Sar 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

**System 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 testing, it was found that sensitive information was disclosed on the website that pointed to a directory that is vulnerable to exploitation. Utilizing a python script, it was possible to force the php file to create a shell code to connect into the network device.

Multiple vulnerabilities were discovered due to outdated binary versions installed on the computer as well as a cron job running that allowed write permissions for local users. Two methods were used to spawn root shells into the system.

The first is a known python exploit that abuses the SUDO commands when it is a version older than 1.9.5p1 to create a shell using root privilege. This was verified to work, and the contents of the proof.txt file were able to be viewed with this shell.

The second method was to use a cron job running every 5 minutes that called a root user bash script, that referred to another bash script that the local user had write privileges for. This was exploited to edit the script to spawn a reverse shell to the attacker computer, generating a root shell giving access to all files and folders on the network device.

1.1-Recommendations

There are some changes that should be made on the external facing services that will harden the network against external threat actors aiming to break in.

Removing any references to other directories or methods to break in should be scrubbed from all pages supplied by the web server, which will allow it to avoid most directory lists and any suggestions of the type of exploit to use against the service.

The sar2html should be updated to a version past 3.2.1 to bypass the vulnerability used to generate a shell using the command functions enabled with it. There are stable versions of 3.2.2 and 4.0.0 that could be chosen for the upgrade to remove this vulnerability.

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.

Once access was gained into the network, numerous binaries were found to be out of date that would require a system update to prevent exploitation. The SUDO binary was successfully exploited to spawn a root shell, with other binaries required a download of gcc compiler to successfully test to validate.

The download would weaken the network against any other attackers, and therefore against the terms of engagement to successfully validate as working. An update of the system that upgrades the binaries to a new version will bypass any possible issues regarding these exploits.

A later test can be conducted to verify if permission is received to download the gcc compiler to validate the exploit paths. It would be recommended to shut down external connections to the website at this time, and an in-person internal test run to prevent these exploits from being used by third parties during the testing period.

The cron jobs set up should also be simplified to make it easier to track where the permissions are being transferred to. Cronjobs should only be utilized, when necessary, with root privileges enabled for them. They should also have all function completed by the directly referenced script in the cronjob file.

This will allow for easier visibility in what is being requested, so that it can be removed when the function is no longer needed. Passing the privileges through a root shell to a local user will allow for exploits to be added that compromise the entire system with only local level privileges.

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 192.168.191.35 was provided ahead of the pentest commencing.

2.2-Service Enumeration

All available TCP ports were scanned to check for any available attack vector for testing to break into the network. The UDP scan of the top 1000 ports returned no positive results for exploitation. This left it with the following ports as possible exploit vectors:

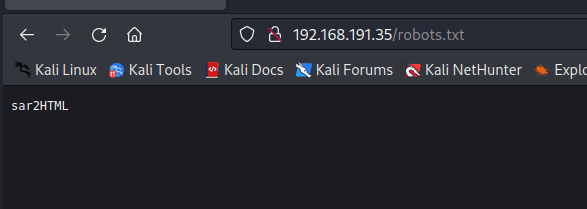
22 Secure Shell Remote Access

80 Web Service

2.3-Penetration Testing

Port enumeration was used to identify the two available TCP ports for exploitation from external attacks. Brute forcing with common admin usernames was set to run against the SSH service, while directory checks and vulnerability scans were run against the web service.

The robots.txt file on the web service disclosed sensitive information regarding both a directory to check, and an exploit to use against the site.



Checking that location found a service with a known vulnerability for the plot feature of the index.php file.

Graphical user interface, text, application, chat or text message

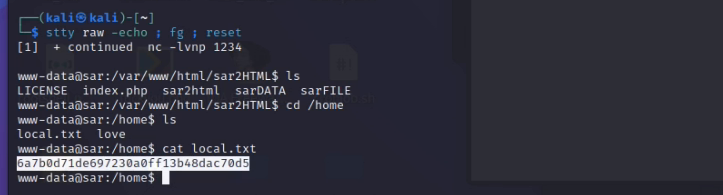
Description automatically generated

The exploit was successfully used against the site to create a web user shell to connect into the network.

Text

Description automatically generated

This represented a compromise of all local level directories and files with the network.

