Sieci komputerowe

Wykład 12: Bezpieczeństwo sieci

Marcin Bieńkowski

Instytut Informatyki
Uniwersytet Wrocławski

Spis treści

Ataki

Poruszane kwestie:

- Nieuprawniony dostęp do informacji
  - przechowywanych na komputerach
  - wysyłanych przez sieć
- Zakłócanie pracy systemu/systemów (ataki DoS)

Atak = próba uzyskania nieuprawnionego dostępu do usług lub/i informacji

Poruszane kwestie:

- Nieuprawniony dostęp do informacji
  - przechowywanych na komputerach
  - wysyłanych przez sieć
- Zakłócanie pracy systemu/systemów (ataki DoS)

Atak = próba uzyskania nieuprawnionego dostępu do usług lub/i informacji

Źródło ataków nr 1: Źle zaprojektowane protokoły

Podsłuchiwanie (1)

Ethernet

- Koncentratory (huby) $\rightarrow$ tryb nasłuchu (*promiscous mode*).
- Przełączniki sieciowe: przepełnianie pamięci CAM.
  - Przełącznik ma sprzętową tablicę haszującą (CAM = *content addressable memory*) z wpisami „adres MAC $\rightarrow$ port”.
  - Zmieniając adres MAC można zalać CAM nowymi wpisami $\rightarrow$ przełącznik przejdzie w tryb uczenia się.
- *ARP spoofing* $\rightarrow$ animacja.
- Częściowe rozwiązanie: dobre i drogie przełączniki.

Podsłuchiwanie (2)

Wyższe warstwy: ataki typu *man-in-the-middle*

- Podsłuchujemy przez „wstawienie” swojego komputera pomiędzy strony nieszyfrowanej komunikacji.
- Jak przekierowac ruch przez siebie?
  - Jesteśmy odpowiednim routerem.
  - *RIP spoofing*
  - Własny serwer DHCP.
  - Własny bezprzewodowy punkt dostępowy (własny punkt nazywający się `stud-wifi`?)
  - Zatruwanie pamięci podręcznej serwera DNS.
  - ...

IP spoofing

Fałszowanie adresu źródłowego IP

- Jedyny problem: sprawić, żeby taki pakiet został przepuszczony przez routery.
- Rozwiązanie: weryfikacja (tzw. *ingress filtering*).
  - Przykładowo z interfejsu sieciowego podłączonego do sieci `192.168.0.0/24` nie powinien nadejść pakiet z źródłowym IP `200.200.200.200`
  - Skuteczne jeśli router jest blisko nadawcy.

Źródło ataków nr 2: Zła implementacja

Błędy implementacji

Brak sprawdzania poprawności wprowadzanych danych

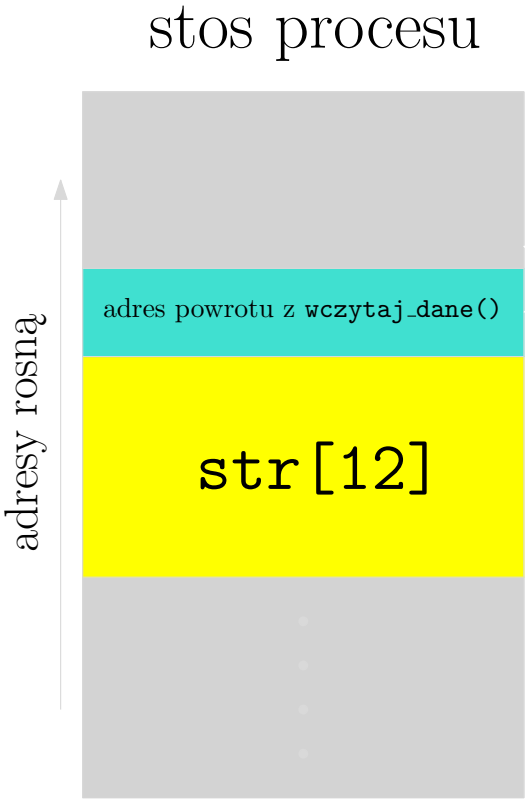
- Błędy przepełnienia bufora, . . /, *SQL injection* ...
- Atakujący zmusza naszą aplikację do wykonania pewnych nieprzewidzianych operacji.

Przykład 1: przepełnienie bufora

```
void wczytaj_dane () {
    char str[12];
    scanf ("\%s", str);
}
```

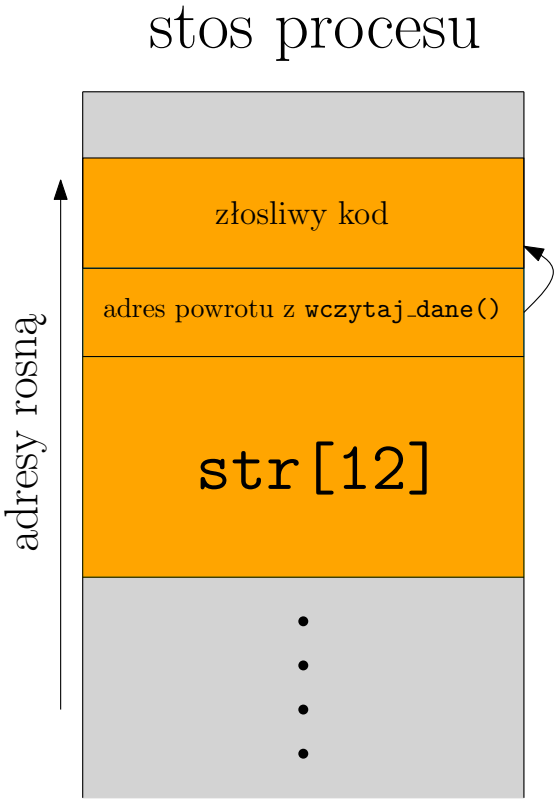
Atakujący wpisuje

- Jakieś 12 znaków
- Odpowiedni adres powrotu
- Kod maszynowy do uruchomienia



Przykład 1: przepełnienie bufora

```
void wczytaj_dane () {
    char str[12];
    scanf ("\%s", str);
}
```



Atakujący wpisuje

- Jakieś 12 znaków
- Odpowiedni adres powrotu
- Kod maszynowy do uruchomienia

Przykład 2: atak typu ../

Pomocniczy skrypt WWW

- `skrypt` jest w katalogu `/var/www/`.
- `skrypt` jest programem wyświetlającym zawartość pliku.
- Przykładowo `http://jakas.domena/skrypt?plik=test` wyświetla zawartość pliku `/var/www/test`.
- Co zrobi wejście na stronę `http://jakas.domena/skrypt?plik=../../etc/passwd`?

