Sieci nie tak znowu lokalne

Konrad Malewski

stopień trudności

Szybki przyrost ilości komputerów na świecie powodują, że zmniejsza się ilość wolnych adresów IP. Najczęściej stosowane rozwiązanie czyli NAT pomimo swoich zalet, posiada również wadę. Jest nią brak możliwości tworzenia połączeń pomiędzy komputerami znajdującymi się w różnych sieciach lokalnych.

Godzina zero. Urządzeń korzystających z publicznych adresów IP ciągle przybywa. Z raportu ([1]) opublikowanego w 2002 roku wynikało, że pula publicznych adresów internetowych zostanie wyczerpana do 2005 roku. Rok 2005 minął spokojnie, a problem adresacji specjalnie nam nie doskwierał. Nowe raporty przesuwają datę apokalipsy na lata od około 2010 do 2026. Pomimo tego, że wg niektórych źródeł w roku 2012 nastąpi już definitywnie koniec świata i problem dostępności IP zejdzie na drugi plan, ustępując pierwszeństwa problemom egzystencjalnym, warto wiedzieć, że to właśnie rozwój lokalnych sieci komputerowych przyczynił się między innymi do przesunięcia momentu, kiedy nie będzie można już podłączyć żadnego nowego komputera do sieci globalnej.

Problem IP i jego rozwiązanie

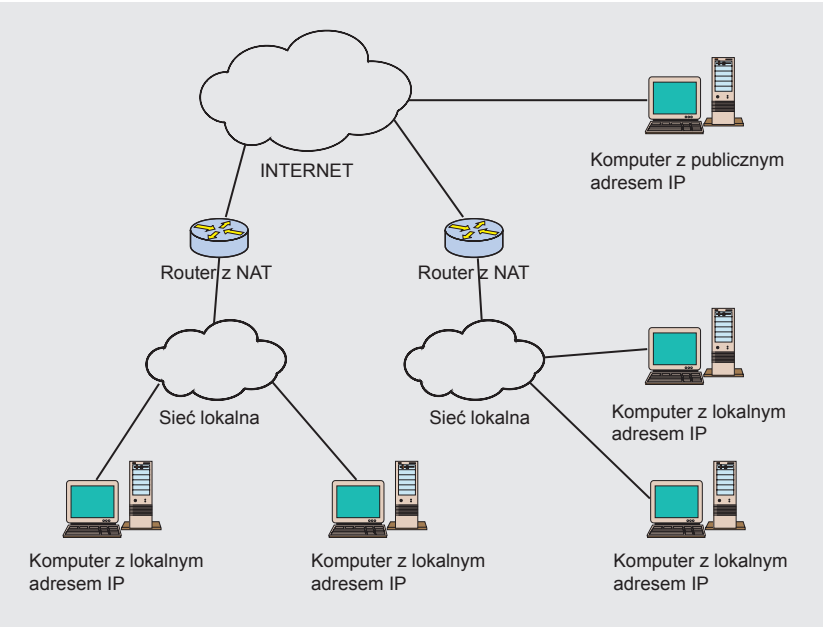
Aby dwa komputery mogły się ze sobą w Internecie skomunikować, muszą posiadać publiczne adresy IP – głosi powszechnie znana teza. Jednak nie jest ona do końca prawdziwa. Z jednej strony prawdą jest, że krążące w Internecie pakiety powinny mieć publiczny adres źródłowy i docelowy, co pociąga za sobą konieczność istnienia dwóch komputerów posiadających zewnętrzne IP – jednego nadającego a drugiego odbierającego. Z drugiej strony istnieją mechanizmy, które potrafią umożliwić maszynom posiadającym lokalną adresację bezproblemową komunikację z dowolnym adresem IP. Jednym z takich mechanizmów jest właśnie NAT (*Network Address Translation*) czyli translacja adresów

Z artykułu dowiesz się...

- Jak działa NAT;
- Jakie są rodzaje NAT i jakie są ich właściwości;
- Jakie są techniki nawiązywania bezpośrednich połączeń UDP oraz TCP pomiędzy lokalnymi sieciami.

Co powinieneś wiedzieć...

- Powinieneś znać model ISO/OSI;
- Powinieneś mieć ogólne pojęcie o zasadzie działania sieci TCP/IP;
- Powinieneś mieć ogólne pojęcie o programowaniu gniazd sieciowych.



Rysunek 1. *Uproszczony model sieci*

sieciowych. NAT nie jest jedynym mechanizmem tego typu. Do umożliwienia surfowania można na przykład użyć serwerów proxy czy też wykorzystać protokół RSIP. Rozwiązania alternatywne często są wykorzystywane w miejscach, w których ten, kto udostępnia Internet chce mieć pełniejszą kontrolę nad przesyłanymi danymi.

Translacja adresów

NAT jest obecnie jednym z najczęściej stosowanych mechanizmów służących do udostępniania połączenia internetowego w sieciach lokalnych. Jest to mechanizm sprawdzony, dość dobrze przetestowany i zrealizowany lepiej lub gorzej w szerokiej gammie routerów sprzętowych. Istnieją oczywiście darmowe implementacje programowe NAT – natd czy iptables. Stosowanie tego rozwiązania posiada następujące cechy:

- NAT oszczędza pule adresów – wiele komputerów może posiadać jeden adres publiczny, pod którym będą widoczne;
- Udostępnianie Internetu dodatkowym maszynom wymaga jedynie przekonfigurowania routera. Nie istnieje potrzeba dokupywania dodatkowych adresów u ISP;
- Stosując NAT w firmie łatwo zmienić dostawcę internetowego;
- NAT *automatycznie* tworzy swojego rodzaju firewall, który nie pozwala na niekontrolowane łączenie się komputerów z internetu z tymi w sieci lokalnej;
- NAT nie wymaga specjalnej konfiguracji aplikacji na komputerach klientów.

Niestety oprócz licznych zalet używając NAT warto pamiętać również o jego niedostatkach:

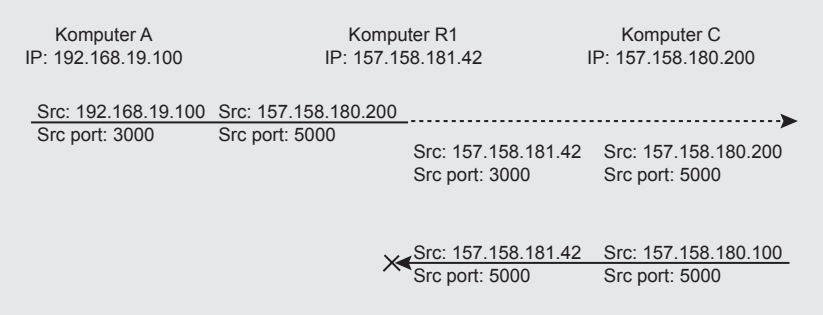
- Stosowanie NAT przy dużej liczbie komputerów i jednym adresie powoduje najrozmaitsze problemy, które niekoniecznie wskazują na konkretną przyczynę;
- Aby działać poprawnie, niektóre usługi wymagają publicznego adresu IP lub/i wymagają dedykowanego portu;
- NAT wymaga więcej zasobów (pamięć, CPU) niż rozwiązanie oparte wyłącznie na routingu;
- Niektóre protokoły nie działają dobrze w środowisku z NAT (np. ftp, ipsec).

Istnieje szereg rodzajów NAT:

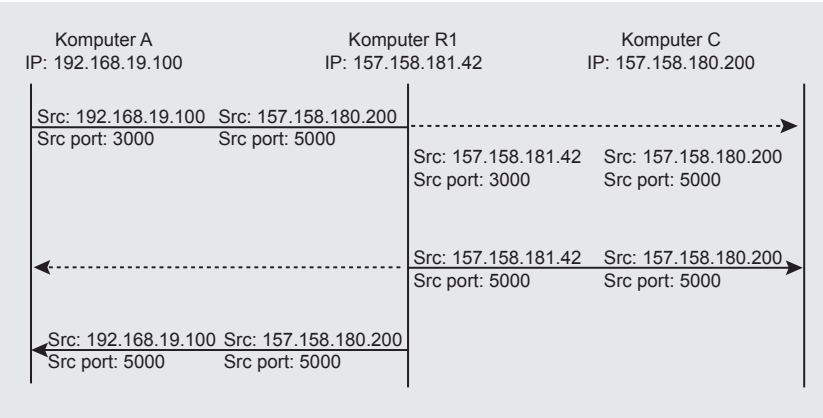
- Jednokierunkowy NAT;
- Dwukierunkowy NAT;
- NAT z translacją portów;
- Dwukrotny NAT.

Różnice pomiędzy poszczególnymi rodzajami NAT oraz zasadę działania najlepiej pokazać za pomocą przykładu. Załóżmy, że w sieci istnieje komputer, który posiada lokalny adres 192.168.19.100 i jest podłączony do Internetu przez NAT, którego adres publiczny to 157.158.181.42. Co się dzieje, gdy komputer z lokalnym adresem IP chce się połączyć np. z komputerem o adresie 157.158.180.200? W przypadku NAT-a komputer z lokalnym IP używa maszyny z usługą NAT jako domyślnego routera, więc chcąc nawiązać jakąkolwiek komunikację poza LAN wyśle do niego pakiety. Przy przejściu pakietu przez router zostanie zamieniony adres źródłowy pakietu na adres publiczny routera. Komputer, do którego dotrze pakiet po zmianie adresu zobaczy połączenie nie z adresu 192.168.19.100, ale z 157.158.181.42.

Jednokierunkowy NAT charakteryzuje się tym, że komputer z poza sieci lokalnej nie może nawiązać połączenia do wewnątrz LAN.



Rysunek 2. *Zasada działania NAT pokazana na przykładzie NAT jednokierunkowego*



Rysunek 3. *Zasada działania dwukierunkowego NAT*

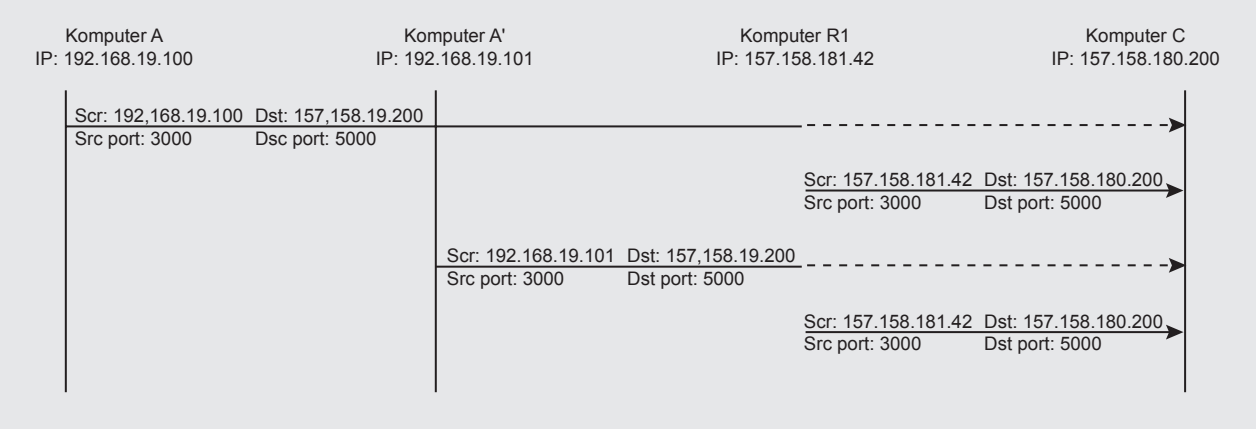
Wszystkie próby nawiązania połączenia do routera są przetwarzane jako próby połączenia z samą maszyną routera.

