Software Design Document
For
Story Driven, Platform and Hardware Independent Cyber Security Training Environment

Çankaya University Faculty of Engineering
Computer Engineering Department

Prepared by:
Sina Barış AYDIN – 201411007
Altuğ BOZKURT – 201311010
Utku TORAMAN – 201411061
18/05/2018

Table of Contents

List of Figures

1. INTRODUCTION

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.

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.

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.

2. SYSTEM OVERVIEW

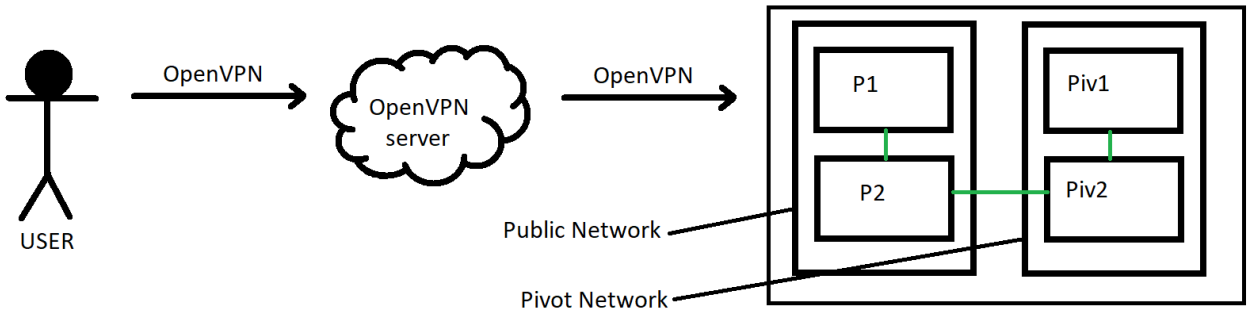
2.1 System Design

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. SYSTEM ARCHITECTURE

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.

3.1.1 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.

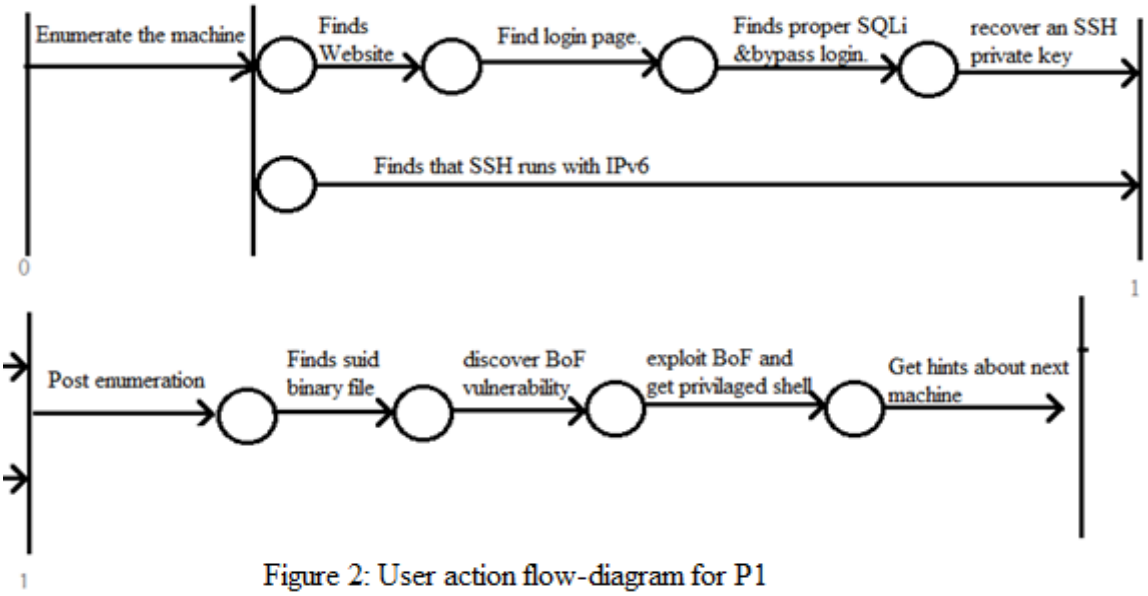


Figure 2: User action flow-diagram for P1

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.

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.

3.1.2 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.

