



**ÇANKAYA UNIVERSITY FACULTY OF  
ENGINEERING  
COMPUTER ENGINEERING DEPARTMENT**

Project Report

CENG 407

Innovative System Design and Development I

**Story Driven, Platform and Hardware  
Independent Cyber Security Training  
Environment**

Prepared by:

Sina Barış AYDIN – 201411007

Altuğ BOZKURT – 201311010

Utku TORAMAN – 201411061

8/12/2017

# Table of Contents

1.	<a href="#">Literature Review</a>	4
1.1	<a href="#">Penetration Testing Definition</a>	4
1.1.1	<a href="#">Penetration Testing Laboratories</a>	4
1.1.2	<a href="#">Known and Commonly Used Attacks</a>	4
2.	<a href="#">Software Requirements Specification</a>	6
2.1	<a href="#">Introduction</a>	6
2.1.1	<a href="#">Purpose</a>	6
2.1.2	<a href="#">Intended Audience and Reading Suggestions</a>	6
2.1.3	<a href="#">Product Scope</a>	6
2.2.	<a href="#">Overall Description</a>	7
2.2.1	<a href="#">Product Perspective</a>	7
2.2.2	<a href="#">User Classes and Characteristics</a>	7
2.2.3	<a href="#">Operating Environment</a>	7
2.2.4	<a href="#">Assumptions and Dependencies</a>	7
2.3.	<a href="#">External Interface Requirements</a>	7
2.3.1	<a href="#">User Interface Requirements</a>	7
2.3.2	<a href="#">Hardware Interfaces</a>	7
2.3.3	<a href="#">Software Interfaces</a>	7
2.4.	<a href="#">System Features</a>	8
2.4.1	<a href="#">Configuration of vulnerabilities and exploit path of Computer 1</a>	8
2.4.1.1	<a href="#">Brief Description</a>	8
2.4.1.2	<a href="#">Vulnerability and Configuration</a>	8
2.4.1.3	<a href="#">Exploitation Path</a>	8
2.4.2	<a href="#">Configuration of vulnerabilities and exploit path of Computer 2</a>	8
2.4.2.1	<a href="#">Brief Description</a>	8
2.4.2.2	<a href="#">Vulnerability and Configuration</a>	9
2.4.2.3	<a href="#">Exploitation Path</a>	9
2.4.3.1	<a href="#">Brief Description</a>	9
2.4.3.2	<a href="#">Vulnerability and Configuration</a>	9
2.4.3.3	<a href="#">Exploitation Path</a>	9
2.4.4.1	<a href="#">Brief Description</a>	10
2.4.4.2	<a href="#">Vulnerability and Configuration</a>	10

2.4.4.3	<a href="#">Exploitation Path</a>	10
2.5.	<a href="#">Other Nonfunctional Requirements</a>	10
2.5.1	<a href="#">Performance Requirements</a>	10
3.	<a href="#">Software Design Description</a>	11
3.1.	<a href="#">Introduction</a>	11
3.1.1	<a href="#">Purpose</a>	11
3.1.2	<a href="#">Scope</a>	11
3.1.3	<a href="#">Overview</a>	11
3.2.	<a href="#">SYSTEM OVERVIEW</a>	11
3.2.1	<a href="#">System Design</a>	11
3.3.	<a href="#">SYSTEM ARCHITECTURE</a>	12
3.3.1	<a href="#">Architectural Design</a>	12
3.3.1.1	<a href="#">Configuration of vulnerabilities and exploit path of P1</a>	12
3.3.1.2	<a href="#">Configuration of vulnerabilities and exploit path of P2</a>	14
3.3.1.3	<a href="#">Configuration of vulnerabilities and exploit path of Piv2</a>	15
3.3.1.4	<a href="#">Configuration of vulnerabilities and exploit path of Piv1</a>	16
3.4.	<a href="#">HUMAN INTERFACE DESIGN</a>	17
3.4.1	<a href="#">Overview of User Interface</a>	17
4.	<a href="#">Conclusion</a>	17
	<a href="#">Appendix A: Glossary</a>	18
	<a href="#">Appendix B: To Be Determined List</a>	20
	<a href="#">References</a>	21

List of figures:

Figure 1: User connection and system network architecture visualization. Figure 2: User action flow-diagram for P1.

Figure 3: User action flow-diagram for P2. Figure 4: User action flow-diagram for Piv2. Figure 5: User action flow-diagram for Piv1.

Figure 6: Possible screenshot for exploit discovery.

# 1. Literature Review

## 1.1 Penetration Testing Definition

A penetration test is an authorized controlled attack on a computer system that aims to find security weaknesses, and ways to access data. An effective penetration test almost always involves one or more skilled hackers, who will attempt to gain access to your systems starting only with what a regular, unauthorized attacker might have. The level of starting access given to the testers and the scope of the test may vary from simply attacking a web application, to getting access to the office building by disguising yourself as maintenance workers, to breaking in during the night. In many parts of the world, transactions of governments and banks are being done via internet and all the data is transferred to online servers. In this situation, providing security personnel with real experience in dealing with intruders, uncovering aspects of security policy that are lacking, providing feedback on the most at risk routes into your system, training developers to make fewer mistakes are crucial and highlights the significance of cyber security [1].

### 1.1.1 Penetration Testing Laboratories

We have decided to create a platform where aspiring testers can practice and improve their skills. Some tools that provide similar online services already exist, such as VulnHub, Pentestit, Hack The Box and OverTheWire [2]. VulnHub allows you to download images of virtual machines that are designed to be vulnerable in order to try to get into the machine and elevate your privileges to the highest level possible. The other two host a virtual private networks that have various machines with different levels of difficulties with the goal of trying to breach the network and breaking into the machines. Hack The Box and OverTheWire also have specifically configured virtual machines that can be accessed via SSH connections. Our goal is to build and improve upon these examples by building a product that utilizes story lines to make it more fun and engaging, is developed using LXC for hardware and OS compatibility, and is community driven.

