Network Research Project

Mohamad Shahir Bin Mohamad Yani

Student code: S39

Unit code: 130623

Trainer Name: James Lim

CONTENTS

Introduction

This report will comprise of creating a script that automates communicating with a remote server and running tasks on the remote server from a local device anonymously. Script would be further broken down into smaller sections and explained followed with screenshots of output.

Network traffic would be captured and monitored during execution of script to better understand the network and security. The traffic would then be analysed on what protocols are used during the running of script and how it impacts the CIA Triad.

The findings will be then discussed on how certain issues can be addressed upon with better alternatives/suggestions to create a more secure network environment.

Methodologies

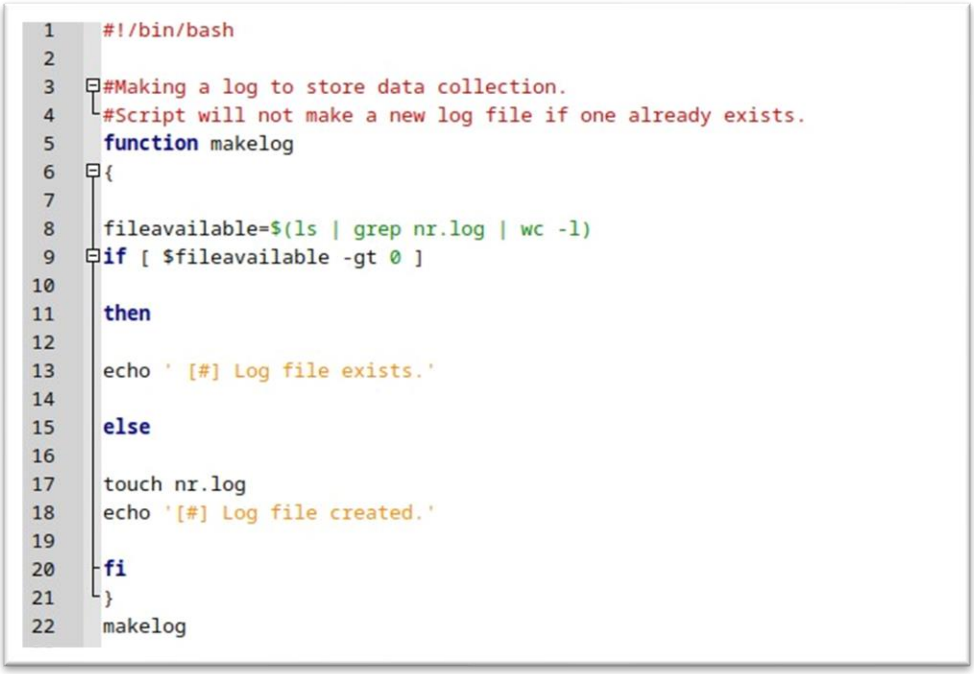
Contents of Script

- Creating a log file
- Installing of relevant programs
- Checking of spoofed IP
- Entering credentials
- Nmap scan
- SSH Pass
- Retrieving file
- Log activity

Link to full script:

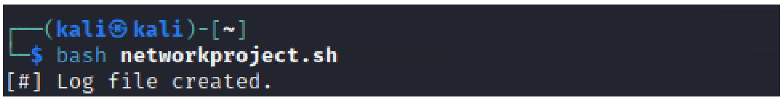
networkproject.sh

Creating A Log File



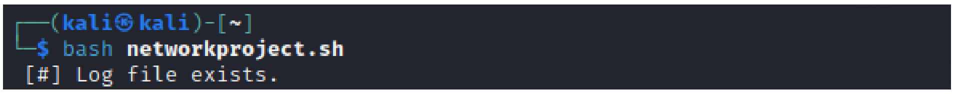
```
#!/bin/bash

#Making a log to store data collection.
#Script will not make a new log file if one already exists.
function makelog
{

fileavailable=$(ls | grep nr.log | wc -l)
if [ $fileavailable -gt 0 ]

then

echo ' [#] Log file exists.'

else

touch nr.log
echo '[#] Log file created.'

fi
}
makelog
```

First function of the script searches the current directory for an existing nr.log file. If file does not exist, it will create a new one. If file already exists, it will not create a new one.



```
┌──(kali㉿kali)-[~]
└─$ bash networkproject.sh
[#] Log file created.
```

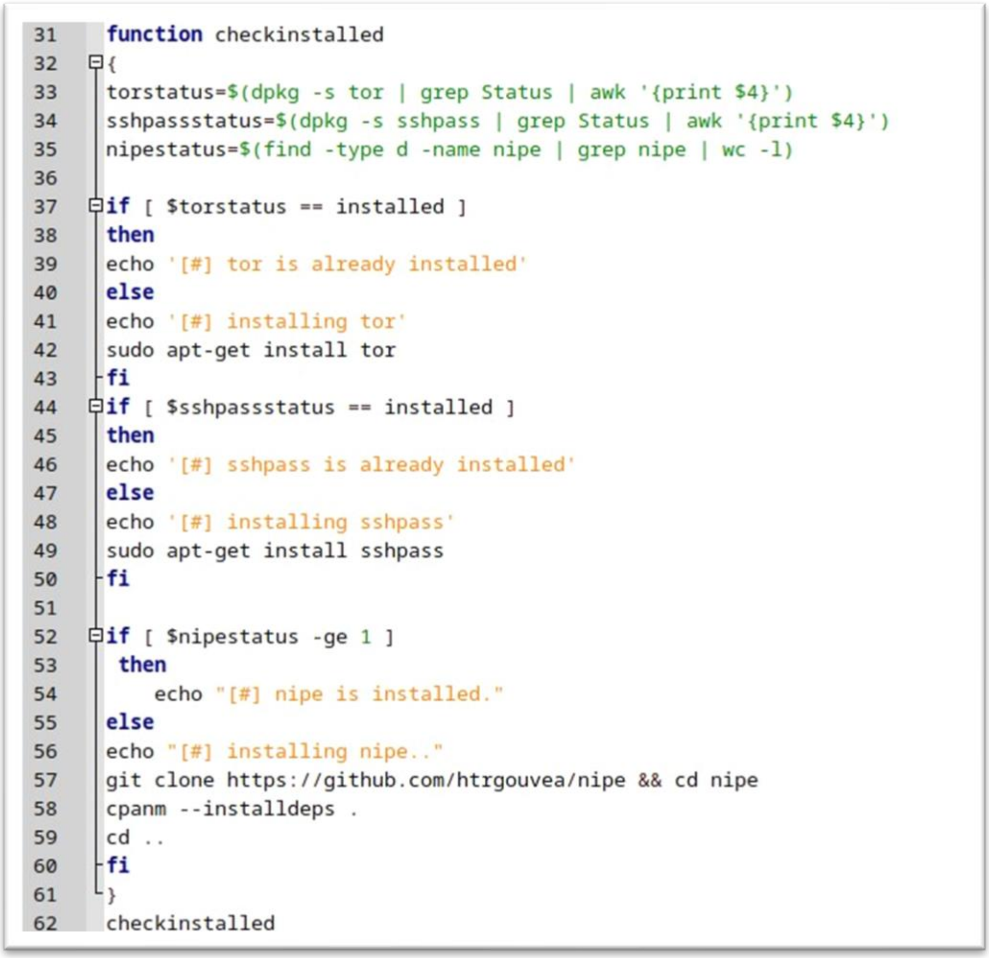
If file does not exist.



```
┌──(kali㉿kali)-[~]
└─$ bash networkproject.sh
 [#] Log file exists.
```

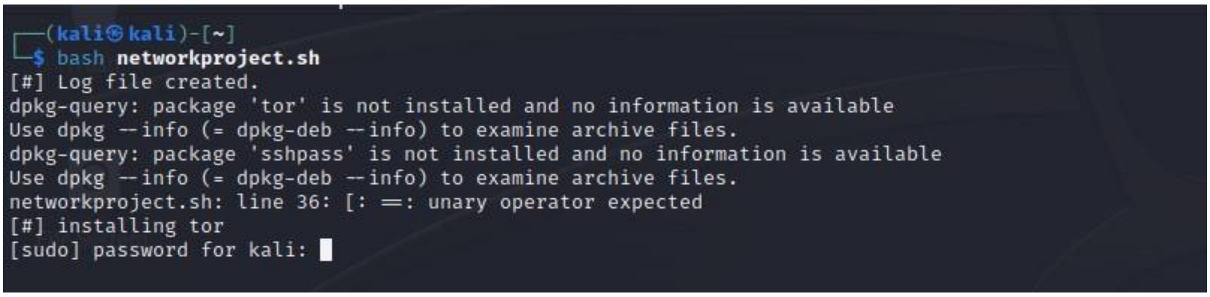
If file already exists.

