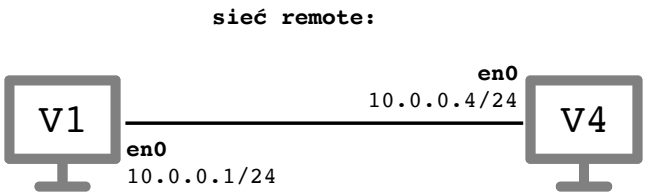
Warsztaty nr 8 z Sieci komputerowych

Konfiguracja początkowa

Celem tej części jest osiągnięcie konfiguracji sieci jak na rysunku poniżej. Maszyna *Virbian1* będzie pełnić rolę serwera, zaś *Virbian4* klienta; powinny być one połączone kartami sieciowymi z wirtualną siecią `remote`.



- Nazwij interfejsy sieciowe na obu maszynach `en0`. Aktywuj te interfejsy i przypisz im adresy IP, takie jak na rysunku.
- Sprawdź wzajemną osiągalność maszyn. Uruchom Wiresharka na obu maszynach nasłuchującego na wszystkich interfejsach.

Tutorial 1

- Na maszynie *Virbian4* poleceniem

  ```
  V4$> telnet 10.0.0.1 7
  ```

  połącz się z serwerem echa maszyny *Virbian1*. Obejrzyj przesyłane pakiety w Wiresharku. Obejrzyj całą komunikację klikając prawym przyciskiem myszy jeden z pakietów należących do połączenia `telnet` i następnie wybierając z menu kontekstowego Wiresharka opcję *Follow* | *TCP stream*.

  Program `telnet` możesz zakończyć naciskając kombinację `Ctrl` + `]` i następnie wpisując `quit`.

- Na maszynie *Virbian1* włącz serwer SSH poleceniem

  ```
  V1#> systemctl start ssh
  ```

  a następnie połącz się z nim z maszyny *Virbian4* poleceniem

  ```
  V4$> ssh 10.0.0.1
  ```

  Na maszynie *Virbian4* wyświetlony zostanie fingerprint klucza publicznego serwera SSH. Możesz wyświetlić fingerprint tego klucza również na maszynie *Virbian1* poleceniem

  ```
  V1$> ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub
  ```

  Sprawdź, czy oba fingerprinty są takie same. Po zaakceptowaniu klucza należy zalogować się podając `user` jako hasło użytkownika `user`. Z jakim portem zostało nawiązane połączenie? Zauważ, że podczas pracy na zdalnej maszynie znak zachęty zawiera czerwony napis `[REMOTE]`.

- Będąc zalogowanym(-ą) na maszynie *Virbian1* przez SSH wykonaj jakieś polecenie, np. wyświetl zawartość katalogu domowego poleceniem `ls -al`. Obejrzyj całą komunikację za pomocą opcji *Follow* | *TCP stream* Wiresharka. Które części komunikacji przesyłane są zaszyfrowane, a które nie? Zamknij połączenie SSH.

- Skonfigurujemy teraz SSH, tak aby możliwe było łączenie się z maszyny *Virbian4* do maszyny *Virbian1* bez podawania hasła. Wygeneruj klucz publiczny i prywatny poleceniem

  ```
  V4$> ssh-keygen
  ```

  Zapisz te klucze w domyślnych plikach (`.ssh/id_ed25519.pub` oraz `.ssh/id_ed25519`). Hasło zabezpieczające klucz pozostaw puste.[1] Obejrzyj właśnie wygenerowane pliki z kluczami.

- Teraz wystarczy dopisać wygenerowany klucz publiczny do pliku `.ssh/authorized_keys` na serwerze SSH (maszynie *Virbian1*). W tym celu skopiuj go poleceniem

  ```
  V4$> scp .ssh/id_ed25519.pub 10.0.0.1:keyfile
  ```

  Na maszynie *Virbian1* dopisz skopiowany właśnie klucz publiczny do pliku `.ssh/authorized_keys` poleceniami

  ```
  V1$> mkdir -p .ssh
  V1$> cat keyfile >> .ssh/authorized_keys
  V1$> rm keyfile
  ```

- Sprawdź, czy działania odniosły skutek, tj. czy możesz zalogować się teraz z maszyny *Virbian4* na maszynę *Virbian1* bez podawania hasła. Polecenie

  ```
  V4$> ssh -v 10.0.0.1
  ```

  wyświetli kolejne etapy nawiązywania połączenia. Obejrzyj je również w Wiresharku. Na końcu zamknij sesję SSH.