### 1.1.2 Known and Commonly Used Attacks

Our lab will consist of 6 machines and will be divided into two sub networks. Each machine will require different sets of skills and will pose differing levels of difficulties. Some of the exploits that will be required to break into the machines can be broken into 3 main category; web-based exploits, daemon misconfigurations and privilege escalation attacks. Some of the popular web-based attacks are SQL injection [3][4][5], Cross-Site Scripting (XSS)[6][7], Local File Inclusion (LFI) and Remote File Inclusion (RFI). The main reason why most of these attacks exists in the wild is the lack of proper sanitization. Developers assume that user input would most likely be as intended, thus they don't handle cases of unexpected user input properly. When it's unintentional, abnormal input strings usually lead to the application crashing, but when that string is properly crafted, it can lead to a compromise of the webserver. SQL injections are a type of attack where the user can issue SQL queries and leak sensitive data from the database. XSS is another attack vector that only affects the end user. This attack allows the attacker to inject their own JavaScript code into a vulnerable page, which can then be used for phishing attacks. These are the most common attacks we can observe in wild. The second type of attack is dependent on the daemons/services running on the server. Many services run on servers for many different purposes. Some enable you to interact with the server remotely, some for file transfer and some for network management in an enterprise. Let's consider a DNS protocol, which is a very simple and crucial protocol for translating domain names into IP addresses. There are name servers all around the world that keep tracks of these records. There are many different record types for different

purposes, for example, DNS keeps mail servers as "MX" records while "A" record simply specifies an IP address for a normal host. DNS is a hierarchical system, each level in hierarchy can be provided with another server with different ownership. There are masters and slaves in this system. Slaves ask the master for a copy of records for a particular part which is called zone and the whole process is called zone transfer [8]. It is the mechanism masters use to update its DNS data. If it's not properly configured, an attacker can pretend to be a slave and ask for this information. This is actually a huge security hole since DNS data the attacker receives can be used to decipher the topology of a company network. This is just one of the examples for daemon misconfigurations and what it could lead to. Lastly, privilege escalation attacks usually come into play after the server is compromised and unauthorized remote access is gained. When you get the initial shell, it is likely to be a limited shell since the external service you have compromised is configured to have minimal privileges on the system. Thus you will have to escalate your privileges to gain full control over the machine. For this, you need to enumerate the system and identify the possible attack surfaces for that purpose. For example, you might encounter an executable running with higher privileges and has SUID bit set. SUID bits enables lower privilege users to execute files without having to be the owner of the file. This file might be vulnerable to buffer overflow which upon exploitation gives you the ability to execute commands with higher privileges. There are many other types of attacks with different attack surfaces but there are also many hardening methods.

## 2. Software Requirements Specification

### 2.1 Introduction

#### 2.1.1 Purpose

Today, cyber security is an internationally important issue. Therefore, many countries concentrated on training cyber security professionals and to encourage people for this training, there are hackathons and capture the flag (CTF) competitions being organized. These competitions run on servers which are specifically configured for this purpose. Also virtual machines can be used for this purpose. This project aims to create a cross-platform environment to learn and practice penetration testing skills with captivating storylines while taking community feedback every step of the way to increase usability of the machines and enhance the education stage.

#### 2.1.2 Intended Audience and Reading Suggestions

Since this project is about intentionally configuring computers or applications running on them with vulnerabilities, functional requirements of this project are explained in a different way than using use case, class or functional diagrams. This is because the product itself is not a software, it is a whole computer system. Instead of that diagrams, vulnerabilities and predicted exploitation paths are clearly explained to help configuration methods to be understood well.

In the system features part, vulnerabilities and exploitation paths of four connected computers in the training environment is explained in four parts [TBD: 1]. These parts includes step by step explanation of vulnerability for developer and exploitation path that user need to follow to access next computer in the network.

#### 2.1.3 Product Scope

Cyber security is a domain which is constantly renewed and has many subdomains like cryptology, forensics, reverse engineering, mobile and networking. In order to be an expert in cyber security, all latest topics need to be followed and all of its subdomains must be fully

known. Therefore, it is crucial to start learning from early. Besides, vulnerabilities are very diverse and they are spread on a very wide platform, some of them depends on the operating system, some of them depends on an older version of the program so it is difficult to create a training simulation environment in a single platform for penetration testing learners.

One way to create these training environments is to set-up pre-configured real servers but this is expensive, another way is to create virtual machines and host them and one other way is to use LXC containers. Target of the project is to create this training environment as LXC containers for making it easier for developers to configure computers and making it cheaper than real servers and easy to use for users because containers are hardware and operating system independent.

Also, creating a realistic network within the containers like in real world networks and amalgamating this pre-configured vulnerable on purpose computers with a storyline builds up more motivation to foster people for learning cyber security and increase training efficiency. Last, deliberately chosen vulnerabilities will be from all possible subdomains to show beginners what cyber security is like.

## 2.2. Overall Description

### 2.2.1 Product Perspective

Until today, penetration testing labs has already existed. Innovation in this project is using LXC containers instead of real servers or virtual machines. This project will consist of pre-configured LXC containers to simulate a network with two computers with different vulnerabilities. First, machines with vulnerabilities will be configured on LXC containers. Later, users need to install a VPN client on their machine to access to our live server with story driven virtual penetration testing laboratory and start their training from first computer in that network. Every computer has its own scenario to gather information about the vulnerability to proceed and connect to the next computer in the network.

### 2.2.2 User Classes and Characteristics

There are two user classes in this project which are standard user and admin. Anyone who has installed a VPN client can download our container and use this training environment but this project is especially done for people who want to train in cyber security so in order to proceed in his training, the user of our lab needs to be experienced about exploits and must have a base network knowledge at least. Also, admin have to know entire topic for every machine in order to configure them. Since, users will be working on pre-configured machines there will be no interaction between admin and standard user.

### 2.2.3 Operating Environment

Since lab to be created is a LXC container, it is operating system and hardware independent.

### 2.2.4 Assumptions and Dependencies

Only dependency is Docker software.

## 2.3. External Interface Requirements

### 2.3.1 User Interface Requirements

The user can install Docker on Linux, Windows or MacOS but whichever operating system s/he will be using main user interface will be command line interface (CLI) of that operating system. There is no graphical user interface.

### 2.3.2 Hardware Interfaces

To use CLI, a monitor and a keyboard is required beside the PC to use its CLI. Also, 2.5GB amount of memory and internet connection is required to download and run our container.

### 2.3.3 Software Interfaces

VPN client need to be installed in order to gain access to our lab.

## 2.4. System Features

The lab will consist of 2 sub networks that are connected to each other through only one machine in the lab. In order to do that, the user will have to pivot through that machine to access the other sub net. This can be done using IPtables that basically allows you to set network rules for any machine in the network [TBD: 2].

### 2.4.1 Configuration of vulnerabilities and exploit path of Computer 1

#### 2.4.1.1 Brief Description

Admin must configure a web site to be hosted on this computer with a SQLi vulnerability and create a buffer overflow vulnerable code. User must find and exploit this vulnerability to gain access to host of this website and then find the buffer overflow vulnerability and exploit it to access root privilege shell. After user has root privileges then he can find necessary information related to the storyline and gather IP number of the next computer in the network.

