Obrona

Przegląd wybranych metod niwelowania ograniczeń sieci lokalnych

Konrad Malewski

stopień trudności

Istnieje wiele aplikacji, które do działania potrzebują publicznego adresu IP. Jest szereg sposobów, aby umożliwić użytkownikom wewnątrz sieci lokalnych korzystanie ze wszystkich zalet publicznego IP. Można powiedzieć, że istnieje pewnego rodzaju walka pomiędzy administratorami sieci komputerowych, a ich użytkownikami.

Ci pierwsi chcą kontrolować każde posunięcie drugich. Z kolei użytkownicy chcieliby mieć nieograniczony dostęp do wszystkich usług – zwłaszcza do modnych ostatnio p2p. W zależności od potrzeb użytkowników administrator ma do dyspozycji różne narzędzia, za pomocą których może przekierowywać porty, spiąć dwie sieci lokalne lub zestawić tunel z komputerem w Internecie powodując, że będzie on widziany w sieci lokalnej. Dodatkowo musi mieć na uwadze fakt, że istnieją techniki, za pomocą których bardziej wykwalifikowany użytkownik może osiągnąć podobny rezultat. Może on mianowicie wejść w posiadanie narzędzi, które zniwelują wysiłki administratora idące ku ograniczaniu ruchu p2p oraz innych usług, do których administrator nie chce dać użytkownikom dostępu.

Spełniając zachcianki użytkownika

Najczęściej użytkownicy sieci lokalnych chcą „hostować" gry. Z punktu widzenia administratora oznacza to ni mniej ni więcej to, że należy przekierować jeden lub więcej portów z interfejsu zewnętrznego do sieci lokalnej. Można tego dokonać na parę sposobów. Pierwszym sposobem jest ręczna konfiguracja. Aby przekierować port UDP wystarczy wydać polecenie:

```
iptables -t nat -A PREROUTING -i IINT -p tcp --
dport PORT_DOCELOWY -j DNAT --to-destination
ADRES_W_SIECI_LOKALNEJ
```

Z artykułu dowiesz się...

- Jak przekierowywać porty z wykorzystaniem urządzenia IGD.
- Jak samodzielnie lub z pomocą administratora utworzyć wirtualną sieć.
- Jak stworzyć tunel IP wykorzystujący różne protokoły wyższych warstw.

Co powinieneś wiedzieć...

- Powinieneś znać model ISO/OSI.
- Powinieneś mieć ogólne pojęcie o zasadzie działania sieci TCP/IP.
- Powinieneś posiadać umiejętność administrowania systemem Linux na poziomie użytkownika.

gdzie zamiast IINT podajemy interfejs wejściowy np. eth0, PORT_DOCELOWY ustalamy dla konkretnej usługi użytkownika, którego adres IP w sieci lokalnej to ADRES_W_SIECI_LOKALNEJ. Dodawanie i usuwanie przekierowań tym sposobem jest nieco uciążliwe, zwłaszcza gdy wielu użytkowników chce przekierować jeden port (co wiąże się z częstym dodawaniem i usuwaniem wpisów), lub nie chce aby przekierowanie było permanentne.

Na szczęście można ten proces zautomatyzować. W tym celu należy uruchomić demona UPNP (z ang.: Universal Plug and Play), z którym będą komunikować się aplikacje klientów tj. Windows Messenger, Emule itp. Pierwszym krokiem idącym do uruchomienia IGD jest ściągnięcie, skompilowanie i skonfigurowanie programu demona ([1] oraz [2]). Plik konfiguracji (o nazwie upnpd.conf) znajduje się domyślnie w katalogu /etc. Aby uruchomić program przy minimalnej konfiguracji należy ustawić poprawną ścieżkę dostępu do programu Iptables. Dla celów demonstracyjnych należy również ustawić parametr duration tak, aby miał wartość inną niż domyślna (np. 50) oraz ustawić zmienną debug_info na 3. Parametr duration określa jak długo mapowanie jest „ważne". Po przekroczeniu tego czasu przekierowanie jest usuwane z routera.

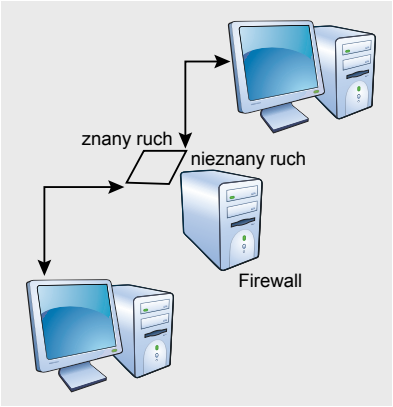
Ostatnią rzeczą, jaką należy wykonać jest dodanie ścieżki multicastowej do domyślnej tablicy routingu:

```
route add -net 239.0.0.0 netmask
255.0.0.0 eth0
```

Uruchomienie demona IGD sprowadza się do wydania komendy:

```
./upnpd -f eth1 eth0
```

Jeśli nie użyjemy paremetru „-f", demon upnp uruchomi się w tle i nie będziemy widzieli komunikatów diagnostycznych. Kolejne parametry programu to nazwa interfejsu zewnętrznego oraz wewnętrznego. Po uruchomieniu demona UPnP wszystkie aplikacje, które działają na komputerach w sieci lokalnej, a są zdolne do wykorzystania zalet IGD, powinny umieć wykryć obecność demona i korzystać z jego funkcji.



Rysunek 1. *Prawdopodobna konfiguracja firewalla.*

Istnieje klient UPnP ([3]), za pomocą którego można ręcznie sterować przekierowaniami na routerze. Program ten potrafi automatycznie wykryć urządzenia zgodne ze specyfikacją IGD. W tym celu wykorzystuje protokół SSDP do wykrywania urządzeń. Nie wgłębiając się zbytnio w szczegóły zobaczmy, jak można ręcznie stworzyć przekierowanie TCP z portu 5000 na routerze do portu 22 na kliencie. Na początku sprawdzamy status połączenia – widoczne na Listingu 1.

