Testing

Below will be a list of tests that will be ran on the application. The tests below will be split into several parts: Client, watch file feature, retrieve file feature, encryption tests, and the server tests. The description of each test will be posted along with the results beside it, and the last column in the box will indicate whether the test has passed or failed.

After the list of tests, screenshots of each test result will be shown below with the corresponding test number.

Client Tests

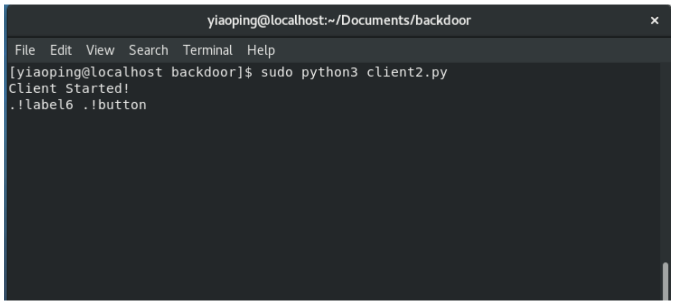
Command Sending

Test #	Description	Results of Test	Pass/Fail
1	The client application can run with no errors through Linux terminal		Pass
2	Client graphical user interface pops up upon running client app		Pass
3	User can enter in Destination IP entry box		Pass
4	User can enter in Source IP entry box		Pass
5	User can enter in name of process title entry box		Pass
6	User can enter in command to send		Pass
7	User can select AES encryption radio button		Pass
8	User can select RSA encryption radio button		Pass
9	User can select Yiao's encryption radio button		Pass
10	Upon sending a proper command to destined IP, user		Pass

	receives correct results back		
11	Test: Sending PWD in command has the result of working directory		Pass
12	Test: Sending ifconfig results in IP of target machine		Pass
13	Sending an unknown command results in calculated error		Pass
14	Sending a command that receives no terminal output results in "No response received but command processed"		Pass
15	Sending command with no IP or command results in no output received. Program does not crash or freeze		Pass
16	Process title successfully changed on server side		Pass
17	Sending empty process title receives no error. Process title does not change		Pass
18	Exit button successfully closes the application		Pass

Test Case 1

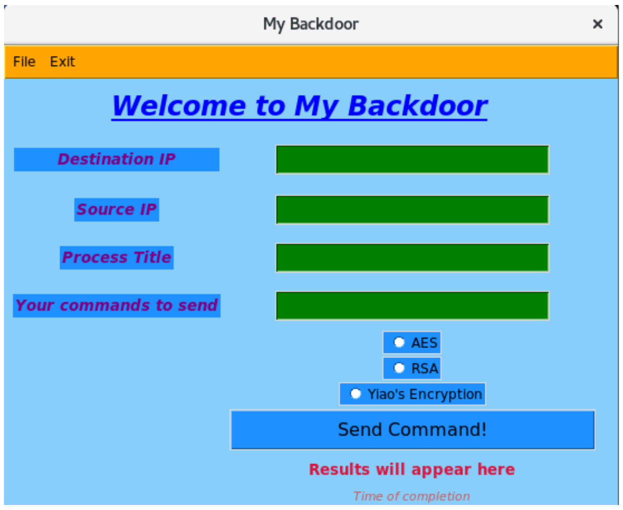
The client backdoor can successfully run without any errors


yiaoping@localhost:~/Documents/backdoor
File Edit View Search Terminal Help

```
[yiaoping@localhost backdoor]$ sudo python3 client2.py
Client Started!
.!label6 .!button
```

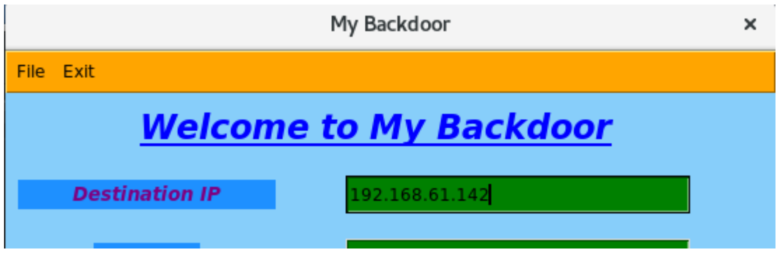
Test Case 2

Client graphical user interface immediately pops up upon running client application



Test Case 3

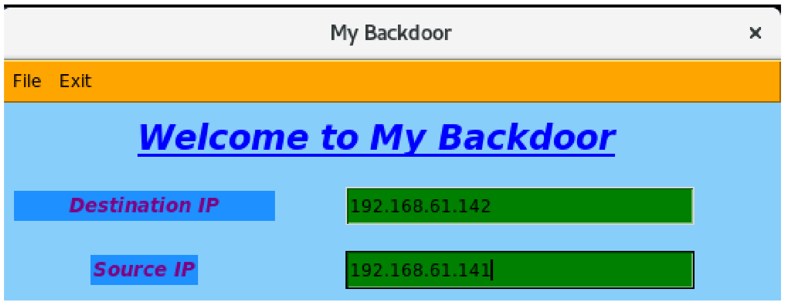
Able to type in IP destination inside IP entry



192.168.61.142

Test Case 4

Able to type in Source IP in IP entry

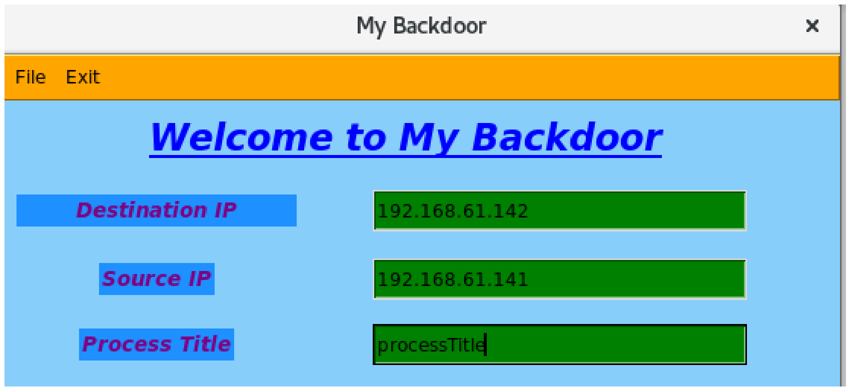


192.168.61.142

192.168.61.141

Test Case 5

Process title entry can have entered data



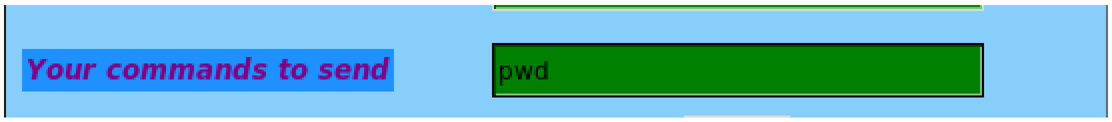
192.168.61.142

192.168.61.141

processTitle

Test Case 6

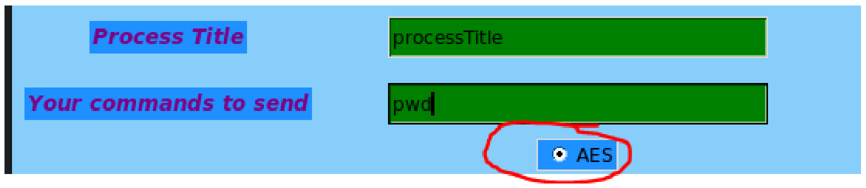
Able to type in the command box



pwd

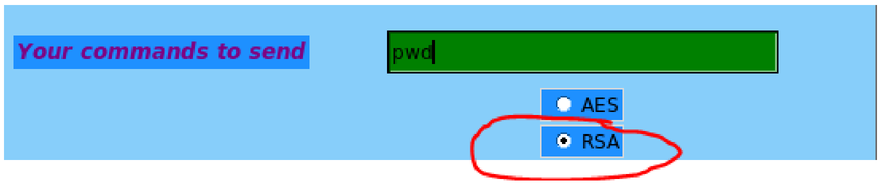
Test Case 7

User can select the AES button for encryption



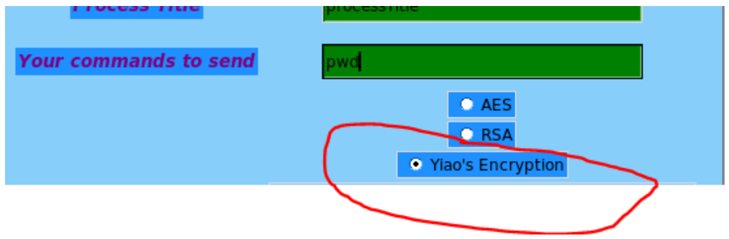
Test Case 8

User can select the RSA button for encryption



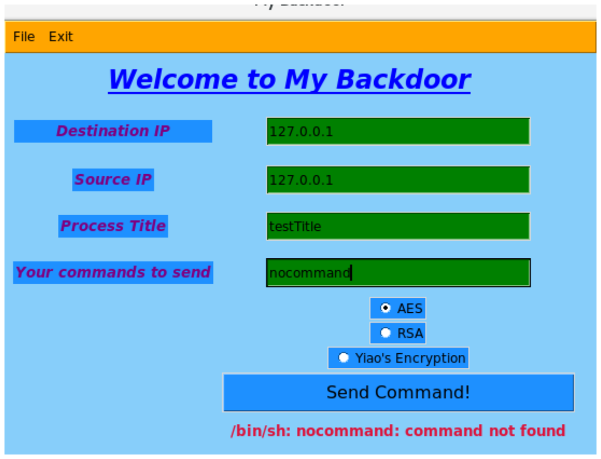
Test Case 9

User can select Yiao's Encryption radio button



Test Case 10

Sending a non-command results in the correct error



Destination IP 127.0.0.1

Source IP 127.0.0.1

Process Title testTitle

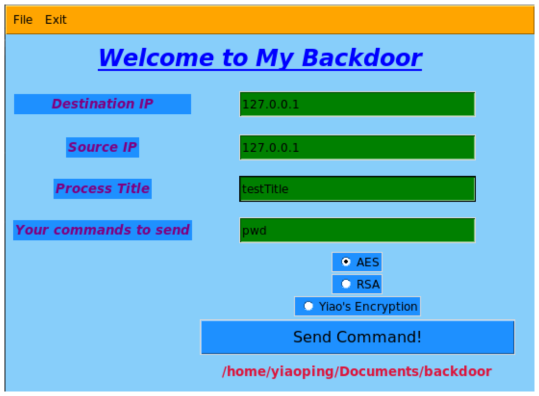
Your commands to send nocommand

[✓ AES] RSA Yiao's Encryption

```
/bin/sh: nocommand: command not found
```

Test Case 11

Sending a command gives the correct results

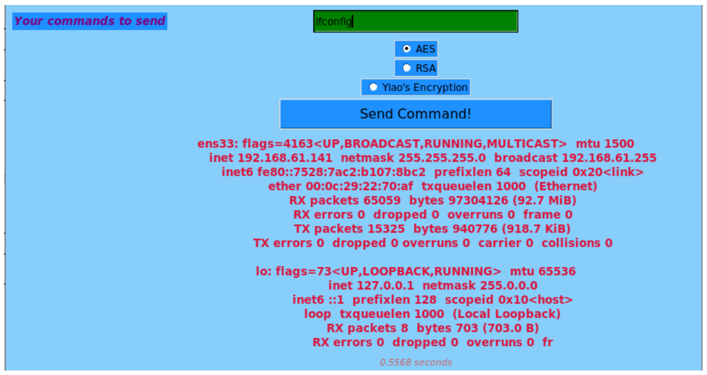


Destination IP: 127.0.0.1
Source IP: 127.0.0.1
Process Title: testTitle
Your commands to send: pwd
[✓ AES] RSA Yiao's Encryption

```
/home/yiaoping/Documents/backdoor
```

Test Case 12

Typing in ifconfig gives the correct results as shown below

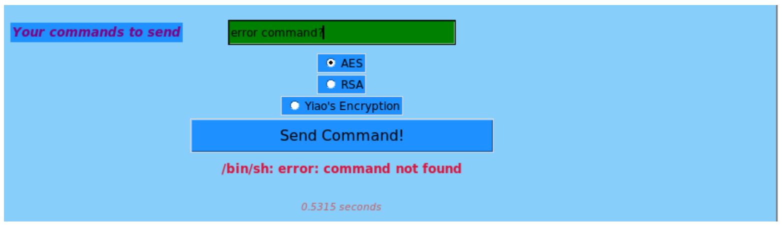


Your commands to send: ifconfig
[✓ AES] RSA Yiao's Encryption

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.61.141  netmask 255.255.255.0  broadcast 192.168.61.255
        inet6 fe80::7528:7ac2:b107:8bc2  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:22:70:af  txqueuelen 1000  (Ethernet)
        RX packets 65059  bytes 97304126 (92.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15325  bytes 940776 (918.7 KiB)
        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 8  bytes 703 (703.0 B)
        RX errors 0  dropped 0  overruns 0  fr
0.5568 seconds
```

Test Case 13

Sending an error results in no command being processed

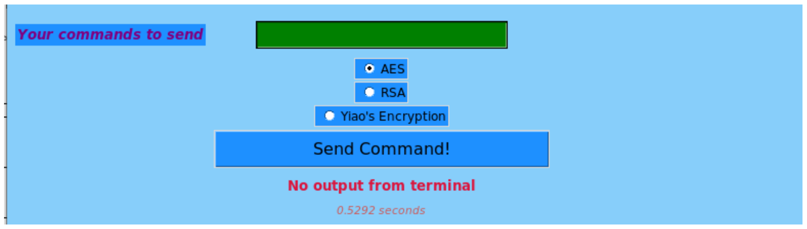


Your commands to send: error command
[✓ AES] RSA Yiao's Encryption