Taką funkcjonalność umożliwia dwukierunkowy NAT. Jeśli komputer 157.158.180.200 wyśle pakiet do 157.158.181.42, to router zamieni adres docelowy na ten z sieci lokalnej. Komputer lokalny zarejestruje połączenie z adresu 157.158.180.200.

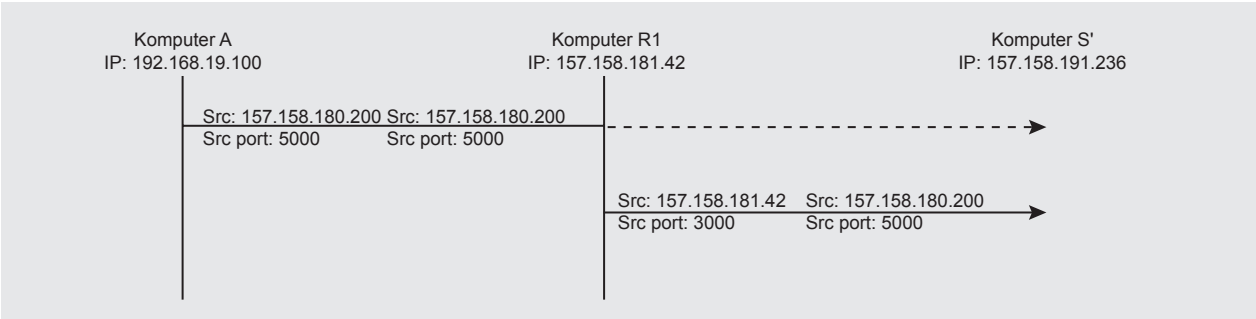
Autorzy poszczególnych implementacji NAT starają się, by pakiety przechodzące przez router były modyfikowane w jak najmniejszym stopniu. I tak np., jeżeli klient NAT w sieci lokalnej próbuje ustanowić połączenie zewnętrzne i używa do tego celu portu źródłowego X, to NAT będzie starał się użyć portu źródłowego X po translacji adresu IP. Aby zrozumieć zasadność istnienia NAT z translacją portów należy rozważyć następujący przypadek: co się stanie jeśli dwóch klientów będzie próbowało otworzyć połączenie zewnętrzne i będą używać tego samego portu źródłowego? NAT powinien jak najmniej ingerować w zawartość przepływających pakietów, ale w tej sytuacji nieingerowanie spowoduje, że po translacji połączenia staną się nierozróżnialne od siebie (oba używałyby portu X jako źródłowego). Jeżeli NAT wykryje taką sytuację, to dokona translacji portu i w przypadku drugiego połączenia NAT oprócz adresu IP zmieni również port źródłowy.

Skoro istnieje możliwość zamiany adresu i portu źródłowego, to nic nie stoi na przeszkodzie, aby zamieniać również adres docelowy. Zamienianie adresu docelowego ma sens między innymi w sytuacji, w której chcemy połączyć dwie sieci, których adresy nakładają się. Jeśli chcielibyśmy umożliwić komunikację pomiędzy tymi dwoma sieciami to albo musielibyśmy zmienić adresy w jednej z nich, albo zastosować podwójny NAT na routerach, zamieniając adresy sieci tak, aby się nie nakładały.

Istnieją podziały NAT ze względu na inne kryteria, jednakże na chwilę obecną poprzestaniemy na tym po-



Rysunek 4. *NAT z translacją portów*



Rysunek 5. *Podwójny NAT*

dziale.

Problem bezpośrednich połączeń

Obecność maszyny pośredniczącej w wymianie pakietów oraz lokalna adresacja znacząco komplikuje klasyczny schemat nawiązywania połączeń pomiędzy komputerami znajdującymi się w różnych sieciach lokalnych. Większość NAT używanych obecnie to NAT z translacją portów. O ile nie ma większych problemów z nawiązaniem sesji komputera w LAN z serwerem posiadającym publiczne IP, o tyle wszystkie żądania połączeń z sieci zewnętrznej zostaną odrzucone przez router łączący komputer z Internetem. Nawet jeśli mamy dostęp do konfiguracji routera i zastosujemy przekierowanie portu do sieci lokalnej, to i tak nie rozwiązuje to wszystkich problemów, ponieważ można przekierować port jedynie do jednego adresu. Czasami przekierowanie jednego portu może być idealnym rozwiązaniem, ale co w przypadku kiedy administrator naszej lokalnej sieci nie ma ochoty zmieniać konfiguracji routera, a w dodatku docelowa maszyna również znajduje się za NAT. Z pomocą przychodzą nam różne techniki, przy użyciu których możemy ustanowić bezpośrednie połączenia i nie będziemy musieli zawracać głowy naszemu administratorowi.

Prosty sposób na P2P

Jeśli jeden z komputerów posiada publiczny adres IP, to problem połączenia w jedną stronę praktycznie nie istnieje. Komputer z publicznym adresem IP jest osiągalny bezpośrednio, więc połączyć się z nim można tak samo, jak z każdym innym w Internecie. Połączenie w drugą stronę sprawia już więcej problemu. Aby je umożliwić należy odwrócić schemat połączenia. A mianowicie komputer z publicznym IP musi zażądać połączenia od komputera zza NAT. Skoro nie może zrobić tego bezpośrednio, to musi istnieć komputer pośredniczący, poprzez który do maszyny za NAT zostanie przesłane żądanie połączenia (Rysunek 6).

Rozwinięciem tego pomysłu, sięgającego jeszcze czasów Napstera, jest przesyłanie danych poprzez pośrednika sieciowego, który posiadałby publiczny adres IP. Klienci, zamiast łączyć się bezpośrednio ze sobą, łączą się z serwerem, który tworzy tunel łączący dwie sieci.

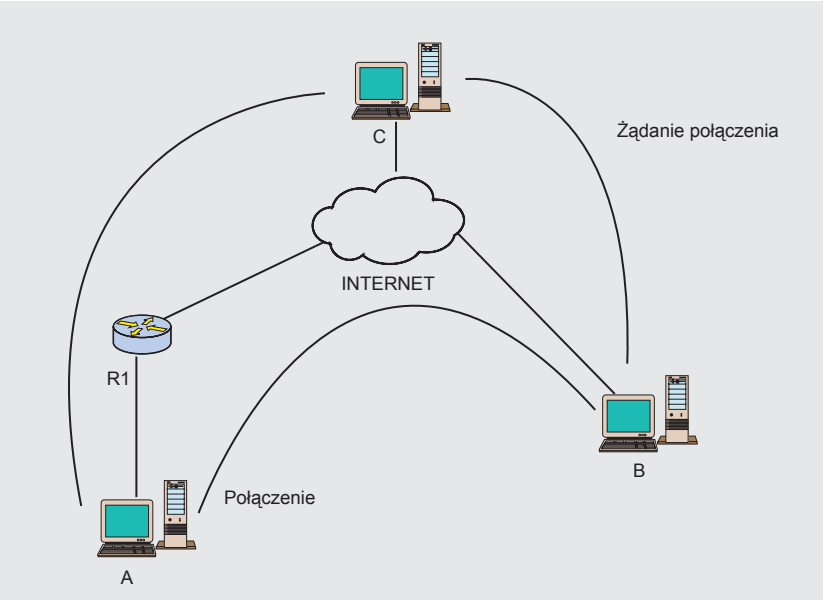
Pomysł ten nosi nazwę *TURN* (skrót ang. *Traversal Using Relay NAT*). Rozwiązanie to jest dość uniwersalne, a na dodatek bardzo skuteczne (Rysunek 7). Posiada jednak pewne wady, a mianowicie wymaga dedykowanego serwera pośredniczącego oraz protokołu wymiany danych oraz informacji sterujących tunelem. To ostatnie wymusza istnienie warstwy pośredniej, z którą komunikowałaby się aplikacja chcąca przesłać dane. Jeżeli rozwiązanie to miałoby zostać wykorzystane na większą skalę np. gdyby na jego bazie chcieć świadczyć usługi jako uniwersalny pośrednik sieciowy, to należy się liczyć również z problemami natury prawnej.

Zestawiamy VPN

Dla własnej potrzeby możemy jednak w bardzo szybki sposób zestawić tunel takiego typu. W najprostszym przypadku musimy jedynie posiadać uprawnienia administratora na maszynie z publicznym adresem IP. Do stworzenia tunelu pomiędzy komputerami A i B użyjemy VPN o nazwie VTUN. VTUN umożliwia stworzenie na komputerze za NAT wirtualnego interfejsu sieciowego, który połączy go z drugim komputerem. Od strony routera łączącego A z internetem tunel widoczny jest jako jedno połączenie A z komputerem C. Dodatkowo VTUN umożliwia szyfrowanie przesyłanych danych. Alternatywnym rozwiązaniem i poniekąd uważanym za bezpieczniejsze jest OpenVPN, który dostępny jest nawet na platformę Windows. Jednak ze względu na prostotę konfiguracji w tym przykładzie zostanie użyty VTUN. Po pomyślnej instalacji VTUN, należy skonfigurować usługę pracującą na maszynie serwera (w tym przykładzie niech będzie to maszyna C), oraz na maszynach klientach (A i B).

Na maszynie C plik konfiguracyjny powinien zawierać:

Plik konfiguracyjny na hoście A jest stworzony analogicznie do hosta B. Różnią się wyłącznie nazwą sekcji (hosta zamiast hostb) oraz adresem IP tunelu (10.1.0.2). Większość opcji w pliku konfiguracyjnym jest samo komentująca się. Obszerną dokumentację jest domyślny plik konfiguracyjny dostarczony wraz z oprogramowaniem. Znaleźć tam można dokładny opis każdego z parametrów.



Rysunek 6. *Schemat odwróconego połączenia*

Mając już przygotowane pliki konfiguracyjne uruchamiamy serwer VTUN na kompuerze C:

```
vtund -s -f vtun-srv.conf
```

oraz na komputerze A:

```
vtund -f vtun-hosta.conf -p hosta
157.158.180.200
```

i na koniec na komputerze B:

```
vtund -f vtun-hostb.conf -p hostb
157.158.180.200
```

Jeśli wszystko poszło dobrze, to serwer C powinien posiadać dwa dodatkowe interfejsy – *tap0* łączący C z A oraz *tap1* łączący C z B.