Powyższa odpowiedź oznacza, że urządzenie UPnP działające na routerze, zostało wykryte. Dodatkowo polecenie zwraca pewną ilość informacji statystycznych oraz informacje o parametrach skonfigurowanego łącza. Należy zwrócić uwagę na fakt, że parametry takie jak MaxBitRateDown nie są wyliczane, a zwracana wartość jest parametrem z pliku konfiguracyjnego. Teraz przystąpmy do tworzenia przekierowania za pomocą programu [3], co można zobaczyć na Listingu 2.

Jest to równoważne wydaniu następującej komendy na routerze:

```
/sbin/iptables -t nat -I PREROUTING
-i eth1 -p TCP --dport 5000 -
```

Listing 1. *Status połączenia*

```
# ./upnpc -s

upnpc : miniupnp test client. (c) 2006 Thomas Bernard
go to http://miniupnp.free.fr/ for more information.
UPnP device :
desc: http://192.168.64.254:49152/gatedesc.xml
st: urn:schemas-upnp-org:device:InternetGatewayDevice:1
UPnP device :
desc: http://192.168.64.254:49152/gatedesc.xml
st: urn:schemas-upnp-org:device:InternetGatewayDevice:1
getting "http://192.168.64.254:49152/gatedesc.xml"
getting "http://192.168.64.254:49152/gateconnSCPD.xml"
Connection Type : IP_Routed
Status : Connected, uptime=10548
MaxBitRateDown : 512000 bps MaxBitRateUp 512000 bps
Bytes: Sent: 490900 Recv: 34710590
Packets: Sent: 8881 Recv: 24517
```

Listing 2. *Tworzenie przekierowania*

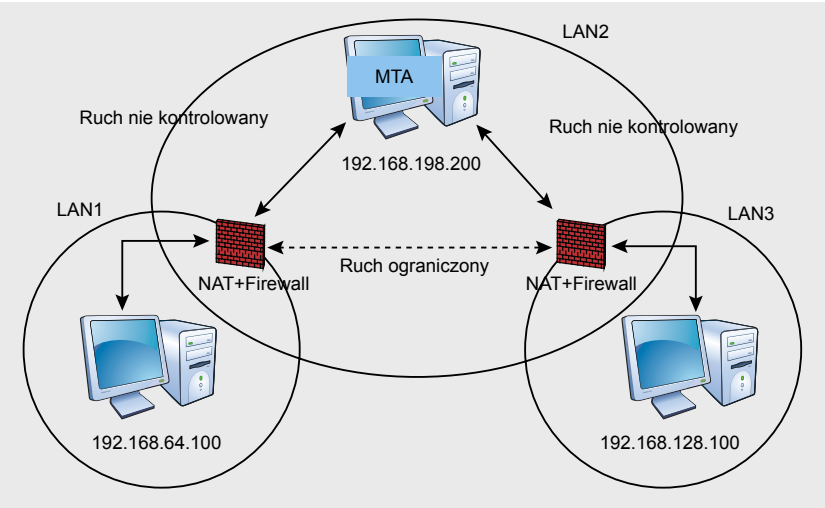
```
#./upnpc -a 192.168.64.100 22 5000 tcp

upnpc : miniupnp test client. (c) 2006 Thomas Bernard
go to http://miniupnp.free.fr/ for more information.
UPnP device :
desc: http://192.168.64.254:49152/gatedesc.xml
st: urn:schemas-upnp-org:device:InternetGatewayDevice:1
UPnP device :desc: http://192.168.64.254:49152/gatedesc.xml
st: urn:schemas-upnp-org:device:InternetGatewayDevice:1
getting "http://192.168.64.254:49152/gatedesc.xml"
getting "http://192.168.64.254:49152/gateconnSCPD.xml"
ExternalIPAddress = 192.168.198.64
InternalIP:Port = 192.168.64.100:22
external 192.168.198.64:5000 is redirected to internal 192.168.64.100:22
```

```
j DNAT --to 192.168.64.100:22
/sbin/iptables -A FORWARD -p TCP -d
192.168.64.100 --dport 22 -j ACCEPT
```

Jeżeli na routerze został podniesiony poziom logowania, to powyższa informacja powinna zostać wyświetlona na ekranie. Po upływie czasu określonego parametrem duration przekierowanie zostanie automatycznie skasowane poprzez usunięcie dwóch wpisów w firewallu. Przy pomocy programu Upnpc użytkownik może także usunąć przekierowanie przed upływem czasu duration, oraz wyświetlić aktywne przekierowania.

Korzystając z rozwiązań opartych na IGD należy mieć na uwadze nie tylko jego zalety, ale i wady, przez które wielu administratorów klasyfikuje tego rodzaju urządzenia jako ryzyko dla bezpieczeństwa sieci lokalnej. Trzeba sobie zdawać sprawę, że przedstawiona wersja IGD została zaprojektowana tak, aby emulować ICS (Microsoft's Internet Connection Service) i robi to bardzo dokładnie. Oznacza to, ni mniej ni więcej, tyle, że odpowiedzialność za bezpieczeństwo i celowość przekierowań zostaje przeniesiona na użytkowników bramy. Daje to potencjalnemu intruzowi możliwość bezproblemowego przeglądania, dodawania oraz usuwania przekierowań. Więcej szczegółów technicznych czytelnik może znaleźć na forum UPnP.