#### 2.4.1.2 Vulnerability and Configuration

- 1) To host the website, mysql, php and apache2 services must be downloaded and installed to the hosted Linux (Ubuntu) container.
- 2) Linux Apache MySQL PHP (LAMP) must be configured to run integrated with each other.
- 3) Hosted website moved to web-root directory (for apache2 the path is: /var/www/html).
- 4) Intentionally, user inputs coming from the query done in Login.php will not be sanitized and used directly in the query that is used in back-end to enable SQLi login bypass vulnerability.
- 5) A new rule added to IP tables to serve SSHDaemon on IPv6.
- 6) A buffer-overflow vulnerable C code shall be written and compiled. After compilation created executable file, suid and x bits are set to this file. The file then is placed between suid binaries in the system. Last, C source code of that executable is deleted.

#### 2.4.1.3 Exploitation Path

- 1) User enumerates the machine to find running services.
- 2) As a result of enumeration user finds hosted website.
- 3) User enumerates website and reaches to login page of the website.
- 4) User injects SQL queries with username and password strings to check if there is any SQLi vulnerability.
- 5) Then user gains unauthorized access by SQLi login bypass.
- 6) User finds SSH private key in the website.
- 7) User finds that SSH daemon is hosted on IPv6 and connects to SSH port on IPv6 with found key.
- 8) After user gains initial shell, s/he does port enumeration process and finds the buffer overflow vulnerable root privileged executable binary file.
- 9) User does reverse engineering on suid binary to exploit the buffer overflow vulnerability and gains access to root privileged shell.

After gaining this shell then user can find necessary information in that computer to connect to next computer.

### 2.4.2 Configuration of vulnerabilities and exploit path of Computer 2

#### 2.4.2.1 Brief Description

User need to anonymously connect to FTP server and exploit a buffer overflow vulnerability hosted on the remote service and exploits a cronjob to gain root access. After gaining root access user gets a hint about port knocking for connecting to 3rd computer.



#### 2.4.2.2 Vulnerability and Configuration

- 1) Buffer-overflow vulnerable C code is written and hosted as a remote service on the server.
- 2) FTP server is set-up. FTP configuration file must be edited to allow anonymous login access and saved.
- 3) A cronjob is set to run a bash script in a specific directory for once in a minute.

#### 2.4.2.3 Exploitation Path

- 1) User enumerates the server to find running services.
- 2) User finds out that FTP port is open.
- 3) User connects to FTP port and tries to login anonymously.
- 4) After User logs in successfully, downloads suspicious C source code and bash script from the server.
- 5) User finds out that source code to be serving remotely on the server.
- 6) User connects to necessary service remotely and checks the functionality.
- 7) User finds buffer-overflow vulnerability in that source code.
- 8) User exploits that vulnerability and gains initial shell.
- 9) With post-enumeration result user finds a cronjob which executes every minute.
- 10) User finds that cronjob executes a bash script under a specific directory.
- 11) User modifies that file and next execution of that cronjob will spawn a new shell for the user.
- 12) To spawn that new shell, user must set up a listener on the server and necessary cronjob should be moved in. Then user catches the reverse shell that spawned with root privileges and gains root access.
- 13) User finds a note including a hint for 3rd computer.

### 2.4.3 Configuration of vulnerabilities and exploit path of Computer 3

#### 2.4.3.1 Brief Description

Admin configures port knocking service and downloads and installs a text editor with a version which is vulnerable to shell escape attack. User must gain access to server with port knocking and after gaining an initial shell he must use shell escape method to spawn a shell and gain root access.

#### 2.4.3.2 Vulnerability and Configuration

- 1) Admin downloads and configures port knocking service.
- 2) Admin configures the LAMP server must to run integrated with each other.
- 3) Files that are going to be hosted on the website are moved into web-root directory.
- 4) For allowing remote code execution, input sanitization will not be done on websites functionality back-end.
- 5) A shell escape attack vulnerable version of text editor must be installed by admin.

#### 2.4.3.3 Exploitation Path

- 1) User enumerates the computer and finds the services running services on it.
- 2) User need to use the hint given on previous machine to implement port knocking.
- 3) After necessary port knocking process is done, user enumerates server again and finds a hosted website on the server.
- 4) User enumerates the site and finds a functionality is vulnerable to remote command execution
- 5) User exploits this vulnerability to gain reverse shell from the website.
- 6) User does post enumeration after gaining initial shell.

- 7) With post enumeration results user finds out that s/he can run some high privilege commands without sudo password.
- 8) The command user can run provides editing a specific file with a text editor. After running that command and opening the text editor, user tries to spawn a shell with shell escape method.
- 9) User gains root access after these processes.

## 2.4.4 Configuration of vulnerabilities and exploit path of Computer 4

### 2.4.3.4 Brief Description

Admin intentionally misconfigures the NFS Share and user uses this misconfiguration to access to server. Then user finds a bash script with root privilege and gains root access with it.

### 2.4.3.5 Vulnerability and Configuration

- 1) Admin setups SSH Service.
- 2) Admin installs and configures RPC-Bind Port Mapper service.
- 3) Admin prepares NFS Shares and does necessary configurations on them.
- 4) Admin creates a simple bash script.

### 2.4.3.6 Exploitation Path

- 1) User enumerates the computer and finds running services.
- 2) After enumeration user finds out that a RPC-Bind (Port Mapper) service is being served.
- 3) User enumerates the service and observes publicly available NFS-Shares.
- 4) User downloads these Shares and enumerates in his/her own computer.
- 5) In order the user to enumerate these files, the permissions must be same with the owner of the Share.
- 6) After user done necessary settings, NFS-Share can be examined in user's computer.
- 7) User places his/her created SSH key with required arrangements and connects to server with SSH.
- 8) User does post enumeration after gaining initial shell and explores a bash script with root permissions.
- 9) After user observes that script, s/he examines that it is a single line simple bash command.
- 10) User creates a file with exact same name of that used bash command and writes shell spawning bash command to that file and saves.
- 11) User makes that file executable and directory of that file is added to PATH variable. In this way, when this root owned script executed, instead of that bash command user created executable file will run with root permissions and user will gain a shell with root permissions.

## 2.5. Other Nonfunctional Requirements

### 2.5.1 Performance Requirements

Since running LXC container use the resources of a PC like a virtual machine, before converting the configured machines into containers, lightweight operating systems and applications must be used.

## 3.1. Introduction

### 3.1.1 Purpose

This software design document describes the architecture and system design of Story Driven, Platform and Hardware Independent Cyber Security Training Environment. This project aims to create a cross-platform environment to learn and practice penetration testing skills with engaging storylines, while taking community feedback every step of the way to increase usability of the machines and increase the educational value. The intended audience is people who want to know what real penetration testing is like, novices who want to improve and experts looking for fun challenges.

### 3.1.2 Scope

The way we have chosen to create this training is environment is by arranging a network of intentionally misconfigured computers using LXC and LXD containers hosted on the cloud. This provides the following benefits:

- easy for developers to configure computers
- no specific hardware or operating system requirements
- scalability

In addition, creating a realistic network within the containers mimicking real world networks and amalgamating these pre-configured vulnerable-on-purpose computers with a storyline builds up more motivation for people to learn cyber security and increase training efficiency. Lastly, deliberately chosen vulnerabilities will be from all possible subdomains to show beginners what cyber security is like.

