COMP 8505 Assignment 2

Linux Backdoor

Xinghua Wei & Tao Yuan

A00978597 & A00952007

Introduction

In this assignment, I will implement a Linux backdoor. This backdoor will allow a user to remote control, execute a command, and receive the command result.

Constraints

There are few constraints I need to follow:

- The backdoor must be hidden. So, users can hardly identify it when looking at the process table
- The backdoor should only receive and send packets that are specified.
- The backdoor must interpret commands sent to it and execute them, and send the results back.
- The backdoor commands sent to it and results back should be encrypted

Design

Tools

To satisfy all the constraints with Python, the following tools will be used:

- Python 3.8
- Scapy
    - Scapy has a sniff function that is great for getting packets. Sniff has an argument prn that allows users to pass a function that executes with each packet sniffed. Also, unlike traditional TCP\UDP connections, sniff does not need to specify buffer size, which gives me the flexibility to control the packets.
- Setproctitle
    - The setproctitle module allows a process to change its title as displayed by system tools such as ps. Therefore, anyone who is looking at the process table can hardly identify the backdoor process.
- Subprocess

    - Subprocess module allows users to spawn new processes, connect to their input, output and error pipes
    - Popen function takes arguments to set up the new process so the parent can communicate with it via pipes.
    - Stdout and Stderr will be pipes to open and will be where the command executes and where the results store
- Crypto
    - Crypto is a python cryptography toolkit
    - AES encryption will be used in this assignment

Detail Design

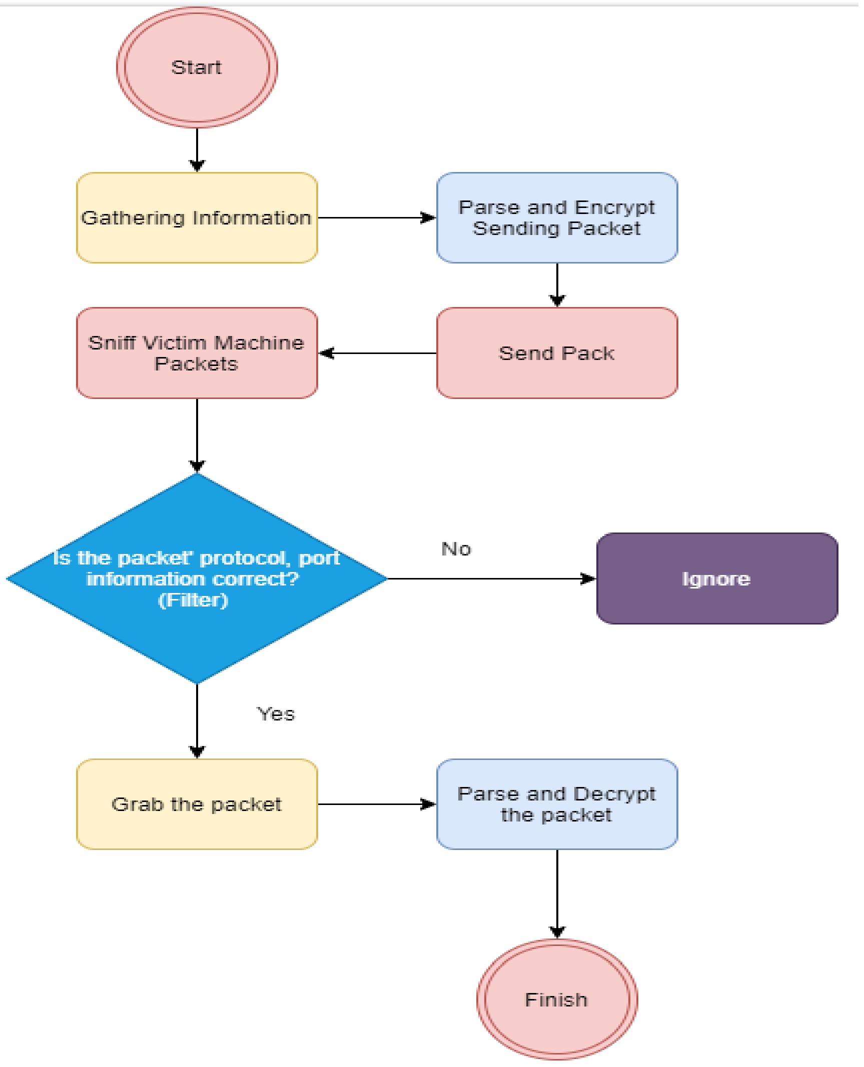
Client (Attacker)

On the client-side, first users need to gather a few pieces of information:

- Target IP address(Destination IP address)
- Target port (Destination port and source port)
    - I will use destination port 8000 and source port 8505 as default
    - UDP will be used as the default
- Attacker IP address(Source IP address)
- The backdoor title which camouflages the backdoor
- The command to send to execute on the victim machine

Once users have all the information, the program will start processing this information and send it to a victim machine.

1. Concatenate the command and process title to one packet
2. Encrypt the packet with AES and UTF-8
3. Send the packet so the victim machine can sniff the packet
4. Sniff the packet in the victim machine using a filter. So, only the information we want would be sniffed.
5. Load the sniffed raw packets and decrypt them into readable information


Start
Gathering Information
Parse and Encrypt Sending Packet
Send Pack
Sniff Victim Machine Packets
Is the packet' protocol, port information correct? (Filter)
No
Ignore
Yes
Grab the packet
Parse and Decrypt the packet
Finish

Server(Victim)

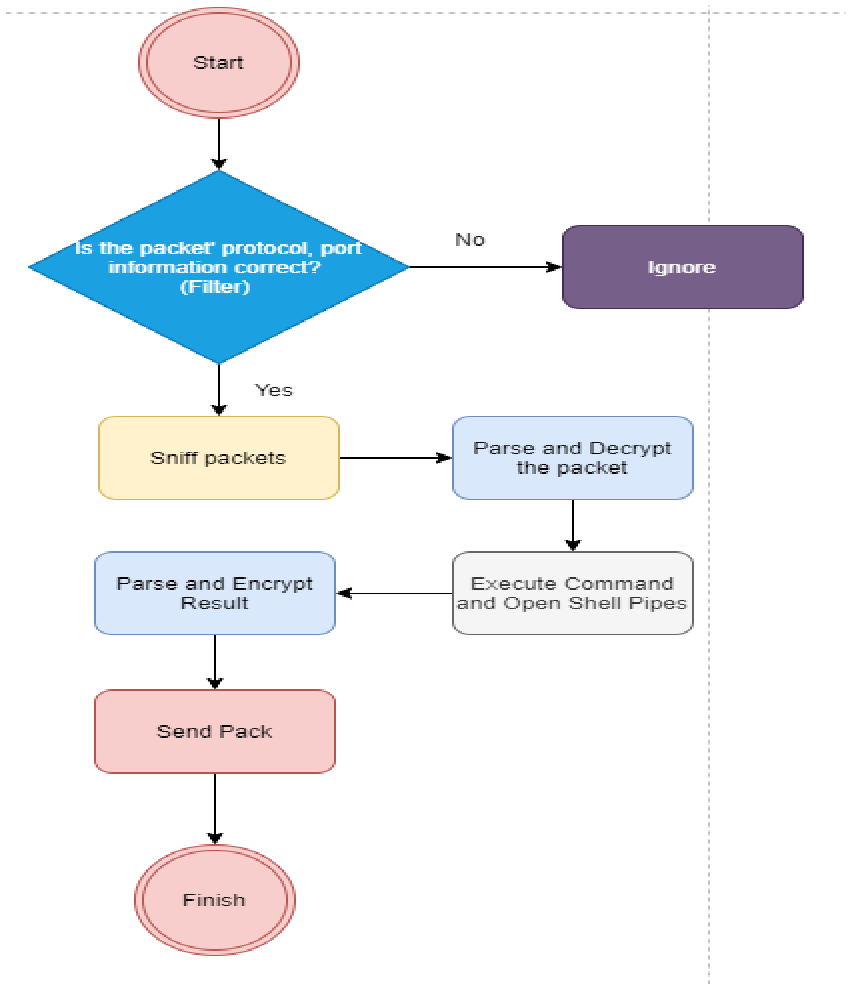
On the server-side, the default protocol, destination port and the source port will be set in this assignment.

