Securing Hadoop RPC using SSL
(v 1.0)

Design Contributors
Kartheek Muthyala
# Table of Contents

1. **Introduction** ..................................................................................................................... 1  
2. **Background** ....................................................................................................................... 2  
   2.1 How SSL Works? .................................................................................................................... 2  
   2.2 Hadoop RPC ...................................................................................................................... 3  
   2.3 SASL support in Hadoop RPC .......................................................................................... 3  
3. **Design** ............................................................................................................................. 3  
   3.1 SSL Configuration ............................................................................................................. 3  
   3.2 SSL Implementation .......................................................................................................... 4  
      3.2.1 Redesigning Server and Client classes ....................................................................... 4  
      3.2.2 Proposed implementation ......................................................................................... 5  
4. **References** ....................................................................................................................... 5
1 Introduction

This document describes the work required for securing remote procedural calls between different services in Hadoop using SSL and is captured under the Apache Jira - HADOOP-13836. Today, RPC connections in Hadoop are encrypted using Simple Authentication & Security Layer (SASL), with the Kerberos ticket based authentication or Digest-md5 checksum based authentication protocols. This proposal is about enhancing this cipher suite with SSL/TLS based encryption and authentication. Secure Sockets Layers (SSL) is a computer networking protocol for securing connections between clients and servers over an insecure network. This is a proposed Internet Engineering Task Force (IETF) standard that has made its way to a number of applications such as web browsing, internet faxing, messaging, VOIP etc. And supporting this cipher suite at the core of Hadoop would give a good synergy with the applications on top and also bolster industry adoption of Hadoop.

2 Background

2.1 How SSL works?

SSL protocol includes two phases – Handshake protocol and Record protocol. The handshake protocol defines how a client and server establish an SSL connection, including the negotiation of which cryptographic systems each host is willing to use for communication as well as the exchange of cryptographic material, such as public keys and session keys for encryption or authentication of transmitted data. The record protocol defines how communicating hosts exchange data using SSL, including specifications for how data is to be prepared for transmission and how it is to be verified or decrypted on receipt [1].

The following entities are required to establish a successful SSL connection between a Server and Client

1. Digital certificates: When a client connects to Server, the server presents the client with a digital certificate which is typically an x509 certificate. The client on receipt of the certificate, validates the server identity using public key encryption. Similarly, client also can optionally present a certificate to the server which gets validated at the server end.
2. KeyStore: KeyStore is an entity managed by Java keytool command, to provide the credentials such as certificates corresponding to public keys and private keys required for authenticating an SSL connection.
3. TrustStore: Truststore validates the credentials provided by the Server, through a third party Certifying Authorities such as Verisign / goDaddy/ GeoTrust.

How different entities above integrate, and communicate is something beyond the scope of this proposal. For establishing an SSL connection between two peers, we just need the following

1. Server.jks – This Keystore configuration file contains the certificates and cryptographic cipher suite information that the server shares with the client
2. Client.jks – This Keystore configuration file contains the certificates and cryptographic cipher suite information that the client shares with the server.
3. trustedCerts.jks – This Truststore configuration file contains the third party certificate information required to validate the incoming digital certificate
4. Key Store password – A Keystore password that is used to protect the store from updating certificates that are stored in its store.
5. Trust Store password – A Truststore password that is used to protect the store from updating certificates that are stored in its tore.
2.2 Hadoop RPC

Hadoop RPC is increasingly being used with data-center middle wares such as MapReduce, HDFS, and HBase because of its simplicity, productivity and high performance. Hadoop RPC is implemented using a model build for inter Process Communication in Hadoop. Hadoop IPC is a client/server system, which is defined at the org.apache.hadoop.ipc.Client and org.apache.hadoop.ipc.Server classes. A server process offers service to others by opening a socket and exposing one or more Java interfaces that remote callers can invoke. The server starts its life by calling RPC.getServer [2]. User server code must indicate the port number and an instance of an object that will receive remote calls. The following diagram represents the main components of org.apache.hadoop.ipc.Server class. A Server has a single listener (Listener), which listens for the incoming connections on the port. On a new Connection, the Server authenticates the connection using SASL if enabled, does basic validations and puts it in a Call Queue. The Call Queue is operated through a set of Handlers asynchronously, which invoke the corresponding Responder for serving the request. The Responder writes the serialized response onto the same connection.