Installing Of Relevant Programs



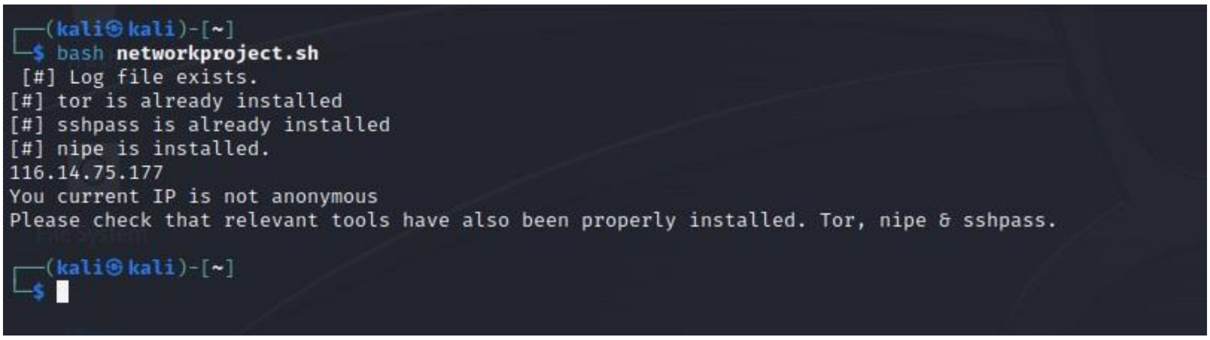
```
function checkinstalled
{
torstatus=$(dpkg -s tor | grep Status | awk '{print $4}')
sshpassstatus=$(dpkg -s sshpass | grep Status | awk '{print $4}')
nipestatus=$(find -type d -name nipe | grep nipe | wc -l)

if [ $torstatus == installed ]
then
echo '[#] tor is already installed'
else
echo '[#] installing tor'
sudo apt-get install tor
fi
if [ $sshpassstatus == installed ]
then
echo '[#] sshpass is already installed'
else
echo '[#] installing sshpass'
sudo apt-get install sshpass
fi

if [ $nipestatus -ge 1 ]
 then
    echo "[#] nipe is installed."
else
echo "[#] installing nipe.."
git clone https://github.com/htrgouvea/nipe && cd nipe
cpanm --installdeps .
cd ..
fi
}
checkinstalled
```

Script then checks the user's server to see if the relevant programs have been installed for the script to run which are tor, sshpass and nipe. If not already installed, script will automatically install the programs. If already installed, the server would inform user that respective programs have been installed.



```
┌──(kali㉿kali)-[~]
└─$ bash networkproject.sh
[#] Log file created.
dpkg-query: package 'tor' is not installed and no information is available
Use dpkg --info (= dpkg-deb --info) to examine archive files.
dpkg-query: package 'sshpass' is not installed and no information is available
Use dpkg --info (= dpkg-deb --info) to examine archive files.
networkproject.sh: line 36: [: ==: unary operator expected
[#] installing tor
[sudo] password for kali: 
```

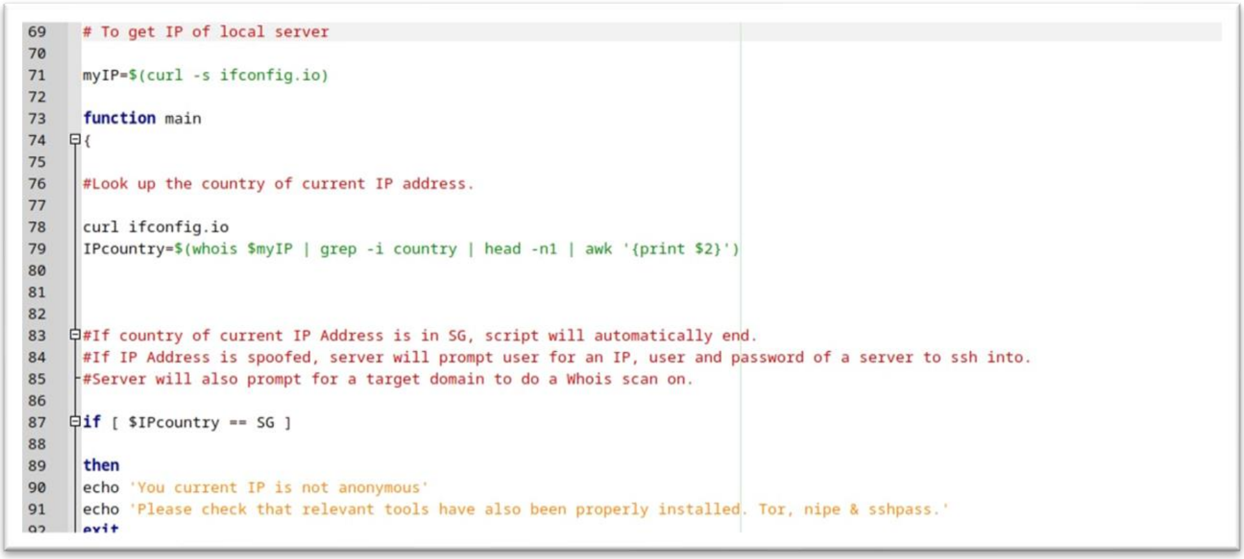
If programs are not installed.



```
┌──(kali㉿kali)-[~]
└─$ bash networkproject.sh
 [#] Log file exists.
[#] tor is already installed
[#] sshpass is already installed
[#] nipe is installed.
116.14.75.177
You current IP is not anonymous
Please check that relevant tools have also been properly installed. Tor, nipe & sshpass.

┌──(kali㉿kali)-[~]
└─$ █
```

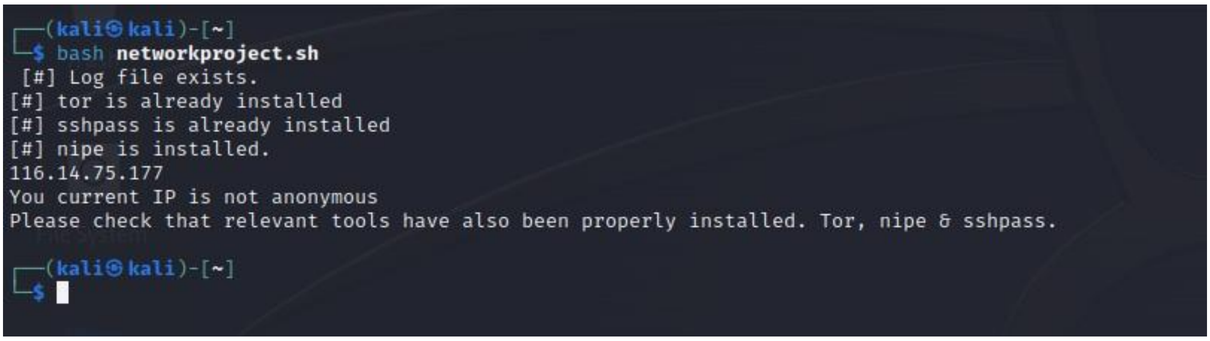
When programs are installed.

Checking Of Spoofed IP



```
# To get IP of local server

myIP=$(curl -s ifconfig.io)

function main
{

#Look up the country of current IP address.

curl ifconfig.io
IPcountry=$(whois $myIP | grep -i country | head -n1 | awk '{print $2}')



#If country of current IP Address is in SG, script will automatically end.
#If IP Address is spoofed, server will prompt user for an IP, user and password of a server to ssh into.
#Server will also prompt for a target domain to do a Whois scan on.

if [ $IPcountry == SG ]

then
echo 'You current IP is not anonymous'
echo 'Please check that relevant tools have also been properly installed. Tor, nipe & sshpass.'
exit
```

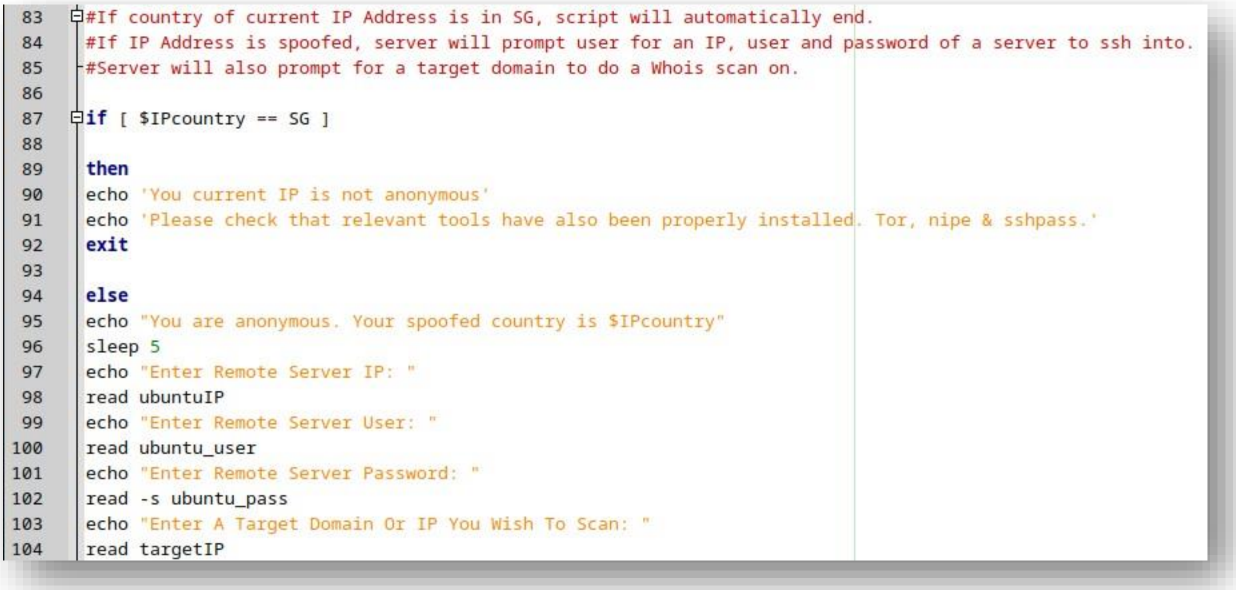
The script then runs a check on the host’s server’s IP address and runs a whois on the IP. If the country of IP is from SG the script will tell the user that the IP is not spoofed (and prompt user to check if programs have been properly installed) and exit immediately. If the IP is spoofed, the script will continue.



```
┌──(kali㉿kali)-[~]
└─$ bash networkproject.sh
 [#] Log file exists.
[#] tor is already installed
[#] sshpass is already installed
[#] nipe is installed.
116.14.75.177
You current IP is not anonymous
Please check that relevant tools have also been properly installed. Tor, nipe & sshpass.

┌──(kali㉿kali)-[~]
└─$ █
```