Na potrzeby nie tylko graczy, ale również użytkowników chcących uniezależnić swoją wirtualną sieć od lokalizacji poszczególnych węzłów zostało stworzone oprogramowanie łączące w sobie VPN, aplikacje p2p oraz prosty system IM. Narzędzie to, noszące nazwę Hamachi [5], dostępne jest zarówno dla systemów spod znaku pingwina, jak i Windows. W telegraficznym skrócie można powiedzieć, że tworzy ono sieć lan w oparciu o Internet. Konfiguracja programu w systemie Linux jest bajecznie prosta. W systemie z Readmont konfigurowanie aplikacji przysparza jeszcze mniej problemów. Program jest dość nowatorski w swoim działaniu. Przy instalacji aplikacji tworzony jest dodatkowy interfejs sieciowy, przez który będą wykonywane połączenia do komputerów w wirtualnej sieci. Klient po uruchomieniu rejestruje się w systemie. Dodatkowo przeprowadzana jest dwukierunkowa autentykacja zarówno serwera jak i klienta. Każdy serwer Hamachi posiada parę kluczy – publiczny i prywatny, służących do potwierdzania autentyczności. Klucze publiczne są zawarte w programach klientów. Klient Hamachi generuje swoją parę kluczy, a jego klucz publiczny udostępniany jest serwerom w celu przeprowadzenia późniejszej autentykacji kolejnych podłączeń klienta. Przy pierwszym uruchomieniu klienta należy stworzyć nową wirtualną sieć, lub dołączyć się do już istniejącej. Każdy komputer w takiej sieci posiada swój pseudo-publiczny adres IP. Zakładając, że mamy skonfigurowanego wcześniej klienta Hamachi, kolejnym etapem przy jego uruchamianiu jest synchronizacja z innymi klientami należącymi do wirtualnych sieci. Oznacza to stworzenie tunelu UDP pomiędzy komputerami w sieciach lokalnych i wykorzystanie go jako nośnika danych do enkapsulowania innych protokołów. Do zestawienia tunelu i przebicia NAT wykorzystywany jest komputer–pośrednik zwany często mediatorem. Szczegóły techniczne tego procesu można poznać np. w artykule „Sieci nie tak znowu lokalne" z Hakin9u 1/2007. Warto jednak wiedzieć, że mediator jest wykorzystywany jednokrotnie – przy przebijaniu NAT.



Rysunek 2. *Schemat komunikacji poprzez serwer pocztowy.*

Dodatkową cechą węzłów łączących hosty w Hamachi jest szyfrowanie przesyłanych danych. Niektórych zdziwić może fakt, że w celu otrzymania możliwości przesyłania danych w postaci niezaszyfrowanej należy wykupić konto premium u producenta. Aplikacja ta likwiduje wszystkie problemy związane z sieciami lokalnymi posiadającymi niepubliczne adresy lokalne. Jednak, jak by tego było mało, Hamachi posiada wbudowany serwer proxy, za pomocą którego można korzystać z Internetu wykorzystując zdalny komputer.

Z perspektywy administratora sieci komputerowej zablokowanie tego typu usługi może być bardzo pożądane. Niektóre z serwerów pośredniczących Hamachi używają do synchronizacji portu 32976. Aby zablokować możliwość połączenia z tymi hostami wystarczy dodać następującą regułkę w firewallu:

```
iptables -I FORWARD -p tcp --dport
32976 -j DROP
```

Powinno to uniemożliwić synchronizację z częścią serwerów Hamachi, jednak nie zlikwiduje to całkowicie łączenia się z ową siecią. Aby w pełni zablokować sieć należy, tak jak w przypadku niektórych sieci p2p, sprawdzić adresy IP pośredników i wpisać je do firewalla. W chwili pisania tego artykułu wystarczyło oprócz

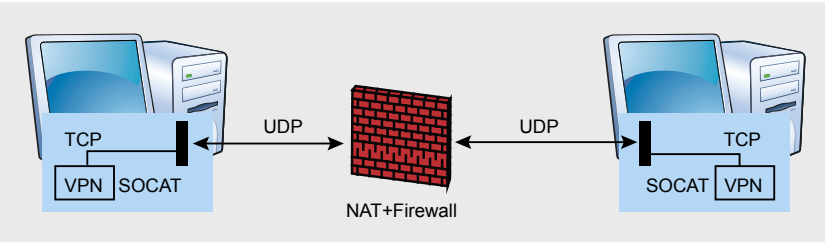
wyżej wymienionej reguły zablokować jeden adres IP, z którym klient Hamachi łączył się na porcie 443:

```
iptables -I FORWARD -d 63.208.197.211
-j DROP
```

Tunelowanie, enkapsulacja – IP w czymkolwiek?

Duża część administratorów konfiguruje firewalle w ten sposób, że znany ruch np. SSH, HTTP, FTP, SMTP nie jest dokładnie analizowany przez reguły firewalla, a pozostałe pakiety przepuszczane są na przykład przez filtry wykrywające ruch p2p, ograniczana jest liczba równoczesnych połączeń, czy w końcu niektóre porty są całkowicie blokowane. Tak skonfigurowanego firewalla można obejść w sposób, jaki od razu przychodzi na myśl – należy ukryć interesujący nas ruch w pakietach, które będą zaklasyfikowane na firewallu jako „znany ruch" (patrz rysunek 1).

Jednym z takich programów jest Mailtunnel [6]. Ten, pochodzący z 1999 roku, program składa się z trzech skryptów napisanych w Perlu. Jego działanie polega na ukrywaniu danych połączenia w wiadomościach email, zaś konfiguracja



Rysunek 3. *Tunelowanie danych TCP w UDP.*

sprowadza się do edycji pliku conf.pl. Celem dla którego powstał program jest stworzenie ukrytego kanału pomiędzy komputerem A, który znajduje się w sieci lokalnej (załóżmy, że w danej sieci administrator przepuszcza tylko pocztę email), a hostem B, który po odebraniu przesyłek od A, przesyła dane do komputera C poprzez stworzone połączenie TCP. W efekcie A łącząc się do localhost:port połączy się z komputerem C. Aby skomplikować przykład, załóżmy że A oraz B znajduje się za NAT, a MTA, którego używają oba komputery znajduje się w jeszcze innym miejscu (patrz rysunek 2).