- Prostym sposobem zaszyfrowania komunikacji jest tunelowanie strumienia danych w protokole SSH. Na maszynie *Virbian4* utworzymy tunel SSH łączący port `7777` lokalnej maszyny (*Virbian4*) z portem `7` maszyny *Virbian1*. W tym celu wykonaj polecenie

  ```
  V4$> ssh -4 -N -L 7777:localhost:7 user@10.0.0.1
  ```

  i pozostaw je uruchomione. Sprawdź, że po wpisaniu na maszynie *Virbian4* polecenia

  ```
  V4$> telnet localhost 7777
  ```

  odpowiada serwer echa maszyny *Virbian1*.

- Na podstawie Wiresharka i wykorzystując (na obu maszynach) polecenie `sudo ss -tnp` sprawdź, że strumień danych od programu `telnet` na maszynie *Virbian4* do serwera echa na maszynie *Virbian1* jest dzielony na trzy etapy. Każdy z etapów jest osobnym połączeniem TCP; znajdź w wyniku polecenia `ss` oba końce każdego z tych połączeń.

  - Połączenie między portem (jakim?) programu `telnet` na maszynie *Virbian4* a portem `7777` klienta ssh na maszynie *Virbian4*.
  - Połączenie między klientem ssh na maszynie *Virbian4* a serwerem ssh na maszynie *Virbian1*.
  - Połączenie między portem (jakim?) serwera ssh na maszynie *Virbian1* a portem `7` serwera echa na maszynie *Virbian1*.

  Który z tych etapów jest szyfrowany? Jaką wersję protokołu IP wykorzystuje każdy z tych etapów?

- Zamknij program `telnet` i sesję SSH tunelującą połączenie.

[1] W prawdziwych zastosowaniach jest to zazwyczaj zły pomysł.

Tutorial 2

W tej części skonfigurujemy zaporę na maszynie *Virbian1*, wykorzystując moduł `nftables` jądra konfigurowany przez polecenie `nft`. Zaporę można konfigurować interaktywnie za pomocą tego programu, lecz wygodniej jest edytować plik konfiguracyjny `/etc/nftables.conf`.

- Upewnij się, że zawartość pliku `/etc/nftables.conf` na maszynie *Virbian1* jest (z dokładnością do białych znaków) taka, jak poniżej.[2]

```
#!/usr/sbin/nft -f

flush ruleset

table inet filter {
  chain input {
    type filter hook input priority filter;
  }
  chain forward {
    type filter hook forward priority filter;
  }
  chain output {
    type filter hook output priority filter;
  }
}
```

- Wykonaj teraz polecenie

```
V1#> /etc/nftables.conf
```

  Powyższe polecenie należy wykonywać na maszynie *Virbian1 po każdej* edycji pliku `/etc/nftables.conf` (nie będzie to zaznaczone poniżej). Spowoduje to skonfigurowanie zapory zgodnie z instrukcjami z tego pliku.

  Obecnie powyższe instrukcje usuwają wszystkie istniejące reguły a następnie implementują (pustą) konfigurację z sekcji `table inet filter {...}`. Aktualną konfigurację zapory możesz wyświetlić poleceniem

```
V1#> nft list ruleset
```

- Zmodyfikujemy teraz plik `/etc/nftables.conf` na maszynie *Virbian1* tak, żeby pakiety przychodzące do maszyny były odrzucane, zaś wychodzące przepuszczane. Dodatkowo będziemy rejestrować wszystkie odrzucone pakiety do pliku dziennika. W tym celu zmień sekcję `chain input {...}` na następującą:

```
chain input {
  type filter hook input priority filter;
  log
  drop
}
```

- W osobnym terminalu uruchom polecenie

```
V1#> journalctl -f -k
```

[2] Wykorzystamy zdefiniowane przez `nftables` domyślne wartości priorytetów (`filter` i `srcnat`). Więcej informacji na temat ich wartości można znaleźć np. na stronie https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks#Priority_within_hook.

wyświetlające bieżące komunikaty jądra maszyny *Virbian1* (czyli w szczególności odrzucone pakiety).

