Simulating Network Lateral Movements through the CyberBattleSim Web Platform[*]

Jonathan Esteban[1]

Massachusetts Institute of Technology, Cambridge, USA
jesteban@mit.edu

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.

Contents

[*]This work was co-funded by "Fondo Europeo di Sviluppo Regionale Puglia POR Puglia 2014 – 2020 – Asse I – Obiettivo specifico 1a – Azione 1.1 (RS) - Titolo Progetto: Suite prodotti Cybersecurity e SOC" and BV TECH S.p.A. This work was co-funded by Cybersecurity at MIT Sloan (CAMS https://cams.mit.edu)

1 Introduction

Modern cyber attacks demand immediate critical decision making [4]. 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 to preserve the system's integrity come at different trade-offs for the system's availability and security. [2] [10]

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. [1]

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 research 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. [8] 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.

1.1 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. [5] [6] 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. [7] 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. [3]

1.2 Contributions

This paper 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.

1.3 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.

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. [9] 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.

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.

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.

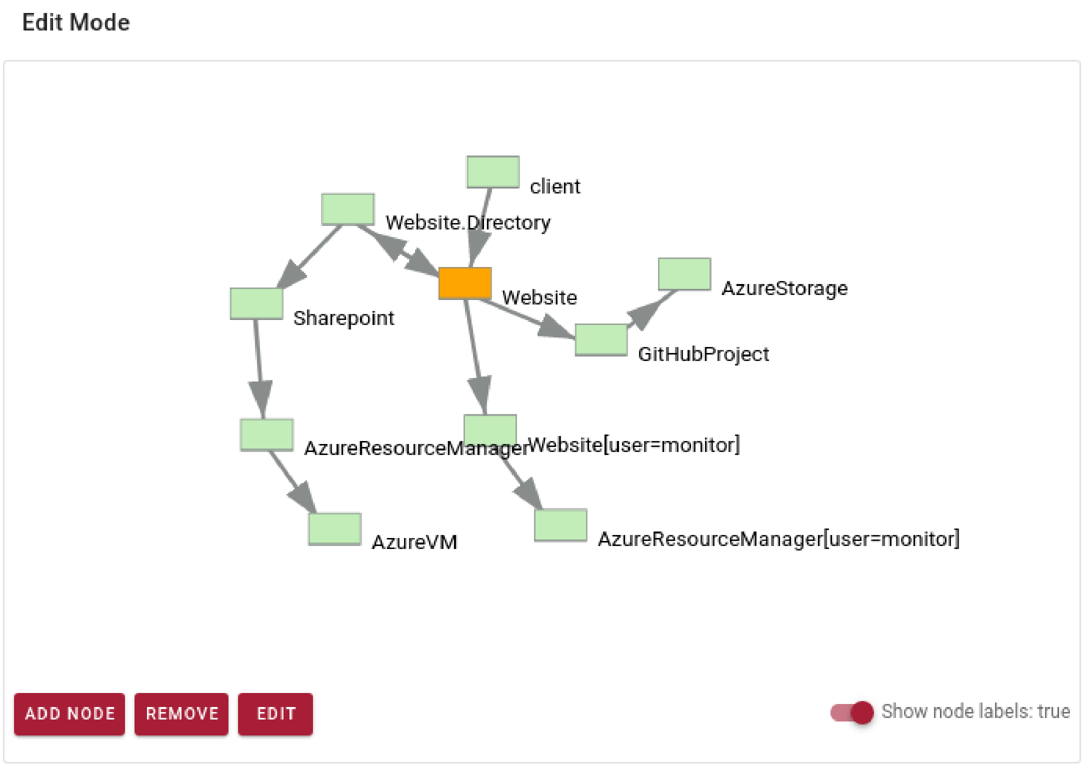


Figure 1: Network modeling interface on sample network topology.

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.

