Ćwiczenie nr: 5
Temat: Tunelowanie, VPN i elementy kryptografii

1. Informacje ogólne

Tunel jest to zestawienie połączenia między dwoma odległymi komputerami tak, by stworzyć wrażenie, że są połączone bezpośrednio. Przesłanki do tworzenia tuneli:

1. wygoda
2. umożliwienie połączenia między komputerami ukrytymi w sieciach prywatnych (za firewallem, lub z adresami prywatnymi - warunek jeden z hostów biorących udział w tworzeniu tunelu musi mieć adres publiczny)
3. bezpieczeństwo - szyfrowanie połączenia: SSL/TLS, SSH
4. możliwość przyspieszenia transmisji – kompresja sprawdza się szczególnie na wolnych łączach

Pojęcia tunelowanie zazwyczaj używa się w odniesieniu do przesyłania poprzez tunel tylko jednego protokołu. Zwykle jest to jeden z typowych protokołów np POP3, SMTP, HTTP, które przesyłają dane w sposób jawny (w tym nazwy użytkowników i hasła) za pomocą bezpiecznych protokołów TLS/SSl lub SSH.

Wirtualna sieć prywatna (VPN) – umożliwia transmisję w sposób bezpieczny wielu protokołów poprzez publiczną sieć. Hosty będące po obu stronach VPN-u „widzą się" tak jakby były w jednej sieci. Takie sieci przeważnie działają w warstwie 3 modelu TCP/IP i tunelują warstwę 3 i wyższe, niektóre z nich umożliwiają także transmisję 2 warstwy (np tworzą wirtualną kartę sieciową).

Kompresja powinna spełniać dwa podstawowe kryteria: niskie zapotrzebowanie na moc procesora i szybkość działania. Kompresji może zostać poddana całość transmisji lub tylko ładunek użyteczny pakietów IP

Szyfrowanie umożliwia zabezpieczenie transmisji. Znane są dwie podstawowe metody szyfrowania:

- symetryczna przy pomocy współdzielonego klucza
  - ten sam klucz służy do zaszyfrowania i rozszyfrowania danych
  - jest to szybka i skuteczna metoda szyfrowania
  - podstawowym problemem jest bezpieczne dostarczenie do obu hostów klucza służącego do szyfrowania i deszyfrowania
  - stosowane klucze są stosunkowo krótkie 128, 256, 512b (dawniej także 40 i 56b)
- asymetryczna
  - polega na użyciu dwóch kluczy
    - klucza publicznego – klucz jest dostępny dla każdego użytkownika i służy do zaszyfrowania transmisji oraz do weryfikacji podpisu
    - klucz prywatny – jest przechowywany w bezpiecznym miejscu i służy do rozszyfrowywania danych zaszyfrowanych kluczem publicznym oraz do tworzenie podpisu elektronicznego
  - konieczne jest stosowanie znacznie dłuższych ciągów bitów (zwykle generowane losowo ciągi liczb pierwszych) od długościach równych lub przekraczających 1kb (1024, 2048, 4096b)
  - wymaga większych mocy obliczeniowych niż klucz symetryczny
  - jest znacznie wygodniejsza w użyciu

Aby wykorzystać zalety i zniwelować wady obu metod kryptograficznych większość powszechnie używanych rozwiązań stosuje kombinację obu technik:

1. bezpieczne połączenie tworzone jest z wykorzystaniem kluczy publicznego i prywatnego (przy tym sprawdzana jest wiarygodność obu stron)
2. generowany jest klucz sesyjny – używany jednorazowo dla konkretnego połączenia i co jakiś czas może być wymieniany dla większego bezpieczeństwa
3. klucz sesyjny przesyłany jest przy pomocy bezpiecznego już połączenia
4. dalsza transmisja odbywa się z użyciem klucza symetrycznego

W oparciu o ten schemat działają protokoły m.in. SSH, SSL/TLS
Powstaje jeszcze jeden problem – weryfikacja wiarygodności stron połączenia.

W przypadku SSH generowane są skróty kryptograficzne klucza publicznego zwane także odciskiem palca (fingerprint) i przy pierwszym połączeniu użytkownik jest zobowiązany go zaakceptować co powoduje jego zapisanie przez klienta SSH w „bazie znanych hostów" (np `~./ssh/known_hosts`). Do kolejnych połączeń nie dojdzie gdy skrót kryptograficzny, którym przedstawia się serwer nie zgadza się z tym zapisanym w bazie. W tym przypadku odpowiedzialność za sprawdzenie wiarygodności serwera spada na użytkownika.

