Ha-Natraj 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-Penetration Testing………………………………………………………………………………………………………………5

**Service Vulnerable**……………………………………………………………………………………………………….10

2.4-Report…………………………………………………………………………………………………………………………………10

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 website suffered from local file inclusion, which exposed local files to public viewing. Among these were the /etc/passwd file, which provided information on 2 of the local users. The other one of use was the Secure Shell remote access log, which allowed for a log poisoning attack to gain network access. This compromised all local level files within the system.

Checking the web user found that there were sudo permissions assigned that granted control of the Apache web server. The configuration file for the Apache web server also had incorrect permissions that allowed for write access to local users.

With the combination of write access to the config file and sudo rights for restarting the server, it was possible to change the user account owner of the website, and create shells for any of the local users on the system that were listed in the password file.

One of the users had sudo rights to the nmap program, which is an exploit vector for privilege escalation. Using these rights it was possible to spawn a root shell to the system. This represented a complete compromise of the network device, as all files and folders were available for download.

Checking for other vulnerabilities discovered a weakness in the sudo command from being an older version. Using the sudo bypass method, it was possible to spawn a root shell from the web user that compromised the network device.

1.1-Recommendations

It is recommended to isolate the web server directory, so it is not possible to use path traversal methods to gain access to sensitive local files. This will prevent both the disclosure of the users through the password file, as well as poisoning the authorization logs to gain shell entry into the system.

The web user account had root level access for starting and stopping the Apache web server. If this is necessary for business function, it can be left as is. Otherwise, the sudo privileges should be removed from the user account.

If they are necessary, then the write permissions for the config files for the Apache should be removed. This will prevent restarting the server to create shells to various users on the network.

The users should also have sudo permissions removed from their accounts that are not necessary. The user account has a nmap service that was sudo privileged that can be exploited in all versions available. There is no update that can fix this exploit; the permissions must be removed from the account to prevent attackers from being able to get root privilege on the account.

The sudo command will also need to be updated to a version past 1.9.5p1 to prevent exploits from being used to bypass authentication and spawning a root shell. This should be done as soon as possible, as there is limited expected impact from such an upgrade.

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 192.168.191.80 was provided ahead of the pentest commencing.

2.2-Service Enumeration

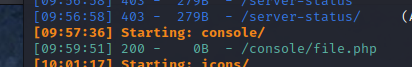
This was mainly accomplished with nmap scan of the base 1000, followed by a scan of all tcp ports on the device. The UDP top ports returned no open ports for the top 1000 ports. This left it with the following ports as possible exploit vectors:

22 Secure Shell Remote access

80 HTTP Web Server

2.3-Penetration Testing

With only two available services showing on the device, enumeration began on the directories for the service while manually checking the pages for any information disclosure. There was a console directory found early on, that looked to have a file to test for possible path traversal.



From there, a custom script was run to check for sql injection and path traversal, with positive results showing that the /etc/passwd file was available for viewing externally.

Text

Description automatically generated

This was verified by checking the website, which validated the local file inclusion vulnerability found by the script.

Text

Description automatically generated

This gained information regarding two of the users, along with the root account. SSH was brute forced for those users with the accounts using a common 1k password list. This did not succeed on any of the user accounts.

While the brute force was running, the website was tested to see what other files were able to be viewed externally. It was found that the Secure Shell authorization log was also available for viewing and shown to be done in real time as the brute force attempts were constantly updating on the log.

Text

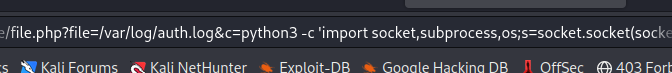
Description automatically generated

Once the users were verified being added into the log, an exploit was then uploaded by authenticating with a php command script that added command functionality to the website.

Text

Description automatically generated

From there, the command sequence was triggered by adding a command into the URL for a shell code to be executed to connect to the attacker computer.