Teraz wystarczy ustawić reguły routingu pozwalające na komunikację A z B oraz włączyć przekazywanie pakietów pomiędzy interfejsami tap0 i tap1 na hoście C. Routing na A i B możemy ustawić na dwa sposoby. Można do tego celu użyć standardowej komendy route, lub też bardziej zaawansowane ip z pakietu *iproute*. Zaletą używania polecenia ip jest możliwość ustawienia adresu źródłowego, który będzie używany do osiągnięcia danej sieci. Na hoście A wywołujemy:

```
ip route add 192.168.18.0/24 dev
tap1 src 192.168.19.100
```

Na hoście B:

```
ip route add 192.168.19.0/24 dev
tap1 src 192.168.18.200
```

Listing 1. *Host C*

```
echo "1" > /proc/sys/net/ipv4/conf
/tap1/forwarding
echo "1" > /proc/sys/net/ipv4/conf
/tap0/forwarding
echo "1" > /proc/sys/net/ipv4/conf
/tap1/proxy_arp
echo "1" > /proc/sys/net/ipv4/conf
/tap0/proxy_arp
ip route add 192.168.19.0/24
dev tap0
ip route add 192.168.18.0/24
dev tap1
```

Natomiast hosta C uczymy przekazywania pakietów oraz żądań ARP i tego gdzie znajdują się sieci lokalne A i B.

Po wykonaniu powyższego zbioru komend tunel powinien już działać. Wysyłanie pakietów *ping* komendą `ping 192.168.18.200` wywołaną na hoście A powinno spowodować przyjście pakietów ICMP Echo Reply z hosta B.

Niektóre programy (na przykład *nmap*) nie respektują parametru src polecenia ip i używają adresu IP interfejsu, przez który wychodzą pakiety. Pakiety dochodzące do drugiego końca tunelu mają adres źródłowy interfejsu tunelującego czyli w naszym przypadku pakiety docierające z A do B mają adres 10.1.0.2. Aby host B umiał odpowiedzieć na taki pakiet, trzeba skonfigurować mu dodatkową ścieżkę routingu do sieci hosta A.

Sytuacja nie jest tak prosta, jeśli nie mamy uprawnień administratora na hoście C. Jednak i w tym przypadku można zestawić tunel. Do tego celu użyjemy dodatkowo tunelowania SSH. Host B połączy się z hostem C przekierowując swój lokalny port 5000 na port 5000 hosta C:

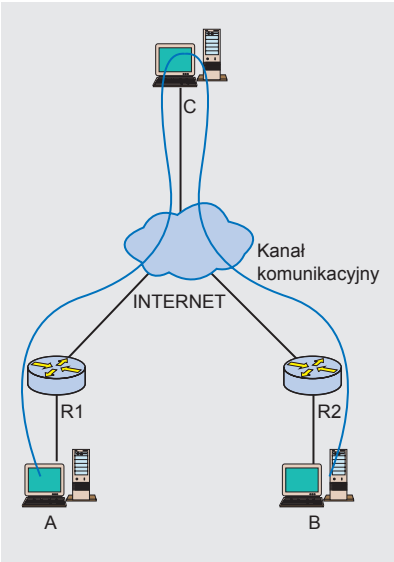
```
ssh -f -N -L5000:localhost:5000 uzytkow
nikB@157.158.180.200
```

Następnie host A wykona połączenie, jednakże przekierowując zdalny port 5000 na swój lokalny:

```
ssh -f -N -R5000:localhost:5000 uzytkow
nikA@157.158.180.200
```

Teraz wystarczy skonfigurować *VTUN* jako serwer na hoście A i użyć *VTUN* w trybie klienta na hoście B. Host B łącząc się na adres *localhost: 5000* łączy się do serwera *VTUN* działającego w drugiej sieci lokalnej. Warto przy tym wiedzieć, że ssh zapewnia dodatkową ochronę przesyłanych danych. De facto w tym rozwiązaniu mamy podwójną ochronę. Pierwszą zapewnia sam mechanizm *VTUN*, drugą *SSH*. Jednak bycie paranoikiem czasami popłaca.

Przedstawione wyżej techniki nie wykorzystywały specyficznych



Rysunek 7. *Zasada działania protokołu TURN*

właściwości NAT-a. Jest to poniekąd ich zaletą, gdyż mamy pewność, że z dużym prawdopodobieństwem, w każdym przypadku uda nam się nawiązać połączenie. Techniki, które polegają na wykorzystaniu zachowania NAT są o wiele ciekawsze, ale nie dają tej pewności.

Drugie podejście

Jedną z takich technik jest *STUN* (*Simple Traversal of User Datagram Protocol* (UDP) *Through Network Address Translators* (NATs)). Jak można wywnioskować z nazwy, sposób na przebijanie NAT dotyczy wyłącznie protokołu UDP. Jednak, jak zostanie pokazane poniżej, technikę tę da się również zastosować wobec protokołu TCP.

Załóżmy, że komputery A i B chcą wysyłać do siebie bezpośrednio datagramy. Aby to osiągnąć łączą się z serwerem pośredniczącym C. Przyjmijmy, że sieć komputera A to 192.168.19.0/24 a sieć komputera B to 192.168.18.0/24. Komputery A i B posiadają adresy w swoich sieciach kolejno 100 i 200.

Interfejsy publiczne routerów R1 i R2 to 157.158.181.42 i 157.158.180.235, a komputer pośredniczący posiada adres 157.158.180.200.

Gdy komputer A wysyła pakiet UDP do komputera C poprzez *router R1*, to router zapamiętuje, że taka sy-

Reklama PSD EXTRA do zrobienia

tuacja miała miejsce i pozwala na dwustronną komunikację pomiędzy komputerami A i C. Na przykład, jeśli A wyśle pakiet z źródłowym portem 3000 i docelowym 5000 do komputera C, to w momencie przechodzenia przez router z *NAT R1*, adres źródłowy (czyli 192.168.19.100) zostanie zamieniony na adres interfejsu zewnętrznego – 157.158.181.42. Jeśli NAT na routerze R1 zamienia również porty źródłowe to zostanie zamieniony również ów port. Aby utrudnić sytuację załóżmy, że R1 jak i R2 zamieniają numery portów połączeń zza NAT. Porty źródłowe pakietów wychodzących są z zakresu 30000-60000. Wracając do naszego pakietu – po translacji adresu źródłowego i portu źródłowego, wędruje on w kierunku maszyny C z ustawionym adresem źródłowym 157.158.181.42 i portem źródłowym 50025. Dodatkowo router R1 zapamiętuje to połączenie. Wszystkie pakiety, które będą pochodzić od maszyny C, a ich port docelowy będzie równy 50025 (port źródłowy równy 5000) zostaną poddane translacji odwrotnej – czyli zostanie przywrócony adres docelowy maszyny A i oryginalny port – 3000. W ten sposób A i C mogą dokonywać dwustronnej komunikacji. Jeżeli B również nawiąże połączenie z C, to C będzie miał łączność zarówno z A i B. Klienci A i B muszą dość często komunikować się z C, aby routery R1 i R2 nie usunęły wpisów w swoich tablicach translacji. Teraz C może poinformować A o adresie i porcie źródłowym B (tym po translacji) i powiadomić B o adresach i portach A. Załóżmy, że żądanie B połączenia na port 5000 komputera C z portu 4000 zostało zamienione tak, że port źródłowy po przejściu przez NAT wynosi 50000. Gdy zaistnieje możliwość komunikowania się A i B za pomocą techniki TURN, komputery mogą dokonać próby połączenia bezpośredniego. W tym miejscu należy trochę poszerzyć naszą wiedzę o mechanizmie NAT. W momencie otwarcia tak zwanego tunelu i zmapowania lokalnego adresu i portu na zewnętrzny adres i port poszczególne NAT inaczej zachowują się w chwili, w której zewnętrzna maszyna (nie należąca do konwersacji) wyśle pakiet do routera łączącego sieć lokalną z Internetem na zmapowany adres i port. Sytuację tą obrazuje rysunek 9.

Listing 2. *Plik konfiguracyjny VTUN na komputerze pośredniczącym C*

```
options {
   port 5000;
   syslog    daemon;
   ifconfig   /sbin/ifconfig;
}
default {
   speed 0;
}
hosta {
   passwd  Ma&^TU;
   type  ether;
   device tap0;
   proto tcp;
   compress  lzo:1;
   encrypt  yes;
   stat  yes;
   keepalive yes;
   up {
   ifconfig "%% 10.1.0.1 netmask 255.255.255.0";
   };
  down {
   # Shutdown tap device.
   ifconfig "%% down";
   };
}
hostb {
   passwd  Ma&^TU;
   type  ether;
   device tap1;
   proto tcp;
   compress  lzo:1;
   encrypt  yes;
   stat  yes;
   keepalive yes;
   up {
   ifconfig "%% 10.2.0.1 netmask 255.255.255.0";
  };
   down {
   ifconfig "%% down";
  };
}
```

Jeszcze więcej rodzajów NAT

Równolegle z zaproponowanym wcześniej podziałem, literatura klasyfikuje NAT ze względu na zachowanie w takich sytuacjach jako:

- Full cone NAT;
- Restricted cone NAT;
- Port restricted cone NAT;
- Symmetric NAT.

Wszystkie mają wspólną cechę. A mianowicie taką, że konsekwencją przejścia pakietu przez router jest zmiana adresu na zewnętrzny, podczas gdy port źródłowy, o ile to jest możliwe, zostanie zachowany. Cechami charakterystycznymi pierwszego jest między innymi to, że host nie należący do konwersacji, chcąc wysłać pakiet do lokalnego komputera, wysyła go na adres zewnętrzny hosta za NAT. Ponadto jeśli host w sieci lokalnej użyje tego samego portu źródłowego do skomunikowania się z innym hostem w Internecie, to NAT nie zmieni portu źródłowego po translacji. W swoim działaniu przypomina częściowo dwukierunkowy NAT, o którym wspomniano wcześniej.

