

---

# LINUX BACKDOOR

---

By:

Jeffrey Sasaki

British Columbia Institute of Technology

COMP 8505 – Assignment 3

Aman Abdulla

May 22, 2015

## Contents

Introduction .....	2
Requirements.....	3
Implementation .....	4
Usage.....	4
State Diagram.....	5
Pseudocode.....	6
client.py.....	6
server.py .....	6
Testing.....	8
Conclusion.....	11

## Introduction

A backdoor is perceived as a negative vulnerability because it allows an attacker to obtain access to a victim's machine without proper credentials. However, a backdoor is more than just a tool of exploitation, it is used far more commonly than one may think.

Generally speaking, the purpose of a backdoor is to allow access to a machine, implemented into the program by the programmer. This is without a doubt a security flaw, however, it is also a tool used for debugging and analytical purposes.

This assignment demonstrates a backdoor program where the attacker is capable of executing shell commands on the victim's machine and returns the response to the attacker.

## Requirements

- Backdoor must camouflage itself so as to deceive anyone looking at the process table.
- Application must ensure that it only receives (authenticate) those packets that are meant for the backdoor itself.
- The backdoor must interpret commands sent to it, execute them and send the results back.
- Incorporate an encryption scheme of your choice into the backdoor.

## Implementation

The program is written in python. There are two programs included in this assignment:

1. client.py (Attacker)
2. server.py (Backdoor Victim)

The client (attacker) program establishes a connection to the server (Victim) and will be able to execute Linux commands against the victim's machine. The messages will be encrypted using the AES encryption scheme while sending data to the server. When the victim sends the message back to the client, it will be encrypted once again; hence, the message will be decrypted to plaintext.

The server (victim) will acquire the encrypted data, decrypt it and execute the command. The command will not appear on the victim's message to emulate a hidden backdoor. The server then encrypts that data, again with AES, and transmit the data back to the client.

The program uses two libraries:

1. pycrypto 2.6.1 – For Encryption
2. setproctitle 1.1.8 – To masquerade process title

It is important to note that to ensure that the attacking machine is also authenticating the packets by utilizing the secret key used (for encryption) as a form of a flag. Any other traffic will be ignored altogether.

## Usage

Prior to running the program, the user must install pycrypto 2.6.1 and setproctitle 1.1.8. The two libraries can be downloaded at:

1. <https://pypi.python.org/packages/source/p/pycrypto/pycrypto-2.6.1.tar.gz>

2. <https://pypi.python.org/packages/source/s/setproctitle/setproctitle-1.1.8.tar.gz#md5=728f4c8c6031bbe56083a48594027edd>

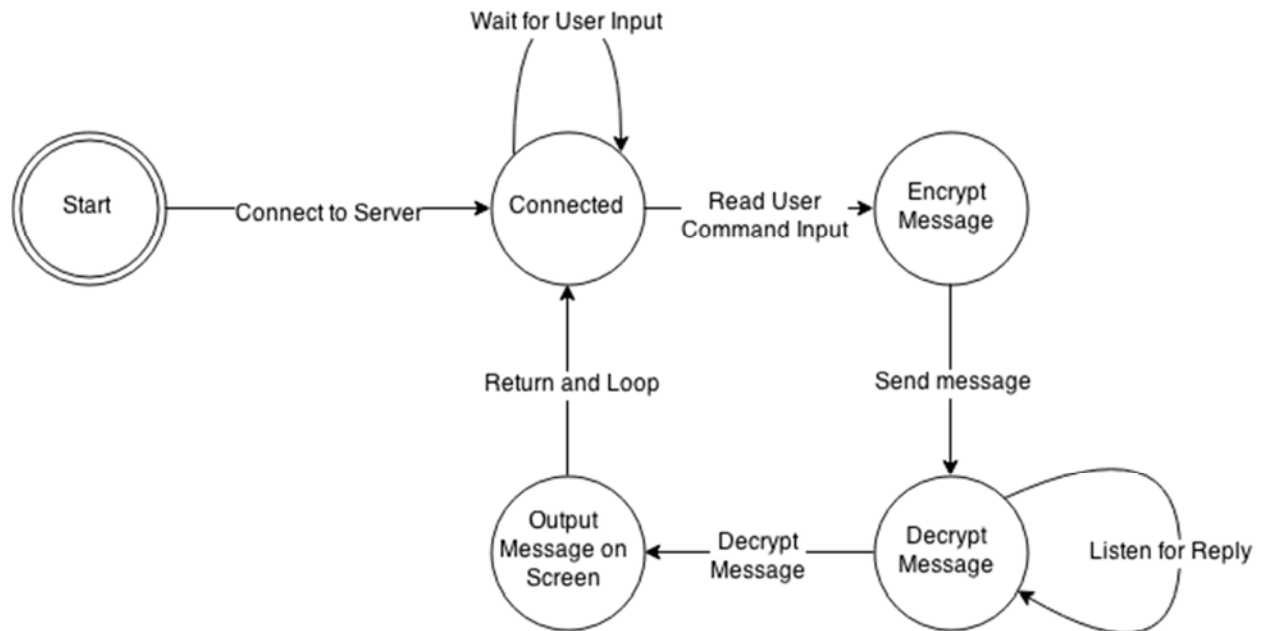
Attacker:      \$ python client.py -d <host\_ip> -p <port>

