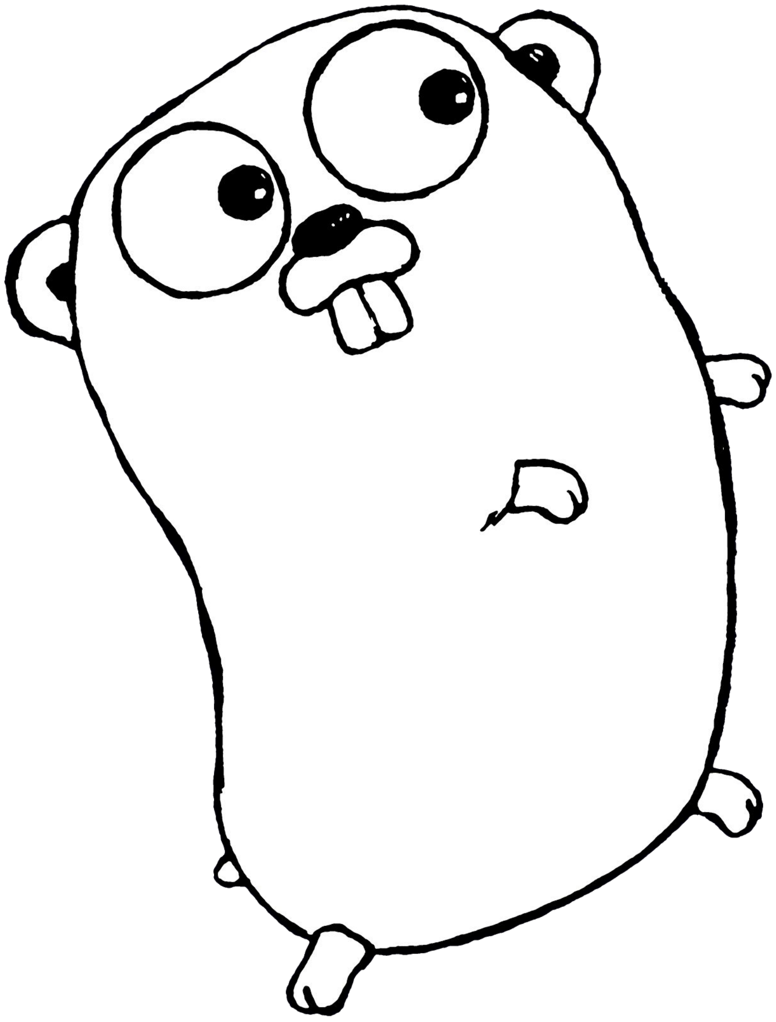


Assignment 3: GoBD

10.16.2015

Ramzi Chennafi
BCIT

Overview

In assignment 3 we were tasked with creating a packet sniffing backdoor. These requirements were and implemented in these ways:

Authentication - Implemented through an initial authentication session at the start of the client's connection.

Encryption - Using golang/aes I performed AES-256 encryption for all data transferred. This was performed with the bdencrypt.go source.

System Interactive - The client was able to control the host system the backdoor was situated on using terminal commands.

Packet Sniffing - The server uses gopacket to sniff for incoming data.

Going past these I added some bonus features, they are as follows:

OS Interoperability - Doesn't matter what type of system the backdoor or the client are placed on, they will work as expected. Of course, the commands the client sends must be specific to the backdoor's host system.

2 Way Communication - Fully interactive 2 way communication between the client and backdoor. The session begins and ends when the user desires it to.

Package Listing

Included with this assignment you should find these files.

bdmain.go

bdencrypt.go

GoBDServer

GoBDDesignDocs.pdf

README

References/* (Contains several packet captures of testing)

These are the source files, the readme, application and design documentation as well as accompanying references for the testing documentation.

Installation

To compile the GoBD program yourself first install golang, follow the guide below for exact directions for installing on your system.

```
https://golang.org/doc/install
```

Once installation has been completed, execute

```
go install GoBD
```

after navigating to the source directory. You should now be able to run the program by typing

```
GoBD
```

You may also choose to use go build GoBD

and execute the created executable by typing:

```
./GoBD
```

Usage

GoBD comes with several flags for program execution, they are as follows.

-mode=[server | client] - sets the program to client or server mode.

-ip=[ip address] - sets the ip address to send data to.

-port=[port number] - sets the port to send data to.

-lport=[port number] - sets the port to listen for incoming data on.

