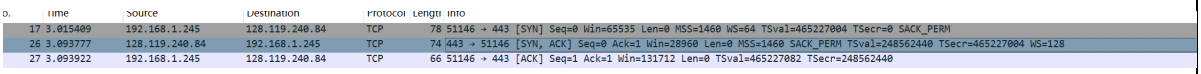
计算机学院 计算机网络 课程实验报告

实验题目：TLS		学号：202200400053
日期：2024-06-07	班级： 2 班	姓名： 王宇涵
Email：1941497679@qq.com		

实验方法介绍：

使用 wireShark 进行抓包并分析，理解了 TLS 的工作原理，巩固了理论课知识.

实验过程描述：

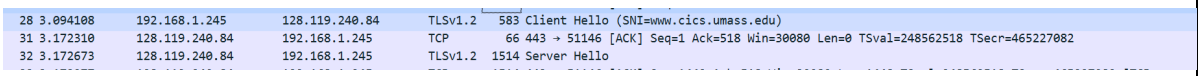


1. 在您的跟踪记录中，包含初始 TCP SYN 消息的包编号是多少？（“包编号”是指 Wireshark 显示左侧 “No.” 列中的编号，不是 TCP 段本身的序列号）。

答：17

2. 在客户端向服务器发送的第一个 TLS 消息之前或之后，TCP 连接是否已经建立？

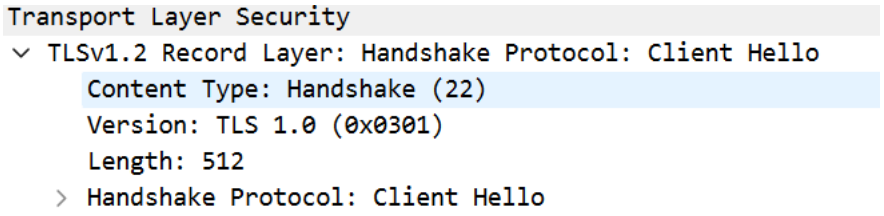
答：TCP 连接在第一个 TLS 消息发送之前已经建立



3. 在您的跟踪记录中，包含 TLS Client Hello 消息的包编号是多少？

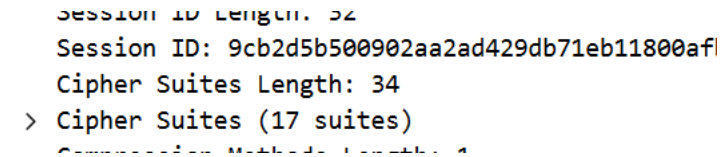
答：28

4. 在 Client Hello 消息中声明，您的客户端运行的 TLS 版本是什么？



答：TLS 1.0

5 在 Client Hello 消息中声明，您的客户端支持多少个加密套件？加密套件是一组相关的加密算法，决定了会话密钥如何派生，以及数据如何通过 HMAC 算法进行加密和数字签名。



答：17 个

6. 您的客户端在 Client Hello 消息中生成并发送了一串 “随机字节” 到服务器。Client Hello 消息中随机字节字段的前两个十六进制数字是什么？请输入两个十六进制数字（不带十六进制前缀 '0x'，并且在需要时使用小写字母）。

```
Random: 421623e04b909a780b955b1a679367e8af0312ec2362979794c50c162089004b
    GMT Unix Time: Feb 19, 2005 01:20:32.000000000 中国标准时间
    Random Bytes: 4b909a780b955b1a679367e8af0312ec2362979794c50c162089004b
```

答 ：4b

7. Client Hello 消息中的 “随机字节” 字段的用途是什么？注意：您需要进行一些搜索和阅读才能回答这个问题；请参阅 RFC 5246 的第 8.6 节（尤其是 RFC 5246 的第 8.1 节）。

答 ： 1. 防止重放攻击：通过使用随机字节，每次连接都会有不同的随机值，从而确保每个会话都是唯一的。这有助于防止重放攻击，因为即使攻击者拦截了之前的会话，随机值的不同使得重放这些会话数据无效。

