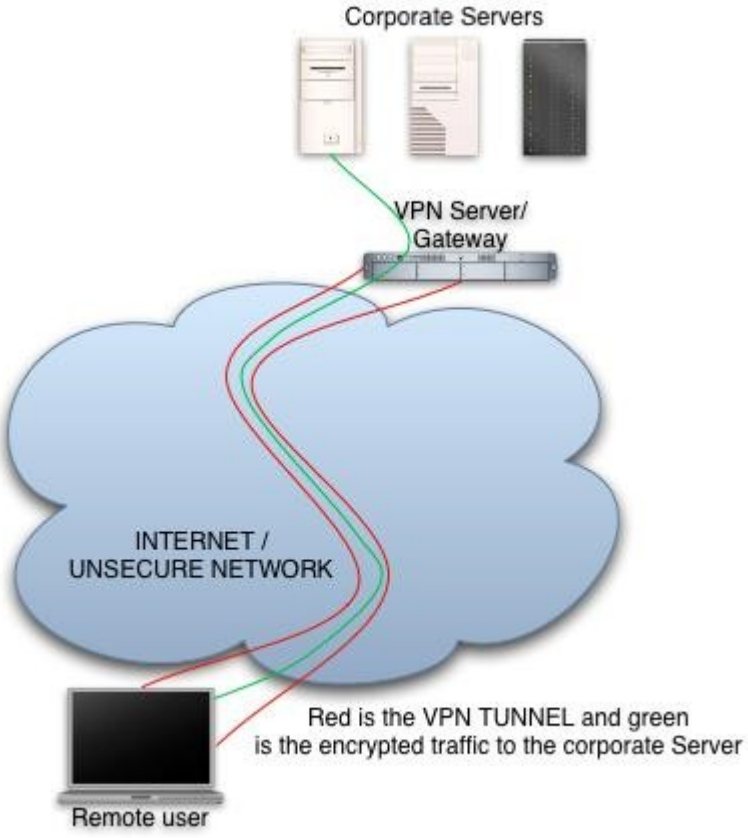
INŻYNIERIA BEZPIECZEŃSTWA – LABORATORIUM

VPN / OpenVPN

1. Czym jest VPN ?

VPN (ang. Virtual Private Network, Wirtualna Sieć Prywatna), można opisać jako tunel, przez który płynie ruch w ramach sieci prywatnej pomiędzy klientami końcowymi za pośrednictwem publicznej sieci (takiej jak Internet) w taki sposób, że węzły tej sieci są przezroczyste dla przesyłanych w ten sposób pakietów. Taki kanał może opcjonalnie kompresować lub szyfrować w celu zapewnienia lepszej jakości lub większego poziomu bezpieczeństwa przesyłanych danych.

Określenie "Wirtualna" oznacza, że sieć ta istnieje jedynie jako struktura logiczna działająca w rzeczywistości w ramach sieci publicznej, w odróżnieniu od sieci prywatnej, która powstaje na bazie specjalnie dzierżawionych w tym celu łącz. Pomimo takiego mechanizmu działania stacje końcowe mogą korzystać z VPN dokładnie tak jak gdyby istniało pomiędzy nimi fizyczne łącze prywatne. Rozwiązania oparte na VPN powinny być stosowane np. w sieciach korporacyjnych firm, których zdalni użytkownicy dosyć często pracują ze swoich domów na niezabezpieczonych łączach. Wirtualne Sieci Prywatne charakteryzują się dość dużą efektywnością, nawet na słabych łączach (dzięki kompresji danych) oraz wysokim poziomem bezpieczeństwa (ze względu na szyfrowanie).



2. Tunel. Tunelowanie połączenia.

Tunel – zestawienie połączenia między dwoma odległymi hostami tak, by stworzyć wrażenie, że są połączone bezpośrednio.

W miarę rozwoju sieci komputerowych najpierw lokalnych, a następnie rozległych, powstało zapotrzebowanie na łączenie ze sobą różnych sieci lokalnych za pośrednictwem publicznych sieci rozległych. Sieci lokalne korzystają jednak z innych protokołów sieciowych niż sieci rozległe. Na przykład popularne sieci lokalne firmy Novell pracują w protokole IPX, sieci rozległe wykorzystują natomiast protokoły Frame Relay, ATM, X.25, a na nich często IP (np. Internet). Łączenie sieci wykorzystujących inny protokół niż sieć rozległą, za pomocą której łączymy sieci rozległe w wirtualną sieć prywatną (VPN), nie jest jedynym przesłaniem wykorzystywania tunelowania. Drugim i często istotniejszym jest bezpieczeństwo. Szczególnie ostatnio tunelowanie bywa często łączone z wykorzystaniem metod kryptograficznych. Często zwykli użytkownicy Internetu stosują tę technikę do własnych potrzeb. Przykładem jest korzystanie z dostępnego w Internecie oprogramowania szyfrującego o nazwie SSH. Oprogramowanie to, oprócz bezpiecznej pracy zdalnej, umożliwia tworzenie dodatkowego kanału szyfrowanego, przez który użytkownik może "tunelować" dowolną inną - potencjalnie narażoną na niebezpieczeństwo podsłuchu - usługę TCP/IP (np. FTP, telnet, IRC). Warunkiem jest jedynie zainstalowane SSH na obu końcach połączenia.

