**Bezpieczna komunikacja w OpenSSL**

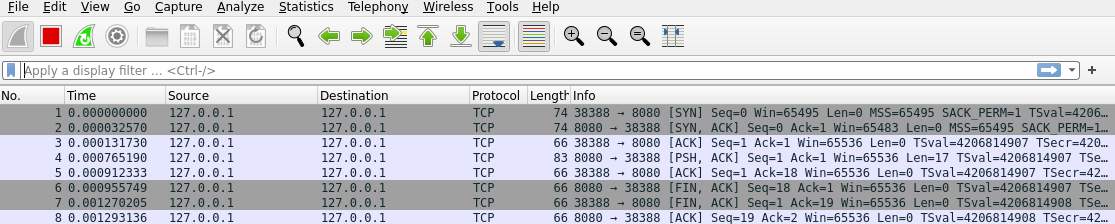
**Laboratorium 4**

**Wiktor Zmiendak**

**ZADANIE 1**

****

****

****

Po uruchomieniu serwera, który nasłuchuje na wskazanym porcie, a następnie klienta, który nawiązuje połączenie w celu monitorowania ruchu użyto sniffera (Wireshark), aby przechwycić i przeanalizować pakiety TCP.

Na podstawie przechwyconych pakietów w Wiresharku można zauważyć typowy przebieg połączenia TCP:

1. Trójfazowe uzgadnianie (3-way handshake): pakiety SYN, SYN-ACK, ACK.
2. Wymiana danych: serwer wysyła wiadomość do klienta.
3. Zamykanie połączenia: FIN-ACK.

Uzyskano poprawny przebieg komunikacji TCP, co potwierdza poprawność działania zarówno serwera, klienta, jak i konfiguracji sniffera.

**ZADANIE 2**

**Obraz zawierający tekst, Czcionka, oprogramowanie, zrzut ekranu

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

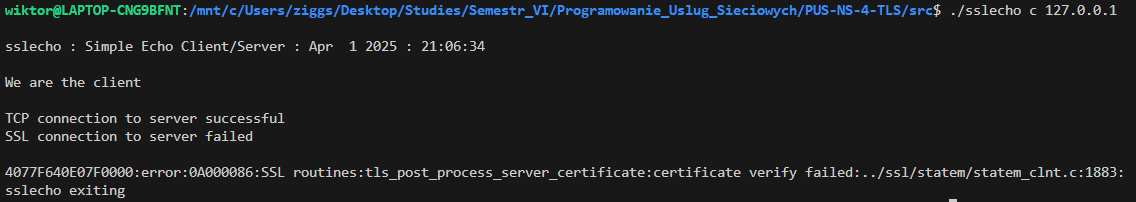
**Obraz zawierający tekst, oprogramowanie, Czcionka, zrzut ekranu

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

**Obraz zawierający tekst, oprogramowanie, Strona internetowa, Czcionka

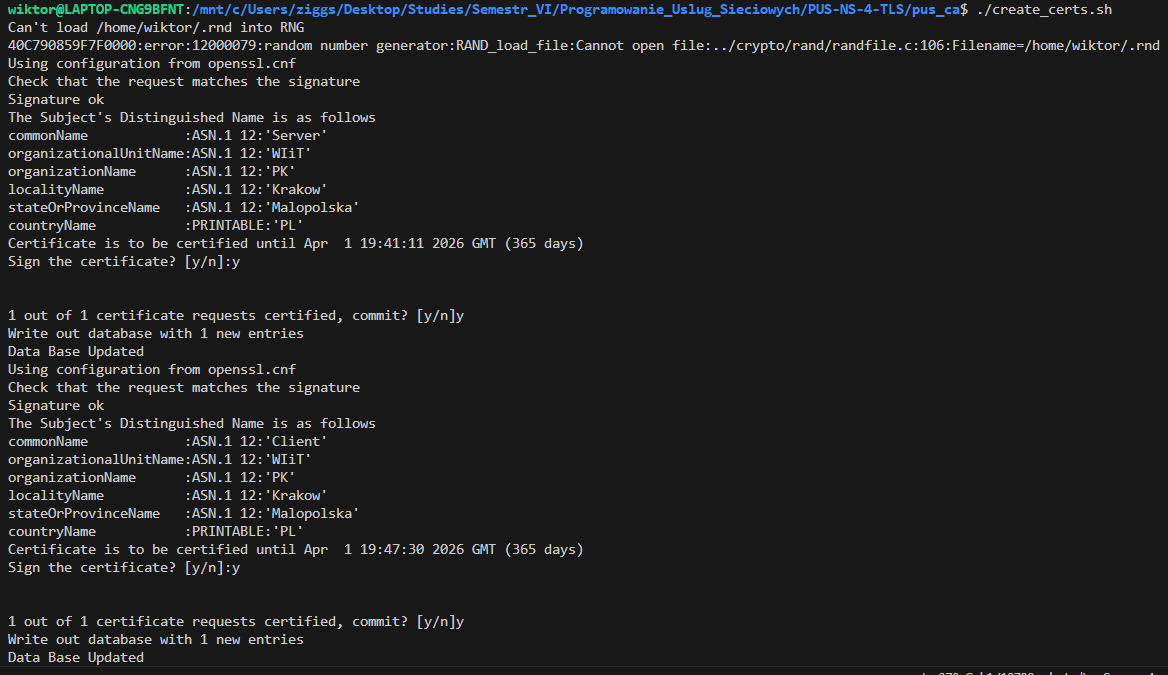
Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

