Simulating Network Lateral Movements through the CyberBattleSim Web Platform

by

Jonathan Esteban

B.S. Computer Science and Engineering,
Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author . . . .
Department of Electrical Engineering and Computer Science
January 13, 2022

Certified by . . . .
Michael Siegel
Principal Research Scientist
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Simulating Network Lateral Movements through the CyberBattleSim Web Platform

by

Jonathan Esteban

Submitted to the Department of Electrical Engineering and Computer Science
on January 13, 2022 , in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Modern cyber attacks demand immediate action plans based on an overwhelming amount of information and options. Microsoft has made available a highly parameterizable model of enterprise networks with the capability of simulating automated cyber-attacks. We provide an extension of this project by means of a web platform. The platform allows a user to model an enterprise network topology, interact with the topology manually, and simulate an automated adversarial agent. Leveraging the CyberBattleSim toolkit, we enable the swift prototyping of different network configurations that can then be analyzed by a defensive security team member either manually or automatically through the automated agent. We demonstrate that the platform can simulate any network topology supported by CyberBattleSim as well as evaluate different Q-Learning strategies. This in turn can provide us with valuable insight regarding the progression of cyber attacks, aiding us at generating appropriate cyber-attack response plans.

Thesis Supervisor: Michael Siegel
Title: Principal Research Scientist

Acknowledgments

This thesis is dedicated to my family and friends, who were each a beacon of inspiration throughout my MIT career.

I would also like to express my everlasting gratitude towards my supervisors, Dr. Michael Siegel and Dr. Keman Huang, for lending me their support and making me feel welcomed at the CAMS (Cybersecurity at MIT Sloan) research laboratory. Additionally, I wish to thank BV TECH S.p.A. for their collaboration in this project.[1] Finally, I would like to thank the CAMS community for providing me with a tremendous amount of invaluable feedback throughout my research and a familial environment that I could fallback to during the tough and enduring Covid-19 pandemic.

Again, I cannot thank them enough, I am incredibly grateful, and I wish them all success along their own professional and personal journeys.

[1]This work was funded in part by "Fondo Europeo di Sviluppo Regionale Puglia POR Puglia 2014 – 2020 – Asse I – Obiettivo specifico 1a – Azione 1.1 (R&S) - Titolo Progetto: Suite prodotti CyberSecurity e SOC" and BV TECH S.p.A.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

Modern cyber attacks demand immediate critical decision making [1]. Determining the optimal response to an adversary's attack to an Industrial Control System (ICS) is a difficult challenge given the overwhelming amount of information and options ICS operators have at their disposal. Actions configured to preserve the system's integrity come at different trade-offs for the system's availability and security. [2] [3]

Furthermore, as an ICS operator imposes security policies during a cyber attack, an adversary is able to acquire new information and change their attacking strategies. This was seen in the case of the Attacks of Ukraine's Power Grids, which suffered two cyber attacks within a year. A post-mortem analysis suggested that based on their experience with the first attack, the attackers were able to adapt to new challenges and improve their adversarial strategy. [4]

The analysis also proposed a series of active defense recommendations. Among these was a call to train both IT and OT network personnel in cybersecurity incident response plans. The authors also recommended the development of active defense models that visualizes and predicts the evolution of cyber attack strategies. This thesis aims at tackling both of these recommendations.

To achieve these goals, we have developed a cyber-attack simulator platform: an interactive web application that could help business professionals and operators improve their decision-making abilities when faced with cyber attack crises. To achieve this, we leveraged Microsoft's CyberBattleSim research toolkit. CyberBattleSim allows for the simulation of post-breach lateral movement during a cyber attack. [5] The toolkit abstracts a fixed network topology into a collection of computer nodes, each with their own predefined vulnerabilities that an automated adversary could exploit in order to continue moving through the network. CyberBattleSim uses OpenAI Gym internally, thus providing an interactive environment for researchers to create and apply different reinforcement learning models on the model network. More information regarding CyberBattleSim can be found in Section A.2

1.0.1 Reinforcement Learning Within Cyber Security

Reinforcement learning is a type of machine learning with which autonomous agents learn how to conduct decision-making by interacting with their environment. [6] [7] Agents may execute actions to interact with their environment, and their goal is to optimize some notion of reward. One popular and successful application is found in video games where an environment is readily available: the computer program implementing the game. [8] The player of the game is the agent, the commands it takes are the actions, and the ultimate reward is winning the game. The best reinforcement learning algorithms can learn effective strategies through repeated experience by gradually learning what actions to take in each state of the environment. The more the agents play the game, the smarter they get at it. Recent advances in the field of reinforcement learning have shown we can successfully train autonomous agents that exceed human levels at playing video games. [9] Additional information on Reinforcement Learning can be found in Section A.1

1.0.2 Contributions

This thesis offers the following contributions. First, a user interface to model the network topology and computer node vulnerabilities. Second, an human-interactive attack simulator that provides a sand-boxed environment to help red team users predict the evolution of cyber incidents and understand the consequences of their response plans. Finally, an automated attack simulator that employs the Q-Learning reinforcement learning technique to evaluate the network's security. The reward function of the automated adversaries is based on the discovery and ownership of computer nodes in the network. Thus, the reinforcement learning model outputs the optimal action to compromise the entire network.

1.0.3 Motivation

Our main motivation for this project lies in enabling security experts to investigate how automated agents interact within simulated network environments. We hope to see this project be utilized by the cyber-security research community to test different automated attack strategies. Lastly, we would like to reciprocate the gesture of Microsoft open-sourcing CyberBattleSim; we hope to extend their contributions by providing a streamlined user-interface that effectively showcases the modeling and simulation components.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Related Work

To our knowledge, there is no publicly available frontend-interface for CyberBattleSim. In fact, research that makes use of the toolkit is very limited. We suppose that this is due to the fact that the CyberBattleSim project is relatively new. However, we are confident that the toolkit's goal of enabling researchers to investigate RL learning in the context of computer networks will garner academic attention in due time.

The paper "Incorporating Deception into CyberBattleSim for Autonomous Defense" by Walter et. al. [10] demonstrates that CyberBattleSim is readily extensible and can be used to investigate the effects of cyber deception within the toolkit. These deceptive elements included Decoys, Honeypots, and Honeytokens, each with their own set of penalties. They investigated how these deception techniques influenced the maximum expected cumulative reward of the automated adversary as well as the percentage of attacker wins and the amount of wasted resources. The paper showed that, as expected, the attacker's rate of progress is inversely proportional to the amount of deceptive elements on the network. Thus, the authors set the stage for other researchers to design advanced autonomous defender agents that can employ deceptive strategies.

Work by Standen et. al on "CybORG: A Gym for the Development of Autonomous Cyber Agents" [11] was similar to that of CyberBattleSim in that the authors developed a network simulation environment (CybORG) that can also employ automated, decision-making agents. In contrast to CyberBattleSim, the CybORG virtual Gym also supports emulation, which allows for a more realistic training of agents. For example, the adversarial action space can support the Metasploit Framework [10][12] and both attacker and defender agents can execute terminal commands. The results of this paper demonstrated that AI agents can be trained on simulated networks and then be run on an emulated infrastructures. It remains to be seen if the project will eventually support blue-agent training. Lastly, the project appears to be in the early stages of development and, at the time of writing, the paper's authors have not made their codebase publicly available.

Previous work on reasoning the optimal strategies to defend against Advanced Metering Infrastructures (AMI) was conducted by Ismail et al. in "A Game Theoretical Analysis of Data Confidentiality Attacks on Smart-Grid" [1]. The authors of the paper were able to construct a game-theoretical model of AMI smart grids to evaluate offensive and defensive patterns. In particular, they worked towards finding the Nash equilibrium within different scenarios, in which neither the attacker nor the defender may improve upon their strategies. The authors were able to derive a set of devices within AMI that would yield the most reward when compromised. In addition, the authors identified the minimum defense budget needed on each device in order to protect them against cyber-attacks.

Also within the space of applying game theory to cyber-security lies "A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security" by Khoury et al [13]. The model presented in the paper leverages multi-agent reinforcement learning (MARL) to run a game between an adversary and cyber-physical system (CPS) operator. The authors propose a hybrid approach based on game theory as a tool to formalize the game interactions between the human and adversaries within a CPS environment. To simulate the virus spread within a CPS network the authors collected a predefined known and discovered vulnerabilities, derived optimal attack sequences and defense policies using MARL and Q-learning, and finally created a simulation framework composed of a network simulator and an reinforcement learning toolkit. From their results, the authors determined that MARL agents learn the best policy for an automated response at run-time.

The risk assessment of attack graphs was studied by Munoz-Gonzalez et al. in "Dynamic Security Risk Management Using Bayesian Attack Graphs". [14] In the paper, the authors were able to leverage Bayesian networks to model attack graphs and evaluate real-world network vulnerabilities. The authors conducted this study to better help system administrators react to cybersecurity threats. They found that using Bayesian networks allowed them to effectively measure the probabilities of consecutive, successful attacks.

This project contributes to the field of network simulation by extending Microsoft's CyberBattleSim project; we present a graphical web interface to model and simulate enterprise networks. To ensure interoperability with CyberBattleSim, we directly exposed CyberBattleSim's inner mechanisms through an application programming interface (API) and created a frontend wrapper for the parent project. Thus, we provide a user-interface for creating network topologies, exploring topologies as a human attacker, and running automated AI strategies to compromise simulated network environments.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Approach

Our technical contributions include a full-stack application that allows for the modeling of network topology and visualization of cyber attacks on this network. This was achieved using the frameworks Vue.js for the frontend and Flask for the backend. The frontend codebase has three main components: network modeling, human-interactive simulation, and AI-agent simulation. The backend Flask server receives actions from the user interface, passes them into the CyberBattleSim toolkit, which in turn relays the response back to the user interface.

3.1 Network Modeling

The network modeling component is where the user can create and tweak the network topology abstraction. This network topology is visualized through a graph, where the nodes represent a computer or computer system in the enterprise network and directed edges point to another node obtained by exploiting a vulnerability or connecting via a leaked credential.

Besides adding or removing nodes to the graph, a user may edit various attributes of a node, including: intrinsic value, vulnerabilities, services, available ports and firewalls. These attributes are defined within the CyberBattleSim project and are described in Table 3.1. Many of these attributes have their own nested properties, allowing the user to finely specify a node's behavior.

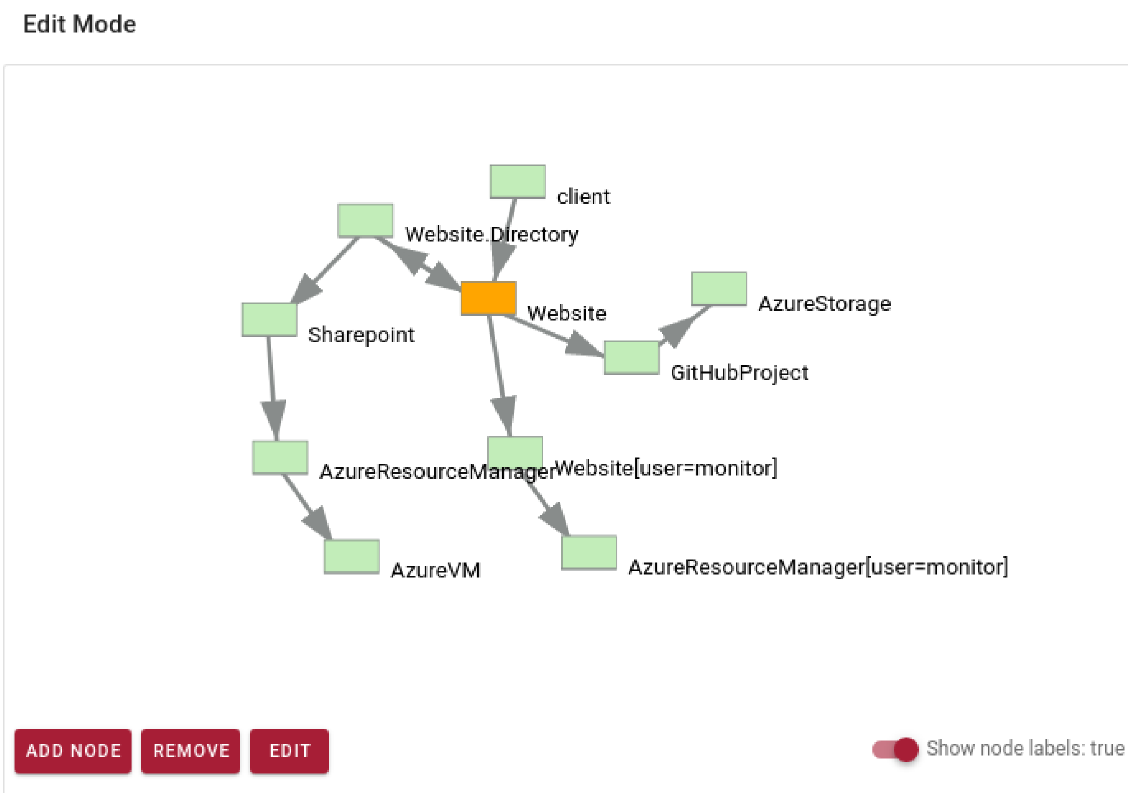


Figure 3-1: Network modeling interface on sample network topology.

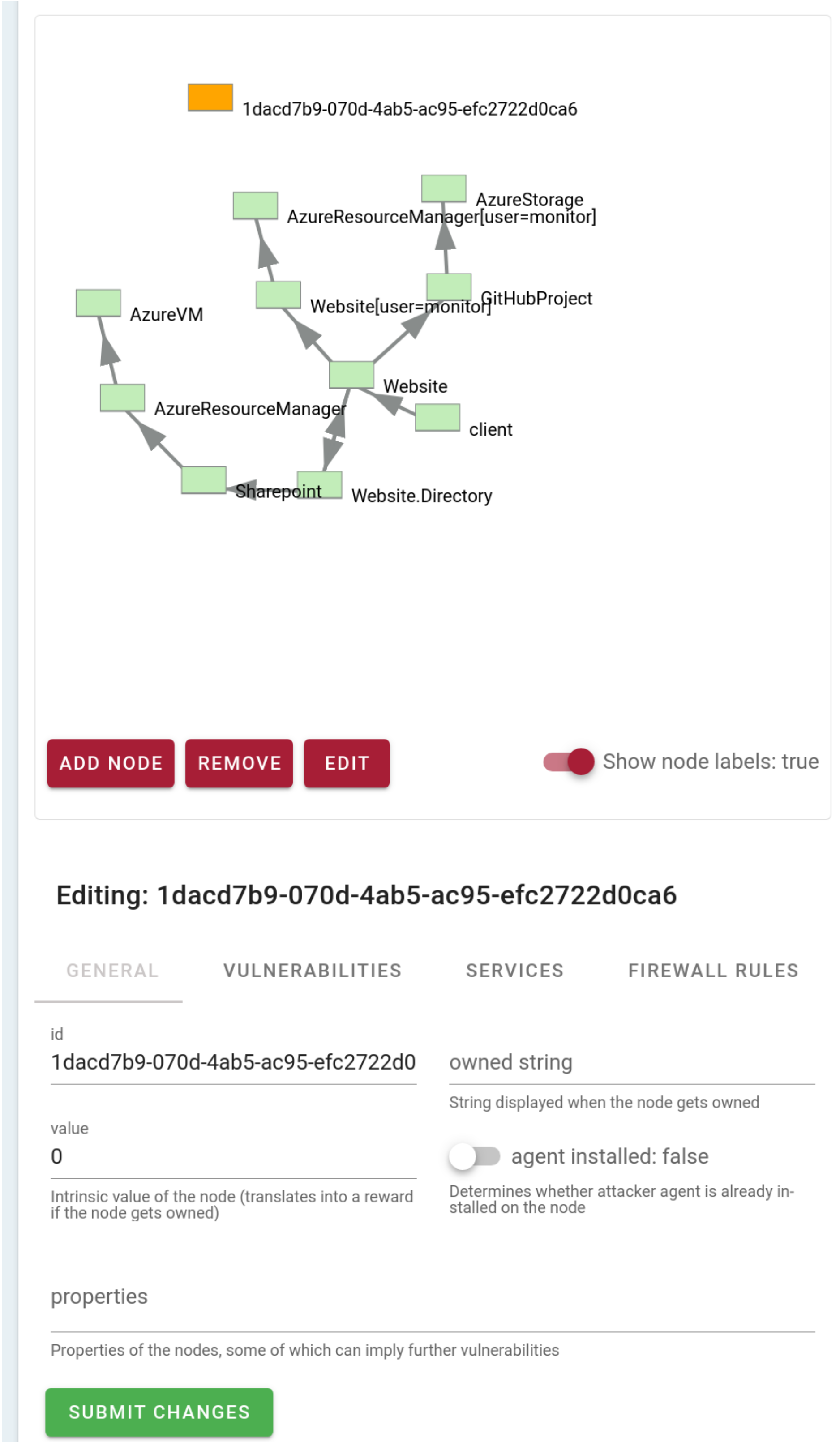


Figure 3-2: The network modeling page, showcasing a newly added node.

