

---

# Bezpieczeństwo sieci

Sieci komputerowe

Wykład 13

---

*Marcin Bieńkowski*



# Założenia

---

- ❖ **Atakujący kontroluje pewną część sieci**
  - ♦ komputer(y)
  - ♦ router(y) / przełącznik(i)
  - ♦ nośniki (kable, fale radiowe)
  - ♦ ...



# Co można zepsuć?

---

## ❖ **Poufność**

- ♦ atakujący czyta nasze dane

## ❖ **Integralność**

- ♦ atakujący podszywa się pod nas
- ♦ atakujący modyfikuje nasze wiadomości

## ❖ **Dostępność**

- ♦ atakujący uniemożliwia nam komunikację



---

# Podśluchiwanie i podszywanie się

---



# Podśluchiwanie we współdzielonym kanale

---

- ❖ **Nieprzełączany Ethernet (oparty na koncentratorach)**
  - ✦ Wszyscy słyszą wszystkie ramki → tryb nasłuchu (*promiscuous mode*).
  
- ❖ **Sieci WLAN ze współdzielonym kluczem (WEP, WPA Personal)**
  - ✦ Klucze sesji są przesyłane przy związywaniu klienta z punktem dostępowym (AP).
  - ✦ Wystarczy je podsłuchać i możemy deszyfrować całą komunikację między AP i klientem.
  
- ❖ **Przełączany Ethernet**
  - ✦ Przełącznik ma sprzętową tablicę haszującą (CAM = *content addressable memory*).
  - ✦ Wpisy „adres MAC → port”.
  - ✦ Zmieniając często adres MAC można zalać CAM nowymi wpisami → przełącznik przejdzie w tryb uczenia się.



# Ataki „man-in-the-middle”: sieci lokalne

---

Podśłuchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

- ❖ **Zatruwanie pamięci podręcznej ARP**



# Ataki „man-in-the-middle”: sieci lokalne

Podśłuchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

## ❖ Zatrutowanie pamięci podręcznej ARP

192.168.1.11  
mac: aa:bb:cc:00:00:01

Tablica ARP:

192.168.1.12 → aa:bb:cc:00:00:02



192.168.1.12

mac: aa:bb:cc:00:00:02





# Ataki „man-in-the-middle”: sieci lokalne

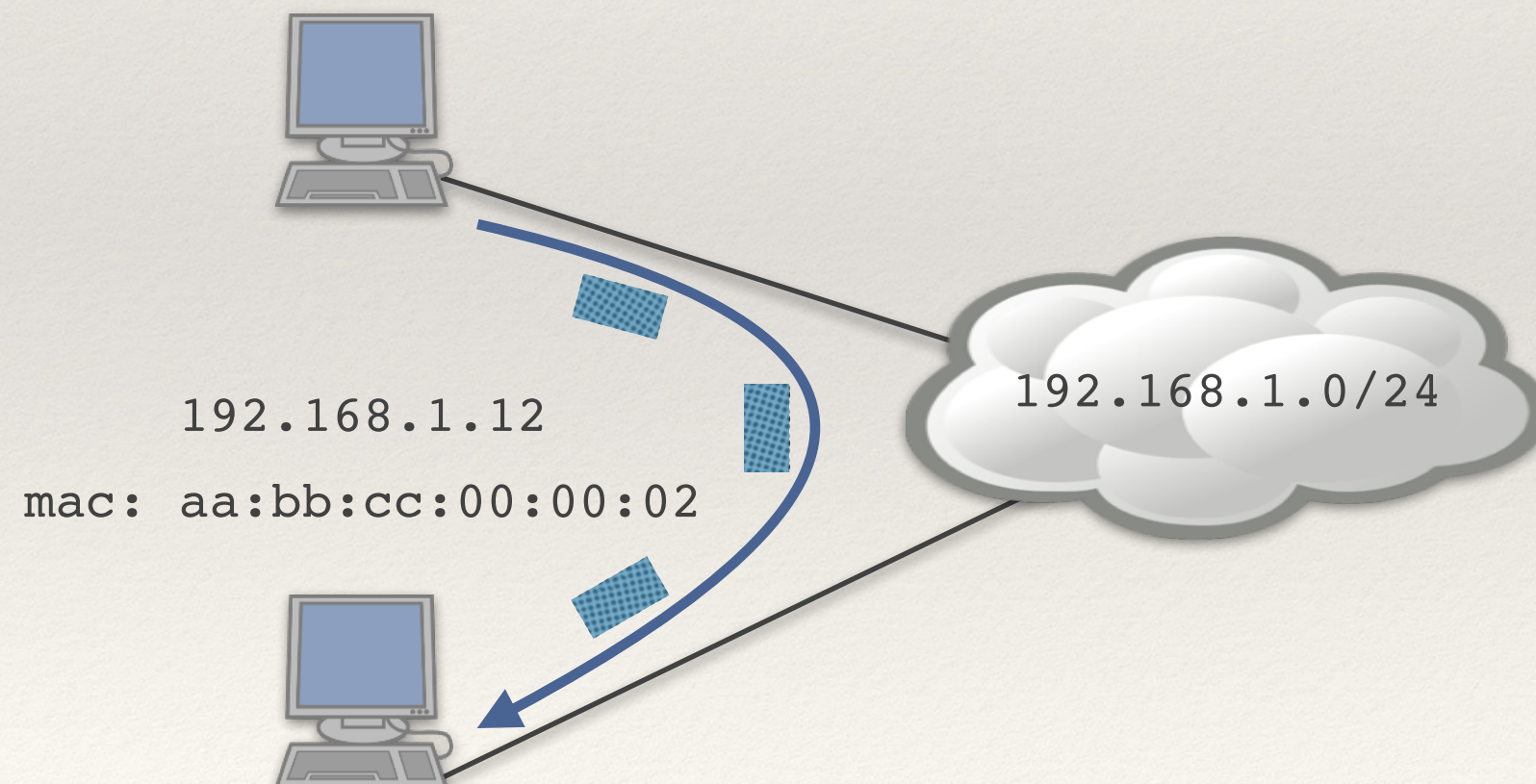
Podśłuchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

## ❖ Zatrutowanie pamięci podręcznej ARP

192.168.1.11  
mac: aa:bb:cc:00:00:01

Tablica ARP:

192.168.1.12 → aa:bb:cc:00:00:02





# Ataki „man-in-the-middle”: sieci lokalne

Podśłuchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

## ❖ Zatrutowanie pamięci podręcznej ARP

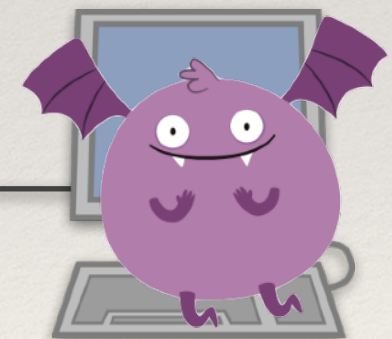
192.168.1.11  
mac: aa:bb:cc:00:00:01



192.168.1.12  
mac: aa:bb:cc:00:00:02



192.168.1.100  
mac: aa:bb:cc:00:00:03

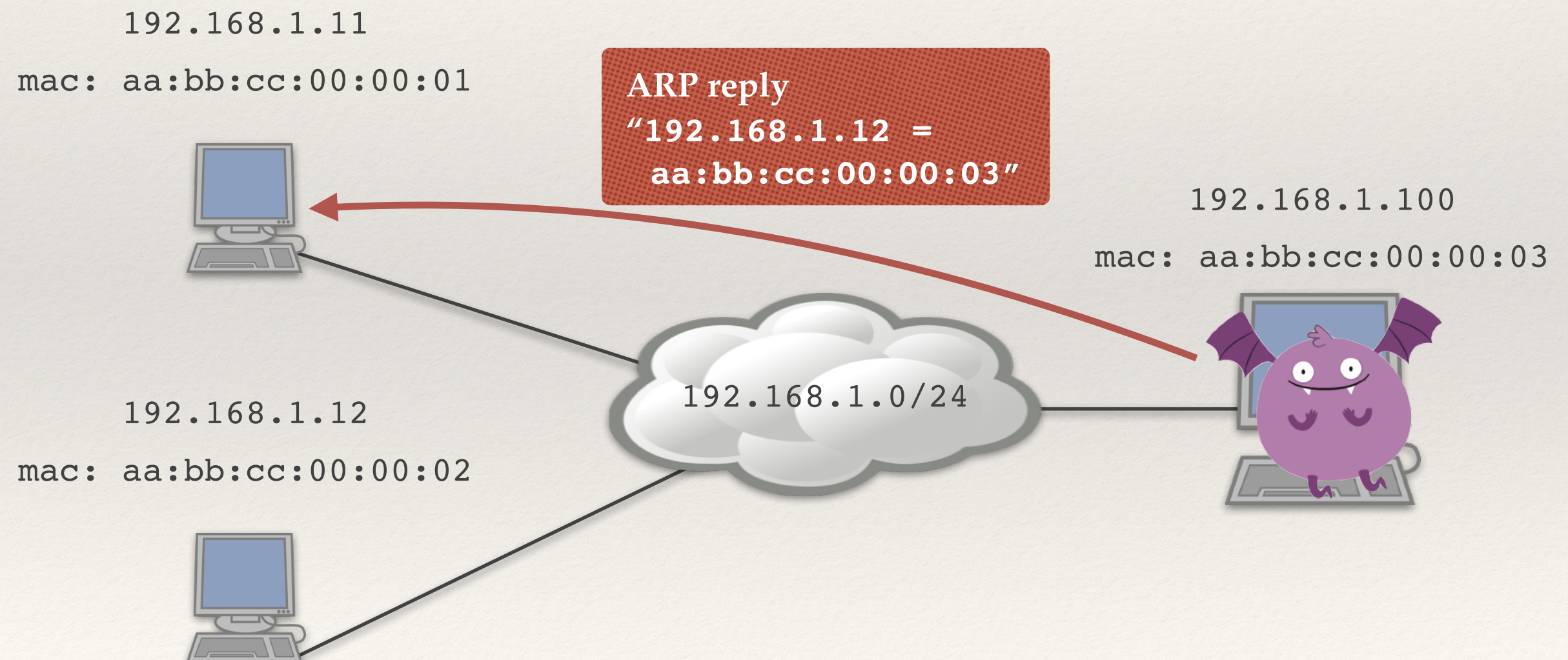




# Ataki „man-in-the-middle”: sieci lokalne

Podśłuchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

## ❖ Zatrutowanie pamięci podręcznej ARP





# Ataki „man-in-the-middle”: sieci lokalne

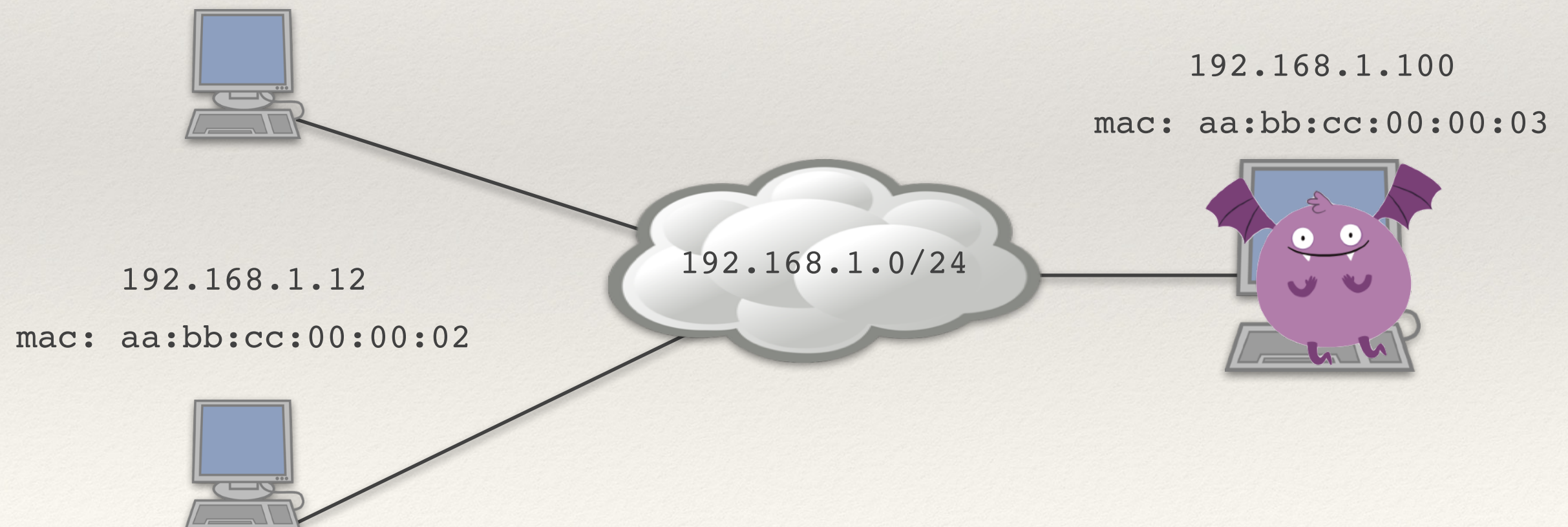
Podśłuchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