Naszym celem będzie zestawienie połączenia VPN pomiędzy A oraz B. Pierwszym krokiem będzie założenie konta email. Od skrzynki email wymaga się w zasadzie tylko tego, aby była szybka, umożliwiała obsługę przez POP3 i SMTP oraz nie posiadała filtru antyspamowego, który usunąłby niektóre pakiety z konwersacji. Załóżmy, że na hoście o adresie 192.168.198.200 założone zostały dwie skrzynki: mtn_relay1 oraz mtn_relay2. Kolejnym krokiem jest skonfigurowanie Fetchmaila na komputerach A oraz B.

Na hoście A plik konfiguracyjny .fetchmailrc powinien zawierać:

```
set daemon 1
poll 192.168.198.200 with proto POP3
user 'mtn_relay1' there with password
'mtnrelay1' is 'mtn' here
```

natomiast na komputerze B:

```
set daemon 1
poll 192.168.198.200 with proto POP3
user 'mtn_relay2' there with password
'mtnrelay2' is 'mtn' here
```

Jeżeli ktoś preferuje korzystanie ze środowiska graficznego, może użyć programu Fetchmailconf do uzyskania podobnego pliku konfiguracyjnego.

Po wpisaniu polecenia „fetchmail" jako użytkownik mtn na komputerze A oraz B poczta skierowana do mtn_relay1@[192.168.198.200] powinna trafiać do /var/spool/mail/mtn na A. Podobnie email wysłany do mtn_relay2@[192.168.198.200] trafi do komputera B. Na obu komputerach fetchmail odpytuje serwer pop3 co 1 sekundę. Kontroluje to liczba, która znajduje się za słowem „deamon" w pliku konfiguracyjnym.

Program Mailtunnel został przystosowany do tunelowania tekstowych danych. Aby go dostosować do wiadomości binarnych należy w miejsce operatorów wczytania linii „<..>" użyć funkcji „read". Zmodyfikowany program znajduje się na [7].

Aby skonfigurować VPNa należy sięgnąć po przepis z numeru Hakin9 6/2006 autorstwa gosub-a. Jeśli jednak zależy nam na prostszej wersji tunelu, w którym szyfrowanie odbędzie się poprzez współdzielony klucz, to możemy pójść nieco na skróty. Na jednym z komputerów generujemy klucz openvpn-owy poleceniem:

```
openvpn --genkey --secret key
```

następnie w sposób, który uniemożliwiłby podsłuchanie transmisji, kopiujemy plik „key" na drugą maszynę. Po czym na maszynie B wydajemy polecenie:

```
openvpn  --dev tun1 --ifconfig 10.4.0.2
10.4.0.1 --verb 5 --secret key --proto
tcp-server --port 666
```

które uruchomi serwer VPN nasłuchując na gnieździe TCP i porcie 666. Mając już działający serwer VPN należy uruchomić program forwardujący pakiety z serwera VPN poprzez email do klienta:

```
./mtn forward localhost 666
```

Na hoście A uruchamiamy klienta tunelu:

```
./mtn listen 5000
```

oraz klienta VPN:

```
openvpn --remote 127.0.0.1 --dev
tun1 --ifconfig 10.4.0.1 10.4.0.2 --verb
5 --secret key --proto tcp-client --
port 5000
```

Po paru chwilach możemy cieszyć się tunelem zestawionym pomiędzy komputerami A oraz B – Listing 3.

Jak możemy się domyślać, przez takie połączenie nie da się komfortowo grać w Quake'a, lecz dla celów SSH w zupełności ono wystarczy. Biorąc pod uwagę zasadę działania przesyłek email można od razu wpaść na pomysł routingu opartego o emaila. Za pomocą przekazywania poczty można skomplikować drogę pakietu unikając wścibskich oczu administratorów, rządu, terrorystów, żony etc... Ponadto zastosowanie takiej techniki jest słuszne w przypadku, gdy wiemy, że kierowanie poczty do konkretnego hosta w sieci będzie szybsze, niż wysyłanie emaili do innego. Tworząc większą liczbę skrzynek pocztowych, które przekazywałyby pomiędzy sobą wiadomości, powodowalibyśmy, że pakiet skierowany z komputera A do C przechodziłby przez szereg MTA.

Do kompletu protokołów brakuje już jedynie tunelowania za pomocą FTP. Jednym z programów, które częściowo tego dokonują jest Ftptunnel [8]. Program ten jest raczej aplikacją typu *proof of concept*, niż programem do tunelowania. Składa się on z czterech skryptów. Jeden z nich (*master.pl*) należy umieścić w katalogu głównym serwera FTP. Po uruchomieniu stworzy on pliki semaforów, których będzie oczekiwał program klienta, a następnie poprosi o wprowadzenie komendy, która zostanie zapisana do pliku o nazwie *request*. Na drugiej maszynie wgrywamy plik slave.pl i modyfikujemy jego część, w której wskazany jest serwer FTP, nazwa użytkownika i hasło. Skrypt *slave.pl* zaloguje się na serwer, pobierze plik *request*, wykona go, a wynik zapisze do pliku *answer*, który następnie zostanie umieszczony na komputerze, na którym uruchomiony jest skrypt master-a.

