Client Identity Federation with HSSO Overview

HADOOP-9533 Centralized Hadoop SSO/Token Server

Author: Larry McCay
Date: 05.04.2013

Overview

In order to meet the requirements of various Hadoop deployment scenarios that are emerging from the enterprise and cloud environments, the Hadoop ecosystem needs to extend the integration capabilities for identity and security policies available for each.

The ability to federated identities that are authenticated by the customer's preferred identity and access control systems is something that will extend these capabilities to meet the needs of enterprise, cloud and hybrid deployment scenarios.

The Hadoop Single Sign On (HSSO) service will provide this ability as a federating identity bridge between the provider's infrastructure and the Hadoop cluster. It will do this while utilizing a protocol inspired by the OAuth 2.0 Authorization Framework – see [RFC6749] in the references section below.

It will allow the integration of new client applications that may include mobile, middleware and 3rd party integration components and connectors. Easing the development and integration of these ecosystem participants will create great opportunities for the Hadoop community.

This document provides a high level overview of use of HSSO within the Hadoop cluster in order to federate an identity that was authenticated by a party outside of the Hadoop cluster. The actual formats of the requests and responses will be a matter of discussion within the community but examples of the general shape of these are provided below for context.

This overview is not meant to serve the purpose of a design document or provide specific implementation details as much as to provide a concise and understandable introduction of the proposed approach and need for collaboration and rationalization of related work within the community.

This document also references related efforts where alignment and rationalization is required.

Additional documents will delve into the details of the various protocols and token formats that are required to meet the needs of this goal of federation.

Security Context

This project intends to facilitate the federation of disparate single-sign-on and various identity tokens into a common Hadoop security context.

This context is comprised of token-based representations of the authenticated user, characteristics of the authentication event and claims about the authenticated user that can be used in the enforcement of access policy.

The tokens used for this security context are JSON Web Tokens (JWT) – see [JWT] in the references section below.

User Attributes

Provisions are made in this proposal to allow for fine-grained access control policy based on attribute server integrations that will include group and other user attributes. Attribute freshness is insured through configurable lifespan of service tokens - without requiring runtime acquisition for every access decision.

Delegation

The ability to delegate use of a Hadoop cluster to a third party on a user's behalf is addressed through the use of restricted cluster access tokens that limit the access the third party has while using the provided token.

IDP Discovery

In order for clients of Hadoop clusters to be able to interact with required services, the client needs to know what the accepted and trusted authentication mechanisms that are both available and compatible with the credentials that the client is able to present.

This document describes a REST based protocol to discover the most appropriate authentication provider (IDP) for a given Hadoop deployment and subsequently gain access to the Hadoop service by acquiring access tokens for the desired services.

Token Exchanges

Once authentication to a trusted IDP is accomplished, a sequence of token exchanges is undertaken in order to establish the required security context to request resources of the target Hadoop service.

Involved Parties

1. REST client code:
    a. C# client applications
    b. Java applications
    c. Knox DSL
    d. Browsers
    e. node.js
    f. curl
2. Hadoop CLI Clients
3. Hadoop SSO (HSSO) Service with roles:
    a. Gatekeeper
        i. Discovery protocol endpoint
    b. Service Registry
    c. Federating SP
        i. Composable Token Endpoints
            1. Pluggable authentication providers (HADOOP-9392)
            2. Common Hadoop Tokens (HADOOP-9392)

ii. Principal Mapping Provider
4. Trusted Identity Providers (IDPs)
5. Attribute Servers

General Flow

The following flow description is current thinking around discovering identity providers that have been configured for use Hadoop cluster identity federation.

The specifics of the token usage following the initial IDP discovery and interaction are informational and illustrate how the Hadoop identity federation is achieved.

It also indicates where rationalization with related efforts needs to be done and a means to eliminate or reduce the amount of cluster topology configuration needed by the client in order to interact with Hadoop services.

