Ano Letivo 2020/2021

**Segurança de Redes e Sistemas de Computadores**

***A DTLS-Enabled Live Streaming Service***

Relatório de Projeto

Alexandre Correia (53298) e Pedro Ferreira (52370)

Grupo 4

## Introdução

Neste projeto é implementada a opção A (*A DTLS-Enabled Live Streaming Service*) que consiste em efetuar uma re-implementação do protocolo de segurança, usando DTLS, em relação ao projeto 1 (*SECURE STREAMING PROTOCOL*), que adiciona segurança ao streaming de vídeos.

Para o efeito, é utilizado o protocolo DTLS (*Datagram Transport Layer Security*), que se baseia no protocolo TLS (*Transport Layer Security*), mas que não tem uma implementação direta e pronta a usar. Por este motivo, explica-se neste relatório a implementação seguida.

## Handshake do DTLS

Antes da transferência de qualquer tipo de dados relativos ao streaming do vídeo é necessário efetuar o *handshake* entre o servidor e o proxy, que também permite autenticar o servidor e/ou proxy.

Para se realizar o *handshake*, é necessário indicar primeiro ao *SSLEngine* os modos de operação e autenticação do servidor e do proxy (*SERVER*, *MUTUAL* ou *PROXY*), o protocolo a usar, assim como as *cipher suites* que o proxy suporta.

Para a realização do *handshake* é necessário usar o *SSLEngine*, que negoceia as *cipher suites* e chaves a usar durante a sessão. O *SSLEngine* gera e consome todas as mensagens relativas ao processo do *handshake* (que está implementado na classe *DTLSHandshake*).

## SSPHeader sobre DTLS

Tendo os parâmetros necessários para a sessão DTLS definidos e olhando para a especificação do *SSP* do projeto 1, repara-se que há um *SSPHeader*.

Este *header*, usando o DTLS, não é necessário pois este segundo protocolo transporta as mensagens relativas ao conteúdo do vídeo no sub-protocolo do TLS designado de *SSL Record Protocol*, que já contém as informações equivalentes ao *SSPHeader*, como se pode verificar na imagem ao lado.

## SSPPayload sobre DTLS

Analisa-se agora a outra parte da especificação do *SSP* do projeto 1 - o *SSPPayload* - que está ilustrado na seguinte imagem:



Neste protocolo vêm-se dois componentes de segurança:

* MAC1: garantir a integridade e a autenticidade dos dados;
* MAC2: previne ataques de DoS - *Denial of Service*.

No caso do DTLS/TLS, quando se cifra uma mensagem, esta é cifrada junto com o MAC da mesma (tal como ilustrado na figura apresentada na secção “SSPHeader sobre DTLS”), garantindo assim que o efeito do MAC1 do projeto 1 é mantido.

A componente de segurança representada pelo MAC2 do projeto 1 não foi adaptada pois o TLS não o suporta. Esta componente é parcialmente suportada com o uso do MAC que é cifrado juntamente com a mensagem a enviar, o que ajuda a prevenir os ataques de DoS. Contudo não é o mais eficiente.

## SHP sobre DTLS

O *SHP* do projeto 1 negoceia as *cipher suites* a utilizar na sessão e garante também que o utilizador tem acesso ao vídeo.

No *handshake* do DTLS os protocolos e *cipher suites* são negociados, ficando apenas a faltar garantir que o utilizador tem acesso ao vídeo, sendo que para este efeito é só necessário implementar a *Password Based-Encryption* (PBE).

Este protocolo é executado logo após o *handshake*.

## Capturas do WireShark

De modo a se confirmar o correto funcionamento do projeto, mostram-se algumas capturas do WireShark nos diferentes modos de autenticação: SERVER (**Fig. 1.**), PROXY (**Fig. 2**.) e MUTUAL (**Fig. 3a e b**).

Nestas capturas dá-se ênfase à direção nas quais os certificados circulam:

servidor -> proxy proxy -> servidor

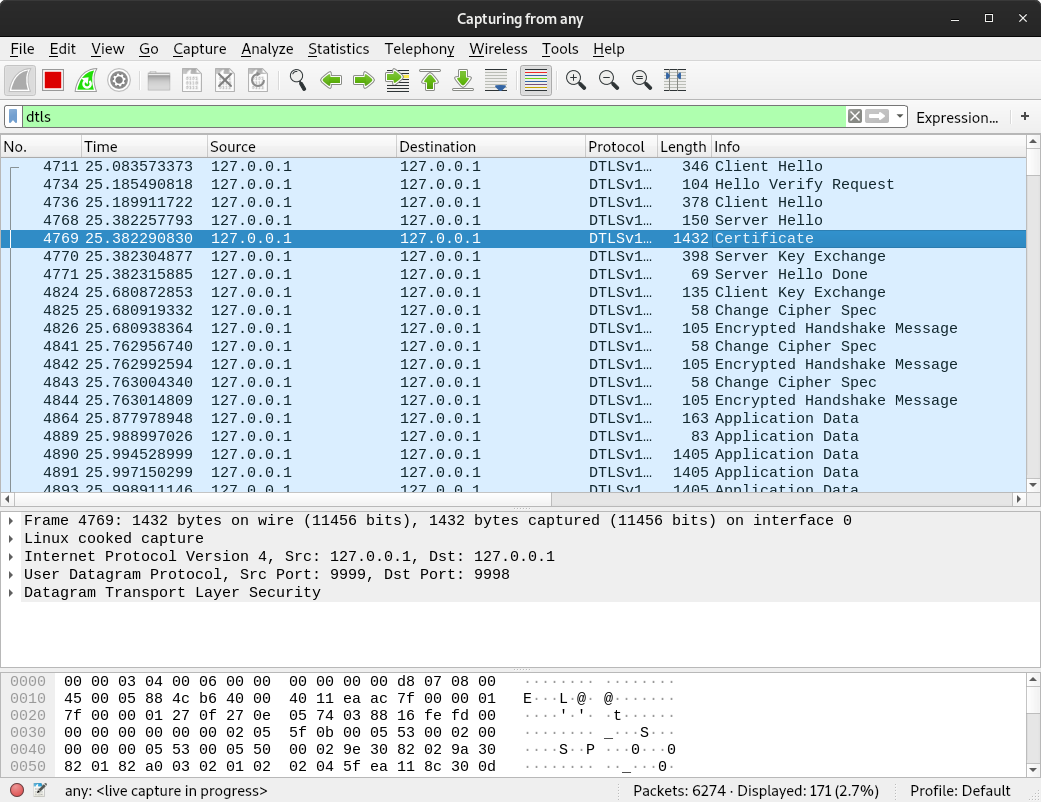


Fig. 1 - SERVER

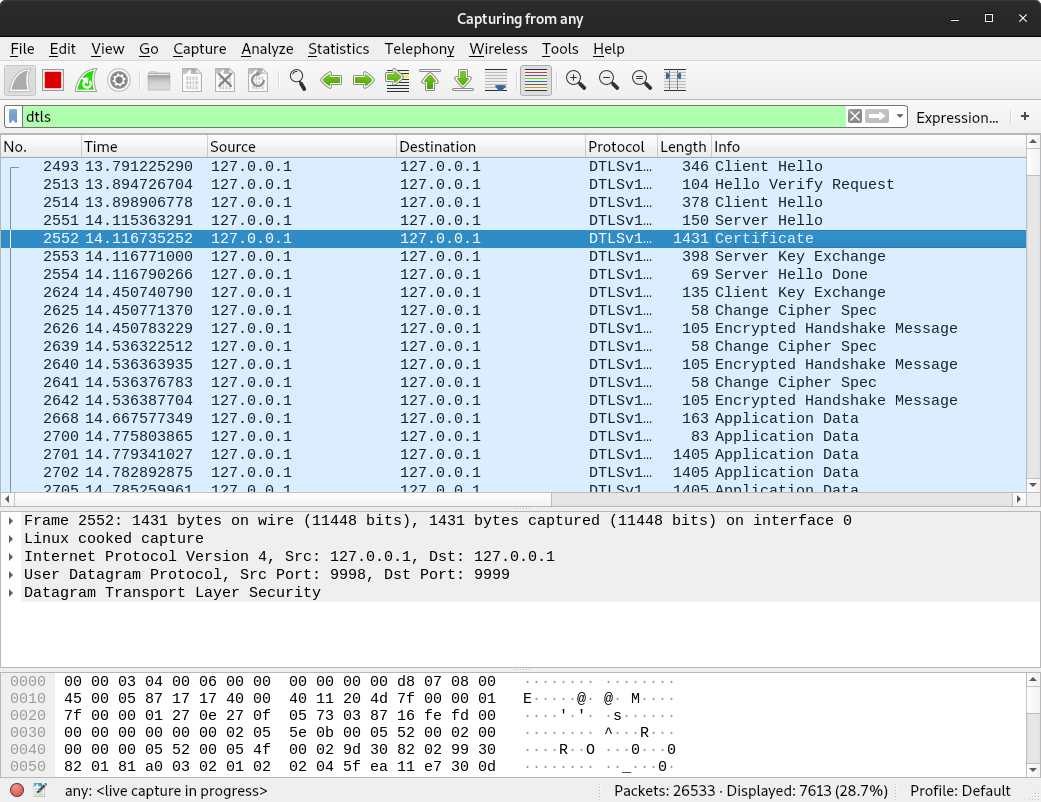


Fig. 2 - PROXY