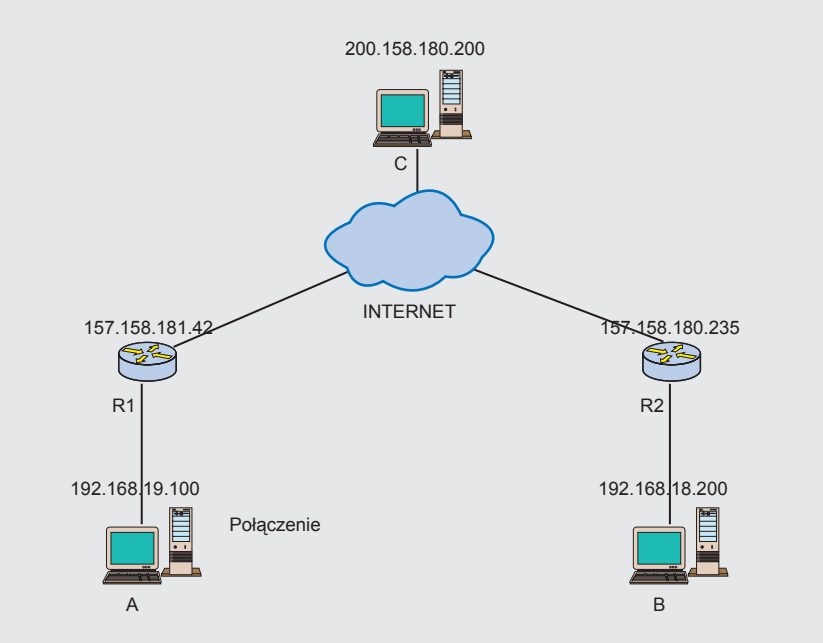
Drugi – *Restricted cone NAT*, zachowuje się tak jak *Full cone NAT* z tą

różnicą, że wysłany przez zewnętrznego hosta pakiet dotrze do sieci lokalnej wyłącznie w przypadku, gdy host z wewnątrz sieci wysłał wcześniej pakiet do zewnętrznego hosta. Port restricted cone NAT dodaje obostrzenie w postaci konieczności zachowania numerów portów. Wysłany przez zewnętrznego hosta pakiet używający portu źródłowego X i portu docelowego Y dotrze do hosta za NAT wyłącznie wtedy, gdy host ten wyśle najpierw pakiet do zewnętrznego hosta, jego port docelowy to X, a jego port źródłowy po opuszczeniu routera NAT będzie zmapowany na Y. Najtrudniejszy do przebicia jest *Symmetric* NAT, gdyż ograniczenie dotyczy w tym przypadku nie tylko adresów, ale także portów źródłowych i docelowych, a wszystkie parametry są unikalne dla każdego połączenia. Tylko komputer, który używa tej samej pary parametrów, odpowiadając pakietem na adres i port źródłowy, skomunikuje się z hostem zza NAT. Dodatkowo, jeśli komputer z sieci lokalnej użyje tego samego portu źródłowego do skomunikowania się z innym adresem, to po opuszczeniu NAT, port źródłowy zostanie zmieniony. Zachowanie wyżej opisanych NAT zobrazowane jest na rysunkach nr 11,12 oraz 13.

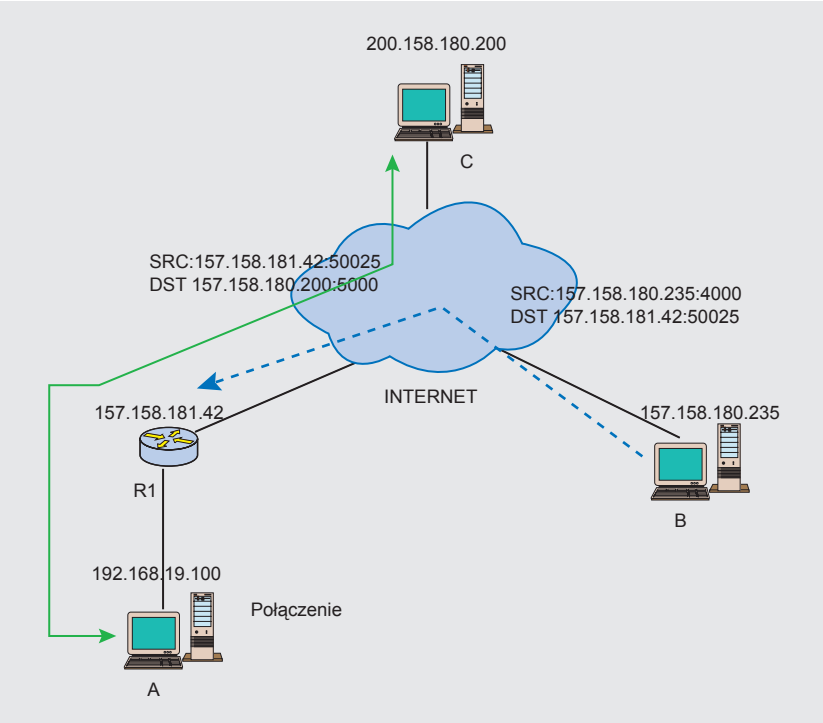
Symetryczny NAT jest często określany jako *not well behaved*. Jednak nawet używając tego typu NAT jesteśmy w stanie tworzyć bezpośrednie połączenia. Polegają one na przewidywaniu portu źródłowego, który zostanie wybrany jako kolejny.

Tunel UDP

Załóżmy, że NAT routerów R1 i R2 to Port *restricted cone NAT*, ponieważ takie są najczęściej spotykane obecnie. *Port restricted cone NAT* ma tą ciekawą właściwość, że jest *well behaved*. Oznacza to ni mniej ni więcej, że tak długo, jak host A będzie używał portu źródłowego równego 3000 do wysłania pakietu, tak długo NAT będzie używał tego samego portu (czyli 50025). Wracając do sytuacji opisanej wcześniej, hosty A i B mają połączenia z C, host A wysyła pakiet w kierunku routera R2 z ustawionym portem źródłowym 3000 i portem docelowym 50000. W momencie translacji – na routerze R1 zostanie zamieniony adres i port źródłowy tak samo, jak w przypadku łączenia się z hostem C. Dodatkowo zostanie odnotowane, aby wpuszczać do wewnątrz ruch pochodzący od routera R2 przychodzący z portu 50000. Router R2 odebrawszy pakiet odrzuci go, gdyż host B jeszcze nie przestawił swojego NAT tak, aby akceptował połączenia z R1. W zależności od konfiguracji routera może zostać wysłany pakiet ICMP port unreachable. Załóżmy, że NAT R2 nie wysłał takiego pakietu. Gdy B wyśle pakiet (używając oczywiście tego samego portu źródłowego) do R1, to ten zamieni adres i port docelowy pakietu



Rysunek 8. *Schemat przykładowej sieci*



Rysunek 9. *Host nie należący do konwersacji łączący się do zmapowanego połączenia na routerze*

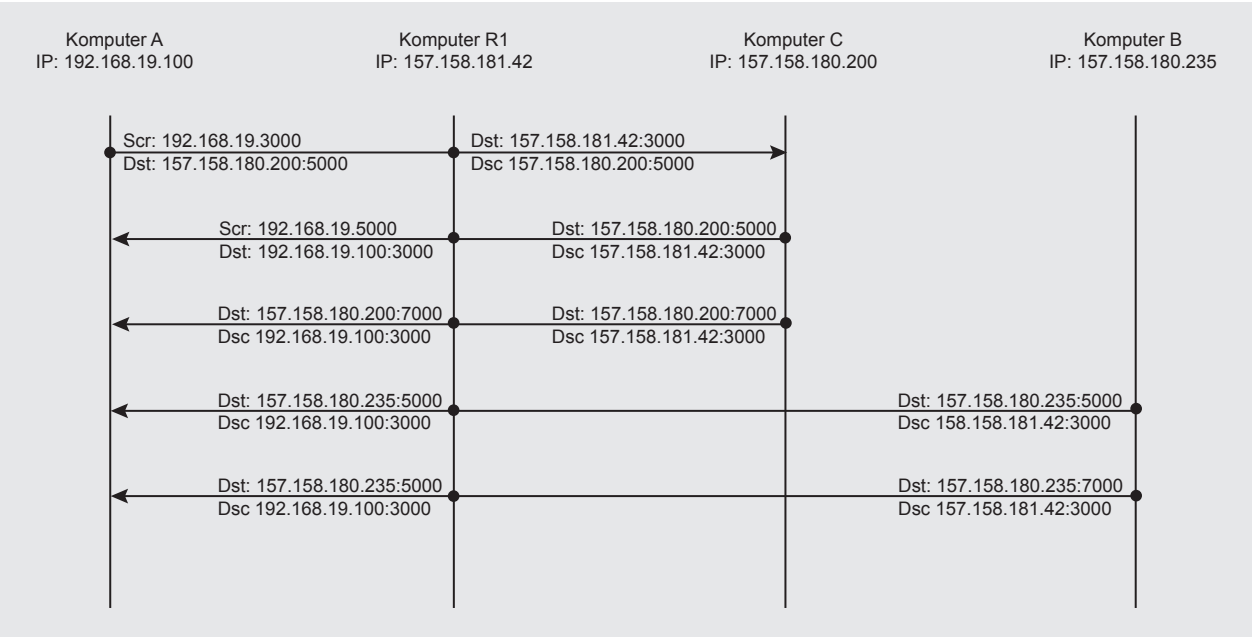
gdyż został wcześniej przeprogramowany przez próbę połączenia się A z B. Teraz A i B mogą się bezpośrednio komunikować wysyłając pakiety UDP na swoje zewnętrzne adresy i porty. W całej komunikacji zaginął wyłącznie jeden pakiet. Jest to cena, jaką należy zapłacić za możliwość bezpośredniego połączenia.

Powyższy schemat ma zastosowanie w przypadku, gdy NAT na R1 i R2 zmieniają porty źródłowe. Sprawa się nieco upraszcza w momencie, gdy NAT nie zamienia portów źródłowych. W tym przypadku nic nie stoi na przeszkodzie, by dokonać próby nawiązania połączenia bez pośrednika. Aby otworzyć *tunel* w NAT R1 należy wysłać z A pakiet w kierunku R2.

