Security Design for Federation

1. Introduction:

Design doc for how to enable Kerberos security for router based federation

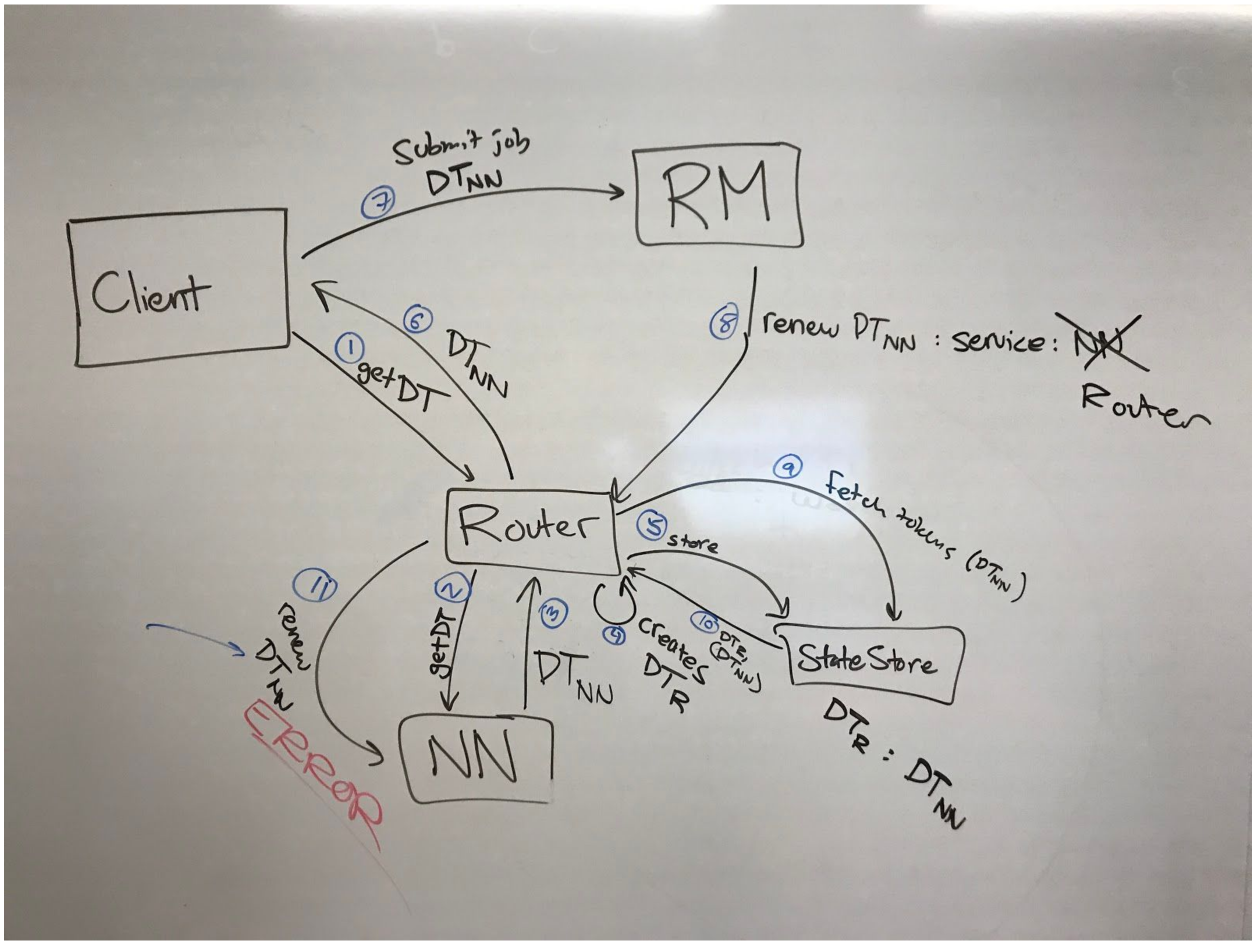
2. Design:

Our design is based on how security is originally implemented in HDFS, where the goal is to provide same level of security when the router is involved to proxy the request between client and hadoop HDFS cluster.

Authentication is provided by Kerberos, Authorization is maintained internally by HDFS and so will not be modified by adding a router as a proxy. Therefore, this design doc only cover the authentication by Kerberos.

KT:
When the client is connecting to the HDFS without the router, the client will be granted a Kerberos ticket to access NameNode. When the router is involved, all connection to the NameNode need to go through the router, therefore, native Kerberos authentication won't work. We proposed to have a authenticated user for the router to proxy for all real users, and therefore making router to authenticate on behalf of these users.