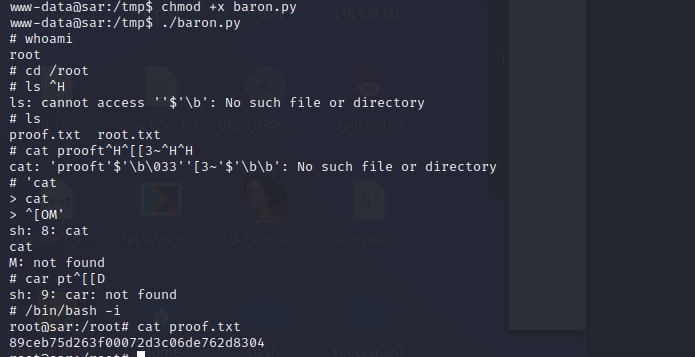


Vulnerability scripts were then uploaded to the /tmp folder to check for all available methods of escalating privilege to the root level for the system. The SUDO version was checked and found to be out of date and vulnerable to the baron samedit exploit.

Text

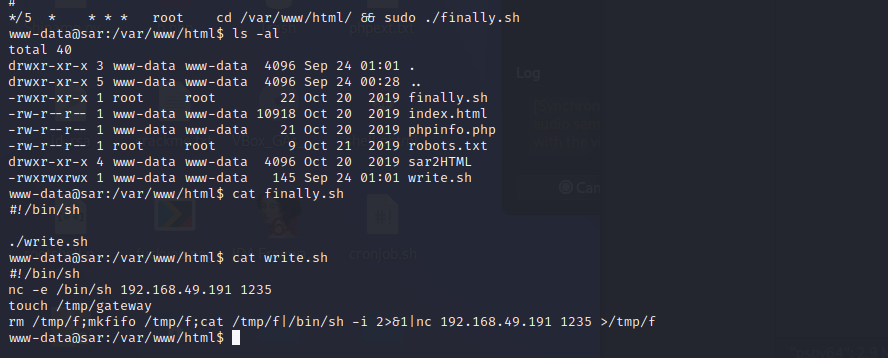
Description automatically generated

The exploit was uploaded to the system and run, to successfully generate a root shell. This represented a total compromise of the network, with all files and folders available for modification and retrieval.

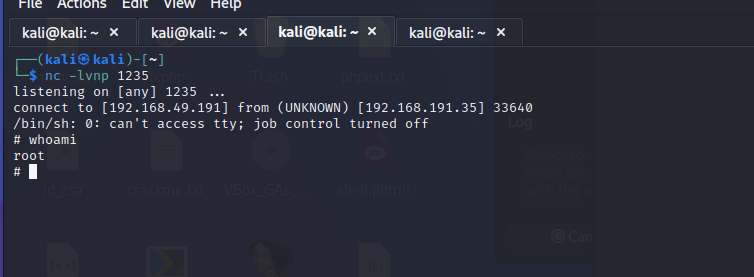


This shell was exited out of, and further vulnerabilities were checked for viability. There were two different exploits that called the c compiler, which is not allowed in the terms of engagement due to weakening the network against further attacks. An upgrade of the system will resolve any possibility of these being activated later if a user installs the gcc compiler on the system

The referenced cronjob was circled back to to discover what the function being called does. Checking the directly referenced one called another shell that was writable at the local level.



Code was added to the file to wait for the cronjob to call the function to spawn a tab with the root permissions of the cron file. After waiting for a few minutes, the exploit was successfully triggered and a root shell was obtained.



**System Vulnerable 192.168.191.35:**

**Vulnerabilities Exploited:**

Sensitive information disclosure on the website allowed for directory and exploit disclosure

Outdated version of sar2html used to generate a shell into the network

Outdated binary versions allowed for successful root shell to be generated

Nested binaries obfuscating the purpose of a cronjob

Local level privilege write access for a bash script being called with root level privileges

**Severity: Critical**

**Proof of Privilege Escalation:**

Local.txt: 6a7b0d71de697230a0ff13b48dac70d5

Proof.txt: 89ceb75d263f00072d3c06de762d8304

2.4-Report: Clean-up

Exploits were uploaded to the /tmp folder for testing vulnerabilities due to outdated binary versions on the system. After they were used, all generated folders were removed along with the exploits from the folder.

Vulnerability scanners were uploaded to the /tmp folder to verify all possible attack vectors on the system and removed once the output of the files was received.

The modified bash script had the extra lines removed from the file to restore it to its previous state. This restored the system to the configuration state it was in before the penetration test commenced.

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.