## ❖ Zatrutowanie pamięci podręcznej ARP

192.168.1.11  
mac: aa:bb:cc:00:00:01

Tablica ARP:

192.168.1.12 → aa:bb:cc:00:00:03





# Ataki „man-in-the-middle”: sieci lokalne

Podsluchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

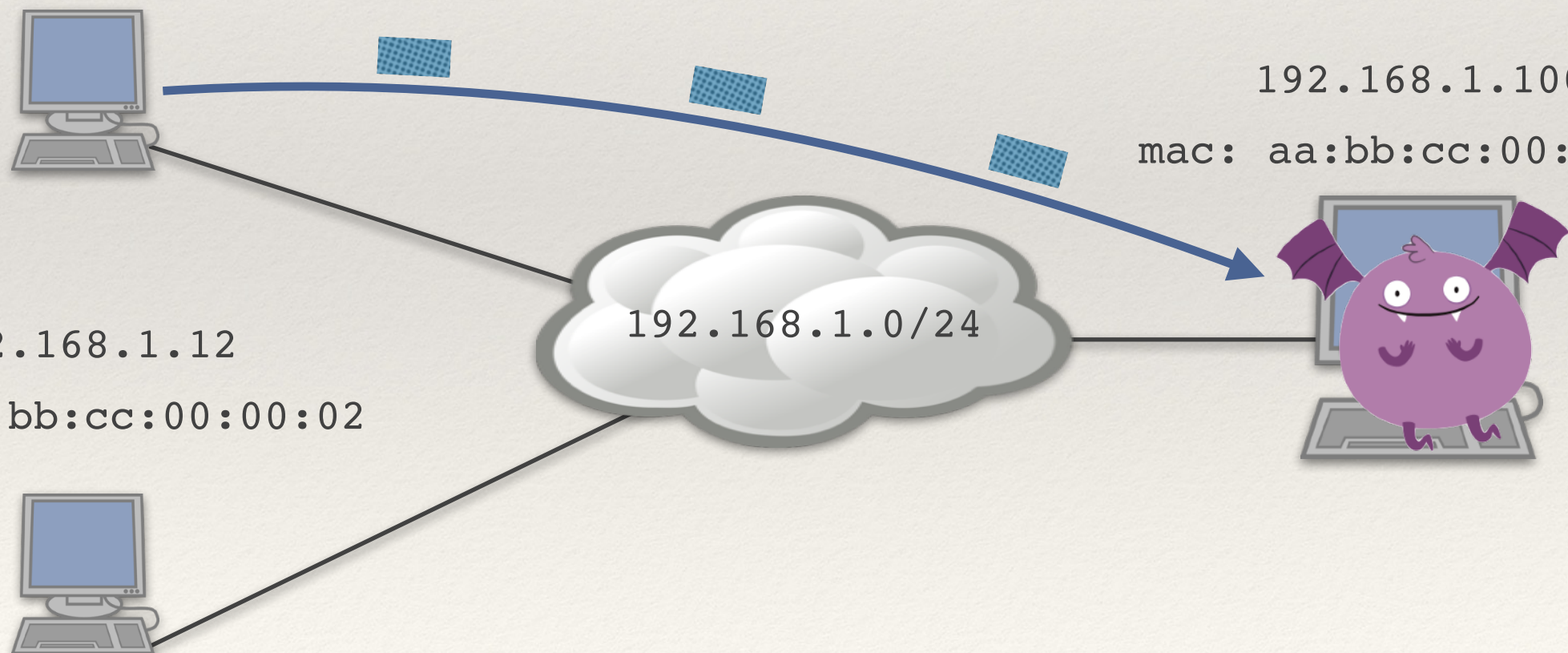
## ❖ Zatrutowanie pamięci podręcznej ARP

192.168.1.11  
mac: aa:bb:cc:00:00:01

Tablica ARP:  
192.168.1.12 → aa:bb:cc:00:00:03

192.168.1.12  
mac: aa:bb:cc:00:00:02

192.168.1.100  
mac: aa:bb:cc:00:00:03





# Ataki „man-in-the-middle”: sieci lokalne

---

Podsluchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.



# Ataki „man-in-the-middle”: sieci lokalne

---

Podsluchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

- ❖ Własny serwer DHCP.
  - ❖ Atakujący rozgłasza swój adres IP jako bramę domyślną.
  - ❖ Dobre przełączniki mogą filtrować takie odpowiedzi.



# Ataki „man-in-the-middle”: sieci lokalne

---

Podsluchujemy przez wstawienie swojego urządzenia „w środku” ścieżki komunikacji.

- ❖ Własny serwer DHCP.
  - ❖ Atakujący rozgłasza swój adres IP jako bramę domyślną.
  - ❖ Dobre przełączniki mogą filtrować takie odpowiedzi.
- ❖ Własny punkt dostępowy nazywający się stud-wifi.
  - ❖ Kto sprawdza chociaż adres MAC punktu dostępowego?
  - ❖ WPA Personal nie uwierzytelnia punktu dostępowego.



# Ataki „man-in-the-middle”: routing

---

- ❖ **RIP spoofing**

- ❖ RIPv1 nie ma uwierzytelniania
- ❖ Wystarczy rozgłaszać trasę do różnych sieci o małym koszcie.



# Ataki „man-in-the-middle”: routing

---

## ❖ RIP spoofing

- ❖ RIPv1 nie ma uwierzytelniania
- ❖ Wystarczy rozgłaszać trasę do różnych sieci o małym koszcie.

## ❖ Nadużycia BGP

- ❖ Są uwierzytelnienia...
- ❖ ... ale nie sprawdzamy, czy dany zakres adresów faktycznie należy do danego AS.
- ❖ Przykład z 2008 roku:
  - ❖ Router BGP z Pakistanu ogłosił, że ma bardzo krótką ścieżkę do zakresu adresów odpowiadających Youtube.
  - ❖ Przez pewien czas cały ruch kierowany do Youtube szedł do Pakistańskich serwerów.
  - ❖ <https://www.youtube.com/watch?v=IzLPKuAOe50>



# IP spoofing

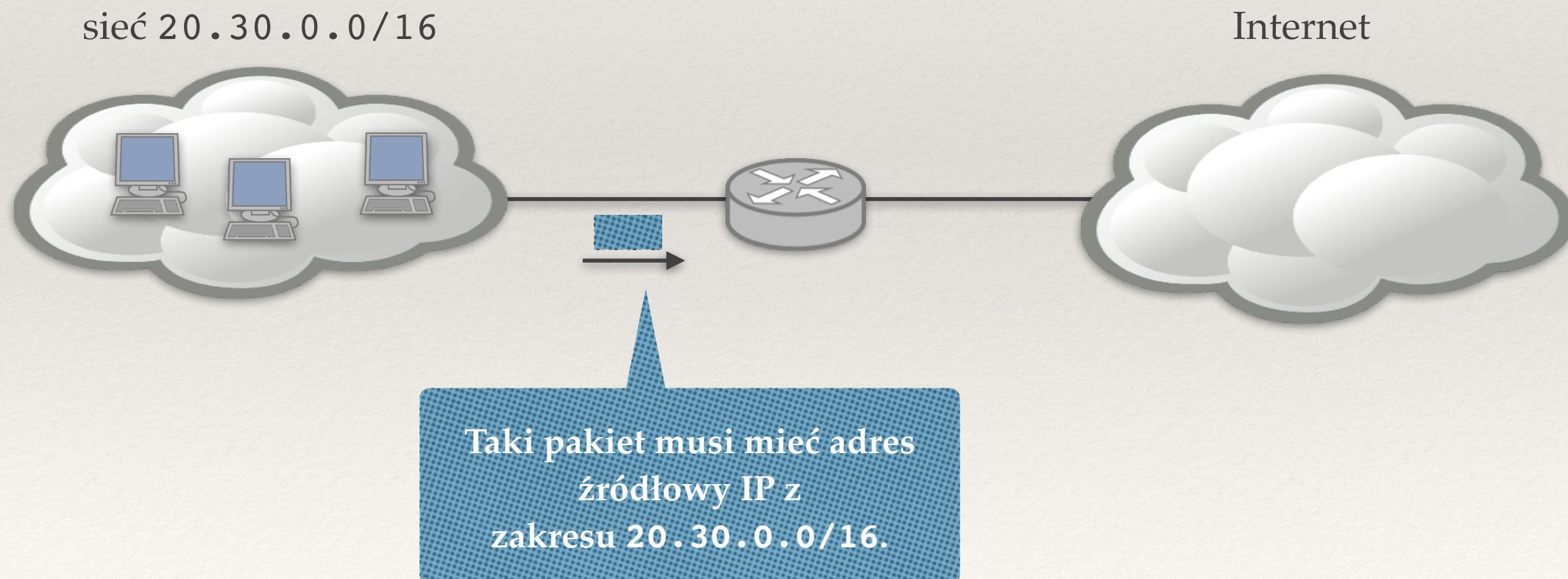
---

- ❖ **Fałszowanie adresu źródłowego IP.**
  - ♦ Unikanie odpowiedzialności za atak.
  - ♦ Uzyskiwanie dostępu do niektórych usług.



# IP spoofing

- ❖ **Fałszowanie adresu źródłowego IP.**
  - ✦ Unikanie odpowiedzialności za atak.
  - ✦ Uzyskiwanie dostępu do niektórych usług.
- ❖ **Rozwiązanie: tzw. *ingress filtering***
  - ✦ Skuteczne jeśli router jest blisko nadawcy.





# Zatruwanie pamięci podręcznej DNS (1)

---

- ❖ Atakujący kontroluje serwer o adresie IP = 11.22.33.44.
- ❖ Resolver DNS (*R*) ma pamięć cache, w której pamięta niedawno odpytywane domeny.
- ❖ Atakujący chce, żeby dostał się tam fałszywy wpis amazon.com → 11.22.33.44



# Zatruwanie pamięci podręcznej DNS (1)

---

- ❖ Atakujący kontroluje serwer o adresie IP = 11.22.33.44.
- ❖ Resolver DNS (*R*) ma pamięć cache, w której pamięta niedawno odpytywane domeny.
- ❖ Atakujący chce, żeby dostał się tam fałszywy wpis amazon.com → 11.22.33.44
- ❖ **Stara wersja ataku:**
  - ✦ Atakujący wysyła do *R* zapytanie o domenę xyz.com, którą posiada.
  - ✦ *R* pyta o xyz.com serwer nazw kontrolowany przez atakującego.
  - ✦ W odpowiedzi na zapytanie o xyz.com atakujący odpowiada dodatkowo rekordem amazon.com → 11.22.33.44.
  - ✦ Współcześnie taki dodatkowy rekord zostanie zignorowany.



# Zatruwanie pamięci podręcznej DNS (2)

---

## Nowa wersja ataku:

- ❖ Atakujący wysyła do  $R$  zapytanie o `amazon.com`
- ❖  $R$  wysyła zapytanie (przez UDP) o `amazon.com` do odpowiedzialnego serwera DNS (o adresie IP =  $X$ )
- ❖ Atakujący wysyła odpowiedzi DNS (datagramy UDP) podszywając się pod  $X$ .
- ❖  $R$  sprawdza, czy w odpowiedzi jest taki sam 16-bitowy ID jak w zapytaniu...
- ❖ ... wystarczy, że atakujący wyśle  $2^{16}$  odpowiedzi ze wszystkimi możliwymi ID.

## Zapobieganie:

- ❖ DNSSEC (kryptograficzne uwierzytelnianie pakietów DNS) → stosowane m.in. przez serwery główne DNS.
- ❖ Ataki na DNS są mniej współcześnie mniej skuteczne ze względu na uwierzytelnianie punktów końcowych połączenia (w TLS).



