Network and System Security

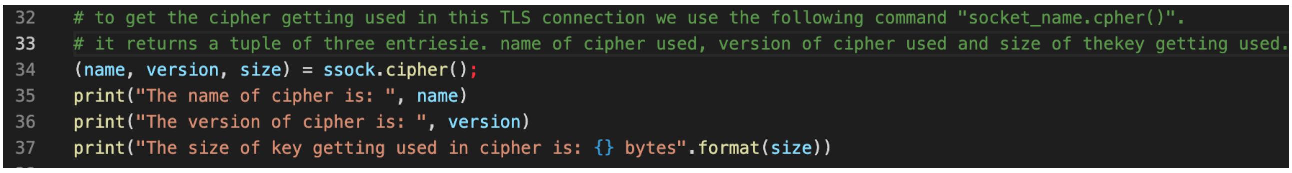
Assignment - 3 (Problem - 1)

Name: Suraj Kiran Mate
Entry No: 2021JCS2387

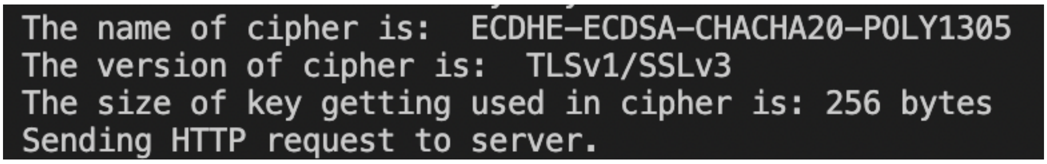
Problem-1: Understanding Transport Layer Security

Task-1: TLS Handshake

1.



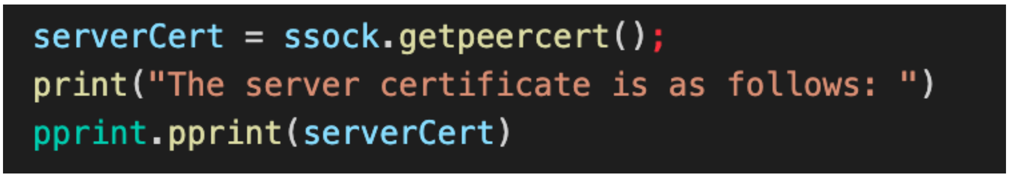
```
# to get the cipher getting used in this TLS connection we use the following command "socket_name.cpher()".
# it returns a tuple of three entriesie. name of cipher used, version of cipher used and size of thekey getting used.
(name, version, size) = ssock.cipher();
print("The name of cipher is: ", name)
print("The version of cipher is: ", version)
print("The size of key getting used in cipher is: {} bytes".format(size))
```



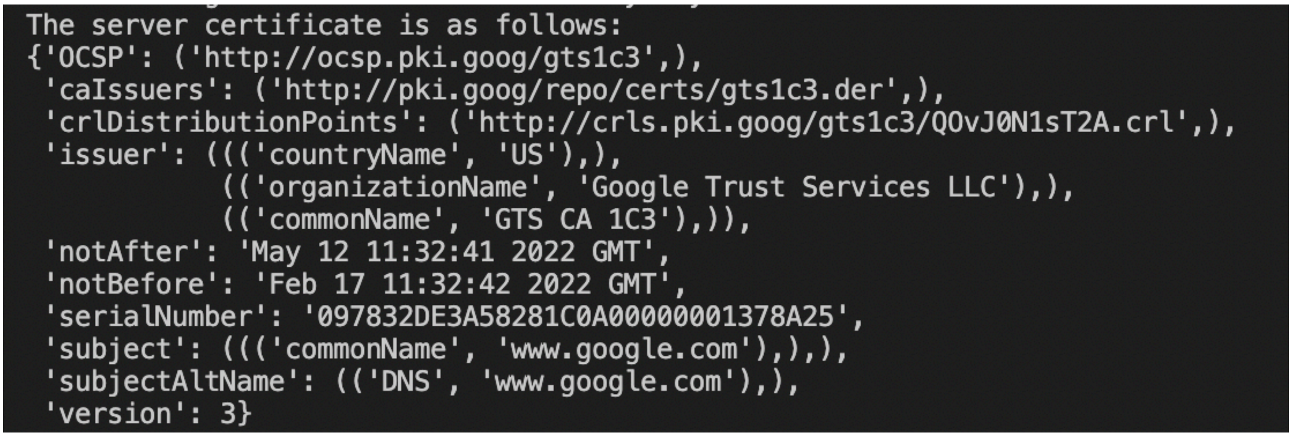
```
The name of cipher is:  ECDHE-ECDSA-CHACHA20-POLY1305
The version of cipher is:  TLSv1/SSLv3
The size of key getting used in cipher is: 256 bytes
Sending HTTP request to server.
```

