

Curs 08

Securizarea rețelei



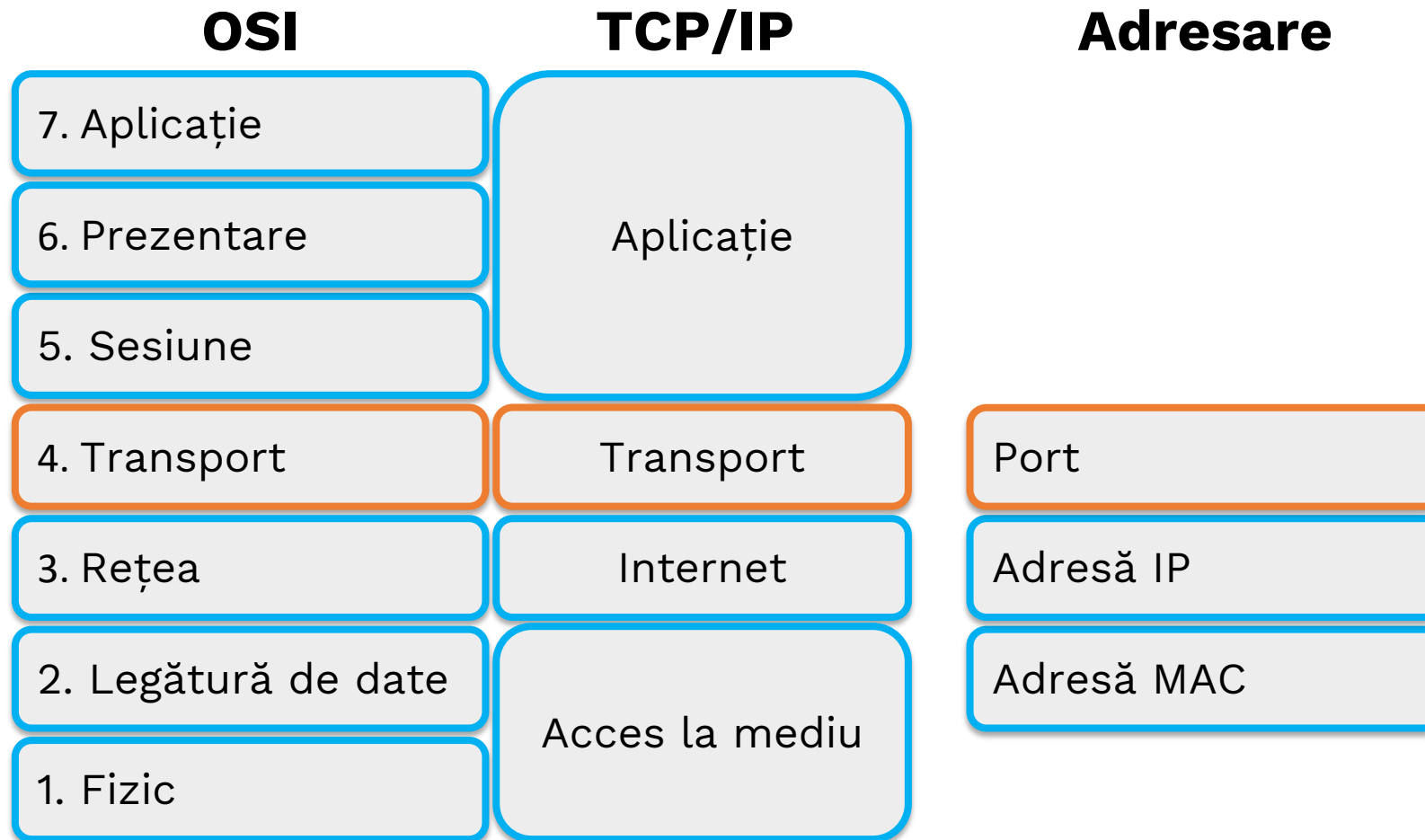
Objective

- TCP și UDP pe scurt
- Ce este un firewall
- Filtrarea pachetelor
- Implementarea în Linux: iptables

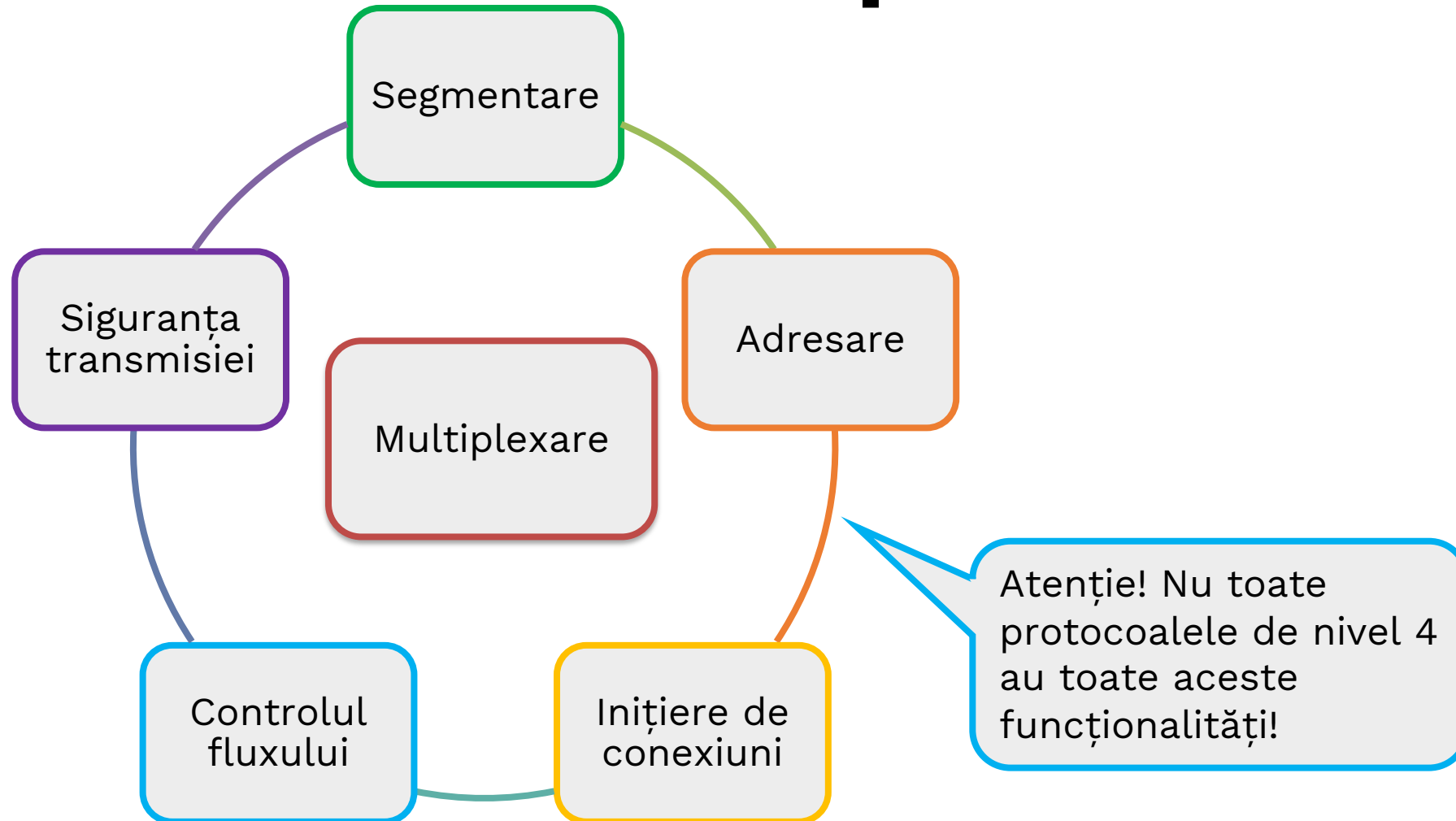
Nivelul transport



Puțină recapitulare



Rolurile nivelului transport



TCP vs UDP

TCP

- Transmission Control Protocol
- Orientat conexiune
- Protocol sigur (reliable)
 - datele ajung garantat la destinație
 - datele ajung în ordine la destinație
- Controlul fluxului
- Controlul congestiei
- Controlul erorii
- Exemple:
 - SSH
 - HTTP

