Infosecprep 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-Penetraion 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 testing, it was found that there was information disclosure on the external website that allowed for retrieving sensitive information for both the user and login credentials. With these, local access was gained to the network that compromised all local privilege files and directories on the device.

Two methods were then used to exploit vulnerabilities on the system that allowed a root privilege shell to be gained on the system. This allowed access to all files and folders on the network.

The first was with a known vulnerability that exploits an out-of-date SUDO version that will create a new shell session with the root user access.

The second was with a SUID setting on the bash binary that creates a persistent shell with root level access to the system.

Other vulnerabilities were found that were attempted to exploit, but due to configuration issues or the need to install a gcc compiler they were not able to be utilized. The vulnerabilities will need to be pathed, to prevent any changes or updates on the system rendering the exploits able to complete successfully.

1.1-Recommendations.

The website should be sanitized of any sensitive information from being publicly disclosed for any attackers enumerating the website. The robots.txt file disallow a directory that when checked showed there was a base64 encoded message that when decoded turned out to be the private key for Secure Shell. This allows for logging in to the network without the need for password authentication.

The username was found on the main page of the website, which coupled with the key allowed for access into the network. Removing both of these pieces of information will harden the network against most attacks from hackers.

The device should have its current configuration backed up with a restoral point created, and then upgraded to a more recent and stable version. There were many vulnerabilities on the device due to out-of-date binaries and programs that this upgrade will resolve.

The groups allowed for local level users should also exclude the lxd, adm, and sudo allowances to prevent exploitation to get root level access on the device. If these groups are necessary for business functions, it is recommended to have a separate higher privileged account created that can have its permissions used with the su command. This will prevent the compromise of a local level user from getting root access to various programs, while allowing for normal business functions to continue.

The password for the local user was not able to be broken into with a 1.4 million password list check but may be able to be cracked with rule base or larger password lists. A policy should be put in place to enforce a 15+ character passphrase to prevent easy brute force attacks against user accounts.

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 192.168.191.89 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 port only returned 10 ports open on the top 1000 ports, none looked available for an exploit. This left it with the following ports as possible exploit vectors:

22 Secure Shell Remote Access

80 Web Service

33060 SQL database

2.3-Penetration Testing

Brute forcing was attempted against both the SQL database and the SSH service that was found during initial enumeration checks. The SSH returned that normal authentication with a user and password was not possible, and the SQL database returned false positives for login attempts. This ruled out brute force as a method to get into the services.

The web service was then checked for all directories and services, which found wordpress installed on the server. Scans for that returned the admin user as existing on the service, but brute forcing the login did not return a valid password using common 1k password list. A 1.4m password list was left running while checking other exploit avenues but did not return a positive result by the time the testing window was completed.

Checking the robots.txt folder showed that there was a secret directory on the service.

Graphical user interface, text, application

Description automatically generated

Checking there found a base64 encoded string of characters.

Graphical user interface, text

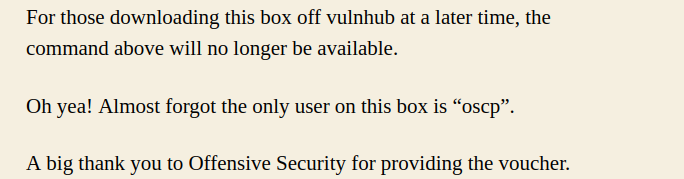
Description automatically generated

Decoding this string resulted in an RSA private key being obtained.

Text

Description automatically generated

Searching for the user of the ssh key through the directories and brute force attempts against the web login led to finding the username listed on the main webpage for the service.



With both pieces of information, it was possible to log in to the network as an authenticated local user. This represented a compromise of all local privilege files and folders on the system.

Text

Description automatically generated

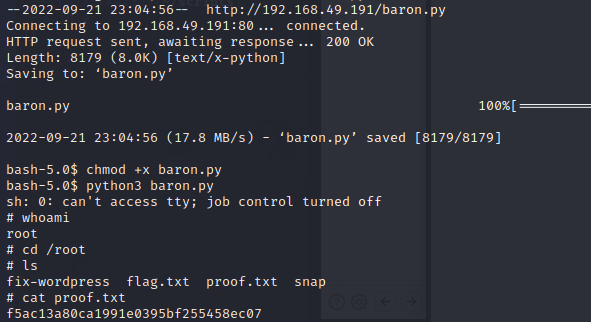
This was proven out with the contents of the local.txt file rendered viewable for the user.

Graphical user interface, text, application

Description automatically generated

Various versions of the binaries and programs on the network device were tested, with there being multiple out-of-date versions on the current installation.

An exploit was uploaded to the /tmp folder to exploit the version of SUDO on the system that was able to spawn a root shell. This represented a complete compromise of the network device, with all files and folders available for retrieval or modification.



Using the privilege, the hash of the user was retrieved from the /etc/shadow file to attempt a brute force against with a 1.4m password list. This password was not able to be cracked with the file, but other rule-based attacks or password lists may be able to do so. This was attempted to prove out some of the other exploits that would require the user password to complete.

One of these was an outdated pkexec version that would have spawned a root shell with the authentication credentials of the local user by abusing the SUID setting on the binary.

Text

Description automatically generated

If the password for the local user was found the pkexec binary can be exploited to gain root access.

Another SUID set binary was found with the bash binary. This was successfully used to create a persistent shell with the elevated privileges inherited by the SUID setting.

Graphical user interface, text

Description automatically generated

The groups associated with the local account were then checked, and a few of the groups allowed for exploit paths that were tested for vulnerabilities.

A few of the exploits were unable to be utilized due to needing compilation through the gcc binary that was not shown on the device. Attempts to download gcc were not made, as this would be out of scope modifications to the system per the terms of engagement of this engagement.

The versions were found to be vulnerable and could be exploited if someone downloaded gcc or a user downloaded it in the future for business purposes. The recommended backup and update will resolve any current concerns regarding these exploits.

The final group that was checked was the lxd group allowance that is not recommended to be added for any local level user due to the exploitations allowed through the privilege.



This will allow for any user with the group privilege to create a container instance with root level privilege afforded them within the instance. If this is needed for normal production functions, it is again recommended to create a separate admin level account that can be sudo’ed into to allow for the privileges in that container.

Graphical user interface, text

Description automatically generated

The allowances allow for exploiting through mounting a container to the root directory. This was attempted, and only failed due to the directory already having a mount connected to it prevent the connection. If this had succeeded, root access would have been provided to the attacker.

Graphical user interface, text

Description automatically generated

**System Vulnerable 192.168.191.89:**

**Vulnerabilities Exploited:**

Information disclosure on public website allowed for login credentials to network

Outdated binary versions allowed for root shell access

SUID permissions on binaries allowed for root shell access

Group privileges allowed for root access within containers

**Severity: Critical**

**Proof of Privilege Escalation:**

Local.txt: 99ed2dc4c46b1926470ad79e9e66c0e2

Proof.txt: f5ac13a80ca1991e0395bf255458ec07

2.4-Report: Clean-up

Enumeration scripts were uploaded to the /tmp directory to check for vulnerabilities on the system that were removed after the output of the checks was completed. Exploits were uploaded to the /tmp directory that were removed after being run. Any spawned directories or files from the exploits were removed form the /tmp directory.

An alpine container image was uploaded to start up lxd for exploitation attempts. The image was removed, and the container stopped on the system. The lxd process is running in an init state, and can be killed or stopped through a restart of the web server.

All information retrieved from the device and found through testing are isolated in their own directory on the attack system and will be removed pending hand-over of all information requested per the terms of engagement for this test.