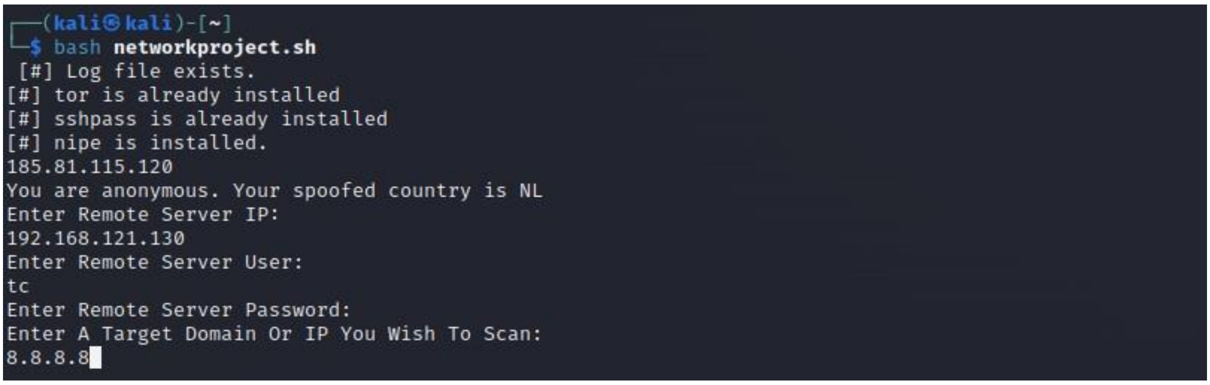
If the current IP has not been spoofed.

Entering Credentials



```
#If country of current IP Address is in SG, script will automatically end.
#If IP Address is spoofed, server will prompt user for an IP, user and password of a server to ssh into.
#Server will also prompt for a target domain to do a Whois scan on.

if [ $IPcountry == SG ]

then
echo 'You current IP is not anonymous'
echo 'Please check that relevant tools have also been properly installed. Tor, nipe & sshpass.'
exit

else
echo "You are anonymous. Your spoofed country is $IPcountry"
sleep 5
echo "Enter Remote Server IP: "
read ubuntuIP
echo "Enter Remote Server User: "
read ubuntu_user
echo "Enter Remote Server Password: "
read -s ubuntu_pass
echo "Enter A Target Domain Or IP You Wish To Scan: "
read targetIP
```

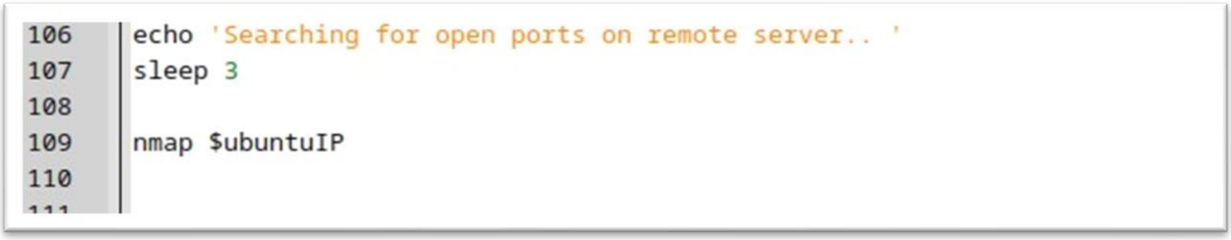
If the IP Address has been spoofed, the script will prompt the user to enter the remote server's IP address, username and password. The script will also request a target IP address to run a whois scan on the remote server.



```
┌──(kali㉿kali)-[~]
└─$ bash networkproject.sh
 [#] Log file exists.
[#] tor is already installed
[#] sshpass is already installed
[#] nipe is installed.
185.81.115.120
You are anonymous. Your spoofed country is NL
Enter Remote Server IP:
192.168.121.130
Enter Remote Server User:
tc
Enter Remote Server Password:
Enter A Target Domain Or IP You Wish To Scan:
8.8.8.8
```

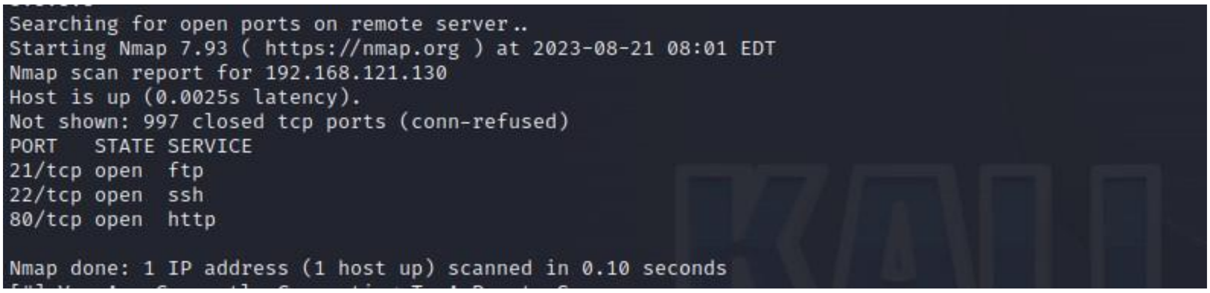
Here it shows the current spoofed IP address or server and country which the address belongs to. The user is then required to enter the remote server's IP address, user, password and target IP to run a whois scan on.

Nmap Scan



```
echo 'Searching for open ports on remote server.. '
sleep 3

nmap $ubuntuIP

```

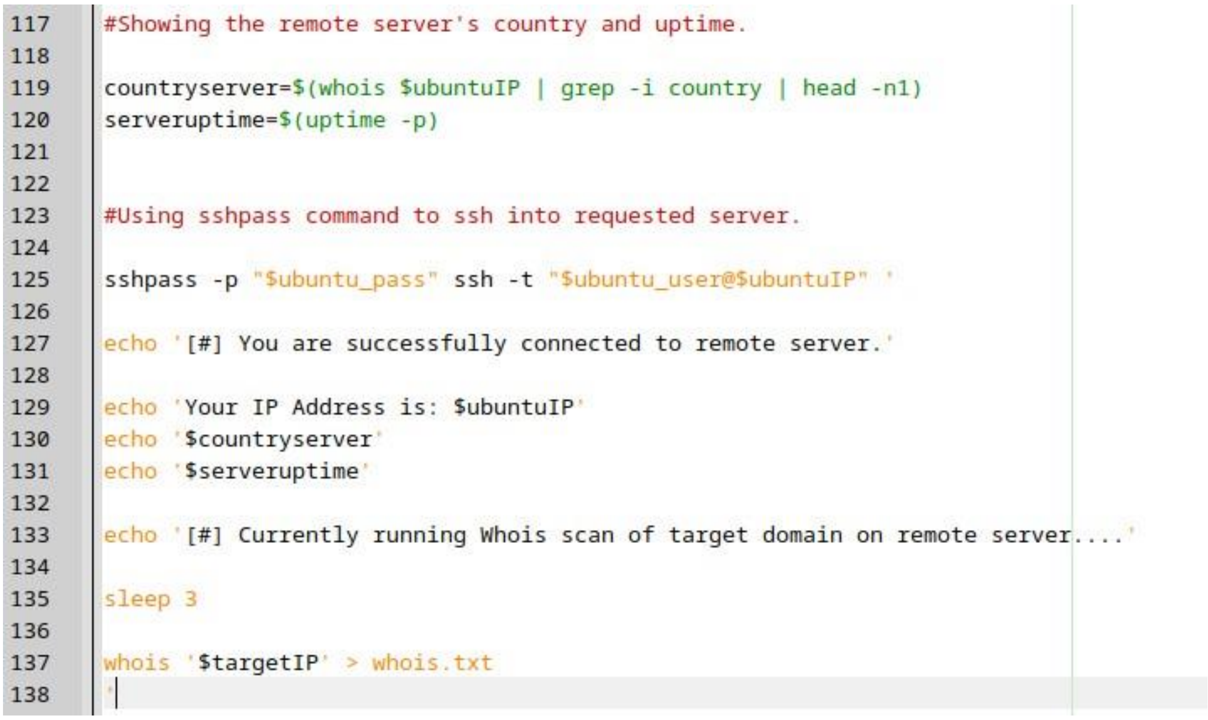
The script will then run a nmap scan on the given remote server to check for open ports.



```
Searching for open ports on remote server..
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-21 08:01 EDT
Nmap scan report for 192.168.121.130
Host is up (0.0025s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds
```

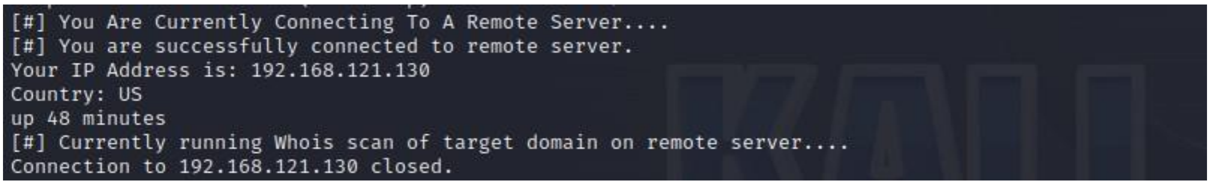
Here it shows that an nmap scan has been done and ports 21, 22 and 80 are open.