3. OpenVPN.

OpenVPN to pakiet VPN stworzony przez Jamesa Yonana. Umożliwia on tworzenie zaszyfrowanych połączeń między hostami – używa do tego celu biblioteki OpenSSL oraz protokołów SSLv3/TLSv1. W przeciwieństwie do innych rozwiązań VPN nie bazuje na protokole IPsec jako medium. Pakiet ten dostępny jest na platformach Linux, BSD, Mac OS X oraz Windows 2000/XP/Vista. Cały pakiet składa się z jednego kodu binarnego dla klienta i serwera, opcjonalnego pliku konfigurującego oraz z jednego lub więcej plików kluczy w zależności od metody uwierzytelnienia.
OpenVPN używa bibliotek OpenSSL do szyfrowania danych i kanałów kontrolnych. Może również korzystać z HMAC by stworzyć dodatkową warstwę zabezpieczenia połączenia. Pakiet jest w stanie również wykorzystać możliwości sprzętowe, by polepszyć stopień i jakość szyfrowania.
OpenVPN oferuje kilka metod uwierzytelnienia użytkowników: poprzez klucze, certyfikaty lub nazwę użytkownika i hasło (opcja z nazwą użytkownika i hasłem może być stosowana, w przypadku klienta bez certyfikatu).

4. OpenSSL.

OpenSSL – wieloplatformowa, otwarta implementacja protokołów SSL (wersji 2 i 3) i TLS (wersji 1) oraz algorytmów kryptograficznych ogólnego przeznaczenie. Udostępniana jest na licencji zbliżonej do licencji Apache. Dostępna jest dla systemów uniksopodobnych (m.in. Linux, BSD, Solaris), OpenVMS i Microsoft Windows.
OpenSSL zawiera biblioteki implementujące wspomniane standardy oraz mechanizmy kryptograficzne, a także zestaw narzędzi konsolowych (głównie do tworzenia kluczy czy certyfikatów, zarządzania certyfikatami i urzędem certyfikacji, (de)szyfrowania, obliczania podpisów cyfrowych i innych).

Za pomocą OpenSSL Crypto Library można m.in. obliczać funkcję skrótu wiadomości oraz szyfrować dane popularnymi algorytmami kryptograficznymi, m.in. Blowfish, AES, IDEA, 3DES.

5. How To : OpenVPN – instrukcja konfiguracji serwera/klienta w sieci VPN.

1. Zainstalować aktualna stabilną wersję oprogramowania OpenVPN ze strony:
http://openvpn.net/index.php/open-source/downloads.html
(instrukcja opracowana w oparciu o wersję 2.1.4).

2. Po poprawnym zainstalowaniu oprogramowania, pierwszym krokiem konfiguracji jest wygenerowanie kluczy (najlepiej uruchamiać kolejne pliki wsadowe poprzez linię poleceń w systemie Windows (menu start->uruchom->cmd), w systemach Vista i 7 należy pamiętać o trybie administratora).

a). Wygenerowanie głównego certyfikatu służacego podpisywaniu pozostałych kluczy (the master Certificate Authority (CA)).
W folderze **\Program Files\OpenVPN\easy-rsa**, który powinien się pojawić w miejscu zainstalowania aplikacji znajduje się plik wsadowy **init-config.bat**

b). Po uruchomieniu go należy dokonać edycji utworzonego pliku **vars.bat** i zamienić odpowiednie linie skryptu na pasujące dla naszej lokalizacji:
KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEY_ORG, oraz KEY_EMAIL
Nie wolno zostawić tych parametrów pustych!

c). Następnie należy kolejno uruchomić pliki:
vars.bat
clean-all.bat
build-ca.bat

Warto zwrócić uwagę, że podczas wykonywania powyższych skryptów większość parametryzowanych parametrów ma domyślne wartości, takie jakie ustawiliśmy w pliku **vars.bar.** Jedynym parametrem, który musimy wpisać własnoręcznie jest **Common Name**. Dla przejrzystości można wpisać "OpenVPN-CA".