Przykład 2: atak typu ../

Pomocniczy skrypt WWW

- `skrypt` jest w katalogu `/var/www/`.
- `skrypt` jest programem wyświetlającym zawartość pliku.
- Przykładowo `http://jakas.domena/skrypt?plik=test` wyświetla zawartość pliku `/var/www/test`.
- Co zrobi wejście na stronę `http://jakas.domena/skrypt?plik=../../etc/passwd`?

Przykład 3: błąd typu *SQL injection*

Aplikacja WWW odwołująca się do bazy

- Skrypt dostaje przez formularz argument `$user` i wykonuje polecenie `mysql_query("SELECT * FROM tab WHERE user = '$user'");`
- Podajemy w tym polu: `x' OR '1'='1`
- Albo lepiej: `x'; DROP TABLE tab; --`

Przykład 3: błąd typu *SQL injection*

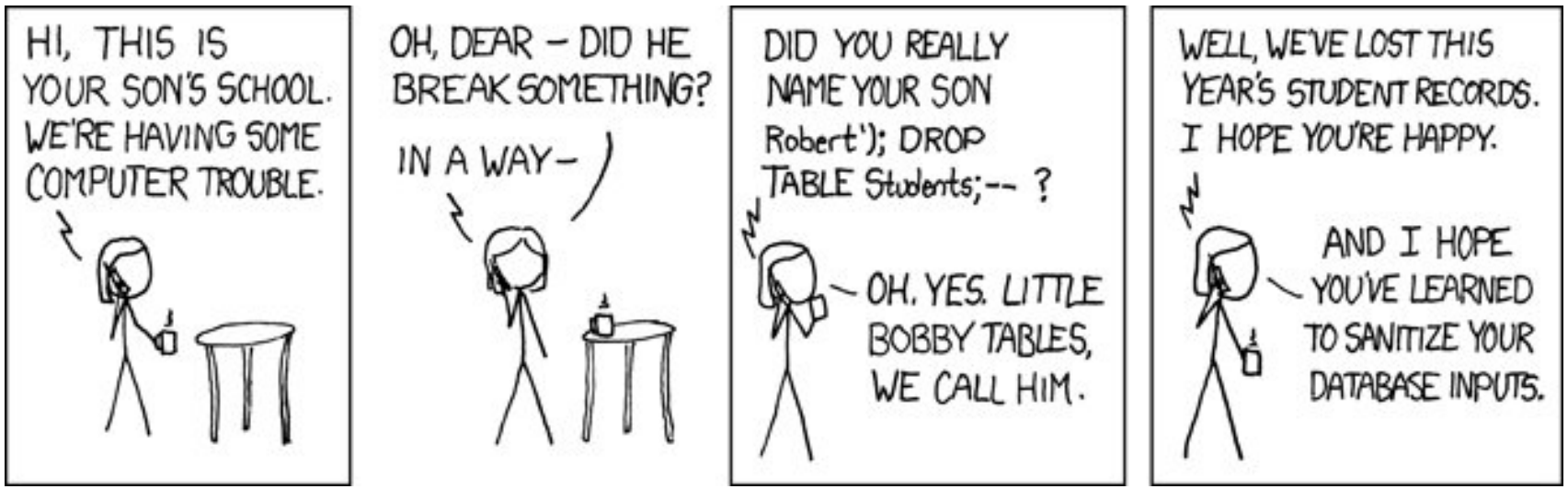
Aplikacja WWW odwołująca się do bazy

- Skrypt dostaje przez formularz argument `$user` i wykonuje polecenie `mysql_query("SELECT * FROM tab WHERE user = '$user'");`
- Podajemy w tym polu: `x' OR '1'='1`
- Albo lepiej: `x'; DROP TABLE tab; --`

Przykład 3: błąd typu *SQL injection*

Aplikacja WWW odwołująca się do bazy

- Skrypt dostaje przez formularz argument `$user` i wykonuje polecenie `mysql_query("SELECT * FROM tab WHERE user = '$user'");`
- Podajemy w tym polu: `x' OR '1'='1`
- Albo lepiej: `x'; DROP TABLE tab; --`



Obrazek ze strony `http://xkcd.com/`

Przykład 4: wysyłanie części RAM procesu

- Protokół definiuje, że możemy otrzymać od serwera jakiś fragment przechowywanej przez niego tablicy.
- Klient prosi go o więcej niż rozmiar tablicy + serwer nie sprawdza tego rozmiaru.
- Serwer odsyła część pamięci procesu.
- Przykład: SSL heartbleed

- Inny wariant: standardy określają minimalną ilość danych do wysłania (np. ramka ethernetowa).
- Jeśli pamięć na wypełnienie nie zostanie wyzerowana, to wyślemy część pamięci procesu.

Błędy w implementacji

Programista:

- ZAWSZE sprawdzać poprawność danych od użytkownika
- W szczególności nigdy nie zakładać, że użytkownik używa przepisowego klienta usługi.

Administrator:

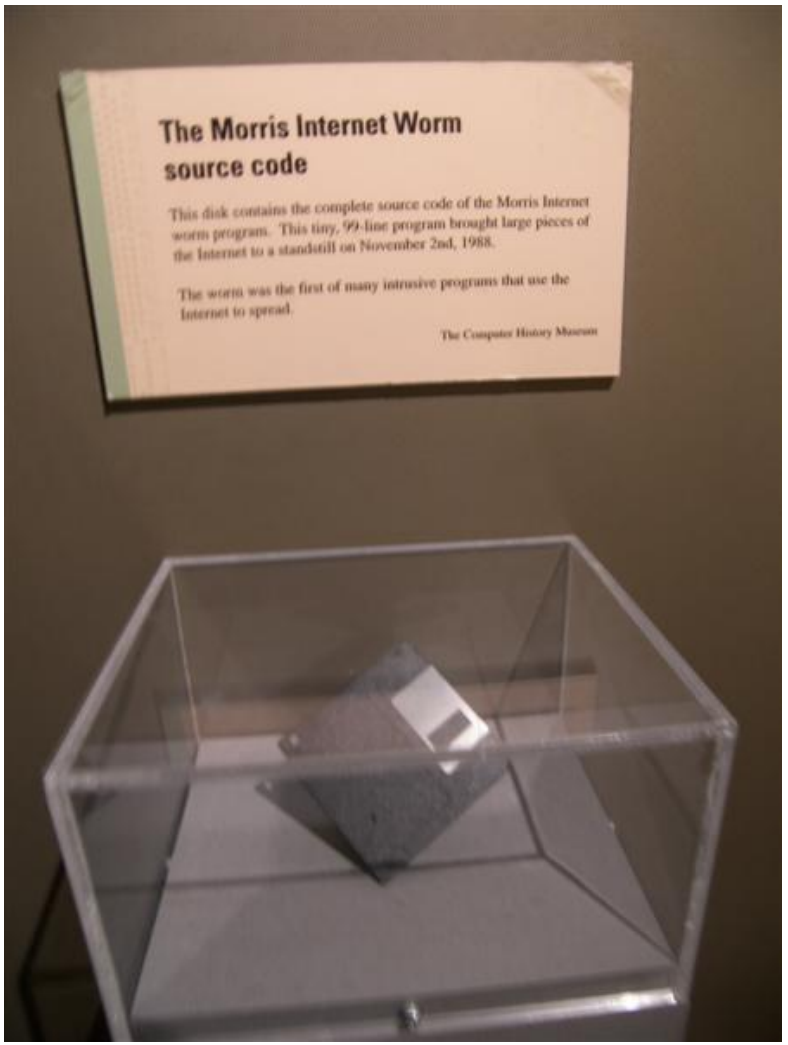
- Częsta aktualizacja systemu
- Różne ogólne techniki zapobiegawcze, np. zabronienie wykonywania programów na stosie.
- Program A działający z uprawnieniami B ma lukę $\rightarrow$ atakujący może wykonać to, na co pozwalają uprawnienia B $\rightarrow$ złota zasada minimalnych uprawnień.
- Tylko potrzebne usługi i tylko tym, którym są one konieczne.

Exploits

- Luki i gotowe sposoby ich wykorzystania (*exploits*) są publikowane na specjalistycznych stronach (np. Security Focus).
  - Te programy są często wykorzystywane przez niedoświadczonych włamywaczy.
  - Często wystarczy drobna zmiana, typu zmiana portu na którym nasłuchuje SSH, żeby taki program nie działał (*security by obscurity*).

Robaki internetowe (*worms*)

- Luki wykorzystywane są też w automatyczny sposób przez *robaki internetowe*.
- Zainfekowane komputery stają się częścią większej sieci (*botnet*)
  - rozsyłanie spamu
  - przeprowadzanie ataków DDoS



Skanowanie portów

Wyszukiwanie otwartych portów

- `nmap`
- Przykładowe rodzaje:
  - `connect` scan,
  - `SYN` scan,
  - ustawianie różnych dziwnych flag (np. sama flaga `ACK`).
- Najczęściej wykorzystywane narzędzie w filmach!
  `https://nmap.org/movies/`

Podatność na ataki

Omówiliśmy dwie przyczyny:

- Złe zaprojektowanie protokołów komunikacyjnych...
- ...i zła ich implementacja.

Ale po co to wszystko?

Każdy system jest tak bezpieczny jak jego najsłabsze ogniwo...

Podatność na ataki

Omówiliśmy dwie przyczyny:

- Złe zaprojektowanie protokołów komunikacyjnych...
- ...i zła ich implementacja.

Ale po co to wszystko?

Każdy system jest tak bezpieczny jak jego najsłabsze ogniwo...

Czynnik ludzki

Po co włamywać się na serwer, skoro można zadzwonić i spytać o hasło?

Kevin Mitnick

Phishing

Uwaga!
Użytkownik ???? przesyła Tobie następującą wiadomość:

Witaj!, KOCHAJMY SIĘ

Dalsza część wiadomości dostępna jest tutaj:
http://www5.nasza-klasa.pl/profile/872276
(http://www5.nasza-klasa.pl.rdir84.com/profile/?873105
Copyright (c) 2006-2008 nasza-klasa.pl, korzystanie z serwisu oznacza akceptacje regulaminu.

Czynnik ludzki, cd.

Wykorzystanie tańczących świnek daje 95% szansy na uruchomienie złośliwego kodu na cudzym komputerze!

Obrazek ze strony `http://www.securingjava.com/chapter-one/chapter-one-7.html`

Denial of Service

czyli blokowanie dostępu do usług

Blokowanie dostępu do usługi

Denial of service

- Cel: wyczerpanie zasobów systemu operacyjnego albo zatkanie łącza.
- Jakie zasoby możemy wyczerpać:
  - moc obliczeniowa,
  - limit jednoczesnych połączeń (`connect()`),
  - *SYN flood* = wysyłanie samych segmentów SYN $\rightarrow$ zapełnienie tworzonej przez `listen()` kolejki połączeń oczekujących na nawiązanie.

Odmiany DoS

- DoS wykonywany z już przejętych komputerów lub/i ze sfałszowanych adresów źródłowych IP.
- DDoS = rozproszony DoS (wykorzystanie wielu komputerów, np. uprzednio zainfekowanych robakiem internetowym).
- Odbity (*reflected*) DoS
  - Wysyłamy do różnych komputerów komunikat z fałszywym adresem źródłowym = adres IP ofiary $\rightarrow$ komputery odpowiadają ofierze.
  - Komunikat = segment SYN, ICMP *echo request* (ping).
  - Przykład: *smurf attack* = wysyłamy ping na adres *broadcast* z podmienionym adresem IP ofiary.

Odmiany DoS

- DoS wykonywany z już przejętych komputerów lub/i ze sfałszowanych adresów źródłowych IP.
- DDoS = rozproszony DoS (wykorzystanie wielu komputerów, np. uprzednio zainfekowanych robakiem internetowym).
- Odbity (*reflected*) DoS
  - Wysyłamy do różnych komputerów komunikat z fałszywym adresem źródłowym = adres IP ofiary $\rightarrow$ komputery odpowiadają ofierze.
  - Komunikat = segment SYN, ICMP *echo request* (ping).
  - Przykład: *smurf attack* = wysyłamy ping na adres *broadcast* z podmienionym adresem IP ofiary.

Obrona przed DDoS

Przed DDoS można się bronić tylko jeśli pochodzi z jednego geograficznego obszaru

- Problem: ustalenie źródła ataków (źródłowe adresy IP są podrobione), potem można zadzwonić do odpowiedniego administratora
- ICMP Traceback
  - Każdy router dla przesyłanego pakietu, z małym prawdopodobieństwem (ok. 1/20.000), wysyła do odbiorcy dodatkowo komunikat ICMP
  - Komunikat zawiera informacje o przesyłanym właśnie przez router pakiecie, informacje o routerze, etc.