Table 3.1: Technical Node Attributes

name	**description**
services	List of port/protocol the node is listening
vulnerabilities	List of known vulnerabilities for the node
value	Intrinsic value of the node (translates into a reward if the node gets owned)
properties	Properties of the nodes, some of which can imply further vulnerabilities
firewall	Firewall configuration of the node
agent installed	Attacker agent installed on the node? (i.e. is the node compromised?)
privilege level	Escalation level
reimagable	Can the node be re-imaged by a defender agent?
owned string	String displayed when the node gets owned
status	Machine status: running or stopped

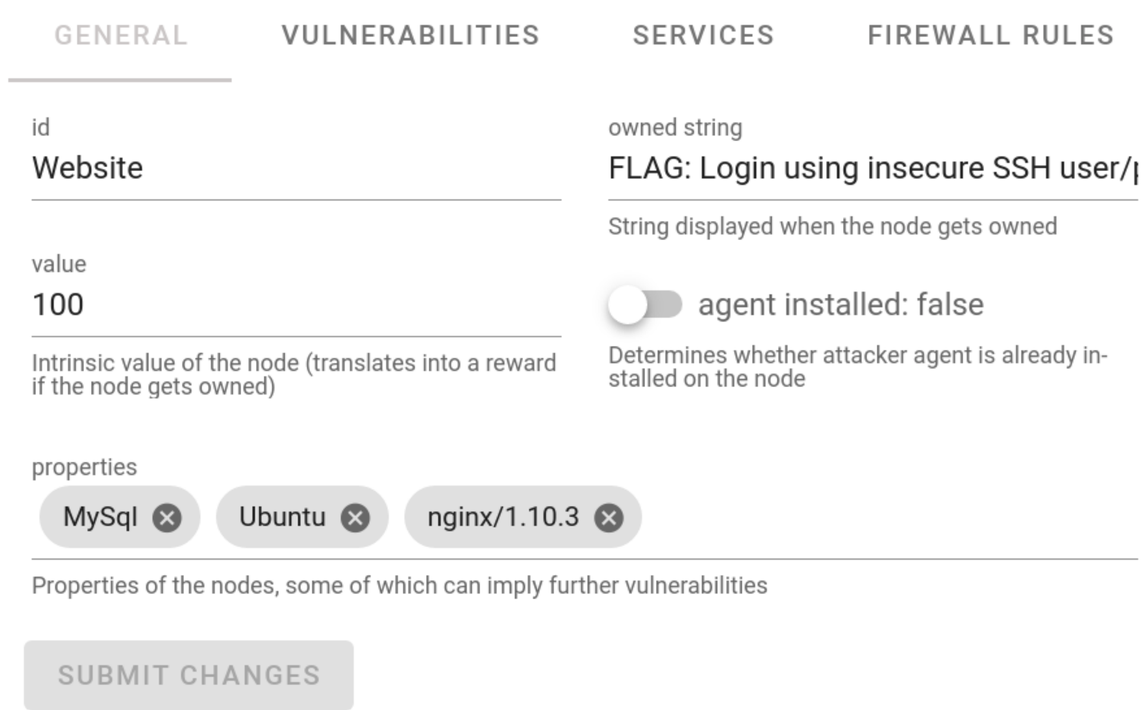


Figure 3-3: "General Properties" tab within the network modeling interface.

3.1.1 General Properties

The main properties of a node are shown within the "General Properties" tab. These properties include: node ID, the intrinsic value of the node, the text displayed when the node is compromised, a boolean representing whether an adversary has already captured the node, and property tags. Newly created nodes are instantiated with universally unique identifiers (UUID) as their ID. This preserves the invariant that no two nodes will have the same ID when created. The frontend form validation also ensures that ID uniqueness is preserved. In order for a proper CyberBattle Simulation to take place, at least one agent should be installed within a node. This ensures that the agent has an initial environment to attack from.

3.1.2 Vulnerabilities

Node vulnerabilities are abstracted with the following details in mind: outcome type, cost of exploit, rate of successful exploitation, rate of detection, and whether the vulnerability requires local or remote access to be executed. An example of a remote

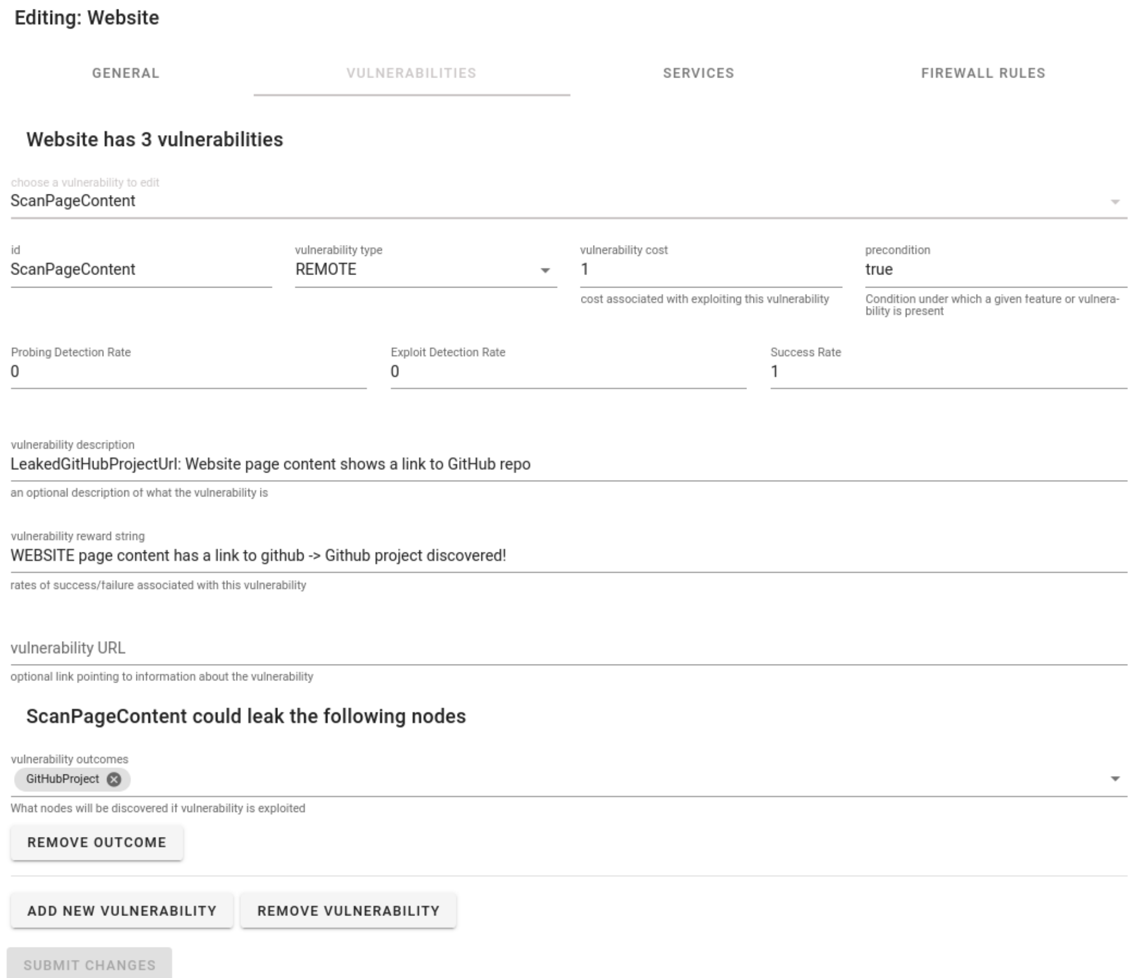


Figure 3-4: "Vulnerabilities" tab within the network modeling interface.

vulnerability could be a publicly hosted site exposing SSH credentials. Conversely, a local vulnerability could be extracting authentication token from a stolen device or escalating to administrator privileges from within the node.

CyberBattleSim provided us with several predefined outcome categories, including: leaked credentials, leaked references to other computer nodes, leaked user data, and privilege escalation on the node. Vulnerabilities can also be labeled as remote or local. Once a vulnerability has been exploited, the outcome is presented to the adversary along with the reward associated with the value of the node.

3.1.3 Services

Along with vulnerabilities, a node may also have running services. Services describe processes which run on an exposed port which can be configured to require credentials for authentication. For example, a web browser may expose an HTTPS service and a file transfer tool may expose an SSH service under a credential.

3.1.4 Firewall Rules

Finally, a user may add firewall rules to a node. Firewall rules can be used to block or allow certain ports. These rules can be defined for both outgoing and incoming traffic. Ports that are not explicitly allowed in the configuration are automatically assumed to be blocked. That said, explicitly blocking a port allows a user to provide a reason for the block.

As the user modifies the enterprise network abstraction model on the frontend, the changes are reflected on the CyberBattleSim backend model. Once the user is satisfied with the current topology, they may now use the human-interactive attack simulator or the automated attack simulator.

3.2 Human-interactive simulation

In red team versus blue team dynamics, the red team consists of offensive security strategists who try to attack a company's cyber-security defenses. The blue team in turn, defends against and responds to the red team's attack. We implemented the use-case of a human red team player who tries to attack an organization's cybersecurity defenses. In the scope of our project, a blue team member would design the network topology, as described in the previous sections, and would hand it over to the red team player for them to try to compromise. The red team player starts off in control of the node that the blue team player has configured to be initially breached. This starting node may have low privileges, and may represent the gateway between public and private domain, such as a web server. On this page, the player is presented with

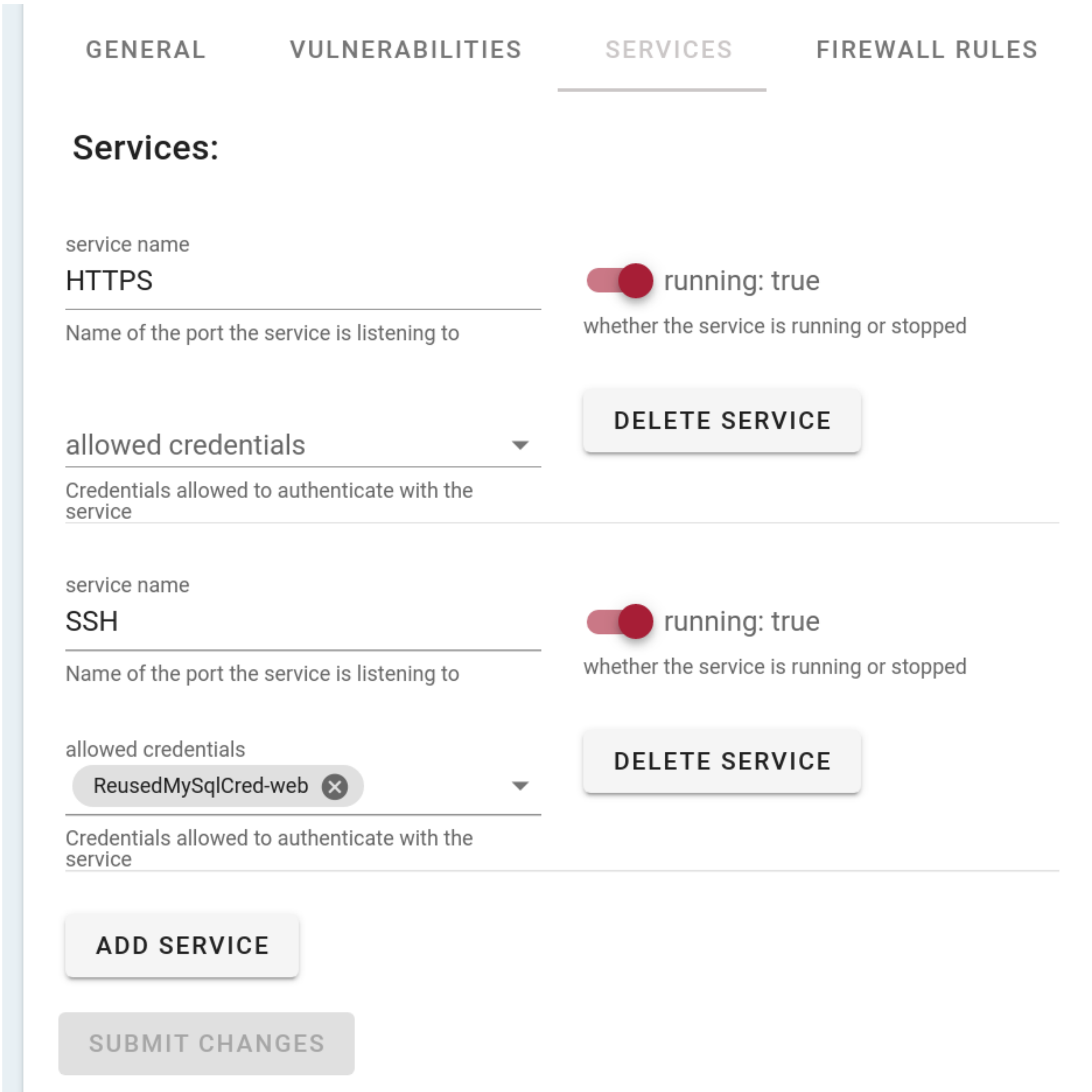


Figure 3-5: "Services" tab within the network modeling interface.

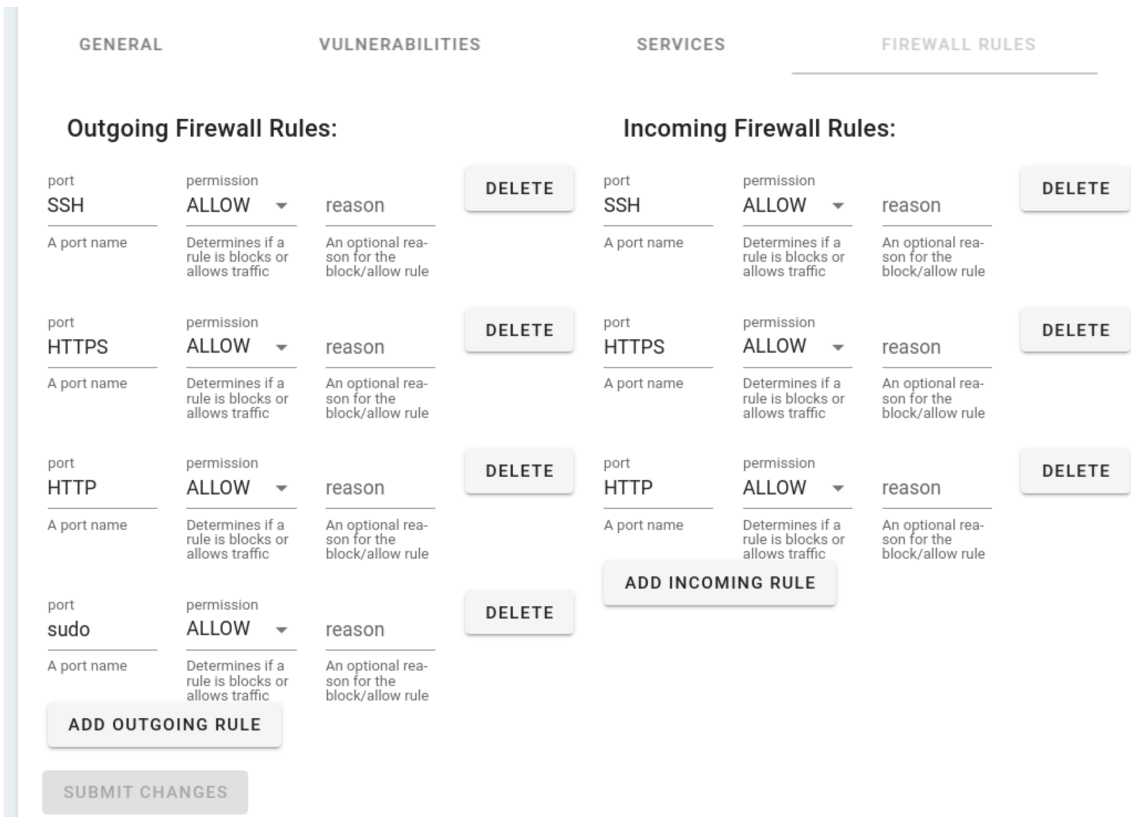


Figure 3-6: "Firewall rules" tab within the network modeling interface.

a sub-graph containing discovered (green) and owned (red) nodes, a list of actions for the currently selected node, and logs that inform the player of rewards or penalties.

The red team player's goal is to maximize their cumulative reward by incrementally discovering and taking ownership of nodes in the network. This component of the platform allows a human to move through the sand-boxed network, discovering new nodes as they exploit new vulnerabilities and acquire hidden credentials. This mode could provide valuable insight into how a human player would approach compromising the network. As designed by Microsoft's CyberBattleSim, the environment is partially observable, meaning that the agent does not know of the nodes and edges of the network graph in advance. The red team player takes actions to gradually explore the network from the nodes it currently owns. We support three kinds of actions, which allows the player to run exploits as well as explore the network that is visible to them. These actions are: running a local attack, running a remote attack, and connecting from a source node via learned credentials. Local actions require that the node where the underlying operation would take place is already owned by the player. After a node gets discovered or owned, the player is given a reward, which represents the intrinsic value of the node.