A client contacts a server at a specified host and port, and invokes methods exposed by the server. The client starts its life through Client.call method which creates a new connection (or pull one from the connection pool) and begin an RPC handshake [3].

The Server and Client classes becomes the core of the inter process communication between different services in Hadoop. The serialized remote procedure calls get sent over the wire protocol defined in these classes.
2.3 SASL support in Hadoop RPC

HADOOP-6419 introduced Simple Authentication and Security Layer to Hadoop RPC. SASL is an authorization framework. It has many variants which are really different, and leaves details undefined. Today Hadoop supports two SASL variants, DIGEST-MD5 and GSSAPI. The DIGEST-MD5 will simply accept username/password login, while GSSAPI will use Kerberos server to authenticate the login [6]. SASL provides rpc encryption through symmetric keys.

3 Design

3.1 SSL Configuration

SSL can be switched on by setting the flag `ipc.secure-ipc.enable` to true in core-site.xml. SSL specific configuration defined in section 2.1 is read from `hadoop.security.sslclient.conf` and `hadoop.security.sslserver.conf` files which are already introduced for enabling HTTPS support.

3.2 SSL Implementation:

SSLEngine (javax.net.ssl.SSLEngine)[5] is sun’s oracle implementation of enabling secure communications for protocols such as SSL / TLS. It provides secure communication modes for Integrity protection, Authentication and Confidentiality between two different end points. It provides secure SSLSocket and SSLServerSocket alternatives to Socket and ServerSocket.

Hadoop IPC Server and Client classes, uses non-blocking SocketChannel and ServerSocketChannel throughout instead of blocking Socket and ServerSocket. Today JVM does not provide standard SSLSocketChannel and SSLServerSocketChannel classes that extend the base socket channel classes. Instead, SSL exchanges must be manually orchestrated using SSLEngine. niossl library [6] written by Corey Baswell and licensed under Apache 2.0, provides a non-blocking implementations for SSLSocketChannel and SSLServerSocketChannel that can be used like SSLSocket and SSLServerSocket.

3.2.1 Redesigning Server and Client classes

The current implementation of the Server and Client classes, have large chunks of SASL-specific code added. It becomes increasingly difficult for us to add SSL specific implementation on top of this. We propose to add some abstraction to the Server and Client classes so that the new SSL implementation can go in separate files. In future, we can pull SASL specific implementation from the existing Server and Client classes to be moved to separate files as SSL does. The below class diagrams Fig3 and Fig4, shows the abstraction desired to make Server and Client classes to easily extend SSL implementation.
Fig3: UML diagram of Server with the new abstractions

Fig3 shows the abstractions AbstractListener and AbstractConnection classes, which were introduced for segregating the common functionalities across different Listeners. The standard ServerSocket creation separated into Listener Class in a new file, and it inherits rest of the behavior from the parent AbstractListener class. Through this we can easily plugin SSLServerSocket implementation through SSLListener extending AbstractListener in a separate file. Similarly, Server.Connection class is abstracted, so that the subclasses can override the behavior of the core readAndProcess() method used for parsing requests on a connection. The factory classes ListenerFactory and ConnectionFactory can choose the right Listener and Connection based on the SSL specific configuration.
Like in Server class, possible abstractions are made in Client class to segregate the implementation of SSL Connection from plain socket connection.

3.2.2 Proposed Implementation

The patch submitted under Jira HADOOP-13836, implements SSLEngine using niossl library in Server.java. It uses the new abstracted interface of Server, and creates an SSLListener. The SSLListener creates a secure server socket channel that wraps over server socket channel. SSLServerSocketChannel needs an SSLContext, actual server socket channel, and a thread pool for executing long-running SSL tasks. For further details on niossl library usage refer to [6].

The patch uses javax.nio.ssl library to create SSLSocket in Client.java. It abstracts SSLSocket creation logic using the new abstraction to SSLConnection class.

References
5. https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/SSLEngine.html