

哈尔滨工业大学(深圳)

《网络与系统安全》

实验报告

实验五

TLS 实验

学 院: 计算机科学与技术学院

姓 名: 覃煜淮

学 号: 220110803

专 业: 计算机类

日 期: 2025 年 4 月

一、实验过程

1.在客户端容器中执行如下命令 `./handshake.py www.baidu.com` 根据执行结果回答下面三个问题。

(1) 客户端和服务端使用的加密算法有哪些，分别起什么作用？

Cipher used: ('ECDHE-RSA-AES128-GCM-SHA256', 'TLSv1.2', 128)

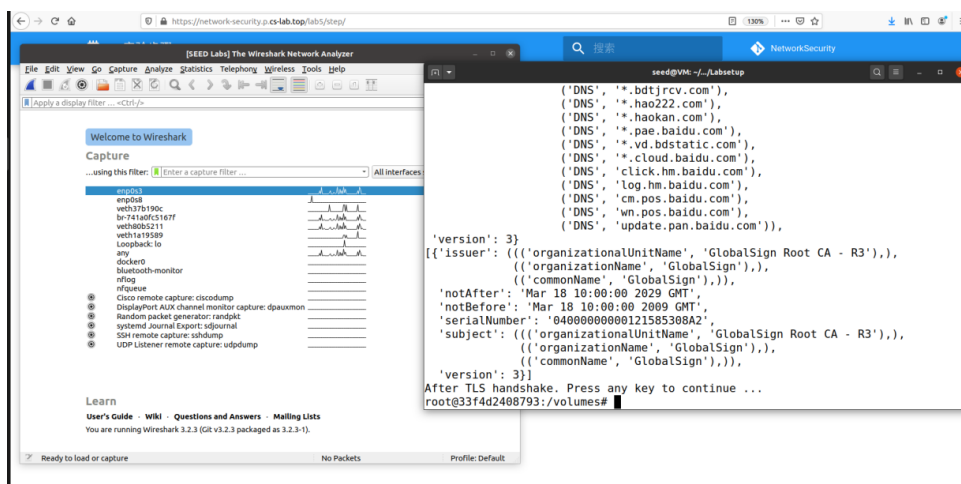
客户端

AES（高级加密标准），具有加密速度快、安全级别高的特点，适用于对大量数据进行加密传输。RSA 算法，基于大整数因式分解的数学难题，安全性较高，广泛应用于密钥交换和数字签名等场景；还有 ECDHE（椭圆曲线 Diffie-Hellman 密钥交换）算法，与 RSA 相比，在提供相同安全性的同时，可以使用更短的密钥，具有更高的安全性和效率。SHA-256（安全哈希算法），具有不可逆性和雪崩效应等特点，能够确保数据的唯一性和完整性

服务器端

AES 算法，其高效的加密性能使其成为服务器端处理大量数据加密的首选。RSA 算法、ECC 算法，它们在保障服务器身份认证和密钥交换的安全性方面发挥着重要作用。SHA-256，其不可逆性使得即使存储的哈希值泄露，攻击者也难以还原出原始数据。

(2) 分析打印出来的服务器端证书



分析：服务器端主要包含以下信息

版本号 (Version) 值为 3, 这表明该证书遵循的是 X.509 标准的第三版。X.509 v3 是目前广泛使用的一种数字证书格式, 它支持扩展功能, 比如可以添加各种扩展项来增强证书的功能和灵活性, 相比之前的版本 (如 v1 和 v2), v3 能够更好地满足现代网络安全的需求。

颁发者 (Issuer) 包含以下信息: organizationUnitName: GlobalSign Root CA - R3 organizationName: GlobalSign 这表示该证书是由 GlobalSign 这家证书颁发机构 (CA) 颁发的, 且具体是由其名为“GlobalSign Root CA - R3”的根证书颁发的。GlobalSign 是一家知名的 CA 机构, 提供各种数字证书服务, 用于保障网络安全通信。