---

# Szyfrowanie: sposób na posłuchiwanie

---



# TLS (1)

---

## Warstwa szyfrująca (TLS)

- ❖ Pomiedzy warstwą transportową a warstwą aplikacji.
- ❖ Większość tradycyjnych usług ma szyfrowane warianty:
  - ♦ HTTPS = HTTP + TLS
  - ♦ POP3 + TLS
  - ♦ SMTP + TLS



## Szyfrowanie oparte o 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 uwierzytelnia się podając swoje hasło.



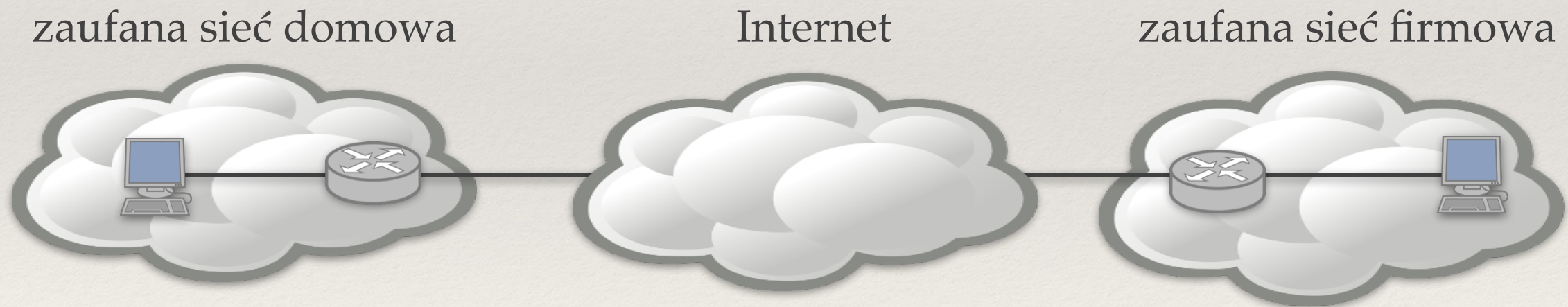
## Uwierzytelnianie serwera = za pomocą certyfikatu

- ❖ Klucz publiczny serwera jest podpisany przez CA i możemy zweryfikować autentyczność tego podpisu.
- ❖ Analogicznie działa uwierzytelnianie serwera w sieci WLAN z szyfrowaniem WPA Enterprise (np. eduroam).



## VPN = Wirtualna sieć prywatna

- ❖ 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.





# Tunelowanie

---

## Model warstwowy

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

## Tunelowanie

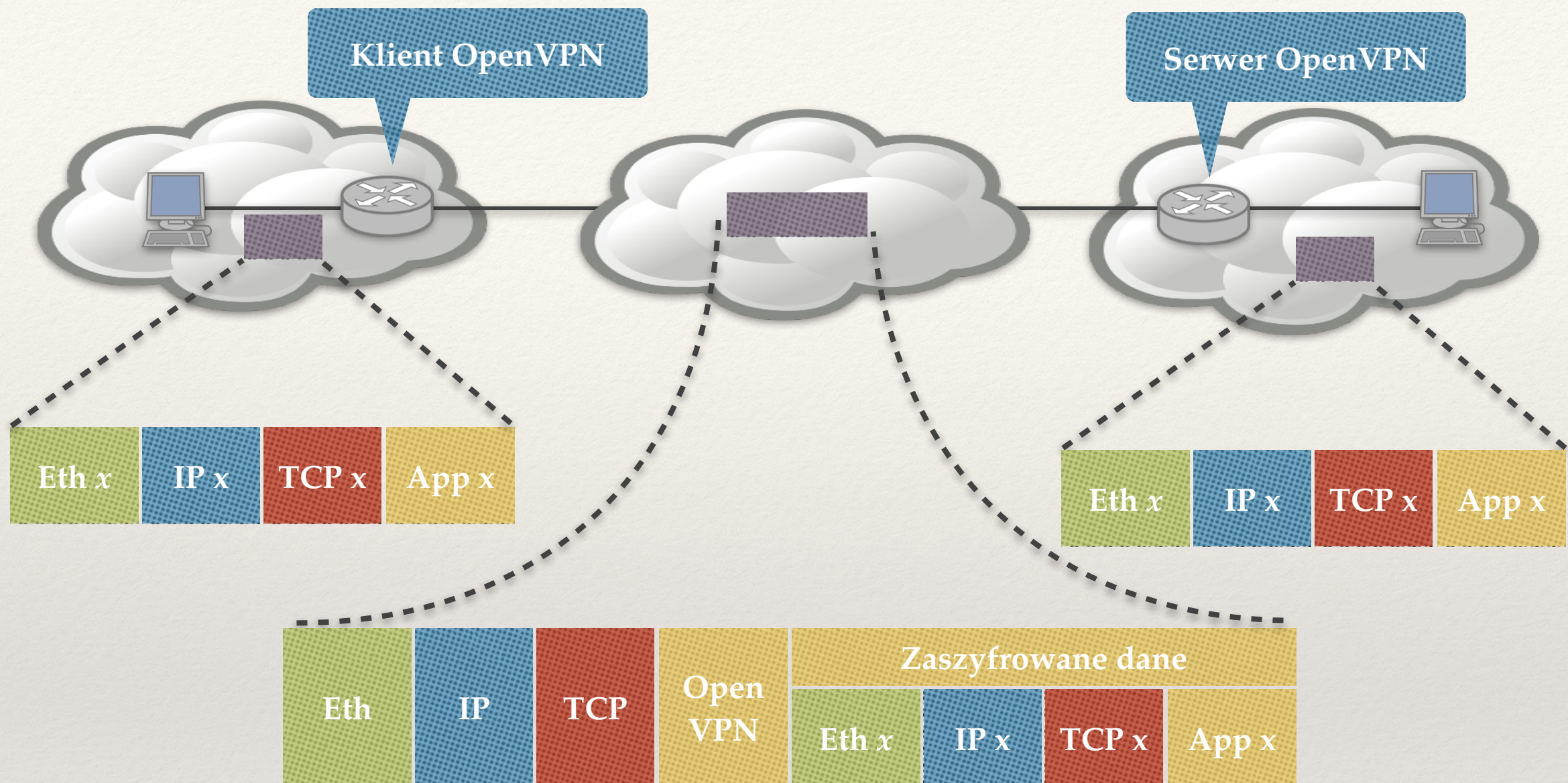
- ❖ Przesyłanie pewnych usług sieciowych za pomocą innych usług sieciowych w sposób łamiący standardowy model warstwowy.
- ❖ Cel: zestawianie wirtualnego (często szyfrowanego) połączenia.
- ❖ Poznaliśmy już tunelowanie pakietów IPv6 w pakietach IPv4.

## VPN: dwa popularne rozwiązania

- ❖ **IPSec**: tunelowanie (szyfrowanych) pakietów IP w danych pakietów IP.
- ❖ **OpenVPN**: tunelowanie (szyfrowanych) pakietów IP (lub ramek Ethernetowych) w danych protokołu UDP lub TCP.



# OpenVPN



- ❖ Logiczne działanie tak jakby klient i serwer VPN stanowiły jedno urządzenie (most).
- ❖ Wersja w której nie tunelujemy nagłówek warstwy drugiej  
→ tak jakby klient i serwer VPN stanowiły jedno urządzenie (router)



# SSH (Secure SHell)

---

**Standardowe narzędzie pracy zdalnej (w terminalu tekstowym).**

- ❖ Następca nieszyfrowanego telnetu.
- ❖ Mechanizmy szyfrowania jak przy TLS.
- ❖ Co z uwierzytelnianiem serwera?



# SSH: uwierzytelnianie serwera

---

Znowu nie wiemy, czy łączymy się z dobrym serwerem!

- ❖ Przy pierwszym połączeniu serwer przesyła nam klucz publiczny, program klienta oblicza funkcje skrótu klucza.

```
The authenticity of host 'ssh-server.example.com  
(12.18.49.21)' can't be established. RSA key  
fingerprint is
```

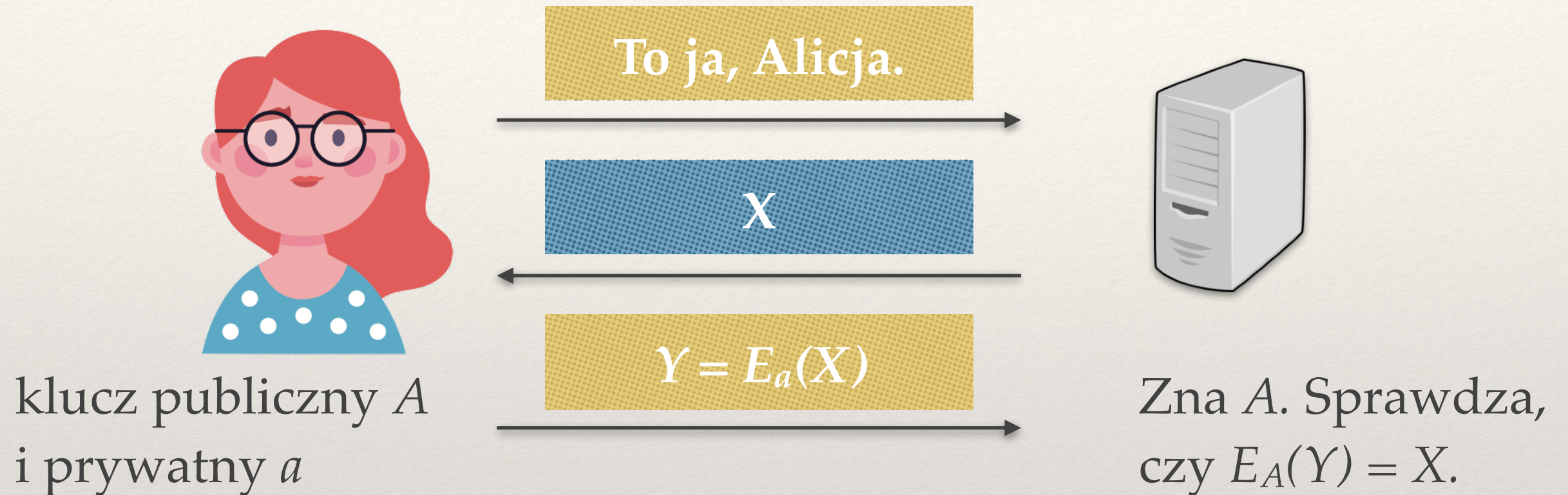
```
98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d. Are  
you sure you want to continue connecting?
```

- ❖ Kto dzwoni do administratora serwera i weryfikuje, czy funkcja skrótu *fingerprint* ma poprawną wartość?
- ❖ Po zaakceptowaniu klucz publiczny serwera zostaje zapisany lokalnie.



# SSH: uwierzytelnianie klienta

- ❖ Klient może po prostu podać hasło do konta. Albo:

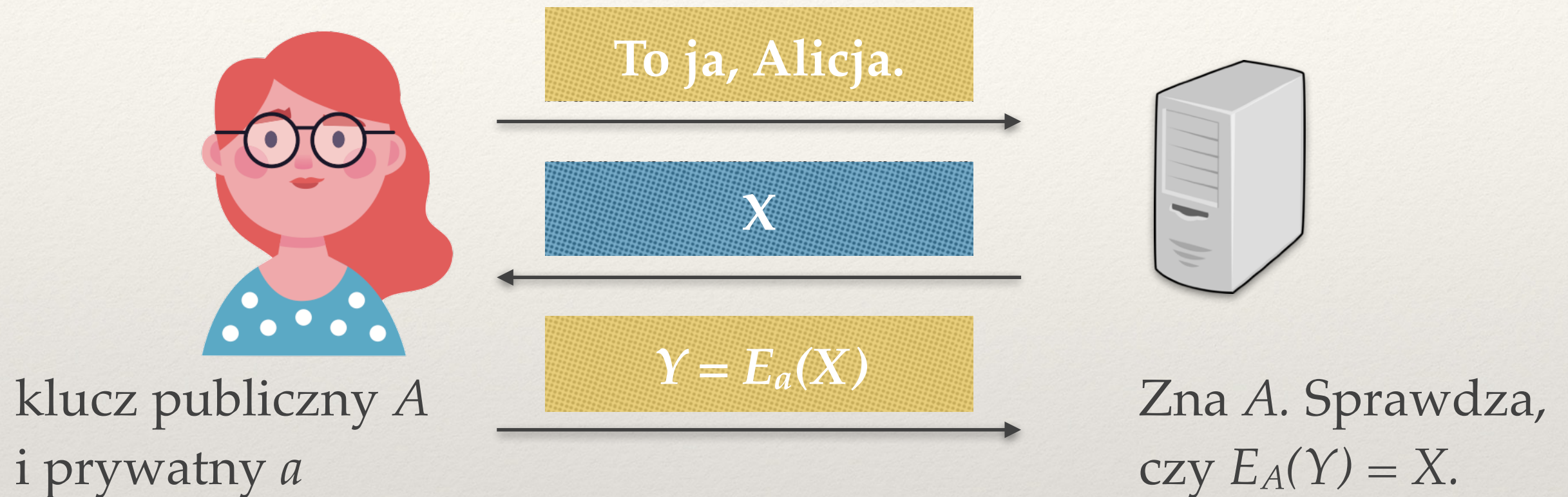


- ❖ Serwer zna klucz publiczny Alicji, bo Alicja zapisała go uprzednio w pliku `authorized_keys` na serwerze.