- Sprawdź, co zapisuje się do pliku dziennika i co wyświetlane jest w Wiresharku, gdy:
  - pingasz z maszyny *Virbian4* maszynę *Virbian1*,
  - pingasz z maszyny *Virbian1* maszynę *Virbian4*.

  W którym przypadku zatrzymywane są komunikaty *ICMP Echo Request*, a w którym komunikaty *ICMP Echo Reply*?

- Zaktualizuj konfigurację zapory dopisując do sekcji `chain input {...}` regułę wpuszczającą pakiety należące do już nawiązanych połączeń. Odpowiednia część pliku konfiguracyjnego powinna teraz wyglądać następująco:

```
chain input {
  type filter hook input priority filter;
  ct state established,related accept
  log
  drop
}
```

  Zwróć uwagę na kolejność reguł: zaakceptowane pakiety nie będą rejestrowane. Które z pingów z poprzedniego punktu będą teraz działać i dlaczego?

- Sprawdź, że nadal nie jest możliwe pingnięcie maszyny *Virbian1* z maszyny *Virbian4*. Co więcej, nie jest nawet możliwe pingnięcie maszyny *Virbian1* z niej samej:

```
V1$> ping 127.0.0.1
```

  ani też połączenie z portem echa lokalnej maszyny:

```
V1$> telnet 127.0.0.1 7
```

  Aby to naprawić, wpuść wszystkie połączenia lokalne oraz komunikaty *ICMP Echo Request* z zewnątrz, zmieniając sekcję `chain input {...}` na następującą:

```
chain input {
  type filter hook input priority filter;
  ct state established,related accept
  iif lo accept
  ip protocol icmp icmp type echo-request accept
  log
  drop
}
```

  Składnię reguł możesz sprawdzić np. w tutorialu na stronie https://wiki.archlinux.org/index.php/Nftables.

  Sprawdź, że pinganie maszyny *Virbian1* z maszyny *Virbian4* jest obecnie możliwe. Sprawdź też, że możliwe stało się połączenie z lokalnym portem echa na maszynie *Virbian1*.

- Za pomocą polecenia

```
V4#> nmap -A -T4 10.0.0.1
```

  sprawdź, jakie porty są dostępne do komunikacji na maszynie *Virbian1*. Przeczytaj uważnie wyświetlane informacje. Oglądając pakiety w Wiresharku i komunikaty o zablokowanych pakietach w pliku dziennika sprawdź, z jakimi portami usiłował połączyć się `nmap`.

Uwaga: powyższe polecenie będzie miało inny efekt, jeśli zostanie wywołane z uprawnieniami zwykłego użytkownika.

- Włącz teraz w zaporze możliwość łączenia się z portem SSH i portem echa, zmieniając sekcję `chain input {...}` na następującą:

```
chain input {
  type filter hook input priority filter;
  ct state established,related accept
  iif lo accept
  ip protocol icmp icmp type echo-request accept
  ct state new tcp dport 7 accept
  ct state new tcp dport 22 accept
  log
  drop
}
```

  Sprawdź, że połączenie SSH z maszyny *Virbian4* do maszyny *Virbian1* jest teraz możliwe. Ponownie wykonaj skan portów poleceniem `nmap` i porównaj wyniki.

Tutorial 3

W tej części skonfigurujemy tunel VPN przy użyciu narzędzia WireGuard.[3] W odróżnieniu od tunelowania SSH, WireGuard tworzy wirtualny interfejs sieciowy, przez który pakiety IP przesyłane są w zaszyfrowanych datagramach UDP.

