Shakabrah Proving Grounds

Penetration Test Report

Table of Contents

1.0-High-Level Summary………………………………………………………………………………………………………………………….3

1.1-Recommendations………………………………………………………………………………………………………………..3

2.0-Methodologies………………………………………………………………………………………………………………………………….5

2.1-Information Gathering………………………………………………………………………………………………………….5

2.2-Service Enumeration…………………………………………………………………………………………………………….5

2.3-Peneration Testing……………………………………………………………………………………………………………….5

**Service Vulnerable**…………………………………………………………………………………………………………8

2.4-Report…………………………………………………………………………………………………………………………………..9

1.0-High-Level Summary

An internal penetration test was performed on the potato network in the Offensive Security Proving Ground Labs. An internal test simulates an attacker that is directly connected into the network, in this case through a VPN tunnel.

The purpose of this test was to simulate an attack where the attacker had access to the network, with attempts made to break into a system and then elevate privileges on the machine.

Over-all, the intent was to enumerate the services on the exposed network, determine an attack vector to get access, and then exploit any flaw found within the system.

During the test, it was found that the php file on the website allowed for remote code injection, that allowed for the retrieval of local files. This included the /etc/passwd file, which allowed for discovering the local users that were allowed access to the device.

Shell code was injected into website that allowed for remote access into the system as the web user account. This allowed for access to all local level files and directories within the network.

Once within the system, two different vectors were found for exploiting privileges to escalated to the root user through an interactive shell.

The first was through a SUID binary that was exploited to gain root shell access. This allowed access to all root level files and directories on the system.

The second method was utilizing an exploit that performed a memory overflow attack against an outdated version of the SUDO command. Using this method, a root shell was spawned for total system control.

1.1-Recommendations

The website index.php file should be upgraded to avoid using common exploit commands that can be abused. Exec(), shell\_exec(), system(), and passthru() should be avoided for the lack of sanitizing in verifying the commands executed. Escapeshellcmd() is a good upgrade to use to prevent most of the common methods of injection. Custom coding to prevent the & or its URL encode of %26 is also a good method to add to a script to prevent standardization from being a weak point for future exploits.

Upgrading the system should help protect against the SUDO version exploit used, and is a good practice to stay ahead of zero-day exploits being discovered. Many times, exploits of older versions can be found through checking the patch notes of the more recent updates for services and programs.

Removing the SUID permissions from files that do not absolutely need the functionality is another good practice to implement. Many binaries can be exploited when given that setting to spawn root shells within the system. It is better to make a sudo call for needed functions to prevent their exploitation.

The Secure Shell remote access service should also be set up with brute force attack protection. There are various mechanisms on the pam.d process that can disable the service for the IP or user when too many failed logins are detected.

2.0-Methodologies

Below are the methods that were undertaken to break into the device, and ultimately achieve root access on the device.

2.1-Information Gathering

The information gathering portion was mostly null, as the network address of 192.168.191.86 was provided ahead of the pentest commencing.

2.2-Service Enumeration

This was mainly accomplished with nmap scan of the base 1000, followed by a scan of all tcp ports on the device. The UDP top 1000 ports were scanned for vulnerabilities, none looked available for an exploit. This left it with the following ports as possible exploit vectors:

22 Secure Shell Remote Access

80 HTTP web access

2.3-Penetration Testing

With only two services available for targeting, brute force with common usernames was utilized against Secure Shell, and various scripts were run against the web site service. There was no rate-limiting enabled on the Secure Shell service, so the program was run with both a 1k password list against multiple users, and then a 1.4m password list against the root user. Neither method returned a positive result by the time the penetration test was completed.

The enumeration of the web site returned a positive result for remote code injection, that returned the results of the /etc/passwd rendered in the response. This was verified by going to the web browser, and using the URL.

Graphical user interface, text, application

Description automatically generated

Various commands were tested against the system, including the ability to view the files and folders of any local privilege directory on the system. Using this method, it was also possible to find the contents of the home files of the users exposed through the passwd file. There were no sensitive ssh keys or password files discovered, but the contents of the local.txt was capable of being viewed.

Graphical user interface, text, application

Description automatically generated

Various shell codes were attempted through the injection, with a command successfully spawning a /bin/sh shell that was able to be upgraded to as bash shell. This shell provided complete access to all local privilege files and directories on the network.

Graphical user interface, text

Description automatically generated

Text

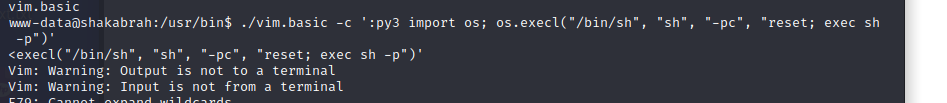
Description automatically generated

Checking for exploits found two methods for spawning a root shell on the system.

The first method utilized a SUID permission set on one of the binaries, that was able to be launched with root privilege.

Text

Description automatically generated



Once inside the process, it was able to be halted by spawning a shell through the process. This shell inherited the permissions of the SUID binary and led to root level access to the network. This represented a complete compromise of the network device.

Graphical user interface, text

Description automatically generated

Backing out of the shell, the second method was attempted. This exploit made use of an outdated SUDO version that allows for a buffer overflow attack to launch a shell with the privileges of root. The exploit was uploaded to the /tmp directory, and successfully launched to spawn a root shell on the target.

Text

Description automatically generated

**System Vulnerable 192.168.191.86**

**Vulnerabilities Exploited:**

Remote code injection through the website .php file

Out of date binary allowing for buffer overflow attack

Improper SUID set binary allowing for root shell access

**Severity: Critical**

**Proof of Privilege Escalation:**

Local.txt: cea7ac68e29c6dd3cb28dcb8a685d1a1

Proof.txt: 20e5b5d21574d2e504c051782038067f

2.4-Report: Clean-up

An exploit for privilege escalation was uploaded to the /tmp folder to spawn a root shell. After this was completed, the file was removed from the directory. Various vulnerability scanners were also uploaded to the /tmp directory. Following the output of the scripts, the files were removed from the /tmp folder. This returned the device to its original state from before the penetration test.