A+

MAS 433 Assignment 2

Wang Xueou (087199E16)

November 5, 2011

Question 1 Solution:
SSL stands for Secure Sockets Layer. Netscape originally developed this protocol to transmit information privately, ensure message integrity, and guarantee the server identity. SSL works mainly through using public/private key encryption on data. It is commonly used on web browsers, but SSL may also be used with email servers or any kind of client-server transaction. For example, some instant messaging servers use SSL to protect conversations.

TLS stands for Transport Layer Security. The Internet Engineering Task Force (IETF) created TLS as the successor to SSL. It is ~~most often used as a setting in email programs, but~~, like SSL, TLS can have a role in any client-server transaction.

The differences between the two protocols are very minor and technical, but they are different standards. TLS uses stronger encryption algorithms and has the ability to work on different ports. Additionally, TLS version 1.0 does not interoperate with SSL version 3.0.

Question 2 Solution:

Contents	Description
Version	v1, v2, or v3
Serial number	Used to uniquely identify the certificate.
Signature algorithm ID	The algorithm used to create the signature
Issuer	(CA) The entity that verified the information and issued the certificate
Valid-from	The date the certificate is first valid from
Valid-to	The expiration date
Subject	The person, or entity identified
Subject public key info	The public key of the subject
Certificate signature	authenticate all the above contents, signed by the issuer (CA)
Optional information	

Question 3 Solution:
Simple TLS handshake

A simple connection example follows, illustrating a handshake where the server (but not the client) is authenticated by its certificate:

1. Negotiation phase:
    - A client sends a ClientHello message specifying the highest TLS protocol version it supports, a random number, a list of suggested CipherSuites and suggested compression methods. If the client is attempting to perform a resumed handshake, it may send a session ID.
    - The server responds with a ServerHello message, containing the chosen protocol version, a random number, CipherSuite and compression method from the choices offered by the client. To confirm or allow resumed handshakes the server may send a session ID. The chosen protocol version should be the highest that both the client and server support. For example, if the client supports TLS1.1 and the server supports TLS1.2, TLS1.1 should be selected; SSL 3.0 should not be selected.
    - The server sends its Certificate message (depending on the selected cipher suite, this may be omitted by the server).
    - The server sends a ServerHelloDone message, indicating it is done with handshake negotiation.
    - The client responds with a ClientKeyExchange message, which may contain a PreMasterSecret, public key, or nothing. (Again, this depends on the selected cipher.) This PreMasterSecret is encrypted using the public key of the server certificate.
    - The client and server then use the random numbers and PreMasterSecret to compute a common secret, called the "master secret". All other key data for this connection is derived from this master secret (and the client- and server-generated random values), which is passed through a carefully designed pseudorandom function.
2. The client now sends a ChangeCipherSpec record, essentially telling the server, "Everything I tell you from now on will be authenticated (and encrypted if encryption parameters were present in the server certificate)." The ChangeCipherSpec is itself a record-level protocol with content type of 20.
    - Finally, the client sends an authenticated and encrypted Finished message, containing a hash and MAC over the previous handshake messages.
    - The server will attempt to decrypt the client's Finished message and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.
3. Finally, the server sends a ChangeCipherSpec, telling the client, "Everything I tell you from now on will be authenticated (and encrypted, if encryption was negotiated)."
    - The server sends its authenticated and encrypted Finished message.
    - The client performs the same decryption and verification.

4. Application phase: at this point, the "handshake" is complete and the application protocol is enabled, with content type of 23. Application messages exchanged between client and server will also be authenticated and optionally encrypted exactly like in their Finished message. Otherwise, the content type will return 25 and the client will not authenticate.

Question 4 Solution:

4.1

4.1.1 Mozilla Firefox 7.0.1

4.1.2 TLS 1.0

4.1.3 CipherSuite supported by the browser:

4.2

4.2.1 TLS 1.0

4.2.2 ~~TLS_RSA_WITH_RC4_128_SHA (0x0005)~~

4.3

4.3.1 RSA public key crtptosystem, RC4 stream cipher, secure hash algorithm (SHA1)

4.3.2 DigiCert High Assurance CA-3

4.3.3 Valid from 9/28/2011; Valid until 10/1/2012

4.3.4

```
Modulus (2048 bits):
b3 de 8f c6 b8 46 81 d5 72 c2 87 4c 45 f8 2a a2
e0 89 b2 6c 0b 6c de b0 94 b2 7a 28 49 40 34 13
ee 6d 7e d4 f7 13 43 81 a4 57 b4 a1 24 9b a9 fa
af 39 1d 85 61 b2 48 cd d8 e7 11 9c c0 f8 42 85
c1 75 d3 19 b9 9f 58 0c 11 61 33 0c b8 bc 59 20
18 b2 93 d2 0c e9 c4 20 b9 ed a7 e0 85 e9 f4 52
f0 91 e4 2c 7c d7 d5 0e c2 e1 43 b5 3e 6c a1 b9
da 39 d5 e2 dd 52 b7 bf 51 07 d8 aa 80 dc ef 45
1a b8 7a 59 67 69 68 52 07 39 fd 1f b6 8c d9 ac
bb 8e c8 cb f8 98 40 68 ff f2 b3 22 b7 a7 24 e8
7d ab 47 2d 38 c6 18 0c 7f 97 8a 62 c1 1a 60 87
ce 95 83 d5 46 ff 1a ed 07 60 a2 86 96 d5 d9 bf
32 76 7b ef 88 dc f0 7b 4f 15 51 6b 59 4c 0e 4d
7d fb 7c 2b 4a 2d a0 d6 47 48 6f fa 83 c0 36 bb
c7 a4 6b 29 78 da ba a9 23 fd e6 d6 d6 66 89 27
d9 11 68 f9 92 5d 6c 1b 61 44 e9 5f 04 da 8d d3

Exponent (24 bits):
65537
```

It is strong.

4.4

If there is no PKC , the attacker man can launch the man-in-the-middle attack. Suppose that Alice is sending her public key to Bob

1. The attacker Eve can change the public key of Alice to Eve's public key.
2. Bob wants to use Alice's public key to establish secret key with Alice. But if Alice's public key has been changed to Eve's public key, Bob would use Eve's public key to establish a secrete key with Eve.
3. Eve establishes a secret key with Alice using the public key of Alice.
4. Eve receives the encrypted and authenticated data from Bob, then decrypts it, and transmits it securely to Alice; Eve receives the encrypted and authenticated data from Alice, decrypts it, and transmits it securely to Alice.

In this way, Eve can decrypt all the communication between Alice and Bob, but Alice and Bob do not notice the existence of Eve.

other algorithms: HMAC, SHA1, AES (or RC4)

4.5 RSA is used to generate the session keys used for the secure connection and also to establish identification during the handshake.

4.6 Normally the **Certification authority** (CA) and **public key certificate** are involved. Everyone knows the public key (for signature) of CA. The webmail server registers its public key with CA. CA signs (the server's information + the server's public key), and give the signature (certificate) to the server. A public key is sent together with the certificate for verification (using CA's public key). Thus my computer can verify the certificate to check whether the server is correct and whether the certificate is within the valid period. My computer also can verify the signature.