Program ten posiada wiele braków, lecz wydaje się, że przy odrobinie pracy uda się za pomocą tej techniki stworzyć połączenie VPN pomiędzy dwoma komputerami. Najprawdopodobniej modyfikacji nie powinno być dużo jeśli użyjemy programu Socat [9], za pomocą którego możemy przekazywać dane pomiędzy dwoma niezależnymi źródłami, począwszy od gniazd, poprzez pliki, a skończywszy na serwerach proxy.

Tunel IP poprzez zbiory w sieciowym systemie plików

Skoro już wspomniałem o programie Socat, trudno nie przytoczyć w tym miejscu jego niewątpliwych zalet, za pomocą których możliwe jest zestawienie dowolnych tuneli. Na przykład, aby osiągnąć podobny efekt, jak w artykule Hakin9 „Tunele TUN/TAP w systemach Linux/Unix" z lutego 2007, można na komputerze zdalnym uruchomić polecenie:

```
socat udp4-listen:53 tcp4:127.0.0.1:
5000
```

natomiast na komputerze za firewallem:

```
socat tcp4-listen:5000 udp:
adreskomputerazdalnego:53,sourceport=53
```

Teraz posiadając działające połączenie TCP (patrz rysunek 3) pomiędzy dwoma komputerami, widziane na routerze jako szereg pakietów UDP, można użyć pakietu Openpvn do stworzenia tunelu.

Jak zostało wspomniane wcześniej, Socat umożliwia wykorzystanie plików jako źródła danych. Mając to na uwadze, nietrudno wyobrazić sobie, jak za pomocą tego programu można stworzyć prosty tunel w oparciu o zdalne zasoby NFS. Do zestawienia tunelu VPN potrzebny będzie nam dodatkowy programik o nazwie Udprelay [10]. Dodatkowo trzeba będzie lekko zmodyfikować jego źródła. Mianowicie w pliku udprelay.c w

linijce 353 przed komendą *bind* należy wstawić:

```
if (setsockopt(sock,SOL_SOCKET,
SO_REUSEADDR,&one,sizeof(one))<0)
{
syslog (LOG_ERR, "setsockopt, %m");
exit (1);->
}
```

Pozwoli to na przypięcie dwóch gniazd do tego samego portu. Mając zmodyfikowany i skompilowany program, możemy przystąpić do tworzenia tunelu. Schemat komunikacji po NFS obrazuje rysunek 4.

Rolą maszyny B jest jedynie udostępnienie jednego katalogu poprzez NFS. W tym katalogu znajdują się dwa pliki *hostA-hostC* oraz *hostC-hostA*. Pierwszy z nich będzie użyty do jednokierunkowej komunikacji od hostaA do hostaC. Do łączności w drugą stronę wykorzystany zostanie drugi plik. Załóżmy, że komputer A oraz C mają skonfigurowanego klienta NFS i zamontowany synchronicznie katalog /mnt/nfs komputera B w katalogu /mnt/nfs-remote. Aby przedstawione na rysunku połączenie zaczęło działać, należy wykonać następujące sekwencje komend. Na początku uruchamiamy serwer openvpn na komputerze A. Serwer powinien nasłuchiwać na porcie TCP: 10000.

Aby zrealizować część numer 1 połączenia należy wydać komendę:

```
socat udp-listen:15000 tcp4:
192.168.64.100:10000
```

Spowoduje to powstanie dwukierunkowego kanału, który przekieruje wszystkie pakiety UDP przychodzące na port 15000 do połączenia TCP na port 10000.

Połączenie UDP-plik (numer 2) realizuje komenda:

```
socat -u udp4-listen:10000,
reuseaddr open:/mnt/nfs_remote/hostA-
hostC,sync
```

Wszystkie pakiety UDP wysłane na port 10000, zostaną zapisane do pliku na dysku sieciowym. Przełącz-

Listing 3. *Tunel zestawiony pomiędzy komputerami A oraz B*

```
hostb:~# ping 10.4.0.1
PING 10.4.0.1 (10.4.0.1) 56(84) bytes of data.

64 bytes from 10.4.0.1: icmp_seq=1 ttl=64 time=5147 ms
64 bytes from 10.4.0.1: icmp_seq=2 ttl=64 time=10093 ms
64 bytes from 10.4.0.1: icmp_seq=3 ttl=64 time=9092 ms
64 bytes from 10.4.0.1: icmp_seq=4 ttl=64 time=8092 ms
64 bytes from 10.4.0.1: icmp_seq=5 ttl=64 time=7092 ms
64 bytes from 10.4.0.1: icmp_seq=6 ttl=64 time=24040 ms
64 bytes from 10.4.0.1: icmp_seq=7 ttl=64 time=23038 ms
```

Listing 4. *Połączenie*

```
debian:~# ping 10.4.0.2
PING 10.4.0.2 (10.4.0.2) 56(84) bytes of data.

64 bytes from 10.4.0.2: icmp_seq=1 ttl=64 time=8607 ms
64 bytes from 10.4.0.2: icmp_seq=2 ttl=64 time=7607 ms
64 bytes from 10.4.0.2: icmp_seq=3 ttl=64 time=6607 ms
64 bytes from 10.4.0.2: icmp_seq=4 ttl=64 time=5607 ms
64 bytes from 10.4.0.2: icmp_seq=5 ttl=64 time=4607 ms
64 bytes from 10.4.0.2: icmp_seq=6 ttl=64 time=3607 ms
64 bytes from 10.4.0.2: icmp_seq=7 ttl=64 time=5623 ms
64 bytes from 10.4.0.2: icmp_seq=8 ttl=64 time=4620 ms
64 bytes from 10.4.0.2: icmp_seq=9 ttl=64 time=3618 ms
64 bytes from 10.4.0.2: icmp_seq=10 ttl=64 time=2614 ms
--- 10.4.0.2 ping statistics ---

13 packets transmitted, 10 received, 23% packet loss, time 12024ms
rtt min/avg/max/mdev = 2614.549/5312.246/8607.296/1789.397 ms, pipe 9
```