Client Interaction with HDFS

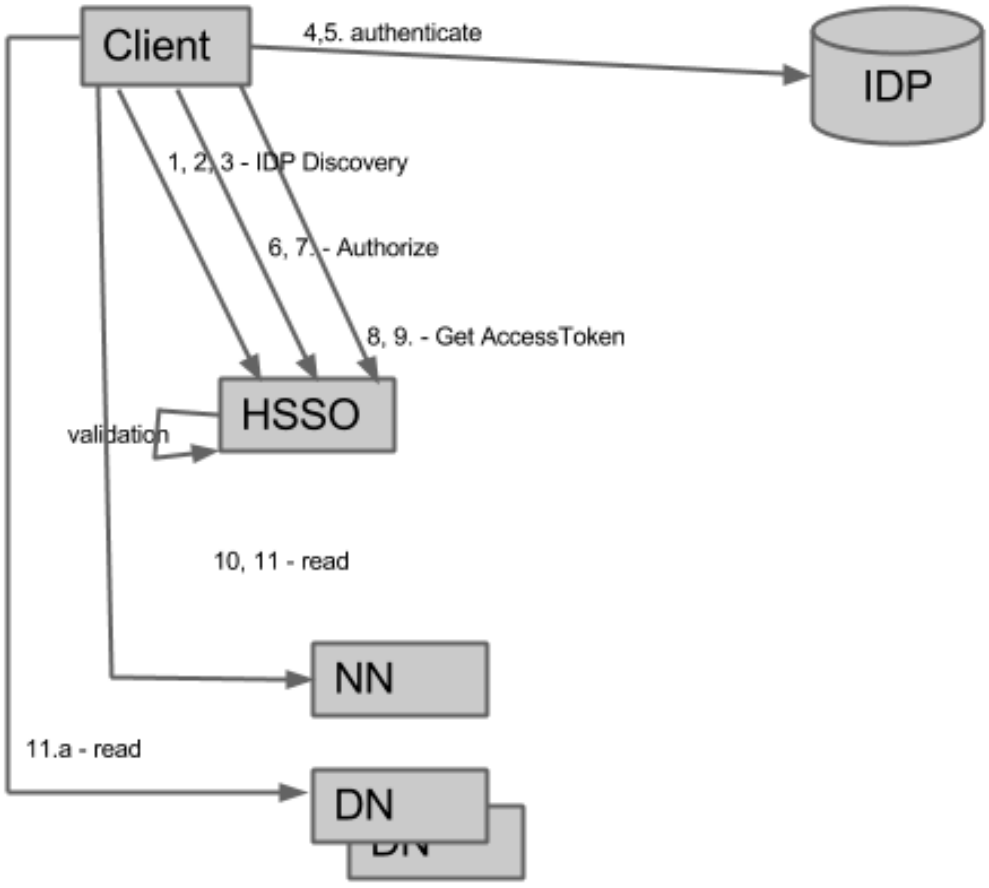

Figure 1. Client HSSO Interaction with HDFS

1. Client issues IDP Discovery Request (IDReq) indicating:

a. Target cluster id (default for)
b. IDP types available
c. User Domain identifier

2. Client issues GET to Discovery endpoint passing the IDReq from #1

```
$ curl
https://www.example.com/gatekeeper/idp_discovery?cluster_id=mycluster&idp_types=BASIC&domain=mydomain
```

listing 1. *non-normative example of IDP Dsicovery Request with curl*

3. The IDP Discovery service utilizes a best match algorithm to identify the list of trusted IDP/s from its configured set of trusted authentication providers and returns the IDP Dsicovery Response (IDRes) which includes:
    a. IDP name
    b. IDP description
    c. IDP URL for credential type and domain
    d. Authorization URL for acquiring a token for accessing the cluster

