Katana 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 testing phase, there were multiple weak points on the various web services hosted. SQL injection was found that allowed the database to be dumped of all contents externally. Among the information was found an admin hash that was cracked for the credentials to log in.

Another web login page allowed any password entered to be authenticated for the admin user. This provided admin privileges to the contents of the ebook store that was on the service.

Information disclosure on the file upload of a web service gave the information regarding the attempts to obfuscate the storage location for file uploads. With the information provided it was possible to upload a shell code to the site and activate it for remote access into the network.

Inside the network was a configuration file viewable from local privilege that contained the admin hash for the account’s password. This was a weak password that was able to be brute forced to find the password for the account.

Checking for privilege escalation found two methods that were successfully carried out to get root shell access to the system.

The first involved an out-of-date version of the SUDO command. Using a buffer overflow exploit, it was possible to spawn a root shell to get access to all files and folder contents on the network.

The second method involved abusing a sudo capability applied to a program on the network device that was able to be exploited for root shell access.

1.1-Recommendations

For the web services, it will be necessary to make sure proper validation of authentication is carried out, and that all fields have a value to check against to make sure authentication is carried out correctly. The port 80 service looks to either not be authenticating properly against the values or is missing a value for the password that allows for automatic authentication with any password on the admin account.

The ebook book.php file will need to be checked for proper sanitization of the inputs to prevent SQL injection attacks from succeeding against the service. The best method to do this is to separate the data from SQL, so it is never interpreted by the SQL parser. Utilizing prepared statements and parameterized queries should prevent any future injections on the site.

Storing the uploaded files with a prepend and migrating to another service is a good method of preventing malicious code from being uploaded and activated on the system. The issue is that this mechanism was displayed on the page when it performed it. The output needs to be removed from the page, so the defense mechanism can work as intended. It would be beneficial to change both the directory and prepend after removing the output, as it is unknown how many attackers have already discovered the parameters.

The passwords hard coded in config files should be removed, or the permissions for read access removed from non-root users if the password needs to be there for business operations. This will prevent accidental disclosure of higher privileged accounts for local access level users.

The SUDO version needs to be upgraded to a version higher than 1.9.5p1 to prevent the buffer overflow exploit from being used against it for root level access on the device. At its current level, there are two different exploit mechanisms that can be used against it to achieve this exploitation.

The SUDO capabilities should also be removed from the python program at this time. There is a known exploit that was used to generate a root shell using the program to spawn a new shell with SUDO privilege.

The local /etc/passwd file contains hashes for the root and local user on the computer system. Given enough time and hardware, it is possible for someone to crack the hashes if the complexity is not strong enough. These hash values should be moved to the /etc/shadow file to prevent accidental disclosure of hash values to lower-level users.

The FTP and SSH services should be updated to have lockout mechanisms in place when they detect too many failed login attempts within a certain timeframe.

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.

The host.allow file can also be updated to only allow access for certain IPs to connect to the service. This method is recommended if there are not too many people who will be connecting to the device from a remote location.

Iptables can be used to limit the rate at which people can connect to the service, which will act as a further barrier against brute force attacks. This setting can easily boost a brute force attack against a known user to take years, if not decades.

Using an IPS program will assist in locking down the FTP service when it detects too many failed attempts to authenticate under a specific user. Once detected, it will lock out the IP address of the person attempting the brute force on the service.

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 192.168.191.83 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:

21 FTP file server

22 Secure Shell Remote Access

80 Web server

7080 Web Server

8088 Web Server

8715 Web Server

2.3-Penetration Testing

Initial scans of the network found multiple open ports that were able to be tested against. Brute force was set with common user credentials and a 1k password list against the FTP and SSH services. While those were running, multiple checks were run against the web server ports for directory enumeration and any vulnerabilities.

From the directory search results on the main port 80 service, a login page was found. While waiting on the results of the scans, a login for admin with password test was entered to try and capture the form to brute force the login with burpsuite. On forwarding the request, it was found that the login completed for the admin account.

Graphical user interface, text, application, website

Description automatically generated

The account was backed out of and various other passwords were used, all of which results in authentication to the admin account on the service.

From there the various .php URLs were collected and sent through for testing for path traversal or SQLi attacks. It was found that the book.php in the ebook directory on the port 80 web server allowed for SQL injection. Using a script, it was possible to dump the contents of the MySQL database for all information. In the contents, it was found there was a hash for the admin account that was able to be cracked.

Graphical user interface, text

Description automatically generated

Further checks on the websites found that there was a location for file uploads on the port 8088 web server under the upload.html directory. On uploading the file, it was discovered there was information disclosure on the page that rendered the defense against malicious code uploads null. It listed the new file name, and that it was going to another web server.

Graphical user interface, text, application, email

Description automatically generated

Another file with a common file name was uploaded using the same method, and the output on the page matched this attempt. With the methods verified, the various web servers were enumerated to check for the php file extension. This led to discovering which web server the files are being moved to, as well as the directory level at which they appear.

Text

Description automatically generated

With that, a listener was set up on the attack machine and the malicious shell code uploaded to the site was activated through the web browser.

Graphical user interface, text, application

Description automatically generated

This exploit was verified to work once a shell into the network was achieved on the attack machine.

Text

Description automatically generated

With this local level access achieved, it was possible to view al local privilege files and folders on the system. This allowed access to the contents of the local.txt file that was stored on the web server home directory.

Text

Description automatically generated

Checking the sudo version show that it was an out-of-date version that is exploitable from a couple different exploits. Using a well-tested python script allowed for a root shell to be spawned. This granted total access to all folders and files on the system, which represented a complete compromise of the network device.

Text

Description automatically generated

This was backed out of, and another method was found using the sudo capabilities granted to the python program.

Text

Description automatically generated

Utilizing a command to spawn a shell with the sudo privileges granted a root shell on the device.

Text

Description automatically generated

It was also found that there was an admin hash in a configuration file that was viewable from local level privileges.

Text

Description automatically generated

This hash was cracked with hashcat, giving the password credentials for the admin user for this web server.

A screenshot of a computer

Description automatically generated with medium confidence

**System Vulnerable 192.168.191.83:**

**Vulnerabilities Exploited:**

Weak admin password for one of the web servers

No password required for admin on one of the web servers

SQL injection allowed on the ebook directory allowed for dumping of all SQL database contents

Information disclosure on the redirect for file uploads showing the final name and directory

Configuration file with password credentials inside for admin account

Out-of-date sudo version allowing for exploits to spawn root shell

Sudo capability setting for the python program allowing for spawning of a root shell

**Severity: Critical**

**Proof of Privilege Escalation:**

Local.txt: a3e8db2104e75cdbc8d08999f1412ae7

Proof.txt: 11c0918ab6a679d63a1a26c3a467f553

2.4-Report: Clean-up

Exploits were uploaded to the /tmp directory that were removed after successfully exploiting the system for a root shell. Vulnerability scanners were uploaded to the /tmp directory that were removed after receiving the results of the checks. A nobody directory remains in the /tmp directory currently. If it is wished to remove the directory and its contents, a chmod -t followed by rm -r <directory> should resolve the issue. The shell php files that were uploaded to the web server were removed after the pentation test was completed.