2021-03-18

Automated Cyber Ranges: Design Features, Architectures, Scenarios and Impacts

Dezhang Wen

Honors Thesis

AUTOMATED CYBER RANGES: DESIGN FEATURES, ARCHITECTURES, SCENARIOS, AND IMPACTS

by

Dezhang Wen

Submitted to Brigham Young University in partial fulfillment

of graduation requirements for University Honors

Information Technology Department

Brigham Young University

April 2021

Advisor: Justin Giboney

Faculty Reader: Amanda Hughes

Honors Coordinator: Derek Hansen

ABSTRACT

AUTOMATED CYBER RANGES: DESIGN FEATURES, ARCHITECTURES, SCENARIOS, AND IMPACTS

Dezhang Wen

Information Technology

Bachelor of Technology

As cybersecurity becomes increasingly important in the digital world, the need for a virtual environment where security professionals can safely practice defending against real-life attacks is gradually rising. This thesis explores, participates in, and expands upon the design and implementation of such an environment, also known as a “cyber range”. We build a model where real-life attacks and defense can be successfully simulated, as well as further improving the process through automation. Ultimately, it proposes and experiments with the idea of an automated cyber range in order to enhance both the efficiency and effectiveness of a security testbed.

ACKNOWLEDGMENTS

I would like to thank Dr. Justin Giboney for providing me the opportunity of working on this project, as well as all of his guidance and help along the way. I also would like to thank Dr. Derek Hansen and Dr. Amanda Hughes for their support and assistance. Lastly, I would like to thank every person who has participated in the project, especially Jacob Siebach, Timothy Smith, Maria Feist, and Kylie Johnson. Without our combined effort, this project would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

Introduction

As the world becomes mediated through digital technologies, cyber-attacks are also becoming more common and dangerous. In 2003, the worldwide expenditure due to cyberattacks, such as botnet and denial-of-service, exceeded thirteen billion dollars and this number has only been going up in the past years (Cashell etc., 2004). Apart from ones performed by hacking enthusiasts and security professionals, there are various malicious attacks conducted at large scales by organized crime groups and even teams backed by government agencies across the globe. Such attacks should not be taken lightly not only because of the pecuniary cost they can bring, but also the threat they possess to people's right of online privacy and freedom.

In order to better combat and prevent cyber-attacks, it is essential to provide a safe testbed where security professionals can safely practice defending against real-life attacks. Such testbeds are known as "cyber ranges". A cyber range is defined as a controlled virtual environment used in cybersecurity training as an efficient way for professionals to gain practical knowledge through hands-on activities (Pham etc., 2016). To accomplish the said goal, features that a cyber range implements include simulating common cyber-attacks, creating real-life security scenarios, testing possible defense techniques, and others. However, designing, building, and evaluating such environments is a challenging problem and one that warrants extensive research and experiment in order to succeed.

1.1 Past Research

The earliest forms of security training were often set in a physical classroom setting, where slides were presented and lectures were given. Over the years, such conventional methods of security training have been proven to be ineffective. Not only are most of the lectures and presentations repetitive, the lack of practical experience involved also greatly hinders people's abilities to translate classroom concepts into real-world applications. Although such methods are still commonly used in the security field today, researchers and industry professionals have been experimenting and developing new ways for better security training, including hands-on competitions, challenges, and exercises. Such events will not only enhance participants' diverse security skills, but also do so in a more practical and fun fashion.

Capture the Flag (CTF) is among one of the most popular security training programs initiated in the recent decade. A CTF competition is one where the participants search for flags in a virtual environment and score points for finding them. Because of its game-like model and extensive hands-on opportunities, CTF has quickly gained interest in the security field and has been adapted to various security training procedures. One of the largest CTF competitions is called the GenCyber Capture the Flag competition, created and sponsored by the United States National Security Agency (NSA) as part of its GenCyber program (McDaniel etc., 2016). Participants who compete in this GenCyber CTF range from middle schoolers all the way up to college students, and other more advanced CTF competitions even attract audiences with full-time security jobs.

Other than CTF, cyber defense competitions are also ever-growing around the nation. While a CTF competition puts the participant in the shoes of the attacker in order to spot the possible vulnerabilities in an application, a cyber defense competition does the opposite. It lets the participants take on a defensive role in trying to defend against various attacks. The very first cyber defense competition, the Cyber Defense Exercise, was create by the NSA to test the ability of students from various military academies to defend certain networks. In recent years, more and more schools and organizations have incorporated this idea into security training, creating more cyber defense competition and at a larger scale. One example is the National Collegiate Cyber Defense Competition, where students from all over the nation compete with each other to protect established network infrastructures (Conkin etc., 2006).