SSH Pass



```
#Showing the remote server's country and uptime.

countryserver=$(whois $ubuntuIP | grep -i country | head -n1)
serveruptime=$(uptime -p)


#Using sshpass command to ssh into requested server.

sshpass -p "$ubuntu_pass" ssh -t "$ubuntu_user@$ubuntuIP" '

echo '[#] You are successfully connected to remote server.'

echo 'Your IP Address is: $ubuntuIP'
echo '$countryserver'
echo '$serveruptime'

echo '[#] Currently running Whois scan of target domain on remote server....'

sleep 3

whois '$targetIP' > whois.txt
'
```

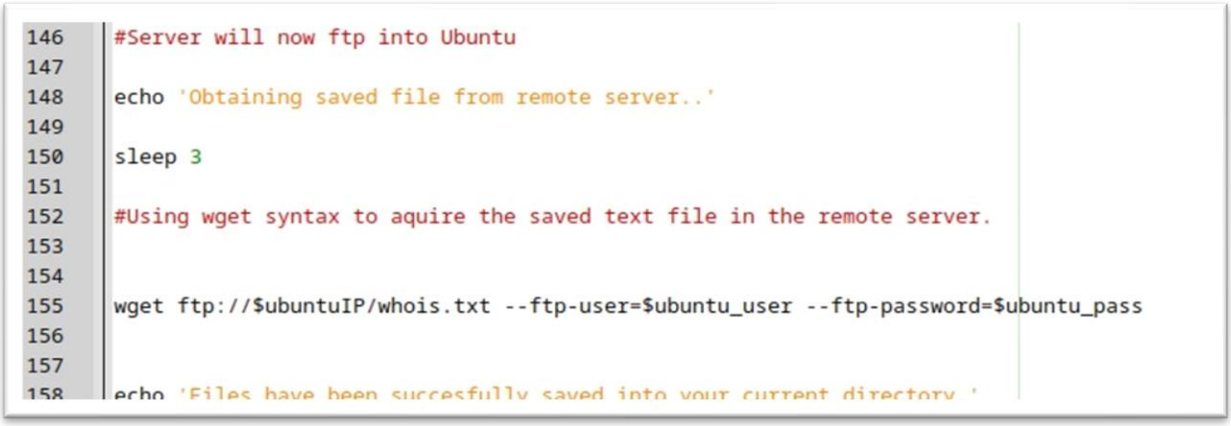
SSH pass is a program that automates tasks in a remote server immediately after connecting from the local server using ssh. Here, the sshpass command helps me echo out the remote server's IP address, country and also it's uptime. It also runs a whois scan on the given target's IP address in the remote server and saves the results in a text file.



```
[#] You Are Currently Connecting To A Remote Server....
[#] You are successfully connected to remote server.
Your IP Address is: 192.168.121.130
Country: US
up 48 minutes
[#] Currently running Whois scan of target domain on remote server....
Connection to 192.168.121.130 closed.
```

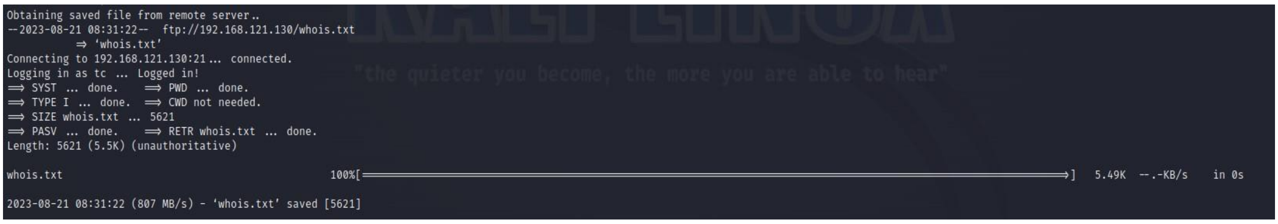
The sshpass will automatically exit the remote server after running all specified tasks.

Retrieving File



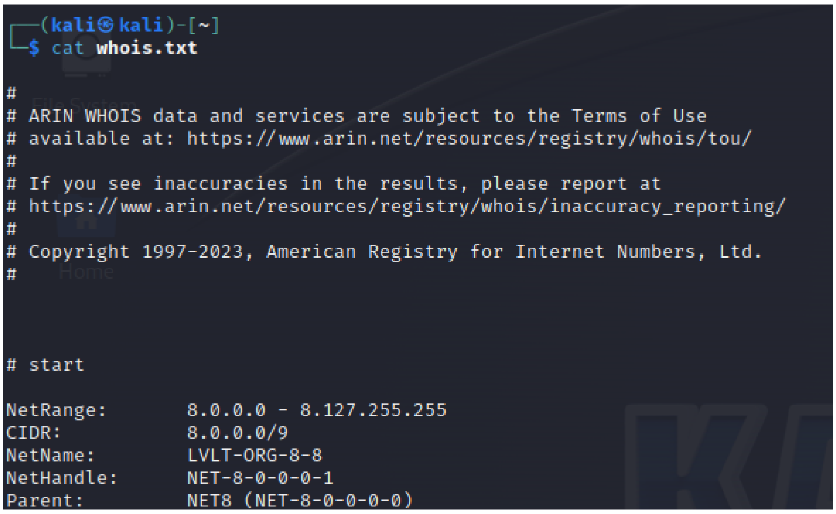
```
#Server will now ftp into Ubuntu

echo 'Obtaining saved file from remote server..'

sleep 3

#Using wget syntax to aquire the saved text file in the remote server.


wget ftp://$ubuntuIP/whois.txt --ftp-user=$ubuntu_user --ftp-password=$ubuntu_pass


echo 'Files have been succesfully saved into your current directory.'
```

The script then uses a wget command to retrieve the saved text file from the remote server through ftp and save it in the local server's current directory.



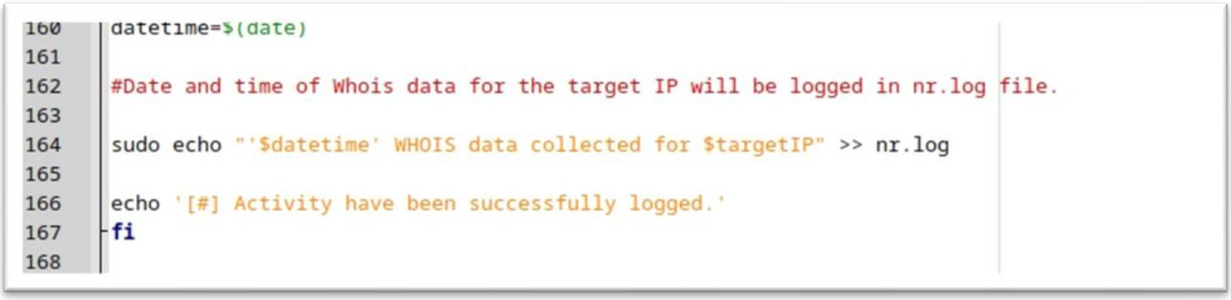
```
Obtaining saved file from remote server..
--2023-08-21 08:31:22--  ftp://192.168.121.130/whois.txt
           ⇒ 'whois.txt'
Connecting to 192.168.121.130:21... connected.
Logging in as tc ... Logged in!
⟹ SYST ... done.    ⟹ PWD ... done.
⟹ TYPE I ... done.  ⟹ CWD not needed.
⟹ SIZE whois.txt ... 5621
⟹ PASV ... done.    ⟹ RETR whois.txt ... done.
Length: 5621 (5.5K) (unauthoritative)

whois.txt                                  100%[===================================================>]  5.49K  --.-KB/s    in 0s

2023-08-21 08:31:22 (807 MB/s) - 'whois.txt' saved [5621]
```

Output showing that a whois.txt has been retrieved.



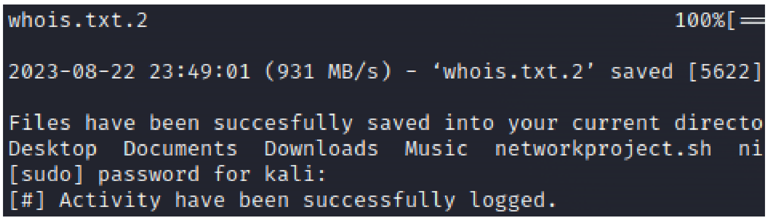
```
┌──(kali㉿kali)-[~]
└─$ cat whois.txt

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2023, American Registry for Internet Numbers, Ltd.
#



# start

NetRange:       8.0.0.0 - 8.127.255.255
CIDR:           8.0.0.0/9
NetName:        LVLT-ORG-8-8
NetHandle:      NET-8-0-0-0-1
Parent:         NET8 (NET-8-0-0-0-0)
```

Log File

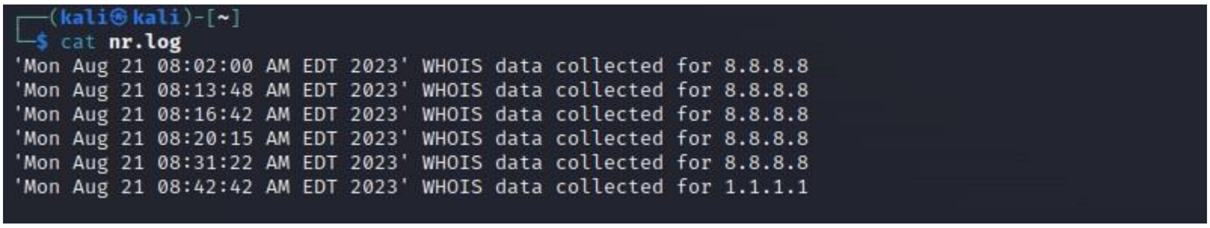


```
datetime=$(date)

#Date and time of Whois data for the target IP will be logged in nr.log file.

sudo echo "'$datetime' WHOIS data collected for $targetIP" >> nr.log

echo '[#] Activity have been successfully logged.'
fi

```

The script will finish of with printing the current date, time and specified remote server of collected information in the nr.log file that was created at the beginning of the script.