From above code we have printed the cipher getting used in between client and server. Here we have formed connection with google server (www.google.com). Google was initially using **RC4** stream cipher but because of its vulnerability as we have seen in WEP (wired equivalent privacy), RC4 found to be weak. Then google started implementing **CHACHA20** as a stream cipher. Here **ECDSA** is the signature algorithm because of its fastness in generation as well as verification of authentication tag.
Still **TLS version 1** is getting used but with updated cipher suite. The size of key getting used is 256 bytes.

2.



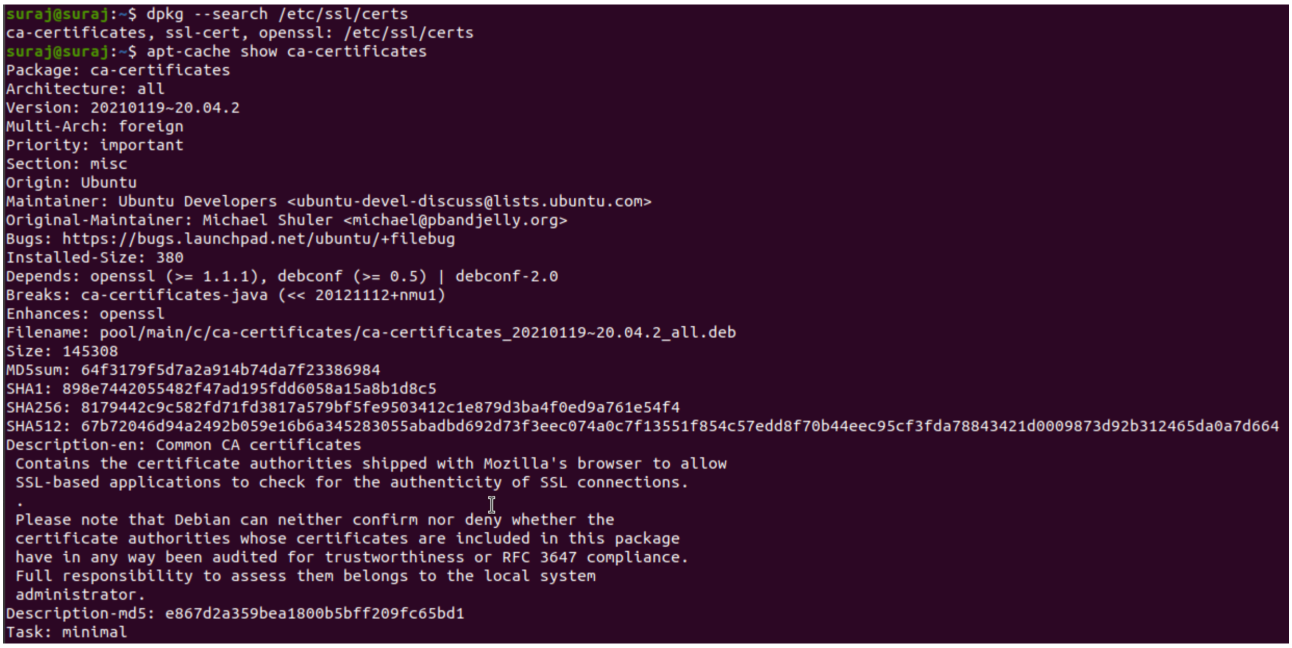
```
serverCert = ssock.getpeercert();
print("The server certificate is as follows: ")
pprint.pprint(serverCert)
```



```
The server certificate is as follows:
{'OCSP': ('http://ocsp.pki.goog/gts1c3',),
 'caIssuers': ('http://pki.goog/repo/certs/gts1c3.der',),
 'crlDistributionPoints': ('http://crls.pki.goog/gts1c3/QOvJ0N1sT2A.crl',),
 'issuer': ((('countryName', 'US'),),
            (('organizationName', 'Google Trust Services LLC'),),
            (('commonName', 'GTS CA 1C3'),)),
 'notAfter': 'May 12 11:32:41 2022 GMT',
 'notBefore': 'Feb 17 11:32:42 2022 GMT',
 'serialNumber': '097832DE3A58281C0A00000001378A25',
 'subject': ((('commonName', 'www.google.com'),),),
 'subjectAltName': (('DNS', 'www.google.com'),),
 'version': 3}
```