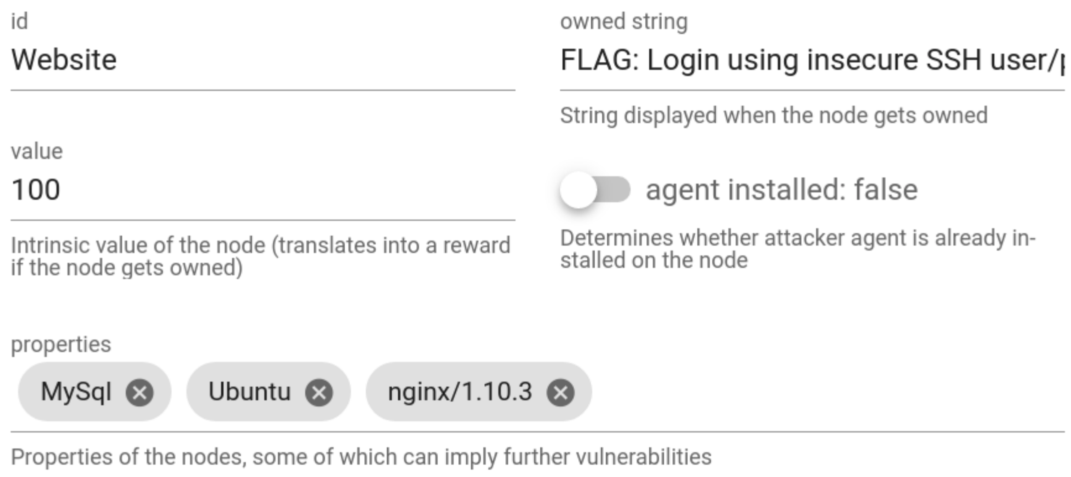
Besides adding or removing nodes to the graph, a user may edit various attributes of a node, including: intrinsic value, vulnerabilities, services, available ports and firewalls. These attributes are defined within the CyberBattleSim project and are described in Table 1. Many of these attributes have their own nested properties, allowing the user to finely specify a node's behavior.

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

Table 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


id
Website

owned string
FLAG: Login using insecure SSH user/
String displayed when the node gets owned

value
100
Intrinsic value of the node (translates into a reward if the node gets owned)

agent installed: false
Determines whether attacker agent is already installed on the node

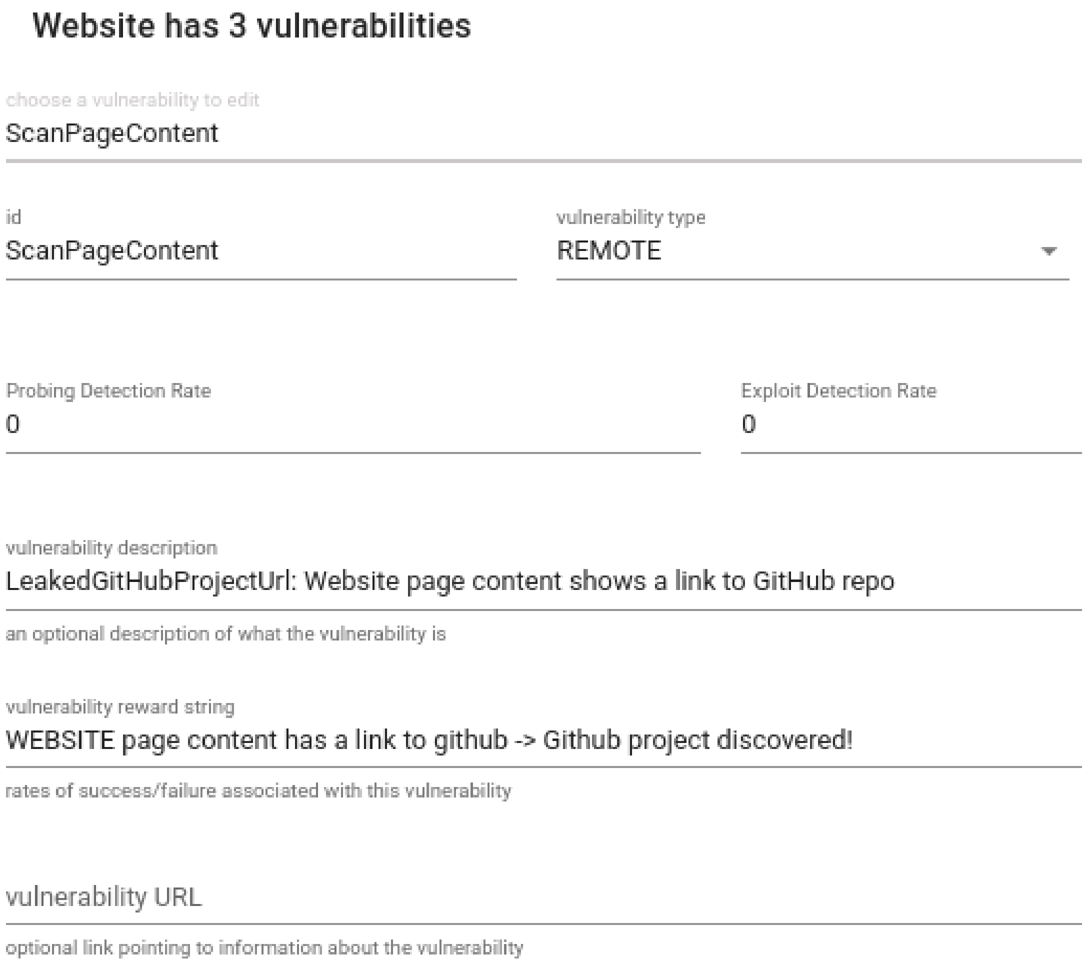
properties
MySql Ubuntu nginx/1.10.3
Properties of the nodes, some of which can imply further vulnerabilities