1.2 Current State

While it is indeed exciting to see new models and methods for security training emerging, the idea of establishing a cyber range by incorporating hands-on exercises into one environment has only come forward recently. There are only a few research projects dealing with designing such environments. While there are many theories and concepts being explored in the security field, technical implementation and design details are still in the early stage.

The Nation Cyber Range (NCP) is one of the largest and earliest cyber range projects that is currently active. It is established by the Defense Advanced Research Projects Agency (DARPA), part of the United States Department of Defense. The NCR, once successfully developed, serves to provide a unique environment for cyber security testing by using different methods to assess resiliency to advanced cyberspace security threats. The NCP approaches the challenge of designing a cyber range by representing an Internet-like environment by employing a multitude of virtual machines and physical devices augmented with traffic emulation, vulnerability scanning, data capture, and penetration tools (Ferguson etc., 2014). While the NCP is still under development and testing, the overarching concept has been quickly adopted by many other researchers and government agencies as a guideline for designing a cyber range.

The Cyber Range Instantiation System (CyRIS) is another example of an on-going cyber range project that is more transparent, up-to-date, and comprehensive. Developed by researchers from the Japan Advanced Institute of Science and Technology (JAIST), CyRIS is a cyber range model designed to automatically prepare and manage a cyber range for cybersecurity education or training based on specifications manually defined (Pham etc., 2016). CyRIS's method to achieve automation in creating and managing a cyber range is to deploy one machine first through pre-defined scripts and then clone that machine a number of times in order to set up the entire environment. If the NCP solves the challenge of designing the overarching architecture of a cyber range, then CyRIS solves the challenge of how to efficiently deploy and implement such an architecture.

While there are numerous other existing research projects on the topic of the cyber range, the above two are the most prominent ones and ones that are widely referenced in the security field. After careful examination and analysis of the current state of cyber ranges, a project overview is presented in the next section.

1.3 Project Overview

First of all, the research question of this project was presented by the BYU Cybersecurity Research Lab: how to successfully design, implement, and automate a cyber range in order to test various cyber attacks and defenses effectively, efficiently and securely. Circling this research question, the entirety of the project could be divided into four major steps: designing, implementing, automating, and testing. Each of the steps was essential to the entirety of the project and will be explained in detail individually below.

- Designing

  The designing portion was the first step in this project. This portion was mainly brainstorming and coming up with the design of the cyber range and was possibly the most important one out of the four. It consisted of thinking about various rudimentary problems and solving them, such as deciding how to set up the cyber range, what different components it would have, how the components would talk to each other, what technologies to use for each component, how to connect the various technologies, and others.

- Implementing

  After the designing was finished, the second step was to implement the design. This step established the overarching architecture of the range, as well as installing and connecting any technologies needed. How the implementation was done largely depended on how the range was designed. Choosing the most efficient and safe tools, algorithms, and applications was the key.

- Automating

  The third step of the project was to automate the range, which technically was still part of the implementation. As stated previously, research surrounding automation in cyber ranges was very scarce and most research is still in early stages. This step was to experiment on how to run the cyber range with as little manual interaction as possible, potentially making it all automated.

- Testing

  After the previous steps were both finished, the last step of the project was to test everything. What tests to run, how the tests were run, and what results to expect also largely depended on the design and partially the implementation. Example tests included port scanning, fingerprint, injection attack, and others.

Design Features

This section of the thesis talks about the design features of the project. Such features include the flow of the range, application choices, and others. The central research question of the project was how to successfully design, implement, and automate a cyber range in order to test various cyber attacks and defenses effectively, efficiently and securely, and it is important to keep this question in mind during the design process.

2.1 Hosts Setup

Similar to NCP's approach to designing a cyber range, the flow of the cyber range started out with setting up the virtual machines, which were used as hosts in the range. Such machines were divided up into two teams: the red team and the blue team. Each of the two team was essential to the overall design and each had its own responsibilities and tasks. First of all, the red team was the offensive team and its main responsibility was to attack. Each machine in the red team was able to deploy various attacks onto specified targets. The targets, secondly, were virtual machines on the blue team. The blue team was the defensive team and its main responsibility was to defend against the red team. Each machine in the blue team was pre-configured with various kinds of vulnerabilities that the red team machines could attack. Ultimately, machines for both the red and blue teams were the main players in the range; they provided the platform for different vulnerabilities to be installed, as well as various types of attacks to be deployed. It was important that all the machines should get set up in an efficient and safe environment. Efficiency was needed because fast recreation and teardown of the machines were required for experimenting with different scenarios; safety was also important because certain attacks were extremely malicious if not carefully handled and it was absolutely necessary to ensure that no damages were done to actual devices, applications, or servers.