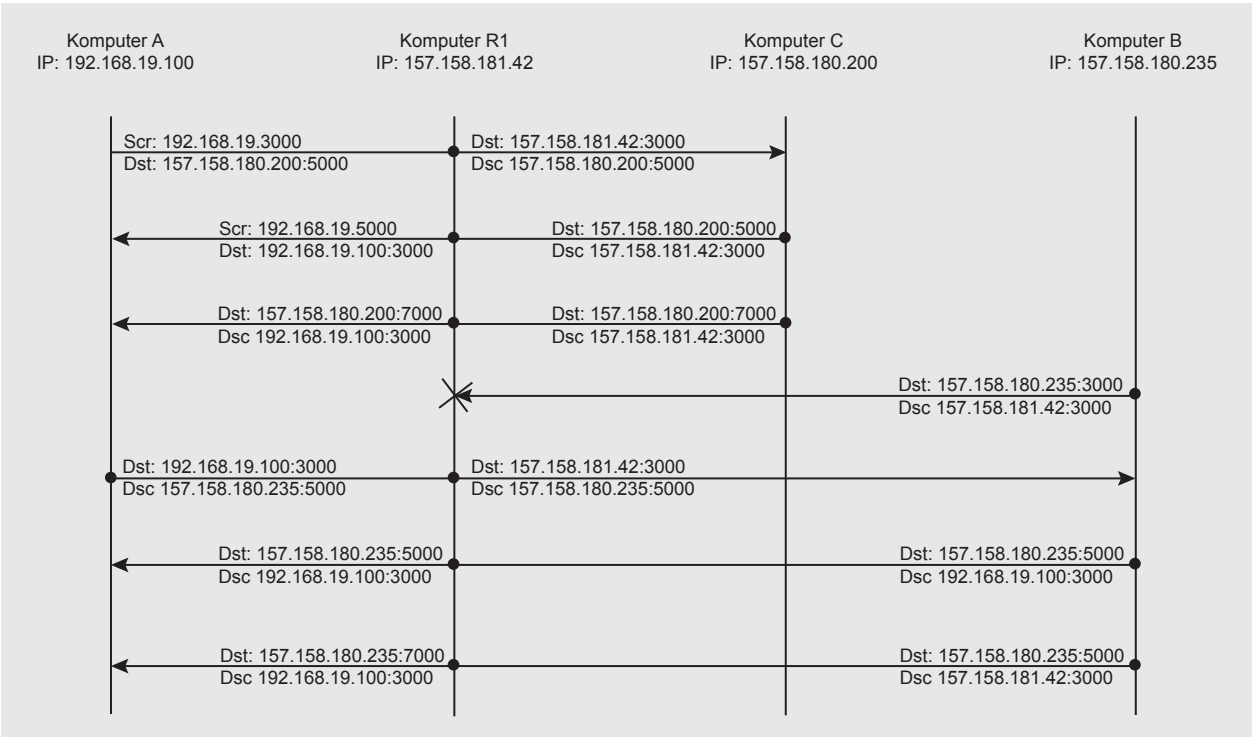
Tunel UDP w praktyce

Do stworzenia tunelu UDP potrzebne będą nam następujące narzędzia: netcat, hping2 oraz skrypt napisany w języku Perl z listingu nr 3.

Skrypt z listingu nr 3 uruchamia-



Rysunek 10. *Działanie Full Cone NAT*



Rysunek 11. *Działanie Restricted Cone NAT*

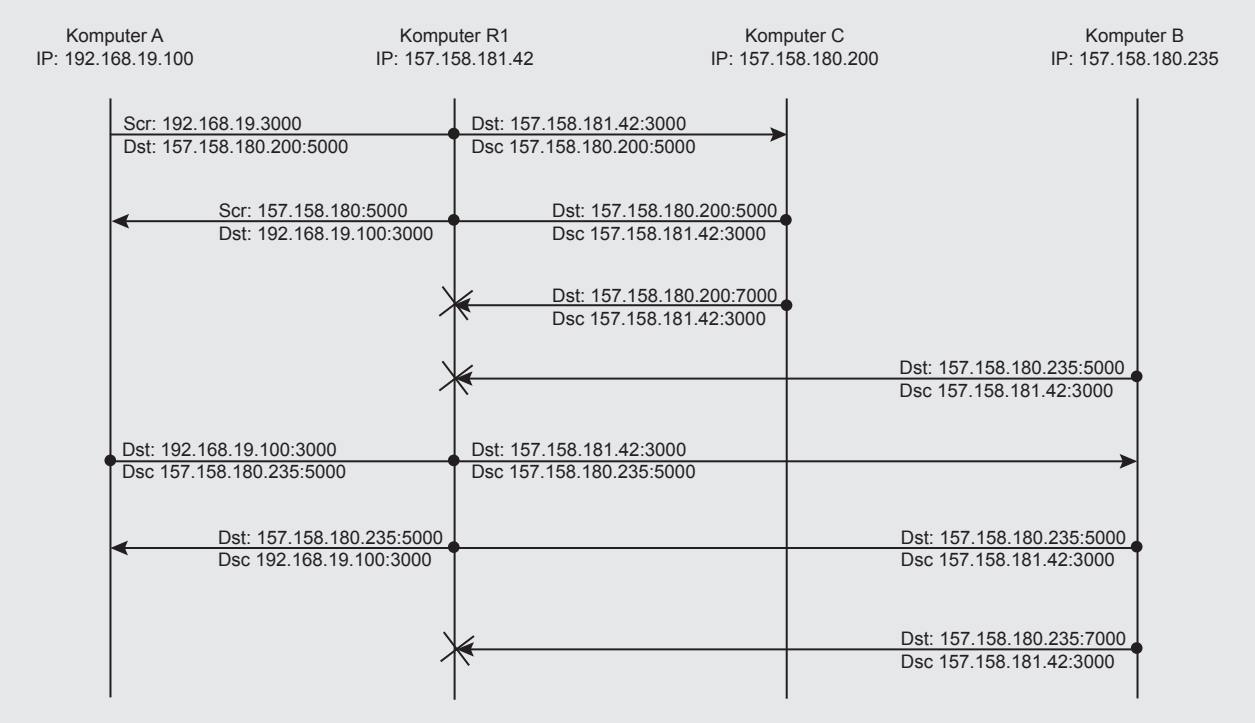
my na komputerze C. Jego zadaniem jest nasłuchiwanie na porcie nr 5000 i wypisywanie na ekran terminala parametrów hostów łączących się z C oraz wiadomości jakie owe hosty wysyłają.

Na komputerze A uruchamiamy *netcata*:

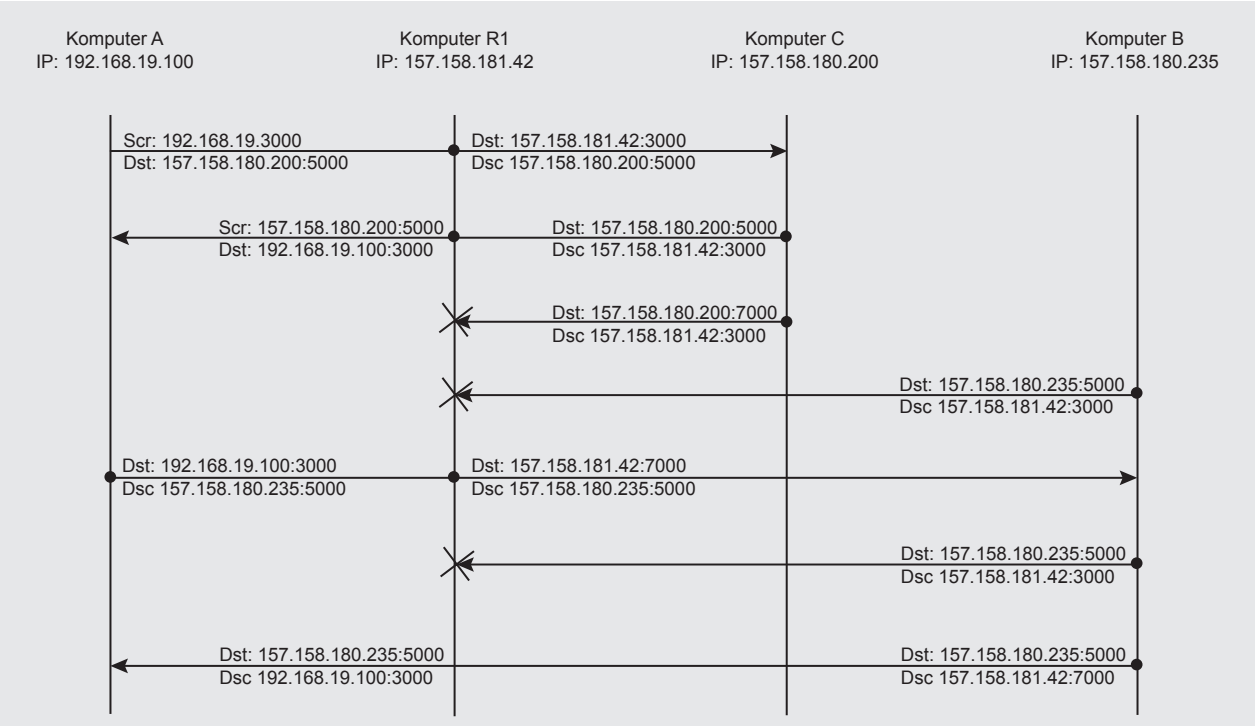
```
netcat -l -u -p 3000
```

Spowoduje to nasłuchiwanie maszyny A na porcie 3000. Zawartość wszystkich pakietów zaadresowanych do A i posiadających port docelowy 3000 zostanie wypisana na ekranie terminala.

Następnie otwieramy tunel w NAT na obu maszynach. Na A musimy użyć portu źródłowego równego 3000, ponieważ na tym porcie netcat