Figure 2: "General Properties" tab within the network modeling interface.

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 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.


Website has 3 vulnerabilities

choose a vulnerability to edit
ScanPageContent

id	ScanPageContent
vulnerability type	REMOTE
Probing Detection Rate	0
Exploit Detection Rate	0

vulnerability description
LeakedGitHubProjectUrl: Website page content shows a link to GitHub repo
an optional description of what the vulnerability is

vulnerability reward string
WEBSITE page content has a link to github -> Github project discovered!
rates of success/failure associated with this vulnerability

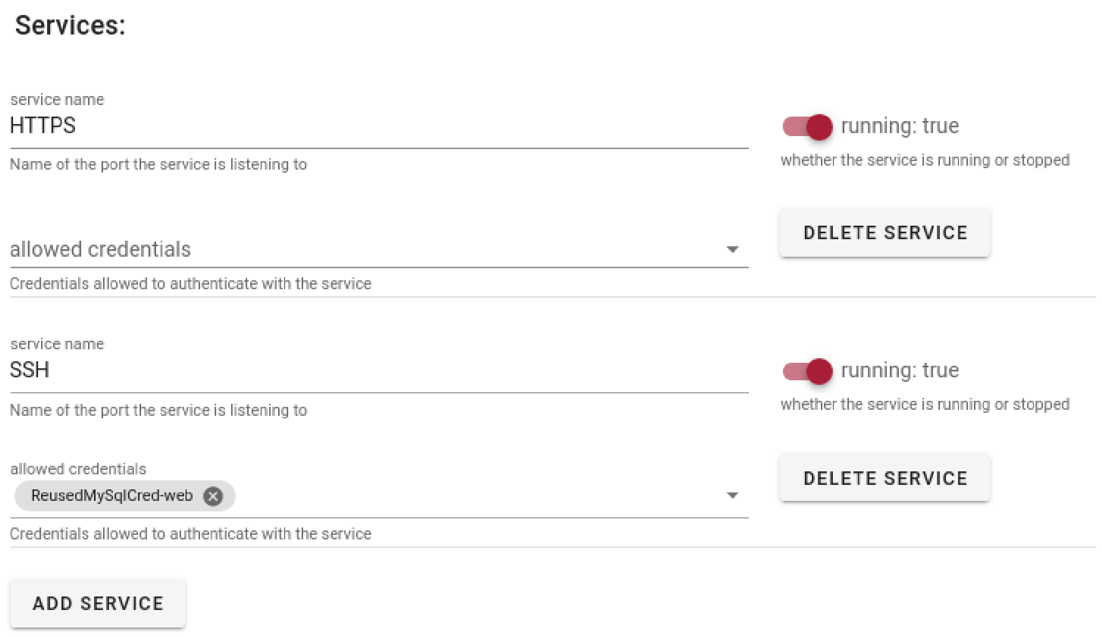
vulnerability URL
optional link pointing to information about the vulnerability

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

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.


