CS 6349 - Network Security – Fall 2017 Programming Project (Initial Draft)

Anirudh Rathinam (net-id: axr167530)
Utsav Vijay Dholakia (net-id: uxd150130)

Project Overview

This project attempts to implement a secure file transfer protocol to transfer data securely between the client and the server. The client authenticates the server allowing data to be uploaded to and downloaded from the server in a secure fashion. The project considers issues such as authentication, maintaining confidentiality and performing integrity checks to ensure secure data transfer between the client and the server.

Functional capabilities and security features of the model:

The functional capabilities are as follows:

- The model performs server authentication ensuring that the communications take place between the intended parties. This is achieved through the use of certificates where the server certificate is generated by a CA.
- The confidentiality of the data is achieved using the SHA-1 keyed hash function. The protocol proposed in this project takes into account various attacks against confidentiality such as traffic monitoring/analysis and man in the middle attacks and provides security measures to deal with such attacks.
- Each time communication is established, a unique session key is generated with the help of which communication takes place.
- In addition to the above, the project also ensures that the integrity of the data is maintained. It takes steps to identify modification of messages and ensures that there is no incorrect data injection without trace. This step is also achieved with the help of a keyed hash mechanism.

With the help of the above, the protocol allows for secure data exchange between the client and server. It allows the client to upload files to and download files from the server in a secure fashion and ensures that the integrity of data is maintained with the help of a keyed hash function.
Various well-known security attacks are considered such as the possibility of data fabrication, man in the middle attacks etc. and the protocol provides adequate preventive measures for these attacks.
The project will be implemented in Java.

Authentication

Authentication is achieved with the help of a Certification Authority (CA). To perform authentication, the certificate issued by the CA is sent to the client. The client uses the CA's public key to extract information from the certificate – namely the public key of the server.

The next step is to ensure that the client is communicating with the correct server. To achieve this, the client generates a random message (m) and encrypts it using the public key of the server. It then sends the encrypted message to the server.

The server can decrypt the message using its private key to obtain the message (m'). This message is then hashed and sent back to the client. The client then compares the hashed value of original message h(m) with h(m'). If they are both the same, the client knows that it is communicating with the correct server and communication is established.

Session Key Establishment

Once the authentication is successful, the client can generate a session key. The session key is then encrypted using the public key of the server and sent to the server. The server decrypts the message to obtain the shared session key.

To ensure that the data has not been tampered with during file transfer, the server can compute the hashed value of the session key it receives and send it to the client. The client can then compute the hash of the original session key and compare the values. If the values match, the client knows that they share the session key. Similarly, the client can send the hashed value back to the server and the server can perform the same comparison with its hashed value. In this way both the client and the server know that they share a common session key.

Data Confidentiality

The data to be exchanged once the session key has been established can be done as using a scheme described as follows:

- The message can be broken down into fixed length chunks (p1...pn) of the size of the message digest.
- We use intermediate blocks ($b_1...b_n$) and ciphertext blocks ($c_1...c_n$) computed as follows:
  b_1 = SHA(session key|IV) $c_1 = p_1$ XOR b_1
  b_2 = SHA(session key|c_1) $c_2 = p_2$ XOR b_2
  ...
  b_n = SHA(session key|c_{n-1}) $c_n = p_n$ XOR b_n
- Since only the ciphertext blocks are being transferred over the network, confidentiality is maintained.

- To obtain the original plaintext message, we simply need to XOR the ciphertext using the intermediate blocks (b1...bn). For example, $p_n = c_n$ XOR b_n.

Ensuring Integrity

To ensure integrity, we hash the plaintext blocks ($p_1...p_n$) and append the hashed value $H(p_i)$ to the ciphertext block. After the data is sent, the receiving party must first decrypt the ciphertext block 'c' to obtain the plaintext block 'p'.

The hash of 'p' is taken and compared with $H(p_i)$ that was sent with the ciphertext block. If the values are equivalent, we can be sure that the message has not been tampered with.

Threats, possible attacks and attack prevention measures

The protocol deals with several threats which are listed below:

- Identity theft/spoofing: The authentication step handles this with the help of the CA using public key cryptography.
- Traffic monitoring: Since the ciphertext is not decipherable to outside observers without knowledge of the shared session key, traffic analysis is no longer a threat.
- Man in the Middle attacks: If a third party intercepts the data, it will not compromise the security since the data has been encrypted and can only be decrypted by the intended parties.
- Data fabrication: This is handled in the integrity check. If the data has been tampered with, the hash values of the decrypted plaintext will not match the value sent along with the ciphertext block.

Hence by doing the above, we can securely transfer files from the client to the server as per the specifications of the project.