



AKADEMIA GÓRNICZO-HUTNICZA

WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ

BEZPIECZEŃSTWO W SIECI

---

## OpenVPN

---

*Autorzy:*

Maciej Kubicki  
Karol Kozicki  
Filip Kania

*Prowadzący:*

dr inż. Piotr Kowalski

26 czerwca 2015

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>2</b>
1.1	Co to jest VPN . . . . .	2
1.2	Najczęściej spotykane rodzaje VPN . . . . .	2
1.3	Główne cele VPNa . . . . .	2
1.4	Pakiet OpenVPN . . . . .	2
<b>2</b>	<b>Instalacja środowiska</b>	<b>3</b>
2.1	Pliki . . . . .	3
2.2	Wykorzystywane pakiety . . . . .	3
2.2.1	VirtualBox . . . . .	3
2.2.2	Vagrant . . . . .	3
2.3	Skrypt startowy . . . . .	3
2.4	Płyty z obrazami . . . . .	3
<b>3</b>	<b>Pliki dołączone do projektu</b>	<b>4</b>
3.1	Skrypty instalacyjne . . . . .	4
3.2	Pliki konfiguracyjne . . . . .	4
<b>4</b>	<b>Topologia sieci</b>	<b>5</b>
4.1	Opis . . . . .	5
<b>5</b>	<b>Generowanie certyfikatów</b>	<b>6</b>
5.1	Certyfikaty dla serwera . . . . .	6
5.2	Certyfikat dla klienta . . . . .	6
<b>6</b>	<b>Plik VagrantFile</b>	<b>6</b>
<b>7</b>	<b>Konfiguracja połączenia OpenVPN</b>	<b>7</b>
7.1	Konfiguracja Serwera . . . . .	7
7.2	Konfiguracja Klienta . . . . .	7
<b>8</b>	<b>Realizacja przy wykorzystaniu połączenia TUN</b>	<b>8</b>
<b>9</b>	<b>Realizacja przy wykorzystaniu połączenia TAP</b>	<b>10</b>
<b>10</b>	<b>Podsumowanie</b>	<b>12</b>

# 1 Wstęp

## 1.1 Co to jest VPN

VPN jest to tunel, przez który przepływa 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. Można opcjonalnie kompresować lub szyfrować przesyłane dane w celu zapewnienia lepszej jakości lub większego poziomu bezpieczeństwa. Wirtualna sieć 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 sposobu działania stacje końcowe mogą korzystać z VPN dokładnie tak, jak gdyby istniało pomiędzy nimi fizyczne łącze prywatne.

## 1.2 Najczęściej spotykane rodzaje VPN

- PPTP
- L2TP
- OpenVPN
- IPsec

## 1.3 Główne cele VPNa

Rozwiązania oparte na VPN stosowane są np. w sieciach korporacyjnych firm, których zdalni użytkownicy pracują ze swoich domów na niezabezpieczonych łączach. Wirtualne Sieci Prywatne charakteryzują się dość dużą efektywnością, nawet na słabych łączach (dzięki możliwości kompresji danych) oraz wysokim poziomem bezpieczeństwa (ze względu na możliwość szyfrowania połączenia). Rozwiązanie to sprawdza się w firmach, których pracownicy często podróżują lub pracują zdalnie. Dzięki temu pracownik ma dostęp do danych, do których miałby dostęp będąc w biurze, na przykład do bazy danych czy też serwera WWW. Kolejną zaletą takiej sieci jest możliwość realizacji kopii zapasowej w odległej lokalizacji.

## 1.4 Pakiet OpenVPN

OpenVPN jest to pakiet oprogramowania, który implementuje techniki tworzenia bezpiecznych połączeń punkt-punkt (VPN) lub strona-strona w sieciach routowanych lub mostkowanych. Umożliwia on tworzenie zaszyfrowanych połączeń (tuneli) między hostami przez sieć publiczną Internet. Używa do tego celu biblioteki OpenSSL oraz protokołów SSLv3/TLSv1.

