Indian Institute of Technology Roorkee

Computer Science and Engineering

Computer Networks Semester Project

Author

Nishant Parmar | 18114053 | CS1

nparmar@cs.iitr.ac.in

Contents

1. Introduction

2. Problem 1

2.1 Problem statement

2.2 Solution

3. Problem 2

3.1 Problem statement

3.2 Solution

4. Conclusion

4.1 Problem 1

4.2 Problem 2

A Tables

Introduction

A computer network is a telecommunications network that connects a collection of computers to allow communication and data exchange between systems, software applications, and users.Two devices are said to be networked when a process in one device is able to exchange information with a process in another device.

In this project we tend to enhance one’s understanding of the nuances and intricacies of the Physical and Network Layers of a Computer Network.

The project is divided int two sections.The first section attempts to show the level of security when using different protocols. The second section statement attempts to analyse network traffic captured using Wireshark and sieve out the top 3 visited websites by a user.

2: **Problem 1**

2.1 Problem Statement

To demonstrate password sniffing over different application layer protocols viz., HTTP,HTTPS, TELNET, FTP and SSH. Show if sensitive data such as username and passwords can be captured by anyonne monitoring the traffic or not in each of them.

2.2 Solution

First we have to capture the network traffic between a server and a host. Here we have used an Open Source Tool Wireshark . Wireshark is a data capturing program that "understands" the structure of different networking protocols. It can parse and display the fields, along with their meanings as specified by different networking protocols.

2.2.a HTTP

HTTP is a protocol which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.

Since there are no any encryption methods used in HTTP, there are chances of someone altering the content. That is the reason why HTTP is considered to be an insecure method prone to data integrity.

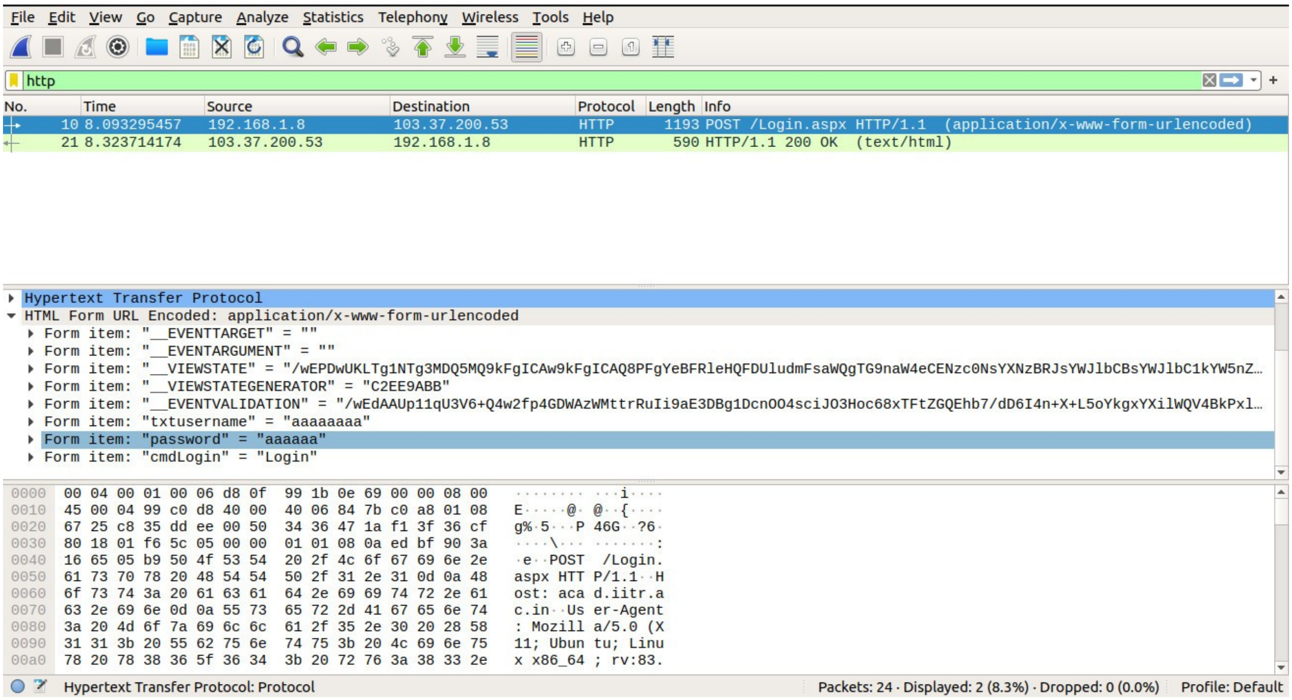
Procedure

We use Wireshark to capture the traffic between a website using http. The user will login into the website and submit login credentials. We filter the packets for http.

Analysis

Fig A shows the http request sent by the host PC to the website. Since the website uses HTTP, the exchanged traffic isn't encrypted hence it becomes very easy to sniff the packets exchanged.

Under **HTML Form URL Encode** we can easily find the USERNAME and PASSWORD beside "txtusername" and "password".