UDP

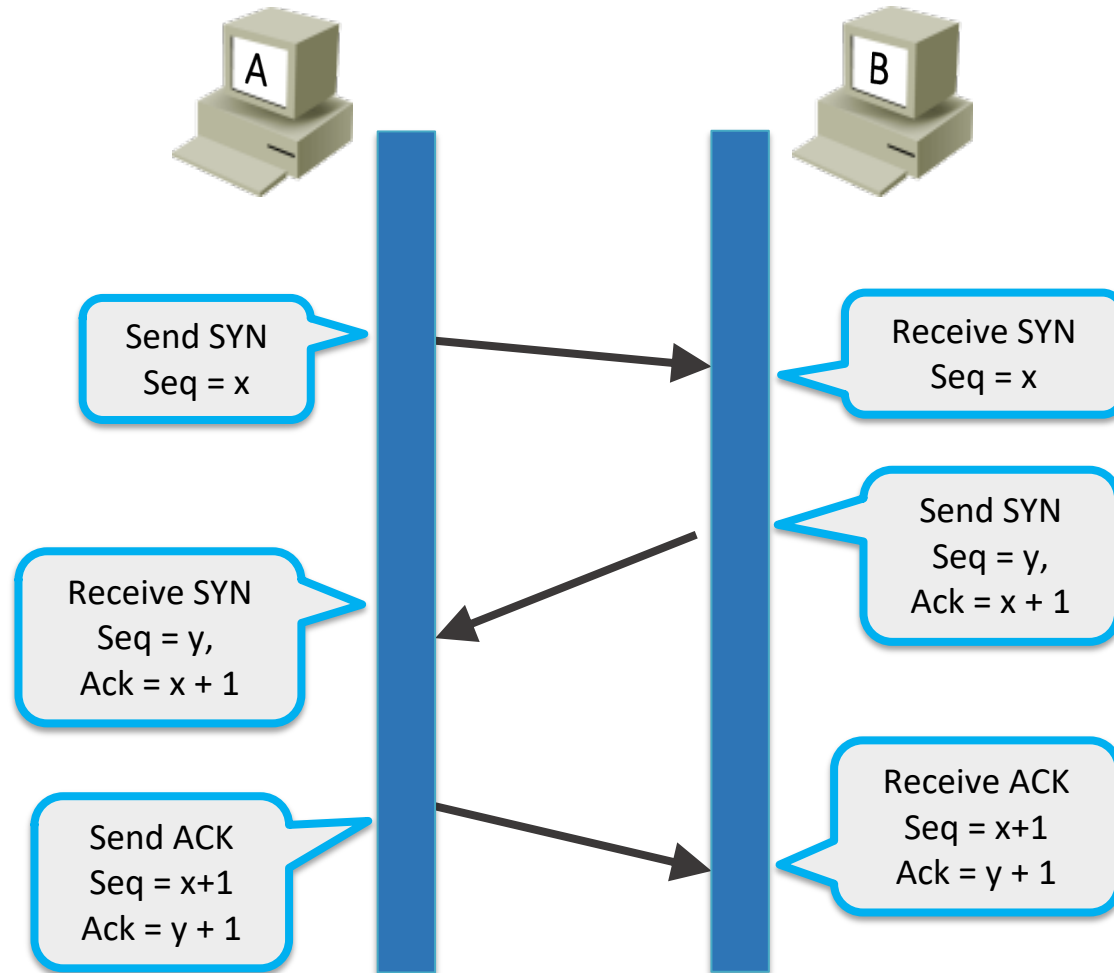
- User Datagram Protocol
- Neorientat conexiune
- Nesigur (unreliable)
 - segmente pierdute
- Fără controlul fluxului
 - segmente fără ordine
- Exemple
 - IPTV
 - VoIP

Antetul TCP – flag-uri

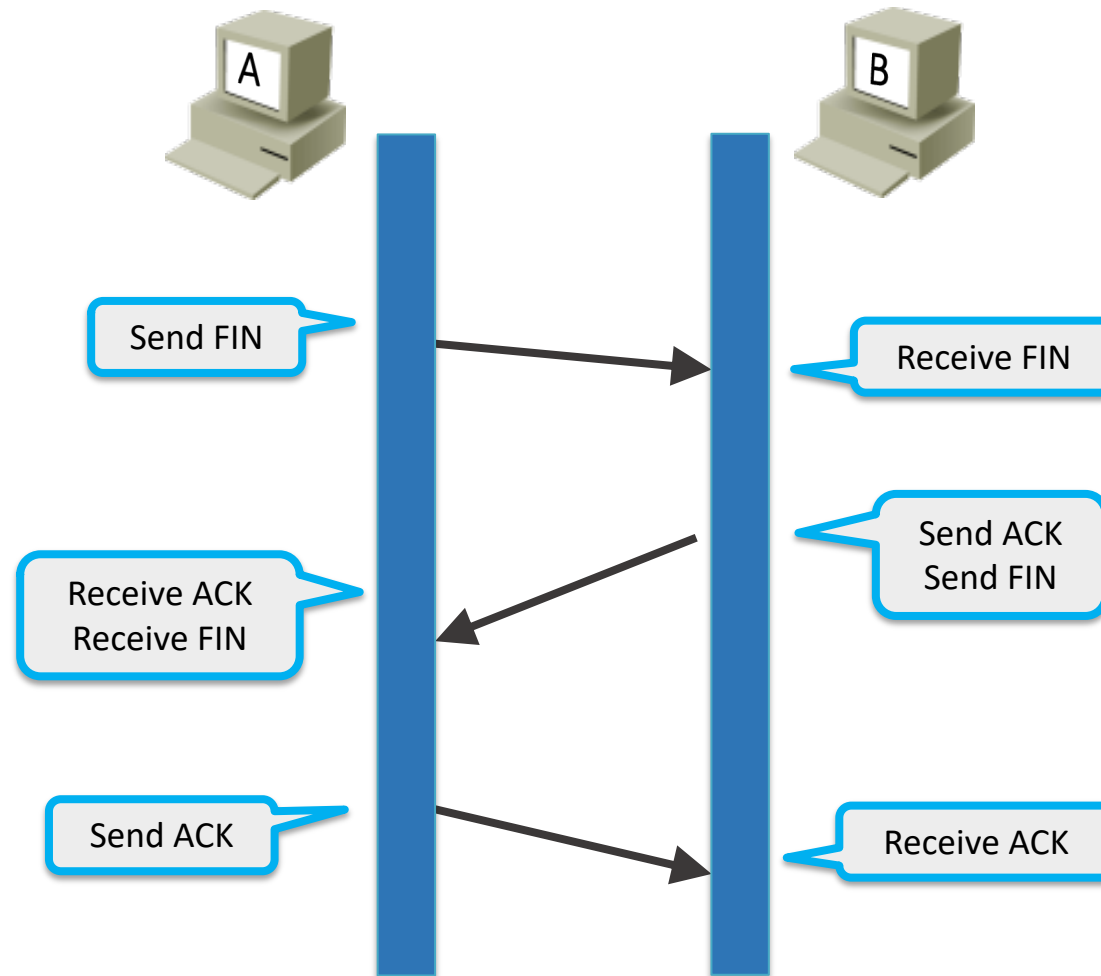
- Grup de 8 biți din antetul TCP
- Identifică diverse stări ale protocolului
- Câteva flag-uri importante sunt:
 - **ACK**
 - activare câmp “Număr de confirmare”
 - **SYN**
 - protocolul de inițiere a conexiunii (handshake)
 - stabilirea/sincronizarea numerelor de secvență
 - **FIN**
 - protocolul de încheiere a conexiunii
 - încheierea transmisiei de la FIN-sender