2. 提供熵以生成会话密钥：客户端和服务器都会生成自己的随机值，并将其用于会话密钥的生成过程。两个随机值的结合（客户端的随机字节和服务器的随机字节）有助于确保生成的会话密钥具有足够的熵和不可预测性，从而增强加密的安全性。

3. 支持密钥交换协议：在一些密钥交换协议中（如使用 Diffie-Hellman 密钥交换），随机字节用于生成临时密钥对，从而进一步增强安全性。

```
  32 3.172673    128.119.240.84    192.168.1.245    TLSv1.2   1514 Server Hello
```

8. 在您的跟踪记录中，包含 TLS Server Hello 消息的包编号是多少？

答：32

9. 服务器从之前 Client Hello 消息中提供的加密套件中选择了哪一个加密套件？

```
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
```

答：选择了 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

10. Server Hello 消息中是否包含随机字节，类似于 Client Hello 消息中的随机字节？如果包含，它们的用途是什么？

```
Random: ad4543685c08b35ca6b696fcd26eaf9a275f67f37730fa82d5a570809ef8ab9f
    GMT Unix Time: Feb 13, 2062 03:20:08.000000000 中国标准时间
    Random Bytes: 5c08b35ca6b696fcd26eaf9a275f67f37730fa82d5a570809ef8ab9f
```

答：1. 生成会话密钥：在 SSL/TLS 握手过程中，客户端和服务器需要协商一个对称密钥（会话密钥），以便进行后续的加密通信。服务器在 Server Hello 消息中发送的随机字节用于生成这个会话密钥。

2. 防止重放攻击：随机字节的引入使得每个握手过程都是唯一的，从而防止了重放攻击。因为每次握手时随机字节都是不同的，即使攻击者截获了一个握手过程，也不能简单地重放它来进行恶意操作。

3. 生成安全参数：随机字节还可能用于生成其他安全参数，如初始化向量（IV），以确保后续的加密通信的安全性。

11. 在您的跟踪中，包含用于 www.cics.umass.edu 服务器的公钥证书的 TLS 消息部分中的数据包编号是多少？

```
  37 3.174259    128.119.240.84    192.168.1.245    TLSv1.2   1294 Certificate, Server Key Exchange, Server Hello Done
  38 3.174380    192.168.1.245     128.119.240.84   TCP         66 51146 → 443 [ACK] Seq=518 Ack=5325 Win=129792 Len=0 TSval=465227161 TSecr=248562520
  39 3.185548    192.168.1.245     128.119.240.84   TLSv1.2    192 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
  40 3.267472    128.119.240.84    192.168.1.245    TLSv1.2    340 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
```

答 ：37

12. 服务器可能返回多个证书。如果返回了多个证书，这些证书是否都是为 www.cs.umass.edu 服务器？如果不是全部都是为 www.cs.umass.edu，则其他证书是为谁的？您可以通过检查返回证书中的 id-at-commonName 字段来确定证书是为谁的。

```
subject: rdnSequence (0)
  rdnSequence: 6 items (id-at-commonName=InCommon RSA Server CA,id-at-organ
    RDNSequence item: 1 item (id-at-countryName=US)
    RDNSequence item: 1 item (id-at-stateOrProvinceName=MI)
    RDNSequence item: 1 item (id-at-localityName=Ann Arbor)
    RDNSequence item: 1 item (id-at-organizationName=Internet2)
    RDNSequence item: 1 item (id-at-organizationalUnitName=InCommon)
    RDNSequence item: 1 item (id-at-commonName=InCommon RSA Server CA)
RDNSequence item: 1 item (id-at-commonName=USERTrust RSA Certification Authority)
```

答：如图，还有 InCommon RSA Server CA 和 USERTrust RSA certification Authority

13. 为 id-at-commonName=www.cs.umass.edu 发布证书的认证机构的名称是什么？

```
RDNSequence item: 1 item (id-at-commonName=InCommon RSA Server CA)
```

答：InCommon RSA Server CA

14. 证书颁发机构（CA）用什么数字签名算法签署了此证书？提示：此信息可以在 www.cs.umass.edu 的证书的 SignedCertificate 字段的签名子字段中找到。

```
signature (sha256WithRSAEncryption)
    Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)
```