http

```
No.     Time          Source          Destination     Protocol  Length Info
     10 8.093295457   192.168.1.8     103.37.200.53   HTTP        1193 POST /Login.aspx HTTP/1.1  (application/x-www-form-urlencoded)
     21 8.323714174   103.37.200.53   192.168.1.8     HTTP         590 HTTP/1.1 200 OK  (text/html)
```

```
▸ Hypertext Transfer Protocol
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "__EVENTTARGET" = ""
  ▸ Form item: "__EVENTARGUMENT" = ""
  ▸ Form item: "__VIEWSTATE" = "/wEPDwUKLTg1NTg3MDQ5MQ9kFgICAw9kFgICAQ8PFgYeBFRleHQFDUludmFsaWQgTG9naW4eCENzc0NsYXNzBRJsYWJlbCBsYWJlbC1kYW5nZ…
  ▸ Form item: "__VIEWSTATEGENERATOR" = "C2EE9ABB"
  ▸ Form item: "__EVENTVALIDATION" = "/wEdAAUp11qU3V6+Q4w2fp4GDWAzWMttrRuIi9aE3DBg1DcnOO4sciJO3Hoc68xTFtZGQEhb7/dD6I4n+X+L5oYkgxYXilWQV4BkPxl…
  ▸ Form item: "txtusername" = "aaaaaaaa"
  ▸ Form item: "password" = "aaaaaa"
  ▸ Form item: "cmdLogin" = "Login"
```

```
0000  00 04 00 01 00 06 d8 0f  99 1b 0e 69 00 00 08 00   ········ ···i····
0010  45 00 04 99 c0 d8 40 00  40 06 84 7b c0 a8 01 08   E·····@· @··{····
0020  67 25 c8 35 dd ee 00 50  34 36 47 1a f1 3f 36 cf   g%·5···P 46G··?6·
0030  80 18 01 f6 5c 05 00 00  01 01 08 0a ed bf 90 3a   ····\··· ·······:
0040  16 65 05 b9 50 4f 53 54  20 2f 4c 6f 67 69 6e 2e   ·e··POST  /Login.
0050  61 73 70 78 20 48 54 54  50 2f 31 2e 31 0d 0a 48   aspx HTT P/1.1··H
0060  6f 73 74 3a 20 61 63 61  64 2e 69 69 74 72 2e 61   ost: aca d.iitr.a
0070  63 2e 69 6e 0d 0a 55 73  65 72 2d 41 67 65 6e 74   c.in··Us er-Agent
0080  3a 20 4d 6f 7a 69 6c 6c  61 2f 35 2e 30 20 28 58   : Mozill a/5.0 (X
0090  31 31 3b 20 55 62 75 6e  74 75 3b 20 4c 69 6e 75   11; Ubun tu; Linu
00a0  78 20 78 38 36 5f 36 34  3b 20 72 76 3a 38 33 2e   x x86_64 ; rv:83.
```

Hypertext Transfer Protocol: Protocol

Packets: 24 · Displayed: 2 (8.3%) · Dropped: 0 (0.0%)

Profile: Default

Figure A:Wireshark capture for http

2.2.b HTTPS

Hypertext Transfer Protocol Secure (**HTTPS**) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network, and is widely used on the Internet.

The principal motivations for HTTPS are utahentication of the accessed website, and protection of the privacy and integrity of the exchanged data while in transit.

Procedure

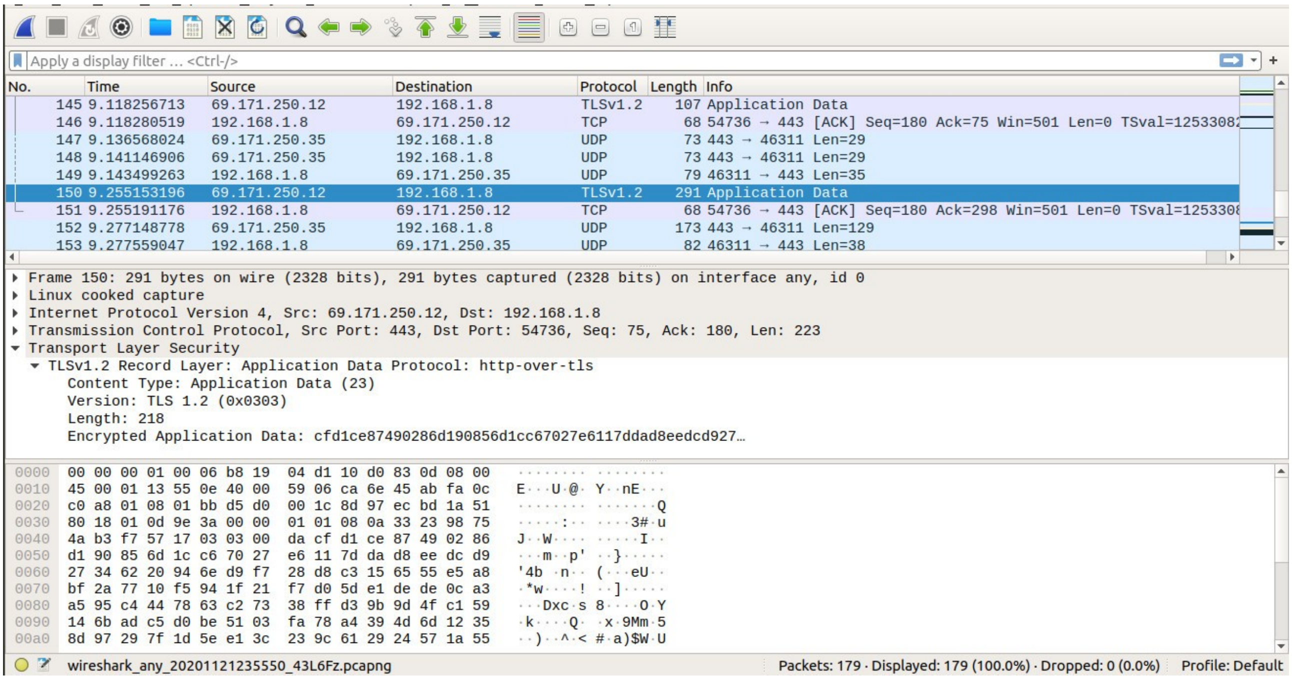
Same as http we use Wireshark to capture packets. We use youtube to analyse https protocol. The host will login to youtube with his credentials and thus we capture the network traffic between a server and a host.

Analysis

Fig B shows the https request sent by the host PC to the website. Since the

website uses HTTPS, the exchanged traffic is encrypted. This makes it next to impossible for the attacker to figure out what are the contents of the packets whcih have been exchanged.

