Computer Networks and Systems Security

Project Assignment #1

Overlayed secure "pay-per-view" real-time media streaming system supported by configurable TLS Channels

Abstract

The objective of this project assignment is the design, implementation and experimental assessment of an overlayed secure "pay-pre-view" real time streaming system, supported by configurable TLS channels.

For the purpose the main idea is to reuse initial protocols (SAPKDP and SRTSP) developed in Project Assignment #1, to create a secure enforcement for overlaying such protocols on top of TLS channels. The TLS channels can be supported over TCP or UDP, depending on the previous implementation objectives achieved in the Project Assignment #1: DTLS/UDP or TLS/TCP.

Keywords: TLS – Transport Layer Security, secure real-time streaming; symmetric and asymmetric cryptography, secure hashing and message authentication codes; password-based encryption; secure datagram-based connectionless communication channel; real-time FFMEG media playing

Development and delivery process:
Delivery process

- **Via a Form-Quiz (Google Form open after 20/Dec/2021 and closed on 30/Jan/2021**
- **The form includes a characterization of the implementation and concluded achievements**
- **URL of shard GitHub repo with the implementation included in the Form**
- **Penalizations considered for source code after 1/Jan/2022**

In the next sections of the document, you can find the reference and guidelines for the project implementation

1. Introduction

The goal of this project is to develop a solution to use SDKDP or SRTSP (or both) protocols as overlaying protocols on top of TLS (for the SDKDP) and DTLS (for the case of SRTSP).
The functionality initially provided for the components of the PA#1 solution must be maintained: Signaling Srver, Proxy, Stream Server, and the Media-Visualization Tool used for demonstration of real-time streamed movies in the Project Assignment #1. All the options for configurations provided for the Project Assignment #1 must be also supported after the implementation of the Project Assignment #2.

Depending on the implemented objectives in Project Assignment #1, students can implement the project in the following options:

- A) Use of configurable TLS channel only implemented for the SADKDP Protocol, involving the support for the Proxy and for the Signaling Server
- B) Use of configurable TLS channel and DTLS channel for the SRTSP protocol, involving the Proxy and the Streaming Server, if this protocol was initially implemented using TCP and UDP (TCP Sockets and Datagram Sockets), for different message flows of the protocol.
- C) Use of DTLS channel for SRTSP protocol, involving the Proxy and the Streaming Server, if this protocol was initially implemented using only UDP (or Datagram Sockets)
- D) TLS and DTLS enabled channels to overlay both protocols (SADKDP and SRTSP), if students decide to have a generic solution for both protocols, involving the Signaling Server, Proxy, and Streaming Server.

For each one of the above options, the following specifications and requirements must be addressed.

2. Options and Requirements

2.1 Option A: Configurable TLS channel for overlayed SADKDP

Beyond all the previous configurations required by the SADKDP protocol in Signalling and Proxy endpoints, as provided in the PA#1 implementation, students will add the support for the overlayed SADKDP on top of configurable TLS channels.
To address this, the solution can be implemented with JSSE Sockets [1]. Each endpoint of the SADKDP protocol must have a configuration files, as follows, allowing a generic configuration of the TLS behavior.

Signaling server and the Proxy must have the necessary keystores (managed with the keytool [2] or KeyStore Explorer tool [3] or openssl tool [4] to manage files for the Public Key Certificates and keystores for Private Keys. The implementation will use RSA-enabled configurations using keys with 2048 bits, in order to support the behavior of a TLS configuration file in the following format (in a text-based file)

```
File tls.conf

authentication: conf    // MUTUAL, PROXY-ONLY  SIGNALING-ONLY
tlsversion: version      // tls1.1, tls1.2, tls1,3
ciphersuites: list of supported ciphersuites
```

Note that a file tls.conf must be configured in a consistent way, for the correct operation and demonstration of the Signaling Server and Proxy. You must investigate the support for the TLS behavior based on the provided configuration and the configuration of ciphersuites to be supported and enabled and how those cipersuites operate.

For the implementation, see also references and lab materials and examples [5] and explanations or clarifications in Lab Classes. For your experimental observations you can use also the Wireshark tool [6]

2.2 Option B: Configurable TLS and DTLS channels for overlayed SRTSP

For this implementation option you can use the support for the use of the Java JSSE package to support TLS (for the TCP message flows) and DTLS (for the UDP message Flows).