The exploit was verified to work when a shell with the web account was spawned, giving local level access to all files and directories on the network device.

A screenshot of a computer

Description automatically generated with medium confidence

From there, the contents of the local.txt file were able to be viewed on the system.

Graphical user interface

Description automatically generated with low confidence

Further exploits for privilege escalation were checked for, and two avenues were found. The SUDO version is out of date and had multiple exploits available for its current version. The Baron Samedit exploit was used, and the exploit succeeded in gaining root access to the computer device. This represented a complete compromise of the system, that allowed full access to all files and directories.

Graphical user interface, text, application

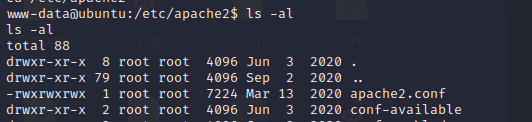
Description automatically generated

Backing out of the root shell, the sudo permissions of the web account were viewed for possible privilege escalations as well.

Text

Description automatically generated

To see if there was a possible way to exploit this, it was looked at for possible calls from the Apache configuration, or any permissions that could be abused. There did not appear to be any calls from the configuration that could be exploited, but the file had write permissions enabled on it.



This was moved to the /tmp folder and downloaded locally, and edited using the two local users on the system. The configuration was backed up and stored in /var/tmp, while the file was over-written with one of the users referenced as the service owner.

Graphical user interface, text, application

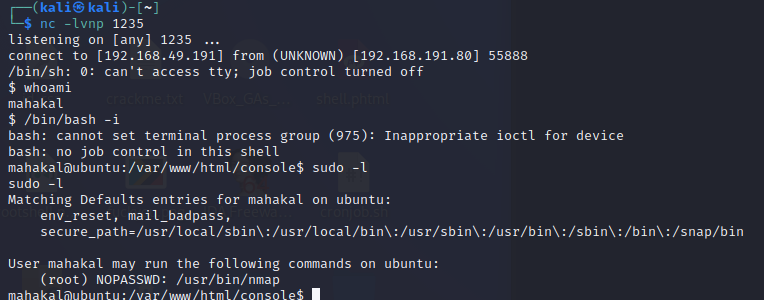
Description automatically generated

Using the restart function of the server allowed it to start up as the local user, with the shell generated giving access for that user.

Text

Description automatically generated

The first user checked had no sudo functions allowed to them, and no proprietary files found that could be utilized by them for privilege escalation. Changing the Apache configuration again to the other user led to finding a SUDO privilege allowed to them without password required.



Using the nmap vulnerability with sudo privilege, enabled a second method of spawning a root shell to the system.

A screenshot of a computer

Description automatically generated with medium confidence

**System Vulnerable 192.168.191.80:**

**Vulnerabilities Exploited:**

Local File Inclusion left sensitive files available for public viewing from external sources

Secure Shell logging being publicly available allowed for log poisoning to spawn a shell into the network

Improper write privileges allowed for changing the configuration of the server to different users

Outdated software allowed for known exploits to gain root level access on the system

SUDO privilege enabled binaries allowed root level shell to be gained

**Severity: Critical**

**Proof of Privilege Escalation:**

Local.txt: 56b56863da1492e410eab0969130de20

Proof.txt: 8e7c0176b70392c121d9a7f67864f8d5

2.4-Report: Clean-up

The Secure Shell authorization log contains a shell command from poisoning the log, that could not be removed without impacting all the log. This will need to be backed up for company records, and then emptied. During the testing, multiple vulnerability check programs were uploaded and then removed after getting the needed outputs. The SUDO exploit generated a temp file for shell access, that was removed after root verification.

The Apache configuration file was not able to be restored and will need local access admin to update the backup file to its proper place. This is due to an attempt to launch the server as root user, which caused the server to crash on restart and not come back up. The original configuration file is located in /var/tmp and can be restored with the following command:

Cp /var/tmp/apache2.conf /etc/apache2/apache2.conf