2.2 Networking Configuration

After the virtual machines for both the red and blue teams were set up and ready to go, the next step was to configure the networking between the machines. For both the red and blue teams, all the machines were in the same network environment; additionally, the first three ports that should be open were port 80 for HTTP, port 443 for HTTPS, and port 22 for SSH. The internet ports needed to be open because certain services or applications needed to be installed on the machines as preparations. The port for SSH needed to be open in order to support manual configurations inside the machines. After these three ports, other networking configurations could then be applied depending on the types of attacks and vulnerabilities tested. For example, if an SQL injection attack were to be tested against a misconfigured File Transfer Protocol (FTP) server, then port 21 for FTP should be open. Therefore, the networking differed due to the different attacks being experimented with, but port 80, 443, and 22 should be opened up for installing services and manual configurations.

2.3 Internal Logic

Once the virtual machines and networking were completely configured, it was time to design the internal logic of the cyber range itself. The first thing to think about was how to set up the attacks. The approach that we took to solve this challenge was to have different types of attacking scripts targeted at different types of vulnerabilities. Such scripts were the attacks used by the red team and they could be either stored physically on the red team machines or remotely stored in an online container where the red team machines could grab them from. After the attacking scripts were written and ready to go, the blue team machines would install the necessary vulnerabilities corresponding to whatever attacks were available. This way, each attacking script had a target blue team machine to deploy an attack onto. It was important to note here that one red machine could have more than one attacking script stored; similarly, each of the blue machines could have more than one vulnerability. This directly related to the next phase in the design, which was automation.

2.4 Redteam Engine / Automation

After the attacking scripts were set up on the red team machines and the vulnerabilities were installed on the blue team machines, now it was time to test out the attacks. Because one of the goals in the research question was to automatically deploy attacks onto various vulnerabilities, it was not enough to manually send out the attack scripts one by one onto the blue machines; we wanted to do so in a way that required as few user interactions as possible, potentially achieving full automation. There are a few main reasons why we would want automation in a cyber range. First of all, it could greatly increase the fairness to different teams and make sure that there were not any errors from the red team. Secondly, automation can greatly increase the efficiency for security training. As of currently, most tasks of the red teams in a cyber range were still being done by actual humans. By making the red team automated, it could reduce the time of setup and response, ultimately making the whole process more fluid and less time-consuming. To accomplish this goal of automation, we took the approach of building a “Redteam Engine”. The engine reached the goal of automated attacks through the following steps:

1. Take out all the available attack scripts from either a local directory or an online container
2. Store a reference to each script into a data structure that allows for easy insertion and deletion
3. Loop through the data structure
4. Each time a reference is found, deploy the attacking script corresponding to the reference onto the target blue machine
5. After the deployment is finished, the reference will be removed and the next reference will be called to deploy the next script

6. After every reference is called and nothing is left, the loop stops and an intelligencer checks the attacking results

The engine should be installed on every red team machine, thus potentially automating the attacks in the cyber range. This was the design features of an automated cyber range, and further changes or improvements are discussed later in this paper.

Architecture Implementation

After successfully designing the overall architecture of the cyber range, implementation came. This section of the paper talks about the technologies used for implementation, the process of implementing each functionality, and what they accomplished in response to the central research question.

3.1 Technologies to Use

Before getting into the details of the architecture implementation of the cyber range, it was necessary to determine what technologies to use and how to use them compatibly. As stated in the "Design Features" section, it was important to choose technologies that were convenient, stable, and safe. Stability was important because the entire range would be built on top of all the technologies chosen, and it was important they should not conflict with each other during any stage or cause any problems. Convenience was needed for potential reproduction and duplication, and safety should always be kept in mind when dealing with potentially malicious software or applications. After careful selection and review, the following technologies were used during stages of the implementation; each of them is further explained later in detail on why they were chosen, how they were used, and what they accomplished for the project.