Współczesne sieci VPN posiadają wiele mechanizmów, których zadaniem jest zapewnienie bezpieczeństwa takiego połączenie:

- Uwierzytelnianie(ang. Authentication) - jest to weryfikacja czy dany system jest w rzeczywistości tym za kogo się podaje. Dotyczy to zarówno serwera, jak i klienta. Do uwierzytelniania stosuje się infrastrukturę klucza prywatnego.
- Autoryzacja - służy do sprawdzenia czy dany klient ma faktyczny dostęp do interesującego go zasobu sieci. Wykorzystuje się hasła, czy też zestaw kluczy,
- Szyfrowanie - mechanizm zapewniający poufność oraz zapobiegający modyfikacji danych przez osoby trzecie.

Dodatkowo różne instancje sieci VPN posiadają swoje metody zapewniające bezpieczeństwo. Dla OpenVPN jest to na przykład klucz HMAC(ang. keyed-Hash Message Authentication Code). Jego działanie polega na podpisywaniu każdego pakietu kontrolującego innym kluczem.

## 2 Instalacja środowiska

### 2.1 Pliki

Wszystkie pliki składające się na projekt umieszczone są w repozytorium GitHub. Repozytorium znajduje się pod adresem:

```
1 https://github.com/KaRi94/OpenVPNNetwork
```

Dostępna jest także wersja w spakowanym pliku, znajdująca się pod adresem:

```
1 https://github.com/KaRi94/OpenVPNNetwork/archive/master.zip
```

### 2.2 Wykorzystywane pakiety

Aby stworzyć środowisko niezbędne do przeprowadzania testów wykorzystaliśmy:

#### 2.2.1 VirtualBox

VirtualBox to darmowa i bardzo dobrze wyposażona wirtualna maszyna, która pozwala bez względu na to, czy korzysta z systemu Microsoftu, czy któregoś z dystrybucji Linuksa na uruchomienie drugiego bądź wielu systemów operacyjnych działających pod kontrolą już istniejącego systemu.

#### 2.2.2 Vagrant

Vagrant jest to narzędzie stworzone z myślą o tworzeniu wirtualnych środowisk dla deweloperów, bazujące na dobrze znanym VirtualBoxie oraz języku Ruby. Narzędzie to znacznie usprawnia oraz przyspiesza tworzenie maszyn wirtualnych, posiada także wiele funkcji usprawniających konfigurację już istniejących maszyn.

### 2.3 Skrypt startowy

Do szybkiego zarządzania topologią sieci naszego projektu i zainstalowania potrzebnego oprogramowania stworzyliśmy prosty skrypt startujący. Jest to plik **network.sh**. W skrypcie mamy zdefiniowane trzy funkcje: *start*, *stop*, *delete*. Do pierwszego uruchomienia używamy komendy:

```
1 ./network.sh start
```

Polecenie to sprawdza czy mamy zainstalowane potrzebne oprogramowanie i w razie potrzeby pobiera wymagane pakiety. Potem rozpoczyna się budowa topologii sieci (budowane są wirtualne maszyny), zdefiniowanej w pliku **Vagrantfile**. Proces ten trwa około 10-15 minut. Kolejną funkcję wywołujemy poleceniem:

```
1 ./network.sh stop
```

Polecenie to służy do zatrzymania wirtualnych maszyn. Do ponownego uruchomienia tych maszyn należy użyć funkcji *start*. Ostatnią funkcję dostępną w skrypcie wywołujemy przy pomocy:

```
1 ./network.sh delete
```

Funkcja *delete* służy do trwałego usunięcia wirtualnych maszyn. Dodatkowo po użyciu tej funkcji pokazują się komendy do skasowania zainstalowanego oprogramowania przy pierwszym uruchomieniu tego skryptu.

### 2.4 Płyty z obrazami

Na potrzeby projektu przygotowaliśmy płyty DVD z wrzuconymi przygotowanymi już środowiskami. Domyślne dane do logowania na każdą z maszyn to *vagrant* : *vagrant*. Niestety wykorzystując wcześniej przygotowane maszyny bezpośrednio w virtualboxie tracimy wszelkie dogoności wynikające z użytkowania vagranta. Musimy pamiętać o samodzielnym uruchomieniu skryptów */home/all\_openvpn.sh*, */home/all\_locals.sh* po każdym restarcie wirtualnych maszyn.