- Protocol: UDP
- Destination port: 8000
- Source Port: 8505

Once the program starts, the following actions will be done:

1. Load the packet with sniff
2. Parse and decrypt the packet to retrieve the command and the title

3. Use subprocess and Popen functions to create pips to execute the command
4. Camouflage the process with the title
5. Read results from stdout and stderr
6. Encrypt the result with AES
7. Send packet so the attacker machine can sniff the packet


Start
Is the packet' protocol, port information correct? (Filter)
No
Ignore
Yes
Sniff packets
Parse and Decrypt the packet
Execute Command and Open Shell Pipes
Parse and Encrypt Result
Send Pack
Finish

Cipher

For cipher, I will use AES with CFB 8 mode (8-bit cipher feedback mode). The following information is required:

- Key: fixed data block size of 16 bytes

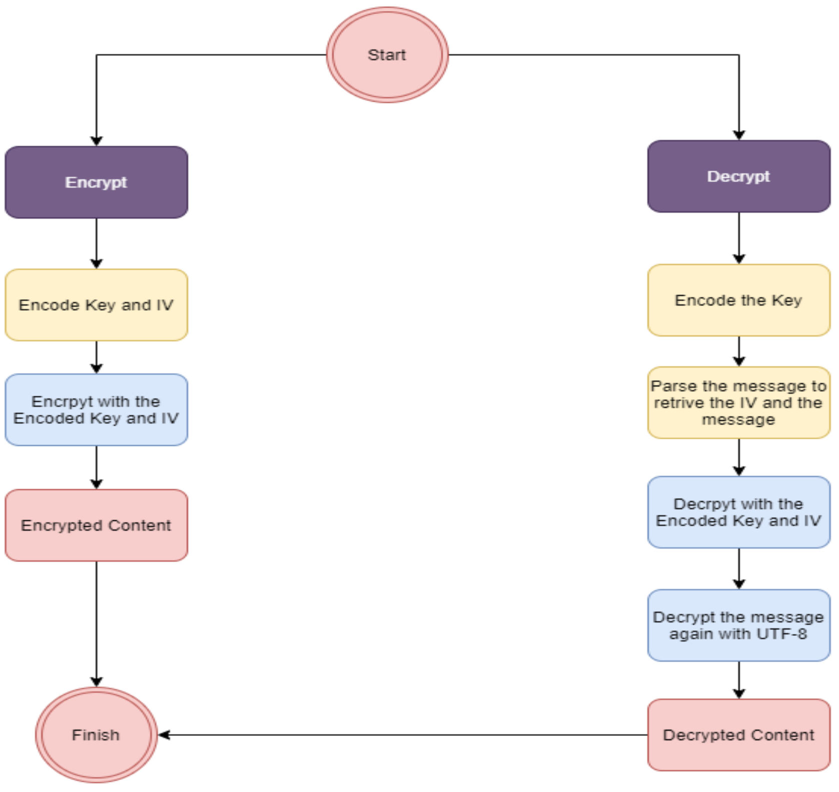
- IV: Initialization Vector is used by several modes to randomize the encryption and produce distinct ciphertexts even if the same plaintext is encrypted multiple times. For CFB mode, it must be 16 bytes long.

Encryption:

1. Encode the key and the initialization vector with UTF-8 because AES in Crypto cannot take a string.
2. Encrypt the message and return the IV and the message

Decryption:

1. Encode the key
2. Parse the message that the former 16 bytes will be the IV and the rest will be the message
3. Decrypt the message with the key and the IV
4. Decode the message again with UTF-8 because it is still in bytes format


Start
Encrypt
Decrypt
Encode Key and IV
Encode the Key
Encrpyt with the Encoded Key and IV
Parse the message to retrive the IV and the message
Decrpyt with the Encoded Key and IV
Encrypted Content
Decrypt the message again with UTF-8
Finish
Decrypted Content

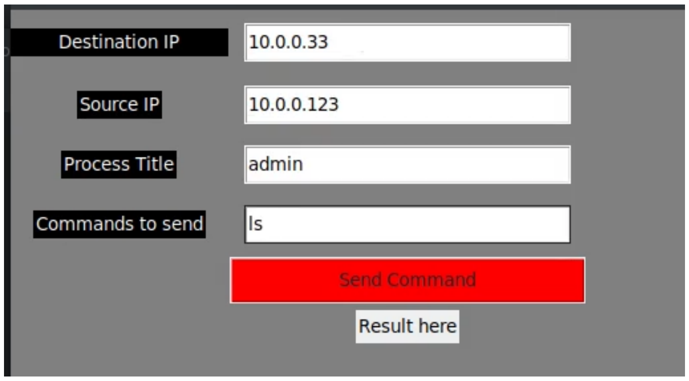
Test

Test Case 1 – The backdoor title (Camouflage the process)

Test case one will demonstrate camouflaging the process title to the title users specified.

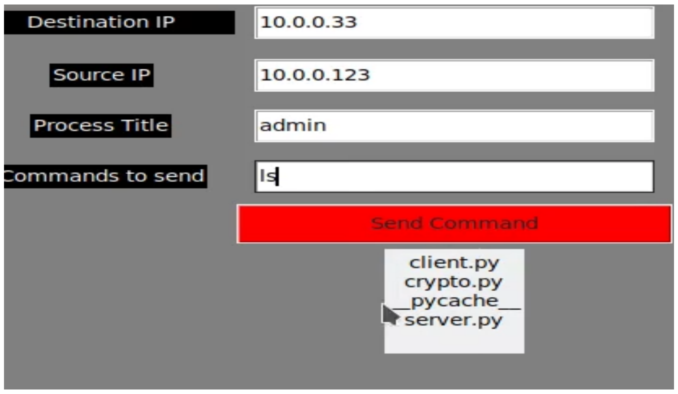
- Destination IP: 10.0.0.33
- Source IP: 10.0.0.123
- Camouflaged title: admin
- Command: ls

Attacker



Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	admin
Commands to send	ls

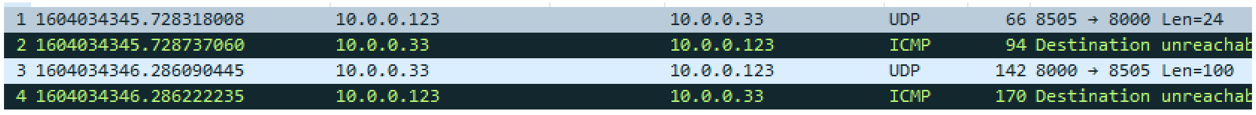
Users specify destination IP, source IP, process title and commands to send.



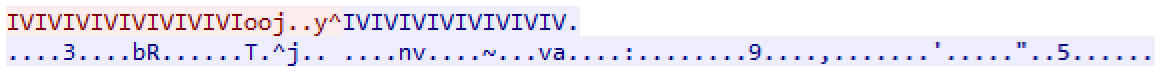
Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	admin
Commands to send	ls

```
client.py
crypto.py
__pycache__
server.py
```

The result is back to the attack machine with lists of files in the backdoor file directory.