答：sha256WithRsAEncryption 算法

15. 让我们看看真实的公钥是什么样子！www.cics.umass.edu 使用的公钥的模数的前四个十六进制数字是什么？输入四个十六进制数字（十六进制数字之间没有空格，不包括任何前导'0x'，在需要时使用小写字母，并包括在'0x'之后的任何前导 0）。提示：此信息可以在 www.cs.umass.edu 的证书的 subjectPublicKeyInfo 子字段中找到。

```
subjectPublicKey [truncated]: 3082010a0282010100b39e7296158da80176a2f1035c7c61f06120f9852aad0d20d4931a30842
```

答：3082

16. 在您的跟踪中查找客户端与 CA 之间的消息，以获取 CA 的公钥信息，以便客户端验证服务器发送的 CA 签名证书确实有效，且未被伪造或篡改。您是否在跟踪中看到这样的消息？如果是，请说明客户端发送给 CA 的第一个数据包的编号是多少？如果没有，请解释为什么客户端没有联系 CA。

答：在 Wireshark 的捕获中，不会看到客户端直接联系 CA 的消息，因为客户端使用的是预先存储的 CA 公钥来验证服务器证书的有效性。客户端不会在 TLS 握手过程中发送任何数据包给 CA。因此，不存在客户端发送给 CA 的第一个数据包的编号。

Server Hello 消息总是以显式的 Server Hello Done 记录结束。

17. 在您的跟踪中，包含 Server Hello Done TLS 记录的 TLS 消息部分的数据包编号是多少？

37	3.174259	128.119.240.84	192.168.1.245	TLSv1.2	1294	Certificate, Server Key Exchange, Server Hello Done

答：37 号

39	3.185548	192.168.1.245	128.119.240.84	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

18.在您的跟踪中，包含公钥信息、Change Cipher Spec 和加密握手消息的 TLS 消息的包编号是多少？该消息是从客户端发送到服务器的。

答：39 号

19.客户端是否向服务器提供了自己由 CA 签名的公钥证书？如果是，包含客户端证书的包编号在您的跟踪中是多少？

答：没有提供自己由 CA 签名的公钥证书.

20. 客户端和服务器用于加密应用数据（在本例中为 HTTP 消息）的对称密钥加密算法是什么？

```
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
```

答：TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

21. 在 TLS 消息中，这个对称密钥加密算法最终是在哪个阶段决定和声明的？

答：在 TLS 握手过程中的 Server Hello 消息中，服务器会选择一个 Cipher Suite，并在 Server Hello 消息中通知客户端。这就是确定加密算法的阶段。

22. 在您的跟踪中，从客户端到服务器传输的第一个加密消息的包编号是多少？

41	3.267670	192.168.1.245	128.119.240.84	TLSv1.2	970	Application Data

答：41

23. 鉴于此跟踪是通过获取 www.cics.umass.edu 的主页生成的，您认为这个加密应用数据的内容是什么？

答：关于获取 www.cics.umass.edu 主页的 HTTP 请求。

24.哪个数据包编号包含了客户端到服务器的 TLS 消息，用于关闭 TLS 连接？由于我们的 Wireshark 跟踪中的 TLS 消息是加密的，我们实际上无法查看 TLS 消息的内容，所以我们只能在这里做出合理的猜测。