   The IDRes is not implemented as an HTTP redirect. This is for enabling the client to interrogate the returned payload and trigger appropriate behavior in a client specific way.

```
{
  "identity-providers":{
    "idp":[
      {
        "endpoints":{
          "idpURL":"https://www.example.com:8080/BASIC/DomainOne",
          "domain":"DomainOne",
          "idpType":"BASIC",
          "authURL":"https://www.example.com/gatekeeper/v1/authorize/BASIC"
        }
      }
    ]
  }
}
```

listing 2*. non-normative example of IDP Discovery Response*

4. Client interrogates IDRes payload and authenticates to the IDP at the IDP URL

5. Depending on its particular implementation, the IDP may redirect the user to Authorization URL or return an authentication response with an IDP specific token (idp_token)
6. Client issues Authorize request to the HSSO service at the Authorization URL returned in step #3 passing the idp_token received in step #5

```
$ curl
https://www.example.com/gatekeeper/v1/authorize/BASIC?idp_token=8w64rjhkdfsksdfk==
```

***listing 3**. non-normative example of HSSO authorize request with curl*

7. HSSO service's federating service provider (SP) endpoint has been composed to accept validate and normalize the incoming idp_token into a representation that can be used to create the requested token.
    a. Verifies the idp_token as trusted and valid according to IDP vendor specifications
    b. Normalizes the idp_token into a standard Java Subject with Hadoop specific principals that can be utilized as the single source from which tokens are created
    c. Passes the normalized token to the token builder appropriate for the token type requested – requested tokens may be of various types (for this flow description we will assume Cluster Access Token). For a description of the token types used in this flow see the Token Definitions section below.


```
{
  "exp":"1366404909",
  "sub":"joe",
  "aud":"HSSO",
  "iss":"https://www.example.com/gatekeeper/v1/authorize/BASIC",
  "tke":"https://www.example.com/gatekeeper/v1/token,
"code":"eyJhbGciOiAiUlMyNTYifQ.eyJpc3MiOiAiZ2F0ZXdheSIsICJwcm4iOiAiaGRmcyIsICJhdWQiOiAiaHR0cHM6Ly9sb2dpbi5oYWRvb3AuZXhhbXBsZS5vcmciLCAiZXhwIjogIjEzNjY0MDQ5MDkifQ.UuNZ9ntDEKV9hDh9btZdNcqTLerl8ykcWPrKIylbIcjMkv0CXmKD2JhF8d-tOC8qX9oGxmscDf-6h-JC00lg0mNGs7x9orXRPh0_pY0uA0AQhIHg1tJfk-bycYV_2RVKtaHr0ciIK_Lq1bHmw8UFDD3GAbL7Oqe4Hlmc_ZYvJYU"
}
```


***listing 4**. non-normative example of authorize response with Cluster Access Token*

The response includes

- the expiration period in milliseconds
- authenticated user's subject
- the audience (target service) for the provided code
- the issuer of the token

- the token endpoint for acquiring service access tokens
- the actual cluster access token code

The token itself is signed and encoded in accordance with standard JWT practices. [JWT-Bearer] (see: http://tools.ietf.org/html/draft-ietf-oauth-jwt-bearer-04)

8. Client extracts the Cluster Access Token from response and passes the token from the "code" attribute in its request for a Service Access Token to the Token Endpoint URL that was returned along with the Cluster Access Token. The Service Access Token Request (SATReq) includes
    a. Service name (HDFS)
    b. Cluster Access Token as Bearer Token

```
$ curl --Header "Authorization: Bearer
eyJhbGciOiAiUlMyNTYifQ.eyJpc3MiOiAiZ2F0ZXdheSIsICJwcm4iOiAiaGRmcyIsICJhdWQiOi
AiaHR0cHM6Ly9sb2dpbi5oYWRvb3AuZXhhbXBsZS5vcmciLCAiZXhwIjogIjEzNjY0MDQ5M
DkifQ.UuNZ9ntDEKV9hDh9btZdNcqTLerl8ykcWPrKIylbIcjMkv0CXmKD2JhF8d-
tOC8qX9oGxmscDf-6h-JC00lg0mNGs7x9orXRPh0_pY0uA0AQhIHg1tJfk-
bycYV_2RVKtaHr0ciIK_Lq1bHmw8UFDD3GAbL7Oqe4Hlmc_ZYvJYU"
https://www.example.com/gatekeeper/v1/token?service-name=HDFS
```