In Transport Layer Security we can see the “Encrypted Application Data” which can not be encrypted.



```
No.  Time         Source          Destination     Protocol Length Info
 145 9.118256713  69.171.250.12   192.168.1.8     TLSv1.2    107 Application Data
 146 9.118280519  192.168.1.8     69.171.250.12   TCP         68 54736 → 443 [ACK] Seq=180 Ack=75 Win=501 Len=0 TSval=1253308
 147 9.136568024  69.171.250.35   192.168.1.8     UDP         73 443 → 46311 Len=29
 148 9.141146906  69.171.250.35   192.168.1.8     UDP         73 443 → 46311 Len=29
 149 9.143499263  192.168.1.8     69.171.250.35   UDP         79 46311 → 443 Len=35
 150 9.255153196  69.171.250.12   192.168.1.8     TLSv1.2    291 Application Data
 151 9.255191176  192.168.1.8     69.171.250.12   TCP         68 54736 → 443 [ACK] Seq=180 Ack=298 Win=501 Len=0 TSval=1253308
 152 9.277148778  69.171.250.35   192.168.1.8     UDP        173 443 → 46311 Len=129
 153 9.277559047  192.168.1.8     69.171.250.35   UDP         82 46311 → 443 Len=38
```

```
▸ Frame 150: 291 bytes on wire (2328 bits), 291 bytes captured (2328 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 69.171.250.12, Dst: 192.168.1.8
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 54736, Seq: 75, Ack: 180, Len: 223
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 218
      Encrypted Application Data: cfd1ce87490286d190856d1cc67027e6117ddad8eedcd927…
```

```
0000  00 00 00 01 00 06 b8 19  04 d1 10 d0 83 0d 08 00   ········ ········
0010  45 00 01 13 55 0e 40 00  59 06 ca 6e 45 ab fa 0c   E···U·@· Y··nE···
0020  c0 a8 01 08 01 bb d5 d0  00 1c 8d 97 ec bd 1a 51   ········ ·······Q
0030  80 18 01 0d 9e 3a 00 00  01 01 08 0a 33 23 98 75   ·····:·· ····3#·u
0040  4a b3 f7 57 17 03 03 00  da cf d1 ce 87 49 02 86   J··W···· ·····I··
0050  d1 90 85 6d 1c c6 70 27  e6 11 7d da d8 ee dc d9   ···m··p' ··}·····
0060  27 34 62 20 94 6e d9 f7  28 d8 c3 15 65 55 e5 a8   '4b ·n·· (···eU··
0070  bf 2a 77 10 f5 94 1f 21  f7 d0 5d e1 de de 0c a3   ·*w····! ··]·····
0080  a5 95 c4 44 78 63 c2 73  38 ff d3 9b 9d 4f c1 59   ···Dxc·s 8····O·Y
0090  14 6b ad c5 d0 be 51 03  fa 78 a4 39 4d 6d 12 35   ·k····Q· ·x·9Mm·5
00a0  8d 97 29 7f 1d 5e e1 3c  23 9c 61 29 24 57 1a 55   ··)··^·< #·a)$W·U
```

wireshark_any_20201121235550_43L6Fz.pcapng   Packets: 179 · Displayed: 179 (100.0%) · Dropped: 0 (0.0%)   Profile: Default

Figure B: Wireshark capture of an HTTPS exchange

2.2.c TELNET

Telnet is an application protocol used on the Internet or local area network to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection.There is no means of protection between the communicating parties, thus anyone listening to the same port as the host has open access to the messages exchanged.

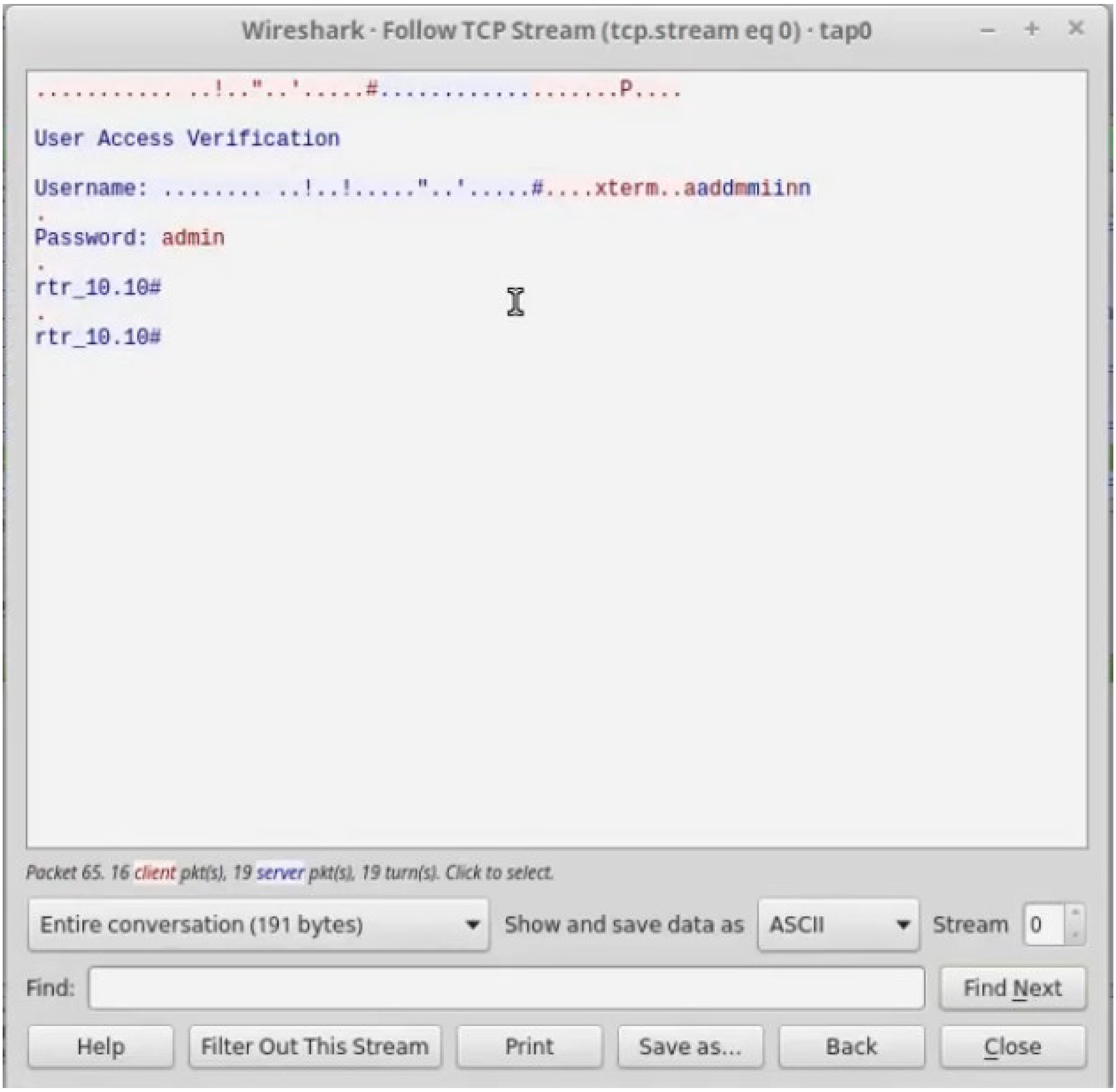
Procedure

We utilise Wireshark to capture the traffic in a telnet session initiated on the localhost itself. The command which will help us do so is telnet localhost followed by entering the necessary credentials.