- Terraform

  Terraform is an open-source infrastructure as code software tool used for building, changing, and versioning infrastructure safely and efficiently. This was used to automate the setup for all the virtual machines and networking configurations.

- Amazon Web Service (AWS)
  - Elastic Compute Cloud (EC2)

    EC2 is a service that allows users to launch and manage AWS resources, such as virtual machines, in the cloud with minimum friction. This was used to host all of the virtual machines, both red team and blue team.
  - Virtual Private Cloud (VPC)

    VPC is a service that allows users to compute inside an isolated virtual network, which will be used to manage the networking configuration of the virtual machines.
- Git / GitHub

  Git / GitHub is a tool used mainly for code storage and version control. It allows for efficient team cooperation within a team, as well as acting as a backup method for the project in case of emergencies or accidents.

3.2 Implementation Sequence Diagram

The implementation of the design was divided into five main stages: deployment, initialization, queue, script, and exploiter. A sequence diagram for the implementation was shown in Figure 1. Each of the individual components of the design was explained in detail in its own section. The deployment stage corresponded to “Hosts Setup” and “Network Configuration” in the design, while the remaining four components corresponded to “Redteam Engine / Automation”.

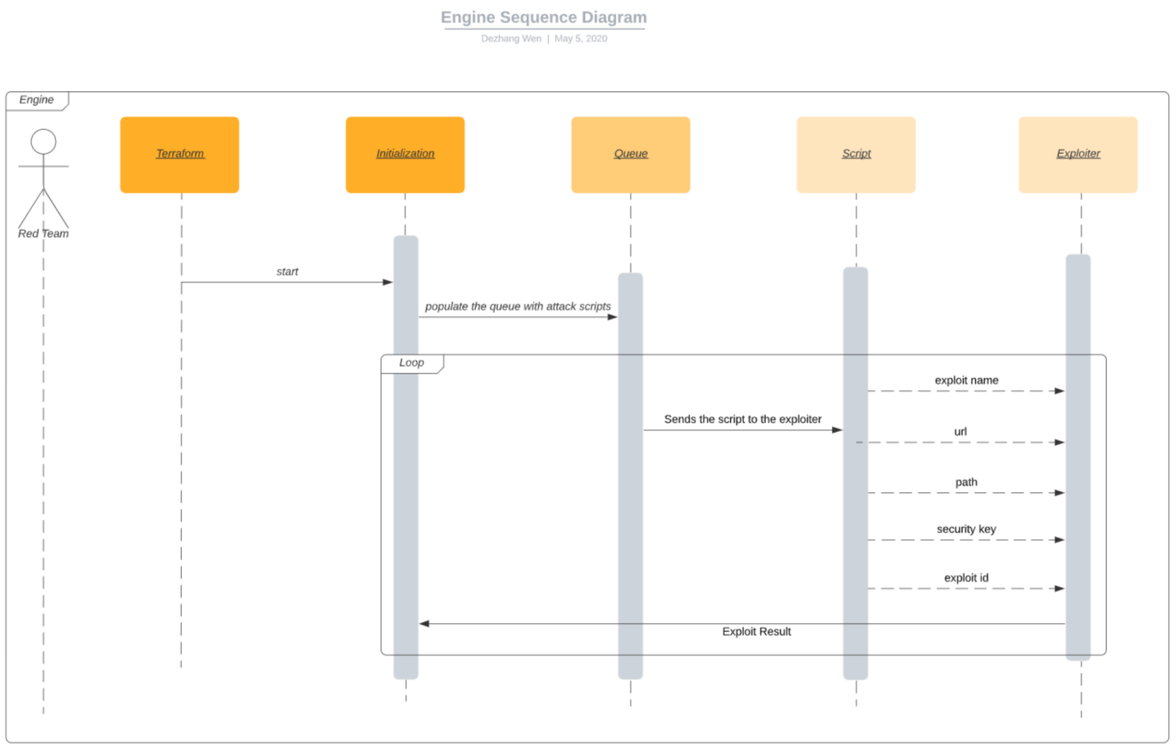


Figure 1: This image is the sequence diagram for the overall design of the cyber range being built. Starting from left to right are the five components of the design: Terraform

for the deployment, initialization using NodeJS, queue to store the attacking scripts, the attacking scripts, and the exploiter that executes each script.

