|  |
| --- |
| Politechnika Wrocławska, Informatyka Stosowana |
| Ochrona komunikacji |
| Cyberbezpieczeństwo, Laboratorium nr.9 - raport |

|  |
| --- |
| Autor: Aleksander Stepaniuk  Nr. Indeksu: 272644 |

**1. Zadania i pytania (TLS)**

**Zadania 1.1-1.3;**

Przebieg zadań na zrzutach ekranu poniżej:

**Pytanie 1.4;**

Oba protokoły działają w głównej mierze tak samo (służą do szyfrowania i uwierzytelniania danych, aby chronić je przed odczytem przez osoby niepożądane), z tą różnicą że TLS jest następcą SSL, a więc wprowadza różnego rodzaju ulepszenia, między innymi w zakresie bezpieczeństwa i wydajności działania protokołu.

**Pytanie 1.5;**

TLS/SSL handshake to process negocjacji parametrów używanych do nawiązywania połączenia między klientem a serwerem. Podczas tego procesu następuje wymiana szczegółów połączenia takich jak wersja protokołu, klucze sesji czy użyte algorytmy szyfrujące. Jest niezbędnym elementem bezpiecznego przesyłania danych zaszyfrowaną drogą.

**Pytanie 1.6;**

Protokół TLS/SSL zapewnia ochronę transmisji danych poprzez ich szyfrowanie oraz uwierzytelnianie stron komunikacji. Zastosowania:

* Szyfrowanie połączenia między klientem a serwerem podczas przesyłania danych logowania przy rejestracji do wybranego serwisu (np. Netflix)
* Szyfrowanie połączenia między klientem a serwerem podczas prowadzenia połączenia głosowego przez aplikację typu Zoom (ochrona danych przed podsłuchem)
* Uwierzytelnianie stron internetowych HTTPS (np. facebook.com)
* Szyfrowanie e-maili (SMTP)
* Bezpieczne połączenie z serwerem VPN (ochrona prywatności)

**Pytanie 1.7;**  
TLS 1.2 i TLS 1.3. Wersja 1.3 uznawana jest obecnie jako standard i jest powszechnie używana w większości systemów.

**Pytanie 1.8;**

TLS 1.3, ponieważ eliminuje przestarzałe algorytmy (uwierzytelniania i szyfrowania), redukuje liczbę podatnych funkcji i usprawnia proces handshake.

**Pytanie 1.9;**

TLS 1.3, ponieważ korzysta z wydajniejszych algorytmów oraz sam proces przebiega szybciej (oraz zużywa mniej energii) ze względu na:

* 1-RTT zamiast 2-RTT
* AES zamiast asymetrycznego RSA
* Krótszy handshake (każda wymiana kluczy korzysta z Diffiego-Hellmana)

**Pytanie 1.10;**

Wireshark przechwycił obydwie wersje protokołu: TLS 1.2 oraz TLS 1.3

**Pytanie 1.11;**

Zwykle z: *ClientHello*, *ServerHello*, wymiany certyfikatów i kluczy sesji. Proces ten różni się w zależności od wersji protokołu:

* **TLS 1.2**:
  1. *ClientHello*: Klient wysyła informacje o obsługiwanych algorytmach szyfrowania, wersji protokołu i danych do wygenerowania klucza.
  2. *ServerHello*: Serwer odpowiada wyborem algorytmu i wersji TLS.
  3. *Certificate*: Serwer wysyła swój certyfikat do uwierzytelnienia.
  4. *Key Exchange*: Klient i serwer wymieniają dane pozwalające na ustalenie wspólnego klucza sesji.
  5. *Finished*: Obie strony kończą proces, potwierdzając poprawność ustaleń.
* **TLS 1.3**:
  1. *ClientHello*: Klient wysyła propozycję algorytmów szyfrowania i dane do wygenerowania klucza (w tym dane dla szyfrowania asymetrycznego).
  2. *ServerHello*: Serwer akceptuje algorytm i generuje klucz sesji na podstawie danych klienta. Certyfikat serwera jest przesyłany w jednym kroku, skracając czas wymiany.
  3. *Finished*: Po weryfikacji certyfikatu połączenie jest szyfrowane, co zmniejsza liczbę rund komunikacji.

**Pytanie 1.12;**

Zależy od wersji oprogramowania zarówno po stronie klienta jak i serwera (wsparcie systemowe oraz konfiguracja oprogramowania mają znaczenie). Z tego powodu warto regularnie aktualizować oprogramowanie aby mieć dostęp do najnowszych, najszybszych i najbezpieczniejszych wersji protokołów takich jak TLS. Proces komunikacji przebiegać będzie po najnowszej wspólnej wersji protokołu uzgodnionej w handshaku (obie strony muszą obsługiwać daną wersję). Dodatkowo administrator sieci może wymusić użycie konkretnej wersji protokołu. Niektóre przeglądarki lub aplikacje pozwalają na wymuszenie używania nowszych wersji. Serwery proxy lub urządzenia pośredniczące mogą wymuszać starsze wersje w przypadku braku wsparcia.

**2. Zadania (OpenVPN)**

**Zadania 2.1-2.4;**

Przebieg zadań na zrzutach ekranu poniżej:

**3. Zadania i pytania (OpenVPN)**

**Zadania 3.1-3.4;**

Przebieg zadań na zrzutach ekranu poniżej:

**Pytanie 3.5;**

Tak, jeśli wiadomość została przesłana w postaci nieszyfrowanej (dla ruchu z interfejsu sieci VPN), można ją odczytać, przechwytując ruch sieciowy. W przypadku analizy ruchu z interfejsu fizycznego widoczne pakiety OpenVPN przesyłane między klientami miały postać zaszyfrowaną i wymagały klucza dostępu.