Services:

service name
HTTPS
Name of the port the service is listening to

running: true
whether the service is running or stopped

allowed credentials
Credentials allowed to authenticate with the service

service name
SSH
Name of the port the service is listening to

running: true
whether the service is running or stopped

allowed credentials
ReusedMySqlCred-web
Credentials allowed to authenticate with the service

Figure 4: ”Services” tab within the network modeling interface.

3.1.4 Firewall Rules

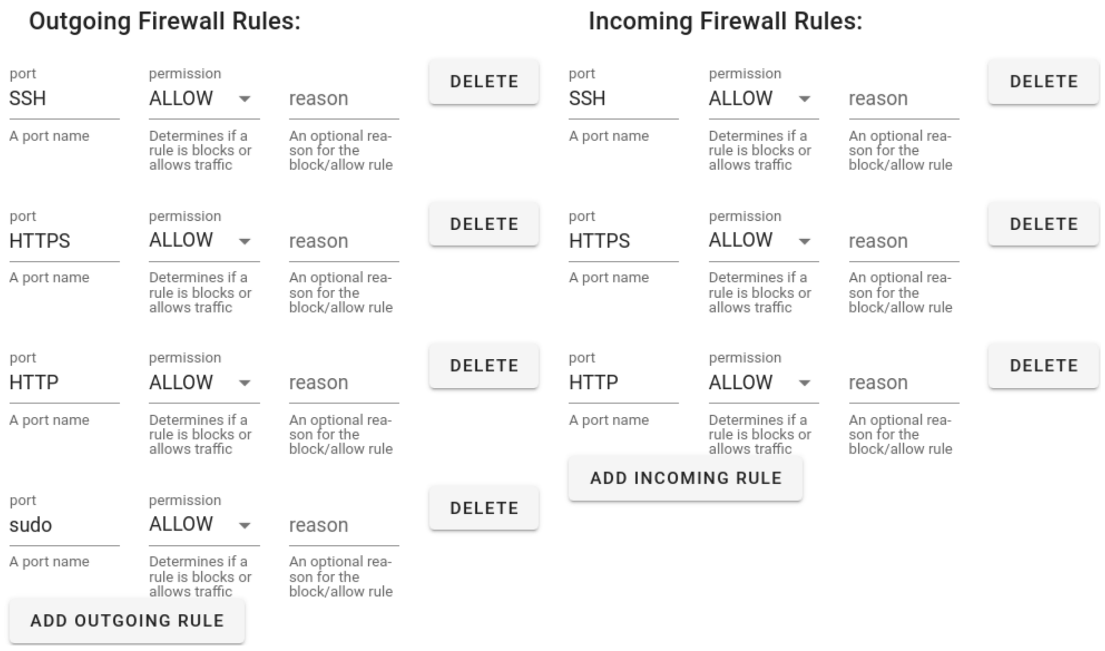
Finally, a user may add firewall rules to a node. Firewall rules can be used to block or allow certain ports. These rules can be defined for both outgoing and incoming traffic. Ports that are not explicitly allowed in the configuration are automatically assumed to be blocked. That said, explicitly blocking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


Outgoing Firewall Rules:

port	permission	reason	
SSH	ALLOW	reason	DELETE
A port name	Determines if a rule is blocks or allows traffic	An optional reason for the block/allow rule	
HTTPS	ALLOW	reason	DELETE
A port name	Determines if a rule is blocks or allows traffic	An optional reason for the block/allow rule	
HTTP	ALLOW	reason	DELETE
A port name	Determines if a rule is blocks or allows traffic	An optional reason for the block/allow rule	
sudo	ALLOW	reason	DELETE
A port name	Determines if a rule is blocks or allows traffic	An optional reason for the block/allow rule	

ADD OUTGOING RULE

Incoming Firewall Rules:

port	permission	reason	
SSH	ALLOW	reason	DELETE
A port name	Determines if a rule is blocks or allows traffic	An optional reason for the block/allow rule	
HTTPS	ALLOW	reason	DELETE
A port name	Determines if a rule is blocks or allows traffic	An optional reason for the block/allow rule	
HTTP	ALLOW	reason	DELETE
A port name	Determines if a rule is blocks or allows traffic	An optional reason for the block/allow rule	

ADD INCOMING RULE

Figure 5: "Firewall rules" tab within the network modeling interface.

