CSC474 - Information Systems Security: Homework4 Solutions

April 27, 2005

1. (40 points) Consider the following threats to Web security and describe how each is countered by a particular feature of SSL.

   (a) Brute-Force Cryptanalytic Attack: An exhaustive search of the key space for a conventional encryption algorithm.

   SSL uses symmetric one-time session keys and it has the capability to negotiate a stronger cipher to be used during session.

   (b) Known-Plaintext Dictionary Attack: Many messages will contain predictable plaintext, such as the HTTP GET command. An attacker constructs a dictionary containing every possible encryption of the known-plaintext message. When an encrypted message is intercepted, the attacker takes the portion containing the encrypted known plaintext and looks up the ciphertext in the dictionary. The ciphertext should match against an entry that was encrypted with the same secret key. If there are several matches, each of these can be tried against the full ciphertext to determine the right one. This attack is especially effective against small key sizes (e.g., 40-bit keys).

   SSL uses per session random numbers (of client and server) to generate the session key. It helps in randomizing the cipher text.

   (c) Replay Attack: Earlier SSL handshake messages are replayed.

   The random numbers used in each session has the first 4 bytes as the time stamp, so they are different for each session.

   (d) Man-in-the-middle attack: An attacker interposes during key exchange, acting as the client to the server and as the server to the client.

   Mutual authentication with certificates.

   (e) Password sniffing: Passwords in HTTP or other application traffic are eavesdropped.

   Passwords are encrypted.

   (f) IP spoofing.

   SSL does not use IP addresses to authenticate the client and server.

   (g) IP Hijacking: An active, authenticated connection between two hosts is disrupted and the attacker takes the place of one of the hosts.

If the attacker hijacks the connection after authentication, he has no way of knowing the encryption key. Therefore, the Alert protocol will detect if the attacker tries to send data as a legitimate user and closes the connection eventually. Even if the attacker hijacks it during handshaking, the attacker does not know the password and hence cannot succeed during the password authentication phase.

(h) SYN Flooding.

Can not be defeated. SSL is not stateless and working on top of TCP.

2. (10 points) Based on what you have learned in the class, is it possible in SSL for the server to reorder SSL record blocks that arrive out of order? If so, explain how it can be done. If not, why not?

   Yes, SSL has a sequence number that can be used to reorder packets. However, TCP will normally take care of it.

3. (10 points) In SSL, each data fragment is compressed before encryption. Can we encrypt the fragment first, and compress it later? Explain your answer.

   No, we can not, because the MAC is added after compression and before encryption. If we changed the order, the MAC would be unencrypted. Moreover, the compression ratio would be small because encryption will randomize the data, hence remove the patterns.

4. Compare the performance of doing PFS as implemented in SSL v3, vs. the recommended modification in §19.13.2 Exportability in SSLv3 vs. doing a Diffie-Hellman exchange for each connection. Assume the ephemeral key pair is of adequate length, rather than done to meet export rules. Assume also that PFS doesn't need to be "perfect", but rather the ephemeral key can change, say, every hour, and the cost of generating the key pair can be amortized over all the connection requests that occur during that hour.

   PFS,SSLv3, as implemented: client does a cheap RSA encrypt, and a cheap RSA verify signature (cheap because both use the public key). The server has an expensive decrypt and an expensive sign on every authentication.

   With the recommended modification: the server only has to do the expensive sign once an hour or so. The disadvantage of the modification is that it requires putting in an expiration time, and therefore requires relatively synchronized clocks.

   With Diffie-Hellman, each side has to do an expensive Diffie-Hellman exponentiation, and in addition the server would have to sign its Diffie-Hellman number on each interaction. If PFS is less than perfect, the server can reuse its Diffie-Hellman b value, and not have to keep re-signing that.

5. Design a variant of Kerberos in which the workstation generates a TGT. The TGT will be encrypted with the user's master key rather than the KDC's master key. How does this

compare with standard Kerberos in terms of efficiency, security, et.? What happens in each scheme if the user changes her password during a login session?

Let Alice be the user. In this variant, Alice's TGT is just $K_{Alice}\{$"Alice"$, S_A\}$, where Alice (actually, her workstation) has invented the session key S_A. This causes a problem for the KDC when it wants to get the session key from the TGT. To avoid this problem, Alice's name/instance/realm (unencrypted) should always be paired with the TGT when transmitted (it is already included in the encrypted TGT).

Since the only purpose of the TGT is to allow use of the short-term session key S_A instead of the long-term master key K_{Alice} when Alice and the KDC talk, and since knowledge of K_{Alice} is all that is required for mutual authentication of Alice and the KDC, there is no difference in security between the two schemes except when Alice changes her password (and thus her key K_{Alice} during a session. If Alice believes someone has learned her password and therefore changes it, with normal Kerberos the TGT obtained by the impostor remains valid until expiration. With the modified scheme, the TGT would be immediately invalidated, though tickets could continue to be used until expiration.

In normal Kerberos, Alice can change her password (and thus her key K_{Alice} during a session, because S_A is used for encryption, and S_A is in the TGT encrypted with K_{KDC} (so the TGT remains valid). In the variant, changing K_{Alice} invalidates the TGT, so to change her password, Alice would have to send a message containing the old TGT, and the new K_{Alice} encrypted with the unchanged S_A. She would then compute a new TGT for herself.

6. §13.5 Replicated KDCs explains that the KDC database isn't encrypted as a unit. Rather each user's master key is independently encrypted with the KDC master key. Suppose replication were done with a simple download (i.e., no cryptographic integrity check is performed). How could a bad guy who is a principal registered with a KDC impersonate Alice, another principal registered with that KDC? Assume he can see and modify the DKC database in transit, but that he does not know the KDC master key.

   All he has to do is replace Alice's encrypted key with his encrypted key.

7. Why is the authenticator field not of security benefit when asking the KDC for a ticket for Bob, but useful when logging into Bob?

   The authenticator in the TGS_REQ is used to prove knowledge of the session key. But without knowledge of the session key, the ticket requester can't decrypt the CREDENTIALS field in the TGS_REP to get the ticket. The authenticator in the AP_REQ is used to prove knowledge of the shared key. This is important because further message in the session may be unencrypted, so there is no other opportunity for authentication.