Szyfrowanie = sposób na podsłuchiwanie

SSL

- Warstwa szyfrująca pomiędzy warstwą transportową a warstwą aplikacji.
- Większość popularnych usług ma swoje warianty wykorzystujące SSL działające na tym samym lub innym porcie (HTTPS, POP3+SSL, SMTP+SSL, ...)

SSL (przypomnienie)

Standardowe podejście oparte od kryptografię asymetryczną

- Serwer wysyła klientowi swój klucz publiczny
- Klient generuje symetryczny klucz sesji (np. AES) i wysyła go szyfrując go kluczem publicznym serwera.
- Od tej pory połączenie szyfrowane jest kluczem sesji.
- Klient się uwierzytelnia podając swoje hasło.

Uwierzytelnianie serwera

- Za pomocą certyfikatu (klucz publiczny serwera jest podpisany przez CA i możemy zweryfikować autentyczność tego podpisu).

SSH (Secure SHell)

- Standardowe narzędzie pracy zdalnej (w terminalu tekstowym).
- Następca nieszyfrowanego `telnet`a.
- Proste użycie: `ssh user@host`
- Mechanizmy szyfrowania jak przy SSL.
- Co z uwierzytelnianiem serwera?

Serwerze, czy to ty?

Znowu nie wiemy, czy łączymy się z dobrym serwerem!

- Przy pierwszym połączeniu:

```
The authenticity of host 'ssh-server.example.com
(12.18.429.21)' can't be established.
RSA key fingerprint is
98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d.
Are you sure you want to continue connecting?
```

  tzn. serwer przesyła nam klucz publiczny, program klienta oblicza funkcje skrótu klucza.
- Czynnik ludzki: kto dzwoni do administratora serwera i weryfikuje, czy funkcja skrótu (*fingerprint*) ma poprawną wartość?
- Po zaakceptowaniu klucz publiczny serwera zapisany w `~/.ssh/known_hosts`.

Serwerze, czy to ty?

Znowu nie wiemy, czy łączymy się z dobrym serwerem!

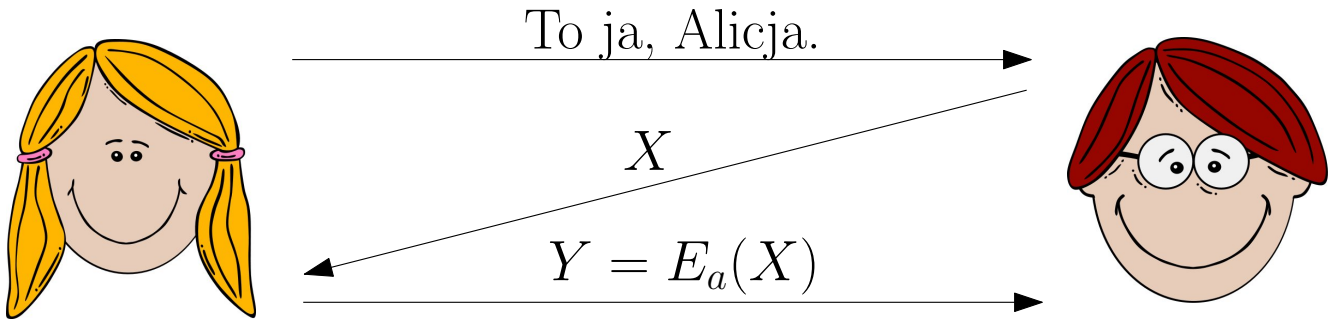
- Przy pierwszym połączeniu:
  ```
  The authenticity of host 'ssh-server.example.com
  (12.18.429.21)' can't be established.
  RSA key fingerprint is
  98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d.
  Are you sure you want to continue connecting?
  ```
  tzn. serwer przesyła nam klucz publiczny, program klienta oblicza funkcje skrótu klucza.
- Czynnik ludzki: kto dzwoni do administratora serwera i weryfikuje, czy funkcja skrótu (*fingerprint*) ma poprawną wartość?
- Po zaakceptowaniu klucz publiczny serwera zapisany w `~/.ssh/known_hosts`.

Uwierzytelnianie klienta

- Może po prostu podać hasło do konta.
- Albo... tak, to już było:
  - klient = Alicja
  - serwer = Bob
  - Serwer zna klucz publiczny klienta, bo klient go uprzednio zapisał w pliku `authorized_keys` na serwerze.

Uwierzytelnianie klienta

- Może po prostu podać hasło do konta.
- Albo... tak, to już było:



klucz prywatny: a — zna: A

klucz publiczny: A — sprawdza, czy $E_A(Y) = X$

  - klient = Alicja
  - serwer = Bob
  - Serwer zna klucz publiczny klienta, bo klient go uprzednio zapisał w pliku `authorized_keys` na serwerze.

Zastosowania

Za pomocą `ssh` można też:

- uruchamiać zdalnie polecenia:
  `ssh serwer "polecenie_zdalne"`
- kopiować pliki na zdalny serwer:
  `scp plik_lokalny serwer:/katalog/zdalny/`
- albo z powrotem:
  `scp -r serwer:/katalog/zdalny/ /katalog/lokalny/`

Co z innymi usługami, nie posiadającymi wariantu SSL?

Tunelowanie

Model warstwowy

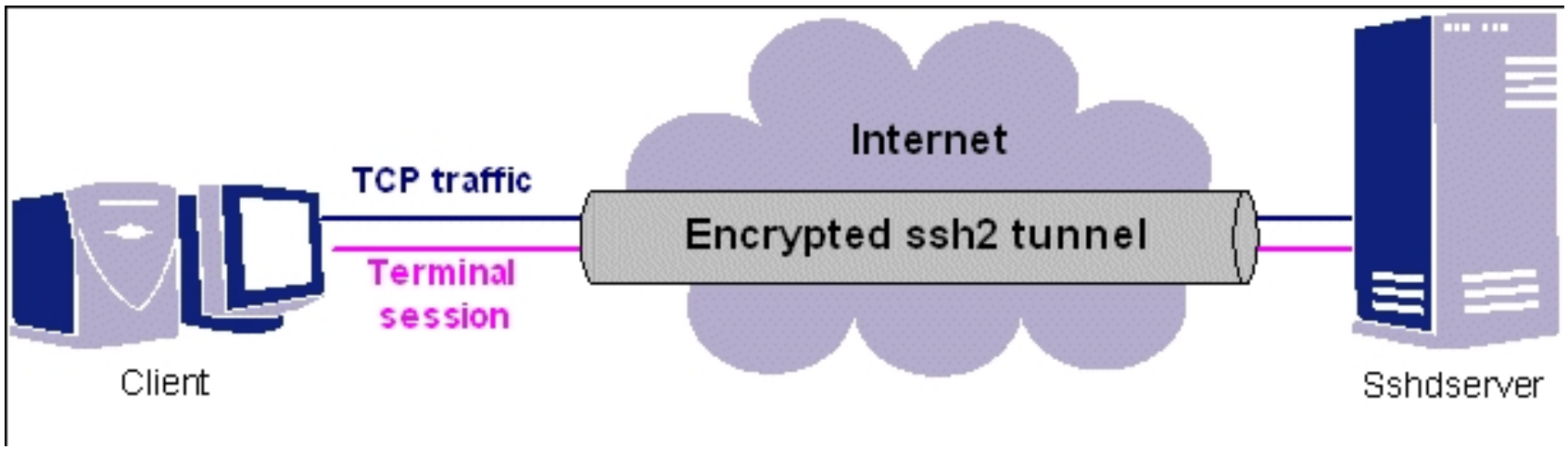
- protokół warstwy $i + 1$ jest przesyłany jako *dane* protokołu warstwy i.

