

Budowa tuneli sieciowych pod Linuxem

Światelko dla pakietów

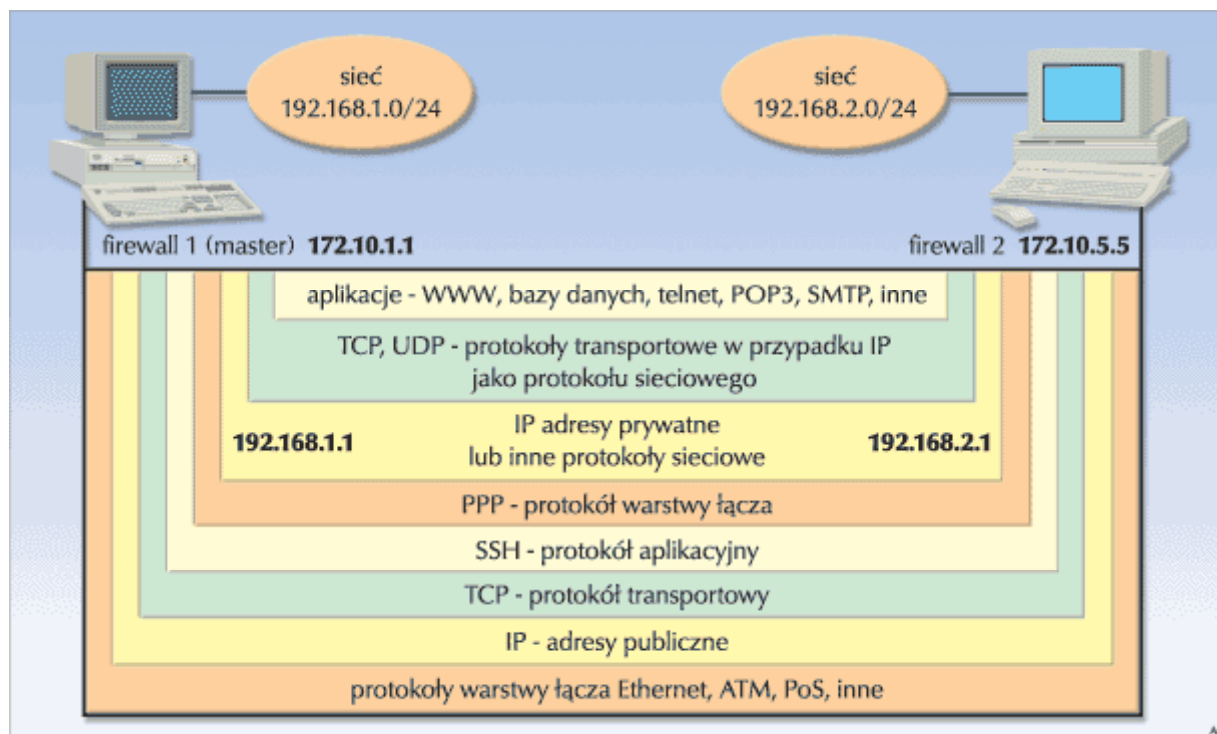
Autor: [Maciej Kosiński](#)

Sieciowe tunele pozwalają w wielu przypadkach obejść techniczne ograniczenia środowisk sieciowych. Korzystając z narzędzi systemu Linux, można na przykład zestawić zaszyfrowane przez SSH bezpieczne połączenie między dwoma sieciami. Innym ciekawym zastosowaniem tej technologii jest przyłączenie odległej stacji do sieci lokalnej firmy.

Tunelowanie możliwe jest dzięki kapsułkowaniu (hermetyzacji), czyli zagnieżdżaniu pakietów jednego protokołu w drugim - w ten sposób można przenieść przez sieć rozległą IP pakiety protokołu sieci lokalnej, np. IPX. W pakietach IP możemy również kapsułkować pakiety IP na przykład w celu przesyłania tunelem pakietów IP multicast poprzez routery, które nie obsługują trasowania multicastowego. Dzięki tunelom może też działać testowa sieć IPv6 wykorzystująca nową, 6-bajtową wersję protokołu IP.

