testing -> Automatic (Azure DevOps Pipelines)
- Glassbox testing approach

### 5.3. Project risk analysis

The table below shows a collection of possible risks involved with the project work as a whole, based on the team's thoughts early in the project. The likelihood and impacts are ranked from 1 to 5, where 1 is least likely/lowest impact and 5 is most likely/highest impact.

#### 5.3.1 Explanation of evaluation

<b>Level</b>	<b>Likelihood, freq. (P)</b>	<b>Impact, significance</b>
1	Very Unlikely $P < 1/365$ Less than once a year	Insignificant impact, does not matter or can be fixed easily
2	Somewhat unlikely $P = 1/365$ to $2/365$ Once to twice a year	Low impact, might lead to slight postponement
3	Likely $P = 3/365$ to $12/365$ Monthly	Medium impact, could lead to longer delays and decrease of quality in work
4	Very likely $P = 13/365$ to $36/365$ One to three times a month	Significant impact, leads to lower quality work and possibly worse grades
5	Almost certain $P > 36/365$ More than three times a month	Critical impact, will miss important parts of the work and could fail us the course

### 5.3.2 Risk evaluation

We have conducted a simple risk analysis to identify likelihood, consequences and mitigating actions for possible risks that could impact the project quality. Below are a few of the risks we could face.

**ID: 1**

**Risk:** Overwhelmed by work

**Description:** Too many tasks to do at once while a deadline is approaching

**Likelihood:** 4

**Impact:** 2

**Action:** Work continuously, divide fairly, do not skip weekly sessions

**ID: 2**

**Risk:** Sickness / dropout

**Description:** A team member could get sick or unable to continue working on the project, leading to a loss of valuable skills and knowledge.

**Likelihood:** 2

**Impact:** 3

**Action:** Have a plan, stick to the rules agreed to beforehand, ask for help from other team members or the supervisor, adjust expectations and deliverables, and ensure not to rely solely on one team member for the success of the project.

**ID: 3**

**Risk:** Limited Experience

**Description:** Team members may have limited experience in specific subjects which could lead to a higher need of introductory reading and learning.

**Likelihood:** 4

**Impact:** 2

**Action:** Watch lectures and tutorials, complete recommended courses, learn by working on tasks and experimenting, and read online resources to gain knowledge and skills. Ask the supervisor or client for help.

**ID: 4**

**Risk:** Data loss

**Description:** Data, documents or files that are crucial to the project could be lost, potentially causing delays and difficulties in completing the project.

**Likelihood:** 1

**Impact:** 3

**Action:** We are using cloud-based storage for documents and files which are identified as safe, and ensures integrated remote backups automatically. Further, we could create multiple local backups of important data, and ensure that accounts and systems used for storing data are secure.



**ID:** 5

**Risk:** Missing deadlines or meetings

**Description:** Failing to deliver on time or attend important meetings could have consequences like failing the course, or otherwise impact project quality.

**Likelihood:** 1

**Impact:** 4

**Action:** Add events and deadlines to calendars, set up reminders, and work continuously to stay on track and meet deadlines.

**ID:** 6

**Risk:** Misunderstandings

**Description:** Poor communication and lack of clarity on the vision for tasks could lead to confusion and double-work among team members.

**Likelihood:** 3

**Impact:** 2

**Action:** Keep each other updated on the progress and goals of tasks, and ensure that everyone is on the same page and collaborating with the same vision.

### 5.3.3 Risk Matrix

Based on the risks found in 5.2. We decided to visualize the risks by using the Risk Matrix [4].

	Impact 1	Impact 2	Impact 3	Impact 4	Impact 5
Likelihood 5					
Likelihood 4		<b>1, 3</b>			
Likelihood 3		<b>6</b>			
Likelihood 2					
Likelihood 1			<b>4</b>	<b>5</b>	<b>2</b>

*Number in **bold** identifies the risk ID from the Risk evaluation*

**Green** indicates acceptable risks

**Yellow** indicates risks where we should consider taking action

**Red** indicates unacceptable risks where actions need to be taken immediately.

## 6. IMPLEMENTATION PLAN

### 6.1 Gantt Chart

This gantt chart displays the schedule of our project, broken down into distinct phases. It highlights the start and end dates, duration, and dependencies of each task within each phase. The chart provides a clear overview of the project timeline and helps us plan and coordinate the various tasks and activities involved. By using this tool, we can ensure that we allocate resources effectively, avoid delays, and complete the project on time and within our time budget. Note that the gantt chart is subject to change. An updated version can be found [here](#).

