Driftingblues6 Proving Grounds

Penetration Test Report

Table of Contents

1. High-Level Summary………………………………………………………………………………………………………………………….3
   1. Recommendations………………………………………………………………………………………………………………..3
2. Methodologies………………………………………………………………………………………………………………………………….5
   1. Information Gathering………………………………………………………………………………………………………….5
   2. Service Enumeration…………………………………………………………………………………………………………….5
   3. Penetration Testing………………………………………………………………………………………………………………5

**Service Vulnerable**……………………………………………………………………………………………………………….9

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 a robots.txt file listed the login page for the website, along with information regarding .zip files being on the website. Checking directories for such a file, led to finding a zip file that was password protected. John the ripper was able to find the password to open up the zip file. Inside were the credentials needed to log on to the website.

Once on the website, a shell was able to be uploaded to the site that allowed backdoor access when triggered from the default directory. With this, a shell was created that compromised local files on the network.

At this point, a vulnerability scanner was uploaded to the device to check for any possible exploit avenues. It suggested some exploits, among which were the ‘dirty cow’ kernel exploits for privilege escalation. The first version was used and was unsuccessful. The dirty cow 2 exploit was then used, and a root user account was successfully created in the /etc/passwd file.

After that we were able to utilize the newly created root account from local privilege to get a root shell into the device. This represented a total compromise of the network device, and the root level contents of the system were available for retrieval.

1.1-Recommendations

It is recommended that all .txt files stored on the website be scrubbed of any notes or other sensitive information that may lead hackers towards checking any particular directory or file type. All files that contain sensitive information should also be removed from being stored on the website as well.

Any default admin accounts should also be deactivated after a new user with admin privileges is created, to prevent brute force attempts if user credentials are not discovered for access. If that is not possible, using a passphrase of 15+ characters should reasonably protect against normal brute-forcing methods.

It may be useful to deactivate file uploads for non-admin users on the website if it is possible. Other methods to prevent shell file upload and activation would be: validating only specific types of files for upload server side, changing the default directory uploads go to, and randomizing the name of the uploaded file. These methods help prevent shell upload vulnerabilities on the website.

There were multiple suggested exploit bases on the current versions of various processes on the system. These can be mitigated with an upgrade of the system to prevent them from occurring. Creating a backup of the current image before the upgrade, or sandboxing an upgraded version to test for stability, would be a good method to go about the upgrade without causing any service degradation.

There should also be scheduled maintenances put into place to periodically review the current versions, and implementing upgrades once the process is in place. New vulnerabilities keep being discovered for programs over a decade old. The only way to prevent future vulnerabilities is to upgrade when a new verified stable release is made, and to validate it before putting it into production.

It is also recommended to check the options available on TextPattern to see if some of the features can be turned off to prevent the current vulnerabilities of the version installed on the server. An upgrade may not prevent the Remote Code Execution (RCE) vulnerabilities detected for this program. The most recent stable version shows as 4.8.8, with verified RCE vulnerabilities showing as recent as 4.8.7. It is very likely the same vulnerability exists in the current version, it just has not been documented. An upgrade is not likely to resolve the exploits on this service.

With an upgrade not being a viable avenue, the weak point of the exploits is that they all require authentication to pull off. For this, enforcing passphrase passwords of 15+ characters on all accounts, and deactivating and creating new accounts with a different name for any exploited user should work to prevent them from being a possible attack vector.

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 192.168.191.219 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 5 ports open on the top 1000 ports, none looked available for an exploit. This left it with the following ports as possible exploit vectors:

80 HTTP web access

2.3-Penetration Testing

With only the web service publicly available for enumeration, the normal checks were run to test for any information or vulnerability that could be found; nikto, dirbuster, dirsearch, gobuster, and zap. While waiting for them to spin up, the robots.txt file was checked and information was found that pointed to a directory on the server, along with information regarding file types to check for.

![](data:application/txt;base64,)

With this information, the directory checking software was halted, and a new search made to check for that specific file type. With the new parameters to search for, the zip file was quickly found and downloaded with curl to view on the local attack box. On download, it was found to be password protected from casual users opening it up. Using the John the ripper program, it was possible to get the hash for the password of the zip file.

Text

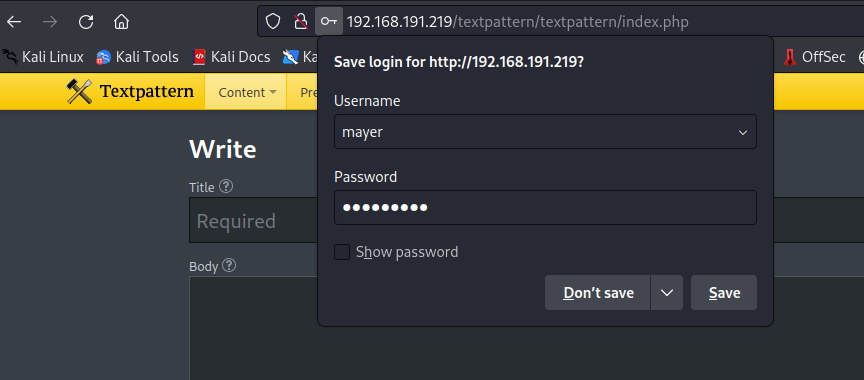
Description automatically generated

With the hash, John the ripper was again run to extract the password for the file. The password was found in 1 second using the regular rockyou.txt file as a password list. Using the password, the creds.txt file was unzipped from the file. Within the file were the username and password credentials for a user.