When we connected to google server we got the certificate like this. We get the server certificate by using **ssock.getpeercert()**. This certificate is showing various certificate parameters. Like the information about issuer, the duration of validity, serial Number of certificate, subject of certificate and the version of certificate.

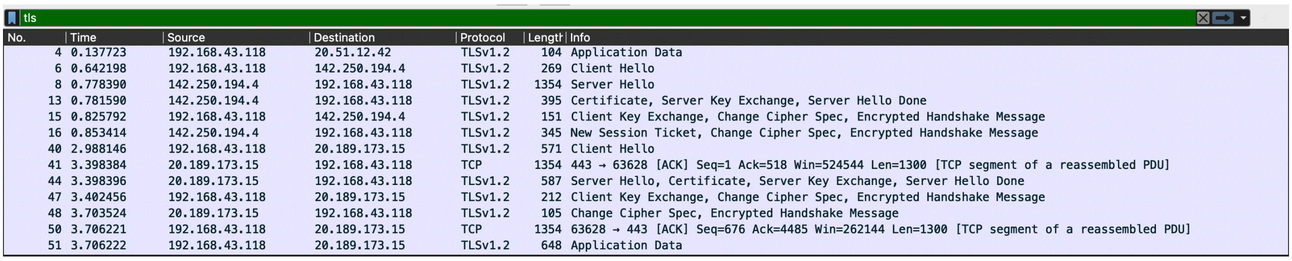
3. In Linux based systems **/etc/ssl/certs.** stores the root certificates along with file **ca-certificate.crt**. So the only purpose of this directory is to store the certificates. We can get more info about this directory on ubuntu by running the following commands.



```
suraj@suraj:~$ dpkg --search /etc/ssl/certs
ca-certificates, ssl-cert, openssl: /etc/ssl/certs
suraj@suraj:~$ apt-cache show ca-certificates
Package: ca-certificates
Architecture: all
Version: 20210119~20.04.2
Multi-Arch: foreign
Priority: important
Section: misc
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Michael Shuler <michael@pbandjelly.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 380
Depends: openssl (>= 1.1.1), debconf (>= 0.5) | debconf-2.0
Breaks: ca-certificates-java (<< 20121112+nmu1)
Enhances: openssl
Filename: pool/main/c/ca-certificates/ca-certificates_20210119~20.04.2_all.deb
Size: 145308
MD5sum: 64f3179f5d7a2a914b74da7f23386984
SHA1: 898e7442055482f47ad195fdd6058a15a8b1d8c5
SHA256: 8179442c9c582fd71fd3817a579bf5fe9503412c1e879d3ba4f0ed9a761e54f4
SHA512: 67b72046d94a2492b059e16b6a345283055abadbd692d73f3eec074a0c7f13551f854c57edd8f70b44eec95cf3fda78843421d0009873d92b312465da0a7d664
Description-en: Common CA certificates
 Contains the certificate authorities shipped with Mozilla's browser to allow
 SSL-based applications to check for the authenticity of SSL connections.
 .
 Please note that Debian can neither confirm nor deny whether the
 certificate authorities whose certificates are included in this package
 have in any way been audited for trustworthiness or RFC 3647 compliance.
 Full responsibility to assess them belongs to the local system
 administrator.
Description-md5: e867d2a359bea1800b5bff209fc65bd1
Task: minimal
```

Here complete info about ca certificates is given.