W SSL/TLS zwykle stosuje się metodę potwierdzania tożsamości na zasadzie dziedziczenia

zaufania w kombinacji z podpisem elektronicznym. Tworzy to PKI (Public Key Infrastructure) Infrastrukturę Klucza Publicznego. PKI opiera się na instytucji zaufania tzw. Urzędzie Certyfikacji (CA – Certification Authority) podpisującym certyfikaty i Urzędach Rejestracji (RA – Registration Authority), które zbierają wnioski i weryfikują
W pierwszym etapie tworzony jest certyfikat, który zawiera:

- nazwę certyfikowanego obiektu
- identyfikator obiektu
- klucz publiczny obiektu
- czas ważności
- identyfikator wystawcy
- podpis wystawcy
- numer seryjny (opcja)
- przeznaczenie (serwer lub klient)

Pełna nazwa obiektu powinna zawierać informacje w formacie X500 (kontekst w strukturze drzewa katalogowego) a X509 opisuje sposób użycia asymetrycznych kluczy kryptograficznych w celu składania podpisów i ich weryfikacji.
Certyfikaty mogą być samopodpisane (ich zastosowanie praktyczne jest ograniczone i ich bezpieczeństwo opiera się na akceptacji użytkownika)
W przypadku certyfikatów podpisanych przez Zaufany Urząd Certyfikacji system w sposób automatyczny weryfikuje certyfikaty przez niego podpisane mając zapisany w bazie certyfikat tego urzędu. (Przeglądarka internetowe łącząc się ze stroną o URL zaczynającym się od https:// sprawdza czy certyfikat serwera został podpisany przez jeden z wbudowanych w jej bazie Urzędów Certyfikacji). CA – (Certyfication Authority) posiada certyfikat główny, który służy tylko i wyłącznie do podpisywania certyfikatów Urzędów Certyfikacji, które służą do podpisywania certyfikatów klientów i serwerów lub pośrednich urzędów certyfikacji. W zależności od przeznaczenia certyfikatu ich cena bywa bardzo wysoka (np dla banków) ale odpowiedzialność CA i RA jest wysoka, gdyż to one gwarantują swoim autorytetem że instytucja, dla której certyfikat został wystawiony jest tą za którą się podaje.

2. Standardy i protokoły

2.1 SSL/TLS

TLS (Transport Layer Security) jest rozwinięciem standardu SSL (Secure Socket Layer) opracowanego przez firmę Netscape. TLS zapewnia poufność transmisji i uwierzytelnienie stron lub przynajmniej jednej ze stron. Używa PKI i standardu X509. Obecnie zalecaną wersją jest TLS 1.1 (RFC4346 z 4366, 4680, 4681).
SSL v3 i TLS1.x używają dwóch protokołów:

- SSL Handshake - nawiązanie połączenia i negocjowanie algorytmów szyfrowania
- SSL Record - definiuje format przesyłanych pakietów danych

Do działania wymagany jest

- w najprostszej wersji - przynajmniej 1 certyfikat (samopodpisany certyfikat serwera) w tym przypadku klient zawsze będzie informował użytkownika o ułomności tego rozwiązania.
- typowo – Certyfikat(y) CA oraz certyfikat serwera przez niego podpisany, jeżeli klient ma w swojej bazie certyfikat CA połączenie zostanie nawiązane bez ostrzeżeń (potwierdzona zostaje utentyczność tylko jednej ze stron – serwera)
- w wersji najbezpieczniejszej - obie strony transmisji posiadają podpisane certyfikaty, podpisane przez CA znajdujące się w bazie obu stron – potwierdzona jest tożsamość obu stron

Bazę certyfikatów CA klienta i serwera uzupełnia lista certyfikatów, które straciły wiarygodność tzw CRL (Certification Revocation List)

Schemat nawiązania połączenie protokołem SSL/TLS (certyfikat posiada tylko serwer)

Z	DO	komunikat	
Klient	Serwer	ClientHello	wersja, obsługiwane met. kryptograficzne, liczba losowa
Serwer	Klient	ServerHello	jak wyżej
Serwer	Klient	Certificate	wysyła certyfikat (opcja ale zwykle występuje)
Serwer	Klient	ServerKeyExchange	info o kluczu publicznym
Serwer	Klient	ServerHelloDone	przejście do następnej fazy
Klient	Serwer	ClientKeyExchange	na podstawie obu liczb losowych generowany jest klucz sesyjny
Klient	Serwer	ChangeCipherSpec	klient zawiadamia że można przejść na protokół szyfrowany