DT:



The DT is more complicated than KT, where the router would store a federated token that maps to all tokens for each NameNode when user start sent request getDelegationToken. Therefore, the client is able to connect to all of the NameNodes under the router federation.

There is an alternative proposal due the asymmetry of the previous proposal between KT and DT. KT fully depends on router to authenticate the user where NameNode would fully trust the router to do the correct things. Whereas the DT is granted by the NameNode and Router only proxy the DT back to the user. Changing the proposal to let router to grant DT for all of its namenodes would require more complicated changes and I am not sure if it worth it.

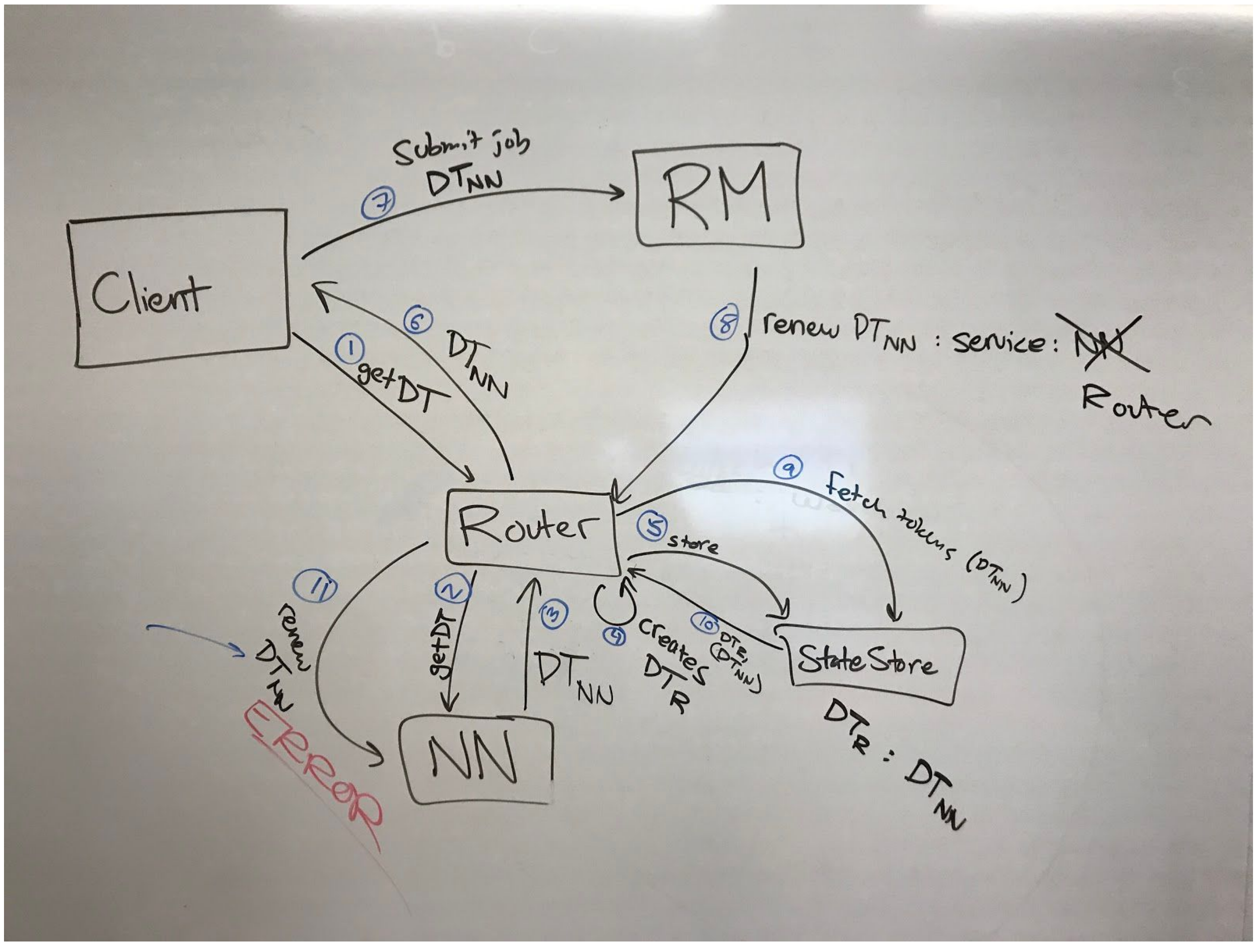
3. Implementation:

KT

1. set up keytab and principal and enable the security

2. proxy as the real user, when new user want to open a connection to the NameNode to get DTs, the router user would proxy as the real user, where the authorization is done towards the real user but the authentication is granted by the router user. Also, the corresponding NameNode need to set config to allow router user to proxy as given group of the real user.

