Trabalho Final Seg Info

Diego V. S. de Matos – 120098723

November 29, 2025

1 Protocolo TLS

O TLS é projetado para usar o TCP para prover um serviço seguro, é caracterizado por ter duas camadas de protocolos como descrito abaixo.

O Protocolo de Registro (*Record Protocol*) fornece os serviços básicos de segurança para os protocolos da camada superior:

- Confidencialidade: via criptografia simétrica
- Integridade: via hash criptográfico
- Autenticação: via criptografia de chave pública

Os Protocolos Superiores (Gerenciamento TLS) é responsável pelo Handshake, que é definido em 4 passos:

1. Inicialização:
   - *client_hello*:
     - cliente envia a versão do TLS compatível
     - uma sequência inicial aleatória de bytes
     - Id de sessão
     - indica conjuntos de cifras que ele suporta
     - Lista de métodos de compressão métodos
   - *server_hello*:
     - servidor responde com os mesmos parâmetros do *client_hello*
     - a cifra escolhida
     - o método de compressão escolhido
2. Autenticação do Servidor: Opcionalmente servidor envia o seu certificado para o cliente autenticar e informações da chave (*ServerKeyExchange*), termina com *server_done*
   - Opcionalmente o servidor solicita um certificado do cliente (*CertificateRequest*)

3. Resposta do Cliente: Caso necessário, o cliente verifica se o certificado do servidor foi emitido por uma autoridade de certificação confiável, se é válido e não revogado e se o domínio é o mesmo. Caso tudo estiver,

4. Finalização: O cliente envia uma mensagem de *change_cipher_spec* e imediatamente envia mensagem *finished* que verifica se a troca de chaves e a autenticação foram bem-sucedidas. O servidor também responde com *change_cipher_spec* e com sua própria mensagem de *finished*.

   O handshake está completo e o cliente e servidor podem começara trocar dados da camada de aplicação de forma segura.

2 Comparação de Implementações

Foi implementado em Python dois programas cliente e servidor para cada caso: com TLS e sem TLS. As rotinas para cada caso são parecidas e suas diferenças principais são:

1) Inicialização dos programas: No caso com TLS o servidor deve carregar os certificado criar seu socket normalmente

Listing 1: Inicializacao do Servidor

```
# Rotina abaixo apenas para o caso com TLS
context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context.load_cert_chain(certfile="cert.pem", keyfile="key.pem")

# Rotina abaixo em comum para ambos os casos
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((HOST, PORT))
server.listen(1)
```

Listing 2: Inicializacao do Cliente

```
# Apenas para o caso com TLS
context = ssl.create_default_context()
context.check_hostname = False
# Para nao verificar o certificado
context.verify_mode = ssl.CERT_NONE
```

2) Em Python, ao importar o pacote *ssl*, temos acesso a uma API para fazer um envólucro na conexão do socket, dessa forma facilitando o envio da mensagem da aplicação pois é feita a encriptação e decriptação de forma automática tanto no cliente como no servidor. Passo não necessário no caso sem TLS.

Listing 3: Conexão do servidor

```
conn, addr = server.accept()
# Conexao segura com TLS usando envolucro python
secure_conn = context.wrap_socket(conn, server_side=True)
```

Listing 4: Conexão do cliente

```
raw_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Conexao segura com TLS usando envolucro python
secure_sock = context.wrap_socket(raw_sock, server_hostname=HOST)
```