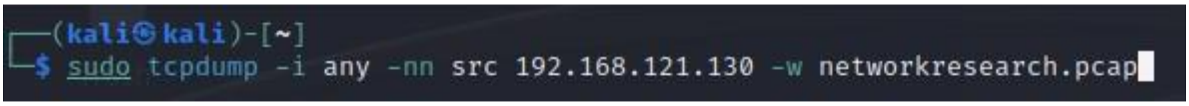
```
whois.txt.2                                         100%[═

2023-08-22 23:49:01 (931 MB/s) - ‘whois.txt.2’ saved [5622]

Files have been succesfully saved into your current directo
Desktop  Documents  Downloads  Music  networkproject.sh  ni
[sudo] password for kali:
[#] Activity have been successfully logged.
```



```
┌──(kali㉿kali)-[~]
└─$ cat nr.log
'Mon Aug 21 08:02:00 AM EDT 2023' WHOIS data collected for 8.8.8.8
'Mon Aug 21 08:13:48 AM EDT 2023' WHOIS data collected for 8.8.8.8
'Mon Aug 21 08:16:42 AM EDT 2023' WHOIS data collected for 8.8.8.8
'Mon Aug 21 08:20:15 AM EDT 2023' WHOIS data collected for 8.8.8.8
'Mon Aug 21 08:31:22 AM EDT 2023' WHOIS data collected for 8.8.8.8
'Mon Aug 21 08:42:42 AM EDT 2023' WHOIS data collected for 1.1.1.1
```

Discussion

To further dive into what happened during the automated attack, I ran a tcpdump from Kali Linux into the remote server to capture packets as such that the remote server is the host while the attack was happening. The syntax is as follows:



```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -i any -nn src 192.168.121.130 -w networkresearch.pcap
```

Using tcpdump requires elevated privileges which is why sudo is required before running command in this case.

The -i flag is to specify the interface in which the packets are captured, which in this case ‘any’ meaning all interfaces.

Tcpdump resolves IP addresses and port numbers by default so by adding the -nn flag, the results will show IP addresses and port numbers instead to easier understand the conversations of the packets send and received.

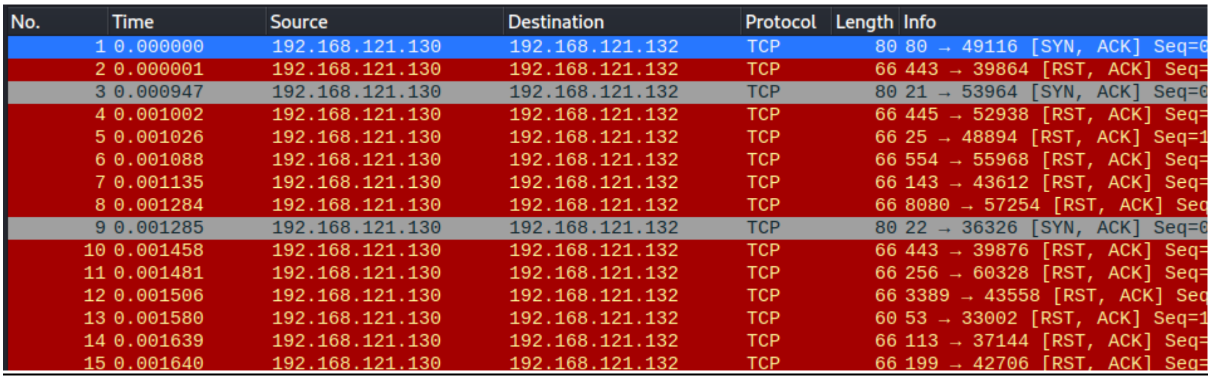
Src <remote server’s IP> is to run the tcpdump such that the packets will be read as the remote server being the host.

Lastly the -w flag is to save the output after the listening is stopped into a file. In this case I saved the output into a .pcap file so that I can display the output on wireshark which makes analysing simpler.

Captured pcap file:

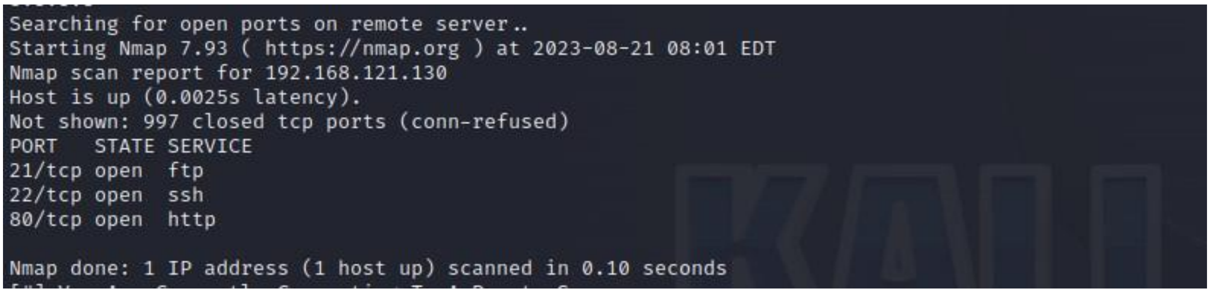
networkresearch.pcap

Wireshark

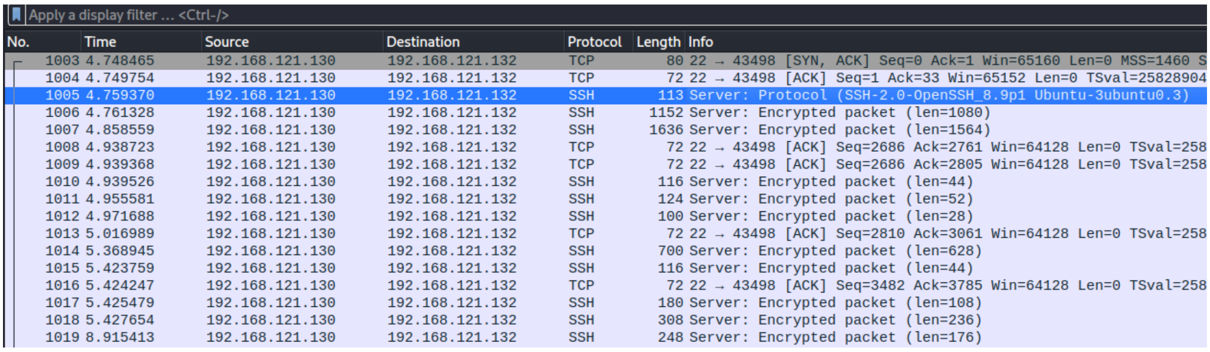


```
No.  Time       Source           Destination      Protocol  Length Info
   1 0.000000   192.168.121.130  192.168.121.132  TCP         80 80 → 49116 [SYN, ACK] Seq=0
   2 0.000001   192.168.121.130  192.168.121.132  TCP         66 443 → 39864 [RST, ACK] Seq=
   3 0.000947   192.168.121.130  192.168.121.132  TCP         80 21 → 53964 [SYN, ACK] Seq=0
   4 0.001002   192.168.121.130  192.168.121.132  TCP         66 445 → 52938 [RST, ACK] Seq=
   5 0.001026   192.168.121.130  192.168.121.132  TCP         66 25 → 48894 [RST, ACK] Seq=1
   6 0.001088   192.168.121.130  192.168.121.132  TCP         66 554 → 55968 [RST, ACK] Seq=
   7 0.001135   192.168.121.130  192.168.121.132  TCP         66 143 → 43612 [RST, ACK] Seq=
   8 0.001284   192.168.121.130  192.168.121.132  TCP         66 8080 → 57254 [RST, ACK] Seq
   9 0.001285   192.168.121.130  192.168.121.132  TCP         80 22 → 36326 [SYN, ACK] Seq=0
  10 0.001458   192.168.121.130  192.168.121.132  TCP         66 443 → 39876 [RST, ACK] Seq=
  11 0.001481   192.168.121.130  192.168.121.132  TCP         66 256 → 60328 [RST, ACK] Seq=
  12 0.001506   192.168.121.130  192.168.121.132  TCP         66 3389 → 43558 [RST, ACK] Seq
  13 0.001580   192.168.121.130  192.168.121.132  TCP         60 53 → 33002 [RST, ACK] Seq=1
  14 0.001639   192.168.121.130  192.168.121.132  TCP         66 113 → 37144 [RST, ACK] Seq=
  15 0.001640   192.168.121.130  192.168.121.132  TCP         66 199 → 42706 [RST, ACK] Seq=
```

Here, you can see that the remote server started communicating with the local server when the script tried to do nmap scan on the remote server to discover open ports. The local server is trying to establish a tcp 3-way handshake with the remote server by sending SYN packet on the scanned ports. If a port is open and listening on the remote server, it will reply to the local server with a SYN, ACK packet. From this screenshot, we can see that port 80, 21 and 22 are open which match the output of the Nmap scan we saw earlier.