**Pytanie 3.6; //work in progress**

W przypadku ruchu na interfejsie VPN pakiety miały jawnie podany prawidłowy adres źródłowy i docelowy – nadawca wiedział, do kogo wysyła wiadomość, a odbiorca, kto ją wysłał. Pakiety przechwycone przy analizie interfejsu hosta miały podany jedynie adres maszyny, z której wysyłano wiadomość oraz serwera – odbiorca nie miał możliwości sprawdzenia, z jakiego adresu wysłano wiadomość, a nadawca nie wiedział, kto wysłał wiadomość.

Na interfejsie sieci VPN ruch jest zaszyfrowany, a na interfejsie hosta może być widoczny w postaci nieszyfrowanej (jeśli nie użyto szyfrowania aplikacyjnego, np. TLS).

Dla ruchu na interfejsie sieci VPN widoczny jest zarówno adres źródłowy jak i docelowy, natomiast na interfejsie hosta pakiety posiadały jedynie adres klienta, który wysłał wiadomość oraz adres serwera, a więc odbiorca wiadomości nie mógł w tym przypadku sprawdzić z jakiego adresu wysłano wiadomość.

**Pytanie 3.7; //to do**

Dla ruchu w sieci VPN można zobaczyć adres źródłowy i docelowy oraz jawny tekst przesyłanej wiadomości, natomiast dla ruchu na interfejsie fizycznym jedynie adres nadawcy oraz serwera, w żaden sposób nie da się podejrzeć tekstu wiadomości. Przesyłanie pliku: Czasy przesyłania pliku: •

* Setting 1 – 2 min 38 s
* Setting 2 – 2 min 16 s
* Setting 3 – 2 min 21 s
* Setting 4 – 2 min 27 s

Na interfejsie hosta można zobaczyć szczegóły transmisji (adresy IP, porty, dane aplikacyjne). Na interfejsie VPN widoczne są tylko zaszyfrowane pakiety i dane metadane (np. adres serwera VPN).

Dla interfejsu fizycznego można zobaczyć: (adres nadawcy, adres serwera, nie widać tekstu samej wiadomości). Dla ruchu w sieci VPN można zobaczyć: (adres źródłowy, adres docelowy, tekst jawny przesyłanej wiadomości). Dodatkowo widać **(coś co widać)**

**4. Zadania i pytania**

**Zadania 4.1-4.3;**

Przebieg zadań na zrzutach ekranu poniżej:

**Pytanie 4.4;**

Istnieje niewielka różnica w czasach przesyłania danych. Jednak jest ona na tyle niewielka, że można stwierdzić, iż wybór konkretnego algorytmu nie ma większego znaczenia przy czasie przesyłania pliku. Różnice w zmierzonych czasach mogą wynikać z innych procesów uruchomionych w tym samym czasie na maszynie, które mogły w niektórych przypadkach spowolnić proces przesyłania.

Tak, szyfrowanie i dodatkowe trasy przez serwer VPN zwiększają opóźnienia. Na czas wpływają m.in. algorytmy szyfrowania, odległość do serwera VPN oraz przepustowość sieci.

**Pytanie 4.5;**

Algorytm AES-256-GCM jest symetrycznym algorytmem powszechnie używanym do szyfrowania danych i uchodzącym za bezpieczny. Algorytm SHA512 jest algorytmem uwierzytelniania używającym klucza o długości 512 bitów i również uchodzi za bardzo bezpieczny. Algorytm AES-128-CBC jest inną odmianą algorytmu AES z kluczem 128- bitowym i pracującym w trybie CBC. Jest jednak mniej bezpieczny niż jego wersja 256- bitowa, ze względu na mniejszą długość klucza. Algorytm SHA1 to algorytm uwierzytelniania o kluczu długości 160 bitów. Jest on mniej bezpieczny niż jego 512-bitowa wersja. DES-EDE-CBC to wariant algorytmu szyfrowania DES-CBC, wykorzystujący 3 klucze. Jest on uznawany za bezpieczny, jednak zaleca się wykorzystywanie „nowszego” AES. Algorytm MD5 używa skrótu o długości 128 bitów. Współcześnie nie gwarantuje już większego bezpieczeństwa i zaleca się zastąpienie go algorytmami z rodziny SHA.

Algorytmy silne (np. AES-GCM, RSA-2048) zapewniają lepsze bezpieczeństwo, ale mogą być wolniejsze. Słabe algorytmy (np. DES) są podatne na ataki i niewskazane.

**Pytanie 4.6;**

Za najbezpieczniejszy zestaw można uznać użycie do szyfrowania algorytmu AES-256- GCM, zaś do uwierzytelniania algorytmu SHA512, ponieważ długości kluczy użyte w procesie szyfrowania oraz uwierzytelniania są wystarczająco długie, aby w stu procentach zapewnić bezpieczeństwo szyfrowanych danych. Zaś najmniej bezpiecznym jest zestaw z algorytmem DES-CBC oraz MD5. Tych algorytmów nie używa się już powszechnie (znane są skuteczne metody ataków na nie) i zaleca się zastąpienie ich m.in. przez bezpieczniejsze AES i SHA.

Najgorsze: brak szyfrowania lub użycie przestarzałych algorytmów (np. MD5, RC4).

Najlepsze: nowoczesne algorytmy, np. TLS 1.3 z AES-256-GCM i PFS (Perfect Forward Secrecy).