Tunelowanie

- Przesyłanie pewnych usług sieciowych za pomocą innych usług sieciowych w sposób łamiący standardowy model warstwowy.
- Cel: zestawianie wirtualnego bezpośredniego połączenia między dwoma odległymi komputerami.

Przykład 1: Segmenty TCP w SSH

- Jeśli mamy konto na zdalnej maszynie, możemy szyfrować połączenia z tymi usługami za pomocą `ssh`.
- Przykładowo `ssh -L 4025:localhost:25 user@zdalny-serwer` przekazuje segmenty TCP skierowane do portu 4025 lokalnego komputera w (zaszyfrowanych) danych SSH do `zdalny-serwer` (do portu 22) a następnie do portu 25 `zdalny-serwer`.



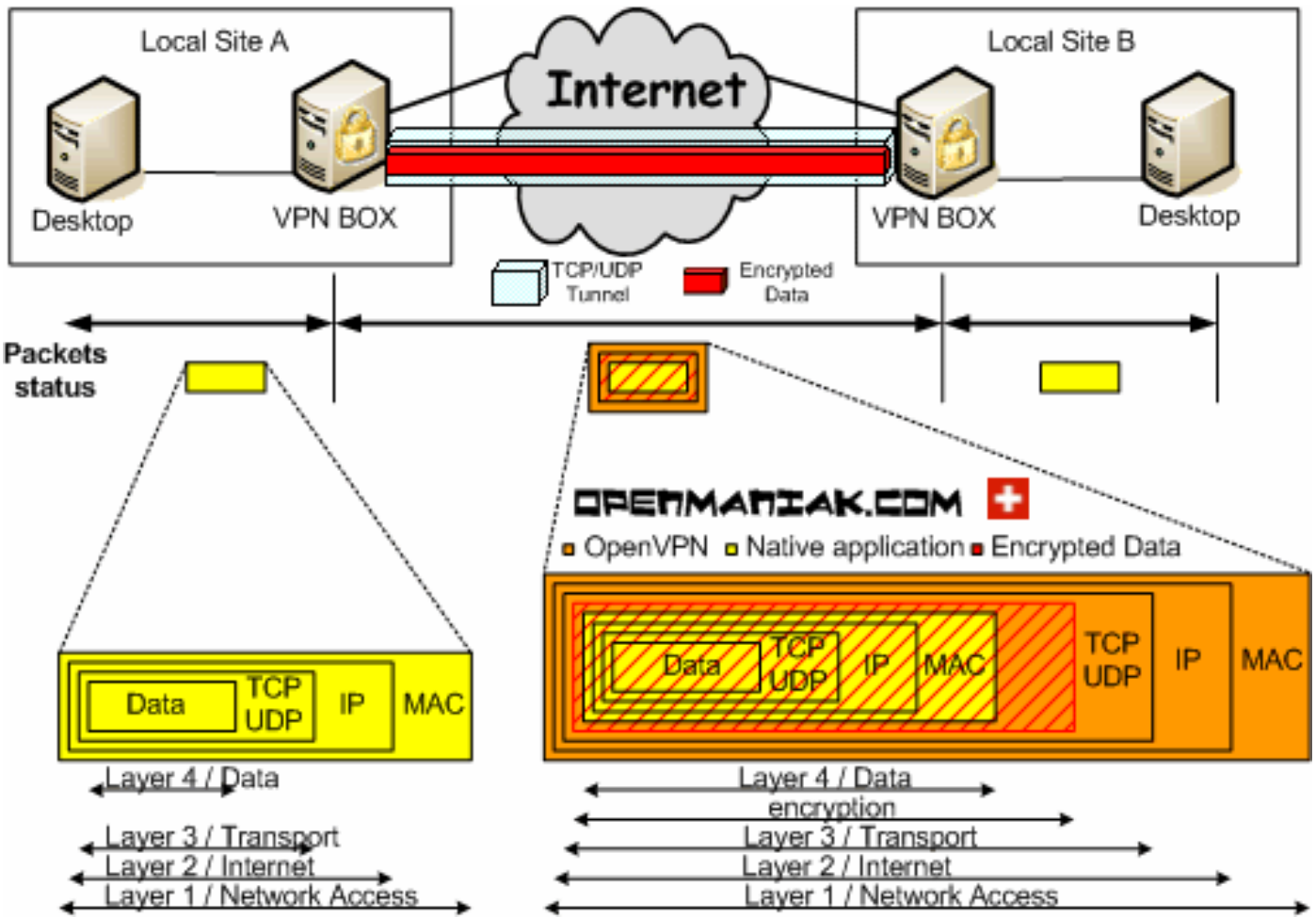
Obrazek ze strony `http://www.ssh.com/support/documentation/online/ssh/adminguide/32/Port_Forwarding.htm`

Przykład 2: VPN

VPN = Wirtualna sieć prywatna (*Virtual Private Network*)

- Zadanie: mamy dwie sieci połączone Internetem i chcemy zrobić z nich jedną logiczną sieć.
- Transmisja wewnątrz każdej z nich jest bezpieczna, ale transmisja w Internecie już nie.
- Przykład: dostęp do firmowej sieci z domu.

Przykład 2: VPN, cd.