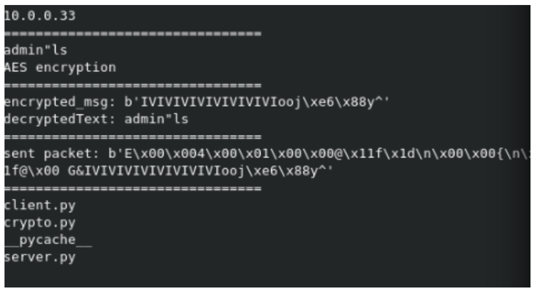


```
1 1604034345.728318008   10.0.0.123   10.0.0.33    UDP    66 8505 → 8000 Len=24
2 1604034345.728737060   10.0.0.33    10.0.0.123   ICMP   94 Destination unreachab
3 1604034346.286090445   10.0.0.33    10.0.0.123   UDP   142 8000 → 8505 Len=100
4 1604034346.286222235   10.0.0.123   10.0.0.33    ICMP  170 Destination unreachab
```



```
IVIVIVIVIVIVIVIVIooj..y^IVIVIVIVIVIVIVIV.
....3....bR......T.^j.. ....nv....~...va....:........9...,.......'....."..5......
```

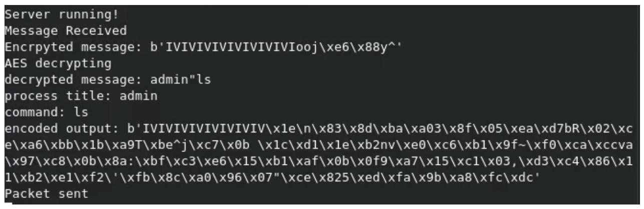
The Wireshark capture shows that the attacker sends the command to the victim and get the result. The result and the command are encrypted.



```
10.0.0.33
================================
admin"ls
AES encryption
================================
encrypted_msg: b'IVIVIVIVIVIVIVIVIooj\xe6\x88y^'
decryptedText: admin"ls
================================
sent packet: b'E\x00\x004\x00\x01\x00\x00@\x11f\x1d\n\x00\x00{\n\
1f@\x00 G&IVIVIVIVIVIVIVIVIooj\xe6\x88y^'
================================
client.py
crypto.py
__pycache__
server.py
```

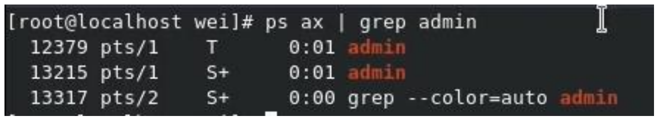
Also, the print out in the console shows the encrypted message being sent and the decrypted result.

Victim



```
Server running!
Message Received
Encrpyted message: b'IVIVIVIVIVIVIVIVIooj\xe6\x88y^'
AES decrypting
decrypted message: admin"ls
process title: admin
command: ls
encoded output: b'IVIVIVIVIVIVIVIV\x1e\n\x83\x8d\xba\xa03\x8f\x05\xea\xd7bR\x02\xc
e\xa6\xbb\x1b\xa9T\xbe^j\xc7\x0b \x1c\xd1\x1e\xb2nv\xe0\xc6\xb1\x9f~\xf0\xca\xccva
\x97\xc8\x0b\x8a:\xbf\xc3\xe6\x15\xb1\xaf\x0b\x0f9\xa7\x15\xc1\x03,\xd3\xc4\x86\x1
1\xb2\xe1\xf2\'\xfb\x8c\xa0\x96\x07"\xce\x825\xed\xfa\x9b\xa8\xfc\xdc'
Packet sent
```

On the victim machine, I printed out the information just to make sure everything is working. So, it shows the command and the title received, and the encrypted message that will be sent back. Because the encoded cipher could be very long, I only print the first 120 bytes.



```
[root@localhost wei]# ps ax | grep admin
  12379 pts/1    T      0:01 admin
  13215 pts/1    S+     0:01 admin
  13317 pts/2    S+     0:00 grep --color=auto admin
```

This image shows that the program successfully camouflaged the process title to the received title “admin.”

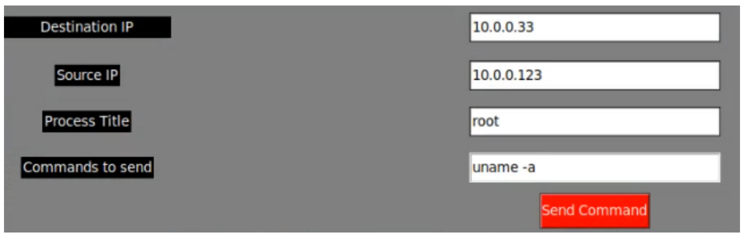
This test means the function to camouflage the process is a success.

Test Case 2 – Encryption

Test case one will demonstrate encryption data transmission, including encrypted command and encrypted result.

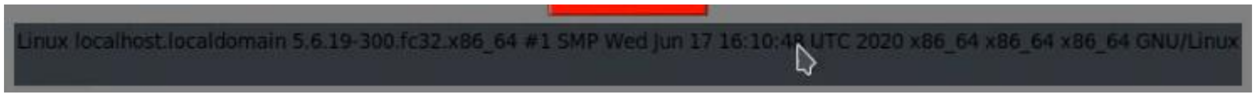
- Destination IP: 10.0.0.33
- Source IP: 10.0.0.123
- Camouflaged title: root
- Command: uname -a
- Encryption

Attacker



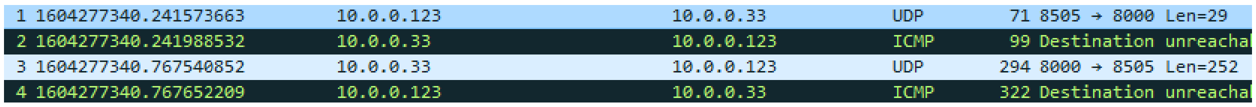
Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	root
Commands to send	uname -a

Users specify destination IP, source IP, process title and commands to send.

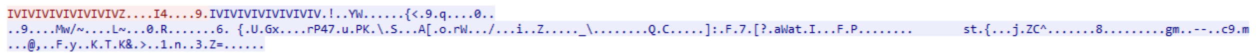


```
Linux localhost.localdomain 5.6.19-300.fc32.x86_64 #1 SMP Wed Jun 17 16:10:48 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

The result is back to the attack machine with the system information of the victim machine.



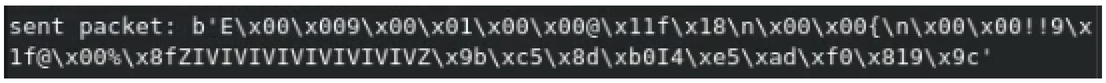
```
1 1604277340.241573663    10.0.0.123    10.0.0.33     UDP     71 8505 → 8000 Len=29
2 1604277340.241988532    10.0.0.33     10.0.0.123    ICMP    99 Destination unreachal
3 1604277340.767540852    10.0.0.33     10.0.0.123    UDP    294 8000 → 8505 Len=252
4 1604277340.767652209    10.0.0.123    10.0.0.33     ICMP   322 Destination unreachal
```



```
IVIVIVIVIVIVIVIVZ....I4....9.IVIVIVIVIVIVIVIV.!..YW......{<.9.q....0..
..9....Mw/~....L~...0.R.......6. {.U.Gx....rP47.u.PK.\.S...A[.o.rW.../...i..Z....._\........Q.C.....]:.F.7.[?.aWat.I...F.P........      st.{...j.ZC^.......8.........gm..--..c9.m
...@,..F.y..K.T.K&.>..1.n..3.Z=......
```

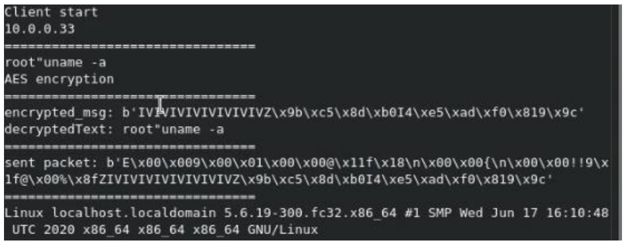
The Wireshark capture shows that the attacker sends the command to the victim and get the result. The result and the command are encrypted.

The encrypted also printed out in the console.