Innym ważnym zastosowaniem tuneli są wirtualne sieci prywatne (VPN - Virtual Private Networks) - przez tunele przesyłane są szyfrowane pakiety IP sieci korporacyjnych. Wprawdzie poufność przesyłanych danych można zapewnić na poziomie poszczególnych aplikacji - np. protokołem SSL (dla transmisji HTTP) czy stosując bezpieczne przekierowanie portów poprzez SSH (Secure Shell) - jednak wiele protokołów usługowych nie ma zaimplementowanych technik autoryzacji i szyfrowania, a indywidualne ich zabezpieczanie przez SSH jest uciążliwe ze względu na małą elastyczność konfigurowania. Ponadto SSH nie umożliwia zaszyfrowania ruchu sieciowego wykorzystującego protokół transportowy UDP. Zabezpieczenie całości transmisji na poziomie protokołu IP, bez wskazywania poszczególnych aplikacji, jest łatwiejsze i bardziej elastyczne. Wadą technologii tunelowania jest narzut spowodowany przesyłaniem dodatkowych nagłówków kapsułkowanych pakietów, który oczywiście zmniejsza wydajność przesyłania danych.

Tunel z klocków - PPP w SSH



Przykład tunelu z zagnieżdżeniem PPP w SSH - bezpieczne, szyfrowane łącze zestawia się kosztem znacznej rozbudowy stosu protokołów

Secure Shell może okazać się pożytecznym narzędziem, jeśli użyje się go do zagnieżdżenia pakietów popularnego protokołu warstwy łącza - PPP. Zestawienie to na pierwszy rzut oka wygląda dziwnie - protokół niższej warstwy tunelowany przez protokół aplikacyjny - ale działa w najprostszym z możliwych zastosowań SSH - zdalnego terminala. Skojarzone są w nim dwie instancje daemona pppd: jedna po stronie klienta SSH inicjująca połączenie,

druga wywołana zdalnie na serwerze SSH. Zobaczmy, jak można zestawić prywatne, szyfrowane łącze PPP między dwoma serwerami uniksowymi pełniącymi rolę routera/firewalla w Internecie.

SSH jest usługą sieciową zbudowaną w architekturze klient-serwer. Umożliwia tworzenie szyfrowanych i uwierzytelnionych połączeń pomiędzy stacjami sieciowymi. Pozwala na zdalny interakcyjny dostęp i/lub wsadowe wykonanie poleceń na odległej stacji analogicznie do znanej z systemu Unix usługi rsh/rlogin. Dzięki mechanizmowi przejmowania lokalnych portów TCP/IP i kojarzenia ich ze wskazanymi usługami zdalnej stacji można za pomocą SSH zabezpieczyć także inne usługi (np. POP3, X11) wykorzystujące protokół TCP.

W SSH zaimplementowano algorytmy szyfrowania: DES, 3DES, BLOWFISH, IDEA, ARCFOUR i TWOFISH (SSH2), a także kompresję przesyłanych danych. Domyślnym algorytmem generowania asymetrycznych kluczy uwierzytelniających użytkownika jest Digital Signature Standard (DSS), w niektórych implementacjach także RSA.

Implementacje SSH wywodzące się z oryginalnego "drzewa" rodowego SSH są dostępne bezpłatnie tylko dla zastosowań niekomercyjnych. Istnieje też wiele implementacji SSH nie podlegających tego typu restrykcjom (m.in. OpenSSH), dostępnych dla większości popularnych systemów operacyjnych.

Najpierw należy odpowiednio skonfigurować SSH na komputerze akceptującym połączenie (przykład konfiguracji klienta ssh1 - p. ramka), czyli przede wszystkim wybrać metodę autoryzacji, np. kluczem publicznym RSA. Następnie na obu końcówkach tunelu zakłada się konta - np. o nazwie vpn - przeznaczone do zestawiania połączeń SSH. Po zalogowaniu się na te konta trzeba programem ssh-keygen (bez parametrów wywołania lub z parametrem -b i długością klucza w bitach) wygenerować dla nich parę kluczy identyfikacyjnych z pustymi hasłami dostępu. Dla ssh1 niezbędne jest skopiowanie wygenerowanego klucza publicznego z konta na końcówce master (klient) na konto vpn końcówki slave (serwer) do pliku \$HOME/.ssh/authorized_keys.