# SSH: uwierzytelnianie klienta

- ❖ Klient może po prostu podać hasło do konta. Albo:



- ❖ Serwer zna klucz publiczny Alicji, bo Alicja zapisała go uprzednio w pliku `authorized_keys` na serwerze.
- ❖ Dlaczego lepsze niż hasło? Nie działa atak powtórzeniowy!



# SSH: 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/`



# Tunelowanie TCP w SSH

---

- ❖ Jeśli mamy konto na zdalnej maszynie, możemy szyfrować połączenia z tymi usługami za pomocą ssh.
- ❖ `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`.



---

# Źródła podatności na ataki

---



# Źródła podatności na ataki

---

- ❖ **Źle zaprojektowane protokoły**
- ❖ **Błędy implementacji**
  - ✦ Najczęściej: Brak weryfikacji poprawności wprowadzanych danych
  - ✦ Atakujący zmusza aplikację do wykonania nieprzewidzianych operacji.
- ❖ **Czynnik ludzki**

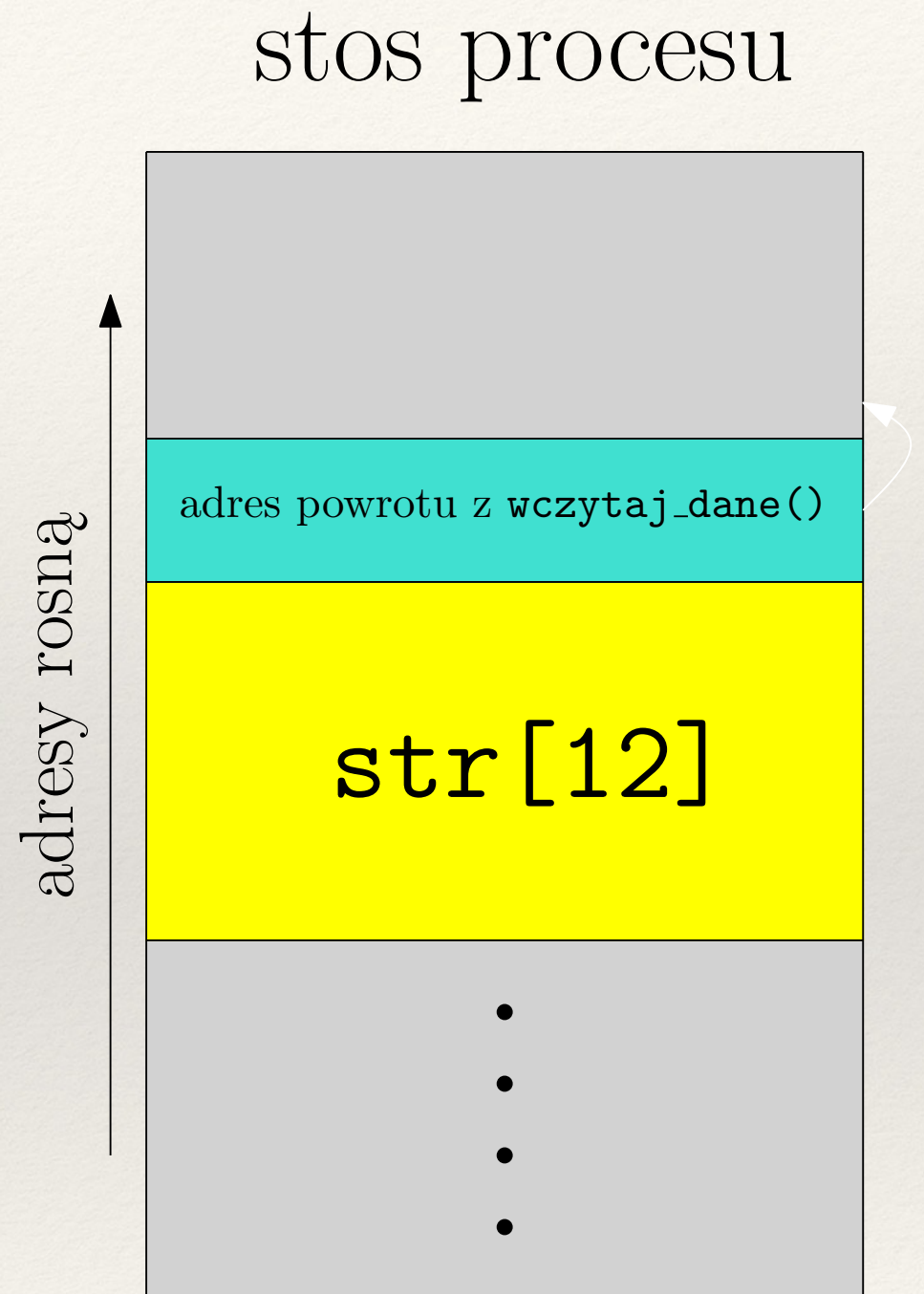


# Błędy implementacji: przepełnienie bufora

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

Atakujący wpisuje:

- ❖ jakieś 12 znaków,
- ❖ odpowiedni adres powrotu,
- ❖ kod maszynowy do uruchomienia.



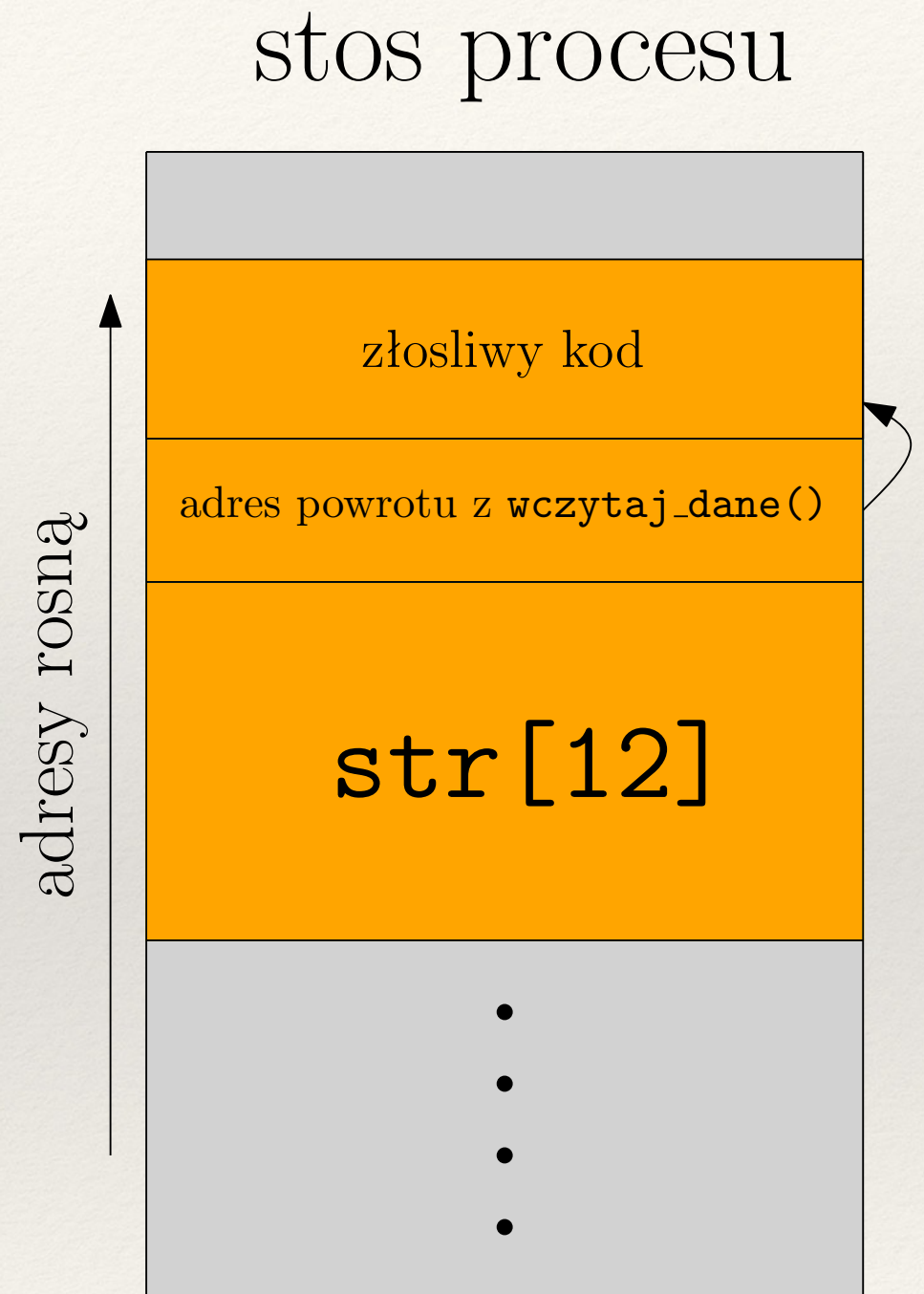


# Błędy implementacji: przepełnienie bufora

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

Atakujący wpisuje:

- ❖ jakieś 12 znaków,
- ❖ odpowiedni adres powrotu,
- ❖ kod maszynowy do uruchomienia.





# Błędy implementacji: atak typu ../

---

## Pomocniczy skrypt WWW

- ❖ skrypt jest w katalogu `/var/www/`.
- ❖ skrypt jest programem wyświetlającym zawartość pliku.
- ❖ Przykładowo `http://example.com/skrypt?plik=test` wyświetla zawartość pliku `/var/www/test`.



# Błędy implementacji: atak typu ../

---

## Pomocniczy skrypt WWW

- ❖ skrypt jest w katalogu `/var/www/`.
- ❖ skrypt jest programem wyświetlającym zawartość pliku.
- ❖ Przykładowo `http://example.com/skrypt?plik=test` wyświetla zawartość pliku `/var/www/test`.
- ❖ Co otrzymamy po wejściu na stronę `http://example.com/skrypt?plik=../../etc/passwd?`



# Błędy implementacji: atak SQL injection

---

Aplikacja WWW odwołująca się do bazy danych

- ❖ Skrypt dostaje przez formularz argument `$user` i wykonuje polecenie
  - ♦ `mysql_query("SELECT * FROM tab WHERE user = '$user'");`



# Błędy implementacji: atak SQL injection

---

## Aplikacja WWW odwołująca się do bazy danych

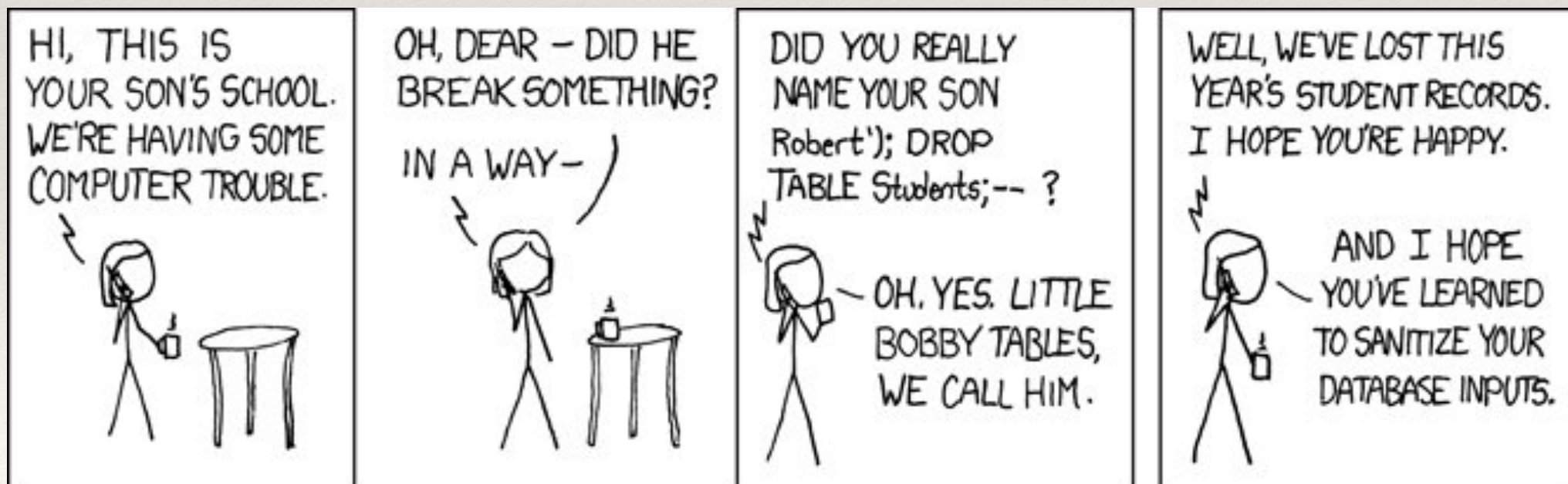
- ❖ Skrypt dostaje przez formularz argument `$user` i wykonuje polecenie
  - ♦ `mysql_query("SELECT * FROM tab WHERE user = '$user'");`
- ❖ Podajemy w tym polu: `x' OR '1'='1`



# Błędy implementacji: atak SQL injection

## Aplikacja WWW odwołująca się do bazy danych

- ❖ 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 <https://xkcd.com/327/>



# Błędy implementacji: wysyłanie części RAM procesu

---

- ❖ **Protokół: otrzymujemy od serwera 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*.