-iface=[net interface name] - sets the net interface to listen to on the server

-visible=[true or false] - sets whether the server should be visible or hidden

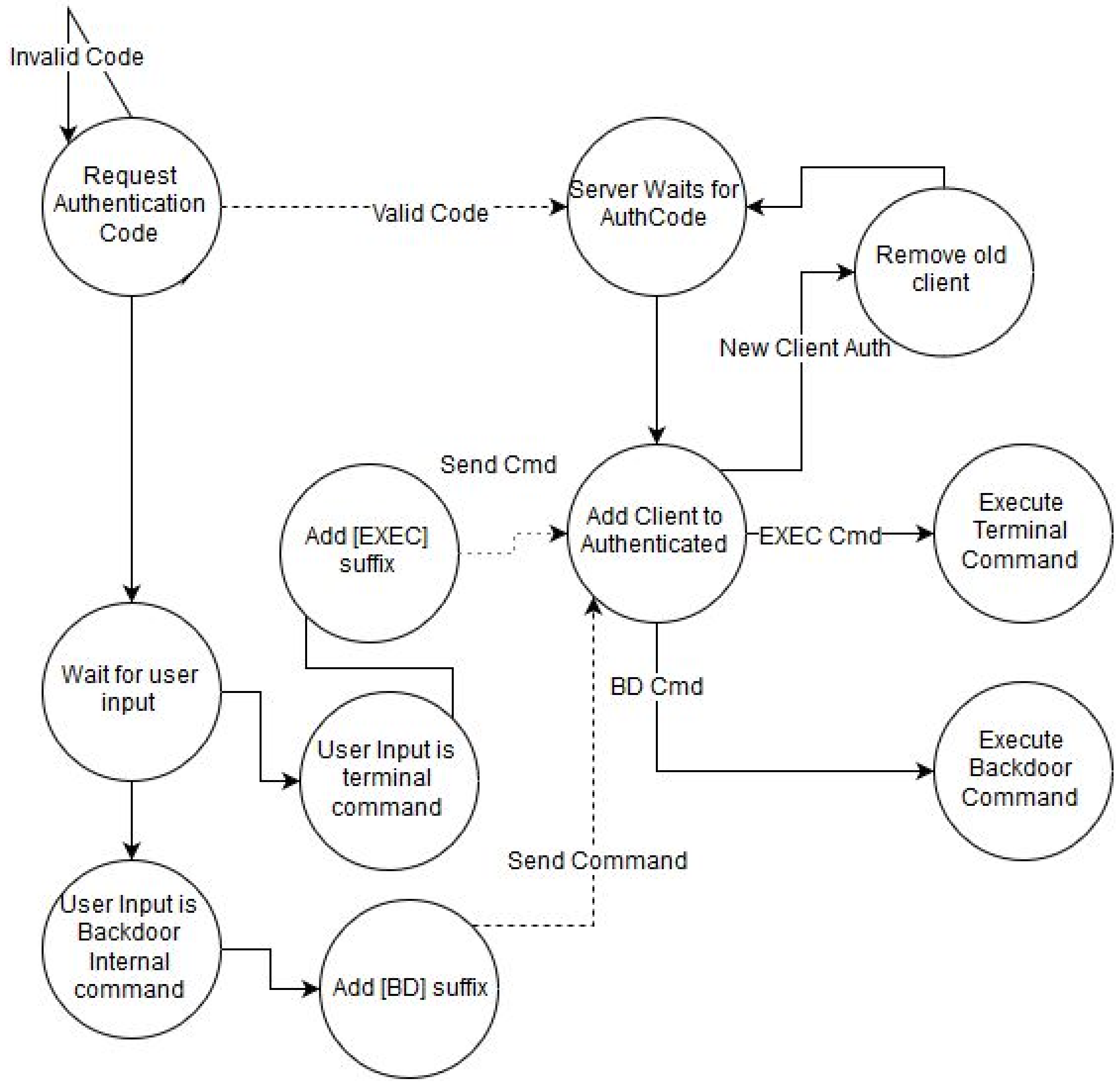
When using client or server mode, they should each have 2 different ports selected, these ports should be a reflection of each other. Take for example this execution:

```
./GoBD -ip=127.0.0.1 -iface=eth0 -port=222 -lport=223 -mode=client
./GoBD -ip=127.0.0.1 -iface=eth0 -port=223 -lport=222 -mode=client
```

It's important that you set the ports to different numbers, but make sure the lport and port combo of each execution match. The program will still work if the same port is used for everything, but you will get some strange behaviour. The server mode of GoBD must be executed with root privileges in order to allow packet sniffing to be performed. There is also one condition to any commands sent, they can only be to one program, no piping.

The authentication code at this point is: DAMNPANDIMENSIONALMICE

Design



Here is an overview of the system design. The design is relatively simple, and largely consists of :

Accept Authentication → receive commands → execute commands → return output

The system differentiates between host system commands and program commands by using a tag system, where each command is tagged with [BD] or [EXEC] for program commands and terminal commands respectively.

All program commands are prefixed by a **!** while terminal commands lack any sort of prefix.

The authentication is a dumb sort of hard coded auth into the system. The server sends no confirmation back if you have the right code, but this code is hard-coded so the client simply checks it for you.

Pseudocode

```
main(){
    process all command flags
        -mode, -ip, -port, -lport, -iface
    intiateTools()
    switch on mode
        server:
            IntiateServer()
        client:
            IntiateClient()
}

intiateClient(){
    retrieve authcode from user
    if correct, sendEncryptedData(authcode)
    else keep retrieving authcode from user

    establish a listening connection for server
    read in user input
        if user input begins with ! send command prefixed by [BD] to server
        if user input begins with just a command, send command prefixed by
[EXEC]
        if user type ?help, print help
    grabOutput(server)
}
```

```
grabOutput(connection){
    read a udp packet from the listening connection
    if the data has the suffix [END], consider that the end of data
    else continue reading for data
}
intiateServer() {
    enable packet sniffing on udp
    while {
        grab a packet off the line
        if packet is udp
            handlePacket(packet)
    }
}
handlePacket(packet) {
    if packet src is authenticated
        data = decrypt_data(packet.data)
        if data is prefixed by [EXEC]
            executeCommand(data)
        if data is prefixed by [BD]
            executeServerCommand(data)
    else if packet is from the right listening port
        data = decrypt_data(packet.data)
        if data == authcode
            set authenticated addr to the src ip of the packet
}
```