Listing 5. *Rezultaty implementacji*

```
vegard@gyversalen:~$ ping -i 900 10.0.3.1
PING 10.0.3.1 (10.0.3.1): 56 data bytes

64 bytes from 10.0.3.1: icmp_seq=0 ttl=255 time=6165731.1 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=255 time=3211900.8 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=255 time=5124922.8 ms
```

Listing 6. *Rezultat na maszynie o adresie 192.168.128.100*

```
debian:~# hping2 192.168.198.200  -S -p 110
HPING 192.168.198.200 (eth0 192.168.198.200): S set, 40 headers + 0 data
                      bytes
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=0 win=5840
                      rtt=8.9 ms
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=1 win=5840
                      rtt=3.8 ms
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=2 win=5840
                      rtt=1.3 ms
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=3 win=5840
                      rtt=1.3 ms
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=4 win=5840
                      rtt=1.2 ms
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=12 win=5840
                      rtt=2.8 ms
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=24 win=5840
                      rtt=2.3 ms
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=35 win=5840
                      rtt=2.6 ms
len=46 ip=192.168.198.200 ttl=63 DF id=0 sport=110 flags=SA seq=46 win=5840
                      rtt=3.7 ms
```

nik „-u” powoduje, że komunikacja gniazdo->plik jest jednokierunkowa. Dodatkowy parametr „sync” zapewnia, że zapisy do pliku nie będą buforowane.

Dane przychodzące do komputera A z komputera C poprzez plik hostC-hostA (połączenie nr 6) są przekazywane za pomocą:

```
socat -u open:/mnt/nfs_remote/host4-
host1,
sync,ignoreeof udp:192.168.64.100:
15001,
sourceport=10001,reuseaddr
```

Dodatkowy parametr „ignoreeof” powoduje, że Socat zakłada iż plik rośnie i ignoruje znacznik końca danych i próbuje odczytać więcej danych (działanie podobne do polecenia *tail -f*). Dane odczytywane z pliku są wysyłane do UDP:192.168.64.100:15001. Jako port źródłowy ustawiona jest wartość 10001. Pakiety te odbierane są przez program Udprelay. Zadaniem połączenia numer 6.5 jest odebranie tych pakietów a następnie wysłanie ich do UDP:192.168.60.100:15000, używając przy okazji portu 10000. Połączenie to realizowane jest przez polecenie:

```
udprelay -d -f /etc/udprelay.conf
```

Plik konfiguracyjny dla Udprelay powinien zawierać jedną linijkę:

```
relay 192.168.64.100 10001
15001 192.168.64.100 15000 10000 oneway
```

Nadaje ona działaniu Udprelay następujące znaczenia: przekazuj pakiety w jednym kierunku.; oczekuj pakietów z adresu 192.168.64.100, portu docelowego 15001 i portu źródłowego 10001; przekaż dane do 192.168.64.100 używając portu docelowego 15000 i portu źródłowego 10000.

Jak pamiętamy, na porcie 15000 nasłuchuje Socat realizujący połączenie numer 1, a na porcie 10000 Socat obsługujący połączenie numer 2. Użycie Udprelay ma na celu zmylenia Socat-a (1) tak aby po odebraniu pakietów od 6.5 odpowiedział pakietami w kierunku (2).

Nadszedł czas na komputer C. Połączenie nr 3 zrealizuje polecenie:

```
socat -u udp-listen:20000,reuseaddr
open:/mnt/nfs_remote/hostC-hostA,sync
```

Połączenie nr 4:

```
socat tcp4-listen:5000,reuseaddr udp:
192.168.128.100:20000,sourceport=25000
```

Połączenie nr 5:

```
socat -u open:/mnt/nfs_remote/hostA-
hostC,
sync,ignoreeof udp:192.168.128.100:
25001,
sourceport=20001
```

Połączenie nr 5.5:

```
udprelay -d -f /etc/udprelay.conf
```

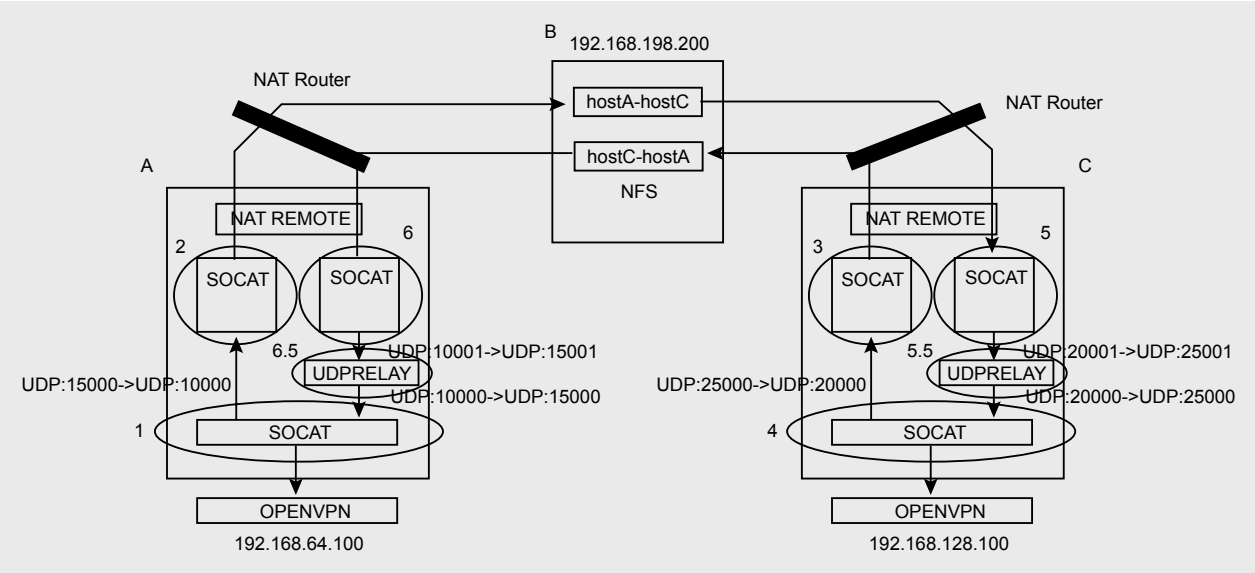
Plik konfiguracyjny powinien zawierać:

```
relay 192.168.128.100 20001
25001 192.168.128.100 25000 20000
oneway
```