Klient	Serwer	Finished	klient jest gotowy na przyjmowanie danych zaszyfrowanych
Serwer	Klient	ChangeCipherSpec	serwer zawiadamia o przejściu na prot. szyfrowany
Serwer	Klient	Finished	próba już zaszyfrowanym kanałem

Ponowne nawiązanie połączenia:
Jeżeli w ClientHello zostanie podany identyfikator poprzedniej sesji to zostanie użyty klucz sesyjny z poprzedniej sesji

2.2 SSH1/2

Podstawowym celem powstania oprogramowania SSH była bezpieczna alternatywa dla przestarzałych i niezabezpieczonych protokołów rsh i telnet umożliwiających zdalne wykonywanie poleceń powłoki systemów uniksowych oraz zdalną pracę. Opierały się na domyślnym zaufaniu hostów (rsh) lub na autoryzacji za pomocą nazwy użytkownika i hasła ale przesyłanych w sposób nieszyfrowany i łatwy do przechwycenia szczególnie w sieciach publicznych. Stopniowo funkcjonalność SSH rozszerzono o tunelowanie protokołów rodziny X oraz bezpieczną transmisję plików scp (lub sftp). Obecnie tunelować przy pomocy SSH można dowolny protokół, który używa pojedynczych portów lub wykorzystać wirtualny interfejs tun linuksa pozwalający tworzyć tunel w warstwie 3. Początkowo SSH było oprogramowaniem komercyjnym. Bardzo szybko powstała wersja OpenSSH (początkowo obsługująca tylko wersję SSH 1), która od końca lat 90-tych ubw. wyparła całkowicie wersję komercyjną. Serwer SSH wykorzystuje standardowo port 22. Jak to zostało opisane wcześniej SSH wykorzystuje zarówno kryptografię asymetryczną (do nawiązania połączenia i przesłania klucz sesyjnego) ja i symetryczną (klucz sesyjny). Przy pierwszym połączeniu klient ssh wymaga akceptacji skrótu kryptograficznego serwera i zapisuje go do bazy a następnie pyta o hasło (w przypadku ssh nazwa użytkownika jest przesyłana automatycznie przy połączeniu) w Putty (klient SSH dla MS Windows) można podać nazwę użytkownika tuż po nawiązaniu połączenia
SSH w wersji 1 używa asymetrycznego klucz RSA a do transmisji symetrycznych kluczy 3DES i Blowfish
Wersja 2 aby uniknąć problemów patentowych z RSA (już nie aktualne) używa asymetrycznych algorytmów DSA i DH oraz wiele różnych algorytmów dla kluczy symetrycznych (IDEA, 3DES, Blowfish, AES, Arcfour pochodna RC4)
Sposób działania i metody kryptograficzne przypominają OpenSSL za wyjątkiem użycia certyfikatów
Można używać kart procesorowych

2.3 OpenVPN

Implementacja tunelu punkt-punkt z wykorzystaniem SSLv3/TLSv1, w odróżnieniu od wielu innych implementacji VPN nie bazuje na IPsec ale wykorzystuje wirtualny interfejs sieciowy tun/tap (tun – warstwa 3 protokół IP, tap – warstwa druga – np ethernet) dostępny jest m.in. dla Linux-a, BSD (w tym Mac OS X), Windows2000 w górę, Solaris.
Autentykacja może odbywać się za pomocą współdzielonego klucza, certyfikatów oraz nazwy użytkownika i hasła. Pakiet składa się z jednego programu binarnego dodatkowo można użyć pliku konfiguracyjnego oraz plików z kluczem współdzielonym lub certyfikatów.
Szyfrowanie bazujące na SSL może być wspomagane przez mechanizm haszujący HMAC (**Hash Message Authentication Code)** w celu zwiększenia bezpieczeństwa. Pracuje w przestrzeni użytkownika i używa do komunikacji protokołu UDP lub TCP wykorzystując oficjalnie przyznany temu protokołowi przez IANA port 1194. Nie wymaga tak jak IPsec tworzenia 2 kanałów – w OpenVPN tunel jest dwukierunkowy i łatwiej przechodzi przez firewalle. Pozwala na równoległe tworzenie wielu tuneli i wspomaga ruting między nimi. Do kompresji używany jest szybki algorytm LZO podobny do gzip. Wspiera akceleracją sprzętową oraz karty procesorowe.