Rysunek 12. *Działanie Port Restricted Cone NAT*



Rysunek 13. *Działanie Symmetric NAT*

nasłuchuje pakietów:

```
hping2 157.158.180.200 -2 -s 3000 -p
5000 -c 1
```

spowoduje to pojawienie się na terminalu C komunikatu

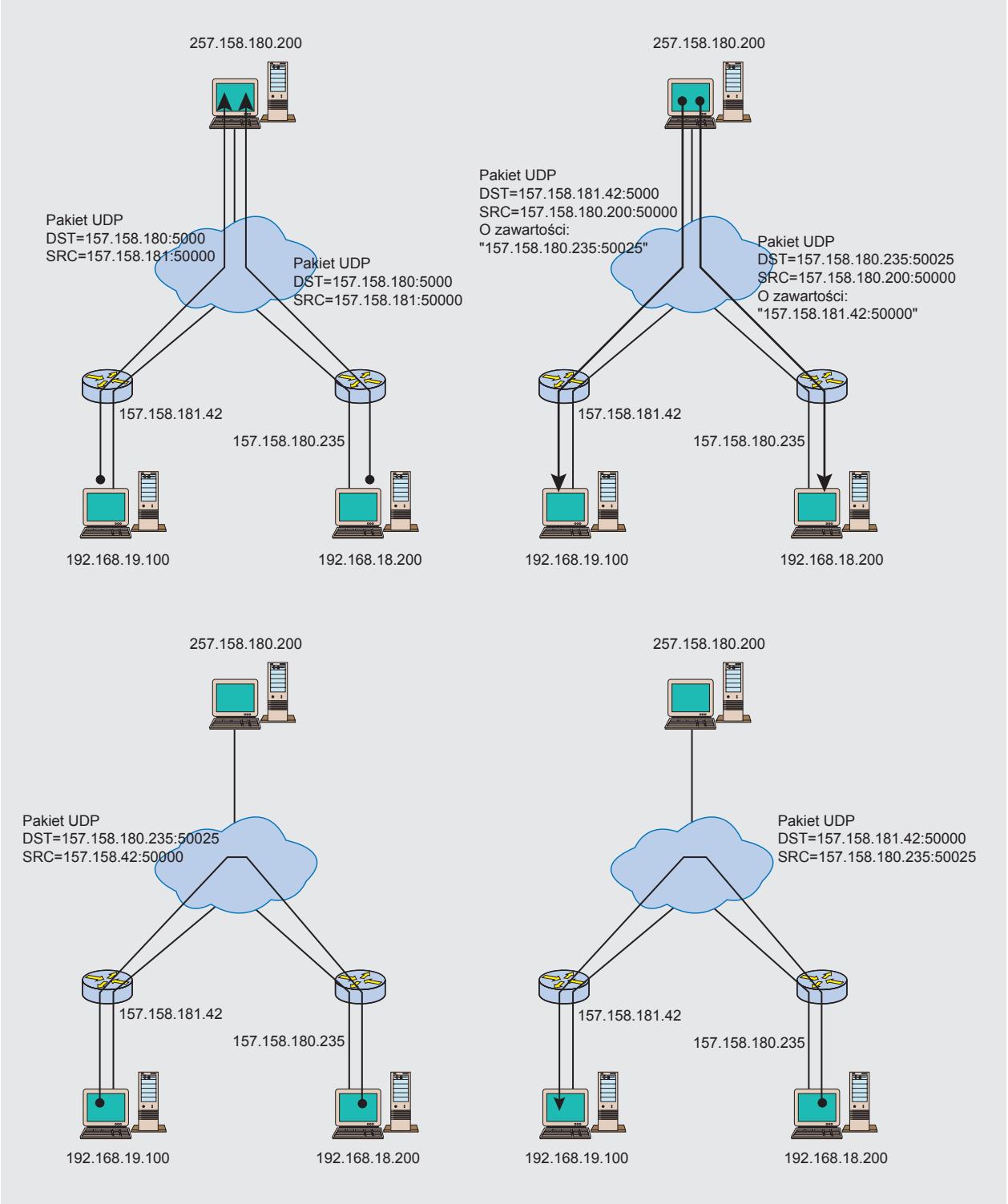
```
[157.158.181.42: 3000]>
```

świadczącego o tym, że pakiet zza NAT dotarł do R1 nie zmienił portu źródłowego. Następnie czynność tą powtarzamy na komputerze B:

```
hping2 157.158.180.200 -2 -s 4000 -p
5000 -c 1
```

co z kolei powinno na C spowodować powstanie komunikatu:

```
[157.158.180.235: 4000]>
```



Rysunek 14. *Poszczególne fazy tworzenia tunelu UDP*

Teraz na komputerze A przestawiamy nat w R1 tak, aby akceptował pakiety od R2:

```
hping2 157.158.180.235 -2 -s 3000 -p
4000 -c 1
```

Na B przestawiamy NAT R2 tak, aby akceptował pakiety od R1:

```
hping2 157.158.181.42 -2 -s 4000 -p
3000 -c 1
```

Teraz możemy na B uruchomić netcata z parametrami:

```
nc -p 4000 157.158.181.42 3000 -u
```

Po wykonaniu ostatniego polecenia, wszystko co wpiszemy w okienko na hoście B pojawi się w okienku netcat hosta A.

Tworzenie tunelu tym sposobem nie jest zbyt użyteczne. Na szczęście powstały narzędzia które automatycznie wysyłają sekwencje wiadomości tworzących połączenie. Dość ciekawym narzędziem jest skrypt perlowy *nat-traverse* dostępny na [13]. Aby osiągnąć identyczny rezultat jak w powyższym przykładzie, wystarczy na maszynie A wykonać polecenie:

```
nat-traverse 3000:157.158.180.235:4000
```

a na maszynie B:

```
nat-traverse 4000:157.158.181.42:3000
```

Program ten posiada jedną ciekawą opcję. Jest nią możliwość wywołania dowolnego polecenia i przekierowanie strumieni – wejściowego i wyjściowego w ten sposób, że wszystko co polecenie wypisze zostanie wysłane przez gniazdo do drugiego komputera. Wszystko co przyjdzie z tunelu zostanie przekierowane jako strumień standardowego wejścia do polecenia. Za pomocą tej opcji można przesyłać katalogi pomiędzy komputerami:

```
hostA# nat-traverse 3000:
157.158.180.235:4000 -cmd=tar cj
katalog
hostB# nat-traverse 4000:
157.158.181.42:3000 -cmd=cat >
katalog.tar.bz2
```

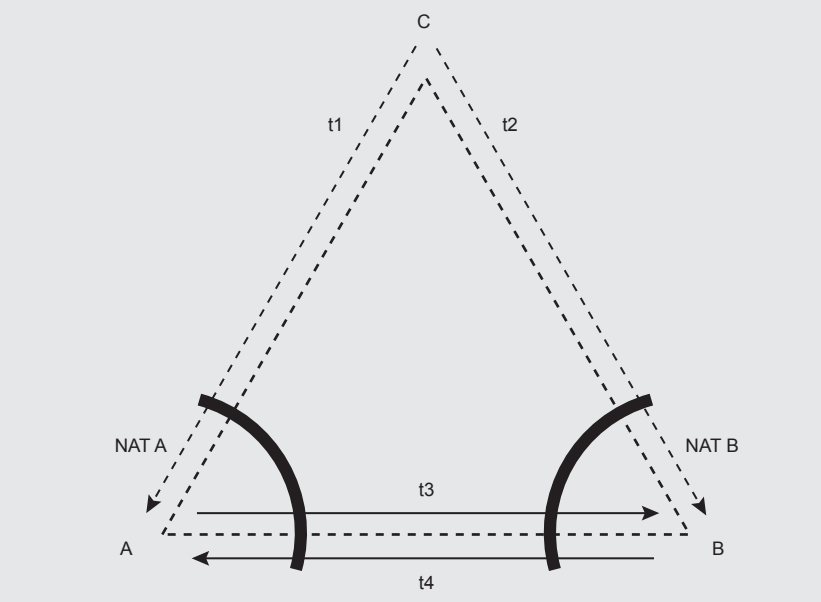
Możemy również za pomocą tego narzędzia przekazywać połączenia TCP. Wystarczy wykorzystać do tego celu program netcat:

```
hostA# nat-traverse 3000:
157.158.180.235:4000 -cmd=nc -lvp 5000
hostB# nat-traverse 4000:
157.158.181.42:3000 -cmd=nc -v
localhost 22
```

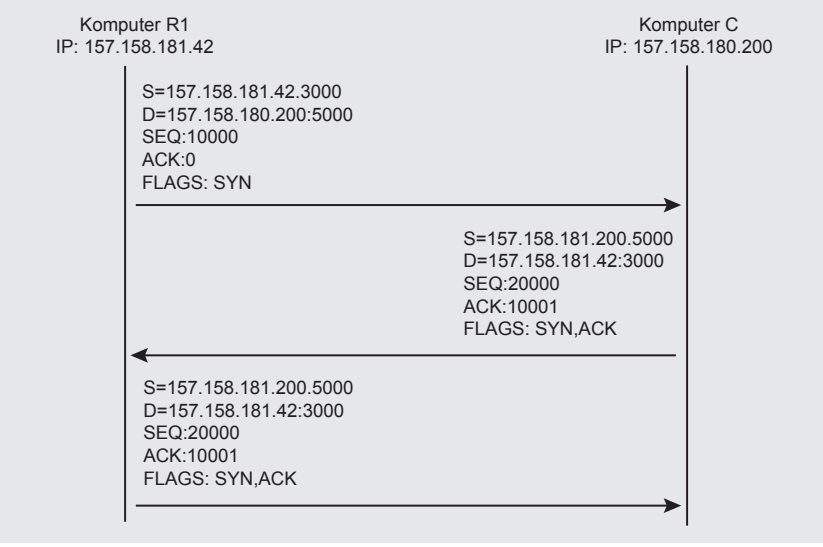
Teraz można połączyć się z hosta A

Listing 3. *Plik konfiguracyjny VTUN na komputerze B*

```
options {
   port 5000;
   timeout 60;
   ifconfig   /sbin/ifconfig;
}
hostb {
   passwd  Ma&^TU;
   type  ether;
   device tap1;
   proto tcp;
   up {
   ifconfig "%%  10.2.0.2
   netmask 255.255.255.0";
   };
   down {
   ifconfig "%% down";
   };
}
```



Rysunek 15. *Czasy transmisji pakietów.*



Rysunek 16. *Trzy fazy nawiązywania połączenia*

Listing 4. *Skrypt w języku Perl dzięki któremu dowiemy się jak mapowane są połączenia w naszym NAT*

```
#!/usr/bin/perl -w
use strict;
use IO::Socket;
my ($sock, $remote, $message);
$sock = IO::Socket::INET->new(LocalPort => 5000, Proto => 'udp') or die
                        "socket: $@";
while ($remote = $sock->recv($message, 1024)) {
   my ($port, $addr) = sockaddr_in($remote);
   my $host_str = inet_ntoa($addr);
   printf "[%s:%5d]> %s\n", $host_str, $port, $message;
} die "recv: $!";
```

do serwera ssh działającego na hoście B poprzez połączenie z interfejsem loopback na port 5000. Za pomocą netcat można również przekazywać połączenia UDP. Skoro można przekazywać połączenia UDP oraz TCP, to nic nie stoi na przeszkodzie aby uruchomić VPN działający poprzez ten tunel. Wystarczy przekazać lokalny port tcp 5000 do zdalnego hosta, na którym działa VTUN oraz połączyć się klientem VTUN do localhost:5000. Należy przy tym pamiętać, że po pierwsze tracimy niezawodność protokołów takich jak TCP a po drugie sam proces enkapsulacji posiada duży narzut danych (TCP w IP w ethernet w UDP).

Jeśli firewall na routerach jest skonfigurowany tak, aby odpowiadać komunikatami ICMP na pakiety UDP których „nie zna", to stworzenie tunelu może się nie powieść, ponieważ niektóre NAT reagują na komunikaty ICMP. Jeśli pakiet hosta A dotrze do NAT B przed wysłaniem przez B UDP w kierunku A, to może dojść do sytuacji, w której R2 odpowie pakietem ICMP (port unreachable) do R1, co w przypadku niektórych NAT może uniemożliwić połączenie.

Przeszkoda ICMP

Jeśli nasz NAT reaguje na tego typu pakiety ICMP, to można zamienić jeden typ ICMP na inny. Wystarczy pierwszemu pakietowi wysłanemu w kierunku drugiego routera ustawić odpowiednio niską wartość TTL. Wartość TTL jest zmniejszana wraz z przejściem przez każdy router. Gdy osiągnie zero, następuje odrzucenie pakietu i wysyłanie komunikatu ICMP nr 36 *time exceeded in-transit*.

Kolejną kwestią jest synchronizacja całej operacji. Jeśli NAT wysyła pakiety ICMP oznacza to, że host w sieci lokalnej nie wysłał jeszcze pakietu przestawiającego NAT. Niektóre wersje NAT w takiej sytuacji zaczynają zmieniać porty źródłowe wychodzących pakietów, co znacznie utrudnia komunikację.