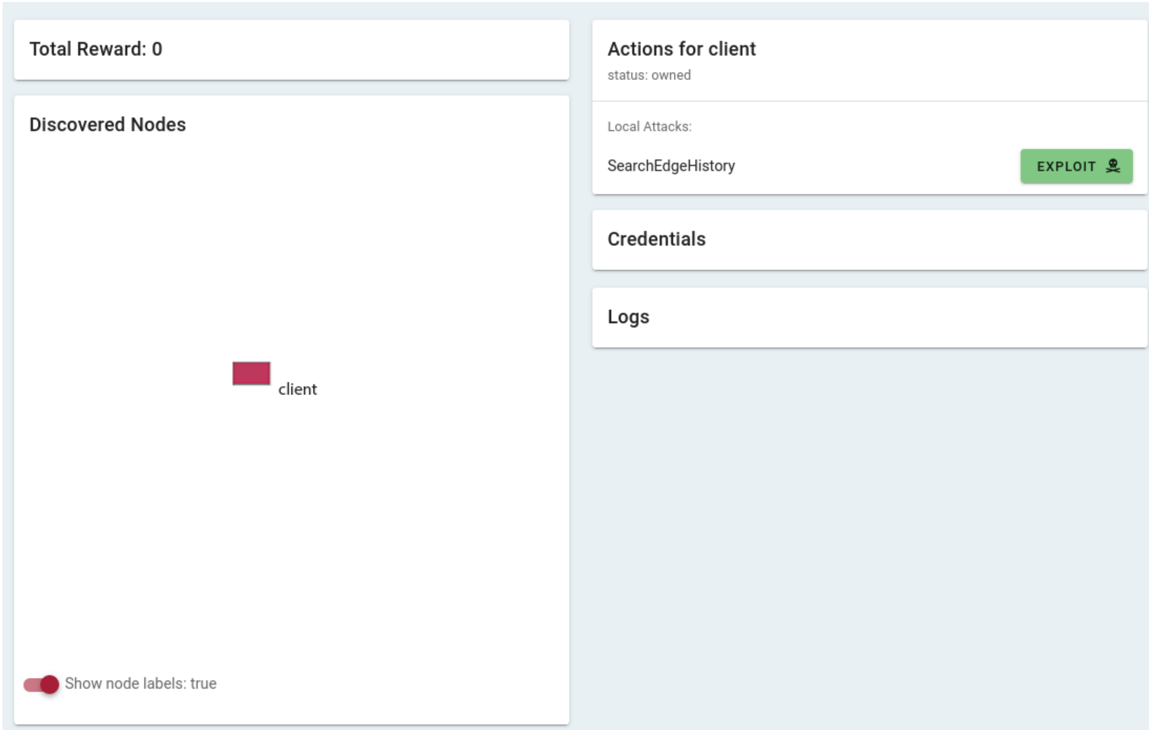


Figure 3-7: Initial, post-breach interface on sample network topology. Only one node has been compromised, accumulated reward is 0, and the logs are empty.

3.3 AI-learning simulation

We also implemented the use-case of an automated AI player playing as the attacker using Q-Learning, a type of reinforcement learning algorithm used by the CyberBattleSim project. In this scenario, a blue team member would design the network topology, input the specific AI learning simulation parameters (as defined in Table 4.1) and run the simulation. This component of the site allows a for an AI adversary to move through the sand-boxed network, discovering new nodes as it exploits new vulnerabilities and discovers hidden credentials. This mode can be used to find Cyber Kill Chains, evaluate different Q-Learning strategies and learn about different attack paths at a faster rate than a human player.

The component's page presents the user with a list of parameters, and once submitted, shows a live progress of the AI learning algorithm. As the simulation is running, the user may view the reward-over-time chart and the sub-network that the AI agent can currently observe and interact with. Once complete, a gallery of figures

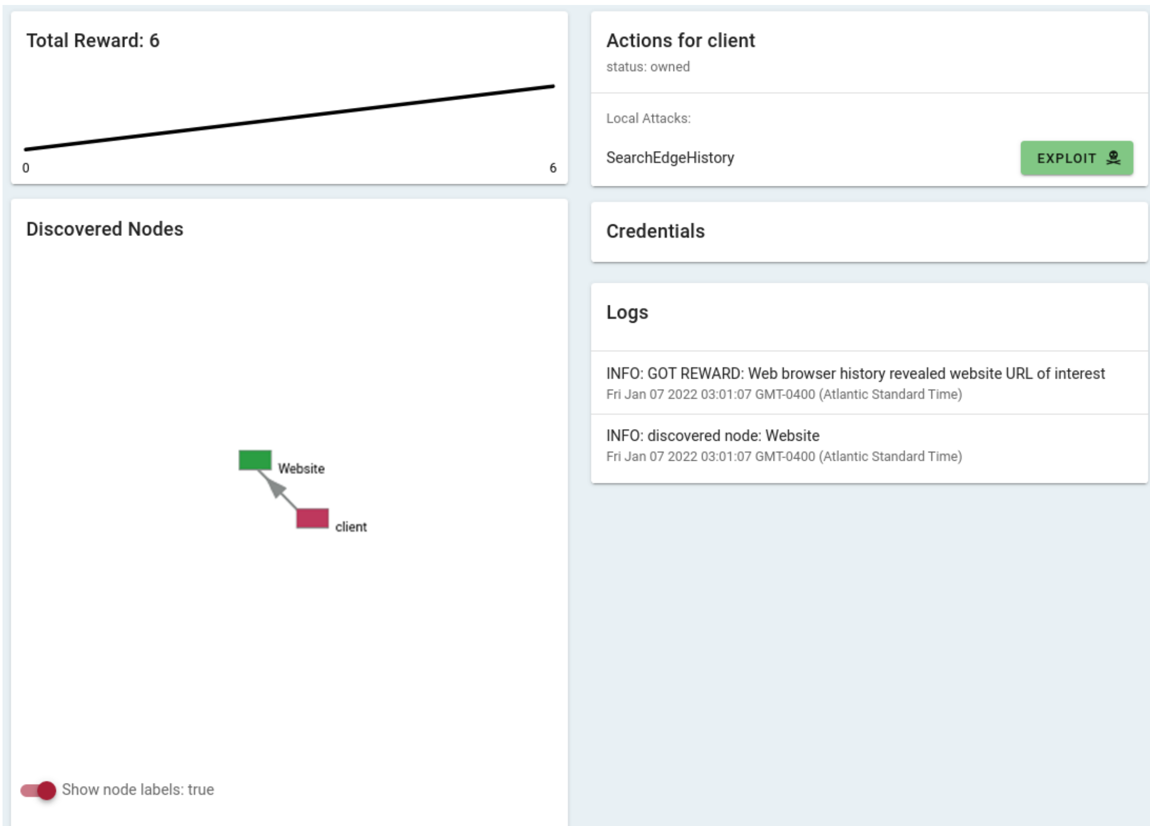


Figure 3-8: Attack progression on sample network topology. One node has been compromised and another node has been revealed, accumulated reward is 6, and the logs have informed a discovery and reward.

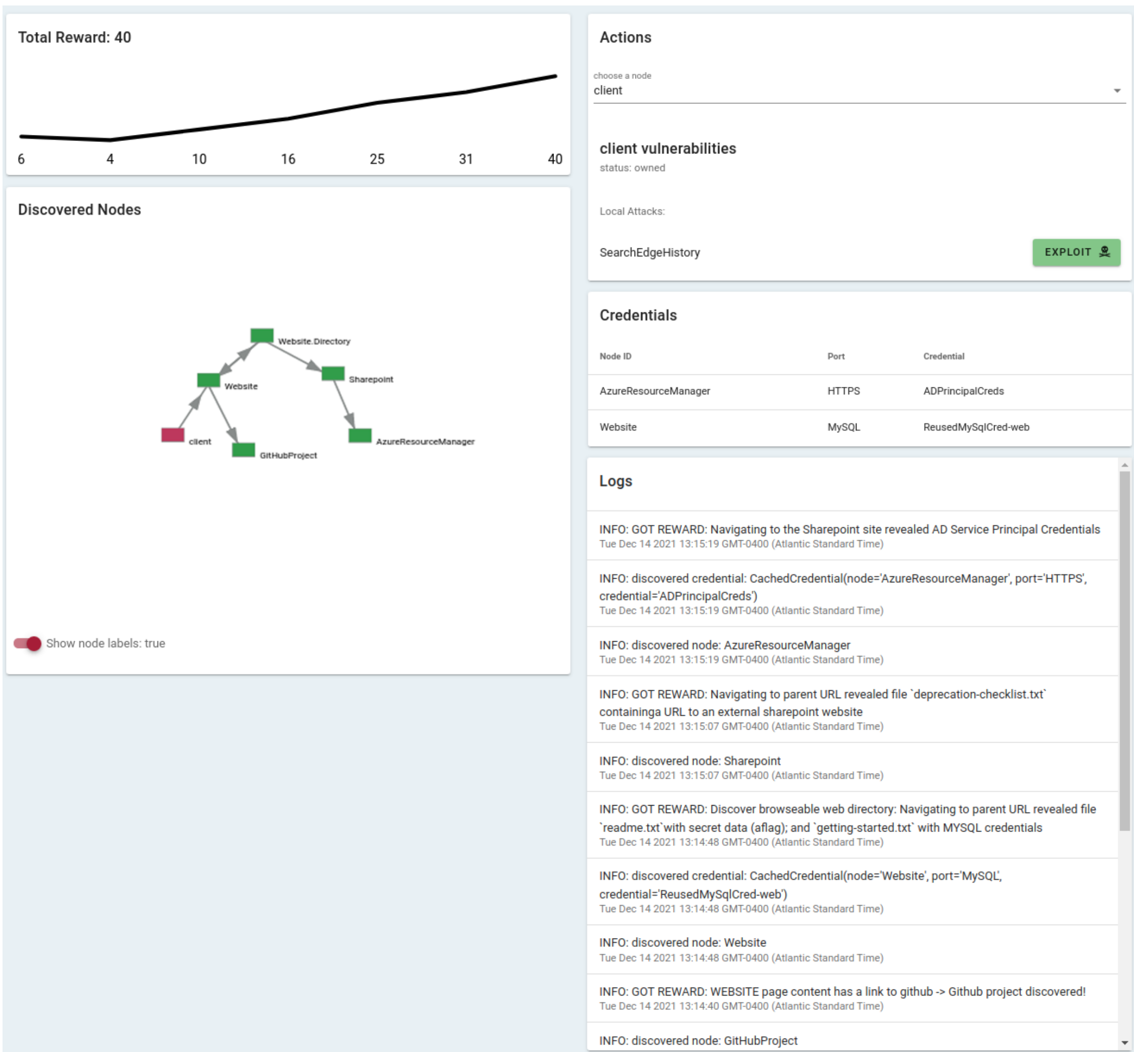


Figure 3-9: Attack progression on sample network topology showcasing a blocked action via a firewall, resulting in a score penalty.

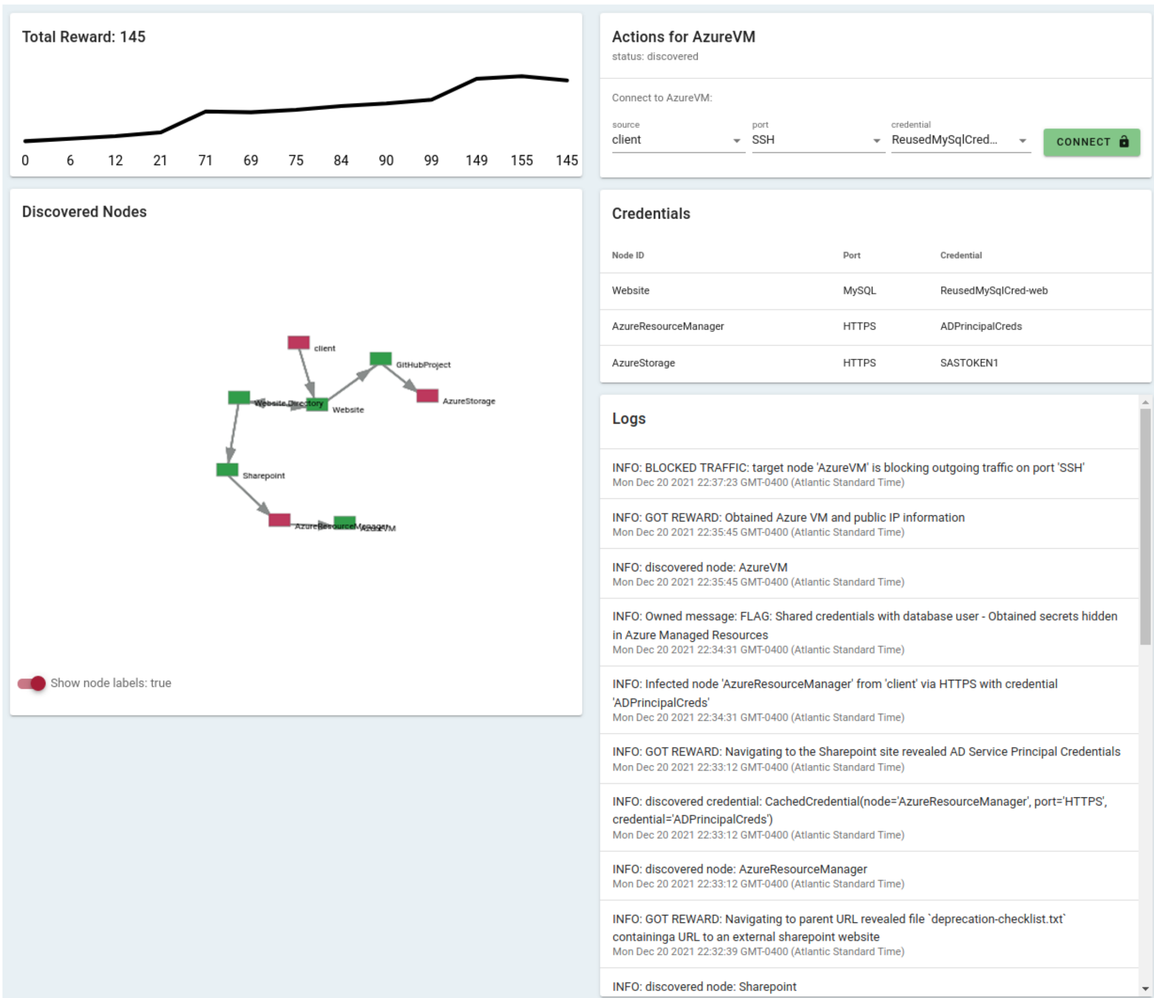


Figure 3-10: Attack progression on sample network topology showcasing a successful connect-and-infect of a "Website" node via a stolen credential. Owned nodes are shown in red.

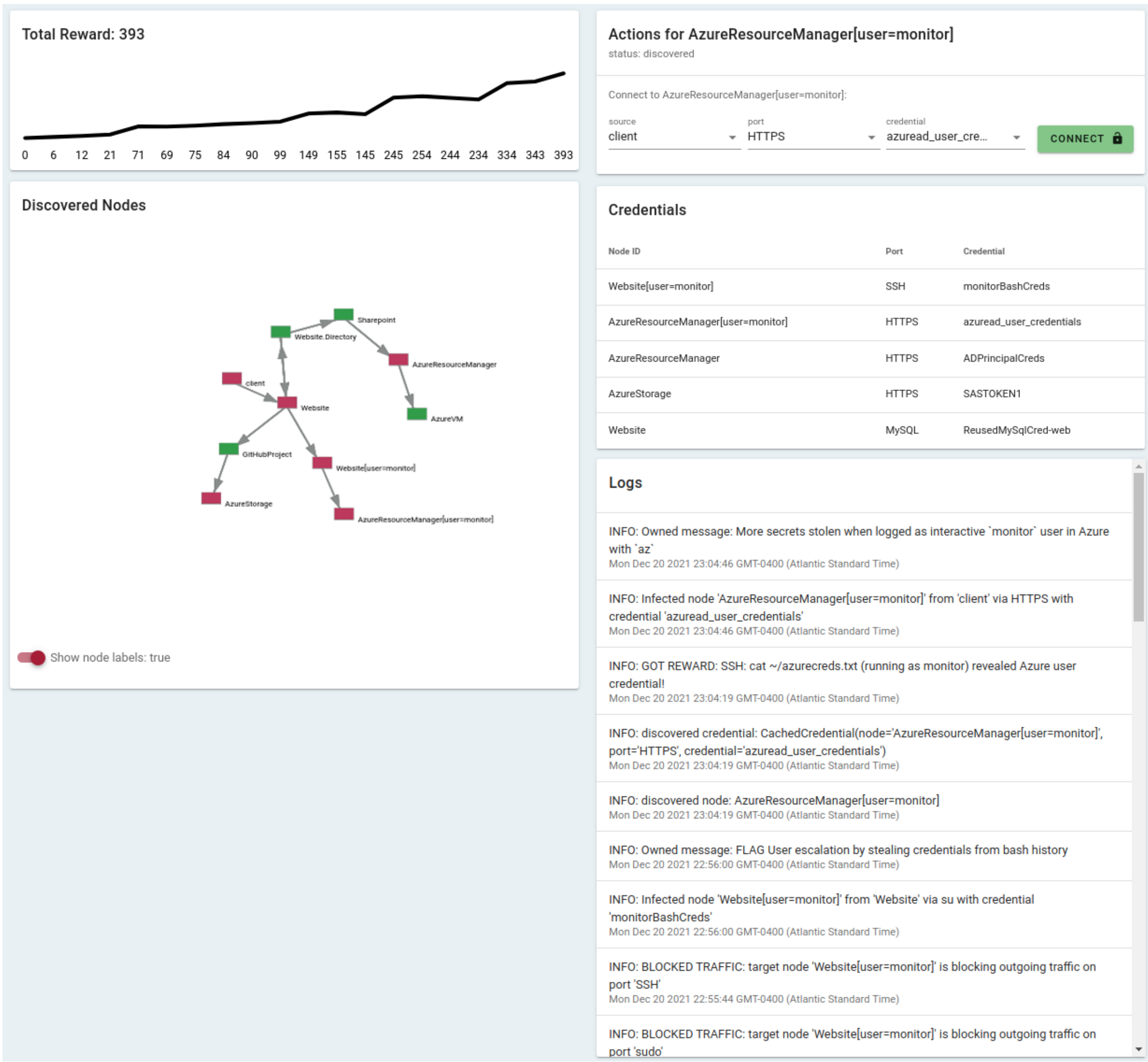


Figure 3-11: Entire network has been compromised and all flags have been acquired.

