MiniTLS
Sprawozdanie wstępne

Sofiya Yedzeika
Matvii Ivashchenko
Katsyaryna Anikevich

1 Cel projektu

Celem projektu jest zaprojektowanie i implementacja prostego, szyfrowanego protokołu komunikacyjnego typu klient–serwer, opartego na protokole TCP (tzw. mini TLS), umożliwiającego bezpieczną wymianę wiadomości z wykorzystaniem mechanizmu wymiany kluczy oraz szyfrowania danych.

2 Założenia projektu

Projekt realizuje wariant W1, wykorzystujący mechanizm encrypt–then–MAC jako sposób zapewnienia integralności i autentyczności wiadomości.

- Komunikacja oparta jest na protokole TCP w architekturze klient–serwer.
- Interakcja z klientem oraz serwerem odbywa się za pośrednictwem wiersza poleceń.
- Komunikacja realizowana jest w sieci dockerowej.

3 Struktura wiadomości protokołu

Protokół wykorzystuje trzy podstawowe typy wiadomości: ClientHello, ServerHello oraz wiadomości szyfrowane (EncryptedData). Zakończenie sesji (EndSession) jest kodowane wewnątrz zaszyfrowanego ładunku wiadomości EncryptedData, dzięki czemu nie jest rozróżnialne przed odszyfrowaniem.
W wariancie encrypt–then–MAC nadawca najpierw szyfruje ładunek wiadomości, a następnie oblicza MAC z szyfrogramu i dołącza go do wiadomości. Odbiorca weryfikuje MAC przed odszyfrowaniem.

3.1 ClientHello

Wiadomość ClientHello jest wysyłana przez klienta w celu zainicjowania połączenia oraz rozpoczęcia wymiany kluczy. Wiadomość nie jest szyfrowana.

- **Type** – identyfikator wiadomości (ClientHello)
- **ClientPublicKey** – klucz publiczny klienta używany w algorytmie wymiany kluczy (np. Diffie–Hellman)

3.2 ServerHello

Wiadomość ServerHello stanowi odpowiedź serwera na ClientHello i kończy etap wymiany kluczy. Wiadomość nie jest szyfrowana.

- **Type** – identyfikator wiadomości (ServerHello)
- **ServerPublicKey** – klucz publiczny serwera używany w algorytmie wymiany kluczy

Po wymianie wiadomości ClientHello oraz ServerHello klient i serwer obliczają wspólny klucz sesyjny.

3.3 Wiadomości szyfrowane

Wszystkie kolejne wiadomości przesyłane po ustanowieniu sesji są szyfrowane przy użyciu wspólnego klucza sesyjnego.

- **Type** – identyfikator wiadomości (EncryptedData)
- **EncryptedPayload** – zaszyfrowana treść wiadomości (np. tekst wysyłany przez klienta); wewnątrz ładunku znajduje się również informacja, czy jest to wiadomość danych, czy sygnał zakończenia sesji (EndSession)
- **MAC** – kod uwierzytelniający (np. HMAC) obliczony z użyciem klucza sesyjnego na podstawie szyfrogramu (encrypt–then–MAC)

3.4 EndSession

Zakończenie sesji jest przesyłane jako zaszyfrowana wiadomość typu EncryptedData, w której odszyfrowany ładunek zawiera informację EndSession. Dzięki temu wiadomości danych oraz zakończenie sesji nie są rozróżnialne przed odszyfrowaniem. Po odebraniu wiadomości EncryptedData i po poprawnej weryfikacji MAC, odbiorca odszyfrowuje ładunek. Jeśli w ładunku znajduje się EndSession, połączenie zostaje zakończone, a ponowne nawiązanie komunikacji wymaga ponownej wymiany wiadomości ClientHello i ServerHello.

4 Wykorzystane algorytmy

4.1 Wymiana kluczy

Do wymiany kluczy pomiędzy klientem i serwerem wykorzystano algorytm Diffie–Hellman. Algorytm ten umożliwia dwóm stronom uzgodnienie wspólnego klucza sesyjnego przy użyciu niezabezpieczonego kanału komunikacyjnego.
Klient oraz serwer generują własne wartości prywatne, na podstawie których obliczane są klucze publiczne przesyłane w wiadomościach ClientHello oraz ServerHello. Po odebraniu klucza publicznego drugiej strony obie strony niezależnie obliczają ten sam klucz sesyjny, który nie jest przesyłany bezpośrednio przez sieć.

4.2 Szyfrowanie wiadomości

Do szyfrowania wiadomości wykorzystano prosty mechanizm symetryczny oparty o wspólny klucz sesyjny uzyskany w procesie wymiany kluczy. W projekcie będzie zastosowano szyfrowanie typu one-time pad (OTP), polegające na operacji XOR pomiędzy treścią wiadomości a kluczem szyfrującym.
Szyfrowanie i deszyfrowanie realizowane są tą samą operacją XOR, co upraszcza implementację.

4.3 Przykładowy scenariusz użycia

Przykładowy przebieg komunikacji pomiędzy klientem a serwerem wygląda następująco:

1. Klient wysyła do serwera wiadomość ClientHello zawierającą swój klucz publiczny.
2. Serwer odpowiada wiadomością ServerHello zawierającą własny klucz publiczny.
3. Klient oraz serwer obliczają wspólny klucz sesyjny na podstawie algorytmu Diffie–Hellman.
4. Klient wysyła do serwera zaszyfrowaną wiadomość typu EncryptedData.
5. Serwer weryfikuje MAC, odszyfrowuje wiadomość przy użyciu klucza sesyjnego i wyświetla jej treść.
6. Jedna ze stron wysyła zaszyfrowaną wiadomość typu EncryptedData, której ładunek po odszyfrowaniu oznacza EndSession, co kończy sesję komunikacyjną.

5 Integralność i autentyczność wiadomości

W celu zapewnienia integralności oraz autentyczności szyfrowanych wiadomości zastosowano mechanizm typu encrypt–then–MAC. W projekcie do obliczania kodu MAC wykorzystano mechanizm HMAC oparty o funkcję skrótu (np. SHA-256) oraz wspólny klucz sesyjny uzyskany w procesie wymiany kluczy.

Dla każdej wiadomości nadawca najpierw szyfruje treść wiadomości, otrzymując szyfrogram (EncryptedPayload). Następnie obliczana jest wartość MAC (Message Authentication Code) na podstawie **szyfrogramu** oraz wspólnego klucza sesyjnego i dołączana do wiadomości (encrypt–then–MAC).

Odbiorca po odebraniu wiadomości najpierw oblicza wartość MAC z otrzymanego szyfrogramu i porównuje ją z wartością dołączoną. Dopiero w przypadku zgodności wykonuje odszyfrowanie wiadomości i interpretuje jej treść.

W przypadku niezgodności wartości MAC wiadomość zostaje odrzucona. Zastosowany mechanizm spełnia wymagania integralności oraz autentyczności dla przesyłanych danych.