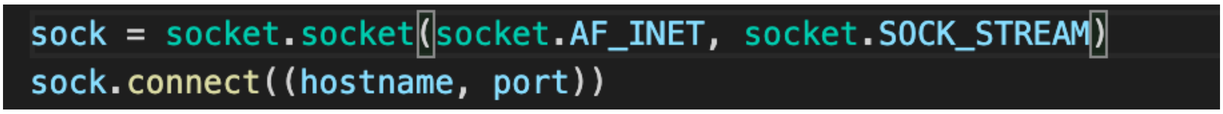
4.



tls

No.	Time	Source	Destination	Protocol	Length	Info
4	0.137723	192.168.43.118	20.51.12.42	TLSv1.2	104	Application Data
6	0.642198	192.168.43.118	142.250.194.4	TLSv1.2	269	Client Hello
8	0.778390	142.250.194.4	192.168.43.118	TLSv1.2	1354	Server Hello
13	0.781590	142.250.194.4	192.168.43.118	TLSv1.2	395	Certificate, Server Key Exchange, Server Hello Done
15	0.825792	192.168.43.118	142.250.194.4	TLSv1.2	151	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
16	0.853414	142.250.194.4	192.168.43.118	TLSv1.2	345	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
40	2.988146	192.168.43.118	20.189.173.15	TLSv1.2	571	Client Hello
41	3.398384	20.189.173.15	192.168.43.118	TCP	1354	443 → 63628 [ACK] Seq=1 Ack=518 Win=524544 Len=1300 [TCP segment of a reassembled PDU]
44	3.398396	20.189.173.15	192.168.43.118	TLSv1.2	587	Server Hello, Certificate, Server Key Exchange, Server Hello Done
47	3.402456	192.168.43.118	20.189.173.15	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
48	3.703524	20.189.173.15	192.168.43.118	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
50	3.706221	192.168.43.118	20.189.173.15	TCP	1354	63628 → 443 [ACK] Seq=676 Ack=4485 Win=262144 Len=1300 [TCP segment of a reassembled PDU]
51	3.706222	192.168.43.118	20.189.173.15	TLSv1.2	648	Application Data

When we run the script we captured some TLS packets as shown above. Here 192.168.43.118 is my public IP address acting as client and 142.250.194.42 is the google server IP address acting as server here. The message exchange in TLS handshake is exactly similar to what we have learned in lectures.



```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((hostname, port))
```

First TCP handshake is initiated and after this the TCP socket created is wrapped using TLS protocol.
The first command creates the TCP socket and the second command connect that socket to given hostname (eg: **www.google.com**) and the specified port number (upto 65535).

The wrapped TLS socket is referred here as **ssock.**

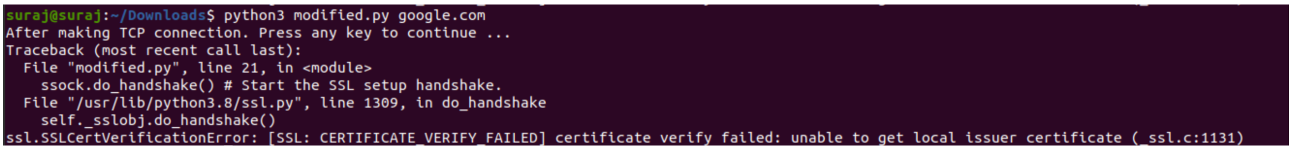
ssock.do_handshake()

This commands triggers the TLS handshake.
TCP follows the **3 way handshake** in which server and client gets synchronised on particular sequence number. Client sends the SYN(synchronised sequence number). Server responds the client with the ACK corresponding to the sequence number. Then client again send the ACK for receiver ACK from server.

But the TLS handshake is completely different. In TLS handshake server and client shares the cipher suite containing multiple cypher algorithms. The best and compatible algorithm is finalised for communication between them. Then client and server authenticate the identity of each other by sharing digital certificate on their public key. They can also generate session keys for symmetric encryption for message exchange.

Task-2: CA's Certificate

1 and 2.



```
suraj@suraj:~/Downloads$ python3 modified.py google.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "modified.py", line 21, in <module>
    ssock.do_handshake() # Start the SSL setup handshake.
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1131)
```

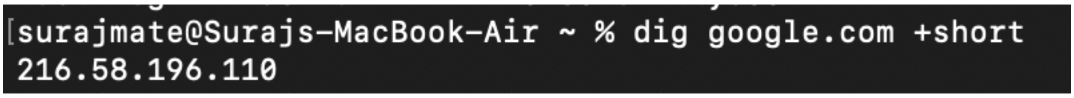
After changing the directory I am getting error like this. Here it is saying **CERTIFICATE_VERIFY_FAILED** because the certs folder is empty and the code is not able to find the required certificates to authenticate the server. Hence the certificate verification is getting failed and no further connection can be established.

