Assignment 3

Karl-Johan Grahn, 03005202

22nd September 2003

1. (a) SSL is vulnerable to man-in-the-middle attacks. Consider the handshake protocol: during establishment of security cababilities, the exchange is initiated by the client. He or she sends a *client_hello* message with, among all, a parameter *CipherSuite*. This is a list that contains the combinations of cryptographic algorithms supported by the client. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec; the choice of key exchange parameter is discussed subsequently.

   After sending the client_hello message, the client waits for the *server_hello* message, which containts the same parameters as the client_hello message. For the server_hello message, the CipherSuite field contains the single cipher suite selected by the server from those proposed by the client.

   The first element of the Cipher Suite parameter is the key exchange method. Anonymous Diffie-Hellman is supported among all methods. If chosen, the base Diffie-Hellman algorithm is used, with no authentication. That is, each side sends it public Diffie-Hellman parameters to the other, with no authentication. This approach is vulnerable to man-in-the-middle attacks, in which the attacker conducts anonymous Diffie-Hellman with both parties.

   An attack can be prevented by choosing another supported key exchange method, for example RSA, Fixed Diffie-Hellman or Ephemeral Diffie-Hellman.

   (b) SSL has some protection against reply attacks. Consider the handshaking protocol: when establishing security cababilities, the client initializes the exchange by sending a *client_hello* message with, among all, a parameter *random*, consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during key exchange to prevent replay attacks.

   After sending the client_hello message, the client waits for the *server_hello* message, which contains the same parameters as the

client_hello message. For the server_hello message, the random field is generated by the server and is independent of the client's random field.

To prevent a reply attack, the server sends a key exchange message with a signature created by taking the hash of a message and encrypting it with the sender's private key. The hash is defined as:

hash(ClientHello.random||ServerHello.random||ServerParameters)

where || is concatenation. So the hash covers not only the encryption parameters, but also the two nonces from the initial hello messages. This ensures against reply attacks and misrepresentation.

2. Yes, message seven is necessary since is concerned with the authentication of the initiator and the responder.

   Consider the Needham-Schroeder public-key authentication protocol. Each agent A possesses a public key, denoted K_a, which any other agent can obtain from a key server, including the server's own public key. The agent also possesses a secret key, K_a^{-1}, which is the inverse of K_a. We will write $\{m\}_k$ for message m encrypted with key k. We denote nonces by N_a and N_b: the subscripts are intended to denote that the nonces were generated by A and B, respectively.

   The Needham-Schroeder public-key protocol involves seven steps, and can be described as follows:

$$
\begin{array}{lcl}
1.\ A \rightarrow S & : & A, B \\
2.\ S \rightarrow A & : & \{K_b, B\}_{K_s^{-1}} \\
3.\ A \rightarrow B & : & \{N_a, A\}_{K_b} \\
4.\ B \rightarrow S & : & B, A \\
5.\ S \rightarrow B & : & \{K_a, A\}_{K_s^{-1}} \\
6.\ B \rightarrow A & : & \{N_a, N_b\}_{K_a} \\
7.\ A \rightarrow B & : & \{N_b\}_{K_b}.
\end{array}
$$

   Here A is an *initiator* who seeks to establish a session with *responder* B, with the help of trusted key server S. In step 1, A sends a message to the server, requesting B's public key. S responds in message 2 by returning the key K_b, along with B's identity (to prevent attacks based upon diverting key deliveries), encrypted using S's secret key (to assure A that this message originated from S). A then seeks to establish a connection with B by selecting a nonce N_a, and sending it along with its identity to B (message 3), encrypted using B's public key. When B receives this message, it decrypts the message to obtain the nonce N_a. It requests (message 4) and receives (message 5) A's

public key. It then returns the nonce N_a, along with a new nonce N_b, to A, encrypted with A's key (message 6). When A receives this message he should be assured that he is talking to B, since only B should be able to decrypt message 3 to obtain N_a. A then returns the nonce N_b to B, encrypted with B's key. When B receives this message he should be assured that he is talking to A, since only A should be able to decrypt message 6 to obtain N_b. Messages 1, 2, 4 and 5 are concerned with obtaining public keys, whereas messages 3, 6 and 7 are concerned with the authentication of A and B.

3. (a) In my view the most serious drawback is the poor defence against attacks involving malicious software. The Kerberos protocol rely on the fact that the Kerberos software is trustworthy. There's nothing to stop a hacker from surreptitiously replacing all client Kerberos software with a version that, in addition to completing the Kerberos protocols, records passwords. This is a problem with any cryptographic software package on an insecure computer, buth the widespread use of Kerberos in these environments makes it a particularly tempting target.

   However, after having used the UNIX-system at my homeuniversity KTH, Sweden, where Kerberos is implemented, I get the impression that Kerberos is a well thought-out protocol with overall good security and high transparency.

   (b) In version 4, tickets provided to clients are encrypted twice, once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful. In version 5 this double encryption has been removed.

   Version 4 requires the use of DES. In version 5, ciphertext is tagged with an encryption type identifier so that any encryption technique may be used. Encryption keys are tagged with a type and a length, allowing the same key to be used in different algorithms and allowing the specification of different variations on a given algorithm.