Bachelor's thesis

Nataniel Gåsøy
Maarten Dijkstra
Christian Bråthen Tverberg

Red vs Blue,

Cyber Security Simulator

May 2019

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Bachelor’s thesis

2019

Nataniel Gåsøy
Maarten Dijkstra
Christian Bråthen Tverberg

Red vs Blue,

Cyber Security Simulator

Bachelor's thesis
May 2019

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Sammendrag av Bacheloroppgaven

Tittel:	**Red VS Blue, Cyber Security Simulator**
Dato:	20.05.2019
Deltakere:	*Christian Bråthen Tverberg* *Maarten Dijkstra* *Nataniel Gåsøy*
Veiledere:	Mariusz Nowostawski
Oppdragsgiver:	The Norwegian Cyber Range.
Kontaktperson:	Muhammad Mudassar Yamin, muhammad.m.yamin@ntnu.no, +4796999968
Nøkkelord:	C#, Unity3D, Cyber sikkerhet, seriøst spill, 2D, Norwegian cyber range, CTF verktøy
Antall sider:	64
Antall vedlegg:	9
Tilgjengelighet:	Åpen

Sammendrag:

Dette prosjektet ble laget som grunnlaget for en avhandling for bachelor i programmering. Det er en type "serious game", eller et spill som har et formål annet enn kun underholdning, i dette tilfellet for trening og undervisning, og fokuserer på cyber sikkerhet. Prosjektet er laget med Unity3D med agil metodikk der en blanding av scrum og kanban tavle er brukt.

Målet med dette prosjektet var å lage en prototype av en simuleringsplatform for cyber sikkerhet der man har en side som prøver å angripe et system, og en side som prøver å beskytte det. Dette inkluderer også et verktøy for å lage scenarioer, samt funksjonaliteten til å spille gjennom de lagde scenarioene over et lokalt nettverk. I et scenario er den angripende og beskyttende siden styrt av deltagere, enten som lag eller individer, på separate maskiner. Dette er et høy-level system som fokuserer på å ta den rette avgjørelsen i et gitt cyber sikkerhets scenario, og hele systemet er 2D basert.

Det ferdige produktet består av to moduler, en for å lage og endre scenarioer og en for å spille gjennom scenarioer. Redigeringsmodulen er klikk-og-dra basert, og blir brukt til å lage topologien til systemet i scenarioet, samt sette attributene og informasjonen relevant for scenarioet. Spilldelen er sanntidsbasert og styrt kun ved klikking, og fokuserer på å først bli kjent med scenarioet presentert for så å utføre eller forhindre angrep på det avhengig av spillerens rolle.

Dette prosjektet er hovedsakelig ment for å brukes av Norwegian Cyber Range, men kan bli brukt av hvem som helst, og fokuset dets er trening og opplæring/undervisning i cybersikkerhet. Dette har vært en fin læringsopplevelse for oss, både når det kommer til cybersikkerhet og hvordan slike scenarioer er satt og, og for Unity3D's UI system.

Summary of Graduate Project

Title:	**Red VS Blue, Cyber Security Simulator**
Date:	20.05.2019
Authors:	*Christian Bråthen Tverberg* *Maarten Dijkstra* *Nataniel Gåsøy*
Supervisor:	Mariusz Nowostawski
Employer:	The Norwegian Cyber Range.
Contact Person:	Muhammad Mudassar Yamin, muhammad.m.yamin@ntnu.no, +4796999968
Keywords:	C#, Unity3D, Cyber Security, Serious Game, 2D, Norwegian cyber range, CTF tool
Pages:	64
Attachments:	9
Availability:	Open

Abstract:

This project was made as the basis for a thesis written for a bachelor in programming degree. It was made as a type of serious game, or a game with a purpose other than pure entertainment, in this case for education and training, and focuses on cyber security. The project was made using the Unity3D engine with an agile method consisting of a mix between scrum and kanban.

The goal for this project was to create a prototype of an attacker vs defender cyber security simulator, with an editor for creating scenarios as well as the functionality to play a created scenario over a local network. In a scenario the attacker and defender sides are controlled by participants, either teams or individuals, on separate machines. It is a high-level system that focuses on making the right decisions regarding cyber security when dealing with a certain scenario, and the entire system is 2D based.

The finished product consists of two modules, one for creating and editing scenarios and one for playing through a scenario. The editor is drag-and-drop based and used for creating the topology of the scenario's system, as well as setting the attributes and information relevant for the scenario. The gameplay part is real-time and click-based, and focuses on getting to know the scenario system and perform or mitigate attacks, based on what role the player has.

This project is mainly intended to used by the Norwegian Cyber Range, but can be used by anyone. Its focus is on training and education regarding cyber security. It has been a great learning experience for us, both in regards to cyber security and how a scenario is set up, as well as Unity3D's UI system.

Preface

We would like to thank the following:

The Norwegian Cyber Range for letting us work with this project, being available for constant discussion during the development period and helping us find testers from the target group of the project.

Mariusz Nowostawski for supervising this project, for giving us helpful feedback and for helping us get in contact with others who could inspect and help us improve our final product.

The testers for testing our system, providing useful feedback and helping us improve it.

Simon McCallum for motivating the development of a LATEX template for GUC's master's theses. This template has been very helpful during the writing of the thesis.

Contents

List of Figures

List of Tables

Listings

1 Introduction

The goal for this project was to create a prototype of a cyber security simulator type serious game consisting of an editor for creating scenarios as well as the functionality to play a created scenario. In a scenario the attacker and defender sides (red and blue) are played by participants, either teams or individuals. It is a high-level system that focuses on making the right decisions regarding cyber security when dealing with a certain scenario.

Figure 1: Game icon

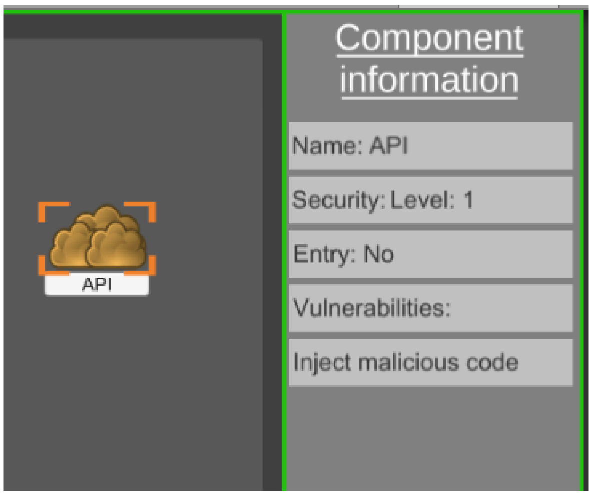
1.1 Project description

The project is a serious game that has two parts, or modules. One module is an editor for creating the topology of a scenario, and the second module is a gameplay section to play the created scenarios. In other words, it is based around creating and playing simulated cyber security scenarios through decision making within a serious game-type environment [3]. The model for the scenarios is a red vs. blue type of capture the flag event (CTF) [4], or an attacker team vs a defender team. The entire game has a 2D user interface (UI) with point-and-click functionality. A scenario in this context is a certain, specified set of actions the defending and attacking team can take as well as the system they will be acting upon. There is also an observer mode that is able to see what actions the attacker and defender are taking, as well as inspect the different components in the system in order to see their current status.

This project focuses on technology development for a serious game in the context of cyber security training. The project team consists of programmers, and it was a bachelor project for a Programming degree with focus on games. Due to time constraints, and the fact that all the team members are programmers, rather than designers, we have not focused much on the aesthetics and chose a simplistic design. The controls are also relatively simple, where all actions are done with left- and right clicking. It was made as a tool for teaching and training decision making regarding cyber security situations to be used by students and others with varying degrees of familiarity with the subject.

The product owner for this project was the Norwegian Cyber Range (NCR) [5]. As the goal was to train decision making, this is a very high-level system, where the details behind how an attack is mitigated or executed is abstracted away and the focus is on what action to take, rather than how to perform it. An example of this abstraction is the use of code injection, as depicted in fig. 2. There is no information of what type of injection this is, how it is executed or the details around it, just that there is a code injection vulnerability that the defender will have to mitigate before the attacker can exploit it.

The game concept was developed during the ongoing research activities at the NCR. The inefficiencies in cyber security exercises was identified by the project owner and this project is in part to address those inefficiencies, and will be used in ongoing research and training activities at the NCR.



Component information

Name: API

Security: Level: 1

Entry: No

Vulnerabilities:

Inject malicious code

Figure 2: A component with a code injection vulnerability

The project was made with the Unity3D game engine. This was a request from the product owner in order to make further development at a later stage easier, and this suited us well as this game engine was one our team was familiar with. The programming language used was C# as this was the main language supported by Unity3D [6].

A user manual for the project can be found in appendix G.

1.2 Terminology

Scenario: A scenario in this context is the topology of the system in question, what vulnerabilities are present in the system, the information regarding the attacker and defender in regards to their starting skills, attributes and resources as well as the system statistics.

NCR: The Norwegian Cyber Range, or NCR for short, is an arena for testing, training and practice for cyber security. They are currently working on expanding their toolbox of systems at their disposal to achieve this, and one of those expansions is this project.

Attacker: The individual or team whose goal is to compromise the system presented through the scenario.

Defender: The individual or team whose goal is to defend the system presented through the scenario, and mitigate potential attacks.

CTF: " There are two main types of Capture the Flags (CTFs) challenges: Jeopardy-style CTFs consisting of completing different challenges from a broad range of categories in order to earn points. Attack-defence CTFs consisting of two teams, one team who defends a network or server, and another who attack it. " [4]

Serious game: A game that has a function beyond purely entertainment, in this case training and education [3].

DSL: Domain specific language. A computer language specialized to a certain application domain to express logic/concepts/relationships. The YAML format used in this project is a part of the DSL of the NCR.

2 Requirements

2.1 Functionality

The main focus for this project is to implement a scenario creator in order to create and edit scenarios, and gameplay in order to play the given scenarios. It is also important to do this in a scalable and dynamic way, as this project is meant to be a prototype that we do not know if will be further developed or not. For this to be accomplished, there are some core features for each part of the system that needs to be implemented.

2.1.1 Initial functional requirements

The initial functional requirements was to make the scenario creator and gameplay as two modules of a separate system. As the project was somewhat ambiguous from the start, the requirements were broad and got narrowed down and specified as the work with the project progressed.

The scenario creator should be a separate module meant to create or edit a scenario. There should be functionality to represent a system topology, including location, connection and system component information. This information should include whether the system component is "safe" or if it is in some way exploitable. The list of basic functionalities and system components as listed by the product owner during the start of the project is as follows:

Policies

Physical environment: buildings, office space, doors, windows, furniture.

Servers: OS, configuration, websites, API.

Clients: Desktops, laptops, mobile, IoT, configuration.

Transmission media: LAN/WAN/WLAN, hub, switch, router.

Internet: DNS, social media, internet traffic.

Vulnerabilities: OWASP web top 10, OWASP mobile top 10, OWASP IoT top 10.

SIMS: Motivation, skills, communication, ethics, training and adaptability.

The difficulty level and resources for each player should also be defined in the scenario creator, and saved as a part of the specific scenario.

The original requirements for gameplay mostly included lists of what the participants should have access to, as well as a logging system to log player actions. The initial plan was therefore to mainly base the vulnerabilities on OWASP top 10 vulnerabilities for web (see table 1).

OWASP top 10, web
Injection
Broken authentication/ session management
Sensitive data exposure
XML external entities (XXE)
Broken access control
Security misconfiguration
Cross-site scripting (XSS)
Insecure deserialization
Using components with known vulnerabilities
Insufficient logging and monitoring

Table 1: OWASP top 10 security risks for web applications [2]

The overall functionality of the gameplay was to have a real-time system with gameplay revolving around one attacker (player/team) trying to compromise a system, while one defender (player/team) were to mitigate and prevent this (a red vs. blue scenario). There should be a considerable focus on human errors and human exploitability, not just digital ones.

From the product owner specifications, the attacking side should have access to a variety of different methods and exploits in order to gain access to and compromise the systems. The initial list consisted of:

- Public exploits.
- Zero day exploits.
- Exploitation tools.
- Insiders.
- Publicly available information.

The defending side, on the other hand, should have access to different methods to prevent the attacker from exploiting the system, and mitigating vulnerabilities. The initial list of these methods consisted of:

- Parameter security.
- IPS/IDS.
- Threat intelligence.
- Firewalls.
- Secure configurations.
- Training.
- Awareness.
- Penetration tests.
- Security operation centers.

We also received a list of functionalities typically found in games, but as this was a very generic list it was not very relevant to this project. Furthermore, the product owner requested the gameplay to be real time as opposed to turn-based in order to add an innovative element to the project. The project was not expected to have any sort of AI. This was partially due to the fact that there would be little time to implement this, and partially due to the fact that this would be a later project already planned by the project owner. There would be no animation as this would be time-consuming and serve no functional purpose.

This system would initially be developed as a local multiplayer, as the finished product would mostly run on local networks. If there was time, this could be changed to run on a dedicated server later, but this was not a priority.

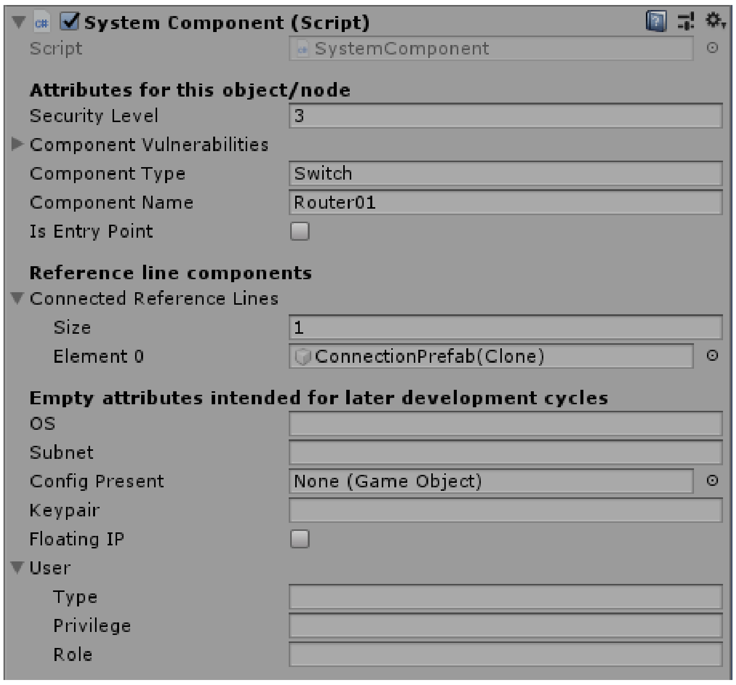
2.1.2 Changes to functional requirements

There were a lot of changes to the requirements throughout this project. As most of these were merely suggested and discussed and not actually planned we did unfortunately not think to document them all, but there were roughly 40 suggested requirement changes. This was to be expected, as the product owner had an overall idea of what he wanted the finished product to be like, but no hard plan or complete list of requirements. The largest changes to the requirements were the shifts in focus areas, as well as the continuous expansion of the backlog with more features like alternate gameplay modes, replay functionality and an observer mode.

Another big change to the requirements was to have this integrated as a part of the NCR toolbox and possibly use the scenario creator for other systems in the toolbox as well, as opposed to being a stand-alone system that would not interact with anything but it's own components. In order to do this we had to adapt the system based on the domain specific language (DSL) developed by a Masters student (an example of this can be seen in appendix C). This mainly led to two changes in the requirement; logging and system component data. A part of accommodating these changes was also to make it as easy as possible to change these systems later as the DSL was still under development and prone to change.

The logging system was initially just intended to log player actions in a non-standard file format. As the system was to be integrated into the NCR toolbox and communicate with other systems, the product owner requested that we log using the YAML format instead, with more information being logged like system state and changes.

As the scenario creator would interact with several other systems, it needed to store data we did not necessarily use in our system (see fig. 3). This also included changing the save- and load functionality in order to accommodate these new data fields, as well as reworking the saving system to more easily add new fields later.



System Component (Script)

Script	SystemComponent
Attributes for this object/node	
Security Level	3
Component Vulnerabilities	
Component Type	Switch
Component Name	Router01
Is Entry Point	☐
Reference line components	
Connected Reference Lines	
Size	1
Element 0	ConnectionPrefab(Clone)
Empty attributes intended for later development cycles	
OS	
Subnet	
Config Present	None (Game Object)
Keypair	
Floating IP	☐
User	
Type	
Privilege	
Role	

Figure 3: A system component in the scenario creator

As the system was to be made more modular with more freedom to the end user, this effectively made the lists of methods the attacker and defender had access to redundant. The focus shifted to making a modular and expansive system where the user could define vulnerabilities and mitigations as they became relevant or needed. This also included the human errors and human vulnerabilities, as these could be added in the same way as digital vulnerabilities. The requirement for the list of vulnerabilities and mitigations in the base system were therefore changed to revolve around the OWASP top 10 vulnerabilities list (see table 1).

In addition to the attacker and the defender screen, the product owner wanted an observer screen. This should be able to keep track of what the attacker and defender are doing, including what node they are inspecting, as well as be able to inspect different nodes in the system topology and see their status. This was to be accomplished with a visual representation of the topology, as well as logging to the screen whenever the attacker or defender performed an action.

2.1.3 Final functional requirements

Scenario creator

The user should be able to make as complex a scenario as desired in the scenario creator given screen space for this. This included system components, connections, vulnerabilities and firewalls as well as other minor components.

The user should be able to add new available system components and vulnerabilities, but this would not be possible during run-time of the game. A default system component button should be present for the user to add a new component during run-time that will have the functionality to change its attributes and data, but the icon will stay default until an appropriate icon has been added using the Unity3D editor.

System components: Selecting a system component from a given list, including a default placeholder that would have a default icon but all other values for a component, and place these on a chosen location within the editing field.

Implement a series of components from a list provided by a PhD student working with the NCR.

Connections: Connect various system components together with a visible connection. These clearly shows which components are connected to which in order to serve their function in the system, and also how an attacker would traverse between the system components. These connections should have the option of having a firewall.

Vulnerabilities: Each system component can have an arbitrary number of vulnerabilities. These vulnerabilities are loaded from a separate file, and the user can select which specific vulnerabilities are present at each individual component, if any at all. The file containing the list of vulnerabilities can be changed to add and remove available vulnerabilities.

Scenario statistics: Various data statistics should be saved with the scenario. This includes what the scenario is named, how long the scenario should last and what resources and state the attacker and defender starts with.

Saving and loading: As the data that needs to be saved and loaded is prone to change, both during the project and in future work, the saving and loading function should be easily located and changed. It should also accommodate data identified by the DSL, and save and load these even if this information is not necessarily used in our system (see fig. 3).

Gameplay

A user should be able to host a lobby, or to join an already hosted lobby. A user cannot host a lobby and join the same lobby, as the host becomes the observer and can therefore not also be a player. This lobby should exist while the user hosting the lobby is still hosting, and become unavailable for everyone once the user stops hosting.

If there is a lobby with two players, an attacker and defender, the host is able to start the selected scenario. This will bring the players to their respective game screens, and the host to the observer screen.

A user in an active lobby with both an attacker and defender should be able to play a given scenario until the game enters a goal state (time runs out, attacker compromises the system or other goal states later defined), a player leaves the lobby or the host terminates the game.

Gameplay: Gameplay is real-time, with a click-based 2D UI system.

Attacker screen: The attacker screen would start off without knowing any of the topology for the system, and would have to traverse and exploit the different system components in order to learn more.

Defender screen: The defender would start with a full overview of the system topology, and would therefore get a head start in discovering weaknesses and mitigating these before the attacker could exploit them

Observer screen: The observer screen would not be able to act upon the system, and would merely be able to inspect the state of the nodes in the system. The observer would also be able to see what node the attacker and defender are currently inspecting, as well as have access to the event log showing all the actions the attacker and defender takes.

Networking: As this was to be a multiplayer system and Unity3D's old networking API, UNet, is deprecated [7], a requirement was to use the new networking system. A user should be able to create a lobby, that other users then could detect and join. The multiplayer aspect of the game were to be strictly local multiplayer, not using a dedicated server.

Logging: The system should log the entire progression of the played scenario. It should also log every action that changes the current system during gameplay in a YAML format.

2.2 Reliability

The system should be stable, both in the scenario creator and gameplay, and should not crash by itself. The players may experience an abrupt stop in gameplay, but this should only be when the host or a player for some reason disconnects or leaves. This should not be caused by an error in our system, but rather be caused by another event our system reacts to such as loss of connectivity or manual termination from one or more users of the system.

2.3 Performance

During run time, the system should not drop under 60 frames per second on a decent running PC. Gameplay should not be too impacted by running on a lower end PC, as the gameplay is UI and click based, which does not require as much processing power and frame rate as other game system but the frame rate could be impacted by this. The system should be responsive to user actions, and should give visual responses that a user's actions have been registered.

2.3.1 Scenario creator

Performance is not as important in the scenario creator as in the gameplay part. Nevertheless, it is still important that the scenario creator have a reasonable response time, so as to not make it feel tedious to work with.

2.3.2 Gameplay

It is crucial that the gameplay part of the system is highly responsive, as this will be a real-time game. It is therefore important to have the system display the visual ques of a player's actions quickly as to avoid frustration with the system, and carry out user actions immediately.

3 Design

3.1 Modules

As the project had two main modules, the scenario creator and the gameplay, it was a natural conclusion for us to have a main menu as the first thing the user sees when the program starts. From here, the user can choose whether to play a scenario, make or edit a scenario or go into the settings menu for the entire program. We chose to use a minimalist design for the program, which we felt worked well with the functionality of the project and did not distract the user.

3.1.1 Scenario creator

The scenario creator would need a sizable information display as well as a list over the components that could be added to the scenario. These components must be defined in advance due to the size and image, but a default component has also been included with a default shape and image. All components can be renamed and have all its effective data changed in the editor.

3.1.2 Gameplay

The gameplay would be three similar screens with different functionality. It was therefore a focus to make these look similar to each other, though not identical. The attacker and defender screens were color coded as to easily distinguish them from each other, and would have the information column on the right side, similar to the scenario creator.

3.2 Scenario creator design

Most of the scenario creator is the "editing space", the area the scenario can be created within. In a new scenario, this starts off as a blank screen as it is the blank canvas upon which the user creates the scenario.

The left-hand menu containing available system components is a list of buttons. We originally intended to have these ready icons that you dragged out of the menu and onto the editing field, but due to technical difficulties this had to be changed to buttons that spawn the corresponding component when pressed.

In order to free up as much space as possible, most of the actions are hidden away in sub-menus the user can access by right-clicking the component they wish to change. This brings up a new list of actions the user can perform to adapt the system components to their needs.

The right-hand column lists information about the currently selected system component, where the user can easily identify the current status on each component.

3.2.1 System components

The system components were designed to be individual nodes that the user can customize based on need. All the attributes, at least the ones relevant for now in the system, as well as name and position can be freely edited within the editing field. Note that dropping a component outside of the editing field will cause it to be deleted, along with its connections lines.

The original plan was to have the components displayed in the left-hand menu for the user to drag-and-drop from the menu as well. However, this broke the references to the components as well as providing other technical difficulties, leading to us opting for a spawning button instead. The plan was also to have a semi-transparent component icon follow the cursor after clicking a component until it was placed in the editing field. However, we did not have time to implement this, so as of the date of delivering the project it simply spawns the components on the left-hand side of the editing field.

The icons for the system components are mainly from Open Source Architecture(OSA) [8], a site containing free of charge icons for security architecture. The rest were either made by us, or edited based on the OSA icons.

3.2.2 Connections

The connections are two straight lines with a 90 degree angle in the intersection that visually represent which components are connected in what way. If the connection has a firewall installed, this will be visible as a brick wall placed on the longest of the two lines representing the connection in order to improve visibility. We looked to other similar projects for inspiration, like ThreatGEN [9] and Uplink [10].

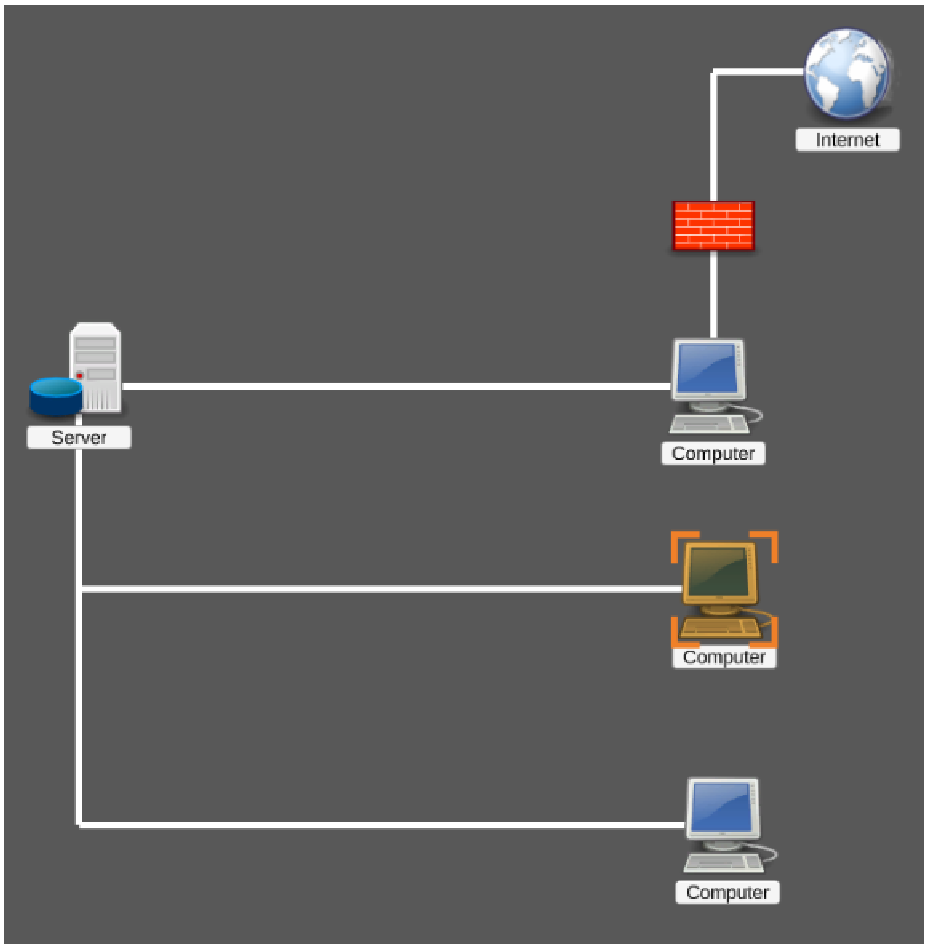


Figure 4: Connections between system components

3.2.3 Menus

For the menus, we wanted to have as little information clutter the base screen as possible. After consulting with the supervisor as well as fellow students, we concluded that having the options in menus and sub-menus was the best way to achieve the desired functionality while at the same time keep the base screen open and free from clutter.

The first menu the user can gain access to was the one brought up from right-clicking a system component. From this menu the user can edit the basic attributes of a system component, like its security level or if it is an entry point. For more involved actions like renaming the component or changing the list of vulnerabilities the component has, a different sub-menu will be opened. While in these sub-menus, the user will not be able to interact with the rest of the system before exiting the sub-menu.

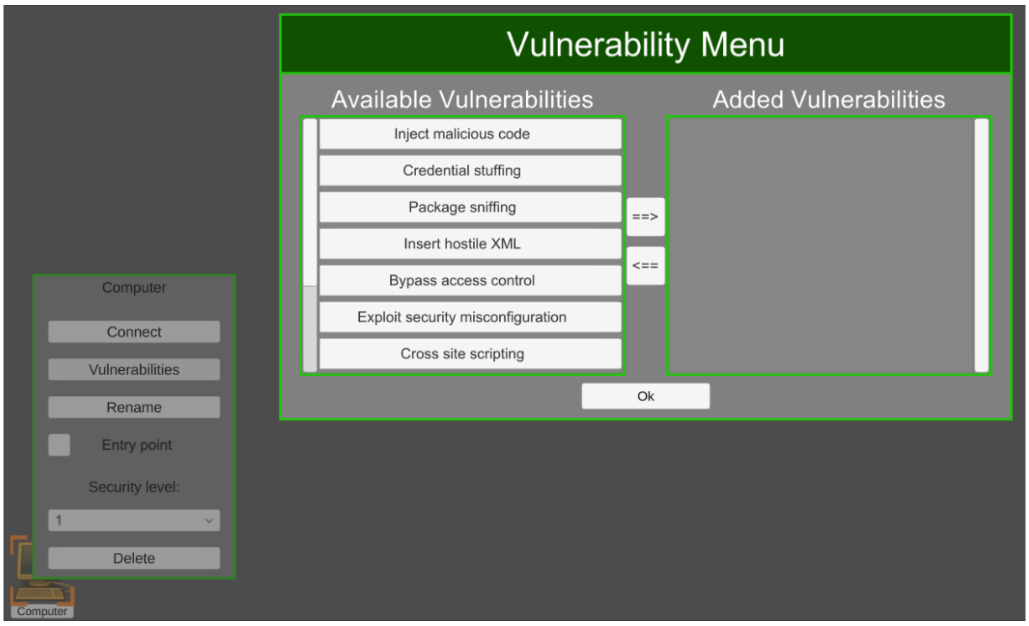


Figure 5: The different type of menus, right-click menu and screen centered sub-menu

3.3 Gameplay

3.3.1 Game lobby

The players have the option of hosting a game as an observer, or join the game a host is serving as an attacker or defender. These options will be given through a matchmaking screen, where the player is also given all possible lobbies to join. To be able to host or join a game the player will need to put in a name here as an identifier for the lobby.

A game lobby was desired to show players that were connected to the same host. It will show the names of the host, attacker(s) and defender(s) that wants to play in that specific game as seen in fig. 6. The attackers and defenders have the option to swap position/roles, for example to give the attacker an opportunity to play as defender, by pressing the name of the other player.

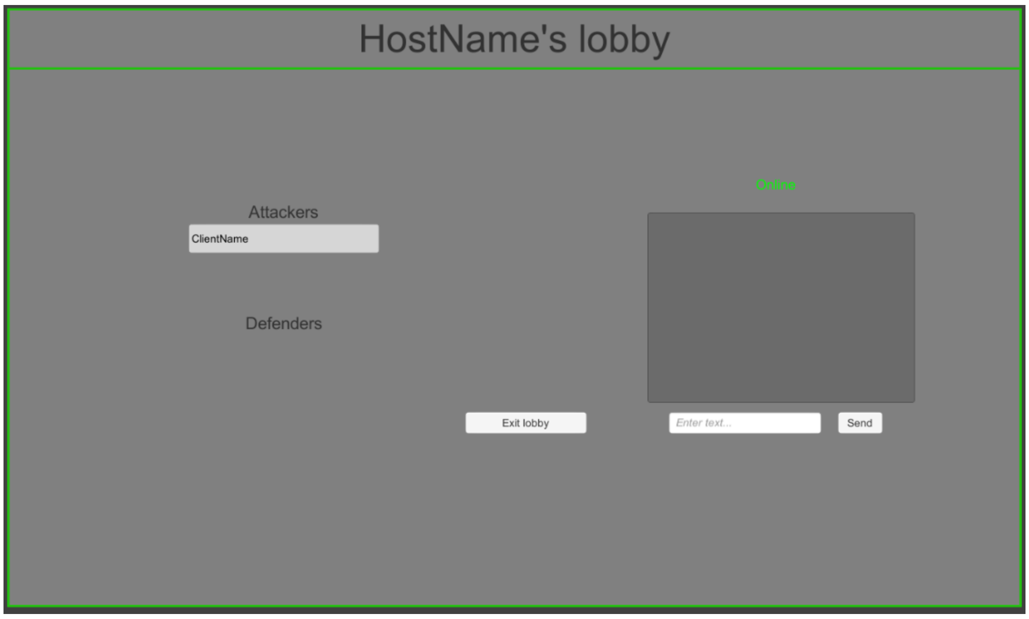


Figure 6: Lobby screen showing the host name at the top player on the attacker team, chat between the people in this lobby and for the host the button to start the game.

3.3.2 Gameplay UI

We wanted an easy to use and simple UI design with the main focus being the topology in the scenario. This led to having a big window with plenty of space to show the system components without making the icons feel small or cramped. Color coding was used to clearly show which role the user has. Within cyber security, the red team are those tasked with testing and attacking a system, whilst blue are those tasked with defending against incoming attacks [11]. Based on this we made our attackers have red out-lines and colors during gameplay and the defender have blue, as shown in fig. 7. Since we also have an observer, we decided to use a green color to create a large enough color difference to avoid confusion. Green was used as the main color in menus in the rest of the product.