### 3.1.3 Overview

Following part of the document includes system overview and system architecture in order. There is no data design section in the document because there is not any data structure or database implemented in the system.

## 3.2. SYSTEM OVERVIEW

### 3.2.1 System Design

This system is not a single software, it is a set of configured and networked operating systems and there are not any uncertainties expected. Thus, we will follow a waterfall lifecycle model for this project. First, we will configure each machine alone then containerize them and do the necessary network connection configurations between them. For each machine, configuration will be complete in two weeks and after all machines configured, network configuration will be complete in the following two weeks. Last, testing will take two weeks according to the plan.

## 3.3. SYSTEM ARCHITECTURE

### 3.3.1 Architectural Design

The lab will consist of two sub networks that are connected to each other through only one machine in the lab. In order to do that, the user will have to pivot through that machine to access the other sub net.

(Public Network: It will be dual homed and Network Address Translated to the interface that reaches out to the internet.) (Pivot Network: Private network that can only be accessed through public

network.)

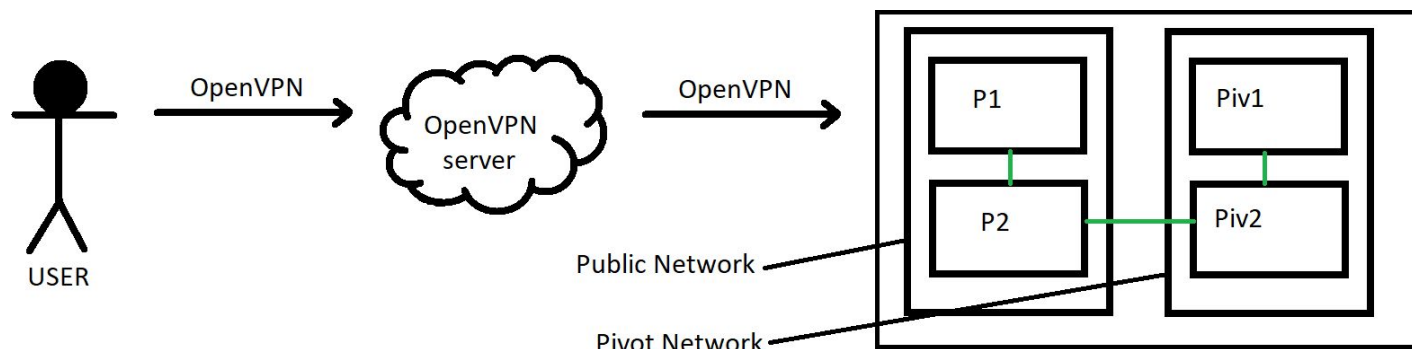


Figure 1: User connection and system network architecture visualization.

#### 3.3.1.1 Configuration of vulnerabilities and exploit path of P1

**Brief Description** Admin must configure a web site to be hosted on this computer with a SQLi vulnerability and create a buffer overflow vulnerable code. User must find and exploit this vulnerability to gain access to host of this website and then find the buffer overflow vulnerability and exploit it to access root privilege shell. After user has root privileges then he can find necessary

information related to the storyline and gather IP number of the next computer in the network.  
Vulnerability and Configuration

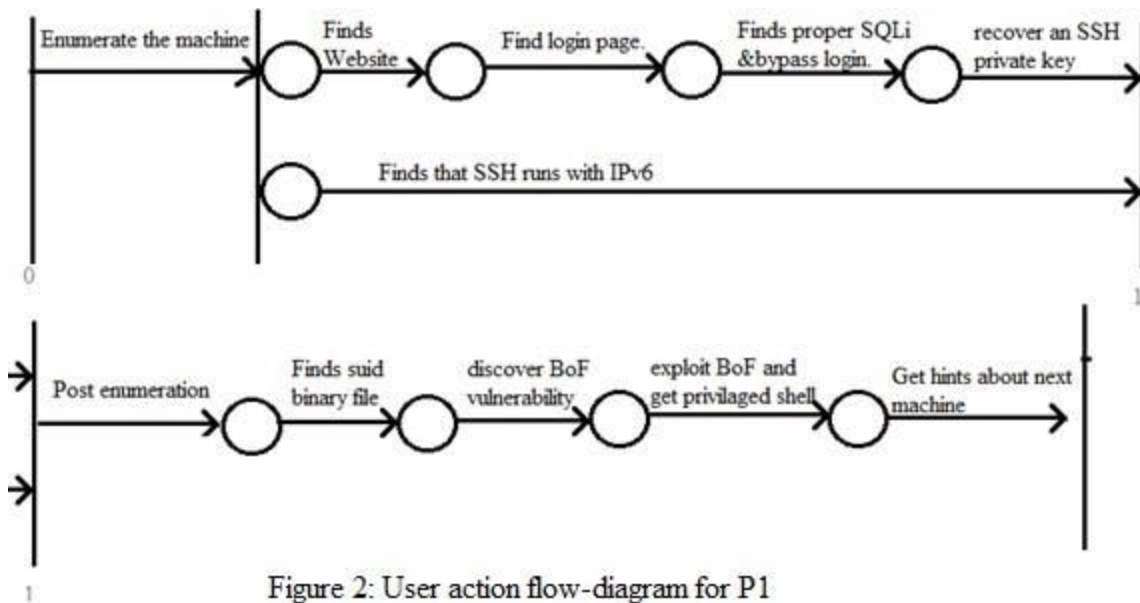


Figure 2: User action flow-diagram for P1

1. To host the website mysql, php and apache2 services must be downloaded and installed to the hosted Linux (Ubuntu) container.
2. Linux Apache MySQL PHP (LAMP) must be configured to run integrated with each other.
3. Hosted website moved to web-root directory (for apache2 the path is: /var/www/html).
4. Intentionally, user inputs coming from the query done in Login.php will not be sanitized and used directly in the query that is used in back-end to enable SQLi login bypass vulnerability.
5. A new rule added to IP tables to serve SSHDaemon on IPv6.
6. A buffer-overflow vulnerable C code shall be written and compiled. After compilation created executable file, suid and x bits are set to this file. The file then is placed between suid binaries in the system. Last, C source code of that executable is deleted.