## 3 Pliki dołączone do projektu

### 3.1 Skrypty instalacyjne

Skrypty użyte w trakcie stawiania wirtualnych a także przy użyciu polecenia *Vagrant provision* maszyn to:

- **all\_locals.sh** - skrypt wykonujący się na każdej maszynie LocalClient. Przy pomocy tego skryptu instalowane jest dodatkowe oprogramowanie na maszynach takie jak tcpdump, czy nmap. Dodatkowo dla warstwy trzeciej ustawiana jest trasa domyślna.
- **all\_openvpn.sh** - skrypt wykonujący się na maszynach OVPNServer oraz dwóch OVPNC-client. Za pomocą tego skryptu instalowane jest dodatkowe oprogramowanie, a także uruchamiany jest OpenVPN, a także dla warstwy trzeciej tworzona jest maskarada.
- **local1.sh** - skrypt wykonujący się na maszynie LocalClient1. Skrypt ten dla warstwy drugiej instaluje serwer DHCP oraz uruchamia go. Natomiast dla warstwy trzeciej instaluje "apacha".
- **openvpn-bridge** - skrypt wykonujący się w trakcie podnoszenia interfejsu *tap*. Używany do zmostowania interfejsów.
- **attacer.sh** - skrypt wykonujący się na maszynie Attacker. Instalacja tcpdumpa.

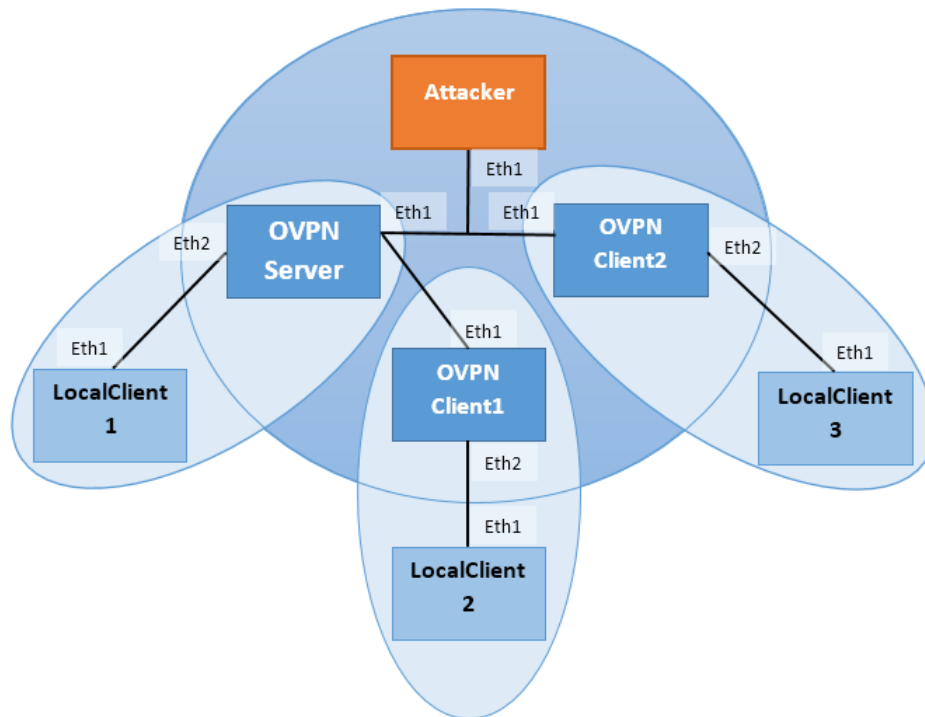
### 3.2 Pliki konfiguracyjne

Pliki konfiguracyjne kopiowane zaraz po uruchomieniu maszyn wirtualnych to:

- **dhcpd.conf** - konfiguracja puli DHCP dla LocalClient (połączenie TAP)
- **isc-dhcp-server.conf** - konfiguracja serwera DHCP dla LocalClient (połączenie TAP)
- niezbędne certyfikaty i klucze wraz z konfiguracją zawarte w katalogu **cert**

## 4 Topologia sieci

Topologia sieci tworzonej przez skrypt startowy przedstawiona jest na poniższej ilustracji. Opis maszyn wirtualnych wchodzących w skład sieci znajduje się w następnej sekcji.