Inițierea conexiunii TCP



Încheierea conexiunii



Rolul unui firewall



Ce este un firewall

- Mecanism folosit pentru blocarea traficului nedorit din rețea
- Poate fi implementat:
 - Pe un dispozitiv de rețea
 - Ruter Cisco
 - Ca un dispozitiv dedicat
 - Cisco ASA
 - Fortinet Fortigate
 - Pe un end-device (host sau server)
 - ZoneAlarm
 - Windows Firewall
 - Netfilter/iptables



De ce avem nevoie de firewall-uri

- Internetul nu este un loc sigur
- Rețeaua locală poate fi oricând ținta unui atac:
 - De recunoaștere
 - Ping sweep
 - Sniffing
 - Port scan
 - De DoS (Denial of Service) sau DDoS (Distributed DoS)
 - Smurf attack
 - SYN flood
 - De acces
 - Atacarea unei parole (cu dicționar sau brute-force)
 - Buffer overflow
 - Man-in-the-middle

Rolul unui firewall

- Atac de recunoaștere
 - Atacatorul încearcă să descopere mașini și serviciile de pe acestea
 - Exemplu: ICMP echo request către o adresă de broadcast descoperă toate mașinile din rețea
 - Un Firewall poate:
 - Bloca porturile vulnerabile
 - Bloca inițierea din exterior a conexiunilor
 - Bloca răspunsul la ICMP echo request

Rolul unui firewall

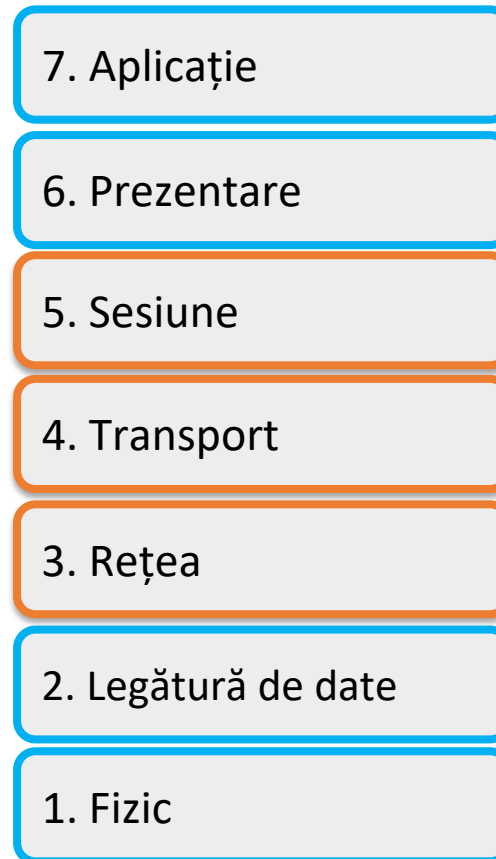
- Atac DoS sau DDoS
 - În general bazate pe generarea unei cantități mari de trafic ce supraîncarcă rețeaua sau serverul
 - Din cauza supraîncărcării, traficul riscă să fie ignorat
 - Un Firewall poate:
 - Monitoriza numărul sesiunilor TCP Half-Open către un server și le poate închide dacă trec de un prag
 - Bloca directed broadcasts

Tipuri de firewall

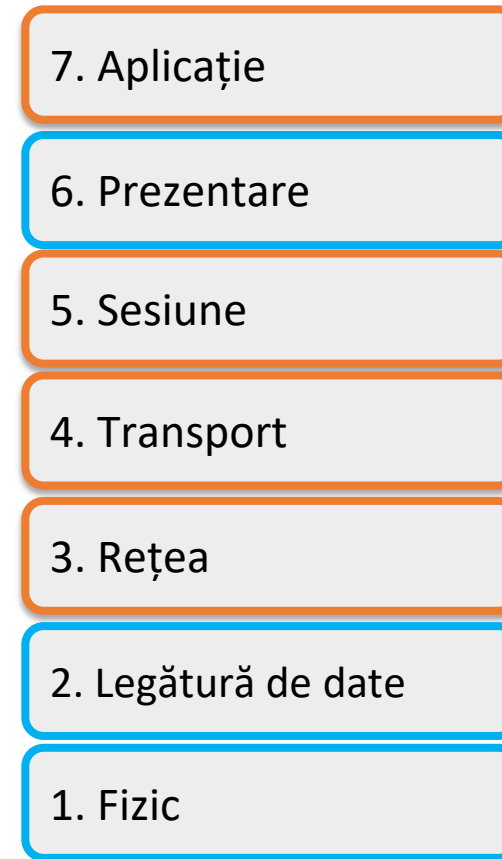
Stateless firewall



Stateful firewall



Firewall de nivel aplicație (Proxy firewall)



iptables



iptables

- Utilitar Linux
- Face parte din proiectul Netfilter
- Permite unei mașini Linux să:
 - Filtreze pachetele
 - Translateze adrese
 - Rescrie câmpurile unui pachet
- Configurat prin scrierea de **reguli**
- Regulile iptables sunt compuse din două secțiuni principale:
 - Șablon – ce valori trebuie să aibă câmpurile din pachet pentru a se acționa asupra lor
 - Acțiune – ce operație va efectua mașina Linux asupra pachetului

Tabele iptables

- **Filter**

- Conține reguli ce spun ce trafic poate să treacă și ce trafic trebuie aruncat
- Exemplu:
 - O adresă externă a eșuat în mod repetat să se conecteze la un server Linux prin SSH
 - Se adaugă o regulă de filtrare care blochează orice trafic de la adresa respectivă