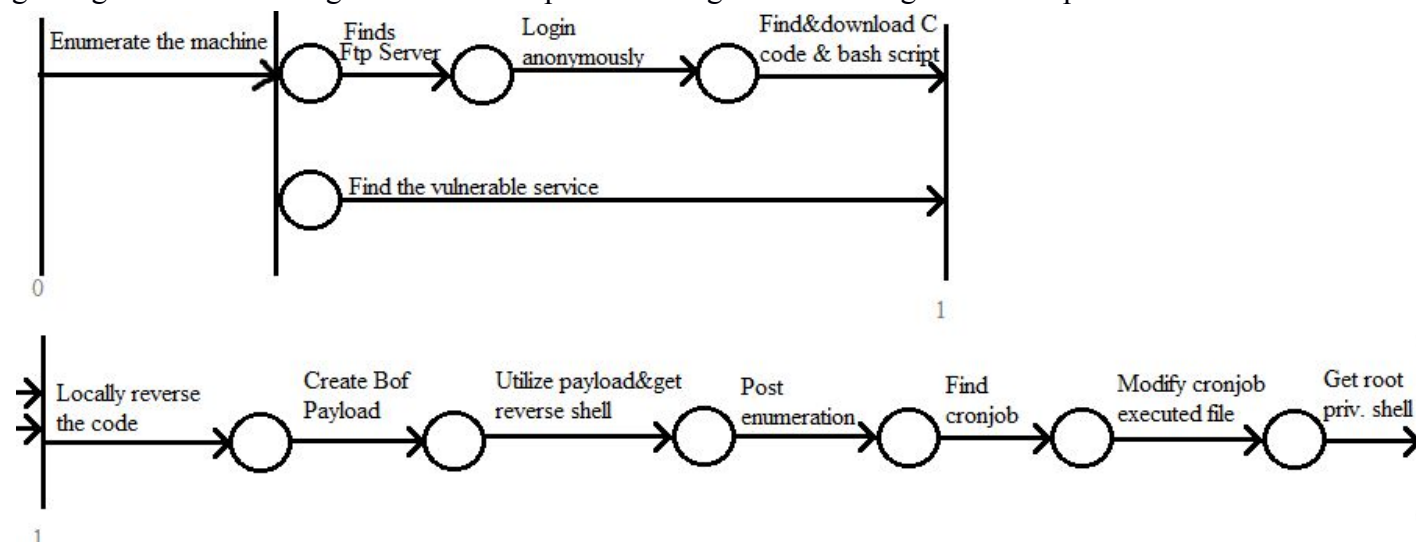
## Exploitation Path

1. User enumerates the machine to find running services.
2. As a result of enumeration user finds hosted website.
3. User enumerates website and reaches to login page of the website.
4. User injects SQL queries with username and password strings to check if there is any SQLi vulnerability.
5. Then user gains unauthorized access by SQLi login bypass.
6. User finds SSH private key in the website.
7. User finds that SSH daemon is hosted on IPv6 and connects to SSH port on IPv6 with found key.
8. After user gains initial shell, s/he does port enumeration process and finds the buffer overflow vulnerable root privileged executable binary file.

9. User does reverse engineering on suid binary to exploit the buffer overflow vulnerability and gains access to root privileged shell. After gaining this shell then user can find necessary information in that computer to connect to next computer.

### 3.3.1.2 Configuration of vulnerabilities and exploit path of P2

Brief Description User need to anonymously connect to FTP server and exploit a buffer overflow vulnerability hosted on the remote service and exploits a cronjob to gain root access. After gaining root access user gets a hint about port knocking for connecting to 3rd computer.



**Figure 3: User action flow-diagram for P2 Vulnerability and Configuration**

1. Buffer-overflow vulnerable C code is written and hosted as a remote service on the server.
2. FTP server is set-up. FTP configuration file must be edited to allow anonymous login access and saved.
3. A cronjob is set to run a bash script in a specific directory for once in a minute.

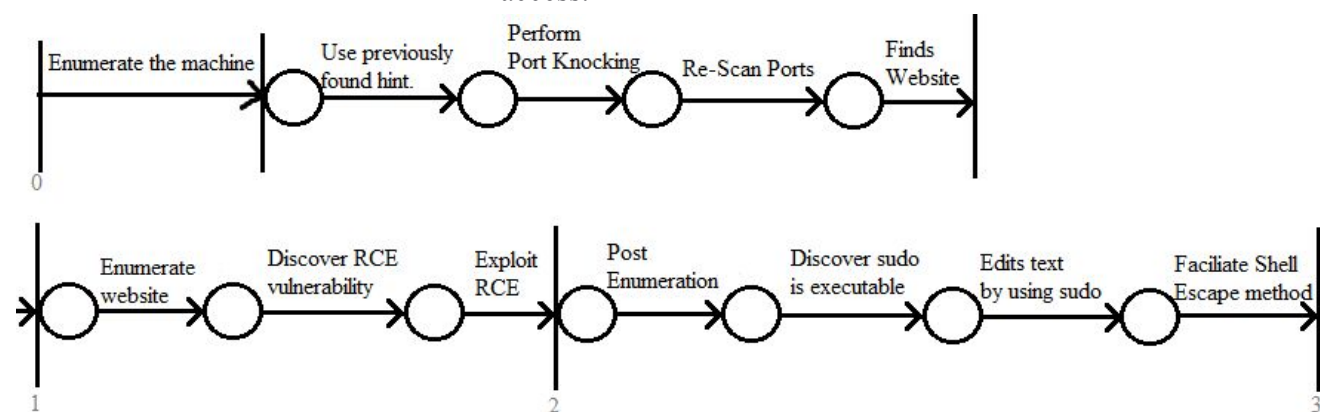
#### Exploitation Path

4. User enumerates the server to find running services.
5. User finds out that FTP port is open.
6. User connects to FTP port and tries to login anonymously.
7. After User logs in successfully, downloads suspicious C source code and bash script from the server.
8. User finds out that source code to be serving remotely on the server.
9. User connects to necessary service remotely and checks the functionality.
10. User finds buffer-overflow vulnerability in that source code.
11. User exploits that vulnerability and gains initial shell.
12. With post-enumeration result user finds a cronjob which executes every minute.
13. User finds that cronjob executes a bash script under a specific directory.
14. User modifies that file and next execution of that cronjob will spawn a new shell for the user.

15. To spawn that new shell, user must set up a listener on the server and necessary cronjob should be moved in. Then user catches the reverse shell that spawned with root privileges and gains root access.
16. User finds a note including a hint for 3rd computer.

### 3.3.1.3 Configuration of vulnerabilities and exploit path of Piv2

**Brief Description** Admin configures port knocking service and downloads and installs a text editor with a version which is vulnerable to shell escape attack. User must gain access to server with port knocking and after gaining an initial shell he must use shell escape method to spawn a shell and gain root access.



**Figure 4: User action flow-diagram for Piv2 Vulnerability and Configuration**

1. Admin downloads and configures port knocking service.
2. Admin configures the LAMP server must to run integrated with each other.
3. Files that are going to be hosted on the website are moved into web-root directory.
4. For allowing remote code execution, input sanitization will not be done on websites functionality back-end.
5. A shell escape attack vulnerable version of text editor must be installed by admin.