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.

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 3) 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

Table 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

interact with. Once complete, a gallery of figures 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.

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.

Table 3: Q-Learning CTF Simulation Parameters as described in Table 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%

4.1 Human Interaction with CTF Network Topology

The replicated CTF environment can be seen in Figure 1. Because every node property listed in Table 1 can be configured, virtually any network topology can be abstracted into CyberBattleSim's model.

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 8 through 11 display the results of running Q-Learning on the CTF Network with the parameters in Table 3. 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.

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. We designed the platform interface with blue team and red team dynamics in mind; a blue team member would design the network topology and would then hand it over to a red team player for them to try to compromise and gain vulnerability insights. 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.

References

[1] 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.

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

[4] 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.

[5] 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.

[6] 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.

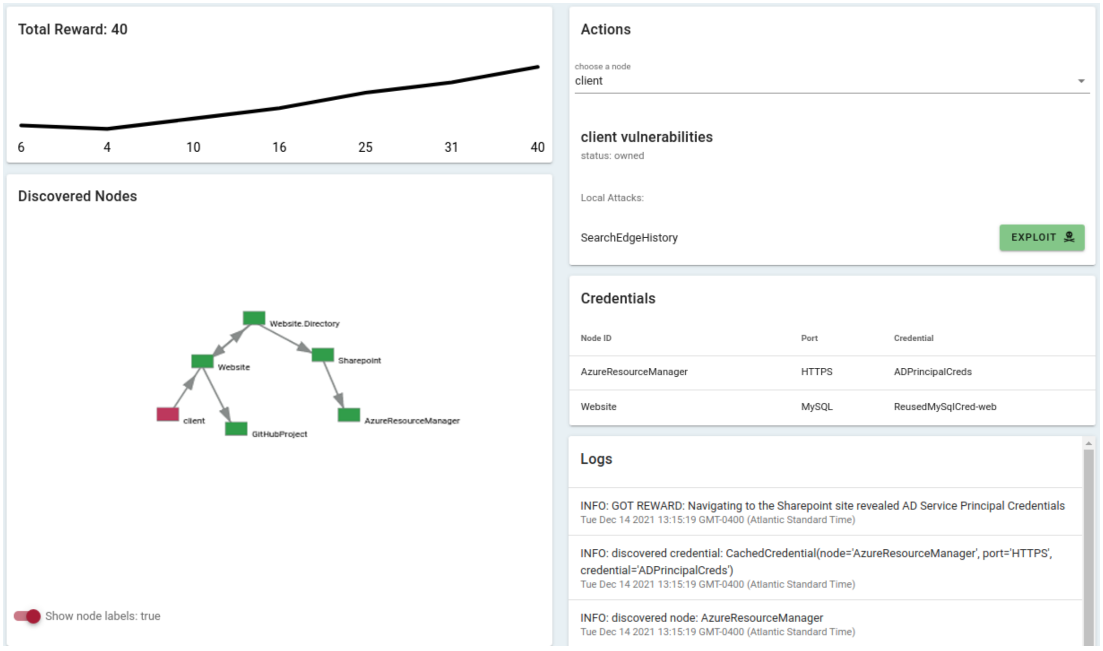
[7] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games, 2019.

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

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

[10] 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.

