A novel stateless authentication protocol

Chris Mitchell

Royal Holloway, University of London

c.mitchell@rhul.ac.uk

1. History

Royal Holloway University of London

- 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

Royal Holloway University of London

- 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.
- Note that the idea of using cookies in this way appears to predate Aura and Nikander – Boyd and Mathuria point out that the idea occurs in the original version of Photuris, published in 1995.

Aura-Nikander focus

Royal Holloway University of London

- The focus of the Aura-Nikander paper was very much on reducing the state **held by a server** in client-server protocols.
- This has also held true for most (all?) subsequent work on designing protocols which minimise state.

Aura-Nikander key ideas I

Royal Holloway University of London

- They start by considering a generic client-server protocol:
  1. $C \rightarrow S$: $Message_1$ [S stores $State_{S1}$]
  2. $S \rightarrow C$: $Message_2$
  3. $C \rightarrow S$: $Message_3$ [S stores $State_{S2}$]
  4. $S \rightarrow C$: $Message_4$
  5. ...
- The need for S to store state can be avoided by including the server's state in the messages:
  1. $C \rightarrow S$: $Message_1$
  2. $S \rightarrow C$: $Message_2 \,||\, State_{S1}$
  3. $C \rightarrow S$: $Message_3 \,||\, State_{S1}$
  4. $S \rightarrow C$: $Message_4 \,||\, State_{S2}$
  5. ...

Aura-Nikander key ideas II

Royal Holloway University of London

- Need to protect integrity (and possibly the confidentiality) of the state information, since it is sent across an unprotected channel.
- Can use MACs (computed using a key K_S known only to the server) and a timestamp (using a server-based clock) to enable the server to check its state information:
  1. $C \rightarrow S$: $Message_1$
  2. $S \rightarrow C$: $Message_2 \,||\, Time_{S1} \,||\, State_{S1} \,||\, \mathrm{MAC}_{K_S}(Time_{S1} \,||\, State_{S1})$
  3. $C \rightarrow S$: $Message_3 \,||\, Time_{S1} \,||\, State_{S1} \,||\, \mathrm{MAC}_{K_S}(Time_{S1} \,||\, State_{S1})$
  4. $S \rightarrow C$: $Message_4 \,||\, Time_{S2} \,||\, State_{S2} \,||\, \mathrm{MAC}_{K_S}(Time_{S2} \,||\, State_{S2})$
  5. ...
- Need to allow for variable delays in receipt of replies – allows for state replay attacks.
- If necessary, can encrypt using server-owned secret key.

Aura-Nikander key ideas III

Royal Holloway University of London

- Aura and Nikander do not state this explicitly, but the MACed (and possibly encrypted) state needs to include the complete session context, e.g.:
  - name of the communicating client;
  - session identifiers;
  - session variables;
  - session keys;
  - ...

Oakley, Photuris, etc.

- Oakley, a protocol proposed for use in the Internet, and some versions of which also avoid 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.
- This work ultimately led to IKEv2, which has similar properties.

Other DoS remediation techniques

- Of course, avoiding stored state is by no means the only way of trying to prevent DoS attacks.
- State elimination addresses state-space-exhaustion attacks.
- Juels and Brainard (NDSS 1999) and Aura, Nikander and Leiwo (SPW 2000) discuss he use of ‘client puzzles’ to make a client do work before requiring a server to do significant work. This addresses attacks attempting to use all a server’s computational resources.

2. Universal state elimination

- As we have already noted, the emphasis of past work has primarily been 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.
- In the future, all machines may function as servers.

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

Royal Holloway University of London

- 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

Royal Holloway University of London

- 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

Royal Holloway University of London

- 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

Royal Holloway University of London

- 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

Royal Holloway University of London

- 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

Royal Holloway University of London

- 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

Royal Holloway University of London

- 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.

Other DoS attack issues

- In a world where client-client (as opposed to client-server) communications dominates, do we need to rethink our DoS countermeasures?
- Can we apply the client puzzle techniques in this environment?
- One obvious approach would involve suggesting that in a client-client world, interactions are typically still of the **requester-provider** form, in which case we can map requesters to clients and providers to servers, and use the same techniques as employed for client-server. Is this appropriate?
- Could we also use the computational asymmetries arising naturally in certain public key crypto schemes as ‘natural’ client puzzles?

And, finally ...

Royal Holloway
University of London

- Should be clear that these ideas are not fully thought through.
- Would welcome collaboration to take ideas further.
- ...
- Questions?