2.4 IPsec (IP security)

W odróżnieniu od pozostałych omawianych typów tuneli i VPN pracuje w warstwie 3 modelu OSI (warstwie sieciowej) co wymaga pracy w trybie jądra systemu operacyjnego. Jest to zestaw protokołów dobrze udokumentowanych przez odpowiednie rfc (4301-4309). Zapewnia m.in. autentykacje, szyfrowanie oraz ustanawianie klucza. IPsec jest obowiązkowym elementem IPv6 stąd wszystkie systemy oferujące IPv6 posiadają także obsługę IPsec. Bezpieczeństwo w IPsec realizowane jest przez:

autoryzacja z wykorzystaniem PKI
szyfrowanie transmisji
sprawdzanie integralności (czy pakiety nie były zmodyfikowane)

Bardziej szczegółowo:

- protokoły związane z zarządzaniem kluczami szyfrującymi

    - IKE(v1/2) – wymiana kluczy (ręczna lub zautomatyzowana)
- protokoły związane z bezpieczństwem
    - Authentication Header (AH) – zapewnia autoryzację, może być stosowany samodzielnie (bez szyfrowania)
    - Encapsulating Security Payload ESP – zapewnia poufność przesyłanego ładunku użytecznego pakietu IP
- metody kryptograficzne
    -

IPsec pracuje w dwóch trybach

w trybie transportowym

Tylko ładunek użyteczny pakietu jest szyfrowany. Trasowanie pakietu pozostaje bez zmian. Nagłówek pakietu nie może być zmieniany (jego skrót kryptograficzny dołączany jest do nagłówka pakietu) co utrudnia wykorzystanie tego rozwiązania w kombinacji z NAT (translacja adresów sieciowych m.in w firewallach) można to częściowo obejść ale utrudnia to konfigurację NAT (odblokowanie IKE – port 500 UDP, IPsec NAT-T – port 4500 UDP i ESP port 50 IP) często określa się to jako „IPsec Passthrough". To rozwiązanie zapewnia komunikację host-host

w trybie tunelu

Cały pakiet jest szyfrowany i enkapsulowany w nowym pakiecie IP. służy do komunikacji sieć- sieć, sieć-host i host-host

W wielu przypadkach wygodne jest użycie tunelowania warstwy drugiej w IPsec możliwe jest to dzięki protokołowi L2TP (layer 2 tunneling protocol)

Implementacje IPsec w różnych systemach operacyjnych:

Microsoft Windows:

począwszy od Windows 2000 Prof (XP home i słabsze wersje Visty
nie zawierają serwera IPsec)
Windows 2008, 2003 i 2000 jeżeli działają jako serwer nie obsługują NAT-T
opcje: IKE, NAT-T, L2TP

Linux:

począwszy od jądra 2.0.x implementacje FreeSwan, OpenSwan, strongSwan
moduł jądra KLIPS lub NETKEY (natywne)
IKE (FreeSwan/OpenSwan v1, strongSwan v2), L2TP, NAT-T

BSD:

KAME
IKE

MacOS X, iPhone

L2TP, problemy z importem certyfikatów lub obsługa tylko PSK (współdzielony klucz)

Wiele komercyjnych rozwiązań opartych o Linuksa lub BSD może posiadać sprzętowe wspomaganie operacji kryptograficznych lub używać modułów kryptograficznych
Osobna grupa to urządzenia dedykowane do tworzenie VPN firm CISCO (system IOS), Nortel, CheckPoint, Juniper i innych

3 Instrukcje do użytego oprogramowania

3.1 OpenSSH

Serwer SSH (OpenSSH) zwykle uruchamiany jest jako demon/usługa systemu operacyjnego pliki konfiguracyjne oraz klucze (wygenerowane przy pierwszym uruchomieniu) przechowywane są w katalogu /etc/ssh

moduli, sshd_config, ssh_host_dsa_key.pub, ssh_host_key.pub, ssh_host_rsa_key.pub
ssh_config, ssh_host_dsa_key, ssh_host_key, ssh_host_rsa_key

sshd_config – jest plikiem konfiguracyjnym serwera
ssh_config – jest plikiem konfiguracyjnym klienta