```
sent packet: b'E\x00\x009\x00\x01\x00\x00@\x11f\x18\n\x00\x00{\n\x00\x00!!9\x
1f@\x00%\x8fZIVIVIVIVIVIVIVIVZ\x9b\xc5\x8d\xb0I4\xe5\xad\xf0\x819\x9c'
```

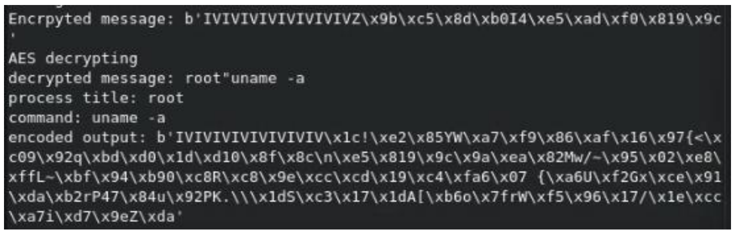
The data was encrypted using AES in bytes and IV combined with it.



```
Client start
10.0.0.33
================================
root"uname -a
AES encryption
================================
encrypted_msg: b'IVIVIVIVIVIVIVIVZ\x9b\xc5\x8d\xb0I4\xe5\xad\xf0\x819\x9c'
decryptedText: root"uname -a
================================
sent packet: b'E\x00\x009\x00\x01\x00\x00@\x11f\x18\n\x00\x00{\n\x00\x00!!9\x
1f@\x00%\x8fZIVIVIVIVIVIVIVIVZ\x9b\xc5\x8d\xb0I4\xe5\xad\xf0\x819\x9c'
================================
Linux localhost.localdomain 5.6.19-300.fc32.x86_64 #1 SMP Wed Jun 17 16:10:48
 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

Also, the print out in the console shows the encrypted message being sent and the decrypted result.

Victim



```
Encrpyted message: b'IVIVIVIVIVIVIVIVZ\x9b\xc5\x8d\xb0I4\xe5\xad\xf0\x819\x9c
'
AES decrypting
decrypted message: root"uname -a
process title: root
command: uname -a
encoded output: b'IVIVIVIVIVIVIVIV\x1c!\xe2\x85YW\xa7\xf9\x86\xaf\x16\x97{<\x
c09\x92q\xbd\xd0\x1d\xd10\x8f\x8c\n\xe5\x819\x9c\x9a\xea\x82Mw/~\x95\x02\xe8\
xffL~\xbf\x94\xb90\xc8R\xc8\x9e\xcc\xcd\x19\xc4\xfa6\x07 {\xa6U\xf2Gx\xce\x91
\xda\xb2rP47\x84u\x92PK.\\\x1dS\xc3\x17\x1dA[\xb6o\x7frW\xf5\x96\x17/\x1e\xcc
\xa7i\xd7\x9eZ\xda'
```

On the victim machine, I printed out the information to make sure everything is working. So, it shows the command and the title received, and the encrypted message sent back. Because the encoded cipher could be very long, I only print the first 120 bytes.

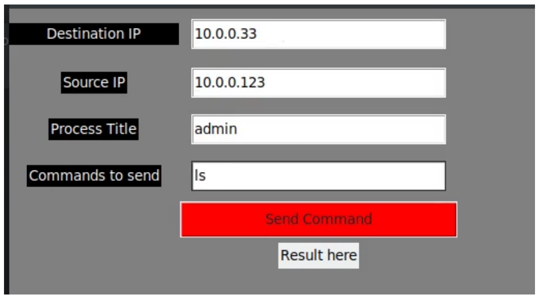
This test means the data encryption function is a success.

Test Case 3 – List files

Test case one will demonstrate list files in the current directory remotely to the victim machine.

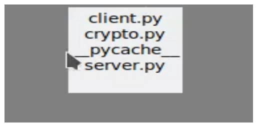
- Destination IP: 10.0.0.33
- Source IP: 10.0.0.123
- Camouflaged title: admin
- Command: ls

Attacker



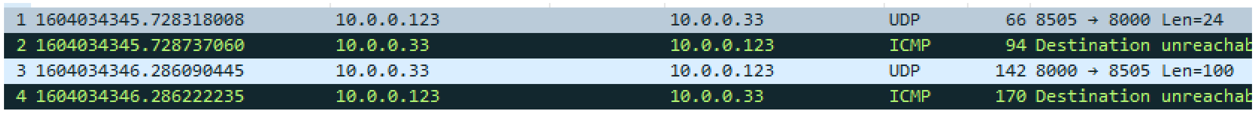
Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	admin
Commands to send	ls

Users specify destination IP, source IP, process title and list files "ls" commands to send.

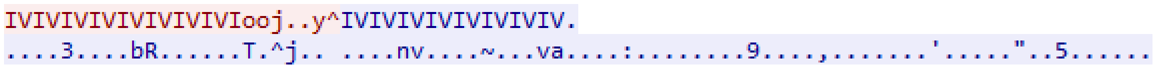


```
client.py
crypto.py
__pycache__
server.py
```

The result is back to the attack machine with lists of files in the backdoor file directory.

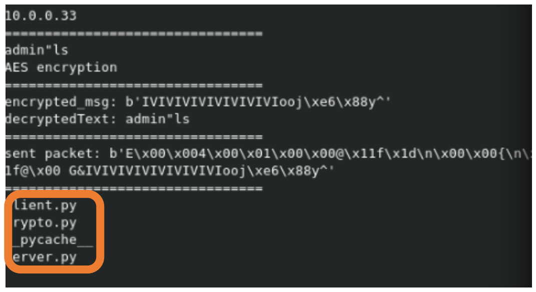


```
1 1604034345.728318008   10.0.0.123   10.0.0.33    UDP    66 8505 → 8000 Len=24
2 1604034345.728737060   10.0.0.33    10.0.0.123   ICMP   94 Destination unreachab
3 1604034346.286090445   10.0.0.33    10.0.0.123   UDP   142 8000 → 8505 Len=100
4 1604034346.286222235   10.0.0.123   10.0.0.33    ICMP  170 Destination unreachab
```



```
IVIVIVIVIVIVIVIVIooj..y^IVIVIVIVIVIVIVIV.
....3....bR......T.^j.. ....nv....~...va....:........9...,.......'.....”..5......
```

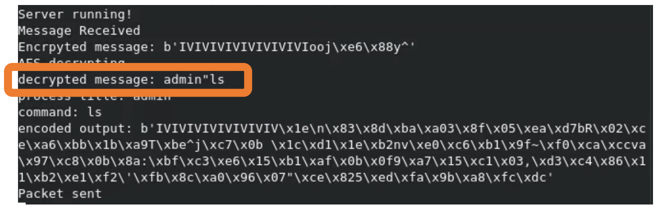
The Wireshark capture shows that the attacker sends the command to the victim and get the result. The result and the command are encrypted.



```
10.0.0.33
==================================
admin"ls
AES encryption
==================================
encrypted_msg: b'IVIVIVIVIVIVIVIVIooj\xe6\x88y^'
decryptedText: admin"ls
==================================
sent packet: b'E\x00\x004\x00\x01\x00\x00@\x11f\x1d\n\x00\x00{\n\
1f@\x00 G&IVIVIVIVIVIVIVIVIooj\xe6\x88y^'
==================================
lient.py
rypto.py
_pycache__
erver.py
```

Also, the print out in the console shows the encrypted message being sent and the decrypted result.

