Hypertext Transport Protocol (HTTP) Session Continuation: Problem Statement

draft-williams-websec-session-continue-prob-00trust200902infoHTTP Session ProblemSecurity AreaInternet-Drafttocindent="no"comments="yes"inline="yes"

One of the most often talked about problems in web security is “cookies”. Web cookies are a method of associating requests with “sessions” that may have been authenticated somehow. Cookies are a form of bearer token that leave much to be desired. This document describes the session “continuation” problem for the HyperText Transport Protocol (HTTP).

1. Introduction

Today most web applications use “cookies” to associate HTTP requests with “sessions”. A “session” is a set of related HTTP requests (and responses), where the relation is to some request(s) that created the session. Some sessions are created by the act of authenticating a user, in which case the primary goal of “sessions” is to avoid having to re-authenticate the user on every request. Other times a session is created when a request is received that is not associated with any session, in which case the primary purpose of “sessions” may be to provide a pseudonymous identifier for an otherwise anonymous user. We call the mechanisms by which requests are strung into sessions “session continuation”.

“Cookies” are server-assigned bearer tokens -- nothing more, nothing less, though some cookies are used just to store things like “shopping cart” state. A bearer token is an octet blob which can be presented as-is, possibly repeatedly, to authenticate a user to some party; mere possession of the bearer token is sufficient to act on the user’s behalf to at least one service. As such they are susceptible to theft via passive attacks (eavesdropping) if not protected in some other way (e.g., by using TLS), or via active attacks such as BEAST and CRIME [http://www.xors.me/?attachment_id=3727], as well as to leakage in various ways [XXX expand].

We would like a session continuation mechanism to replace or augment cookies that has better security semantics than bearer tokens. In particular we would like a system that is not susceptible to theft via active attacks like BEAST and CRIME. We believe that such a scheme should use cryptographic algorithms and cryptographic session keys, and should be amenable to being keyed by HTTP- and web-authentication mechanisms. A new session continuation mechanism should be suitable for use in web and non-web HTTP applications, and should work even for unauthenticated sessions.

1.1. Motivation

The motivation for this document and the related session continuation protocol I-D.draft-williams-websec-session-continue-proto document is as follows. We want:

- A variable authentication token instead of (or in addition to) web cookies, for resistance to BEAST, CRIME, and other adaptive chosen plaintext active attacks on TLS.
- The ability to explicitly logout and destroy all session state even if the session has been compromised, assuming there is no Man In The Browser (MITB).
- The ability to manage sessions.
- The ability to negotiate replay protection.
- Cryptographic binding ("channel binding" RFC5056) to the lower transport layer (TLS, where available).
- Cryptographic binding to the user authentication mechanisms (where the authentication mechanism can export a secret value).
- The ability to use HTTP/Negotiate RFC4559 in such a way that a) new HTTP(S) connections need not result in re-authentication, b) does not strobgly bind requests in a single HTTP connection to the HTTP/Negotiate authentication that precedes them.

1.2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

2. Requirements

Any session continuation scheme to replace or augment cookies must provide the following functionality:

1. Support for authenticated and unauthenticated sessions alike.
2. Support for http: and https: both.
3. Session continuation must be possible to implement without keeping state on the server side (see below), and it must be possible to keep some state on the server and some on the client.
4. Resistance to active attacks on https. [NOTE: This should probably NOT be a requirement. Instead we should be happy to note where a proposed protocol provides this.]
5. Session continuation must be expressed via HTTP headers.
6. Session continuation header values must be cryptographically difficult for attackers to spoof, and servers must be able to validate these values.
7. Session continuation header values used with TLS must be cryptographically distinct from those used without TLS such that no such values taken from HTTP requests sent without TLS can be used in HTTP requests with TLS.

8. Session continuation must provide protection against man-in-the-middle (MITM) attacks when using TLS. (This is important when using anonymous Diffie-Hellman cipher suites for TLS, as well as when using server certificates from low-value Public Key Infrastructures (PKI).
9. Must support explicit session termination ("logout"), initiated by either party, client or server. Once a session is logged out there should be no way to use it again, even if any session keys are compromised. Note that this is not a deployment requirement, just a protocol requirement; a fully stateless deployment may not be able to implement faithful logout.
10. Must work across all types of proxies. Proxies that can modify the plaintext HTTP requests and responses can (but should not) interfere with any session continuation protocol.
11. Sessions should be tied to "origins"; multi-origin sessions (sharing sessions across servers) are allowed, but there are user interface considerations.

Can you move a session from one server to another? No, probably not. Servers can share sessions, so we need to at least be able to scope sessions to sets of servers or DNS sub-domains. This appears to require that sessions have names. Once we have proper session continuation we may well end up needing a mechanism by which to authenticate to a service as a user of a given session on a foreign service that is "friends" with the first.