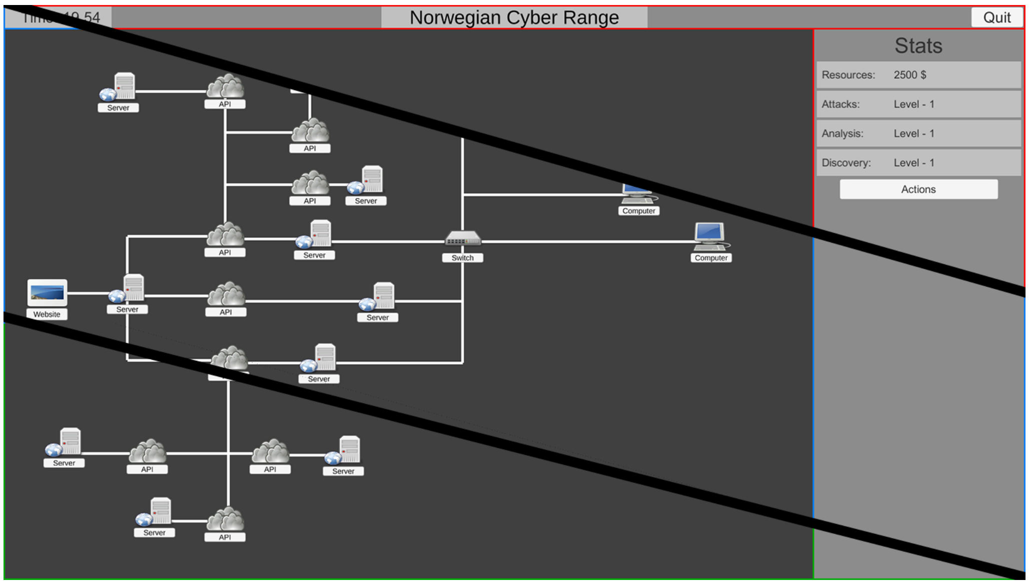


Figure 7: Combined image of the three different UI color themes. Red is the attacker (top right), blue is the defender (middle), and green is the observer (bottom left).

To make the game a little more difficult to get an initial overview, we decided that the defender should start with all the nodes visible upon start whilst the attacker has no nodes visible. For the defender this can cause a slight confusion, simulating what it might be like when being newly hired. As for the attacker, this might simulate a lack of knowledge of the infrastructure of the network, forcing the attacker to decide between brute force or information gathering.

4 Development Process

A mixture of scrum with a kanban board [12] was used for the development of this project. An agile method seemed the most appropriate for this project, as the product owner had a general idea of what he wanted and the project was ambiguous. The issue board functionality GitLab provides was used for the kanban board, where we put up the tasks after dividing the larger tasks into several smaller ones.

4.1 Roles

Group leader: Nataniel.
Secretary: Nataniel.
Git manager: Maarten.
Build and deployment manager: Maarten.
UI and design: Maarten.
Developers: Christian, Nataniel, Maarten.

4.1.1 Role explanation

Group leader:

- Encourage group discussion in order to make decisions throughout the project.
- Keep track of time and meetings, be a group representative on meetings that do not require the entire group to be present.
- Responsible for communication with product owner and supervisor.

Secretary:

- Take notes during the meetings with the product owner and with the supervisor.

Git manager:

- Managing Git.
- Responsible for merging and pull requests.

Build and deployment manager:

- Make sure the system builds.
- Make an installer for the final product.

UI and design:

- Responsible for consistency in the UI and design.

Developers:

- Develop the system.
- Each developer was responsible for testing their code and its functionality under development.

4.1.2 Workload

We were initially wondering how we would split the workload, as the project we were making had two parts to it, the scenario creator and the game play. However, we soon discovered that UNet, Unity3D's networking API, was deprecated and would start losing support in 2021 [7]. This led to networking requiring more work than previously anticipated, and it was decided that we would need one person dedicated to networking. We concluded that Christian would be working on this, while Maarten worked on the gameplay aspect and Nataniel worked on the scenario creator.

4.2 Working hours

As we were going to be working together in the same room, we decided to have a regular working schedule in order to ensure that we were present at the same time. For the daily workflow, we decided to work with regular work times, 8 hours a day from 09.00 to 17.00 with half an hour lunch breaks working together in the Game lab (A253). We decided to work a 09.00-17.00 schedule, as these are normal working hours that mostly fitted the team member's schedules.

4.3 Sprints

The work was sprint-based, with 2-week long sprints. At the start and end of a sprint, as well as halfway through a sprint we planned a meeting in order to keep up to date on what the others were doing and how the overall project was coming along. As time was limited, we started the first implementation sprint halfway through the planning sprint. This was a conscious choice, as the first sprint would mainly focus on prototyping and visual mock-ups. This would heavily influence the second part of the planning process, as this would give us a clearer understanding of what to focus on.

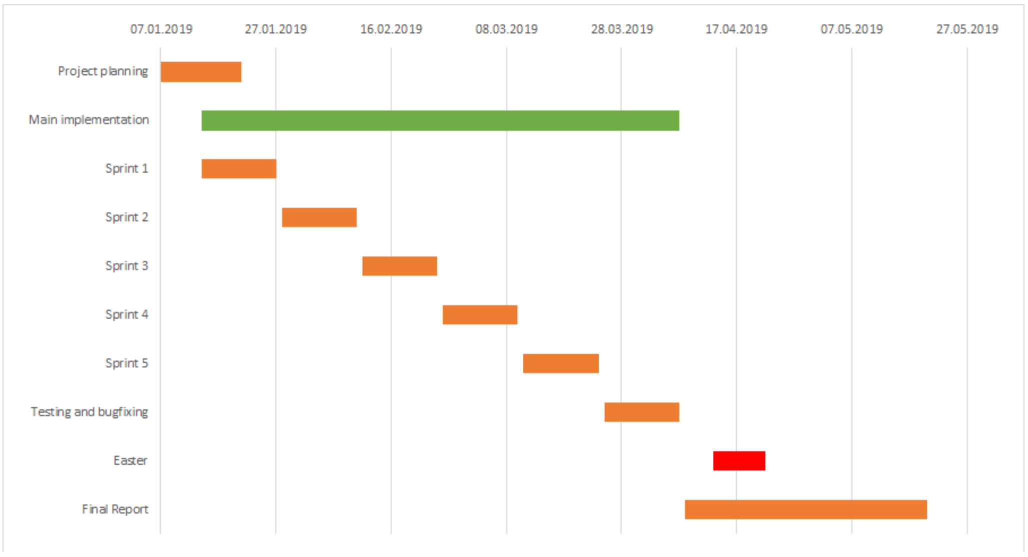


Figure 8: Gantt chart

4.3.1 Planning

For the first sprint, the project planning sprint, there were 3 milestones. The first milestone was to have a project plan written down, where the initial visual and functional requirements were specified. The second milestone was having a visual prototype accepted by the product owner in order to know what style we were aiming for, and the third milestone was to have a functional prototype accepted by the product owner.

4.3.2 Implementation

Sprint two through five was the main implementation stage, where most of the functionality and architecture for the system was implemented. The main milestones for this period was having initial versions and the final versions completed for the different components as well as having networking operational. This included the first and final version for the scenario creator with the editor, the first and final version of the gameplay that utilizes the scenario created as well as having networking for the gameplay. Later, two more milestone were added. The first of these was to successfully implement saving to file for the scenario creator, and loading from file for the scenario creator and for gameplay. The second was to create a spectator mode, where one might spectate what the attacker and defender does, as well as the status for each system component at a given time.

4.3.3 Testing and fixing bugs

After the implementation phase we planned a sprint for testing and fixing bugs, where we also looked for feedback on the design of the project as well as finding assets and improving the design and visual aspects of the project. The plan was to have this finished by Easter, so that the entire time period after Easter could be dedicated to writing the thesis.

This sprint was not completed in the planned time, partially due to the product owner wishing for more functionality we wished to implement, and partially due to the implementation time of other functionality that was started but not yet completed. Another reason for testing being delayed was that we got the opportunity to test our system at the Norwegian Cyber Security Challenge 2019 [13]. Here our system would be used as one of the challenges the participants would meet in order to determine who would represent Norway in the European Cyber Security Challenge 2019 [14].

4.4 Tools

4.4.1 Game Engine

The product owner requested that we make this project using the Unity3D game engine with the Unity3D framework, as this is a prototype and there are plans to expand upon it later. This expansion will most likely be done by students, and as Unity3D is the most used game engine in the programming course (at the time of writing this report), the product owner wanted to use Unity3D in order for the next group to more easily pick up the project.

This also made the choice of programming language easy, as Unity3D now only fully supports C# [6]. JavaScript used to be supported, but was deprecated in 2017 and thus was not an option for us. Additionally, the entire team were more comfortable programming in C# than JavaScript.

4.4.2 Git

Git was used for version control. Version control was important for our project in order to have consistency, and as this was a tool that we were familiar with using this came natural for us to use. We chose Gitlab for storing our repositories as it provides the tools we wished to have for issue tracking, as we here could have the kanban board directly in the repository for ease of access.

Using git with Unity3D is prone to create merge errors. To mitigate this, we decided to work with different scenes in Unity3D as to not have overlapping work that caused merge errors, perform all merges on the same computer to avoid metadata conflicts as well as being on a separate branch while working on features. In retrospect, we should have had a developer/merge branch as well, where all merging would be done before pushing this to master instead of merging directly on the master branch.

We considered using Unity3D collaborate in order to mitigate some of these conflicts, but as we would lose many tools such as branches, commit messages and revert functionality by doing this we opted not to.

4.4.3 Issue tracker

We used Gitlab as our issue tracking tool, as we already had our project repository there and Gitlab provided an integrated issue tracker. The issue tracker was also used to host our kanban board [12].

The kanban board we had 7 steps that a task could be located in: backlog, development, waiting for review, in review, product owner review, bug/issue and closed. This gave us a good workflow progression, where everything started in the backlog on the left side of the board, and steadily progressed to the right towards closed. The only times something went from right to left was if a task had been marked as closed but a bug was later discovered, and consequently when a bug was fixed and the task was moved to waiting for review. It was important to note that most of these tasks would not go through all the steps, as many of them were technical and not something the product owner needed to review. Bugs/issues was also a step most tasks did not get placed in as tasks would not be moved from development to waiting for review with known bugs.

4.4.4 Documentation

For the code documentation, we decided to comment in the Doxygen style [15] in order to automatically generate the documentation, which is located in the git repository. The parts of the documentation that was not direct documentation of code has been written and stored on Overleaf, an online LaTeX editor [16]. This allowed us to have the documentation stored in the cloud, as well as to perform collaborative editing.

4.4.5 Development environment

As the product owner wanted the product to be made with the Unity3D game engine, it was a natural choice to work in Windows using Microsoft Visual Studio as our coding environment. While it was possible to use Unity3D on Linux by mimicking the Windows libraries provided by the Wine software, there was no official Unity3D support for Linux at that date [17].

4.4.6 Communication tools

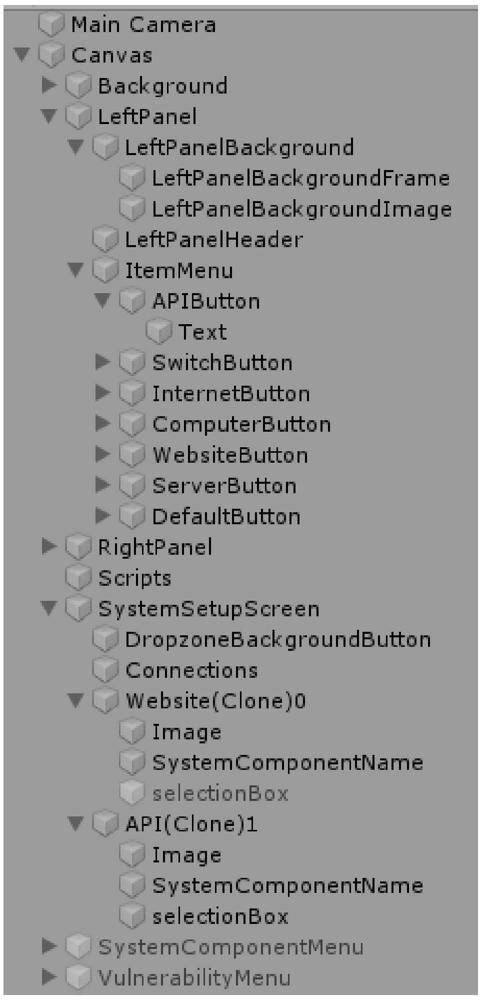
As we were working together in the same room for most of the working period of this project, we mostly communicated without the use of a messaging service and simply turned around to the person we needed to discuss something with. During the start of the project, we mostly used Facebook Messenger as our communication tool. As discord proved to be an easier tool to use, as well as for keeping track of various subjects at the same time, we changed from Messenger and over to a dedicated discord server early during development. We also communicated with the supervisor using discord.

5 Implementation

5.1 Unity3D

We started developing this project in the latest released version of the Unity3D game engine available, which was version 2018.3.2f1. During development, newer versions of Unity3D were released. We decided to not update our version of Unity3D but keep the same version for consistency, as well as to avoid unforeseen conflicts that could present itself in new releases.

Working with Unity3D, or any existing game engine or framework will heavily influence the architecture of the projects developed. One of the most influential aspects to this in our project is Unity3D's hierarchical structure, as can be seen in the example of fig. 9, and the fact that it is a component based system. Using a component design allows isolating the various features in a single compact service. In that way the components are decoupled, and functions independent of each other [18].



```
Main Camera
▼ Canvas
  ▶ Background
  ▼ LeftPanel
    ▼ LeftPanelBackground
        LeftPanelBackgroundFrame
        LeftPanelBackgroundImage
      LeftPanelHeader
    ▼ ItemMenu
      ▼ APIButton
          Text
      ▶ SwitchButton
      ▶ InternetButton
      ▶ ComputerButton
      ▶ WebsiteButton
      ▶ ServerButton
      ▶ DefaultButton
  ▶ RightPanel
    Scripts
  ▼ SystemSetupScreen
      DropzoneBackgroundButton
      Connections
    ▼ Website(Clone)0
        Image
        SystemComponentName
        selectionBox
    ▼ API(Clone)1
        Image
        SystemComponentName
        selectionBox
  ▶ SystemComponentMenu
  ▶ VulnerabilityMenu
```

Figure 9: An example of the hierarchy structure of a Unity3D `scene`

In Unity3D, the containers for sets of components is called a `GameObject`. Custom scripts can be placed on these `GameObjects` if they derive from the built-in class `MonoBehaviour`. This is a necessary step to connect the script to the internal Unity3D pipeline, and will let Unity3D handle the component management. You can store a particular `GameObject` composition in an object called a `prefab`, which makes it easy to reference the object as well as instantiate copies of it during run-time. Unity3D also has a container for `GameObjects` called `scenes`. These are used to separate different parts of the game, like different screens or levels [19].

5.2 Program flow

As the system has two clearly separated parts (see fig. 10), we decided to do this as two modules that would mostly be independent of each other. As it made little sense to separate these into two systems, a main menu was implemented. This menu is a separate scene that holds the settings menu as well as connecting to the scenario creator scene and the matchmaking scene for gameplay.

Every part of the system can be exited in order to return to the main menu, and from the main menu the program can be terminated. As we wanted the program flow to be intuitive, we tried to keep to the standard file path traversal when making the progression system. The exit/quit button is at the bottom of the screen, main actions or the main focus is centered on the screen and buttons and actions are clearly marked.

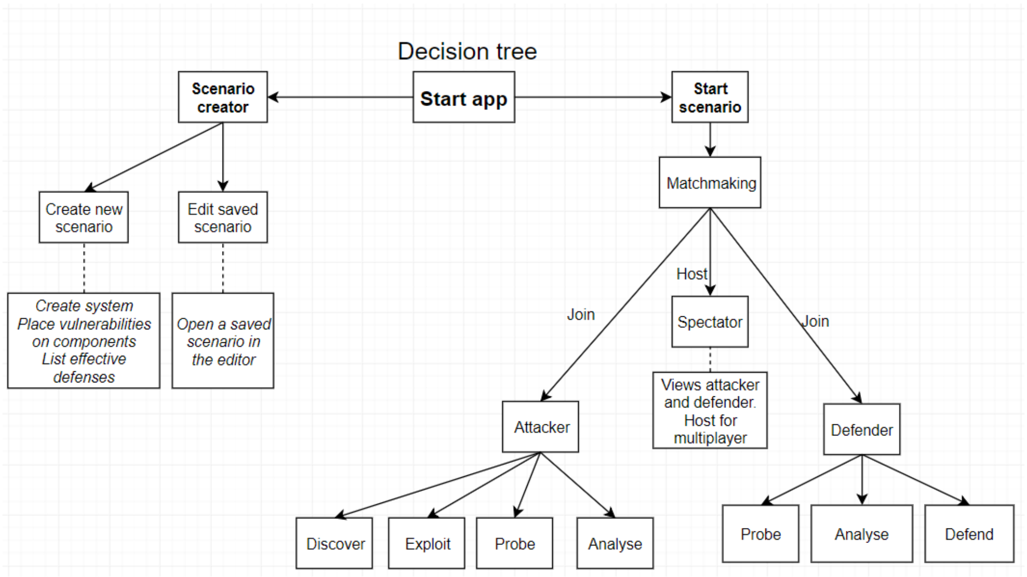


Figure 10: The decision tree of the system, overview

For a more detailed flowchart of the system, please see appendix F.

5.3 Scenario creator

One of the two modules for our project, the scenario creator is meant to be the editor module of the system where the scenarios can be made or edited. When entering the scenario creator from the main menu, a prompt is asking whether the user want to start a new scenario or load an existing one. The only difference this makes is whether or not there are any components present in the editing screen when the user enters the scenario creator. This is a mandatory first step as it needs to be decided at the start of entering the editor. These visual representations can be seen in fig. 11 where the right-hand items shows the object that is hovered over (red) and selected (orange) compared to their default states seen in the left-hand column.

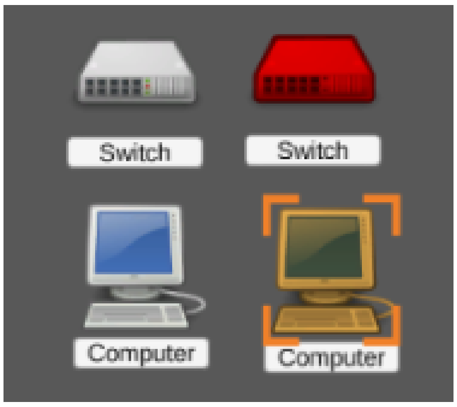


Figure 11: Visualizations of objects the user is interacting with

The scenario creator has a few scripts in an empty `GameObject` in the hierarchy. This `GameObject`'s only function is to hold these scripts as they control the saving and loading of a scenario. As these scripts are not specific to a component in the system, they stand as independent scripts in their own `GameObject`. This is done as the scripts have to be present in the hierarchy to be used, and it is not possible to have a script directly in the hierarchy. It is also easier to keep track of these stand-alone scripts when they are all collected in the same place. This `GameObject` also holds the script for storing the currently selected object in the editor scene, as well as the previously selected one. This allows others scripts to access the currently selected object, as several functions and scripts behaviour depends on the currently selected component.

5.3.1 System components

The system components are implemented as independent `GameObjects` instantiated from `prefabs` upon clicking the corresponding button, and can be placed anywhere within the editing field. In addition to the system component specific script, each system component also has the `HighlightObject` script and `DraggableObject` script, allowing the user to change its position by drag-and-dropping it as well as highlighting selectable object when hovered over. This script also ensures that the currently selected object is clearly visualized (see fig. 11).

The system component script holds all the attributes for each individual component. This includes both the attributes we currently use in our system, as well as empty attributes requested by the product owner. The attributes we are not using in our system is by default set to null/false/0 as they are not utilized, simply implemented.

The components should not be able to be placed outside of the editing field, as this field corresponds to the size of the gameplay screen. System components dropped outside of the editing screen are therefore deleted. This ensures that there are no components outside of the scope, as well as giving the user an easier method of deleting a system component than through the right-click menu.

Listing 5.1: The attributes of a system component.

```
[Header("Attributes for this object/node")]
public int securityLevel;
public List<AttackTypes> componentVulnerabilities = new List<
    AttackTypes>();
public string componentType;   // Must match the prefab name
public string componentName;
public bool isEntryPoint;
private TMP_Text displayedName;

[Header("Reference line components")]
public List<GameObject> connectedReferenceLines = new List<
    GameObject>();
List<GameObject> connectedSystemComponents = new List<
    GameObject>();
private GameObject connectedSystemCommponent;
private Transform lineToEnd;
private Transform lineFromStart;
private GameObject firewall;

[Header("Empty attributes intended for later development
    cycles")]
public string OS = null;
public string subnet = null;
public GameObject configPresent = null;
public string keypair = null;
public bool floatingIP = false;
public User user = new User();
```

5.3.2 Connections

The connection lines are made up of two "boxes" connected at one edge, in order to represent a line with a 90 degree angle bend (see fig. 4). The scaling of the lines were done using vector calculations based on the positions of the two components it connects, and rotated using quaternions [20]. Code listing 5.2 shows a piece of the calculation of a connection line's position, rotation and scale (specifically when the target system component is positioned down and to the right from the original component's position). It is part of the *SetConnectionLines* function, which takes the positions of the two components and the connection line instance to be changed and updates the connection line position.

In order to easily perform actions on the connection line, such as moving or deleting it, the connection line saves references to the two system components it connects. In addition to this, the system components saves a list of references of all the connection lines that are attached to it.

Listing 5.2: Example of connection line calculation.

```
lineToEnd.localScale = new Vector3(XPosDiff, 0.05f, 1.0f);
lineFromStart.localScale = new Vector3(0.05f, YPosDiff, 1.0f);
connectionLine.transform.rotation = Quaternion.identity;

lineFromStart.position = new Vector3(startPos.x, endPos.y +
    YPosDiff * 50);
lineToEnd.position = new Vector3(lineFromStart.position.x +
    XPosDiff * 50, endPos.y);
```

A problem that occurred during development was that the connection lines would sometimes not be selectable at all. This proved to be a trouble with rotation, as rotations along the X and Y dimensions would sometimes rotate the object 180 degrees around and show the backside of the connection line. This led to Unity3D not registering the user trying to interact with the line, as only the front side can be interacted with. This was resolved by only rotating in the Z dimension and compensate for with scaling.

Another problem with the connection lines were that they were sometimes rendered over system components due to Unity3D's hierarchical architecture. This was resolved by grouping all the connection lines under the same empty `GameObject` and making sure that they were set at the first sibling, in order to ensure that all connection lines would always be drawn behind all system components.

5.3.3 Menus in the scenario creator

The menus were made after the core functionalities of placing system components and connecting them was finished. It was decided to use menus as this would save the most space in the base screen and would still be an efficient and intuitive way to edit various attributes for the scenario. They are mostly blank pictures with buttons and input fields on them, keeping in mind a simplistic design, and are split into two different kinds namely right-click menus, and centered menus (both of these can be seen in fig. 5).

The right-click menus are the system component menu and the connection line menu. These are the menus that will spawn where the user right-click on a system component or connection line.

The centered menus are the menus that appear in the center of the screen. These, unlike the right-click menus, appear in the same space every time. They also have a semi-transparent image attached to their background ensuring that the player cannot interact with any objects outside of these menus while they are active and open. These menus include the rename menu, the vulnerabilities menu and the menu the user sees when opening the scenario creator, asking if the user wishes to load an existing scenario or make a new one from scratch.

5.4 Saving and loading a scenario

We decided to load to and save scenario from a separate file as this was the easiest solution that would make the saved scenarios available both from the scenario creator and gameplay module. As we wanted a human readable save file format, we chose to save the game data using the JSON format.

5.4.1 Saving a scenario

In order to properly parse the desired data to JSON and save it to a file, a separate `Save` class was made, containing the data fields we wanted to save. Part of this save file can be seen in code listing 5.3. As nested lists are not supported by the JsonUtility [21], the JSON parser we used for this project, we also had to write a separate wrapper in order to contain the nested list.

Listing 5.3: Structure of the Save class.

```
[Header("Saved system component data")]
public List<string> systemComponentNamesList = new List<string
    >();
public List<Vector3> systemComponentPositionsList = new List<
    Vector3>();
public List<string> systemComponentTypesList = new List<string
    >();
public List<int> systemComponentSecurityLevelsList = new List<
    int>();
public List<bool> isEntryPointList = new List<bool>();
public List<VulnerabilityWrapper>
    systemComponentVulnerabilyWrappersList = new List<
    VulnerabilityWrapper>();

[Header("Saved reference line data")]
public List<Vector3> connectionLinePosition = new List<Vector3
    >();
public List<GameObject> referenceFromObject = new List<
    GameObject>();
public List<GameObject> referenceToObject = new List<
    GameObject>();
public List<string> referenceFromObjectName = new List<string
    >();
public List<string> referenceToObjectName = new List<string>()
    ;
public List<bool> hasFirewall = new List<bool>();
public List<Vector3> firewallPositionsList = new List<Vector3
    >();

[Header("Empty attributes intended for later development
    cycles")]
public List<string> OSList = new List<string>();
public List<string> subnetList = new List<string>();
public List<GameObject> configPresentList = new List<
    GameObject>();
public List<string> keypairList = new List<string>();
public List<bool> floatingIPList = new List<bool>();
public List<User> usersList = new List<User>();
```

This gave us complete control of what would be saved in the file, and what would not. As the parsing of JSON objects would not work for us if we did not know what we were saving and what we would later be loading, this suited our needs perfectly. It also ensured that we only needed to save the relevant data for building the designed scenario, instead of having to save a lot of unnecessary data we would never use.

We are also saving the empty attributes that are not used in our system, as the product owner wanted these implemented for aligning our system with other systems from the NCR toolbox.

5.4.2 Loading a scenario

As with the saving of data, the loading of data is done by reading the entire JSON string from the file and parsing this into an instance of the Save class. This is then used to load the required information for the components, connections and attributes from a file and into the system. The loading of all data for system components can be seen in code listing 5.4. This script also includes the instantiate code for all of the system components. These are located in separate functions as they can be connected to an in-game button using the Unity3D editor.

Listing 5.4: Loading system component data from file.

```csharp
for (int i = 0; i < loadFromJSON.systemComponentPositionsList.
    Count; i++)
{
    VulnerabilityWrapper vulnerabilityWrapper = loadFromJSON.
        systemComponentVulnerabilyWrappersList[i];

    pos = loadFromJSON.systemComponentPositionsList[i];
    LoadObject(loadFromJSON.systemComponentTypesList[i]);
    SystemComponent targetComponent = target.GetComponent<
        SystemComponent>();
    targetComponent.componentType = loadFromJSON.
        systemComponentTypesList[i];
    targetComponent.isEntryPoint = loadFromJSON.isEntryPointList[i
        ];

    targetComponent.securityLevel = loadFromJSON.
        systemComponentSecurityLevelsList[i];
    targetComponent.componentName = loadFromJSON.
        systemComponentNamesList[i];

    foreach (var vulnerability in vulnerabilityWrapper.
        vulnerabilityWrapperList)
    {
        targetComponent.componentVulnerabilities.Add(vulnerability
            );
    }

    /// Empty attributes intended for later development cycles
    targetComponent.OS = loadFromJSON.OSList[i];
    targetComponent.subnet = loadFromJSON.subnetList[i];
    targetComponent.configPresent = loadFromJSON.configPresentList
        [i];
    targetComponent.keypair = loadFromJSON.keypairList[i];
    targetComponent.floatingIP = loadFromJSON.floatingIPList[i];
    targetComponent.user = loadFromJSON.usersList[i];
    /////////////////////////////////////////////////////
}
```

Due to Unity3D controlling its own garbage collection, we could not load over scenario with components in it due to the references to objects still being in memory despite us trying to delete everything before loading the desired scenario. This is a problem that only occurred when the editing screen for the scenario creator was not empty upon loading, which would not be a problem as the user would have to load a scenario upon entering the scenario creator.

5.5 Import attacks and defenses from file

We wanted it to be easy to add new attacks and defenses to the game in the future, and decided to use XML for this. C# has built in support for XML through the *System.XML* namespace [22], so we did not need to rely on external libraries. One of the benefits of XML is how easy it is to handle nested and complex data structures. It is also very easy to use both manually and through code. In order to have a built project with working attacks and defenses however, we had to use AssetBundles [23]. This told Unity3D to build with the XML file, and add it to the game files.

In order to access the XML file and load our attacks and defenses, we had to first access the file within the asset bundle, and then parse the XML file as shown in listing 5.5. We first extract the root element for each category within the XML file, then simply loop through each category to extract the values we need and insert them into lists or dictionaries so we can access them later.

Listing 5.5: XML parsing snippet

```
static VulnerabilityLogic()
{
    /// Loads the assetbundle that contains the xml file.
    var assetBundle = AssetBundle.LoadFromFile(Path.Combine(
        Application.streamingAssetsPath, "Assets/AssetBundles/xml"
        ));
    /// Loads the xml file as a text asset
    var asset = assetBundle.LoadAsset("VulnerabilityLogic.xml") as
         TextAsset;

    /// Creates an XmlReader from the loaded asset using string
        reader
    using (var reader = XmlReader.Create(new StringReader(asset.
        text)))
    {
        XElement attackStrings = null;
        XElement defenseStrings = null;
        XElement vulnerabilityPairs = null;

        /// Reads through entire xml file
        while (reader.Read())
        {
            switch (reader.NodeType)
            {
                /// When we encounter a base element, we store
                    them XElement
                case XmlNodeType.Element:
                    if (reader.Name == "_AttackStrings")
                        attackStrings = XElement.ReadFrom(reader)
                            as XElement;
                    else if (reader.Name == "_DefenseStrings")
                        defenseStrings = XElement.ReadFrom(reader)
                             as XElement;
                    else if (reader.Name == "_VulnerabilityPairs")
                        vulnerabilityPairs = XElement.ReadFrom(
                            reader) as XElement;
                    break;
            }
        }
    ...
}
```

5.6 Gameplay

From the initial requirements we knew that the Game would need to be modular. There would be no guarantee that the same `GameObject` would be present during any two scenarios. We decided to go for an event-driven messaging system using event codes and interfaces. When something would happen during gameplay, e.g: The attacker attacks a server, information about that would be packaged into a message, which gets broadcast to every object who implemented the interface for that event. This meant that every `GameObject` would implement interfaces for the events they want to listen to, and broadcast any message that other `GameObjects` need to listen to. This made every object self contained with no direct communication between them.

5.6.1 Events

All events in the game have their unique unsigned short value. Event types are categorized in static classes where each category has its own offset set to the second byte. We have Logging-events, Network-events and Game-events. Logging has an offset of 0, network has an offset of 256, and game has an offset of 512 as shown in listing 5.6. We used hexadecimal to set the offset to make it easier to increment the byte by one, instead of using normal decimal and having to add 256 each time. There is no real benefit of doing it either way however. The offsets gives us a limit of 256 events per event type. All message types are set in unsigned short to reduce size (2 bytes vs 4 with uint), as we do not need millions of different event codes. With unsigned short, we can have a maximum of 65'536 different events.

Listing 5.6: Static class containing game event codes.

```
public static class Game
{
    private const ushort offset = 0x0200;

    public const ushort Start = offset + 1;

    public const ushort Attack = offset + 2;
    public const ushort AttackResponse = offset + 3;
    public const ushort Defense = offset + 4;
    public const ushort DefenseResponse = offset + 5;
    public const ushort Discover = offset + 6;
    public const ushort DiscoverResponse = offset + 7;
    public const ushort Analyze = offset + 8;
    public const ushort AnalyzeResponse = offset + 9;
    public const ushort Probe = offset + 10;
    public const ushort ProbeResponse = offset + 11;

    public const ushort SaveFile = offset + 12;
}
```

5.6.2 Messages

Every event sent between `GameObjects`, and over network, has its data packaged into a message. A standard message, as shown in listing 5.7, contains name of the sender, the target (or receiver), the message type (or event type), and the player type of the sender. There are messages that need more info than a standard message. These are derived from the `Message` class and have added variables to them.

Listing 5.7: Base Message class.

```
public class Message
{
    public string targetName;
    public string senderName;

    public ushort messageType;

    public PlayerManager.PlayerType playerType;

    /// <summary>
    /// Default constructor
    /// </summary>
    /// <param name="target">Target of the message</param>
    /// <param name="sender">Sender of the message</param>
    /// <param name="type">Message type</param>
    public Message(string target, string sender, ushort type)
    {
        targetName = target;
        senderName = sender;
        messageType = type;

        playerType = Object.FindObjectOfType<NetworkingManager>().
            playerType;
    }
}
```

5.6.3 Messaging Manager

We implemented a custom messaging system to handle interaction between `GameObjects` within the game. This is done through the messaging manager. The messaging manager takes a message, and sends it to everyone that listens to the specific event stored in that message. This is done by finding every `MonoBehaviour` [24], and using `Linq` [25] we only get those who implement the interface for that event. Then we simply loop through all the listeners and call their function, sending with the message in the parameter seen in listing 5.8. The `BroadcastMessage` function takes the base class message in as a parameter. Any message that isn't the base class gets cast to the base class, and if the function called (`OnDiscover` in listing 5.8 as an example), we simply cast it back to what it was with no information lost because of it.