# Błędy implementacji

---

## Co można zrobić?

- ❖ Zawsze sprawdzać poprawność danych od użytkownika
  - ✦ Sprawdzanie danych po stronie klienta nie jest bezpieczne: np. nie można zakładać, że użytkownik używa przepisowego klienta usługi.
- ❖ Ogólne techniki na poziomie jądra: np. zabronienie wykonywania programów na stosie.
- ❖ Zasada minimalnych uprawnień:
  - ✦ program A działający z uprawnieniami B ma lukę;
  - ✦ atakujący może wykonać to, na co pozwalają uprawnienia B.



# Exploits

---

- ❖ Luki i gotowe sposoby ich wykorzystania (*exploits*) są publikowane na specjalistycznych stronach (CIRCL, VulDB, 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*).



# Skanowanie portów

---

## Wyszukiwanie otwartych portów

- ❖ nmap
- ❖ Przykładowe rodzaje:
  - ♦ connect scan
  - ♦ SYN scan
  - ♦ ustawianie różnych dziwnych flag (np. RST+ACK)
- ❖ Najczęściej wykorzystywane narzędzie w filmach!  
<https://nmap.org/movies/>

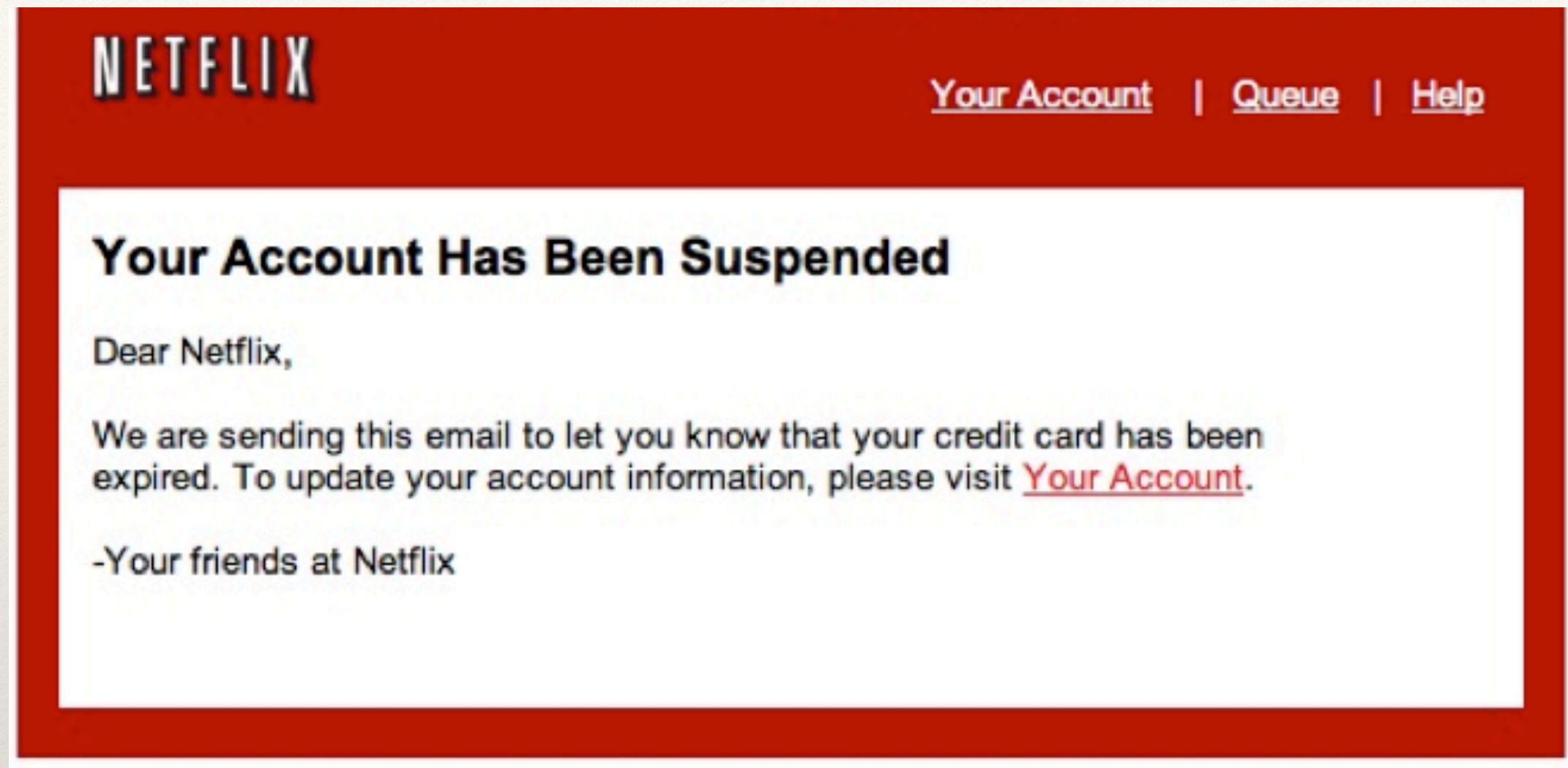


*Po co włamywać się na serwer, skoro można  
zadzwonić i spytać o hasło?*

*— Kevin Mitnick*



# Phishing



Obrazek ze strony <http://resources.infosecinstitute.com/category/enterprise/phishing/>

Odnośnik „Your Account” prowadzi na stronę atakującego <http://www.netflix.domena.com/>.

- ❖ Wygląda bardzo podobnie do <http://www.netflix.com/>.
- ❖ Atakujący może mieć dla niej również poprawny certyfikat HTTPS.



# Czynnik ludzki

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>



---

# Blokowanie dostępu

---



# Denial of Service (DoS) = Blokowanie dostępu do usług

---

## Po co?

- ❖ Głównie wymuszenia okupu od dochodowych stron, instytucji finansowych itp.
- ❖ Czasem zemsta: spamerzy atakujący serwisy antyspamowe.



# Proste ataki DoS

---

- ❖ **Zakłócanie fizycznego kanału**
  - ♦ Łatwe zakłócanie sieci bezprzewodowej



# Proste ataki DoS

---

- ❖ **Zakłócanie fizycznego kanału**
  - ♦ Łatwe zakłócanie sieci bezprzewodowej
- ❖ **Wyczerpywanie mocy obliczeniowej**



# Proste ataki DoS

---

- ❖ **Zakłócanie fizycznego kanału**
  - ♦ Łatwe zakłócanie sieci bezprzewodowej
- ❖ **Wyczerpywanie mocy obliczeniowej**
- ❖ **Zalewanie łącza pakietami**
  - ♦ Np. komunikatami *ICMP echo*.
  - ♦ Problem dla atakującego: musi być w stanie wysyłać szybciej komunikaty niż ofiara odbierać.



# Odbity (reflected) DoS

---

- ❖ Zapytania z (fałszywym) adresem źródłowym równym adresowi ofiary.

ofiara

IP = 12.34.56.78



serwer  
DNS

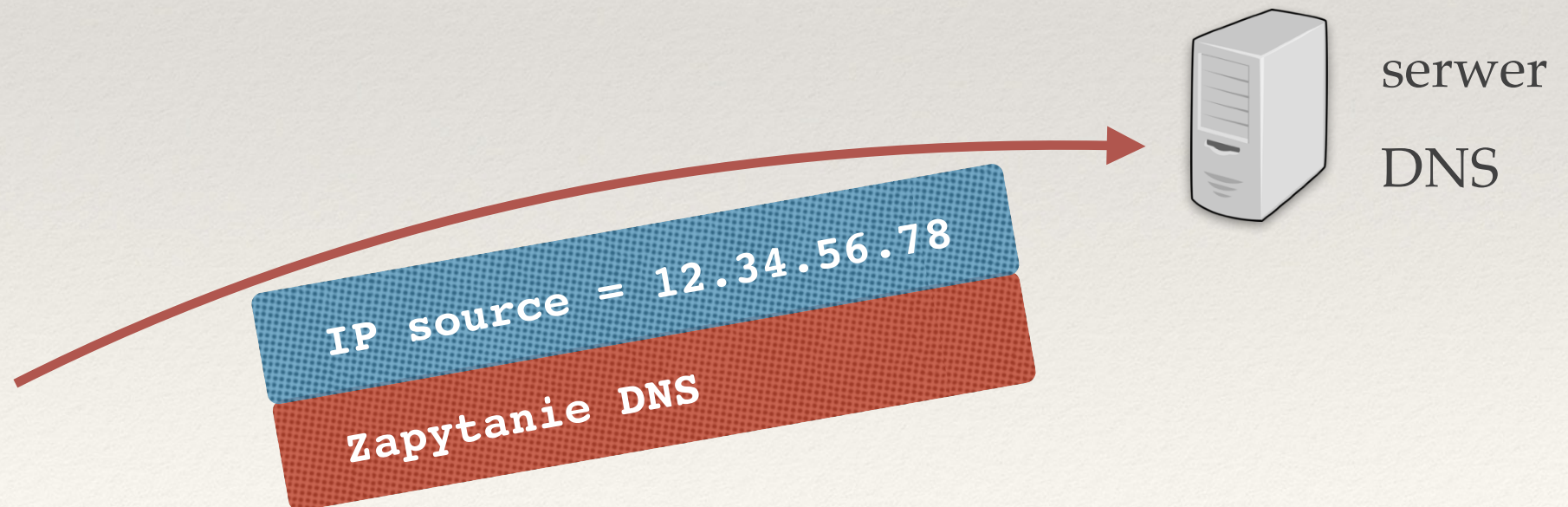


# Odbity (reflected) DoS

- ❖ Zapytania z (fałszywym) adresem źródłowym równym adresowi ofiary.