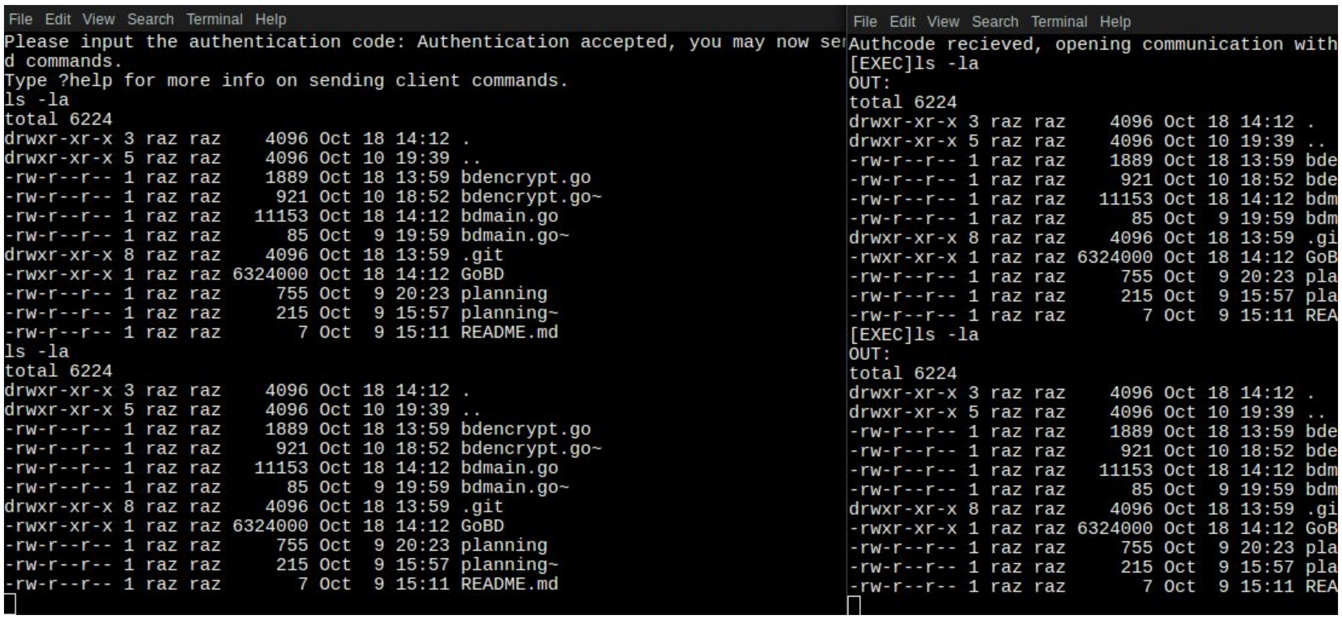
```
executeServerCommand(cmd){
    remove the [BD]! from the command
    split the arguments of the command
    switch arguments
        if setprocess
            call SetProcessName(args[1])
        if exit
            exit backdoor server
    send output from commands back encrypted to the client
}
executeCommand(cmd){
    remove the [EXEC] from the command
    split the arguments
    output = exec(arguments)
    return output from commands to client encrypted
}
SetProcessName(name) {
    set arg0 of the program to name
}
intiateTools() {
    initiate all the required ciphers for encryption
}
decrypt_data(){
    decrypt passed in data using the ciphers initiated by intiatetools
}
encrypt_data(){
    encrypt passed in data using the ciphers initiated by intiatetools
}
```

Testing

Number	Name	Descrip.	Tools Used	Pass/Fail
1	Encryption	Check if data is encrypted	Wireshark, terminal	Pass
2	2 Way Communication	The client and server can freely interact.	Terminal, Wireshark	Pass
3	OS Interoperability	Cross platform possible using GoBD	Terminal, Wireshark	Pass
4	Authentication	Authentication works properly	Wireshark, Terminal	Pass
5	Commands Executed	System commands work.	Terminal, ps, Wireshark	Pass
6	Internal Commands Executed	Program commands work.	Terminal, ps, Wireshark	Pass
7	Process Name	Process name is changed.	Terminal, ps	Pass
8	Packet Sniffing	Backdoor uses packet sniffing.	Wireshark, Terminal, Ps, netstat	Pass
9	Hidden Mode	Server exists without terminal session	Terminal, ps	Pass

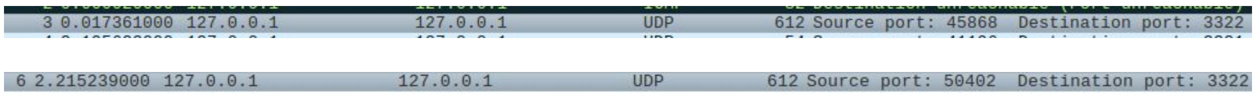
Test 1: Encryption

For this test I checked whether encryption was working properly. To test this I started up the client and backdoor and sent the ls -la command twice. Now if we take a look at a packet transfer of these two commands, they should be the same.



```
Please input the authentication code: Authentication accepted, you may now sen
d commands.
Type ?help for more info on sending client commands.
ls -la
total 6224
drwxr-xr-x 3 raz raz    4096 Oct 18 14:12 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bdencrypt.go
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bdencrypt.go~
-rw-r--r-- 1 raz raz   11153 Oct 18 14:12 bdmain.go
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdmain.go~
drwxr-xr-x 8 raz raz    4096 Oct 18 13:59 .git
-rwxr-xr-x 1 raz raz 6324000 Oct 18 14:12 GoBD
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 planning
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 planning~
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 README.md
ls -la
total 6224
drwxr-xr-x 3 raz raz    4096 Oct 18 14:12 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bdencrypt.go
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bdencrypt.go~
-rw-r--r-- 1 raz raz   11153 Oct 18 14:12 bdmain.go
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdmain.go~
drwxr-xr-x 8 raz raz    4096 Oct 18 13:59 .git
-rwxr-xr-x 1 raz raz 6324000 Oct 18 14:12 GoBD
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 planning
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 planning~
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 README.md
```

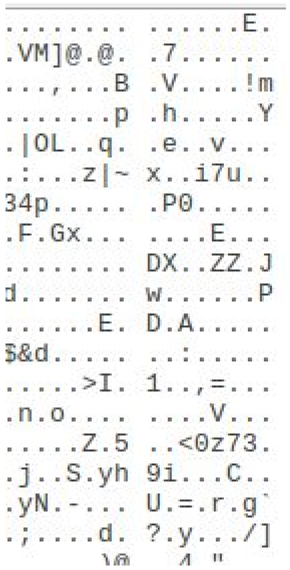
```
Authcode recieved, opening communication with
[EXEC]ls -la
OUT:
total 6224
drwxr-xr-x 3 raz raz    4096 Oct 18 14:12 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bde
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bde
-rw-r--r-- 1 raz raz   11153 Oct 18 14:12 bdm
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdm
drwxr-xr-x 8 raz raz    4096 Oct 18 13:59 .gi
-rwxr-xr-x 1 raz raz 6324000 Oct 18 14:12 GoB
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 pla
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 pla
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 REA
[EXEC]ls -la
OUT:
total 6224
drwxr-xr-x 3 raz raz    4096 Oct 18 14:12 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bde
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bde
-rw-r--r-- 1 raz raz   11153 Oct 18 14:12 bdm
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdm
drwxr-xr-x 8 raz raz    4096 Oct 18 13:59 .gi
-rwxr-xr-x 1 raz raz 6324000 Oct 18 14:12 GoB
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 pla
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 pla
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 REA
```



