                    Draft LDAP Single Sign On Token Processing
                         draft-wibrown-ldapssotoken-00

Abstract

   LDAP Single Sign On Token is a SASL (Simple Authentication and
   Security Layer RFC 2222 [RFC2222]) mechanism to allow single sign-on
   to an LDAP Directory Server environment. Tokens generated by the
   LDAP server can be transmitted through other protocols and channels,
   allowing a broad range of clients and middleware to take advantage of
   single sign-on in environments where Kerberos v5 or other Single Sign
   On mechanisms may not be avaliable.

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 October 10, 2016.

Copyright Notice

Table of Contents

1. Introduction

The need for new, simple single sign-on capable systems has arisen
with the development of new technologies and systems. For these
systems we should be able to provide a simple, localised and complete
single sign-on service. This does not aim to replace Kerberos V5.
It is designed for when Kerberos is too invasive for installation in
an environment.

Tokens generated by this system should be able to be transmitted over
different protocols allowing middleware to relay tokens to clients.
Clients can then contact the middleware natively and the middleware
can negotiate the client authentication with the LDAP server.

   This implementation will provide an LDAP extended operation to create
   tokens which a client may cache, or relay to a further client. The
   token can then be sent in a SASL bind request to the LDAP server.
   The token remains valid over many binds. Finally, Tokens for a
   client are always able to be revoked at the LDAP Server using an LDAP
   extended operation, allowing global logout by the user or
   administrator.

2. Requirements Language

   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 RFC 2119 [RFC2119].

3. Format

   This document has two components. A SASL Mechanism, and LDAP
   extended operations.

   There is no strict requirement for the two to coexist: The LDAP
   Operation is an implementation of the service providing tokens, and
   the SASL Mechanism to authenticate them.

   In theory, an alternate protocol and database could generate and
   authenticate these tokens.

4. SASL Component

4.1. Token formats

   Token formats are server implementation specific: As they are the
   only entity that will decrypt and consume them, they have the option
   to provide these in any format they wish.

   This means the client will only see an opaque data structure, and
   will only need to transmit this opaque structure as part of the
   authentication request.

   For the token system to operate correctly the server MUST generate
   tokens that contain at least these three values:

   o Date Time Issued

   o Date Time Until

   o User Unique Id

As the client does not ever see the contents the User Unique Id can
be anything within the database that uniquely identifies the user
that is the holder of the token.

The User Unique Id MUST be an UTF8 String.

The token format MUST be encrypted. The token format can be
decrypted with either a asymmetric or symmetric keying system.

The token format MUST have a form of data authentication. This can
be through authenticated encryption, or validation of a hash.

The Date Time Issued MUST be a complete timestamp in UTC, to prevent
issues with changing timezones.

Without these guarantees, the token system is not secure, and is
vulnerable to credential forgery attacks.

Here is an EXAMPLE ASN.1 format that would be encrypted and sent to
the client:

```
LDAPSSOToken ::= SEQUENCE {
    DateTimeIssued GeneralizedTime,
    DateTimeUntil  GeneralizedTime,
    UserUniqueId   UTF8String }
```

                              Figure 1

This would be encrypted with AES-GCM and transmitted to the client.

Another example would be to use a fernet token Fernet Specification
[FERNETSPEC].

Version || Timestamp || IV || Ciphertext || HMAC

                              Figure 2

Timestamp can be considered to be the DateTimeIssued as:

"This field is a 64-bit unsigned big-endian integer. It records the
number of seconds elapsed between January 1, 1970 UTC and the time
the token was created."

We can then create a Cipher text containing:

Date Time Until || User Unique Id

                              Figure 3

The Date Time Until is a 64-bit unsigned big-endian integer. It is,
like Date Time Issued, the number of seconds since January 1, 1970
UTC, and the token creation time added to the number of seconds of
the requested life time.

This example format satisfies all of our data requirements for the
sso token system.

4.2. SASL Client

The client will request a token from the authentication server. The
acquisition method for the token is discussed in section XXX.

For authentication, the client MUST send the token as it was
received. IE changes to formatting are not permitted.

