Potato 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…………………………………………………………………………………………………………………………..5

**System Vulnerable**……………………………………………………………………………………………………………..12

2.4-Report…………………………………………………………………………………………………………………………………12

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 FTP server allowed for anonymous access. This led to finding the backup config of the index.php file, which allowed for a look-up of the method used to authenticate logins in the file. With this, it was possible to log in to the web server using just the admin user account, with no password needed.

Once in the web dashboard, an exploit was performed to change the file retrieval system for logs to instead retrieve the /etc/passwd file. This is due to no server side validation of the value being supplied by the client. This value can be edited within the browser, or through proxies such as burpsuite. There were hashes stored within it, that were able to be cracked for remote access to the system itself through Secure Shell. This allowed access to the local file contents.

Checking the sudo allowances for the user, found that a file could be run, with allowances for another directory for all values. Using this, it was possible to create a file on the home directory to execute a bash shell. Running the allowed command as sudo, let the shell be created with root privileges. These privileges allowed the root shell to have total access to the system.

1.1-Recommendations

It is recommended that the anonymous access to the FTP server should be disabled, allowing only authenticated logins. All backup configs and credentials should also be removed from the FTP server as well, in case someone successfully brute forces an account.

For the website, it is recommended to disable any default admin accounts on it, and only use unique usernames. This will slow down any attempts at brute forcing a login to the device. The search feature for retrieving log files should also validate server-side, to make sure only the supplied values are allowed as possible retrievals. Given the radial button nature of the page, it should not cause a hardship in making only those values able to be retrieved.

It was also found that passwords are being stored within the login file index.php, based on the backup configuration. It would be a good idea to not store any passwords on files, or stored on computers if possible.

There were also hashed that were being stored in the /etc/passwd file. All hashes should be updated to only be visible within the /etc/shadow file. This will prevent people without root access from being able to see the hashes of users on the system.

The password for webadmin was also relatively weak, requiring only a few seconds to crack with john the ripper. A more stringent password policy should be instituted, that prevents any passwords located on the rockyou.txt list from being used. Passphrases of 15+ characters would be a good idea at this point in time.

The final recommendation would be to remove unneeded sudo privileges for accounts that do not need them. The /bin/nice file does not appear to be needed and can likely be removed. Barring that, it should at least have its sudo authorization removed, as it should not need to run as root.

The final /notes/\* sudo allowance should be removed entirely, as wildcards allow for path traversals to occur on it. This means they can use the sudo powers on any directory and any file, with root privileges. All sudo privileges should be targeted to specific files only, and only if it is truly needed to run them properly.

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 192.168.191.101 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 3 ports open, none looked available for an exploit. This left it with the following ports as possible exploit vectors:

22 Secure Shell Remote access

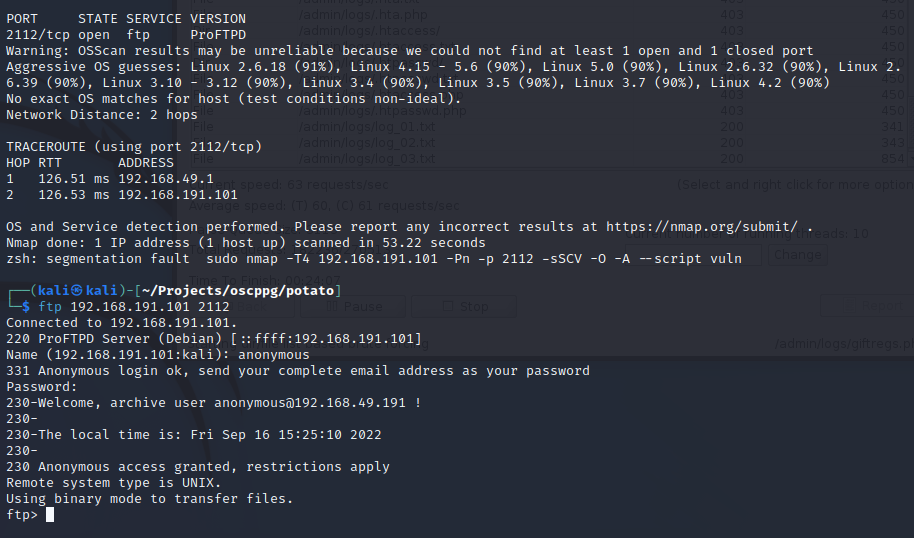
80 HTTP web access

2112 FTP server

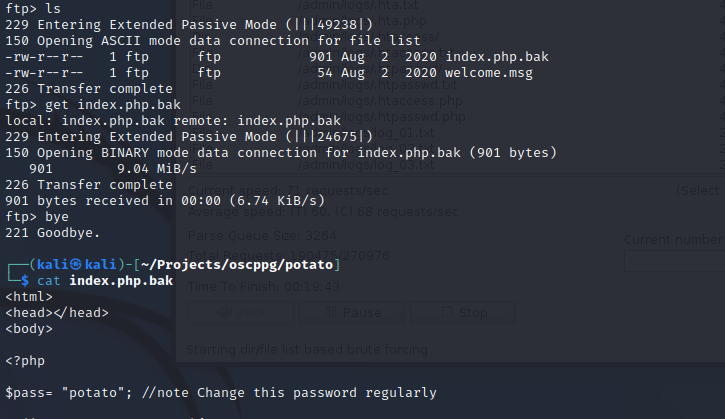
2.3-Penetration Testing

Performed the initial intrusive scanning with nmap, nikto, dirbuster, and gobuster. The directories of the web page were found, along with the logon section for dashboard access. The FTP server did not come up until a full port scan was performed, which is a good safety measure to have in place. Brute force was attempted against common usernames on ssh with a password list of 1k values. None of them returned a success.

Once nmap was found on port 2112, anonymous login was attempted against the device. This led to a successful login, and finding a backup of the configuration for the login .php file.



With this, it was possible to find the backup file of the index.php file, that is being used to authenticate users attempting to log in to the web portal. It was also found that past passwords have been getting stored on the .php file as well.

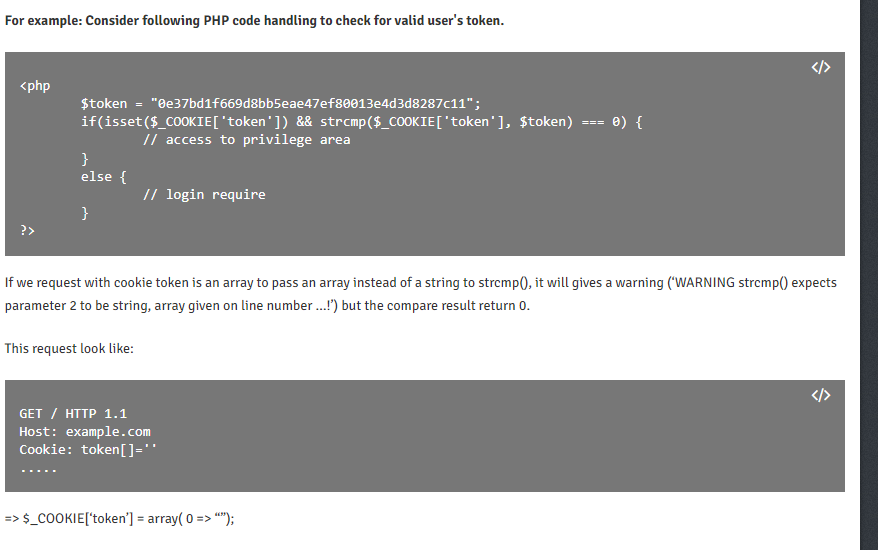


With the file stored locally, it was possible to review the configuration being used for authenticating users attempting to log in to the website.

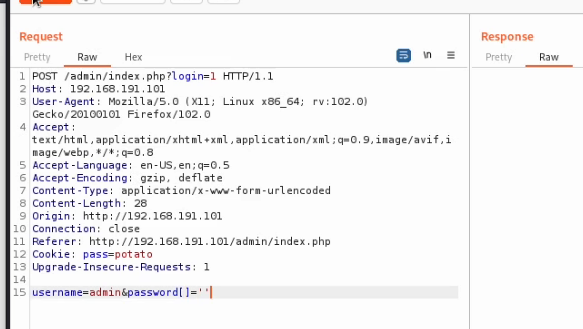
Text

Description automatically generated

With this, it was possible to compare the basic structure of the login, to determine the value it supplies to the server in order to validate the login. This was found on a publicly available website that shows the value comparisons, along with the value to supply on them.



Using the token[]=’’ value from the website, it was possible to authenticate into the website as admin.



Sending this through the burpsuite proxy, enabled admin access to the dashboard of the website.