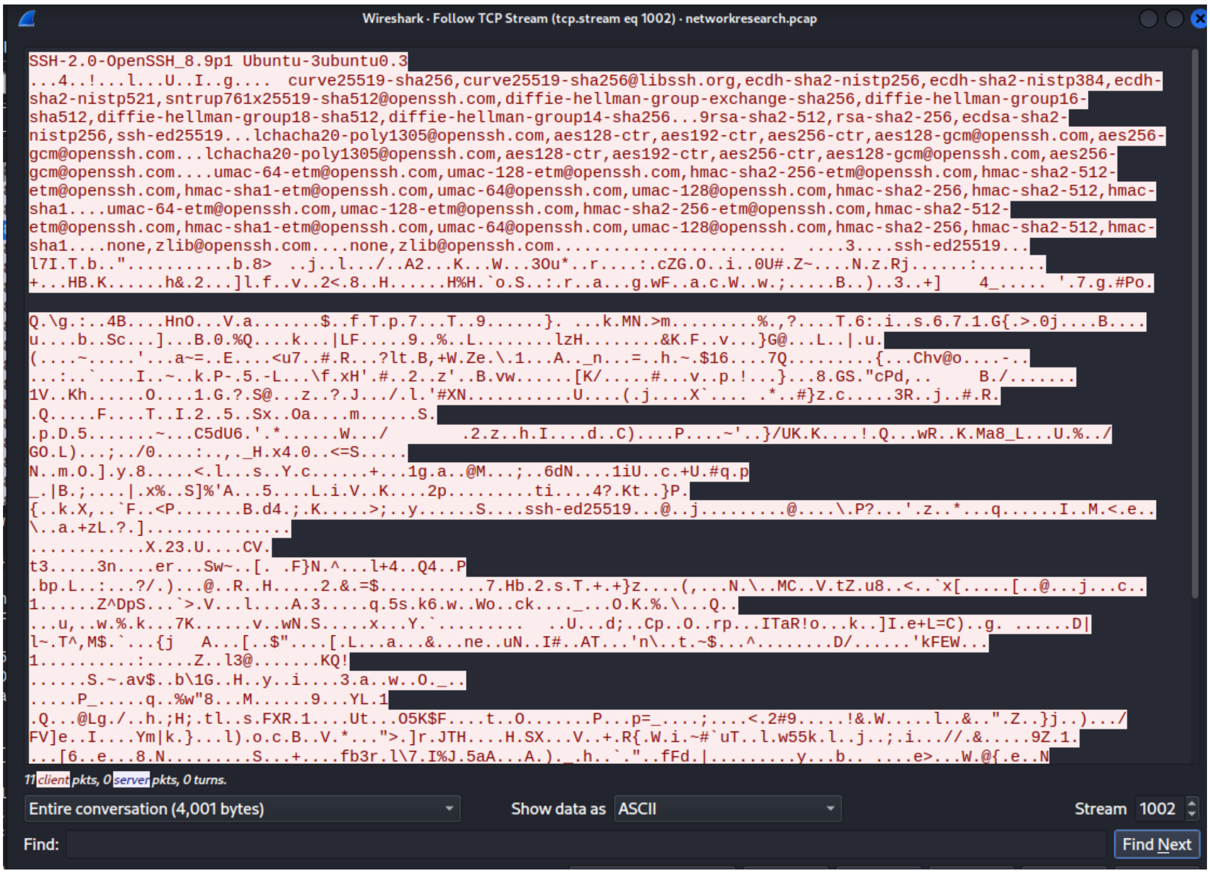


```
Searching for open ports on remote server..
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-21 08:01 EDT
Nmap scan report for 192.168.121.130
Host is up (0.0025s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds
```



```
No.   Time      Source           Destination      Protocol Length Info
 1003 4.748465  192.168.121.130  192.168.121.132  TCP        80 22 → 43498 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 S
 1004 4.749754  192.168.121.130  192.168.121.132  TCP        72 22 → 43498 [ACK] Seq=1 Ack=33 Win=65152 Len=0 TSval=25828904
 1005 4.759370  192.168.121.130  192.168.121.132  SSH       113 Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.3)
 1006 4.761328  192.168.121.130  192.168.121.132  SSH      1152 Server: Encrypted packet (len=1080)
 1007 4.858559  192.168.121.130  192.168.121.132  SSH      1636 Server: Encrypted packet (len=1564)
 1008 4.938723  192.168.121.130  192.168.121.132  TCP        72 22 → 43498 [ACK] Seq=2686 Ack=2761 Win=64128 Len=0 TSval=258
 1009 4.939368  192.168.121.130  192.168.121.132  TCP        72 22 → 43498 [ACK] Seq=2686 Ack=2805 Win=64128 Len=0 TSval=258
 1010 4.939526  192.168.121.130  192.168.121.132  SSH       116 Server: Encrypted packet (len=44)
 1011 4.955581  192.168.121.130  192.168.121.132  SSH       124 Server: Encrypted packet (len=52)
 1012 4.971688  192.168.121.130  192.168.121.132  SSH       100 Server: Encrypted packet (len=28)
 1013 5.016989  192.168.121.130  192.168.121.132  TCP        72 22 → 43498 [ACK] Seq=2810 Ack=3061 Win=64128 Len=0 TSval=258
 1014 5.368945  192.168.121.130  192.168.121.132  SSH       700 Server: Encrypted packet (len=628)
 1015 5.423759  192.168.121.130  192.168.121.132  SSH       116 Server: Encrypted packet (len=44)
 1016 5.424247  192.168.121.130  192.168.121.132  TCP        72 22 → 43498 [ACK] Seq=3482 Ack=3785 Win=64128 Len=0 TSval=258
 1017 5.425479  192.168.121.130  192.168.121.132  SSH       180 Server: Encrypted packet (len=108)
 1018 5.427654  192.168.121.130  192.168.121.132  SSH       308 Server: Encrypted packet (len=236)
 1019 8.915413  192.168.121.130  192.168.121.132  SSH       248 Server: Encrypted packet (len=176)
```

Here, you can see when local server did an ssh (Secure Shell) on the remote server using the OpenSSH service on ubuntu. However, because ssh is a secure protocol, the packet is encrypted and not recorded in plaintext.



Wireshark · Follow TCP Stream (tcp.stream eq 1002) · networkresearch.pcap

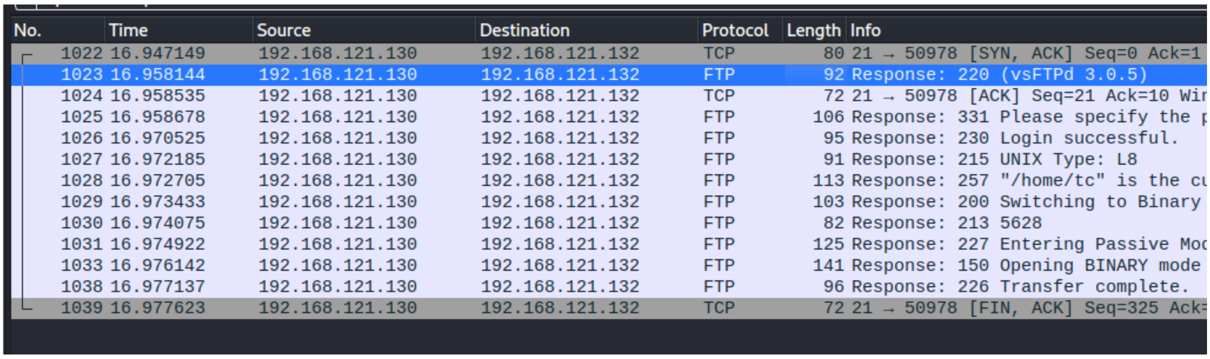
```
SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.3
...4..!...l...U..I..g....  curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-
sha2-nistp521,sntrup761x25519-sha512@openssh.com,diffie-hellman-group-exchange-sha256,diffie-hellman-group16-
sha512,diffie-hellman-group18-sha512,diffie-hellman-group14-sha256...9rsa-sha2-512,rsa-sha2-256,ecdsa-sha2-
nistp256,ssh-ed25519...lchacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-
gcm@openssh.com...lchacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-
gcm@openssh.com....umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-
etm@openssh.com,hmac-sha1-etm@openssh.com,umac-64@openssh.com,umac-128@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-
sha1....umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-
etm@openssh.com,hmac-sha1-etm@openssh.com,umac-64@openssh.com,umac-128@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-
sha1....none,zlib@openssh.com....none,zlib@openssh.com........................  ....3....ssh-ed25519...
l7I.T.b..".........b.8>  ..j..l.../..A2...K...W...3Ou*..r....:.cZG.O..i..0U#.Z~....N.z.Rj......:.......
+...HB.K......h&.2...]l.f..v..2<.8..H......H%H.`o.S..:.r..a...g.wF..a.c.W..w.;.....B..)..3..+]    4_.....  '.7.g.#Po.

Q.\g.:..4B....HnO...V.a.......$..f.T.p.7...T..9......}.  ...k.MN.>m.........%.,?....T.6:.i..s.6.7.1.G{.>.0j....B....
u....b..Sc...]...B.0.%Q....k...|LF.....9..%..L........lzH........&K.F..v...}G@...L..|.u.
(....~.....'...a~=..E....<u7..#.R...?lt.B,+W.Ze.\.1...A.._n...=..h.~.$16....7Q.........{...Chv@o....-..
...:..`....I..~..k.P-.5.-L...\f.xH'.#..2..z'..B.vw......[K/.....#...v..p.!...}...8.GS."cPd,..     B./.......
1V..Kh......O....1.G.?.S@...z..?.J.../.l.'#XN..........U....(.j....X`....  .*..#}z.c.....3R..j..#.R.
.Q.....F....T..I.2..5..Sx..Oa....m......S.
.p.D.5.......~...C5dU6.'.*......W.../        .2.z..h.I....d..C)....P....~'..}/UK.K....!.Q...wR..K.Ma8_L...U.%../
GO.L)...;../0....:..,._H.x4.0..<=S.....
N..m.O.].y.8.....<.l...s..Y.c......+...1g.a..@M...;..6dN....1iU..c.+U.#q.p
_.|B.;....|.x%..S]%'A...5....L.i.V..K....2p.........ti....4?.Kt..}P.
{..k.X,..`F..<P.......B.d4.;.K.....>;..y......S....ssh-ed25519...@..j.........@....\.P?...'.z..*...q......I..M.<.e..
\..a.+zL.?.]...............
............X.23.U....CV.
t3.....3n....er...Sw~..[.  .F}N.^...l+4..Q4..P
.bp.L..:...?/.)...@..R..H.....2.&.=$...........7.Hb.2.s.T.+.+}z....(,...N.\..MC..V.tZ.u8..<..`x[.....[..@...j...c..
1......Z^DpS...`>.V...l....A.3.....q.5s.k6.w..Wo..ck...._...O.K.%.\...Q..
...u,..w.%.k...7K......v..wN.S.....x...Y.`.........   ..U...d;..Cp..O..rp...ITaR!o...k..]I.e+L=C)..g.  ......D|
l~.T^,M$.`...{j   A...[..$"....[.L...a...&...ne..uN..I#..AT...'n\..t.~$...^........D/......'kFEW...
1.........:.....Z..l3@.......KQ!
......S.~.av$..b\1G..H..y..i....3.a..w..O._..
.....P_.....q..%w"8...M......9...YL.1
.Q...@Lg./..h.;H;.tl..s.FXR.1....Ut...O5K$F....t..O.......P...p=_....;....<.2#9.....!&.W.....l..&..".Z..}j..).../
FV]e..I....Ym|k.}...l).o.c.B..V.*..."">.]r.JTH....H.SX...V..+.R{.W.i.~#`uT..l.w55k.l..j..;.i...//.&.....9Z.1.
...[6..e...8.N.........S...+....fb3r.l\7.I%J.5aA...A.)._.h..`."..fFd.|.........y...b..  ....e>...W.@{.e..N
```