Adresy IP utworzonego tunelu będą należeć do sieci `172.16.0.0/24`. Będzie to `172.16.0.1` dla maszyny *Virbian1* i `172.16.0.4` dla maszyny *Virbian4*.

- Przed właściwą konfiguracją należy otworzyć odpowiedni port w zaporze maszyny *Virbian1*. Serwer WireGuarda domyślnie nasłuchuje na porcie UDP 51820. Zmodyfikuj plik `/etc/nftables.conf` na maszynie *Virbian1*, dopisując do sekcji `chain input {...}` następującą regułę (przed wierszami `log` i `drop`):

```
ct state new udp dport 51820 accept
```

  Nie zapomnij wykonać pliku `/etc/nftables.conf` po jego edycji.

- Na obu maszynach wygeneruj pary kluczy WireGuarda:

```
V1$> wg genkey > v1-privkey
V1$> wg pubkey < v1-privkey > v1-pubkey
V4$> wg genkey > v4-privkey
V4$> wg pubkey < v4-privkey > v4-pubkey
```

  Obejrzyj zawartość plików `v*-privkey` i `v*-pubkey`.

- Wymień klucze publiczne: z maszyny *Virbian4* skopiuj swój klucz publiczny na maszynę *Virbian1* i następnie pobierz jej klucz publiczny za pomocą SCP:

```
V4$> scp v4-pubkey 10.0.0.1:
V4$> scp 10.0.0.1:v1-pubkey .
```

[3] Jeśli masz starszą wersję Virbiana, konieczne będzie doinstalowanie pakietu `wireguard` za pomocą `apt`.

- Skonfiguruj interfejs WireGuarda na maszynie *Virbian1*. W tym celu wykonaj następujące polecenia:

```
V1#> ip link add wg0 type wireguard
V1#> ip link set up dev wg0
V1#> ip addr add 172.16.0.1/24 dev wg0
V1#> wg set wg0 private-key v1-privkey listen-port 51820
V1#> wg set wg0 peer $(cat v4-pubkey) allowed-ips 172.16.0.4/32
```

  Jeśli pomylisz się w powyższych poleceniach, można usunąć cały interfejs poleceniem `ip link del wg0` i skonfigurować go od początku.

  Ostatnie polecenie określa, że maszyna *Virbian1* będzie akceptowała przez ten tunel pakiety od klienta WireGuarda posługującego się kluczem prywatnym `v4-privkey` (tj. pasującym do `v4-pubkey`) i adresem IP `172.16.0.4`.

  Wyświetl aktualny stan wirtualnego interfejsu i tunelu poleceniami

```
V1$> ip addr show wg0
V1$> ip route
V1#> wg show
```

- Skonfiguruj interfejs WireGuarda na maszynie *Virbian4*:

```
V4#> ip link add wg0 type wireguard
V4#> ip link set up dev wg0
V4#> ip addr add 172.16.0.4/24 dev wg0
V4#> wg set wg0 private-key v4-privkey
V4#> wg set wg0 peer $(cat v1-pubkey) \
        allowed-ips 172.16.0.0/24 endpoint 10.0.0.1:51820
```

  Parametr `endpoint` wskazuje adres i port, pod którym maszyna *Virbian4* będzie szukać serwera WireGuarda. Wyświetl aktualny stan tunelu poleceniem `wg show`.

- Sprawdź łączność przez tunel poleceniami

```
V4$> ping 172.16.0.1
V4$> telnet 172.16.0.1 7
```

  a następnie ponownie wykonaj `wg show` na obu maszynach. Co zmieniło się w wyświetlanych informacjach? Zwróć uwagę na pola *latest handshake*, *transfer* oraz *endpoint*.