TASK NR.	Task title	Phase 1 - Planning			Phase 2 - F&R				Phase 3 - Project Implementation					Phase 4 - Writing & Finalizing					Phase 5 - Presentation					
		02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	Planning	09.01	16.01	23.01	30.01	06.02	13.02	20.02	27.02	06.03	13.03	20.03	27.03	03.04	10.04	17.04	24.04	01.05	08.05	15.05	22.05	29.05	05.06	12.06
1.1	Project plan																							
1.2	Contract																							
1.3	Courses/AttackIQ																							
1.4	Submission of plan / contract																							
2	Familiarization and research																							
2.1	Familiarization with Azure Devops Environment																							
2.2	Proof-of-Concept security test																							
2.3	Research into automation																							
2.4	Initial deployment to VM																							
2.5	Security testing frameworks																							
2.6	Assessment of the threat environment																							
3	Project implementation & first draft																							
3.1	Implementing frameworks																							
3.2	Automation																							
3.3	Minimum Viable Product																							
3.4	Test development & Mapping to MITRE																							
3.5	Testing																							
3.6	Gathering results																							
3.7	Ensure proper documentation																							
4	Writing and finalizing																							
4.1	Writing																							
4.2	Quality Control																							
5	Presentation																							
5.2	Presentation																							

Figure 1: Gantt-chart

#### Milestones

- Project Plan: 31.01
- First draft: 31.03
- Final Thesis: 20.05
- Presentation: First half of June

## 6.2 Project breakdown

The following diagram visualizes the project's main areas of work and breaks them down into smaller subsections. We decided to only include the more general parts of the project, the parts that we know that we will work with and complete.