The client MAY transform the token if acting in a proxy fashion.
However this transformation must be deterministic and able to be
reversed to satisfy the previous requirement.

```
+-------+                 +-------------+              +--------+
| LDAP  |                 | HTTP server |              | Client |
|       |                 |             | <- Login --  |        |
|       | <-- Bind --     |             |              |        |
|       | - Success ->    |             |              |        |
|       | <- Req Token    |             |              |        |
|       | -- Token -->    |             |              |        |
|       | <- Unbind -     |             |              |        |
|       | - Success ->    |             |              |        |
|       |                 | Html Escape |              |        |
|       |                 |             | -- Safe -->  |        |
|       |                 |             |     Token    |        |
|       |                 |             |              | Store  |
|       |                 |             | < Request +- |        |
|       |                 | Reverse esc |    Token     |        |
|       | < Token Bind |              |              |        |
|       | - Success -> |              |              |        |
|       | <- Operation |              |              |        |
|       | <- Unbind -  |              |              |        |
|       | - Success -> |              |              |        |
|       |                 |             | - Response > |        |
+-------+                 +-------------+              +--------+
```
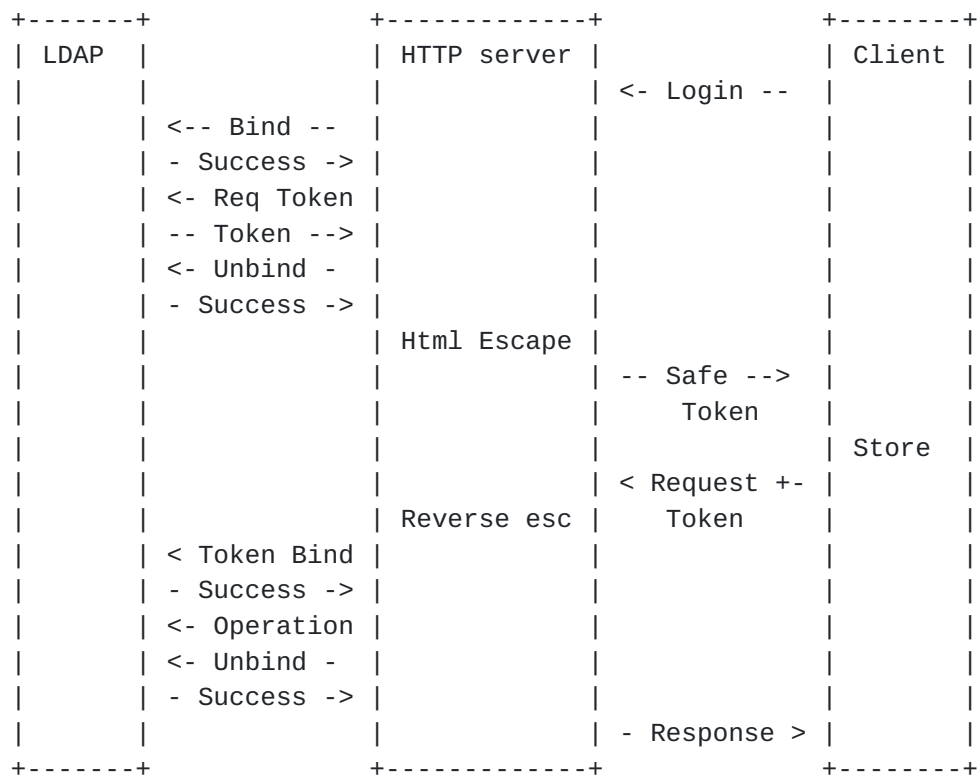
                           Figure 4

This example shows how a client is issued with a token when
communicating with a web server via the HTTP intermediate. The
Client does not need to be aware of the SASL/LDAP system in the

background, or the token's formatting rules. Provided the HTTP
server in proxy, if required to transform the token, is able to undo
the transformations, this is a valid scenario. For example, HTML
escaping a base64 token.

4.3. SASL Authentication

The client issues a SASL bind request with the mechanism name
LDAPSSOTOKEN.

The client provides the encrypted token that was provided in the
LDAPSSOTokenResponse Token Field.

The token is decrypted and authenticated based on the token format
selected by the server. The server MAY attempt multiple token keys
and or formats to find the correct issuing format and key.

If the token decryption fails, the attempt with this key and format
MUST be considered to fail.

If the values have been tampered with, IE hash authentication fails,
the attempt with the key and format MUST be considered to fail.

The token decryption MUST return a valid DateTimeUntil,
DateTimeIssued and User Unique Id. If this is not returned, the
decryption MUST be considered to fail.

If all token formats and keys fail to decrypt, this MUST cause an
invalidCredentials error.

The DateTimeUntil field is checked against the servers current time.
If the current time exceeds or is equal to DateTimeUntil,
invalidCredentials MUST be returned.

The User Unique Id is validated to exist on the server. If the User
Unique Id does not exist, invalidCredentials MUST be returned.

The DateTimeIssued field is validated against the User Unique Id
object's attribute or related attribute that contains "Valid Not
Before". If the value of "Valid Not Before" exceeds or is equal to
DateTimeIssued, invalidCredentials MUST be returned.

Only if all of these steps have succeeded, then the authentication is
considered successful.