W katalogu użytkownika (zwykle .ssh) znajduje się plik known_hosts zawierający listę hostów z ich adresami i kluczami publicznym zapisywane po zatwierdzeniu pierwszego połączenia i sprawdzane przy kolejnych
W przypadku używania logowania z użyciem tylko klucz (bez hasła) jest jeszcze plik authorized_keys z listą hostów i ich kluczy, które dopuszczone są do logowania w tym trybie (wymagana opcja PubKeyAuthentication yes w pliku sshd_config)

ssh-keygen – program służący do generowania kluczy
ssh-keygen -t <algorytm szyfrowania klucza publicznego>

Przykłady użycia:

`ssh-keygen -t rsa` – generuje klucze rsa i zapisuje w plikach .ssh/id_rsa (prywatny) i ssh/id_rsa.pub (publiczny)

użycie opcji klienta SSH

`ssh <opcje> [użytkownik@]serwer <komenda>`

opcje	parametr	opis
-1		tylko wersja 1 SSH
-2		tylko wersja 2 SSH
-l	login	tak samo jak użykownik@
-p	port	port do połączenia jeśli nie 22
-R	port:host:hostport	tunel zwrotny - zdalny serwer nasłuchuje na porcie „port” i przekazywane do komputera lokalnego z którego kierowane jest połączenie na adres host i port hostport
-L	port:host:hostport	tunel – lokalny komputer nasłuchuje na porcie „port” i przekazuje przez SSH do komputera zdalnego z którego kierowane jest połączenie na adres host i port hostport
-w	localtun[:reomotetun]	tunelowanie za pomocą ssh wirtualnego interfejsu tun
-f		przejście do działania w tle po ustanowieniu połączenia wyjście jest przekierowane do /dev/null
-c	algorytm	3des-cbc, aes128-cbc, aes192-cbc, aes256-cbc, aes128-ctr, aes192-ctr, aes256-ctr, arcfour128, arcfour256, arc-four, blowfish-cbc, and cast128-cbc
-o	opcja	
	Compresion	rodzaj kompresji
	CompresionLevel	poziom kompresji

Przykłady użycia:

`ssh student@10.0.2.186 -L 8080:localhost:80`

tworzy tunel pomiędzy lokalnym komputerem nasłuchując na porcie 8080 i przekazuje pakiety do komputera 10.0.2.186 (gdzie jesteśmy zalogowani jako student) ten przekazuje już bez szyfrowania połączenie do adresu loclahost (127.0.0.1) na port 80 (WWW)

Połączenie można sprawdzić wywołując URL http://localhost:8080/ z lokalnego komputera

`ssh student@10.0.2.186 -R 8080:localhost:80`

tworzy tunel zwrotny pomiędzy zdalnym komputerem nasłuchując na porcie 8080 (na zdalnym komputerze 10.0.2.186 gdzie jesteśmy zalogowani jako student) i przekazuje pakiety do lokalnego komputera przez SSH ten przekazuje już bez szyfrowania połączenie do adresu loclahost (127.0.0.1) na port 80 (WWW)

Połączenie można sprawdzić wywołując URL http://localhost:8080/ ze zdalnego (10.0.2.186) komputera lub http://10.0.2.186:8080/ z każdego innego będącego w tej samej sieci co 10.0.2.186

ssh -f -w 0:1 10.0.2.186 true

Tworzy wirtualną sieć prywatną pomiędzy lokalnym komputerem na interfejsie tun0 i zdalnym komputerem 10.0.2.186 wykorzystując tam interfejs tun1. Konieczne jest jeszcze skonfigurowanie adresów IP obu interfejsów (ifconfig tun0 addr_1 addr_2 dla lokalnego i ifconfig tun1 addr_2 addr_1dla zdalnego) oraz tras routowania (route add siec/maska gateway)