W przypadku ssh2 wygenerowane zostaną pliki id_dsa_1024_a i id_dsa_1024_a.pub (liczbę 1024 można zastąpić inną długością klucza). Na końcówce master w podkatalogu \$HOME/.ssh2 należy wykonać polecenie:

```
echo "IdKey id_dsa_1024_a" > identification
```

Plik id_dsa_1024_a.pub powinno się skopiować na końcówkę slave do podkatalogu domowego konta vpn, nadając mu inną nazwę (np. master.pub) i tam też należy utworzyć plik o nazwie authorization i zawartości:

```
Key master.pub
```

W połączeniu z odpowiednią konfiguracją serwera ssh na końcówce slave pozwoli to na logowanie się bez podania hasła na konto vpn, dzięki czemu nasz tunel będzie uruchamiany wsadowo. Na końcówce pełniącej rolę klienta należy nadać odpowiednie uprawnienia do uruchamiania daemona pppd z konta przeznaczonego dla tuneli. Można to zrobić na kilka sposobów: skorzystać z pakietu sudo i dopisać do /etc/sudoers linię zezwalającą na wykonanie daemona /usr/sbin/pppd bez hasła roota, ale z jego uprawnieniami (vpn jest nazwą naszego konta):

```
vpn ALL=NOPASSWD: /usr/sbin/pppd
```

lub też stworzyć grupę ppp, dopisać do niej konto vpn, nadać odpowiednie uprawnienia serwerowi PPP:

```
chmod 4750 /usr/sbin/pppd
```

a następnie uruchomić pppd na kliencie (master) poleceniem:

```
pppd pty 'ssh2 -l vpn -t slave /usr/sbin/pppd nodetach' 192.168.1.1:192.168.2.1
```

i dodać do tablicy routingu trasy prowadzące do sieci wewnętrznych.

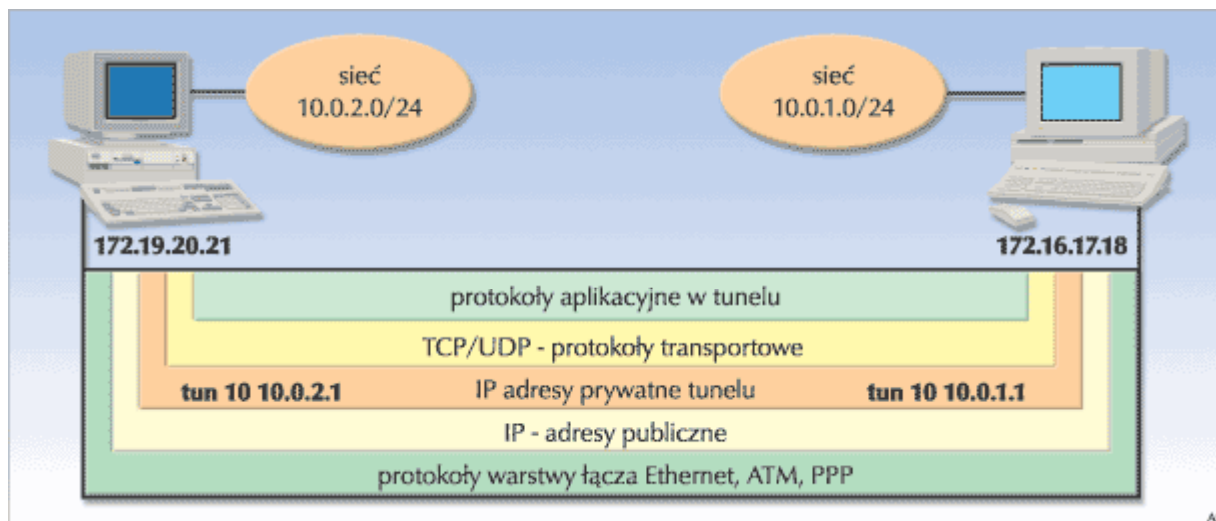
Właściwe tworzenie tunelu odbywa się dopiero podczas wykonania ostatniego polecenia, gdy proces daemona ppp na końcówce master zostaje parametrem pty przyłączony do pseudoterminala, a nie do urządzenia fizycznego, np. portu szeregowego (alokację pseudoterminala umożliwia dopiero wersja 2.3.8 pppd). Aby protokół PPP mógł się komunikować przez przydzielony pseudoterminale, niezbędne jest również wykonanie polecenia uruchamiającego PPP po drugiej stronie łącza. Przy użyciu ssh2 uruchamiamy pppd na końcówce slave na koncie vpn. Parametr -t klienta ssh wymusza alokowanie pseudoterminala, nawet jeśli jako inny parametr podane jest polecenie do wykonania na zdalnym hoście. Parametr "nodetach" dotyczy serwera pppd na końcówce slave - nie pozwala na jego przejście do pracy na drugim planie i wymusza komunikowanie się poprzez pseudoterminale z pppd na końcówce master. Adresy wystarczy podać lokalnej instancji PPP - protokół IPCP powinien doprowadzić do przydzielenia ich po obu stronach łącza.

