Solstice 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…………………………………………………………………………………………………………………………..5

**System Vulnerable**……………………………………………………………………………………………………….10

2.4-Report…………………………………………………………………………………………………………………………………11

1.0-High-Level Summary

An internal penetration test was performed on the SSBCUTE 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 the websites were vulnerable to local file inclusion that was viewable from the browser through path traversal. With this method, it was possible to view the /etc/passwd file. It was also possible to view the apache logs for port 80 through port 8593. It was also possible to inject a shell into the port 80 access logs, that allowed commands to be run from the 8593 index.php file.

With this shell code, a netcat shell was able to be formed to the system as the web user of www-data. This caused a local compromise of the system, that rendered the contents of local.txt accessible.

From there further searching was started that uncovered a SUID enabled folder on /var/tmp/sv. The index.php file within that folder was able to be modified, that allowed for copying an exploitable binary with SUID privileges into the /var/tmp folder.

Once the modified index.php was triggered by sending traffic to a local server running with the file, the binary was copied over and able to be utilized for root access to the machine. This caused a complete compromise of the network device.

1.1-Recommendations

To prevent the path traversal that led to local file inclusion, it is recommended to prevent user supplied values to make it to the filesystem APIs. Utilizing a whitelist to validate supplied information against should also prevent it from being able to move between directories on the box as well. It could also be possible to configure it to take any supplied and append it to the expected base directory. It is also recommended to check against both ascii characters, as well as other encoding methods; such as URL.

It is also recommended to sanitize the access.logs in case another method is discovered for path traversal. For this, it should validate any supplied GET parameters and remove any parameters that are outside of normal range (this could be [,],(,) ) Doing this will prevent injected shell code that can be accessed by loading the logs on the website.

Going through and removing SUID set directories and files is also recommended. This should only be enabled for the files that absolutely must have it, and should be avoided on directories in general.

Going through and adjusting permissions on php and other config files is also recommended, as anyone gaining access can modify or deface websites and configurations should they gain access.

It is also recommended to disable the anonymous ftp access on the port 2121 ftp server. While it was not privileged enough to add or remove any files on the server, there is no reason to leave it as a possible vector for attack.

The phpIPAM version should also be upgraded at this time, as there are numerous exploits that affect the current version of 1.4 that are available. The only reason they couldn’t be used at this time, is due to there not being any functionality on the web ports using that service. To future proof the network, an upgrade will prevent a future vulnerability should the phpIPAM website be built out.

It may also be useful to separate the directories and files between the various websites among the different ports. The crossover between port 80 with a vulnerable file on port 8593 allowed for sending traffic to port 80 and validating the input on port 8593. By keeping them separated, the vulnerabilities of one server will not impact other servers.

It is also recommended to flush the access.logs for the apache server, and likely remove read access from all but root. This will remove the code that was uploaded to the server with flushing the file after backing it up for records purposes. By leaving write permissions for the access.log but removing the read access for all but root, will allow for it to continue to log access but not allow for local file inclusion on the index.php file.

2.0-Methodologies

Below is 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 192.168.191.72 was provided ahead of the pentest commencing.

2.2-Service Enumeration

Checking for TCP connections on the device, found that the following services were open:

21 FTP file server

22 Secure Shell Remote access

25 SMTP EXIM4 mail server

80 HTTP Website

2121 FTP file server

3128 HTTP Website

8953 HTTP Website

62524 HTTP Website

2.3-Penetration Testing

Performed initial enumeration on the ftp servers and found that the port 2121 server allowed for anonymous access. No rights were available for adding files to the server, there was no files on the device to test if it was possible to retrieve files from the server at this time.

Text

Description automatically generated

From there, brute force was attempted against both the Secure Shell service, as well as both FTP servers. This used the common names of root, admin, and administrator with a 1k long password list. This returned no results.

Enumerating port 25 for Exim listed a couple possible exploits, but none of the ones attempted panned out with any results.

From there, it was enumerating the websites and the various ports using the typical programs and methods: nikto, gobuster, dirbuster, dirsearch, zap, and manual review of any returned directories and files. No comments or accidental disclosure was found on the websites for any of the ports.

The php enabled sites were collected into a list, and customer scripts were run against them for various vulnerabilities on the URLs. The sqlmap script did not show any positive results for the sql injections possible. When running the path traversal curl script, it was found that path traversal was possible on the index.php file to get the contents of the /etc/passwd file.

Text

Description automatically generated

This provided a user name for the system. Brute force attempts with the user against the FTP server and SSH did not return a positive results with a 1k password list. It may have been possible to break in with a longer password list, such as the rockyou.txt file.

Validating the results of the scan showed that local file inclusion was possible through the website as well.

Text

Description automatically generated

The port 8593 website looked to be tied into the port 80 one, as the log for port 80 was available for viewing on the port 8593 page. After verifying the access.log could be viewed through the path traversal method on index.php, a malicious netcat message was sent out that contained the ability to add cmd functionality to the URL.

Text

Description automatically generated

The php code injected into the log was not visible on the web page, but the GET request was added in.

Text

Description automatically generated

From there, it was just necessary to craft a payload with a shell component that was URL encoded to have the device reach out to the attack computer to establish a reverse shell into the network.

Graphical user interface, text

Description automatically generated

With this shell access, the local system was compromised and the contents of the local.txt were able to be exfiltrated.

The SUID enabled files were checked, but none returned any promising results. The EXIM 4 exploits that were attempted did not return positive results for privilege escalation. Next was enumerating the typical directories for any interesting files or directories listed in them. In the /var/tmp folder, a SUID set directory of sv was found as an exploit vector.

Text

Description automatically generated

Going inside the directory found that there was a php file that had write access on it.

Text

Description automatically generated

With this, it was possible to modify the code to allow for an exploitable binary to be copied over with SUID properties.

Text

Description automatically generated

Checking for the file found that there were a few local services running that were using this file. From there, sending traffic to one of them with the curl function successfully triggered the file into activating.



Once it was activated, it was possible to verify the SUID enabled binary was created in the requested location.

Text

Description automatically generated

The only remaining task to do was to activate the exploitable binary, which resulted in a root shell being created. This represented a complete compromise of the network device.

Graphical user interface, text, application, chat or text message

Description automatically generated

**System Vulnerable 192.168.191.72:**

**Vulnerabilities Expoited:**

Anonymous login to the port 2121 FTP server

Path traversal on the port 8593 website which led to local file inclusion

Shared database between the various websites hosted

No sanitization of the access.logs which allowed for command injection

SUID enabled directories allowing for the creation of SUID enabled binaries

Write access on a file within a SUID enabled directory

**Severity: Critical**

**Proof of Privilege Escalation:**

Local.txt: 91305e3404360b442a3bb47583f3c179

Proof.txt: 68093675de5a8455dbbaf35f85706e62

2.4-Report: Clean-up

The SUID enabled file was removed after utilizing it to gain root access on the device. The exploits that were uploaded and attempted were removed after they failed to escalate privileges. The exploit code is still loaded on the access.logs, which is why it is recommended to back them up to storage to flush the code out with a new log started up. It is also recommended to immediately change the permissions of any created log files to write permission only for non-root users.