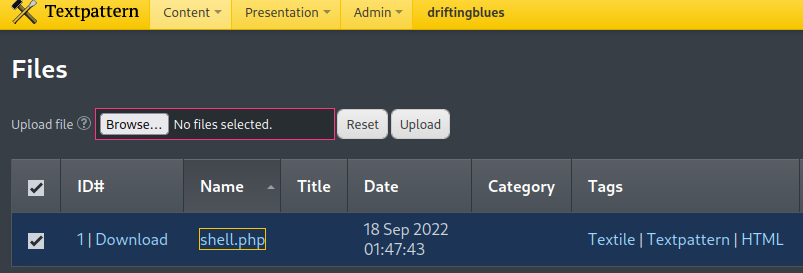
Text

Description automatically generated

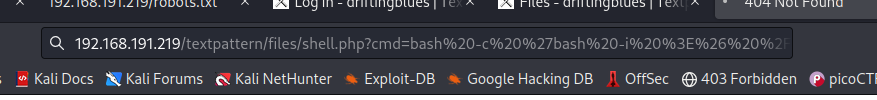
These credentials were checked against the website, as the robots.txt file seemed to hint that the two would be related. On testing, access was granted as a non-admin user to the site.



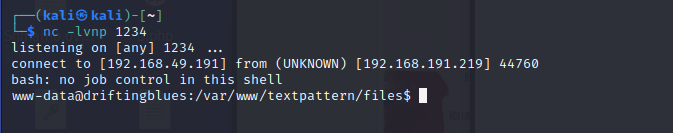
From there, the version of the Textpattern service was viewable in the corner. A search for exploits for this version turned up some Remote Code Execution vulnerabilities for the service. Following the notes in the exploits, a php shell was created and uploaded to the website that would form a reverse shell.



With the file being stored in the default directory, it was possible to directly trigger this shell to execute.

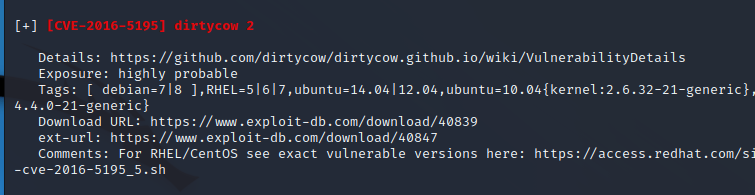


Once the shell file was executed, a connection to the network device was formed with local user credentials. This led to a compromise of the local system files and content.

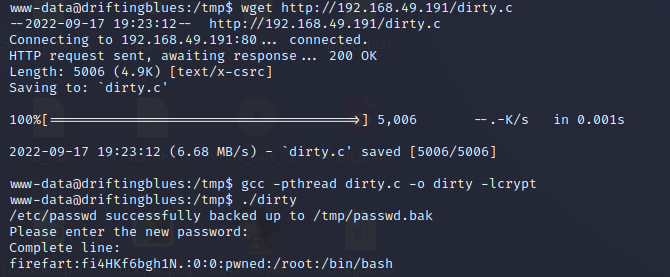


Normal checks for exploit paths did not return any positives. There were no common SUID enabled files/directories found, there were no interesting files to check that the web user had access to. The vulnerability scanner linpeas was then uploaded to the /tmp directory to check for common vulnerabilities on the server.

The scan returned similar results to the manual checks on files and folders. It did find an issue with the kernel version, and returned some suggested vulnerabilities to attempt to exploit. Of these, dirty cow and dirty cow 2 were suggested paths to check for.



Dirty cow was uploaded and tested against the system with no successful results of privilege escalation. On testing with dirty cow 2, the exploit successfully added a user to the /etc/passwd file with root level privileges. This was then used to elevate the local user to a root user with the su command, and root privilege was gained for the system. This represented a complete compromise of the network device, and all files and folders were then accessible.

1

**System Vulnerable 192.168.191.219:**

**Vulnerabilities Exploited:**

Information disclosure on .txt files publicly available on the website

Files with sensitive information publicly available on the website

RCE vulnerability for authenticated users logged in on the website

Default upload directories used for storing files

No server side checks for file types allowed for upload on the site

Outdated versions for kernel allowing for privilege escalation

**Severity: Critical**

**Proof of Privilege Escalation:**

Proof.txt: bef21420b8bfd3a5f4f7e6ec57a8a5df

2.4-Report: Clean-up

Clean up was accomplished after successfully exploiting the system to root. This included moving the backed-up version of /etc/passwd back to its original place. This was validated by checking the contents after the move to ensure the added user was not on there. After that, the two dirty cow exploits were removed from the /tmp directory. Linpeas was removed after getting a .txt print out copy to review for that attack box; this .txt file was also removed from the system. The shell.php file was then removed from the files directory. This restored the system to the state it was in before the penetration test began.