- **Nat**

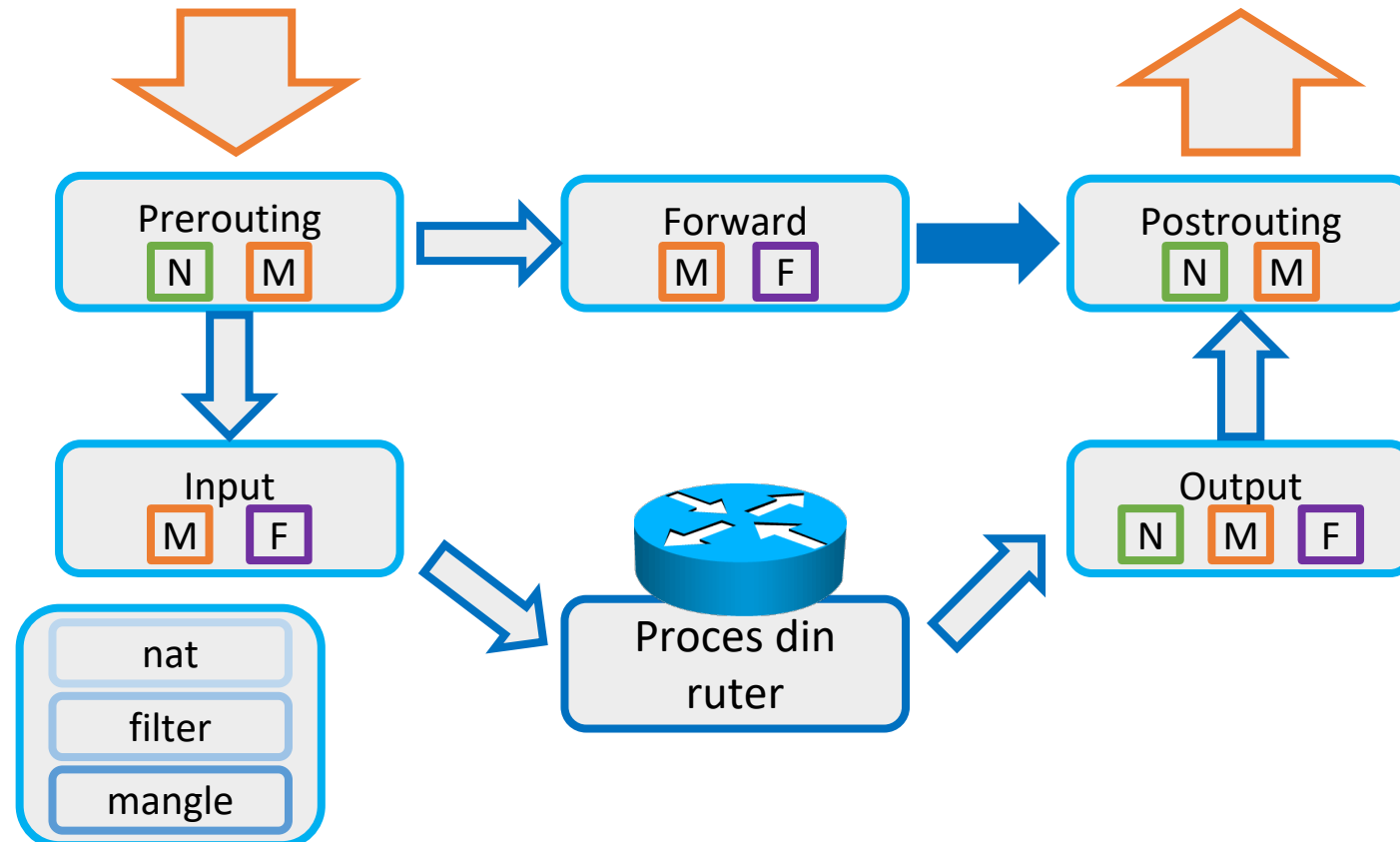
- Conține reguli pentru translatarea adreselor în procesul de NAT
- Exemplu:
 - O adresă privată trebuie să acceseze un server din Internet
 - Se adaugă o regulă de NAT care rescrie adresa sursă privată cu o adresă publică
 - La întoarcere, pachetul va fi rescris invers
 - (Mult) mai multe detalii în cursul viitor

- **Mangle**

- Conține reguli pentru alterarea specializată a pachetelor

Lanțuri iptables

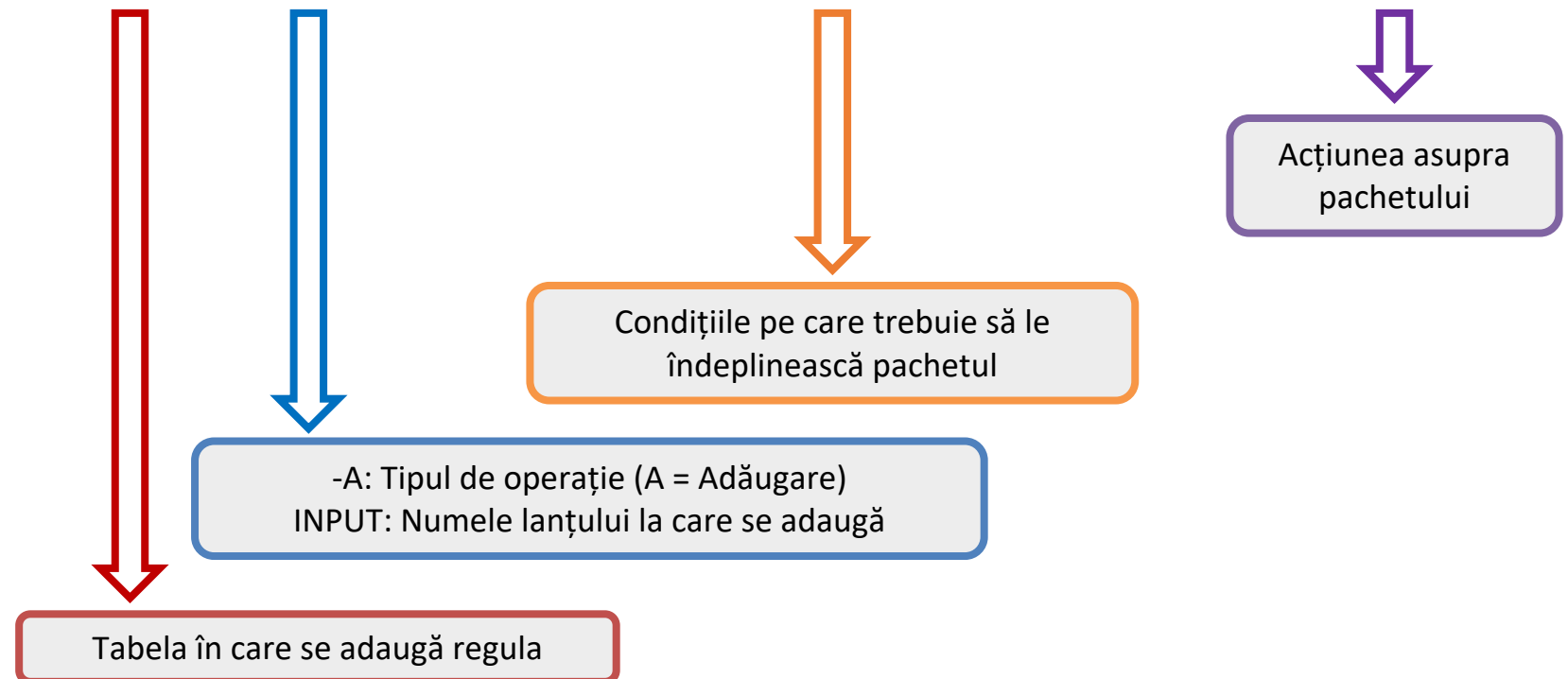
- Liste de reguli aplicate implicit unui anumit subset de trafic



Filtrarea pachetelor cu iptables

- Regulile sunt configurate de fapt prin comenzi iptables

```
ubuntu# iptables -t filter -A INPUT -s 10.0.0.0/8 -p icmp -j DROP
```



Filtrarea pachetelor cu iptables

```
ubuntu# iptables -t filter -A INPUT -s 10.0.0.0/8 -p icmp -j DROP
```

- Tabela este implicit filter
 - Regula putea fi deci scurtată ca fiind:

```
ubuntu# iptables -A INPUT -s 10.0.0.0/8 -p icmp -j DROP
```
- Opțiunile permise pentru acest parametru sunt:
 - filter
 - nat
 - mangle
 - raw
 - Folosită pentru configurarea excepțiilor de monitorizare a conexiunilor

Filtrarea pachetelor cu iptables

```
ubuntu# iptables -t filter -A INPUT -s 10.0.0.0/8 -p icmp -j DROP
```

- INPUT poate fi înlocuit cu orice alt lanț predefinit
- Pot fi create și lanțuri noi de către administrator
- Operațiile permise sunt:

| | | |
|-----------|----------------|--------------------------------|
| -A | --append | Adăugarea unei reguli la final |
| -D | --delete | Ștergerea unei reguli |
| -L | --list | Afișarea regulilor |
| -F | --flush | Ștergerea tuturor regulilor |
| -N | --new-chain | Crearea unui lanț nou |
| -X | --delete-chain | Ștergerea unui lanț |
| -P | --policy | Schimbarea politicii implicite |

Filtrarea pachetelor cu iptables

```
ubuntu# iptables -t filter -A INPUT -s 10.0.0.0/8 -p icmp -j DROP
```