3 and 4.

After Putting the required CA certificate into the certs folder the error is resolved.

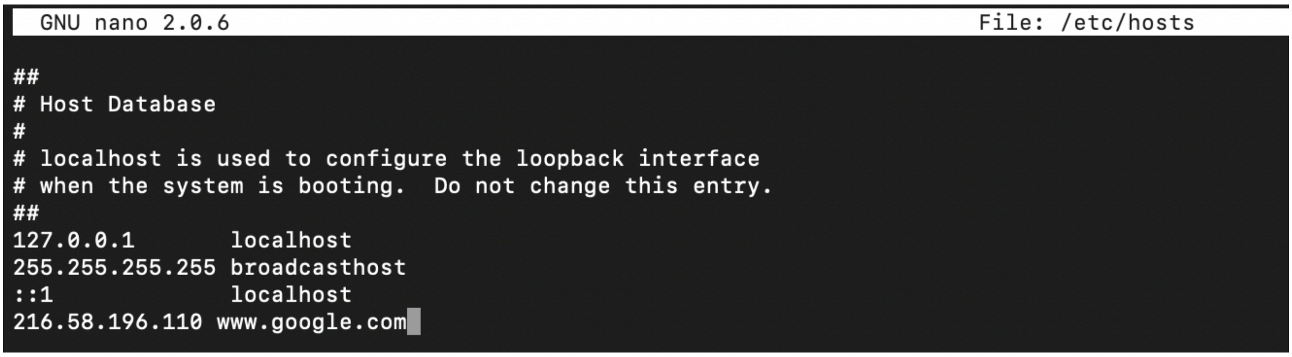
Task-3: Hostname

1. The IP address of google server is **216.58.196.110**.



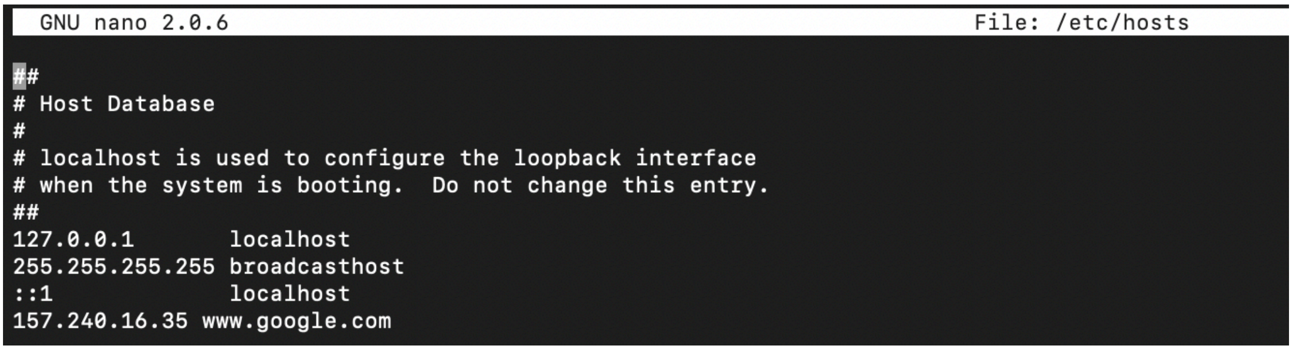
```
[surajmate@Surajs-MacBook-Air ~ % dig google.com +short
216.58.196.110
```

2.



```
  GNU nano 2.0.6                                              File: /etc/hosts

##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
216.58.196.110 www.google.com
```



