# 11/14 Kerberos

Kristin Stephens

November 17, 2011

## 1 Overview

Using encryption and authentication alone is not enough to be secure. The following is a list of several attacks and defenses beyond just using encryption and authentication. Typically to help not having to think too hard all these defenses are incorporated into a secure channel.

- Replay
  - Sequence number
  - Timestamp
  - Nonce
- Reflection
  - To/From
  - Separate keys
- Deletion (Can't be prevented, but can at least detect it.)
  - Sequence number
  - Nonce
  - ACK

## 2 Reflection

In cryptography we can't actually authenticate a person, authentication is based on knowledge. B assumes A sent a message because they share a secret and the message is created using that secret. Unless the secret is compromised, only A or B could create a message with it.

**Attack** $A$ and $B$ share a symmetric key $k$. If $A$ sends to $B$ message $\{M\}_k$, $Eve$ can take the message $\{M\}_k$ and send it back to $A$. $A$ will assume $B$ sent the message.

**Defense** Include in the message the from and to. $\{A, B, M\}_k$

## 3 Key Exchange

There are two main methods for Alice and Bob to exchange keys. The first through a system like Kerberos where the trusted authority, we will call Trent in the rest of this paper, decides on a key and tells Alice and Bob what the *symmetric key* is. The second is through a *public key system (PKI)* where Trent signs the public keys of Alice and Bob and forwards the keys to them.

Neither model is secure if Trent is evil. In the symmetric key case Trent can snoop on the communication between Alice and Bob without detection. In the public key case a man in the middle attack (by Trent) can only be detected if there is an out-of-band channel between $A$ and $B$ to confirm the public keys and or messages.

The ability to trust Trent aside, each method has pros and cons. In terms of *availability* the PKI model is preferred because to communicate Alice and Bob only need to talk to Trent once. In the Kerberos model, Trent is needed to start any new communications. So if Trent is down no new communications can be initiated. In the case of *revocation* Kerberos is actually preferred. Revoking access in Kerberos is trivial, Trent simply does supply a key. Unfortunately comparing Kerberos and PKI in terms of revocation might not be an apples to apples comparison, so we are not sure if Kerberos is strictly better than PKI in terms of revocation.

In terms of certificate authorities. The certificate generators are Trent (e.g. Verisign) and their public keys are hard coded into your browser. This means that the browsers are automatically set to accept any certificates from these authorities.

## 4 Kerberos

Below is the series of messages Kerberos follows.

**This part is confusing and my notes don't make sense. Need to ask Dave and then fix.**

- $A \to T$: $\{A, T, t_0, B, t_{exp}, N_A\}_{k_{T,A}}$
- $T \to A$: $\{T, A, N_A, A, B, k_{A,B}, t_{exp}\}_{k_{T,A}}$
- $T \to B$: $\{T, B, t_1, A, B, k_{A,B}, t_{exp}\}_{k_{T,B}}$

One change to this sequence is the message from $T \to B$ can instead by sent to $A$ and $A$ can forward it.

**Problems** There are several problems with Kerberos. (1) It trusts timestamps for its security measures. This means the clocks on the machines have to be reasonably in sync. (2) It uses the user's password as the encryption key. (3) It did not have good randomness in the key generator.

## 5 Cross-frame Authentication

Various sites for a time did mash ups like iGoogle where there is an integrator and various widgets from third parties. The widgets are usually an iFrame to some third party. The widgets and integrator need to communicate, however there is the browser's same origin policy that requires only those with the same origin can communicate. To overcome this communication problem developers found they could use URL fragments, along with the fact the integrator could navigate a frame and the frames could navigate the outer most page.

**URL Fragment** A URL fragment is the identifier after a URL that is not actually sent to the server (e.g. the c in "foo.com/a/b#c"). An integrator can write to a widget by setting its fragment. The widget can read the fragment. The widget can do the same to the outer most page. Due to same origin though they can both write but not read the fragment. However this still enables two way communication.

**Secure Communication** Since the same origin policies makes this a write, but not read communication channel it is like it is encrypted. This means that the communication cannot be eavesdropped. However it does not have authentication. To add authentication they follow this protocol:

- $A$ generates nonce $N_A$
- $A$ sets the container's fragment to $\{N_A, URI_A\}$
- The container responds by writing to $A$'s fragment $\{N_A, N_C\}$
- $A$ now sends it's message by setting the container's fragment to $\{N_C, M\}$

But this is flawed! This is the Needham-Schroeder protocol from the 1970s, and an attack was found in 1976.

**Attack** An attack can occur if *Evil* can convince $A$ to communicate with it.

- $A \rightarrow Evil : \{N_A, URI_A\}_{k_{Evil}}$
- $Evil \rightarrow B : \{N_A, URI_A\}_{k_B}$
- $B \rightarrow A : \{N_A, N_B\}_{k_A}$
- $A \rightarrow Evil : \{N_B, M\}_{k_{Evil}}$
- $Evil \rightarrow B : \{N_B, M'\}_{k_B}$

**Defense** Put the name of who is sending the message in the double nonce message.

The URL fragment technique is actually no longer used. This kind of communicate is now done through the post-message.