Most events have extra function calls for the observer. This is because the observer logs the game events and also needs to send these messages to all clients currently connected. This is in order to update them with the same information.

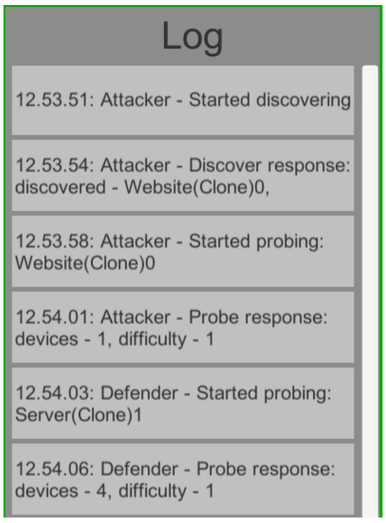
Listing 5.8: Code snippet from the messaging manager showing code run on a *Discover* event

```
public static void BroadcastMessage(Message message, int index =
    -1)
{
    ...
    switch (message.messageType)
    {
        ...
        case MessageTypes.Game.Discover:
            {
                var objects = Object.FindObjectsOfType<
                    MonoBehaviour>().OfType<IDiscover>();
                foreach (var o in objects)
                    o.OnDiscover((DiscoverMessage) message);

                if (_NetworkingManager.playerType == PlayerManager
                    .PlayerType.Observer)
                {
                    LogSaving.Save((DiscoverMessage)message);
                    _NetworkingManager.sb.BroadcastMessage(message
                        );
                }
                break;
            }
        ...
    }
}
```

5.6.4 Logging

The product owner requested logs on the events that happened during gameplay. This was due to future work which would involve machine learning and AI that would use this data in order to play the game. The log files are split between attacker and defender and are written in the YAML [26] format. An example of this can be seen in listing 5.9. The observer also gets the logs on screen. However, these are more verbose as well as formatted differently as seen in fig. 12.



```
12.53.51: Attacker - Started discovering
12.53.54: Attacker - Discover response: discovered - Website(Clone)0,
12.53.58: Attacker - Started probing: Website(Clone)0
12.54.01: Attacker - Probe response: devices - 1, difficulty - 1
12.54.03: Defender - Started probing: Server(Clone)1
12.54.06: Defender - Probe response: devices - 4, difficulty - 1
```

Figure 12: Log visible to the observer during gameplay

Listing 5.9: Example of a log file in YAML

```
Event:

  - Event_type: Discover
        From: Attacker
        To: Root
        Probability: 0,8

  - Event_type: Discover Response
        From: Website(Clone)0
        To: Attacker
        Discovered: Website(Clone)0

  - Event_type: Probe
        From: Attacker
        To: Website(Clone)0

  - Event_type: Probe Response
        From: Website(Clone)0
        To: Attacker
        Difficulty: 1
        Child nodes: 1
        Number of Vulnerabilities: 0
```

5.6.5 Gameplay UI

The Gameplay UI is split into 3 screens; the attacker, defender and the observer. For these different views we created an abstract [27] base class that contains references to all UI related `GameObjects`, as well as both abstract functions and normal, public functions. The abstract functions are required to be overridden by the derived classes while the inherited functions can either be used normally, or replaced by using the *new* keyword [28] if different code is needed. This is because we have other scripts that call UI functions, and having a base class with those functions makes it easier. Everything that needs a reference to the UI gets the current UI script from the player manager as it's base class, and calls functions available from there.

Every player type (attacker, defender, observer) has their own UI script tailoring the UI and its functionalities based on their needs. For example: The attacker and defender both have on-click menus when right-clicking on a node to show options on what they can do. The observer doesn't need this and thus doesn't have that. The observer, however, does have more information displayed about nodes than both the attacker and defender, and thus has different code than the other two. Every player type uses tooltips though, which is shared between all derived classes through the base class.

For displaying tooltips we made a separate script that is attached to any `GameObject` that has an existing UI element attached to it, and that needs to have a tooltip. It uses Unity3D's built-in pointer handlers to enable and disable tooltips on enter and exit, as shown in listing 5.10. This will display a tooltip message next to the object that is hovered over. It sends the string as a parameter to the UI script, as well as it's position and `RectTransform` [29] to correctly offset it next to the object hover over.

Listing 5.10: PointerHandler script used for tooltips.

```
public class PointerHandler : MonoBehaviour, IPointerEnterHandler,
    IPointerExitHandler
{
    public string tooltipInfo;

    private BaseUI uiScript;

    ...
    public void Start()
    {
        uiScript = FindObjectOfType<PlayerManager>().GetUIScript()
            ;
    }

    ...
    public void OnPointerEnter(PointerEventData eventData)
    {
        uiScript.EnableToolTip(tooltipInfo, transform.position,
            GetComponent<RectTransform>().rect);
    }

    ...
    public void OnPointerExit(PointerEventData eventData)
    {
        uiScript.DisableToolTip();
    }
}
```

5.7 Networking

There was some discussion about using a dedicated server vs hosting through the local network, but we ended up using a local multiplayer for this project. Since our gameplay had the observer mode which would not be acting upon the system, we concluded that this would be the best place to put the server of the system.

All clients send a message through the network to the server which responds by sending a message back to all other clients in the game that needs this new information. The server will always have a full overview of the important information in the game and receive the new information when events happen.

5.7.1 Server

When a player has started a lobby as a host they will be listening to all incoming messages from any IP in the local network through a `TcpListener` [30]. When the client first asks if this specific host has available spots in the lobby a message also containing the name of the lobby is sent over to give the client a user-friendly view of that specific game.

Listening to incoming calls requires the thread's attention for a longer period, which made us utilize separate threads for different purposes. This way the host can listen to potential new clients while still being able to interact with the UI. The standard for working with `Tasks` and `CancellationTokens` can be found in the .NET framework [31], and was used to keep different actions for the network behaviours on their own threads listing 5.11.

The server needs to have control over all clients that are connected. This is done with a `NativeList` for holding the connections through the `UdpCNetworkDriver` from Unity3D's new networking system [32], and a list for keeping control of each players id with names for the lobby seen in the listing 5.11.

Listing 5.11: The main attributes of a Server. They are used to keep control over UI, connections, new clients and threads.

```
[Header("Reference for manager")]
NetworkingManager nm;

[Header("Core attributes for connection and sending gamedata")
    ]
public UdpCNetworkDriver m_ServerDriver;
public NativeList<NetworkConnection> m_connections;

[Header("Listener for clients to find host")]
TcpListener server;

[Header("List for clients in lobby/game")]
private List<(string name, int connectionNumber)>
    connectionNames;

[Header("Attributes to keep multiplayer on its own thread to
    not interrupt user interaction with the game")]
private Task findConnections;
private CancellationTokenSource cancellationTokenSource;
private CancellationToken cancellationToken;
```

When the game has started the server will mostly be listening to incoming messages sent from its connected clients and act upon them depending on their message type. With many different types of messages we could quickly set them up by dynamically getting their type and broadcasting them through our `MessagingManager` as seen in listing 5.12.

Listing 5.12: Code for server to loop through the connections, pop messageEvents and handle the messages received based on their event type.

```
for (int i = 0; i < m_connections.Length; ++i)
{
    NetworkEvent.Type cmd;
    /// Pop all events for the connection
    while ((cmd = m_ServerDriver.PopEventForConnection(
        m_connections[i], out DataStreamReader strm)) !=
        NetworkEvent.Type.Empty)
    {
        if (cmd == NetworkEvent.Type.Connect)
        {
            Debug.Log("Server - Connectevent");
        }
        if (cmd == NetworkEvent.Type.Data)
        {
            /// A DataStreamReader.Context is required to keep
                track of current read position since
                DataStreamReader is immutable
            var readerCtx = default(DataStreamReader.Context);
            byte[] bytes = strm.ReadBytesAsArray(ref readerCtx,
                strm.Length);
            string data = Encoding.ASCII.GetString(bytes);
            if (data.Contains("|"))
            {
                var str = data.Split('|');
                var type = Type.GetType(str[1]);
                dynamic msg = JsonUtility.FromJson(str[0], type);

                MessagingManager.BroadcastMessage((Message) msg, i
                    );
            }
        }
        else if (cmd == NetworkEvent.Type.Disconnect)
        {
            DisconnectClient(i);

            if (i >= m_connections.Length)
                break;
        }
    }
}
```

5.7.2 Client

Just like the server, the client has functions in the network that requires it's own thread for listening functionality. This gives the user an option to stop listening for new lobbies to be found on the local network and start hosting a new lobby.

When players want to join a host they will initially start a thread that is looking for currently available lobbies on the local network. If lobbies are found they will be displayed as buttons in a list on the right side as seen in fig. 13. These are also saved in a `List` for further use as seen in listing 5.13 named `lobbyPairs`.

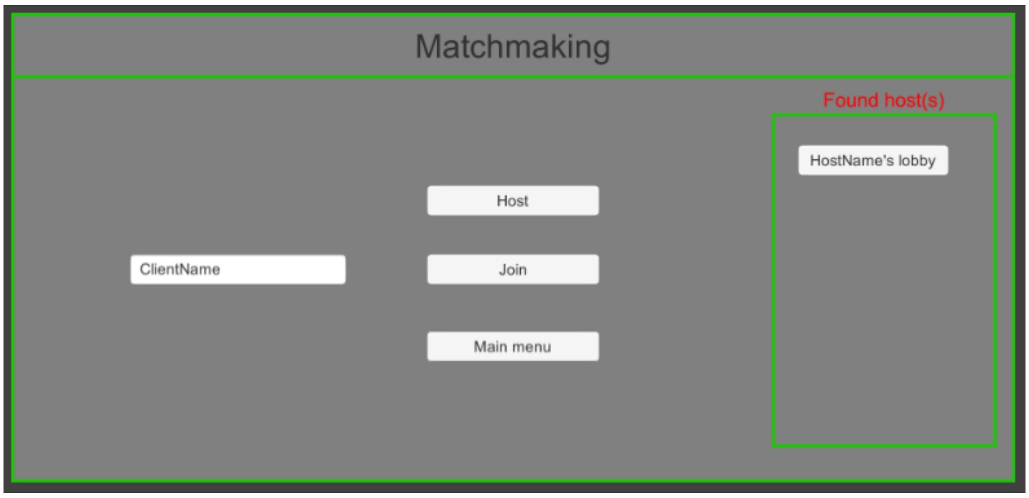


Figure 13: Matchmaking screen. Lobby field on right side has 1 lobby shown that was found on the local network.

The client has the option of swapping position with a player in the other team in the lobby. For this functionality the server needed a list of names for everyone in the lobby as seen in listing 5.11 named `connectionNames`. This option gives both the client wanting to swap and the player they wants to swap with a timed countdown before the swap disappears. To make this behaviour synchronized we made enums (see listing 5.13) to help keep the status correct when sending different messages across the network. The code in listing 5.14 shows how a client will start a swap by sending a `SwapMessage` with the request type to start a new swap, setting the current status to $SwapInfo.Waiting$ as found in listing 5.13 and then wait for the swap to get either accepted or declined. If it was accepted a final message will be sent back about swap being acknowledged and completed.

Listing 5.13: The main attributes of a Client. They are used to keep control over UI, ips, serverconnection, a dictionary for lobbies tasks. Enums to keep swap status.

```
private NetworkingManager nm;

///  results contain the ipaddresses possible to connect to.
readonly List<string[]> results = new List<string[]>();

public static NetworkEndPoint ServerEndPoint { get; private
    set; }

public UdpCNetworkDriver m_ClientDriver;
public NetworkConnection m_clientToServerConnection;
private bool m_clientWantsConnection;

/// <summary>
/// Task to look for a host on the network, tokens to cancel
    tasks.
/// </summary>
private Task setupHostIp;
private CancellationTokenSource cancellationTokenSource;
private CancellationToken cancellationToken;

private Dictionary<string, string> lobbyPairs = new Dictionary
    <string, string>();


/// <summary>
/// Keeps info about current swap(Request, Accept, Decline,
    Acknowledge)
/// </summary>
private SwapInfo swapInfo;

public enum SwapInfo
{
    Null,
    Waiting,
    Accepted
}

public enum SwapMsgType
{
    Request,
    Accept,
    Decline,
    Acknowledge
}
```

Listing 5.14: SendSwapMessage sends a swapmessage to request swap. SendMessage shows how that message is sent.

```
/// <summary>
/// Sends message to client with that player wants to swap
    position with.
/// </summary>
public void SendSwapMessage(string target)
{
    swapInfo = SwapInfo.Waiting;

    SendMessage(new SwapMessage(target, nm.userName, MessageTypes.
        Network.Swap, SwapMsgType.Request));
}

/// <summary>
/// SendMessage will send message about actions happening on the
    current client, and server will send information about this to
     other clients.
/// </summary>
/// <param name="msg"></param>
public void SendMessage(dynamic msg)
{
    if (m_clientToServerConnection.IsCreated)
    {
        var str = JsonUtility.ToJson(msg);
        str = str + "|" + msg.GetType();

        var writer = new DataStreamWriter(str.Length, Allocator.
            Temp);

        writer.Write(Encoding.ASCII.GetBytes(str));
        m_ClientDriver.Send(m_clientToServerConnection, writer);
        writer.Dispose();
    }
}
```

6 Deployment

Our project will be deployed in the internal systems for the Norwegian Cyber Range. This will include the source code, as well as an installer made using Inno setup [33], a free installer for windows programs. Inno Setup Compiler uses a custom scripting language in order to build and compress a program into an installer. It has an easy to follow wizard which we used to create the initial script. The only extra we needed to add to the script was to include created scenarios to the installer as these were not added automatically. Once the script was ready, we needed to compile it and then run it to create the installer.

6.1 Repository

The source files for this project can be found in the following GitLab repository:
http://prod3.imt.hig.no/LordXyroz/cyber-security-simulator.

This repository contains only the source code and Unity3D metadata files, as the complete code for running the system includes a lot of Unity3D code which we do not change, and which would be much too large. Most of these files are included in the *.gitignore* file (http://prod3.imt.hig.no/LordXyroz/cyber-security-simulator/blob/master/.gitignore), and contain files such as visual studio files, files built by visual studio, auto generated VS/MD/-Consulo solution and project files, build files, user-specific files, different packages and plugins.

7 Testing and User Feedback

7.1 Testing during development

Due to the project heavily relying on visual elements, we decided to not utilize automated tests as the data might be right while the visuals were off, and vice versa. This would make automated testing both difficult and time consuming, as there were many factors that would need to be considered, mostly due to the difficulties of writing automated tests for visual elements in the program. In order for the system to be operational both the visuals and the data must be correct. This led to the decision of relying on manual testing during the development process for this project, as well as manual testing and user testing after a functional version of the project was ready.

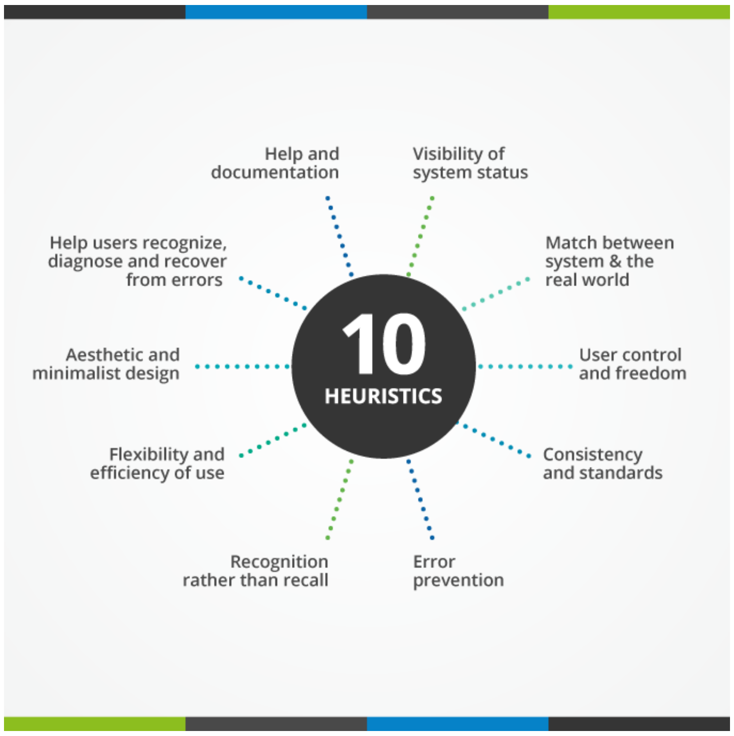
7.2 User testing

As user testing came quite late in the development, we decided to focus on cost- and time effective methods of conducting user tests and to improve the usability of the system. As we would not have the time to implement new functionality, this was put in the backlog to be discussed later for future work.

After consulting with our supervisor as well as a Master's student working on design we chose to use heuristic evaluation and cognitive walkthrough, as well as to utilize the opportunity we had gotten to test our system with our target audience at the Norwegian Cyber Security Challenge.

7.2.1 Heuristic evaluation

The Heuristic evaluation method is a method for testing the user interface (UI) of the system, and focuses on finding usability problems using a holistic view. After identifying a problem, they are given a severity level (low, medium, high, critical) and categorized based on what heuristic they infringe, or what recognized usability principle they do not comply with [34]. While there are several models for Heuristic evaluations, one of the most known ones and the one we were recommended to use was the one made by Jacob Nielsen, based on his previous model made together with Rolf Molich [1], with the 10 Heuristics seen in fig. 14.


10
HEURISTICS
Help and documentation
Visibility of system status
Help users recognize, diagnose and recover from errors
Match between system & the real world
Aesthetic and minimalist design
User control and freedom
Flexibility and efficiency of use
Consistency and standards
Recognition rather than recall
Error prevention

Source: https://www.invespcro.com/blog/heuristic-evaluation-your-complete-guide/

Figure 14: Nielsen's 10 user interface design Heuristics [1]

.

While one practice is to use these heuristics as a guideline to establish design-specific heuristics, we decided against this as most of our problems were with general design principles which these 10 heuristics cover very well. Examples of the problems we found were having the asterisk on the right side of mandatory input fields, giving a warning if the user tries to exit without saving and to be able to exit or return from every stage of the system. We fixed some of these problems in the system, but unfortunately we did not have time to fix everything.

The results of the heuristic tests can be seen in appendix D.

7.2.2 Cognitive walkthrough

A cognitive walkthrough is an evaluation method where a user new to the system is asked to perform a specific task or a set of tasks while thinking out loud, or narrating their train of thoughts [35]. During these tests, the user should not get any feedback or help on their progress, the observer is only to take notes until the user finishes (or gives up on) the task.

After a cognitive walkthrough is completed with a subject, a discussion ensues to be able to collect as much information as possible and relevant improvements are made to the system. It is a good idea to perform cognitive walkthroughs with different subjects but the same task or set of tasks, with changes being done to the system between each subject to make sure that the overall usability of the system is improving.

As this was to be a short walkthrough, the steps we wanted the subjects to do was to:

- Enter the scenario creator.
- Load a scenario.

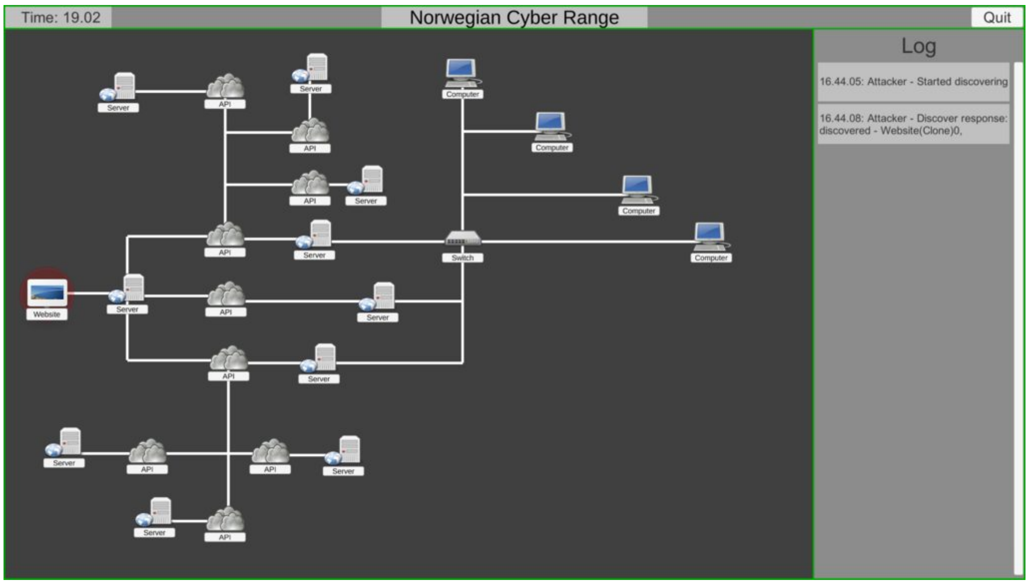
- Add a new component to the topology.
- Connect the component to the system.
- Add two vulnerabilities.
- Make the component an entry point.
- Rename the component.
- Save the scenario with a new name.
- Exit to the main menu.

After watching the subject perform this set of tasks, as well as having a discussion about it afterwards, the feedback was that the interface was easy and intuitive to use. The subject did not struggle with any of the steps, the only deviation was that the subject tried to drag the component button into the scene instead of clicking it.

7.2.3 Norwegian Cyber Security Challenge 2019

The Norwegian Cyber Security Challenge 2019 [13] was hosted by the Norwegian Cyber Range, which gave us a good opportunity to perform user tests on participants here. Our system was used as one of the tasks in the challenge in order to judge their reasoning and decision making when trying to exploit an unfamiliar system. The topology and details for the scenario was designed by the NCR, as they would use this system to evaluate the participants together with an interview. In order to make the evaluation as fair as possible, there would be no defender playing against the participants. They would all play the role of attacker and would act upon the system without active opposition.

The plan was to have a system that would be hard to solve within the given time period, even without an opponent, so the system had to have some size. At the same time it should not be too large, as this is not meant to be an impossible task, simply a difficult one. The objective for this task was to discover and compromise one of the computer components in the scenario within the time limit of 20 minutes.



```
16.44.05: Attacker - Started discovering

16.44.08: Attacker - Discover response:
discovered - Website(Clone)0,
```

Figure 15: The topology for the scenario used in the Norwegian Cyber Security Challenge

We performed this test with 25 participants who used our system before answering a short questionnaire about their thoughts on the system and how easy it was to use. As this test and questionnaire covered both our project and another done by a member of the NCR, many of the questions on the questionnaire were not relevant to us. What we were most interested in were the usability of the game (rated 1-10) and the input field for feedback and improvements to the system. It was also of interest to us to see how realistic (rated 1-10) they felt the scenario and the experience was. The results of this questionnaire can be found in its entirety in appendix E.

Of the 25 participants, only one did not finish the scenario within the given time. Most of the ones who finished used 10-15 minutes to complete the scenario, with a few people completing it in less time. From observing the participants using the system, given that they had never seen it before and never used it, they seemed to grasp the mechanics quickly. It was interesting to see that almost all of the participants started to upgrade their tools before progressing past the second node. Another interesting thing we noticed was that many of the participants started using the breadth-first method, before changing to depth-first halfway through.

The evaluator for the Cyber Challenge was pleased with our system and how it effectively can give information of what strategies the different individuals utilize on a superficial level, as well as helping to identify who the potential risk takers are. As each decision requires both time and resources, each participant had to evaluate the risk of using time and resources to upgrade their tools vs trying to progress without upgrading.

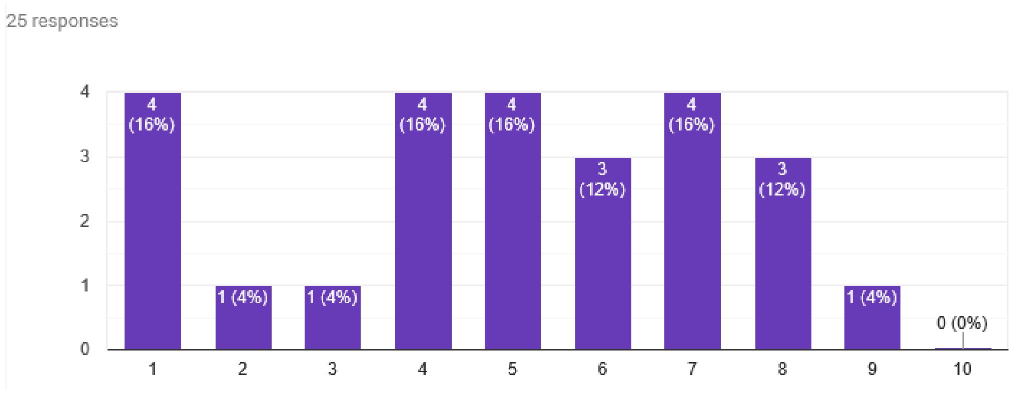


Figure 16: How the users rated the overall usability of the game, from appendix E

From fig. 16 we saw that some of the users found our system hard to use. From the written feedback (see appendix E) we discovered that this was due to a lacking explanation from our part. The thing most of the complaints on usability mentioned was that they used some time to understand that right-clicking a component gives you options for that component. As we considered this an intuitive thing due to the standard methods of interacting with graphical interfaces we did not mention this to the participants. In retrospect, this is something we should have mentioned, but it is not something we wish to change in the system as the confusion was with the explanation and not the system itself.

8 Discussion

8.1 Development decisions

There were a lot of development decisions made, both before and during development. This section covers some of the more influential decisions that were made.

8.1.1 Choice of development environment

As previously stated in section 4.4.1, the product owner requested that we make this project using the Unity3D game engine with the Unity3D framework, as this is a prototype and there are plans to expand upon it later. This expansion will most likely be done by other students, and as Unity3D is the most used game engine in the programming course (at the time of writing this thesis), the product owner wanted to use Unity3D in order for the next group to more easily pick up the project at a later date.

Other game engines like Unreal Engine and Godot were initially considered as alternatives to using the Unity3D game engine, but were disregarded due to the wishes of the product owner.

The process for creating a 2D UI system in Unreal Engine is not that different from making the UI elements in Unity3D, but due to the connection system when using blueprints, the visual scripting representation used in Unreal Engine [36], this could very quickly become messy. This is both due to all the connections and due to the system having an arbitrary number of elements with an arbitrary number of components. This would also change the programming language to C++, but as the team has more experience using C++ than C# this would not have been a problem.

As an alternative to these big, popular engines we could have used an open source engine like Godot. It seems to be quite easy to get into Godot, which would have been an advantage would we have gone with this option [37]. While it appears that doing UI in the Godot engine does not look to be difficult, it would quickly become more complicated with the amount of details and components we need for some of our UI elements. Additionally, since Godot is much smaller than the other engines we looked at, it would be more time-consuming to find help/solutions regarding issues with the engine since the user- and knowledge base is smaller.

8.1.2 Attacks and defenses from a file

In our first few iterations the attacks and defenses, as well as which defense stops which attack, were hard-coded to make development easier. We changed this later in the development to be read from a file to make it easier to add new attacks and defenses, and link which defense mitigates which attack. This was done in XML format [22] in order to have a well structured document where we added the values we need. We didn't have time to implement everything regarding an attack or defense in the XML document, only the displayed string values and which defenses stop which attacks are stored.

The hardest part with making attacks and defenses modular is finding a good way of keeping a good type safety. We could use strings instead of an enum for attacks and defenses, but it's prone to be mistyped. This could create problems and exceptions during gameplay which might result in crashes.

Unity3D has a feature called Editor Scripting [38] where you can create code which will run and automate processes in the editor. Using this feature you could create functionality where you'd create a new attack or defense, and it would automatically check for syntax and make sure everything works as intended. This could then be extended to create a custom editor window where you'd create and manage attacks and defenses during runtime.

8.1.3 Events and messaging system

We knew from the start that our project had to be modular. One of the difficulties of this was to find a way where we could have the individual `GameObjects` interact with each other. Since a big part of the project was the scenario creator, and user-generated scenarios, there was no way of predicting which network component would be present during a scenario, let alone how many of each. This led us to take a look at different ways of doing an event- or messaging system.

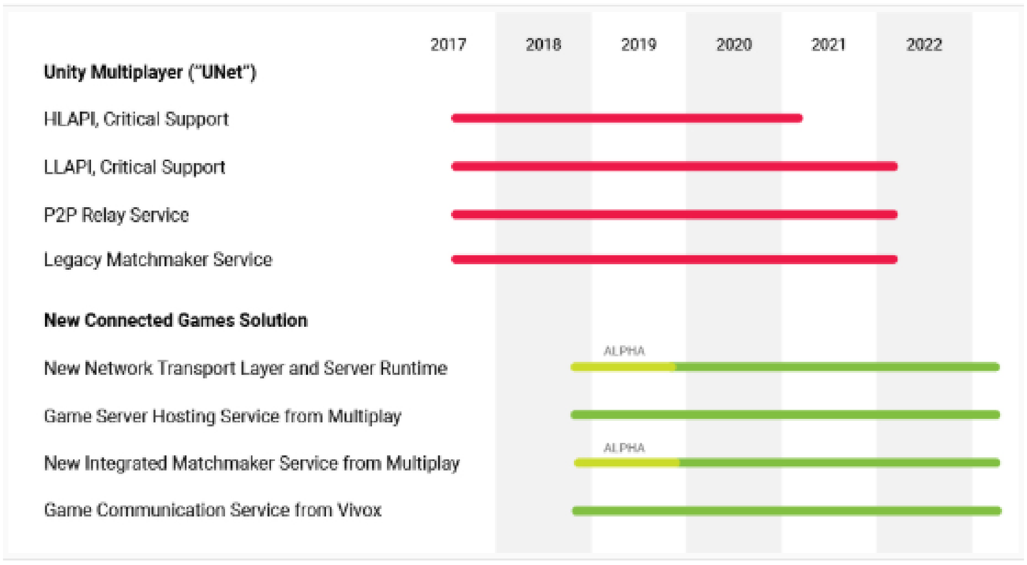
One way to tackle messaging was to use the built-in messaging that Unity3D has through `GameObject.SendMessage` [39] and `GameObject.BroadcastMessage` [40]. The benefit of having something already existing was that we wouldn't need to implement our own version. This would call the function specified as a parameter and call every function with that name that exist on the `GameObject` through `SendMessage`, or this `GameObject` and all children through `BroadcastMessage`. The downside to this was that it didn't fit our need to have interaction between different `GameObjects`. It also uses reflection [41], which is slower than accessing variables and methods directly [42].

Another way we looked at was using callbacks [43]. This would allow each script to tell an event manager to call a function when x happens, and allow for great modularity as each object would subscribe on creation. The downside though, is that we would have to manage and remove callbacks as objects would get destroyed, something that requires a lot of work and is prone to errors.

In the end we settled for using a messaging system based on event codes and interfaces. Classes that wanted to listen to a particularly event have to implement an interface, and the messaging manager would then find every instance of that interface and call the function provided by it. This made the system very modular as we could simply add event types and interfaces without needing to add a lot of boilerplate code. The biggest downside, however, is that it uses `FindObjectOfType` [44] which can be slow in a scene with many objects. A better way of doing it would be managing all objects ourselves, but this requires a substantial amount of work and code, and we did not consider this a necessary part for this project.

8.1.4 Networking

We had originally planned on using Unity3D's networking API, UNet to implement the networking part of our game. However, after some research we learned that UNet was deprecated and would start losing support in 2021 (see fig. 17). After discussing the situation with the product owner and supervisor, we decided to use the new Unity3D multiplayer features for our system. This is largely due to the fact that our system would likely be used and further built upon later, so it would not make sense to use a multiplayer system that would lose support after a few years.



Source: `https://support.unity3d.com/hc/en-us/articles/360001252086-UNet-Deprecation-FAQ`

Figure 17: Timetable of when UNet will lose support

.

Since Unity3D's new networking features was in alpha at the beginning of the project we had little to no explanation on how to use different features. It was too early for the community to have shared enough experience for us to utilize, so we ended up with a lot of trial and error. The time spent experimenting here could possibly have been shortened by not using Unity3D's networking but rather work with .NET networking. Creating networking behaviours using asynchronous sockets could be a simpler way to go. One simple example we could expand from can be found here: [45]. In the example they are using sockets to connect through `ProtocolType.Tcp`, then sending a message before they get a response. For our project we would need to separate these for the client to be able to get responses from the server that an action happened from another client while clients still being able to send messages through the server themselves.