By right clicking on the packet(any of the TELNET line) then selecting Follow and from the drop down menu selecting the TCP stream we can see the login ID and Password.

Analysis

Figure C shows the output of the packet. Such crucial information can aid the attacker in finding sensitive information about the contents of one’s PC. Therefore, it is clearly possible to find what was exchanged in case of Telnet.



```
........... ..!.."..'.....#..................P....

User Access Verification

Username: ........ ..!..!....."..'.....#....xterm..aaddmmiinn
.
Password: admin
.
rtr_10.10#
.
rtr_10.10#
```

Packet 65. 16 client pkt(s), 19 server pkt(s), 19 turn(s). Click to select.

Entire conversation (191 bytes) ▾ Show and save data as ASCII ▾ Stream 0

Find:

Figure C: Wireshark capture of a Telnet exchange

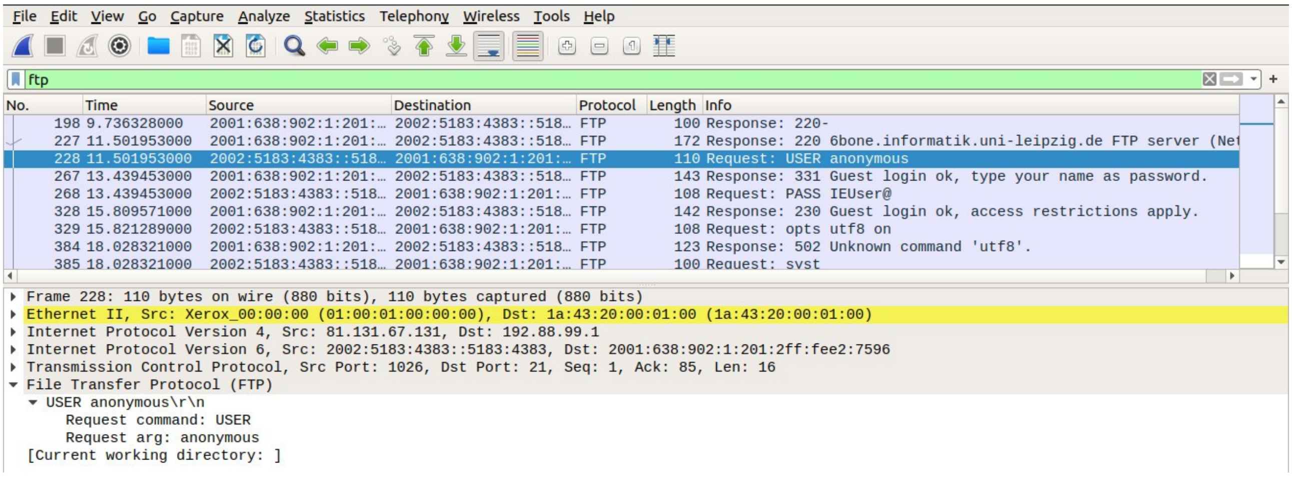
2.2.d FTP

Procedure

There is no good website that run on FTP .Thus I have downloaded FTP capture from here . U can find a file named “FTPv6-1.cap (Microsoft Network Monitor) FTP packets (IPv6) ” which can be downloaded and analysed.Filter packets from ftp and select User and Pass from info

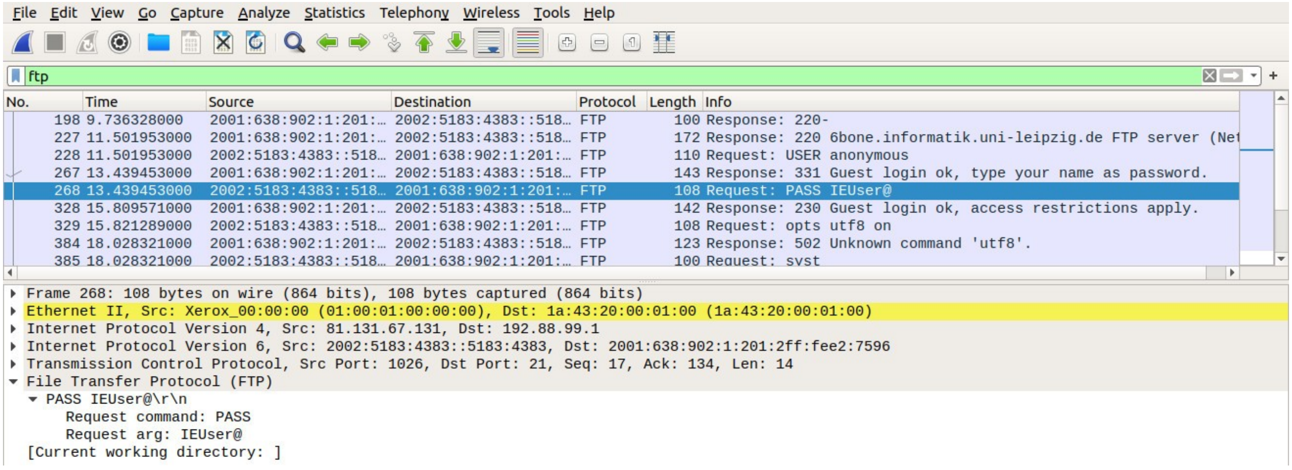
Analysis

From Fig D.a and Fig D.b we can see that the USERNAME and PASSWORD can easily be found.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ftp
No.   Time          Source                   Destination              Protocol Length Info
  198 9.736328000   2001:638:902:1:201:…     2002:5183:4383::518…     FTP        100 Response: 220-
  227 11.501953000  2001:638:902:1:201:…     2002:5183:4383::518…     FTP        172 Response: 220 6bone.informatik.uni-leipzig.de FTP server (Ne
  228 11.501953000  2002:5183:4383::518…     2001:638:902:1:201:…     FTP        110 Request: USER anonymous
  267 13.439453000  2001:638:902:1:201:…     2002:5183:4383::518…     FTP        143 Response: 331 Guest login ok, type your name as password.
  268 13.439453000  2002:5183:4383::518…     2001:638:902:1:201:…     FTP        108 Request: PASS IEUser@
  328 15.809571000  2001:638:902:1:201:…     2002:5183:4383::518…     FTP        142 Response: 230 Guest login ok, access restrictions apply.
  329 15.821289000  2002:5183:4383::518…     2001:638:902:1:201:…     FTP        108 Request: opts utf8 on
  384 18.028321000  2001:638:902:1:201:…     2002:5183:4383::518…     FTP        123 Response: 502 Unknown command 'utf8'.
  385 18.028321000  2002:5183:4383::518…     2001:638:902:1:201:…     FTP        100 Request: syst