Table 3.2: Q-Learning Parameters

name	**description**
iteration count	Maximum number of iterations in each episode
episode count	Number of training episodes
gamma	Gamma discount factor
learning rate	Determines the weight of successful actions.
epsilon	Explore vs Exploit: 0.0 to exploit the learned policy only without exploration vs 1.0 to explore purely randomly
epsilon decay	Epsilon gets multiplied by this value after each episode
attacker reward	Creates goal to reach at least the specified cumulative total reward
low availability	Creates goal to bring the availability to lower than the specified Service Level Agreement (SLA) value
own at least	Creates goal to own at least the specified number of nodes
own at least percent	Creates goal to own at least the specified percentage of the network nodes

is shown. These figures include progression of total reward, network observability over time, as well as duration of episodes.

3.4 Backend Routing

The backend of the project involves a simple Flask server that relays user-submitted data into CyberBattleSim's internal model. All data is sanitized on the frontend and backend to keep the network model's preconditions consistent. Each action that the user can make on the frontend has a corresponding API route exposed on the backend server. The source code of CyberBattleSim was modified lightly to allow for the serialization and deserialization of the data being transmitted.

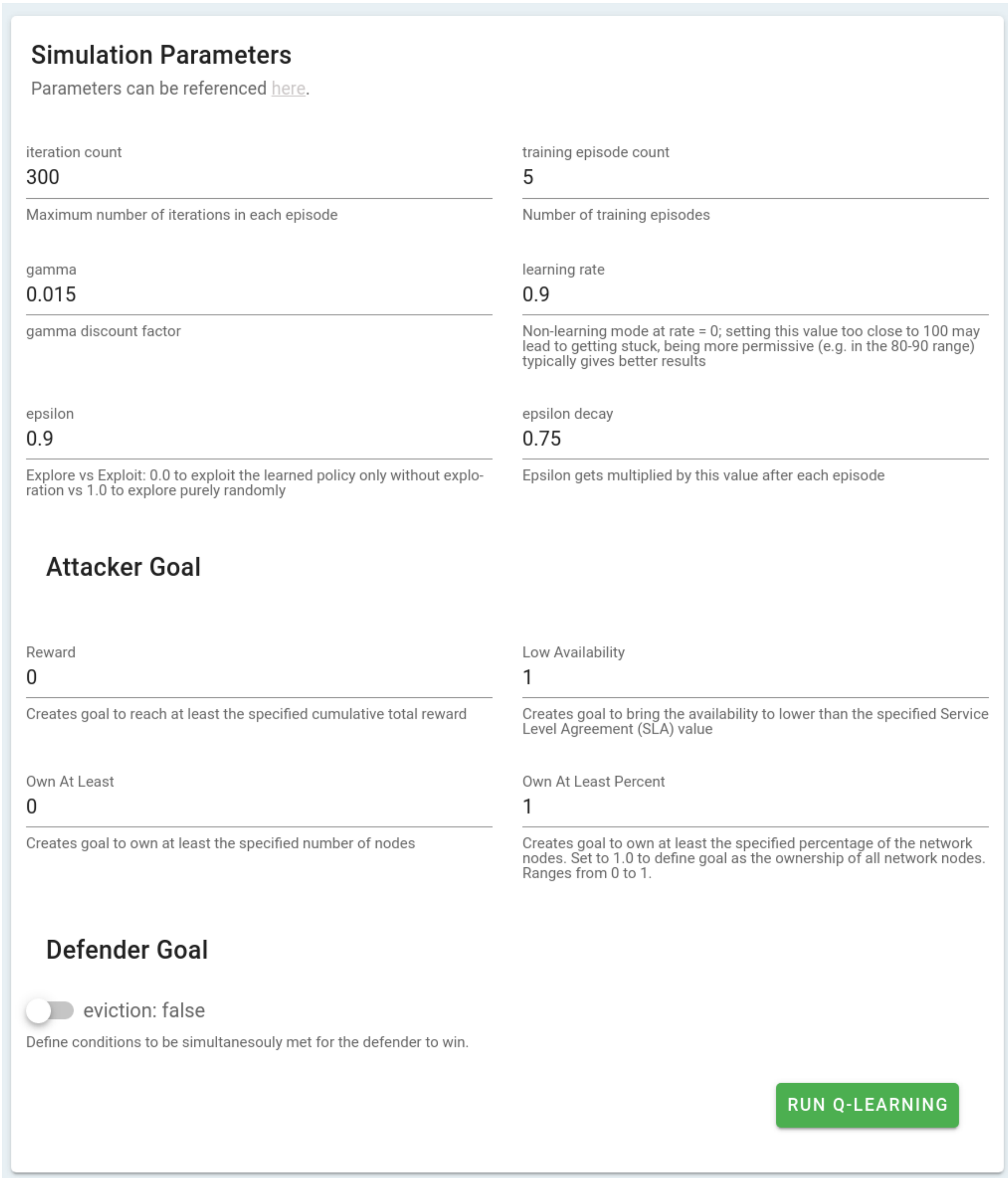


Figure 3-12: Simulation Parameters

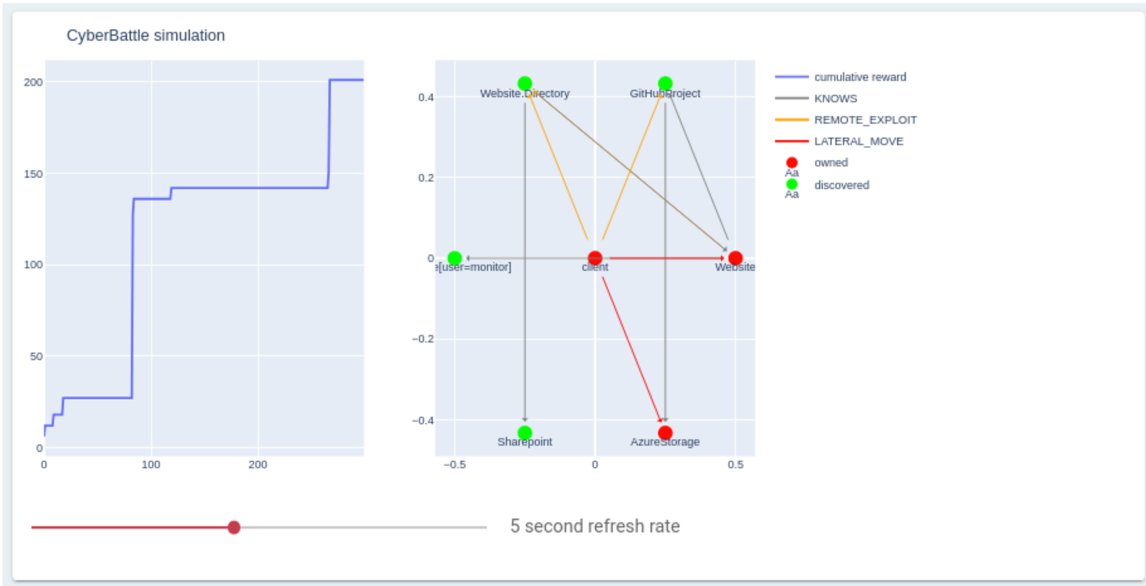


Figure 3-13: Simulation Running

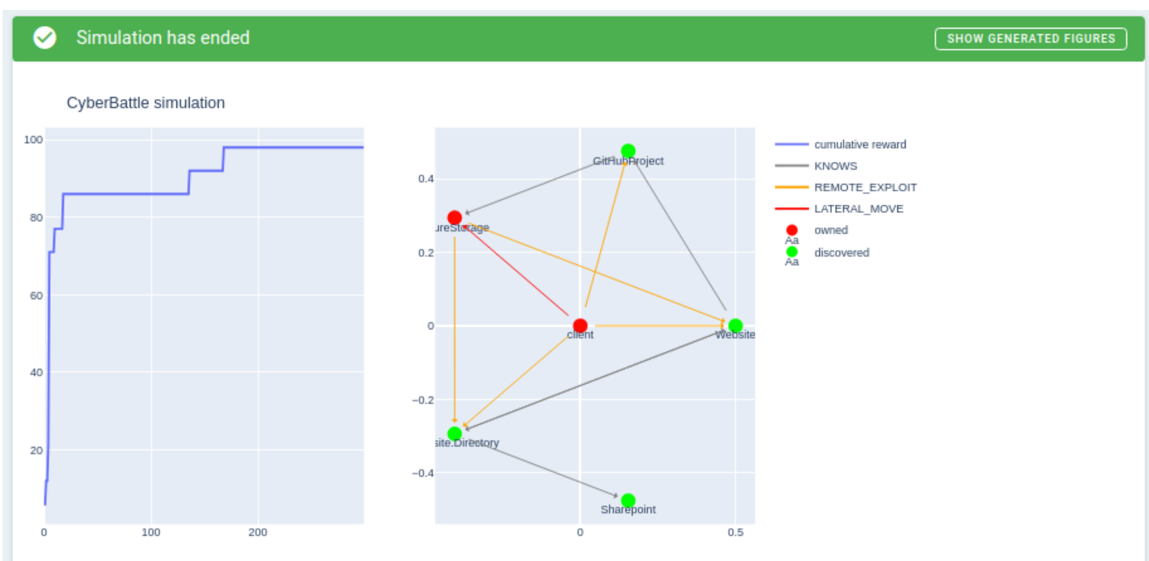


Figure 3-14: Simulation Ended

Chapter 4

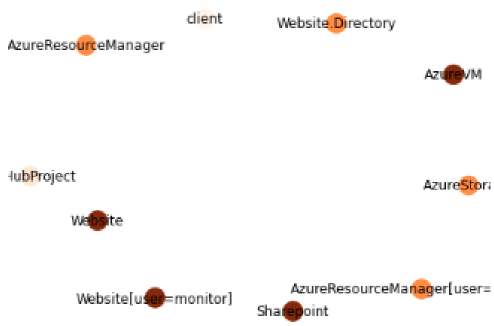
Results

The goal for this project was to create a web platform in which a user can model network topologies and interact with them either manually or via an AI agent. Crucially, the platform must be highly interoperable with the CyberBattleSim project. Our metric for success was to replicate CyberBattleSim's capture-the-flag (CTF) topology with the network modeling component and be able to carry out the same agent actions supported by CyberBattleSim. These actions enable a human or AI agent to manipulate the environment.

4.1 Human Interaction with CTF Network Topology

Testing the Human-interactive component involved going through the solution to the CTF provided by CyberBattleSim (shown in Figure 4-1) and applying each action.

The replicated CTF environment can be seen in Figure 3-1. Because every node property listed in Table 3.1 can be configured, virtually any network topology can be abstracted into CyberBattleSim's model. The step-by-step walk-through of the CTF solution can be seen in Appendix B. Thus we have shown that we can both model and manually interact with network topologies that are compatible with the CyberBattleSim project.



Solution to the CTF

This is the list of actions taken to capture 7 of the 8 flags from the CTF game.

Source	Action	Result
WEBSITE	page content has a link to github	Discover Github project
GITHUB	navigate github history	**FLAG** Some secure access token (SAS) leaked in a reverted git commit (`CredScan`)
AZURESTORAGE	access blob using SAS token	
WEBSITE	view source HTML	Find URL to hidden .txt file on the website, extract directory path from it
	navigate to parent URL and find 3 files	**FLAG** Discover browseable web directory
	- readme.txt file	Discover secret data (the flag)
	- getting-started.txt	Discover MYSQL credentials
	- deprecation-checklist.txt	Discover URL to external sharepoint website
SHAREPOINT	Navigate to sharepoint site	**FLAG** Finding AD Service Principal Credentials on Sharepoint
CLIENT-AZURE	`az resource` with creds from sharepoint	Obtain secrets hidden in azure managed resources
		Get AzureVM info, including public IP address
CLIENT	`ssh IP`	Failed attempt: internet incoming traffic blocked on the VM by NSG
CLIENT	SSH into WEBSITE with mysql creds	**FLAG** Shared credentials with database user
		FLAG Login using insecure SSH user/password
WEBSITE/SSH	`history`	**FLAG** Stealing credentials for the monitoring user
	`sudo -u monitor`	Failed! monitor not sudoable. message about being reported!
CLIENT	SSH into WEBSITE with 'monitor creds	Failed! password authentication disabled! looking for private key
CLIENT	SSH into WEBSITE as 'web'	
	`su -u monitor` using password	**FLAG** User escalation by stealing credentials from bash history
	`cat ~/azurecreds.txt`	Get user credentials to Azure
CLIENT	`az resource` with monitor's creds	Steal more secrets

Figure 4-1: Sample solution for Toy Capture-The-Flag Network Topology.

Table 4.1: Q-Learning CTF Simulation Parameters. Descriptions found in Table 3.1

name	**value**
iteration count	300
episode count	5
gamma	0.015
learning rate	0.9
epsilon	0.9
epsilon decay	0.75
attacker reward	0
low availability	1
own at least	0
own at least percent	100%

4.2 Q-Learning AI Interaction with CTF Network Topology

We applied Q-Learning to the CTF Network Topology to demonstrate the platform's capability of running CyberBattleSim reinforcement learning techniques on network models. Figures 4-2 through 4-9 display the results of running Q-Learning on the CTF Network with the parameters in Table 4.1. The plots to the left of each figure show accumulated reward over time. Meanwhile, network graphs to the right of each figure show the sub-network available to the AI agent, with discovered nodes shown in green and owned nodes shown in red. The results demonstrate that the web platform can be used to evaluate different Q-Learning strategies without the need of using the CyberBattleSim platform directly.