有效期 (Validity) notAfter: Mar 18 10:00:00 2029 GMT notBefore: Mar 18 10:00:00 2009 GMT 这给出了证书的有效时间范围。该证书从 2009 年 3 月 18 日 10 点开始生效, 到 2029 年 3 月 18 日 10 点失效。在这段时间内, 客户端可以信任该证书, 用于验证服务器的身份以及进行安全通信。一旦超过这个时间范围, 证书就不再有效, 客户端通常会拒绝与使用该过期证书的服务器进行安全连接, 除非有特殊的配置或豁免。

序列号 (Serial Number) 040000000012158530BA2 序列号是 CA 为每个颁发的证书分配的唯一标识符。它用于在 CA 的数据库中唯一地标识这个证书, 方便进行证书的管理、吊销等操作。当需要查询证书的状态 (比如是否被吊销) 时, 序列号是一个重要的参考信息。

主题 (Subject) 包含以下信息: organizationUnitName: GlobalSign Root CA - R3 organizationName: GlobalSign 这表示该证书的持有者 (主题) 是 GlobalSign 的“GlobalSign Root CA - R3”。这通常意味着该证书是一个根证书, 用于签署其他证书 (如中间证书或服务器证书)。根证书在证书链中处于最高层级, 客户端通常会预先安装受信任的根证书, 当验证服务器证书时, 会追溯到根证书来建立信任关系。

(3) 抓包分析 TLS 握手协议

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------------------------|---------------|-------------|----------|--------|--|
| 1 | 2022-08-12 03:12:10.0.2.15 | 34.122.121.32 | 10.0.2.15 | TCP | 74 | 40558 -> 80 [SYN] Seq=2529053097 Win=0 Len=0 MSS=1460 SACK... |
| 2 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 3 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 4 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 5 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 6 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 7 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 8 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 9 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 10 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 11 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 12 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 13 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 14 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 15 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 16 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 17 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 80 -> 36744 [SYN, ACK] Seq=3291853698 Win=0 Len=0 MSS=1460 SACK... |
| 18 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TLSv1.2 | 270 | Client Hello |
| 19 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 20 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TLSv1.2 | 2974 | Server Hello |
| 21 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 22 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 23 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 24 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 25 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 26 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 27 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 28 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 29 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 30 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 31 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 32 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TLSv1.2 | 85 | Encrypted Alert |
| 33 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 34 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 35 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |
| 36 | 2022-08-12 03:12:10.0.2.15 | 35.224.170.84 | 10.0.2.15 | TCP | 60 | 443 -> 55464 [ACK] Seq=32256002 Ack=2247210072 Win=65535 Len=0 |

分析: TLS 握手是在建立了 TCP 连接之上上了。从上图中可以看到, 在程序运行之后先建立了 TCP 连接, 从 15-17 这三个包是用于建立 TCP 连接的, 而这时候还没有 TLS 连接。

按一下任意键之后开始建立 TLS 连接。从第 18 个包开始建立 TLS 连接, 到第 32 个包结束。

综上所述, 通过执行以上操作, 我们完成了 TLS handshake 任务

2.更改证书文件路径, 请同学们将 www.baidu.com 网站的测试过程截图保存 (如果不将证书拷贝过来应该有报错信息, 拷贝过来之后应该正常), 也可选用