```
/bin/sh: error: command not found

0.5315 seconds
```

Test Case 14

Sending no command at all results in proper output



Your commands to send

[✓ AES] RSA Yiao's Encryption

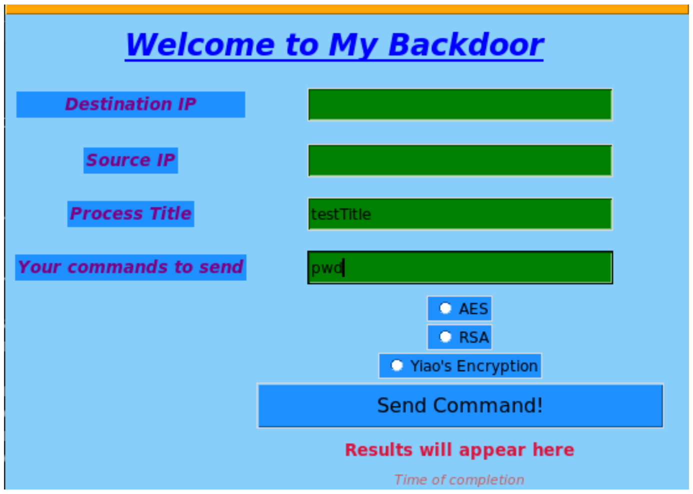
Send Command!

No output from terminal

0.5292 seconds

Test Case 15

No IP entered in results in proper output



Welcome to My Backdoor

Destination IP

Source IP

Process Title: testTitle

Your commands to send: pwd

AES RSA Yiao's Encryption

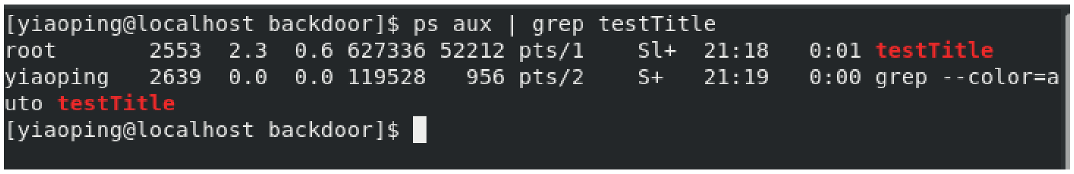
Send Command!

Results will appear here

Time of completion

Test Case 16

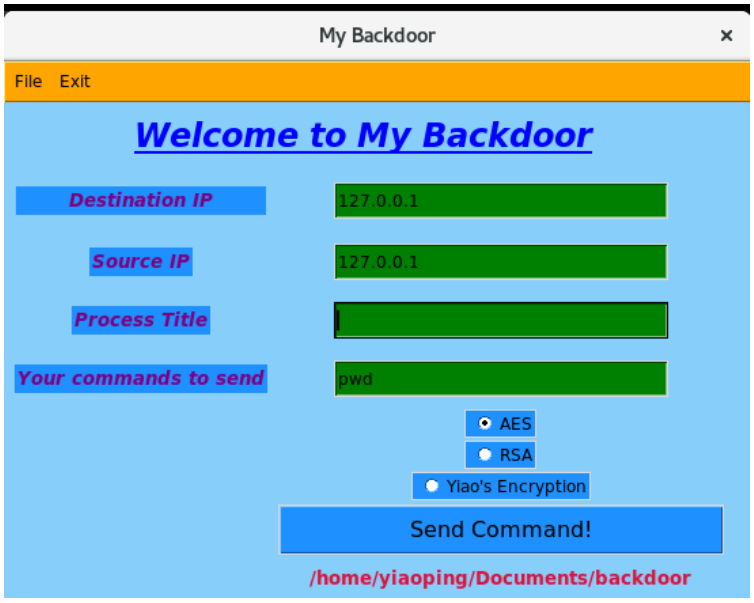
Renaming the process title from Client results in server process name change



```
[yiaoping@localhost backdoor]$ ps aux | grep testTitle
root       2553  2.3  0.6 627336 52212 pts/1    Sl+  21:18   0:01 testTitle
yiaoping   2639  0.0  0.0 119528   956 pts/2    S+   21:19   0:00 grep --color=a
uto testTitle
[yiaoping@localhost backdoor]$ 
```

Test Case 17

No process title results in regular process title name and no crash



Welcome to My Backdoor

Destination IP: 127.0.0.1

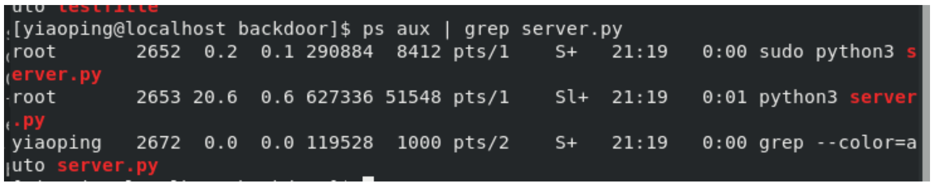
Source IP: 127.0.0.1

Process Title:

Your commands to send: pwd

[✓ AES] RSA Yiao's Encryption

/home/yiaoping/Documents/backdoor



```
uto testtitle
[yiaoping@localhost backdoor]$ ps aux | grep server.py
root       2652  0.2  0.1 290884  8412 pts/1    S+   21:19   0:00 sudo python3 s
erver.py
root       2653 20.6  0.6 627336 51548 pts/1    Sl+  21:19   0:01 python3 server
.py
yiaoping   2672  0.0  0.0 119528  1000 pts/2    S+   21:19   0:00 grep --color=a
uto server.py
```

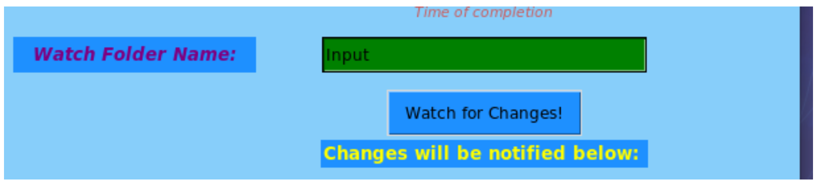
Watch for file changes

1	User can enter in any input in watch folder entry box		Pass
2	User can press watch button with folder entered in box		Pass
3	User can press watch button with no item in entry box		Pass
4	User receives notification on changes occurred in folder		Pass
5	User can successfully watch a different path of folder		Pass
6	Deletion of file successfully notified		Pass

7	Modification of file successfully notified		Pass
8	Creation of file successfully notified		Pass
9	Able to watch a different folder after already watching a folder		Pass
10	Typing in incorrect folder does not result in error		Pass
11	Typing in no folder does not result in program crash		Pass
12	Combine sending command first, then watch file		Pass
13	Creating a file with no data displays message that no data is in file		Pass
14	Creation of folder results in proper message stating folder created		Pass
15	Deletion of directory gives correct result		Pass
16	Modification of folder name gives the correct response		Pass

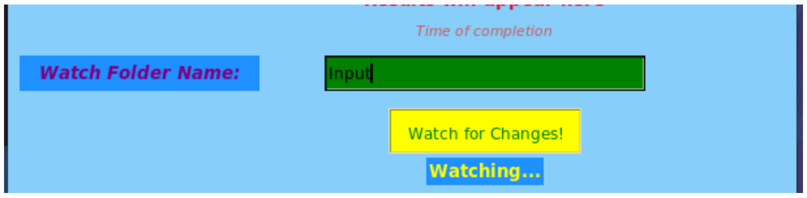
Test Case 1

Any text can be entered in the folder entry box



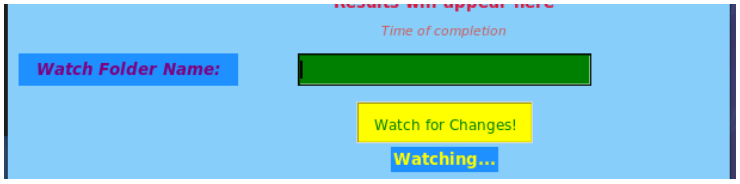
Test Case 2

User can press the watch button being notified.



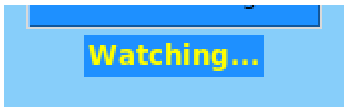
Test Case 3

No input being entered results in no crash when button is pressed in watch folder



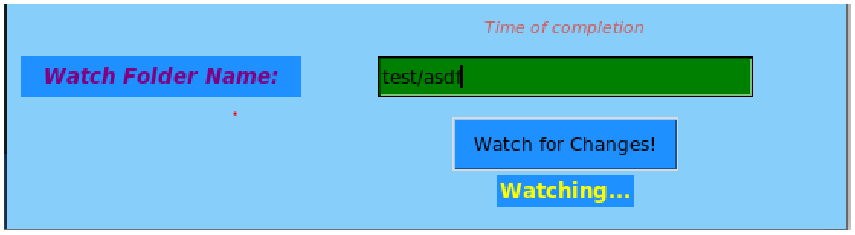
Test Case 4

Watch notification displayed to client user upon button click

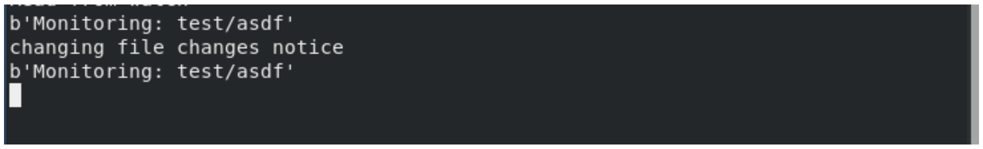


Test Case 5

User can watch for changes in a folder on server

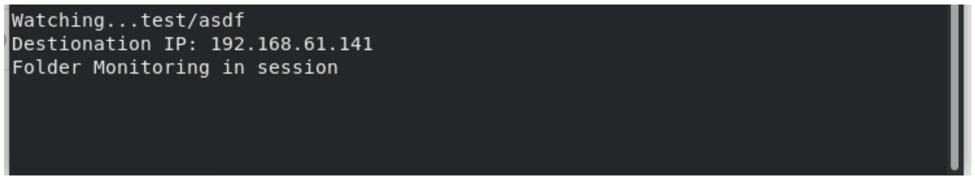


Client Side:



```
b'Monitoring: test/asdf'
changing file changes notice
b'Monitoring: test/asdf'
```

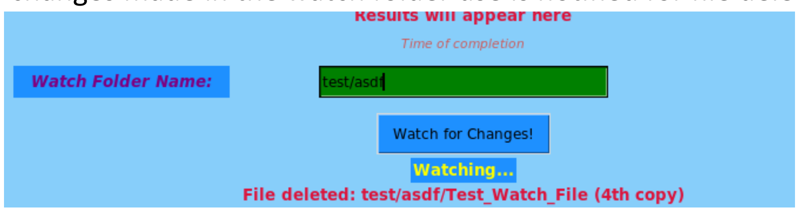
Server Side:



```
Watching...test/asdf
Destionation IP: 192.168.61.141
Folder Monitoring in session
```

Test Case 6

Any changes made in the watch folder use is notified for file deletion



Results will appear here

Time of completion

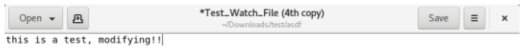
test/asdf

Watching...

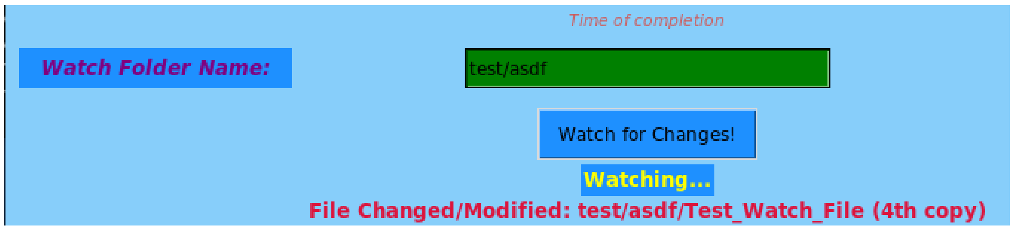
File deleted: test/asdf/Test_Watch_File (4th copy)

Test Case 7

Modification of file notifies the client user