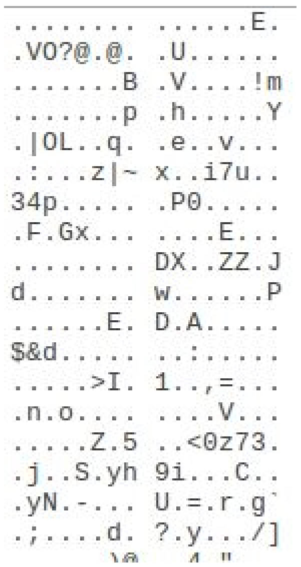
```
  3 0.017361000  127.0.0.1          127.0.0.1          UDP       612 Source port: 45868  Destination port: 3322

6 2.215239000  127.0.0.1          127.0.0.1          UDP       612 Source port: 50402  Destination port: 3322
```

First execution of the command and the second, these two packets were sent after each other. To the sides, we see the data from each packet, excluding the header, the two packets are exactly the same. This test is a success.



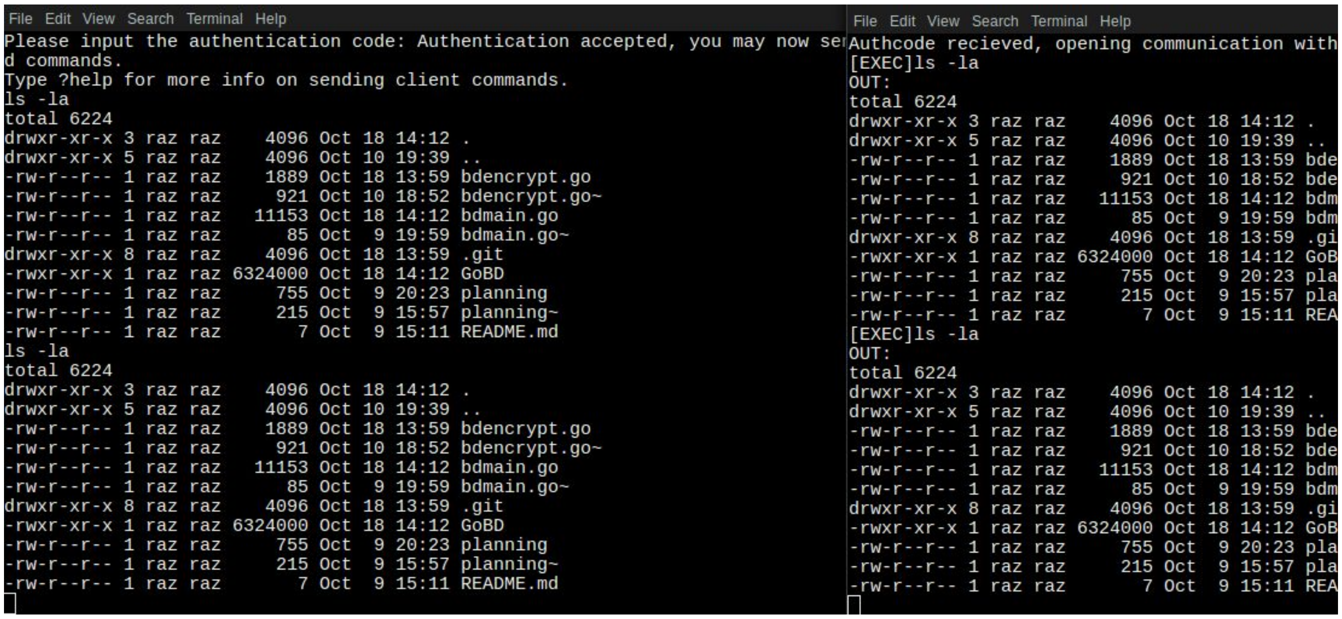
```
........ ......E.
.VM]@.@. .7......
...,...B .V....!m
.......p .h.....Y
.|OL..q. .e..v...
.:...z|~ x..i7u..
34p..... .P0.....
.F.Gx... ....E...
........ DX..ZZ.J
d....... w......P
......E. D.A.....
$&d..... ..:.....
.....>I. 1..,=...
.n.o.... ....V...
.....Z.5 ..<0z73.
.j..S.yh 9i...C..
.yN.-... U.=.r.g`
.;....d. ?.y.../]
```



```
........ ......E.
.VO?@.@. .U......
.......B .V....!m
.......p .h.....Y
.|OL..q. .e..v...
.:...z|~ x..i7u..
34p..... .P0.....
.F.Gx... ....E...
........ DX..ZZ.J
d....... w......P
......E. D.A.....
$&d..... ..:.....
.....>I. 1..,=...
.n.o.... ....V...
.....Z.5 ..<0z73.
.j..S.yh 9i...C..
.yN.-... U.=.r.g`
.;....d. ?.y.../]
```

Test 2: Two Way Communication

For this test I'd like to return to the picture grabbed from our encryption test. As you can see here, the client can continuously make commands and the server responds accordingly. This test is a pass.