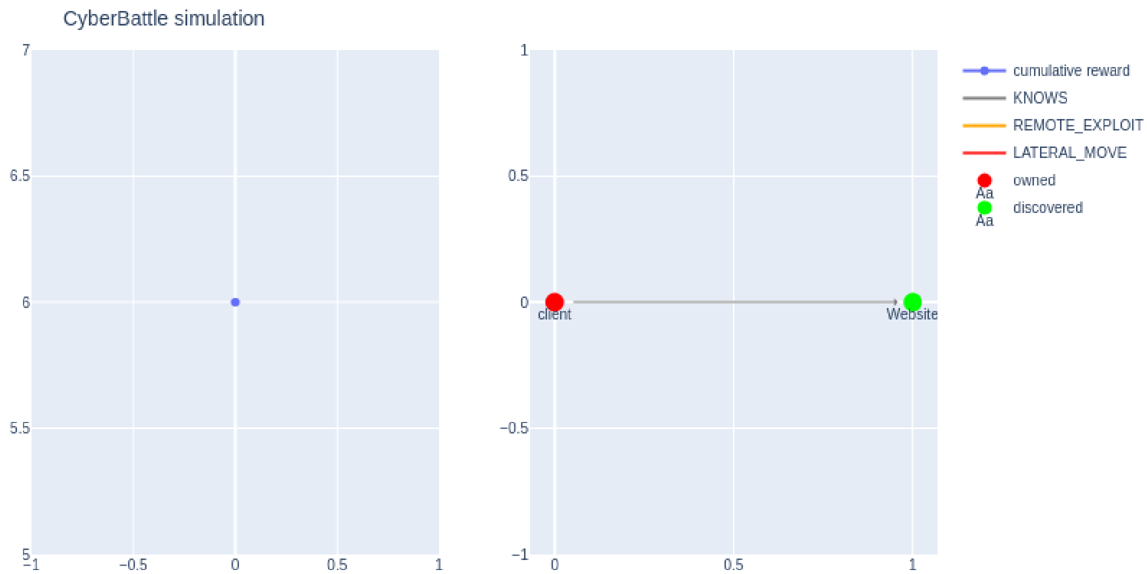


Figure 4-2: Step 1 of attack progression under Q-Learning AI agent

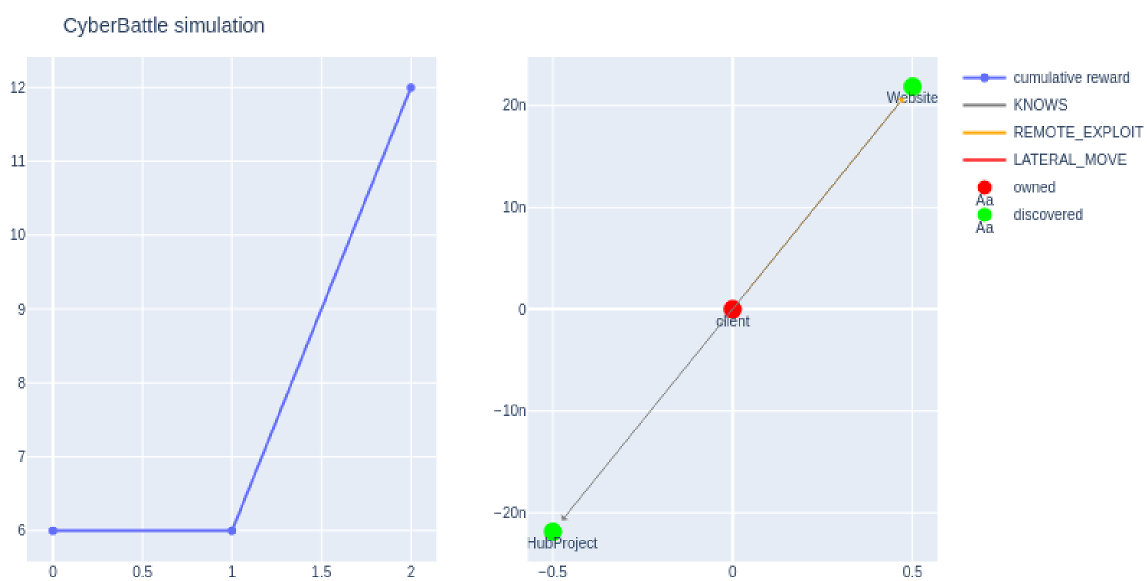


Figure 4-3: Step 2 of attack progression under Q-Learning AI agent

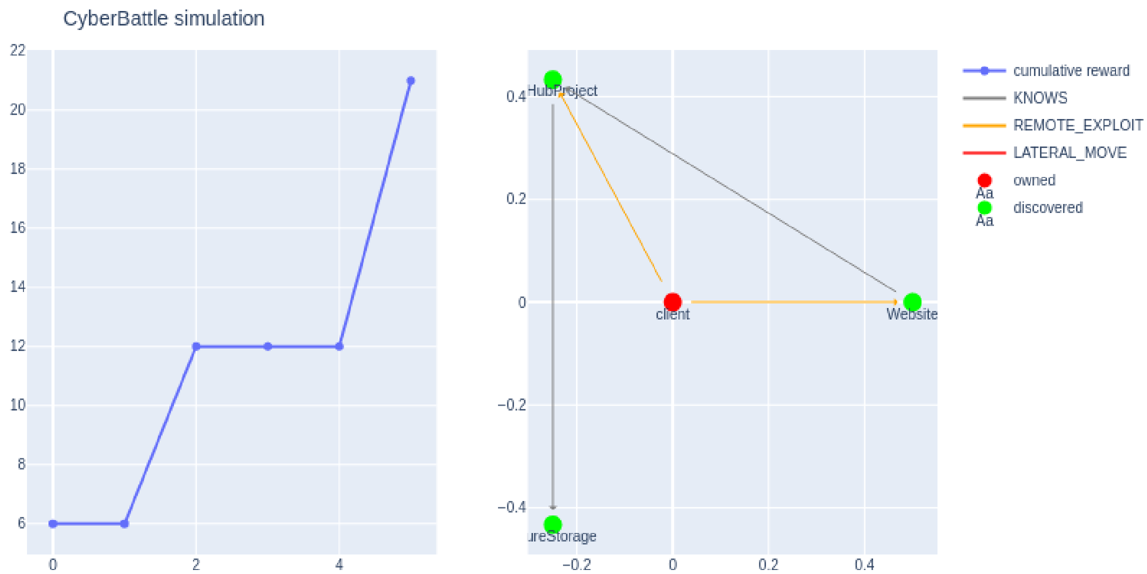


Figure 4-4: Step 3 of attack progression under Q-Learning AI agent

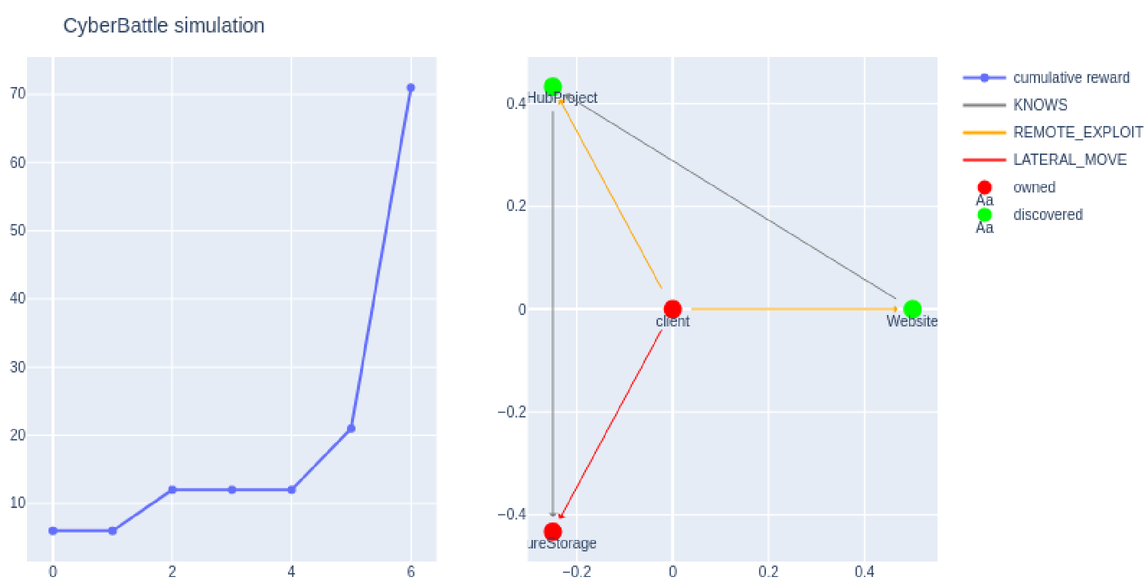


Figure 4-5: Step 4 of attack progression under Q-Learning AI agent

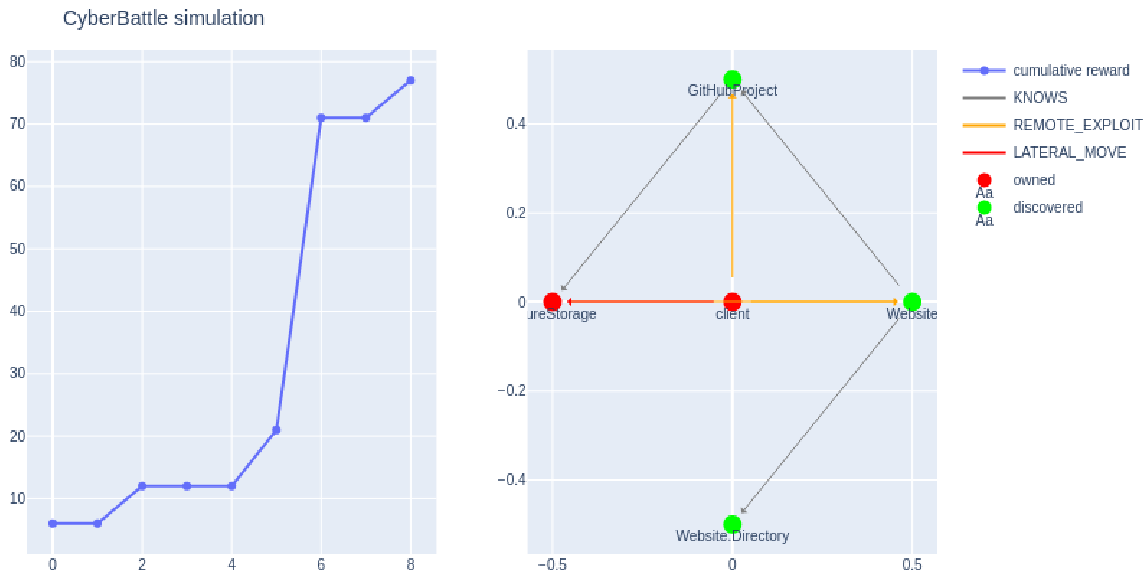


Figure 4-6: Step 5 of attack progression under Q-Learning AI agent

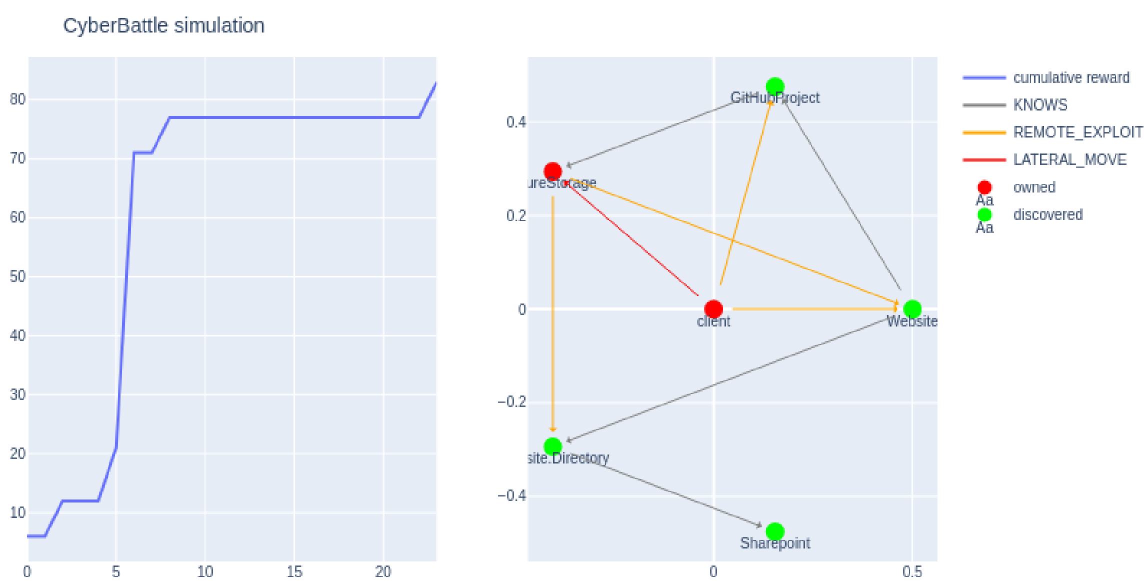


Figure 4-7: Step 6 of attack progression under Q-Learning AI agent

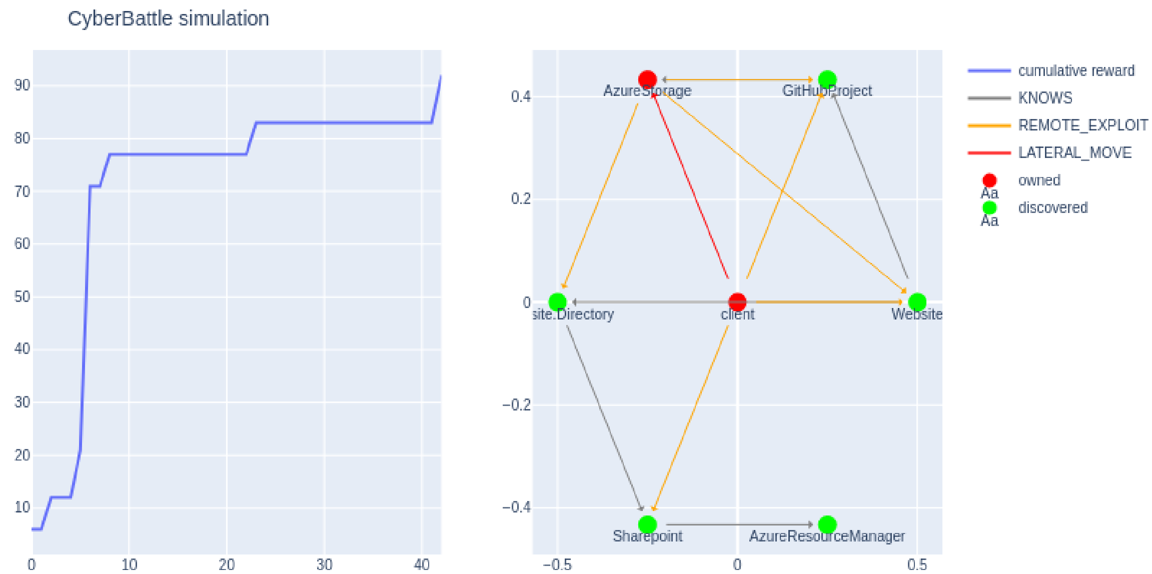


Figure 4-8: Step 7 of attack progression under Q-Learning AI agent

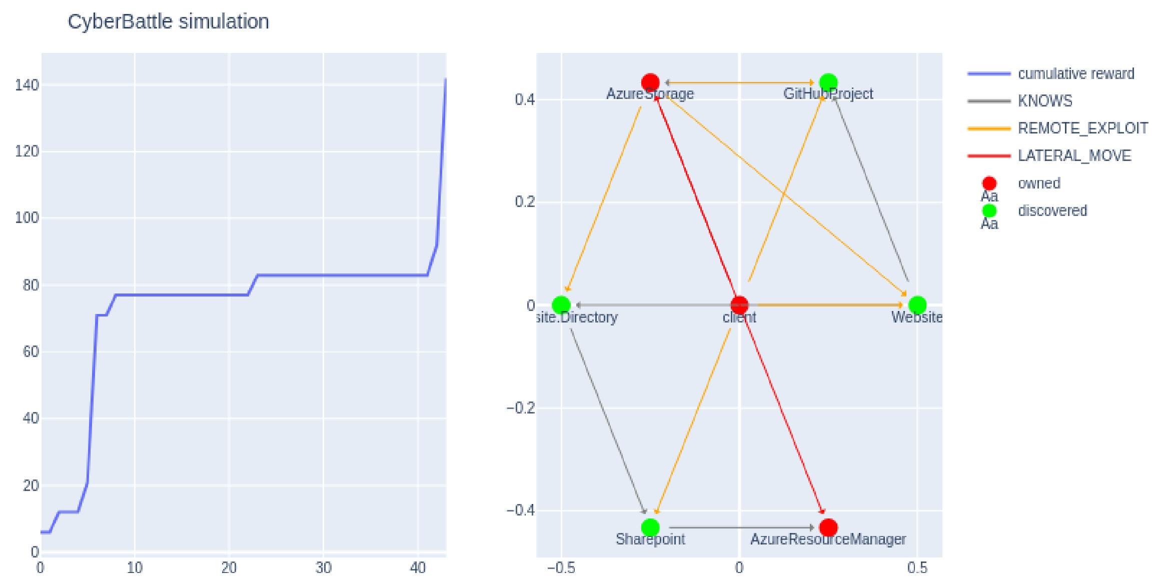


Figure 4-9: Step 8 of attack progression under Q-Learning AI agent

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Conclusions

This project provides a way to build and simulate enterprise networks, making it possible to frame cybersecurity challenges in the context of reinforcement learning via a web platform. This tool shows that high-level abstractions of cyber security concepts can help us understand how real cyber-agents would behave in actual enterprise networks.

5.1 Future Work

Future work on the CyberBattleSim web platform includes adding support for other AI algorithms. Permitting other types of AI agents would allow the user to compare different attacker strategies. Microsoft's CyberBattleSim project has already provided a suite of agents as starting points, thus this task would be a matter of extending the current API to support these agents. In addition, adding support for a defender agent could prove to be worthwhile, as CyberBattleSim readily supports defensive players. Finally, at the time of writing, the CyberBattleSim continues to be developed. Thus future work could include adding frontend support to new features planned for the project, such as simulating network traffic and file systems.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Supplementary Information

A.1 Reinforcement Learning

Reinforcement learning is a technique within machine learning in which autonomous agents learn how to conduct decision-making by interacting with their environment and accumulating knowledge. [15] Agents may perform actions to interact with their environment in order to optimize a reward function. Reinforcement learning algorithms can learn effective strategies through repeated experience by gradually learning what actions to take within each state of the environment. The more agents interact with the environment, the better they optimize obtaining reward.

Reinforcement Learning can be applied in the context of cyber security. [16] In this case, the automated agent is the attacker or a defender, which evolve in the environment that is provided by a simulated computer network. The actions available to the agents are the network and computer commands. An automated attacker's goal would be to compromise the network while an automated defender's goal would be to circumvent the attacker's actions by executing a set of protective measures.

Reinforcement Learning techniques can be readily applied using OpenAI Gym, a software tool that provides interactive environments for researchers to develop, train, and evaluate machine learning algorithms. [17]

A.2 CyberBattleSim

Microsoft developed CyberBattleSim in an attempt to leverage AI and machine learning to solve cybersecurity challenges, in particular, autonomous systems. In a simulated enterprise network, the CyberBattleSim toolkit serves to investigate how reinforcement learning techniques can be applied to improve security within an network environment. CyberBattleSim uses the Python-based OpenAI Gym interface, which allows for the training of automated agents using reinforcement learning algorithms. [5]

Thus, using CyberBattleSim, we are able to construct a highly abstract simulation of computer systems, making it possible to frame cybersecurity challenges in the context of reinforcement learning. [11] CyberBattleSim provides a network model in which cyber-agents can interact and evolve in a sand-boxed, simulated environments. This type of high-level abstraction prevents direct application to real-world systems, which safeguards against potential nefarious use of automated agents trained with it. With that said, it can still prove to be useful for gaining insights with respect to how real cyber-agents would behave in an actual enterprise network.

A.2.1 How CyberBattleSim works

CyberBattleSim focuses on threat modeling the post-breach lateral movement stage of a cyber-attack. The environment consists of a network of computer nodes. It is parameterized by a fixed network topology and a set of predefined vulnerabilities that an agent can exploit to laterally move through the network. The simulated attacker's goal is to take ownership of some portion of the network by exploiting these planted vulnerabilities. While the simulated attacker moves through the network, a defender agent watches the network activity to detect the presence of the attacker and contain the attack.

To illustrate, the graph in figure A-1 depicts a toy example of a network with machines running various operating systems and software. Each machine has a set of properties, a value, and pre-assigned vulnerabilities. Black edges represent traffic running between nodes and are labelled by the communication protocol.