Teraz prześledźmy, co się stanie, gdy uruchomimy openvpn-a, który połączy się do TCP:192.168.128.100:5000.

W tej sytuacji pakiet zostanie odebrany przez (4) i wysłany do (3) na port UDP:20000. (3) zapisze dane do pliku (hostC-hostA) na dysku sieciowym NFS. Po pewnym czasie (6) zorientuje się, że plik się powiększył, odczyta go i prześle do (6.5), który wyśle go w pakiecie UDP:192.168.64.100:15000 używając portu 10000 do (1). (1) przekaże dane do serwera Openvpn komputera A. Openvpn w odpowiedzi wyśle dane do (1), który odpowie w kierunku, z którego przyszły dane – czyli (2). (2) zapisze dane do pliku hostA-hostC. Plik ten jest monitorowany przez (5), który zauważając dane wyśle je do (5.5). Dane te w pakiecie UDP:192.168.128.100:25000 (port źródłowy 20000) trafią do (4) i dalej z powrotem do klienta openvpn. Voila...

Połączenie takie ma jeden mankament – co zobaczymy na Listingu 4.



Rysunek 4. *Schemat komunikacji poprzez serwer NFS*

Komfort gry w Quake'a w dalszym ciągu byłby zachwiany. Ponadto trudno nie wspomnieć o bezpieczeństwie takiego tunelu, a raczej braku takiego bezpieczeństwa. Cała komunikacja zostaje bowiem utrwalona w dwóch plikach zdalnego katalogu.

IP w czymkolwiek

Techniki tunelowania za pomocą protokołu HTTP oraz ICMP zostały opisane w Hakin9u nr 2/2007, w związku z tym niniejszy artykuł pominie te protokoły. Zainteresowanym czytelnikom polecam lekturę tego artykułu oraz dokumentacje do [11] [12] [13] jako uzupełnienie.

Jednym z ciekawszych pomysłów, aczkolwiek dość trudnym w realizacji, jest użycie papieru jako medium transmisyjnego i wykorzystanie gołębi pocztowych do przekazywania pakietów z jednego miejsca w inne. Pomysł ten został opisany w dokumencie RFC numer 1149. Pierwsze próby implementacji podobno przyniosły dość obiecujące rezultaty, co zobaczymy na Listingu 5.

W tym przypadku należy definitywnie stwierdzić. Używając takiego tunelu nie da się grać w Quake'a.

Paradygmat – „szukaj i niszcz”

Istnieje stosunkowo niewiele narzędzi do wyszukiwania ukrytych kanałów. Być może spowodowane jest to tym, że *de facto* nie wiadomo czego należy szukać. Jednym z lepszych (a zarazem sprawdzonych) narzędzi jest snort_inline opisany w Hakin9u 1/2007. Należy mieć na uwadze fakt, że wykrywanie tuneli sprowadza się w większości przypadków do analizy statystycznej płynącego przez router ruchu i wykrywania wszelkiego rodzaju anomalii. Dlatego odpowiednia konfiguracja systemu IDS jest w stanie wykryć ukryty tunel. Dla protokołu TCP zostało napisane narzędzie o nazwie Tcpstatflow [15]. Jego zadaniem jest analiza ruchu i alarmowanie w momencie przekroczenia progu parametru ruchu takiego, jak czas trwania komunikacji, liczba bajtów przychodzących, ilość pakietów przychodzących. Na przykład, aby wykryć sesje HTTPS, które trwają ponad 60 sekund, wystarczy wydać polecenie:

```
debian:~# ./tcpstatflow -y 60 port 443.
/tcpstatflow[5771]: listening on eth0
```

Gdy taka sesja zostanie wykryta program wyświetli komunikat:

```
Potencial tunnel = 192.168.198.128:
3863->
192.168.198.200:443: packets rx=82
tx=86,
 bytes rx=9080 tx=1983, seconds=61
```

Nie jest to jedyne narzędzie, za pomocą którego można badać ruch przechodzący przez router. Pozostałe narzędzia, których pokaźna lista znajduje się w [14], skoncentrowane są raczej na prowadzeniu statystyk połączeń, portów etc., lub pokazywaniu bieżącej działalności „w kablu”. Przykładem może być chociażby jnettop, czy popularny iptraf. Za pomocą obu tych narzędzi administrator jest w stanie zobaczyć, jakie połączenia utrzymują użytkownicy i podjąć odpowiednie kroki.