```
this is a test, modifying!!
```



Time of completion

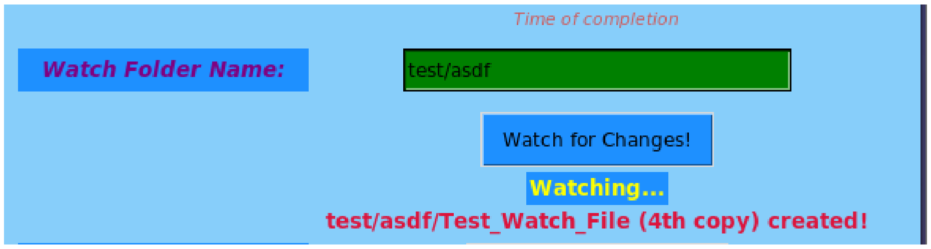
test/asdf

Watching...

File Changed/Modified: test/asdf/Test_Watch_File (4th copy)

Test Case 8

Creation of file notifies the client user


Time of completion

Watch Folder Name: test/asdf

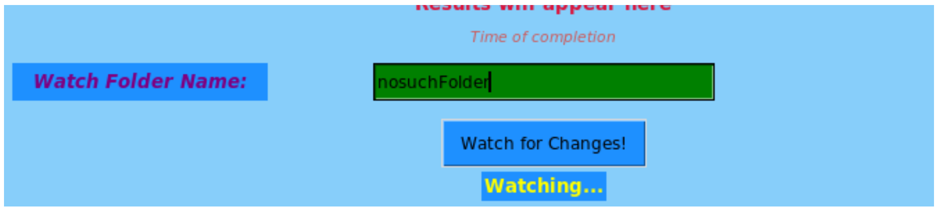
Watch for Changes!

Watching...

test/asdf/Test_Watch_File (4th copy) created!

Test Case 10

Typing in no such folder results in proper message and no crash


Results will appear here

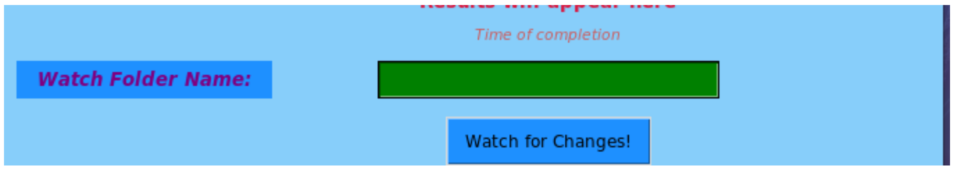
Time of completion

Watch Folder Name: nosuchFolder

Watch for Changes!

Watching...

Test Case 11


Results will appear here

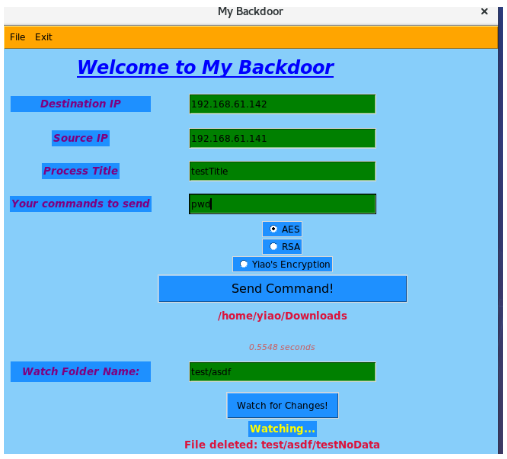
Time of completion

Watch Folder Name:

Watch for Changes!

Test Case 12

Combination of sending command and watching file works properly


My Backdoor

File Exit

Welcome to My Backdoor

Destination IP: 192.168.61.142

Source IP: 192.168.61.141

Process Title: testTitle

Your commands to send: pwd

[✓ AES] RSA Yiao's Encryption

Send Command!

/home/yiao/Downloads

0.5548 seconds

Watch Folder Name: test/asdf

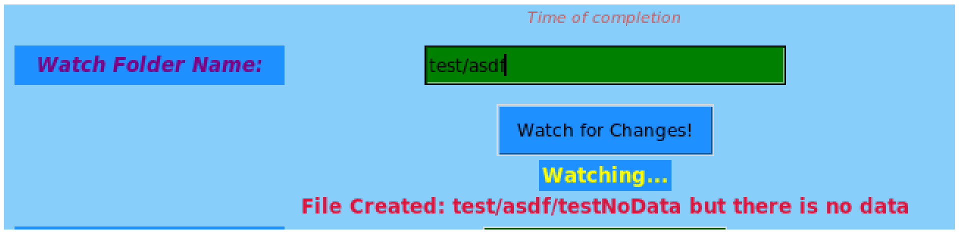
Watch for Changes!

Watching...

File deleted: test/asdf/testNoData

Test Case 13

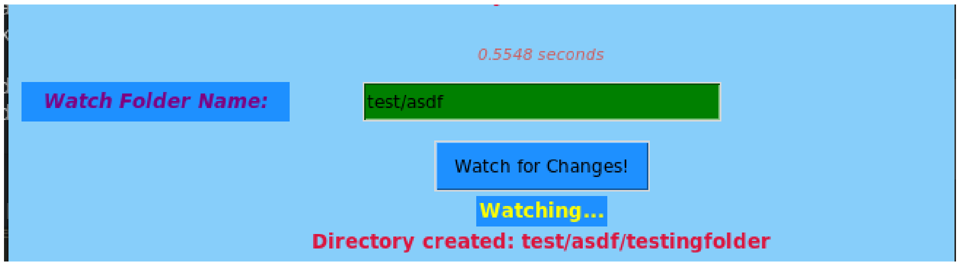
Creation of empty file results in correct message



File Created: test/asdf/testNoData but there is no data

Test Case 14

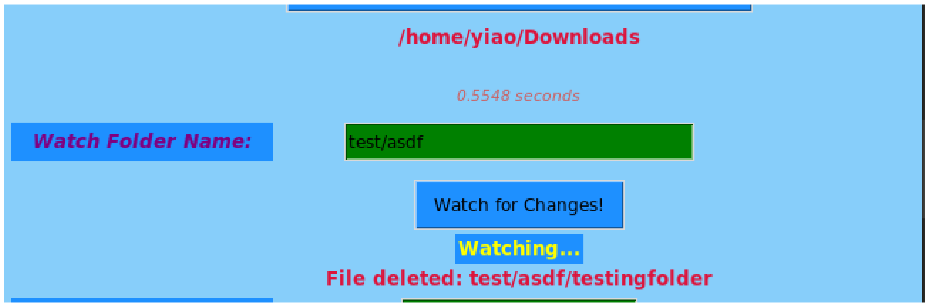
Creating a folder results in proper client message of directory created



Directory created: test/asdf/testingfolder

Test Case 15

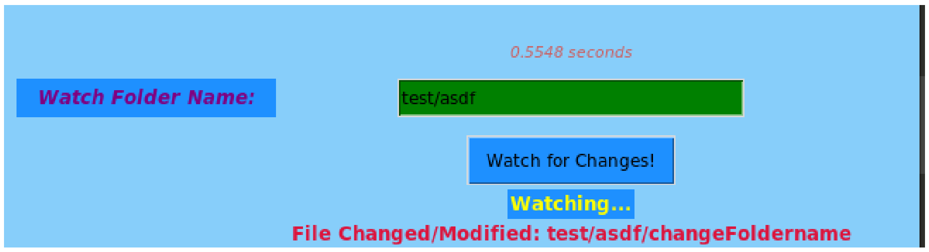
Deletion of directory results in proper client message of folder creation



File deleted: test/asdf/testingfolder

Test Case 16

Modification of folder name results in proper name



0.5548 seconds

test/asdf

Watching...

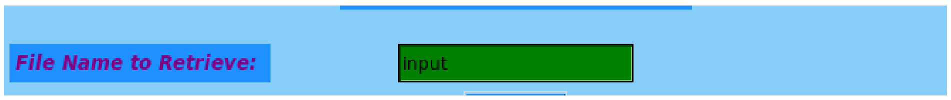
File Changed/Modified: test/asdf/changeFoldername

File Retrieval

1	Any input can be entered in Get file entry box		Pass
2	Upon entering in successful file, user can press get file button		Pass
3	No input is entered in get file. Pressing get file button results in correct error displayed to user		Pass
4	Typing in a non-existent file does not result in error		Pass
5	User can grab any number of files consecutively		Pass
6	If server is not running and user attempts to grab file, client application does not crash		Pass
7	User can use a combination of get file along with watch file at same time		Pass
8	User can use combination of get file and send commands to server at same time		Pass

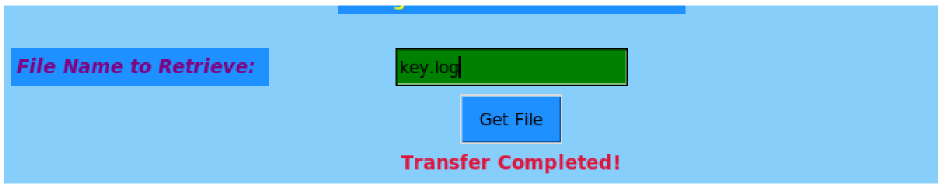
Test Case 1

User can type in any entry in the input box of file retrieval



Test Case 2

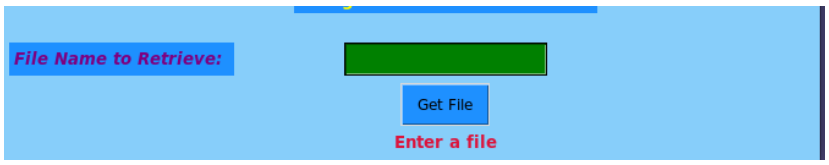
User successfully presses button to get file retrieval



Transfer Completed!

Test Case 3

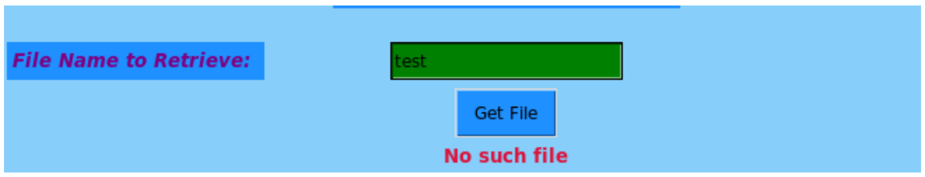
Entering no file in results in proper error message



Enter a file

Test Case 4

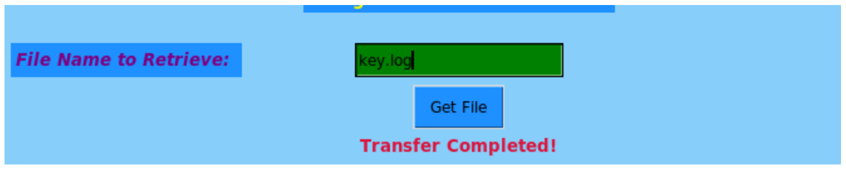
If no such file exists, correct error resulted

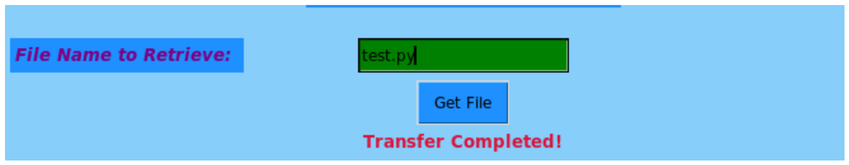


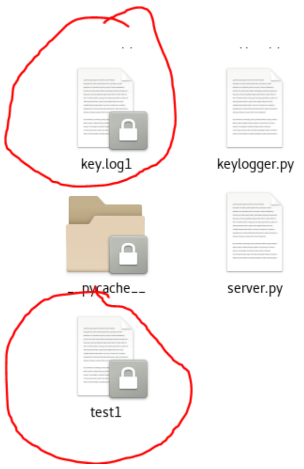
No such file

Test Case 5

User can get any consecutive number of file retrieval. Key.log is grabbed first, then test1.

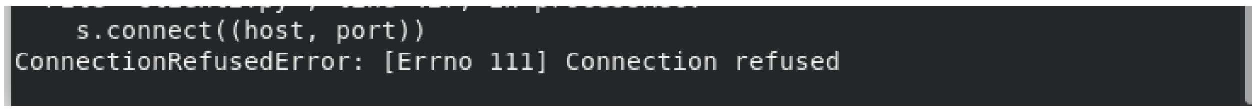






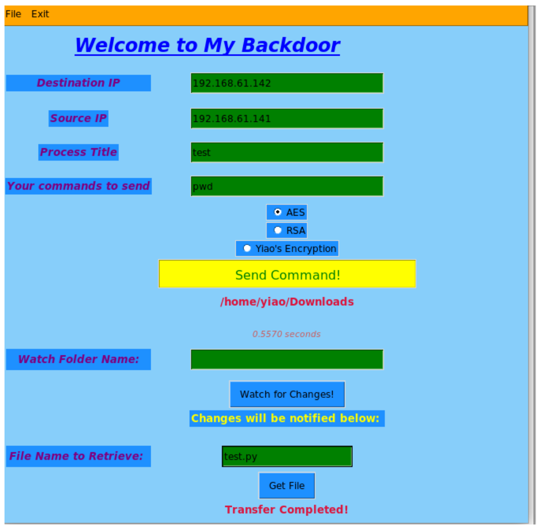
Test Case 6

User is unable to connect if server is not running, but client application does not crash



```
    s.connect((host, port))
ConnectionRefusedError: [Errno 111] Connection refused
```

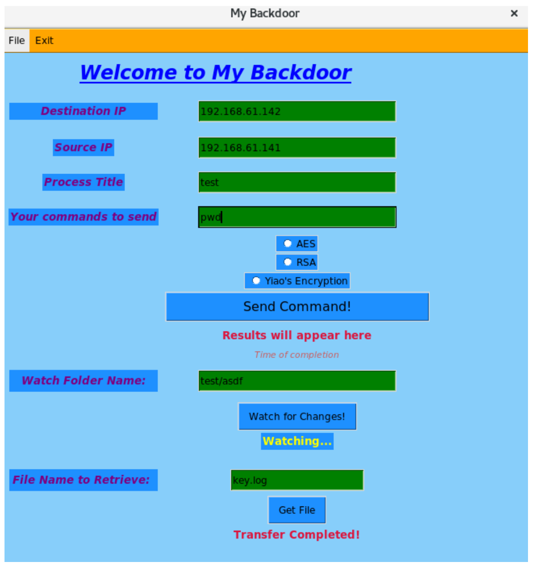
Client still running:



Destination IP: 192.168.61.142
Source IP: 192.168.61.141
Process Title: test
Your commands to send: pwd
[✓ AES] RSA Yiao's Encryption
/home/yiao/Downloads
0.5570 seconds
Watch Folder Name:
Changes will be notified below:
File Name to Retrieve: test.py
Transfer Completed!

Test Case 7

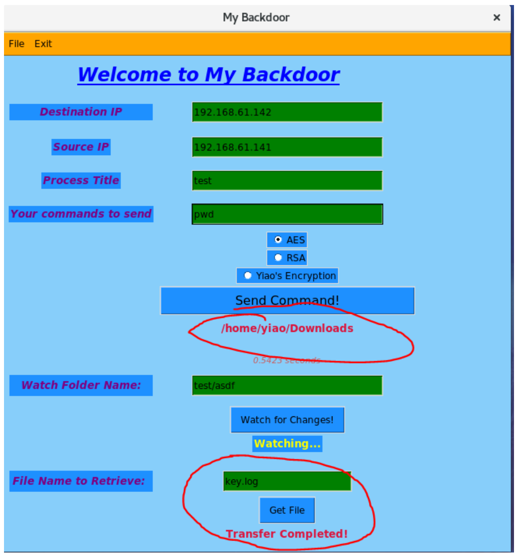
Grab file works consecutively after using the watch folder feature.



Destination IP: 192.168.61.142
Source IP: 192.168.61.141
Process Title: test
Your commands to send: pwd
AES RSA Yiao's Encryption
Results will appear here
Time of completion
Watch Folder Name: test/asdf
Watching...
File Name to Retrieve: key.log
Transfer Completed!

Test Case 8

User can successfully use all 3 features of watch folder, send a command, and retrieve a file.



Destination IP: 192.168.61.142
Source IP: 192.168.61.141
Process Title: test
Your commands to send: pwd
[✓ AES] RSA Yiao's Encryption

/home/yiao/Downloads
0.5423 seconds

Watch Folder Name: test/asdf

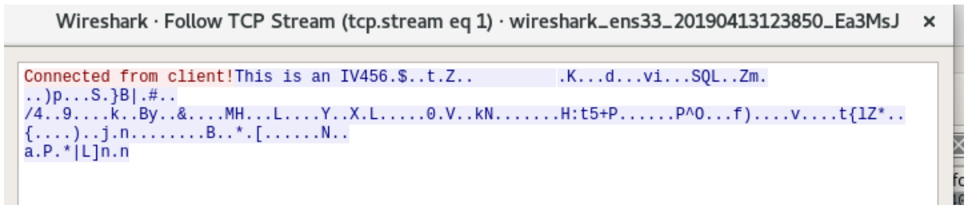
Watching...

File Name to Retrieve: key.log

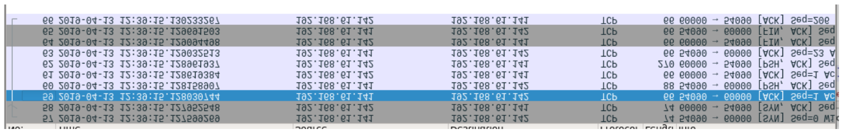
Transfer Completed!

Wireshark Captures:

TCP connection established and transfer of data made.


Wireshark · Follow TCP Stream (tcp.stream eq 1) · wireshark_ens33_20190413123850_Ea3MsJ