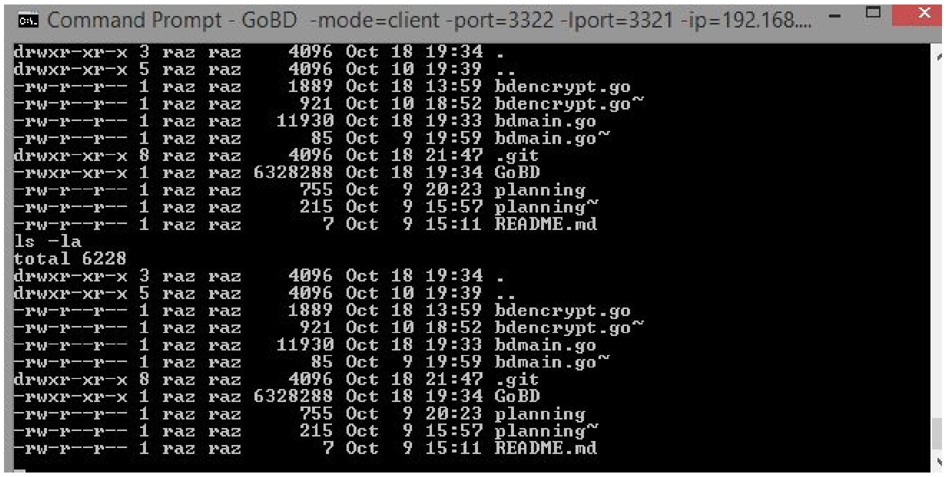


```
Please input the authentication code: Authentication accepted, you may now se
d commands.
Type ?help for more info on sending client commands.
ls -la
total 6224
drwxr-xr-x 3 raz raz    4096 Oct 18 14:12 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bdencrypt.go
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bdencrypt.go~
-rw-r--r-- 1 raz raz   11153 Oct 18 14:12 bdmain.go
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdmain.go~
drwxr-xr-x 8 raz raz    4096 Oct 18 13:59 .git
-rwxr-xr-x 1 raz raz 6324000 Oct 18 14:12 GoBD
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 planning
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 planning~
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 README.md
ls -la
total 6224
drwxr-xr-x 3 raz raz    4096 Oct 18 14:12 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bdencrypt.go
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bdencrypt.go~
-rw-r--r-- 1 raz raz   11153 Oct 18 14:12 bdmain.go
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdmain.go~
drwxr-xr-x 8 raz raz    4096 Oct 18 13:59 .git
-rwxr-xr-x 1 raz raz 6324000 Oct 18 14:12 GoBD
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 planning
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 planning~
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 README.md
```

```
Authcode recieved, opening communication with
[EXEC]ls -la
OUT:
total 6224
drwxr-xr-x 3 raz raz    4096 Oct 18 14:12 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bde
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bde
-rw-r--r-- 1 raz raz   11153 Oct 18 14:12 bdm
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdm
drwxr-xr-x 8 raz raz    4096 Oct 18 13:59 .gi
-rwxr-xr-x 1 raz raz 6324000 Oct 18 14:12 GoB
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 pla
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 pla
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 REA
[EXEC]ls -la
OUT:
total 6224
drwxr-xr-x 3 raz raz    4096 Oct 18 14:12 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bde
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bde
-rw-r--r-- 1 raz raz   11153 Oct 18 14:12 bdm
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdm
drwxr-xr-x 8 raz raz    4096 Oct 18 13:59 .gi
-rwxr-xr-x 1 raz raz 6324000 Oct 18 14:12 GoB
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 pla
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 pla
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 REA
```

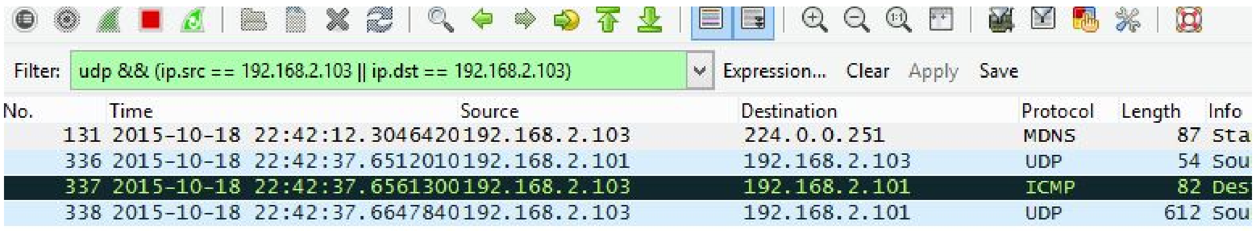
Test 3: OS Interoperability

In this test we execute a client on the windows machine and a server on a linux machine. Below is the windows machine executing a ls -la command.



```
Command Prompt - GoBD  -mode=client -port=3322 -lport=3321 -ip=192.168....
```

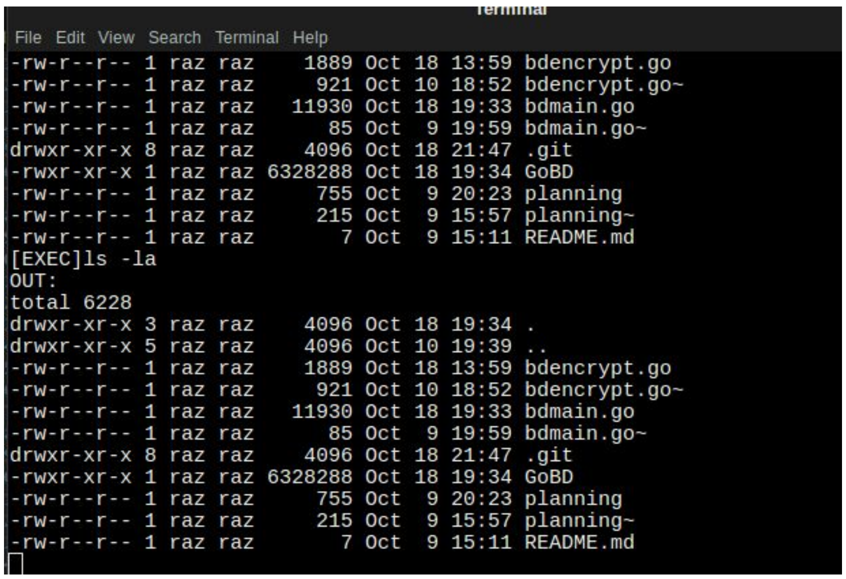
```
drwxr-xr-x 3 raz raz    4096 Oct 18 19:34 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bdencrypt.go
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bdencrypt.go~
-rw-r--r-- 1 raz raz   11930 Oct 18 19:33 bdmain.go
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdmain.go~
drwxr-xr-x 8 raz raz    4096 Oct 18 21:47 .git
-rwxr-xr-x 1 raz raz 6328288 Oct 18 19:34 GoBD
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 planning
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 planning~
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 README.md
ls -la
total 6228
drwxr-xr-x 3 raz raz    4096 Oct 18 19:34 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bdencrypt.go
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bdencrypt.go~
-rw-r--r-- 1 raz raz   11930 Oct 18 19:33 bdmain.go
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdmain.go~
drwxr-xr-x 8 raz raz    4096 Oct 18 21:47 .git
-rwxr-xr-x 1 raz raz 6328288 Oct 18 19:34 GoBD
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 planning
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 planning~
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 README.md
```



```
Filter: udp && (ip.src == 192.168.2.103 || ip.dst == 192.168.2.103)

No.  Time                        Source          Destination    Protocol  Length  Info
 131 2015-10-18 22:42:12.3046420192.168.2.103    224.0.0.251    MDNS          87  Sta
 336 2015-10-18 22:42:37.6512010192.168.2.101    192.168.2.103  UDP           54  Sou
 337 2015-10-18 22:42:37.6561300192.168.2.103    192.168.2.101  ICMP          82  Des
 338 2015-10-18 22:42:37.6647840192.168.2.103    192.168.2.101  UDP          612  Sou
```