▸ Frame 228: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
▸ Ethernet II, Src: Xerox_00:00:00 (01:00:01:00:00:00), Dst: 1a:43:20:00:01:00 (1a:43:20:00:01:00)
▸ Internet Protocol Version 4, Src: 81.131.67.131, Dst: 192.88.99.1
▸ Internet Protocol Version 6, Src: 2002:5183:4383::5183:4383, Dst: 2001:638:902:1:201:2ff:fee2:7596
▸ Transmission Control Protocol, Src Port: 1026, Dst Port: 21, Seq: 1, Ack: 85, Len: 16
▾ File Transfer Protocol (FTP)
  ▾ USER anonymous\r\n
      Request command: USER
      Request arg: anonymous
  [Current working directory: ]
```

Figure D.a: Username in FTP website


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ftp
No.   Time          Source                   Destination              Protocol Length Info
  198 9.736328000   2001:638:902:1:201:…     2002:5183:4383::518…     FTP        100 Response: 220-
  227 11.501953000  2001:638:902:1:201:…     2002:5183:4383::518…     FTP        172 Response: 220 6bone.informatik.uni-leipzig.de FTP server (Ne
  228 11.501953000  2002:5183:4383::518…     2001:638:902:1:201:…     FTP        110 Request: USER anonymous
  267 13.439453000  2001:638:902:1:201:…     2002:5183:4383::518…     FTP        143 Response: 331 Guest login ok, type your name as password.
  268 13.439453000  2002:5183:4383::518…     2001:638:902:1:201:…     FTP        108 Request: PASS IEUser@
  328 15.809571000  2001:638:902:1:201:…     2002:5183:4383::518…     FTP        142 Response: 230 Guest login ok, access restrictions apply.
  329 15.821289000  2002:5183:4383::518…     2001:638:902:1:201:…     FTP        108 Request: opts utf8 on
  384 18.028321000  2001:638:902:1:201:…     2002:5183:4383::518…     FTP        123 Response: 502 Unknown command 'utf8'.
  385 18.028321000  2002:5183:4383::518…     2001:638:902:1:201:…     FTP        100 Request: syst

▸ Frame 268: 108 bytes on wire (864 bits), 108 bytes captured (864 bits)
▸ Ethernet II, Src: Xerox_00:00:00 (01:00:01:00:00:00), Dst: 1a:43:20:00:01:00 (1a:43:20:00:01:00)
▸ Internet Protocol Version 4, Src: 81.131.67.131, Dst: 192.88.99.1
▸ Internet Protocol Version 6, Src: 2002:5183:4383::5183:4383, Dst: 2001:638:902:1:201:2ff:fee2:7596
▸ Transmission Control Protocol, Src Port: 1026, Dst Port: 21, Seq: 17, Ack: 134, Len: 14
▾ File Transfer Protocol (FTP)
  ▾ PASS IEUser@\r\n
      Request command: PASS
      Request arg: IEUser@
  [Current working directory: ]
```

Figure D.b: Password in FTP website

2.2.e SSH

SSH or **Secure Shell** is a cryptographic network protocol for operating network services securely over an unsecured network.SSH provides a secure channel over an unsecured network by using a client–server architecture, connecting an SSH client application with an SSH server.