Victim:        \$ python server.py -p <port> &

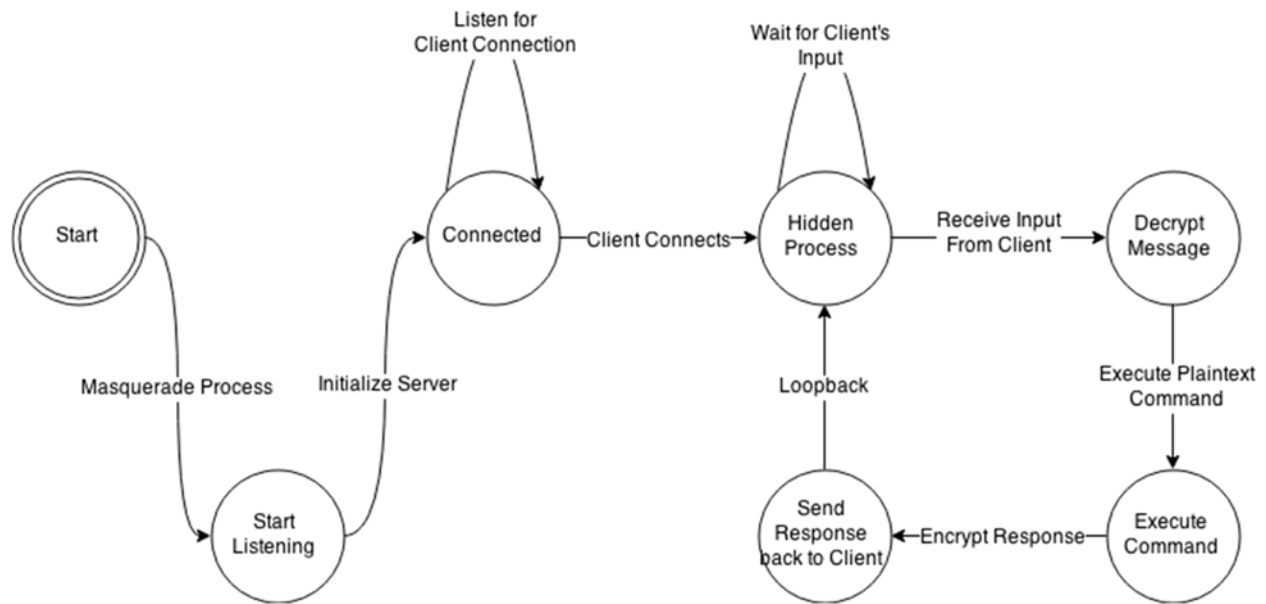
The ampersand (&) will denote that the backdoor will be executed in the background.