d). Generowanie klucza i certyfikatu dla serwera odbywa się poprzez wywołania pliku
build-key-server server
Ponownie większość parametrów jest ustawiona domyślnie. Jako **Common Name** warto wpisać "server". Dwa kolejne zapytania: "Sign the certificate? [y/n]" i "1 out of 1 certificate requests certified, commit? [y/n]" wymagają zatwierdzenia poprzez wpisanie y.

e). Generowanie klucza i certyfikatu dla 3 różnych klientów (w ćwiczeniu będziemy potrzebować tylko jednego zestawu, ale warto utworzyć kilka dla różnych stacji klienckich)
build-key.bat client1
build-key.bat client2
build-key.bat client3

Należy pamiętać o wpisaniu poprawnej nazwy dla **Common Name** np.: "client1", "client2", lub "client3". Należy zawsze używać unikatowych nazw dla każdego klienta.

f). Generowanie parametrów Diffiego-Hellmana dla serwera OpenVPN (więcej o parametrach na stronie : Diffie Hellman). Parametry dla naszej długości klucza uzyskujemy poprzez wykonanie skryptu:
build-dh.bat

3. Po wygenerowaniu wszystkich kluczy znajdują się one w folderze **\Program Files\OpenVPN\easy-rsa\keys**. Klucze te należy przekopiować zgodnie z poniższą tabelką do folderu **\Program Files\OpenVPN\config :**

Nazwa pliku:	**Na jakim komputerze potrzebny:**	**Cel:**	**Plik tajny:**
ca.crt	serwer + wszystkie stacje klienckie	Root CA certificate	Nie
ca.key	Tylko stacja na której klucze zostały wygenerowane	Root CA key	Tak
dh{n}.pem	Tylko serwer	Diffie Hellman parameters	Nie
server.crt	Tylko serwer	Server Certificate	Nie
server.key	Tylko serwer	Server Key	Tak
client1.crt	Tylko stacja klienta nr 1	Client1 Certificate	Nie
client1.key	Tylko stacja klienta nr 1	Client1 Key	Tak
client2.crt	Tylko stacja klienta nr 2	Client2 Certificate	Nie
client2.key	Tylko stacja klienta nr 2	Client2 Key	Tak
client3.crt	Tylko stacja klienta nr 3	Client3 Certificate	Nie
client3.key	Tylko stacja klienta nr 3	Client3 Key	Tak

Należy pamiętać o kopiowaniu kluczy na komputery które ich wymagają poprzez bezpieczny kanał!!!

4. Tworzenie konfiguracji na serwerze vpn, wykonujemy poprzez stworzenie pliku server.ovpn również w folderze **\Program Files\OpenVPN\config** na komputerze, który będzie pełnił funkcję serwera.

plik server.ovpn:

```
################################################
# ustawienie portu na którym nasłuchujemy połączeń
port 1194
# ustawienie interfejsu połączenia
dev tap
# ustawienie protokołu połączenia
proto udp
# nazwy certyfikatow
ca ca.crt
```

```
cert server.crt
key server.key
# plik z parametrami Diffie hellman
dh dh1024.pem
# Konfiguracja trybu serwera oraz adresacji podsieci VPN
# adres 10.8.0.1 zostanie przypisany serverowi reszta jest dostępna dla
klientów
server 10.8.0.0 255.255.255.0
# Ustawienie zapisywania w pliku, przypisań wirtualnych adresów ip dla
poszególnych klientów, w celu odzyskania ich w razie restartu połączenia
ifconfig-pool-persist ipp.txt
# ustawienie czasowego pingowania pomiędzy oboma stronami w celu
podtrzymania połączenia oraz monitorowania zerwania połączenia
keepalive 10 120
# parametry dotyczące zachowania parametrów podczas restartów
persist-key
persist-tun
# ustawienie pliku z krótkim statusem obecne połączenia
status openvpn-status.log
# włączenie kompresji transmisji.
comp-lzo
# ustawienie poziomu szczegółowości logowania
verb 3
```

5. Tworzenie konfiguracji klienta VPN, wykonujemy poprzez stworzenie pliku client.ovpn również w folderze **\Program Files\OpenVPN\config** na wszystkich komputerach, które będą pełnił funkcję klienta. Adres `remote 192.168.2.104` należy zamienić na odpowiadający adresowi maszyny pełniącej rolę serwera.