```
Connected from client!This is an IV456.$..t.Z..          .K...d...vi...SQL..Zm.
..)p...S.}B|.#..
/4..9....k..By..&....MH...L....Y..X.L.....0.V..kN.......H:t5+P......P^O...f)....v....t{1Z*..
{....)..j.n........B..*.[......N..
a.P.*|L]n.n
```



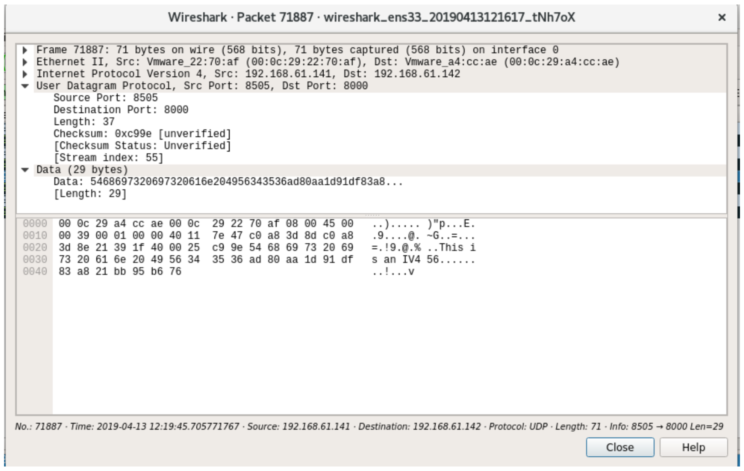
Encryption Tests

1	Check in Wireshark to ensure commands are encrypted with AES upon selecting AES option		Pass
2	Upon selecting RSA option, check to ensure commands are encrypted in RSA		Pass

3	Upon selecting Yiao's encryption, check Wireshark to ensure commands are encrypted		Pass
4	Do not click a radio button for encryption. Send the commands with no encryption selected. Data should be automatically encrypted in AES		Pass
5	Data is properly decrypted by AES on server side		Pass
6	Data is properly decrypted by RSA on server side		Pass
7	Data is properly decrypted by Yiao's encryption server side		Pass
8	Encryption with AES for watch folder name		Pass
9	Encryption with AES for file retrieval		Pass
10	Encryption with AES on server side for return of information		Pass

Test Case 1

Below shows the screenshot of a Wireshark capture between the server and the client. The source and destination IP address are displayed, with the packet data displayed below.

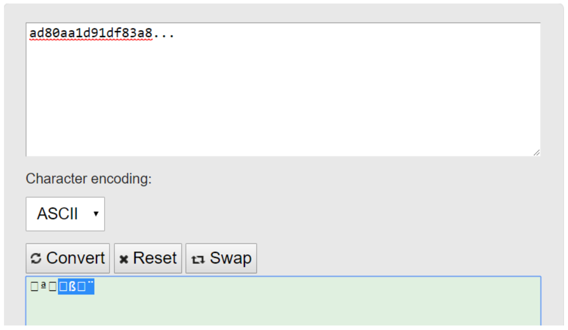


Wireshark · Packet 71887 · wireshark_ens33_20190413121617_tNh7oX

```
▸ Frame 71887: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0
▸ Ethernet II, Src: Vmware_22:70:af (00:0c:29:22:70:af), Dst: Vmware_a4:cc:ae (00:0c:29:a4:cc:ae)
▸ Internet Protocol Version 4, Src: 192.168.61.141, Dst: 192.168.61.142
▾ User Datagram Protocol, Src Port: 8505, Dst Port: 8000
    Source Port: 8505
    Destination Port: 8000
    Length: 37
    Checksum: 0xc99e [unverified]
    [Checksum Status: Unverified]
    [Stream index: 55]
▾ Data (29 bytes)
    Data: 546869732069732061 6e204956343536ad80aa1d91df83a8...
    [Length: 29]

0000  00 0c 29 a4 cc ae 00 0c  29 22 70 af 08 00 45 00   ..)..... )"p...E.
0010  00 39 00 01 00 00 40 11  7e 47 c0 a8 3d 8d c0 a8   .9....@. ~G..=...
0020  3d 8e 21 39 1f 40 00 25  c9 9e 54 68 69 73 20 69   =.!9.@.% ..This i
0030  73 20 61 6e 20 49 56 34  35 36 ad 80 aa 1d 91 df   s an IV4 56......
0040  83 a8 21 bb 95 b6 76                               ..!...v
```

No.: 71887 · Time: 2019-04-13 12:19:45.705771767 · Source: 192.168.61.141 · Destination: 192.168.61.142 · Protocol: UDP · Length: 71 · Info: 8505 → 8000 Len=29

After the IV, if we attempt to decrypt the information from hexadecimal to ascii, we get the following:



```
ad80aa1d91df83a8...
```

Character encoding: ASCII

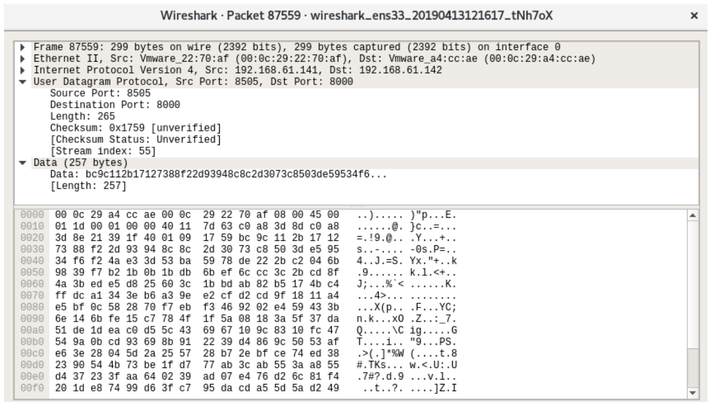
```
□ª□□ß□¨
```

As you can see, the above shows AES encrypted information.

```
We want AES encryption!
b'This is an IV456\xad\x80\xaa\x1d\x91\xdf\x83\xa8!\xbb\x95\xb6v'
```

Test Case 2

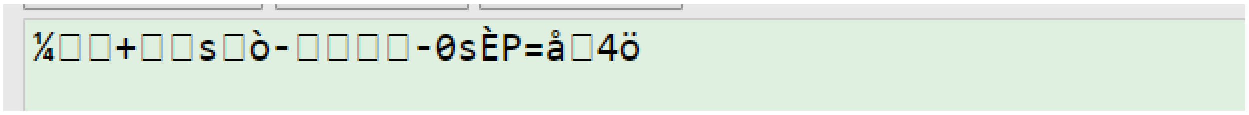
Same as before, the below shows RSA encryption



Wireshark · Packet 87559 · wireshark_ens33_20190413121617_tNh7oX

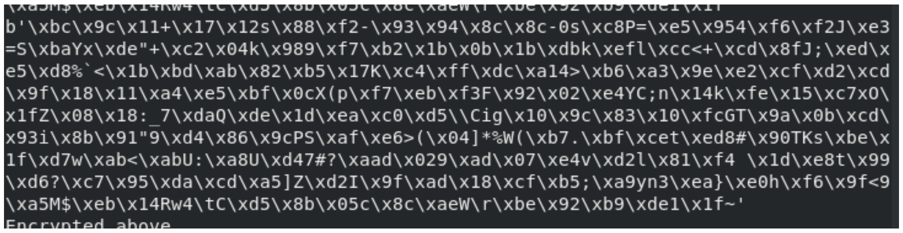
```
▸ Frame 87559: 299 bytes on wire (2392 bits), 299 bytes captured (2392 bits) on interface 0
▸ Ethernet II, Src: Vmware_22:70:af (00:0c:29:22:70:af), Dst: Vmware_a4:cc:ae (00:0c:29:a4:cc:ae)
▸ Internet Protocol Version 4, Src: 192.168.61.141, Dst: 192.168.61.142
▾ User Datagram Protocol, Src Port: 8505, Dst Port: 8000
    Source Port: 8505
    Destination Port: 8000
    Length: 265
    Checksum: 0x1759 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 55]
▾ Data (257 bytes)
    Data: bc9c112b17127388f22d93948c8c2d3073c8503de59534f6...
    [Length: 257]

0000  00 0c 29 a4 cc ae 00 0c  29 22 70 af 08 00 45 00   ..)..... )"p...E.
0010  01 1d 00 01 00 00 40 11  7d 63 c0 a8 3d 8d c0 a8   ......@. }c..=...
0020  3d 8e 21 39 1f 40 01 09  17 59 bc 9c 11 2b 17 12   =.!9.@.. .Y...+..
0030  73 88 f2 2d 93 94 8c 8c  2d 30 73 c8 50 3d e5 95   s..-.... -0s.P=..
0040  34 f6 f2 4a e3 3d 53 ba  59 78 de 22 2b c2 04 6b   4..J.=S. Yx."+..k
0050  98 39 f7 b2 1b 0b 1b db  6b ef 6c cc 3c 2b cd 8f   .9...... k.l.<+..
0060  4a 3b ed e5 d8 25 60 3c  1b bd ab 82 b5 17 4b c4   J;...%`< ......K.
0070  ff dc a1 34 3e b6 a3 9e  e2 cf d2 cd 9f 18 11 a4   ...4>... ........
0080  e5 bf 0c 58 28 70 f7 eb  f3 46 92 02 e4 59 43 3b   ...X(p.. .F...YC;
0090  6e 14 6b fe 15 c7 78 4f  1f 5a 08 18 3a 5f 37 da   n.k...xO .Z..:_7.
00a0  51 de 1d ea c0 d5 5c 43  69 67 10 9c 83 10 fc 47   Q.....\C ig.....G
00b0  54 9a 0b cd 93 69 8b 91  22 39 d4 86 9c 50 53 af   T....i.. "9...PS.
00c0  e6 3e 28 04 5d 2a 25 57  28 b7 2e bf ce 74 ed 38   .>(.]*%W (....t.8
00d0  23 90 54 4b 73 be 1f d7  77 ab 3c ab 55 3a a8 55   #.TKs... w.<.U:.U
00e0  d4 37 23 3f aa 64 02 39  ad 07 e4 76 d2 6c 81 f4   .7#?.d.9 ...v.l..
00f0  20 1d e8 74 99 d6 3f c7  95 da cd a5 5d 5a d2 49    ..t..?. ....]Z.I
```

Attempting to decode this give us the following:



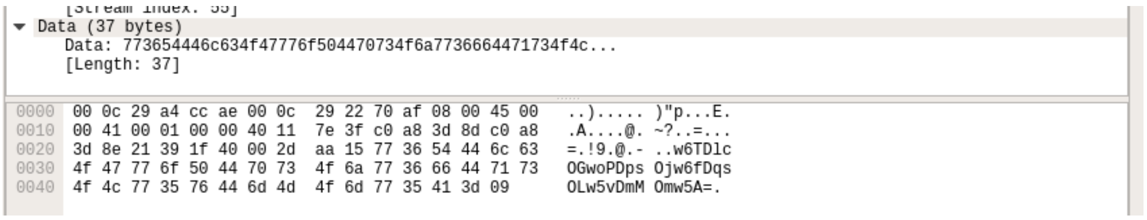
```
¼□□+□□s□ò-□□□□-0sÈP=å□4ö
```

Client encrypted RSA



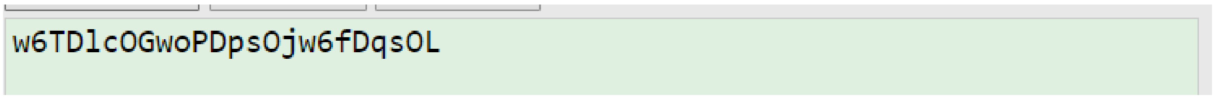
```
b'\xbc\x9c\x11+\x17\x12s\x88\xf2-\x93\x94\x8c\x8c-0s\xc8P=\xe5\x954\xf6\xf2J\xe3
=S\xbaYx\xde"+\xc2\x04k\x989\xf7\xb2\x1b\x0b\x1b\xdbk\xefl\xcc<+\xcd\x8fJ;\xed\x
e5\xd8%`<\x1b\xbd\xab\x82\xb5\x17K\xc4\xff\xdc\xa14>\xb6\xa3\x9e\xe2\xcf\xd2\xcd
\x9f\x18\x11\xa4\xe5\xbf\x0cX(p\xf7\xeb\xf3F\x92\x02\xe4YC;n\x14k\xfe\x15\xc7xO\
x1fZ\x08\x18:_7\xdaQ\xde\x1d\xea\xc0\xd5\\Cig\x10\x9c\x83\x10\xfcGT\x9a\x0b\xcd\
x93i\x8b\x91"9\xd4\x86\x9cPS\xaf\xe6>(\x04]*%W(\xb7.\xbf\xcet\xed8#\x90TKs\xbe\x
1f\xd7w\xab<\xabU:\xa8U\xd47#?\xaad\x029\xad\x07\xe4v\xd2l\x81\xf4 \x1d\xe8t\x99
\xd6?\xc7\x95\xda\xcd\xa5]Z\xd2I\x9f\xad\x18\xcf\xb5;\xa9yn3\xea}\xe0h\xf6\x9f<9
\xa5M$\xeb\x14Rw4\tC\xd5\x8b\x05c\x8c\xaeW\r\xbe\x92\xb9\xde1\x1f~'
```

Test Case 3



```
▾ Data (37 bytes)
    Data: 773654446c634f47776f504470734f6a7736664471734f4c...
    [Length: 37]

0000  00 0c 29 a4 cc ae 00 0c  29 22 70 af 08 00 45 00   ..)..... )"p...E.
0010  00 41 00 01 00 00 40 11  7e 3f c0 a8 3d 8d c0 a8   .A....@. ~?..=...
0020  3d 8e 21 39 1f 40 00 2d  aa 15 77 36 54 44 6c 63   =.!9.@.- ..w6TDlc
0030  4f 47 77 6f 50 44 70 73  4f 6a 77 36 66 44 71 73   OGwoPDps Ojw6fDqs
0040  4f 4c 77 35 76 44 6d 4d  4f 6d 77 35 41 3d 09      OLw5vDmM Omw5A=.
```

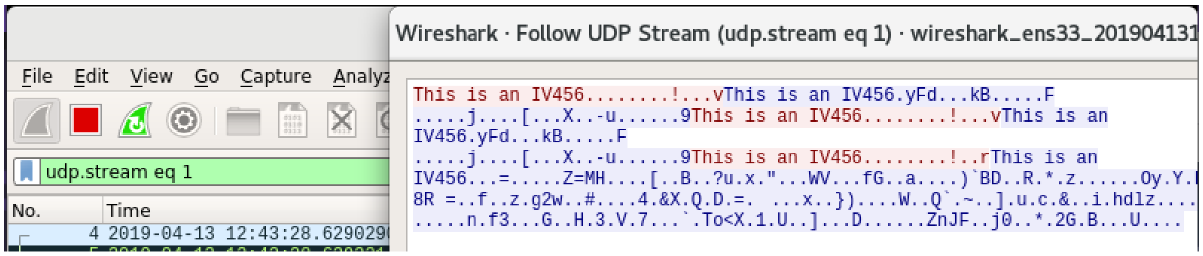
As suspected, attempting to decrypt a transformation algorithm will give us proper letters and numbers as opposed to encrypted AES and RSA information.