- Selectarea traficului se face pe baza informațiilor din pachet
- Fără specificarea unui protocol, se pot face reguli conținând:
 - Interfața de intrare (**-i**)
 - Interfața de ieșire (**-o**)
 - Adresa IP destinație (**-d**)
 - Adresa IP sursă (**-s**)

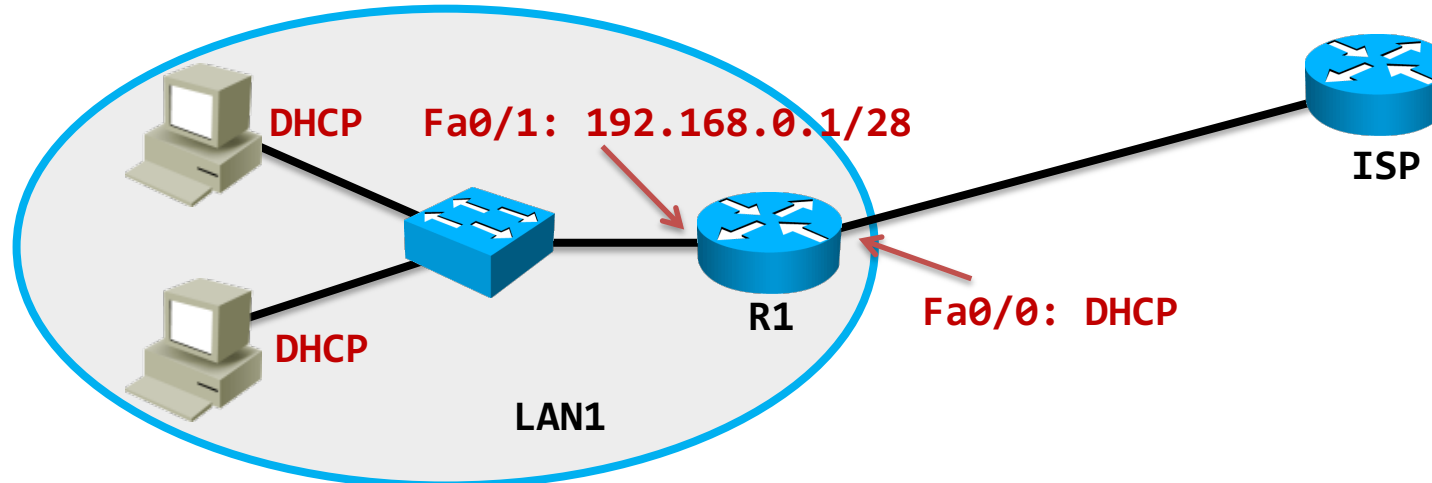
Filtrarea pachetelor cu iptables

```
ubuntu# iptables -t filter -A INPUT -s 10.0.0.0/8 -p icmp -j DROP
```

- Reprezintă operația ce va fi făcută asupra pachetului
- În terminologia iptables, **j** vine de la **jump** și **DROP** este un **target**
- Poate fi omisă
 - În acest caz regula nu face nimic, însă contorul regulii va fi incrementat
- Target-uri uzuale sunt:
 - ACCEPT: pachetul este acceptat
 - DROP: pachetul este aruncat
 - LOG: este adăugată în log-urile sistemului o înregistrare

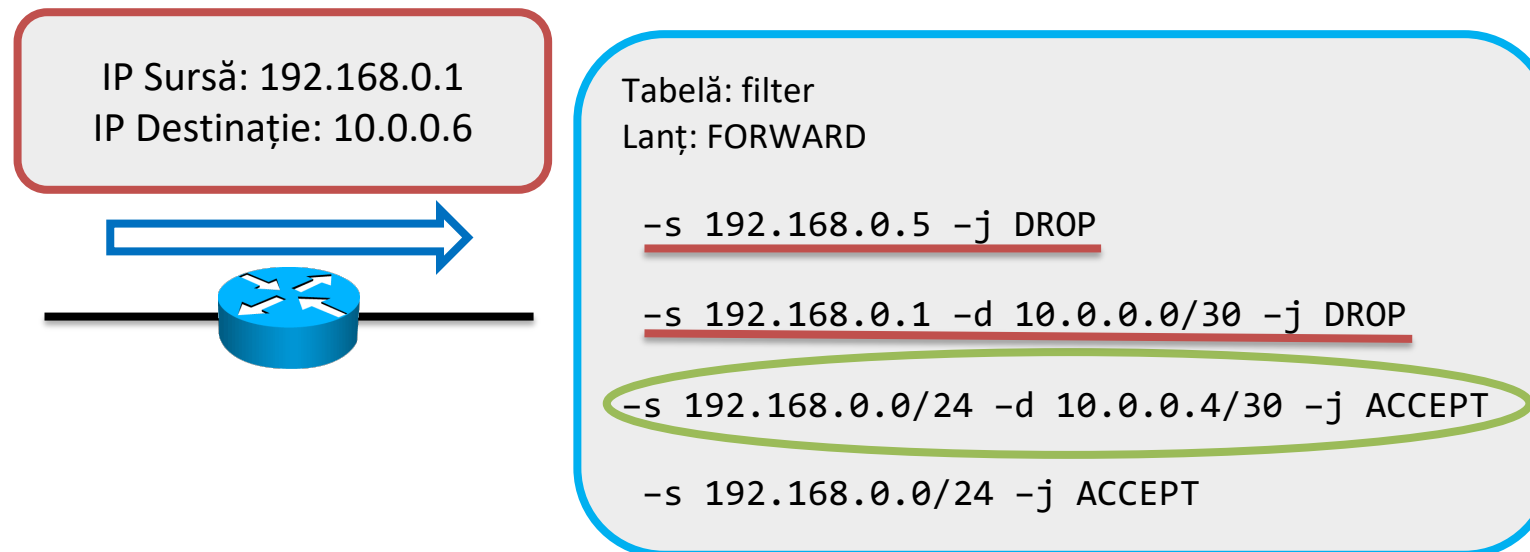
Exercițiul 1

- Să se scrie o regulă iptables care permite trecerea traficului de la stația 192.168.10.1 către serverul 192.168.10.40.
 - **R:** `iptables -A FORWARD -s 192.168.10.1 -d 192.168.10.40 -j ACCEPT`
- Să se scrie o regulă care blochează orice trafic destinat ruterului R1 de la stațiile din rețeaua LAN1. Traficul ce doar tranzitează ruterul trebuie să fie permis.
 - **R:** `iptables -A INPUT -s 192.168.0.0/28 -j DROP`



Cum funcționează iptables

- La întâlnirea unui pachet, acesta este evaluat secvențial conform fiecărei reguli dintr-un lanț
- Dacă se face match pe o regulă cu un target ACCEPT sau DROP, procesarea se termină și pachetul este acceptat sau aruncat
- Ce se întâmplă dacă nu se face match pe nicio regulă?