Graphical user interface, text, application, email

Description automatically generated

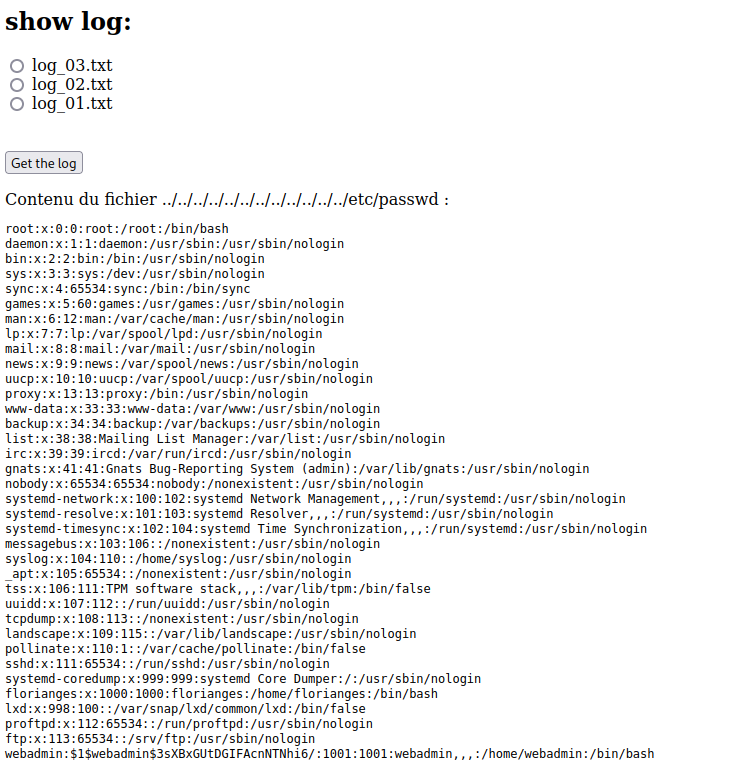
Checking through the different directories eventually led to discovering the Logs section. This had the option to pull from 3 different values with radial buttons. While radial buttons does prevent unforeseen search queries, the value submitted is only validated client-side; not server-side.

Due to this, it was possible to change the value that the browser was supplying for the requested file, so that I could instead view the contents of the /etc/passwd file.

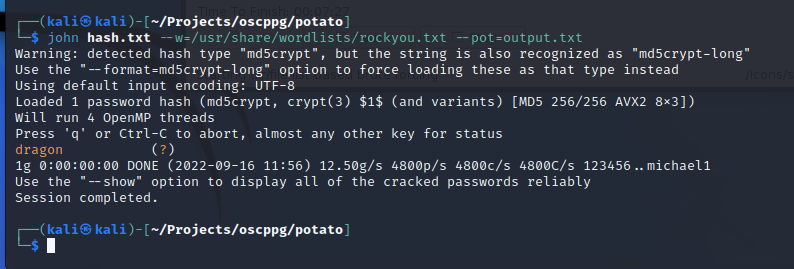
Graphical user interface, text

Description automatically generated

Adjusting the value of the log radial button, then allowed it to execute the adjusted value supplied by the web browser.



The password was not overly complex, and john the ripper was able to find the password associated with the hash in just a few seconds.

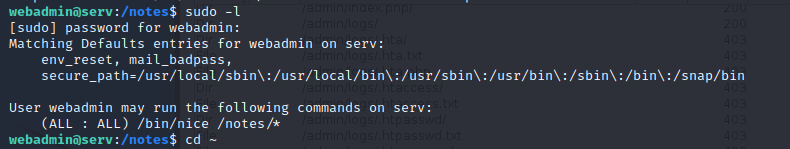


Using the supplied password, it was possible to login to the device over the Secure Shell remote connection on port 2112. This also allowed for compromise of the contents of the local file stored on the computer.

Text

Description automatically generated

There were no SUID exploits showing on the device, but checking for sudo permissions for the user found an exploit vector for root privilege.



This had both an executable file that could be utilized with root privilege, as well as a wildcard directory. The wildcard directory is what enabled the privilege escalation, as it is possible to change the directory and file it is pointing to, just by backing out of the directory.

With these tools, it just needed a directory where write privilege was allowed, and then to point the wildcard towards it.

Text

Description automatically generated

Once the sudo program of /bin/nice was executed against the wild card directory of notes it was appended with the directory of the shell command created. This caused a total compromise of the host.

**System Vulnerable 192.168.191.101:**

**Vulnerabilities Exploited:**

Anonymous access to FTP server

Configs or credentials stored on a publicly accessible server

Index.php authentication occurring client-side, with a value supplied to the server

Client-side validation of the values supplied for the requested file on Logs

Weak password policy allowing for brute-forcing the hash

Wildcard directory being given sudo privileges for the user

Sudo privilege being given to executable files that do not require it

**Severity: Critical**

**Proof of Privilege Escalation:**

local.txt: 4733673ec0e1aae639b89e772ef63d4f

proof.txt: 2902b95bfa751cec91bf48aeb35a2761

2.4-Report: Clean-up

Clean up was accomplished by removing the shell command that was created on the home directory of the webadmin user. This was the only file that was created on the device to allow for successful exploitation. All other aspects of the system remain the same from before the penetration test. It is recommended for an immediate change of the webadmin user to a stronger password until a new password policy is put in place.