Victim



```
Server running!
Message Received
Encrpyted message: b'IVIVIVIVIVIVIVIVIooj\xe6\x88y^'
AES decrypting
decrypted message: admin"ls
process title: admin
command: ls
encoded output: b'IVIVIVIVIVIVIVIV\x1e\n\x83\x8d\xba\xa03\x8f\x05\xea\xd7bR\x02\xc
e\xa6\xbb\x1b\xa9T\xbe^j\xc7\x0b \x1c\xd1\x1e\xb2nv\xe0\xc6\xb1\x9f~\xf0\xca\xccva
\x97\xc8\x0b\x8a:\xbf\xc3\xe6\x15\xb1\xaf\x0b\x0f9\xa7\x15\xc1\x03,\xd3\xc4\x86\x1
1\xb2\xe1\xf2\'\xfb\x8c\xa0\x96\x07"\xce\x825\xed\xfa\x9b\xa8\xfc\xdc'
Packet sent
```

It shows the command and the title received, and the encrypted message that will be sent back. Because the encoded cipher could be very long, I only print the first 120 bytes.

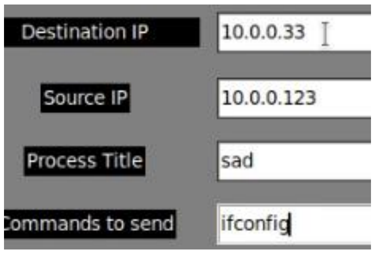
This test means the list file function is a success.

Test Case 4 – Ifconfig

Test case one will demonstrate Ifconfig command in the current directory remotely to the victim machine.

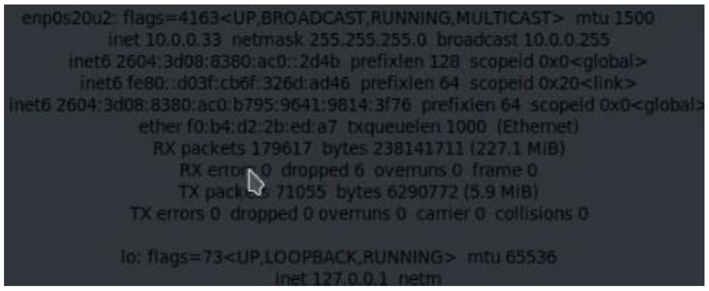
- Destination IP: 10.0.0.33
- Source IP: 10.0.0.123
- Camouflaged title: sad
- Command: ifconfig

Attacker



Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	sad
Commands to send	ifconfig

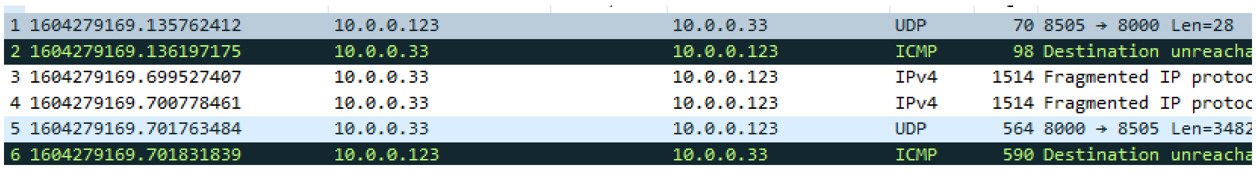
Users specify destination IP, source IP, process title and Ifconfig commands to send.



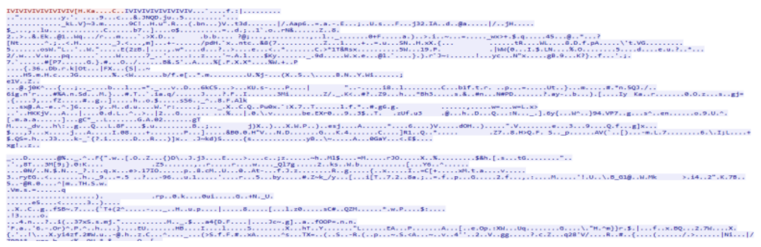
```
enp0s20u2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.33  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 2604:3d08:8380:ac0::2d4b  prefixlen 128  scopeid 0x0<global>
        inet6 fe80::d03f:cb6f:326d:ad46  prefixlen 64  scopeid 0x20<link>
        inet6 2604:3d08:8380:ac0:b795:9641:9814:3f76  prefixlen 64  scopeid 0x0<global>
        ether f0:b4:d2:2b:ed:a7  txqueuelen 1000  (Ethernet)
        RX packets 179617  bytes 238141711 (227.1 MiB)
        RX errors 0  dropped 6  overruns 0  frame 0
        TX packets 71055  bytes 6290772 (5.9 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  netm
```

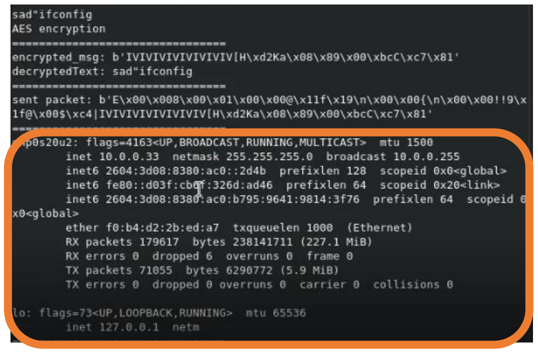
The result is back to the attack machine with Ifconfig of the victim machine. It shows victim machine IP address, network card names, IPv6 address, flags and etc.



```
1 1604279169.135762412    10.0.0.123    10.0.0.33     UDP     70 8505 → 8000 Len=28
2 1604279169.136197175    10.0.0.33     10.0.0.123    ICMP    98 Destination unreacha
3 1604279169.699527407    10.0.0.33     10.0.0.123    IPv4  1514 Fragmented IP protoc
4 1604279169.700778461    10.0.0.33     10.0.0.123    IPv4  1514 Fragmented IP protoc
5 1604279169.701763484    10.0.0.33     10.0.0.123    UDP    564 8000 → 8505 Len=3482
6 1604279169.701831839    10.0.0.123    10.0.0.33     ICMP   590 Destination unreacha
```



The Wireshark capture shows that the attacker sends the command to the victim and get the result. The result and the command are encrypted. Because the Ifconfig result is large, so the encryption will be extensive too.

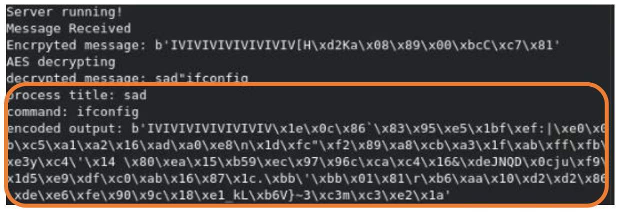


```
sad"ifconfig
AES encryption
================================
encrypted_msg: b'IVIVIVIVIVIVIVIV[H\xd2Ka\x08\x89\x00\xbcC\xc7\x81'
decryptedText: sad"ifconfig
================================
sent packet: b'E\x00\x008\x00\x01\x00\x00@\x11f\x19\n\x00\x00{\n\x00\x00!!9\x
1f@\x00$\xc4|IVIVIVIVIVIVIVIV[H\xd2Ka\x08\x89\x00\xbcC\xc7\x81'
================================
p0s20u2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.33  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 2604:3d08:8380:ac0::2d4b  prefixlen 128  scopeid 0x0<global>
        inet6 fe80::d03f:cb0f:326d:ad46  prefixlen 64  scopeid 0x20<link>
        inet6 2604:3d08:8380:ac0:b795:9641:9814:3f76  prefixlen 64  scopeid 0
x0<global>
        ether f0:b4:d2:2b:ed:a7  txqueuelen 1000  (Ethernet)
        RX packets 179617  bytes 238141711 (227.1 MiB)
        RX errors 0  dropped 6  overruns 0  frame 0
        TX packets 71055  bytes 6290772 (5.9 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  netm
```

Also, the print out in the console shows the encrypted message being sent and the decrypted result.