Early on we discussed whether or not to have a dedicated server set up for the project. One of the good things about having that would be growth, as we could expand the game much more. Instead of having games only played over local network we could have games played over continents, giving the game a bigger potential. With the local network finding lobbies has to be done through looking for IP-addresses on the local network before you try to connect to them. The way we do this makes it possible that you not necessarily find the host you want. There is an insecurity regarding it sometimes finding the host and other times not, so one might need to directly ping the IP-address of the host before connecting to it in the game.

On the other side there were a couple of reasons for not using this kind of server. One being that our supervisor with more experience in this field advised us to go for local network first as this was within the requirement we received from the project owner early on in the project. Also considering our experience working with this prior to the bachelor, we decided to use local networking. Using a server that the school already possess could make this option cheaper but nevertheless more expensive than using local network. [46] Unity3D started working with Multiplay in 2017, Unity3D's new networking will use Multiplay features, which will according to their words deliver:

"resilient and scalable dedicated game servers hosted through optimized multi-cloud and bare metal infrastructure"[47].

Using Multiplay features will not be an option for us during the time we have through the bachelor project, as this feature is in alpha during the project period we can't be certain that we would achieve the result we wanted for this project. We would also need to apply for access to these features with no given specification given in advance of whether we get the access or not, and if we get it within time needed to be capable of finishing the project.

One major problem we stumbled upon was when we were transmitting the level data to the clients upon them connecting to the server. The server had no problem sending the data to the clients, only the clients wouldn't accept the packet. After digging through Unity3D's networking code we found a constant called "MTU" that dictated the maximum size of receiving packets. This was originally set to 1400, meaning 1400 bytes of data per packet could be sent, which our level data exceeded. Since the maximum packet size for IPv4 TCP is at 64 KB [48], we changed the constant to reflect that size. It does lower performance since lower network layers still divide the packets into smaller chunks, but since we do not need to send a lot of data this was not a major problem for us.

After working a couple of weeks with networking we could end up with behaviour constituting networking that was really difficult to understand. After testing and debugging we understood that the problem was with the tasks not being cancelled at all. Adding a `cancellationtoken` and `tokensource` fixed this problem [31]. When a thread in the background is not finished/completed a program may experience code being run at times it definitely shouldn't have. We had to spend time making sure that whatever action the user took the system would end every thread and piece of code correctly, and at correct times so no unexpected action will be taken that comes from somewhere else than the clients themselves playing the game.

Developing functionality that uses networking, which can experience unknown and unexpected exceptions, made our choice of using try/catch more relevant than if/else. The major tasks these functionalities use can experience many different exceptions, which could need to be handled differently. For low exception throwing using try/catch could actually be faster than having if/else as it does not need the extra check like the if-statement, and for several occasions when different exceptions needs to be handled it looks cleaner and is easier to read using a try/catch. Looking at the code example in listing 8.1 we can see that the if check can be run without problems, but if the file is removed or deleted elsewhere before the `File.Delete` is run in this code-snippet we would get an exception; So for this specific case we would **need** to have a try/catch as the other option could end with the program crashing. There were also snippets of code where an if statement would make it easier to read and understand without giving a single comment about it, see example in listing 8.2.

Listing 8.1: Checking if file exists before running some code and delete file.

```
if(File.Exists("file.txt"))
{
    /// Some other code...
  File.Delete("file.txt");
}
```

Listing 8.2: try/catch vs if/else for destroying an object in-game.

```
try
{
    Destroy(Canvas.transform.Find("SwapLoadingScreen").gameObject)
        ;
}
catch (Exception e)
{
    /// Will get Exception message about object being null.
    Debug.Log(e);
}

/// VS

GameObject loadingScreen = Canvas.transform.Find("
    SwapLoadingScreen").gameObject;
if (loadingScreen != null)
{
    Destroy(loadingScreen);
}
```

8.1.5 Real-time vs turn based

An important decision was to choose between real-time and turn based gameplay. The product owner initially wanted a real-time system, but as many of the initial requirements were broadly defined and open for changes, we decided to research this question some more by looking at other similar games. We could not find any other released game with a similar concept that utilized a real-time system, all the games we found used a turn based system. We discussed this with the supervisor, and he agreed that a turn-based system would make more sense for the project we are developing. This would make the players evaluate the situation and their options more instead of making impulsive decisions, as well as being able to go through the transpired scenario later. This would enable a greater learning experience, in both our and the supervisor's opinion, as this would enable the users to point out exactly what steps were done right and what steps were done wrong. After presenting this to the product owner, however, he still requested a real-time system as this would add an innovative element to the project, leading to the decision of making the system real-time.

8.1.6 Connection lines

One of the bigger challenges to the scenario creator was how to represent the connection between system components (see section 5.3.2). The initial plan was to visualize the connection lines using the `LineRenderer` Component from Unity3D. The `LineRenderer` takes an array of two or more points in 3D space, and draws a straight line between each one [49].

This would work with our initial plan of simply visualizing the connection between the objects without doing operations on them. However, this proved to not be the case for long as we did not find a good way to delete the connecting line with all corresponding references during runtime using `LineRenderer`. We also discovered that the connection line should have an attribute stating whether or not there is a firewall present, as well as a visual representation of this as seen in fig. 4. These two factors led to us disregarding the original plan of using the `LineRenderer` as it does not provide all the functionality we require.

The connections were originally intended to just be a straight line connecting the two relevant components. After some testing, we concluded that this was a bad approach, as it looked and felt quite messy. We looked to other similar projects for inspiration, like ThreatGEN [9] and Uplink [10], and after making some visual mock-ups concluded that having angled lines instead of straight ones would look much better for our system. The firewall on a connection is automatically placed on the longest of the connecting lines in order to improve visibility (see fig. 18.

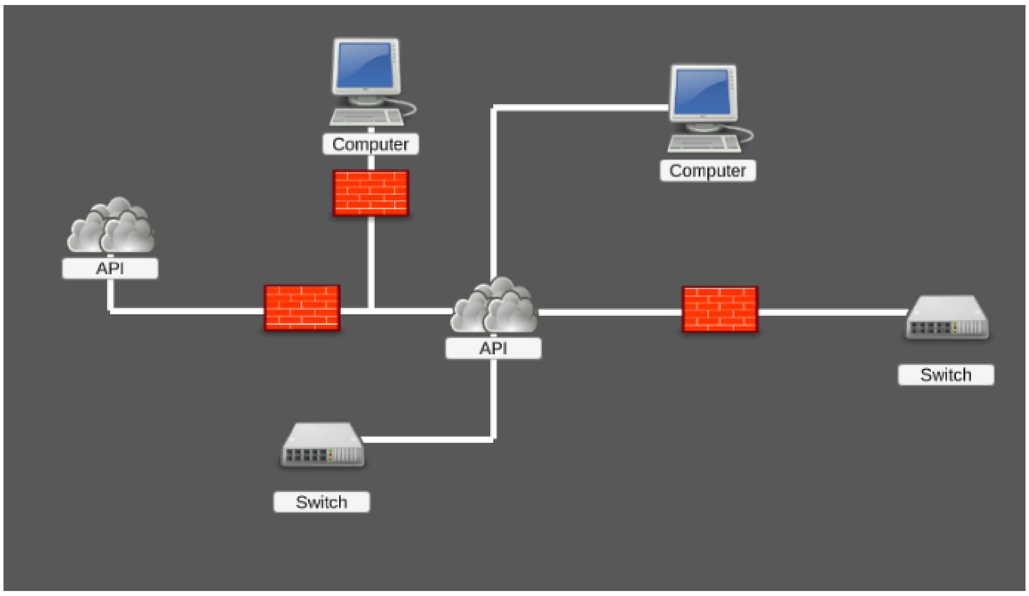


Figure 18: Connections to a single system component, with firewalls

After some discussion, we concluded that the best way to do this was to have two separate boxes represent the connection line and save this shape as a `prefab`, and scale the boxes in order to have them match the shape of a line. This made us able to have a corner on the line, save attributes to it as well as making it selectable and thus easliy deletable.

We considered functionality to choose which side of the system component the connection line would originate from, including if the user wanted the connection line to originate from a corner. After looking at some visual mock ups and considering this, we disregarded this idea as we felt it would end up looking quite messy. As we went for a minimalist design, the clear lines going directly out from the center of the system component was more aligned with this and looked much cleaner.

8.2 System design

As none of us are designers, we chose to go for a minimalist look from the start, and experimented with different ways of achieving this. The current design of the system is the result of making several visual mock-ups and prototypes for how the system could look, and consulting with the product owner based on these.

8.2.1 Specifications

The specifications did not provide much details for the design of the system and how it should look. The product owner showed us some screenshots from other existing games that were very different from each other, ranging from the interface based system of Uplink [10] and the mobile based interface of Hackers [50] to the SIM like style of CyberCIEGE [51]. These provided us with some inspiration based on a few of their details and gave us a starting point to work from.

8.2.2 Choice of colors

There are a lot of people that have some form of color blindness, and the most common of them is red-green color blindness [52]. While our game does not currently mix that many colors, it is still important to design colors and patterns to make it easier for those with color blindness. In a game like this that is heavily reliant on UI interaction, there should be an option to change the color theme based on your color blindness. We did not implement this because we focused on the technical aspect of the game, but for a full version we would like to see this feature implemented. The fig. 19 shows how a red-green color blind person might see our game.

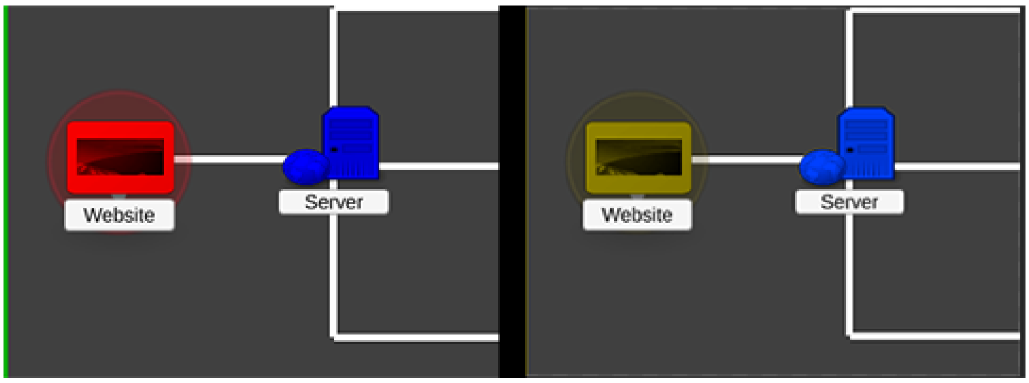


Figure 19: The different colors between normal (left) and red-green color blindness (right)

8.2.3 Similar projects

Our main source of inspiration from similar projects came from ThreatGEN [9], as well as details from Uplink [10]. We were hoping to research ThreatGen closer as we liked the initial look we saw from the steam screenshots. Unfortunately the initial release time for January did not happen for this game, and as of the delivery-date of this thesis it is still not released.

8.3 Changes during development

8.3.1 Requirements

There were a lot of changes during the development of the system. The area that received the most changes were the requirements since the initial requirements were very broad, and due to the fact that the plan for the finished project shifted from being a separate system to be an integrated part of the NCR toolbox.

The level of abstraction

As stated in the project description (section 1.1), the goal of this system is to train decision making. This is a very high-level system, where the details behind how an attack is mitigated or executed is abstracted away and the focus is on the action to take, rather than how to perform it. After some meetings and prototypes, we thought we had a pretty good idea of what was required of us and what we were supposed to create, including the level of abstraction. After working with this for a while, it appeared we had not abstracted it enough according to what the product owner had originally planned. This was solved by making some changes to the models we had designed and several more meetings in order to make sure that we were on the same page regarding the abstraction level.

Later in the development process, when several others working with the NCR were included in some of our meetings, it quickly became apparent that they all had their own idea of how detailed and abstract the system should be. As this was relatively late in the implementation process, and these ideas mostly conflicted with each other and the model we had made we had to decline making these changes. We were open to suggestions should they agree on what changes they wanted, but as this did not happen, the second iteration of the level of abstraction and the last updated version that was agreed upon between the team and the product owner was used.

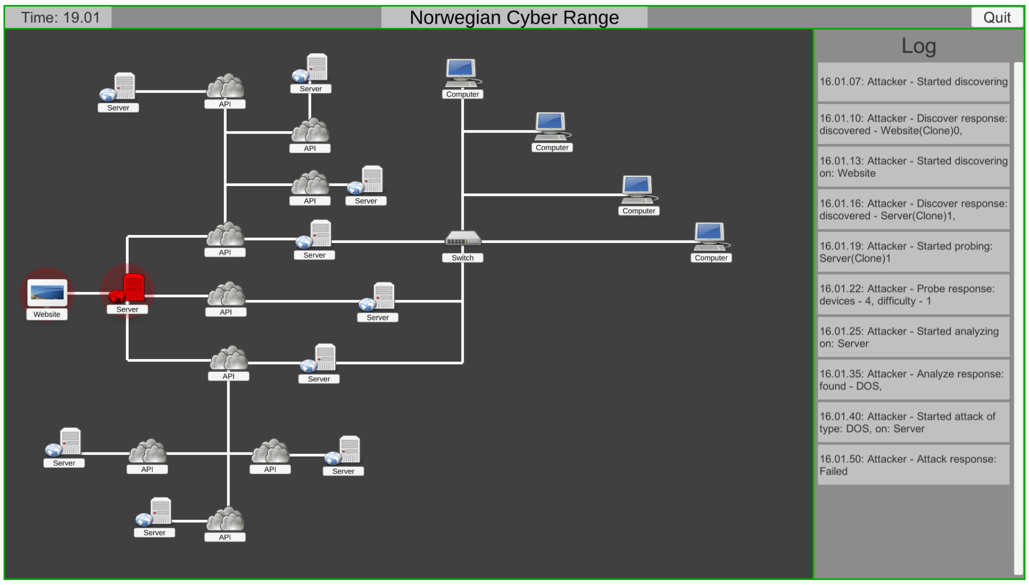
Additional scenes

From the start the plan was to make two different scenes in addition to the main menu, these were the gameplay scene and the editor scene for the scenario creator. This was discussed with the product owner while planning and designing the solution, as well as shown on a prototype of how the finished system would operate.

Around half-way through the development period, some of the participants in our meetings working with the NCR requested that we include several more modes. This included general modes like an observer mode, a very high-level game mode designed to educate hospital personnel, a very low-level game mode for people experienced in cyber security and various other ideas.

After a brief discussion it was clear that some of these would require a lot of extra work, and would not be feasible to include in the time period we had. This included the game mode designed to educate hospital personnel, which would have to be an entirely new system and the low-level game mode for people experiences in cyber security as the system here would have to be vastly expanded upon to include the necessary functionality. As the overall system was built with the purpose of training decision making for people with cyber security knowledge in an attacker vs. defender mode we found no easy way to convert this to the requested modes. In order to fit this for training hospital personnel in secure conduct at the workplace, much of the scenario creator and almost all of the gameplay would have to be reworked to suit these needs. For the low-level mode, much more information and functionality would have to be implemented in order to support the larger number of options and consequences this entails.

Some of the suggestions, however, were suited to be implemented into our system, and thus were discussed whether we should use time and resources on this or not. One of the modes we decided to include was the observer mode. It made sense to have an observer mode in the system we were creating as it would give the one leading the scenario insight in what the teams were doing at a given time. It was also relatively easy to implement compared to other suggestions, as it shared many characteristics with the attacker and defender scene resulting in it not having to be made from scratch as well as not requiring much extra functionality. Once the decision to implement an observer mode was made, as well as the decision to have this as a separate screen for the one running the scenario we started discussing how we wanted it to reflect in the game. The initial idea was to have an event log displayed while the attacker and defender were playing, but after a while we decided to also graphically display the system topology of the scenario. This visual representation also includes the "location" of the attacker and defender based on what nodes they have selected at any given time. The observer also had the option to inspect a node, in order to see what vulnerabilities it has as well as the mitigations the defender has implemented.



16.01.07: Attacker - Started discovering

16.01.10: Attacker - Discover response: discovered - Website(Clone)0,

16.01.13: Attacker - Started discovering on: Website

16.01.16: Attacker - Discover response: discovered - Server(Clone)1,

16.01.19: Attacker - Started probing: Server(Clone)1

16.01.22: Attacker - Probe response: devices - 4, difficulty - 1

16.01.25: Attacker - Started analyzing on: Server

16.01.35: Attacker - Analyze response: found - DOS,

16.01.40: Attacker - Started attack of type: DOS, on: Server

16.01.50: Attacker - Attack response: Failed

Figure 20: The observer mode

In addition to this, a fourth scene had to be made. This was the matchmaking lobby for creating a game, where the desired scenario would be loaded and the defender and attacker would connect to a game hosted by the observer. The decision to host on the observer machine was made shortly after the decision to implement an observer mode was made. This would help mitigate cheating in the system as none of the players would act as the host, as well as making it easier to collect the logging system as everything happening in the game would go through the observer.

Dynamic attack/defense creation

During the initial prototyping we had a discussion between the group members of how we wanted to implement the different attacks and defenses. We ended up using enums as it would be easier to ensure the different types were correct when used for the game logic, as well as ensuring smaller messages would have to be sent.

During a meeting with the product owner at the end of the final sprint he wished for dynamic creation of attacks/defenses during run-time. Since the core game-logic revolved around using enums, and there only were 2 days until the planned development process was to be finished, we decided we would look into it but could not promise to be able to implement it in time.

One of the ways for implementation we looked at was changing enums to using strings instead. This would make it easy for the user to customize specific attacks and defenses, but would make it easy for user error and misspelling and thus could cause errors during gameplay, corrupting the files and making the scenario unplayable. It would have taken time to develop checks that would ensure concurrency between strings and rewrite the game-logic using this. As the game logic depends on having attack-defense pairings, making an attack would quickly lead to internal errors unless defense types were specified upon attack creation. This would require logic to ensure the defense types were made before the attacks, as well as paired correctly. In addition to this, the networking messaging system can only send packages of up to 64KB of data [48]. This is not a problem with enums and predefined attacks, defenses and pairs where only the enum itself has to be sent, but with dynamically made attacks and defenses the entire string containing the information for the attack or defense as well as the pair data relevant for it would have to be sent. This would lead to us having to find and implement another method of sending the game data in order to create the game scenario upon start. We therefore decided to not implement this and focus our attention on fixing known errors and other parts of the game instead.

Expanding the scenario creator

As the scenario creator was originally meant to just handle the information used in the gameplay system we were creating, it did not originally have as much information as the finished product has. This is reflected in our design as well, an example of this is all the attributes for system components that are present in the logic but not editable through the interface in the scenario creator (see fig. 3). The attributes are still present in the logic of the system, and will be saved along with the attributes we use in our system, but will have to be set manually until the interface has been expanded to edit these as well.

8.3.2 Tools

As we were working with the development of the project, we used a Kanban board [12] in order to keep track of who were doing what tasks, what had to be done and what was yet to be done. This worked well for a time, but slipped out for us as we moved from the development phase and moved into tracking and fixing bugs. This was partially due to the fact that we were working right next to each other for the entire period, and as we discussed what bugs to focus on we also orally decided who would try to fix what bugs and were usually informed when the bug was fixed. We also started using discord for tracking the discovered bugs that had yet to be fixed as this was more accessible to us.

8.3.3 Logging

As the product owner requested event logs for training an AI, we created a simple way of logging data to the screen for the observer. This consisted of time, event type, who sent it and who received it, as well as extra details regarding the event where applicable. Initially this was only sent to the observer and displayed in-game in a list, but the product owner wanted it to be on file to make it easier to use for training. Later we got notified that this data should be in a specific format, namely YAML [26]. This required us to change a lot about how the logging worked. We decided not to change the logging to screen since this was a nice way to know what was going on during gameplay, we only removed the saving of those logs to file and instead saved the YAML format to files.

We had to change quite a bit about our logging. This resulted in creating an entire class to handle logging to file. We needed to divide the logs between attacker and defender, and format the strings that were to be saved in YAML format. We could have used external libraries to handle the formatting for us, but many of the C# libraries were outdated. On the asset store for Unity3D there exists a library for YAML serialization (YAMLDotNet for Unity [53]), but at the time, learning an external library would take more time than just hard-coding the format.

8.4 Feedback

8.4.1 Testing

Performing the heuristic evaluations helped us identify both some minor and major bugs as well as inconsistencies and quality of life improvements we could make to improve the system's usability. Several of these fixes were easily done, like increasing the text size and checking the color of the text contrast compared to the background. We used ContrastChecker [54] in order to make sure the text is clearly visible, also for people with visual impairments like color blindness. There were also things we discovered that we did not have time to implement, like a help button, visual cue for when a connection is initiated and have the system components clicked in place when spawned rather than just spawning it in a given location.

The cognitive tests did not provide that many results as we did not have time to perform them with different sets of instructions or for different parts of the system. This was only done for the scenario creator, but according to our test subjects the controls were intuitive and easy to use.

During the user testing at the Norwegian Cyber Challenge, one of the things we got the most feedback on regarding our system was that it was not very intuitive for several of the participants to have to right-click on a node in the system in order to bring up an action menu. As we thought this was an intuitive action, we did not explain this in advance. In retrospect, it would have been a good idea to have a quick demo for each group before they started in order to familiarize them with the controls of the system.

8.4.2 NCR

We have generally received positive feedback from the NCR during the development of this project. They have continuously wished for new functionalities during the development process and have requested changes, but have been understanding of the time constraint we are facing. Especially after running the game as a part of the Norwegian Cyber Security Challenge 2019 [13] we were met with positive feedback from the jury and the NCR team, with talk of continuing to use the system for different cyber security challenges in Norway and maybe abroad. There is also talk of extending our system to the Norwegian Defense Forces.

9 Future work

As this was designed to be a prototype, a lot of future work has been planned and discussed. As it is likely that another group of students will pick up this project at a later date, an extensive listing of possible future work has been included in this chapter.

9.1 General work for the system

Exit warning: One of the general things that can be added is to have a warning prompt for all quit/exit buttons, asking the user if they really want to quit.

Online deployment: The system could be deployed on a cloud service/dedicated server. This would be the next step from simply having it as a local multiplayer. It could also be hosted on a website, requiring the users to simply enter the web page instead of downloading the application in advance.

Lower level gameplay: The functionality of the system could be expanded to support a lower level system with more technical details. This would require adding a lot of functionality to both the gameplay and the scenario creator, however. This could include component specific information like internet ports and OS computers are running on. It could also include privilege levels for both the attacker and defender (guest/user/admin).

Help button: A help prompt/button could be implemented, that would give the user a list of basic operations available for the current scene and answers to some frequently asked questions (FAQ).

Support for colorblindness: As there are a lot of colors used in this game to represent the different teams, as well as progress on actions and other indicators, it might be an idea to implement a color blind mode. This could be an option toggled on/off in the settings menu.

9.2 Scenario creator

Scrollable map: One of the functionalities we have considered for the scenario creator is to have a scrollable map, where the user can change the map position they view within the editing window. This would in theory expand the map to an arbitrary size as the map would then not have a physical limit, the only limit being the view window showing how much of the map you can see at a given time.

Edit connection lines: Another functionality we considered making was editing options for the connection lines. This would include adding more corners to the connecting lines and moving the positions of the lines manually.

Exit warning: One other thing that could be added is to have a warning if you are trying to exit the scenario creator if you have unsaved changes.

Visualize a started connection: Have a functionality for visualizing when a connection is in progress could be implemented. One method of doing this is to have a connection line follow the cursor after a connection has been initiated, and stop following the mouse if the a connection is established or the connection process was canceled.

Spawning components on click: When spawning a component at the present moment, the system component will be spawned at a certain location within the editing screen. This leads to all new components being spawned on top of each other, which could lead to confusion. A solution to this could be to have a semi-transparent icon of the component follow the mouse, and be "spawned" at the location clicked within the editing screen, or removed if the user clicks outside of the editing field.

9.3 Gameplay

Win/lose condition: The most important functionality for gameplay at this point is to add a clearly defined win/lose condition for the attacker and defender. As we did not have time to implement this, the game only stops by itself now if the timer runs out.

Attacker capabilities: There aren't that many different options for the attacker to take during gameplay at this point. We would've liked to improve this by adding more options. One option would be a social engineering aspect. The attacker has to make contact with insiders to get information, and discover different entrance points to the network. Another option would be interactive attacks. Having the attacker do mini games or even sudo coding to have something to do while a progress bar fills.

Defender capabilities: As with the attacker, the defender also has only a few options to do during gameplay. One option would be adding a way to train the "users" of network equipment to decrease security risks. Another option would be to let the defender add firewalls and other defense options in the network as well as physical items like cameras and security card readers. Having the defender doing mini games as well is another idea which would improve the game.

Objectives and score: Related to this is the functionality to add objectives and scores. A part of having a proper gameplay in place is to have a scoring system. This should be based on time used, resources remaining as well as specified objectives for the attacker and the defender.

Expand multiplayer: Another functionality that could be added in the future is to have more players in the game at the same time. This would be best achieved by adding several players for each team, as opposed to the current system where each team only supports one player.

Gameplay AI: In order to be able to play through scenarios alone, this system would require a gameplay AI. This is already being planned as the product owner stated from the start that this would be a later project.

Wait notification: In the matchmaking lobby, the system could give players a sign that they are waiting for host to start game. New users might be confused as they have no action to start the gameplay, as this can only be done by the observer mode.

Event system: Have an event system in place, where certain triggers will cause certain events to transpire. This could be a resource gain for the attacker if certain system components are exploited, system changes at given times or other similar events.

Attacks and defenses entirely from file: One improvement point is having attack and defense options be read entirely from a file. Currently to add a new option you would need to add it to an enum, create a prefab, and add vulnerability pairing and strings to an XML in order to create a new type. One way of changing this is to add cost, duration and description to the XML file, as shown in listing 9.1, and read them during the initial loading of the game.

UI improvements: The UI itself is very basic with no textures or animation. This would be one of the main items to improve upon when all core functionality is in place. One way of improving would be to not have a lot of duplicate objects. Currently there are three versions of the UI present in the scene, where one of them gets activated based on the player type. The main difference between these three versions is color, while also having a few smaller changes where the UI displays different things for different player types. These should all be changed through scripting, removing a lot of unnecessary duplicates from the scene. Another thing to add is animations to UI interactions, and textures and icons to the UI objects to make it look more polished.

Gameplay balance and probability: To make the game's probability calculation and balance more realistic, a custom random number generator would need to be implemented. This randomization would then depend on charts and numbers generated through research, and preferably read from file to make changing values easier.