Obrazek ze strony `http://openmaniak.com/openvpn.php`

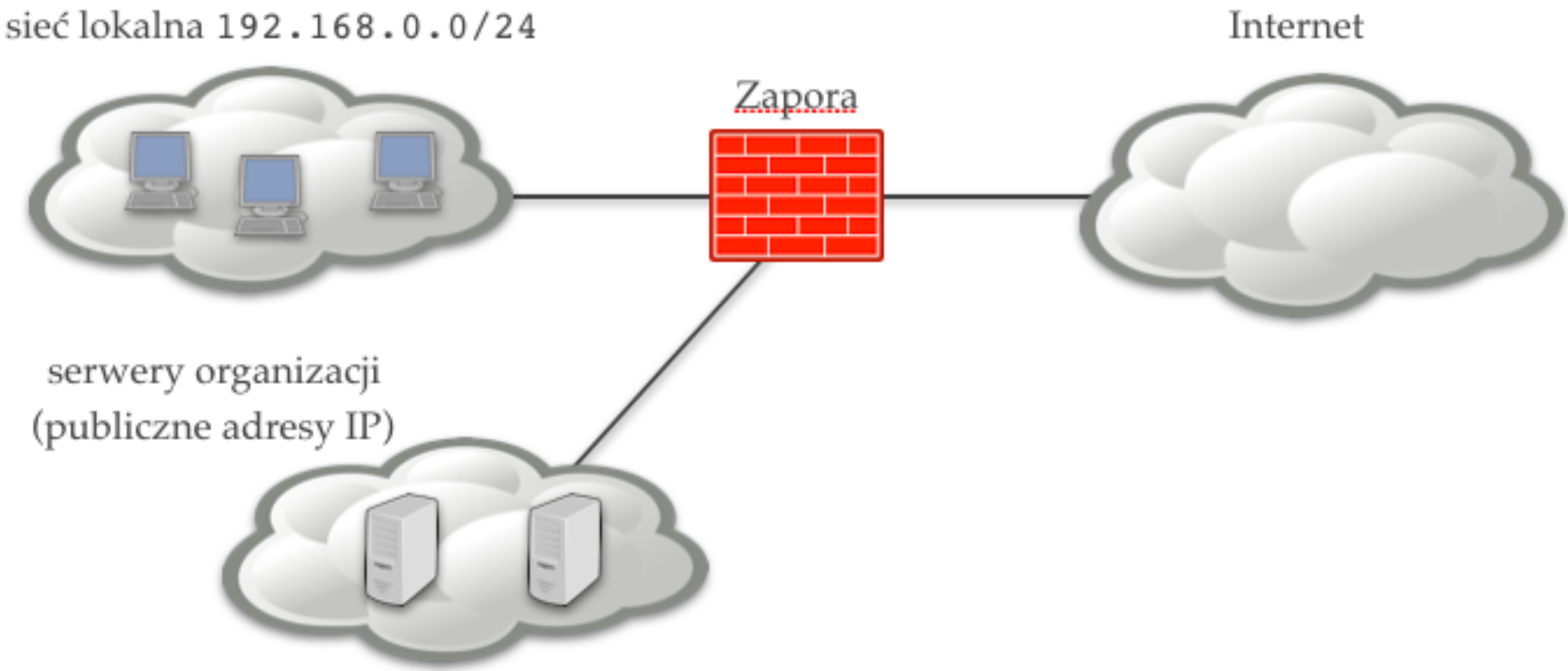
Zapora (firewall)

Po co?

- Pierwsza linia obrony
- Rejestrowanie i kontrolowanie dostępu do usług
- Rejestrowanie ataków i skanów portów

Gdzie?

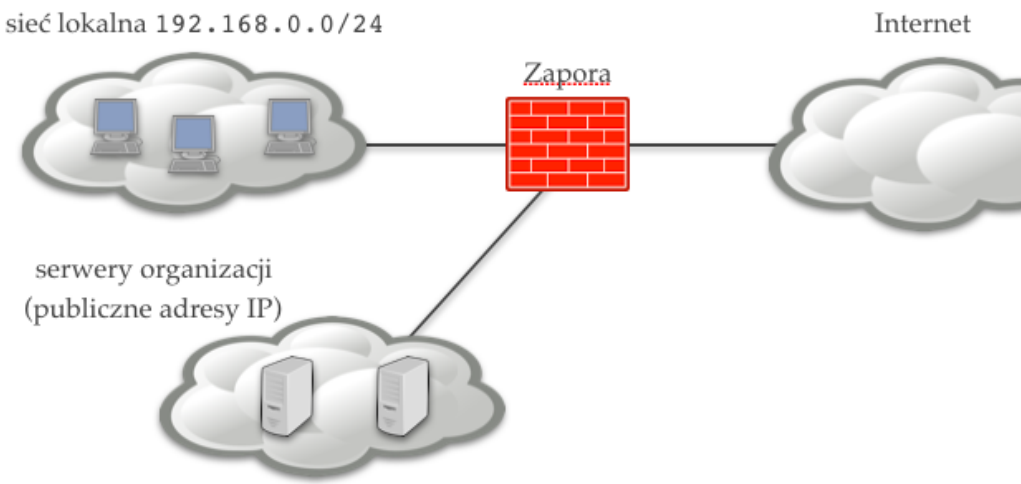
Często zapora jest osobnym komputerem, pełniącym też funkcję routera:



Polityka

Zapora realizuje pewną politykę bezpieczeństwa. Przykładowo:

- Wpuszcza pakiety do serwerów organizacji (do określonych portów).
- Wpuszcza pakiety do portu SSH serwerów tylko z sieci wewnętrznej.
- Nie wpuszcza pakietów do sieci wewnętrznej.
- ...



Zapora = filtr pakietów

Podział ze względu na warstwę działania:

- (3) filtry proste
- (4) filtry stanowe
- (5) filtry działające w warstwie aplikacji

Klasyfikacja filtrów pakietów

Filtry proste

- Analizują tylko nagłówki IP.
- Szybkie, bardzo nieprecyzyjne.

Filtry stanowe

- Analizują nagłówki IP i TCP
- Śledzą trójstanowe uzgodnienie, pamiętają stan połączenia
- „Rozumieją” numery sekwencyjne $\rightarrow$ lepsza odporność na fałszowanie nagłówków.

Filtry działające w warstwie aplikacji

- Analizują zawartość segmentów i datagramów
- Np. w przypadku FTP „rozumieją” że trzeba otworzyć odpowiedni port na dane.
- To coś innego niż *zapory aplikacji* (które analizują *wywołania systemowe* aplikacji)

Klasyfikacja filtrów pakietów

Filtry proste

- Analizują tylko nagłówki IP.
- Szybkie, bardzo nieprecyzyjne.