Udało się nawiązać bezpieczne połączenie TLS 1.3 między klientem a serwerem uruchomionymi lokalnie (127.0.0.1). W przeciwieństwie do poprzedniego zadania, gdzie obserwowano jedynie próbę nawiązania połączenia TCP, tutaj Wireshark prawidłowo rozpoznał protokół TLS 1.3 . Widać w sekwencji pakietów komunikaty Client Hello (pakiet 4) inicjujący uzgadnianie TLS oraz Server Hello (pakiet 6) będący odpowiedzią serwera. Można również zidentyfikować inne komunikaty związane z uzgadnianiem TLS, takie jak Change Cipher Spec oraz Application Data (pakiety 6, 8, 9, 11, 13, 14), które zawierają już zaszyfrowane dane wymieniane między klientem a serwerem.

****

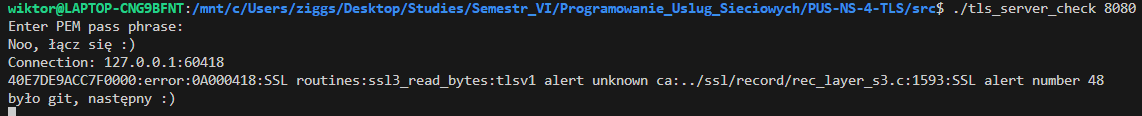
Podczas łączenia się z localhost, nazwa hosta w żądaniu klienta zgadza się z nazwą, dla której prawdopodobnie wystawiony jest certyfikat serwera. Natomiast przy połączeniu z 127.0.0.1, ta nazwa nie pasuje, co powoduje nieudaną weryfikację certyfikatu przez klienta i w konsekwencji zerwanie połączenia SSL

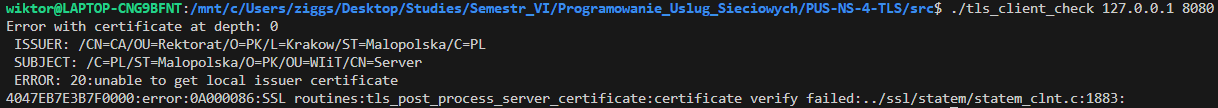
**ZADANIE 3**

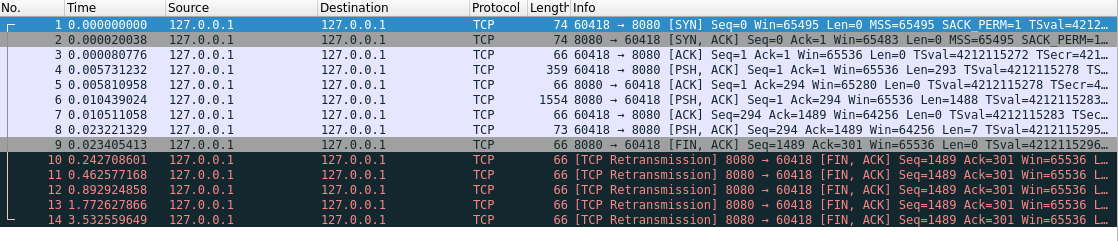
****

Po rozpakowaniu archiwum pus\_ca.tar i przejściu do katalogu pus\_ca, uruchomiłem skrypt create\_certs.sh. Skrypt ten, wykorzystując konfigurację z pliku openssl.cnf, wygenerował niezbędne klucze prywatne i certyfikaty dla centrum certyfikacji (CA), serwera i klienta. Podczas działania skryptu, dla każdego żądania certyfikacyjnego (serwera i klienta), wyświetlane były informacje o danych podmiotu (Distinguished Name) oraz dacie ważności certyfikatu. Po potwierdzeniu podpisania certyfikatów, baza danych wystawionych certyfikatów (index.txt) została zaktualizowana. W wyniku działania skryptu w odpowiednich katalogach (certs, private, tmp) utworzone zostały pliki zawierające certyfikaty (w formacie .pem), klucze prywatne (zaszyfrowane), żądania certyfikacyjne oraz łańcuchy certyfikatów, zgodnie z opisem zadania.