4.4. Valid Not Before Attribute

The management and details of the "Valid Not Before" attribute are
left to the implementation to decide how to implement and manage.
The implementation should consider how an administrator or
responsible party could revoke tokens for users other than their own.
The Valid Not Before SHOULD be replicated between LDAP servers to
allow correct revocation across many LDAP servers. For example,
Valid Not Before MAY be an attribute on the User Unique Id object, or
MAY be on another object with a unique relation to the User Unique
Id.

5. LDAP Component

5.1. Token Generation

An ldap extended operation is issued as per Section 4.12 of RFC 4511
[RFC4511].

The LDAP OID to be used for the LDAPSSOTokenRequest is
2.16.840.1.113730.3.5.14.

The LDAP OID to be used for the LDAPSSOTokenResponse is
2.16.840.1.113730.3.5.15.

A User Unique Id is selected. This may be the Bind DN, UUID or other
utf8 identifier that uniquely determines an object.

The extended operation must fail if the connection SSF is 0.

Tokens must not be generated for Anonymous binds. This means, tokens
may only be generated for connections with a valid bind dn set.

Token requests MUST contain a requested lifetime in seconds. The
server MAY choose to ignore this lifetime and set it's own value.

A token request of a negative or zero value SHOULD default to a
server definied minimum lifetime.

The token is created as per an example token format in 4.1. This
value is then encrypted with an encryption algorithm of the servers
choosing. The client does not need to be aware of the encryption
algorithm.

The DateTimeIssued, DateTimeUntil and User Unique Id are collected in
the format required by the token format we are choosing to use in the
server. The token is then generated by the chosen algorithm.

The encrypted token is sent to the client in the LDAPSSOTokenResponse
structure, along with the servers chosen valid life time as a guide

for the client to approximate the expiry of the token. This valid
life time value is in seconds.

If the token cannot be generated due to a server error,
LDAP_OPERATION_ERROR MUST be returned.

5.1.1. Token Generation Extended Operation

```
LDAPSSOTokenRequest ::= SEQUENCE {
    ValidLifeTime INTEGER }

LDAPSSOTokenResponse ::= SEQUENCE {
    ValidLifeTime INTEGER,
    EncryptedToken        OCTET STRING
}
```

                              Figure 5

5.2. Token Revocation

An ldap extended operation is issued as per Section 4.12 RFC 4511
[RFC4511].

The LDAP OID to be used for LDAPSSOTOKENRevokeRequest is
2.16.840.1.113730.3.5.16.

The extended operation MUST fail if the connection is anonymous.

The extended operation MUST fail if the connection SSF is 0.

The extended operation MUST only act upon the "Valid Not Before"
attribute related to the bind DN of the connection.

Upon recieving the extended operation to revoke tokens, the directory
server MUST set the current BindDN's related "Valid Not Before"
attribute timestamp to the current datetime. This will have the
effect, that all previously issued tokens are invalidated.

This revocation option must work regardless of directory server
access controls on the attribute containing "Valid Not Before".

5.2.1. Token Revocation Extended Operation

The extended operation requestValue MUST not be set for LDAP SSO
Token revocation.

The extended operation does not provide a response OID. The result
is set in the LDAPResult.

5.3. Binding

This assumes the successful completion of the SASL Authentication
with a succesful result.

The Object is retrived from the User Unique Id, and a Bind DN
Determined.

The current Bind DN MUST be set to the Bind DN of the object that is
determined, and the result ldap success is returned.

6. Requirements

The SASL mechanism, LDAPSSOTOKEN, MUST be registered to IANA as per
RFC 2222 [RFC2222] Section 6.4

7. References

7.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/
            RFC2119, March 1997,
            <http://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[FERNETSPEC]
            Maher, T. and K. Rarick, "Fernet Specification", 2013,
            <https://github.com/fernet/spec/blob/master/Spec.md>.

[RFC2222]   Myers, J., "Simple Authentication and Security Layer
            (SASL)", RFC 2222, DOI 10.17487/RFC2222, October 1997,
            <http://www.rfc-editor.org/info/rfc2222>.

[RFC4511]   Sermersheim, J., Ed., "Lightweight Directory Access
            Protocol (LDAP): The Protocol", RFC 4511, DOI 10.17487/
            RFC4511, June 2006,
            <http://www.rfc-editor.org/info/rfc4511>.

Authors' Addresses

   William Brown
   Red Hat Asia-Pacific Pty Ltd
   Level 1, 193 North Quay
   Brisbane, Queensland 4000
   AU

   Email: wibrown@redhat.com

   Simo Sorce (editor)
   Red Hat, Inc.

   Email: simo@redhat.com

   Kieran Andrews (editor)
   The University of Adelaide
   Adelaide, South Australia 5005
   AU

   Email: kieran.andrews@adelaide.edu.au