3.3 Deployment

As the first stage, the deployment establishes the foundation of the entire project; it consists of the deployment of virtual machines for both the red and blue teams, as well as the networking configurations. During this stage of the implementation, the red team machines inherite most of the other design features, specifically the queue, the scripts, and the exploiter. Further details about this are explained in later sections. On the other hand, the blue team machines mainly consisted of various vulnerabilities that the red team machines would exploit. Therefore, it was important to differentiate the two teams from each other in the deployment process so that it is obvious which machine belongs to which team.

The two major technologies used in the deployment stage were Terraform and AWS, specifically EC2 and VPC. While EC2 and VPC were used to host the virtual machines and manage the network, Terraform was used to automate the process of the setup. If the user had to go into the AWS Console and manually prepare everything, that would be extremely inefficient and defeating the purpose of an automated cyber range. In order to achieve automatic setup, rather than duplicating a manually created machine like the CyRIS project, Terraform was used to accomplish the goal. The fact that the entire cyber range could be treated as an infrastructure and Terraforms compatibility with AWS made the entire process straightforward and convenient. Therefore, a Terraform script was created so that when it ran with one simple command, it set up everything needed for the deployment stage.

The Terraform script can be broken down into five major components: networking which consists of VPC and subnetting, security groups, hosts, and tasks. Each component is directly related to the configuration of the virtual machines and networks that are being deployed in AWS.

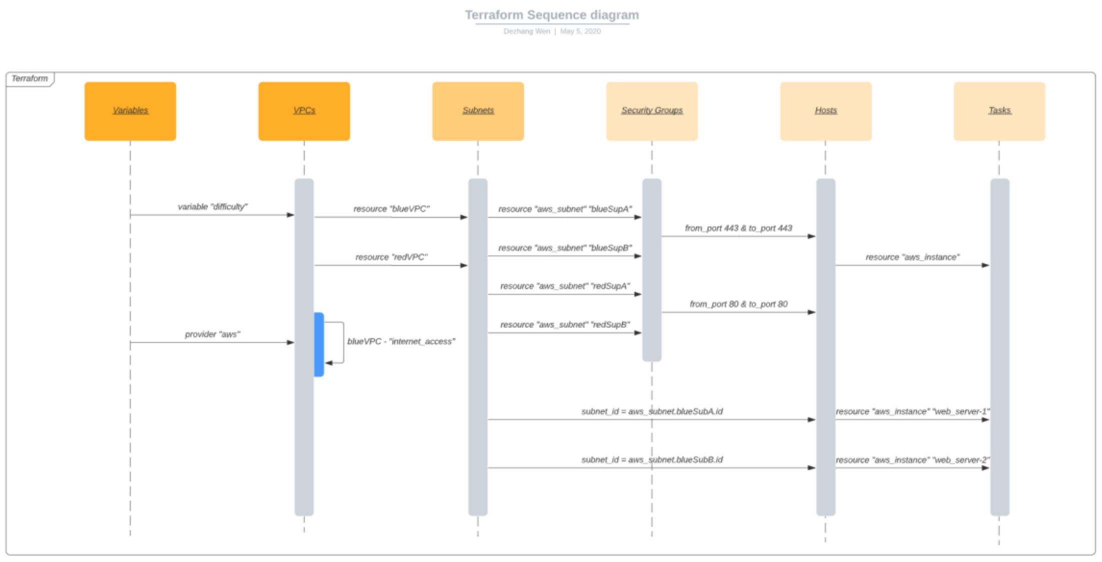

Terraform Sequence diagram
Dezhang Wen | May 5, 2020
Terraform
Variables
VPCs
Subnets
Security Groups
Hosts
Tasks
variable "difficulty"
resource "blueVPC"
resource "redVPC"
provider "aws"
blueVPC - "internet_access"
resource "aws_subnet" "blueSupA"
resource "aws_subnet" "blueSupB"
resource "aws_subnet" "redSupA"
resource "aws_subnet" "redSupB"
from_port 443 & to_port 443
from_port 80 & to_port 80
resource "aws_instance"
subnet_id = aws_subnet.blueSubA.id
subnet_id = aws_subnet.blueSubB.id
resource "aws_instance" "web_server-1"
resource "aws_instance" "web_server-2"

Figure 2: Sequence Diagram of the Terraform Script. From left to right: Variables, VPCs, Subnets, Security Groups, Hosts, and Tasks.

3.4 Initialization

After successfully running the Terraform script, the virtual machines and network for both the red team and the blue were ready to go. Next was to start the initialization stage of the cyber range. This was also the first stage of "Redteam Engine/Automation" from the design.