#### Exploitation Path

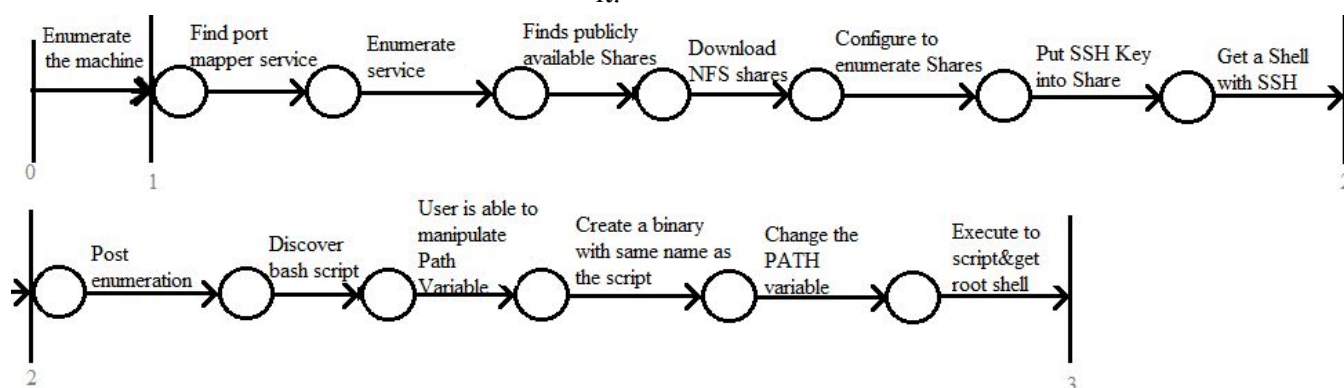
6. User enumerates the computer and finds the services running services on it.
7. User need to use the hint given on previous machine to implement port knocking.
8. After necessary port knocking process is done, user enumerates server again and finds a hosted website on the server.
9. User enumerates the site and finds a functionality is vulnerable to remote command execution
10. User exploits this vulnerability to gain reverse shell from the website.
11. User does post enumeration after gaining initial shell.
12. With post enumeration results user finds out that s/he can run some high privilege commands without sudo password.
13. The command user can run provides editing a specific file with a text editor.
14. After running that command and opening the text editor, user tries to spawn a shell with shell escape method.



15. User gains root access after these processes.

### 3.3.1.4 Configuration of vulnerabilities and exploit path of Piv1

**Brief Description** Admin intentionally misconfigures the NFS Share and user uses this misconfiguration to access the server. Then user finds a bash script with root privilege and gains root access with it.



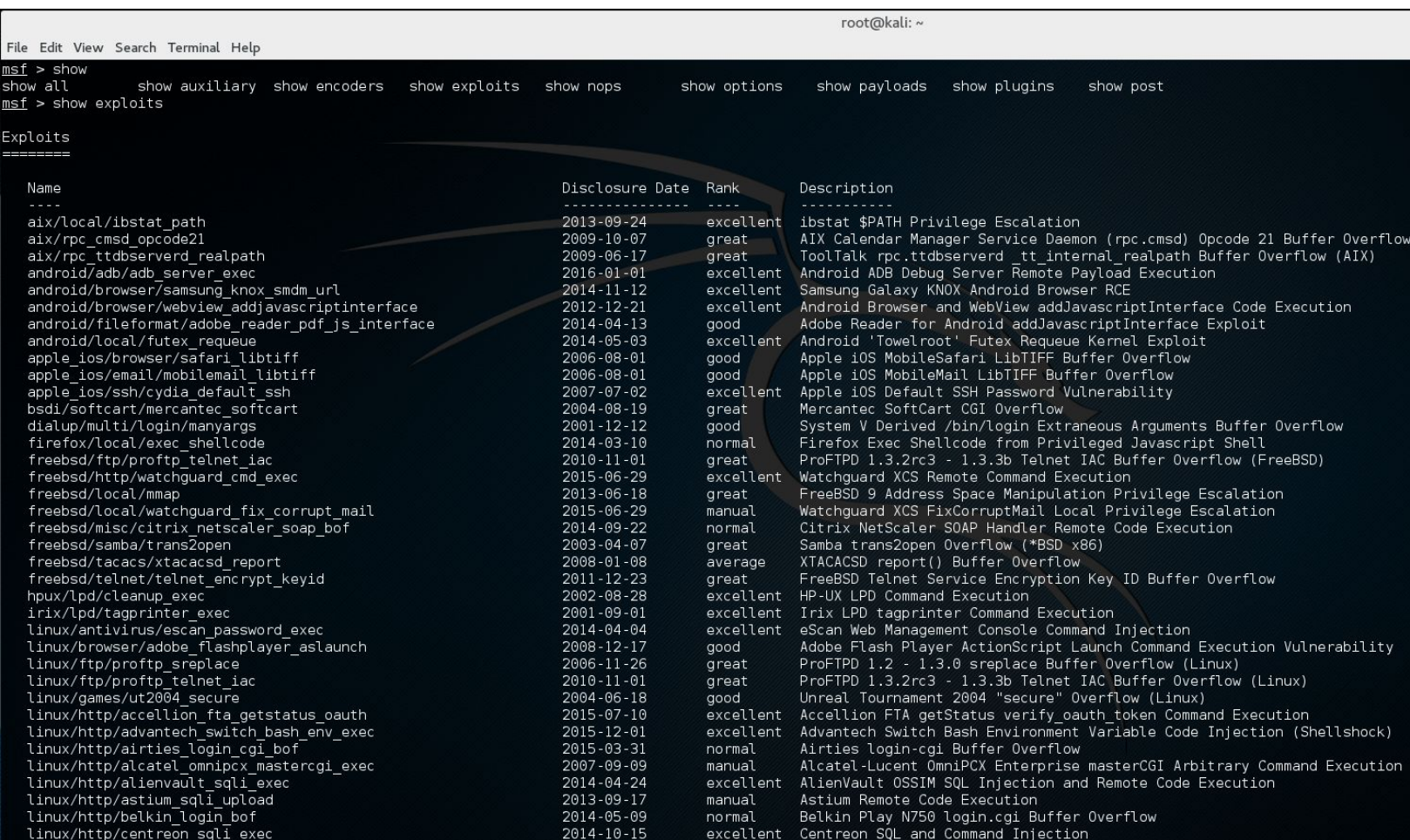
**Figure 5: User action flow-diagram for Piv1 Vulnerability and Configuration**