```
w6TDlcOGwoPDpsOjw6fDqsOL
```

Test Case 4

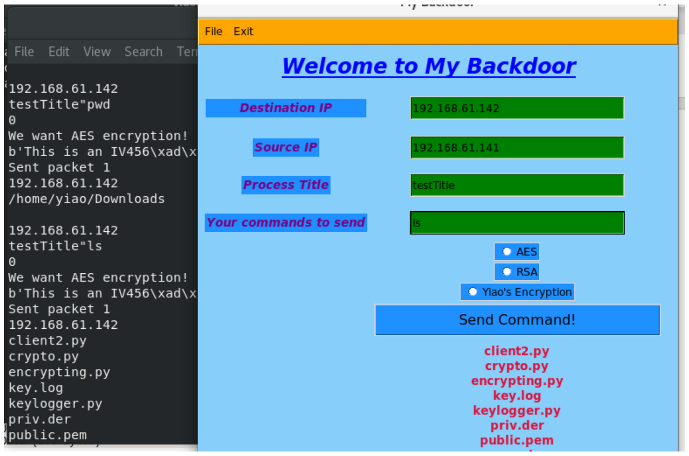
Wireshark capture showing AES encryption when no radio button is selected



Wireshark · Follow UDP Stream (udp.stream eq 1) · wireshark_ens33_20190413

```
This is an IV456........!...vThis is an IV456.yFd...kB.....F
.....j....[...X..-u......9This is an IV456........!...vThis is an
IV456.yFd...kB.....F
.....j....[...X..-u......9This is an IV456........!..rThis is an
IV456...=.....Z=MH....[..B..?u.x."...WV...fG..a....)`BD..R.*.z......Oy.Y.
8R =..f..z.g2w..#....4.&X.Q.D.=.  ...x..})....W..Q`.~..].u.c.&..i.hdlz....
.....n.f3...G..H.3.V.7...`.To<X.1.U..]...D......ZnJF..j0..*.2G.B...U....
```

Client UI application showing no radio button selected but AES encryption is chosen by default, shown in the background terminal.



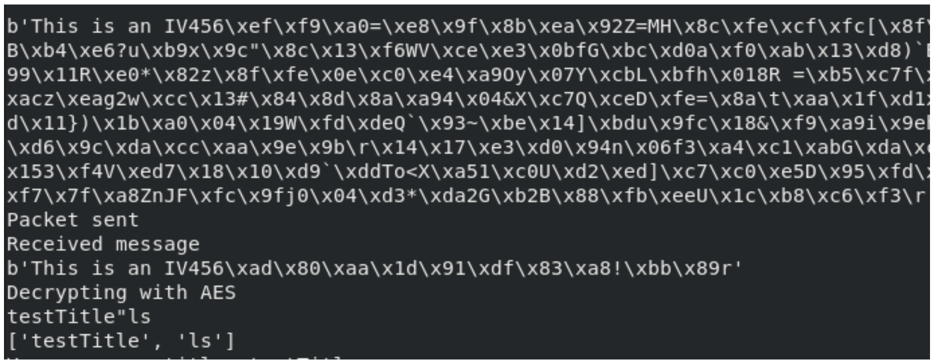
```
192.168.61.142
testTitle"pwd
0
We want AES encryption!
b'This is an IV456\xad\x
Sent packet 1
192.168.61.142
/home/yiao/Downloads

192.168.61.142
testTitle"ls
0
We want AES encryption!
b'This is an IV456\xad\x
Sent packet 1
192.168.61.142
client2.py
crypto.py
encrypting.py
key.log
keylogger.py
priv.der
public.pem
```

Welcome to My Backdoor

Destination IP: 192.168.61.142
Source IP: 192.168.61.141
Process Title: testTitle
Your commands to send: ls
AES
RSA
Yiao's Encryption

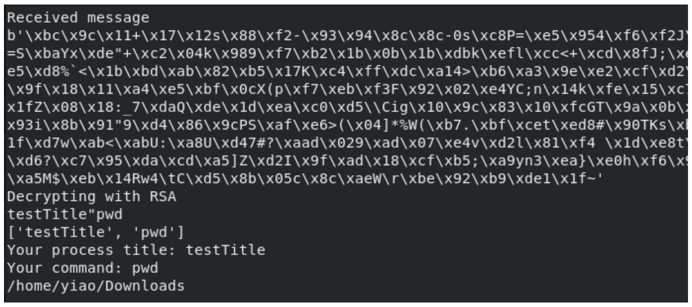
client2.py
crypto.py
encrypting.py
key.log
keylogger.py
priv.der
public.pem

Test Case 5



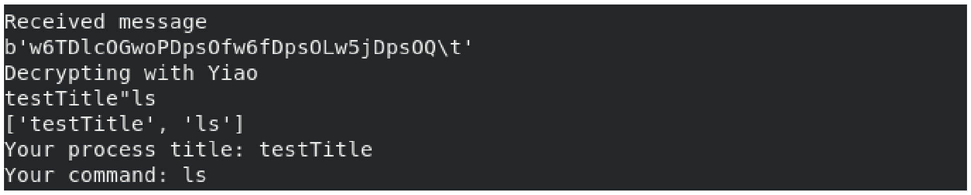
```
b'This is an IV456\xef\xf9\xa0=\xe8\x9f\x8b\xea\x92Z=MH\x8c\xfe\xcf\xfc[\x8f
B\xb4\xe6?u\xb9x\x9c"\x8c\x13\xf6WV\xce\xe3\x0bfG\xbc\xd0a\xf0\xab\x13\xd8)`
99\x11R\xe0*\x82z\x8f\xfe\x0e\xc0\xe4\xa9Oy\x07Y\xcbL\xbfh\x018R =\xb5\xc7f\
xacz\xeag2w\xcc\x13#\x84\x8d\x8a\xa94\x04&X\xc7Q\xceD\xfe=\x8a\t\xaa\x1f\xd1
d\x11})\x1b\xa0\x04\x19W\xfd\xdeQ`\x93~\xbe\x14]\xbdu\x9fc\x18&\xf9\xa9i\x9e
\xd6\x9c\xda\xcc\xaa\x9e\x9b\r\x14\x17\xe3\xd0\x94n\x06f3\xa4\xc1\xabG\xda\x
x153\xf4V\xed7\x18\x10\xd9`\xddTo<X\xa51\xc0U\xd2\xed]\xc7\xc0\xe5D\x95\xfd\
xf7\x7f\xa8ZnJF\xfc\x9fj0\x04\xd3*\xda2G\xb2B\x88\xfb\xeeU\x1c\xb8\xc6\xf3\r
Packet sent
Received message
b'This is an IV456\xad\x80\xaa\x1d\x91\xdf\x83\xa8!\xbb\x89r'
Decrypting with AES
testTitle"ls
['testTitle', 'ls']
```

Test Case 6



```
Received message
b'\xbc\x9c\x11+\x17\x12s\x88\xf2-\x93\x94\x8c\x8c-0s\xc8P=\xe5\x954\xf6\xf2J
=S\xbaYx\xde"+\xc2\x04k\x989\xf7\xb2\x1b\x0b\x1b\xdbk\xefl\xcc<+\xcd\x8fJ;\x
e5\xd8%`<\x1b\xbd\xab\x82\xb5\x17K\xc4\xff\xdc\xa14>\xb6\xa3\x9e\xe2\xcf\xd2
\x9f\x18\x11\xa4\xe5\xbf\x0cX(p\xf7\xeb\xf3F\x92\x02\xe4YC;n\x14k\xfe\x15\xc
x1fZ\x08\x18:_7\xdaQ\xde\x1d\xea\xc0\xd5\\Cig\x10\x9c\x83\x10\xfcGT\x9a\x0b\
x93i\x8b\x91"9\xd4\x86\x9cPS\xaf\xe6>(\x04]*%W(\xb7.\xbf\xcet\xed8#\x90TKs\x
1f\xd7w\xab<\xabU:\xa8U\xd47#?\xaad\x029\xad\x07\xe4v\xd2l\x81\xf4 \x1d\xe8t
\xd6?\xc7\x95\xda\xcd\xa5]Z\xd2I\x9f\xad\x18\xcf\xb5;\xa9yn3\xea}\xe0h\xf6\x
\xa5M$\xeb\x14Rw4\tC\xd5\x8b\x05c\x8c\xaeW\r\xbe\x92\xb9\xde1\x1f~'
Decrypting with RSA
testTitle"pwd
['testTitle', 'pwd']
Your process title: testTitle
Your command: pwd
/home/yiao/Downloads
```

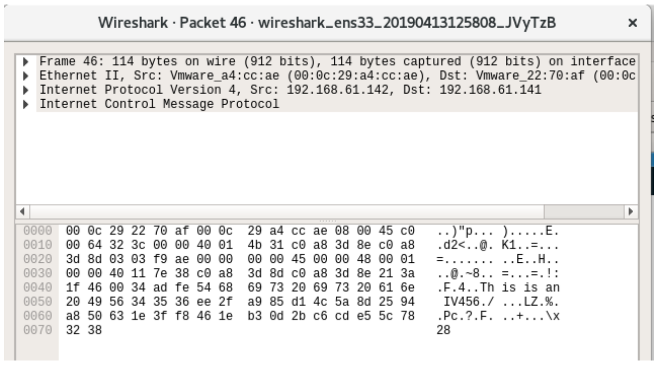
Test Case 7



```
Received message
b'w6TDlcOGwoPDpsOfw6fDpsOLw5jDpsOQ\t'
Decrypting with Yiao
testTitle"ls
['testTitle', 'ls']
Your process title: testTitle
Your command: ls
```

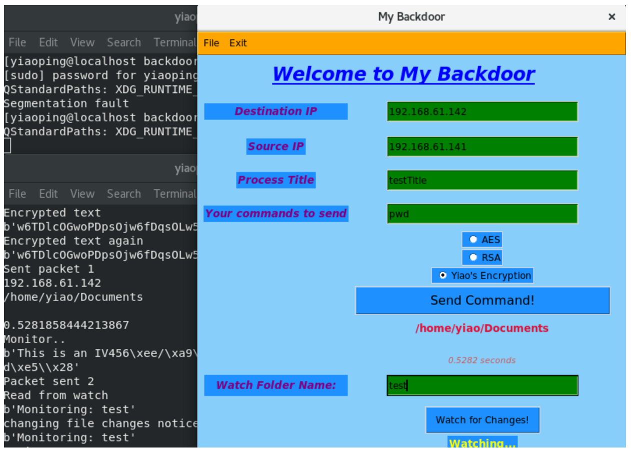
Test Case 8

AES encryption for watching a folder name



```
Wireshark · Packet 46 · wireshark_ens33_20190413125808_JVyTzB

Frame 46: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface
Ethernet II, Src: Vmware_a4:cc:ae (00:0c:29:a4:cc:ae), Dst: Vmware_22:70:af (00:0c
Internet Protocol Version 4, Src: 192.168.61.142, Dst: 192.168.61.141
Internet Control Message Protocol

0000  00 0c 29 22 70 af 00 0c  29 a4 cc ae 08 00 45 c0   ..)"p... ).....E.
0010  00 64 32 3c 00 00 40 01  4b 31 c0 a8 3d 8e c0 a8   .d2<..@. K1..=...
0020  3d 8d 03 03 f9 ae 00 00  00 00 45 00 00 48 00 01   =....... ..E..H..
0030  00 00 40 11 7e 38 c0 a8  3d 8d c0 a8 3d 8e 21 3a   ..@.~8.. =...=.!:
0040  1f 46 00 34 ad fe 54 68  69 73 20 69 73 20 61 6e   .F.4..Th is is an
0050  20 49 56 34 35 36 ee 2f  a9 85 d1 4c 5a 8d 25 94    IV456./ ...LZ.%.
0060  a8 50 63 1e 3f f8 46 1e  b3 0d 2b c6 cd e5 5c 78   .Pc.?.F. ..+...\x
0070  32 38                                              28
```

Below shows watching for folder being encrypted in AES



```
[yiaoping@localhost backdoor
[sudo] password for yiaoping
QStandardPaths: XDG_RUNTIME_
Segmentation fault
[yiaoping@localhost backdoor
QStandardPaths: XDG_RUNTIME_
```

```
Encrypted text
b'w6TDlcOGwoPDpsOjw6fDqsOLw5
Encrypted text again
b'w6TDlcOGwoPDpsOjw6fDqsOLw5
Sent packet 1
192.168.61.142
/home/yiao/Documents

0.5281858444213867
Monitor..
b'This is an IV456\xee/\xa9\
d\xe5\\x28'
Packet sent 2
Read from watch
b'Monitoring: test'
changing file changes notice
b'Monitoring: test'
```

My Backdoor

Welcome to My Backdoor

Destination IP: 192.168.61.142

Source IP: 192.168.61.141

Process Title: testTitle

Your commands to send: pwd

AES [ ] RSA [ ] [✓ Yiao's Encryption]

/home/yiao/Documents

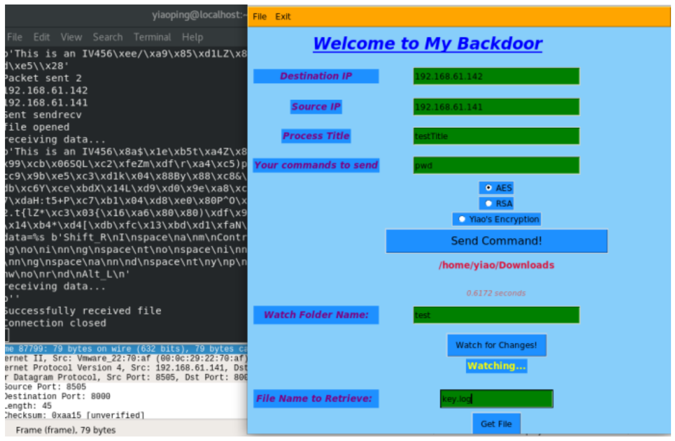
0.5282 seconds

Watch Folder Name: test

Watching...

Test Case 9

Below shows the client retrieving the key.log file, and in the background the terminal displays the encrypted information in AES, receives the information, and decrypts it and writes it to file.

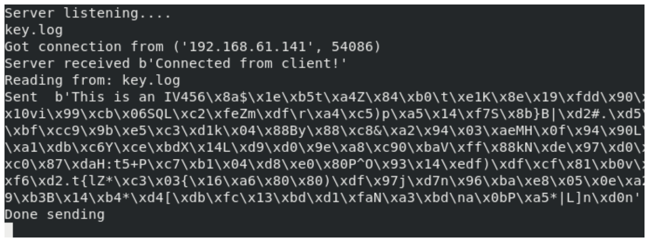