### 4.1 Opis

W skład sieci wchodzi siedem elementów.

1. OVPNServer - Server OpenVPN
2. OVPNClient1 - Router 01 OpenVPN
3. OVPNClient2 - Router 02 OpenVPN
4. LocalClient1 - Server DHCP/WWW połączony do OVPNServer
5. LocalClient2 - Klient OpenVPN połączony przez OPVNClient1
6. LocalClient3 - Klient OpenVPN połączony przez OPVNClient 2
7. Attacker - Atakujący mający możliwość podsłukiwania pakietów wymienianych pomiędzy połączonymi instancjami OpenVPN

## 5 Generowanie certyfikatów

Certyfikaty zostały wygenerowane wcześniej aby oszczędzić czas w trakcie inicjalizacji środowiska. Algorytm postępowania przy generowaniu certyfikatów jest następujący:

### 5.1 Certyfikaty dla serwera

Do wygenerowania certyfikatów potrzebujemy pakietów: OpenVPN, Easy-RSA, OpenSSL. Jeśli ich nie posiadamy to instalujemy:

```
1 sudo apt-get install openvpn openssl
```

Następnie kopiujemy katalog Easy-RSA do katalogu OpenVPN:

```
1 cp -prv /usr/share/doc/openvpn/examples/easy-rsa/2.0 /etc/openvpn/easy-rsa
```

Przechodzimy do tego katalogu, tworzymy kopie zapasową pliku vars, który możemy edytować w celu zmiany podstawowych informacji o serwerze (lokalizacji, czas ważności certyfikatu, wielkość klucza). Wczytujemy plik:

```
1 source ./vars
2 ./clean-all
```

Komenda clean czyści poprzednie certyfikaty. Teraz wydajemy polecenie:

```
1 ./build-ca
```

Potwierdzamy lub zmieniamy dane podane w pliku vars.

```
1 ./build-key-server 'nazwa serwera'
```

Wykonujemy następne polecenie i potwierdzamy jeszcze raz dane z vars, dodatkowo możemy ustalić hasło i na koniec podpisujemy certyfikat. Wygenerowane pliki znajdują się w folderze /keys.

### 5.2 Certyfikat dla klienta

Do generacji certyfikatu klienta używamy polecenia:

```
1 ./build-key 'nazwa klienta'
```

Postępujemy jak w przypadku generacji certyfikatu dla serwera, potwierdzamy dane, ustalamy hasło i podpisujemy. Certyfikat jest w folderze /keys.

## 6 Plik VagrantFile

W pliku VagrantFile znajduje się główne serce konfiguracji wszystkich elementów topologii. To właśnie w tym miejscu tworzymy sieci wewnętrzne, nadajemy adresy IP interfejsą na każdym z elementów topologii. Plik składa się z widocznie podzielonych siedmiu segmentów, każdy z nich zawiera informacje o sposobie konfiguracji każdej z maszyn.

W konfiguracji wykorzystaliśmy proste funkcje oferowane przez vagranta (dokładny opis można znaleźć w obszernej dokumentacji vagrant[1]):

- zmienna *config.vm.box* przechowuje nazwę i wersję dystrybucji Linuxa którą instalujemy na każdej z maszyn wirtualnych
- zmienna *maszyna.vm.hostname* przechowuje nazwę maszyny
- funkcja *maszyna.vm.network* tworzy sieć, przyjmuje jako parametr rodzaj sieci w virtualboxie oraz adres IP
- funkcja *maszyna.vm.provision* odpowiedzialna jest za wiele czynności w zależności od przesłanych parametrów. W naszym przypadku wykorzystujemy funkcje do kopiowania plików z maszyny lokalnej na wirtualną oraz aby wykonać skrypt.

