A novel stateless authentication protocol

Chris Mitchell
Royal Holloway, University of London
c.mitchell@rhul.ac.uk

Security Protocols Workshop
Cambridge, 2nd April 2009

1. History

- It has long been recognised that a requirement for stored state is an undesirable feature in almost any protocol.
- During the 1990s considerable efforts were made to devise protocols which minimise the requirements for stored state at the server in client-server protocols.
- One major goal was to minimise the threat of DoS attacks.

State in the new world

- Whilst preventing exhaustion of table space was the original motivation for state elimination, there are other good reasons.
- It can greatly simplify network protocols by simplifying the associated state machines.
- The cost is slightly longer messages (messages are the new repository of state).
- Of course, this is not new at all – http cookies are hardly a revolutionary new idea!

Aura and Nikander, 1997

- Aura and Nikander published a key paper ‘Stateless connections’ in ICICS 1997.
- They describe how protocols can be made stateless by ‘passing the state information between the protocol principals along[side] the messages’.
- Such state information (forming a cookie – as in http) can be protected using a MAC computed using a server secret key.

Oakley, Photuris, etc.

- Oakley, a protocol proposed for use in the Internet, and which also avoids the need for server state, was proposed at around the same time.
- Photuris, that can be regarded as a development of Oakley, is a session key management protocol defined in RFC 2522.

2. State elimination

- The emphasis of past work has been primarily on eliminating stored state at the server.
- However, in the new world of transient relationships, and peer/peer communications (not just client/server), it is necessary to try to protect both parties engaging in a protocol.

Simple idea

- Well, we could use time-stamp based protocols, e.g. of the form:

  $A \rightarrow B$: $t_A \,||\, f_{K_{AB}}(t_A || i_B)$

  where t_A is a timestamp, f is a MAC function, K_{AB} is a secret key shared by A and B, and i_B is an identifier for B.
- Such protocols are widely known and analysed (can be used twice for mutual authentication).
- Note also that || denotes concatenation (need to be careful here!).

Problems

- This approach requires securely synchronised clocks.
- This doesn't seem like a good solution for our transient relationship scenario – who defines how clocks should be synchronised?
- Anyway, it doesn't prevent replays in a short time window.

State elimination strategy

- If we want to avoid timestamps (and the associated problems) we need to go back to the 1997 Aura-Nikander paper.
- Whilst the emphasis then (and since) has been on eliminating server state, the ideas presented there work just as well in eliminating client state.
- Key idea: ‘passing the state information between the protocol principals along[side] the messages’.

3. Some failed ideas

- We use shared secret-based unilateral authentication protocols throughout as simple examples.
- We believe (hope!) that these protocols can be extended/modified to use asymmetric cryptography and/or provide mutual authentication.

Idea 1

- Use a two-pass nonce-based unilateral authentication protocol, modified to be stateless:

$A \rightarrow B$: $n_A \,||\, f_{K_A}(i_B || n_A)$

$B \rightarrow A$: $n_A \,||\, f_{K_A}(i_B || n_A) \,||\, f_{K_{AB}}(n_A || i_A)$

where n_A is a nonce chosen by A, K_A is a key known only by A (and used only for cookies), and other notation is as before.

- The string $[n_A \,||\, f_{K_A}(i_B || n_A)]$ functions as a cookie.
- We have moved A's stored state into the message.

Problems

- Good point is that A now only has to remember a single secret K_A.
- The main problem is that A cannot verify whether the cookie $[n_A \,||\, f_{K_A}(i_B||n_A)]$ is fresh.
- B can use the cookie to keep sending responses which will be accepted.
- Even worse, a third party could intercept and replay B's original response, which will be accepted.

Idea 2

- Use a timestamp instead of a nonce in a two-pass protocol.

  $A \rightarrow B$: t_A

  $B \rightarrow A$: $t_A \,||\, f_{K_{AB}}(t_A || i_A)$

  where t_A is a timestamp chosen by A, and other notation is as before.
- We don’t need synchronised clocks – only A checks the timestamp!

Problems

- Unfortunately, this scheme allows Gong-style **preplay** attacks.
- Suppose *C* wishes to impersonate *B* to *A* at some future time.
- *C* (pretending to be *A*) engages in the protocol with *B*, using a future value of *A*'s clock.
- *C* can now replay this message to *A* at the future specified time, and successfully impersonate *B*.

4. A fixed idea

- Combine the two ideas – use cookies and a timestamp-based nonce.

  $A \rightarrow B$: $t_A \,||\, f_{K_A}(i_B || t_A)$

  $B \rightarrow A$: $t_A \,||\, f_{K_A}(i_B || t_A) \,||\, f_{K_{AB}}(t_A || i_A || f_{K_A}(i_B || t_A))$

  where notation is as before.

- As in the previous case, we don't need synchronised clocks – only *A* checks the timestamp (which could just be a counter).

Discussion I

- We could include a session identifier in the cookie.
- This would enable *A* to match the response to a higher-layer protocol communications request (e.g. from an application).

Discussion II

- Replays within a time window are still possible.
- Two obvious ways of fixing this:
  1. Keep a log of recently accepted messages (not so nice – re-introduces state, albeit of a bounded size).
  2. Keep track of the timestamp/counter of the most recently received (accepted) message and only accept ‘newer’ messages.

5. Next steps

- Where do we go from here?
- There are many unresolved issues, e.g.:
  - Devise a mutual authentication scheme;
  - Provide schemes using other types of crypto;
  - Prove the protocols secure in an appropriate model (of course – fix them first if they get broken);
  - Consider possible applications.

Protocol design thoughts

- Think about application to various communications models – if all interactions are request-response, then stored state may be completely unnecessary.
- Even where a connection is set up, only a party wishing to initiate message transmissions, rather than responding to a request, needs to maintain state.

And, finally ...

- Should be clear that these ideas are not fully thought through.
- Would welcome collaboration to take ideas further – e.g. to help write paper for proceedings.
- ...
- Questions?

Open_TC EC Contract No: IST-027635

Royal Holloway University of London

The Open-TC project is co-financed by the EC.

If you need further information, please visit our website www.opentc.net or contact the coordinator:

Technikon Forschungs- und Planungsgesellschaft mbH
Richard-Wagner-Strasse 7, 9500 Villach, AUSTRIA
Tel. +43 4242 23355 – 0
Fax. +43 4242 23355 – 77
Email coordination@opentc.net