Z perspektywy administratora można też podejść do problemu od drugiej strony, a mianowicie skoncentrować się nie na wykrywaniu i usuwaniu tuneli kopanych przez użytkowników, ale na podjęciu pewnych kroków prewencyjnych, które utrudnią lub całkowicie uniemożliwią stworzenie ukrytego kanału. Wiedząc, jak działają tunele, łatwo przeciwdziałać ich powstawaniu. Można przykładowo na maszynie firewalla zainstalować proxy ICMP, które

będzie pośredniczyło w wymianie tych pakietów pomiędzy siecią lokalną, a Internetem. Dzięki takiej funkcji moduł mod_icmp [15] skutecznie uniemożliwi tworzenie tuneli opartych o ten protokół (chyba że ktoś wykorzysta ten protokół do stworzenia ukrytego kanału wykorzystującego odstępy czasu do przekazania informacji [16]). Kolejną rzeczą, którą może zrobić administrator, jest ograniczenie liczby prób komunikacji przez niektóre usługi. Opisany w artykule Mailtunnel łatwo można unieszkodliwić nakładając ograniczenia na liczbę prób nawiązania połączenia na porty 25 i 110.

Można to osiągnąć wykorzystując standardowe moduły Iptables. Na początku blokujemy wszystkie połączenia wychodzące na interesujące nas porty:

```
iptables -I FORWARD -s 192.168.128.100
-p tcp -m multiport --destination-port
25,110 --syn -j DROP
```

następnie zezwalamy na określoną liczbę pakietów syn/minutę dla połączeń na odpowiednie porty:

```
iptables -I FORWARD -s 192.168.128.100
-p tcp --dport 110 -m limit --limit
5/min --syn -j ACCEPT
iptables -I FORWARD -s 192.168.128.100
-p tcp --dport 25 -m limit --limit
10/min --syn -j ACCEPT
```

Teraz możemy zaobserwować oczekiwany rezultat na maszynie o adresie 192.168.128.100 – Listing 6.

Jak widać pierwszych 5 pakietów zostało przepuszczonych. Spowodowane jest to domyślną wartością „limit-burst" modułu „limit" w Iptables, która jest ustawiona na 5. Następnie pakiety TCP z ustawioną flagą SYN są przepuszczane z prędkością 1 na 1/5 min. Dzięki takiemu zabiegowi znacznie trudniej będzie stworzyć tunel oparty o protokoły SMTP oraz POP. Podobny zabieg można zastosować wobec innych protokołów.

Kolejną rzeczą, którą administrator powinien rozważyć, jest uruchomienie transparentnego serwera proxy, który przechwytywałby żądania użytkowników. Na taki serwer można nałożyć dodatkowe ograniczenia, które uniemożliwią tworzenie niektórych rodzajów tunelów opartych o HTTP. Stosowanie tego rozwiązania posiada dodatkową zaletę, a mianowicie serwer proxy odciąży trochę nasze łącze, ponieważ część danych będzie znajdować się w cache serwera.

Jeżeli kopanie tuneli jest w sieci zabronione ze względu na duże zużycie zasobów przez programy p2p, to panaceum na wszystkie problemy może się okazać sprawiedliwy podział łącza [16]. Użytkownicy, którzy wykopali sobie tunel i wykorzystują go do ściągania darmowego i legalnego oprogramowania po prostu wykorzystają całe przydzielone im pasmo.

To nie koniec

Przedstawione w artykule pomysły nie są wszystkimi, jakie można zrealizować. Użytkownicy, chcąc obejść zabezpieczenia administratora, wymyślają coraz to nowsze sposoby na pokonanie narzuconych im ograniczeń. W odpowiedzi administratorzy stosują coraz bardziej wymyślne techniki wykrywania i unieszkodliwiania tuneli. Z pewnością nie można powiedzieć, że widać koniec tej wojny. Wręcz przeciwnie, można odnieść wrażenie, że jest to dopiero jej początek. ●

O autorze

Konrad Malewski, absolwent informatyki Politechniki Śląskiej. Obecnie doktorant informatyki na AGH. Administrator amatorskich sieci komputerowych. Zarówno w pracy, jak i w życiu prywatnym interesuje się programowaniem oraz bezpieczeństwem aplikacji sieciowych.

W sieci

- *http://linux-igd.sourceforge.net/* - Tutaj znajdziemy demona IGD pod Linux.
- *http://pupnp.sourceforge.net/* - Biblioteka oraz SDK, potrzebne do uruchomienia IGD.
- *http://miniupnp.free.fr/* - Klient IGD, za pomocą którego możemy konfigurować przekierowania.
- *http://www.upnp.org/download/UPnPDA10_20000613.html* – Dokument opisujący szczegółowo architekturę UPnP
- *http://www.hamachi.cc/* - Strona producenta programu Hamachi wykorzystywanego do tworzenia tuneli na potrzeby gier.
- *http://www.detached.net/mailtunnel.html* - Program służący do tunelowania połączeń TCP w przesyłkach email.
- *http://home.agh.edu.pl/~koyot/mailtunnel/mailtunnel-0.3.tar.gz* - zmodyfikowana na potrzeby stworzenia VPN-a wersja programu Mailtunnel
- http://gray-world.net/tools/ftp-tunnel.tgz – program do tunelowania za pomocą FTP.
- *http://www.dest-unreach.org/socat/* - program przekazujący dane pomiędzy różnymi źródłami.
- *ftp://sunsite.unc.edu/pub/Linux/system/network/misc/udprelay-linux.tgz* – UDP bouncer
- *http://sourceforge.net/projects/proxytunnel/*
- *http://www.nocrew.org/software/httptunnel.html*
- *http://gray-world.net/tools/icmptunnel.tar.gz*
- *http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html* – Lista narzędzi służących do monitorowania sieci i nie tylko
- *http://www.geocities.com/fryxar/*
- *http://www.multicians.org/timing-chn.html* – Ukryte kanały, które wykorzystują zdarzenia czasowe do przekazania informacji.
- *http://sp9wun.republika.pl/linux/shaperd_cbq.html* – Strona domowa demona, za pomocą którego ustawimy „sprawiedliwy podział łącza"