3.2 OpenVPN

sposób użycia

openvpn <opcje>

opcja	parametry	informacje
--genkey		generowanie współdzielonego klucza (razem z --secret)
--secrect	plik static.key	dzielony klucz zapisany jest do pliku static.key

opcje przy połączeniu

server		praca w trybie serwera (tylko z certyfikatami)
client		praca w trybie klienta (tylko z certyfikatami)
-- proto	tcp/udp	używany protokół
--port	domyślnie 1194	port
--remote	adres IP	adres IP lub nazwa serwera
--local	adres IP	adres na którym nasłuchuje serwer
--dev	tun/tap	wirtualny interfejs sieciowy
--comp-lzo		z kompresją LZO
--verb	0-9	ilość komunikatów debugujących
--ifconfig	LadrIP RadrIP	adresy IP odpowiednio lokalnego i zdalnego hosta
--ca	ca-cert.pem	certyfikat urzędu CA
--cert	cert.pem	certyfikat klienta/servera
--key	key.pem	plik z kluczem prywatnymi
--secret	plik.key	plik z kluczem współdzielonym (tylko w trybie z kluczem dzielonym)
--config	plik	użycie pliku konfiguracyjnego
--daemon		przejście do trybu usługi/demona

w pliku konfiguracyjnym parametry umieszcza się bez poprzedzających znaków „--” np

```
##### client.conf######
client
remote 10.0.2.2
port 1194
proto tcp
```

Tryb bez autoryzacji szyfrowania i kompresji
Komputery StudentA i StudentB (odpowiednio nazwy lub adresy ip tych komputerów)

Na StudentA
```
openvpn --remote StudentB --dev tun0 --ifconfig 192.168.1.1 192.168.1.2 --verb 3
```

Na StudentB
```
openvpn --remote StudentA --dev tun0 --ifconfig 192.168.1.2 192.168.1.1 --verb 3
```

zostanie ustanowiony tunel z wykorzystaniem interfejsów tun0 na obu komputerach StudentA i StudentB
interfejs tun0 na StudentA będzie miał adres IP 192.168.1.1 a na StudentB 192.168.1.2
--verb 3 jest przydatne w teście połączenia w dalszych pomiarach lepiej użyć --verb 0
UWAGA adresów **`192.168.1.2 192.168.1.X nie zmieniać`**

3.3 wget

wget <opcje> URL

niektóre opcje:
-O (duże „o”) skierowanie pobieranego pliku do pliku o innej nazwie
„-” standardowe wyjście
-o (małe „o”) - skierowanie komunikatów programu do pliku

Generowanie plików

---------- plik tekstowy ----------
skrypty gen_txt.sh

```
#!/bin/bash
for ((a=1, b=1; a <= 1800000 ; a++, b++))
do
  echo "0123456789Asdfghjklawsfdhkjdfchiu3wryhuiqwefnlewhqiuewhriuehfisbnfanbsia"
  echo "hbrefiouhweoiuasbvibqabfasdobvaoirbfvoiabvoiasbdviqbreniberv;bfnjkjaewfdb6"
  echo "lk984hozdgbn5ihsvn dfm,n skldnfvbsdjbskfbkjbvc,mzx d566%*#@)@EUREU@"
done
```

Uruchomienie
	/bin/bash gen_txt.sh >/srv/www/htdocs/plik.txt

---------- plik binarny ----------
skrypty gen_bin.sh

```
#!/bin/bash
dd if=/dev/urandom of=/srv/www/htdocs/binary count=600000
```

Uruchomienie
	/bin/bash gen_bin.sh
	zapisuje plik binarny /srv/www/htdocs/binary o wielkości 512B*600000

Literatura:

[1] Stawowski Mariusz, Projektowanie i praktyczne implementacje sieci VPN, Warszawa, ArsKom, 2004
[2] Karbowski Marcin, Podstawy kryptografii, Gliwice, Helion, 2006

Scenariusze ćwiczeń

Scenariusz Nr 1 OpenVPN

Sprzęt:

1. StudentA:
   Komputer PC

Procesor	AMD Athlon64 32(4)00+
Architektura	x86_64 (64b)
RAM	1GB
SO	OpenSuSE 11.1 x86_64
użytkownik	root
hasło	1

lub

   Komputer PC

Procesor	Intel Crore2 Quad Q8600
Architektura	x86_64 (64b)
RAM	4GB
SO	OpenSuSE 11.1 x86_64
użytkownik	root
hasło	1

2. StudentB
   jw

Oprogramowanie:

openvpn z bibliotekami SSL i LZO

Wykonanie ćwiczenia:

Przygotowanie:

1. w trybie graficznym systemu linux uruchomić ksysguard (monitor wydajności) na StudentA i StudentB
2. uruchomić serwer http apache komendą /etc/init.d/lighttpd start (na StudentA)
3. skopiować wyznaczony przez prowadzącego plik do katalogu /srv/www/htdocs (na StudentA) i nadać mu nazwę „plik”
   pliki
   a) plik tekstowy o wielkości ok 1G
   komendą

```
for ((a=1 , b=1; a<1800000l a++, b++)) ; do echo
”abcdegghijklmnoprstuwxyz1234567890abcdegghijklmnoprstuwxyz1234567890abcdegghijklm
noprstuwxyz1234567890abcdegghijklmnoprstuwxyz1234567890abcdegghijklmnoprstuwxyz123
4567890abcdegghijklmnoprstuwxyz1234567890” >>/srv/www/htdocs/plik ; done
```

   lub pobrać skrypt generujący plik /srv/www/htdocs/plik z URL http://www.eti.agh.edu.pl/sieci/pliki/gen_txt.sh
   z konsoli zmienić jego uruchomić go komendą

```
sh ./gen_txt.sh
```

   b) plik binarny o wielkości ok 300MB
   komendą

```
dd  if=/dev/urandom of=/srv/www/htdocs/plik count=600000
```

   c) plik skompresowany o wielkości ok 100MB
   d) plik video o wielkości (jeden z http://tempus.metal.agh.edu.pl/~opal/sieci/video/)
   zanotować wielkość pliku

UWAGA generowanie plików trwa kilka minut

4. zanotować wielkość pliku
5. ew wygenerować klucz współdzielony
   - w katalogu /etc/openvpn uruchomić **`openvpn --genkey --secret key`** lub skorzystać z klucza /mnt/mpi/key
   - UWAGA ten sam klucz musi być przeniesiony na drugi komputer.
6. lub/i certyfikat serwera (opcja dodatkowa dla ambitnych do 5**e** i 5**f**)
   - w katalogu /etc/openvpn uruchomić
     - **`/usr/share/openvpn/easy-rsa/build-ca`**
     - **`/usr/share/openvpn/easy-rsa/build-key-server server`**

- **`openssl dhparam -out dh4096.pem 4096`**

UWAGA na dysku sieciowym dostępny jest klucz symetryczny do wykorzystania

Wykonanie ćwiczenia:

1. Pobrać 4 razy (pierwszy pomiar czasu się nie liczy) za pomocą programu wget plik z komputera StudentA
   - połączenie bez tunelu będzie odniesieniem

   `wget http://StudentA/plik -O - >/dev/null`
2. Zestawić połączenie wg jednej z wersji a-f
3. przesłać wyznaczony plik 4 razy (pierwszy pomiar się nie liczy) za pomocą komendy

   `wget http://192.168.1.1/plik -O - >/dev/null`
4. Zestawić połączenie wg innej z wersji a-f
5. przesłać wyznaczony plik 4 razy (pierwszy pomiar się nie liczy) za pomocą komendy

   `wget http://192.168.1.1/plik -O - >/dev/null`

a) połączenie bez szyfrowania, autoryzacji i kompresji
b) połączenie bez szyfrowania i autoryzacji ale z kompresją (--comp-lzo)
c) połączenie z szyfrowaniem przy pomocy współdzielonego klucza ale bez kompresji (--secret key)
d) połączenie z szyfrowaniem przy pomocy współdzielonego klucza z kompresją (--comp-lzo -- secret key)
e)połączenie z użyciem certyfikatów bez kompresji (--ca ca.pem --cert cer.pem --key.pem)
f)połączenie z użyciem certyfikatów bez kompresji (--ca ca.pem --cert cer.pem –key.pem --comp-lzo)

Wyniki pomiarów:

opracować statystycznie
ocenić wpływ ustawionych parametrów na szybkość transmisji i obciążenie procesora
wyciągnąć wnioski

Scenariusz Nr 2 SSH

Sprzęt:

1. StudentA:

Komputer PC

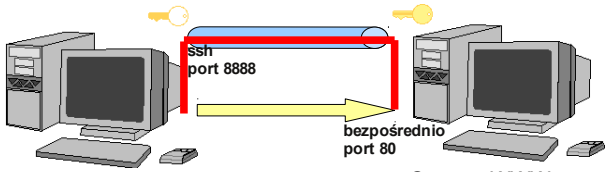
Procesor	AMD Athlon64 32(4)00+
Architektura	x86_64 (64b)
RAM	1GB
SO	OpenSuSE 11.1 x86_64
użytkownik	root
hasło	1

lub

Komputer PC

Procesor	Intel Crore2 Quad Q8600
Architektura	x86_64 (64b)
RAM	4GB
SO	OpenSuSE 11.1 x86_64
użytkownik	root
hasło	1