Streaing server and the Proxy must have the necessary keystores (managed with the keytool [2] or KeyStore Explorer tool [3] or openssl tool [4] to manage files for the Public Key Certificates and keystores for Private Keys. The implementation will use RSA-enabled configurations using keys with 2048 bits, in order to support the behavior of a TLS configuration file in the following format (in a text-based file)

```
File tls.conf

authentication: conf    // STREAMSERER-ONLY
tlsversion: version      // tls1.1, tls1.2, tls1,3
ciphersuites: list of supported ciphersuites
```

Note that a file tls.conf must be configured in a consistent way, for the correct operation and demonstration of the Streaing Server and Proxy. You must also note that in this case we wil require only the use of a TLS behavior using server-side authentication only (Streaming Server). You must investigate the support for the TLS behavior based on the provided configuration and the configuration of ciphersuites to be supported and enabled and how those cipersuites operate. The configuration file may be the same for the TLS and DTLS support.

Streaming server and the Proxy must have the necessary keystores (managed with the keytool [2] or KeyStore Explorer tool [3] or openssl tool [4] to manage files for the Public Key Certificates and keystores for Private Keys (in this case only used in the Streaming Server side). The implementation will use RSA-enabled configurations using keys with 2048 bits, in order to support the behavior of a TLS configuration file in the following format (in a text-based file)

For the implementation, see also references and lab materials and examples [5] and explanations or clarifications in Lab Classes. For your experimental observations you can use also the Wireshark tool [6]

2.3 Option C: Configurable DTLS channel for overlayed SRTSP

This is a particular case of Option B (see above), where you will only use DTLS in the case of that the support you have for SRTSP from the Project Assignment #1 is all supported by Datagran Sockets and UDP.

2.4 Option D: Configurable TLS and DTLS channels for overlayed SADKDP and SRTSP

This is a particular case of all the options above (see above), if you plan to implement a solution inlcuing TLS channels and DTLS channels supporting both overlayed protocols SADKDP and SRTSP.

2.5 Option for PA#1 implementations with a complete implementation of SAPKDP and SRTSP both supported by UDP

In this case the PA#2 must be addressed in the following way:

- DTLS/UDP support for overlayed SRTSP must support the following configuration (in configuration files in the Streaming Server side and in the Proxy Side for the DTLS interaction between Proxy and Streaming Server

```
File tls.conf

authentication: conf    // STREAMSERER-ONLY
tlsversion: version      // tls1.1, tls1.2, tls1,3
ciphersuites: list of supported ciphersuites
```

- DTLS/UDP support for overlayed SAPKDP must support the following configuration (in configuration files in the Signaling Serer side and in the Proxy Side for the DTLS interaction between Proxy and Signaling Server

```
File tls.conf

authentication: conf    // MUTUAL, PROXY-ONLY, SIGANLING-ONLY
tlsversion: version      // tls1.1, tls1.2, tls1,3
ciphersuites: list of supported ciphersuites
```

3. Evaluation Criteria

CRITERIA-1: Option C with the provided configurations and correct operation and demonstration, including the quality of the video stream visualization with the used media tool: 13 points

CRITERIA-2: Options A, B, or C with the provided configurations and correct operation and demonstration, including the quality of the video stream visualization with the used media tool: 15 points

CRITERIA-3: Robustness of the solution, applicable for the options A, B or C: 1 point

CRITERIA-4: Modularity of the solution, applicable for the options A, B or C 1 point

CRITERIA-5: Use of Certification Chains (with at least two certificates and only one root-certificate used in trusted stores (or trusted keystores) in the the TLS or DRLS endpoints: 1 point

CRITERIA-5: Option D with the provided configurations and correct operation and demonstration, including the quality of the video stream visualization with the used media tool: 17 points

References

[1] Java Secure Socket Extension (JSSE) Reference Guide (use the documentation according to your Java version used for the development f your Project Assignment #2). Please note that if you have Java 8 – Jaba 10 you ptobably need to search for an update release including support for TLS 1.3. After Java 11 the support for TLS 1.3 was included in the first release.
Ex: For Java 10:
https://docs.oracle.com/javase/10/security/java-secure-socket-extension-jsse-reference-guide.htm#JSSEC-GUID-F069F4ED-DF2C-4B3B-90FB-F89E700CF21A
Ex: for Java 8:
https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html (for Java 8)

[2] https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html (please see th documenttaion for your own Java version, however the keytool use and specification didn't change significantly in different Java vesrions)

[3] https://keystore-explorer.org/

[4] https://www.openssl.org/

[5] SRSC Lab Materials on the following topics and contents:

Lab 6 – Practice with X509 Certificates

Lab 7 – TLS: Analysis, Java Programming with JSSE and Development Tools

Lab 8 – TLS Traffic analysis

[6] Wireshark Tool: https://www.wireshark.org/