Recommendations:

1. Session continuation SHOULD use proof-of-possession of secret session key(s).
2. Session continuation header values SHOULD include a cryptographically-secure value (indistinguishable from random) that can be validated by the server and is hard for attackers to guess.
3. Session continuation header values should be salted with a nonce to defeate BEAST- and CRIME-style active attacks.

2.1. Statelessness

Session continuation protocols for HTTP MUST allow for stateless implementation on the server side, at least when TLS is used. Statelessness is not a requirement of deployments; implementations SHOULD support both, stateful and stateless servers. This generally means that any state must be encrypted and encoded into a session state cookie that is re-sent by the client to the server on every request. The server, of course, must be the one to assign such state, and it must use an encryption key known only by the server.

Server-side statelessness is NOT REQUIRED in actual deployments, but the ability to implement session continuation in a stateless fashion on the server side is REQUIRED.

Note that statelessness implies that there is no way to implement replay protection. In the case of session continuation with TLS this is not a concern because TLS itself protects against replays. But when session continuation is used without TLS then statelessness really does mean that there can be no replay protection (of course, this is also thus with web cookies). Therefore servers that require replay protection must either require the use of TLS or must use stateful sessions.

Note also that statelessness makes session logout a no-op on the server-side, which means that a compromised session can continue to be used even after a client attempts to logout. A session continuation protocol MUST allow for storing some state on the server, and some on the client, allowing deployments where the only state stored is the existence of a session.

Probabilistic data structures (e.g., Bloom filters) MAY be used to record logouts. This may require the ability to expire and refresh sessions to render the logout system scalable. In other words, HTTP responses MUST be allowed to replace session server state stored on the client side.

3. IANA Considerations

This document does not specify any protocols and has no IANA considerations.

4. Security Considerations

This document does not specify any protocols and is Informational. There are, however, few security considerations to document here.

We seek to improve security on the web (as well as for non-web HTTP applications) by:

1. reducing the need for expensive HTTP authentication exchanges (e.g., HTTP/Negotiate), thereby removing an obstacle to their use;
2. reducing exposure to session credentials theft via attacks on TLS such as BEAST and CRIME;
3. reducing exposure to session credentials theft when not using TLS;
4. introducing a replacement for / augmentation of cookies that will give browsers a chance to pursue better security policies.

As discussed in Section 2.1, there is a security consideration regarding session continuation without TLS and with server-side statelessness: there can be no replay protection in this case. However, this is not a loss of security relative to web cookies. Applications must use TLS if they require integrity protection.

5. Normative References

rfc2119 (http://xml.resource.org/public/rfc/bibxml/reference.RFC.2119.xml)

6. Informative References

rfc2616 (http://xml.resource.org/public/rfc/bibxml/reference.RFC.2616.xml)

rfc2617 (http://xml.resource.org/public/rfc/bibxml/reference.RFC.2617.xml)

rfc5246 (http://xml.resource.org/public/rfc/bibxml/reference.RFC.5246.xml)

rfc5056 (http://xml.resource.org/public/rfc/bibxml/reference.RFC.5056.xml)

rfc5929 (http://xml.resource.org/public/rfc/bibxml/reference.RFC.5929.xml)

rfc5849 (http://xml.resource.org/public/rfc/bibxml/reference.RFC.5849.xml)

rfc4559 (http://xml.resource.org/public/rfc/bibxml/reference.RFC.4559.xml)

<reference anchor='I-D.draft-williams-websec-session-continue-proto'> <front> <title>Hypertext Transport Protocol (HTTP) Session Continuation Protocol</title> <author initials='N.' surname='Williams' fullname='Nicolas Williams'> <organization /> </author> <date month='January' day='1' year='2013' /> <abstract><t>Abstract One of the most often talked about problems in web security is âcookiesâ. Web cookies are a method of associating requests with âsessionsâ that may have been authenticated somehow. Cookies are a form of bearer token that leave much to be desired. This document proposes a session âcontinuationâ protocol for HyperText Transport Protocol (HTTP).</t></abstract> </front> <seriesInfo name='Internet-Draft' value='draft-williams-websec-session-continue-proto-00' /> <format type='TXT' target='http://www.ietf.org/internet-drafts/draft-draft-williams-websec-session-continue-proto-00.txt' /> </reference>

7. Acknowledgements

The author thanks Yaron Sheffer, Yoav Nir, and Phillip Hallam-Baker, all of whom are practically co-authors, and invited to be listed as such. The term “session continuation” is Phillip’s. The motivation, requirements and recommendations text is a group effort.