DT



Create a token store for federated token
Implement getDT, renewDT, cancelDT
declare a federated token interface
manage the API for federated token store

Procedure:

Get Delegation Token:
Client send getDT request to router
router send getDT request to each namespaces, store all DT and create a federated token that maps to these DTs.

router send back the DT to the default NN back to the client(the reason why we choose not just return the federated token is to keep backward compatibility)

Renew Delegation Token:
client send renewDT request to router
router sends renewDT to each nameservice and return the smallest expiration date to the client

Cancel Delegation Token:
client send cancelDT request to router
router sends cancelDT to each nameservice

InvokeMethod:
Whenever the client is sending operations to the NN through the router, the router would search for the token stored for this user and add it to the UGI

Reference:

https://redmondmag.com/Articles/2012/02/01/Understanding-the-Essentials-of-the-Kerberos-Protocol.aspx?Page=1

https://issues.apache.org/jira/browse/HADOOP-4487

https://steveloughran.gitbooks.io/kerberos_and_hadoop/content/sections/what_is_kerberos.html
https://blog.cloudera.com/blog/2017/12/hadoop-delegation-tokens-explained/

https://www.slideshare.net/steve_l/hadoop-and-kerberos-the-madness-beyond-the-gate?qid=3fbf042e-3782-426f-b338-80dbcd6b729c&v=&b=&from_search=14d

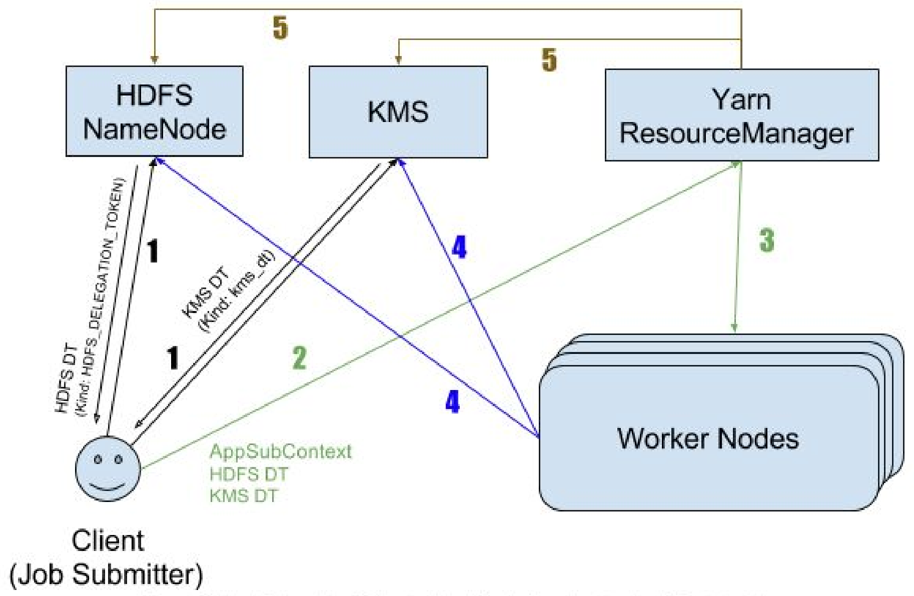


Figure 3: How Delegation Token is Used for Authentication In A Typical Job

For brevity, the steps of Kerberos authentication and the details about the task distribution are omitted. There are 5 steps in general in the graph:

1. The client wants to run a job in the cluster. It gets an HDFS Delegation Token from HDFS NameNode, and a KMS Delegation Token from the KMS.
2. The client submits the job to the YARN Resource Manager (RM), passing the Delegation Tokens it just acquired, as well as the ApplicationSubmissionContext.
3. YARN RM verifies the Delegation Tokens are valid by immediately renewing them. Then it launches the job, which is distributed (together with the Delegation Tokens) to the worker nodes in the cluster.
4. Each worker node authenticates with HDFS using the HDFS Delegation Token as they access HDFS data, and authenticates with KMS using the KMS Delegation Token as they decrypt the HDFS files in encryption zones.
5. After the job is finished, the RM cancels the Delegation Tokens for the job.