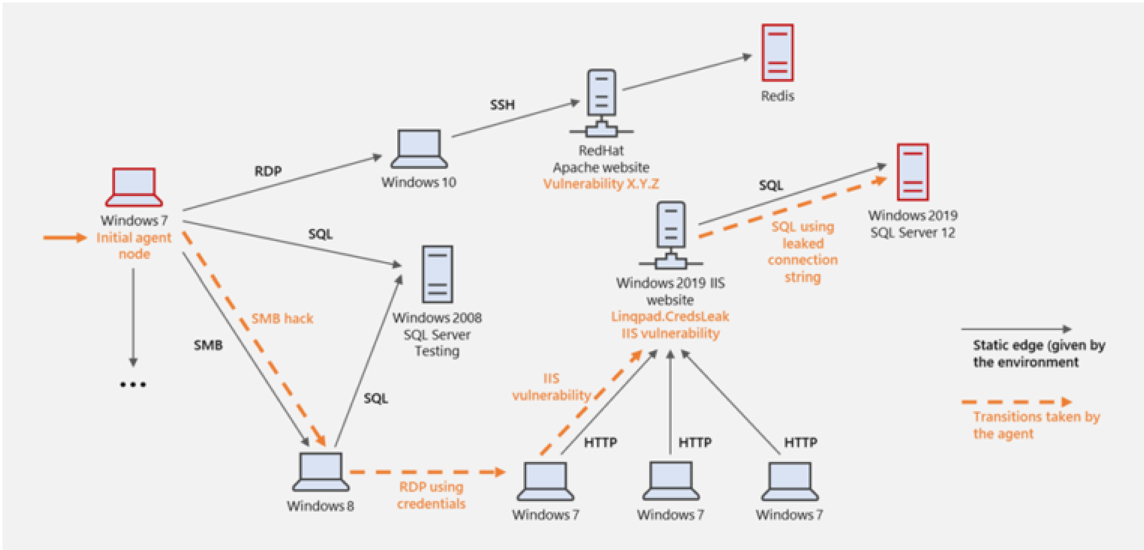


Figure A-1: Visual representation of lateral movement in a computer network simulation

Suppose the agent represents the attacker. The post-breach assumption means that one node is initially infected with the attacker's code (we say that the attacker owns the node). The simulated attacker's goal is to maximize the cumulative reward by discovering and taking ownership of nodes in the network. The environment is partially observable: the agent does not get to see all the nodes and edges of the network graph in advance. Instead, the attacker takes actions to gradually explore the network from the nodes it currently owns. There are three kinds of actions, offering a mix of exploitation and exploration capabilities to the agent: performing a local attack, performing a remote attack, and connecting to other nodes. Actions are parameterized by the source node where the underlying operation should take place, and they are only permitted on nodes owned by the agent. The reward is a float that represents the intrinsic value of a node (e.g., a SQL server has greater value than a test machine).

In the depicted example, the simulated attacker breaches the network from a simulated Windows 7 node (on the left side, pointed to by an orange arrow). It proceeds with lateral movement to a Windows 8 node by exploiting a vulnerability in the SMB file-sharing protocol, then uses some cached credential to sign into another Windows 7 machine. It then exploits an IIS remote vulnerability to own the IIS server, and finally uses leaked connection strings to get to the SQL DB.

This environment simulates a heterogeneous computer network supporting multiple platforms and helps to show how using the latest operating systems and keeping these systems up to date enable organizations to take advantage of the latest hardening and protection technologies in platforms like Windows 10. The simulation Gym environment is parameterized by the definition of the network layout, the list of supported vulnerabilities, and the nodes where they are planted. The simulation does not support machine code execution, and thus no security exploit actually takes place in it. We instead model vulnerabilities abstractly with a precondition defining the following: the nodes where the vulnerability is active, a probability of successful exploitation, and a high-level definition of the outcome and side-effects. Nodes have preassigned named properties over which the precondition is expressed as a Boolean formula.

Vulnerability outcomes

There are predefined outcomes that include the following: leaked credentials, leaked references to other computer nodes, leaked node properties, taking ownership of a node, and privilege escalation on the node. Examples of remote vulnerabilities include: a SharePoint site exposing ssh credentials, an ssh vulnerability that grants access to the machine, a GitHub project leaking credentials in commit history, and a SharePoint site with file containing SAS token to storage account. Meanwhile, examples of local vulnerabilities include: extracting authentication token or credentials from a system cache, escalating to SYSTEM privileges, escalating to administrator privileges. Vulnerabilities can either be defined in-place at the node level or can be defined globally and activated by the precondition Boolean expression.

Measuring progress

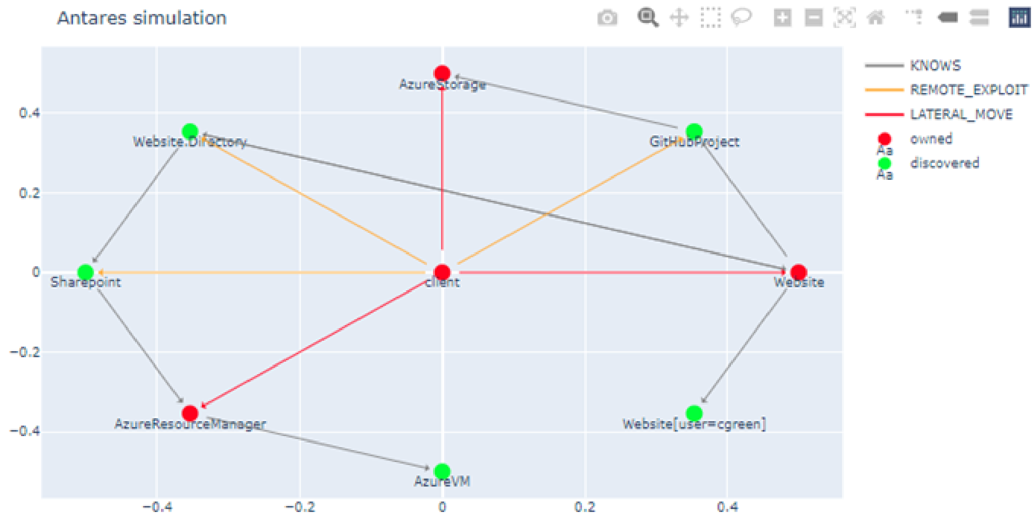
CyberBattleSim provides a basic stochastic defender that detects and mitigates ongoing attacks based on predefined probabilities of success. CyberBattleSim implements mitigation by reimaging the infected nodes, a process abstractly modeled as an operation spanning multiple simulation steps. To compare the performance of the agents, we look at two metrics: the number of simulation steps taken to attain their goal and the cumulative rewards over simulation steps across training epochs.

With the Gym interface, CyberBattleSim can easily instantiate automated agents and observe how they evolve in such environments. Figure A-2 shows the outcome of running a random agent on this simulation—that is, an agent that randomly selects which action to perform at each step of the simulation.

In [17]:

```
# 11
outcome = c2.run_attack('Website', 'CredScanBashHistory')
dbg.plot_discovered_network()
```

```
discovered node: Website[user=cgreen]
discovered credential: CachedCredential(node='Website[user=cgreen]', port='SSH', credential='cgreenBashCr
GOT REWARD: FLAG: SSH history revealed credentials for the monitoring user (cgreen)
```



In [18]:

```
print_all_attacks()
```

```
id                    status      local_attacks            remote_attacks
--------------------  ----------  -----------------------  ------------------------------------------------
client                owned       ['SearchEdgeHistory']    []
Website               owned       ['CredScanBashHistory']  ['ScanPageSource', 'ScanPageContent']
AzureStorage          owned       []                       ['AccessDataWithSASToken']
AzureResourceManager  owned       []                       ['ListAzureResources']
GitHubProject         discovered                           ['CredScanGitHistory']
Website.Directory     discovered                           ['NavigateWebDirectory', 'NavigateWebDirectory
Sharepoint            discovered                           ['ScanSharepointParentDirectory']
AzureVM               discovered                           []
Website[user=cgreen]  discovered                           []
```