Zanim uruchomimy pppd na obu stacjach, może się okazać, że niezbędne jest usunięcie niektórych wpisów z plików /etc/ppp/options, szczególnie tych wskazujących na konkretny fizyczny port szeregowy - w przeciwnym wypadku pppd odmówi połączenia przez pseudoterminale. Jeśli na którymś z komputerów protokół PPP jest używany również do połączeń szeregowych, lepiej będzie stworzyć osobne skrypty wywołujące go do konkretnych zadań lub osobne pliki konfiguracyjne.

Tak skonstruowany tunel jest prosty i łatwy w budowie, a komponenty są dostępne na wszystkie platformy uniksowe. Możliwe jest zbudowanie prywatnego, szyfrowanego łącza danych dla dowolnych protokołów sieciowych i

aplikacyjnych poprzez Internet, bezpiecznego na tyle, na ile bezpieczne są stacje na jego końcach oraz SSH jako aplikacja szyfrująca transmisję i uwierzytelniająca połączenia. Wadą jest duży narzut spowodowany zdublowaniem całego stosu protokołów począwszy od warstwy łącza PPP.

IP over IP



Przykład tunelu IPIP - pakiety IP, przekazywane między stacjami o adresach prywatnych, zagnieżdżone są w pakietach IP przygotowanych przez urządzenia dysponujące adresami z puli publicznej

Dzięki tej na pierwszy rzut oka bezsensownie wyglądającej kombinacji możemy pozornie "przenieść" stację sieciową. Uzyskujemy w ten sposób "mobilną" stację - zapewniamy jej stały adres IP niezależnie od aktualnego jej położenia i przydzielanego adresu IP. Ma to oczywiście znaczenie wszędzie tam, gdzie dostęp do zasobów jest możliwy tylko z określonych adresów IP. Od strony fizycznej metoda hermetyzacji pakietów różni się bardzo od opisanej wcześniej. Nie ma w niej zagnieżdżonych ramek protokołu warstwy łącza, a jedynie pakiety IP. Tunel nie jest obsługiwany przez aplikacje użytkownika, lecz przez jądro Linuksa. Od strony administracyjnej wymagane jest więc zbudowanie jądra systemu obsługującego taką hermetyzację - przy kompilacji jądra należy wybrać opcję Networking options/IP tunnelling (tunelowanie może być zintegrowane z jądrem lub dołączane jako moduł).