A Additional Figures



Credentials

Node ID	Port	Credential
AzureResourceManager	HTTPS	ADPrincipalCreds
Website	MySQL	ReusedMySqlCred-web

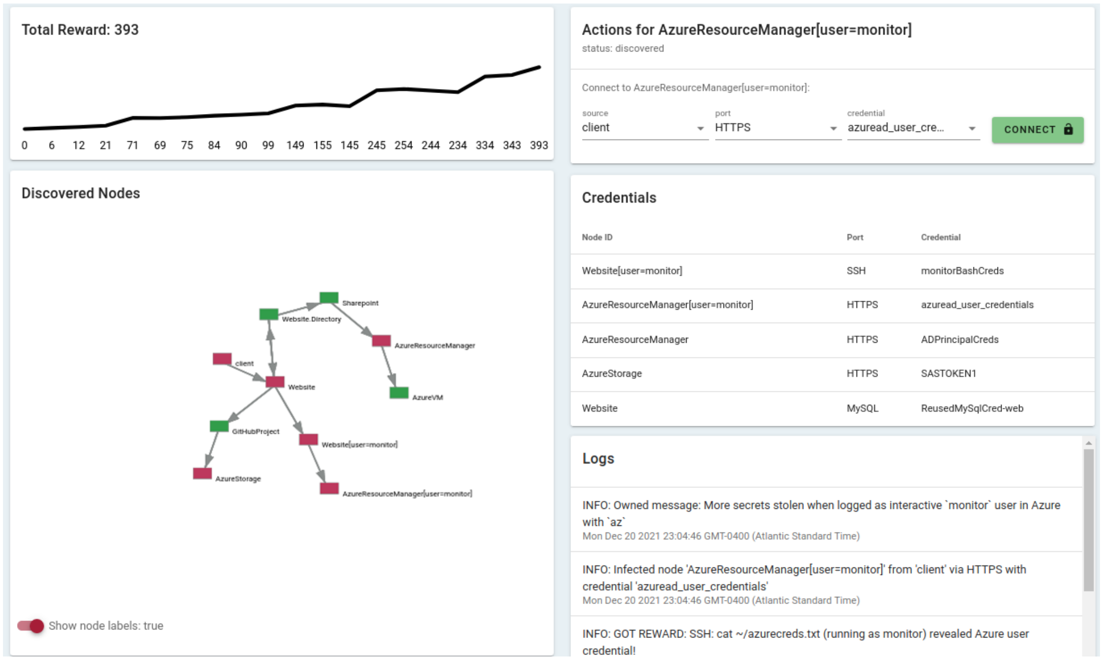
Logs

```
INFO: GOT REWARD: Navigating to the Sharepoint site revealed AD Service Principal Credentials
Tue Dec 14 2021 13:15:19 GMT-0400 (Atlantic Standard Time)

INFO: discovered credential: CachedCredential(node='AzureResourceManager', port='HTTPS',
credential='ADPrincipalCreds')
Tue Dec 14 2021 13:15:19 GMT-0400 (Atlantic Standard Time)

INFO: discovered node: AzureResourceManager
Tue Dec 14 2021 13:15:19 GMT-0400 (Atlantic Standard Time)
```

Figure 6: Attack progression on sample network topology showcasing a blocked action via a firewall, resulting in a score penalty.



Credentials

Node ID	Port	Credential
Website[user=monitor]	SSH	monitorBashCreds
AzureResourceManager[user=monitor]	HTTPS	azuread_user_credentials
AzureResourceManager	HTTPS	ADPrincipalCreds
AzureStorage	HTTPS	SASTOKEN1
Website	MySQL	ReusedMySqlCred-web

Logs

```
INFO: Owned message: More secrets stolen when logged as interactive `monitor` user in Azure
with `az`
Mon Dec 20 2021 23:04:46 GMT-0400 (Atlantic Standard Time)

INFO: Infected node 'AzureResourceManager[user=monitor]' from 'client' via HTTPS with
credential 'azuread_user_credentials'
Mon Dec 20 2021 23:04:46 GMT-0400 (Atlantic Standard Time)

INFO: GOT REWARD: SSH: cat ~/azurecreds.txt (running as monitor) revealed Azure user
credential!
```

Figure 7: Entire network has been compromised and all flags have been acquired.