Filtry stanowe

- Analizują nagłówki IP i TCP
- Śledzą trójstanowe uzgodnienie, pamiętają stan połączenia
- „Rozumieją” numery sekwencyjne $\rightarrow$ lepsza odporność na fałszowanie nagłówków.

Filtry działające w warstwie aplikacji

- Analizują zawartość segmentów i datagramów
- Np. w przypadku FTP „rozumieją” że trzeba otworzyć odpowiedni port na dane.
- To coś innego niż *zapory aplikacji* (które analizują *wywołania systemowe* aplikacji)

Klasyfikacja filtrów pakietów

Filtry proste

- Analizują tylko nagłówki IP.
- Szybkie, bardzo nieprecyzyjne.

Filtry stanowe

- Analizują nagłówki IP i TCP
- Śledzą trójstanowe uzgodnienie, pamiętają stan połączenia
- „Rozumieją” numery sekwencyjne $\rightarrow$ lepsza odporność na fałszowanie nagłówków.

Filtry działające w warstwie aplikacji

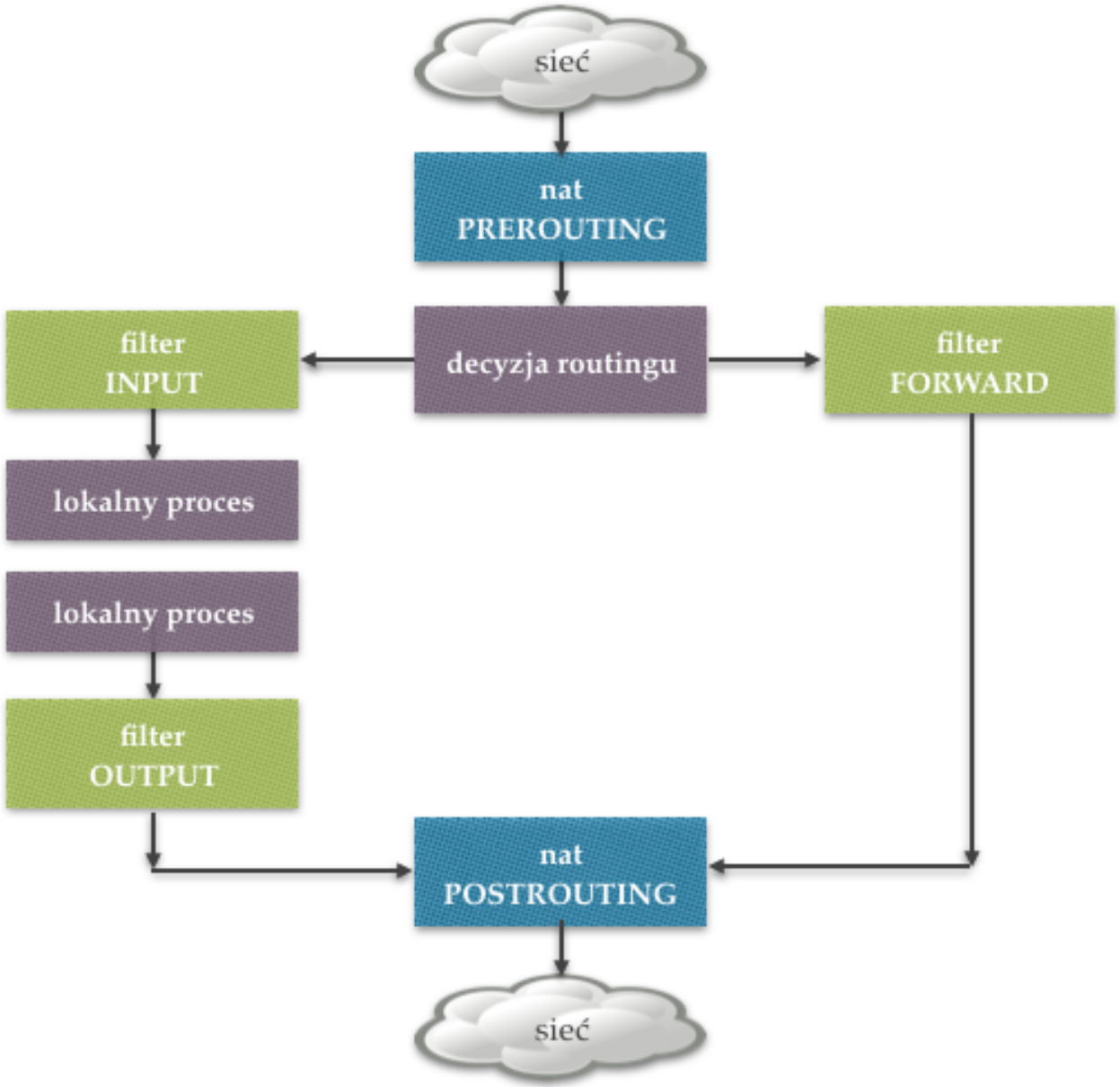
- Analizują zawartość segmentów i datagramów
- Np. w przypadku FTP „rozumieją” że trzeba otworzyć odpowiedni port na dane.
- To coś innego niż *zapory aplikacji* (które analizują *wywołania systemowe* aplikacji)

Netfilter / iptables

Netfilter / iptables = filtr pakietów w Linuksie

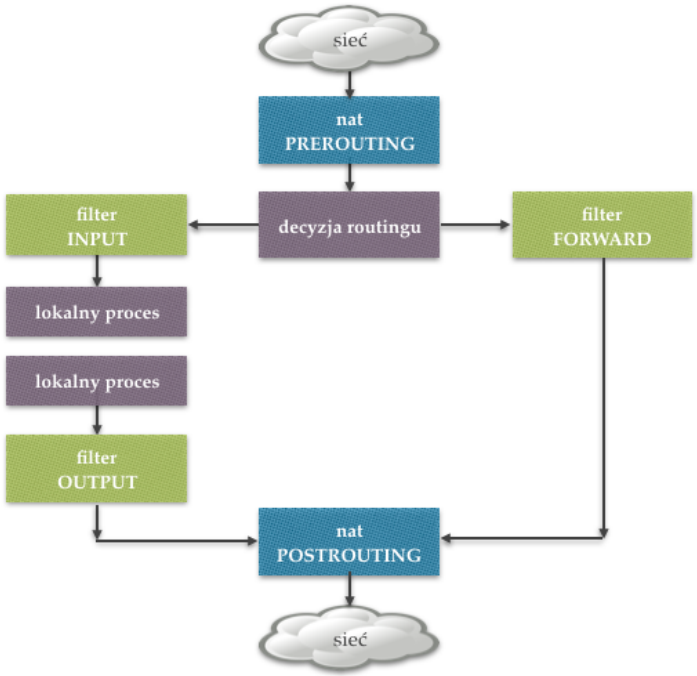
- Filtr stanowy.
- Elementy filtra działającego w warstwie aplikacji (możliwość śledzenia połączeń niektórych protokołów (np. FTP).