Protokół do budowy wirtualnych sieci prywatnych (VPN), zaproponowany przez konsorcjum w składzie: Ascend Communications, 3Com, ECI Telematics, U.S. Robotics i Microsoft Corp., złożony jest z dwu podprotokołów: jeden odpowiada za zarządzanie połączeniem, a drugi transmituje dane (najczęściej szyfrowanym łączem), tworząc kanał sieci prywatnej. Kanał kontrolny jest przypisany do portu TCP 1723, zaś kanał transmisji danych używa portu IP numer 47 (czyli opisanego w artykule tunelowania GRE). "Przezroczysta" transmisja w kanale danych jest negocjowana standardowymi mechanizmami protokołu PPP. Również mechanizmy kompresji i szyfrowania używane w kanale danych pochodzą z protokołu PPP. Skorzystanie z serwera PPTP pod kontrolą Windows NT wymaga zainstalowania oprogramowania RAS (Remote Access Server) i wygenerowania kluczy sesji. PPTP był krytykowany za niezbyt bezpieczną implementację, m.in. za nielosowe klucze sesji (Microsoft twierdzi, że ulepszył ten mechanizm), a także brak mechanizmu uwierzytelniania pakietów i łatwy do złamania protokół autoryzacji dostępu MSCHAP. Implementacja PPTP jest dostępna również dla systemu operacyjnego Linux.

Spróbujmy zbudować odpowiednią konfigurację tunelu dla sieci przedstawionej na rysunku. Konfigurację nowego tunelu rozpoczynamy od zainicjowania odpowiednich modułów jądra:

```
insmod ipip.o
```

```
insmod new_tunnel.o
```

Następnie konfigurujemy urządzenie reprezentujące tunel, nadajemy mu własny adres IP oraz wskazujemy przeciwległy koniec:

```
ifconfig tunl0 10.0.1.1
```

```
pointopoint 172.19.20.21
```

a następnie dodajemy odpowiedni wpis w tablicy routingu:

```
route add -net 10.0.2.0 netmask 255.255.255.0 dev tunl0
```

Analogicznie postępujemy z drugim routerem:

```
ifconfig tunl0 10.0.2.1
```

```
pointpoint 172.16.17.18
```

```
route add -net 10.0.1.0 netmask 255.255.255.0 dev tunl0
```

To wszystko. Tunel możemy wyłączyć po dowolnej stronie poleceniem:

```
ifconfig tunl0 down
```

Mamy zatem w naszej sieci lokalnej komputery z sieci odległej, nawet wtedy, gdy dysponujemy niewielką pulą adresową lub pojedynczym publicznym adresem IP (np. dla usługi SDI). Dzięki temu obejmujemy zasięgiem naszej sieci lokalnej również inną, oddaloną fizycznie lokalizację, a posługując się maskaradą IP, możemy równie dobrze korzystać z usług internetowych w obu sieciach lokalnych. Niestety, nie ma szyfrowania kapsułkowanych pakietów. Jeśli zależy nam na poufności przesyłanych tunelem danych, musimy zadbać o ich utajnienie na poziomie poszczególnych aplikacji.

Tunel pod Cisco

Tunele IPIP są specyficzne dla Linuksa i nie są obsługiwane przez inne routery. Jeśli po drugiej stronie tunelu ma znajdować się np. router Cisco, musimy użyć GRE (Generic Routing Encapsulation) - metody hermetyzacji pakietów opracowanej przez Cisco Systems. GRE zapewnia przesyłanie pakietów adresowanych do większego grona odbiorców (IP broadcast i IP multicast), co pozwala na budowę tuneli dla transmisji IP multicast. Może też tunelować pakiety protokołu IPv6.

Spróbujmy zbudować tunel GRE, wykorzystując topologię z poprzedniego przykładu. Niezbędne będzie skompilowanie jądra Linuksa z zaznaczonymi opcjami: Networking options/Kernel/User netlink socket oraz "IP: GRE tunnels over IP" i "IP: broadcast GRE over IP" - może być w formie modułu. Do konfigurowania tunelu potrzebny będzie pakiet iproute2 Aleksa Kuzniecowa. Po stronie routera o adresie IP 172.16.17.18 wykonujemy polecenia:

```
ip tunnel add netb mode gre remote 172.19.20.21 local 172.16.17.18 ttl 255
```

```
ip link set netb up
```

```
ip addr add 10.0.1.1 dev netb
```

```
ip route add 10.0.2.0/24 dev
```

```
netb
```