ofiara

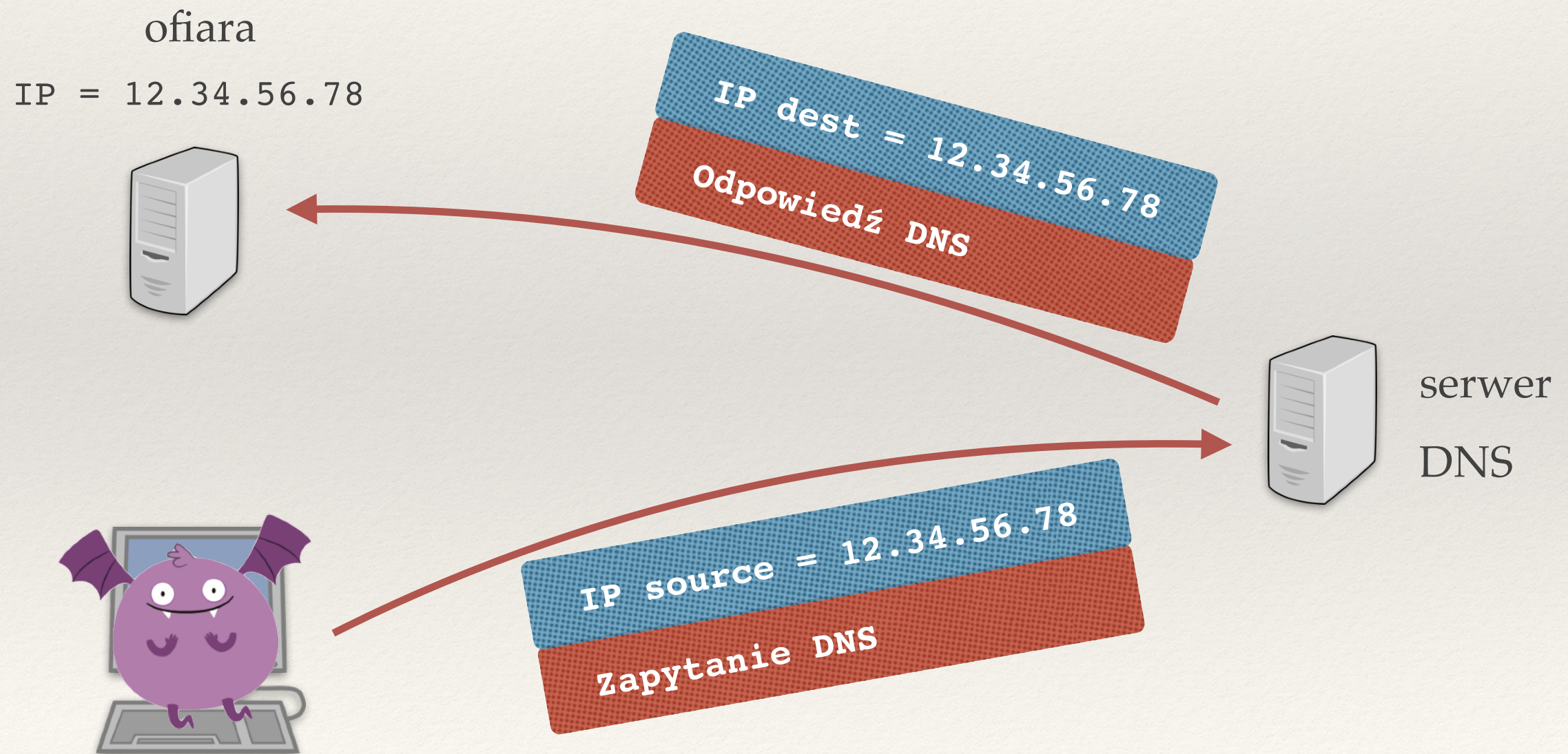
IP = 12.34.56.78





# Odbity (reflected) DoS

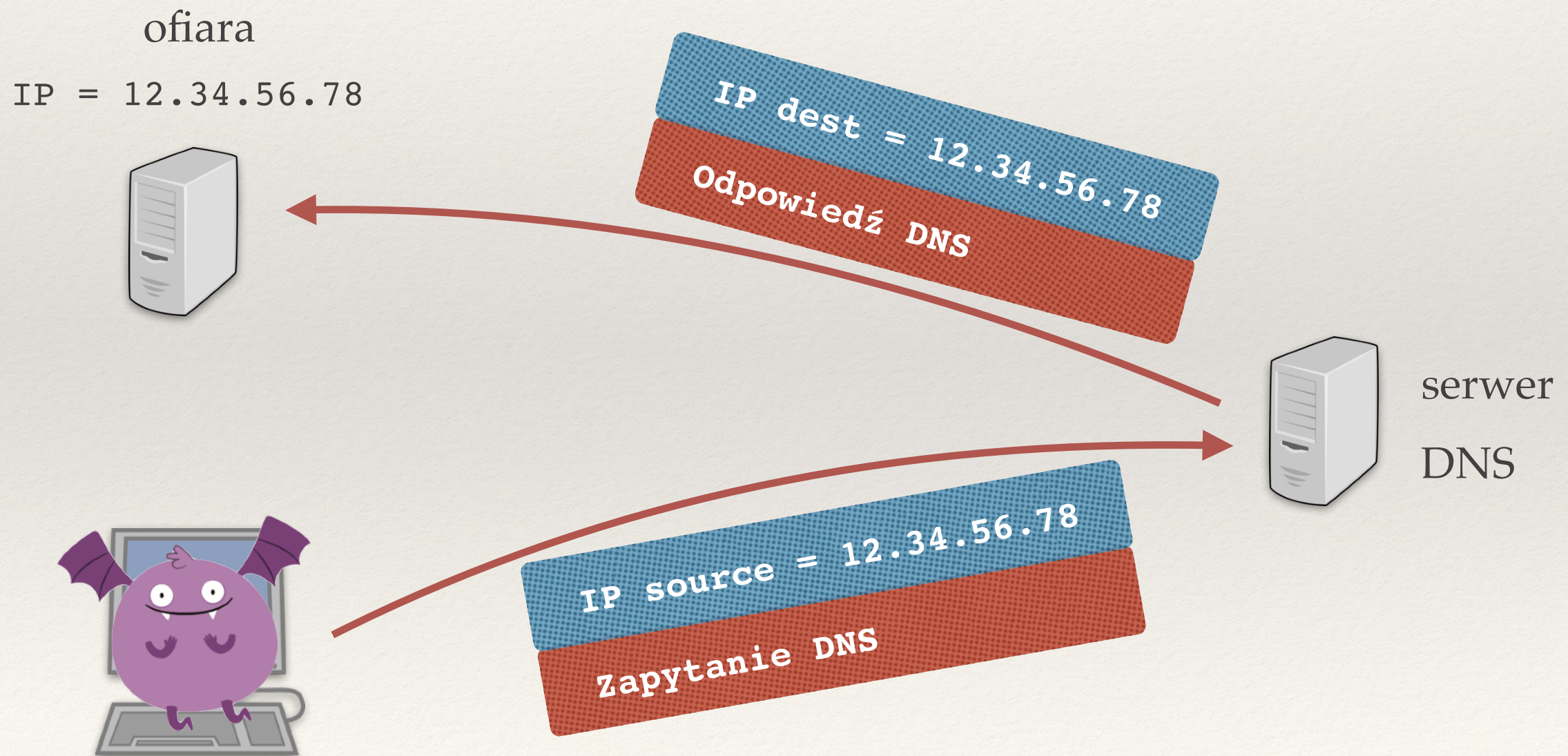
- ❖ Zapytania z (fałszywym) adresem źródłowym równym adresowi ofiary.





# Odbity (reflected) DoS

- ❖ Zapytania z (fałszywym) adresem źródłowym równym adresowi ofiary.
- ❖ Ofierze trudniej wyśledzić atakującego.
- ❖ Odpowiedź może być większa niż zapytanie (DNS do 70:1, NTP do 20:1)





# Wariant odbitego DoS: smurf attack

---

- ❖ Wykorzystuje ICMP echo (ping).
- ❖ Rozmiar odpowiedzi podobny do rozmiaru zapytania...
- ❖ ... ale możemy wysłać *jeden* pakiet do przełącznika na adres broadcast!
- ❖ Przełącznik go zwielokrotni i wyśle do wszystkich uczestników sieci lokalnej.



# Wariant odbitego DoS: smurf attack

---

- ❖ Wykorzystuje ICMP echo (ping).
- ❖ Rozmiar odpowiedzi podobny do rozmiaru zapytania...
- ❖ ... ale możemy wysłać *jeden* pakiet do przełącznika na adres broadcast!
- ❖ Przełącznik go zwielokrotni i wyśle do wszystkich uczestników sieci lokalnej.
- ❖ Wszyscy z LAN odpowiedzą ofierze



# Wyczerpywanie zasobów

---

## Limit jednoczesnych połączeń

- ❖ Nawiażujemy połączenie TCP i nic nie wysyłamy.
- ❖ Opcjonalnie: wiele protokołów na początku wysyła długość komunikatu: wysyłamy długość a potem już nic.

## Wysyłamy dużo komunikatów TCP SYN

- ❖ Jądro utrzymuje tworzoną przez `listen()` kolejkę połączeń TCP oczekujących na nawiązanie.
- ❖ Kolejka ma ograniczoną wielkość.

## Wysyłamy dużo fragmentów pakietów IP

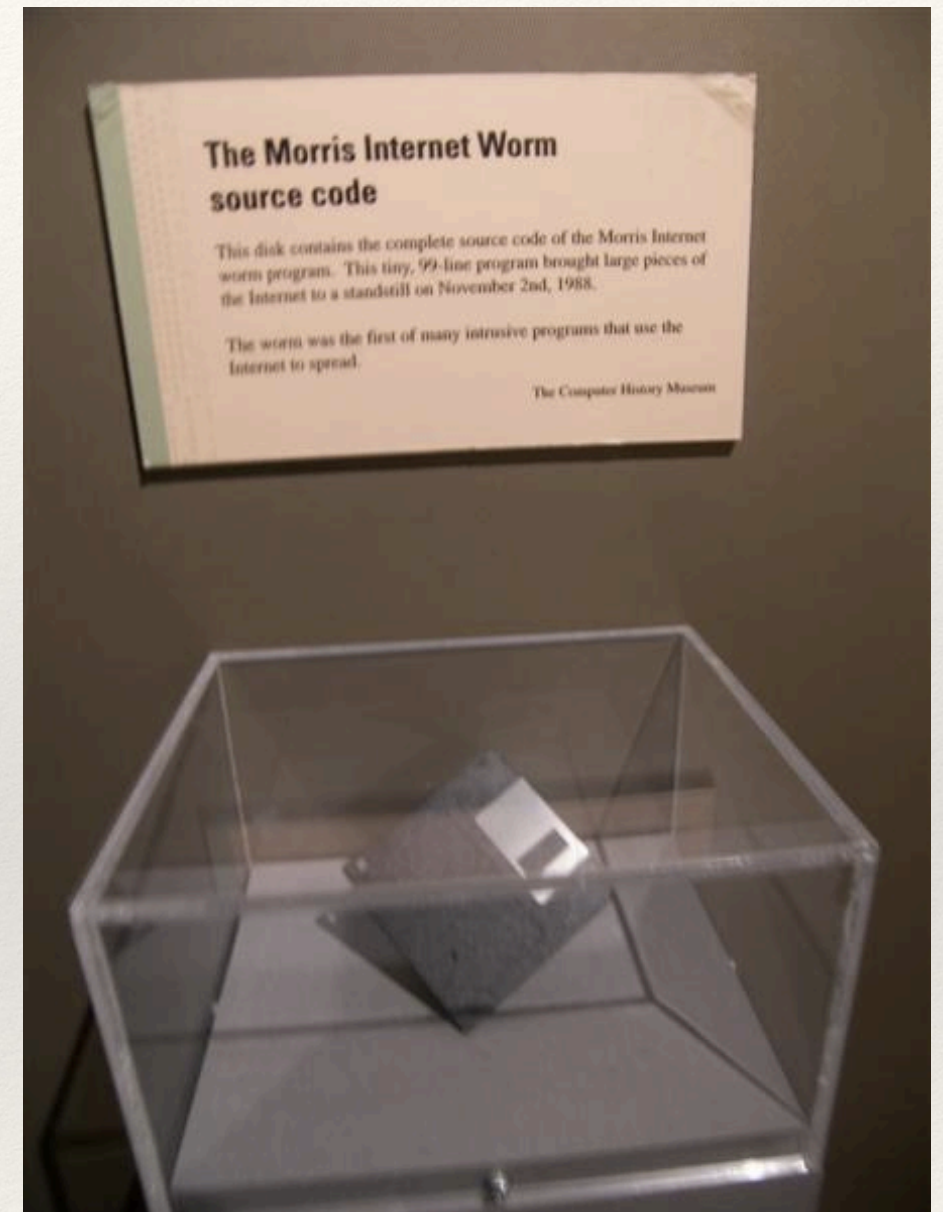
- ❖ Jądro musi utrzymywać te pakiety do późniejszego ich łączenia.



# Rozproszony DoS (DDoS)

## DDoS = Distributed DoS

- ❖ Wykonywany z wielu komputerów.
- ❖ Często są pod kontrolą atakującego, bo uprzednio zostały zainfekowane robakiem internetowym (*botnet*).
- ❖ Robaki wykorzystują znalezione luki w bezpieczeństwie niektórych usług.



Obrazek ze strony  
[https://en.wikipedia.org/wiki/Morris\\_worm](https://en.wikipedia.org/wiki/Morris_worm)



# Obrona przed DDoS

---

- ❖ Tylko jeśli pochodzi z jednego geograficznego obszaru.
- ❖ Główny problem: ustalenie źródła ataków (bo źródłowe adresy IP są zazwyczaj 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.