listing 5. *non-normative example of Service Access Token request with curl*

9. HSSO service's Token Endpoint is composed to:
    a. accept and validate Cluster Access Tokens
    b. extract relevant identity and authentication event attributes
    c. optionally perform principal mapping based on configuration
    d. integrate with the configured Attribute Server to acquire additional attributes for policy enforcement
    e. create a Service Access Token with these attributes to represent the effective security context for a Hadoop service interaction with the configured expiration
    f. return the requested Service Access Token along with the Service URL for accessing the requested service (Cluster Topology information may also be available through a separate topology API)

```
{
  "expires_in":1366409437840,
  "token_type":"Bearer",
  "service_url":"https:///www.example.com:50070/",
"access_token":"kmFgIiQa80jkGtWbUNoJ6Q+yyxXqAzGK50+7jetZYuv7_LLOtpzPMRYT
eLVUNdUnWy-Kthon6XT-KI"
}
```

listing 6. *non-normative example o f Service Access Token response*

10. Upon receiving the Service Access Token and Service URL the client is ready to interact with the target Hadoop service

11. Client issues request to target Hadoop service at the Service URL utilizing a Bearer token within the HTTP Authentication header to carry the Service Access Token [JWT-BEARER]
    a. In this particular scenario a secondary read is made to the DataNode also passing the Service Access Token as a Bearer Token

Client Interaction with MapReduce

The steps to interact with MapReduce are identical to those listed above for HDFS up until the details of step 11.

In this section we will explore the client's interaction with JobTracker and the subsequent interaction with NameNode and DataNode by the job itself. We also describe how delegation tokens are to be extended to leverage the service access token while otherwise remaining largely unchanged.