```
b'This is an IV456\xee/\xa9\x85\xd1LZ\x8
d\xe5\\x28'
Packet sent 2
192.168.61.142
192.168.61.141
Sent sendrecv
file opened
receiving data...
b'This is an IV456\x8a$\x1e\xb5t\xa4Z\x8
x99\xcb\x06SQL\xc2\xfeZm\xdf\r\xa4\xc5)p
cc9\x9b\xe5\xc3\xd1k\x04\x88By\x88\xc8&\
db\xc6Y\xce\xbdX\x14L\xd9\xd0\x9e\xa8\xc
7\xdaH:t5+P\xc7\xb1\x04\xd8\xe0\x80P^O\x
2.t{lZ*\xc3\x03{\x16\xa6\x80\x80)\xdf\x9
\x14\xb4*\xd4[\xdb\xfc\x13\xbd\xd1\xfaN\
data=%s b'Shift_R\nI\nspace\na\nm\nContr
ng\no\ni\nn\ng\nspace\nt\no\nspace\ni\nn
\nn\ng\nspace\na\nn\nd\nspace\nt\ny\np\n
nw\no\nr\nd\nAlt_L\n'
receiving data...
b''
Successfully received file
Connection closed

me 87799: 79 bytes on wire (632 bits), 79 bytes ca
ernet II, Src: Vmware_22:70:af (00:0c:29:22:70:af)
ernet Protocol Version 4, Src: 192.168.61.141, Dst
r Datagram Protocol, Src Port: 8505, Dst Port: 800
Source Port: 8505
Destination Port: 8000
Length: 45
Checksum: 0xaa15 [unverified]
Frame (frame), 79 bytes
```

Welcome to My Backdoor

Destination IP: 192.168.61.142
Source IP: 192.168.61.141
Process Title: testTitle
Your commands to send: pwd
[✓ AES] RSA Yiao's Encryption
/home/yiao/Downloads
0.6172 seconds
Watch Folder Name: test
Watching...
File Name to Retrieve: key.log

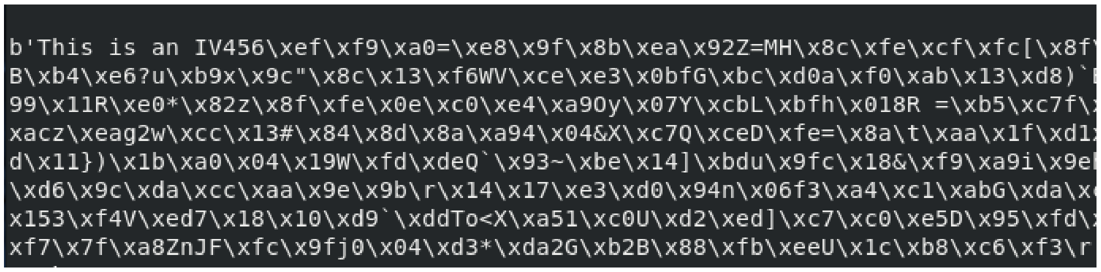
Server Side:



```
Server listening....
key.log
Got connection from ('192.168.61.141', 54086)
Server received b'Connected from client!'
Reading from: key.log
Sent  b'This is an IV456\x8a$\x1e\xb5t\xa4Z\x84\xb0\t\xe1K\x8e\x19\xfdd\x90\x
x10vi\x99\xcb\x06SQL\xc2\xfeZm\xdf\r\xa4\xc5)p\xa5\x14\xf7S\x8b}B|\xd2#.\xd5
\xbf\xcc9\x9b\xe5\xc3\xd1k\x04\x88By\x88\xc8&\xa2\x94\x03\xaeMH\x0f\x94\x90L
\xa1\xdb\xc6Y\xce\xbdX\x14L\xd9\xd0\x9e\xa8\xc90\xbaV\xff\x88kN\xde\x97\xd0\x
xc0\x87\xdaH:t5+P\xc7\xb1\x04\xd8\xe0\x80P^O\x93\x14\xedf)\xdf\xcf\x81\xb0v\x
xf6\xd2.t{lZ*\xc3\x03{\x16\xa6\x80\x80)\xdf\x97j\xd7n\x96\xba\xe8\x05\x0e\xa
9\xb3B\x14\xb4*\xd4[\xdb\xfc\x13\xbd\xd1\xfaN\xa3\xbd\na\x0bP\xa5*|L]n\xd0n'
Done sending
```

Test Case 10

AES encryption shown below for any command sent from server



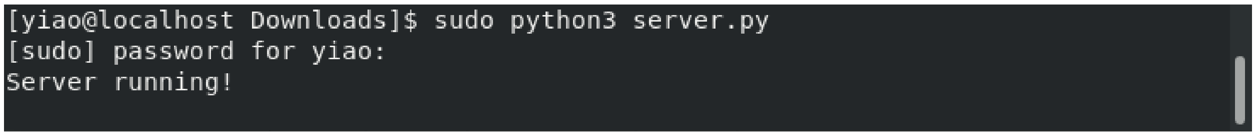
```
b'This is an IV456\xef\xf9\xa0=\xe8\x9f\x8b\xea\x92Z=MH\x8c\xfe\xcf\xfc[\x8f
B\xb4\xe6?u\xb9x\x9c"\x8c\x13\xf6WV\xce\xe3\x0bfG\xbc\xd0a\xf0\xab\x13\xd8)`
99\x11R\xe0*\x82z\x8f\xfe\x0e\xc0\xe4\xa9Oy\x07Y\xcbL\xbfh\x018R =\xb5\xc7f\
xacz\xeag2w\xcc\x13#\x84\x8d\x8a\xa94\x04&X\xc7Q\xceD\xfe=\x8a\t\xaa\x1f\xd1
d\x11})\x1b\xa0\x04\x19W\xfd\xdeQ`\x93~\xbe\x14]\xbdu\x9fc\x18&\xf9\xa9i\x9e
\xd6\x9c\xda\xcc\xaa\x9e\x9b\r\x14\x17\xe3\xd0\x94n\x06f3\xa4\xc1\xabG\xda\x
x153\xf4V\xed7\x18\x10\xd9`\xddTo<X\xa51\xc0U\xd2\xed]\xc7\xc0\xe5D\x95\xfd\
xf7\x7f\xa8ZnJF\xfc\x9fj0\x04\xd3*\xda2G\xb2B\x88\xfb\xeeU\x1c\xb8\xc6\xf3\r
```

Server Tests

1	Server can successfully run through terminal		Pass
2	Upon receiving command, server terminal displays received message		Pass

3	Upon receiving file watch notice, displays the folder that it’s watching from client		Pass
4	Upon receiving request for file transfer, displays data of file and file name it’s retrieving		Pass
5	Server can process proper AES decryption		Pass
6	Server processes proper RSA decryption		Pass
7	Server processes proper Yiao encryption		Pass
8	IP Spoofed: Server unable to send commands back to client if IP was spoofed (wrong source IP entered)		Pass
9	Check process title to ensure correct changes made		Pass
10	Keylogger: Keylogger is recording keystrokes. Test on internet browser		Pass
11	Keylogger: Keylogger file recorded as key.log in directory		Pass
12	Keylogger: Contents of keylogger available for file grabbing		Pass

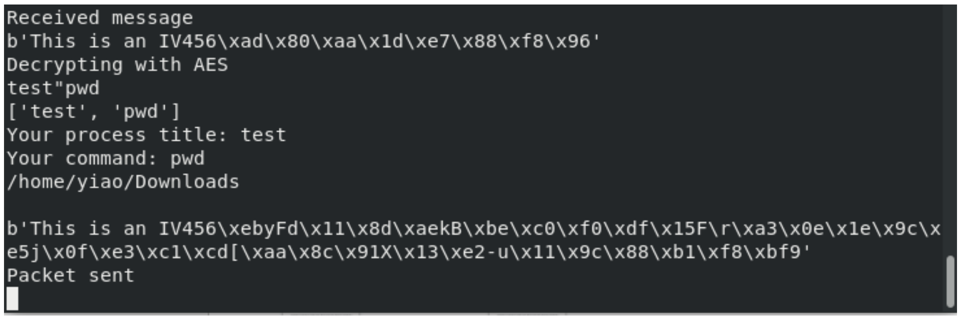
Test Case 1



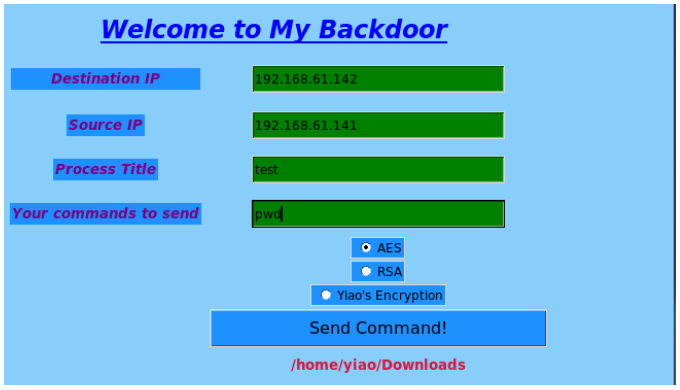
```
[yiao@localhost Downloads]$ sudo python3 server.py
[sudo] password for yiao:
Server running!
```

Test Case 2

Below shows the encrypted message that was received. It then decrypts it and prints out what was sent, which would be the process title, then the command that was received. It runs the command then sends it back encrypted to the client.



```
Received message
b'This is an IV456\xad\x80\xaa\x1d\xe7\x88\xf8\x96'
Decrypting with AES
test"pwd
['test', 'pwd']
Your process title: test
Your command: pwd
/home/yiao/Downloads

b'This is an IV456\xebyFd\x11\x8d\xaekB\xbe\xc0\xf0\xdf\x15F\r\xa3\x0e\x1e\x9c\x
e5j\x0f\xe3\xc1\xcd[\xaa\x8c\x91X\x13\xe2-u\x11\x9c\x88\xb1\xf8\xbf9'
Packet sent
```



Welcome to My Backdoor

Destination IP: 192.168.61.142

Source IP: 192.168.61.141

Process Title: test

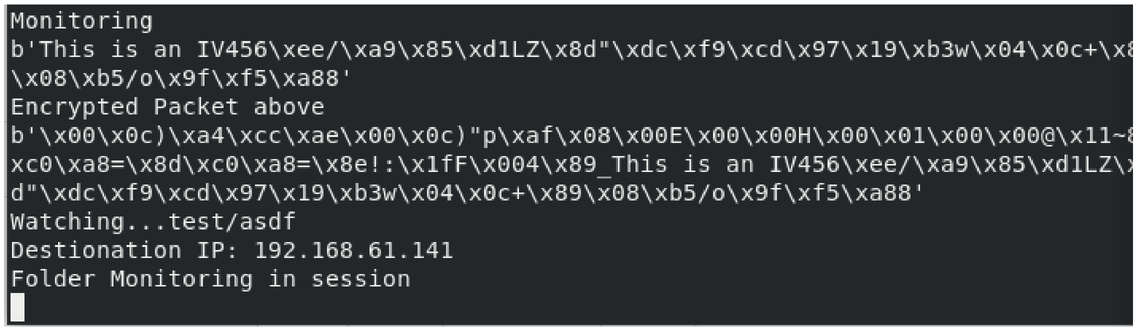
Your commands to send: pwd

[✓ AES] RSA Yiao's Encryption

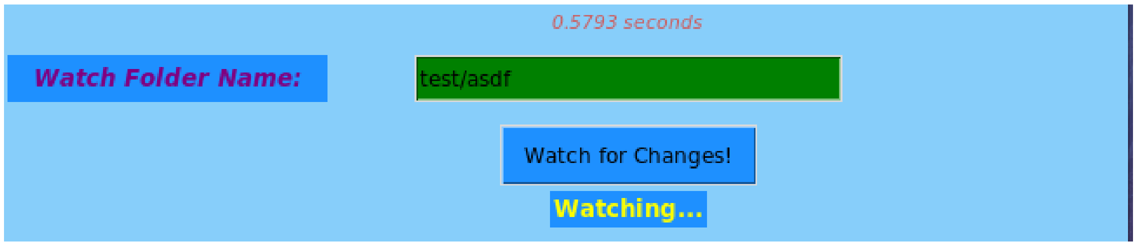
/home/yiao/Downloads

Test Case 3

Receives encrypted message of the folder name that is being watched. Sends encrypted message back to client notifying it that it’s currently watching the named folder.

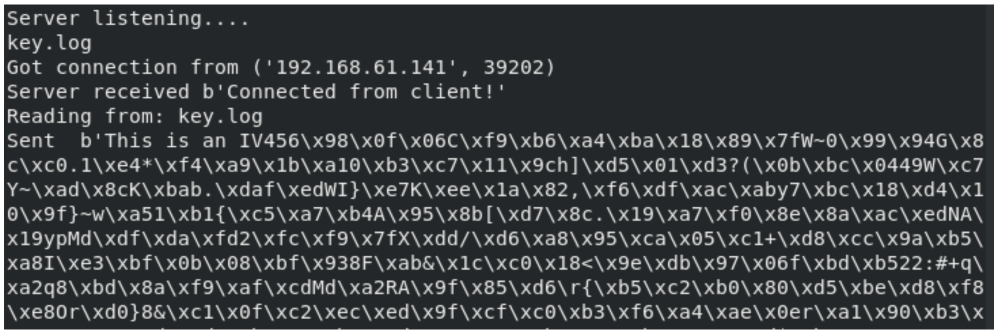