Jedyną ważną opcją z punktu widzenia osoby nie zainteresowanej samą istotą konfiguracji jest zmienna *MODE* zdefiniowana na samym początku pliku. Zmienna ta odpowiada za to czy generowana topologia będzie połączona OpenVPN na warstwie III (:tun) 0 tryb routingu, czy na warstwie II (:tap) - tryb mostkowania.

## 7 Konfiguracja połączenia OpenVPN

Przy tworzeniu połączenia musimy zdecydować poprzez którą warstwę modelu OSI chcemy się połączyć. Mamy do wyboru warstwę II - łącza danych(tap) oraz warstwę III - sieci (tun). W zależności od tego wyboru będziemy operowali na ramach ethernetowych, bądź pakietach IP.

### 7.1 Konfiguracja Serwera

Pliki konfiguracyjne:

- **server\_II.conf** - dla warstwy drugiej
- **server\_III.conf** - dla warstwy trzeciej

W plikach konfiguracyjnych serwera musimy podać szereg niezbędnych rzeczy. Na początku wybieramy rodzaj interfejsu, którego będziemy używali(tun dla warstwy III, tap dla warstwy II). Wybieramy topologie. Ustawiamy adres serwera, protokół używany do komunikacji(dla warstwy II użyliśmy UDP, a dla warstwy III TCP), używany port, klasę adresu IP dla tunelu VPN. Kolejną niezbędną rzeczą jest wybór certyfikatów, które będą użyte w procesie uwierzytelniania. W konfiguracji serwera znajduje się też szereg przydatnych rzeczy takich jak:

- ustawienie algorytmu szyfrującego
- utrzymanie interfejsu w stanie up przy restarcie
- konfiguracja DNS dla DHCP
- plik do którego są wyrzucane logi serwera

### 7.2 Konfiguracja Klienta

Pliki konfiguracyjne:

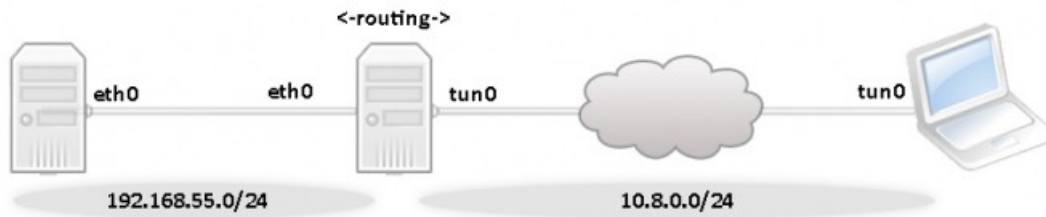
- **client1/client\_II.conf** - dla warstwy drugiej
- **client1/client\_III.conf** - dla warstwy trzeciej
- **client2/client\_II.conf** - dla warstwy drugiej
- **client2/client\_III.conf** - dla warstwy trzeciej

Pliki konfiguracyjne dla klientów nie różnią się znacząco od plików konfiguracyjnych serwera. Na samym początku ustawiamy używany interfejs, tryb pracy jako klient. Następnie ustawiamy adres serwera, port, certyfikaty(certyfikat CA, certyfikat klienta), klucz klienta. Tutaj też jest szereg przydatnych rzeczy jak szyfrowanie, czy też podtrzymanie interfejsu w stanie up przy restarcie.



## 8 Realizacja przy wykorzystaniu połączenia TUN

TUN (network TUNnel) - tryb routowania, tworzy osobną prywatną podsieć która łączy się z siecią po stronie serwera ustalając statyczny routing. Schemat działania połączenia TUN przedstawiony jest na poniższym obrazku.