其他网站做测试。

由之前实验任务的图可知：

```

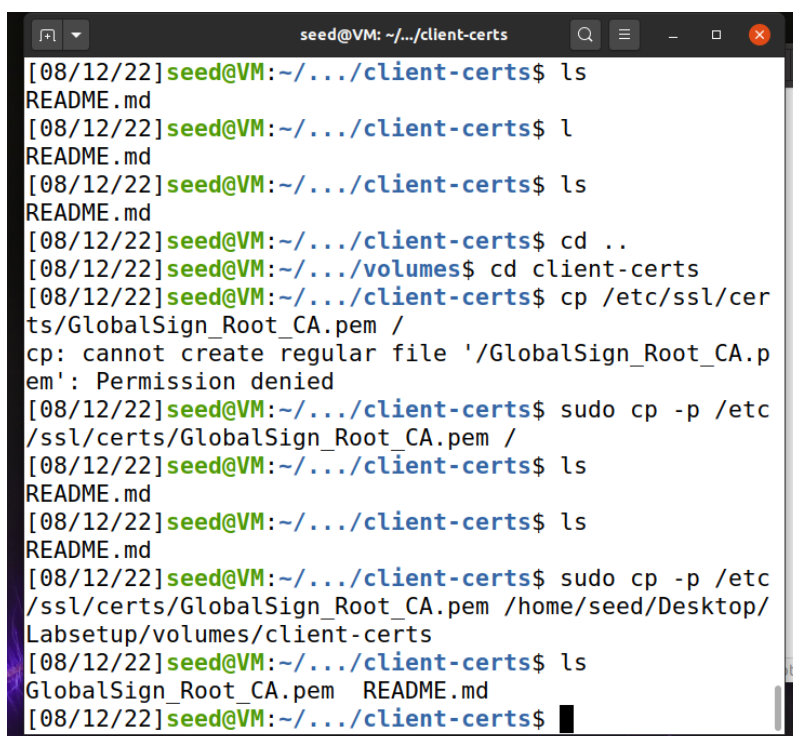
('DNS', '*.gz.baidubce.com'),
('DNS', '*.smartapps.cn'),
('DNS', '*.bdtjrcv.com'),
('DNS', '*.hao222.com'),
('DNS', '*.haokan.com'),
('DNS', '*.pae.baidu.com'),
('DNS', '*.vd.bdstatic.com'),
('DNS', '*.cloud.baidu.com'),
('DNS', 'click.hm.baidu.com'),
('DNS', 'log.hm.baidu.com'),
('DNS', 'cm.pos.baidu.com'),
('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 ...
[08/12/22] seed@VM: ~/.../volumes$ sudo vi handshake.py
[08/12/22] seed@VM: ~/.../volumes$

```

CSDN @XLYcm

CA 证书是：GlobalSign_Root_CA.pem

于是把 GlobalSign_Root_CA.pem 文件拷贝到 clien-certs 目录下：



```

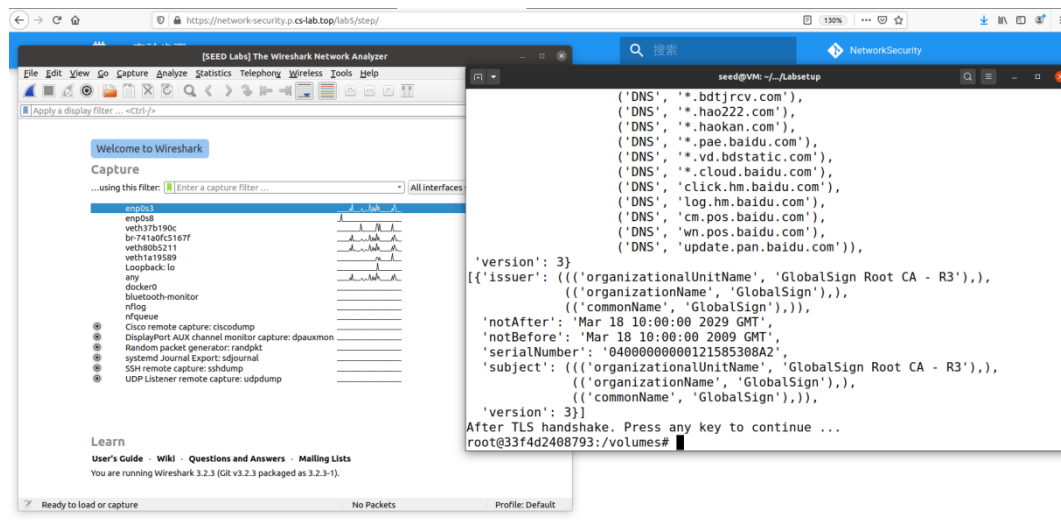
seed@VM: ~/.../client-certs
[08/12/22] seed@VM: ~/.../client-certs$ ls
README.md
[08/12/22] seed@VM: ~/.../client-certs$ l
README.md
[08/12/22] seed@VM: ~/.../client-certs$ ls
README.md
[08/12/22] seed@VM: ~/.../client-certs$ cd ..
[08/12/22] seed@VM: ~/.../volumes$ cd client-certs
[08/12/22] seed@VM: ~/.../client-certs$ cp /etc/ssl/cer
ts/GlobalSign_Root_CA.pem /
cp: cannot create regular file '/GlobalSign_Root_CA.p
em': Permission denied
[08/12/22] seed@VM: ~/.../client-certs$ sudo cp -p /etc
/ssl/certs/GlobalSign_Root_CA.pem /
[08/12/22] seed@VM: ~/.../client-certs$ ls
README.md
[08/12/22] seed@VM: ~/.../client-certs$ ls
README.md
[08/12/22] seed@VM: ~/.../client-certs$ sudo cp -p /etc
/ssl/certs/GlobalSign_Root_CA.pem /home/seed/Desktop/
Labsetup/volumes/client-certs
[08/12/22] seed@VM: ~/.../client-certs$ ls
GlobalSign_Root_CA.pem README.md
[08/12/22] seed@VM: ~/.../client-certs$