Pierwsza linia deklaruje urządzenie netb reprezentujące tunel GRE i jego końcówki - zdalną i lokalną, a także czas życia pakietu trasowanego przez tunel. Druga linia aktywuje zadeklarowane wcześniej urządzenie. Trzecia przyznaje adres prywatny z sieci lokalnej końcówce tunelu. Czwarta dodaje do tablicy routingu wpis wskazujący na trasę do docelowej sieci lokalnej po drugiej stronie tunelu. Analogicznie, na routerze o adresie 172.19.20.21 wydajemy następującą sekwencję poleceń:

```
ip tunnel add neta mode gre remote 172.16.17.18 local 172.19.20.21 ttl 255
```

```
ip link set neta up
```

```
ip addr add 10.0.2.1 dev neta
```

```
ip route add 10.0.1.0/24 dev
```

```
neta
```

- www.ajlc.waterloo.on.ca/... - bezpieczny zdalny dostęp
- kubarb.phsx.ukans.edu/... - Virtual Private Networks FAQ
- linas.org/... - Linux Tunnel Mini HOWTO
- www.synack.net/... - skrypt w Perlu realizujący tunelowanie PPP przez SSH dla wersji PPP starszych niż 2.3.8
- www.linux-mag.com/... - artykuł poświęcony budowie VPN dla końcówek linuxowych przy użyciu oprogramowania SecureVPS
- poptop.lineo.com/... - serwer PPTP dla Linuksa
- cag.lcs.mit.edu/... - klient PPTP dla Linuksa

Tunel jest zestawiony po zakończeniu opisanych czynności na obu końcówkach. Wyłączyć go można (np. na pierwszym routerze) poleceniami:

```
ip link set netb down
```

```
ip tunnel del netb
```

```
* * *
```

Tunele to jedna z najpopularniejszych technologii pozwalających obejść ograniczenia występujące w sieciach

routowalnych, spowodowane niedostatkami routerów czy brakiem potrzebnej puli adresów. Przez tunel możemy "przenieść" logicznie stacje sieciowe z jednej sieci lokalnej do innej, rozciągając usługi sieci lokalnej na większy obszar bez konieczności rozbudowy fizycznej infrastruktury. Tunele umożliwiają budowę bardziej zaawansowanych intra- i ekstranetów - ze wszystkimi szykanami, łącznie z dynamicznym trasowaniem. Pozwalają też utajnić transmisję na poziomie protokołu sieciowego. Wszystko to za cenę narzutu związanego z dodatkowym przetwarzaniem nagłówków zagnieżdżonych pakietów. I chociaż komplikują one i tak już skomplikowaną topologię Sieci, to rozwiązują jednak wiele innych poważnych problemów.

Przykładowa zawartość pliku /etc/sshd_config na komputerze umownie traktowanym jako "serwer" (ssh1 i ssh2 są to kolejne wersje protokołu usługowego SSH)

::

```
Port 22
ListenAddress 0.0.0.0
HostKey /etc/ssh_host_key
RandomSeed /etc/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin no
IgnoreRhosts yes
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging yes
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
UseLogin no
Przykładowa zawartość pliku /etc/ssh2/sshd_config:
*:
Port 22
ListenAddress 0.0.0.0
Ciphers AnyStd
IdentityFile identification
AuthorizationFile
authorization
HostKeyFile hostkey
PublicHostKeyFile
hostkey.pub
RandomSeedFile random_seed
```

ForwardAgent yes
ForwardX11 yes
PasswordGuesses 3
MaxConnections 0
PermitRootLogin no
AllowedAuthentications publickey,password
ForcePTTYAllocation no
VerboseMode no
PrintMotd yes
CheckMail yes
UserConfigDirectory "%D/.ssh2"
SyslogFacility AUTH
Ssh1Compatibility yes
RequireReverseMapping yes
UserKnownHosts yes
subsystem-sftp sftp-server