---

*Zapora*

---



# Po co?

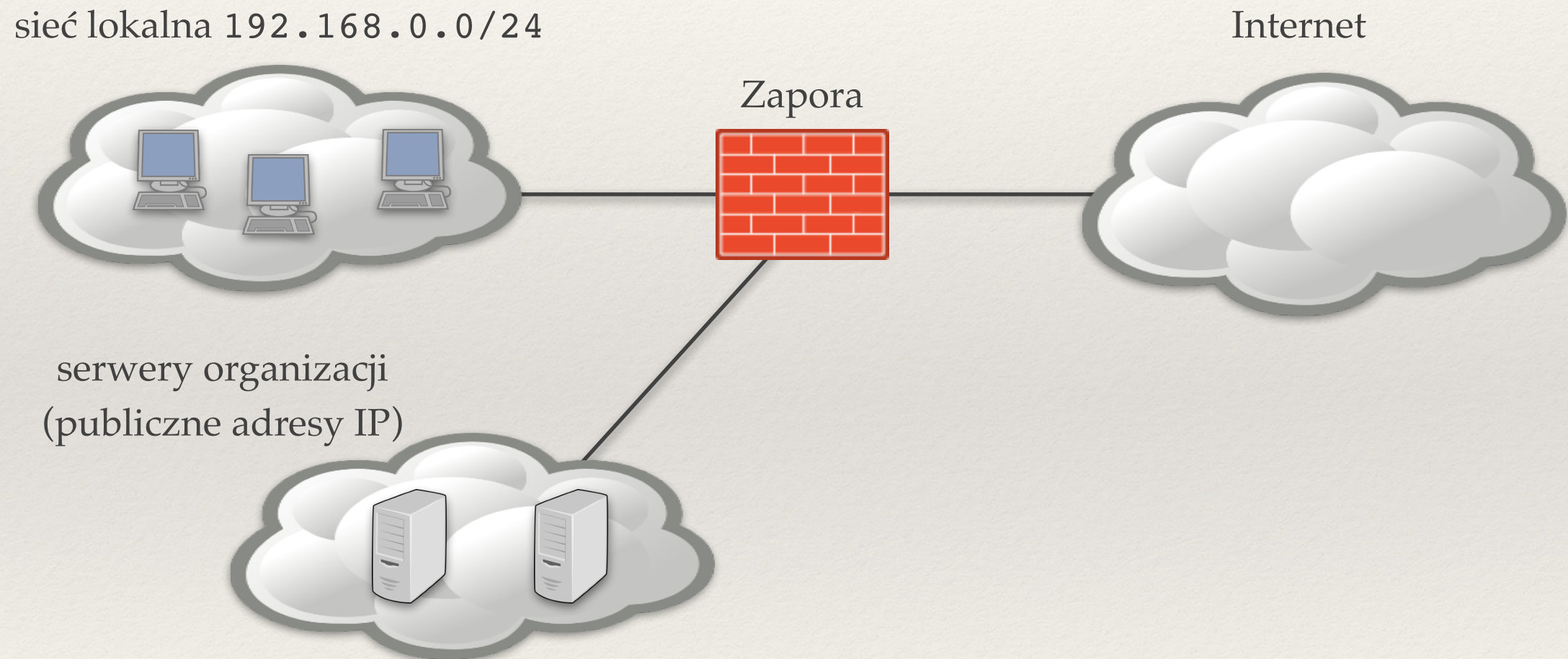
---

- ❖ **Pierwsza linia obrony.**
  - ♦ Częściowe zapobieganie fałszowaniu adresów źródłowych IP.
- ❖ **Rejestrowanie i kontrolowanie dostępu do usług.**
  - ♦ Rejestrowanie ataków i skanów portów.



# Gdzie?

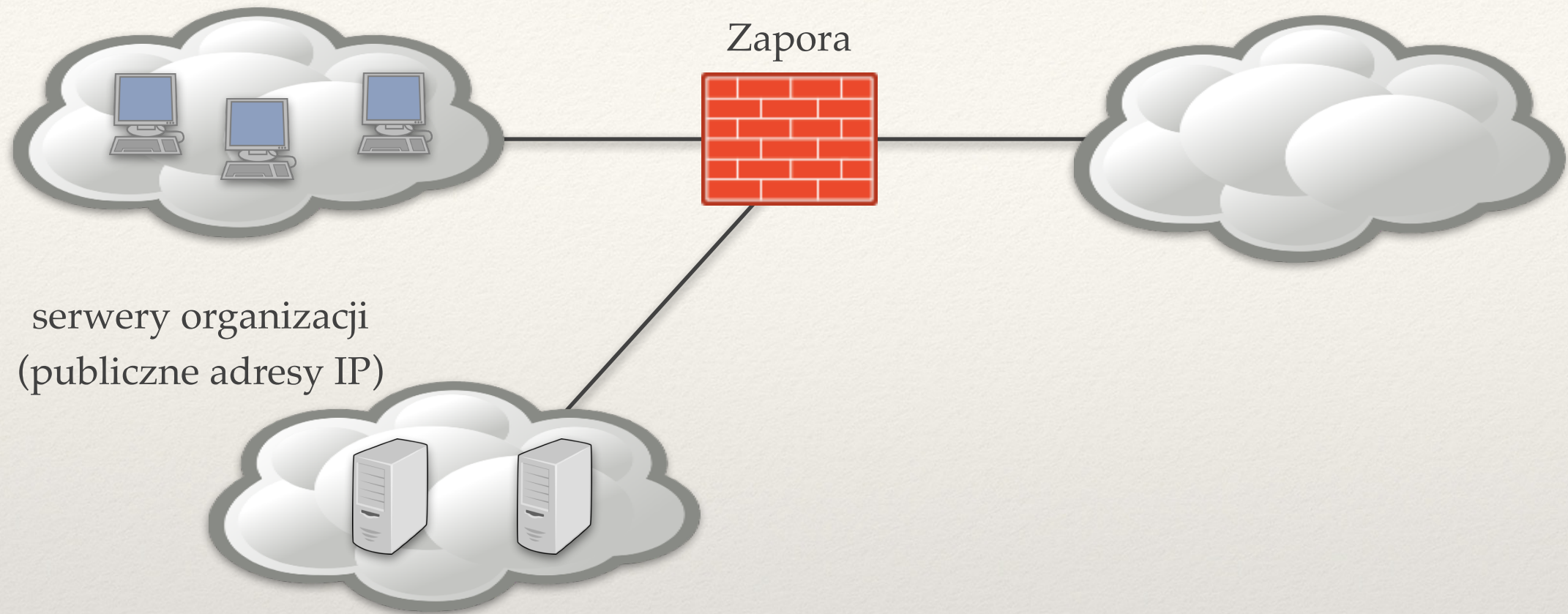
- ❖ Często zapora jest osobnym urządzeniem, pełniącym też funkcję routera:





sieć lokalna 192.168.0.0/24

Internet



**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.



# Klasyfikacja filtrów pakietów

---

## Filtry proste (warstwa sieciowa)

- ❖ Analizują tylko nagłówki IP (+ porty)
- ❖ Szybkie, bardzo nieprecyzyjne.

## Filtry stanowe (warstwa transportowa)

- ❖ Analizują nagłówki IP i TCP
- ❖ Śledzą nawiązywanie połączenia TCP, pamiętają stan połączenia.

## Filtry działające w warstwie aplikacji

- ❖ Analizują zawartość segmentów i datagramów
- ❖ Np. w przypadku FTP „rozumieją”, że trzeba otworzyć port na dane.
- ❖ Nie mylić z zaporami aplikacji, które analizują *wywołania systemowe* aplikacji.

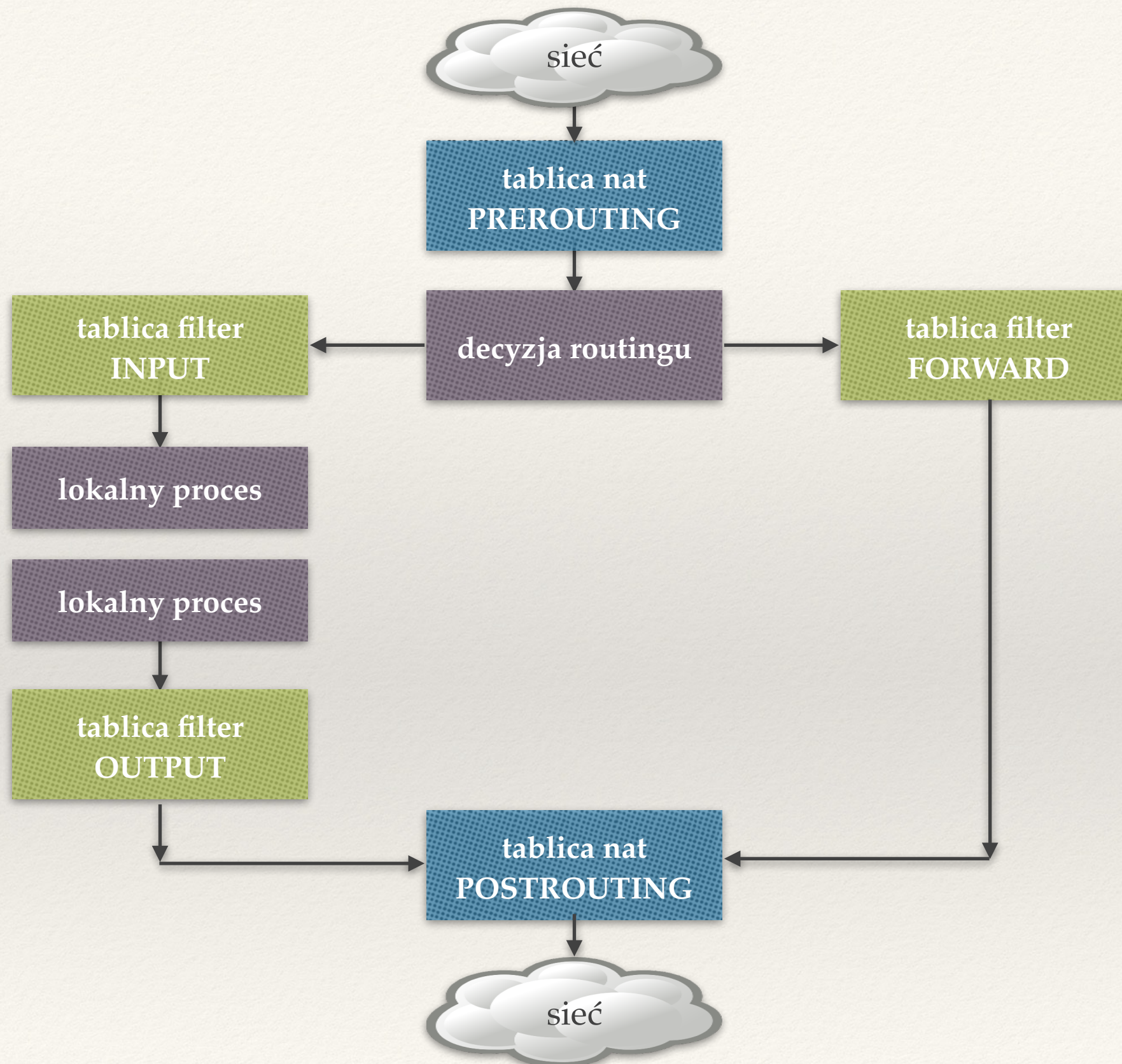


## Filtr pakietów w Linuksie

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



# Przetwarzanie pakietów (1)

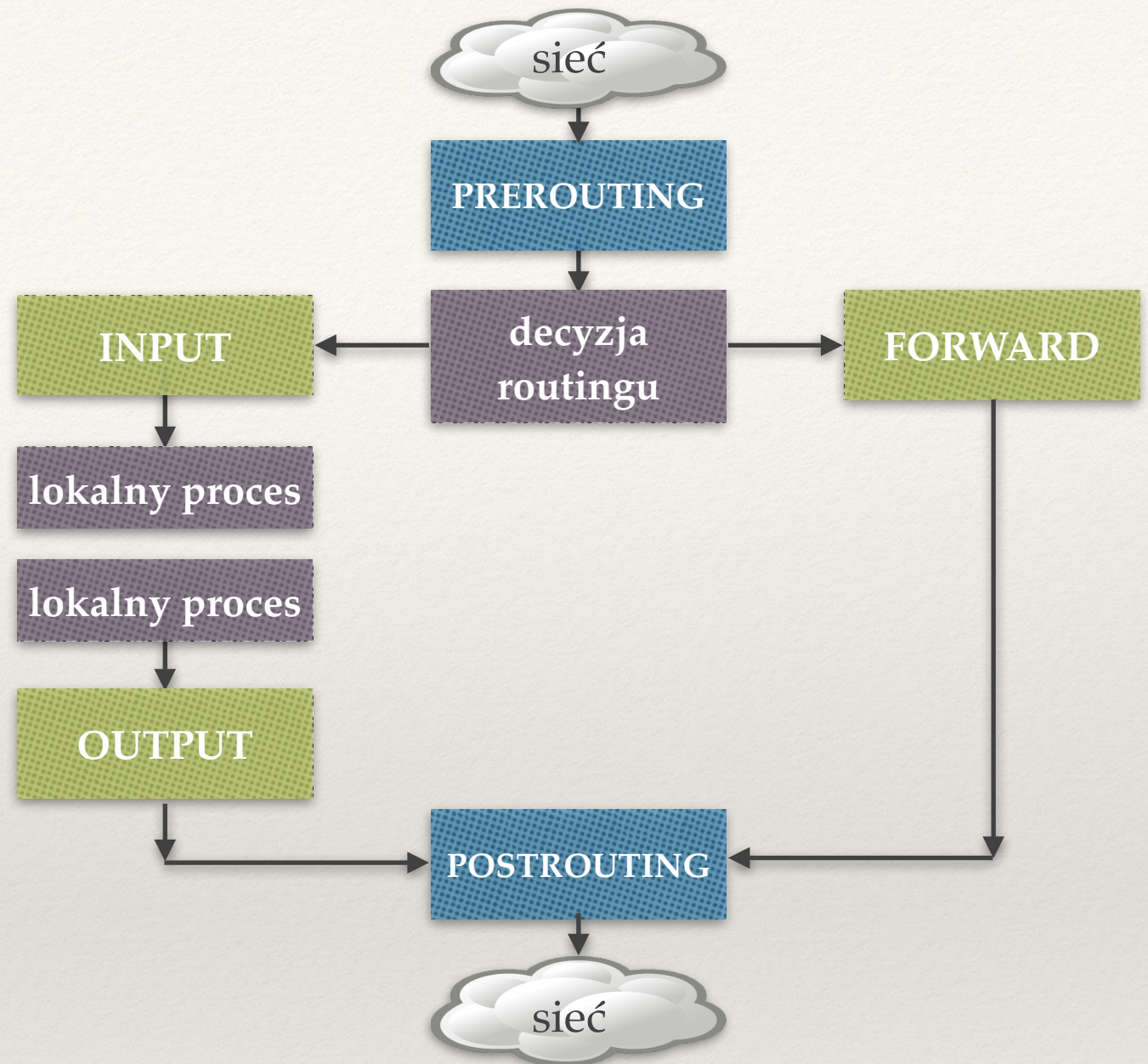




# Przetwarzanie pakietów (2)

## 5 głównych modułów

- ❖ filter INPUT
- ❖ filter OUTPUT
- ❖ filter FORWARD
- ❖ nat PREROUTING
- ❖ nat POSTROUTING



Do każdego modułu możemy dopisać łańcuch reguł

- ❖ Łańcuchy przetwarzane w kolejności priorytetów.
- ❖ Reguły w łańcuchach przetwarzane po kolei.