Listing 9.1: Example layout of improved XML file containing gameplay logic.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<GameplayLogic>
  <AttackTypes>
    <AttackType key="Injection">
      <string>Inject malicious code</string>
      <cost>10</cost>
      <duration>10</duration>
      <description>Perform a code injection attack</description>
    </AttackType>
  </AttackTypes>
  <DefenseTypes>
    <DefenseType key="Sanitize_Input">
      <string>Sanitize user input</string>
      <cost>10</cost>
      <duration>10</duration>
      <description>Implement input sanitization</description>
    </DefenseType>
  </DefenseTypes>
  <VulnerabilityPairings>
    <AttackType key="Injection">
      <DefenseType key="Sanitize_Input"/>
    </AttackType>
  </VulnerabilityPairings>
</GameplayLogic>
```

9.4 Observer

Replay functionality: A replay functionality could be added to the observer mode. This would replay a given scenario based on the logs from a certain run of that scenario.

Snapshot: Another functionality to add to the observer mode is to take a snapshot/record the state of the game at a given time. This could later be used for discussion and reflection purposes.

10 Conclusion

This project has given us an insight in how it is to work with an agile method for a product owner in a realistic scenario. This was an actual product the Norwegian Cyber Range wanted to develop and use, as opposed to a theoretical one that would be discarded. This gave us a distinct advantage in the learning experience compared to a project that would simply be made for the sake of making a project. The focus on what we have learned has been on cyber security scenarios and making UI systems using Unity3D. We have also learned about methods for working as a team with a product and a real product owner. This presented us with new challenges compared to a school task, where the tasks are pre-defined and the pre-set requirements are final.

Working with Unity3D's new networking system while it was still in the alpha proved to be a challenge. There was less documentation than what we are used to, as well as less experience from the community regarding solving various issues. It might have been a better idea to use .NET networking, or at least have researched this further before starting to use the new Unity3D networking system, as there is better documentation and testing available.

Our estimation on how long time the implementation of certain elements would take were not as good as we would have hoped. This was in part due to lack of experience in some of the areas we worked on, as well as discovering more features that were needed during the implementation.

We consider the project to have achieved its intended purpose, to be a prototype for a learning and training platform, but there is definitely room for improvement. Nevertheless, the system is for the most part intuitive to use, and was made in a modular and scalable way which should make it easier to add new features and functionality later.

Bibliography

[1] Jakob Nielsen. *Usability Engineering*. Academic Press, 1994.

[2] The OWASP web Security Team. Owasp top 10 web application security risks. https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf, 2017. Last visited Jan. 2019.

[3] Gary Bente Johannes Breuer. Why so serious? on the relation of serious games and learning. *Eludamos Journal for Computer Game Culture*, 4(1):7.24, 2010.

[4] Capture the flag. https://www.cybersecuritychallenge.org.uk/competitions/capture-the-flag, 2019. Last visited Apr. 2019.

[5] Norwegian cyber range. https://www.ntnu.no/web/ncr/start, 2019. Last visited Apr. 2019.

[6] The Unity team. Experienced programmer, but new to unity? you're already ahead of the game. https://unity3d.com/programming-in-unity, 2019. Last visited Mar. 2019.

[7] Don Glover. Unet deprecation faq. https://support.unity3d.com/hc/en-us/articles/360001252086-UNet-Deprecation-FAQ, 2018. Last visited Feb. 2019.

[8] Osa open security architecture. http://www.opensecurityarchitecture.org/cms/. Last visited Apr. 2019.

[9] Threatgen: Red vs. blue. https://store.steampowered.com/app/994670/ThreatGEN_Red_vs_Blue/. Last visited Apr. 2019.

[10] Uplink. https://store.steampowered.com/app/1510/Uplink/, 2006. Last visited Apr. 2019.

[11] Cybersecurity red team versus blue team — main differences explained. https://securitytrails.com/blog/cybersecurity-red-blue-team, 2018. Last visited May 2019.

[12] Inc. Planview. What is a kanban board? https://www.planview.com/resources/articles/what-is-kanban-board/, 2019. Last visited Apr. 2019.

[13] Norwegian cyber security challenge 2019. https://www.ntnu.no/ncsc, 2019. Last visited Apr. 2019.

[14] European cyber security challenge. https://www.europeancybersecuritychallenge.eu/, 2019. Last visited Apr. 2019.

[15] Doxygen. http://www.doxygen.nl/, 2019. Last visited Apr. 2019.

[16] Overleaf, about us. https://www.overleaf.com/about. Last visited Apr. 2019.

[17] The Unity team. Running unity on linux through wine. https://wiki.unity3d.com/index.php/Running_Unity_on_Linux_through_Wine, 2019. Last visited Apr. 2019.

[18] Porter. Unity: Now you're thinking with components. https://gamedevelopment.tutsplus.com/articles/unity-now-youre-thinking-with-components--gamedev-12492, 2013. Last visited Apr. 2019.

[19] Creating and using scripts. https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html, 2018. Last visited Apr. 2019.

[20] Rotation and orientation in unity. https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html, 2019. Last visited May 2019.

[21] Jsonutility. https://docs.unity3d.com/ScriptReference/JsonUtility.html, 2019. Last visited May 2019.

[22] System.xml namespace. https://docs.microsoft.com/en-us/dotnet/api/system.xml?view=netframework-4.8, 2019. Last visited Apr. 2019.

[23] Assetbundles. https://docs.unity3d.com/2018.3/Documentation/Manual/AssetBundlesIntro.html, 2017. Last visited Apr. 2019.

[24] Monobehaviour. https://docs.unity3d.com/ScriptReference/MonoBehaviour.html, 2019. Last visisted Apr. 2019.

[25] Getting started with linq in c#. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/getting-started-with-linq, 2015. Last visited Apr. 2019.

[26] The official yaml web site. https://yaml.org/, 2016. Last visited Apr. 2019.

[27] abstract (c# reference. https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/abstract, 2015. Last visited Apr. 2019.

[28] new modifier (c# reference). https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/new-modifier, 2015. Last visisted Apr. 2019.

[29] Recttransform. https://docs.unity3d.com/ScriptReference/RectTransform.html, 2019. Last visited Apr. 2019.

[30] Tcplistener class. https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcplistener?view=netframework-4.8, 2019. Last visited Apr. 2019.

[31] Cancellationtoken. https://docs.microsoft.com/en-us/dotnet/api/system.threading.cancellationtoken?view=netframework-4.8. Last visited May 2019.

[32] Unity real-time multiplayer alpha repository. https://github.com/Unity-Technologies/multiplayer, 2019. Last visited Apr. 2019.

[33] Martijn Laan Jordan Russell. Inno setup. http://www.jrsoftware.org/isinfo.php, 1998. Last visited May 2019.

[34] Heuristic evaluation: How to conduct a heuristic evaluation. https://www.interaction-design.org/literature/article/heuristic-evaluation-how-to-conduct-a-heuristic-evaluation, 2019. Last visited Apr. 2019.

[35] How to conduct a cognitive walkthrough. https://www.interaction-design.org/literature/article/how-to-conduct-a-cognitive-walkthrough, 2017. Last visited Apr. 2019.

[36] Tommy Tran. Unreal engine 4 blueprints tutorial. https://www.raywenderlich.com/663-unreal-engine-4-blueprints-tutorial, 2017. Last visited Mar. 2019.

[37] Ariel Manzur Juan Linietsky and the Godot community. Design the gui. https://docs.godotengine.org/en/3.1/getting_started/step_by_step/ui_game_user_interface.html, 2019. Last visited Mar. 2019.

[38] An introduction to editor scripting. https://unity3d.com/learn/tutorials/topics/scripting/introduction-editor-scripting, 2018. Last visited May 2019.

[39] Gameobject.sendmessage. https://docs.unity3d.com/2018.3/Documentation/ScriptReference/GameObject.SendMessage.html, 2018. Last visited Apr. 2019.

[40] Gameobject.broadcastmessage. https://docs.unity3d.com/2018.3/Documentation/ScriptReference/GameObject.BroadcastMessage.html, 2018. Last visited Apr. 2019.

[41] Reflection (c#). https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection, 2015. Last visited Apr. 2019.

[42] Why is reflection slow? https://mattwarren.org/2016/12/14/Why-is-Reflection-slow/, 2016. Last visited Apr. 2019.

[43] Callback (computer programming). https://en.wikipedia.org/wiki/Callback_(computer_programming), 2019. Last visited Apr. 2019.

[44] Object.findobjectsoftype. https://docs.unity3d.com/2018.3/Documentation/ScriptReference/Object.FindObjectsOfType.html, 2018. Last visited Apr. 2019.

[45] Asynchronous client socket example. https://docs.microsoft.com/en-us/dotnet/framework/network-programming/asynchronous-client-socket-example, 2017. Last visited May 2019.

[46] Openstack documentation. https://www.ntnu.no/wiki/display/skyhigh, 2018. Last visited May 2019.

[47] Multiplay - the game server scaling specialists. https://multiplay.com/about/. Last visited May 2019.

[48] Darpa internet program protocol specification. https://tools.ietf.org/html/rfc791, 1981. Last visited Apr. 2019.

[49] Line renderer. https://docs.unity3d.com/Manual/class-LineRenderer.html, 2017. Last visited Apr. 2019.

[50] Hackers. https://play.google.com/store/apps/details?id=com.tricksterarts.hackers&hl=en, 2019. Last visited Apr. 2019.

[51] Cyberciege. https://my.nps.edu/web/c3o/cyberciege, 2004. Last visited Apr. 2019.

[52] Facts about color blindness. https://nei.nih.gov/health/color_blindness/facts_about, 2015. Last visited May 2019.

[53] Yamldotnet for unity. https://assetstore.unity.com/packages/tools/integration/yamldotnet-for-unity-36292, 2016. Last visited May 2019.

[54] Constrast checker main page. https://contrastchecker.com/. Last visited Apr. 2019.

A Project Agreement

This is the agreement for the project between the product owner (Muhammad Mudassar Yamin) and the students (Christian Bråthen Tverberg, Maarten Dijkstra, Nataniel Gåsøy).

This document includes the time period for the entire project (the product and the thesis), as well as details of ownership rights for the finished product.

NTNU

Vår dato Vår referanse

Norges teknisk-naturvitenskapelige universitet

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og *NTNU cyber range* (oppdragsgiver), og *Christian Bråthen Tverberg, Maarten Dijkstra, Nataniel Gåsøy* (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 7.1.2019 til 20.05.2019 .

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): *Mariusz Nowostawski*

Oppdragsgivers kontaktperson (navn): *Muhammad Mudassar Yamin*

NTNU

Vår dato Vår referanse

Norges teknisk-naturvitenskapelige universitet

Student(er) (signatur): [illegible] dato 10.1.2019

[illegible] dato 10.1.2019

Christian Bråthen Tverberg dato 10.1.2019

Oppdragsgiver (signatur): [illegible] dato 11.1.2019

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): [illegible] dato 11.01.2019

B Project Plan

This is the original project plan for this project. This includes the initial requirements for the project, the goals and extent of the project, the group organization and the initial progress plan.

Project plan
CSS
Cyber security simulator

Christian Bråthen Tverberg
Maarten Dijkstra
Nataniel Gåsøy

January 2019

Contents

1 Introduction

This thesis will be about creating a cyber security simulation-type serious game. It will be made for the NTNU cyber range, and is meant to be a tool to help train decision making regarding cyber security threats. It will support multiplayer mode, where one player/team is playing as the defending/secure party and the other player/team is playing as the attacker/malicious hacker.

2 Goals and framework

The main goal is to make a functional simulation that generates scenarios from a scenario generator screen. A scenario will be a certain, specified set of actions both the defending and attacking team can take as well as the system they will be acting upon. After this, the game will start, and the attackers and defender are controlled by the players. There are plans to further develop this by implementing an AI that can control the part the player is not, allowing for single player mode as well as adding levels, but this will not be a part of this bachelor thesis.

The project will be made using the Unity game engine. This was the wish of the product owner in order for future work to be done easier, and this suited us well as this is the game engine we as a team is most familiar with. The programming language used will be C# as this is the main language supported by Unity.

2.1 Background

The background for this project is to develop a tool that the cyber range can use both for training in decision making and to further develop later. As the cyber range is new at NTNU Gjøvik, it requires tools that can be used here. This also corresponded to a bachelor size project, making this a win-win situation. The Norwegian Cyber Range is an area for testing, training and practice for cyber security scenarios. This will mostly be used by a supervisor to create scenarios for students, and use these as educational tools.

2.2 Project goal

The goal is to develop a functioning framework that can generate a scenario based on user specifications. This will include the difficulty level, how many resources each party will have, the initial system that the players will act upon. The game will have two stages, the scenario creation screen and the active gameplay stage where the attacker and defender compete against each other.

2.3 Framework

As stated, the main framework will be the Unity engine. Here we will create a graphical interface for choosing the variables for scenario generation, as well as the "game screens" for the attacking and defending player(s). This will include status and available actions.

The focus of which vulnerabilities that will be accounted for will be based upon the OWASP list of 10 most common cyber security risks. If there is time, the OWASP list for mobile and IoT (internet of things) will also be included, but this is not a priority.

OWASP
Injection
Broken authentication/ session management
Sensitive data exposure
XML external entities (XXE)
Broken access control
Security misconfiguration
Cross-site scripting (XSS)
Insecure deserialization
Using components with known vulnerabilities
Insufficient logging and monitoring

Table 1: OWASP top 10 security risks for web applications [1]

The task was originally to make a real-time cyber security game, it may however make more sense to develop a turn based strategy game with regards to education purposes. This also depends on the size of the team. The original design document specified a 1 vs 1 game, but it may be an idea to have several players on each team. This is something that must be discussed with the product owner and tested out, so it is too early to make a decision on this yet.

3 Extent of project

The project will be a simulation generation tool, as well as having multiplayer functionality to play the generated scenario. It will not support single player mode, as it will not have a player control AI. The generated scenario will have a clear win condition both for the defending and attacking party. This will be a zero-sum game, meaning that one cannot win without the other loosing.

3.1 Field of study

This will be a Bachelor thesis for the bachelor degree in programming, with focus on games. While this will be a game that focuses on cyber security, it will be more focused on the what and not the how. This means that the actual attack and the defense for that kind of attack will be used with focus on decision making, but there will be no closer look as to how certain types of defense mitigates certain types of attack.

This project can be classified as a serious game, as it will focus on practical skills and have an educational purpose.

3.2 Limiting the task (task boundaries)

This thesis will not include a lot of animations. It will mainly be a screen with options and status display, where there will be no characters or classic environment to animate.

There will not be developed a player controller AI for this project. While there may be some AI included for functionality purposes, and to make the player experience less tedious, there will not be an AI in place that is designed to act as an opponent for the player.

In order to not overscope the project, after we have made the framework and the simulation generation tools, we will implement one and one type of threat. We will also implement the appropriate mitigations as these become applicable.

There will not be a deep look at how the different attacks and defenses work, but mainly an overview of certain defenses that can mitigate certain attacks.

The project will be developed for a local multiplayer. This may later be changed to be running on a dedicated server, but this is not a priority. As the Unet multiplayer in unity is deprecated, we will be using the new model for multiplayer.

4 Project organization

The project will mainly be split into 4 parts: **Planning, development, testing** and **documentation**. The planning part is the early stages of the project, where we plan the progression of the project together with the product owner and the supervisor. This includes milestones, deadlines, refining the scope of the project and what the finished product should look like. The development phase is where we expect to spend most of the time, as this will include prototypes, making the actual product and refactoring the product after a certain point. The testing part is expected to be brief, but we will put in more time here as it will also include the time it takes to fix bugs discovered during testing. Documentation will be done throughout the project, but at a certain point in time development and testing will be stopped and the full focus of the group will be on putting together and refining the documentation.

4.1 Team roles

Group leader: Nataniel.
Secretary (notes from meetings): Nataniel

Git manager: Maarten.
Developers: Christian, Nataniel, Maarten.

4.2 Group rules and routines

Rules:

1. Work days: Monday through Friday.
2. Work hours: 9.00 am through 17.00 (with 30 - 60 min lunch break 13.00).
3. If a group member is more than an hour late without giving notification, he owes the other group members a soda/energy drink.
4. Merging parts of the project will be done on one computer only, to avoid merging errors (especially in the meta files).
5. Written code should be commented on a function by function basis.
6. Every file/change should be added and committed individually with a descriptive commit message.

Routines:

- Meetings every Monday in order to discuss what will be done this week. (Time to be determined).
- Meeting Friday to summarize the week.
- Weekly/bi-weekly meeting with supervisor and/or product owner.
- In case of discontent with team members, or other issues, a group meeting will be held to try and solve it. If no solution can be found, a meeting will be organized between the group and the supervisor.
- Before pushing to the git repository, check that the code compiles and is commented.
- When delayed or absent for normal work hours, a message to the other group members will be sent on either Facebook or Discord before 10.00 am the same day, explaining why.

5 Planning, follow up and reporting

This section mostly covers how we plan to work in regards to method and milestones. It will cover the development method we will be utilizing, as well as important milestones we will reach during this project.

5.1 Project structure

We will mainly use the scrum method for working, with daily scrum meeting at the start of the day to keep everyone up to speed on what each individual member is working on. This includes what they did yesterday, what they are doing today and if they are facing a problem they need help to resolve. We are planning to have 2 week sprints, with meetings on fridays at the end of the sprint as well as having mid-sprint meetings the other fridays. This will effectively give us a sprint start meeting every other monday, end of sprint meeting every other friday as well as an update meeting on the friday midway through the sprint.

We think that scrum will benefit us as we will then get a better overview of what the entire team is working on, and thus have a better overview of how the project is coming along. As there may be changes underway due to the loose constraints of the project, a more rigid model like waterfall is ill suited for this project. A more agile model like scrum, on the other hand, is much better suited to handle sudden changes to the requirements of the product.

Gitlab has Issue Boards which we will use to get a better overview over the backlogs, what is under development, bugs found, what is finished, what needs a review and who's working on what part.

5.2 Mile stones

- Created Git repository with read-me.
- Created Unity project and pushed to repository.
- Finished the planning report.
- Visual mock-up of the product, in order to show the product owner.
- First functional prototype of the scenario creator.
- First functional prototype of the gameplay.
- Simple gameplay of a local multiplayer.
- Scenario creator operational.
- Attacker/defender screens.
- Improve gameplay to get a functioning full play model.
- More scenarios.
- Testing and bug fixes (2 weeks).
- Refactoring (done 5.4).
- First report draft (done 12.4).
- Final report finished (done 20.5).

6 Quality assurance

6.1 Documentation and source code

We will have another member of the team review source code while working, as well as properly comment each section of the code written. We will also be taking notes along the entire development process in order to supplement the final document when writing the thesis report.

6.2 Risk analysis (identify, analyze, measures, follow up)

- Merge errors.
- Sick team member(s)
- Crash/Data loss
- Development delays.
- Product owner/supervisor unavailable for a time.
- New Unity version.

Probability/risk	low	medium	high
Highly likely	Product owner/ supervisor unavailable for a time		
Likely	Merge errors/ New Unity version	Sick team member(s)/ development delays	
Unlikely		Crash/data loss	

In order to counteract merge errors, we will be merging everything on the same computer. We considered using Unity collaborate, but as this does not give most of the functionality that git does (branches, commits, commit messages, revert functionality etc.) we opted to not use this.

For sick team members, we will have daily scrums in order to know what each member is doing. This will mitigate a workflow halt should one member get sick and not be able to work on a critical feature without anyone being able to take over.

In order to cope with data loss, it is important that we push to appropriate branches often. This will mitigate the impact should there be local loss of data.

For development delays, it is important to implement key features first. Desired features comes after this, and optional features comes last, after the key and desired features are implemented and functioning.

If the supervisor/product owner becomes unavailable for meetings for a time, contact via e-mail/discord will be used. Should communication be unavailable and an important question arises, Christopher Frantz will be asked as substitute to the supervisor. Should the product owner be unavailable, Basel Katt will be asked as a substitute.

In order to avoid issues with a new Unity version during development, we will not update Unity during the development. This ensures that the Unity version is consistent for the entire project. The version used will be Unity 2018.3.2f1.

6.3 Evaluation

In order to assure that the project goes along with the product owner's wishes, there will be a continuous discussion with him during the development. This will include how we plan to do the project, what to focus on as well as discussions of showcased unfinished product during the development. Based on this, we will ask for feedback and adjust the project accordingly. This will ensure that we keep on the right track based on what the product owner wishes for the final product.

To make sure that we have a functional product, there will be tests during the development phase. This includes tests when we have implemented a new feature as well as when we are doing larger merges. We have also planned a couple of weeks for testing and bug fixes at the end of the development period, before we switch our focus to the final report. Where we will reflect over what went well and why that happened, whether the final product was what we planned or not, if our milestones were met at wanted time and the stuff that should not have happened/could be done better.

At the end of the development time, we will make at least 1 demo, both to showcase the finished product to the product owner. This will also serve to showcase our product for the final presentation of our bachelor project.

7 Progress plan

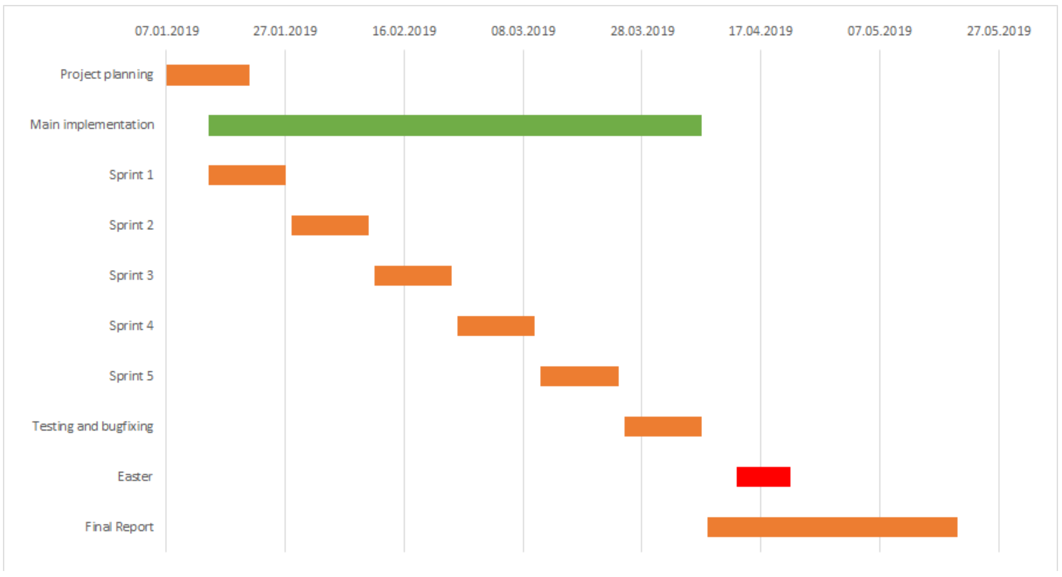


Figure 1: Gantt chart

References

[1] The OWASP web Security Team. Owasp top 10 web application security risks. `https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf`, 2017. Last visited: 2019-01-16.

C YAML example for the NCR DSL

This is an example of the YAML format we were to adapt our system to in order to conform to the DSL. At the time of publishing this thesis, the YAML format for the DSL is outdated as the DSL is updated. This is due to the fact that this was developed at the same time as our system, and while this is not the newest version available it is the version we have worked with during this project.

```
options:
  cloud-platform: heat
  heat:
    authenticator: /path/to/openstack-rc-file

scenario:
  phases: 1                                     # Required - Default=1
  type: attack-defense                          # Required - Possible
values: {generic, jeopardy, attack-defense, Wargame, redteam-vs-blueteam}
  properties:
    name: Fancy-name-of-CTF
    startDate: 01.01.1970
    endDate: 07.01.1970
    startTime: 12:00
    endtime: 23:59
    events:                                     # Events/Injects are
added as a property of a scenario
      ddosStart:
        type: attack                            # Required - Events
should be categorised to enable automation
        description: >
          This is the description that describes the purpose of this
event. This is
          a multiline description.
        activateCondition:                      # Required - Condition
upon which the event is activated
          type: time
          timeafterstart: 01:00:00              # Timestamp formats:
+<timestamp> indicate duration after start. <timestamp> is just
        DeliveryMethod: Puppet
        file: attack-module.pp
      reporting:
        type: report
        description: >
          foobar
        activateCondition:
          type: objectiveComplete
          objective: objectiveXX
        DeliveryMethod: email
        file: null

resources:
  ubuntu-node:                                  # Required - Object name
is used to refer to this object from elsewhere
    type: node                                  # Required - [node,
router, vulnerability, ]
    properties:
      flavor: m1.medium                         # Optional - m1.medium
and m1.small is default for Windows and Linux respectively
      os: Ubuntu Server 18.04 LTS (Bionic Beaver) amd64         #
Required - The OS. Maps to images in OpenStack
      networks:                                 # Required - Can be empty
if Node is not connected to any networks.
        router01:                               # Required - The router
to connect to
          subnet: ctf-lan                       # Required - Comma
separated list e.g subnet: a,b,c
          public_ip: no                         # Optional - Whether or
not the node should have a Floating IP
```

```
          port_security:                        # Required -
            tcp:                                # Optional
            - 22
            - 80
            udp:
            - 53
      role:                                     # Optional -
        name: web                               # Name will be used to
look up corresponding file. Must exist.
      userAccount:                              # Optional - Define User
accounts on the system
        type: employee                          #
        name: John Doe
        groups:                                 # Optional - Which groups
the user should be a member of
          - www-data
          - employees
      vulnerability:                            # Optional - Which
vulnerabilities should the Node have.
        - shellshock                            # Required - Name is used
as ID in DB. If not also declared as a resource, the DB default is used.
        - heartbleed

  shellshock:                                   # Object of type
vulnerability. This definition will override the default
    type: vulnerability
    properties:                                 # The idea here is to
have vulnerability specific properties that can be override what is
default in the DB.
      bash_version: x.y.z                       # Optional

