Vegeta1 Proving Grounds

Penetration Test Report

Table of Contents

1.0-High-Level Summary………………………………………………………………………………………………………………………….3

1.1-Recommendations………………………………………………………………………………………………………………..3

2.0-Methodologies………………………………………………………………………………………………………………………………….4

2.1-Information Gathering………………………………………………………………………………………………………….4

2.2-Service Enumeration…………………………………………………………………………………………………………….4

2.3-Peneration Testing……………………………………………………………………………………………………………….4

**System Vulnerable**………………………………………………………………………………………………………..8

2.4-Report………………………………………………………………………………………………………………………………….8

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 multiple information disclosures with content hosted on the website. At least one of the credentials recovered was utilized to gain access into the network, compromising the local privilege files and folders on the system.

Once in the network an exploitation was found with the editing permissions for the /etc/passwd file being used to create a root level user. This new user’s permissions were then utilized to gain access to all root files and folders on the system.

1.1-Recommendations

The website should be scrubbed of all content that provides information regarding usernames and passwords. Any honeypot or rabbit hole content can remain hosted, and it is recommended to use an IPS to detect the values provided to create blacklist timeouts to identify and hinder attackers.

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 permissions of the /etc/passwd file should be set for root user only, to prevent any local level users from being able to add a root user into the file. This will prevent attackers form being able to get root privilege with only local user credentials.

The ptrace binary will also need an update to a more recent version, as there is an exploit that exists for the current outdated version. The exploit requires access to the sudo binary, which is currently disallowed for the user logged in with.

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 192.168.191.73 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:

22 Secure Shell Remote Access

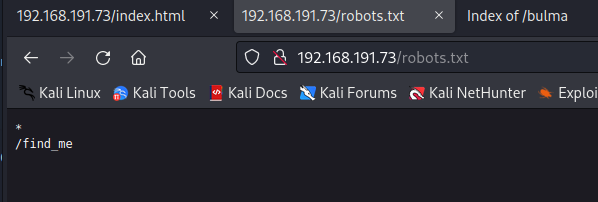
80 Web Service

2.3-Penetration Testing

During the test, enumeration was begun to determine services that were able to be reached externally from the network. This turned up two ports available for exploitation avenues: the SSH service and the web server.

Brute force was set to run against the SSH service while enumeration was done to determine all available files and folders on the website. Common vulnerabilities were also checked for on the php files found in the directories. No results were returned.

Checking the robots.txt file showed a folder that possibly contained some interesting information.



Heading over to that location found a string of text in a base64 format. This was then decoded to reveal a QR code.

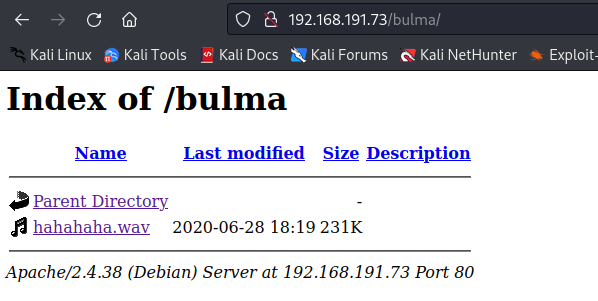
Qr code

Description automatically generated

Checking the QR code information revealed a password credential. This was then used for new brute force attacks against the SSH service with the common usernames, as well as a list of character names related to the website content. No positive results were returned from that, and it is believed this was a rabbit hole credential.

With no further information returned from the common list used, new enumeration and brute force attempts were started utilizing larger lists to search for any further information that could be found.

With these more robust lists, a new directory was found on the website with a .wav file hosted on it.



Listening to the file revealed the content to be morse code. A decoder was then used on the file, which provided a username and password credential.

Graphical user interface, text, application

Description automatically generated

These credentials were verified as valid with a successful login to the SSH service. This provided local level access to all local files and folders on the system and allowed viewing of the contents of the local.txt file as proof of privilege.

Text

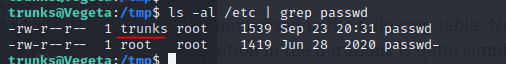
Description automatically generated

Various exploit methods were then attempted to elevate privileges on the system, with two methods found for root exploitation possible.

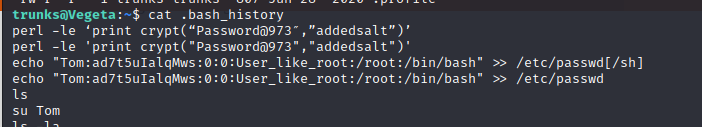
The first was an exploit for the ptrace version loaded on the system. Reading through the exploit showed that the sudo command would be needed for the exploit to function. The sudo command was moved to a different location, and the permissions adjusted to prevent local level use.

This is a good method to bypass the exploit but could be made vulnerable if another user changes the permissions for a business use of new binaries or programs installed. The upgrade would resolve any possible issues arising from changes in the network rendering it vulnerable to the exploit.

The second method was discovering that the /etc/passwd file is writeable from the user logged into. With this permission vulnerability, a user was added into the file with root level privileges.



This was verified through the bash history which showed the account has previously added a user into the file.



The user was created into the file, and root access was successfully achieved through the new user. This represented a complete compromise of the network device, with access to all files and folders on the system.

Graphical user interface, text

Description automatically generated

**System Vulnerable 192.168.191.73:**

**Vulnerabilities Exploited:**

Sensitive information disclosure on the web service for user credentials

Write permissions for user on sensitive files that should be root permission only.

**Severity: Critical**

**Proof of Privilege Escalation:**

Local.txt: 1a61c1383bafdf78f62ae22997faa585

Proof.txt: c564e325882beb5a968d3ee2d55d0db4

2.4-Report: Clean-up

Enumeration scripts were uploaded to the network device to validate all possible vulnerabilities on the system. Once the results of the checks were completed, the scripts were deleted from the /tmp folder they were uploaded to.

The user added into the /etc/passwd file was removed with the root level access that was provided. This restored the file to its original state.

With these changes, the system has been restored to the original state from before the penetration testing 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.