Akademia Górniczo-Hutnicza

Wydział Fizyki i Informatyki Stosowanej

Bezpieczeństwo w sieci

OpenVPN

Autorzy:
Maciej Kubicki
Karol Kozicki
Filip Kania

Prowadzący:
dr inż. Piotr Kowalski

18 czerwca 2015

Spis treści

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 Głowne 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 uwierzytelnianie stosuje się infrastrukture 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 modyfikacjie 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 pozwola bez względu na to, czy korzysta z systemu Microsoftu, czy któreś 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 VirtualBoksie oraz języku Ruby. Narzędzie to znacznie usprawnia oraz przyspiesza tworzenie maszyn wirtualnych, posiada także wiele funkcji usprawniających konfiguracje 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 funkcje wywołujemy poleceniem:

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

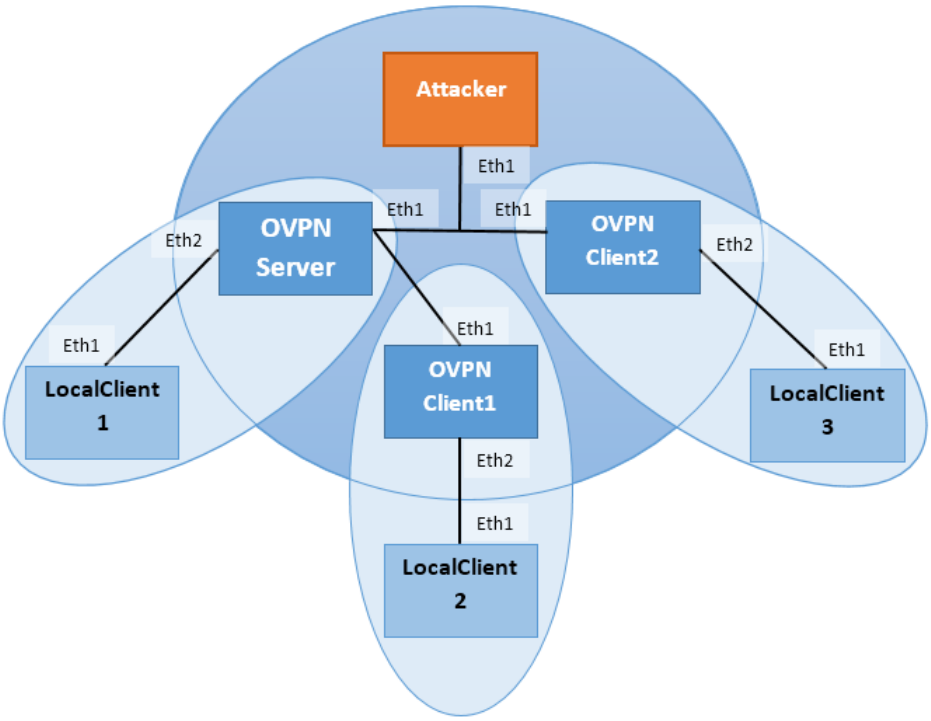
Polecenie to służy do zatrzymania wirtualnych maszyn. Do ponownego uruchomienia tych maszyn należy użyć funkcji *start*. Ostatnią funkcję dostępną w skrypcie wywołujemy przy pomocy:

```
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.

3 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.



3.1 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 połaczony przez OPVNClient1
6. LocalClient3 - Klient OpenVPN połączony przez OPVPNCLient 2
7. Attacker - Atakujący mający możliwośc podsługiwania pakietów wymienianych pomiędzy połączonymi instancjami OpenVPN

4 Pliki dołączone do projektu

4.1 Skrypty instalacyjne

Skrypty użyte w trakcie stawiania wirtualnych maszyn to:

- **all_locals.sh** - skrypt wykonujący się na każdej maszynie LocalClient
- **all_openvpn.sh** - skrypt wykonujący się na maszynach OVPNServer oraz dwóch OVPNClient
- **local1.sh** - skrypt wykonujący się na maszynie LocalClient1
- **openvpn-bridge** - skrypt wykonujący się w trakcie podnoszenia interfejsu *tap*
- **attacer.sh** - skrypt wykonujący się na maszynie Attacker

4.2 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**

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 posiadmy to instalujemy:

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

Następnie kopiujemy katalog Easy-RSA do katalogu OpenVPN:

```
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 serwrze(lokalizacji, czas ważności certyfikatu, wielkość klucza). Wczytujemy plik:

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

Komenda clean czyści poprzednie certyfikaty. Teraz wydajemy polecenie:

```
./build-ca
```

Potwierdzamy lub zmieniamy dane podane w pliku vars.

```
./build-key-server 'nazwa servera'
```

Wykonujemy nasępne polecenie i potwierdzamy jeszcze raz dane z vars, dodatkowo możemu ustalić hasło i na koniec podpisujemy certyfikat. Wygenerowane plik znadjudą się w folderze /keys.