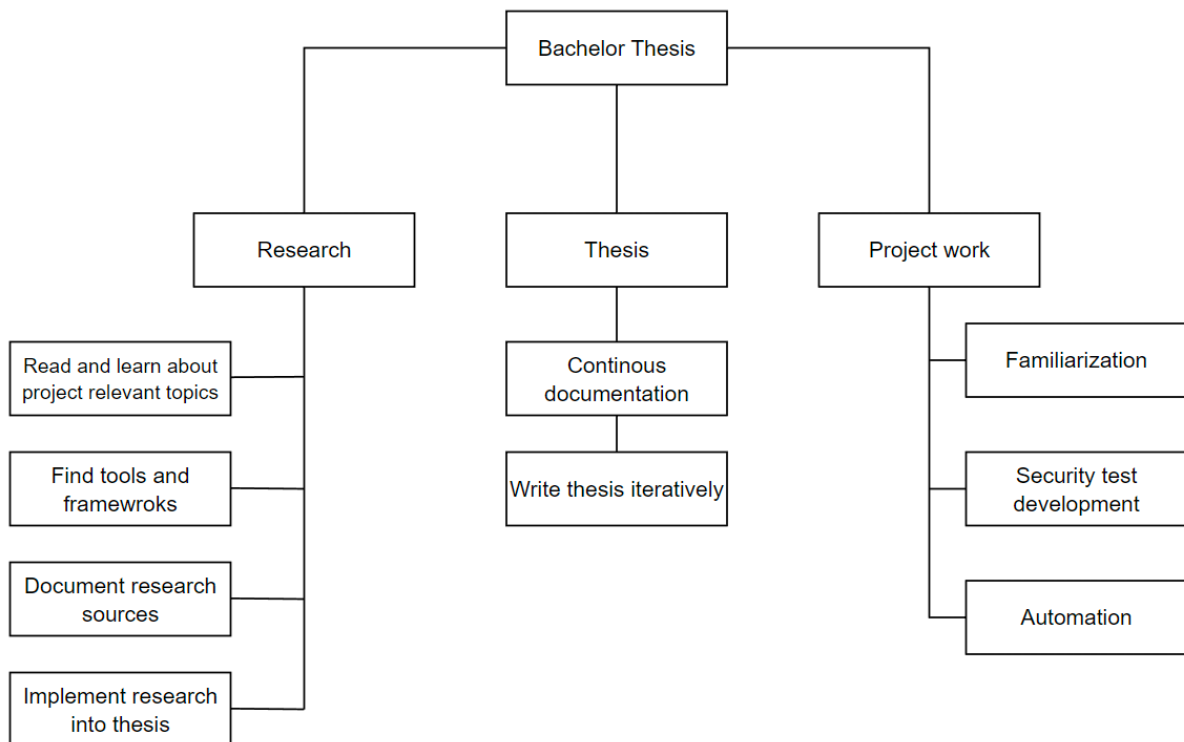


Figure 2: Project breakdown

### 6.1.1 Security test development

The most basic condition for our thesis is the capability of executing (software) security tests, also known as penetration tests or pentests, on one host against another on the same local network. The test being executed from a VM (Virtual Machine) can range from simple recon tools executed with PowerShell, to more extensive testing provided by open-source security testing frameworks like Atomic Red Team [6]. Figure 3 shows a basic illustration of such a test.

### External security test

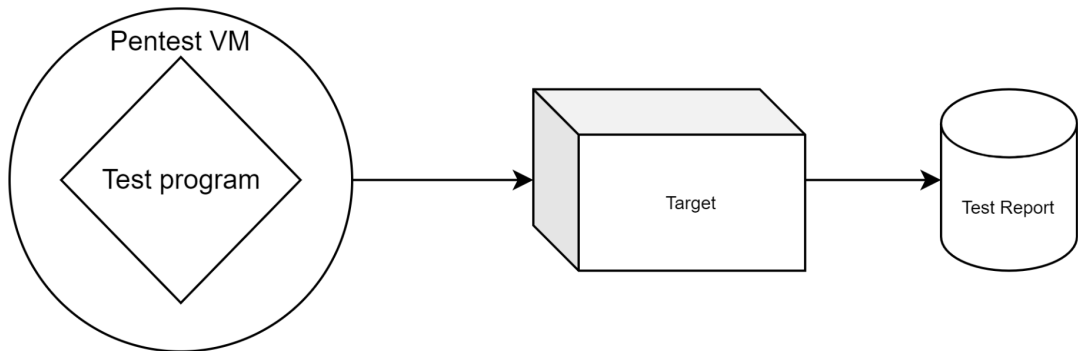


Figure 3: External security test

Some testing will be impractical to execute from an external VM. A proper test of a systems in depth defenses will also include tests that assume some level of system compromise, and will therefore need to be executed internally on the target (see figure 4). For example a security test attempting privilege escalation might be ran by an existing system user, thereby assuming that an attacker already obtained initial access through this user's credentials.

### Internal Security tests

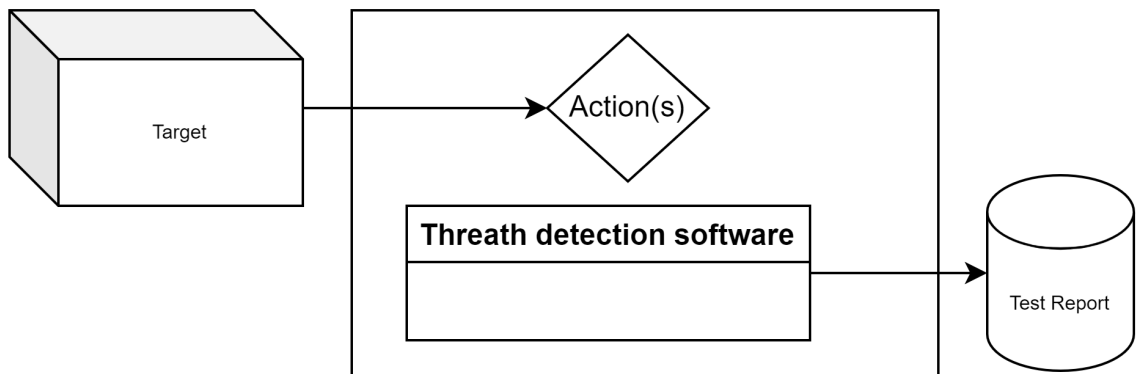


Figure 4: Internal security test

### 6.1.2 Automation

Automation of our solution is part of the optional requirements. As we progress with the core functionality of our solution we will look for ways to automate the process as well. It is not entirely clear at this stage exactly how or what tools we will use to achieve this. We are however looking into the viability of using Azure Devops' [7] Repos and Pipelines features for accomplishing this.

## 7. CONFIRMATION

---

I acknowledge that I have reviewed the project plan and agree to its terms and conditions. I am committed to following the outlined steps, my role and team responsibilities and the timeline to ensure my contribution to a successful completion of the project.