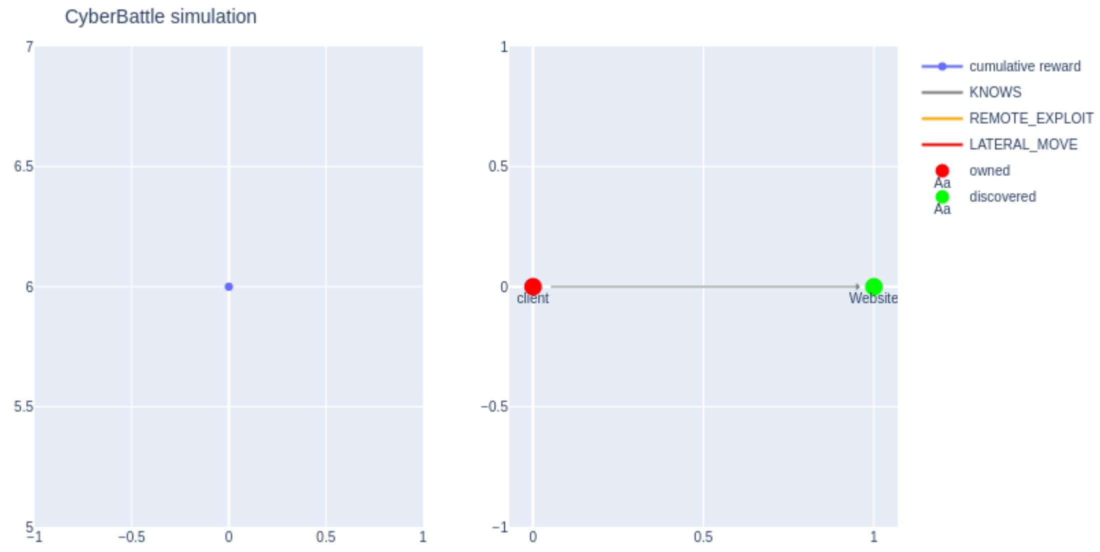


Figure 8: Step 1 of attack progression under Q-Learning AI agent

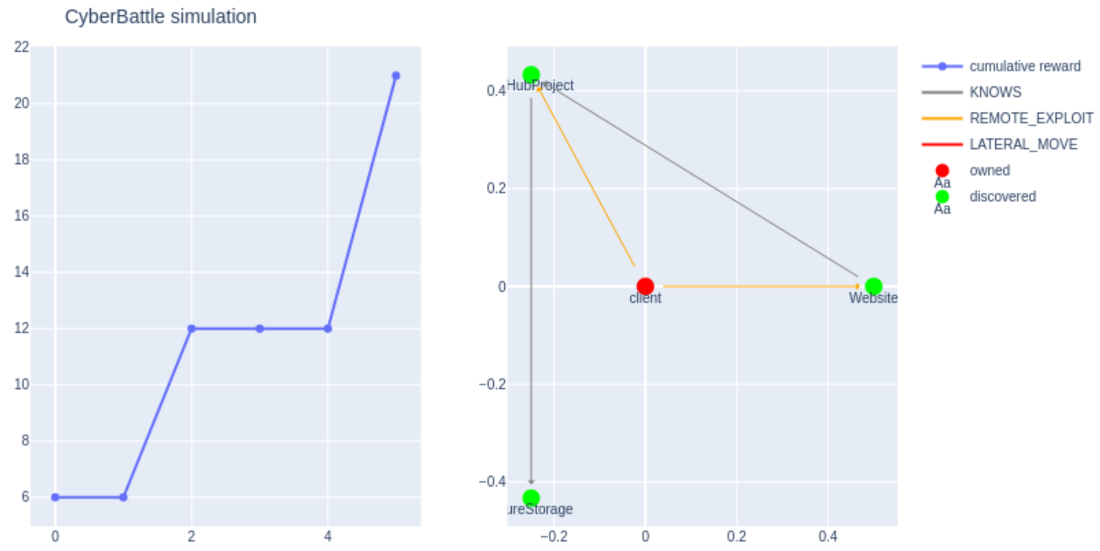


Figure 9: Step 3 of attack progression under Q-Learning AI agent