  router01:
    type: router                                # Required
    properties:
      networks:
        ctf-lan:                                # Required
          cidr: 192.168.1.0/24                  # Required
          gatewayIP: 192.168.1.1                # Optional
          routes:                               # Optional
            - 192.168.3.0/24: 10.11.12.13       # Key is dst subnet,
value is nexthop ipaddr
        #ctf-dmz:
        #  cidr: 10.0.1.0/24
        #  gatewayIP: 10.0.1.1
```

D Heuristic tests

These are the results from the heuristic testing. The observer mode was not tested, as there are few things the user can do here.

Heuristic Evaluation Worksheet

Page / Function being evaluated: Attacker view

Heuristic	Comments
Visibility of system status	User don't know what time and cost different upgrades cost.(critical) Component name can't be too long as the whole name won't be seen then.(critical-high)
Match between system and the real world	Display components given name instead of components system name(critical) Use monitary resources instead of GB(High)
User control and freedom	Terminate ongoing process(medium-low)
Consistency and standards	Left click to select, right click to provide options.(critical) Time should always show two decimals, when seconds is 5 it is only showing 1.(high) Right-click menu not closing when another item is selected.(critical)
Error prevention	Current component selected should be called selected, not target(medium)
Recognition rather than recall	
Flexibility and efficiency of use	
Aesthetic and minimalist design	
Help users recognize, diagnose, and recover from errors	Trying to do an action you can't afford(with resources), an error message should show.(critical)
Help and documentation	

Heuristic Evaluation Worksheet

Page / Function being evaluated: Defender view

Heuristic	Comments
Visibility of system status	Make clear what the entry point(s) are.(critical) User don't know what time and cost different upgrades cost.(critical)
Match between system and the real world	
User control and freedom	Terminate ongoing process(medium-low)
Consistency and standards	Right-click menu not closing when another item is selected(critical)
Error prevention	
Recognition rather than recall	Trying to do an action you can't afford(with resources), an error message should show.(critical)
Flexibility and efficiency of use	
Aesthetic and minimalist design	
Help users recognize, diagnose, and recover from errors	Add a help button(critical)
Help and documentation	Clearly state what kind of defense mitigates what kind of attack.(high)

Heuristic Evaluation Worksheet

Page / Function being evaluated: Scenaio Creator

Heuristic	Comments
Visibility of system status	Components are stacked so no longer clear how many components there are (Critical) Maybe show a line while connecting Have a visual cue to show what the entry point is Show number of components on the board Save and quit look like they are under component information
Match between system and the real world	Color associations (Medium) Wording about available components to add components instead
User control and freedom	
Consistency and standards	Consistent sizing of text and button type (Medium) Place required asterisk on the right side of the field Always have a menu or quit button (Critical)
Error prevention	Consistent icon usage with actions Cannot delete only one vulnerability (High) Ask if you want to save before quitting (Critical) Show last save time Color checker for text and background (Medium)
Recognition rather than recall	Either remove tooltips or make it a hover next to the mouse (Critical)
Flexibility and efficiency of use	
Aesthetic and minimalist design	Make it more clear that it is a dialog box and that background buttons cannot be clicked
Help users recognize, diagnose, and recover from errors	Add a help button
Help and documentation	

E Cyber Challenge User Test Results

These are the questions and answers from the questionnaire the 25 participants in the Norwegian Cyber Challenge answered after trying our system for a specific scenario as an attacker, with no active defender.

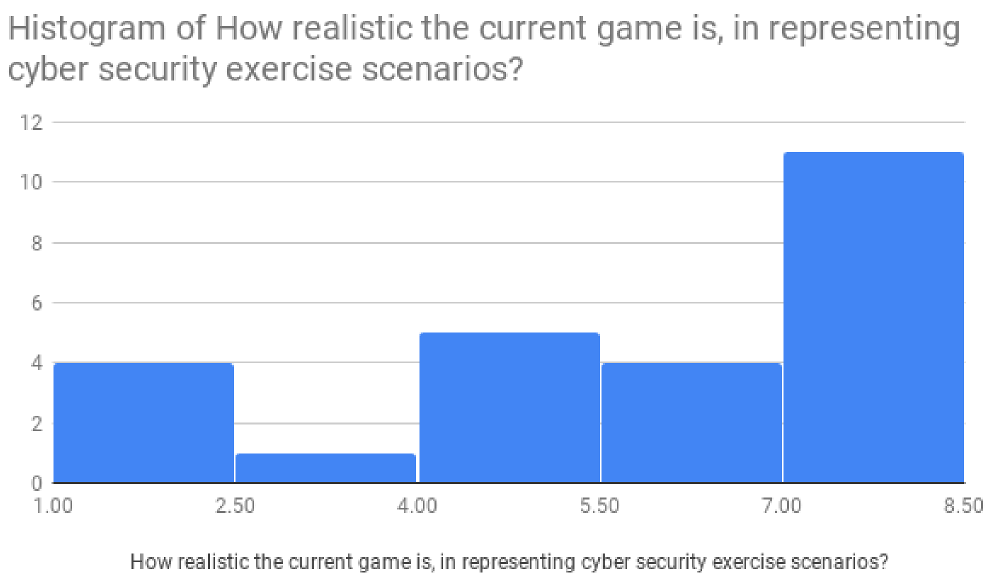


Figure 21: As the question of "How realistic the current game is" is not visible in the pdf, this has been reconstructed using the data.

Cyber Security Strategy Game

QUESTIONS RESPONSES 25

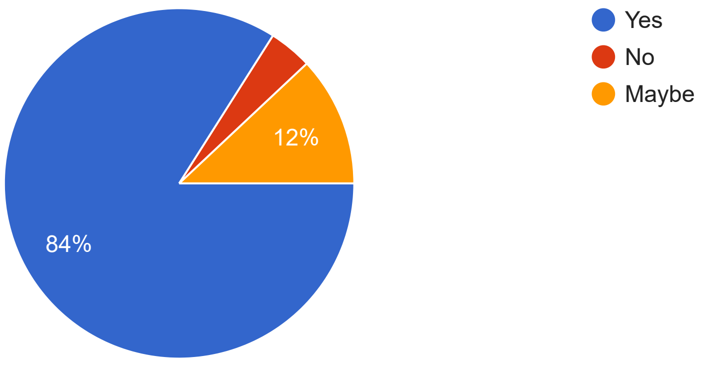
25 responses

SUMMARY INDIVIDUAL

Accepting responses

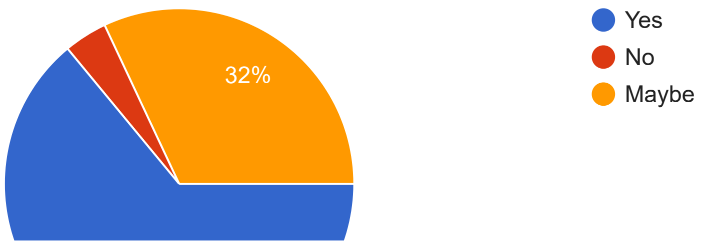
Do you think computer games can help in cyber security education?

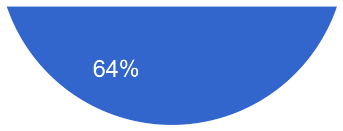
25 responses



Do you think practicing attack strategies in games is useful before launching a real attack?

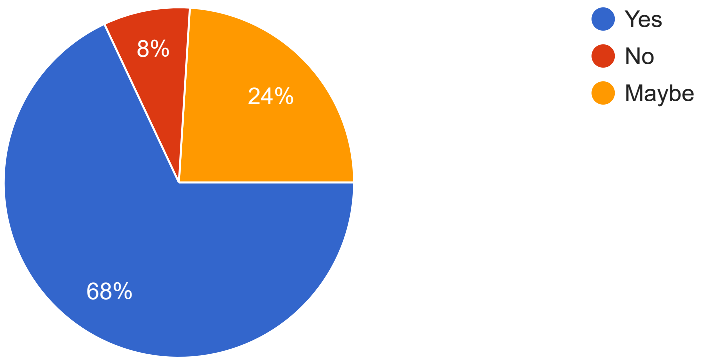
25 responses





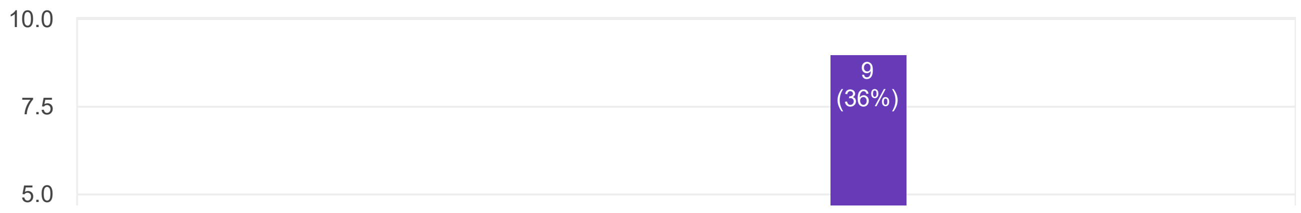
Do you think practicing defense strategies in games is useful before defending against a real attack?

25 responses



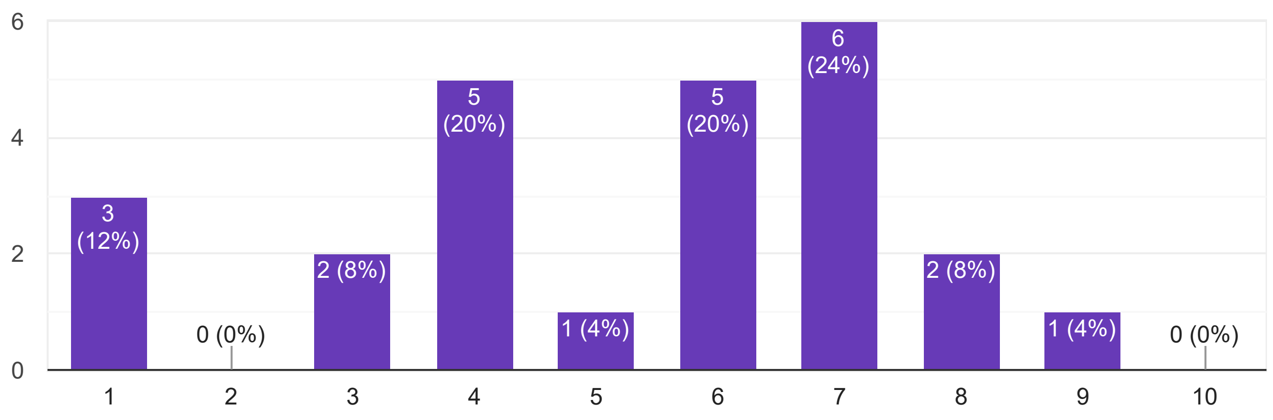
How realistic the current game is, in representing cyber security exercise scenarios?

25 responses



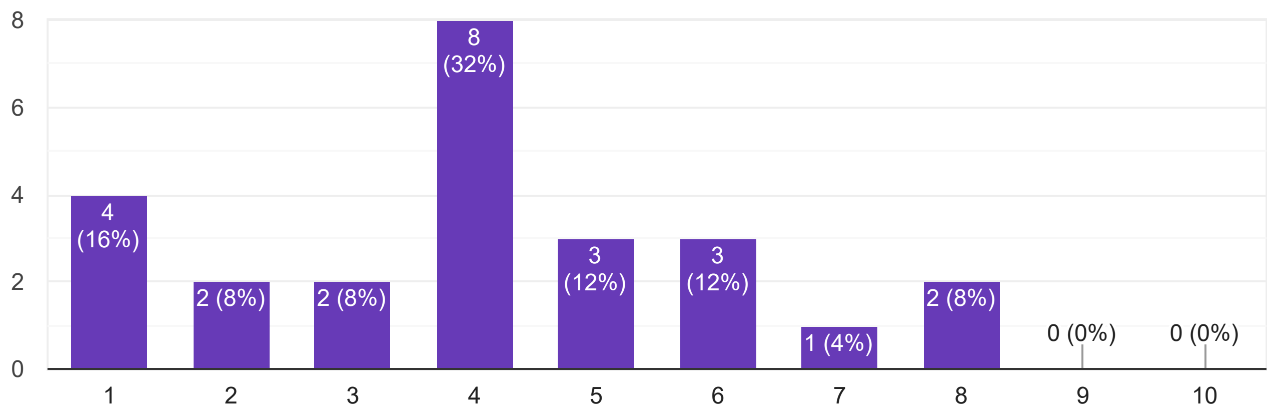
How realistic the current game is in devising cyber attack strategies?

25 responses



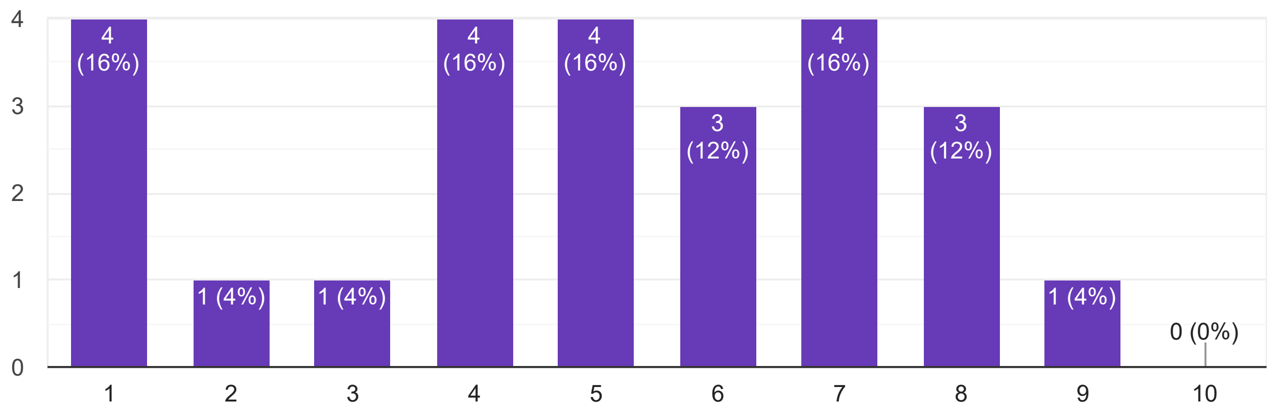
How realistic the current game is in devising cyber defense strategies?

25 responses



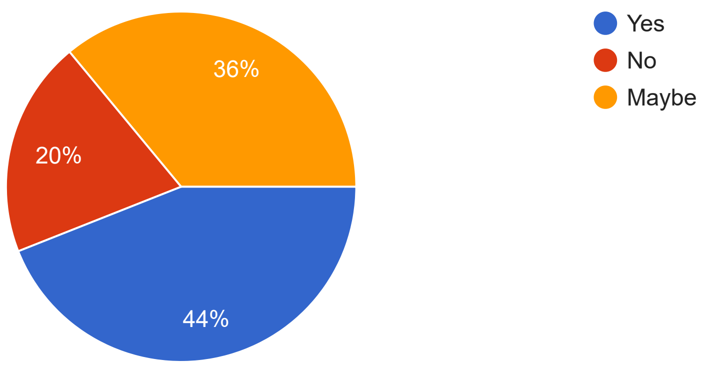
How do you rate the overall usability of the game?

25 responses



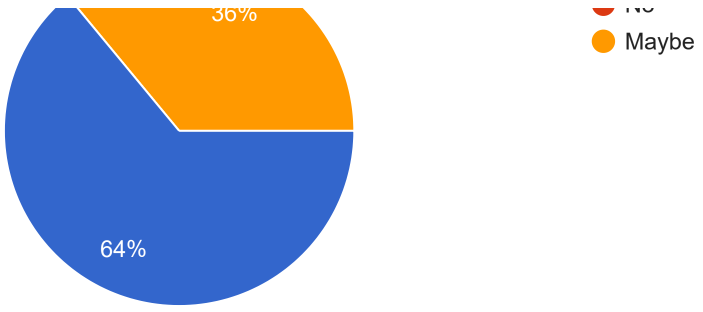
Do you think current game can be useful for cyber security education?

25 responses



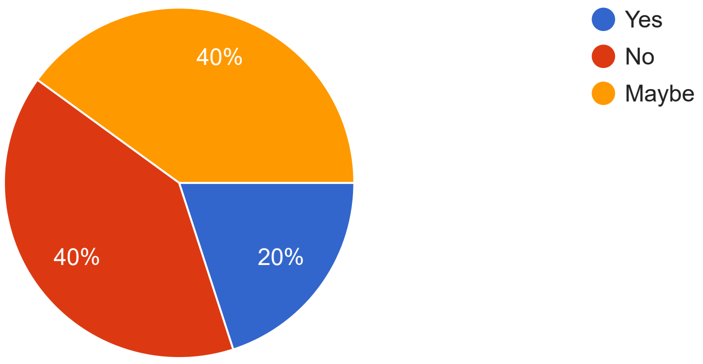
Do you think playing/practicing cyber security exercise scenarios in a simulated/modeled game is an efficient way for conducting cyber security exercises?

25 responses



Do you think your cyber security exercise operational strategy decision making skills are improved after playing this game?

25 responses



What can be improved in the game, do you have any recommendations?

25 responses

No

Some quick instructions like "Right click on a node to perform actions on the node" or a regular button in the GUI that appears when selecting the node.

.

Would be nice if there was some user interactable exploits

Unsure about what to do right before the game

I find it unrealistic that a DoS should help exploit a website/server. It was a bit unclear how to "start" the game, so just doing the initial node discovery for us would be better. Basically giving us "scope". Scope would also be nice to add, so that one fail if one does not follow the scope and scan or attack services outside the given scope.

So the GUI is clunky, you should be able to do multiple things at a time. Also there is no back button. its also really easy to see which hosts to exploit as they get reveled just by the fact that you know there has to be free space, so exploiting the hosts at the bottom is pointless as it cant reveal an adjacent host. Id recommend making the map scrollable to fix that last issue

In the beginning I didn't realize right clicking did something, maybe a quick demo would be good. I was confused by DOS being a way to exploit a server and gain further access, so I waited a bit before using that attack.

Slippe å klikke på x for å fjerne vinduet som dukker opp etter eksemoelvis "attack". (Klikk utenfor vindu for å lukke)

It could be more information

Some more clear instructions, took me a while to realise that you had to right-click on services to discover more actions

Game was good, but give a brief "how to" on controls beforehand, seeing a menu called "actions" and it not having all the actions I wanted was a bit weird.

It's just like cookiecliker. Fun clicking but kinda useless fun

- More indication of what the end goal actually is
- Does not make sense that exploiting a DoS vulnerability is necessary to discover nodes past a server
- Unexpected that services, servers and infrastructure was combined in the same graph. Compromising a switch could be relevant if you need it to gain access to separate networks, but this is not a common attack scenario and there is often other ways into an adjacent network.

The final exploitable system could be placed in a more randomized location. It is fairly obvious that the machine is somewhere on the right hand side of the game area, giving feedback that the further you go right, the closer you are to exploiting the system. Maybe having the root node in the middle would be a better starting location? It would not give any hints as to where the exploitable target is.

no

Burde bruke mer realistiske angrep

A better described usage and end goal

It was a fun game, but maybe better explanation in the beginning?

It was hard to understand what input that was allowed. It took me a long time to realize that I could right click on the discovered items.

not sure

No idea

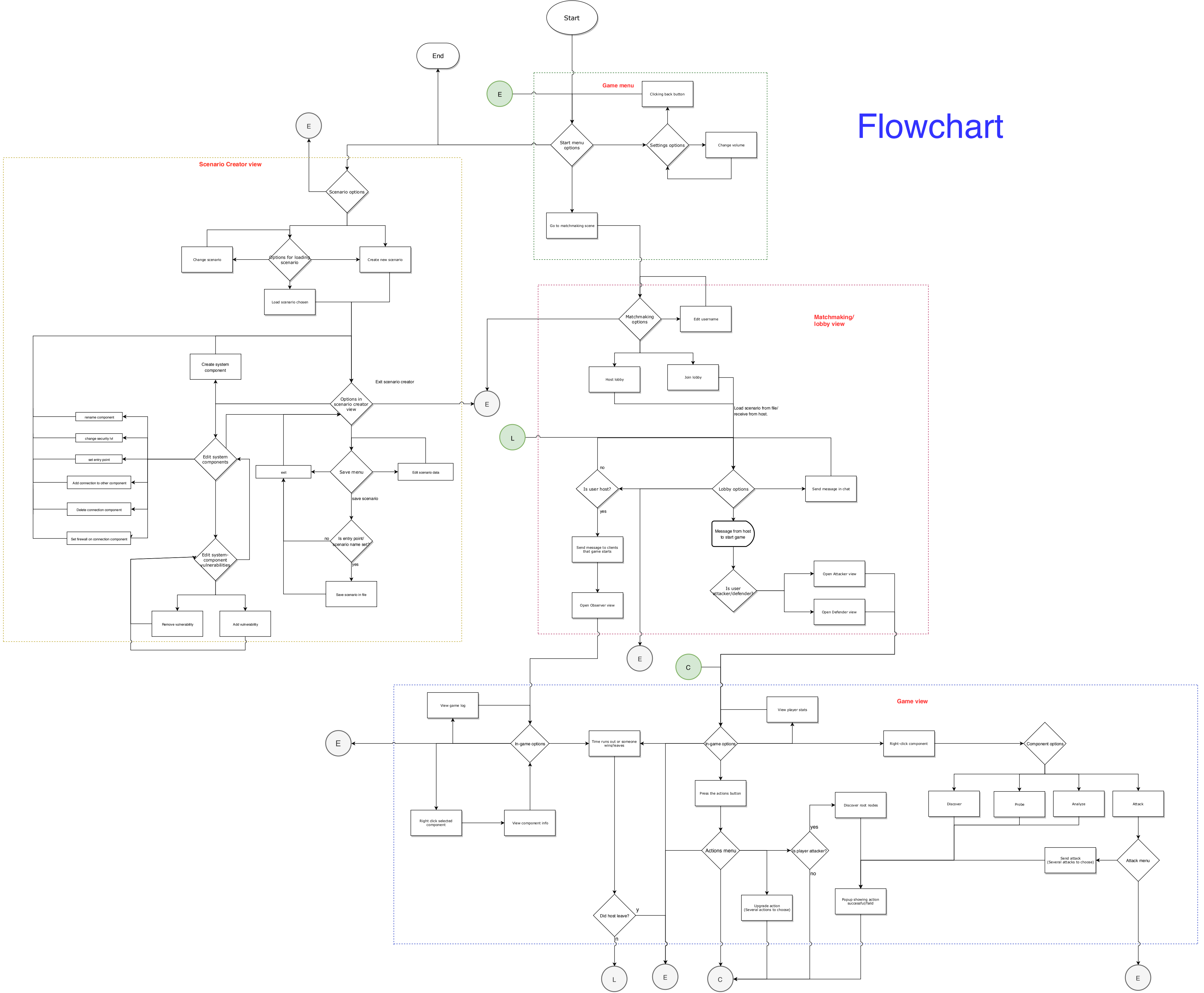
more options

Make it clear when you can do certain thing, like discover new devices before attacking on the switch but not on other things like API's? GOt confused

F System flowchart

Our flowchart shows the result of every action that be taken in the system. It has 4 main views that are shown as a set of actions/decisions within a colored square, these views are Main menu, Matchmaking/lobby, Scenario creator and Game. These are connected through arrows and circular teleports that reference to a different section of the chart, these circles contain a letter showing where they are going. Naturally the arrows going into the grey circles are the ones teleporting away and the arrows going out of the green circles has just teleported. There can be several arrows overlapping eachother, but they will **always** go in the same direction. The rectangles represents a process whereas a diamond shape represents a decision. The bullet shape represents a process that is started from another source when the current player has these specific decisions, such as the host sending message to the clients to start game.

Flowchart



Start
End
Game menu
Start menu options
Settings options
Clicking back button
Change volume
Go to matchmaking scene
E
Matchmaking/ lobby view
Matchmaking options
Edit username
Host lobby
Join lobby
Load scenario from file/ receive from host.
L
Lobby options
Send message in chat
Is user host?
no
yes
Send message to clients that game starts
Open Observer view
Message from host to start game
Is user attacker/defender?
Open Attacker view
Open Defender view
C
Scenario Creator view
Scenario options
Options for loading scenario
Change scenario
Create new scenario
Load scenario chosen
Create system component
Options in scenario creator view
Exit scenario creator
Edit system components
rename component
change security lvl
set entry point
Add connection to other component
Delete connection component
Set firewall on connection component
Save menu
exit
Edit scenario data
save scenario
Is entry point/ scenario name set?
Save scenario in file
Edit system-component vulnerabilities
Remove vulnerability
Add vulnerability
Game view
View game log
In-game options
Right click selected component
View component info
Time runs out or someone wins/leaves
Did host leave?
y
n
View player stats
Right-click component
Component options
Press the actions button
Actions menu
Is player attacker?
Discover root nodes
Upgrade action (Several actions to choose)
Popup showing action successful/faild
Discover
Probe
Analyze
Attack
Attack menu
Send attack (Several attacks to choose)

G User manual

This is the extensive user manual for the system developed during this bachelor project.

Bachelor 2019

Red VS Blue, Cyber Security Simulator
User manual

Christian Bråthen Tverberg
Maarten Dijkstra
Nataniel Gåsøy

January-May 2019

Contents

1 Introduction

This document is intended to be a user manual for the real-time cyber security simulator prototype designed and implemented during our bachelor thesis.

2 Scenario creator

This section is for the scenario creator, and will cover process behind making a scenario. This includes how to add and connect components, set attributes as well as saving a scenario with the player and game statistics.

2.1 Making a scenario

In order to make a scenario, a plan must first be in place. This plan must include what the goal of the scenario will be, what resources and advantages the attacker and defender has from the beginning as well as what components will be involved.

The next step is to make the scenario. Either load an existing scenario or start a new one. Place the components desired for the scenario (from the menu on the left-hand side) by clicking the corresponding button and drag these to the desired place. Note that if an object is dropped outside of the editing field (The marked area in the center of the screen) it will be deleted.

Right-click on the components to change it's attributes (name, security level, if it is an entry point, list of vulnerabilities). From here it is also possible to delete the component, or make a connection between it and another component. In order to connect to another component, right-click a component, select "connect" and click the component you want to connect it to.

The connecting reference lines can also be attributed (firewall) or deleted by right-clicking on it.

After the topology is finished, and the attributes have been set, press save in order to save this scenario. In the save menu, you choose the name for the scenario as well as setting the attributes and resources for the attacking and defending team.

2.2 Scenario creator manual

This section will explain in more detail the different aspects of the scenario creator.

In this scenario creator screenshot (1), the menu seen on the left-hand side is a series of buttons, each representing the available system components. Pushing one of these will instantiate an object of the related system component from the folder of prefabs. This will be an empty component with only the base attributes of the corresponding system components.

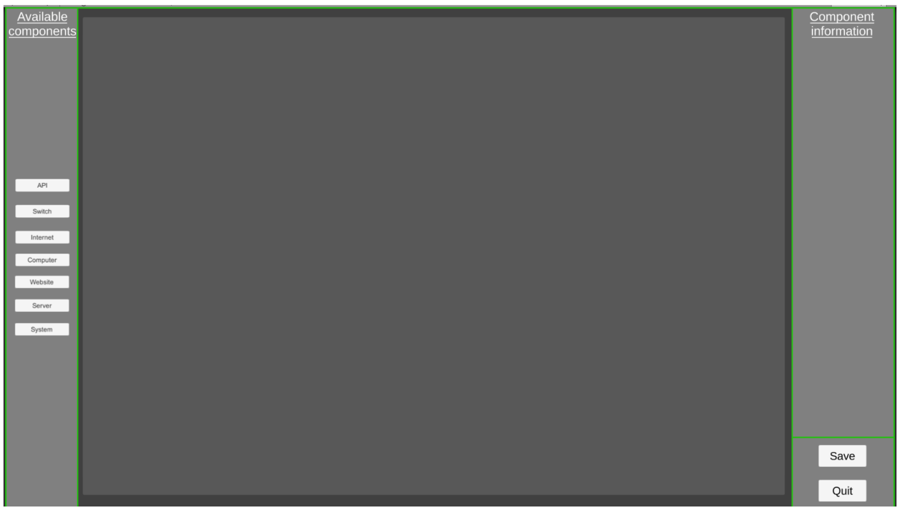


Figure 1: An empty scenario creator screen

On the right-hand side, the information of the selected system component is displayed. This information includes component type, security level, if it is an entry point and the list of vulnerabilities this component has. At the bottom of the right-hand menu, there is a button for saving the newly made scenario as well as a quit button to exit the scenario.

The next figure (2) shows the system component menu. This is accessed by right-clicking a system component within the editing field. This menu consists of the component type, connect button, vulnerabilities button, rename button, entry point toggle, security level dropdown and delete button.

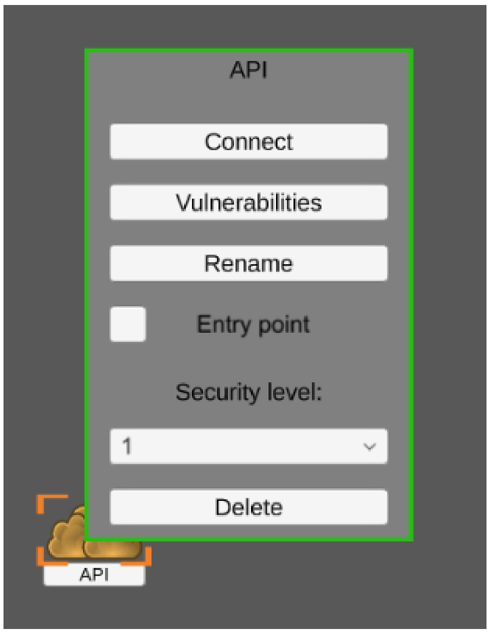


Figure 2: A system component's right-click menu