### 7.1 Signatures

Jardar Hollås:

*Jardar Hollås*

Date: 31.01.2023

Petter Jørgensen:

*Petter Jørgensen*

Date: 31.01.2023

Charlotte Larsen:

*Charlotte Larsen*

Date: 31.01.2023

Tryggvi T. Zabelberg:

*Tryggvi T. Zabelberg*

Date: 31.01.2023

## Bibliography

---

[1] About us | Sopra Steria - <https://www.soprasteria.com/about-us>

[2] MITRE ATT&CK Framework - <https://attack.mitre.org/>

[3] Git | Microsoft Azure - <https://azure.microsoft.com/en-us/products/devops/repos>

[4] Risk Matrix - <https://www.uib.no/en/hms-portalen/142418/risk-matrix#risk-acceptance-criteria>

[5] Jia-Chun Lin - <https://www.ntnu.edu/employees/jia-chun.lin>

[6] Atomic Red Team - <https://atomicredteam.io/learn-more/>

[7] Azure DevOps Services - <https://azure.microsoft.com/en-us/products/devops>

## **Appendix F**

# **Additional Listings**

```

1  $resourceGroupName = $env:DEPLOY_RG
2  $location           = "West Europe"
3  $vmName             = $env:VM_NAME
4  $scriptName         = "CalderaSetup"
5  $containerIP        = $env:CONTAINERIP
6  $fileUri = @"(https://rsbppipelinestorage.blob.core.windows.net/script/CalderaAgentSetup.ps1)"
7
8  $settings = @{"fileUri" = $fileUri};
9  $protectedSettings = @{
10     "commandToExecute" = `
11         "powershell -ExecutionPolicy Unrestricted" + `
12         "-File CalderaAgentSetup.ps1" + `
13         "-containerip ${containerIP}"
14     };
15
16 Write-Host $vmName
17 Write-Host $containerIP
18
19 #run command
20 Set-AzVMExtension -ResourceGroupName $resourceGroupName `
21     -Location $location `
22     -VMName $vmName `
23     -Name $scriptName `
24     -Publisher "Microsoft.Compute" `
25     -ExtensionType "CustomScriptExtension" `
26     -TypeHandlerVersion "1.10" `
27     -Settings $settings `
28     -ProtectedSettings $protectedSettings;
29
30
31
32 <# For Pipeline
33
34 - stage: 'AGENT_INSTALLATION'
35     dependsOn: 'DEPLOY'
36     jobs:
37     - deployment: 'Azure_AGENT_INSTALLATION'
38         displayName: 'Azure AGENT INSTALLATION'
39         environment: $(environment)
40         strategy:
41             runOnce:
42                 deploy:
43                     steps:
44                     - checkout: Bachelor
45                     - task: AzurePowerShell@5
46                       inputs:
47                         azureSubscription: '$(azureSubscription)'
48                         ScriptType: 'FilePath'
49                         ScriptPath: '$(LocalRepo)/Scripts/Start-AzureCustomExtensionV2.ps1'
50                         azurePowerShellVersion: 'LatestVersion'
51                         pwsh: true
52                     env:
53                         CONTAINERIP: $(containerIP)
54 #>

```

Listing 17: Agent-Configuration.ps1



```

1 param($containerip)
2 Start-Transcript -path C:\output.txt -append
3 $containerip | Out-File -FilePath C:\containerip.txt
4 $server="http://"+$containerip+":8888";
5 $url="$server/file/download";
6 Write-Host $url
7 $wc=New-Object System.Net.WebClient;
8 $wc.Headers.add("platform","windows");
9 $wc.Headers.add("file","sandcat.go");
10 $data=$wc.DownloadData($url);
11 get-process | ? {$_.modules.filename -like "C:\Users\Public\splunkd.exe"} | stop-process -f;
12 rm -force "C:\Users\Public\splunkd.exe" -ea ignore;
13 [io.file]::WriteAllBytes("C:\Users\Public\splunkd.exe",$data) | Out-Null;
14 Start-Process -FilePath C:\Users\Public\splunkd.exe -ArgumentList "-server $server -group red" -WindowStyle h
15 Stop-Transcript