70	3.477860	192.168.1.245	128.119.240.84	TLSv1.2	896	Application Data
71	3.480462	192.168.1.245	128.119.240.84	TCP	78	51148 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=465227463 TSecr=0 SACK_PERM
72	3.483224	192.168.1.245	128.119.240.84	TCP	78	51150 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=465227465 TSecr=0 SACK_PERM
73	3.487400	192.168.1.245	128.119.240.84	TCP	78	51154 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=465227468 TSecr=0 SACK_PERM
75	3.489948	192.168.1.245	128.119.240.84	TCP	78	51155 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=465227470 TSecr=0 SACK_PERM
76	3.492405	192.168.1.245	128.119.240.84	TCP	78	51156 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=465227472 TSecr=0 SACK_PERM
79	3.499882	192.168.1.245	128.119.240.84	TCP	78	51158 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=465227479 TSecr=0 SACK_PERM
114	3.556988	192.168.1.245	128.119.240.84	TCP	66	51146 → 443 [ACK] Seq=2378 Ack=29611 Win=129600 Len=0 TSval=465227533 TSecr=248562902
115	3.556988	192.168.1.245	128.119.240.84	TCP	66	51146 → 443 [ACK] Seq=2378 Ack=32507 Win=126720 Len=0 TSval=465227533 TSecr=248562902
116	3.556988	192.168.1.245	128.119.240.84	TCP	66	51146 → 443 [ACK] Seq=2378 Ack=33559 Win=125632 Len=0 TSval=465227533 TSecr=248562902
117	3.557276	192.168.1.245	128.119.240.84	TCP	66	[TCP Window Update] 51146 → 443 [ACK] Seq=2378 Ack=33559 Win=129728 Len=0 TSval=465227533 TSecr=248562902
119	3.557796	192.168.1.245	128.119.240.84	TCP	66	51148 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=465227534 TSecr=248562903
120	3.558108	192.168.1.245	128.119.240.84	TLSv1.2	655	Client Hello (SNI=www.cics.umass.edu)

答: 推测为数据包编号为 70, 因为数据包为 120 时又重新开始 TLS 握手, 因此最后一个加密信息应该为请求关闭 TLS 连接的信息, 它的数据包编号为 70

分析：

在进行实验过程中，我们观察了 TLS 握手过程的各个阶段，并在 Wireshark 中分析了捕获到的数据包。

首先，我们确定了 TCP 连接的建立，然后观察了客户端和服务器之间的 TLS 握手。在 TLS 握手过程中，我们注意到客户端发送了 Client Hello 消息，其中包含有关其支持的 TLS 版本和加密套件的信息。服务器选择了加密套件，并发送了 Server Hello 消息。

然后，我们查看了包含服务器证书的 TLS 消息部分，并了解了服务器的公钥证书由 InCommon RSA Server CA 签名。我们还观察到了密钥交换过程和会话密钥的生成。通过查看 TLS 记录，我们发现了加密的应用数据，根据实验的目的，我们推测这些加密的应用数据是关于获取 www.cics.umass.edu 主页的 HTTP 请求。

最后，我们也尝试确定了 TLS 连接的关闭。虽然 Wireshark 中的 TLS 消息是加密的，但根据 TLS 连接关闭的模式，我们推测了关闭 TLS 连接的消息的数据包编号为 70。这种推测基于观察到在数据包编号为 120 时重新开始了 TLS 握手，因此最后一个加密信息应该是用于请求关闭 TLS 连接的消息。

结论：

TLS 协议是一种用于保护网络通信安全的加密协议，它提供了认证、机密性和完整性保护。在实验中，我们观察了 TLS 握手过程的各个阶段，包括客户端和服务器之间的通信和协商。TLS 握手过程包括 Client Hello、Server Hello、证书交换、密钥协商等步骤，通过这些步骤，客户端和服务器可以建立安全的通信连接，并协商出加密算法和会话密钥。在 TLS 连接建立后，数据包将通过使用对称密钥加密算法进行加密，以确保通信的安全性。

在 TLS 握手过程中，客户端发送 Client Hello 消息，其中包含有关其支持的 TLS 版本和加密套件的信息。服务器选择了加密套件，并发送了 Server Hello 消息，其中包含了服务器证书以及密钥协商参数。通过查看证书的内容，我们了解了证书的签发机构、公钥等信息，以及如何使用这些信息进行安全通信。TLS 协议还使用随机字节来增加通信的安全性，防止重放攻击和提供熵以生成会话密钥。

总的来说，TLS 协议是保护网络通信安全的重要协议，它通过加密和认证机制确保了数据的安全传输。通过这个实验，我对 TLS 协议的原理有了更深入的了解，并学会了如何使用 Wireshark 工具来分析 TLS 握手过程和加密通信。