Figure A-2: A random agent interacting with the simulation

Appendix B

Additional Figures

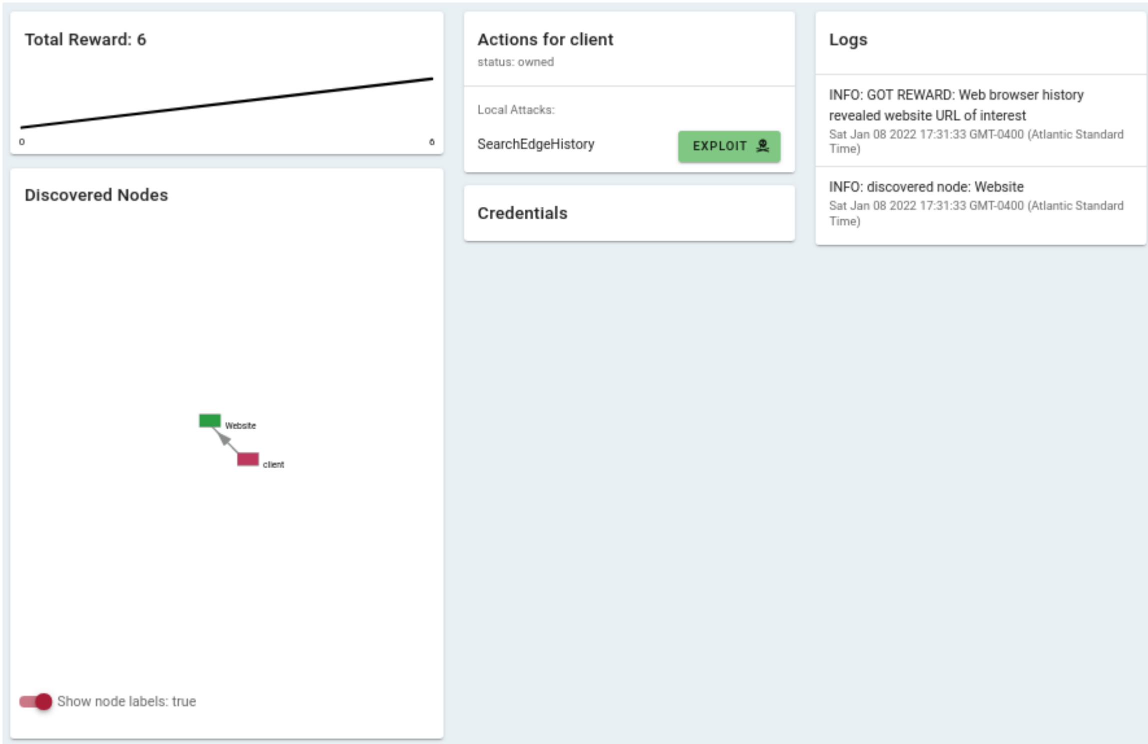


Figure B-1: Initial Environment

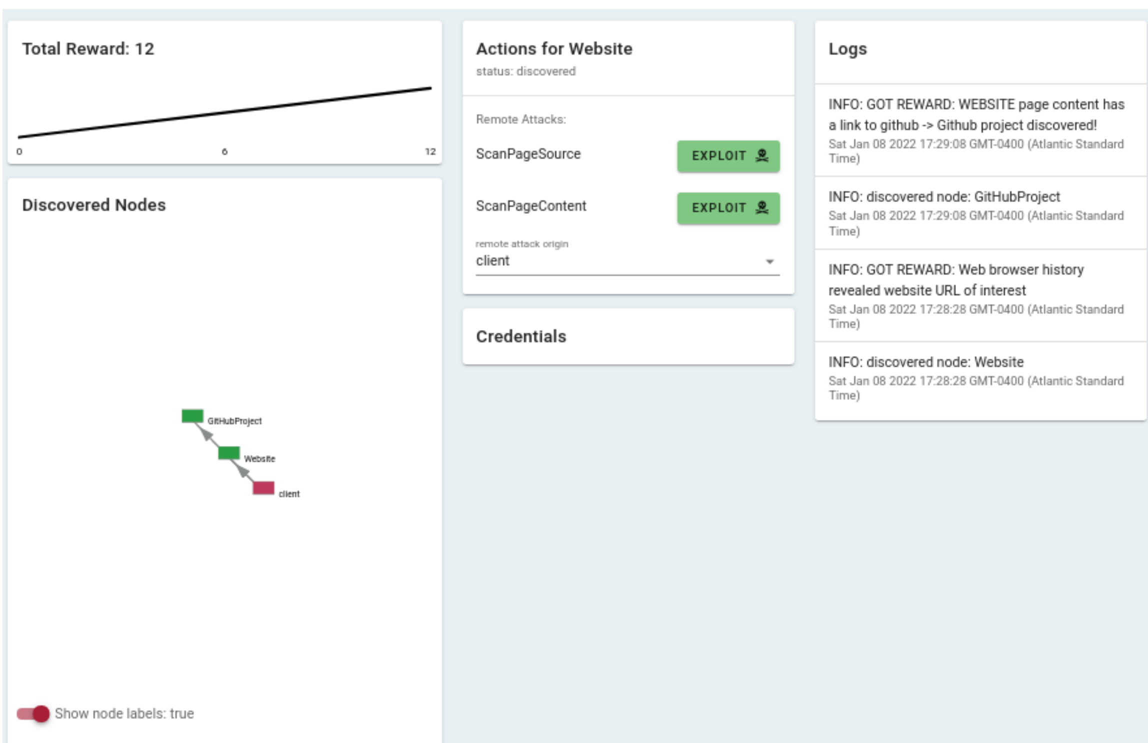


Figure B-2: Page content has a link to GitHub

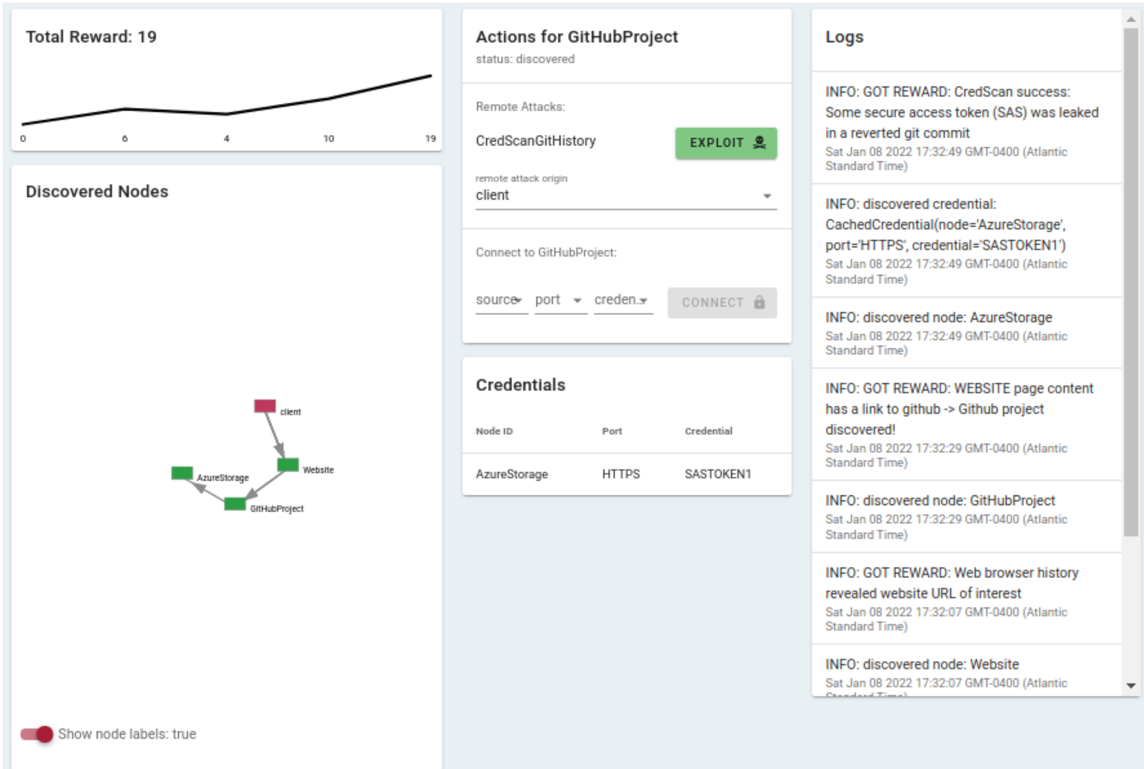


Figure B-3: Navigate GitHub history

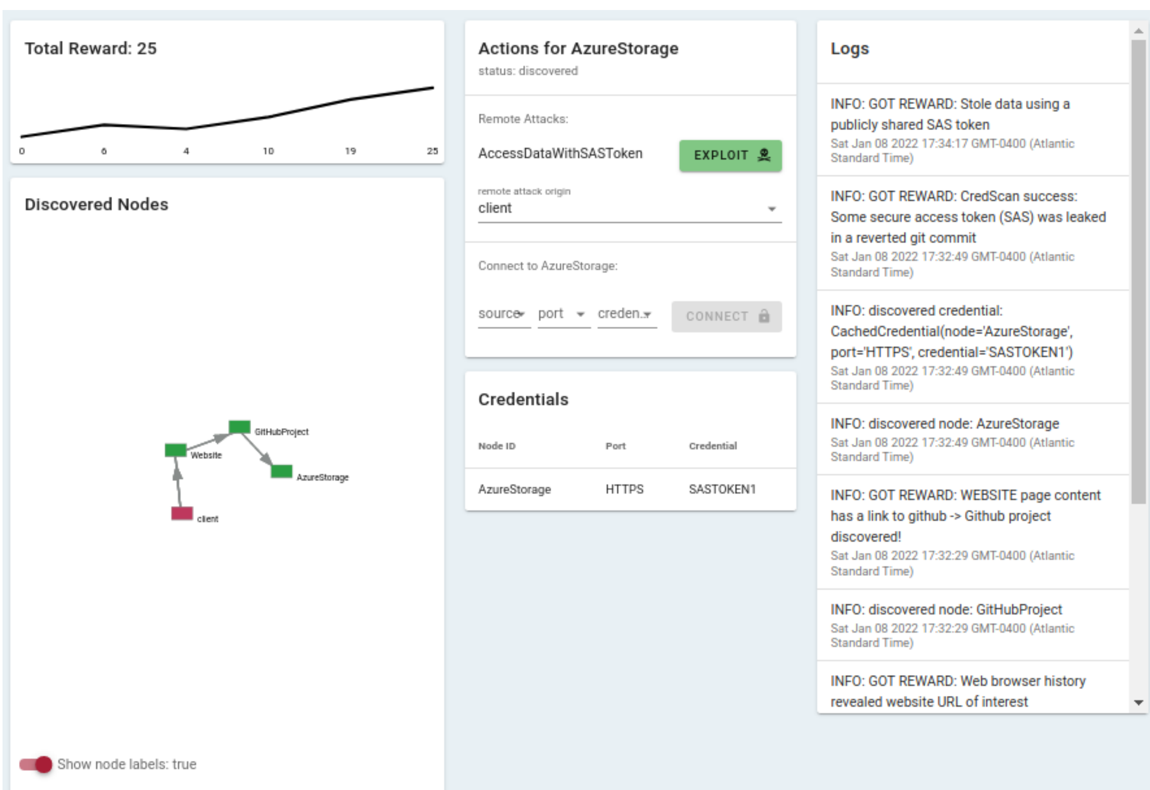


Figure B-4: Solutions Actions

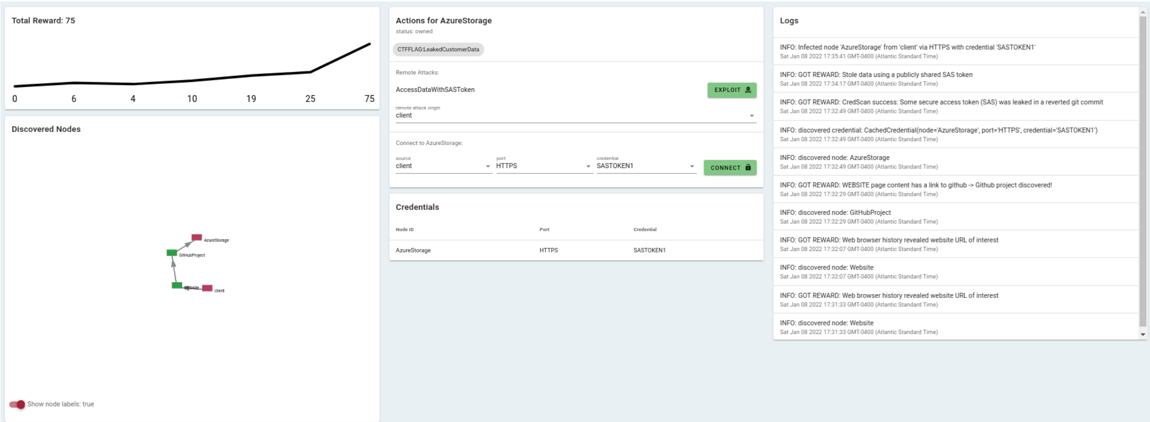


Figure B-5: Access blob using SAS token

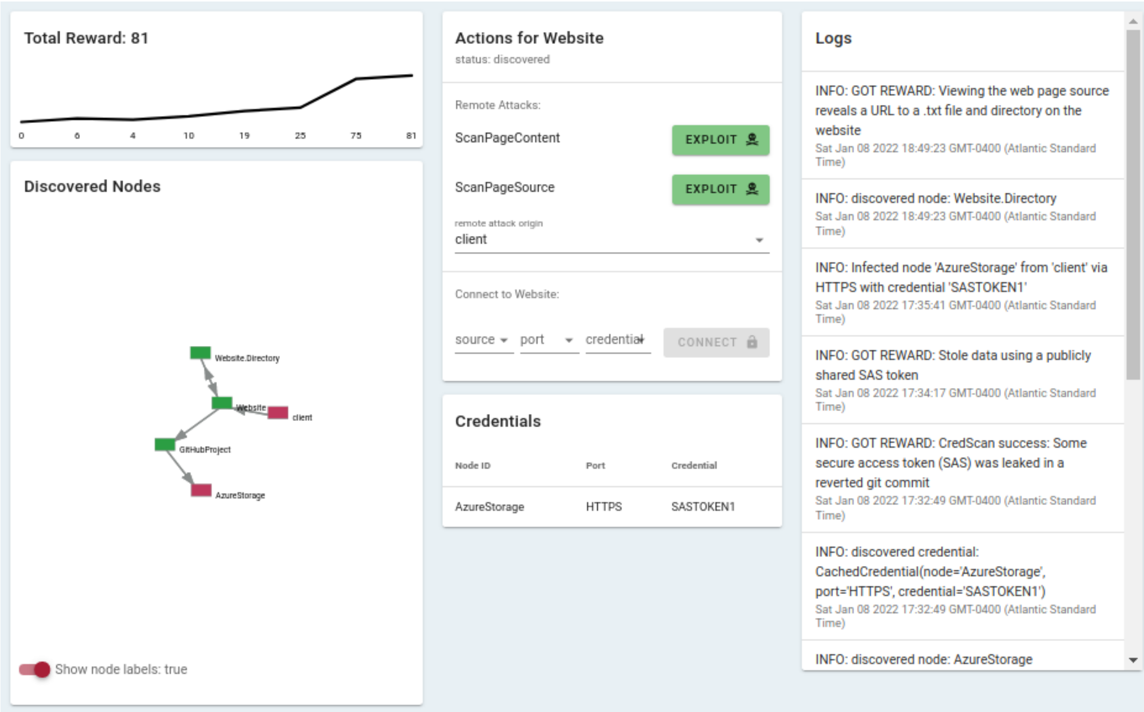


Figure B-6: Navigate to parent URL and find 3 files

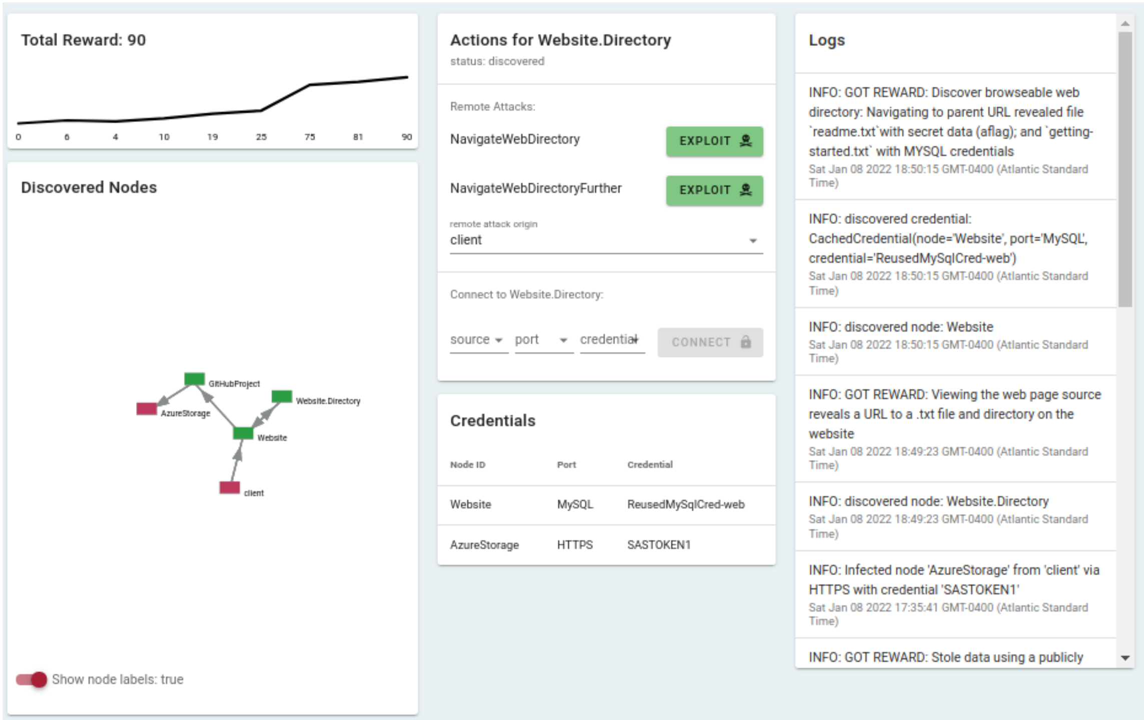


Figure B-7: Navigate to parent URL and find 3 files (cont.)

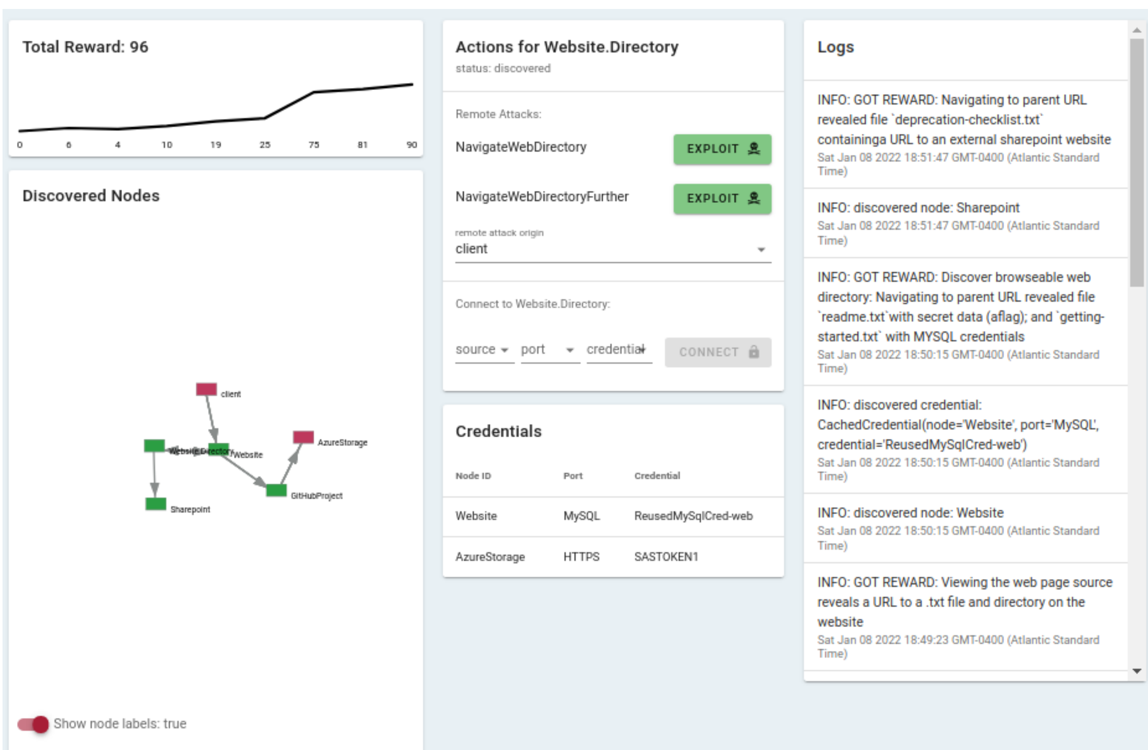


Figure B-8: Navigate to parent URL and find 3 files (cont.)