Politici iptables

- Fiecare lanț predefinit are o politică implicită
 - Lanțurile create de utilizator NU pot avea politică implicită
- Politica este acțiunea implicată (*target*) ce este aleasă pentru fiecare pachet ce nu găsește o regulă în cadrul lanțului (*match*)
- Politicile implicite sunt **ACCEPT**
- Politica unui lanț poate fi modificată:
 - `iptables -P FORWARD DROP`



Exercițiul 2

- Ruterele de la marginea unei rețele private implementează de obicei antispoofing:
 - Nu permit intrarea în rețea a pachetelor cu adrese private
 - Nu permit ieșirea din rețea a pachetelor cu adrese private
- Configurați o politică antispoofing folosind iptables
 - **R:**
 - `iptables -A FORWARD -s 192.168.0.0/16 -j DROP`
 - `iptables -A FORWARD -s 172.16.0.0/12 -j DROP`
 - `iptables -A FORWARD -s 10.0.0.0/8 -j DROP`
 - `iptables -A FORWARD -d 192.168.0.0/16 -j DROP`
 - `iptables -A FORWARD -d 172.16.0.0/12 -j DROP`
 - `iptables -A FORWARD -d 10.0.0.0/8 -j DROP`

Extensii iptables

- Adesea adresele IP și interfețele fizice nu sunt suficiente pentru a implementa cerințele de securitate
 - Se poate permite accesul doar către serviciul de HTTP?
 - Se poate permite stabilirea conexiunilor TCP doar într-o direcție?
 - Se pot bloca ping-urile către interior păstrând încă posibilitatea de a da ping către exterior?
- Iptables permite activarea de **extensii**, module ce oferă noi posibilități în specificarea regulilor
- Extensiile se activează cu `-p` (protocol) sau `-m` (module)
- Extensiile cele mai importante sunt:
 - `tcp`
 - `udp`
 - `icmp`

Extensii iptables

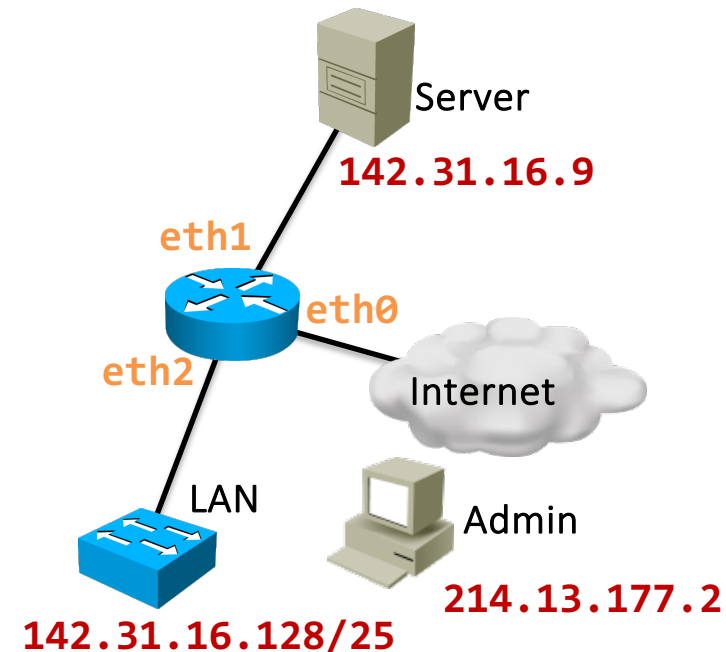
- Extensia tcp permite filtrarea traficului după:
 - Port destinație **--dport --destination-port**
 - Port sursă **--sport --source-port**
 - Flag-uri TCP (SYN, ACK, FIN, etc.) **--tcp-flags, --syn**
- Extensia icmp permite filtrarea traficului după:
 - Tipul pachetului ICMP **--icmp-type <type>** unde type poate fi:
 - echo-request
 - echo-reply
 - time-exceeded
 - Pentru toate valorile lui type, puteți rula:
linux# iptables -p icmp -h

Exercițiul... final

- Fie topologia de mai jos
 - Ruterul este o mașină Linux ce a fost deja configurată cu regulile iptables din stânga
 - Determinați comenzile necesare pentru a rezolva fiecare „ticket”
 - Switch-ul reprezintă o rețea de host-uri cu adresare DHCP

```
Chain: INPUT; Policy: ACCEPT
-i eth0 -j DROP
```

```
Chain: FORWARD; Policy: DROP
-i eth0 -j ACCEPT
```



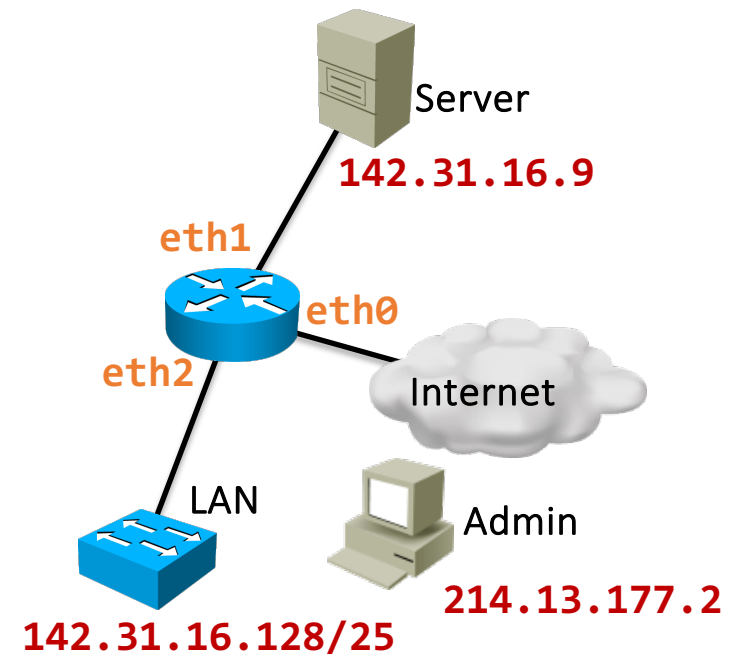
Exercițiul... final

- Ticket #1
 - Stațiile din LAN nu pot comunica cu Server.
 - Care este motivul? Care este soluția?

```
linux# iptables -P FORWARD ACCEPT
```

```
Chain: INPUT; Policy: ACCEPT
-i eth0 -j DROP
```

```
Chain: FORWARD; Policy: DROP ACCEPT
-i eth0 -j ACCEPT
```



Exercițiul... final

- Ticket #2

- Configurați iptables a.î. doar stațiile din LAN să poată folosi serviciul de HTTP de pe Server.

```
linux# iptables -F FORWARD
```

```
linux# iptables -A FORWARD -s 142.31.16.128/25 -p tcp --dport 80 -j ACCEPT
```

```
linux# iptables -A FORWARD -d 142.31.16.9 -p tcp --dport 80 -j DROP
```

Chain: INPUT; Policy: ACCEPT

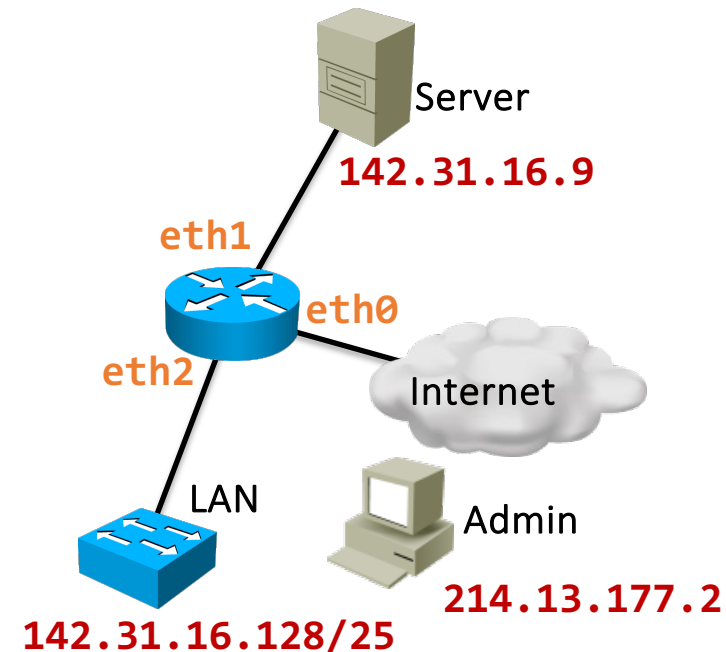
```
-i eth0 -j DROP
```

Chain: FORWARD; Policy: ACCEPT

```
-i eth0 -j ACCEPT
```

```
-s 142.31.16.128/25 -p tcp --dport 80 -j ACCEPT
```

```
-d 142.31.16.9 -p tcp --dport 80 -j DROP
```