1

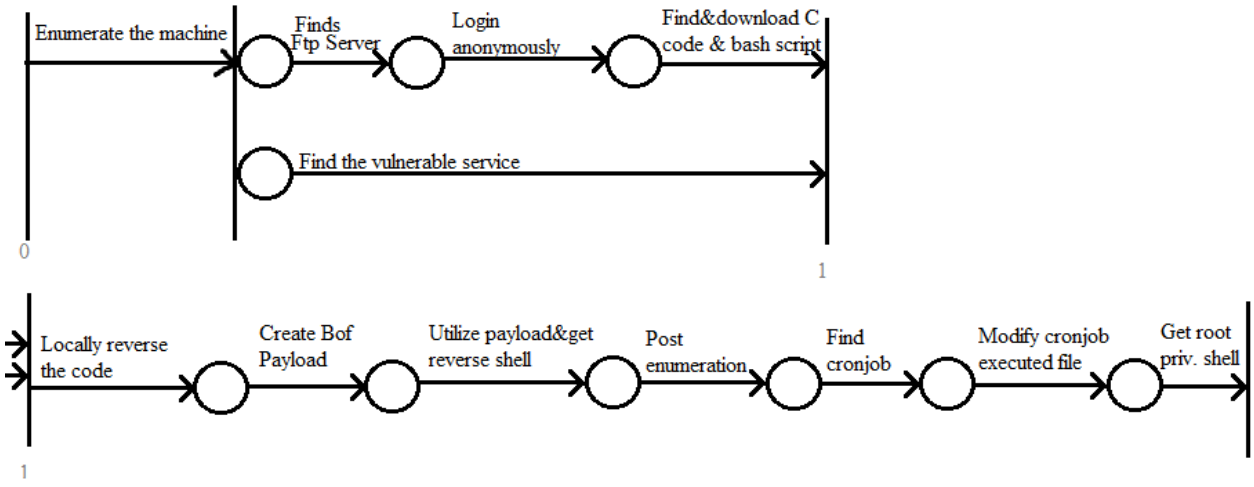


Figure 3: User action flow-diagrom for P2

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

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 logins succesfuly, 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.

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.

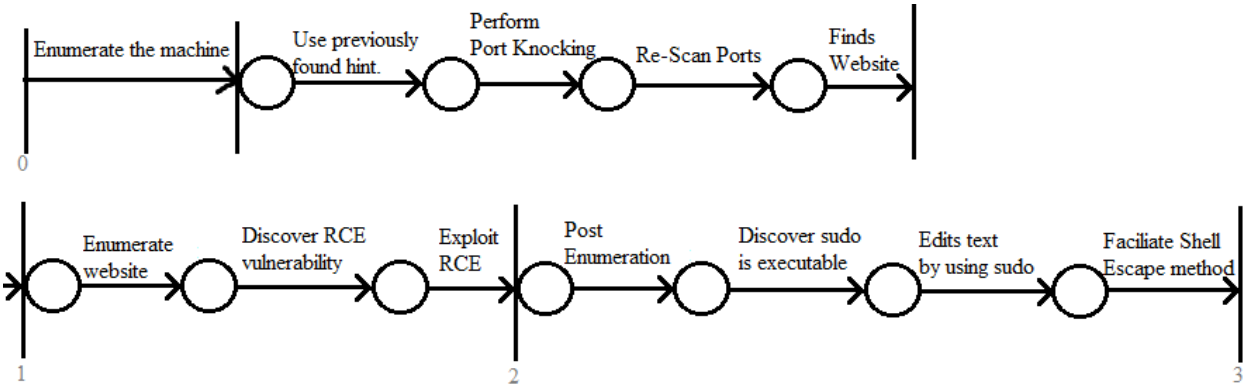


Figure 4: User action flow-diagram for Piv2

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

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.
9. After running that command and opening the text editor, user tries to spawn a shell with shell escape method.
10. User gains root access after these processes.

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.