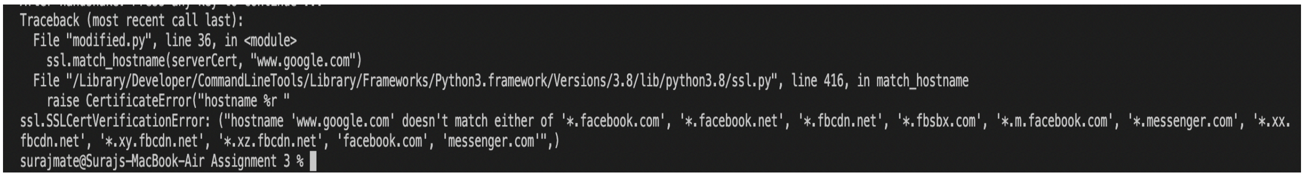
```
 GNU nano 2.0.6                                                   File: /etc/hosts

##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
157.240.16.35 www.google.com
```

This is the correct entry of IP address and its corresponding hostname. In this setting there will be no problem in the hostname verification since the hostname and the corresponding IP address are genuine. The connection is getting formed with the authenticated server.

But if we make any modifications in the IP address for the same hostname the it can be vulnerable to different attack.

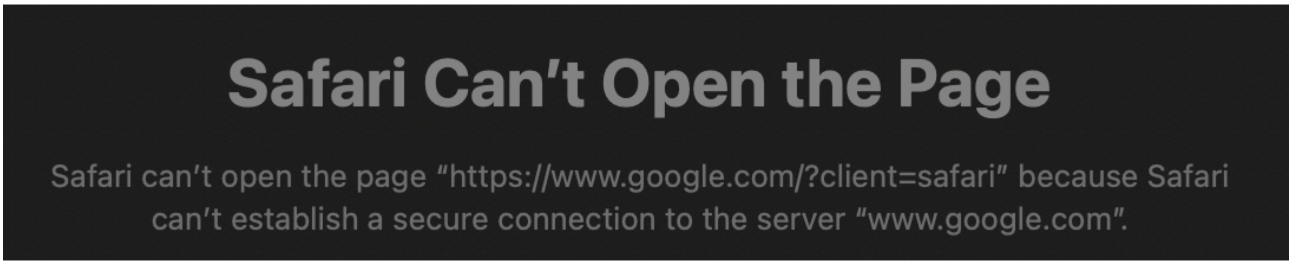
We found the IP address for www.facebook.com as above. Now what we are doing is we are simply replacing the original IP address of google.com in the **/etc/hosts** file with the IP address of facebook.com. Hence the entry modified look like below:
Here the last entry is **157.240.16.35 www.google.com**.
In this setting when we run the client side program with **check_hostname = False** then it will successfully form a connection with the IP address 157.240.16.35 means the client will successfully connect to Facebook.com.
But when we keep the **check_hostname = True** then it will not form connection and given an error like below:



```
Traceback (most recent call last):
  File "modified.py", line 36, in <module>
    ssl.match_hostname(serverCert, "www.google.com")
  File "/Library/Developer/CommandLineTools/Library/Frameworks/Python3.framework/Versions/3.8/lib/python3.8/ssl.py", line 416, in match_hostname
    raise CertificateError("hostname %r "
ssl.SSLCertVerificationError: ("hostname 'www.google.com' doesn't match either of '*.facebook.com', '*.facebook.net', '*.fbcdn.net', '*.fbsbx.com', '*.m.facebook.com', '*.messenger.com', '*.xx.
fbcdn.net', '*.xy.fbcdn.net', '*.xz.fbcdn.net', 'facebook.com', 'messenger.com'",)
surajmate@Surajs-MacBook-Air Assignment 3 %
```

In above error we can clearly see that hostname check failed because the IP address was of www.facebook.com and we saved it by the hostname of www.google.com. Hence the hostname was not matching. Thus the error is generated in this case.

Thus during every connection if the hostname is not verified from the certificate, then it can lead to connecting the client to non-authenticated server. Attacker can take advantage of this in the similar way as we have seen in lectures where the IP address and domain name mapping in the local DNS server is changed.