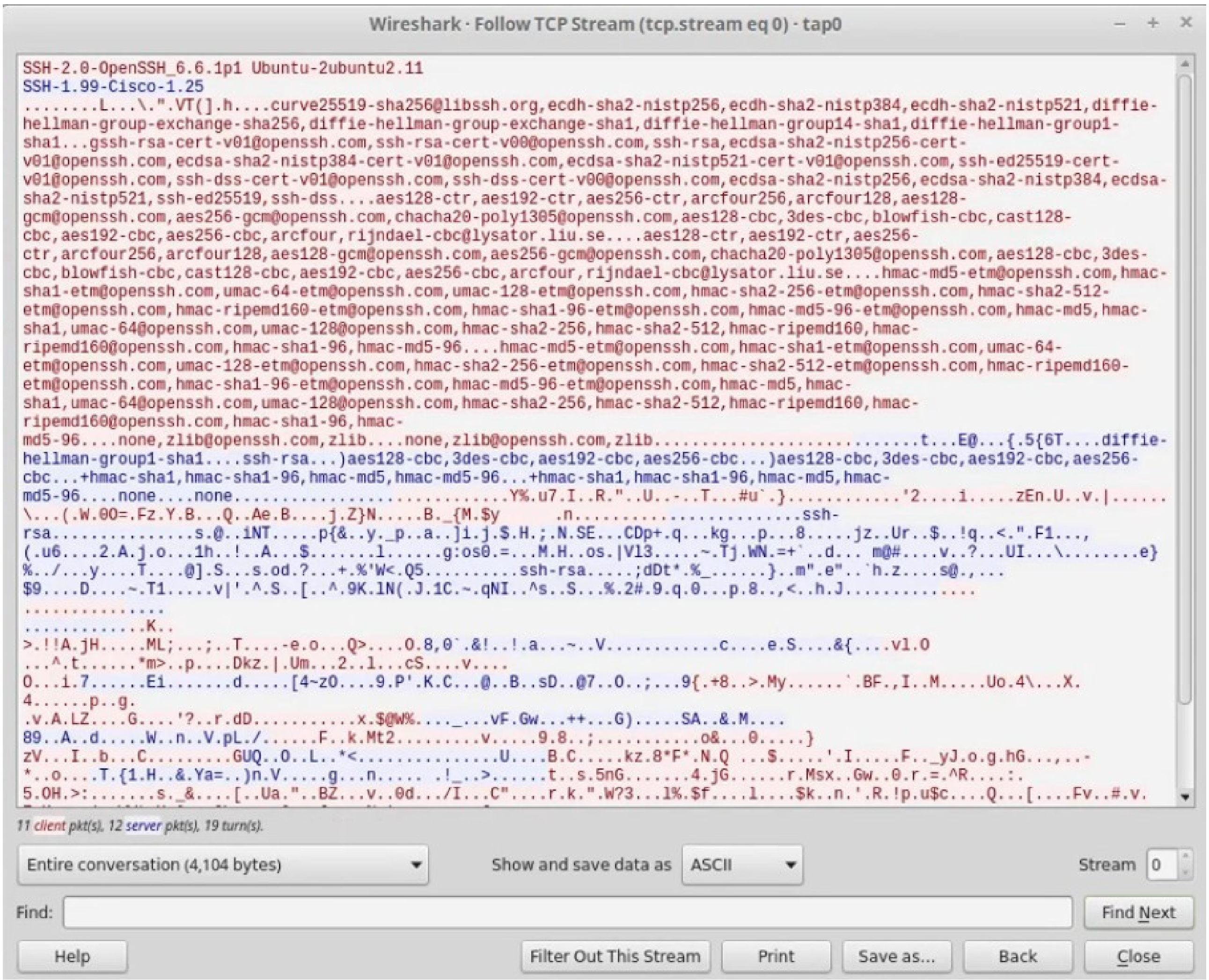
Procedure

We will perform the exact same operations as we did in case of Telnet, i.e., log into localhost itself. The command to do so is ssh username@localhost, where username is the username of an existent user on a computer. The password is the one used to log into the computer.

By right clicking on the packet(any of the SSH line) then selecting Follow and from the drop down menu selecting the TCP stream we can see the login ID and Password are encrypted.

Analysis

Fig E shows the output of one of the packets transferred via a protected SSH tunnel. The Ssh is encrypted the credentials can not be found.This makes it very difficult to get the exchage in SSH.



Wireshark · Follow TCP Stream (tcp.stream eq 0) · tap0

```
SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.11
SSH-1.99-Cisco-1.25
........L...\.".VT(].h....curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-
hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-
sha1...gssh-rsa-cert-v01@openssh.com,ssh-rsa-cert-v00@openssh.com,ssh-rsa,ecdsa-sha2-nistp256-cert-
v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-nistp521-cert-v01@openssh.com,ssh-ed25519-cert-
v01@openssh.com,ssh-dss-cert-v01@openssh.com,ssh-dss-cert-v00@openssh.com,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-
sha2-nistp521,ssh-ed25519,ssh-dss....aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-
gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com,aes128-cbc,3des-cbc,blowfish-cbc,cast128-
cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se....aes128-ctr,aes192-ctr,aes256-
ctr,arcfour256,arcfour128,aes128-gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com,aes128-cbc,3des-
cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se....hmac-md5-etm@openssh.com,hmac-
sha1-etm@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-ripemd160-etm@openssh.com,hmac-sha1-96-etm@openssh.com,hmac-md5-96-etm@openssh.com,hmac-md5,hmac-
sha1,umac-64@openssh.com,umac-128@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-ripemd160,hmac-
ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96....hmac-md5-etm@openssh.com,hmac-sha1-etm@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-ripemd160-
etm@openssh.com,hmac-sha1-96-etm@openssh.com,hmac-md5-96-etm@openssh.com,hmac-md5,hmac-
sha1,umac-64@openssh.com,umac-128@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-ripemd160,hmac-
ripemd160@openssh.com,hmac-sha1-96,hmac-
md5-96....none,zlib@openssh.com,zlib....none,zlib@openssh.com,zlib.............................t...E@...{.5{6T....diffie-
hellman-group1-sha1....ssh-rsa...)aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc...)aes128-cbc,3des-cbc,aes192-cbc,aes256-
cbc...+hmac-sha1,hmac-sha1-96,hmac-md5,hmac-md5-96...+hmac-sha1,hmac-sha1-96,hmac-md5,hmac-
md5-96....none....none.............................Y%.u7.I..R."..U..-..T...#u`.}...........'2....i.....zEn.U..v.|......
\...(.W.0O=.Fz.Y.B...Q..Ae.B....j.Z}N.....B._{M.$y      .n.........................ssh-
rsa...............s.@..iNT.....p{&..y._p..a..]i.j.$.H.;.N.SE...CDp+.q...kg...p...8.....jz..Ur..$..!q..<.".F1...,
(.u6....2.A.j.o...1h..!..A...$.......l......g:os0.=...M.H..os.|V13.....~.Tj.WN.=+`..d... m@#....v..?...UI...\........e}
%../...y....T....@].S...s.od.?...+.%'W<.Q5..........ssh-rsa.....;dDt*.%_......}..m".e"..`h.z....s@.,...
$9....D....~.T1.....v|'.^.S..[..^.9K.lN(.J.1C.~.qNI..^s..S...%.2#.9.q.0...p.8..,<..h.J.............
...............
.............K..
>.!!A.jH.....ML;...;..T....-e.o...Q>....0.8,0`.&!..!.a...~..V...........c....e.S....&{....v1.0
...^.t......*m>..p....Dkz.|.Um...2..l...cS....v....
O...i.7......Ei.......d.....[4~zO....9.P'.K.C...@..B..sD..@7..O..;...9{.+8..>.My......`.BF.,I..M.....Uo.4\...X.
4......p..g.
.v.A.LZ....G....'?..r.dD...........x.$@W%...._...vF.Gw...++...G).....SA..&.M....
89..A..d.....W..n..V.pL./......F..k.Mt2.........v.....9.8..;...........o&...0.....}
zV...I..b...C.........GUQ..O..L..*<...............U....B.C.....kz.8*F*.N.Q ...$.....'.I.....F.._yJ.o.g.hG...,..-
*..o....T.{1.H..&.Ya=..)n.V.....g...n..... .!_..>......t..s.5nG.......4.jG......r.Msx..Gw..0.r.=.^R....:.
5.OH.>:.......s._&....[..Ua."..BZ...v..0d.../I...C"....r.k.".W?3...1%.$f....l....$k..n.'.R.!p.u$c....Q...[....Fv..#.v.
```

11 client pkt(s), 12 server pkt(s), 19 turn(s).