Exercițiul... final

- Ticket #3
 - Configurați iptables a.î. doar Admin să poată accesa prin SSH ruterul

```
linux# iptables -F INPUT
```

```
linux# iptables -A INPUT -s 214.13.177.2 -p tcp --dport 22 -j ACCEPT
```

```
linux# iptables -A INPUT -p tcp --dport 22 -j DROP
```

Chain: INPUT; Policy: ACCEPT

```
-i eth0 -j DROP
```

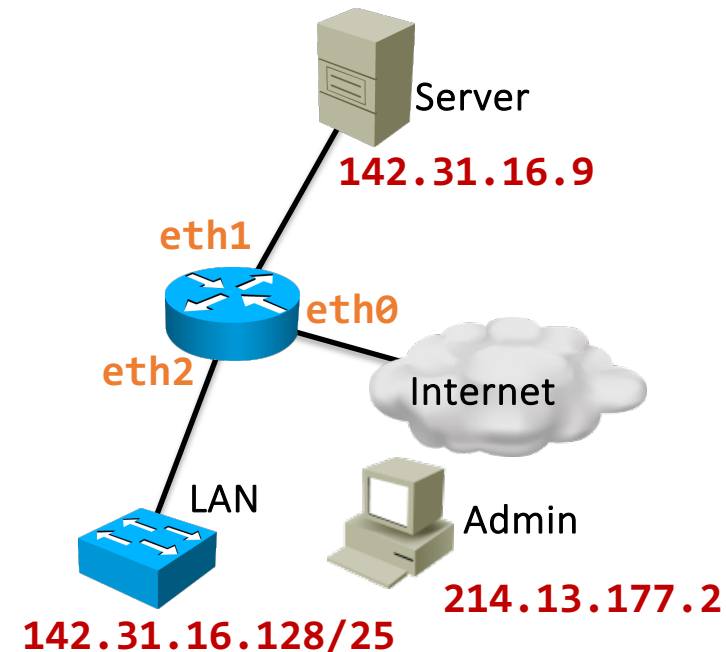
```
-s 214.13.177.2 -p tcp --dport 22 -j ACCEPT
```

```
-p tcp --dport 22 -j DROP
```

Chain: FORWARD; Policy: ACCEPT

```
-s 142.31.16.128/25 -p tcp --dport 80 -j ACCEPT
```

```
-d 142.31.16.9 -p tcp --dport 80 -j DROP
```



Exercițiul... final

- Ticket #4

- Configurați iptables a.î. sesiunile TCP din LAN să poată fi inițiate doar dinspre interior

```
linux# iptables -A FORWARD -p tcp --syn -j DROP ???
```

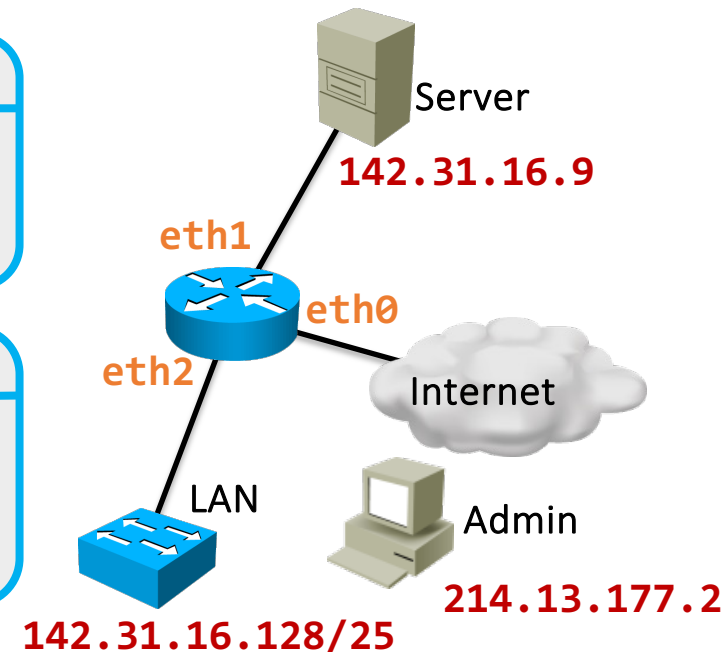
```
linux# iptables -A FORWARD -i eth0 -p tcp --syn -j DROP
```

Chain: INPUT; Policy: ACCEPT

```
-s 214.13.177.2 -p tcp --dport 22 -j ACCEPT
-p tcp --dport 22 -j DROP
```

Chain: FORWARD; Policy: ACCEPT

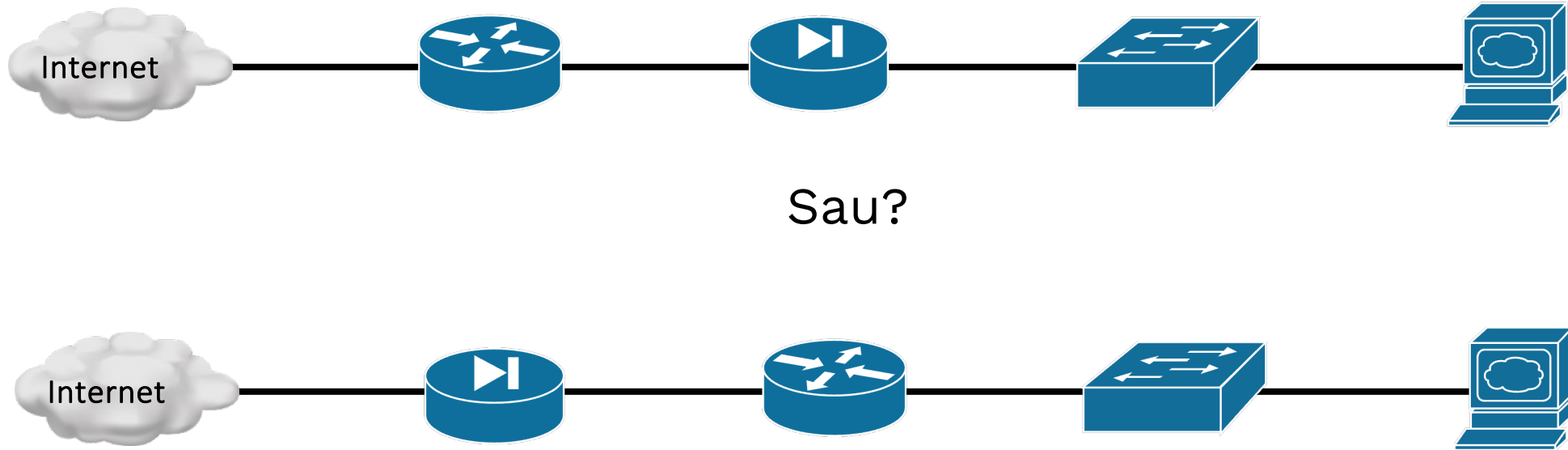
```
-s 142.31.16.128/25 -p tcp --dport 80 -j ACCEPT
-d 142.31.16.9 -p tcp --dport 80 -j DROP
-i eth0 -p tcp --syn -j DROP
```