The connect button is used to connect two system components. the vulnerabilities button will bring up the vulnerabilities menu. Here a list of vulnerabilities is read from the vulnerabilities XML file, and displayed in the menu. Selecting a vulnerability and pressing the "==¿" button will apply the selected vulnerability to the system component. The rename button opens the rename menu, which enables giving the system component a new name instead of the default name (which is the same as the type). The entry point toggle lets the user specify if this system component is an entry point or not. The security level describes how secure this particular system component is (or how vulnerable it is, depending on the angle you view it from). Finally, the delete button enables you to delete a specific system component.

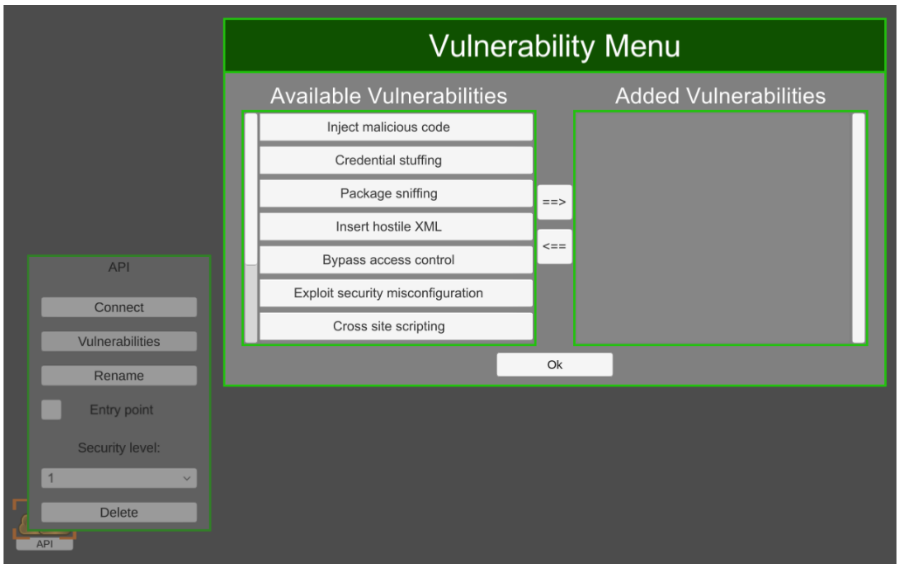


Figure 3: Vulnerability menu

In the vulnerability menu, seen in figure (3), the user can specify the vulnerabilities for the selected system component. The available vulnerabilities is read from an XML file, and displayed in a scrollable menu. From here, the user can add the desired vulnerabilities provided that they exist in the XML file, or remove a vulnerability from the list of added vulnerabilities.

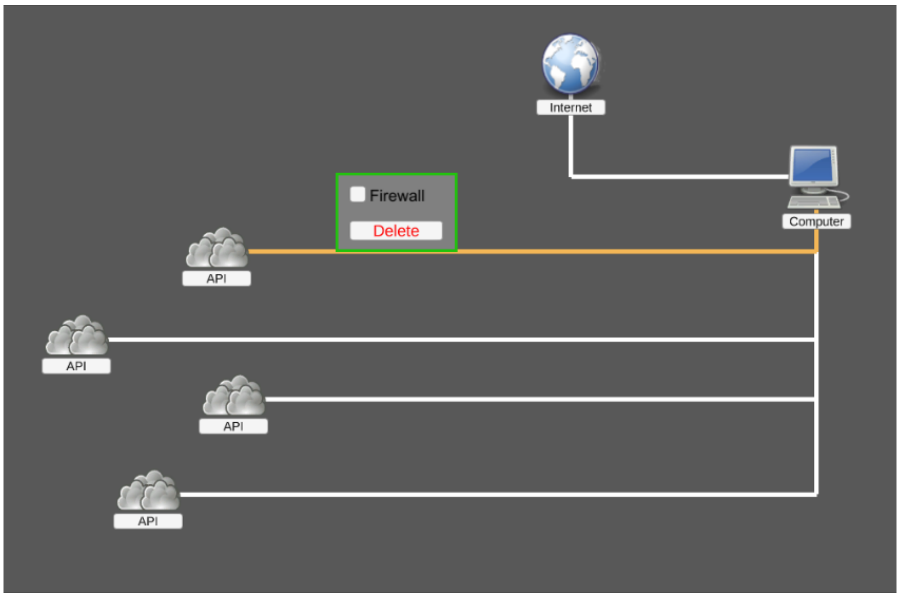
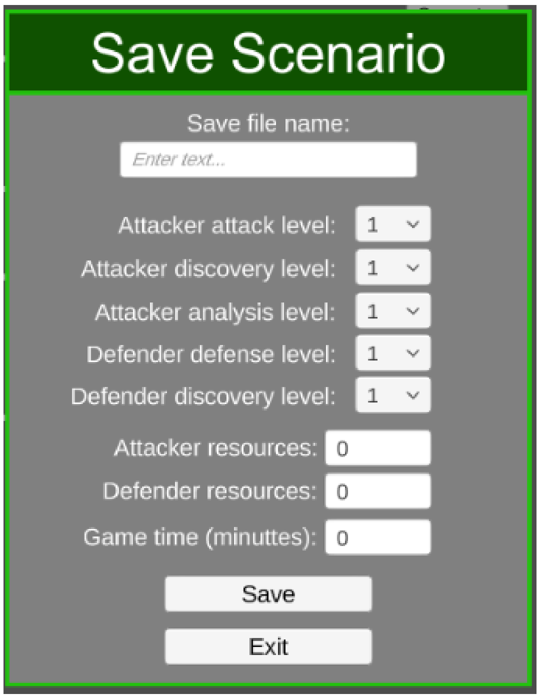


Figure 4: Connections and connection line menu

Right-clicking the connections, as seen in figure (4), brings up the menu for reference connections. Here the user can choose if the connection has a firewall or not, as well as delete the connection.


Save Scenario

Save file name:

Enter text...

Attacker attack level:	1
Attacker discovery level:	1
Attacker analysis level:	1
Defender defense level:	1
Defender discovery level:	1
Attacker resources:	0
Defender resources:	0
Game time (minuttes):	0

Figure 5: Save menu

Figure (5) shows the save menu. Here the user chooses a name for the new scenario created. If a file with that already exists, the user is asked if they want to overwrite this file or not. The statistics for attacker and defender, as well as their available resources and the duration of the scenario, is also set here for the scenario.

3 Gameplay

3.1 Setting up and connecting to a game

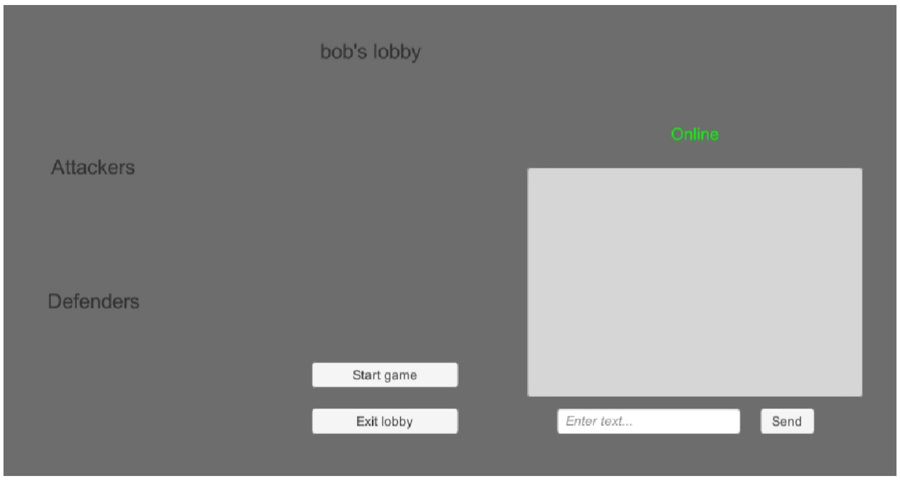


Figure 6: View of an empty lobby from host's perspective.

To play the game you would need to either host a lobby and play as an observer, or join a lobby as a client and play as either attacker or defender. As a host you have the button for starting the game which you can do when each team has a player connected. As a client that joins a game you will get a list of available lobbies in the local network(if there are anyone hosting at the moment). If you are sure that a host is hosting yet you can't find that lobby you would need to follow a few steps outside the game. Firstly open the command prompt window, then use the command "$ping < IP >$", where "¡IP¿" is the hosts IP. You can find this IP by using either the command "$ipconfig$" and use the value next to "$IPv4$" or "$arp - a$" and use the value next to "interface" in a command prompt window on the hosts pc. This will fix the problem.

Before the game starts you have the option to swap between playing as defender/attacker by clicking on the other player, the swap will go through if the other player choose to accept. When there is a player on both teams(attacker/defender) the host can start the game and everyone will go into their own view according to their mode(attacker/defender/observer).

3.2 Game screen

3.2.1 Attacker POV

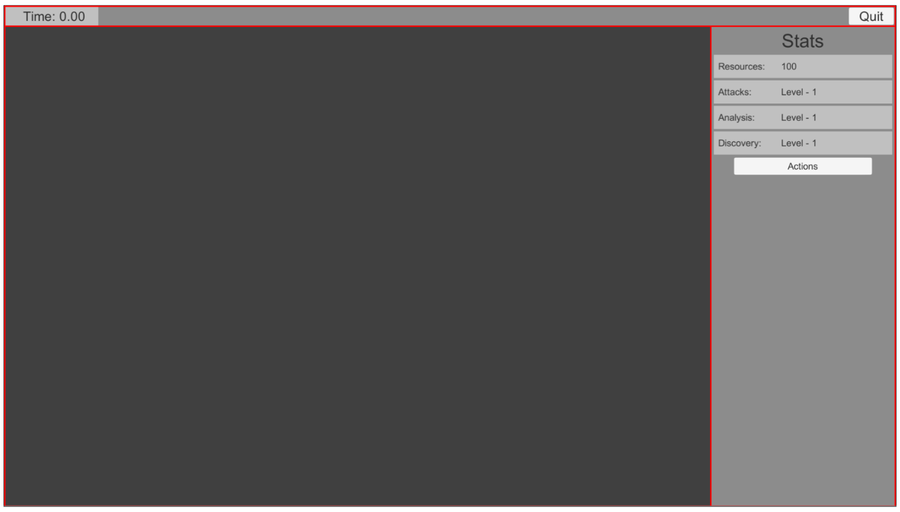


Figure 7: Attacker's initial view

Upon starting a session the Attacker (red team) will be met by a blank screen with no nodes (Figure 7). The right side panel is used for showing either user information, showing initial actions the attacker can take, information about a selected node, or different attacks to send to a node. The first entry here is the amount of resources the attacker has to spend on different actions. The second entry shows the attacker's attack level. The third entry is the attackers analysis level. The fourth entry is the attackers discovery level. Increasing a level in one of these will increase it's success chance. Hovering over any of these fields will display a small tooltip explaining what the entries mean. The final entry is a button which will lead to an actions menu.

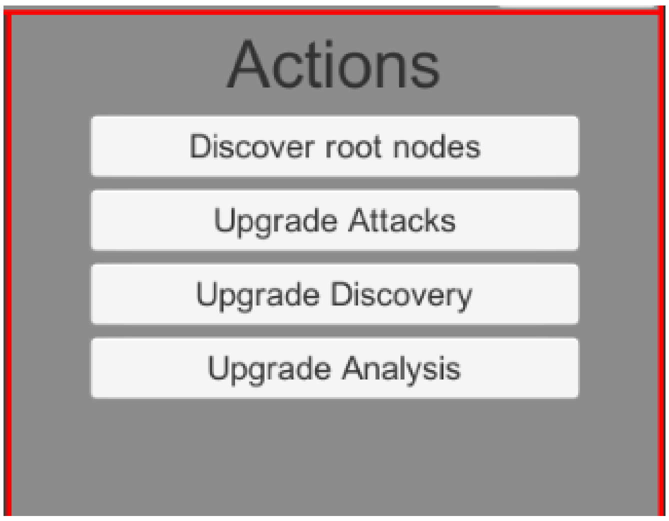


Figure 8: Attacker's actions menu

Clicking on the actions button will change the right panel into showing the different actions the attacker can do (Figure 8). Discovering root nodes does what it sounds it does. It tries to find an access point to the network in the scenario. Discovering an entry point will always succeed, but delving deeper into the network might not. The other three buttons are for upgrading their respective actions. Each upgrade level costs more and takes longer than the previous, but will increase the chance of success doing any of those actions. Currently, the success chance increases linear, and the max level on any action is level three.

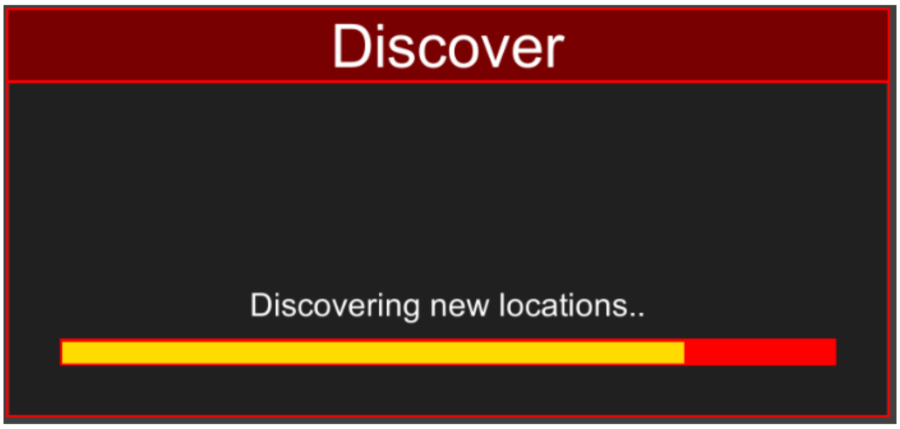
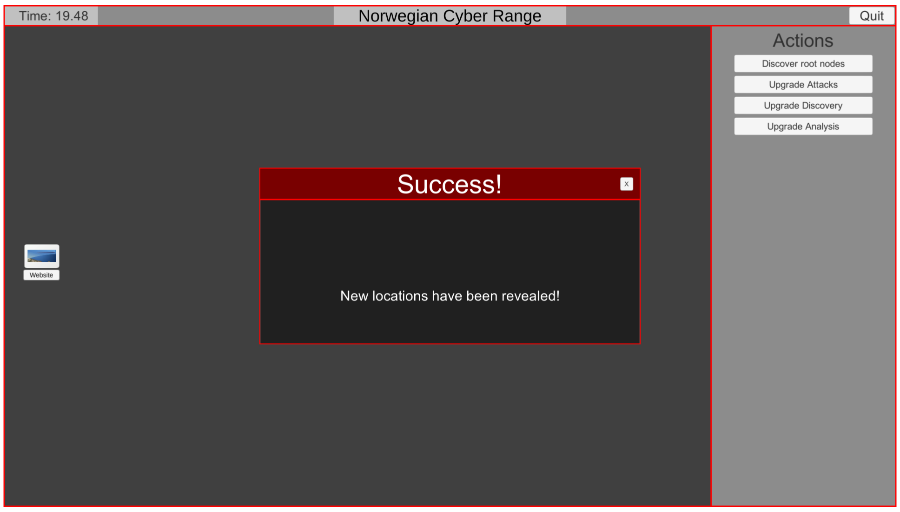


Figure 9: Progress window

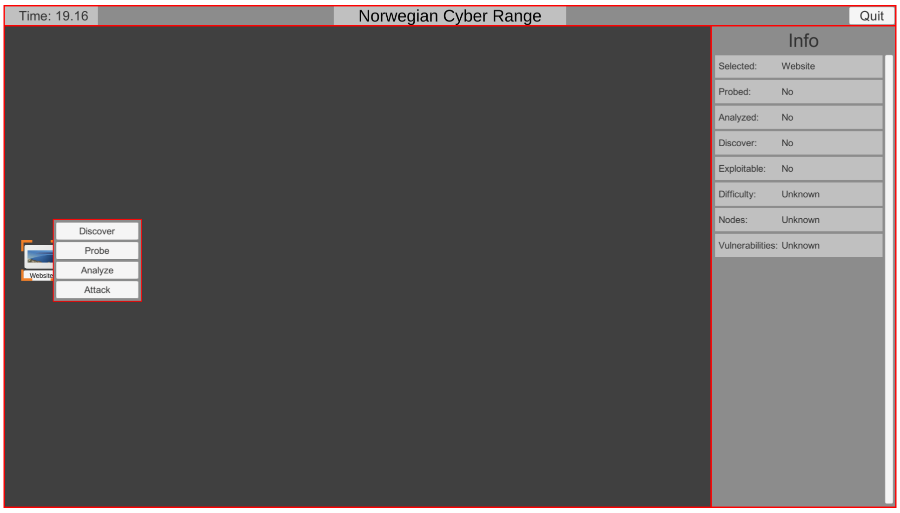
Doing any action requires time, and starting one will pop up a progress window showing what you do and the current progress on the task (Figure 9). Here we are almost done with a discovery action. Different actions take a different amount of time to complete, and since you are only one attacker, you can only do one thing at a time.



New locations have been revealed!

Figure 10: Discovery success!

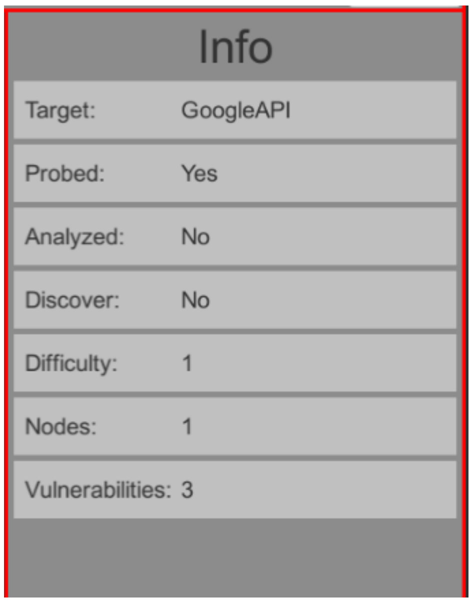
Once an action has been completed either a success or failure message will pop up, and here we succeeded in finding a root node (Figure 10). There will always be at least one entry point to network, and here we found a node called 'Website'.



Selected:	Website
Probed:	No
Analyzed:	No
Discover:	No
Exploitable:	No
Difficulty:	Unknown
Nodes:	Unknown
Vulnerabilities:	Unknown

Figure 11: Node view

When we click on a node (here 'Website') the right panel with be filled with the info we have on it. With a right-click on a selected node a menu will popup next to the node (Figure 11). The info panel will show you the information you currently have available on the node. Each entry has a tooltip explaining the entry. The popup menu has four available actions you can take on the node. Discover will try and find any node that is connected to the selected one. Probe will get you surface level information of the node (Difficulty, number of connected nodes and number of vulnerabilities). Analyze will let you find the specific vulnerabilities on that node. Attack will let you choose which attack you want to send to the node. Every action except Probe has a chance for failure, or a partial success. Probe will always succeed.

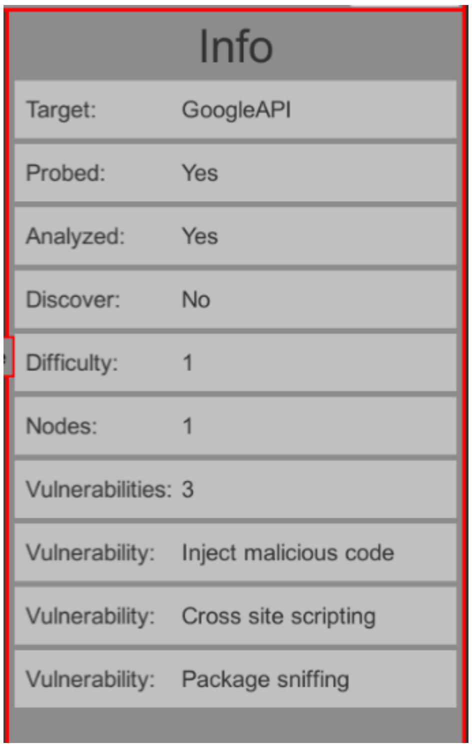


Info

Target:	GoogleAPI
Probed:	Yes
Analyzed:	No
Discover:	No
Difficulty:	1
Nodes:	1
Vulnerabilities:	3

Figure 12: Info panel detailed after probing

Here we see the information available for the 'GoogleAPI' node after probing (Figure 12). The difficulty is one, there is one connected node, and there are three vulnerabilities on this node.



Target:	GoogleAPI
Probed:	Yes
Analyzed:	Yes
Discover:	No
Difficulty:	1
Nodes:	1
Vulnerabilities:	3
Vulnerability:	Inject malicious code
Vulnerability:	Cross site scripting
Vulnerability:	Package sniffing

Figure 13: Info panel detailed after analysis

Here we see the information available for the 'GoogleAPI' node after analyzing it (Figure 13). We were lucky and found all three vulnerabilities. This information might get outdated as the game progresses as the defender might add defenses to the node, in which case you have to probe and analyze again to get the new information.

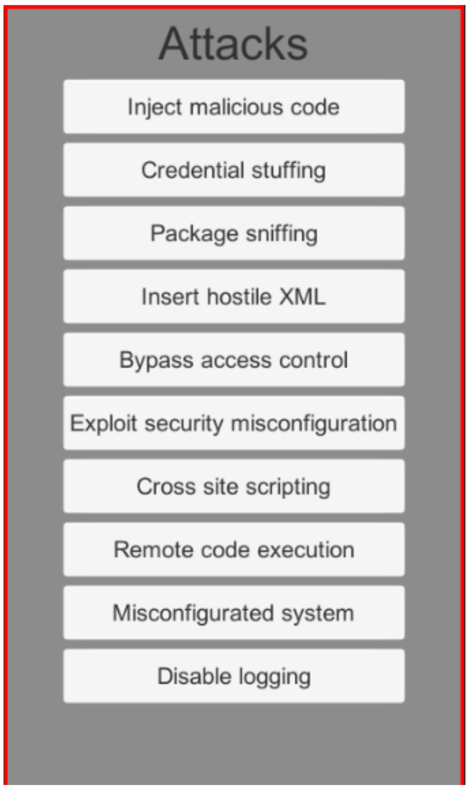


Figure 14: Attack options

When clicking on the Attack button from the popup menu (Figure 11), the right panel will be populated with the different attack options available to you (Figure 14). The best way of attacking is knowing which vulnerabilities the node has. Attacking blindly might succeed if you are lucky, but more often than not you will fail.

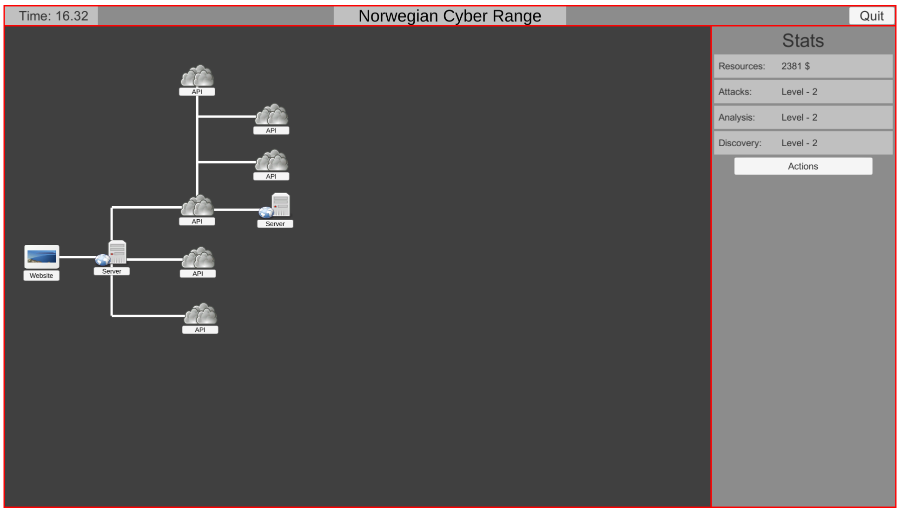


Figure 15: Progress

This is an example of what your game screen might eventually look like. We have several different nodes with individual icons, and lines connecting them.

3.2.2 Defender POV

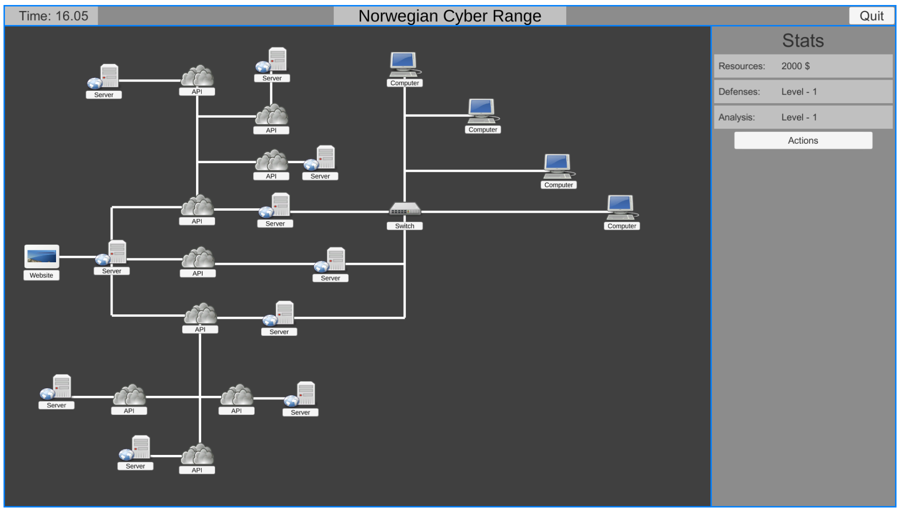


Figure 16: Defender's initial view

Upon starting a session the Defender (blue team) will be met by all the nodes in the network (Figure 16). The right side panel is used for showing either user information, showing initial actions the defender can take, information about a selected node, or different defenses to implement in a node. The first entry here is the amount of resources the defender has to spend on different actions. The second entry shows the defender's defense level. The third entry is the defenders analysis level. Increasing a level in one of these will increase it's success chance. Hovering over any of these fields will display a small tooltip explaining what the entries mean. The final entry is a button which will lead to an actions menu.

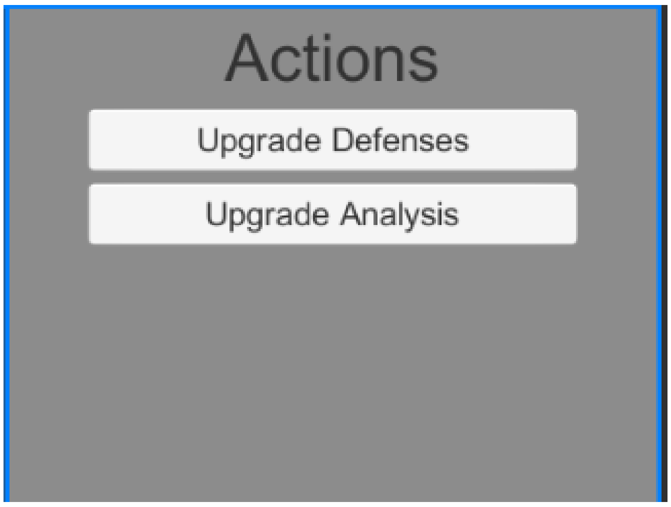
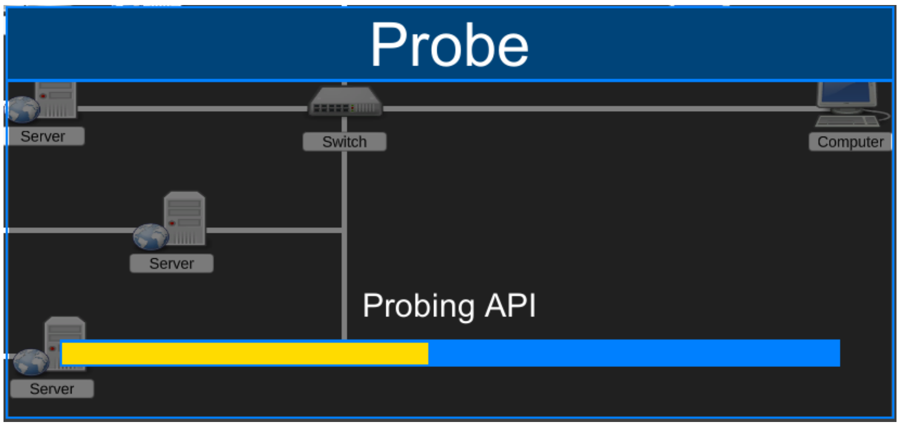


Figure 17: Defender's initial actions

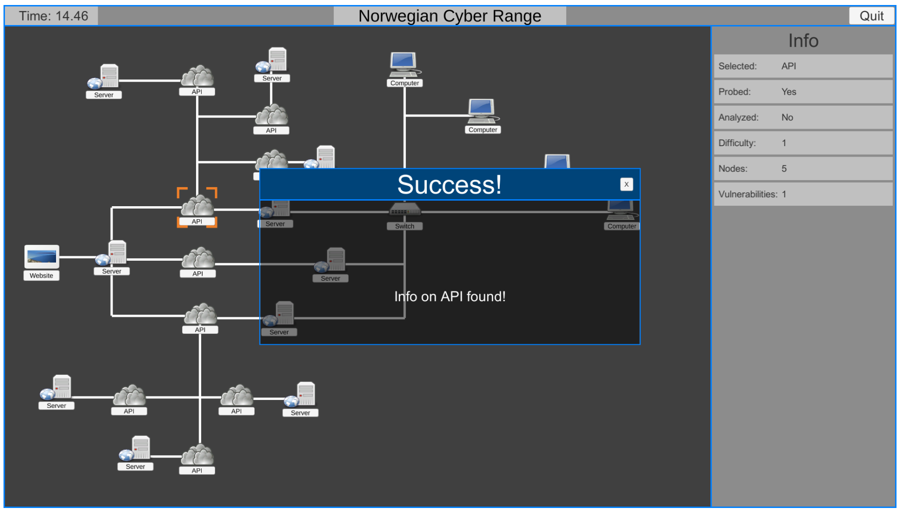
Clicking on the actions button will change the right panel into showing the different actions the defender can do (Figure 8). Unlike the attacker, the defender always see all nodes, so there is no need to discover. The two buttons are for upgrading their respective actions. Each upgrade level costs more and takes longer than the previous, but will increase the chance of success doing any of those actions. Currently, the success chance increases linear, and the max level on any action is level three.



Probing API

Figure 18: Progress window

Doing any action requires time, and starting one will pop up a progress window showing what you do and the current progress on the task (Figure 18). Here we are almost done with a probe action. Different actions take a different amount of time to complete, and since you are only one defender, you can only do one thing at a time.



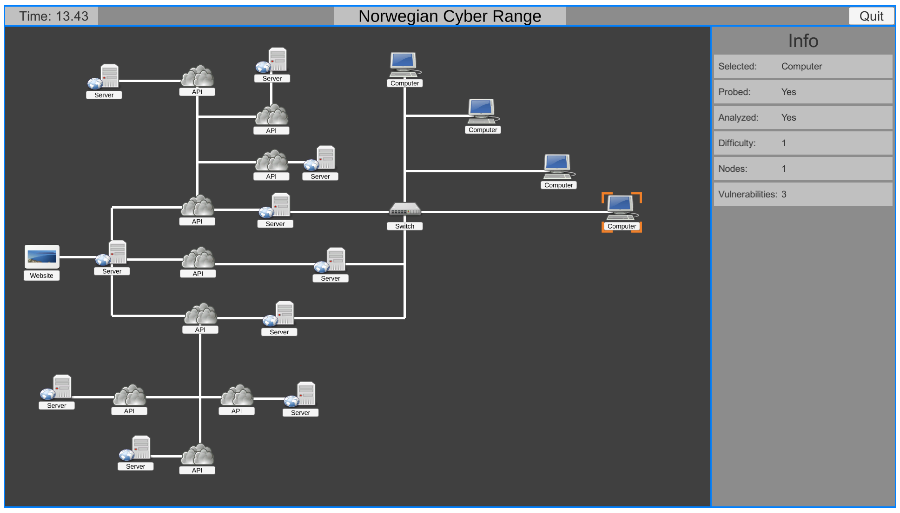
Selected:	API
Probed:	Yes
Analyzed:	No
Difficulty:	1
Nodes:	5
Vulnerabilities:	1

Success!

Info on API found!

Figure 19: Probe success!

Once an action has been completed either a success or failure message will pop up, and here we succeeded in probing an API (Figure 19).

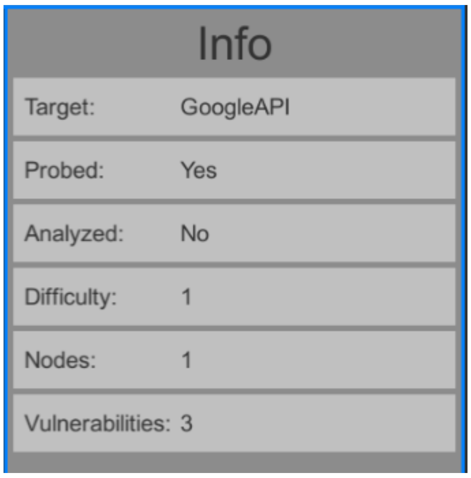


Info

Selected:	Computer
Probed:	Yes
Analyzed:	Yes
Difficulty:	1
Nodes:	1
Vulnerabilities:	3

Figure 20: Node view

When we click on a node (here 'Computer') the right panel with be filled with the info we have on it. With a right-click on a selected node a menu will popup next to the node (Figure 20). The info panel will show you the information you currently have available on the node. Each entry has a tooltip explaining the entry. The popup menu has three available actions you can take on the node. Probe will get you surface level information of the node (Difficulty, number of connected nodes and number of vulnerabilities). Analyze will let you find the specific vulnerabilities on that node. Defend will let you choose which defense you want to implement on the node. Every action except Probe has a chance for failure, or a partial success. Probe will always succeed.

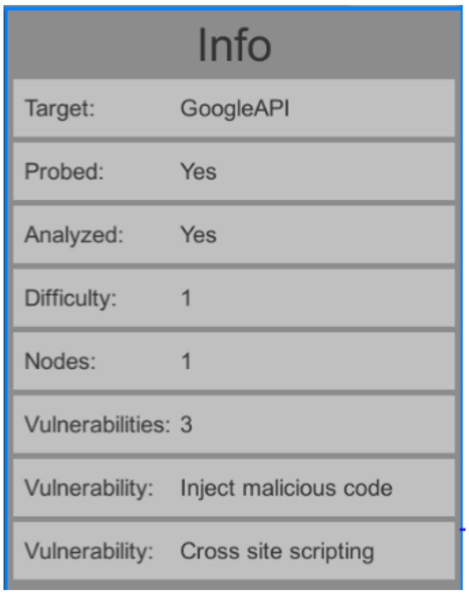


Info

Target:	GoogleAPI
Probed:	Yes
Analyzed:	No
Difficulty:	1
Nodes:	1
Vulnerabilities:	3

Figure 21: Info panel detailed after probing

Here we see the information available for a 'GoogleAPI' node after probing (Figure 21). The difficulty is one, there is one connected node, and there are three vulnerabilities on this node.



Info

Target:	GoogleAPI
Probed:	Yes
Analyzed:	Yes
Difficulty:	1
Nodes:	1
Vulnerabilities:	3
Vulnerability:	Inject malicious code
Vulnerability:	Cross site scripting

Figure 22: Info panel detailed after analyzing

Here we see the information available for a 'GoogleAPI' node after analyzing it (Figure 22). We weren't lucky and only found two of three vulnerabilities This information might get outdated as the game progresses as you might add defenses to the node, in which case you have to probe and analyze again to get the new information.

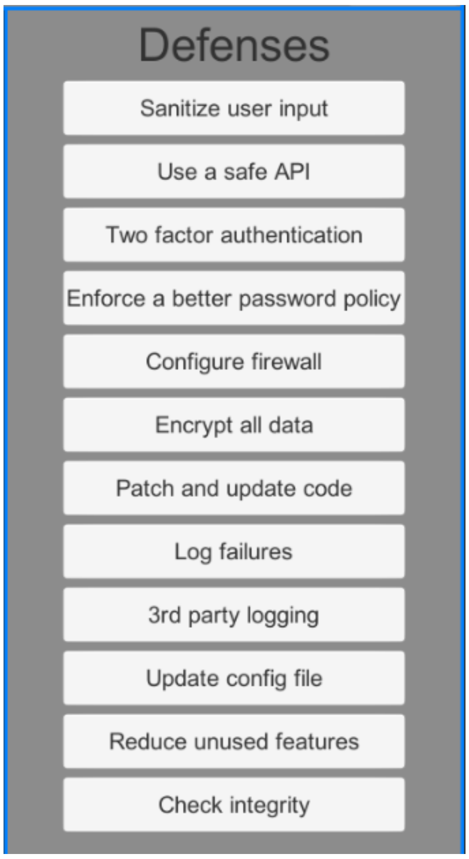


Figure 23: Defense options

When clicking on the Defense button from the popup menu (Figure 20), the right panel will be populated with the different defense options available to you (Figure 23). The best way of defending is knowing which vulnerabilities the node has. Defending blindly might succeed if you are lucky, but more often than not you will fail.

3.2.3 Observer POV

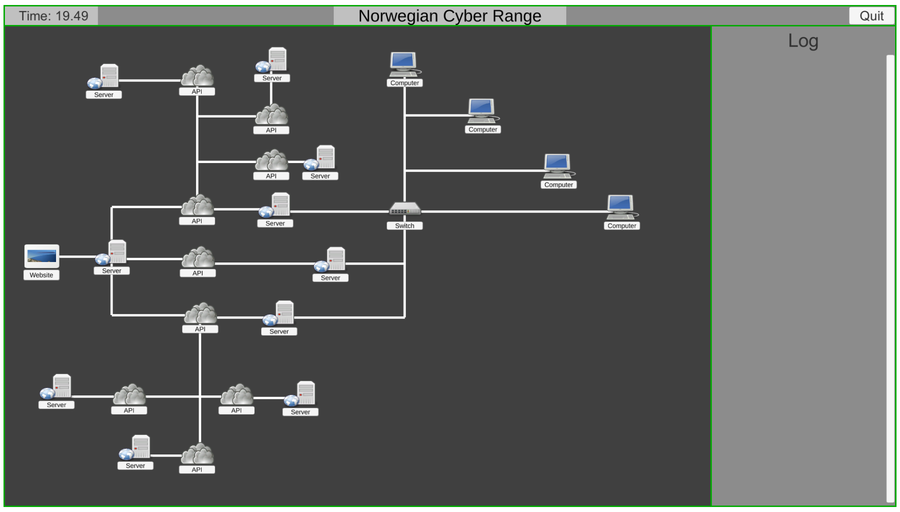
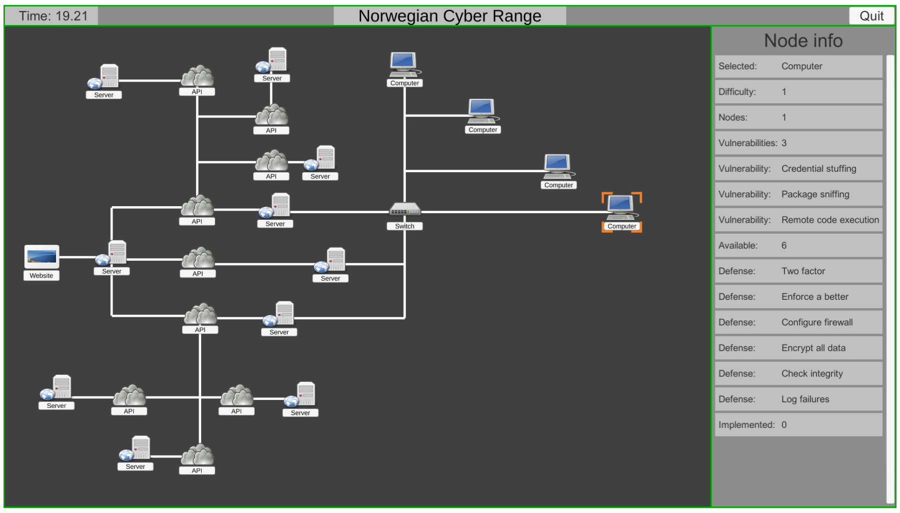


Figure 24: Observer's initial view

Upon starting a session the Observer will be met by all the nodes in the network (Figure 24). The right side panel is used for showing either the log of events, or information about a selected node. The log will automatically scroll to the bottom to display the newest events during gameplay. Most events will also be saved to log files, divided between attacker and defender, to be able to be reviewed at a later point.

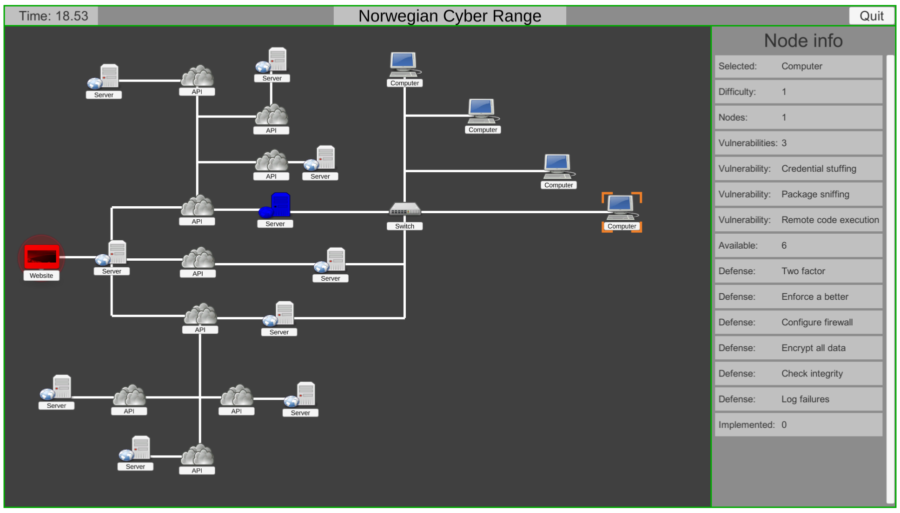


Node info

Selected:	Computer
Difficulty:	1
Nodes:	1
Vulnerabilities:	3
Vulnerability:	Credential stuffing
Vulnerability:	Package sniffing
Vulnerability:	Remote code execution
Available:	6
Defense:	Two factor
Defense:	Enforce a better
Defense:	Configure firewall
Defense:	Encrypt all data
Defense:	Check integrity
Defense:	Log failures
Implemented:	0

Figure 25: Node info

When we click on a node (here 'Computer') the right panel with be filled with all the info on it (Figure 25). Each entry has a tooltip explaining the entry. This info will automatically be updated as the game progresses.

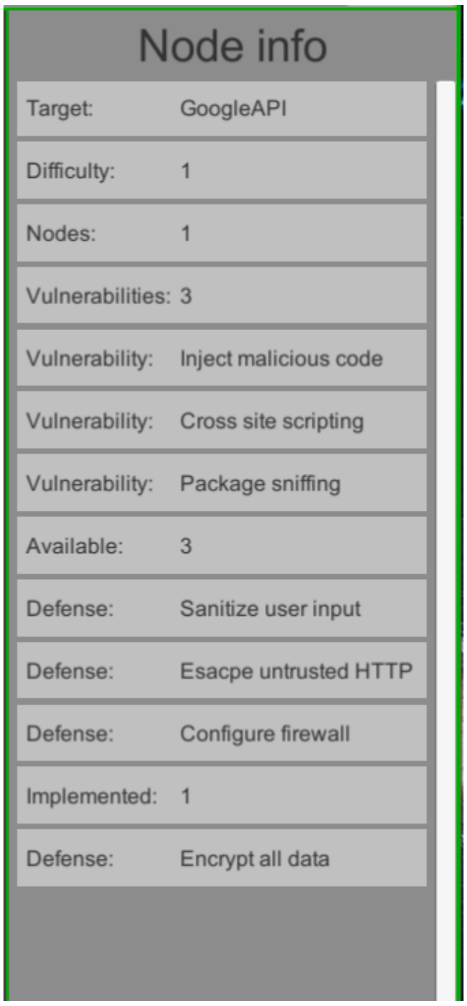


Node info

Selected:	Computer
Difficulty:	1
Nodes:	1
Vulnerabilities:	3
Vulnerability:	Credential stuffing
Vulnerability:	Package sniffing
Vulnerability:	Remote code execution
Available:	6
Defense:	Two factor
Defense:	Enforce a better
Defense:	Configure firewall
Defense:	Encrypt all data
Defense:	Check integrity
Defense:	Log failures
Implemented:	0

Figure 26: Attacker and Defender targeting

During gameplay, when either the attacker or defender is targeting a node, the nodes color will change depending on who is targeting it (Figure 26). If they target the same node, the most recent targeting will display, but if the last one deselects or targets another node, the color will be changed back to the previous color.

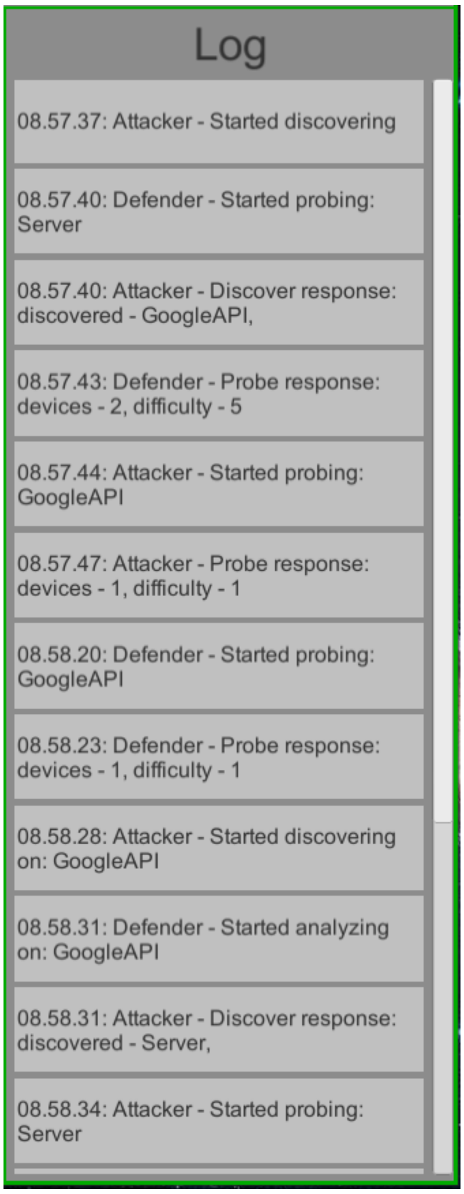


Node info

Target:	GoogleAPI
Difficulty:	1
Nodes:	1
Vulnerabilities:	3
Vulnerability:	Inject malicious code
Vulnerability:	Cross site scripting
Vulnerability:	Package sniffing
Available:	3
Defense:	Sanitize user input
Defense:	Esacpe untrusted HTTP
Defense:	Configure firewall
Implemented:	1
Defense:	Encrypt all data

Figure 27: Updated info during gameplay

Here is an example of a node having changed during gameplay (Figure 27). We can still see the vulnerabilities, but the defender has implemented a defense since earlier (Figure 25).

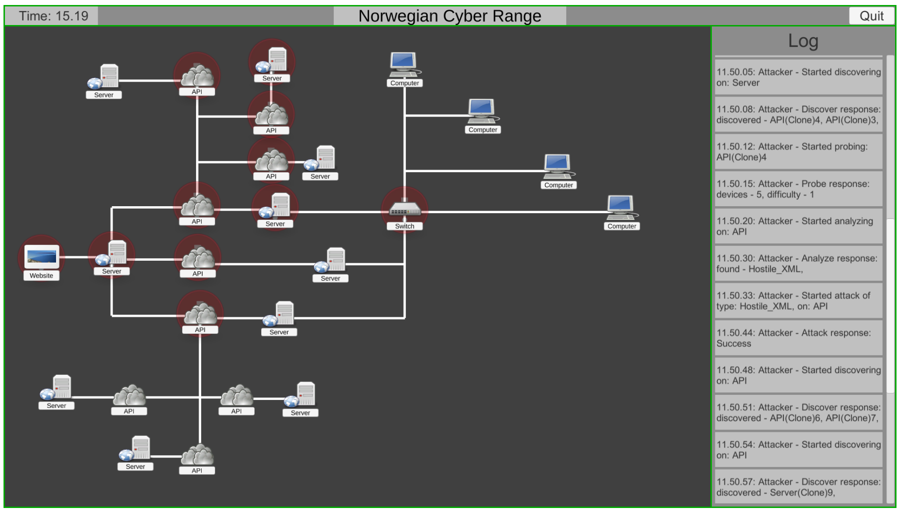