Entire conversation (4,104 bytes)

Show and save data as ASCII

Stream 0

Find:

Figure E: Encrypted data of SSH exchange

3: **Problem 2**

3.1 Problem Statement

An organization wants to keep track of that its employee's access what all websites in their day-to-day life. Your task is to identify the top 3 websites being visited by the employees collectively as a whole. This can be simulated using a packet sniffer tool such as Wireshark. Try to capture packet in your system for about 4-5 hours each and collectively based on the IPs and MAC addresses of the systems, analyse the top 3 regularly visited websites.

3.2 Solution

The first objective was to capture our network traffic for about 4-5 hours and save the output in a csv file.Then we create a python script to get top visited websites along with the frequency.The final result is shown in terminal. IP 182.79.130.13 has the most visits standing in at 15825; with 2190 visits to 182.79.148.16 and finally, 172.217.163.174 has 955 visits. The first 2 websites are of Bharti Airtel and third is one of google.Since 1st and 2nd are same with diff. IPs 4th one at 734 i.e channel i is also added.

Pre-requisite

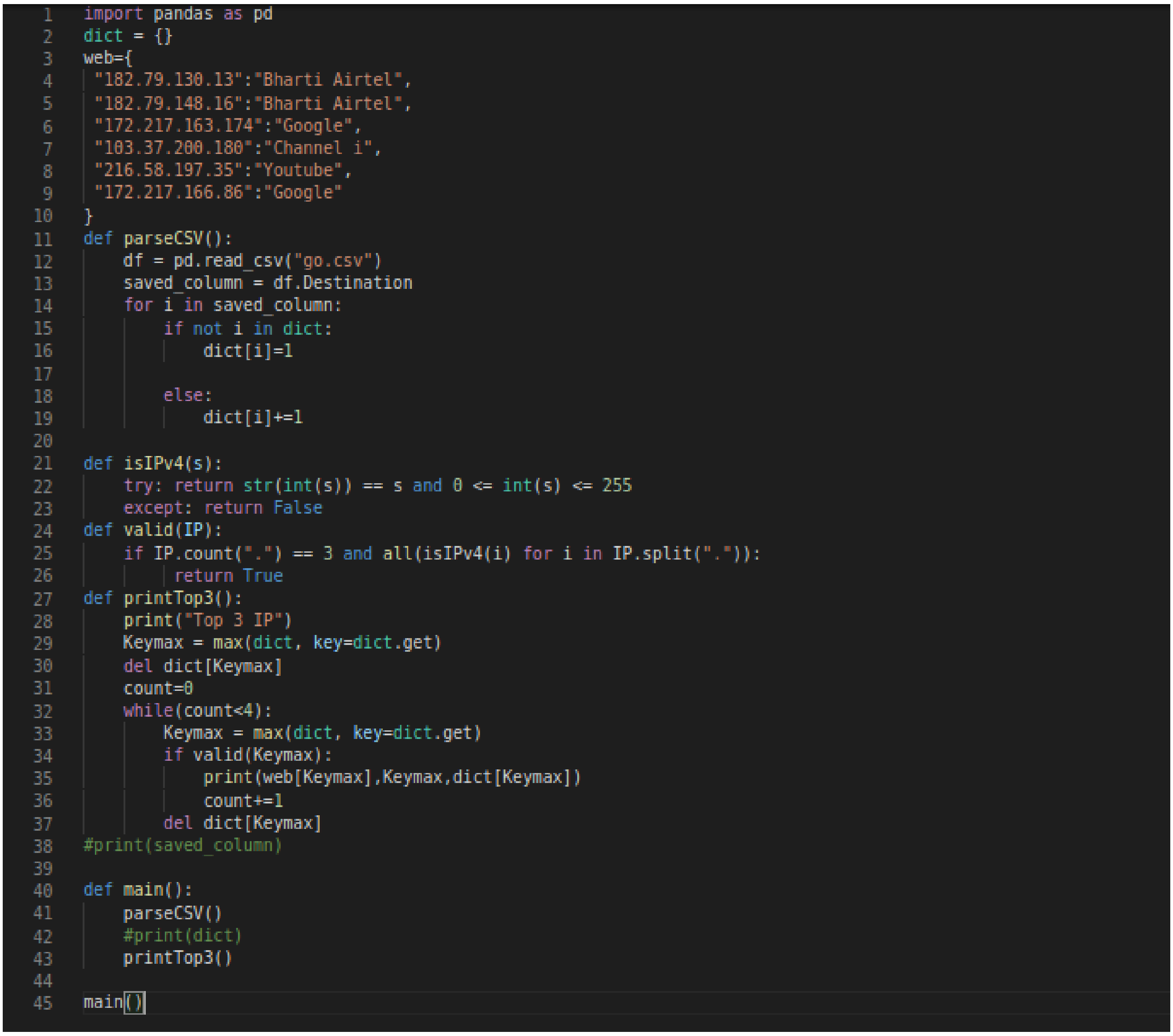
Pandas library should be installed to run the script.Also this code was built in Python 2.7.17 so it should run in later versions.

Lastly Wireshark software is neaded to analyze the network traffic.

