INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 4, Issue 3)
Available online at: www.ijariit.com

Datagram transport layer security in Java

Rekha Kumawat
rekhakumawat19@gmail.com
Vivekanand Education Society's Institute of Technology, Mumbai, Maharashtra

ABSTRACT

This paper explains transport layer security implemented in Java 9 based on datagram protocol. Building secure software requires the use of a wide variety of security controls, at different layers of the application. Java 9 security enhancement support transport-independent and light-weight DTLS version 1.0 and 1.2. Datagram Transport Layer Security (DTLS) protocol is designed to construct "TLS over datagram" traffic that doesn't require or provide reliable or in-order delivery of data. DTLS is a datagram-compatible variant of TLS. The DTLS API provides communications privacy for datagram protocols that allow client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery through SSL Engine. DTLS are the best solution for encrypting and transmitting real-time data. The use of the DTLS API in Java provides access to the application data in each DTLS message.

Keywords: *Java, Security API, DTLS, DTLS implementation*

1. INTRODUCTION

The Java security APIs spans a wide range of areas. Transport security is a point-to-point security mechanism that can be used for authentication, message integrity, and confidentiality. The advantages of using transport-layer security include the following:

• It is relatively simple, well-understood, standard technology.
• It applies to both a message body and its attachments.

Java already has support for TLS which provides secure communication based on reliable transport layer such as TCP. But there is no support for secure communication over datagram transport layer such as UDP. TLS cannot be used over datagram transport layer because it cannot tolerate out of order packets and data loss. The reason that TLS cannot be used directly in datagram environments is simply that packets may be lost or reordered. TLS has no internal facilities to handle this kind of unreliability, and therefore TLS implementations break when rehosted on datagram transport. The purpose of DTLS is to make only the minimal changes to TLS required to fix this problem. Java 9 will add the support for DTLS (Datagram Transport Layer Security) which can work over datagram transport layer. This will largely help applications which use UDP. The JDK provides APIs and an implementation of the DTLS protocols that includes functionality for data encryption, message integrity, and server and client authentication. The new API should be transport-independent and similar to javax.net.ssl.SSLEngine.

It creates client and server on the same host in the same JVM and tries to exchange text messages between them via DatagramSocket, and then validates that exchange is done correctly. The implementation uses SSLEngine class to translate application data (text message) to network (encrypted) data and vice-versa. DatagramSocket class is used to send and receive the network data. A client-server handshake also should be done separately before app data exchange. The DTLS implementation should consume or produce at most one TLS record for each unwrap or wrap operation so that the record can be delivered in the datagram layer individually or can be reassembled more easily if the delivery is out of order. The DTLS API should provide access to the application data in each DTLS message.

2. DTLS HANDSHAKE

Before application data can be sent or received, the DTLS protocol requires a handshake to establish cryptographic parameters. This handshake requires a series of back-and-forth messages between the client and server by the SSLEngine object. Up to 15 handshake messages distributed over 6 flights are needed to establish a secure connection. The DTLS handshake includes the following stages:

1. Negotiating the cipher suite
2. (Optional) Authenticating the server's identity (optional)
3. Agreeing on encryption mechanisms

The DTLS handshake, shown in Figure 1, is nearly identical to that of TLS.

2.1 How it works

DTLS uses the cookie exchange technique. DTLS incorporates mechanisms to retransmit the packet. For instance, if the client sends a ClientHello, it expects to receive a HelloverifyRequest message from the server. If this doesn't happen, it means that either the ClientHello or the HelloVerifyRequest is lost, hence after a certain amount of time the client will resend the ClientHello. The purpose of the HelloVerifyRequest is though not related with the

retransmission mechanism that both client and server integrate. Its purpose is to avoid Denial of Service (DoS) attacks that in UDP are very easy to be performed through the exchange of a cookie by exchanging a cookie.

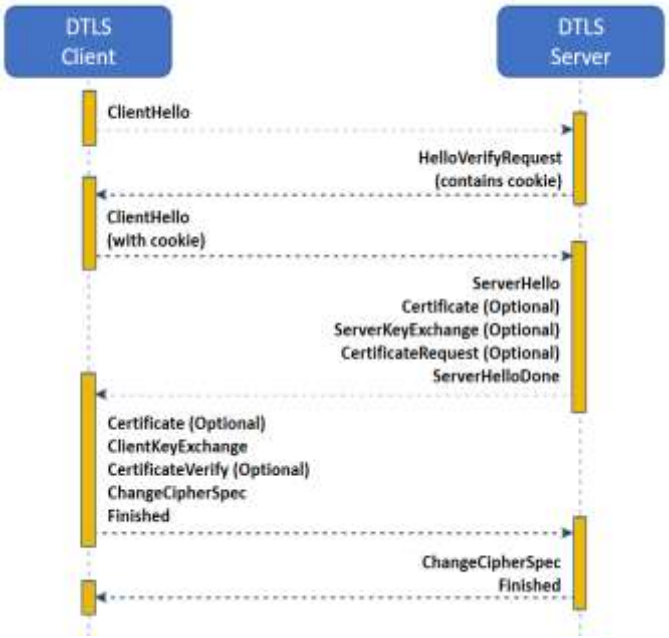

Fig. 1: DTLS Handshake

The final part of the handshake might seem similar to TLS but since the transport is not reliable, the last message could be lost. If, for example, the last flight is lost, the server believes that the handshake has been completed. Since the client is waiting for the server finished message, it will retransmit the last flight completing the handshake. This situation does not happen with TLS since there is no packet loss and needs to be handled with a retransmission timer in DTLS. The DTLS server might optionally send a HelloRequest to ask the client a renegotiation in the same way as it happens with TLS.

