DC-2 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**…………………………………………………………………………………………………………9

2.4-Report…………………………………………………………………………………………………………………………………..9

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 there were two services showing on initial scanning. Brute force was attempted against the Secure Shell service with common usernames and a password list of 1k, as well as on the wordpress site.

A wordpress specific scanner was then used to find three viable usernames, and brute force was attempted against both services with these new usernames. Neither service allowed access.

A customer word list was then made utilizing the values specified on the website main page. Redoing the brute force against the two services received two valid usernames and their passwords for the website, and a valid username and password for the Secure Shell service.

Logging into the network through secure shell initially restricted the commands that could be run, but this was broken out of by spawning a shell inside of another program. This allowed for a normal bash session without the restrictions.

While there were vulnerabilities discovered due to out-of-date version of various binaries and programs, there were missing packages or programs that would have been required to be installed. This would fall outside the scope of this engagement for allowed practices, so the vulnerabilities were noted and the test bypassed these techniques.

The credentials received from the website brute force were then used to gain the privilege of a second user account. This second user had a SUDO privilege that was able to be exploited for a root shell on the network device. With this, all files and folders were available for modification or retrieval.

1.1-Recommendations

The wordpress site will need to be updated to a more recent version, due to many vulnerabilities found for the version current running. Fully testing the website fell outside the scope of the engagement, so fully vetting all the files and plugins would need to be done under a different type of penetration test focusing on web application.

With the scans that were completed, the users were able to be found and the passwords were located on the front page of the website. The main webpage should have this content removed to prevent other individuals from creating a custom wordlist that will be able to brute force the users found.

It is also suggested to change the usernames for the webpage to a more complex variety to prevent common user lists from being able to find the ones with accounts. The wordpress logins by their very nature disclose whether an account exists or not, which allows for easy enumeration when there are common usernames used for the website.

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.

The system had various exploitable paths that will need to be patched with a system upgrade. This should be done during a maintenance window with a backup of the current image saved in case recovery is needed.

Currently many of the exploits attempted failed only due to required binaries/functions not being on the system. Should they be added for future functionality, then the exploits would be able to escalate privilege to root without the proper upgrade completed.

Both initial connection to the network and lateral exploitation to a second user were accomplished due to passwords being reused between different services.

To prevent the compromise of one service resulting in the whole network being breached, it is recommended for each service and login have different passwords for the various users connecting to them. This will compartmentalize the services, with each one needing to be exploited separately to harden the over-all network.

SUDO privileges for users should also be used only when necessary to prevent exploitation of the binary or program from being able to spawn a root shell. If the functionality of root level privilege is required for a user utilizing a program, a separate admin account should be utilized for the local level user to elevate their privilege to to carry out the function. This will prevent a local level account’s compromise from leading to complete access to all files and folders on the system.

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 192.168.191.194 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 scan did not return any possible services to exploit in the top 1000 ports. This left it with the following ports as possible exploit vectors:

80 Web Service

7744 Secure Shell Remote Access

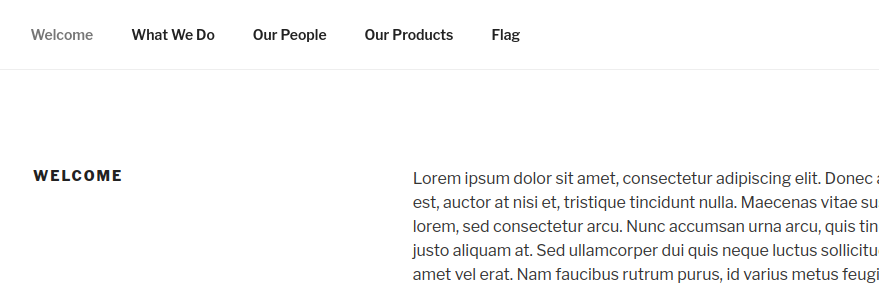
2.3-Penetration Testing

During the testing there were two services found to be vulnerable to attack from external sources: the web service and Secure Shell. Brute force attempts against common admin names and passwords were set to run while setting up enumeration for the directories of the website, searching for any files of interest, and checking for SQL injection and other regular OWASP vulnerabilities.