```
Log
08.57.37: Attacker - Started discovering
08.57.40: Defender - Started probing:
Server
08.57.40: Attacker - Discover response:
discovered - GoogleAPI,
08.57.43: Defender - Probe response:
devices - 2, difficulty - 5
08.57.44: Attacker - Started probing:
GoogleAPI
08.57.47: Attacker - Probe response:
devices - 1, difficulty - 1
08.58.20: Defender - Started probing:
GoogleAPI
08.58.23: Defender - Probe response:
devices - 1, difficulty - 1
08.58.28: Attacker - Started discovering
on: GoogleAPI
08.58.31: Defender - Started analyzing
on: GoogleAPI
08.58.31: Attacker - Discover response:
discovered - Server,
08.58.34: Attacker - Started probing:
Server
```

Figure 28: Log of events

During gameplay, every action taken in-game will be logged by the observer (Figure 28). The log will be sorted by time, and you are able to scroll up or down to see what happened when.

Once the attacker gets progress into the network, the nodes the attacker has access to will have a red circle behind then (Figure 29. This is in order to highlight the current progress of the attacker.



```
11.50.05: Attacker - Started discovering
on: Server
11.50.08: Attacker - Discover response:
discovered - API(Clone)4, API(Clone)3,
11.50.12: Attacker - Started probing:
API(Clone)4
11.50.15: Attacker - Probe response:
devices - 5, difficulty - 1
11.50.20: Attacker - Started analyzing
on: API
11.50.30: Attacker - Analyze response:
found - Hostile_XML,
11.50.33: Attacker - Started attack of
type: Hostile_XML, on: API
11.50.44: Attacker - Attack response:
Success
11.50.48: Attacker - Started discovering
on: API
11.50.51: Attacker - Discover response:
discovered - API(Clone)6, API(Clone)7,
11.50.54: Attacker - Started discovering
on: API
11.50.57: Attacker - Discover response:
discovered - Server(Clone)9,
```

Figure 29: Progress in gameplay

3.3 Win/Lose condition?

(To be implemented)
The win condition for each side is determined by the scenario creator. There will be different types of win/lose conditions like ”attacking or defending a certain node”, ”Have x amount of successful attacks” etc. The scenario creator will have the option to create descriptions for each side to explain the goal, which either side can read whenever they want during gameplay.

H Notes from meetings with the supervisor

These are the notes taken during the meetings with the supervisor during this bachelor thesis.

Bachelor 2019

Meetings with supervisor

Christian Bråthen Tverberg
Maarten Dijkstra
Nataniel Gåsøy

January-June 2019

Contents

1 Introduction

These are notes from the meeting with the supervisor, Mariusz Nowostawski, during the course of this bachelor project.

2 14.01.2019, 09.00

2.1 Agenda for the meeting

- Present the project plan draft.
- Discuss the project (how it will be/ what is planned to be put in/ features and functionality)
- Real-time vs turn based.
- How to structure the scenario creator.
- How to structure the balance and interaction between attacker and defender.
- How to do the multiplayer part of the game.
- Resources: should there be a static resource amount from the start (budget), or should the players be able to use time to gather resources? (defenders gain resources over time, attackers sell what they find to gain more resources?)

2.2 Thoughts so far

- Saving scenarios: JSON (only saving what is enabled) vs binary/txt (enabled/disabled per item), Roller Coaster Tycoon 2 saves scenarios as binary
- Turn-based vs real-time
- Discovering local hosts might be harder, but better for end-user
- Income of resources can most likely be easy to implement at later stages
- Difficulties with undefined win condition

2.3 Notes from the meeting

- Real time, with every action taking a certain time to complete? Product owner request.
- Mock everything (how to set up the system (building blocks/ restrictions/))
- win/ lose (time based/ score based/ system event check/ complete) checklist, can be set up by/in the scenario creator/
- Make a prototype to test how to do scoring, with a hard coded scenario.
- Save scenarios in human readable format (JSON? XML? evaluate and reason for choice)
- Have functionality to snapshot/ record the state of the game (for reflections by users? review session?)
- Resources dynamic (start with static, develop further to dynamic) (attacker set up raspberry pi for sniffing/ black market sale and purchase/)
- Represent the different components as 2D or 3D? have 3D boxes and structures, with wires connecting them. (3D gives extra space, might be more messy. 2D might be more aligned with working with security on a flat screen. 3D might be hard to gain an overview on a 2D screen. context switch easier in 2D, switching screens)
- Start with local LAN multiplayer, maybe later have global server (local may be prone to cheating, i.e. time change. how secure? get time from internet instead of local machine
- May or may not be an opponent? test/ trial environment.

3 21.01.2019, 08.45

3.1 Agenda for the meeting

- Template for the bachelor thesis document.
- Multiplayer local or on dedicated server (skyhigh?)
- Dedicated server for the game? or local host (viewer mode) and clients (attacker and defender).
- Unity is removing Unet, so we should use the new one instead to think about further use for the future. UNet will no longer be in use after first half of 2022. The new network will be better for scalability, inconsistent connection quality and hackable clients. How to use the new networking?

- Discuss new version of project plan
- Take a look at current list of component - vulnerabilities - attacks - defenses.

3.2 Notes from the meeting

Feedback on the project plan:

- Explain terminology (scenario/cyber range (”instructors will use this tool to make scenarios and train students”)/)
- Be consistent (use ”it” more, don’t use space after a slash ”this/that”
- Unity best tool for the job? alternatives? elaborate and explain
- Avoid using etc.
- What things are essential? what can be skipped if there is not enough time? explain and discuss what (potentially) will not be done (OWASP already ordered in a top 10 list, utilize this).
- not zero sum game? probably remove (want flexibility)
- Specify that there will not be developed a gameplay AI for the gameplay (player vs player, not player vs AI. might want to use AI for details and small tasks later).
- Explain how the game will be played in regard to number of players (single player teams/teams cooperating on the same station/co-op for the teams over different machines)
- Turn.based vs realtime/single player vs teams
- milestones, state what will be completed for decisions like ”finished planning”, ”mock-up”. specify deliverables. what is the definition of done? i.e. ”simple gameplay of a local multiplayer game test”
- What can go wrong? Development delays. New unity versions (freeze it under development). What is product owner/ supervisor does not have time for meeting (fallback person - Yamin -¿ Basel / Mariusz -¿ Christopher?)
- Evaluation: talk with product owner. What will be delivered? how will we evaluate that the product we delivered is good? Playtest and measure feedback. lag/refresh rate? Hard to evaluate this project, as it will not be a finished product but rather the components for one (machinery for the scenario builder/.
  develop a demo!!! evaluation = show that the components work with a demo.

Feedback for the rest of the meeting:

- Template for the bachelor is run by Simon and Christopher (front page is out of our control, the rest is based on the template and what we write)
- Dedicated server? have to use resources for server code. (start with local, make the decision later). Might need a dedicated server for full release, not necessarily part of our bachelor.
- Need a dedicated server if we plan on doing it very securely. Again, security of the application is not a main concern for this bachelor thesis, but to have an actual functional product.
- Unity multiplayer: use the new method for multiplayer, not the dedicated one.

4 05.02.2019 10.15

4.1 Agenda for the meeting

- Hand in date for the bachelor thesis moved from 20.5 to 15.5?
- User defined attacks? if attacks and defences, and 3 levels on each of them, lots of logic behind it? predefined list of attacks and defences instead? (need to check in order to not do unnecessary work).
- Commenting style. doxygen? /// or /** */?
- Discuss technical mockup
- A good way of connecting defence with network components? How to check if a components has certain defenses. (Hard code IDs of components in the scenario creator? Use this as name, generated but editable by user in scenario creator?)
- Commission assets for the system components? for attacks and defenses?

4.2 Notes from the meeting

- JSON good for static setup, i.e. the static scenarios saved.
- Scenario is scripted, following a template. We just wary the wiring.
- One template that cover all scenarios (at least in the beginning. may need to modify/expand/have more templates later).
- Predefined safer. type safe, better for UI. We are doing overhead, so not that often new type of attack/vulnerabilities found. can be changed in code, maybe have this extracted as a separate library. isolate as a separate model, making changes easier

(not string-checking, but against drop-down. alternative map with string key-value pairs

- Do/say in the report that attacks/defenses can be abstracted, making updates to this list easier. No need to recompile the whole program, just that abstracted part.
- Preset allows localization, making language change easier. Nice feature for decoupling core unity models and external libraries for constants (type safety, but easier to swap).
- Need to state in the report the reason for commenting style, doxygen? /// or /** */.
- Up to the cyber range to buy assets. If the cyber range wants to find and buy an asset pack, we can implement that.
- Turn based easier to analyze later than for a real-time system. Mariusz suggests turn-based in order to get a feel for it (not jumping straight into deep water). Post action review makes it easier to take it step-by-step in turn based system.
  It also makes it easier to develop AI later, both for gameplay and for giving player suggestions based on the gamestate.
- Link: fully featured game object? line renderer might be best suited for debugging. need to investigate further.
  Using fully fledged game object might be the best solution, for editability and possibility to delete it later.

5 14.02.2019 09.15

5.1 Agenda for the meeting

- Discuss assets from the product owner
- Product owner wants us to find free icons.
- Product owner wants real-time (innovative).
- How would "observer" function? log? screen?
- Uint vs enum for messaging system?

5.2 Notes from the meeting

- Budget items: time, resources, quality (probability of success)
- Adding probability? Makes leveling components easier/makes scenarios more exciting; make the probability of defense lower over time?

- Point system: checkbox (point for success)? Bullet points (reward based on how fast objectives were reached)?
- Rewards: Points/In-game currency/In-game information/Private data (retract in-game currency + private data from defender if attacker gets these).
- Give points for attack based on type and probability? (one cheap attack often succeeds, but gives less reward. one expensive attack succeeds less often, give much higher reward).
- Have a meeting next Wednesday at the cyber range, presenting our project to another group. Discuss our projects, as they are similar.

6 19.02.2019 10.00

6.1 Agenda for the meeting

- Inform Basel's decision of supporting asset packs (and other necessities) up to 2000nok.

6.2 Notes from the meeting

- Consider pipelines/continuous integration/unit testing
- Benefit of automated testing? manual testing better?
- Game testing better than unit testing?
- is automated testing beneficial? will it take more time than it is worth?
- In report: "Always manual testing before committing.
- Should there be a common style for the scenarios? common file format? cyber range already has scenarios, should our scenario share these traits? (strategy issues, not necessarily implement issues)
- Our system should be a module of the cyber range setup. Is the game a separate system? should it be integrated into the cyber range with some sort of unification/cross format for scenarios?
- UI lies having more corners should not be a priority, as this would not bring more functionality to the project.
- Unify some interfaces into a common interfaces (group similar interfaces with a tag?)
- Match on strings? Types? Enums? Number?
  Initially , go with type safety (do not use string input, prone to typos. This would lead to things going wrong, usually with no clear indication as to what has gone wrong. Add a validator, with a try-catch, to check that

the type entered is actually present in the list. Alternatively, could use a stand-alone tool that validates the text/JSON/XML files).

7 20.02.2019 12.15

7.1 Meeting with a team doing a physical project

The supervisor, Mariusz, suggested that we meet with another team doing a project with similarities to ours. This team is working on a physical/digital hacking game where the participants are playing a digital game with physical components. The goal of the digital game is to score higher than you opponent. The goal of the metagame, however, is to hack the physical components of the game in order to turn the balance in your favor.
This meeting is mainly focusing on exchanging information that can be mutually beneficial to both of the projects, especially in the areas they are similar (mostly in the digital game department).

7.2 Notes from the meeting

- Meeting ended up mostly being brainstorming for the other team's project.
- Seems their project will gravitate towards changing the rule set and some aspects of an existing board game, which will not be compatible with our project further.

8 26.02.2019 10.00

8.1 Agenda for the meeting

- We were asked to attend the meeting for the board gaming group Tuesday 26.2 at 11.00 (the same day as this meeting)
- Last meeting with product owner revealed they want us to align our project with a DSL developed by a master student.
- Implement empty attribute fields for the system components that are not relevant for our system (have the fields available, but set them to null).

8.2 Notes from the meeting

- As Mariusz is unavailable for meetings in person for the next 2 weeks, there will be no official meeting but rather virtual meetings over discord.

9 15.03.2019 13.30

9.1 Agenda for the meeting

- Update meeting, discussing what has happened.

9.2 Notes from the meeting

- Make sure to highlight in the thesis and presentation that the visual UI is mostly to showcase the underlying system.
- Discuss in the report what we would have included and implemented if we had more time (and how)
- 2 use cases in the thesis, at least one simple scenario and one more complex scenario (possibly more).
- Document in the thesis why we did not make automatic tests (visual), how and when we did the manual tests.
- Either implement or discuss having a calculating script, giving a rough estimate of the difficulty level of the scenario,
- Explain in the thesis how we organized the work, how we made sure only 1 person worked on a scene etc.
- Figure out how Unity does random.range (and document this)

10 23.04.2019 13.45

10.1 Agenda for the meeting

- Update meeting, discussing what has happened.
- Discuss the project as is now, where no more implementations are planned.
- Is Unity a framework as well as an engine? (for professional programming report, as well as thesis).
- Discuss whether we should publish the project (based on suggestion from the product owner).
- Discuss the state of the bachelor thesis template, if the thesis should be written in past or present tense.
- Get some feedback on the design from another student.

10.2 Notes from the meeting

- Past tense is preferred.
- Cognitive walkthrough: task a tester do certain tasks with the system, while thinking out loud (Do with a teacher/classmate). Can be short. Fix the identified issues after every cognitive walkthrough session, but don't change the task between these. Do not speak during testing, as this will influence the results of the testing.
- Heuristic evaluation (10 different aspects/principles). Evaluate what works and what does not based on these different aspects (add this to the list of future work). Should have 4, one for each of the types of actions (scenario creator/attacker/defender/observer).
- have text size consistency, as well as button consistency (persistent icon usage).
- When starting connection, have the line follow the mouse pointer.
- Do not spawn all icons on the same place. Suggestion: have it "ghost icon" following the mouse after the spawn button has been pressed, until it is "placed" (clicked) somewhere.
- Clear indication of when a vulnerability has been added.
- Cannot delete only one vulnerability.
- Visual representation (in the editor) of entry points.
- visual representation that save was successful.
- Give a warning if an attempt to exit the scenario creator without saving.
- Have asterix for required fields on the right side of the input field.
- Show less of the background when in an internal menu for the scenario creator.
- Write up the fact that you are in the scenario creator, i.e. with a header in the menu stating that this is the scenario creator.
- Be able to exit the scenario creator menu back to the main menu (return button)
- Fix text color (inconsistencies/red text on buttons that are not quit/)
- Have a list of the number of components you have currently included in the scenario creator screen?
- Have a (non-functional) help prompt, to show that a help function has been thought of (put the functionality for this under future work).

- Use contrast checker (i.e. contrastchecker.com) to check the visibility of text (checking the text color with the background color to see if the text would be visible for people with visual disabilities.
- Have a separation on the right-side menu, in order to separate component information and the save/quit buttons.
- Have standardized text font (16 or higher). Make the menus larger in order to fit this.
- Show the time and date this document/scenario was last saved.
- Remove the tooltip text (when there is no relevant information). Replace this with a hover information box that follows the mouse?

11 2.05.2019 13.45

11.1 Agenda for the meeting

- Delete the deployment chapter from our thesis? As we have not actually deployed anything.
- Who to list as the contact person?

11.2 Notes from the meeting

- Have deployment through the NCR repo.
- Ask product owner who to list as the contact person.

12 6.05.2019 14.00

12.1 Agenda for the meeting

- Discuss the first draft for the thesis.

12.2 Notes from the meeting

- Make the text reflect to the note (”as you can see on...” also note that making the notes help us organize etc.).
- Make background for code listing lighter.
- Have the code listings clearer and more self-explanatory. Maybe cut the code listings down, in order to highlight just the certain areas we talk about. Maybe also explain it more in the text. (refactor example: main attributes of a client)

- 5.1. cut surplus stuff, cut "at that time". Do not use more words than necessary.
- 4.1 Explain what the duties, rights and obligations for each role. Have a small, separate section. Add build, deployment, testing (alternatively, state developers should test code), UI
- DSL: domain specific language to express some logic/concepts/relationships. State that the YAML is part of the DSL (there is more to it, but the YAML was the part most relevant to us).
- Separate future work into a separate chapter (between discussion and conclusion. have conclusion without sections).
- Use tt for variables used in text, in order to have them stick out.
- Use tilde ĩn order to have the space between word and a citing to always have a small space there.
- Latex package clever ref (cref) in order to reference things in the text more easily have consistency and styling for all the references.
- Use the term Unity3D everywhere, not Unity as this can be interpreted ambiguously.

I Notes from meetings with the product owner

These are the notes taken during the meetings with the product owner during this bachelor thesis.

Bachelor 2019

Meetings with product owner

Christian Bråthen Tverberg
Maarten Dijkstra
Nataniel Gåsøy

January-June 2019

Contents

1 Introduction

These are the notes from the meetings with the product owner, Muhammad Mudassar Yamin, during the course of this bachelor project.

2 11.01.2019, 11.00

2.1 Agenda for the meeting

- Sign project agreement.
- Set milestones.
- Discuss the structure and details of the planned product (+ show and discuss early sketches).
- Discuss the main focus of the task.

2.2 Notes from the meeting

- Decided to not do a lot of animations (no characters or environment, screen based game).
- 3 screens (scenario/ attacker/ defender).
- Decision tree based progress.
- Abstract physical security measures.
- Focus on OWASP top 10 vulnerabilities.
- Workers as an element/ sims for the game (motivation/ skills/ communication).
- Device creation with checkboxes(?).

It has been established that the main focus of the task is to create the engine that will control the scenario creator. It will mainly focus on covering the OWASP top 10 vulnerabilities, both from the attacker and defender's point of view. For testing the simulator builder, the plan is to test it with a bank website. Here the website of the bank is exposed to the internet, and utilizes several APIs.

"Defender: A bank is under attack, your responsibility is to keep the bank website secure (WAF to protect against internet, IES, IDS)".

"Attacker: Your task is to exploit the bank website (between bank website and internet, exploit one of the APIs etc)"

3 17.01.2019, 13.00

3.1 Agenda for the meeting

- Multiplayer: LAN/ local? cross-campus (with dedicated server)?/
- Go through the OWASP list of risks and prevention methods.
- Have each OWASP (web, mobile, IoT) as different modules? Focus on one at a time.
- Several levels to each component? (i.e. firewall level 1, 2, 3)? (if time, is this something we should focus on?)
- Discuss game design document/ notes.

OWASP web	**Prevention**
Injection	Keep (untrusted) data separated from commands/ queries
	Use prepared statements (sanitazion) ?
	Use a safe API (Which avoids use of the interpreter / uses a parameterized interface)
	Whitelist/ blacklist server-side input validation
Broken authentication/ session management	A single set of strong authentication and session management controls
	Where possible, use multi-factor authentication
	Enforce strong passwords
	Enforce password expiration dates
	Limit/ increasingly delay failed login attempts
Sensitive data exposure	Encrypt all sensitive data (in REST and in transit)
	Store passwords using strong, adaptive, salted hashing functions
XML external entities (XXE)	Do not evaluate external entity references within XML documents
	Patch/ upgrade all XML processors and libraries in use
	Disable XML External entity and DTD processing in all XML parsers in the application
	SAST tools (+ manual review) can help detect XXE in source code
Broken access control	Consistent and easy to analyze authorization module that is invoked from all your business functions
	Deny by default (except public resources)
	Log access control failures
	Rate limit API and controller access to minimize the harm from automated attack tools
	Log access control failures, alert admins when appropriate
Security misconfiguration	Deploy all new software updates and patches, run scans and do audits periodically
	An automated process to verify the effectiveness of the configurations and settings in all environments
Cross-site scripting (XSS)	Separating untrusted data from active browser content
	Escaping untrusted HTML request data
	Enabling a content security policy (CSP)
Insecure deserialization	Selecting an approach for protecting each user accessible object
	Don't accept serialized objects from untrusted sources
	Implement integrity checks (such as digital signatures)
	Logging deserialization exceptions and failures (alerting if a user deserializes constantly)
Using components with known vulnerabilities	Remove unused dependencies, unnecessary features, components, files and documentation
	Upgrade versions
	Add a security wrapping around components
	Monitor for libraries/ components that are unmaintained
Insufficient logging and monitoring	Use extensive logging (login, access control failures, server-side input validation failures
	Establish effective monitoring and alerting
	Establish or adopt an incident response and recovery plan

Table 1: OWASP top 10 security risks for web applications [4]

OWASP mobile	**Prevention**
Improper platform usage	Secure coding and configuration practices must be used on server-side.
Insecure data storage	Using threat-modelling for the mobile app, OS, platforms and frameworks. It is important to see how the app handles the following features:
	URL caching(request/response)
	Keyboard press caching
	Copy/paste buffer caching
	Logging
	HTML4 data storage
	Browser cookies
	Analytical data sent to 3rd parties.
Insecure communication	Use best practices e.g.:
	Assume that the network layer is not secure
	Apply SSL/TLS to transport channels for sensitive information
	Always require SSL chain verification.
	Alert users through the UI if the mobile app detects an invalid certificate.
Insecure authentication	Avoid the following insecure design patterns:
	Authenticating a user locally.
	Less authentication requirements on mobile.
	Use of any spoof-able values for authenticating a user(geo-location).
	(Remember Me) functionality storing password on the device.
Insufficient cryptography	Avoid storage of sensitive data
	Apply cryptographic standards
	Follow the NIST guidelines on recommended algorithms.
Insecure authorization	Verify the roles and permissions of the authenticated user
	Backend code should independently verify that incoming message match up with identifiers.
Client code quality	Maintain consistent coding patterns
	Write easy to read code that is well documented
	Prioritize solving buffer overflows and memory leaks over other 'code quality' issues.
Code tampering	During runtime try to detect a rooted device. Ways to check for rooted device:
	Check for test-keys(Check to see if build.prop includes the line "ro.build.tags=test-keys").
	Check for OTA certificates(Check to see if the file /etc/security/otacerts.zip exists)
	Check for several publicly known rooted apk's.
	Check for SU binaries.
	Attempt SU command directly.
Reverse engineering	Use an obfuscation tool, i.e.:
	Narrow down what methods/code segments to obfuscate.
	Tune the degree of obfuscation to balance performance impact.
	Obfuscate string tables as well as methods.
Extraneous functionality	Perform manual secure code review with tasks like:
	Verify that all test code is not included in the final build.
	Examine the app's configuration settings to discover any hidden switches.
	Examine all log statements to ensure nothing overly descriptive about the backend is being written to the logs.

Table 2: OWASP top 10 security risks for mobile applications [2]

OWASP IOT	**Prevention**
Weak, guessable or hardcoded passwords	Enforce strong passwords
	Enforce password expiration dates.
	Use two-factor authentication
	Not famously known/ publicly available passwords.
	Never give away password and don't store passwords in browsers.
Insecure network services	Avoid unneeded network services running on the device.
	Make sure ports not in use are closed
Insecure ecosystem interfaces	Better authentication/authorization.
	Better encryption and the need of having input/output filtering.
Lack of secure update mechanism	Set up a good, secure update mechanism.
Use of insecure or outdated components	Don't allow usage of outdated components.
	Deny access if something is found insecure.
	Update as soon as possible, if possible.
Insufficient privacy protection	Don't store passwords/other sensitive data on the devices.
	Encrypt all data at rest and transport
	Anonymize data where possible
Insecure data transfer and storage	Get better encryption or access control at any times(at rest/in transit/during processing).
Lack of device management	Get better security support for deployed device, including update management, systems monitoring, asset management and response capabilities.
Insecure default settings	Review default settings, and change to suit the environment
Lack of physical hardening (poor physical security)	Use minimal number of device access ports
	Sensitive application functions should not be accessible through USB

Table 3: OWASP top 10 security risks IoT [3] [1]

3.2 Notes from the meeting

- Have it in the cloud, in order to run the game for both players wherever the machines are. Need to check what kind of connection and protocols unity uses for their multiplayer functionality.
- Skill level? motivation? resources?
- Make buttons scriptable for AI training later
- Have levels of skills and systems (i.e. firewalls, lvl 1 - 2 - 3)
- Make a menu classification system for defender and attacker
- Have much focus on the SIMS as well (focus on employee security as well as technical security)

- Have different categories for the simulator (i.e. technical/ student category where you see the system topology, awareness category where you see a simple screen etc.)

3.3 E-mails with product owner

- We have thought about this in an over-complicated manner. Instead of specific types for attacks, we have clarified with product owner through email (18.01.2019) that he wants a more general attack and defence set of options (i.e. SQL injection, Cross-site scripting) etc. from OWASP as well as for the SIMS (human factor).
- There is to be a large focus on human factors (30-50% of the vulnerabilities)

4 31.01.2019, 13.00

4.1 Agenda for the meeting

- Use Design document from project owner as appendix in bachelor (original user specification document)?
- Turn-based vs real time (turn based might be better for reflection and learning)
- Team vs single player teams?
- Showcase visual mockup
- At the end, have a demo? good show for the capabilities of the system
- User defined attacks? if attacks and defenses, and 3 levels on each of them, lots of logic behind it? predefined list of attacks and defences instead? (need to check in order to not do unnecessary work).

4.2 Notes from the meeting

- DARPAs cyber grand challenge
- Assets? commission from design students?

5 07.02.2019, 12.00

5.1 Agenda for the meeting

- Show visual demo.
- The cyber range buys assets? If the cyber range wants to find and buy an asset pack, we can implement that.

- Turn based easier to analyze later than for a real-time system. Mariusz suggests turn-based in order to get a feel for it (not jumping straight into deep water). Post action review makes it easier to take it step-by-step in turn based system.
  It also makes it easier to develop AI later, both for gameplay and for giving player suggestions based on the gamestate.

5.2 Notes from the meeting

- Cyber security seminar in two weeks(?), have demo for then.
- Present the paper+game June 25th, at a seminar?
- Positive feedback for the visual and technical demos
- Have data here, especially from testing phase.
- Create a flowchart for the entire system (gameplay+scenario creator)
- Be able to send a random attack (will affect attacker's economy)
- Log in a way to be able to re-run the scenario based on the log output (extensive enough logging, every step/action)
- Aiming for a simulation, new attacks and defenses will be regarded as future work
- Real-time is the innovative part of the game. Product owner still wants to have it real-time, even after the suggestion from students and school supervisor
- Only one action should be allowed at the time, both for attacker and defender.

6 14.02.2019, 12.00

6.1 Agenda for the meeting

- Discuss flowchart
- How would observer function? log? screen? (size of package sent)
- Probability of success for attacks and defense. higher level of attack and defense increases probability of success.

6.2 Notes from the meeting

- Have restrictions on what can be connected to what in the scenario creator?
- Assets are fine, just buy them (as long as the total price is below 2000NOK)
- Consider having a replay ability? Read from the log, replay the actions real-time.
- Have the defender screen see the scenario all the time, color the network components based on where the attacker and defender clicks (where they are operating at the moment, color coded).
- Use probability, but remember to document (and log) it.
- Attacker be able to sell resources during gameplay (if any has been obtained)? sell personal data/exploits on the black market.

7 21.02.2019, 12.00

7.1 Agenda for the meeting

- Nothing new to present at this point.

7.2 Notes from the meeting

- A master student making a DSL (domain specific language) for the system that will be utilizing the real-time cyber strategy game. We were asked to present the work so far to him.
- There is a request to align our project with the DSL plan, with regards to components, attributes etc.

7.3 Extra meeting 22.02.2019

An extra meeting for the projects concerning the NCR (Norwegian Cyber Range). There is a presentation of the DSL project we are to (possibly) be aligning with.

- The YAML document will specify the vulnerabilities, the system, events that could trigger (as well as their trigger),
- This system only supports known vulnerabilities (does not support zero-day vulnerabilities). Can be worked around by putting "new" vulnerabilities being put into the database used for the DSL.
- Implement attribute fields for the system components that are not relevant for our system (have the fields available, butsave them as empty objects (0/null).

8 28.02.2019, 12.00

8.1 Agenda for the meeting

- Discuss the draft for what we will and will not include from the DSL model
- Discuss system components (API/Server/Switch/Firewall/Internet...)

8.2 Notes from the meeting

- Include all the items on the list in case they will be used later, but save them as empty objects (0/null).
- Talk with another PhD student about the list of the components (the student in question is product owner for another bachelor with similarities to our project).
- Add empty buttons to the menu for the scenario types we are not covering (all scenario types that are not red vs blue)?

9 07.03.2019, 12.00

9.1 Agenda for the meeting

- Discuss the presentation of the scenario creator on Friday (08.03.2019).

9.2 Notes from the meeting

- Save events as a YAML file (separate files for attacker and defender)
- Observer be able to check the status of objects.
- Prepare a small presentation for tomorrow.

9.3 Feedback from the presentation

- If an attack on a node is successful, what happens then? does the state of the node change?
- Menu in the scenario creator to set up defender and attacker data and/or stats.
- Must a node be compromised in order to gain access to other nodes in the network?
- Implement privilege escalation? (at this time, privilege level is not relevant for the gameplay as it has no effect)?

- For future work, use different probability model or decision models in order to determine whether an attack is successful or not (currently there is only a random number generator).
- Find out what the default number generator used by unity is, and state this clearly in our documentation.
- Make sure to mention in the report that the purpose for this project is to make a framework, not a complete system. The game is made in order to showcase the scenario creator, not to be a full-fledged game in and of itself.
- Start with finding assets to the things we already know we need, find assets for other things later when a more complete list is made (Computer, internet, API, switch, firewall...).

10 14.03.2019, 12.00

10.1 Agenda for the meeting

- Meet another PhD student in order to get some feedback and scenarios.

10.2 Notes from the meeting

- Meeting was cancelled.

11 21.03.2019, 12.00

11.1 Agenda for the meeting

- Present the system and project so far to Simon McCallum.
- Discuss list of components based on the DSL and what the product owner wishes to have included.
- Be able to put a list of implemented defenses on each system component.

11.2 Notes from the meeting

- Could not establish contact with Simon, sending the game document (user manual) instead as a way of getting feedback.
- Product owner just wants the vulnerabilities on the component that an attacker can exploit, and not the list of defenses that a component has (from the scenario creator).

12 04.04.2019, 12.00

12.1 Agenda for the meeting

- Update on the progress of the project, as we are in the finishing stages of implementation and will not start implementing any new functionality.
- Update on feedback from Simon.

12.2 Notes from the meeting

- Show-casing the integration of all the different components into the same program, as opposed to the previous presentations where each component for the system was presented separately.
- There has been no contact with Simon.
- Have the 2 scenarios discussed ready by June, for the presentation of the thesis.
- Make vulnerabilities dynamically in the scenario creator instead of just having the pre-defined list?
- Have product owner inspect the show-casing scenarios after these have been made for validation.
- Consider having the project published (on i.e. Steam).

13 11.04.2019, 12.00

13.1 Agenda for the meeting

- Update on the progress of the project.
- Discuss plan for user testing.

13.2 Notes from the meeting

- Norwegian cyber challenge on the 27th of April, possibility for testing the system on this day? Run a specified scenario.
- They are not looking for win- or lose situations, only reflections around the scenario so there is no issue that we have not had time to implement defined win- and lose conditions.
- Have a scenario that is hard to solve (during the given time), but is more focused on their methodology and their reflection on the situation.

- Have the "Norwegian cyber range" written in the observer mode. Maybe have a logo present? Depends on if we can get a logo in time from the Cyber range.
- Use a pre-defined scenario we will create, based on the specifications of the product owner.
- Product owner wants a restriction system in place, that will make it impossible to proceed from one node to another if a vulnerability is not found (if there are vulnerabilities available).

14 02.05.2019, 13.00

14.1 Agenda for the meeting

- Discuss the report.
- Should this be deployed for the NCR now?
- Who should be listed as the contact person? product owner? an NCR leader? both?

14.2 Notes from the meeting

- The source code and installer should be shared on the NCR repo
- The contact person listed shall be determined later

14.3 E-mails with product owner

- The product owner should be listed as the contact person.
- The source code and installer will be uploaded to the NCR internal systems.

References

[1] Checkmarx. Owasp top 10 for iot - explained. https://www.checkmarx.com/wp-content/uploads/2015/07/OWASP_TOP_10_IoT_Explained.pdf, 2018. Last visited: 2019-01-16.

[2] The OWASP mobile Security Team. Owasp top 10 mobile application security risks. https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10, 2017. Last visited: 2019-01-16.

[3] The OWASP IoT Security Team. Owasp top 10 iot application security risks. https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project, 2018. Last visited: 2019-01-16.

[4] The OWASP web Security Team. Owasp top 10 web application security risks. https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf, 2017. Last visited: 2019-01-16.