```

后使用如下命令获取一个 hash 值，并创建一个名为该 hash 值的软链接。

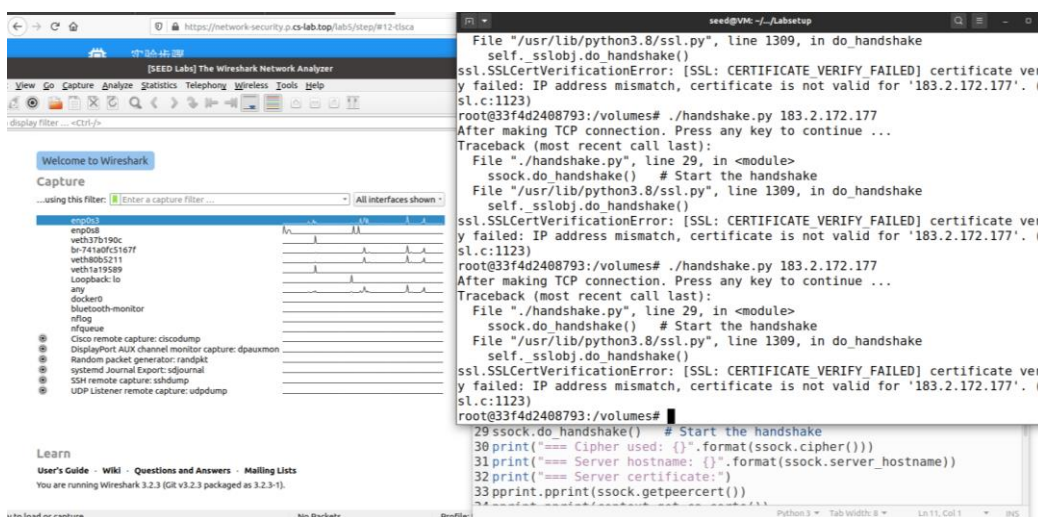
```
openssl x509 -in GlobalSign_Root_CA.pem -noout -subject_hash
```