11 client pkts, 0 server pkts, 0 turns.

Entire conversation (4,001 bytes) Show data as ASCII Stream 1002

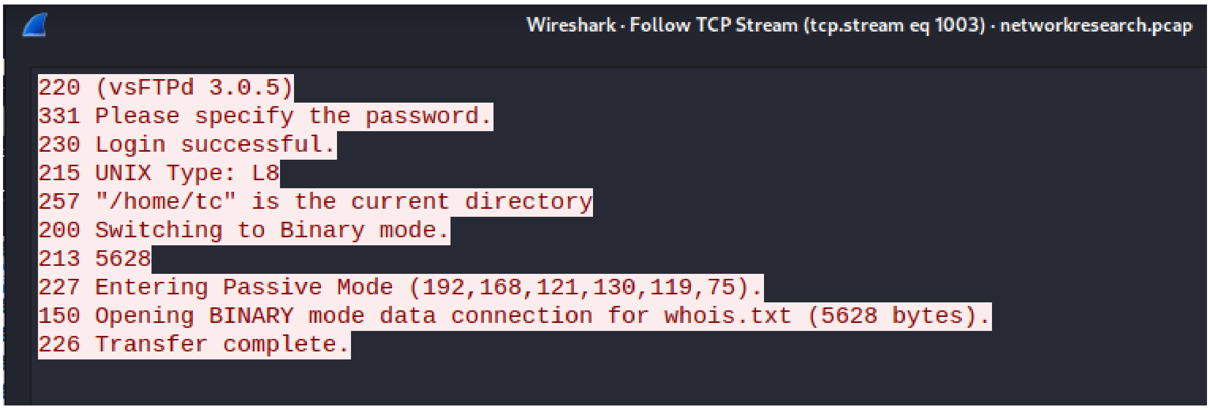
When following the TCP stream of the SSH, it is in encrypted and not in readable format.

FTP



No.	Time	Source	Destination	Protocol	Length	Info
1022	16.947149	192.168.121.130	192.168.121.132	TCP	80	21 → 50978 [SYN, ACK] Seq=0 Ack=1
1023	16.958144	192.168.121.130	192.168.121.132	FTP	92	Response: 220 (vsFTPd 3.0.5)
1024	16.958535	192.168.121.130	192.168.121.132	TCP	72	21 → 50978 [ACK] Seq=21 Ack=10 Wir
1025	16.958678	192.168.121.130	192.168.121.132	FTP	106	Response: 331 Please specify the p
1026	16.970525	192.168.121.130	192.168.121.132	FTP	95	Response: 230 Login successful.
1027	16.972185	192.168.121.130	192.168.121.132	FTP	91	Response: 215 UNIX Type: L8
1028	16.972705	192.168.121.130	192.168.121.132	FTP	113	Response: 257 "/home/tc" is the cu
1029	16.973433	192.168.121.130	192.168.121.132	FTP	103	Response: 200 Switching to Binary
1030	16.974075	192.168.121.130	192.168.121.132	FTP	82	Response: 213 5628
1031	16.974922	192.168.121.130	192.168.121.132	FTP	125	Response: 227 Entering Passive Moc
1033	16.976142	192.168.121.130	192.168.121.132	FTP	141	Response: 150 Opening BINARY mode
1038	16.977137	192.168.121.130	192.168.121.132	FTP	96	Response: 226 Transfer complete.
1039	16.977623	192.168.121.130	192.168.121.132	TCP	72	21 → 50978 [FIN, ACK] Seq=325 Ack=

However, when we follow the FTP stream, it is very different compared to what we saw in the SSH protocol. Everything is in plaintext, and we can see exactly what happened during the connection. Let's further breakdown and see what we can understand from the screen capture below.



Wireshark · Follow TCP Stream (tcp.stream eq 1003) · networkresearch.pcap

```
220 (vsFTPd 3.0.5)
331 Please specify the password.
230 Login successful.
215 UNIX Type: L8
257 "/home/tc" is the current directory
200 Switching to Binary mode.
213 5628
227 Entering Passive Mode (192,168,121,130,119,75).
150 Opening BINARY mode data connection for whois.txt (5628 bytes).
226 Transfer complete.
```

Firstly, we can see the service used for the FTP which is vsFTPd 3.0.5. Which stands for Very Secure File Transfer Protocol Daemon. But is it really secure?

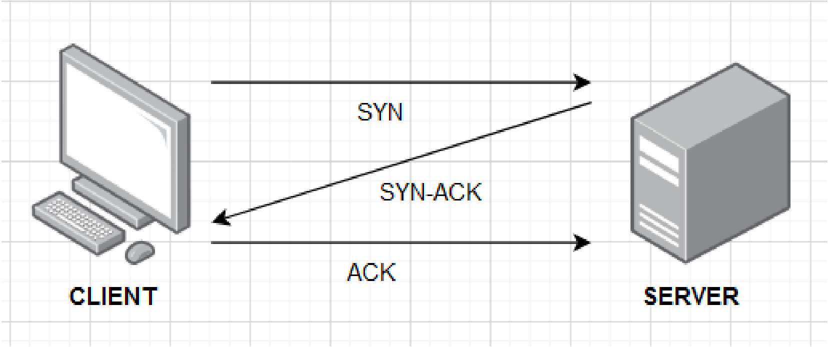
Also, we can see that the local server has logged in successfully into the remote server via ftp.

The local server then goes into the /home/tc directory and successfully manages to get the 'whois.txt' file from the remote server.

From the example given, we can see that FTP protocol is not exactly the most secure in terms of not leaving any network footprints. Let's further dive in on what is FTP and how it works.

What is FTP?

File Transfer Protocol (FTP), as its name suggests, is mainly used for sending and receiving files between different servers over a network. For an FTP connection to be established, there is usually a client and a server. FTP is done through TCP/IP network, so an internet connection is required on both sides. There are two types of connections that happens during FTP protocol. A control connection happens first where the client and server communicate to get a stable connection. Then, a data connection happens which is used for the transfer of files. The client starts by sending a connection request (SYN packet) to the server usually on port 21, and the server responds (SYN, ACK) with client acknowledging (ACK) the respond and they both create a stable connection to begin the data transfer process.

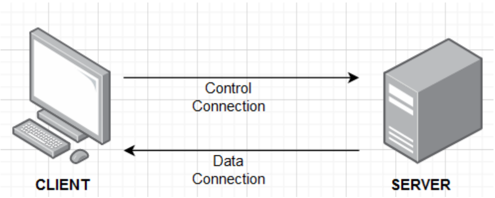


Usually, a username and password are required to log on to the server to establish a connection but there are servers that make their files publicly accessible to everyone without a need for credentials. These servers are known as anonymous FTP.

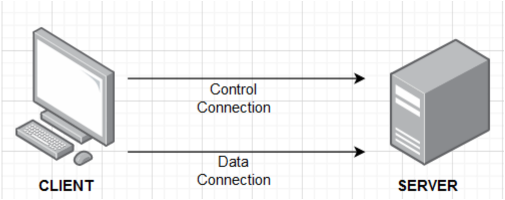
There are two modes of FTP data connection:

1. Active Mode: In an active connection, the client connects to the server from a random port number to the server's ftp port. The client then tells the server what port the server should connect to and the server connects to the client to the specified port.
2. Passive Mode: In a passive connection, the client first connects to the server's ftp port. The server then tells the client what port it should connect to for the data connection. The client then establishes another connection to the specified port number. This is usually more common due to client's firewalls blocking connections from the server.