1. Admin setups SSH Service.
2. Admin installs and configures RPC-Bind Port Mapper service.
3. Admin prepares NFS Shares and does necessary configurations on them.
4. Admin creates a simple bash script. **Exploitation Path**
5. User enumerates the computer and finds running services.
6. After enumeration user finds out that a RPC-Bind (Port Mapper) service is being served.
7. User enumerates the service and observes publicly available NFS-Shares.
8. User downloads these Shares and enumerates in his/her own computer.
9. In order the user to enumerate these files, the permissions must be same with the owner of the Share.
10. After user done necessary settings, NFS-Share can be examined in user's computer.
11. User places his/her created SSH key with required arrangements and connects to server with SSH.
12. User does post enumeration after gaining initial shell and explores a bash script with root permissions.
13. After user observes that script, s/he examines that it is a single line simple bash command.
14. User creates a file with exact same name of that used bash command and writes shell spawning bash command to that file and saves.
15. User makes that file executable and directory of that file is added to PATH variable. In this way, when this root owned script executed, instead of that bash command user rated executable file will run with root permissions and user will gain a shell with root permissions.

## 3.4. HUMAN INTERFACE DESIGN

### 3.4.1 Overview of User Interface

All inputs and outputs will be in text format so User will only use Command Line Interface.



Exploits			
Name	Disclosure Date	Rank	Description
-----	-----	----	-----
aix/local/ibstat_path	2013-09-24	excellent	ibstat \$PATH Privilege Escalation
aix/rpc_cmsd_opcode21	2009-10-07	great	AIX Calendar Manager Service Daemon (rpc.cmsd) Opcode 21 Buffer Overflow
aix/rpc_ttdbserverd_realpath	2009-06-17	great	ToolTalk rpc.ttdbserverd tt_internal_realpath Buffer Overflow (AIX)
android/adb/adb_server_exec	2016-01-01	excellent	Android ADB Debug Server Remote Payload Execution
android/browser/samsung_knox_smdm_url	2014-11-12	excellent	Samsung Galaxy KNOX Android Browser RCE
android/browser/webview_addjavascriptinterface	2012-12-21	excellent	Android Browser and WebView addJavaScriptInterface Code Execution
android/fileformat/adobe_reader_pdf_js_interface	2014-04-13	good	Adobe Reader for Android addJavaScriptInterface Exploit
android/local/futex_requeue	2014-05-03	excellent	Android 'Towelroot' Futex Requeue Kernel Exploit
apple_ios/browser/safari_libtiff	2006-08-01	good	Apple iOS MobileSafari LibTIFF Buffer Overflow
apple_ios/email/mobilemail_libtiff	2006-08-01	good	Apple iOS MobileMail LibTIFF Buffer Overflow
apple_ios/ssh/cydia_default_ssh	2007-07-02	excellent	Apple iOS Default SSH Password Vulnerability
bsdi/softcart/mercantec_softcart	2004-08-19	great	Mercantec SoftCart CGI Overflow
dialup/multi/login/manyargs	2001-12-12	good	System V Derived /bin/login Extraneous Arguments Buffer Overflow
firefox/local/exec_shellcode	2014-03-10	normal	Firefox Exec Shellcode from Privileged Javascript Shell
freebsd/ftp/proftp_telnet_iac	2010-11-01	great	ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (FreeBSD)
freebsd/http/watchguard_cmd_exec	2015-06-29	excellent	Watchguard XCS Remote Command Execution
freebsd/local/mmap	2013-06-18	great	FreeBSD 9 Address Space Manipulation Privilege Escalation
freebsd/local/watchguard_fix_corrupt_mail	2015-06-29	manual	Watchguard XCS FixCorruptMail Local Privilege Escalation
freebsd/misc/citrix_netscaler_soap_bof	2014-09-22	normal	Citrix NetScaler SOAP Handler Remote Code Execution
freebsd/samba/trans2open	2003-04-07	great	Samba trans2open Overflow (*BSD x86)
freebsd/tacacs/xtacacs_report	2008-01-08	average	XTACACSD report() Buffer Overflow
freebsd/telnet/telnet_encrypt_keyid	2011-12-23	great	FreeBSD Telnet Service Encryption Key ID Buffer Overflow
hpux/lpd/cleanup_exec	2002-08-28	excellent	HP-UX LPD Command Execution
irix/lpd/tagprinter_exec	2001-09-01	excellent	Irix LPD tagprinter Command Execution
linux/antivirus/escan_password_exec	2014-04-04	excellent	eScan Web Management Console Command Injection
linux/browser/adobe_flashplayer_aslaunch	2008-12-17	good	Adobe Flash Player ActionScript Launch Command Execution Vulnerability
linux/ftp/proftp_sreplace	2006-11-26	great	ProFTPD 1.2 - 1.3.0 sreplace Buffer Overflow (Linux)
linux/ftp/proftp_telnet_iac	2010-11-01	great	ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (Linux)
linux/games/ut2004_secure	2004-06-18	good	Unreal Tournament 2004 "secure" Overflow (Linux)
linux/http/accellion_fta_getstatus_oauth	2015-07-10	excellent	Accellion FTA getStatus verify.oauth.token Command Execution
linux/http/advantech_switch_bash_env_exec	2015-12-01	excellent	Advantech Switch Bash Environment Variable Code Injection (Shellshock)
linux/http/airties_login.cgi_bof	2015-03-31	normal	Airties login.cgi Buffer Overflow
linux/http/alcatel_omnipcx_mastercgi_exec	2007-09-09	manual	Alcatel-Lucent OmniPCX Enterprise masterCGI Arbitrary Command Execution
linux/http/alienvault_sqli_exec	2014-04-24	excellent	AlienVault OSSIM SQL Injection and Remote Code Execution
linux/http/astium_sqli_upload	2013-09-17	manual	Astium Remote Code Execution
linux/http/belkin_login_bof	2014-05-09	normal	Belkin Play N750 login.cgi Buffer Overflow
linux/http/centreon_sqli_exec	2014-10-15	excellent	Centreon SQL and Command Injection

Figure 6: Possible screenshot for exploit discovery

## 4. Conclusion

As a result of the work we've put into this project so far, we've learned a significant amount about the current state of cyber security and penetration testing. We've been continuously researching about current exploits out there and how to create a penetration testing lab that really resembles something one would encounter in real life. Personally participating in similar penetration testing labs and talking with others in the field gave us ideas about what features a successful product would need to have. We've also learned a great deal about container technologies such as Docker, LXC and LXD; how they work, how to network them and how they compare against Virtual Machines. Writing Literature Reviews, Software Requirements Specifications and Software Design Documents helped guide us through the IEEE standard of Software Development.