The initialization stage of the design was divided into three main steps, as shown in the following order:

1. Grab an available attacking script stored in either a local directory or a remote container.
2. Create an object based on the reference to local path of the current attacking script. The object consists of various attributes based on the script, which will be further discussed in the next stage.
3. Pushing object created into the queue, which is the next stage.
4. Keep populating the queue with every object created until there is no more available script to create object from
5. Call the queue

3.5 Queue

The third stage of the design of the cyber range was the queue. This queue was implemented as a ring buffer using a double-linked list as the data structure and was mainly used to store the script objects. The objects were first created in the initialization stage and were passed into the queue after finishing initialization. Each object in the queue consisted of several main attributes as shown in the following list:

- Name of the current attacking script
- Reference/path to the current attacking script
- Type of the current attacking script
- Target of the current attacking script
- Time at which time the script will be executed

Every time the queue was called, the “while” loop inside the queue ran under the condition that as long as there was something inside the queue, it would keep cycling. Once an object was found, the queue would pass the object, along with all the attributes inside the object to the exploiter, where the script would be called and the attack would be executed.

3.6 Scripts

The fourth stage of the design of the cyber range was the scripts used by the red team for attacking. While the actual contents of the script had nothing to do with either the queue or the exploiter, it was important to introduce them in between the queue and the exploiter because they were being passed from the queue to the exploiter as objects, which were created based on their respective attributes.

There were various types of scripts that could be tested in the range. Depending on the attack being tested, example script types included port scanning, SQL injection, fingerprinting, denial-of-service, and others. They could be either written manually or obtained from free sources online. One important thing to note was that no matter what scripts were being tested in the range, they must be accessible by the red machine sending out the attack; additionally, each script must have a unique name, the reason for which is explained in the next stage. In this specific project, we stored the scripts locally on the red machines for a more convenient approach.

3.7 Exploiter

Last but not least was the final stage of the design, the exploiter. The main function of the exploiter was to correctly execute the attacking script onto the correct target based on the object passed in from the queue. Once the queue found at least one object, it would then pass the object to the exploiter. The exploiter would know which exact attacking script the object was based on because of the unique name of each script. After the script was successfully executed, the exploiter generated a certain response and went back to the queue, where the next object was sent out. The cycle continued until every single object in the queue was sent out to the exploiter for execution, thus accomplishing the goal of automation.

This concluded the implementation section of the cyber range. When everything was successfully implemented, the cyber range was ready to go and tests could be deployed.

Testing Scenario

After the implementation of the cyber range was finished, it was time to test it in various scenarios and see if things would turnout as expected. The testing scenario present here was an Nmap scan test. The blue machine being tested on was configured to have public exposed ports and vulnerable services, so that they should be discovered after the attack was deployed.

4.1 Nmap

The scenario that we were testing was to run the range and then automatically deploy an Nmap scan on a designated blue machine. In order to do this, there were a few preparatory tasks that needed to be done. These tasks should exist every time a unique test was being conducted; although such tasks might vary from test to test, it was important to complete them beforehand to ensure the success of the test.

1. Installation of Nmap on Red Machine

   Nmap is an open-source network scanner used to discover hosts and services on a network through packet traffic. It was used to discover any public exposed ports or services running on the target blue machine.
2. Installation of NodeJS on Red Machine