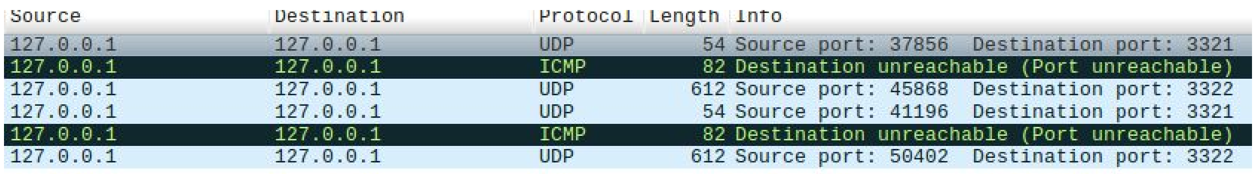
As you can see, we sent a packet between the two machines. If we look at the linux terminal we see the output the command in action. This test is a success.



```
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bdencrypt.go
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bdencrypt.go~
-rw-r--r-- 1 raz raz   11930 Oct 18 19:33 bdmain.go
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdmain.go~
drwxr-xr-x 8 raz raz    4096 Oct 18 21:47 .git
-rwxr-xr-x 1 raz raz 6328288 Oct 18 19:34 GoBD
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 planning
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 planning~
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 README.md
[EXEC]ls -la
OUT:
total 6228
drwxr-xr-x 3 raz raz    4096 Oct 18 19:34 .
drwxr-xr-x 5 raz raz    4096 Oct 10 19:39 ..
-rw-r--r-- 1 raz raz    1889 Oct 18 13:59 bdencrypt.go
-rw-r--r-- 1 raz raz     921 Oct 10 18:52 bdencrypt.go~
-rw-r--r-- 1 raz raz   11930 Oct 18 19:33 bdmain.go
-rw-r--r-- 1 raz raz      85 Oct  9 19:59 bdmain.go~
drwxr-xr-x 8 raz raz    4096 Oct 18 21:47 .git
-rwxr-xr-x 1 raz raz 6328288 Oct 18 19:34 GoBD
-rw-r--r-- 1 raz raz     755 Oct  9 20:23 planning
-rw-r--r-- 1 raz raz     215 Oct  9 15:57 planning~
-rw-r--r-- 1 raz raz       7 Oct  9 15:11 README.md
```

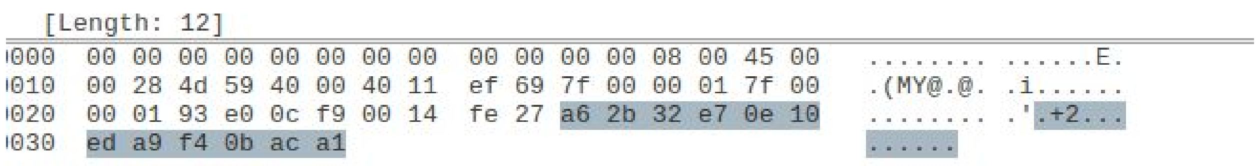
Test 4: Authentication

At the start of a client session it must authenticate. In my program this authentication code is hard coded as “DAMNPANDIMENSIONALMICE”. If we take a look at our previous packet capture we in fact see an auth packet being sent along.



```
Source        Destination   Protocol Length Info
127.0.0.1     127.0.0.1     UDP          54 Source port: 37856  Destination port: 3321
127.0.0.1     127.0.0.1     ICMP         82 Destination unreachable (Port unreachable)
127.0.0.1     127.0.0.1     UDP         612 Source port: 45868  Destination port: 3322
127.0.0.1     127.0.0.1     UDP          54 Source port: 41196  Destination port: 3321
127.0.0.1     127.0.0.1     ICMP         82 Destination unreachable (Port unreachable)
127.0.0.1     127.0.0.1     UDP         612 Source port: 50402  Destination port: 3322
```

If we count the number of encrypted characters in the data of the packet, we can see that it is in fact the same length as our authcode. This along with the output on each side, shows that the program is successful.



```
  [Length: 12]
000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
010  00 28 4d 59 40 00 40 11  ef 69 7f 00 00 01 7f 00   .(MY@.@. .i......
020  00 01 93 e0 0c f9 00 14  fe 27 a6 2b 32 e7 0e 10   ........ .'.+2...
030  ed a9 f4 0b ac a1                                  ......
```