```
Monitoring
b'This is an IV456\xee/\xa9\x85\xd1LZ\x8d"\xdc\xf9\xcd\x97\x19\xb3w\x04\x0c+\x8
\x08\xb5/o\x9f\xf5\xa88'
Encrypted Packet above
b'\x00\x0c)\xa4\xcc\xae\x00\x0c)"p\xaf\x08\x00E\x00\x00H\x00\x01\x00\x00@\x11~8
xc0\xa8=\x8d\xc0\xa8=\x8e!:\x1fF\x004\x89_This is an IV456\xee/\xa9\x85\xd1LZ\x
d"\xdc\xf9\xcd\x97\x19\xb3w\x04\x0c+\x89\x08\xb5/o\x9f\xf5\xa88'
Watching...test/asdf
Destionation IP: 192.168.61.141
Folder Monitoring in session
```



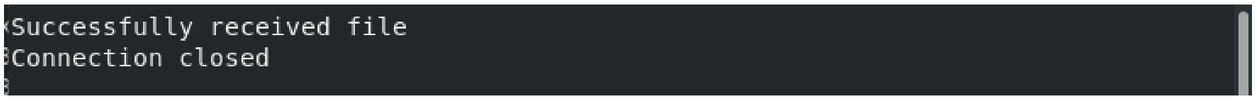
Test Case 4

Socket connection is created and connects to the client upon receiving request for file retrieval. Encrypts the contents of the file and sends it through socket to the client application.

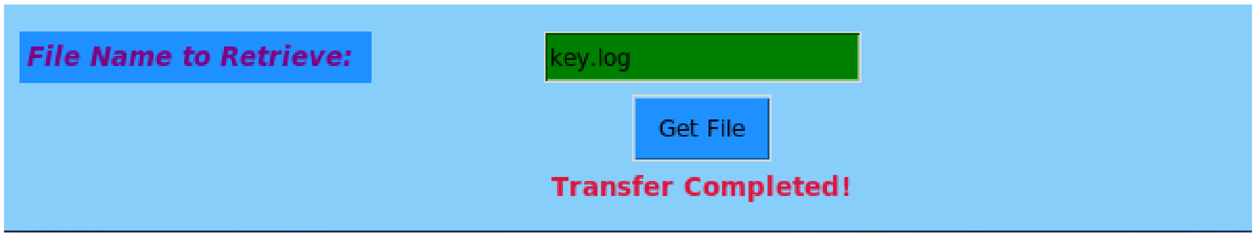


```
Server listening....
key.log
Got connection from ('192.168.61.141', 39202)
Server received b'Connected from client!'
Reading from: key.log
Sent  b'This is an IV456\x98\x0f\x06C\xf9\xb6\xa4\xba\x18\x89\x7fW~0\x99\x94G\x8
c\xc0.1\xe4*\xf4\xa9\x1b\xa10\xb3\xc7\x11\x9ch]\xd5\x01\xd3?(\x0b\xbc\x0449W\xc7
Y~\xad\x8cK\xbab.\xdaf\xedWI}\xe7K\xee\x1a\x82,\xf6\xdf\xac\xaby7\xbc\x18\xd4\x1
0\x9f}~w\xa51\xb1{\xc5\xa7\xb4A\x95\x8b[\xd7\x8c.\x19\xa7\xf0\x8e\x8a\xac\xedNA\
x19ypMd\xdf\xda\xfd2\xfc\xf9\x7fX\xdd/\xd6\xa8\x95\xca\x05\xc1+\xd8\xcc\x9a\xb5\
xa8I\xe3\xbf\x0b\x08\xbf\x938F\xab&\x1c\xc0\x18<\x9e\xdb\x97\x06f\xbd\xb522:#+q\
xa2q8\xbd\x8a\xf9\xaf\xcdMd\xa2RA\x9f\x85\xd6\r{\xb5\xc2\xb0\x80\xd5\xbe\xd8\xf8
\xe8Or\xd0}8&\xc1\x0f\xc2\xec\xed\x9f\xcf\xc0\xb3\xf6\xa4\xae\x0er\xa1\x90\xb3\x
```

Client:

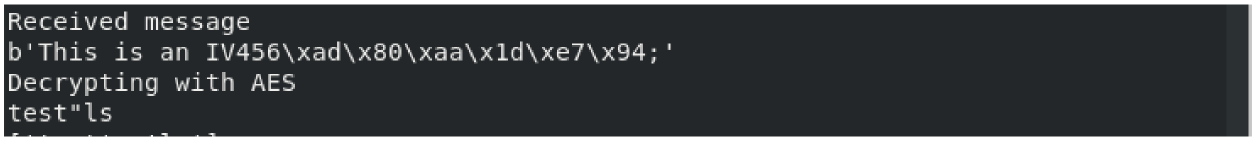


```
Successfully received file
Connection closed
```

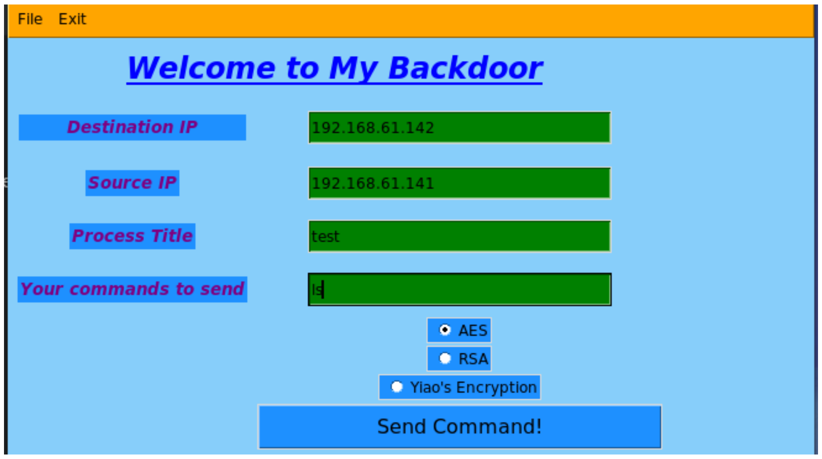


Test Case 5

AES encryption: Receives the message with encrypted process title “test” and “ls” command. Decrypts and displays information.



```
Received message
b'This is an IV456\xad\x80\xaa\x1d\xe7\x94;'
Decrypting with AES
test"ls
```



File Exit

Welcome to My Backdoor

Destination IP: 192.168.61.142

Source IP: 192.168.61.141

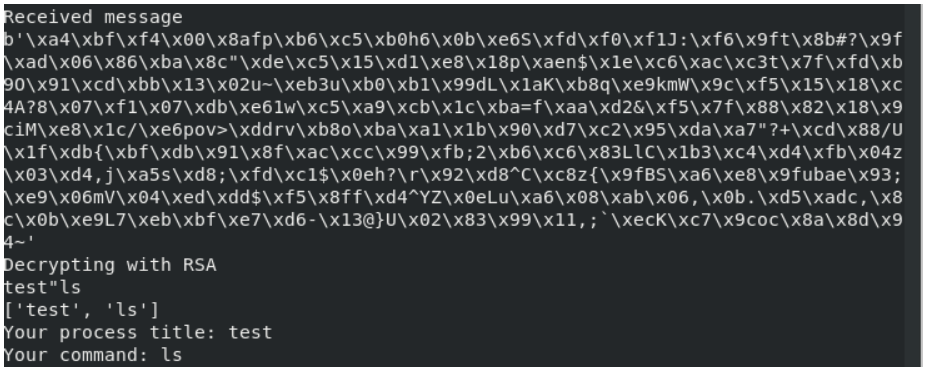
Process Title: test

Your commands to send: ls

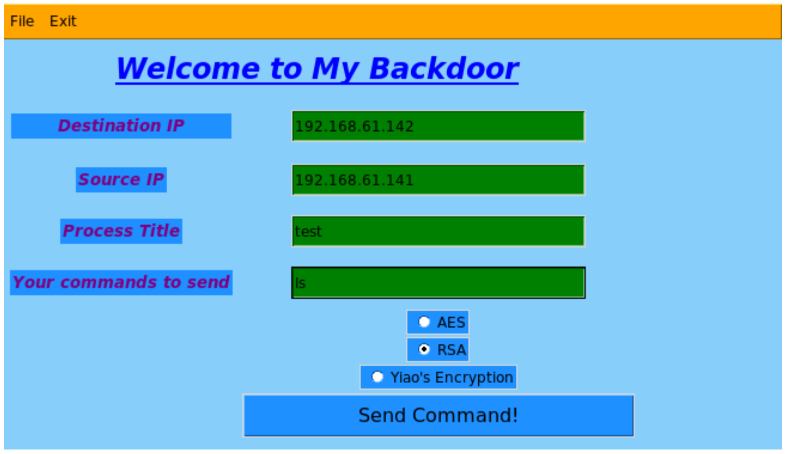
[✓ AES] RSA Yiao's Encryption

Test Case 6

RSA Encryption: Receives the message with encrypted process title and test command as in test case 5. Notice that RSA is quite a bit longer than AES here due to the more inefficient algorithm. It decrypts it and displays the information.



```
Received message
b'\xa4\xbf\xf4\x00\x8afp\xb6\xc5\xb0h6\x0b\xe6S\xfd\xf0\xf1J:\xf6\x9ft\x8b#?\x9f
\xad\x06\x86\xba\x8c"\xde\xc5\x15\xd1\xe8\x18p\xaen$\x1e\xc6\xac\xc3t\x7f\xfd\xb
9O\x91\xcd\xbb\x13\x02u~\xeb3u\xb0\xb1\x99dL\x1aK\xb8q\xe9kmW\x9c\xf5\x15\x18\xc
4A?8\x07\xf1\x07\xdb\xe61w\xc5\xa9\xcb\x1c\xba=f\xaa\xd2&\xf5\x7f\x88\x82\x18\x9
ciM\xe8\x1c/\xe6pov>\xddrv\xb8o\xba\xa1\x1b\x90\xd7\xc2\x95\xda\xa7"?+\xcd\x88/U
\x1f\xdb{\xbf\xdb\x91\x8f\xac\xcc\x99\xfb;2\xb6\xc6\x83LlC\x1b3\xc4\xd4\xfb\x04z
\x03\xd4,j\xa5s\xd8;\xfd\xc1$\x0eh?\r\x92\xd8^C\xc8z{\x9fBS\xa6\xe8\x9fubae\x93;
\xe9\x06mV\x04\xed\xdd$\xf5\x8ff\xd4^YZ\x0eLu\xa6\x08\xab\x06,\x0b.\xd5\xadc,\x8
c\x0b\xe9L7\xeb\xbf\xe7\xd6-\x13@}U\x02\x83\x99\x11,;`\xecK\xc7\x9coc\x8a\x8d\x9
4~'
Decrypting with RSA
test"ls
['test', 'ls']
Your process title: test
Your command: ls
```



Destination IP: 192.168.61.142

Source IP: 192.168.61.141

Process Title: test

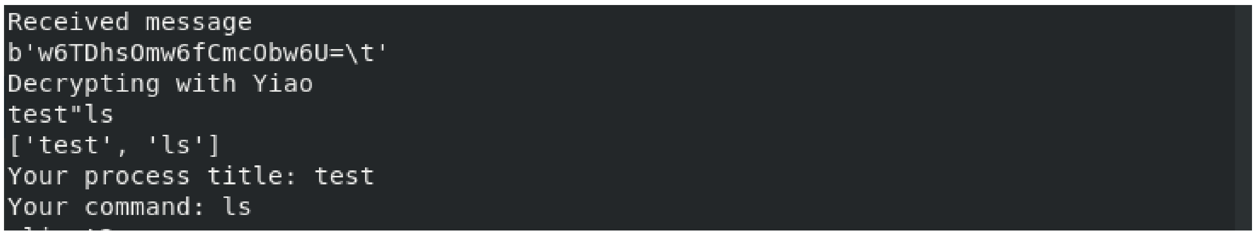
Your commands to send: ls

AES [✓ RSA] Yiao's Encryption

Test Case 7

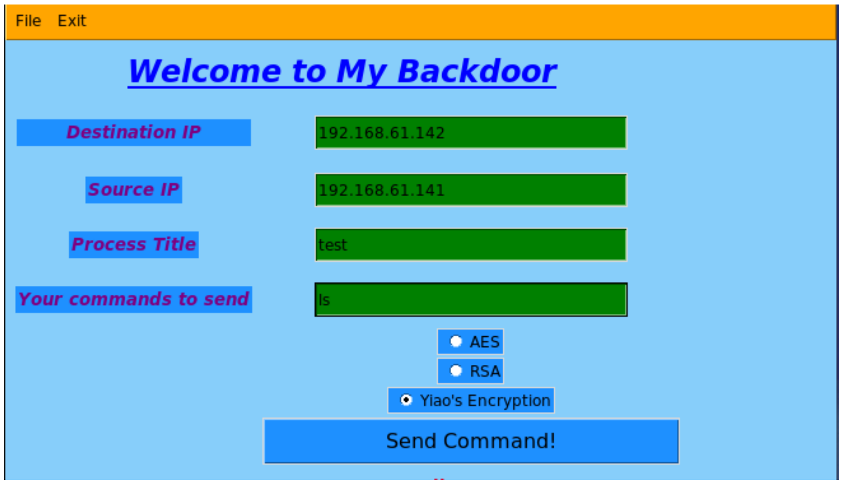
Yiao's Encryption: Same title and command as above with different encryption.

Server Side:



```
Received message
b'w6TDhsOmw6fCmcObw6U=\t'
Decrypting with Yiao
test"ls
['test', 'ls']
Your process title: test
Your command: ls
```

Client Side:



Destination IP: 192.168.61.142

Source IP: 192.168.61.141

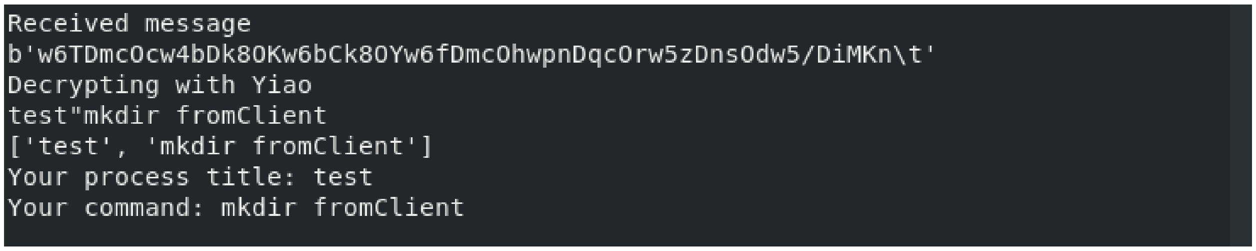
Process Title: test

Your commands to send: ls

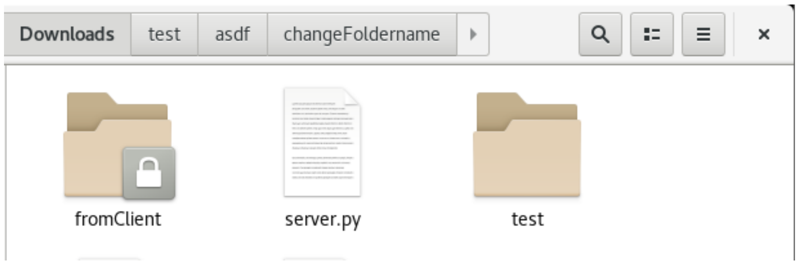
AES RSA [✓ Yiao's Encryption]

Test Case 8

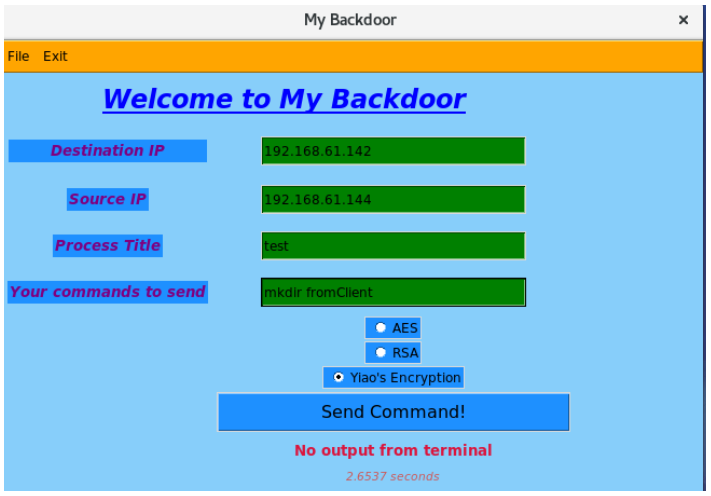
To complete this test case, I've sent a command with a spoofed IP from 192.168.61.144 resulting in creating a directory in the folder. The server receives the command fro Yiao's encryption, processes it, then runs the command. The new folder is now created.



```
Received message
b'w6TDmcOcw4bDk8OKw6bCk8OYw6fDmcOhwpnDqcOrw5zDnsOdw5/DiMKn\t'
Decrypting with Yiao
test"mkdir fromClient
['test', 'mkdir fromClient']
Your process title: test
Your command: mkdir fromClient
```



Client Side:



My Backdoor

Welcome to My Backdoor

Destination IP: 192.168.61.142

Source IP: 192.168.61.144

Process Title: test

Your commands to send: mkdir fromClient

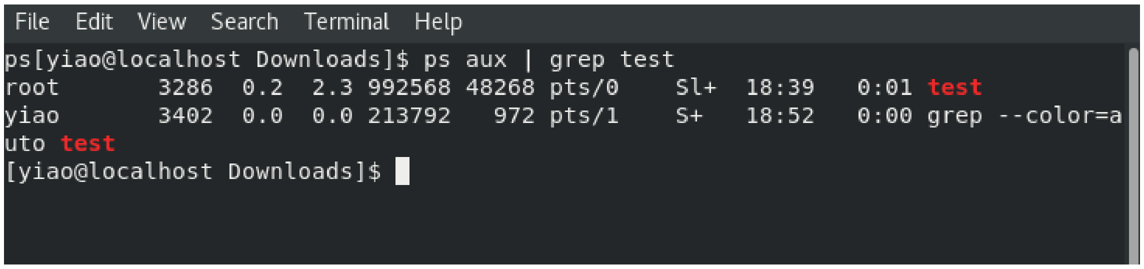
( ) AES
( ) RSA
(●) Yiao's Encryption

No output from terminal

2.6537 seconds

Test Case 9

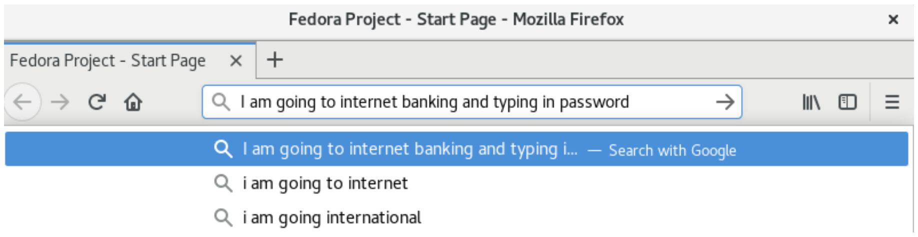
Using the above commands, sent a process title as "test". Ran the following command in the server and the process title now shows up as the requested spoofed name.



```
ps[yiao@localhost Downloads]$ ps aux | grep test
root        3286  0.2  2.3 992568 48268 pts/0    Sl+  18:39   0:01 test
yiao        3402  0.0  0.0 213792   972 pts/1    S+   18:52   0:00 grep --color=a
uto test
[yiao@localhost Downloads]$ 
```

Test Case 10

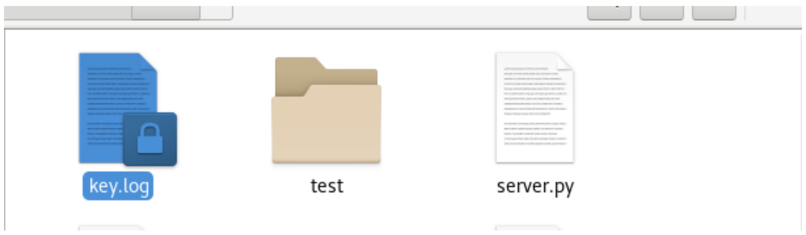
To test the keylogger, we delete the key.log file that was originally there, then run the server again. Upon running the server, we open Firefox and type in the following:


Fedora Project - Start Page - Mozilla Firefox
Fedora Project - Start Page

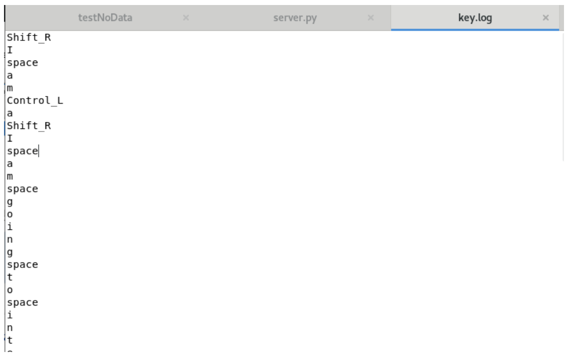
I am going to internet banking and typing in password

I am going to internet banking and typing i... — Search with Google
i am going to internet
i am going international

Looking into the folder, a key.log file now appears.

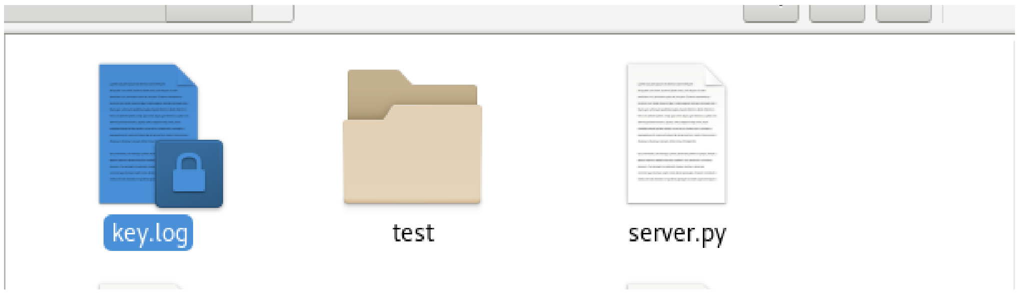


When we open the key.log file, below are the contents, same as what we had typed into Firefox.



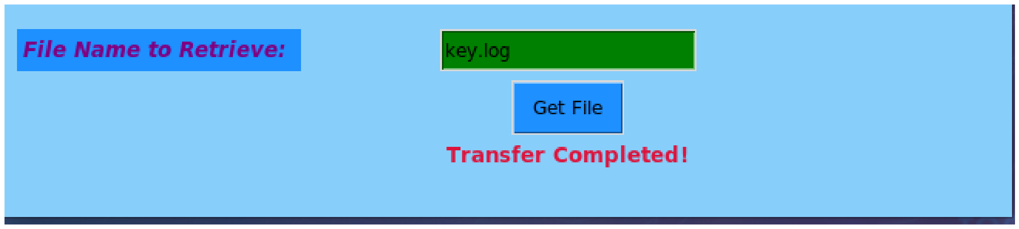
```
Shift_R
I
space
a
m
Control_L
a
Shift_R
I
space
a
m
space
g
o
i
n
g
space
t
o
space
i
n
t
```

Test Case 11

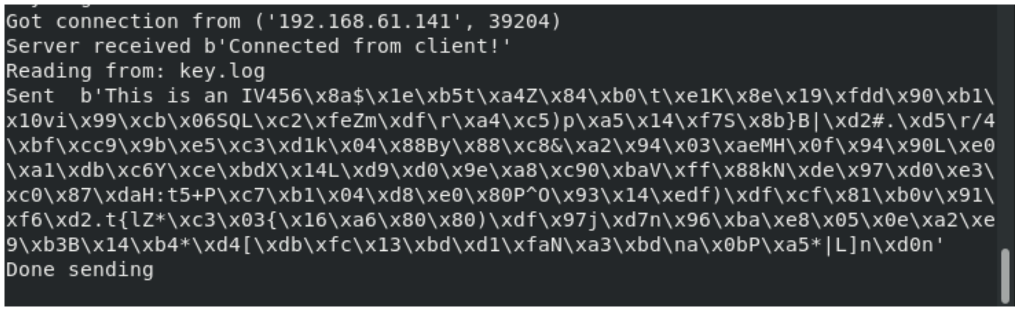


Test Case 12

Client Side:

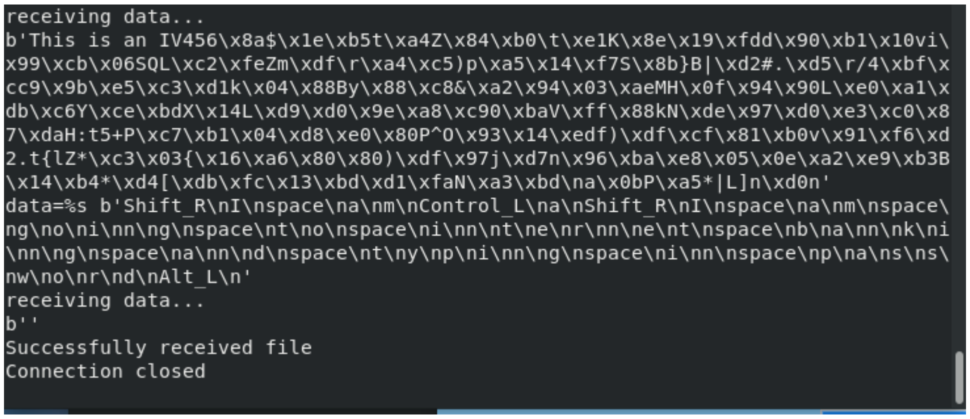


Server Side:



```
Got connection from ('192.168.61.141', 39204)
Server received b'Connected from client!'
Reading from: key.log
Sent  b'This is an IV456\x8a$\x1e\xb5t\xa4Z\x84\xb0\t\xe1K\x8e\x19\xfdd\x90\xb1\
x10vi\x99\xcb\x06SQL\xc2\xfeZm\xdf\r\xa4\xc5)p\xa5\x14\xf7S\x8b}B|\xd2#.\xd5\r/4
\xbf\xcc9\x9b\xe5\xc3\xd1k\x04\x88By\x88\xc8&\xa2\x94\x03\xaeMH\x0f\x94\x90L\xe0
\xa1\xdb\xc6Y\xce\xbdX\x14L\xd9\xd0\x9e\xa8\xc90\xbaV\xff\x88kN\xde\x97\xd0\xe3\
xc0\x87\xdaH:t5+P\xc7\xb1\x04\xd8\xe0\x80P^O\x93\x14\xedf)\xdf\xcf\x81\xb0v\x91\
xf6\xd2.t{lZ*\xc3\x03{\x16\xa6\x80\x80)\xdf\x97j\xd7n\x96\xba\xe8\x05\x0e\xa2\xe
9\xb3B\x14\xb4*\xd4[\xdb\xfc\x13\xbd\xd1\xfaN\xa3\xbd\na\x0bP\xa5*|L]n\xd0n'
Done sending
```

Client Terminal:



```
receiving data...
b'This is an IV456\x8a$\x1e\xb5t\xa4Z\x84\xb0\t\xe1K\x8e\x19\xfdd\x90\xb1\x10vi\
x99\xcb\x06SQL\xc2\xfeZm\xdf\r\xa4\xc5)p\xa5\x14\xf7S\x8b}B|\xd2#.\xd5\r/4\xbf\x
cc9\x9b\xe5\xc3\xd1k\x04\x88By\x88\xc8&\xa2\x94\x03\xaeMH\x0f\x94\x90L\xe0\xa1\x
db\xc6Y\xce\xbdX\x14L\xd9\xd0\x9e\xa8\xc90\xbaV\xff\x88kN\xde\x97\xd0\xe3\xc0\x8
7\xdaH:t5+P\xc7\xb1\x04\xd8\xe0\x80P^O\x93\x14\xedf)\xdf\xcf\x81\xb0v\x91\xf6\xd
2.t{lZ*\xc3\x03{\x16\xa6\x80\x80)\xdf\x97j\xd7n\x96\xba\xe8\x05\x0e\xa2\xe9\xb3B
\x14\xb4*\xd4[\xdb\xfc\x13\xbd\xd1\xfaN\xa3\xbd\na\x0bP\xa5*|L]n\xd0n'
data=%s b'Shift_R\nI\nspace\na\nm\nControl_L\na\nShift_R\nI\nspace\na\nm\nspace\
ng\no\ni\nn\ng\nspace\nt\no\nspace\ni\nn\nt\ne\nr\nn\ne\nt\nspace\nb\na\nn\nk\ni
\nn\ng\nspace\na\nn\nd\nspace\nt\ny\np\ni\nn\ng\nspace\ni\nn\nspace\np\na\ns\ns\
nw\no\nr\nd\nAlt_L\n'
receiving data...
b''
Successfully received file
Connection closed
```