最后执行结果：



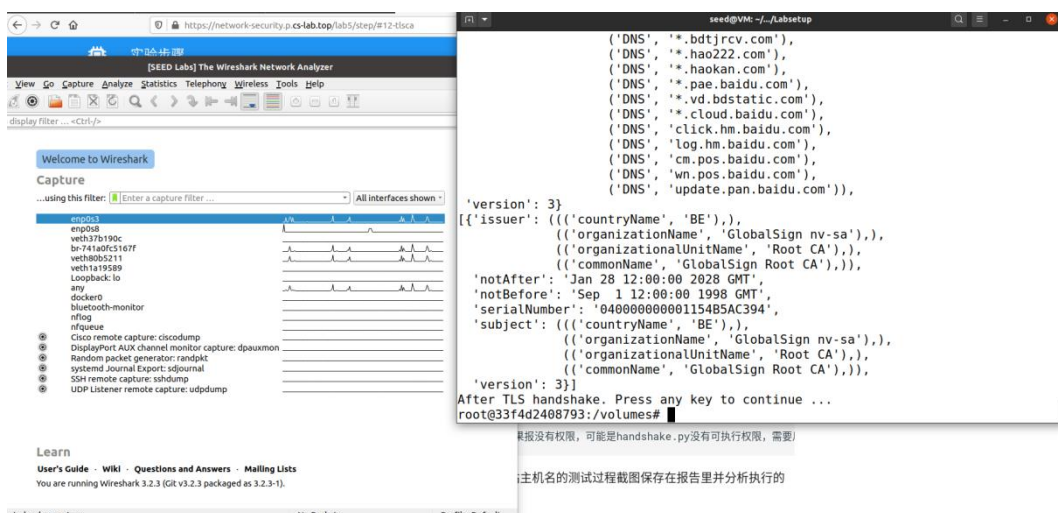
3. 请同学们将修改 `www.baidu.com` 网站主机名的测试过程截图保存在报告里并分析执行的结果，也可选用其他网站做测试。

主机名检测设置为 True:



客户端严格校验服务器证书的主机名，发现 `www.baidu1.com` 不在证书的 SAN (`www.baidu.com`、`www.a.shifen.com`) 中，拒绝连接

主机名检测设置为 False:



客户端忽略主机名校验，即使域名不匹配，仍建立 TLS 连接。

攻击者可通过 DNS 欺骗或伪造 /etc/hosts 诱导客户端连接到恶意服务器，导致中间人攻击（MITM）。

4.请分析 TLS 客户端编程和 server.py 的代码，说明客户端和服务端程序的关键步骤。

TLS 客户端（client.py）关键步骤

```
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
```

```
context.load_verify_locations(capath=cadir) # 加载 CA 证书目录
```

```
context.verify_mode = ssl.CERT_REQUIRED # 强制验证服务器证书
```

书

```
context.check_hostname = True # 校验主机名
```

作用：

指定使用 TLS 客户端协议。

加载受信任的 CA 证书，用于验证服务器证书。

启用证书和主机名验证，防止中间人攻击（MITM）。

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
sock.connect((hostname, port))
```

作用：

通过 TCP 连接到服务器的 hostname:port

```
ssock = context.wrap_socket(sock, server_hostname=hostname)
```

```
ssock.do_handshake() # 显式触发 TLS 握手
```

关键点：

server_hostname：指定预期的主机名，用于证书校验。

握手过程会验证服务器证书的有效性，包括 CA 签名和主机名匹配。

```
request = b"GET / HTTP/1.0\r\nHost: " + hostname.encode('utf-8')  
+ b"\r\n\r\n"
```

```
ssock.sendall(request)
```

```
response = ssock.recv(2048) # 读取服务器响应
```

作用：

应用层数据（HTTP）通过加密的 TLS 通道传输。

```
ssock.shutdown(socket.SHUT_RDWR)
```

```
ssock.close()
```

作用：

安全关闭 TLS 和 TCP 连接。

TLS 服务器 (server.py) 关键步骤

```
context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
```

```
context.load_cert_chain(SERVER_CERT, SERVER_PRIVATE) # 加载服务器证书和私钥
```

作用：

指定使用 TLS 服务器协议。

加载服务器证书和私钥，用于身份认证和密钥交换。

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
sock.bind(('0.0.0.0', 4433)) # 绑定所有接口的 4433 端口
```

```
sock.listen(5)
```

作用：

监听来自任意 IP 的 TCP 连接，端口需与客户端一致。

```
newsock, fromaddr = sock.accept()
```

```
ssock = context.wrap_socket(newsock, server_side=True)
```