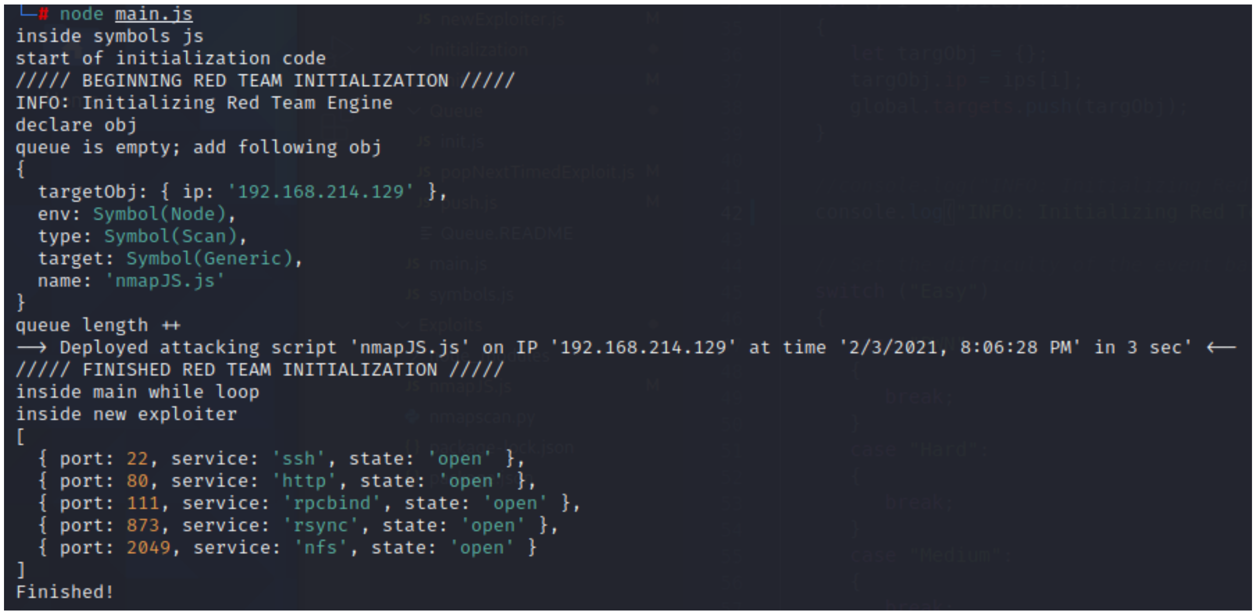
   Because the "Redteam Engine" was entirely written in NodeJS, it was necessary to install NodeJS on the red machines in order to run the engine.
3. A Written Nmap Script

After the installation of Nmap and NodeJS were finished on the red machine, the next step was to write the actual Nmap script. In this scenario, we used Python to write the Nmap script and then tested it to ensure that the script itself had no errors.

4. Installation of Python on Red Machine

   As stated above, because the Nmap script was written in Python, the red machine deploying the attack needed to have Python installed in order for the attacking script to run.

After all the preparatory tasks were finished, it was time to start the range and run the test. In order to start the range, a simple command was run and the result is shown below:



```
└─# node main.js
inside symbols js
start of initialization code
///// BEGINNING RED TEAM INITIALIZATION /////
INFO: Initializing Red Team Engine
declare obj
queue is empty; add following obj
{
  targetObj: { ip: '192.168.214.129' },
  env: Symbol(Node),
  type: Symbol(Scan),
  target: Symbol(Generic),
  name: 'nmapJS.js'
}
queue length ++
──→ Deployed attacking script 'nmapJS.js' on IP '192.168.214.129' at time '2/3/2021, 8:06:28 PM' in 3 sec' ←──
///// FINISHED RED TEAM INITIALIZATION /////
inside main while loop
inside new exploiter
[
  { port: 22, service: 'ssh', state: 'open' },
  { port: 80, service: 'http', state: 'open' },
  { port: 111, service: 'rpcbind', state: 'open' },
  { port: 873, service: 'rsync', state: 'open' },
  { port: 2049, service: 'nfs', state: 'open' }
]
Finished!
```

Figure 3: Starting the range and deploying a Nmap script onto a vulnerable blue machine

As shown in the image above, the flow of the range was displayed. The initialization code was the first to be executed. After that, an object was added to the queue with all the necessary attributes. Next, the Python script was deployed on the blue machine with the corresponding address. When the attack finished, certain responses were sent back. In this particular scenario, it was shown that the blue machines had several public exposed ports and services; after obtaining this information, we could then deploy further attacks aimed at such services and ports. Right now, the architecture that we designed did not support building up attacks based on each other, and each of the attacks were individual by itself. But for future development, we can have the architecture get results from a test like this and then automatically deploy other attacks that are relevant.

Impacts

5.1 Potential Improvements / Changes

Looking back at the progress made so far in this project, there were a few potential improvements and changes that would have made the project more efficient and optimized.

1. Remote Container for Storing Attacking Scripts

   When designing the project at the beginning, we originally had the thought of placing all the attacking scripts for the red team in a remote container. But in the actual implementation, we chose to store them locally on each individual red machine that was used to attack. Even though this was more convenient because there was only one machine on the red team during testing, it would not remain so in a cyber range with a larger scale. If a cyber range were to have ten machines on the red team, it would be a hassle to manually store the attacking scripts in each one of them. Therefore, to store all the script in a remote container and have each red machine pull from it would be much more efficient.