### State Diagram

client.py – Attacker



server.py – Victim



## Pseudocode

### client.py

Parse command-line argument

Connect to server

While client is connected to the server

    Input command

    Encrypt and Send command

    Decrypt Response

### server.py

Masquerade process

Parse command-line argument

Listen for connection

While client is connected to the server

Listen for client's Input

Decrypt message and execute command

Send command's response to client



## Testing

```

jeff@localhost:~/Documents/8505/a3-backdoor
File Edit View Search Terminal Help
[jeff@localhost a3-backdoor]$ python client.py -d 107.170.240.112 -p 1234
[remote shell]$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 107.170.240.112 netmask 255.255.255.0 broadcast 107.170.240.255
    inet6 fe80::601:52ff:fea2:1201 prefixlen 64 scopeid 0x20<link>
    ether 04:01:52:a2:12:01 txqueuelen 1000 (Ethernet)
    RX packets 35054 bytes 115520188 (110.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 33208 bytes 14550314 (13.8 MiB)
    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 0 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 fra
me 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[remote shell]$ ls -l
total 480
-rw----- . 1 root root 7692 Dec 3 06:01 anaconda-ks.cfg
drwxr-xr-x. 7 1000 1000 4096 May 24 21:26 pycrypto-2.6.1
-rw-r--r--. 1 root root 446240 May 14 17:48 pycrypto-2.6.1.tar.gz
-rw-r--r--. 1 root root 1943 May 25 00:13 server.py
drwxrwxr-x. 5 1000 1000 4096 May 25 00:15 setproctitle-1.1.8
-rw-r--r--. 1 root root 23208 May 14 15:22 setproctitle-1.1.8.tar.gz

[remote shell]$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

[remote shell]$ aisejfoiasjef
/bin/sh: aisejfoiasjef: command not found

[remote shell]$ exit
[jeff@localhost a3-backdoor]$

```

Screenshot of the attacker's screen (client.py)

```
root@cymba:/root
[root@cymba ~]# python server.py -p 1234 &
[1] 10246
[root@cymba ~]# ls -l
total 480
-rw-----. 1 root root 7692 Dec 3 06:01 anaconda-ks.cfg
drwxr-xr-x. 7 1000 1000 4096 May 24 21:26 pycrypto-2.6.1
-rw-r--r--. 1 root root 446240 May 14 17:48 pycrypto-2.6.1.tar.gz
-rw-r--r--. 1 root root 1943 May 25 00:13 server.py
drwxrwxr-x. 5 1000 1000 4096 May 25 00:15 setproctitle-1.1.8
-rw-r--r--. 1 root root 23208 May 14 15:22 setproctitle-1.1.8.tar.gz
[root@cymba ~]# ps -aux | grep backdoor
root      10246  0.0  2.5 199992 12896 pts/0    S    00:22   0:00 backdoor
root      10253  0.0  0.4 112996 2248 pts/0    S+   00:24   0:00 grep --color=auto backdoor
[root@cymba ~]#
```

Screenshot of the victim's machine (server.py)

```
jeff@localhost: ~/Documents/8505/a3-backdoor
File Edit View Search Terminal Help
[jeff@localhost a3-backdoor]$ python client.py -d 107.170.240.112 -p 1234
0:0:000,00/W0sd
[remote shell]$ id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

[remote shell]$ iptables -L -nvx
Chain INPUT (policy ACCEPT 37758 packets, 115316973 bytes)
  pkts      bytes target    prot opt in      out     source
stination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts      bytes target    prot opt in      out     source
stination

Chain OUTPUT (policy ACCEPT 36112 packets, 14640278 bytes)
  pkts      bytes target    prot opt in      out     source
stination

[remote shell]$
```

The id command shows that the attacker has root privilege, and the iptables command (which needs super user privilege to execute) to support it. It also implies that the targeted machine ran the server program as a super user.