Przygotowany przez nas skrypt startowy *VagrantFile* ma zmienną *MODE* która domyślnie ustawia konfigurację Open VPN na połączenie w trybie TUN. ( *MODE* = : *tun* )

Tabela 1: Tabela interfejsów dla połączenia tun

Nazwa maszyny	Eth1	Eth2
OVPN Server	192.168.1.10	10.0.10.10
OVPN Client1	192.168.1.20	10.0.20.20
OVPN Client2	192.168.1.30	10.0.30.30
Local Client1	10.0.10.11	-
Local Client2	10.0.20.21	-
Local Client3	10.0.30.31	-
Attacker	192.168.1.50	-

Przekazywanie pakietów między interfejsami:

```
1 sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
2 sudo /sbin/iptables -t nat -I POSTROUTING -s $INTERNAL_NETWORK -j MASQUERADE
```

Instalacja dodatkowych programów:

```
1 sudo apt-get install apache2 -y <-na LocalClient1
2 sudo apt-get install nmap -y <-na LocalClient1/2/3
```

Ważne, by po wyłączeniu Vagrant'a oraz jego ponownym uruchomieniu załadować ponownie skrypty:

```
1 vagrant provision
```

Po pomyślnej instalacji (i ewentualnym ponownym przeładowaniu skryptów), można przystąpić do testowania topologii. Początkowo warto wejść na maszynę wirtualną *OVPNServer* będącą serwerem oraz sprawdzić czy widnieje interfejs *tun0*

```
1 test ! -c /dev/net/tun && echo TUN jest niedostępny || echo TUN jest dostępny
2 TUN jest dostępny
```

```
1 /sbin/ifconfig
```

Następnie testujemy połączenie za pomocą tcpdump. Na serwerze uruchamiamy nasłuchiwanie na interfejsie *tun0*:

```
1 tcpdump -i tun0 -vvv
```

Dla przykładu uruchamiamy ping na *OVPNClient1* w stronę serwera i na serwerze zostają przechwycone dane:

```

1 04:10:54.462685 IP (tos 0x0, ttl 64, id 47963, offset 0, flags [DF], proto ICMP (1), length 84)
2    10.10.10.2 > 10.10.10.1: ICMP echo request, id 3177, seq 4, length 64
3 04:10:54.462739 IP (tos 0x0, ttl 64, id 13991, offset 0, flags [none], proto ICMP (1), length 84)
4    10.10.10.1 > 10.10.10.2: ICMP echo reply, id 3177, seq 4, length 64

```

Wydanie polecenia ping powoduje odpowiedź *echo – reply*, co sugeruje, że połączenie zostało nawiązane poprawnie.

Dzięki OpenVPN realizowanym za pomocą tun dostajemy także dostęp do innych zasobów sieciowych. Na przykład możemy wykorzystać OpenVPN do połączenia się z serwerem w firmie i zdalnej pracy z dowolnego miejsca, dzięki czemu mamy dostęp do jego zasobów - serwera WWW, plików, baz danych itp. Aby zaprezentować taką możliwość wykorzystaliśmy LocalClient 1 który znajduje się w sieci lokalnej OVPN Server aby uruchomić na nim serwer Apache.

Aby sprawdzić połączenie skorzystamy z oprogramowania NetCAT na OVPNCClient1 by przesłać przykładowe dane i sprawdzić czy otrzymamy odpowiedź z serwera apache:

```

1 nc 10.0.10.11 80
2 Test polaczenia z apache

1 HTTP/1.1 400 Bad Request
2 Date: Thu, 18 Jun 2015 10:23:21 GMT
3 Server: Apache/2.4.10 (Debian)
4 Content-Length: 301
5 Connection: close
6 Content-Type: text/html; charset=iso-8859-1
7
8 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
9 <html><head>
10 <title>400 Bad Request</title>
11 </head><body>
12 <h1>Bad Request</h1>
13 <p>Your browser sent a request that this server could not understand.<br />
14 </p>
15 <hr>
16 <address>Apache/2.4.10 (Debian) Server at 127.0.1.1 Port 80</address>
17 </body></html>

```

Pomimo tego że serwer apache znajduje się w odrębnej sieci lokalnej udało nam się nawiązać połączenie z OVPNCClient1.

Połączenie TUN zapewnia bezpieczną wymianę informacji. W tym przypadku do wymiany pakietów użyliśmy protokołu TCP. Aby sprawdzić jak będzie wyglądała komunikacja dwóch klientów odbywająca się przez szyfrowany tunel, należy zalogować się na maszynę wirtualną o nazwie *Attacker*. Następnie włączamy nasłuchiwanie pakietów na interfejsie *eth1* wykonując polecenie:

```

1 sudo tcpdump -i eth1 -vvv

```

Aby zaobserwować wymianę pakietów należy oczywiście wygenerować ruch pomiędzy dwoma miejscami w topologii.