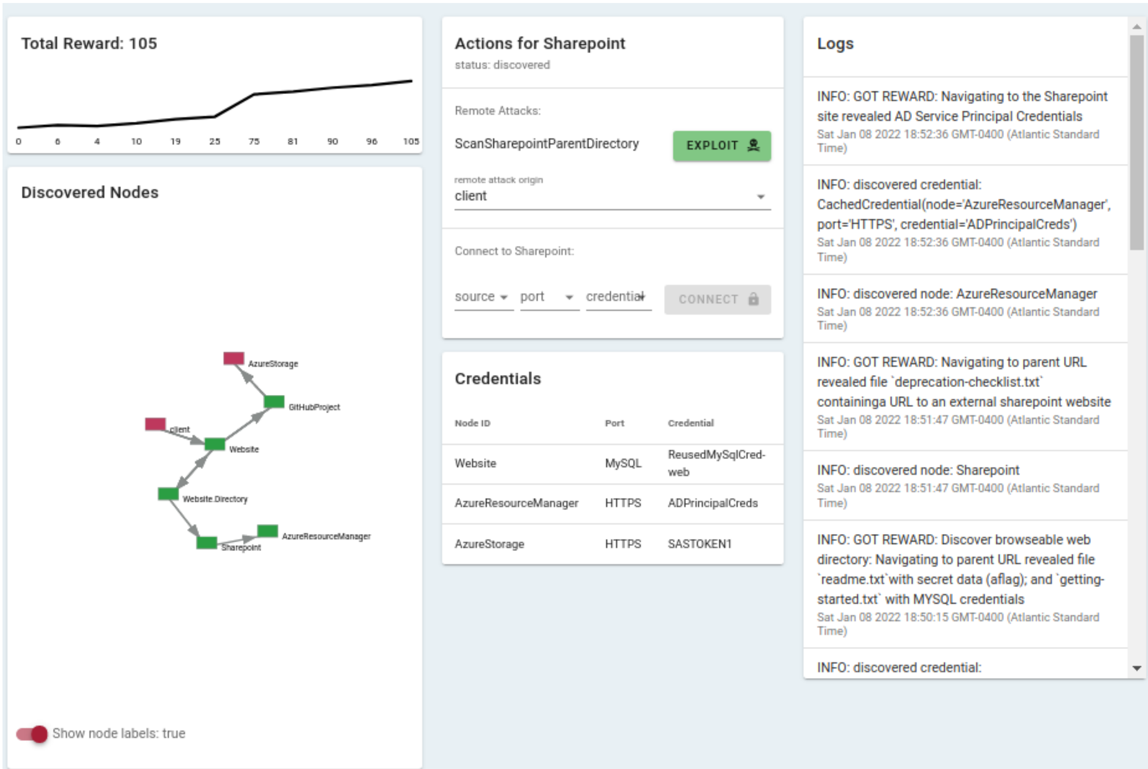


Figure B-9: Navigate to Sharepoint site

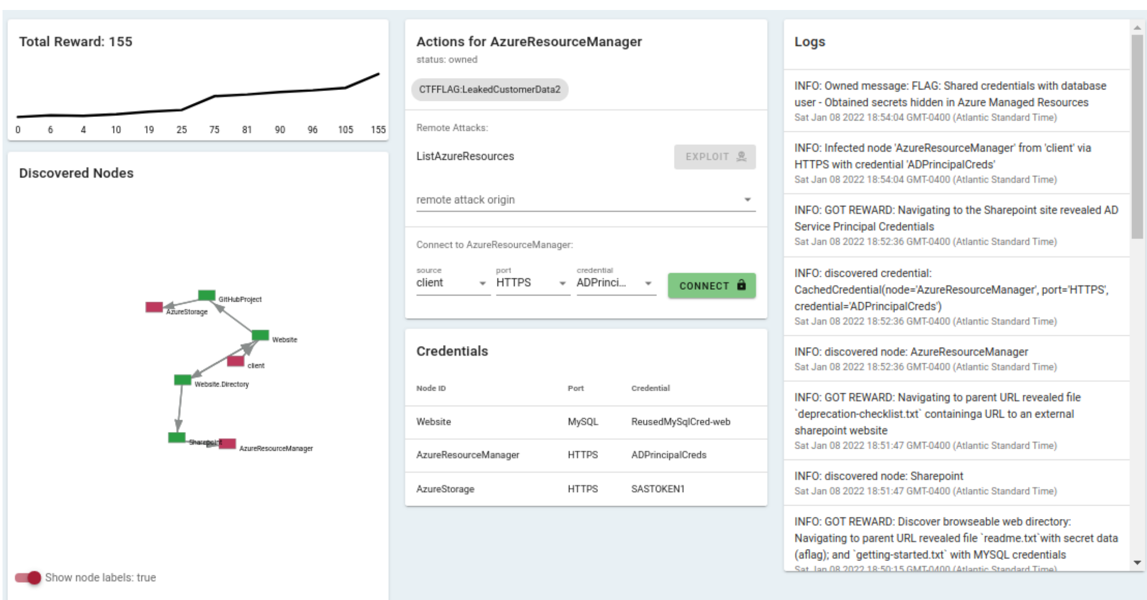


Figure B-10: Azure resource with credentials from Sharepoint

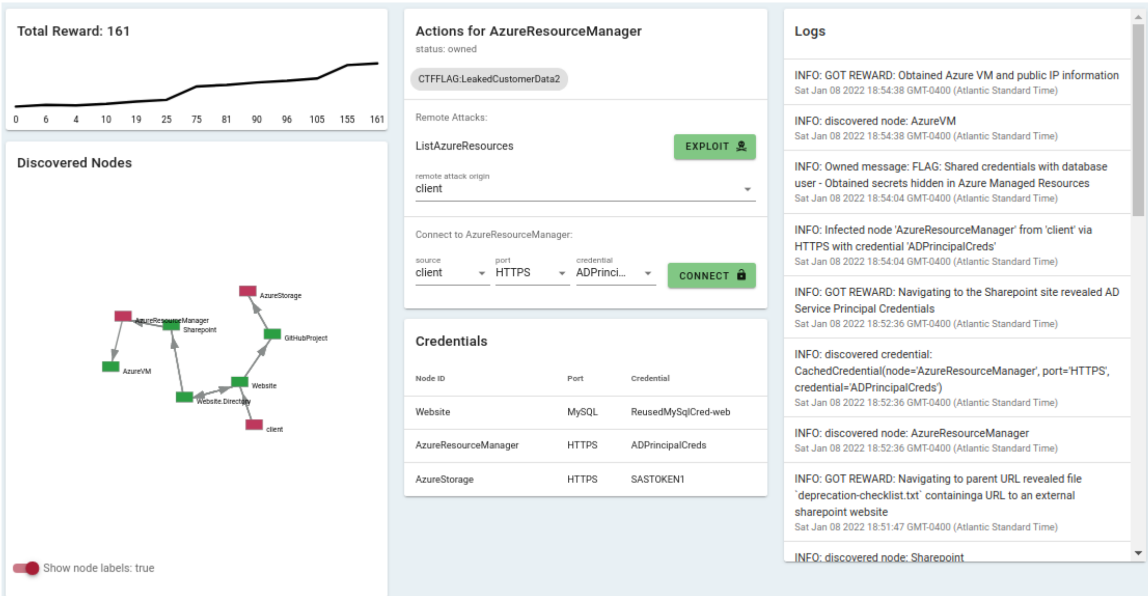


Figure B-11: Obtain Azure VM and public IP information

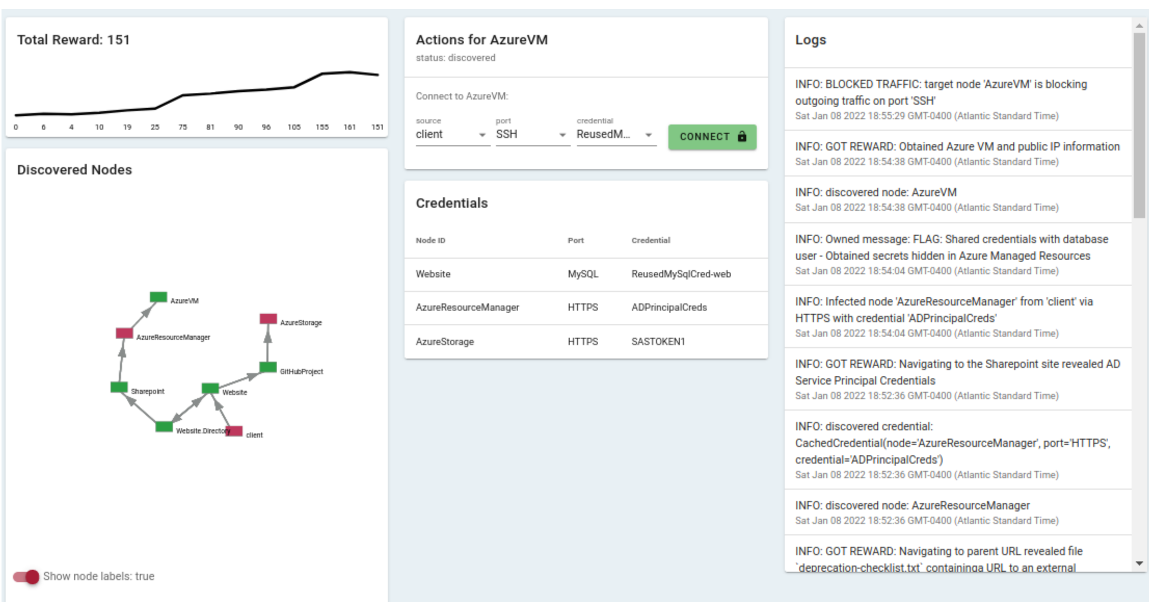


Figure B-12: SSH into IP

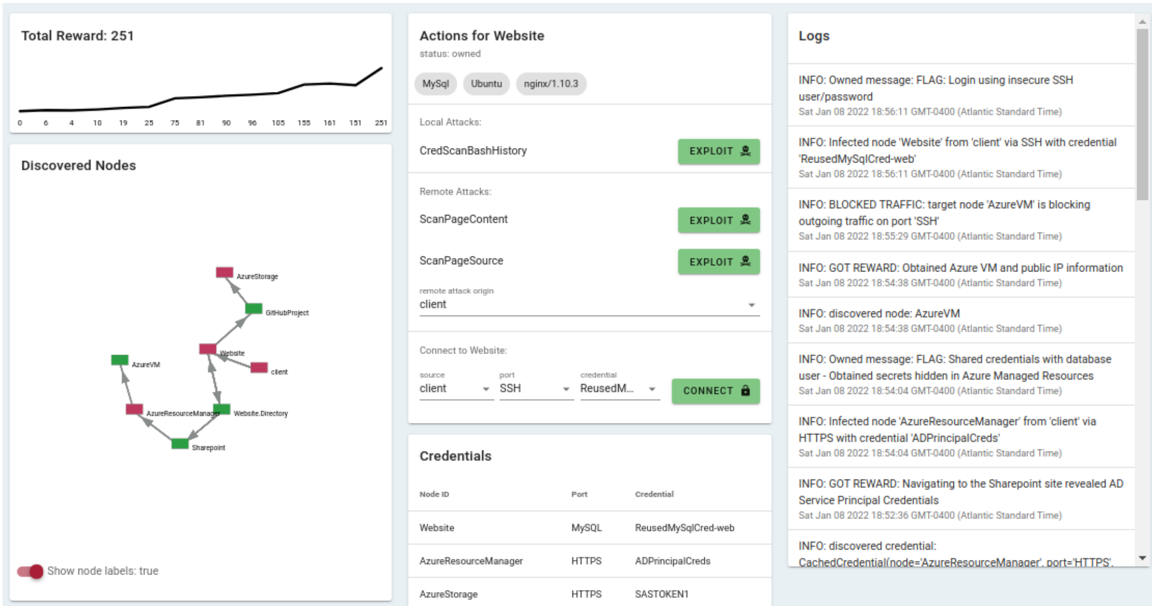


Figure B-13: SSH into Website with MySQL credentials

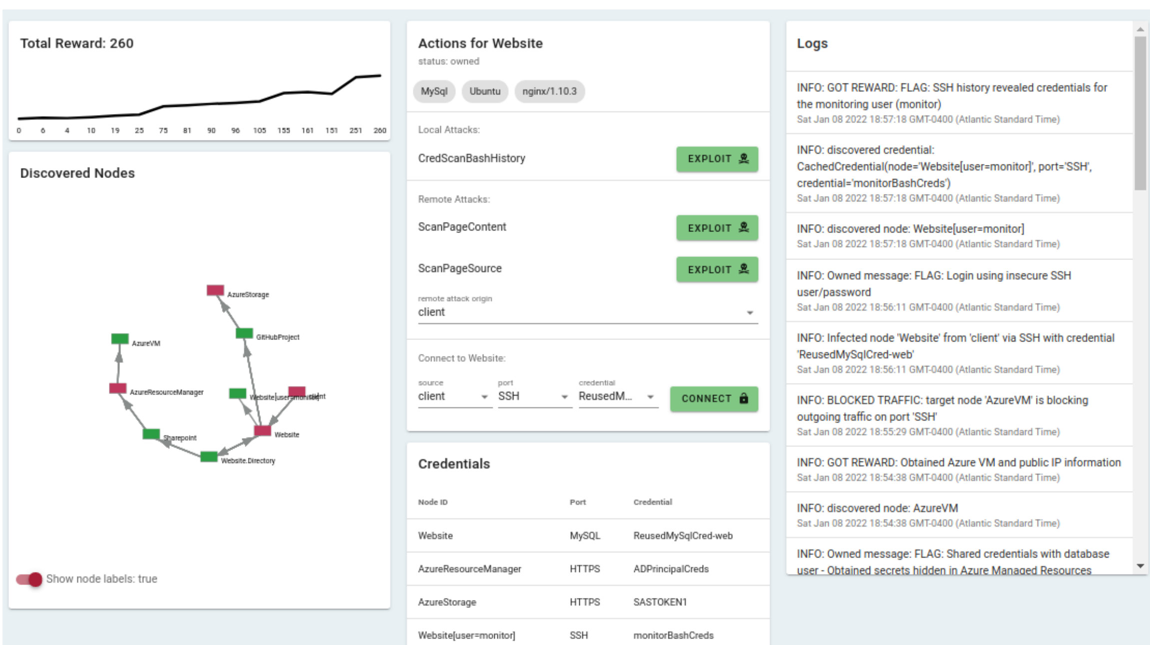


Figure B-14: Search SSH History

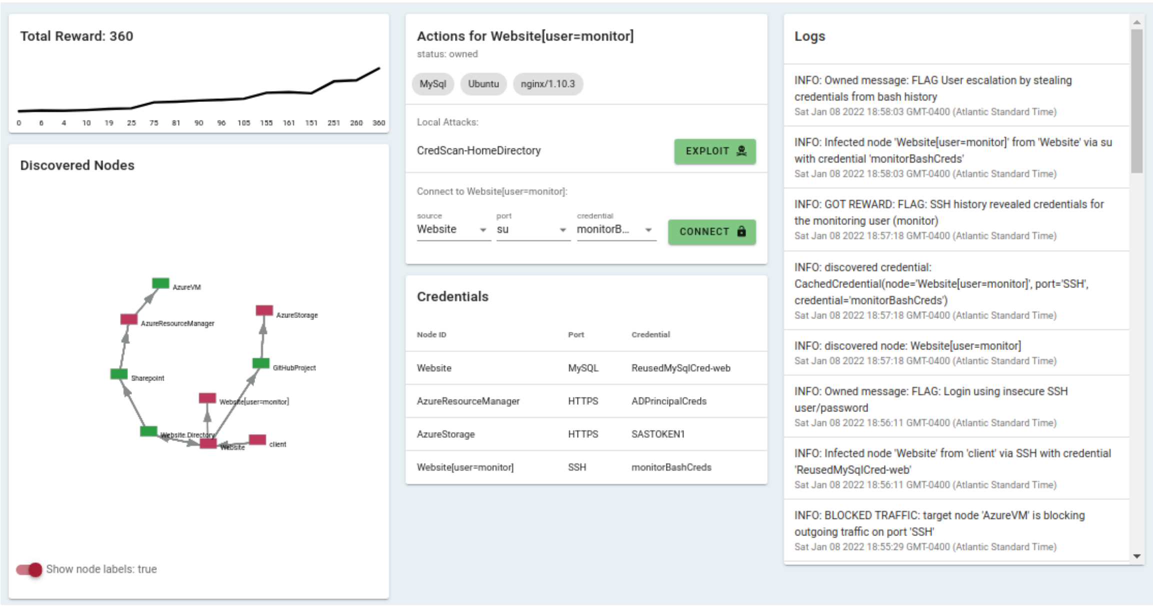


Figure B-15: execute command: su -u website monitor using stolen password

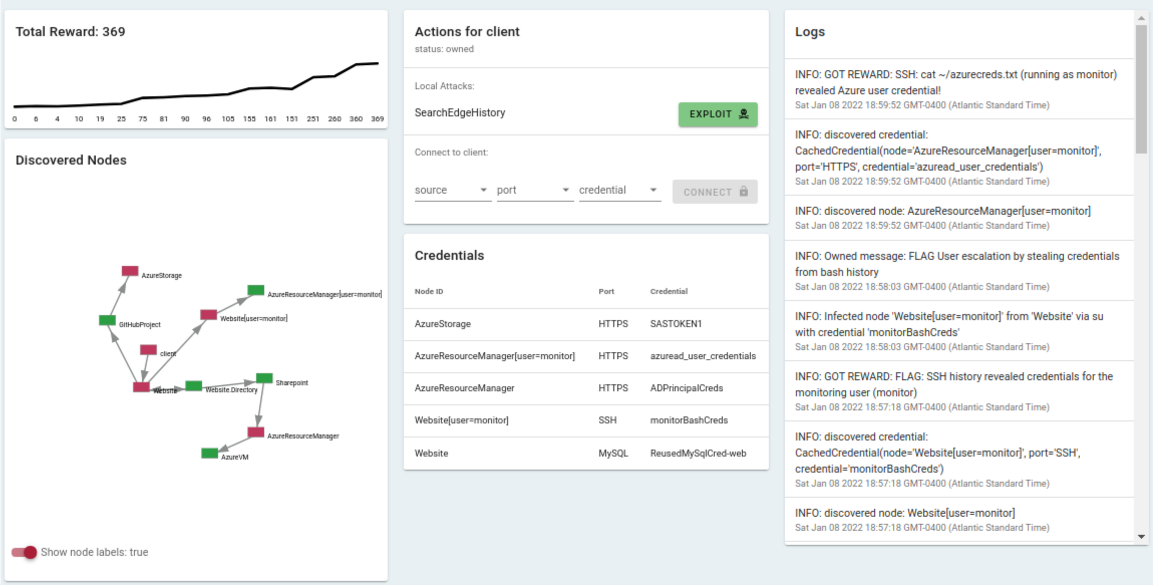


Figure B-16: execute command: cat ~/azurecreds.txt

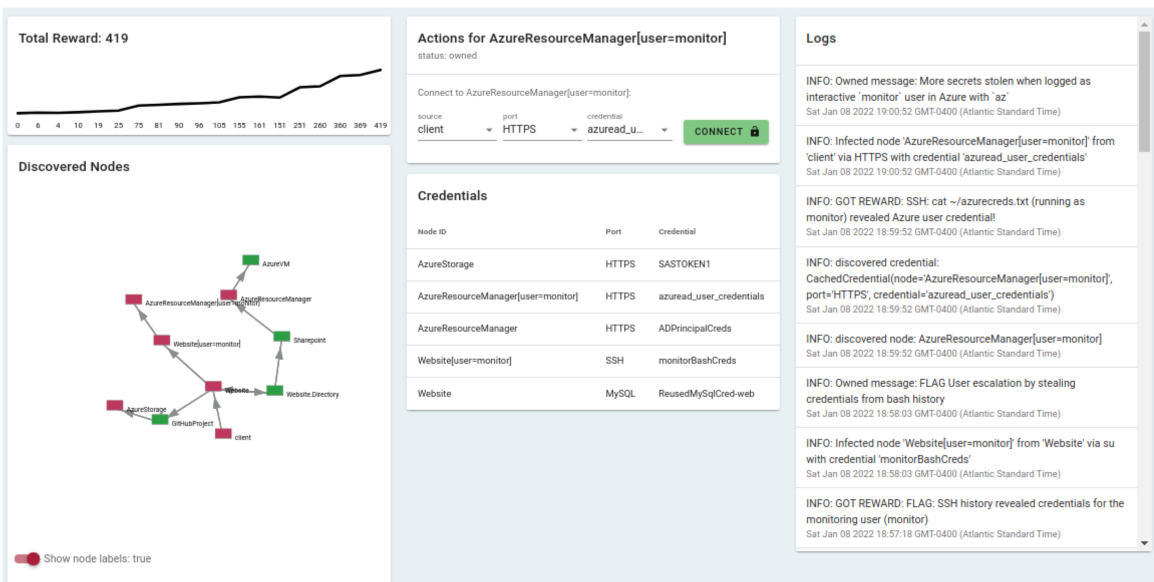


Figure B-17: Access Azure Resource Manager with monitor's credentials

Bibliography

[1] Ziad Ismail, Jean Leneutre, David Bateman, and Lin Chen. A game theoretical analysis of data confidentiality attacks on smart-grid ami. *Selected Areas in Communications, IEEE Journal on*, 32:1486–1499, 07 2014.

[2] Xiaohe Fan, Kefeng Fan, Yong Wang, and Ruikang Zhou. Overview of cyber-security of industrial control system. In *2015 international conference on cyber security of smart cities, industrial control system and communications (SSIC)*, pages 1–7. IEEE, 2015.

[3] Heng Zhang, Yuanchao Shu, Peng Cheng, and Jiming Chen. Privacy and performance trade-off in cyber-physical systems. *IEEE Network*, 30(2):62–66, 2016.

[4] Defense Use Case. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 388:1–29, 2016.

[5] Microsoft Defender Research Team et al. Cyberbattlesim, 2021.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[7] Thanh Thi Nguyen and Vijay Janapa Reddi. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–17, 2021.

[8] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games, 2019.

[9] Vlad Firoiu, Tina Ju, and Josh Tenenbaum. At human speed: Deep reinforcement learning with action delay, 2018.

[10] Erich Walter, Kimberly Ferguson-Walter, and Ahmad Ridley. Incorporating deception into cyberbattlesim for autonomous defense. *arXiv preprint arXiv:2108.13980*, 2021.

[11] Maxwell Standen, Martin Lucas, David Bowman, Toby J. Richer, Junae Kim, and Damian Marriott. Cyborg: A gym for the development of autonomous cyber agents, 2021.

[12] David Maynor. *Metasploit toolkit for penetration testing, exploit development, and vulnerability research.* Elsevier, 2011.

[13] Joseph Khoury and Mohamed Nassar. A hybrid game theory and reinforcement learning approach for cyber-physical systems security. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9, 2020.

[14] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, 2012.

[15] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *CoRR*, abs/1811.12560, 2018.

[16] Xiaorui Liu, Juan Ospina, and Charalambos Konstantinou. Deep reinforcement learning for cybersecurity assessment of wind integrated power systems. *IEEE Access*, 8:208378–208394, 2020.

[17] Iker Zamora, Nestor Gonzalez Lopez, Victor Mayoral Vilches, and Alejandro Hernandez Cordero. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo, 2017.