- W Wiresharku zaobserwuj pakiety przesyłane podczas wykonywania poleceń `ping` i `telnet`. Zwróć uwagę, że widać tam pakiety rozszyfrowane (w sieci `172.16.0.0/24`) i zaszyfrowane (w sieci `10.0.0.0/24`). Interesują nas te drugie.
  - Jaki protokół warstwy transportowej i jakie porty wykorzystuje WireGuard? Jakie są adresy IP nadawcy i odbiorcy?
  - Jakie typy pakietów przesyła WireGuard? Co można odczytać w ich zawartości?
  - Czy utworzony przez WireGuarda tunel jest widoczny za pomocą polecenia `ss -tnp`, tak jak było to w przypadku tunelowania SSH? Dlaczego?

- Wyłącz tunel na maszynie *Virbian4*:

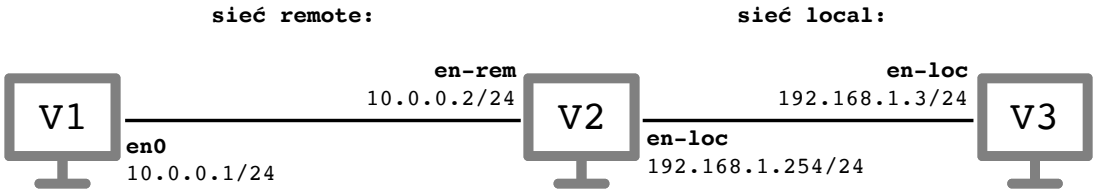
```
V4#> ip link set down dev wg0
V4#> ip link del wg0
```

  Jak zmienił się wynik działania polecenia `wg show` na obu maszynach? Wyłącz maszynę *Virbian4*.

- Konfigurację WireGuarda na maszynie *Virbian1* pozostaw bez zmian; wykorzystamy ją później.

Tutorial 4

- Utwórz teraz dwie dodatkowe maszyny *Virbian2* i *Virbian3* i skonfiguruj je jak na rysunku poniżej. Maszyna *Virbian1* będzie nadal pełnić rolę serwera, *Virbian2* będzie routerem z funkcją NAT, zaś maszyna *Virbian3* będzie maszyną w sieci lokalnej.



  - *Virbian2* powinna mieć dwie karty sieciowe: `en-rem` podłączoną do wirtualnej sieci `remote` i `en-loc` podłączoną do sieci `local`.
  - *Virbian3* powinna mieć jedną kartę sieciową `en-loc` podłączoną do wirtualnej sieci `local`.
  - Przypisz adresy IP do interfejsów sieciowych obu maszyn tak jak na rysunku.
  - Ustaw domyślną bramę na maszynie *Virbian3* wskazującą na maszynę *Virbian2*.
  - Uruchom Wiresharka na dwóch nowych maszynach nasłuchującego na wszystkich interfejsach. Sprawdź, że z maszyny *Virbian2* możesz pingnąć obie sąsiednie maszyny.

- Sprawdź, co się dzieje, gdy z maszyny *Virbian3* próbujesz pingnąć maszynę *Virbian1*. W Wiresharku na maszynie *Virbian1* sprawdź, że pakiety *ICMP Echo Request* docierają do celu, ale odpowiedzi nie wracają. Dlaczego?

- Włącz przekazywanie pakietów IP i skonfiguruj źródłowy NAT na maszynie *Virbian2*. Podobną konfigurację tworzyliśmy parę warsztatów wcześniej, ale teraz posłużymy się plikiem `/etc/nftables.conf`. Na jego końcu (po definicji `table inet filter { ... }`) wpisz następujące instrukcje:

```
table inet my_table {
  chain my_rules {
    type nat hook postrouting priority srcnat;
    ip saddr 192.168.1.0/24 snat ip to 10.0.0.2
  }
}
```

  i wykonaj polecenie `/etc/nftables.conf`.

- Sprawdź, że z maszyny *Virbian3* możesz teraz z powodzeniem skontaktować się z maszyną *Virbian1* wykonując polecenia

```
V3$> ping 10.0.0.1
V3$> telnet 10.0.0.1 7
V3$> ssh 10.0.0.1
```

  Obejrzyj w Wiresharku na maszynie *Virbian2*, jak zmienia się źródłowy adres IP pakietów *ICMP Echo Request* w momencie przejścia przez NAT (każdy pakiet będzie wyświetlony dwukrotnie: raz przed podmianą źródłowego adresu IP i raz po niej).

