Introduction to Offensive Security Proposal

For the term project, I have decided to go forward with Red and exploit all the attacks as mentioned in the vulnhub website. For this project, first I will discuss what exactly is red.

Red is a Vulnhub machine that I created in which your machine was completely taken over by Red and you, Blue, are trying to regain control. It starts by finding a unusual Local File Inclusion (LFI) backdoor on a WordPress site, which leads us to find some credentials. The credentials by themselves don't work but using a password mutation technique with Hashcat, we are able to gain access to the machine. However, Red is doing whatever it takes to defend his takeover and he throws a couple of things your way in order to stop you.

The red can be easily accessed from the website: https://www.vulnhub.com

This website contains a lot of vulnerable machines that are vulnerable by design (red for instance) and we need to develop (multiple) exploits to become the root. This is going to a intermediate difficulty project. I will be using Kali Linux for the implementation.

I will be performing the following tasks:

Task 1: Finding the IP and running Nmap Scan:

Finding the IP: Use tools like `ifconfig` or `ip a` to identify the IP address of the target machine.

Nmap Scan: Run an Nmap scan using the command `nmap <target_IP>` to identify open ports and services.

Task 2: Fixing site for a better view:

This task will likely involve improving the appearance or functionality of a website. It could include fixing broken links, optimizing images, or enhancing the user interface.

Task 3: Finding LFI hint:

LFI (Local File Inclusion) vulnerabilities allow an attacker to include files on a server through the web browser. Look for parameters in URLs that might be susceptible to LFI.

Task 4: Running Gobuster to find backdoor:

Use Gobuster to perform directory/file brute-forcing. For example, `gobuster dir -u http://<target_IP> -w /usr/share/wordlists/dirb/common.txt`.

Task 5: Using WFUZZ to find correct parameter:

WFUZZ is a web application security fuzzer. Use it to identify the correct parameters in URLs. For instance, `wfuzz -c -z file,/path/to/wordlist.txt --u http://<target_IP>/FUZZ`.

Task 6: Parameter found, LFI test:

Once you've identified the vulnerable parameter, attempt LFI attacks to include files and retrieve sensitive information.

Task 7: Enumerating users, looking for WordPress credentials

Enumerate users on the target system. Look for files or configurations related to WordPress, as these might contain credentials.

Task 8: Using PHP wrappers to read PHP files and finding credentials

- PHP wrappers can be used to include remote files. Experiment with `php://input` or other wrappers to read PHP files and locate credentials.

Task 9: Reviewing Hashcat password mutation rules

- Review Hashcat rules to understand how it mutates passwords during brute-force attacks. Modify rules as needed.

Task 10: Using hydra to find password:

Hydra is a password-cracking tool. Use it to perform brute-force attacks on login forms or services to discover passwords.

Task 11: Checking Message from Red and Access as ippsec:

This task implies that there's a message from a user or system called "Red." Examine the message to gather information or instructions. Once done, access the system or user account named "ippsec."

Task 12: Fake flag, but we are not listening to Red:

This task suggests that there might be a misleading or false flag present. In hacking scenarios, a "flag" usually refers to a piece of information or a key that proves the success of a particular step. In this case, the instruction is not to focus on the information provided by "Red" and instead explore other avenues.

Task 13: Losing shell access and password changed:

This task indicates a security challenge where you lose your current access to the system, and the password is changed. You'll need to figure out how to regain access, possibly by exploiting another vulnerability or finding a new entry point.

Task 14: Avoiding getting kicked out of SSH and Finding and reviewing secret directory:

To prevent being kicked out of an SSH session, consider methods like using tools such as `tmux` or `screen` to create persistent sessions. Once secure, look for a secret directory that might contain sensitive information or additional vulnerabilities.

Task 15: SSH session ended, avoided getting kicked out:

Confirm that you've successfully maintained your SSH session without being forcibly disconnected.

Task 16: Using pspy to confirm root backdoor:

`pspy` is a tool that allows you to observe processes on a system without relying on traditional monitoring tools. Use `pspy` to detect any suspicious or privileged processes that might indicate the presence of a root backdoor. Investigate further to confirm and understand how it works.

Task 17: Building C reverse shell:

Create a C program that establishes a reverse shell. A reverse shell is a type of shell in which a target machine initiates a connection back to the attacker's machine, allowing the attacker to gain remote access. You'll need to compile the C program and execute it on the target system.

Task 18: Writing C reverse shell file and root shell returned:

Write the C reverse shell code, compile it, and run it on the target machine. This should result in obtaining a root shell, providing you with escalated privileges on the system.

Task 19: Upgrading Shell and Fixing machine:

Upgrade your shell for better interactivity and perform any necessary cleanup or remediation to fix vulnerabilities on the target system. Upgrade your existing shell to a more interactive one, such as a fully interactive TTY shell. This enhances your control over the target system. Additionally, identify and fix any vulnerabilities or misconfigurations on the target machine to ensure its security.