plik client.ovpn:

```
################################################
# ustawienie połączenia jako klienta w wyniku czego część ustawień jest
pobierana z serwera
client
# ustawienie interfejsu połączenia
dev tap
# ustawienie protokołu połączenia
proto udp
# przykładowe ustawienie adresu i portu zdalnego hosta pełniącego rolę
#serwera VPN
remote 192.168.2.104 1194
#ustawienie ciągłego wznawiania połączenia
resolv-retry infinite
# opcja ustawiająca brak konieczności przypisywania stałych portów
nobind
# parametry dotyczące zachowania parametrów podczas restartów
persist-key
persist-tun
# nazwy certyfikatow dla clienta 1 – dla pozostałych klientów należy
#zmienić na odpowiednie
ca ca.crt
```

```
cert client1.crt
key client1.key
# sprawdzenie poprawności wygenerowania certyfikatu serwera
ns-cert-type server
# włączenie kompresji transmisji.
comp-lzo
# ustawienie poziomu szczegółowości logowania
verb 3
```

Po utworzeniu pliku konfiguracji należy uruchomić nakładkę graficzną na oprogramowanie OpenVPN – „OpenVPN GUI". Gdy aplikacja się uruchomi, pojawi się ikonka wśród ikon systemowego zasobnika (System Tray). Aby uruchomić połączenie, w pierwszym kroku należy uruchomić serwer połączenia poprzez kliknięcie ikonki aplikacji prawym przyciskiem myszki i kliknięcie connect. Następnie należy powtórzyć łączenie na stacjach klienckich.
Najprostszym sposobem weryfikacji poprawności połączenia (poza zielonym kolorem ikonki), jest wysłanie pinga z jednego komputera drugiego za pomocą nowego adresu IP z klasy 10.8.0.0.
Dane dotyczące połączenia lub jego braku można wyczytać z loga tworzonego przez oprogramowanie.

6. Weryfikacja połączenia poprzez dokonanie przykładowej komunikacji.

W przypadku chęci weryfikacji połączenia na konkretnym porcie oraz chęci sprawdzenia poprawności transmisji, można skorzystać z prostego zestawu aplikacji serwer-klient (w przypadku tego ćwiczenia aplikacje są wykonane w języku Java).
Poniższe kody aplikacji trzeba uruchomić na dwóch różnych komputerach w sieci VPN. Uruchamianie powinno zacząć się od aplikacji napisanej dla serwera. W efekcie działania tych aplikacji komunikat tekstowy powinien zostać wysłany z klienta do serwera i wyświetlony w konsoli.
Aby połączenie doszło do skutku w kodzie klienta należy podmienić adres IP na odpowiadający drugiemu komputerowi. Należy sprawdzić wpierw rzeczywisty adres IP, jeśli połączenie będzie udane należy sprawdzić adres wirtualny. Wynik działania w obu przypadkach powinien być jednakowy.

Plik serwera (Server.java):

```
import java.io.DataInputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
	private ServerSocket serverSocket = null;
	private Socket clientSocket = null;
	DataInputStream in = null;

	public void start(int port) throws IOException {
		serverSocket = new ServerSocket(port);
		clientSocket = serverSocket.accept();
		in = new DataInputStream(clientSocket.getInputStream());
		System.out.println(in.readLine());
```

```
	}

	public static void main(String[] args) {
		Server server = new Server();
		int port = 1234;
		try {
			server.start(port);
		} catch (IOException e) {}
	}

}
```

Plik klienta (Client.java):

```
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;

public class Client {
	private Socket socket = null;
	private OutputStream out = null;

	void connect(String host, int port) throws UnknownHostException,
			IOException {
		socket = new Socket(host, port);
	}

	void sendMsg(String msg) throws IOException {
		if (socket != null) {
			out = socket.getOutputStream();
			byte[] b = msg.getBytes();
			out.write(b);
		} else {
			throw new Error("no connection");
		}
	}

	void disconnect() throws IOException {
		out.close();
		socket.close();
	}

	public static void main(String[] args) {
		Client client = new Client();
		String host = "192.168.2.104";  // adres serwera
	//	String host = "10.8.0.1";       // adres wirtualny serwera
		int port = 1234;
		try {
			client.connect(host, port);
			client.sendMsg("Witaj swiecie");
```

```
                client.disconnect();
            } catch (Exception e) {
            }
        }
}
```

7. Zadania do samodzielnego wykonania.

1 Przerobić skrypty generujące klucze uwierzytelniające połączenie z serwerem VPN na jeden parametryzowany plik wsadowy.
2 Przy połączeniu z niezabezpieczona siecią wifi przechwycić komunikacje z wcześniejszego zadania z użyciem vpn i bez.
3 Wygenerować klucze w oparciu o Certificate Signing Request (CSR) (klucze generowane na stacji klienckiej i wykorzystanie mechanizmu CSR do podpisania kluczy, zamiast używania bezpiecznego kanału).