# Przykładowy łańcuch (pakiety wychodzące)

---

❖ Akceptuj wszystkie pakiety wychodzące

```
table inet filter {  
    chain jakaś_nazwa {  
        type filter hook output priority 0;  
        accept  
    }  
}
```



# Przykładowy łańcuch (pakiety przychodzące)

---

```
table inet filter {  
    chain jakaś_nazwa {  
        type filter hook input priority 0;  
        tcp dport 22 accept  
        reject  
    }  
}
```

- ❖ Akceptuj wszystkie pakiety przychodzące do portu SSH.
- ❖ Odrzuć resztę.

Czy to wystarczy?

- ❖ Pakiety wychodzące np. do portu 80 będą wypuszczane, ale odpowiedzi na nie będą odrzucane!



# Stan połączeń

---

## Nieprecyzyjna heurystyka (stosowana w prostych filtrach)

- ❖ Wpuść pakiety przychodzące do portów  $\geq 32678$ .

## Filtry stanowe

- ❖ Możemy wpuszczać pakiety związane wysłanymi pakietami.
- ❖ W szczególności segmenty TCP już nawiązanego połączenia.



# Przykładowy łańcuch (pakiety przychodzące)

---

```
table inet filter {  
    chain jakaś_nazwa {  
        type filter hook input priority 0;  
        ct state established accept  
        tcp dport 22 ct state new accept  
        reject  
    }  
}
```

- ❖ Akceptuj wszystkie pakiety przychodzące do portu SSH.
- ❖ Akceptuj odpowiedzi na już wysłane pakiety.
- ❖ Odrzuć resztę.



# Odrzucanie pakietów

---

## Zgodnie ze standardem:

- ❖ Przez reject.
- ❖ Zapora odpowie komunikatem ICMP *icmp-port-unreachable*

## Wyrzucenie pakietu bez żadnego komunikatu:

- ❖ Przez drop.
- ❖ Utrudnia pracę skanerów portów → muszą czekać na timeout dla połączenia.



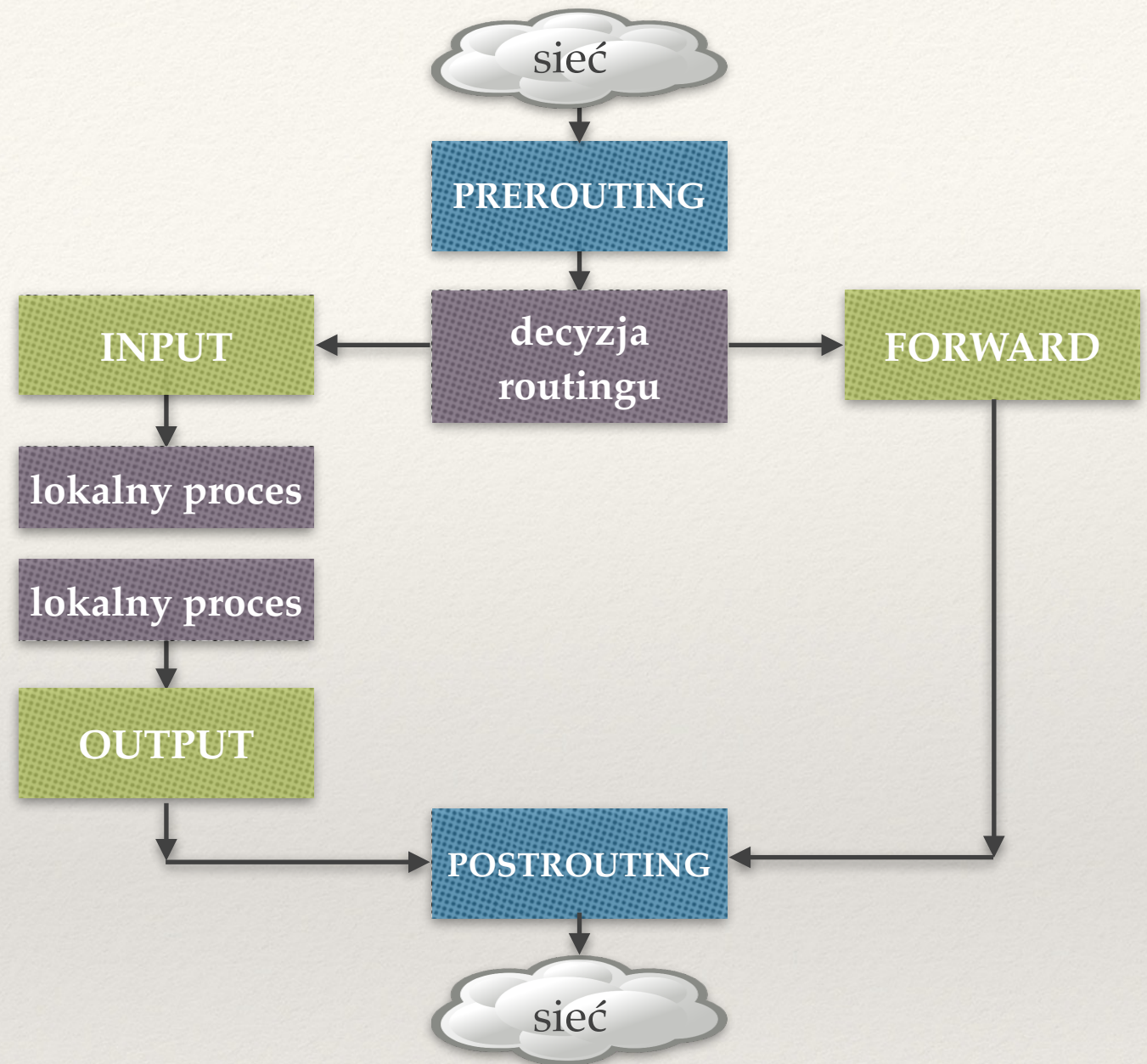
# NAT

## nat POSTROUTING:

- ❖ Podmiana źródłowych adresów IP.
- ❖ Ostatnia czynność przed wysłaniem pakietu.

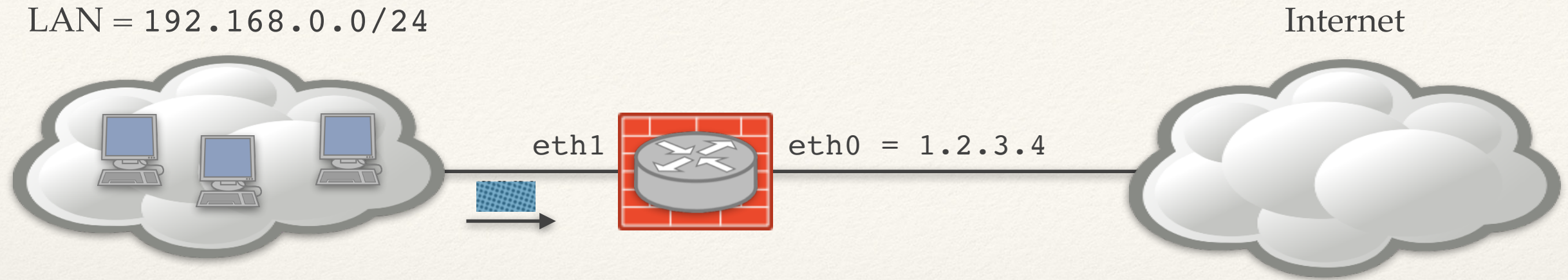
## nat PREROUTING:

- ❖ Podmiana docelowych adresów IP
- ❖ Przykładowo: przekierowanie do innego komputera.





# Źródłowy NAT (SNAT)

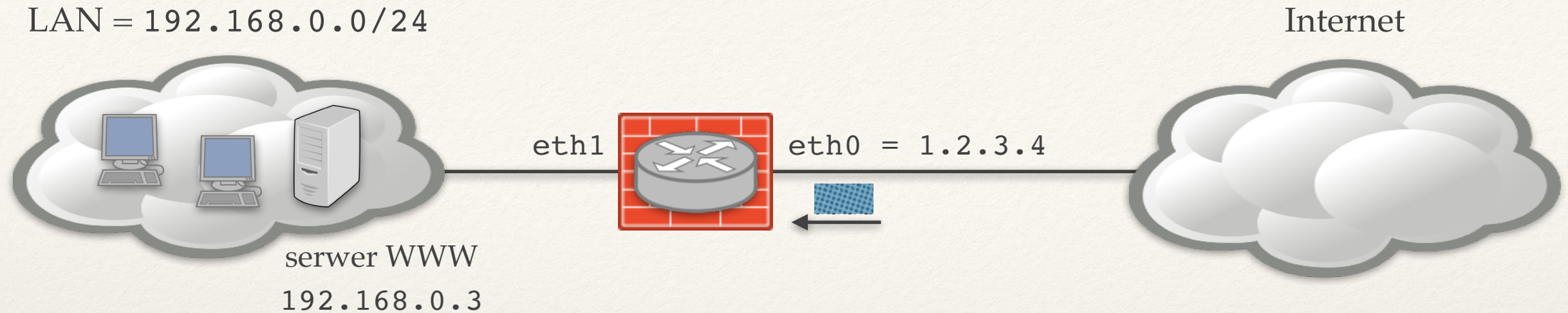


Podmiana adresu źródłowego pakietu wychodzącego z LAN:

```
table ip nat {  
    chain jakaś_nazwa {  
        type nat hook postrouting priority 0;  
        ip saddr 192.168.0.0/24 oif eth0 snat 1.2.3.4  
    }  
}
```



# Docelowy NAT (DNAT)



Podmiana adresu docelowego pakietu wychodzącego z LAN:

```
table ip nat {  
    chain jakaś_nazwa {  
        type nat hook prerouting priority 0;  
        iif eth0 tcp port 80 dnat 192.168.0.3:80  
    }  
}
```



# Lektura dodatkowa

---

- ❖ Kurose & Ross: rozdział 8.
- ❖ Tanenbaum: rozdział 8.



# Zagadnienia

---

- ❖ Co to jest pamięć CAM i jak stosuje się ją w przełącznikach? Jak można ją przepełnić?
- ❖ Opisz atak typu ARP spoofing.
- ❖ Co oznacza termin IP spoofing? Na czym polega metoda weryfikacji tak zmodyfikowanych pakietów (ingress filtering)?
- ❖ Na czym polega atak RIP spoofing?
- ❖ Na czym polega zatrucie pamięci podręcznej serwera DNS?
- ❖ Jak wygląda uwierzytelnianie serwera SSH?
- ❖ Na czym polega uwierzytelnianie użytkownika przez SSH z wykorzystaniem kluczy RSA?
- ❖ Przedstaw przykładowe ataki wykorzystujące brak sprawdzania poprawności wprowadzanych danych.
- ❖ Wyjaśnij pojęcia: robak internetowy, exploit, botnet.
- ❖ Na czym polega phishing?
- ❖ Co to jest skanowanie portów? Po co się je wykonuje?
- ❖ Co to są ataki DoS i DDoS?
- ❖ Na czym polega atak typu odbity (reflected) DoS?
- ❖ Jak działa i do czego jest wykorzystywany ICMP Traceback?
- ❖ Podaj przykłady tunelowania.
- ❖ Rozwiń skrót VPN. Do czego służy?
- ❖ Porównaj wady i zalety filtrów pakietów: prostych, stanowych i działających w warstwie aplikacji.
- ❖ Do czego służą moduły `input`, `output`, `forward` w filtrze Netfilter / `nftables`?
- ❖ W jakich łańcuchach zapory Linuksa wykonywany jest źródłowy a w jakich docelowy NAT?