Aby pakiety dotarły do obu NAT dokładnie w tym samym momencie należy zapewnić aby:

```
t1+t3=t2+t4
```

z czego wynika że każdy z hostów musi przed wysłaniem pakietu poczekać:

```
T = max(t1+t3, t2+t4) - z
```

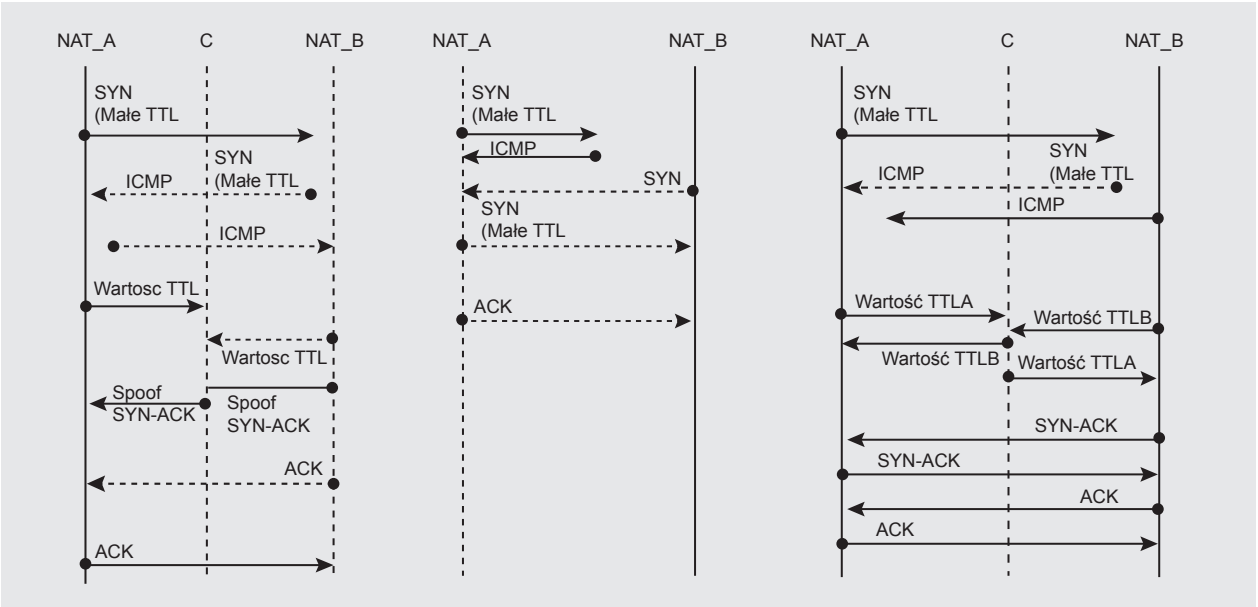
gdzie z jest sumą czasów transmisji pakietów danego hosta z C oraz czasu wędrówki pakietu do przeciwległego NAT

Jednak w większości przypadków tak dokładna synchronizacja nie jest potrzebna. Wystarczy sprawić, aby hosty A oraz B wysłały pakiety w tym samym momencie czyli:

```
T = max(t1, t2) - z
```

Czas na TCP

Techniki UDP hole punching są zazwyczaj skuteczne, ponieważ pro-



Rysunek 17. *Używanie TTL do stworzenia połączenia P2P*

tokół UDP nie jest tak skomplikowany jak na przykład TCP. Jeśli używamy innych protokołów, takich jak TCP, to sytuacja się skomplikuje, ponieważ posiadaon określony schemat nawiązywania połączenia, którego poprawności mogą pilnować nie tylko poszczególne implementacje NAT, ale również systemy antywłamaniowe, *firewall'e* itp. Jeśli już opanujemy technikę przebijania NAT za pomocą UDP, to zawsze możemy zastosować enkapsulacje TCP czy nawet całego IP w UDP tworząc pełny tunel IP. Pomimo iż TCP jest protokołem bardziej złożonym niż UDP istnieją metody pozwalające na stworzenie bezpośrednich połączeń pomiędzy komputerami zza różnych NAT.

Na początek należy przyjrzeć się metodom nawiązywania połączenia TCP i zastanowić się, co przy stosowaniu technik może zakończyć się niepowodzeniem na poszczególnych etapach połączenia.

Zainicjalizowanie połączenia następuje w trzech etapach. W pierwszym etapie klient wysyła pakiet z ustawioną flagą SYN, która świadczy o chęci nawiązania połączenia. Jeśli zdalny host zechce zaakceptować połączenie, to w drugim etapie odsyła pakiet z ustawioną flagą SYN oraz ACK. W trzecim etapie inicjator połączenia potwierdza odebranie informacji od serwera wysyłając pakiet z ustawioną flagą ACK. Trzystopniowy schemat nawiązywania połączenia ma dodatkowo na celu zsynchronizowanie numerów sekwencyjnych używanych do zapewnienia poprawności transmitowanych danych. Flaga SYN ustawiana w pakietach informuje, że zawiera on numer sekwencyjny. Każde połączenie TCP ma dwa takie numery. Każda ze stron posiada własną liczbę reprezentującą pewnego rodzaju licznik wysłanych danych.

Aby zastosować techniki przebijania firewall-i używając protokołu TCP, należy spowodować, by obie strony połączenia przeszły poprawnie przez trzy etapy. Jeśli któryś z etapów zostanie pominięty, to nawiązanie połączenia wymaga zastosowania dodatkowych operacji. Należy wiedzieć, że niektóre implementacje NAT pilnują tego, aby poszczególne kroki połączenia następowały dokładnie po sobie. Np. nie wpuszczą pakietu z ustawioną flagą SYN do sieci lokalnej, jeśli oczekiwany jest pakiet z ustawionymi flagami *SYN-ACK*. Dodatkowo NAT mogą pilnować zachowania poprawności numerów sekwencyjnych w trzech etapach połączenia. Jądra *Linux* z serii 2.4 są bardzo tolerancyjne jeśli chodzi o przestrzeganie tych zasad.

Aby pakiety przechodziły z jednej sieci lokalnej do drugiej wystarczy wysyłać dużą ilości odpowiednio sformatowanych ramek z A oraz B. Na hoście A uruchamiamy program nemesis, który będzie wysyłał ramki z ustawionymi flagami *SYN-ACK*:

```
#while true; do nemesis tcp -x 3000
-y 3000 -fSA -s 0 -a 0 -S
192.168.19.100 -D 157.158.180.235; done
```

Z hosta B musimy wysyłać ramki TCP z ustawioną flagą SYN:

```
#while true; do nemesis tcp -x 3000
-y 3000 -fS -s 0 -a 0 -S 192.168.18.200-D
157.158.181.42; done
```

Aby sprawdzić czy do B docierają ramki A z ustawionymi flagami *SYN-ACK*, to najprościej posłużyć się *tcpdumpem*:

```
#tcpdump -n 'tcp[13]==0x12'
```

Filtr *tcpdump* powinien przepuścić tylko te ramki, które mają ustawione flagi SYN oraz ACK.

Nie takie TCP straszne

Próba połączenia dwóch komputerów w dwóch różnych sieciach lokalnych wykorzystujących TCP może przebiegać prawie analogicznie do przypadku posługiwania się protokołem UDP. Przypuśćmy, że NAT na R1 i R2 nie modyfikuje portów źródłowych. Każdy z klientów A i B otwiera połączenie nasłuchujące (dla uproszczenia załóżmy, że na porcie 3000). Następnie obydwa komputery wykonują połączenie do C na port 5000 używając portu 3000 jako portu źródłowego (muszą przed połączeniem użyć polecenia „*bind*" na otwartym gnieździe). W tym momencie klienci A i B nasłuchują na porcie 3000 oraz posiada-

W Sieci

- *http://news.zdnet.com/2100-1009_22-842973.html* – artykuł przewidujący wyczerpanie adresów IP w 2005roku;
- *http://bgp.potaroo.net/ipv4/* – szczegółowy raport przewidujący wykorzystanie adresów IP w przyszłości;
- *http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_8-3/ipv4.html* – raport na temat „konsumpcji" adresów IP;
- *http://nutss.gforge.cis.cornell.edu/pub/imc05-tcpnat/* – analiza i opis technik przebijania NAT i firewalli;
- *http://www.brynosaurus.com/pub/net/p2pnat/* – opis technik przebijania NAT z użyciem UDP i TCP;
- *http://freshmeat.net/projects/nat-traverse/* – biblioteka wykorzystująca opisane w artykule techniki;
- *http://nutss.gforge.cis.cornell.edu/stunt.php* – strona domowa STUNT (Simple Traversal of UDP Through NATs and TCP);
- *http://freshmeat.net/projects/stund/* – strona domowa klienta i serwera używającego STUN;
- *http://vtun.sourceforge.net/* – oprogramowanie pozwalające stworzyć wirtualny tunel ethernetowy enkapsulowany w TCP lub UDP;
- *http://openvpn.net/* – implementacja VPN bardziej zaawansowana niż VTUN;
- Linux Server Hacks, Oreilly 2003 – znajdziemy w tej książce wiele ciekawych porad. Między innymi opis tunelowania VPN w SSH, który został skrótowo opisany w artykule;
- *http://samy.pl/chownat/* – strona z której możemy pobrać program chownat;
- *http://linide.sourceforge.net/nat-traverse/* – tutaj znajdziemy skrypt nat-traverse;
- *http://www.cis.nctu.edu.tw/~gis87577/xDreaming/XSTUNT/index.html* – tutaj znajduje się biblioteka XSTUNT oraz przykładowy program.

ją połączenie z hostem C (port źródłowy 3000 a port docelowy 5000). Od serwera S dowiadują się o swoich publicznych interfejsach. Host A dowiaduje się, że publiczny interfejs B to 157.158.180.235 i używa on portu 3000 do połączenia się z C (przy założeniu, że NAT R1 i R2 nie modyfikuje portów źródłowych przy translacji). Podobne informacje otrzymuje host B. Teraz A i B mogą dokonać próby połączenia bezpośredniego. A łączy się do publicznego interfejsu hosta B używając portu 3000 jako źródłowego. Podobnie postępuje B. Jeśli NAT nie jest symetryczny, to powinien użyć portu 3000 przy łączeniu się z A do publicznego interfejsu B. W momencie przechodzenia pakietu z ustawioną flagą SYN przez router, NAT zapamiętuje informację o mapowaniu i będzie wpuszczał pakiety, które pochodzą od NAT drugiego hosta. Przebieg nawiązywania połączenia jest w większości identyczny jak w przypadku komunikacji za pomocą UDP.

Kolejność wysyłania i odbierania pakietów ma tutaj dość duże znaczenie. Mogą tutaj zajść następujące przypadki:

- Pakiet z SYN komputera A dotrze do R2 przed wysłaniem przez B pakietu inicjalizującego połączenie do A (lub przypadek odwrotny);
- A i B wyślą pakiety równocześnie co spowoduje, że B otrzyma pakiet z SYN pochodzący od A i vice versa. Jeśli pakiety są idealnie zsynchronizowane w czasie, dochodzi do tak zwanego równoległego otwarcia połączenia.