```

Listing 18: CalderaAgentSetup.ps1

```

1 resource "azurerm_network_interface" "primary" {
2   name                = "${var.prefix}-w10-primary"
3   location            = var.location
4   resource_group_name = var.resource_group_name
5   internal_dns_name_label = "${var.prefix}-w10"
6
7   ip_configuration {
8     name                = "primary"
9     subnet_id          = var.subnet_id
10    private_ip_address_allocation = "Dynamic"
11  }
12 }
13
14 resource "azurerm_network_interface_security_group_association" "spoke-nsg" {
15   network_interface_id = azurerm_network_interface.primary.id
16   network_security_group_id = var.nsg_id
17 }
18
19 resource "azurerm_network_interface_application_security_group_association" "lab-asg" {
20   network_interface_id = azurerm_network_interface.primary.id
21   application_security_group_id = var.asg_id
22 }

```

Listing 19: 1-network-interface.tf

```

1  locals {
2    custom_data_content = "${file("${path.module}/files/winrm.ps1")}"
3  }
4
5  resource "azurerm_virtual_machine" "windows10" {
6    name                = "${var.prefix}-w10"
7    location            = var.location
8    resource_group_name = var.resource_group_name
9    network_interface_ids = [azurerm_network_interface.primary.id]
10   vm_size             = var.vm_size
11   delete_os_disk_on_termination = true
12   delete_data_disks_on_termination = true
13
14   storage_image_reference {
15     publisher = var.vm_image.publisher
16     offer     = var.vm_image.offer
17     sku      = var.vm_image.sku
18     version  = "latest"
19   }
20
21   storage_os_disk {
22     name          = "${var.prefix}-w10-disk"
23     caching      = "ReadWrite"
24     create_option = "FromImage"
25     managed_disk_type = "Standard_LRS"
26   }
27
28   os_profile {
29     computer_name = "${var.prefix}-w10"
30     admin_username = var.admin_username
31     admin_password = var.admin_password
32     custom_data    = local.custom_data_content
33   }
34
35   os_profile_windows_config {
36     provision_vm_agent = true
37     enable_automatic_upgrades = false
38
39     # Immediately logs onto VM upon creation
40     additional_unattend_config {
41       pass          = "oobeSystem"
42       component     = "Microsoft-Windows-Shell-Setup"
43       setting_name  = "AutoLogon"
44       content       = <AutoLogon><Password><Value>${var.admin_password}</Value>
45                    </Password><Enabled>true</Enabled><LogonCount>1</LogonCount>
46                    <Username>${var.admin_username}</Username></AutoLogon>
47
48     }
49
50     # Unattend config is to enable basic auth in WinRM, required for RDP to work
51     additional_unattend_config {
52       pass          = "oobeSystem"
53       component     = "Microsoft-Windows-Shell-Setup"
54       setting_name  = "FirstLogonCommands"
55       content       = "${file("${path.module}/files/FirstLogonCommands.xml")}"
56     }
57     winrm {
58       protocol = "HTTP"
59     }
60   }
61 }
62 }

```

Listing 20: 2-virtual-machine.tf

```
1 #####
2 # Outputs
3 #####
4
5 output "vm_name" {
6     value = azurerm_virtual_machine.windows10.name
7 }
```

Listing 21: Module outputs.tf

```
1 variable "resource_group_name" {
2     description = "The name of the Resource Group where the Windows Client resources will be created"
3 }
4
5 variable "location" {
6     description = "The Azure Region in which the Resource Group exists"
7     default = "westeurope"
8 }
9
10 variable "vm_image" {
11     description = "The Windows image"
12     type = map(string)
13 }
14
15 variable "vm_size" {
16     description = "The VM size"
17 }
18
19 variable "prefix" {
20     description = "The name prefix used for infrastructure resources"
21 }
22
23 variable "subnet_id" {
24     description = "The Subnet ID which the Windows Client's NIC should be created in"
25 }
26 }
27
28 variable "nsg_id" {
29     description = "The Network Security Group ID which the Domain Controller's NIC should be created in"
30 }
31
32 variable "asg_id" {
33     description = "The Application Security Group ID which the Domain Controller's NIC should be created in"
34 }
35
36 variable "admin_username" {
37     description = "The username associated with the local administrator account on the virtual machine"
38 }
39 variable "admin_password" {
40     description = "The password associated with the local administrator account on the virtual machine"
41     sensitive = true
42 }
```

Listing 22: Module variables.tf



 **NTNU**

Norwegian University of  
Science and Technology