Security — Wed, April 30th

Matt Bishop Lecture
transcribed by J. Fritz Barnes

June 2, 1997

1 Overview

- Authentication Protocols
  - Secret Key (Kerberos)
  - Public Key (X.509)

2 Authentication Protocols

This lecture covers authentication protocols. These protocols are designed for networks. The behind these protocols is that the protocol allows a "subject" usually a service/server as discussed below to validate that said user is who they really are. This lecture discusses the design of Kerberos a Secret-Key system and the X.509 protocol which uses Public Key Cryptography. We will be look at the protocols as well as the design and implementation flaws.

3 Secret-Key Authentication — Kerberos

3.1 Needham-Schroeder

Kerberos is a secret key system. It is based on the Needham-Schroeder protocol. There are two versions of the Needham-Schroeder protocol one which uses a secret key and the other which uses a public key cipher. We will discuss the secret-key form Needham-Schroeder which is the basis of the Kerberos implementation.

In Kerberos we are concerned with how to authenticate without placing a key on the network. The operation of the Needham-Schroder protocol utilizes an authentication server which acts as a broker between two users who are trying to get "acquainted." Our too users will be Alice and Bob. Bob would like to authenticate that Alice is who she says she is. The Authentication server contains keys for both Alice and Bob.

The stages in the protocol consist of:

1. **Ask for Credentials.** Alice sends a message to the Authentication Server which consists of: the sender, the recipient, and a nonce (i_a, a random number).

2. The Authentication server generates a random session key (S_k). This is returned in the following message: $\{i_a, B, S_k, \{A, S_k\}K_b\}K_a$. The nonce is returned to Alice with Bob's identifier and the session key. Additionally credentials is provided that encrypts Alices id and the session key with Bob's key. The whole message is encrypted with a key only Alice knows and sent to Alice.

3. Alice now attempts to authenticate with Bob. She sends a message to Bob consisting of the credentials returned from the server that contains Alice's id, the session key and is encoded with Bob's key. ($\{A, S_k\}K_b$).

The above falls under strong authentication. A problem with the protocol as outlined above is that it is vulnerable to replay. Craig could listen to the messages sent from Alice to Bob and replay them later to appear to be Alice from Bob's perspective. In order to solve this particular problem the following solution is utilized:

- Bob sends a nonce (i_b) to Alice which is enciphered with the session key.

- Using a well-known function f, Alice applies f to the nonce and sends a message back to Bob with $f(i_b)$ encoded with the session key.

This prevents replay since by definition i_b will not be the same for two different sessions, so therefore Craig cannot pretend to be Alice by just storing her previous message to Bob.

Dorothy Denning and Giovanni Maria Sacco proposed that the problem with this technique would be if someone "gets" the session key. In order to preempt this problem one can add origination times to messages. When one decides to perform this task one needs to consider how large a window is a valid time. Implementation issues related to this aspect are concerned with the unreliability of time (clocks on different computers are liable to have differences in the time.) Additionally this technique requires the storage of nonces. This is vulnerable to a flooding attack since when the "buffer" is full old nonces will be forgotten.

3.2 Kerberos

The Kerberos protocol builds on the Needham-Schroeder technique discussed above. There are two servers utilized in this system. One is the Authenticator, the other is the Ticket Granting Service. In this system tickets are created. The tickets do not verify who it was given too.

A client interacting with the Kerberos system is shown in Figure 1. The first message is sent from the client to the authentication server to make a request. The Authentication Server replies with among other things a session key. The client can use this later to apply to the Ticket Granting Service for a ticket (3), upon which the Ticket Granting Service will (possibly) grant it with a message (4). This Ticket can be used to talk to a service (5), e.g. the printer in Figure 1.

Implementations must be concerned with the following aspects:

- Where passwords are stored
- Built for single-user clients
- Encipherment overhead
- Session Key Generation

NOTE: Kerberos provides Authentication, but is not meant as an authorization system. Authorization should be performed in a seperate manner. NOTE: DCE provides Authorization but includes extra features to provide these services.

4 Public Key Authentication — X.509

This system uses public key certificates to authenticate a client. If Alice wishes to authenticate herself to Bob she sends the following token:

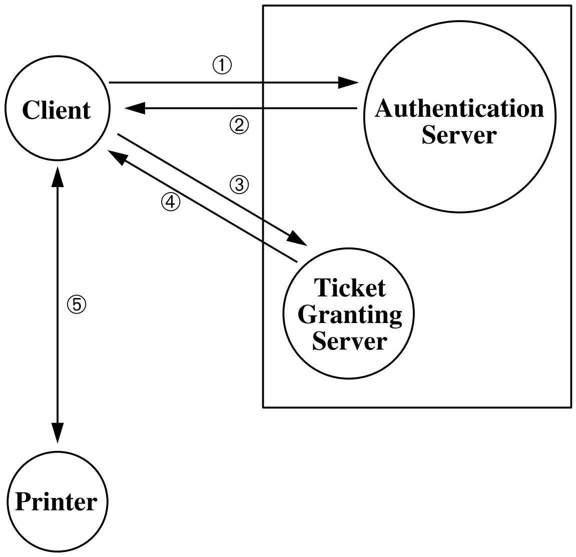

Figure 1: Interactions in Kerberos

$$A\{T_i, i_a, B, i_b, sgnData, \{EncData\}P_b\}\ data$$

In this technique we Alice digitally signs the token. This token contains a time stamp, two nonces, information about how the data passed is enciphered which is enciphered itself using Bob's public key. NOTE: If Craig intercepts it can send the information on pretending to be Alice. It won't be able to read the information Alice sent, but the reply sent by Bob might reveal what Alice was talking about.

To solve the above problem one can sign the token contents, i.e. add a field $\{hash\}S_k$ where $hash$ is a 1-way hash of the rest of the token and S_k is the session key.

4.1 One way client authenticate to server

The goals of the one way client authenticate to server are:

- Validate the Identity of Alice
- A generated token

- Identity of Bob. The message should only be useful to Bob.
- Integrity of Token
- Does not require a reply from Bob

In this protocol Alice sends the following message:

$$A, A\{T, i_a, B, sgnData, \{encData\}P_b\}$$

4.2 Dual Authentication

In the Dual authentication Alice wants to prove who she is to Bob and Bob proves that he really is Bob to Alice. The technique used to solve this problem is referred to as a layer technique, or Two-way handshake. First Alice sends the message as described for the one-way authentication. Afterwards, Bob performs the same role as Alice in the one-way authentication and sends a message to Alice.

4.3 Three-Way Handshake

The one-way and two-way authentication techniques have a problem with clock drift. In Unix systems there is an inherent problem with creating usable timestamps between systems. The three-way handshake remove the need for a timestamp.

In this protocol the dual authentication is performed. A third token is sent from Alice which consists of a publicly signed version of Bob's nonce. Bob then checks this against the nonce he sent previously to Alice to verify the correctness.

There is a basic flaw in this technique. The basic flaw consists of the Monkey in the Middle Attack. Using this attack, Craig tricks Alice into intitiating a conversation with Craig. He takes Alices message and sends it to Bob pretending to be Alice. Bob sends the proper responce back to Craig.

The fix for the Monkey in the middle attack is to add a last message from Alice that contains the i'_b and C's identity (since she is talking to Craig. When Craig passes this on to Bob, Bob will realize that Alice isn't really trying to talk to him and can ignore it.