The URL and IPs of available websites or websites that can we visited should also be updated in web dictionary (In Python Script)

Procedure

Start the wireshark to capture traffic for about 4-5 hours.Export file as CSV file .Run network.py as “python network.py” .The result can be seen in the terminal(NOTE:the csv file and python script must be in same folder and you must be in same directry in terminal)

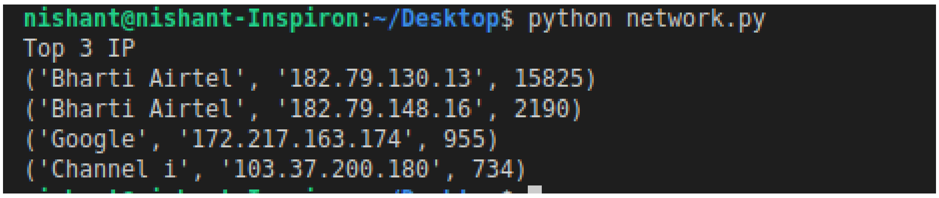


```python
import pandas as pd
dict = {}
web={
 "182.79.130.13":"Bharti Airtel",
 "182.79.148.16":"Bharti Airtel",
 "172.217.163.174":"Google",
 "103.37.200.180":"Channel i",
 "216.58.197.35":"Youtube",
 "172.217.166.86":"Google"
}
def parseCSV():
    df = pd.read_csv("go.csv")
    saved_column = df.Destination
    for i in saved_column:
        if not i in dict:
            dict[i]=1

        else:
            dict[i]+=1

def isIPv4(s):
    try: return str(int(s)) == s and 0 <= int(s) <= 255
    except: return False
def valid(IP):
    if IP.count(".") == 3 and all(isIPv4(i) for i in IP.split(".")):
        return True
def printTop3():
    print("Top 3 IP")
    Keymax = max(dict, key=dict.get)
    del dict[Keymax]
    count=0
    while(count<4):
        Keymax = max(dict, key=dict.get)
        if valid(Keymax):
            print(web[Keymax],Keymax,dict[Keymax])
            count+=1
        del dict[Keymax]
#print(saved_column)

def main():
    parseCSV()
    #print(dict)
    printTop3()

main()
```

Figure F: Snapshot of code

Explanation

We use a function named “valid(IP)” [Line 21-26] in Figure F so that only valid IPv4 comes in count.Also we filter out Destination IPs ans exclude host IP [Line 29-30](freq. of host IP will be maximum).The solution can be seen in terminal as shown in Figure G.Top 3 IPv4 with their frequencies and website names can be seen.



```
nishant@nishant-Inspiron:~/Desktop$ python network.py
Top 3 IP
('Bharti Airtel', '182.79.130.13', 15825)
('Bharti Airtel', '182.79.148.16', 2190)
('Google', '172.217.163.174', 955)
('Channel i', '103.37.200.180', 734)
```

Fig G: Terminal snapshot

Note: I have considered only IPv4 when finding destination IPs. IPv6 has been neglected.

4:Conclusion

Problem 1

We want our website to be ***secure*** for a number of reasons. Not only do you want to protect potentially sensitive information, but we’ll want to make sure that our visitors are comfortable browsing through our site. We need protocols which encrypt the exchanges to make sure that the information is not compromised.

To safeguard the details of an exchange,we need more platforms with secure protocols such as HTTPS and SSH in use.

Problem 2

The software Wireshark is easy to use and one can easily analyze network traffic by capturing the packets. In this problem we figure out the top 3 IPs visited by the user(s) of a network.

Reference

[1] https://www.youtube.com/watch?v=bEXEEfbNADs

[2] https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

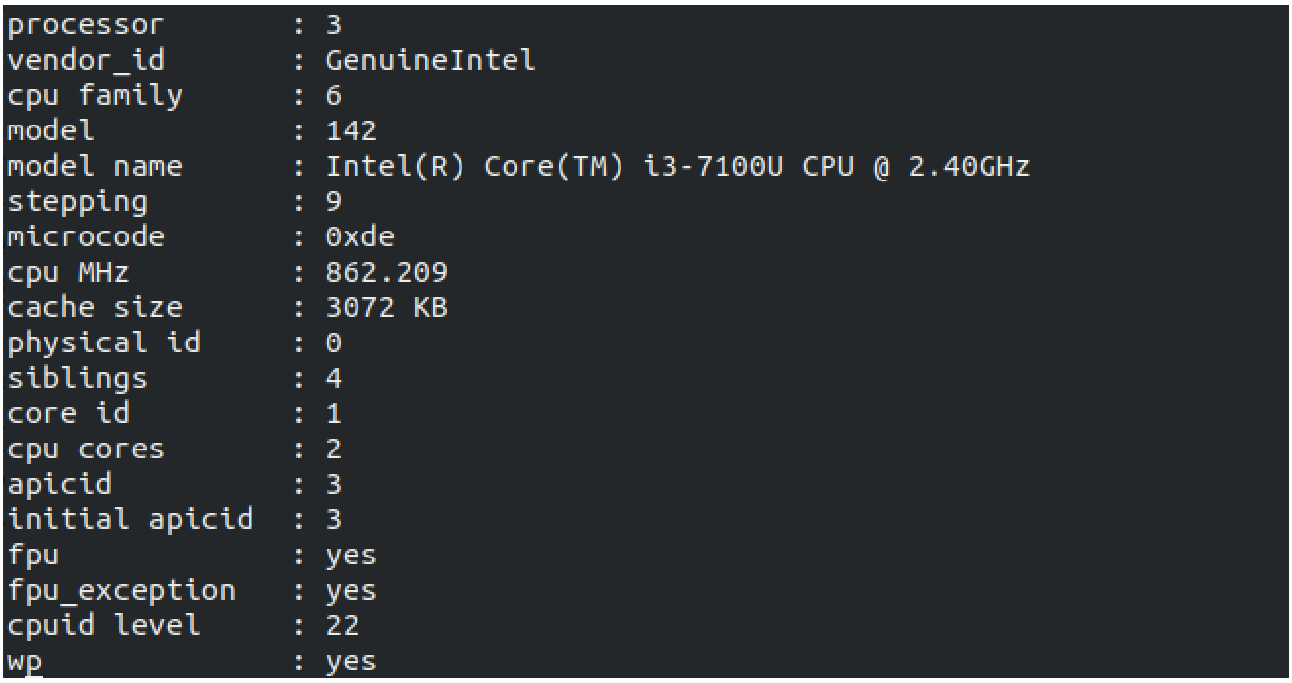
[3] https://en.wikipedia.org/wiki/HTTPS

[4] https://en.wikipedia.org/wiki/Telnet

[5] https://www.tutorialspoint.com/validate-ip-address-in-python

[6] https://wiki.wireshark.org/SampleCaptures

Table



```
processor       : 3
vendor_id       : GenuineIntel
cpu family      : 6
model           : 142
model name      : Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz
stepping        : 9
microcode       : 0xde
cpu MHz         : 862.209
cache size      : 3072 KB
physical id     : 0
siblings        : 4
core id         : 1
cpu cores       : 2
apicid          : 3
initial apicid  : 3
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
```

System Configuration