Victim



```
Server running!
Message Received
Encrpyted message: b'IVIVIVIVIVIVIVIV[H\xd2Ka\x08\x89\x00\xbcC\xc7\x81'
AES decrypting
decrypted message: sad"ifconfig
process title: sad
command: ifconfig
encoded output: b'IVIVIVIVIVIVIVIV\x1e\x0c\x86`\x83\x95\xe5\x1bf\xef:|\xe0\x0
b\xc5\xa1\xa2\x16\xad\xa0\xe8\n\x1d\xfc"\xf2\x89\xa8\xcb\xa3\x1f\xab\xff\xfb\
xe3y\xc4\'\x14 \x80\xea\x15\xb59\xec\x97\x96c\xca\xc4\x16&\xdeJNQD\x0cju\xf9\
x1d5\xe9\xdf\xc0\xab\x16\x87\x1c.\xbb\'\xbb\x01\x81\r\xb6\xaa\x10\xd2\xd2\x86
\xde\xe6\xfe\x90\x9c\x18\xe1_kL\xb6V}~3\xc3m\xc3\xe2\x1a'
```

It shows the command and the title received, and the encrypted message that will be sent back. Because the encoded cipher could be very long, I only print the first 120 bytes. The following is the complete encrypted data.

This test means the Ifconfig command is a success.

Test Case 5 – Create, Execute and Remove an Executable

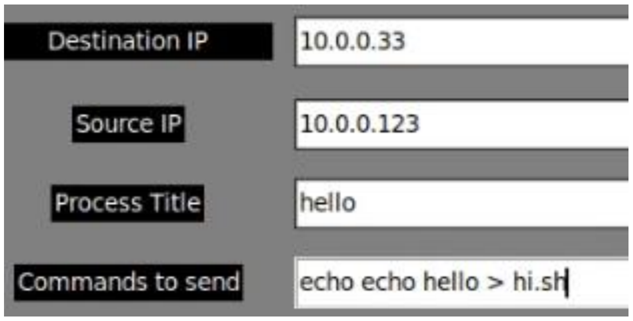
Test case one will demonstrate list files in the current directory remotely to the victim machine.

- Destination IP: 10.0.0.33
- Source IP: 10.0.0.123
- Camouflaged title: sad
- Command: echo / bash / less / rm

Attacker

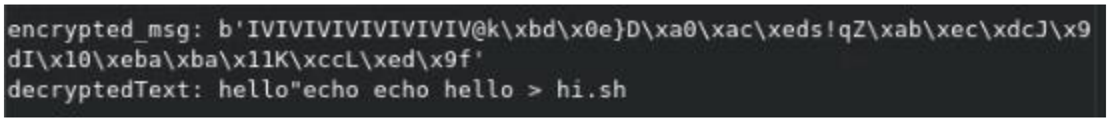
Create Executable

Users specify destination IP, source IP, process title and Ifconfig commands to send. First, I will send "echo echo hello > hi.sh" to create a executable bash file.



Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	hello
Commands to send	echo echo hello > hi.sh

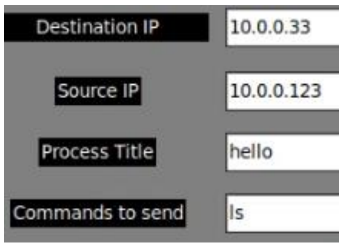
The console shows the plaintext command and encrypted command.



```
encrypted_msg: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa0\xac\xeds!qZ\xab\xec\xdcJ\x9
dI\x10\xeba\xba\x11K\xccL\xed\x9f'
decryptedText: hello"echo echo hello > hi.sh
```

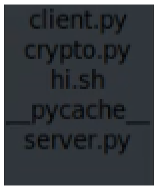
Validate The Executable

Next, I need to check if the file has been created. I use "ls" command to list all files in that directory.



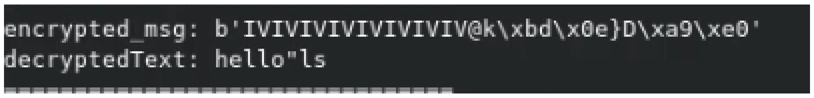
Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	hello
Commands to send	ls

The result shows that the hi.sh has been created.



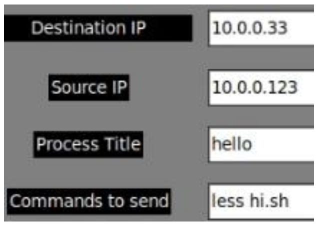
```
client.py
crypto.py
hi.sh
__pycache__
server.py
```

The console shows the plaintext command and encrypted command.



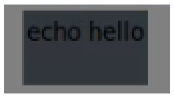
```
encrypted_msg: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa9\xe0'
decryptedText: hello"ls
```

Then I want to use less command to check the content of the executable.



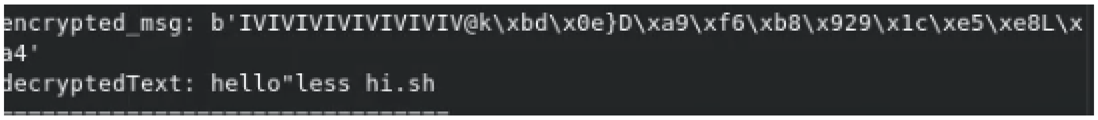
Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	hello
Commands to send	less hi.sh

The result shows that the executable is what I expected.



```
echo hello
```

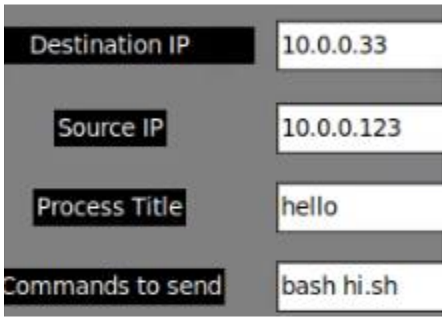
The console shows the plaintext command and encrypted command.



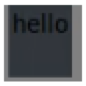
```
encrypted_msg: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa9\xf6\xb8\x929\x1c\xe5\xe8L\x
a4'
decryptedText: hello"less hi.sh
```

Execute The Executable

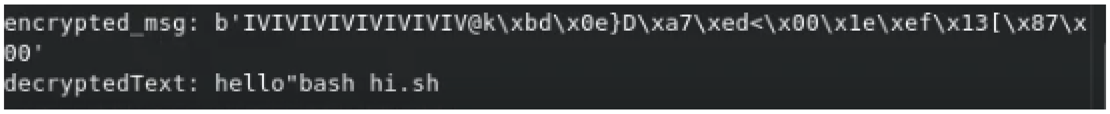
Then I need to run the hi.sh remotely. I used bash hi.sh command.


Destination IP 10.0.0.33
Source IP 10.0.0.123
Process Title hello
Commands to send bash hi.sh

The result shows that it echoed hello in the console.