3 Captura dos Pacotes

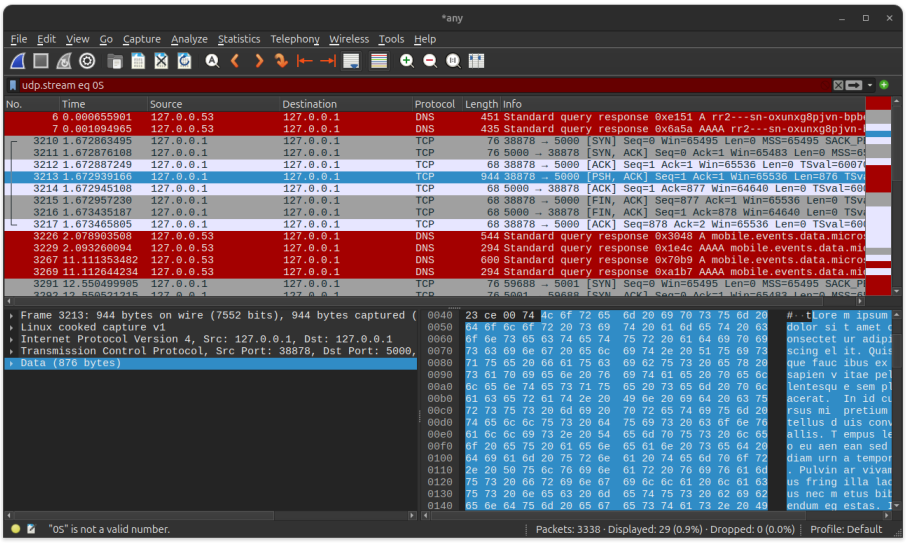

Figure 1: Pacotes enviados sem tls

Na figura 1 conseguimos analisar os pacotes para o caso sem TLS. Observamos uma sequência de pacotes padrão do protocolo TCP (SYN, ACK) entre o cliente e servidor em localhost (127.0.0.1) nas portas 38879 → 5000 seguido imediatamente do conteúdo da aplicação e está em *plain text*.

Já para o caso com TLS observamos na figura 2 a conexão entre o cliente e servidor em localhost (127.0.0.1) nas portas 59699 → 5001 respectivamente. Os três primeiros pacotes são referentes ao TCP, o quê já era esperado pois o TLS é executado em cima do TCP. É seguido dos pacotes de *Client Hello*, *Server Hello* e *Change Chipher Spec*, mostrando que de fato o handshake do TLS funcionou como esperado. Como o meu programa está rodando em localhost estou pulando o passo de verificar o certificado, dessa forma o próximo pacote já é o conteúdo cifrado.

O wireshark não capturou os pacotes de fim de conexão, acredito que seja por quê eu matei o processo diretamente.

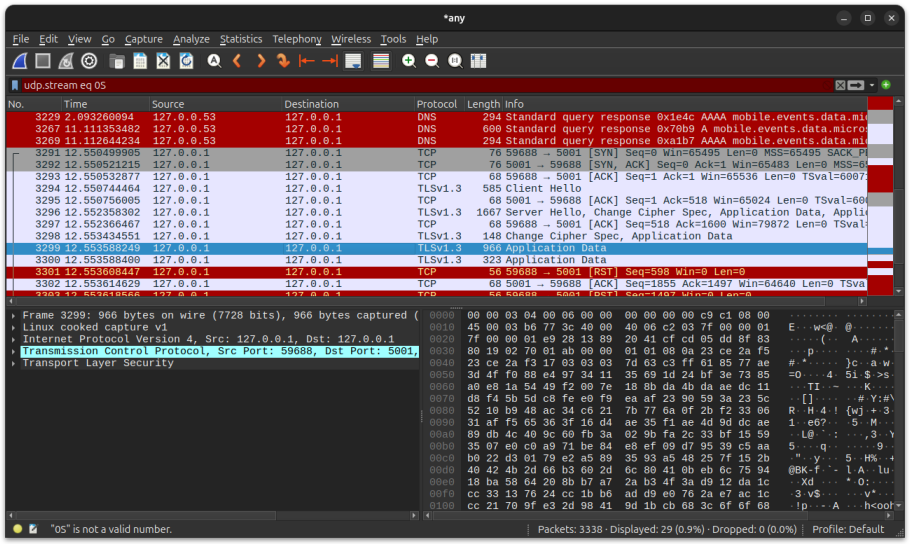

Figure 2: Pacotes enviados com tls