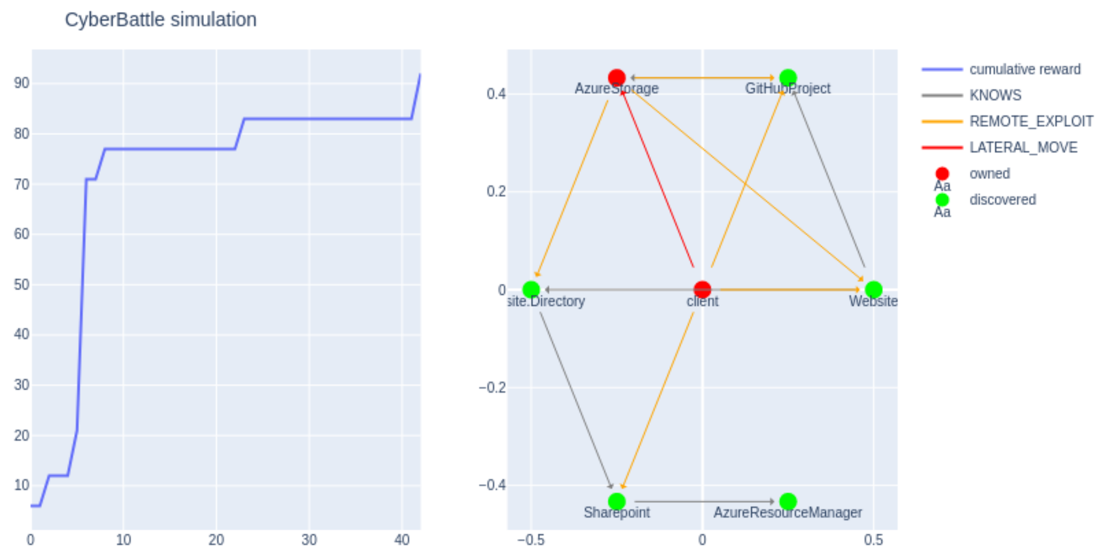


Figure 10: Step 4 of attack progression under Q-Learning AI agent

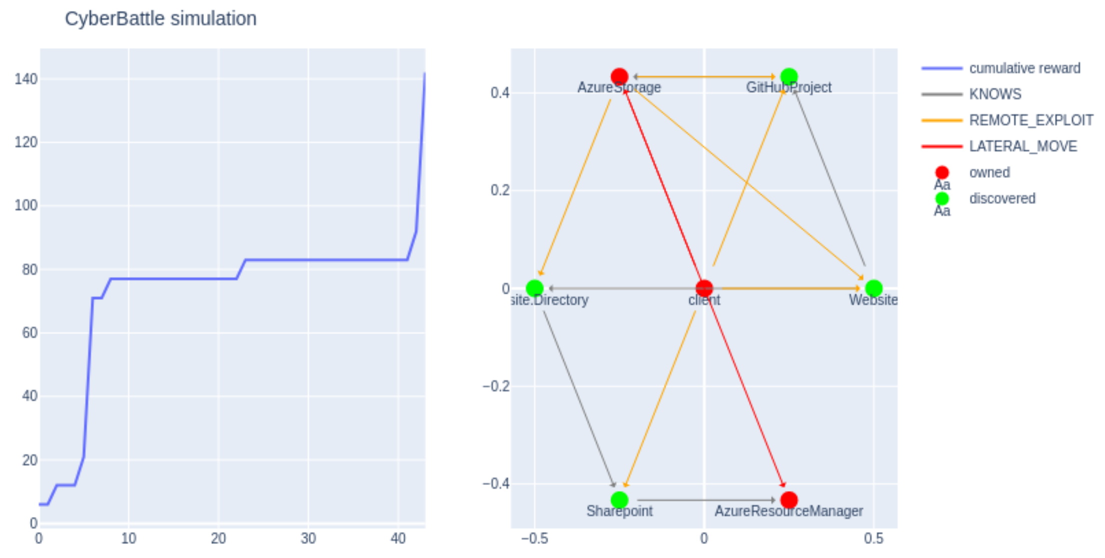


Figure 11: Step 5 of attack progression under Q-Learning AI agent