## Appendix A: Glossary

**Admin:** A person responsible for running technically advanced information systems. Short for Administrator.

**Bash:** Bash is a Unix Shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell.

**Buffer-overflow:** Buffer-overflow is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations.

**Capture The Flag:** Is a special kind of information security competitions. There are three common types of CTFs: Jeopardy, Attack-Defence and mixed.

**Cron-job:** The software utility cron is a time-based job scheduler in Unix-like computer operating systems.

**Corss-Platform:** Able to be used on different types of computers or with different software packages.

**Cryptology:** The study of codes, or the art of writing and solving them.

**Cyber Security:** Cybersecurity is the body of technologies, processes and practices designed to protect networks, computers, programs and data from attack, damage or unauthorized access. In a computing context, security includes both cybersecurity and physical security.

**Daemon:** In multitasking computer operating systems, a daemon is a computer program that runs as a background process, rather than being under the direct control of an interactive user.

**Docker Container:** A standardized unit of software. Package software into standardized units for development, shipment and deployment.

**Docker Hub:** Docker Hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.

**Docker:** Docker is the company driving the container movement and the only container platform provider to address every application across the hybrid cloud.

**Enumeration:** Is a complete, ordered listing of all the items in a collection.

**Exploit:** A software tool designed to take advantage of a flaw in a computer system, typically for malicious purposes such as installing malware.

**Forensic:** Scientific tests or techniques used in connection with the detection of crime.

**File Transfer Protocol (FTP) :** The File Transfer Protocol (FTP) is the standard network protocol used for the transfer of computer files between a client and server on a computer network. **Hackathon:** An event, typically lasting several days, in which a large number of people meet to engage in collaborative computer programming.

**Hardware:** Computer hardware is the physical parts or components of a computer, such as the monitor, keyboard, computer data storage, graphic card, sound card and motherboard.

**Initial Shell:** First shell user gets to execute arbitrary commands on the system.

**Lightweight System:** The term lightweight is sometimes applied to a program, protocol, device, or anything that is relatively simpler or faster or that has fewer parts than something else

**Network:** A group or system of interconnected computers.

**Network File System (NFS):** The Network File System (NFS) is a client/server application that lets a computer user view and optionally store and update files on a remote computer as though they were on the user's own computer.

**Operating System:** The low-level software that supports a computer's basic functions, such as scheduling tasks and controlling peripherals.



**Penetration Testing:** A penetration test, colloquially known as a pen test, is an authorized simulated attack on a computer system, performed to evaluate the security of the system.

**Port Knocking:** Port knocking is a method of externally opening ports on a firewall by generating a connection attempt on a set of prespecified closed ports.

**Port:** A port is an endpoint of communication in an operating system.

**Privilege:** A special right, advantage, or immunity granted or available only to a particular person or group.

**Remote Code Execution:** Remote code execution is the ability an attacker has to access someone else's computing device and make changes, no matter where the device is geographically located. **Reverse Engineering:** The reproduction of another manufacturer's product following detailed examination of its construction or composition.

**Root Privilege (Root Permission):** A permission level that grants access to every possible authorization on computer.

**RPC-Bind:** is a close analog of BIND service.

**Sanitization:** Process made to prevent code injection problems.

**Server:** A server is a computer program that provides services to other computer programs (and their users) in the same or other computers.

**Shell Escape Attack:** Escape and stringify an array of arguments to be executed on the shell.

**Shell Script:** A shell script is a computer program designed to be run by the Unix shell, a command-line interpreter.

**Shell Spawn:** Calling a shell to communicate with kernel directly.

**Shell:** A Unix shell is a command-line interpreter or shell that provides a traditional Unix-like command line user interface.

**SQL Injection Attack (SQLi):** SQL Injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements (also commonly referred to as a malicious payload) that control a web application's database server (also commonly referred to as a Relational Database Management System – RDBMS).

**SQL Query:** A query is an inquiry into the database using the SELECT statement

**SSH Key:** SSH keys serve as a means of identifying yourself to an SSH server using public-key cryptography and challenge-response authentication.

**Secure Shell (SSH):** SSH, also known as Secure Socket Shell, is a network protocol that provides administrators with a secure way to access a remote computer.

**Standard User:** User with restricted permissions.

**Sub Network:** A part of a larger network such as the Internet.

**Suid Bit: SUID (Set owner User ID up on execution)** is a special type of file permissions given to a file.

**Virtual Machine:** A virtual machine (VM) is an operating system (OS) or application environment that is installed on software, which imitates dedicated hardware.

**Vulnerability:** A vulnerability is a weakness which allows an attacker to reduce a system's information assurance.

## Appendix B: Installation Guide

This training environment runs on an Ubuntu Server which is open for VPN connection.

1. First, user needs to have a VPN client to establish a connection.
2. We prefer OpenVPN which is a free software and available for multiple platforms (<https://openvpn.net/>).
3. After installing OpenVPN, user must get the file named "connection.config" from our website. [(link to config file)](<https://pentestingcontainer.wordpress.com/>)
4. Next, user must enter and run these code lines below into terminal in order: (Windows users can run same command on PowerShell)
5. ``sudo openvpn connection.config``
6. ``sudo dhclient tap0``

Now user should be connected to the first container of the training environment and be able to use the training environment without a problem.

## Appendix C: User Manual

Important Note: This training environment is not for beginners. In order to advance you need to have a basic knowledge of attacks and exploits. You can improve your skills from other environments like OverTheWire War Games.

Since this environment runs on an Ubuntu server without a Graphical User Interface there is only Command Line Interface to navigate so knowing basic Linux commands is crucial. You need to find the hidden file to finish this training. You will find hints as you proceed.

## References

- [1] E. Byres, J.Low, "The Myths and Facts behind Cyber Security Risks for Industrial Control Systems", p. 1-3
- [2] <https://www.hackthebox.eu/> , <https://www.vulnhub.com/> , <http://overthewire.org/wargames/> Available: 13.11.2017
- [3] [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection) Available: 13.11.2017
- [4] C. Sharma, Dr.S.C. Jain, "SQL Injection Attacks on Web Applications" Page 12-68, Volume:4, Issue 3,March 2014
- [5] K. Natarajan, S. Subramani, "Generation of SQL-injection free secure algorithm to detect and prevent SQL-injection attacks ", Procedia Technology 4 ( 2012 ) 790 – 796
- [6] S. Gupta, B.B.Gupta, "Smart XSS Attack Surveillance System for OSN in Virtualized Intelligence Network of Nodes of Fog Computing", International Journal of Web Services Research, 2017, p.1-3
- [7] E. Kirda, N. Jovanovic, C. Kruegel, and G. Vigna, "Client-side cross-site scripting protection," Computers & Security, vol. 28, no. 7, pp. 592–604, 2009.
- [8] <https://www.sans.org/reading-room/whitepapers/dns/securing-dns-zone-transfer-868>
- [9] <https://www.docker.com/what-docker>