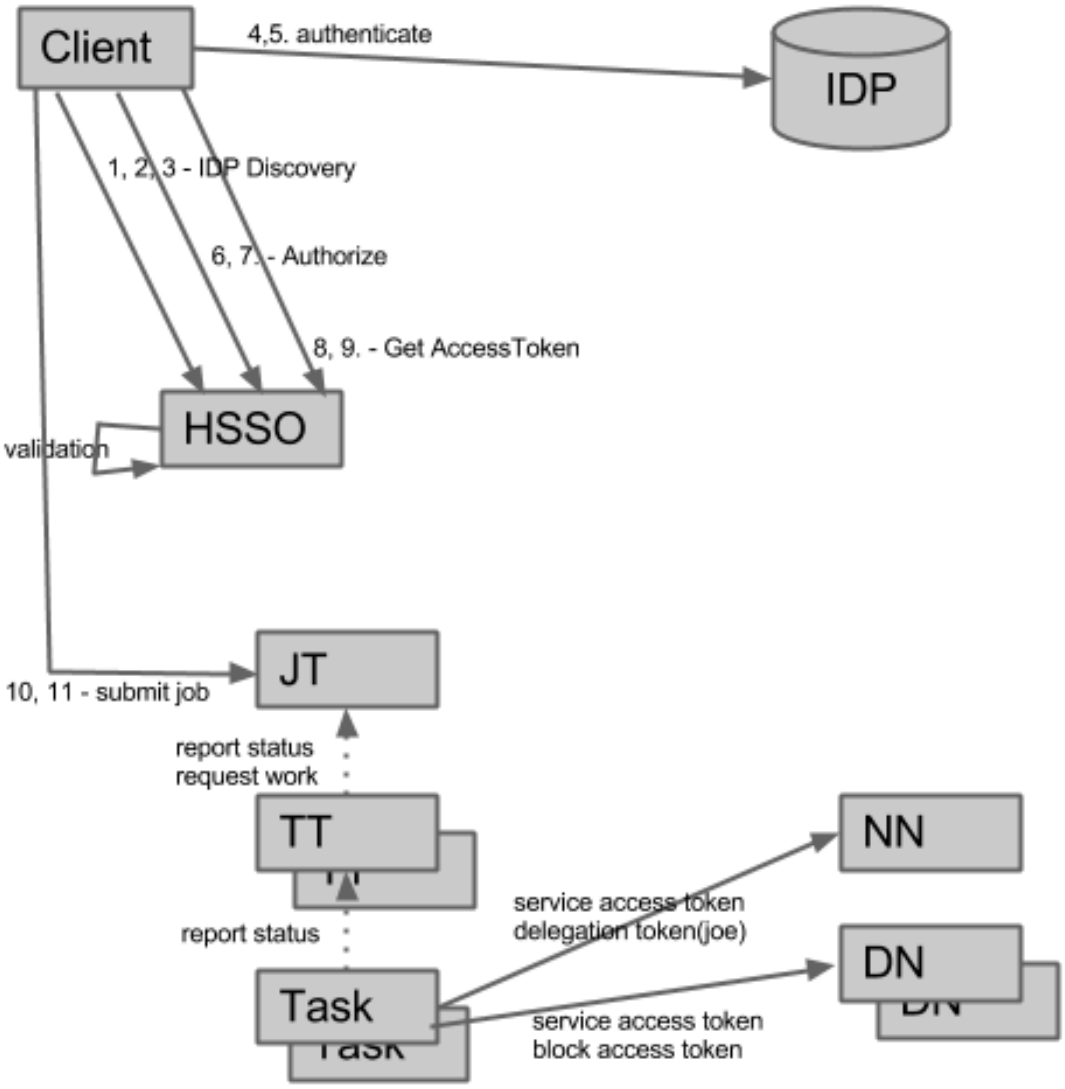

Figure 2. Client HSSO Interaction with MapReduce

While packaging the MapReduce job, the client writes the job configuration, the job classes/jar, the input splits and related metadata into their specific staging directory.

Delegation Tokens and Service Access Tokens

In order to access HDFS and other services, credentials must be provided in order for the job to access those services on the user's behalf.

The client must request a delegation token from the NameNode to provide to JobTracker for the job's restricted use to HDFS.

The NameNode will receive the service access token from the client and need to include that in the creation of the delegation token. (see open questions #5). The

delegation token will now contain the relevant claims that have been verifiably signed by the HSSO service and therefore trusted.

The expiration of the service access token is superseded by the validity period and subsequent renewal capabilities of the delegation token.

The format of the delegation token now takes a form similar to:

TokenID = {expirationDate, KeyID, ownerID, blockID, accessModes, serviceAccessToken} (again, see open questions #5).

The delegation tokens are signed with the private key of the NameNode – rather than with a shared secret – this allows for a simplified mechanism for rolling signtature keys. PKI allows the signer to make their public key available merely by publishing it. The use of the Credential Management Framework (a separate document will cover this in detail) will allow the publishing and acquisition of new public keys from a trusted service. (a separate document for in-cluster trust establishment will cover this in detail)

Token Definitions

1. **Cluster Access Token** – this token represents the authenticated identity as well as characteristics of the authentication event itself. Typical JWT claims as to expiry, issuer, audience, etc – as well as additional claims that represent the strength and number of factors involved in the authentication. It is used to acquire Service Access Tokens. Cluster Access Tokens expiration should be relative to the client usecases being supported by the particular token endpoint but are longer lived than Service Access Tokens.

   By utilizing the concept of scopes within a Cluster Access Token, we provide the ability to restrict the issuance of Service Access Tokens to those services represented as authorized scopes within the Cluster Access Token.