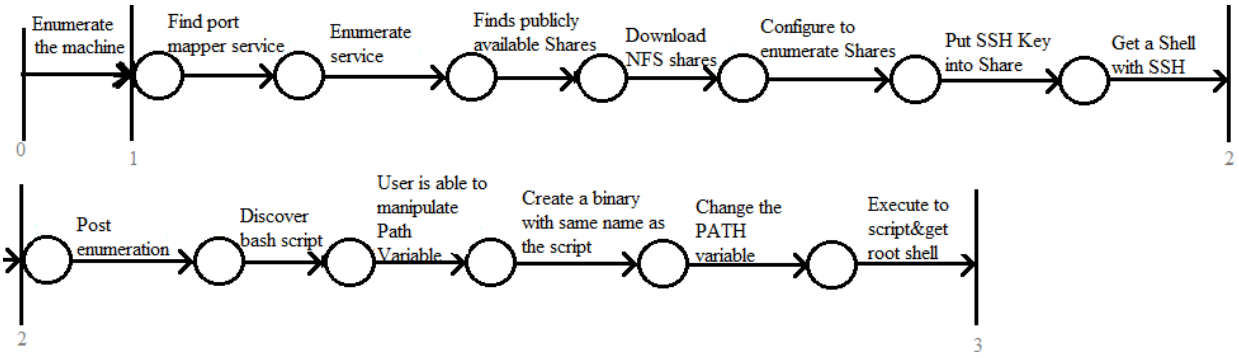


Figure 5: User action flow-diagram for Piv1

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

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.

4. HUMAN INTERFACE DESIGN

4.1 Overview of User Interface

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

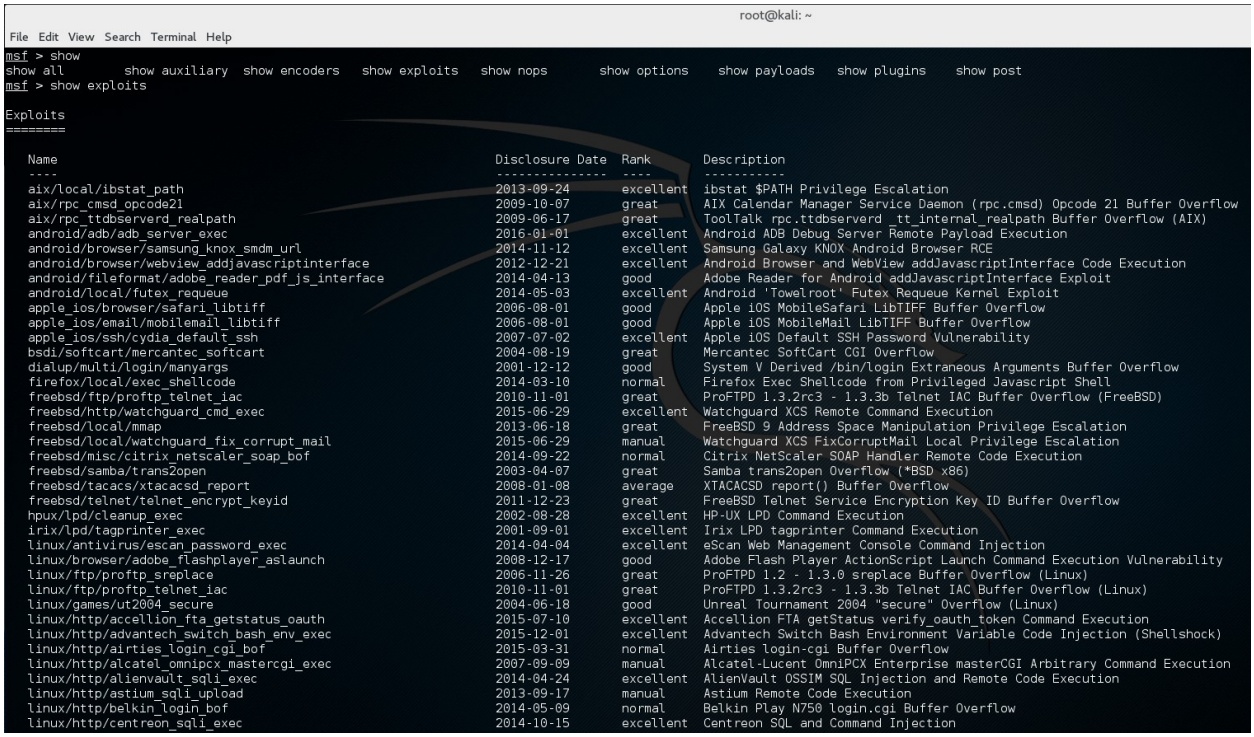


Figure 6: Possible screenshot for exploit discovery

5. Installation Guide & User Manual

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. (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.

User Manual

Important Note: This training environment is not for begginers. 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(http://overthewire.org/wargames/bandit/)

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.

Appendix B: 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.
Cross-Platform: Able to be used on different types of computers or with different software packages.
Cryptograpgy: 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 cyber security 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.
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: A port is an endpoint of communication in an operating 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.

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