5.2 Certyfikat dla klienta

Do generacji certyfikatu klienta używamy polecenia:

```
./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 głowne serce konfiguracji wszsytkich 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 zmiennej $config.vm.box$ przechowywana jest nazwa i wersja dystrybucji Linuxa którą instalujemy na każdej z maszyn wirtualnych. Jedyną ważną opcją z punktu widzenia osoby nie zainteresowanej samą istotą konfiguracji jest zmienna $MODE$ zdefuniowana 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 warstwe II - łącza danych(tap) oraz warstwe III - sieci (tun). W zależności od tego wyboru będziemy operowali na ramkach ethernetowych, bądź pakietach IP.

7.1 Konfiguracja Serwera

Pliki konfiguracyjne:

- **server_II.conf** - dla warstwy drugiej
- **server_III.conf** - dla warstwy trzeciej

W plikach konfigurujących serwer 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

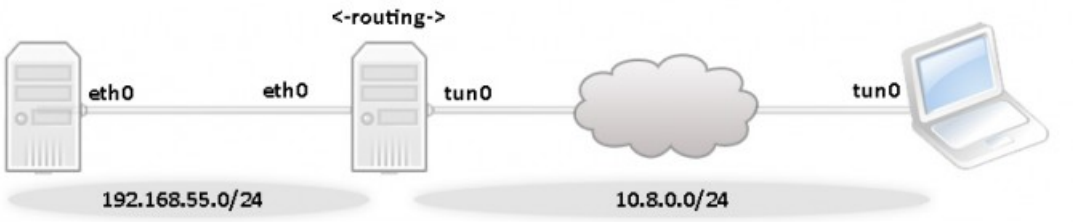
Pliki konfiguracyjne:

- **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 konfigurujących serwer. 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 szerg 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	10.0.0.10
OVPN Client1	192.168.1.20	10.0.0.20
OVPN Client2	192.168.1.30	10.0.0.30
Local Client1	10.0.0.11	-
Local Client2	10.0.0.21	-
Local Client3	10.0.0.31	-
Attacker	192.168.1.50	-

Konfiguracja którą ustawiliśmy w skryptach:
Przekazywanie pakietów między interfejsami:

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

Instalacja dodatkowych programów:

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

Ważne, by po wyłączniu Vagrant'a oraz jego ponownym uruchomieniu załadaować ponownie skrypty:

```
vagrant provision
```

Po pomyślnej instalacji /i ewentualnym ponownym załadowaniu skryptów /, można przystąpić do testowania topologii. Początkowo warto wejść na maszynę wirtualną $OVPNServer$ będącą serwerem oraz sprawdzamy tun jest włączony, i czy widnieje interfejs $tun0$

```
test ! -c /dev/net/tun && echo TUN jest niedostepny || echo TUN jest dostepny
TUN jest dostepny
```

```
/sbin/ifconfig
```

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

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

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

```
04:10:54.462685 IP (tos 0x0, ttl 64, id 47963, offset 0, flags [DF], proto ICMP (1), length 84)
    10.10.10.2 > 10.10.10.1: ICMP echo request, id 3177, seq 4, length 64
04:10:54.462739 IP (tos 0x0, ttl 64, id 13991, offset 0, flags [none], proto ICMP (1), length 84)
    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 sieciowcyh. Na przykład możemy wykorzystać OpenVPN do połączenia się z serwerem w firmie i zdalej 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 nardzędzia do skanowania NetCAT na OVPNClient 1 by przesłać przykładowe dane i sprawdzić czy otrzymamy odpowiedź z serwera apache:

```
nc 10.0.10.11 80
Test polaczenia z apache
```

```
HTTP/1.1 400 Bad Request
Date: Thu, 18 Jun 2015 10:23:21 GMT
Server: Apache/2.4.10 (Debian)
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.10 (Debian) Server at 127.0.1.1 Port 80</address>
</body></html>
```

Jak widać pomimo tego że serwer apache znajduje się w sieci lokalnej udało nam się nawiązać połączenie z OVPN Client1.

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 połączniu bezpośrednio do sieci LAN serwera otrzymując IP z tej samej podsieci co reszta urządzeń. Schemat działania połączenia TAP przedstawiona jest na poniższym obrazku.

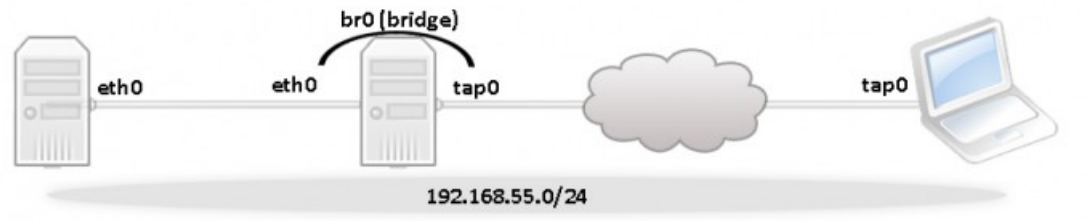


Tabela 2: Tabela interfejsów dla połączenia tun

Nazwa maszyny	Eth1	Eth2
OVPN Server	192.168.1.10	10.0.0.10
OVPN Client1	192.168.1.20	10.0.0.20
OVPN Client2	192.168.1.30	10.0.0.30
Local Client1	10.0.0.11	-
Local Client2	10.0.0.21	-
Local Client3	10.0.0.31	-
Attacker	192.168.1.50	-

Aby uruchomić środowisko w którym wykorzystane jest połączenie TAP należy na początku wejść do pliku $VagrantFile$ i zmienić wartośc 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 maszyneę 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 bridgującym, 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:

```
ip addr show eth1
```

Zostaną wylistowane wszystkie adresy IP przypisane do interfejsu *eth*1 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 masyne wirtualną o nazwie *Attacker*. Następnie włączamy nasłuchiwanie pakietów na interfejsie *eth*1 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:

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