Capturing from wlp3s0b1 [Wireshark 1.12.4 (Git Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: `ip.addr==107.170.240.112` Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
29	4.010771000	192.168.1.67	107.170.240.112	TCP	74	33285→1234 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=18976814 TSecr=0 WS=128
30	4.046178000	107.170.240.112	192.168.1.67	TCP	74	1234→33285 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10661364 TSecr=18976814
31	4.046264000	192.168.1.67	107.170.240.112	TCP	66	33285→1234 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=18976849 TSecr=10661364
32	4.078380000	107.170.240.112	192.168.1.67	TCP	110	1234→33285 [PSH, ACK] Seq=1 Ack=1 Win=28992 Len=44 TSval=10661400 TSecr=18976849
33	4.078520000	192.168.1.67	107.170.240.112	TCP	66	33285→1234 [ACK] Seq=1 Ack=45 Win=29312 Len=0 TSval=18976882 TSecr=10661400
40	12.292096000	192.168.1.67	107.170.240.112	TCP	78	33285→1234 [PSH, ACK] Seq=1 Ack=45 Win=29312 Len=12 TSval=18985095 TSecr=10661400
41	12.325242000	107.170.240.112	192.168.1.67	TCP	66	1234→33285 [ACK] Seq=45 Ack=13 Win=28992 Len=0 TSval=10669645 TSecr=18985095
42	12.328090000	107.170.240.112	192.168.1.67	TCP	1262	1234→33285 [PSH, ACK] Seq=45 Ack=13 Win=28992 Len=1196 TSval=10669650 TSecr=18985095
43	12.328220000	192.168.1.67	107.170.240.112	TCP	66	33285→1234 [ACK] Seq=13 Ack=1241 Win=31616 Len=0 TSval=18985131 TSecr=10669650
47	17.859485000	192.168.1.67	107.170.240.112	TCP	74	33285→1234 [PSH, ACK] Seq=13 Ack=1241 Win=31616 Len=8 TSval=18990663 TSecr=10669650
48	17.897626000	107.170.240.112	192.168.1.67	TCP	598	1234→33285 [PSH, ACK] Seq=1241 Ack=21 Win=28992 Len=532 TSval=10675220 TSecr=18990663
49	17.897766000	192.168.1.67	107.170.240.112	TCP	66	33285→1234 [ACK] Seq=21 Ack=1773 Win=34048 Len=0 TSval=18990701 TSecr=10675220
57	24.827033000	192.168.1.67	107.170.240.112	TCP	82	33285→1234 [PSH, ACK] Seq=21 Ack=1773 Win=34048 Len=16 TSval=18997630 TSecr=10675220
58	24.861355000	107.170.240.112	192.168.1.67	TCP	170	1234→33285 [PSH, ACK] Seq=1773 Ack=37 Win=28992 Len=104 TSval=10682183 TSecr=18997630
59	24.861468000	192.168.1.67	107.170.240.112	TCP	66	33285→1234 [ACK] Seq=37 Ack=1877 Win=34048 Len=0 TSval=18997665 TSecr=10682183
166	53.753956000	192.168.1.67	107.170.240.112	TCP	86	33285→1234 [PSH, ACK] Seq=37 Ack=1877 Win=34048 Len=20 TSval=19026557 TSecr=10682183
167	53.792354000	107.170.240.112	192.168.1.67	TCP	146	1234→33285 [PSH, ACK] Seq=1877 Ack=57 Win=28992 Len=80 TSval=10711113 TSecr=19026557
168	53.792489000	192.168.1.67	107.170.240.112	TCP	66	33285→1234 [ACK] Seq=57 Ack=1957 Win=34048 Len=0 TSval=19026596 TSecr=10711113

Frame 47: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

Ethernet II, Src: Apple\_e3:41:04 (4c:8d:79:e3:41:04), Dst: Actionte\_f5:7e:90 (10:5f:06:f5:7e:90)

Internet Protocol Version 4, Src: 192.168.1.67 (192.168.1.67), Dst: 107.170.240.112 (107.170.240.112)

Transmission Control Protocol, Src Port: 33285 (33285), Dst Port: 1234 (1234), Seq: 13, Ack: 1241, Len: 8

Data (8 bytes)

Data: 676e6b7243446f3d  
[Length: 8]

0000 10 5f 06 f5 7e 90 4c 8d 79 e3 41 04 08 00 45 00 ...L.y.A...E.  
0010 00 3c 1b 7d 40 00 40 06 01 39 c0 a8 01 43 6b aa (...).@..g...K.  
0020 10 70 02 05 04 d2 d2 e3 8b 63 83 49 89 48 80 18 .p.....c.L.F..  
0030 00 f7 4a a1 00 00 01 01 08 0a 01 21 c6 47 00 a2 .....a.l.G..  
0040 5e 52 67 6e 6b 72 43 44 6f 3d .RankrCD a=

Frame (frame), 74 bytes Packets: 532 · Displayed: 18 (3.4%) Profile: Default

Wireshark capture of the data being encrypted during transmission

## Conclusion

The backdoor implementation only works with Linux-based systems. With further enhancements to the program, it is capable of backdooring other operating systems. In addition to that, the “exit” command was implemented into both the client and server program. That confirms that the backdoor program is capable of much more powerful exploitation techniques, such as screenshotting, file transfer, data sniffing, and other malicious activities to the victim’s machine. It is important to note that this is only the basics of a backdoor, where it executes basic commands of a backdoor program.