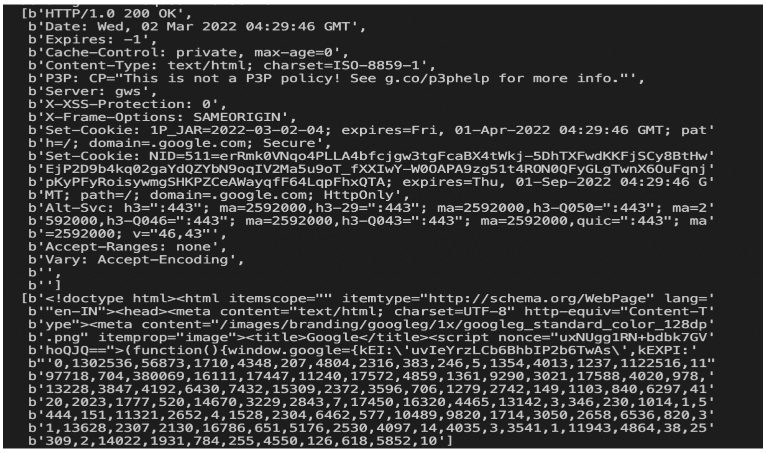
Safari Can’t Open the Page

Safari can’t open the page “https://www.google.com/?client=safari” because Safari can’t establish a secure connection to the server “www.google.com”.

One additional point, when I keep this incorrect mapping in **/etc/host**. I am not able to open www.google.com on my local web browser. This may be due to the hostname verification might be on by default in Safari browser. Thus for incorrect mapping it is not able to form the connection with intended server. Hence safari browser is returning the warning of not being able to form the secure connection with www.google.com.

Task-4: Communicating Data

1.



```
[b'HTTP/1.0 200 OK',
 b'Date: Wed, 02 Mar 2022 04:29:46 GMT',
 b'Expires: -1',
 b'Cache-Control: private, max-age=0',
 b'Content-Type: text/html; charset=ISO-8859-1',
 b'P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."',
 b'Server: gws',
 b'X-XSS-Protection: 0',
 b'X-Frame-Options: SAMEORIGIN',
 b'Set-Cookie: 1P_JAR=2022-03-02-04; expires=Fri, 01-Apr-2022 04:29:46 GMT; pat'
 b'h=/; domain=.google.com; Secure',
 b'Set-Cookie: NID=511=erRmk0VNqo4PLLA4bfcjgw3tgFcaBX4tWkj-5DhTXFwdKKFjSCy8BtHw'
 b'EjP2D9b4kq02gaYdQZYbN9oqIV2Ma5u9oT_fXXIwY-W0OAPA9zg51t4RON0QFyGLgTwnX6OuFqnj'
 b'pKyPFyRoisywmgSHKPZCeAWayqfF64LqpFhxQTA; expires=Thu, 01-Sep-2022 04:29:46 G'
 b'MT; path=/; domain=.google.com; HttpOnly',
 b'Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000,h3-Q050=":443"; ma=2'
 b'592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma'
 b'=2592000; v="46,43"',
 b'Accept-Ranges: none',
 b'Vary: Accept-Encoding',
 b'',
 b'']
[b'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang='
 b'"en-IN"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-T'
 b'ype"><meta content="/images/branding/googleg/1x/googleg_standard_color_128dp'
 b'.png" itemprop="image"><title>Google</title><script nonce="uxNUgg1RN+bdbk7GV'
 b'hoQJQ==">(function(){window.google={kEI:\'uvIeYrzLCb6BhbIP2b6TwAs\',kEXPI:'
 b"'0,1302536,56873,1710,4348,207,4804,2316,383,246,5,1354,4013,1237,1122516,11"
 b'97718,704,380069,16111,17447,11240,17572,4859,1361,9290,3021,17588,4020,978,'
 b'13228,3847,4192,6430,7432,15309,2372,3596,706,1279,2742,149,1103,840,6297,41'
 b'20,2023,1777,520,14670,3229,2843,7,17450,16320,4465,13142,3,346,230,1014,1,5'
 b'444,151,11321,2652,4,1528,2304,6462,577,10489,9820,1714,3050,2658,6536,820,3'
 b'1,13628,2307,2130,16786,651,5176,2530,4097,14,4035,3,3541,1,11943,4864,38,25'
 b'309,2,14022,1931,784,255,4550,126,618,5852,10']
```

When we added the data send and receive http request to client code. We get some output as shown above.

'HTTP/1.0 200 OK' it indicates that the http get request is successfully executed. Then the time of communication is printed. Other options like content type, server domain name and other cookie related information is given.

2. We can fetch the image using HTTP request just by changing the URL in hostname. We can use the image url to fetch that image and we will the bytes of required image in the terminal.

Thank you !