- Sprawdź, że z maszyny *Virbian1* nie jest możliwe bezpośrednie dotarcie do maszyny *Virbian3* (w szczególności *Virbian1* nie wie, jak trafić do sieci `192.168.1.0/24`). Usuniemy to ograniczenie w następnej części.

Wyzwanie

Na maszynie *Virbian1* sprawdź poleceniem `wg show`, że funkcjonuje ona nadal jako serwer WireGuard z aktywnym interfejsem `wg0` (o adresie `172.16.0.1/24`). Na jakim adresie IP i porcie nasłuchuje ten serwer?

Skonfigurujemy teraz jako klienta WireGuarda maszynę *Virbian3*, która jest ukryta za NAT-em realizowanym na maszynie *Virbian2*.

- Poleceniem `wg genkey` wygeneruj parę kluczy dla maszyny *Virbian3*. Plik z kluczem publicznym nazwij `v3-pubkey`.
- Korzystając z SSH wymień klucze publiczne pomiędzy maszynami *Virbian3* i *Virbian1*.
- Na maszynie *Virbian1* dodaj maszynę *Virbian3* jako nowego klienta WireGuard:

  ```
  V1#> wg set wg0 peer $(cat v3-pubkey) allowed-ips 172.16.0.3/32
  ```

- Skonfiguruj interfejs `wg0` na maszynie *Virbian3*, tak jak zrobiliśmy to wcześniej w przypadku maszyny *Virbian4*, ale:
  - adresem IP interfejsu powinno być `172.16.0.3/24` oraz
  - jako klucz prywatny podaj klucz wygenerowany dla maszyny *Virbian3*.
- Spróbuj najpierw zbadać łączność z maszyny *Virbian1*:

  ```
  V1$> telnet 172.16.0.3 7
  ```

  Dlaczego to polecenie nie działa, skoro tunel WireGuard jest skonfigurowany?
- Teraz spróbuj z maszyny *Virbian3* pingnąć maszynę *Virbian1*:

  ```
  V3$> ping -c 1 172.16.0.1
  ```

  Obejrzyj w Wiresharku na maszynie *Virbian2* pakiety WireGuard przesyłane podczas wykonania tego polecenia. Ile wpisów widzisz dla pojedynczego komunikatu *ICMP Echo Request* i odpowiedzi na niego?
  - Jakie adresy źródłowe i docelowe mają pakiety po obu stronach NAT?
  - Jaki protokół i port jest użyty w warstwie transportowej? Co widać w polu danych?

  Porównaj te informacje z wynikiem działania polecenia `wg show` na maszynie *Virbian1*: gdzie w tym wyniku widać dane maszyny *Virbian2* i dlaczego?
- Teraz spróbuj ponownie z maszyny *Virbian1* połączyć się z serwerem echa maszyny *Virbian3*:

  ```
  V1$> telnet 172.16.0.3 7
  ```

  Dlaczego tym razem jest to możliwe, choć przedtem nie było? Dlaczego wystarczyło polecenie `ping` a nie było konieczne wykonanie polecenia `telnet 172.16.0.1 7` z maszyny *Virbian3*?

- Jeśli odczekasz ok. 3 minuty nie przesyłając nic za pośrednictwem WireGuarda, to połączenie przez tunel WireGuard z maszyny *Virbian1* do *Virbian3* znowu przestanie działać. Żeby temu zapobiec możesz na maszynie *Virbian3* uruchomić polecenie

  ```
  V3#> wg set wg0 peer $(cat v1-pubkey) persistent-keepalive 25
  ```

  Po jego uruchomieniu sprawdź w Wiresharku na maszynie *Virbian2*, że maszyna *Virbian3* wysyła co 25 sekund do *Virbian1* komunikat *WireGuard Keepalive*. Dlaczego te komunikaty pozwalają na utrzymanie działającego tunelu?

Materiały do kursu znajdują się w systemie SKOS: https://skos.ii.uni.wroc.pl/.

Marcin Bieńkowski