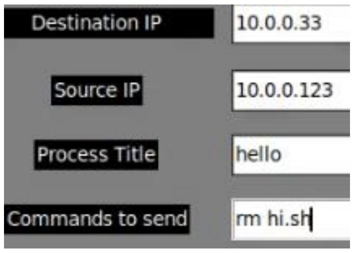

```
hello
```

The console shows the plaintext command and encrypted command.

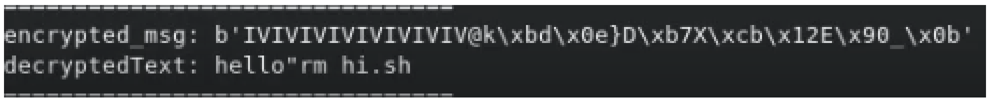

```
encrypted_msg: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa7\xed<\x00\x1e\xef\x13[\x87\x
00'
decryptedText: hello"bash hi.sh
```

Remove The Executable

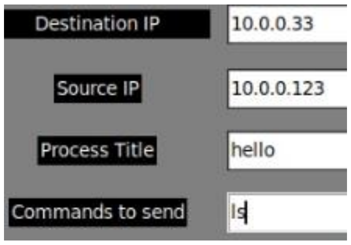
Now, after executing it, I want to remove it from the victim machine. I use rm hi.sh command.


Destination IP 10.0.0.33
Source IP 10.0.0.123
Process Title hello
Commands to send rm hi.sh

The console shows the plaintext command and encrypted command.

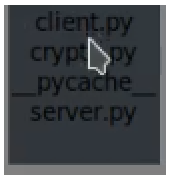

```
--------------------------------
encrypted_msg: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xb7X\xcb\x12E\x90_\x0b'
decryptedText: hello"rm hi.sh
--------------------------------
```

Finally, I use ls command to check if the hi.sh was deleted or not.



Destination IP	10.0.0.33
Source IP	10.0.0.123
Process Title	hello
Commands to send	ls

The result shows that hi.sh was removed.

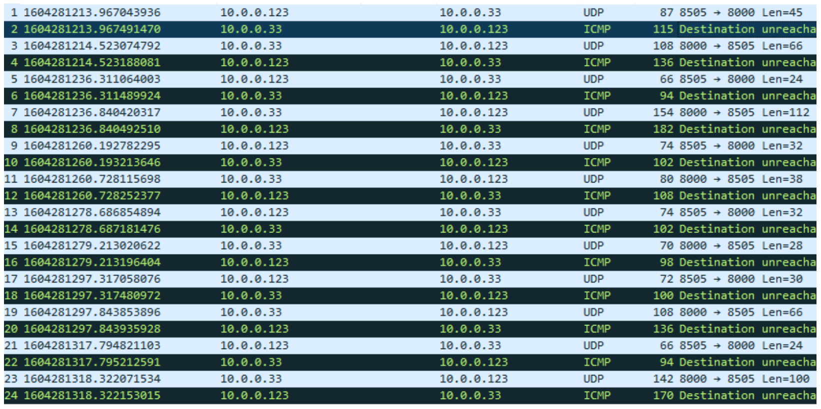


```
client.py
crypto.py
__pycache__
server.py
```

In total, I used six commands for all the steps:

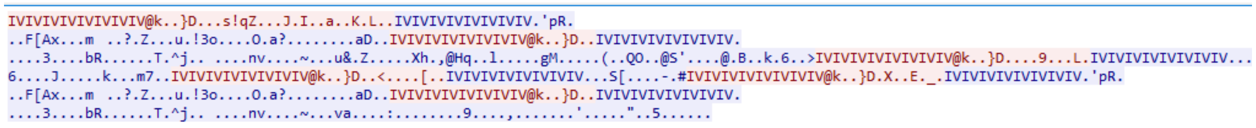
1. Echo echo hello > hi.sh
2. Ls
3. Less hi.sh
4. Bash hi.sh
5. Rm hi.sh
6. Ls

Each step will generate 2 UDP requests. So in total, there will be 12 UDP requests. (ignore the ICMP bad requests.)



```
 1 1604281213.967043936   10.0.0.123    10.0.0.33     UDP     87 8505 → 8000 Len=45
 2 1604281213.967491470   10.0.0.33     10.0.0.123    ICMP   115 Destination unreacha
 3 1604281214.523074792   10.0.0.33     10.0.0.123    UDP    108 8000 → 8505 Len=66
 4 1604281214.523188081   10.0.0.123    10.0.0.33     ICMP   136 Destination unreacha
 5 1604281236.311064003   10.0.0.123    10.0.0.33     UDP     66 8505 → 8000 Len=24
 6 1604281236.311489924   10.0.0.33     10.0.0.123    ICMP    94 Destination unreacha
 7 1604281236.840420317   10.0.0.33     10.0.0.123    UDP    154 8000 → 8505 Len=112
 8 1604281236.840492510   10.0.0.123    10.0.0.33     ICMP   182 Destination unreacha
 9 1604281260.192782295   10.0.0.123    10.0.0.33     UDP     74 8505 → 8000 Len=32
10 1604281260.193213646   10.0.0.33     10.0.0.123    ICMP   102 Destination unreacha
11 1604281260.728115698   10.0.0.33     10.0.0.123    UDP     80 8000 → 8505 Len=38
12 1604281260.728252377   10.0.0.123    10.0.0.33     ICMP   108 Destination unreacha
13 1604281278.686854894   10.0.0.123    10.0.0.33     UDP     74 8505 → 8000 Len=32
14 1604281278.687181476   10.0.0.33     10.0.0.123    ICMP   102 Destination unreacha
15 1604281279.213020622   10.0.0.33     10.0.0.123    UDP     70 8000 → 8505 Len=28
16 1604281279.213196404   10.0.0.123    10.0.0.33     ICMP    98 Destination unreacha
17 1604281297.317058076   10.0.0.123    10.0.0.33     UDP     72 8505 → 8000 Len=30
18 1604281297.317480972   10.0.0.33     10.0.0.123    ICMP   100 Destination unreacha
19 1604281297.843853896   10.0.0.33     10.0.0.123    UDP    108 8000 → 8505 Len=66
20 1604281297.843935928   10.0.0.123    10.0.0.33     ICMP   136 Destination unreacha
21 1604281317.794821103   10.0.0.123    10.0.0.33     UDP     66 8505 → 8000 Len=24
22 1604281317.795212591   10.0.0.33     10.0.0.123    ICMP    94 Destination unreacha
23 1604281318.322071534   10.0.0.33     10.0.0.123    UDP    142 8000 → 8505 Len=100
24 1604281318.322153015   10.0.0.123    10.0.0.33     ICMP   170 Destination unreacha
```

Wireshark also shows that each command and transmission is encrypted.

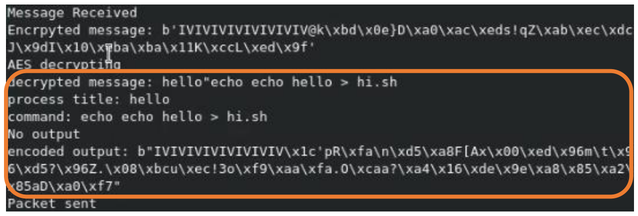


```
IVIVIVIVIVIVIVIV@k..}D...s!qZ...J.I..a..K.L..IVIVIVIVIVIVIVIV.'pR.
..F[Ax...m  ..?.Z...u.!3o....O.a?........aD..IVIVIVIVIVIVIVIV@k..}D..IVIVIVIVIVIVIVIV.
....3....bR......T.^j..  ....nv....~...u&.Z.....Xh.,@Hq..l.....gM.....(..QO..@S'....@.B..k.6..>IVIVIVIVIVIVIVIV@k..}D....9...L.IVIVIVIVIVIVIVIV...
6....J.....k...m7..IVIVIVIVIVIVIVIV@k..}D..<....[..IVIVIVIVIVIVIVIV...S[....-.#IVIVIVIVIVIVIVIV@k..}D.X..E._.IVIVIVIVIVIVIVIV.'pR.
..F[Ax...m  ..?.Z...u.!3o....O.a?........aD..IVIVIVIVIVIVIVIV@k..}D..IVIVIVIVIVIVIVIV.
....3....bR......T.^j..  ....nv....~...va....:........9....,.......'....."..5......
```

Victim

In total, I used six commands for all the steps, so the victim machine will react six times.