2. Designated Machine for the Redteam Engine

   Redteam Engine was designed in a way that every machine on the red team had to be installed with it. While there were ways to automate the installation of the engine onto every machine through tools like Terraform, we could still further improve the overall efficiency by having a designated machine specifically for the engine. For example, if a red machine were to deploy a XSS attack onto a blue machine, it would just need to call the engine stored on another machine in the same network and let that engine deploy the attack instead.

3. Difficulty Levels

   An idea that originally came up during the design of the cyber range was to give each attacking script a difficulty level. This was to give the tester the ability to choose a difficulty level at the beginning of the initialization stage and only the scripts with the matching difficulty level would be initialized into the queue. Due to time constraints and technical difficulty, the functionality was not implemented but it would be a great improvement to add to the project for more customizability.

5.2 Future Research

Since this project is still ongoing and will likely continue its development in the future, there are a few research routes that can be sought after to further extend the functionalities of the cyber range.

1. Reinforcement Learning for Cyber Security

   As one of the rising topics in the field of cyber security, reinforcement learning for cyber security is the idea of using machine learning to dynamically create defending mechanisms that are responsive, adaptive, and scalable. One of the current challenges for reinforcement learning in cyber security is how to efficiently train and test such algorithms in a safe environment. Due to its nature, a cyber range would be the perfect test bed for reinforcement learning. One method would be incorporating reinforcement learning into a blue machine and then feed it attacking scripts constantly to enforce machine learning. This will potentially lead to autonomous defense from the blue machine, which will further automate both the attacking and defending side of the range.

2. Cyber Security Education / Training

   As stated in the introduction, security education and training are mostly classroom-based right now. Although hands-on training experiences such as CTF competitions do exist and are proven to be more effective, most of the current methods of teaching and training are still through lectures and seminars. In order to provide more hands-on learning opportunities, cyber ranges would be a great place to do so. Because of tools like Terraform, the creation and teardown of a cyber range can be automated and students can freely play inside a range without worrying about potential security risks.

5.3 Ethical Impacts

Ethics has always been a large factor in the cyber security field. Since a large amount of the applications and services in the cyber security field directly relates to ethical topics, such as personal privacy and online integrity, potential ethics impacts should never be ignored. In the case of a cyber range, there are several ethical impacts that should be considered when practicing such technologies.

First of all, it is important to note that while most cyber ranges used virtual machines for hosts, these machines were still being hosted by companies that offer the cloud service providing the virtual machines, such as AWS in our case. It is critical to keep in mind to not direct any of the attacking tests at the hosting companies, purposely or by accident. This would result in devastating consequences and should be avoided at all times.

Secondly, certain high-risk attacks, such as distributed denial-of-service attacks, should only be practiced inside a secured cyber range and should never be practiced in a real-life situation. Large-scale attacks like DDOS are dangerous and should only be tested in a safe environment like a cyber range.

Conclusion

In conclusion, cyber range is a relatively new technology that has emerged in the recent decade. While its design and implementation are still being researched and experimented by many, the potential of such technology is limited. The advancement it can bring to the security industry and education should not be under-estimated. In this paper, a design and implementation of a cyber range were conducted and a test scenario of it was successfully ran. Many potential improvements and changes still exist, and future research and applications of a cyber range is looking bright as ever.

Reference

B. Ferguson, A. Tall and D. Olsen, *National Cyber Range Overview*, 2014 IEEE Military Communications Conference, Baltimore, MD, USA, 2014, pp. 123-128, doi: 10.1109/MILCOM.2014.27.

Cuong Pham, Dat Tang, Ken-ichi Chinen, and Razvan Beuran. 2016. *CyRIS: a cyber range instantiation system for facilitating security training*. In Proceedings of the Seventh Symposium on Information and Communication Technology. Association for Computing Machinery, New York, NY, USA, 251–258. DOI:https://doi.org/10.1145/3011077.3011087

Cashell, Brian, et al. *The economic impact of cyber-attacks. Congressional research service documents, CRS RL32331 (Washington DC)* 2 (2004).

A. Conklin, *Cyber Defense Competitions and Information Security Education: An Active Learning Solution for a Capstone Course*, Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), Kauai, HI, USA, 2006, pp. 220b-220b, doi: 10.1109/HICSS.2006.110.

L. McDaniel, E. Talvi and B. Hay, *Capture the Flag as Cyber Security Introduction*, 2016 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, USA, 2016, pp. 5479-5486, doi: 10.1109/HICSS.2016.677.