**ZADANIE 4**

****

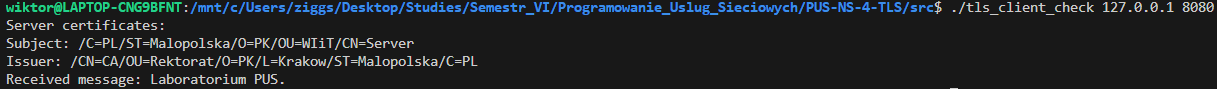
****

****

Widzimy niepowodzenie pierwszego połączenia, które wyniknęło z braku zaufanego certyfikatu. Dodanie certyfikatu CA do cert\_dir oraz wykonanie c\_rehash powinno umożliwić poprawną weryfikację serwera przez klienta.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

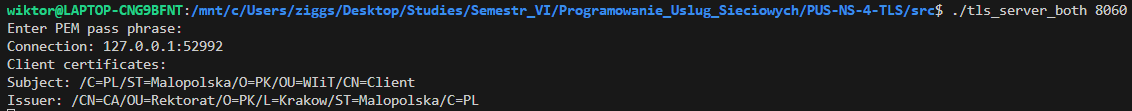
****

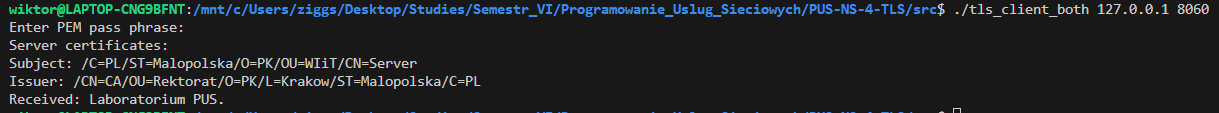
**Obraz zawierający tekst, Czcionka, numer, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

Po dodaniu certyfikatu CA, ruch TLS zawiera udaną wymianę kluczy i potwierdzenie nawiązania szyfrowanej sesji. Wireshark wykazał retransmisje, które mogły być wynikiem nieudanego uwierzytelnienia w pierwszej próbie. Ostatecznie, po poprawnym skonfigurowaniu certyfikatów, połączenie TLS zostało nawiązane pomyślnie.

**ZADANIE 5**

****

****

**Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

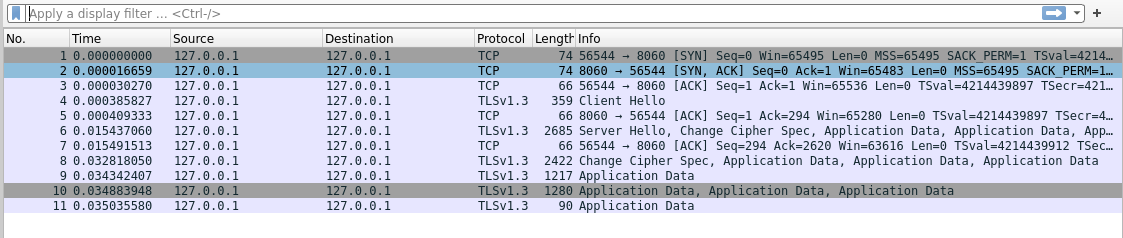
Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

Po zmodyfikowaniu kodów źródłowych tls\_server\_check.c i tls\_client\_check.c w celu załadowania certyfikatów i kluczy zarówno serwera, jak i klienta, oraz skonfigurowaniu serwera do wymagania uwierzytelnienia klienta, uruchomiono serwer i klienta. Zarówno serwer, jak i klient załadowały swoje certyfikaty (server\_chain.pem, client\_chain.pem) oraz klucze prywatne (server\_keypair.pem, client\_keypair.pem). Serwer został również skonfigurowany do weryfikacji certyfikatu klienta poprzez załadowanie katalogu cert\_dir z zaufanymi certyfikatami CA i ustawienie flagi SSL\_VERIFY\_PEER | SSL\_VERIFY\_FAIL\_IF\_NO\_PEER\_CERT. Klient również załadował katalog z zaufanymi certyfikatami i ustawił funkcję verify\_callback do obsługi procesu weryfikacji certyfikatu serwera

W porównaniu do zadania 4, w Wiresharku można zaobserwować dodatkowe komunikaty związane z uwierzytelnianiem klienta. Po wymianie Client Hello i Server Hello, a przed wysłaniem Application Data, powinny pojawić się komunikaty związane z Certificate Request, Certificate oraz CertificateVerify.

**ZADANIE 6**

1. W celu poprawy zamykania połączenia TLS w poprzednich zadaniach należy dołączyć funkcję SSL\_shutdown() która powinna rozwiązać sprawę. Komunikacja po poprawie:



1. Obraz zawierający tekst, numer, Czcionka, oprogramowanie

   Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Po uruchomieniu klienta i server tls\_client i tls\_server faktycznie widzimy że wireshark z jakiś powodów nie rozpoznaje protokołu TLS