W jednym i drugim przypadku zajdzie sytuacja, w której host posiadający gniazdo nasłuchujące na porcie P i próbujący nawiązać połączenie używając portu P jako źródłowego otrzyma pakiet z ustawioną flagą SYN. W zależności od implementacji stosu *TCP/IP* przychodzący pakiet zostanie skojarzony albo z gniazdem nasłuchującym, albo z gniazdem, które próbowało nawiązać połączenie. Nie mniej jednak oba przypadki doprowadzają do nawiązania połączenia. W pierwszym przypadku – pakiet zostanie zinterpretowany jako próba nawiązania połączenia. W odpowiedzi zostanie odesłany pakiet z ustawioną flagą *SYN-ACK*. Funkcja connect wywołana na maszynie, która zaakceptowała połączenie, zwróci błąd, gdyż adresy i porty źródłowe oraz docelowe są używane przez inne połączenie. W przypadku, gdy pakiet SYN zostanie skojarzony z gniazdem próbującym nawiązać połączenie, gniazdo nasłuchujące nie będzie w ogóle wykorzystywane. Funkcja connect zwróci status świadczący o udanym połączeniu. Pakiet, który został wysłany przez drugą maszynę nie zawierał flagi *ACK*, więc stos *TCP/IP* wyśle pakiet z ustawionymi flagami *SYN-ACK* (zachowując oryginalny i ustawiając własny numer sekwencyjny pakietu). Druga strona, w momencie odebrania pakietu z ustawionymi flagami *SYN-ACK*, potwierdza synchronizację numerów sekwencyjnych, a całe połączenie jest gotowe do „*użytku*".

Przy stosowaniu tej techniki może dojść do sytuacji, w której oba hosty wyślą w tym samym czasie pakiet *SYN* do siebie, a następnie oba jednocześnie potwierdzą otrzymane numery sekwencyjne pakietem *SYN-ACK* a następnie *ACK*. Implementacje *TCP/IP* różnie radzą sobie z tą sytuacją. Możliwa jest sytuacja, w której z obu stron wywołania connect się nie powiodą, natomiast połączenie zostanie „*magicznie stworzone*" gniazdami nasłuchującymi (po obu stronach accept się powiedzie).

Stosowanie tej techniki pociąga za sobą konieczność istnienia dwóch gniazd przypiętych do tego samego portu. Niektóre systemy tego nie potrafią u muszą w zamian użyć sekwencyjnej wersji wyżej opisanego algorytmu. Host A zgłasza chęć połączenia się z B poprzez serwer C. B wykonuje połączenie do publicznego interfejsu A. Połączenie to się nie udaje, ponieważ R1 nie wpuszcza pakietów od B. Następnie B tworzy gniazdo nasłuchujące i czeka na połączenia od A. Jeśli NAT po stronie B nie zlikwiduje wpisów o połączeniu `B->A` w momencie niepowodzenia połączenia, to A może zerwać połączenie z C i dokonać próby połączenia się na publiczny interfejs B (używając tego samego portu źródłowego, którego używał łącząc się z C). NAT po stronie B powinien wpuścić żądanie A umożliwiając gniazdu nasłuchującemu komputera B nawiązanie połączenia.

Wprawny czytelnik niewątpliwie zauważy, że skoro NAT R1 i R2 nie zmieniają portów źródłowych to pośrednik C nie jest potrzebny. Uwaga ta jest słuszna zarówno dla protokołu UDP jak i również TCP. Jeśli tylko hosty A i B znają adresy swoich publicznych interfejsów, ustaliły port, na którym chcą dokonać połączenia i zaczną całą operację w tym samym czasie – połączenie zostanie zrealizowane. W przypadku braku translacji portu pośrednik służy wyłącznie do wymiany informacji o adresach IP i synchronizacji całej operacji.

TCP – inne podejście

Złożoność protokołu TCP sprawia, że istnieją alternatywne podejścia do tego prezentowane wyżej. Podobnie jak w przypadku UDP można posłużyć się metodą polegającą na ustaleniu takiej wartości TTL, aby pakiet mógł swobodnie opuścić sieć lokalną, ale na tyle małej, aby nie dotarł do routera docelowego. Gdy oba hosty wyślą takie pakiety, spowoduje to skonfigurowanie ich routerów NAT tak, aby akceptowały pakiety należące do przyszłego połączenia. Po otwarciu „*tunelów*" w NAT można postąpić na trzy sposoby. Pierwszy polega na poinformowaniu przez A oraz B serwera pośredniczącego o numerach sekwencyjnych użytych do otwarcia tunelu. Następnie serwer pośredniczący wysyła dwa pakiety ze sfałszowanymi adresami źródłowymi. Pakiety mają ustawione flagi SYN-ACK i ich zadaniem jest zasymulowanie drugiego etapu połączenia A z B i B z A. Stosowanie tej techniki pociąga za sobą konieczność znalezienia ISP, który pozwala na wysyłanie sfałszowanych ramek, co może stanowić największy problem. Kolejny pomysł polega na tym, aby po otwarciu tunelu w NAT przez host A (pakietem SYN), wyłącznie host B dokonał próby bezpośredniego połączenia się do publicznego interfejsu hosta A. W odróżnie-

niu od poprzedniego sposobu, pośrednik służy wyłącznie do synchronizacji. Trzecie podejście wymaga otwarcia tunelu po obu stronach – zarówno NAT hosta A jak i B. Dodatkowo wymagana jest wymiana poprzez pośrednika informacji o numerach sekwencyjnych użytych do otwarcia „*tunelu*". Następnie hosty A oraz B wysyłają po jednym pakiecie każdy. Pakiet powinien posiadać ustawione flagi *SYN-ACK* oraz numer sekwencyjny zdalnej maszyny uzyskany od pośrednika.

Jedna z wyżej opisanych metod została zaimplementowana w bibliotece *XSTUNT*. Do tworzenia bezpośrednich połączeń wykorzystuje ona pośredniczący który oprócz synchronizacji służy również do wykrycia typów NAT łączących się klientów. Serwer do poprawnego działania wymaga dwóch adresów IP, które wykorzystywane one są do określenia typu NAT. Aby uruchomić serwer wystarczy wydać polecenie:

```
./XSTUNTServer 157.158.180.200
 157.158.180.201
```

Na stronie *XSTUNT* dostępny jest przykład klienta oraz serwera biblioteki. Serwer rejestruje się w komputerze pośredniczącym i oczekuje na połączenie od klienta. Klient łącząc się do serwera (poprzez pośrednika) wysyła do niego tekst wprowadzany interaktywnie z klawiatury. Serwer odebrawszy dane od klienta wysyła je z powrotem do klienta. Oczywiście połączenie pomiędzy klientem a serwerem jest połączeniem bezpośrednim pomiędzy sieciami lokalnymi.

Aby sprawdzić działanie przykładu ze strony wystarczy na hoście A wydać polecenie:

```
./xecho server to_jest_id_mojego_
serwera 157.158.180.200 157.158.180.201
```

Natomiast na komputerze B należy uruchomić ten sam program lecz z innymi parametrami:

```
./xecho client client_id to_jest_id_
mojego_serwera 157.158.180.200
157.158.180.201
```

jeśli wszystko pójdzie dobrze to tekst wprowadzony do okienka terminala hosta B zostanie odebrany przez serwer A oraz odesłany z powrotem do klienta B.

Czarne chmury na horyzoncie i nie tylko

Jeśli NAT posiada maszynę stanową, za pomocą której filtruje pakiety, które nie powinny zostać przetworzone w danym stanie, to możliwe jest, że nie uda nam się nawiązać połączenia. Na przykład, jeśli jedynie akceptowalną przez NAT sekwencją pakietów jest *SYN, SYN-ACK, ACK*, to *NAT* nie zaakceptuje drugiego przychodzącego pakietu SYN prezentowanego w drugim podejściu czy wychodzącego pakietu *SYN-ACK* z trzeciego podejścia, nie mówiąc już o metodzie, która nie stosuje metody z TTL. Kolejny problem może stwarzać interpretowanie pakietów ICMP oraz tych z ustawioną flagą RST, które mogą spowodować zmianę wewnętrzną stanu połączenia w NAT. Ale warto mieć na uwadze, że nawet najbardziej oporne NAT można w sprzyjających warunkach „*przebić*". Wystarczy zastosować mechanizmy przewidywania portów, dobrze zsynchronizować moment nawiązywania połączenia lub zmodyfikować stos TCP/IP po stronie klientów tak, aby był w stanie nawiązywać połączenia TCP pozwalając na małe odstępstwa od zasad protokołu. Opisanie wszystkich pomysłów, które wykorzystywano do przebijania NAT zajęłoby znacznie więcej miejsca niż niniejszy artykuł, a nawet wystarczyłoby na małą książkę.

Czekając na koniec świata

Przebijanie mechanizmu translacji adresów i portów jest bardzo ciekawą i edukującą czynnością. Można nauczyć się wiele nie tylko o mechanizmie działania NAT, ale również poznać zasady rządzące stosami TCP/IP. Producenci oprogramowania oraz urządzeń VoIP wykorzystują wyżej wymienione techniki w swoich rozwiązaniach praktycznie od momentu, kiedy technologia ta stała się powszechnie stosowana.Problem nawiązywania bezpośrednich połączeń jest zatem jak najbardziej aktualny. ●

O autorze

Konrad Malewski, absolwent informatyki Politechniki Śląskiej. Obecnie doktorant informatyki na AGH. Administrator amatorskich sieci komputerowych. Zarówno w pracy jak i prywatnie interesuje się programowaniem oraz bezpieczeństwem aplikacji sieciowych.
Kontakt z autorem:
kmalewski@gmail.com

Terminologia

- NAT – (ang. *Network Address Translation*), zwana także „*maskaradą*". Usługa działająca na routerze pozwalająca komputerom w sieci lokalnej na transparentny dostęp do innej sieci (zazwyczaj Internet). Czasami można spotkać się ze stwierdzeniem host NAT, host za NATem. W pierwszym przypadku chodzi po prostu o komputer, na którym działa usługa NAT. Drugie określenie jest wykorzystywane do określenia komputera korzystającego z tejże usługi.
- STUN – (ang. *Simple Traversal of User Datagram Protocol Through Network Address Translators*). Generalnie oznacza protokół, dzięki któremu host za NAT może wykryć swój publiczny adres, typ NAT, sposób mapowania jego połączeń etc. Mianem tym określa się również technikę tworzenia połączeń pomiędzy sieciami lokalnymi za pomocą protokołu UDP.
- TURN – (ang. *Traversal Using Relay NAT*). Protokół za pomocą którego dwa hosty znajdujące się w różnych sieciach lokalnych przesyłają do siebie dane wykorzystując pośrednika.

Pakiet SYN, Pakiet SYN-ACK, Pakiet ACK – Nagłówek protokołu TCP posiada pola będące flagami, które wykorzystywane są między innymi do nawiązywania, kończenia oraz synchronizacji połączenia. Pakiet SYN stanowi wiadomość TCP z ustawioną flagą SYN. Pakiety SYN-ACK oraz pakiet ACK analogicznie.