Przetwarzanie pakietów (1)



Przetwarzanie pakietów (2)

- 5 głównych modułów (*chains*)
  - filter INPUT
  - filter OUTPUT
  - filter FORWARD
  - nat PREROUTING
  - nat POSTROUTING
- Każdy moduł to ciąg reguł (warunek + akcja).
- Reguły sprawdzane po kolei.
- Domyślna polityka modułu (niepasujące pakiety).



Przykład: Stacja robocza z usługą SSH

Przykład

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Czy to bedzie działać?

- Połączenia z zewnątrz do portu SSH: tak.
- Pakiety wychodzące np. do portu 80 bedą wypuszczane, ale odpowiedzi na nie bedą odrzucane!

Jak sobie z tym radzić?

Przykład: Stacja robocza z usługą SSH

Przykład

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Czy to bedzie działać?

- Połączenia z zewnątrz do portu SSH: tak.
- Pakiety wychodzące np. do portu 80 bedą wypuszczane, ale odpowiedzi na nie bedą odrzucane!

Jak sobie z tym radzić?

Stan połączeń

W przypadku filtrów prostych: heurystyki

- Zazwyczaj lokalne porty są >= 32678.
- Można zatem dodać regułę

```
iptables -A INPUT -p tcp -m multiport
    --dports 32768:65535 -j ACCEPT
```

- Nieprecyzyjne i nie bardzo bezpieczne.

W przypadku filtrów stanowych

- `iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT`

Odrzucanie połączeń

Pakiety można odrzucać na dwa sposoby:

- Zgodny z RFC: przez `-j REJECT`. Nasz serwer odpowie wtedy komunikatem IMCP `icmp-port-unreachable`
- „Ciche” porzucenie pakietu przez `-j DROP`. Serwer nie odpowiada nic. Utrudnia to pracę skanerów portów, gdyż muszą wtedy czekać na *timeout* dla połączenia.

Przykładowe warunki

W cześci warunku reguły mogą znajdować się następujące elementy:

- `-i eth0`: pakiet przychodzi z interfejsu `eth0`
- `-o eth1`: pakiet wychodzi przez interfejs `eth1`
- `-p tcp|udp|icmp`: pakiet jest pakietem danego protokołu
- `-s 10.1.2.0/24`: pakiet pochodzi z danego adresu IP (klasy adresów)
- `-d 10.0.2.0/24`: pakiet idzie do danego adresu IP
- `--dport`: pakiet jest wysłany na określony port
- `--sport`: pakiet jest wysłany z określonego portu

Przykład: zapobieganie fałszowaniu adresów IP

Na interfejsie `eth0` należącym do sieci zewnętrznej nie powinny pojawiać się pakiety z adresami źródłowymi należącymi do lokalnej sieci (`192.168.0.0/24`).

```
iptables -A INPUT -i eth0 -s 192.168.0.0/24 -j DROP
```

Analizowanie danych warstwy aplikacji

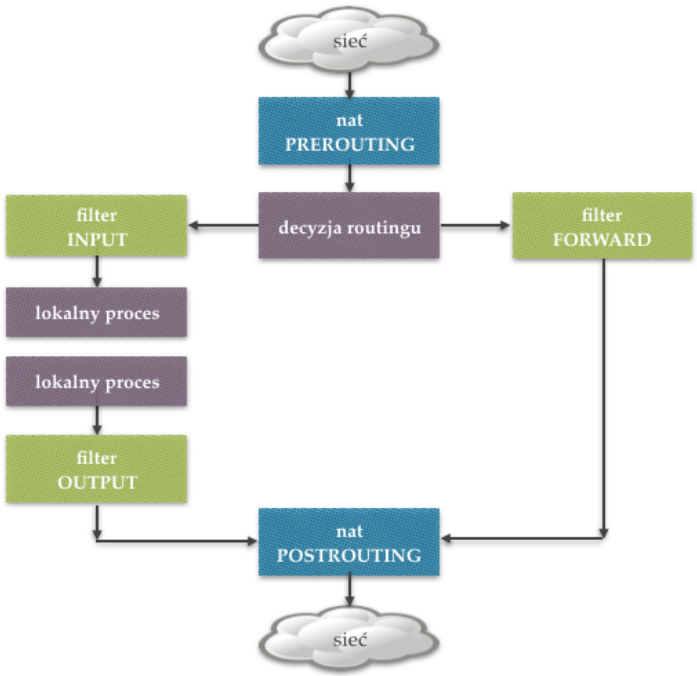
Po włączeniu odpowiednich modułów, za pomocą reguły
`iptables -A INPUT -m state -state RELATED -j ACCEPT`
wpuszczane będą pakiety jakoś związane z istniejącymi połączeniami.

Przykładowo moduł `ip_conntrack_ftp` analizuje transmisję sterującą protokołu FTP i otwiera odpowiednie porty dla połączenia, którym będą przesyłane dane.

Moduły nat

Operacje związane z NAT

- Moduł `nat POSTROUTING`: podmiana źródłowych adresów IP (ostatnia czynność przed wysłaniem pakietu)
- Moduł `nat PREROUTING`: podmiana docelowych adresów IP (np. przekierowanie do innego komputera)



Źródłowy NAT

Podmiana adresu źródłowego pakietu:

- `iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4`
- Maskarada = szczególny przypadek źródłowego NAT. Niech `eth1` będzie połączony z siecią lokalną (z prywatnymi adresami), zaś `eth0` z Internetem. Polecenie:

  `iptables -t nat -A POSTROUTING -i eth1 -o eth0 -j MASQUERADE`

  - Źródłowy adres pakietów będzie równy adresowi IP `eth0`.
  - Przeznaczone dla interfejsów, których adresy mogą zmieniać się dynamicznie.

Docelowy NAT

Podmiana adresu docelowego pakietu:

- `iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 5.6.7.8`
- Przekierowanie = szczególny przypadek DNAT. Przykładowo przekierowanie ruchu wychodzącego do serwerów WWW do lokalnego serwera proxy:

```
iptables -t nat -A PREROUTING -i eth0
    -p tcp --dport 80
    -j REDIRECT --to-port 8080
```

Lektura dodatkowa

- Kurose, Ross: rozdział 8