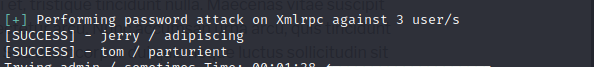
A login page for the website was found for wordpress and scanning for users on the application found three possible users to search for. New brute force attempts were launched with this list, while checking for further vulnerabilities.

The 1k password list did not match for any of the users on any of the services, so a 1.4 mil passwordlist search for the users was launched against the services to run in the background, while creating a customer list using values found on the website.

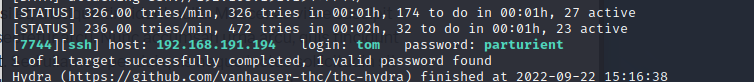
The strange nature of the main web page was targeted for the customer wordlist, that was then used against both services in an attempt to break in with no further information coming from the website enumerating.



The two brute force attempts succeeded with the new password list supplied, finding two usernames and password matches on the website:



Along with a username and password combination found for the Secure Shell service. This password matched between the two different services for this user.

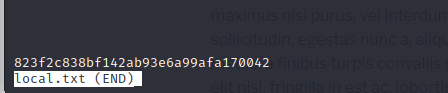


Logging on there were initially very few commands that were able to be completed, due to being in a rbash shell for the user. Various bypass methods were tried until finally succeeding by exploiting the vi binary to spawn a shell inside the running process to break out.

Text

Description automatically generated

With this shell it was possible to move around freely at the local level, and the contents of the local.txt file were retrieved for proof of local level compromise.



From here various checks were made that identified multiple binaries and programs that were older versions susceptible to known exploits for the services. These exploits were uploaded and attempted against the system.

Some were ineligible due to being the wrong version of a program, some were missing specific functions called on the system to complete their exploitation. So while vulnerable, they could not be fully exploited due to the terms of engagement preventing adding programs that would make the system more vulnerable to attack.

During the checks, config files were scraped to find any mentions of password with contents in them. During the checks, it was found that the wp-config.php had some hard-coded credentials inside them for referencing a database.

Text

Description automatically generated

Checking the SQL database with these credentials allowed for access into the service.

Text

Description automatically generated

With this access achieved, the hashes stored within the database were available for retrieval and cracking attempts remotely.

A screenshot of a computer

Description automatically generated with medium confidence

The hashes were not able to be broken with the typical 1.4m password list, but may be susceptible to larger lists or rule-based lists and higher capacity hardware.

With no further luck on finding any credentials or stored keys to use for a different user, the credentials for the second user found on the website were tried to move laterally within the network to the privileges of the other user.

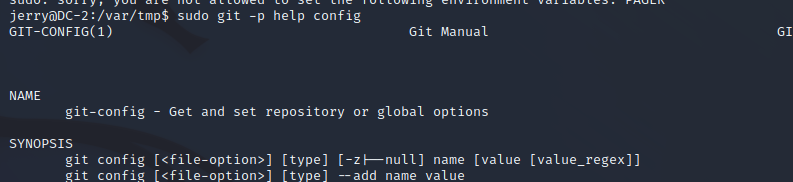


With this access, checks were redone for the new user and a SUDO privilege was found associated with the account.

Text

Description automatically generated

The git binary help menu was called using the SUDO privileges associated with the account.



A root shell was then spawned by breaking out of the help function page with a call to spawn a shell with the SUDO privileges inherited.

Graphical user interface, text

Description automatically generated

With the root shell that spawned, this represented a complete compromise of the network device which rendered all files and folder available for modification or retrieval.

**System Vulnerable 192.168.191.194:**

**Vulnerabilities Exploited:**

Information disclosure with account passwords listed on the website

No lockouts on the SSH service to prevent brute-force attempts

Reused passwords between different services

SUDO enabled binary allowed for a root shell to be created

Common usernames used for a website that gives error messages indicating if the user exists

**Severity: Critical**

**Proof of Privilege Escalation:**

Local.txt: 823f2c838bf142ab93e6a99afa170042

Proof.txt: 4b9ef956f0d861d126e5741ad71fed31

2.4-Report: Clean-up

After gaining access to the system, various exploits were attempted to take advantage of the outdated versions of various binaries on the system. The temporary files created from these exploits were deleted, as well as the exploits after the attempts were completed.

Enumeration programs were uploaded to check for further vulnerabilities on the system, then removed once the results of the scans were completed. With this completed, the network device was returned to the state it was in before the penetration test began.

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.