2. StudentB
   jw



Oprogramowanie:
OpenSSH

Wykonanie ćwiczenia:
Przygotowanie:

1. w trybie graficznym systemu linux uruchomić ksysguard (monitor wydajności) na StudentA i StudentB
2. uruchomić serwer http apache komendą /etc/init.d/apache2 start (na StudentA)
3. skopiować wyznaczony przez prowadzącego plik do katalogu /srv/www/htdocs (na StudentA) i nadać mu nazwę „plik”
   pliki
   a) plik tekstowy o wielkości ok 1G
   komendą

```
for ((a=1 , b=1; a<1800000l a++, b++)) ; do echo
”abcdegghijklmnoprstuwxyz1234567890abcdegghijklmnoprstuwxyz1234567890abcdegghijklm
noprstuwxyz1234567890abcdegghijklmnoprstuwxyz1234567890abcdegghijklmnoprstuwxyz123
4567890abcdegghijklmnoprstuwxyz1234567890” >>/srv/www/htdocs/plik ; done
```

   lub pobrać skrypt generujący plik /srv/www/htdocs/plik z URL http://www.eti.agh.edu.pl/sieci/pliki/gen_txt.sh
   z konsoli zmienić jego uruchomić go komendą

```
sh ./gen_txt.sh
```

   b) plik binarny o wielkości ok 300MB
   komendą

```
dd  if=/dev/urandom of=/srv/www/htdocs/plik count=600000
```

   c) plik skompresowany o wielkości ok 100MB
   d) plik video o wielkości (jeden z http://tempus.metal.agh.edu.pl/~opal/sieci/video/)
4. zanotować wielkość pliku
5. ew wygenerować klucze tylko jeżeli adresów tych komputerów nie ma w `~/.ssh/authorized_keys`
   - w katalogu /etc/openvpn uruchomić **`ssh-keygen -t rsa <lub dsa>`**
   - i dopisać klucz do tego pliku cat rsa.pub >> **`authorized_keys`**

UWAGA generowanie plików trwa kilka minut

Wykonanie ćwiczenia:

1. uruchomić serwer WWW najlepiej lighttpd
   komendy z poziomu konsoli roota
   `service apache2 stop` - dla pewności zatrzymujemy apacha
   `service lighttpd restart` - uruchomienie lighttpd
2. Pobrać 4 razy (pierwszy pomiar czasu się nie liczy) za pomocą programu wget plik z komputera StudentA
   - połączenie bez tunelu będzie odniesieniem

```
wget http://StudentA/plik -O - >/dev/null
```

3. Zestawić połączenie wg jednej z wersji a,c,e,g ze StudentaB do StudentA

ssh StudentB -L 8888:localhost:80 <odpowiednie opcje>

4. przesłać wyznaczony plik 3 razy za pomocą komendy
   `wget http://StudentB:8888/plik -O - >/dev/null`
5. Zestawić połączenie wg odpowiadającej wersji z kompresją i poziomem kompresji X=3 6 i 9
   ssh StudentB -L 8888:localhost:80 <odpowiednie opcje>
6. przesłać wyznaczony plik 3 razy za pomocą komendy
   `wget http://StudentB:8888/plik -O - >/dev/null`

wersja	metoda szyfrowania	kompresja	uwagi
a)	Blowfish	nie	-c blowfish-cbc -o ”Compression no”
b)	Blowfish	tak	-c blowfish -C -o ”CompressionLevel X” - X warość od 0-9
c)	3DES	nie	-c 3des -o ”Compression no”
d)	3DES	tak	-c 3des -C -o ”CompressionLevel X” - X warość od 0-9
e)	Arcfour	nie	-c arcfour -o ”Compression no”
f)	Arcfour	tak	-c arcfour -C -o ”CompressionLevel X” - X warość od 0-9
g	Cast	nie	-c cast128-cbc -o ”Compression no”
h	Cast	tak	-c cast128-cbc -C -o ”CompressionLevel X” - X warość od 0-9

UWAGA Proszę zwrócić uwagę, na którym hoscie jest konsola przy komendzie wget …

Wyniki pomiarów:

opracować statystycznie
ocenić wpływ ustawionych parametrów na szybkość transmisji i obciążenie procesora
wyciągnąć wnioski