UNIVERSITÀ DI PISA

Master's Degree in Computer Engineering

Cybersecurity – Project Report

A.Y. 2018/2019

Students

Adriano Botti

Alessio Schiavo

Index

Project Specifications

The project consists in developing a secure client/server file-transfer application.

Functional requirements:

- The client must be able to upload/download any file sized up to max 4GB on/from the server.
- The client must be able to retrieve a list of the files currently hosted on the server.

Non-Functional requirements:

- The exchange of files must be memory efficient both for client and server, which implies using **incremental encryption;**
- The server must authenticate with a public key certified by a Certification Authority;
- Client must authenticate somehow, for example:
  - With a public key certified by a certification authority.
  - With a password pre-installed on server.
  - With a public key pre-installed on the server.
- Key establishment protocol must establish one (or more) symmetric session key(s) with public-key cryptography;
- Session protocol must use session key(s) to communicate;
- Communication must be **confidential**, **authenticated**, and **protected** against **replay attacks;**
- No coding vulnerabilities (use secure coding principles);
- Manage malformed messages;
- Use C or C++ language, and OpenSSL library for crypto algorithms;

Design Choices

The functional requirements, and the non-functional ones that do not involved a decision, were fully met. Regarding non-functional requirements which needed a choice, we proceeded as follows:

- The client authenticates with a public key certified by a Certification Authority (Simple Authority Software was used to issue certificates);
- The symmetric key exchange protocol establishes one session key, through public key cryptography using **RSA-2048 scheme**;
- The session protocol uses the established symmetric session key, encrypting and decrypting each message exchanged with **AES-CBC-128 block cipher in CBC mode**;
- Communication is confidential, with means of the symmetric session key, authenticated, with means of certificates exchange and HMAC (**SHA-256**), and protected against replay attacks, by means of a **counter**, initialized at 0 at the beginning of each session.

BAN Logic Proof of Key Exchange Protocol

Key Exchange Protocol

1. $M1 \quad C \rightarrow S : < Cert_c , Nc >$
2. $M2 \quad S \rightarrow C : < Cert_S, Ns >$
3. $M3 \quad C \rightarrow S : < \sigma_c(Ns) >$
4. $M4 \quad S \rightarrow C : < N_c, \{Ksc\}_{KpubC} , \sigma_S(N_c, \ \{Ksc\}_{KpubC}) >$

Idealized Protocol:

1. $M1 \quad C \rightarrow S : < Cert_c >$
2. $M2 \quad S \rightarrow C : < Cert_S >$
3. $M3 \quad C \rightarrow S : \{N_s\}_{KprivC}$
4. $M4 \quad S \rightarrow C : \{N_c, \ \{K_{sc}\}_{KpubC}\}_{KprivS}$

Assumptions:

1. $S \mid \equiv (S \overset{Ksc}{\longleftrightarrow} C)$
2. $S \mid \equiv \#(S \overset{Ksc}{\longleftrightarrow} C)$
3. $C \mid \equiv S \ \Rightarrow (S \overset{Ksc}{\longleftrightarrow} C)$
4. $C \mid \equiv S \ \Rightarrow \#(S \overset{Ksc}{\longleftrightarrow} C)$
5. $S \mid \equiv \xrightarrow{KpubC} C$
6. $C \mid \equiv \xrightarrow{KpubS} S$
7. $C \mid \equiv \#(N_c)$
8. $S \mid \equiv \#(N_s)$

Goal: Key Authentication

1. $S \mid \equiv (S \overset{Ksc}{\leftrightarrow} C)$
2. $C \mid \equiv (S \overset{Ksc}{\leftrightarrow} C)$

Proof

Goal 1 is satisfied by assumption 1.

Messages 1 and 2 can be ignored, since they are sent in the clear. From them we derive the assumptions 5 and 6.

M3:

$$\frac{S \mid \equiv \xrightarrow{KpubC} C, S \lhd \{N_c\}_{KprivC}}{S \mid \equiv C \mid\sim N_c}$$

M4:

$$\frac{C \mid \equiv \xrightarrow{KpubS} S, C \lhd \left\{N_c, \left\{(S \overset{Ksc}{\leftrightarrow} C)\right\}_{KpubC}\right\}_{KprivS}}{C \mid \equiv S \mid\sim < N_c, \left\{(S \overset{Ksc}{\leftrightarrow} C)\right\}_{KpubC} >}$$

then

$$\frac{C \mid \equiv S \mid\sim < N_c, \left\{(S \overset{Ksc}{\leftrightarrow} C)\right\}_{KpubC} >, C \mid \equiv \#(N_c)}{C \mid \equiv S \mid \equiv K_{sc}}$$

so

$$\frac{C \mid \equiv S \mid \equiv K_{sc},\ C \mid \equiv S \Rightarrow (S \overset{Ksc}{\leftrightarrow} C)}{C \mid \equiv K_{sc}}$$

Which is goal 2.

Messages Format

Session

- Command “list”
  - $C \rightarrow S : < iv, counter, \{"list"\}_{Ksc}, \ hmac(iv \mid counter \mid \{"list"\}_{Ksc}) >$
    [command_message]

  - $S \rightarrow C : < iv, counter + 1, \{list\_of\_files\}_{Ksc}, \ hmac(iv \mid counter + 1 \mid \{list\_of\_files\}_{Ksc}) >$
    [list_message]

- Command “upload”
  - $C \rightarrow S : < iv, counter, \{"upload"\}_{Ksc}, \ hmac(iv \mid counter \mid \{"upload"\}_{Ksc}) >$
    [command_message]

  - $C \rightarrow S : < iv, counter + 1, \{filePath \mid fileSize\}_{Ksc}, \ hmac(iv \mid counter + 1 \mid \{filePath \mid fileSize\}_{Ksc} >$
    [fileinfo_message]

  Sending/receiving each file chunk:

  - $C \rightarrow S : < iv, counter + 2, \{chunk\}_{Ksc}, \ hmac(iv \mid counter + 2 \mid \{chunk\}_{Ksc} >$
    [filechunk_message]

- Command “download”
  - $C \rightarrow S : < iv, counter, \{"download"\}_{Ksc}, \ hmac(iv \mid counter \mid \{"download"\}_{Ksc}) >$
    [command_message]

  - $C \rightarrow S : < iv, counter + 1, \{filePath\}_{Ksc}, \ hmac(iv \mid counter + 1 \mid \{filePath\}_{Ksc} >$
    [filepath_message]

  - $S \rightarrow C : < iv, counter + 2, \{fileExists\}_{Ksc}, \ hmac(iv \mid counter + 2 \mid \{fileExists\}_{Ksc} >$
    [fileExists_message]

  Sending/receiving file chunks:

  - $S \rightarrow C : < iv, counter + 3, \{chunk\}_{Ksc}, \ hmac(iv \mid counter + 3 \mid \{chunk\}_{Ksc} >$
    [filechunk_message]

- Command “quit”
  - $C \rightarrow S : < iv, counter, \{"quit"\}_{Ksc}, \ hmac(\ iv \mid counter \mid \ \{"quit"\}_{Ksc}\ ) >$
    [command_message]

  - $S \rightarrow C : < iv, counter + 1, \{quit\_ack\}_{Ksc}, \ hmac(\ iv \mid counter + 1 \mid \ \{quit\_ack\}_{Ksc}\ ) >$
    [quitack_message]

Field name	***Size [# of bytes]***
iv	*16*
counter	*4*
filePath	*100*
fileSize	*4*
hmac(…)	*32*
chunk	*4096*
<list>	*4*
<upload>	*6*
<download>	*8*
<quit>	*4*
list_of_files	*-*