Active Mode



Passive Mode



Conclusion

As we have seen how the script works and know how the FTP works, lets discuss the advantages and disadvantages of FTP protocol.

Advantages:

1. Relatively easy to use.
2. Fast.
3. Allow Transfer of multiple files and folders.
4. Allows resuming of file transfer if connection is lost.
5. No Limitations of size of files.

Disadvantages:

1. Traffic is not encrypted and can be read in plaintext.
2. Old protocol and uses multiple port connections that can be hindered by firewalls.
3. No data integrity verification.

How does it impact the CIA Triad?

The CIA Triad refers to 3 important factors that contributes to information security. Confidentiality, Integrity and Availability. Let's see how FTP impacts the CIA Triad on each factor respectively.

Confidentiality:

As we have seen previously, data sent through FTP is not encrypted and can be read in plaintext. This directly impacts the confidentiality aspect of the data being transferred. Sensitive data such username, passwords and other contents of files can be easily retrieved by anyone that has access to the network.

Integrity:

FTP has no data integrity verification and files sent through FTP are susceptible to corruption such as incomplete file transactions, data compressions, file format issues and much more. This might impact the integrity of files and could be a serious issue if file contains important information or is a software which requires an uncorrupted file to be executed.

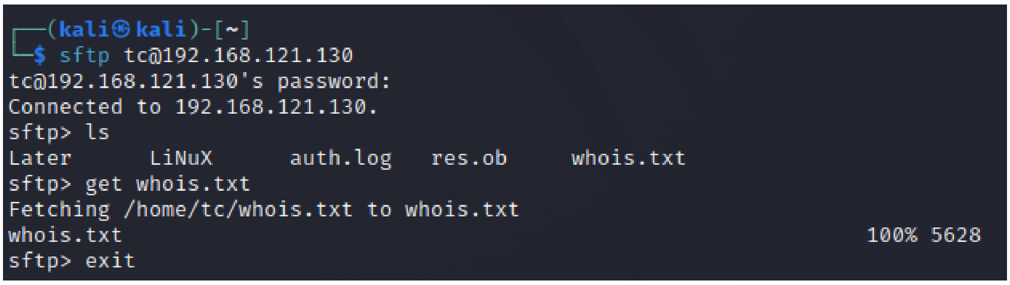
Availability:

Some FTP services do require a username and password for access. Like in the example given above, it was required of me to know the username and password of the remote server to retrieve the file from the server. Also, the FTP port is required to be open in the remote server. All these factors impact the availability aspect of the information directly.

In conclusion, traditional FTP might negatively impact the CIA Triad in terms of confidentiality and integrity due to lack of encryption and lack of data integrity verification respectively. Fortunately, there are alternative methods and further steps we can take to handle these risks and further improve security.

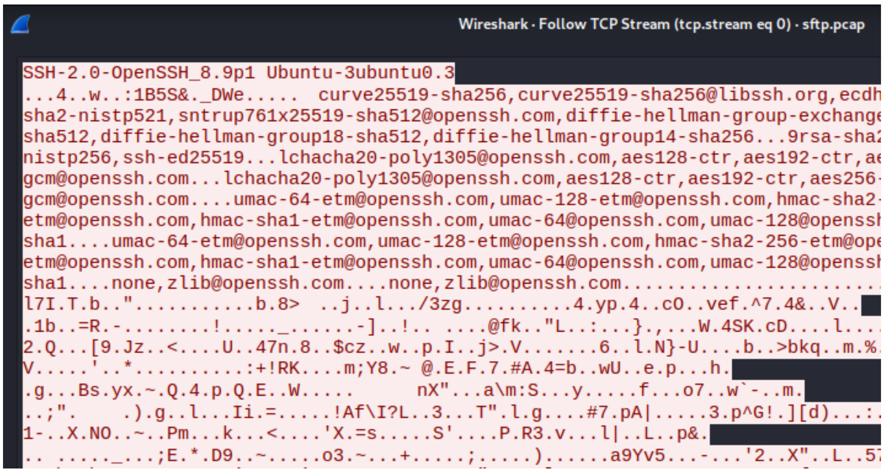
Recommendations

Encryption



```
┌──(kali㉿kali)-[~]
└─$ sftp tc@192.168.121.130
tc@192.168.121.130's password:
Connected to 192.168.121.130.
sftp> ls
Later       LiNuX       auth.log    res.ob      whois.txt
sftp> get whois.txt
Fetching /home/tc/whois.txt to whois.txt
whois.txt                                              100% 5628
sftp> exit
```

In this example, I am using sftp (Secure File Transfer Protocol) also known as SSH File Transfer Protocol to retrieve the same file that I got running the automated script. This is a better alternative compared to FTP because data received during this process is encrypted.



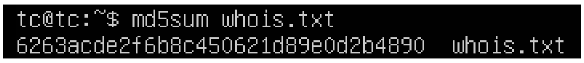
Wireshark · Follow TCP Stream (tcp.stream eq 0) · sftp.pcap

```
SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.3
...4..w..:1B5S&._DWe.....  curve25519-sha256,curve25519-sha256@libssh.org,ecdh
sha2-nistp521,sntrup761x25519-sha512@openssh.com,diffie-hellman-group-exchange
sha512,diffie-hellman-group18-sha512,diffie-hellman-group14-sha256...9rsa-sha2
nistp256,ssh-ed25519...lchacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,ae
gcm@openssh.com...lchacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-
gcm@openssh.com....umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-
etm@openssh.com,hmac-sha1-etm@openssh.com,umac-64@openssh.com,umac-128@openssh
sha1....umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@ope
etm@openssh.com,hmac-sha1-etm@openssh.com,umac-64@openssh.com,umac-128@openssh
sha1....none,zlib@openssh.com....none,zlib@openssh.com......................
l7I.T.b.."...........b.8>  ..j..l.../3zg..........4.yp.4..cO..vef.^7.4&..V..
.1b..=R.-.........!....._.......-]..!..  ....@fk.."L..:...}.,...W.4SK.cD....l...
2.Q...[9.Jz..<....U..47n.8..$cz..w..p.I..j>.V.......6..l.N}-U....b..>bkq..m.%.
V.....'..*..........:+!RK....m;Y8.~ @.E.F.7.#A.4=b..wU..e.p...h.
.g...Bs.yx.~.Q.4.p.Q.E..W.....      nX"...a\m:S...y.....f...o7..w`-..m.
..;".    .).g..l...Ii.=.....!Af\I?L..3...T".l.g....#7.pA|.....3.p^G!.][d)...:.
1-..X.NO..~..Pm...k...<....'X.=s.....S'....P.R3.v...l|..L..p&.
.. ....._...;E.*.D9..~.....o3.~...+.....;.....)......a9Yv5...-...'2..X"..L..57
```

The image above shows the tcp stream captured during the sftp file transfer. As you can see, the data is encrypted and is not easily readable. This will directly resolve the confidentiality issue we discussed on how FTP impacts the CIA Triad. Sensitive data such as usernames, passwords and file contents will no longer be easily compromised if network is intercepted.

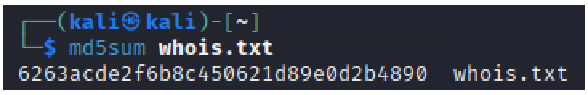
Integrity Check

Remote server:



```
tc@tc:~$ md5sum whois.txt
6263acde2f6b8c450621d89e0d2b4890  whois.txt
```

Local server:



```
┌──(kali㉿kali)-[~]
└─$ md5sum whois.txt
6263acde2f6b8c450621d89e0d2b4890  whois.txt
```

To further ensure the integrity of retrieved file, a hash comparison can be used on the original file and the retrieved file. An md5sum is done on both the remote server before the transfer and on the local server after the transfer in the example shown. This helps ensure that the contents of the file have not been corrupted during the transfer. This helps resolve the issue that we have impacting the Integrity segment of the CIA Triad. Although this does not actually directly change anything about the FTP protocol itself, it’s an extra step we can take to further improve overall information security.

References

Vivek Gite, (2023). sshpass – Login to ssh server with a password using a shell script [online] Available at: https://www.cyberciti.biz/faq/noninteractive-shell-script-ssh-password-provider/ [Accessed 22 Aug. 2023].

PenTest-duck, (2019). (Almost) All The Ways to File Transfer [online] Available at: https://medium.com/@PenTest_duck/almost-all-the-ways-to-file-transfer-1bd6bf710d65 [Accessed 22 Aug. 2023].

Ricardo Gerardi, (2020). An introduction to using tcpdump at the Linux command line [online] Available at: https://opensource.com/article/18/10/introduction-tcpdump [Accessed 24 Aug. 2023].

Cory Mitchell, (2023). What Is File Transfer Protocol (FTP) and What Is It Used for? [online] Available at: https://www.investopedia.com/terms/f/ftp-file-transfer-protocol.asp [Accessed 25 Aug 2023].

Geeksforgeeks.org (2021). File Transfer Protocol (FTP) [online] Available at: https://www.geeksforgeeks.org/file-transfer-protocol-ftp/ [Accessed 25 Aug 2023].

Fortinet.com (2023) CIA Triad [online] Available at: https://www.fortinet.com/resources/cyberglossary/cia-triad [Accessed 25 Aug 2023].