DAMNPANDIMENSIONALMICE\n = 24 characters

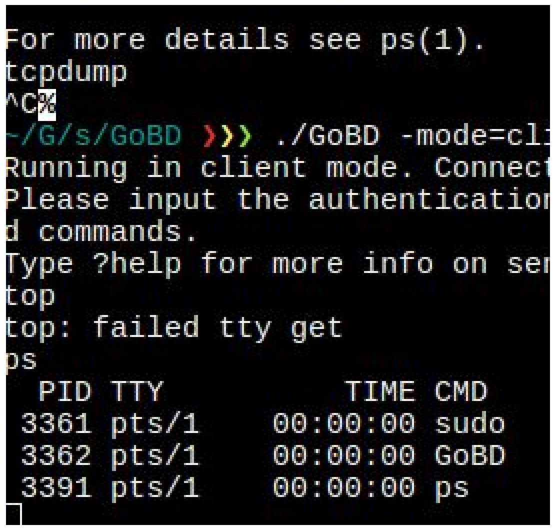
a6 2b 32 e7 0e 10 ed a9 f4 0b ac a1 = 24 characters

Test 5: Commands Executed

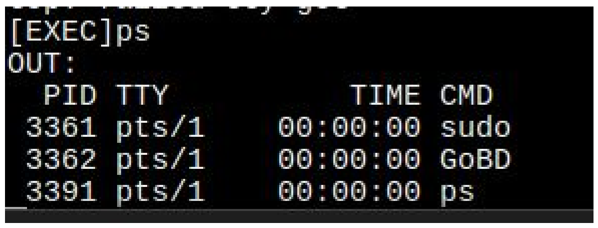
On this test we are checking if the backdoor commands executed, actually work. In this case, we're going to execute the following command

ps

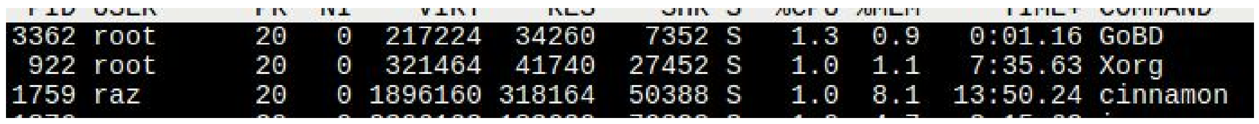
By doing this, we see a list of currently running processes. Comparing this to top, we see that GoBD has the same pid as listed by both the client and server.



```
For more details see ps(1).
tcpdump
^C%
~/G/s/GoBD ❯❯❯ ./GoBD -mode=cli
Running in client mode. Connect
Please input the authenticatio
d commands.
Type ?help for more info on se
top
top: failed tty get
ps
  PID TTY          TIME CMD
 3361 pts/1    00:00:00 sudo
 3362 pts/1    00:00:00 GoBD
 3391 pts/1    00:00:00 ps
```



```
[EXEC]ps
OUT:
  PID TTY          TIME CMD
 3361 pts/1    00:00:00 sudo
 3362 pts/1    00:00:00 GoBD
 3391 pts/1    00:00:00 ps
```

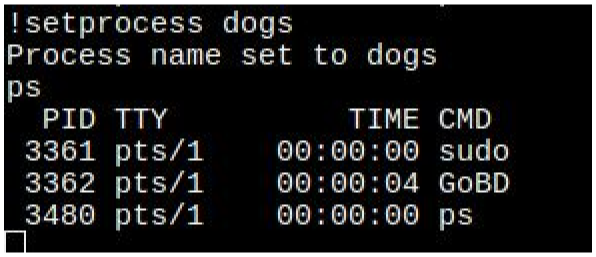


```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3362 root      20   0  217224  34260   7352 S   1.3  0.9   0:01.16 GoBD
  922 root      20   0  321464  41740  27452 S   1.0  1.1   7:35.63 Xorg
 1759 raz       20   0 1896160 318164  50388 S   1.0  8.1  13:50.24 cinnamon
```

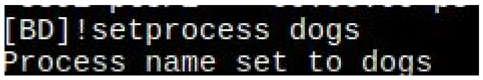
This clearly shows the test is a success, and commands are being executed properly.

Test 6 & 7: Internal Commands Executed & Process Name

In this test we performed the same sort of execution as test 5. As you can see below we executed the internal command “!setprocess dogs” which sets the internal backdoor process name to “dogs”.

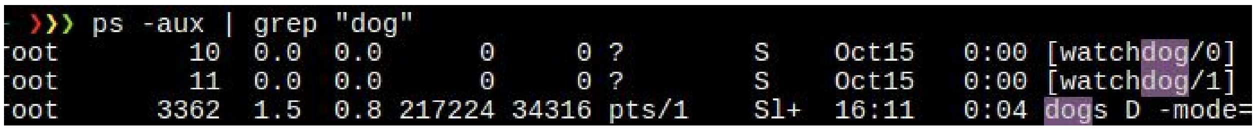


```
!setprocess dogs
Process name set to dogs
ps
  PID TTY          TIME CMD
 3361 pts/1    00:00:00 sudo
 3362 pts/1    00:00:04 GoBD
 3480 pts/1    00:00:00 ps
```



```
[BD]!setprocess dogs
Process name set to dogs
```

If we look to the ps -aux | grep “dog” listing we find our program just sitting there, with its new process name.

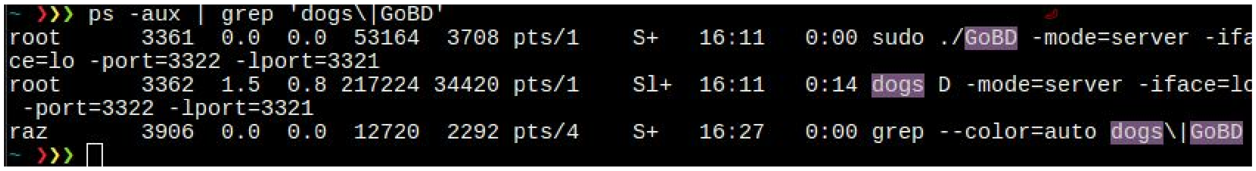


```
- ››› ps -aux | grep "dog"
root        10  0.0  0.0      0     0 ?        S    Oct15   0:00 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S    Oct15   0:00 [watchdog/1]
root      3362  1.5  0.8 217224 34316 pts/1    Sl+  16:11   0:04 dogs D -mode=
```

This test is a success.