关键点：

server_side=True：表明这是服务器端 TLS 套接字。

自动完成 TLS 握手，验证客户端证书（若配置了双向认证）

```
data = ssock.recv(1024) # 读取客户端请求（如 HTTP）
```

```
ssock.sendall(html.encode('utf-8')) # 发送响应（如 HTML 页面）
```

作用：

数据通过 TLS 加密传输。

```
ssock.shutdown(socket.SHUT_RDWR)
```

```
ssock.close()
```

作用：

安全关闭连接。

5.请分别用 client.py 和浏览器两种方式访问服务器，并记录你观察的结果（截图）

```
root@9381fc175dc2:/volumes# ./client.py www.bank32A.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
=== Server hostname: www.bank32A.com
=== Server certificate:
{'issuer': (((('commonName', 'www.modelCA.com'),),
              (('organizationName', 'Model CA LTD.'),),
              (('countryName', 'US'),)),),
 'notAfter': 'Apr 19 08:05:42 2035 GMT',
 'notBefore': 'Apr 21 08:05:42 2025 GMT',
 'serialNumber': '1001',
 'subject': (((('countryName', 'US'),),
                (('organizationName', 'hitsz Inc.'),),
                (('commonName', 'www.hitsz.edu.cn'),)),),
 'subjectAltName': (('DNS', 'www.hitsz.edu.cn'),
                    ('DNS', 'www.hitszA.edu.cn'),
                    ('DNS', 'www.hitszB.edu.cn')),
 'version': 3}
[{'issuer': (((('commonName', 'www.modelCA.com'),),
              (('organizationName', 'Model CA LTD.'),),
              (('countryName', 'US'),)),),
 'notAfter': 'Apr 19 06:23:42 2035 GMT',
 'notBefore': 'Apr 21 06:23:42 2025 GMT',
 'serialNumber': '6E4D6C8A68D2D1774F97A268A66664D4AC951A4F',
 'subject': (((('commonName', 'www.modelCA.com'),),
                (('organizationName', 'Model CA LTD.'),),
                (('countryName', 'US'),)),),
 'version': 3}]
After TLS handshake. Press any key to continue ...
[b'\nHTTP/1.1 200 OK',
 b'Content-Type: text/html',
 b'',
 b'\n<!DOCTYPE html><html><body><h1>This is Bank32.com!</h1></body></html>\n']
root@9381fc175dc2:/volumes#
```

```
seed@VM: ~/Labsetup
mitm-proxy-10.9.0.143 | root@1687a0cb67d5:/# server-10.9.0.43 | root@7b715550a
c:/# mitm-proxy-10.9.0.143 exited with code 137
server-10.9.0.43 exited with code 137
client-10.9.0.5 | root@33f4d2408793:/# client-10.9.0.5 exited with code 137
[04/21/25]seed@VM:~/../Labsetup$ dockps
a939d783bc48 mitm-proxy-10.9.0.143
9381fc175dc2 client-10.9.0.5
24e2ca66be7d server-10.9.0.43
[04/21/25]seed@VM:~/../Labsetup$ docksh 24
root@24e2ca66be7d:/# ./server.py
bash: ./server.py: No such file or directory
root@24e2ca66be7d:/# cd volumes
root@24e2ca66be7d:/volumes# ./server.py
Enter PEM pass phrase:
TLS connection established
"Request: b'GET / HTTP/1.0\r\n\r\nHost: www.bank32.com\r\n\r\n\r\n'"
TLS connection established
"Request: b'GET / HTTP/1.0\r\n\r\nHost: www.bank32.com\r\n\r\n\r\n'"
TLS connection fails
TLS connection established
"Request: b'GET / HTTP/1.0\r\n\r\nHost: www.bank32.com\r\n\r\n\r\n'"
TLS connection established
"Request: b'GET / HTTP/1.0\r\n\r\nHost: www.bank32A.com\r\n\r\n\r\n'"

```

二、遇到问题及解决方法

可能是第一次连续上了四学时的本门课程的实验课，导致到这次实验之后理解和注意力没那么集中，参考了网上相关博客和指导书一齐完成实验

三、对本次实验的建议

很好的一次实验体验