   The expiration period will be a configurable element of the endpoint that is serving cluster access tokens. In general, it would be longer lived than service access tokens and an appropriate default length could be in the order of days. Maybe 5 days would be an appropriate length.
2. **Service Access Tokens** – these tokens are used to actually interact with specific Hadoop services. They include a representation of the user identity just as cluster access tokens do but they contain additional information about the identity. This additional information is made available from a pluggable integration with Attribute Servers. These integrations provide attributes about the identity that can be used during the enforcement of fine-grained access policy. Since this additional information should be as dynamic as possible without requiring acquisition for every interaction, Service Access Token should be expired frequently. Service Access Tokens indicate the specific service for which they allow access as the audience (aud) claim within the token. The

validation of an incoming request requires that the audience be verified as the receiving service. Any service that receives a token that has a audience claim that does not belong to that service must be rejected with a 403 – unauthorized status.

Open Questions

1. How closely do we want to align with the OAuth profiles? The Cluster Access Token is very similar in form and intent to the OAuth Authorization Code and the Service Access Token is a form of OAuth Access Token.
2. How do we rationalize this with the work being done in HADOOP-9421 Generalize SASL Support with Protocol Buffer?
    a. This effort introduces a discovery protocol for clients to determine the SASL authentication mechanisms available for authenticating to a service.
3. How do we rationalize this proposal with RPC calls for things like:
    a. #1 above
    b. Bearer token equivalent
    c. Service URL publishing
4. How do we rationalize the token plans with HADOOP-9392 Token based authentication and Single Sign On?
    a. We need to be able utilize the pluggable authentication portion of the authentication framework in a way that will fit into the composable chains for processing token requests in HSSO. This may require the authentication mechanism to be decoupled from the Token Service described in the HADOOP-9392 design document. Perhaps the Token Service may be leveraged within the composable endpoints as well but they need to be able to work on a normalized representation of the authenticated user (see step 7b. above) rather than coupled tightly to the output of the AuthenticationService in the framework – also described in the design doc for Hadoop-9392.
    b. How do we rationalize the token types described in this flow and its targeted usecases as described in the Token Definitions section above?
5. How exactly do we want to accommodate the service access token with the concept of Delegation tokens? Possibilities include:
    a. Adding the service access token directly to the delegation token as noted earlier in this document. This would require the NameNode to accept delegation tokens and extract the service access token from it for verification and for that verification to ignore the expiration of the service access token. The expiration would be driven by the delegation token expiration and renewal mechanisms.
    b. NameNode could extract the relevant claims from the service access token and add them directly to the Delegation token. This would include attributes to be used for authorization policy enforcement as well. This would require some rework of the delegation token and

that NameNode continue to solely rely on delegation tokens for job interactions.

c. Replace the Delegation token with the service access token. They do both serve the same purpose as alternatives to authentication mechanism specific tokens for accessing a service. This would require a similar expiration/renewal mechanism as used for delegation tokens. Perhaps, it could just use the existing approach.
d. Service access tokens are persisted and presented along with delegation tokens that remain more or less the same.

6. Do we require separate access tokens for masters and related slave services? For instance, do we need to get a service access token for NameNode and then subsequently get another one for the target DataNode?
    a. This document assumes that we will have one service access token that is used across both masters and slaves. The use of a service access token for "HDFS" and a block access token together provide enough information for access to the DataNode.
    b. We could break them out separately. This would require more token requests to HSSO and an additional token to manage from a client or client's user agent perspective. It would however target each token to a specific service which may be considered a cleaner and more concise design.

References:

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", draft-ietf-oauth-json-web-token (work in progress), December 2012.

[JWT-BEARER] Jones, M., "JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0",)", draft-ietf-oauth-jwt-bearer-04(work in progress), December 2012.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[HADOOP-SEC] O'Malley, O., Zhang, Kan., Radia, S., Marti, R., Harrell, C., "Hadoop Security Design", October 2009.