Programowanie Sieciowe - Projekt

Dokumentacja wstępna

Zespół 32: **Katarzyna Kanicka**, Jan Mizera, Andrii-Stepan Pryimak

Cel projektu

Celem projektu jest zaprojektowanie oraz implementacja szyfrowanego protokołu opartego na protokole TCP, tzw. mini TLS.

Założenia

- Architektura klient-serwer
- Serwer jest w stanie obsłużyć kilku klientów równocześnie - maksymalna ich ilość zostaje podana jako parametr uruchomienia
- Klient inicjuje połączenie z serwerem poprzez wysłanie wiadomości *ClientHello*, na którą serwer odpowiada wiadomością *ServerHello*
- Wiadomości *ClientHello* oraz *ServerHello* służą do wymiany kluczy szyfrujących, nie są one szyfrowane
- Sesja może zostać zakończona zarówno przez klienta jak i serwer, poprzez wysłanie wiadomości *EndSession*
- Wszystkie wiadomości między klientem a serwerem, w tym *EndSession*, są szyfrowane

Projekt zostanie zrealizowany zgodnie z wariantem W1. Do zapewnienia integralności i autentyczności zaszyfrowanych wiadomości zostanie wykorzystany mechanizm encrypt-then-mac. Protokół zostanie zaimplementowany w Pythonie.

Struktura wiadomości

Każda wiadomość poprzedzona jest nagłówkiem *header*, który identyfikuje jej typ. Dzięki temu odbiorca jest w stanie rozpoznać strukturę wiadomości i ustalić jej długość.
Kod MAC będzie generowany przez HMAC, z funkcją haszującą SHA-256.

ClientHello

header [0x01] - 1 B	module p - 4 B	base g - 4 B	client public key - 4 B

Klient za pomocą *ClientHello* będzie inicjować połączenie serwerem, przesyłając moduł p, podstawę g i swój klucz. Na podstawie tych danych będzie możliwa wymiana kluczy między klientem i serwerem (algorytm Diffiego-Hellmana). Same dane p i g będą zwykłymi int32 dla uproszczenia.

ServerHello

header [0x02] - 1 B	server public key - 4 B

Serwer, po otrzymaniu wiadomości od klienta, oblicza swój klucz i dokańcza wymianę, wysyłając *ServerHello*..

EndSession

header [0x04] - 1 B	length - 4 B	msg. type - 1B	message - [length-1] B	MAC - 32 B

W celu zakończenia sesji, wysłany zostanie zaszyfrowana wiadomość z msg.type 0x02. Struktura tej wiadomości jest taka sama jak w pozostałych nie da się rozpoznać że to jest koniec sesji bez odszyfrowania. Pole MAC jest weryfikowane przez odbiorcę.

Pozostałe wiadomości

header [0x04] - 1 B	length - 4 B	msg. type - 1B	message - [length-1] B	MAC - 32 B

Reszta wymienianych przez klienta i serwer komunikatów będzie zawierać msg.type 0x01. Oczytana wiadomość jest odszyfrowana dopiero po weryfikacji MAC.

header [0x04] - wiadomość zaszyfrowana
msg. type : 0x01 - Zwykła wiadomość (Dane)
msg. type : 0x02 - Zakończ sesję (EndSession)

Aby uniemożliwić analizę ruchy i ukryć moment zakończenia sesji, wszystkie wiadomości przesyłane po etapie Handshake mają identyczną strukturę zewnętrzną. Rzeczywisty typ operacji jest zawarty wewnątrz zaszyfrowanego bloku danych

Wykorzystane algorytmy

Wymiana kluczy

Algorytm Diffiego-Hellmana - Po tym jak uzgodnione zostaną dwa publiczne parametry: liczba pierwsza p oraz podstawa g, obydwie strony losują swoje tajne klucze a i b. Każda ze stron oblicza odpowiednio swój klucz publiczny ($A = g^a \pmod p$ i $B = g^b \pmod p$). Po wymianie kluczy publicznych każda ze stron jest w stanie obliczyć ten sam klucz sesji ($K = B^a \pmod p = A^b \pmod p$).

Szyfrowanie

One Time Pad (OTP) - Szyfrowanie i deszyfrowanie poprzez operacją bitową XOR między wiadomością a kluczem. Klucz jest jednorazowy i jego długość jest równa długości wiadomości.

Przykładowe użycie

1. Klient generuje parametry p, g i klucz publiczny. Wysyła te dane w wiadomości ClientHello.
2. Serwer odbiera parametry oraz klucz publiczny klienta. Na tej podstawie generuje swój klucz publiczny i prywatny. Wysyła swój klucz publiczny do klienta w wiadomości ServerHello.
3. Obie strony obliczają klucz sesji K.
4. Wysyłanie szyfrowanych wiadomości:
    a. Klient szyfruje wiadomość algorytmem OTP wykorzystując K jako ziarno generatora pseudolosowego oraz numer wiadomości w sesji, zapewniając unikalność klucza.
    b. Klient oblicza kod MAC zaszyfrowanej wiadomości.
    c. Klient wysyła długość wiadomości, zaszyfrowaną wiadomość i kod MAC w wiadomości z nagłówkiem 0x04.
5. Serwer odczytuje długość, weryfikuje kod MAC i deszyfruje wiadomość.
6. Jedna ze stron wysyła wiadomość EndSession z kodem MAC kończąc tym samym połączenie.

Realizacja integralności i autentyczności

Będzie ona realizowana za pomocą mechanizmu **Encrypt-then-MAC**. Najpierw wiadomość zostaje zaszyfrowana, w naszym przypadku za pomocą OTP. Następnie na podstawie otrzymanego szyfrogramu oraz klucza sesyjnego obliczany zostaje kod MAC. Obie operacje są wykonywane sekwencyjnie i niezależnie od siebie.

Odbiorca wiadomości na początku weryfikuje kod MAC, co zapewnia integralność i autentyczność. Jeśli kod MAC okaże się niepoprawny, oznacza to, że treść wiadomości została zmieniona w trakcie przesyłania lub że klucz sesyjny użyty do weryfikacji jest niezgodny.