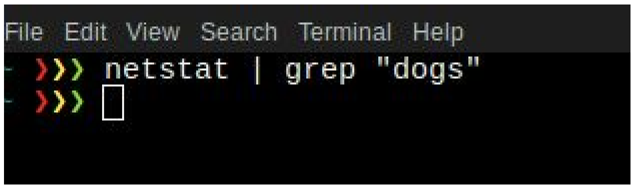
Test 8: Packet Sniffing

The marker for packet sniffing is the existence of a listening socket. So to show that the server lacks this, I executed both programs.



```
~ ››› ps -aux | grep 'dogs\|GoBD'
root      3361  0.0  0.0  53164  3708 pts/1    S+   16:11   0:00 sudo ./GoBD -mode=server -ifa
ce=lo -port=3322 -lport=3321
root      3362  1.5  0.8 217224 34420 pts/1    Sl+  16:11   0:14 dogs D -mode=server -iface=lo
 -port=3322 -lport=3321
raz       3906  0.0  0.0  12720  2292 pts/4    S+   16:27   0:00 grep --color=auto dogs\|GoBD
~ ›››
```

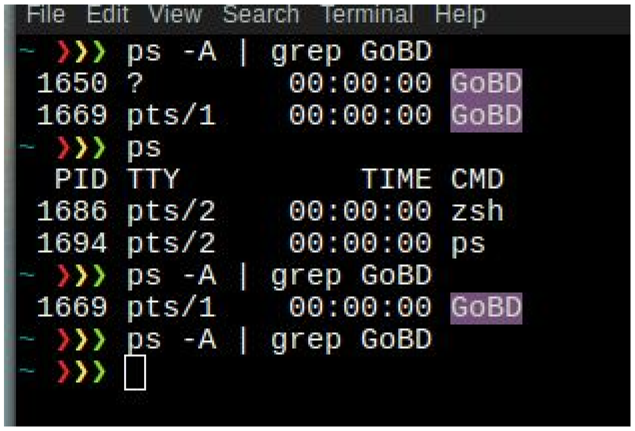
After doing so I ran a netstat | grep ‘dogs’ to find the server on the listener list. This output nothing. Yet after performing this, the two are still able to use commands. This shows that the server is performing packet sniffing.



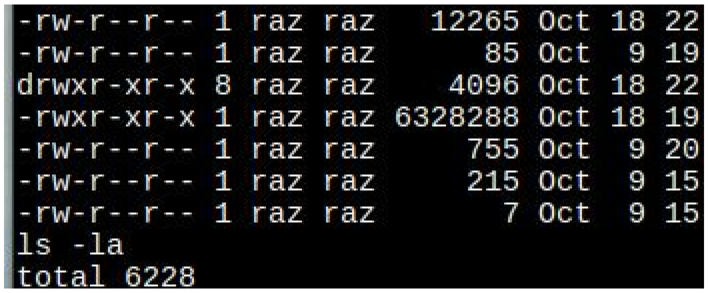
```
~ ››› netstat | grep "dogs"
~ ›››
```

Test 9: Hidden Mode

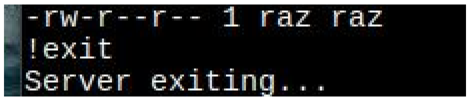
In this test I will demonstrate the hidden mode, a feature that causes the backdoor to exist separate from any terminal instance, something that is a definite need on a backdoor.



```
~ ❯❯❯ ps -A | grep GoBD
 1650 ?        00:00:00 GoBD
 1669 pts/1    00:00:00 GoBD
~ ❯❯❯ ps
  PID TTY          TIME CMD
 1686 pts/2    00:00:00 zsh
 1694 pts/2    00:00:00 ps
~ ❯❯❯ ps -A | grep GoBD
 1669 pts/1    00:00:00 GoBD
~ ❯❯❯ ps -A | grep GoBD
~ ❯❯❯ 
```

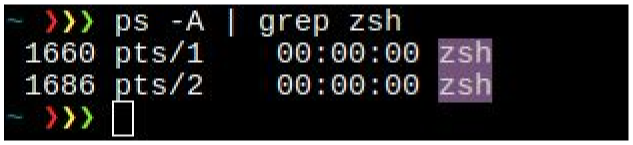


```
-rw-r--r-- 1 raz raz   12265 Oct 18 22
-rw-r--r-- 1 raz raz      85 Oct  9 19
drwxr-xr-x 8 raz raz    4096 Oct 18 22
-rwxr-xr-x 1 raz raz 6328288 Oct 18 19
-rw-r--r-- 1 raz raz     755 Oct  9 20
-rw-r--r-- 1 raz raz     215 Oct  9 15
-rw-r--r-- 1 raz raz       7 Oct  9 15
ls -la
total 6228
```



```
-rw-r--r-- 1 raz raz
!exit
Server exiting...
```

In a previous terminal, I executed the server program. As you can see from the image below ps does not list the server as part of the terminal, however doing a ps -A lists two processes as running. Meanwhile the terminal below is still able to execute commands. Finally, the client terminal executes a !exit command, which closes the backdoor. As you can see, the process is gone for the server, and once we kill the client, the last process is gone



```
~ ❯❯❯ ps -A | grep zsh
 1660 pts/1    00:00:00 zsh
 1686 pts/2    00:00:00 zsh
~ ❯❯❯ 
```

Finally, if we look at the processes, we can see that only two terminals are active, meaning there cannot be a third terminal running executing the program. This test is a success.

References

For pcap references of several of these tests, please refer to the folder titled "References", several of the tests have available pcaps to doubly prove their authenticity.