Otrzymane dane ze sniffera wyglądają następująco:

```

1 20:31:32.549092 IP (tos 0x0, ttl 64, id 49360, offset 0, flags [DF], proto TCP (6), length 187)
2    192.168.1.20.48075 > 192.168.1.10.openvpn: Flags [P.],
3    cksum 0xeb41 (correct), seq 135:270, ack 136, win 32044, options[nop,nop,TS val 378467 ecr 384687], length 135
4 20:31:32.550621 IP (tos 0x0, ttl 64, id 14318, offset 0, flags [DF], proto TCP (6), length 187)
5    192.168.1.10.openvpn > 192.168.1.20.48075: Flags [P.],
6    cksum 0xd4a5 (correct), seq 136:271, ack 270, win 32038, options[nop,nop,TS val 384937 ecr 378467], length 135

```

Jak widać korzystanie i skonfigurowanie tun'a nie jest zbyt problematyczne. Przedstawiona konfiguracja jest absolutną podstawą a sam OpenVPN posiada znacznie więcej opcji konfiguracyjnych, z których można korzystać w zależności od potrzeb.

## 9 Realizacja przy wykorzystaniu połączenia TAP

TAP - tryb mostkowania (bridge) polega na połączeniu bezpośrednio do sieci LAN serwera otrzymującego IP z tej samej podsieci co reszta urządzeń. Schemat działania połączenia TAP przedstawiona jest na poniższym obrazku.

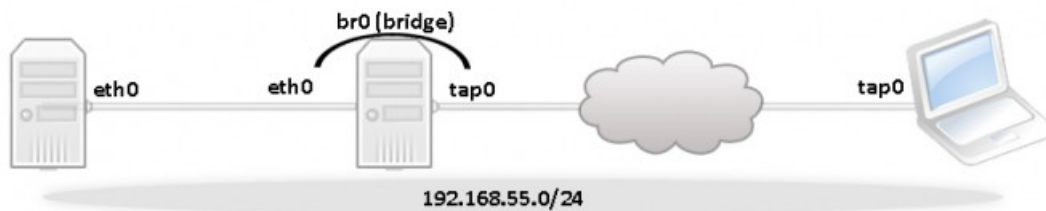


Tabela 2: Tabela interfejsów dla połączenia tun

Nazwa maszyny	Eth1	Eth2
OVPN Server	192.168.1.10	10.0.10.10
OVPN Client1	192.168.1.20	10.0.20.20
OVPN Client2	192.168.1.30	10.0.30.30
Local Client1	10.0.10.11	-
Local Client2	10.0.20.21	-
Local Client3	10.0.30.31	-
Attacker	192.168.1.50	-

Aby uruchomić środowisko w którym wykorzystane jest połączenie TAP należy na początku wejść do pliku *VagrantFile* i zmienić wartość zmiennej *MODE* na : *tap* a następnie uruchomić skrypt startowy.

Podczas pobierania niezbędnych pakietów zostanie pobrany *bridge-utils* - potrzebny do zmostkowania interfejsów.

Po zakończeniu inicjalizacji środowiska nasza topologia wygląda następująco:

Po pomyślnej instalacji, można przystąpić do testowania topologii. Początkowo warto wejść na maszynę wirtualną *OVPNServer* będącą serwerem oraz sprawdzić czy istnieje interfejs *tap0* i czy następuje poprawna wymiana pakietów pomiędzy serwerem a klientami VPNa. Wchodzimy wydając polecenie:

```
1 vagrant ssh OVPNServer
```

Aby sprawdzić istniejące interfejsy wydajemy polecenie:

```
1 /sbin/ifconfig
```

Interesującymi interfejsami w naszym przypadku są *tap0* o którym już wspomnieliśmy oraz *br0* - będący interfejsem bridującym, łączącym na warstwie drugiej *tap0* z *eth2*.

