Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2014

N. Williams
Cryptonector

July 8, 2013

# Hypertext Transport Protocol (HTTP) Session Continuation: Problem Statement
## draft-ietf-websec-session-continue-prob-00

Abstract

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).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2014.

Table of Contents

## 1. Introduction

Today most web applications use "cookies" to associate Hypertext Transfer Protocol (HTTP) [RFC2616] 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 Transport Layer Security (TLS) [RFC5246]), 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 document is as follows. We need:

- 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). (This feature being optional for stateless

servers.)

- The ability to manage sessions. For example, the ability to query session state.

- The ability to negotiate replay protection.

- Cryptographic binding ("channel binding" [RFC5056]) to the lower transport layer (TLS, where available), so as to reduce the cost of session continuation cryptographic protection. (That is, once a channel is bound to a session then there should be no further need to use cryptography for binding requests and responses to the session.)

- Cryptographic binding to user authentication mechanisms (e.g., 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 strongly 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, e.g., CRIME and BEAST. Specifically, an active attack on HTTPS that allows an attacker to recover traditional web cookies should not be sufficient to recover session continuation state with which to impersonate the user to the server. [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, at least when coupled with (bound to) user and/or server authentication, whether in HTTP or at the application layer. (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 the client. 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, at least those that do not destructively modify data nor metadata authenticated by session continuation.

11. Sessions should be tied to "origins"; multi-origin sessions (sharing sessions across servers) are allowed, but there are user interface considerations.

12. It must be possible to share sessions across multiple servers responding to the same origin (e.g., behind a load balancer). In particular, server implementations should agree on encrypted state cookie formats and/or stored state lookup procedures.

[[anchor1: 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 (this too will have user interface considerations).]]

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. For example, message authentication codes (MAC) using a shared session key and applied to all the data (e.g., request/response bodies) and metadata (e.g., some or all headers).

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 true of 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 generally (see note on probabilistic structures below) 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., adaptive 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.

### 4.1. On Origin

[[anchor2: We should probably discuss the notion of "origin" extensively.]]

### 4.2. User Interface Considerations

At its simplest, a session continuation protocol should only have an impact on how users manage sessions, namely: instead of (or in addition to) deleting cookies to destroy sessions, users may see an interface by which to logout sessions.

Any functionality for extending the scope of sessions, or adding origins to sessions, may require more extensive user interfaces.

## 5. Proposed Session Continuation Protocols

- [I-D.hallambaker-httpsession]
- [I-D.draft-williams-websec-session-continue-proto]
- <https://github.com/hueniverse/hawk>
- OAuth [RFC5849]
- and various others (the author begs forgiveness for those left out).

## 6. 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.

# 7. References

## 7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

## 7.2. Informative References

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.

[RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, July 2010.

[RFC5849] Hammer-Lahav, E., "The OAuth 1.0 Protocol", RFC 5849, April 2010.

[RFC4559] Jaganathan, K., Zhu, L., and J. Brezak, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, June 2006.

[I-D.hallambaker-httpsession] Hallam-Baker, P., "HTTP Session Management", draft-hallambaker-httpsession-01 (work in progress), May 2013.

[I-D.draft-williams-websec-session-continue-proto] Williams, N., "Hypertext Transport Protocol (HTTP) Session Continuation Protocol", draft-williams-websec-session-continue-proto-00 (work in progress), January 2013.

Author's Address

Nicolas Williams
Cryptonector, LLC

Email: nico@cryptonector.com