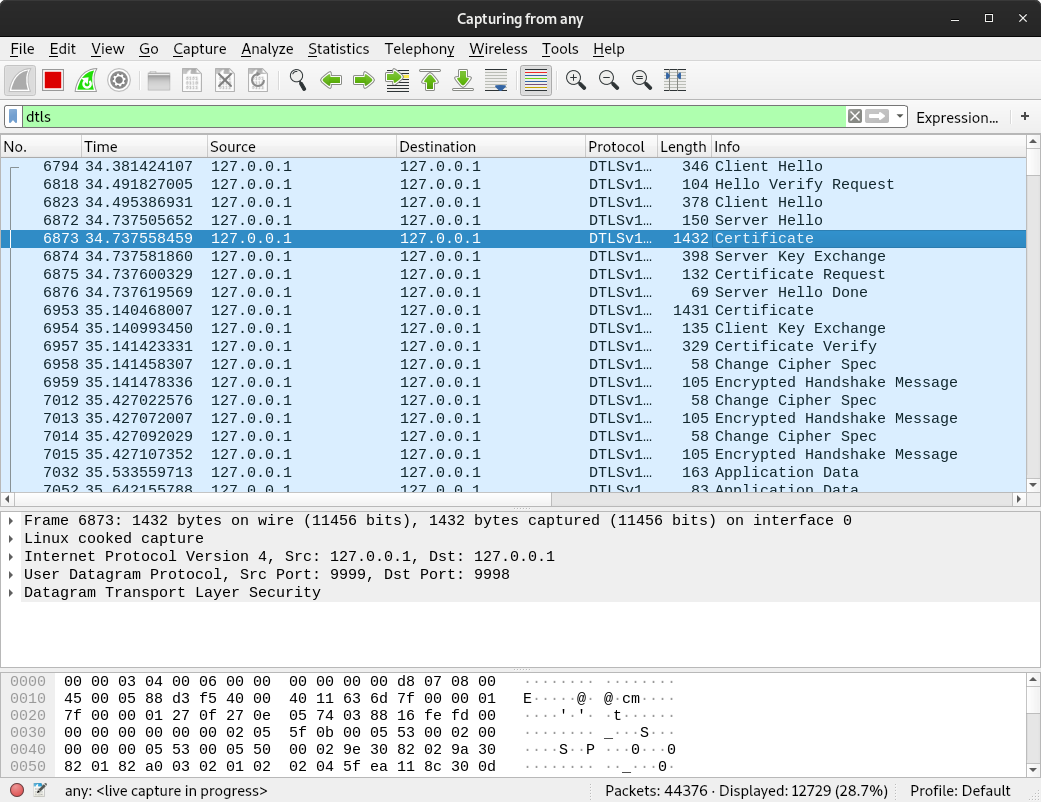


Fig. 3a - MUTUAL (certificado do servidor)

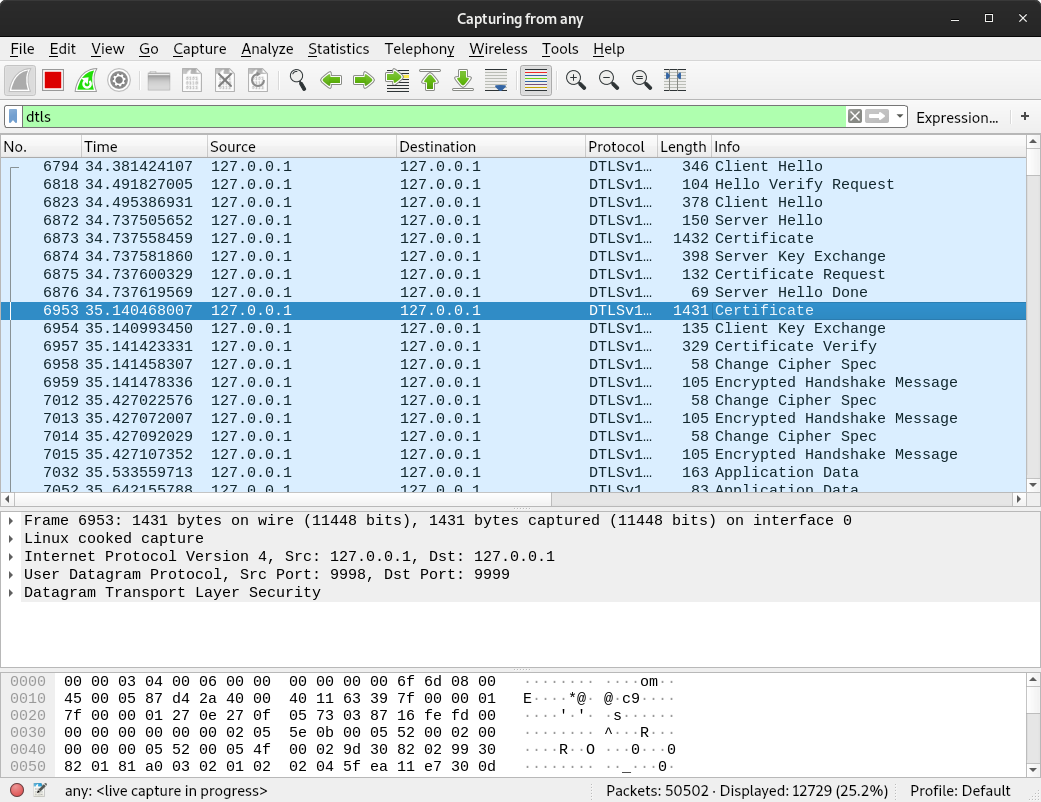


Fig. 3b - MUTUAL (certificado do proxy)