Hive Security Thoughts

Rationale

- Provide access to arbitrary users
- Access control restrictions must be simple to manage and to enforce

Assumptions

- Authentication in scope
- Authorization is out of scope
  - Authorization is a solved problem in Hadoop for accessing files
  - Fine grained Access-Control that may involve different objects/tables/data in a query is at best solved at the Meta store (Howl).
- File system permissions are natively enforced by HDFS for the user of the request.

Authentication

- Will work differently for various Hive Interfaces
  - CLI
  - Hive Thrift Interface
  - Hive Web Interface
- Pluggable authentication support (Simple, Kerberos)
  - For Simple authentication, ugi parameter is provided
  - The delegation token is persisted after successful Kerberos authentication

Control Flow

- Client makes a call to Hive
- Use one of the hooks in the code to enforce authentication
  - SessionState.start()
  - Driver.new()
  - Servlet Filter for HWI?
- Get a list of Authentication providers from HiveConf
- Call authenticate on the provider which returns an Authentication Token
- Token will have the Principal among others
- Save the authenticated token in the session state

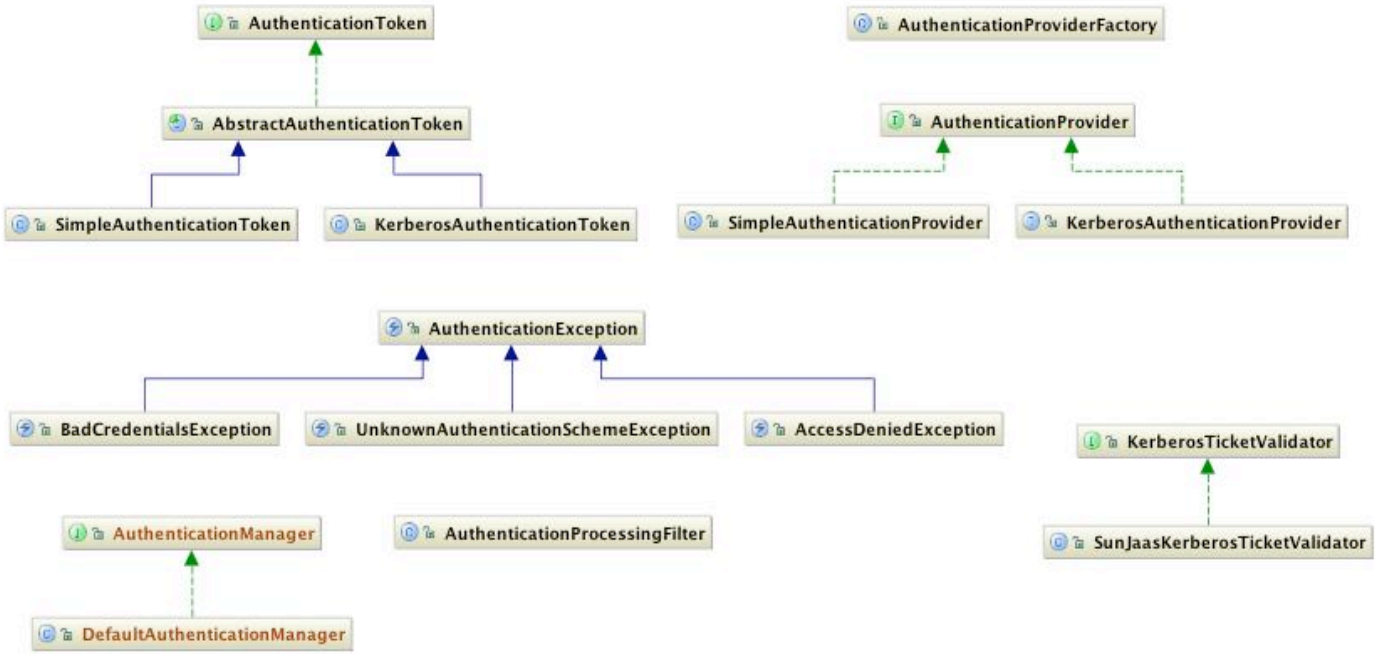
CLI

- Option 1: Delegate to Hadoop
  - User must have Kerberos ticket before using Hive CLI (The user must have TGT in its ticket cache)
  - PIG works this way
  - Queries fail when Hive talks to Hadoop
- Option 2: Build Authentication
  - Reuse what is already done in Hadoop core

Hive Server/HWI

- Hive will run as a super-user (Hadoop proxy user) enabling DO AS operations.
  - Impersonate the target user while accessing data on HDFS.
  - It'll execute jobs for other users using sudo (Java doAs(..)).
- Will perform Kerberos based authentication on all requests
  - Not sure how to do this in Thrift, Client?
  - Reuse what is already done in Hadoop core
  - HWI will use SPNEGO over HTTP on all requests

Class Diagram



Powered by yFiles

Thanks!