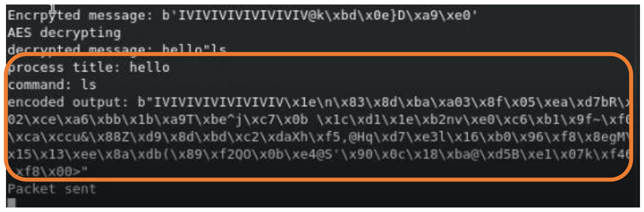
Echo echo hello > hi.sh



```
Message Received
Encrpyted message: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa0\xac\xeds!qZ\xab\xec\xdc
J\x9dI\x10\xeba\xba\x11K\xccL\xed\x9f'
AES decrypting
decrypted message: hello"echo echo hello > hi.sh
process title: hello
command: echo echo hello > hi.sh
No output
encoded output: b"IVIVIVIVIVIVIVIV\x1c'pR\xfa\n\xd5\xa8F[Ax\x00\xed\x96m\t\x9
6\xd5?\x96Z.\x08\xbcu\xec!3o\xf9\xaa\xfa.O\xcaa?\xa4\x16\xde\x9e\xa8\x85\xa2\
x85aD\xa0\xf7"
Packet sent
```

It shows the command and the title received, and the encrypted message that will send back.

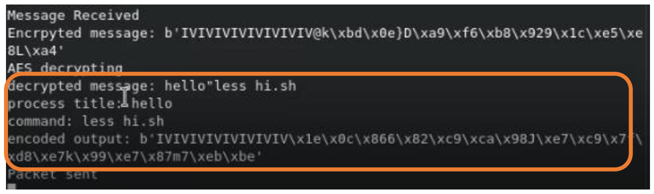
Ls



```
Encrpyted message: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa9\xe0'
AES decrypting
decrypted message: hello"ls
process title: hello
command: ls
encoded output: b"IVIVIVIVIVIVIVIV\x1e\n\x83\x8d\xba\xa03\x8f\x05\xea\xd7bR\x
02\xce\xa6\xbb\x1b\xa9T\xbe^j\xc7\x0b \x1c\xd1\x1e\xb2nv\xe0\xc6\xb1\x9f~\xf0
\xca\xccu&\x88Z\xd9\x8d\xbd\xc2\xdaXh\xf5,@Hq\xd7\xe3l\x16\xb0\x96\xf8\x8egM\
x15\x13\xee\x8a\xdb(\x89\xf2QO\x0b\xe4@S'\x90\x0c\x18\xba@\xd5B\xe1\x07k\xf4
\xf8\x00>"
Packet sent
```

It shows the list file command, and the title received, and the encrypted message that will send back.

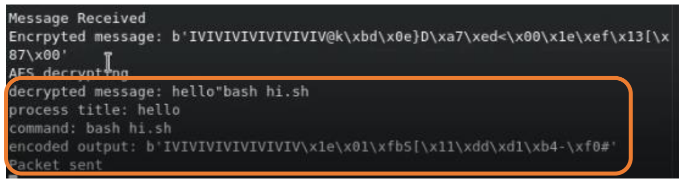
Less hi.sh



```
Message Received
Encrpyted message: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa9\xf6\xb8\x929\x1c\xe5\xe
8L\xa4'
AES decrypting
decrypted message: hello"less hi.sh
process title: hello
command: less hi.sh
encoded output: b'IVIVIVIVIVIVIVIV\x1e\x0c\x866\x82\xc9\xca\x98J\xe7\xc9\x7f\
xd8\xe7k\x99\xe7\x87m7\xeb\xbe'
Packet sent
```

It shows the “less hi.sh” command, the title received, and the encrypted message sent back.

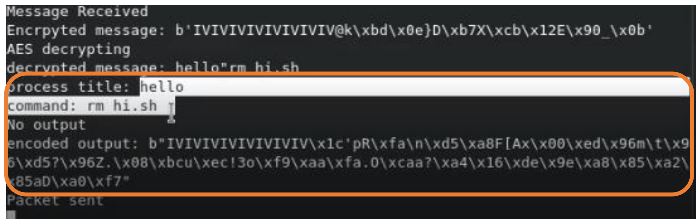
Bash hi.sh



```
Message Received
Encrpyted message: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa7\xed<\x00\x1e\xef\x13[\x
87\x00'
AES decrypting
decrypted message: hello"bash hi.sh
process title: hello
command: bash hi.sh
encoded output: b'IVIVIVIVIVIVIVIV\x1e\x01\xfbS[\x11\xdd\xd1\xb4-\xf0#'
Packet sent
```

It shows the “bash hi.sh” command, the title received, and the encrypted message sent back.

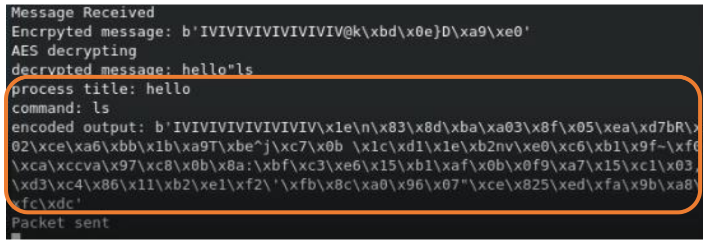
Rm hi.sh



```
Message Received
Encrpyted message: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xb7X\xcb\x12E\x90_\x0b'
AES decrypting
decrypted message: hello"rm hi.sh
process title: hello
command: rm hi.sh
No output
encoded output: b"IVIVIVIVIVIVIVIV\x1c'pR\xfa\n\xd5\xa8F[Ax\x00\xed\x96m\t\x9
6\xd5?\x96Z.\x08\xbcu\xec!3o\xf9\xaa\xfa.O\xcaa?\xa4\x16\xde\x9e\xa8\x85\xa2\
x85aD\xa0\xf7"
Packet sent
```

It shows the remove command, the title received, and the encrypted message sent back.

Ls



```
Message Received
Encrpyted message: b'IVIVIVIVIVIVIVIV@k\xbd\x0e}D\xa9\xe0'
AES decrypting
decrypted message: hello"ls
process title: hello
command: ls
encoded output: b'IVIVIVIVIVIVIVIV\x1e\n\x83\x8d\xba\xa03\x8f\x05\xea\xd7bR\x
02\xce\xa6\xbb\x1b\xa9T\xbe^j\xc7\x0b \x1c\xd1\x1e\xb2nv\xe0\xc6\xb1\x9f-\xf
\xca\xccva\x97\xc8\x0b\x8a:\xbf\xc3\xe6\x15\xb1\xaf\x0b\x0f9\xa7\x15\xc1\x03
\xd3\xc4\x86\x11\xb2\xe1\xf2\'\xfb\x8c\xa0\x96\x07"\xce\x825\xed\xfa\x9b\xa8
xfc\xdc'
Packet sent
```

It shows the list file command, the title received, and the encrypted message sent back.

This test means the function to create, execute and remove an executable is a success.