3. DTLS API

The API used for DTLS is mostly the same as for TLS, because of the mapping of generic functions to protocol-specific ones. A new API will likely be added to set the maximum application datagram size. If the size is not specified explicitly, however, then the DTLS implementation should adjust the size automatically. If a fragment is lost two or three times, the implementation may reduce the size of the maximum application datagram size until it is small enough.
Java API for Datagram Transport Layer Security (DTLS) version 1.0 and 1.2 introduced in JEP 219 through the java.net package.

Java.net

Java.net provides the classes for implementing networking applications.

Java.net has two major classes:

1. DatagramSocket: This class represents a socket for sending and receiving datagram packets.

Hierarchy:

java.lang.Object
- java.net.DatagramSocket

Class Definition :

public class DatagramSocket extends Object
implements Closable

2. DatagramPacket: This class represents a datagram packet. Datagram packets are used to implement a connectionless packet delivery service.

Hierarchy:

java.lang.Object
- java.net.DatagramPacket

public final class DatagramPacket
extends Object

4. IMPLEMENTATION

It creates client and server on the same host in the same JVM and tries to exchange text messages between them via DatagramSocket, and then validates that exchange is done correctly. The implementation uses SSLEngine class to translate application data (text message) to network (encrypted) data and vice-versa. DatagramSocket class is used to send and receive the network data. A client-server handshake also should be done separately before app data exchange.

.

There are three important constructs to use when adding DTLS support to your application using OpenSSL: SSL_CTX, SSL, and BIO. The SSL context (SSL_CTX) structure contains all of the information used to execute a DTLS handshake and create a session.

To go from zero to a configured and connected DTLS session, you must:

1. Create and configure a DTLS SSL_CTX.
2. Create a BIO from the SSL_CTX to use for I/O (sending and receiving packets).
3. Create an SSL instance from the DTLS BIO instance to start the DTLS connection and then interact with a peer(s).

4.1 Necessary changes of Java9, for DTLS support

SSLContext.getInstance ("DTLS"): A DTLS security provider is available. This can be provided for older Java versions as well.

- SSLEngineResult.HandshakeStatus.NEED_UNWRAP_AGAIN available next
- to SSLEngineResult.HandshakeStatus.NEED_UNWRAP
- engine.getSSLParameters().setMaximumPacketSize(n) didn't exist before.

4.2 DTLS Setup Code

The SSL context structure requires some credentials (a certificate and private key) in order to work. Following commands are used to setup DTLS connection:

```
openssl req -x509 -newkey rsa:2048 -days 3650 -nodes \
-keyout client-key.pem -out client-cert.pem
openssl req -x509 -newkey rsa:2048 -days 3650 -nodes \
-keyout server-key.pem -out server-cert.pem
```

This will create a client and server certificate and private key file. We'll use these to create the DTLS context. For convenience, we'll wrap up the DTLS relevant information into a struct that stores the context, BIO, and SSL instance:

```
Typedef struct{
SSL_CTX *ctx;
SSL *ssl;
BIO *bio;
}DTLSParams;
```


5. CONCLUSION

The Java 9 JDK provides new DTLS APIs and an implementation of the DTLS protocols that includes functionality for data encryption, message integrity, and server and client authentication. The goal of this paper is to present the implementation of the Datagram level security. Datagram Transport layer security support in Java enables developer an opportunity to insert datagram level security for applications which uses UDP. DTLS Handshake is the first stage for making the connection between peers. DatagramPacket and DatagramSocket are two major classes provided by the java.net package for implementing datagram level security. Datagram Transport Layer security is implemented using SSLEngine whose instance define protocol name and protocol version. Support of DTLS makes Java faster than TLS support.

6. REFERENCES

[1] Mastering Java 9 by Peter Verhas, Dr. Edward Lavieri.
[2] JEP 219: Datagram Transport Layer Security http://openjdk.java.net/jeps/219.
[3] Java Platform, Standard Edition What's New in Oracle JDK 9.
[4] DTLS support in JDK 9 Martin Toshev Prague, 19- 20 October 2017.
[5] Java Secure Socket Extension (JSSE) Reference Guide - Oracle Docs
[6] SSLEngine based DTLSConnector · Issue #421.
[7] Security enhancements in Java 9 - Distributed Computing in Java 9, By Raja Malleswara Rao Pattamsetti, June 2017
[8] Introduction to DTLS (Datagram Transport Layer Security) Pixelstech.net.
[9] Java 9 - Oracle Help Center.