Wydanie polecenia ping skierowanego na którąkolwiek maszynę wirtualną powoduje odpowiedź *echo - reply*, co sugeruje, że połączenie zostało nawiązane poprawnie.

Dzięki zmostkowaniu interfejsów na każdej z maszyn zawierających instancję OpenVPN stworzona została jednolita sieć lokalna o adresie 10.0.0.0/24. Aby zilustrować działanie tak zestawionej topologii zainstalowaliśmy na maszynie *LocalClient1* serwer DHCP, nasłuchujący na interfejsie *eth1* i nadający adresy IP z puli 10.0.0.100 - 10.0.0.110. Gdyby topologia sieci była stworzona przy wykorzystaniu połączenia TUN to wszelka próba komunikacji na warstwie drugiej zostałaby odseparowana pomiędzy lokalnymi sieciami, natomiast w tym przypadku możemy sprawdzić, że jest możliwość otrzymania adresu IP z serwera DHCP na maszynie wirtualnej znajdującej się za OpenVPN'em.

Aby to uzyskać należy wejść na któryś z lokalnych klientów np *LocalClient2* i wywołać polecenie:

```
1 sudo dhclient -i eth1
```

Komenda ta wyśle zapytanie o adres IP, natomiast serwer DHCP przydzieli wolny adres z puli 10.0.0.100 – 10.0.0.110. Aby sprawdzić czy adres IP został poprawnie przydzielony możemy wydać polecenie:

```
1 ip addr show eth1
```

Zostaną wylistowane wszystkie adresy IP przypisane do interfejsu *eth1* a wśród nich któryś z puli DHCP.

Ważną kwestią połączenia jest bezpieczeństwo. Połączenie TAP zapewnia tak samo bezpieczną wymianę informacji jak TUN. W tym przypadku do wymiany pakietów użyliśmy protokołu UDP. Aby sprawdzić jak będzie wyglądała komunikacja dwóch klientów odbywająca się przez szyfrowany tunel, należy zalogować się na maszynę wirtualną o nazwie *Attacker*. Następnie włączamy nasłuchiwanie pakietów na interfejsie *eth1* wykonując analogiczne polecenie jak podczas sniffowania połączenia TUN. Aby zaobserwować wymianę pakietów należy oczywiście wygenerować ruch pomiędzy dwoma końcami topologii.

Otrzymane dane ze sniffera wyglądają następująco:

```
1 22:48:33.214693 IP (tos 0x0, ttl 64, id 28901, offset 0, flags [DF], proto UDP (17), length 177)
2   192.168.1.20.41254 > 192.168.1.10.openvpn: [udp sum ok] UDP, length 149
3 22:48:33.216369 IP (tos 0x0, ttl 64, id 43661, offset 0, flags [DF], proto UDP (17), length 177)
4   192.168.1.10.openvpn > 192.168.1.20.41254: [udp sum ok] UDP, length 149
```

Widzimy, że komunikacja polega na wymianie pakietów UDP jednak nie jesteśmy w stanie stwierdzić co znajduje się w ich wnętrzu. Na tym właśnie polega szyfrowanie połączenia.

## 10 Podsumowanie

Pakiet OpenVPN zapewnia bezpieczne połączenie pomiędzy hostami przez publiczną sieć. Przedstawione przykłady są tylko nielicznymi ciekawymi z pośród ogromnych możliwości. Z racji tego, że adresacja IPv4 została już dawno wyczerpana i coraz szerzej wykorzystywany jest NAT, OpenVPN będzie wykorzystywany coraz bardziej. Obecnie połączenie tunelowana VPN jest najpowszechniej wykorzystywane w pracy zdalnej, jednak jest jeszcze wiele potencjalnych wykorzystania dla takiego połączenia. Oczywiście, że wraz ze wzrostem popularności VPN rośnie liczba zagrożeń dla takiej metody komunikacji, chociażby ze strony hakerów, jednak tunelowanie wydaje się być nadal bezpieczne.

## Literatura

- [1] <https://docs.vagrantup.com/v2/>
- [2] <https://openvpn.net/index.php/open-source/documentation.html>