Perimeter defence

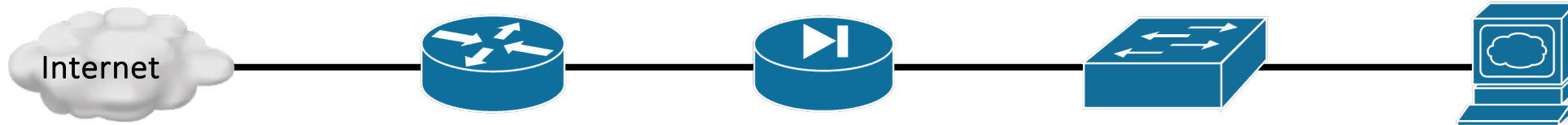


Customer edge design

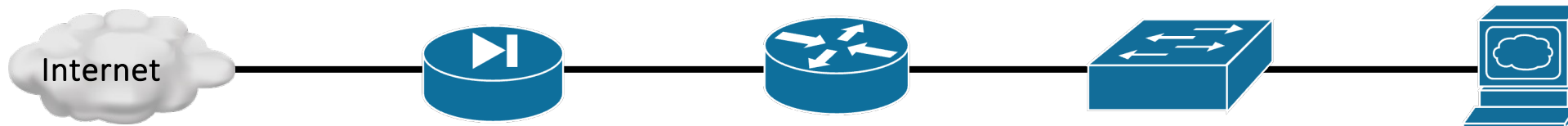


Customer edge design

- Decizia este cel mai adesea impusă de ISP



- Este varianta cea mai frecventă pentru rețele medii și mari, datorită cerințelor de rutare ce nu pot fi adesea îndeplinite de firewall



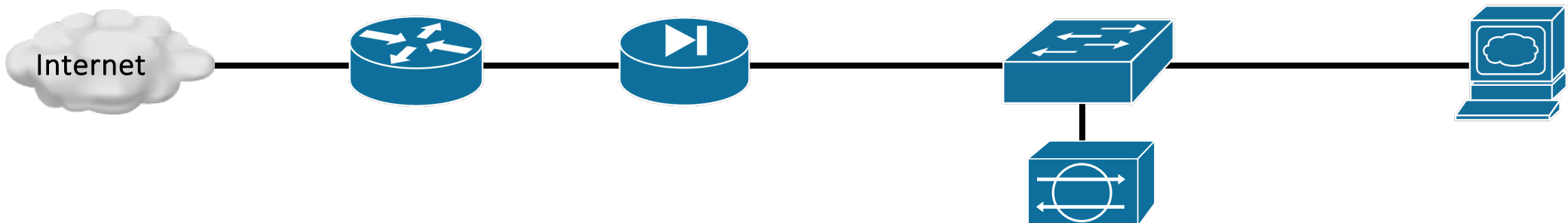
- Pentru rețele cu rutare statică sau cu cerințe scăzute de complexitate, putem colapsa ruterul și firewall-ul, sau folosi compartimentarea funcțiilor cu protecția ruterului de către firewall

Intrusion Prevention System

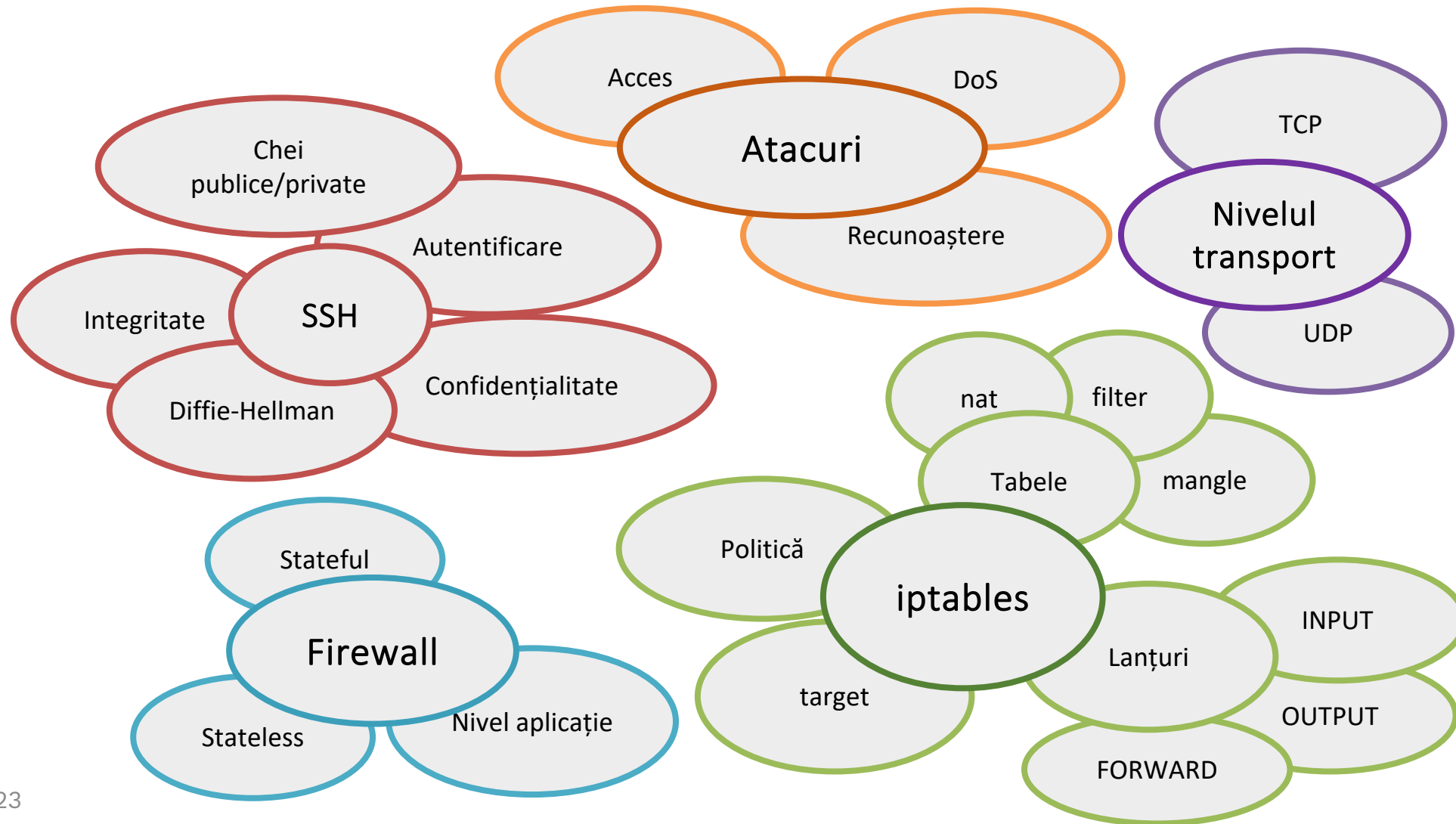
- Pentru inspectarea conținutului pachetelor putem adăuga în topologie un IPS, Intrusion Prevention Systems



- Un IPS poate deveni un punct de gâtuire a traficului în rețea
- Putem asigura doar detecția prin inspectia traficului out-of-band, prin folosirea unui Intrusion Detection System



Sumar



SSH



RL

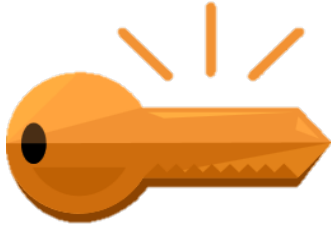
crunch it connected

SSH

- **S**ecure **S**Hell
- Protocol folosit pentru accesul sigur la distanță
- Permite execuția de comenzi pe mașina accesată
- Două versiuni majore existente: SSH-1 și SSH-2
 - SSH-1 are vulnerabilități majore
 - Cursul va aborda în continuare versiunea SSH-2
- Rol similar cu protocolul Telnet
- Funcționează pe portul TCP 22



Conceptele securității



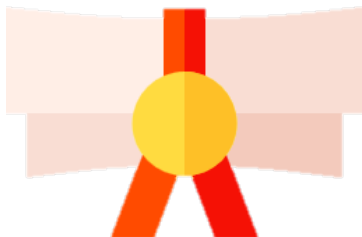
Autentificare

Sursa și destinația sunt cine spun că sunt



Confidențialitate

Doar sursa și destinația pot vizualiza informația



Integritate

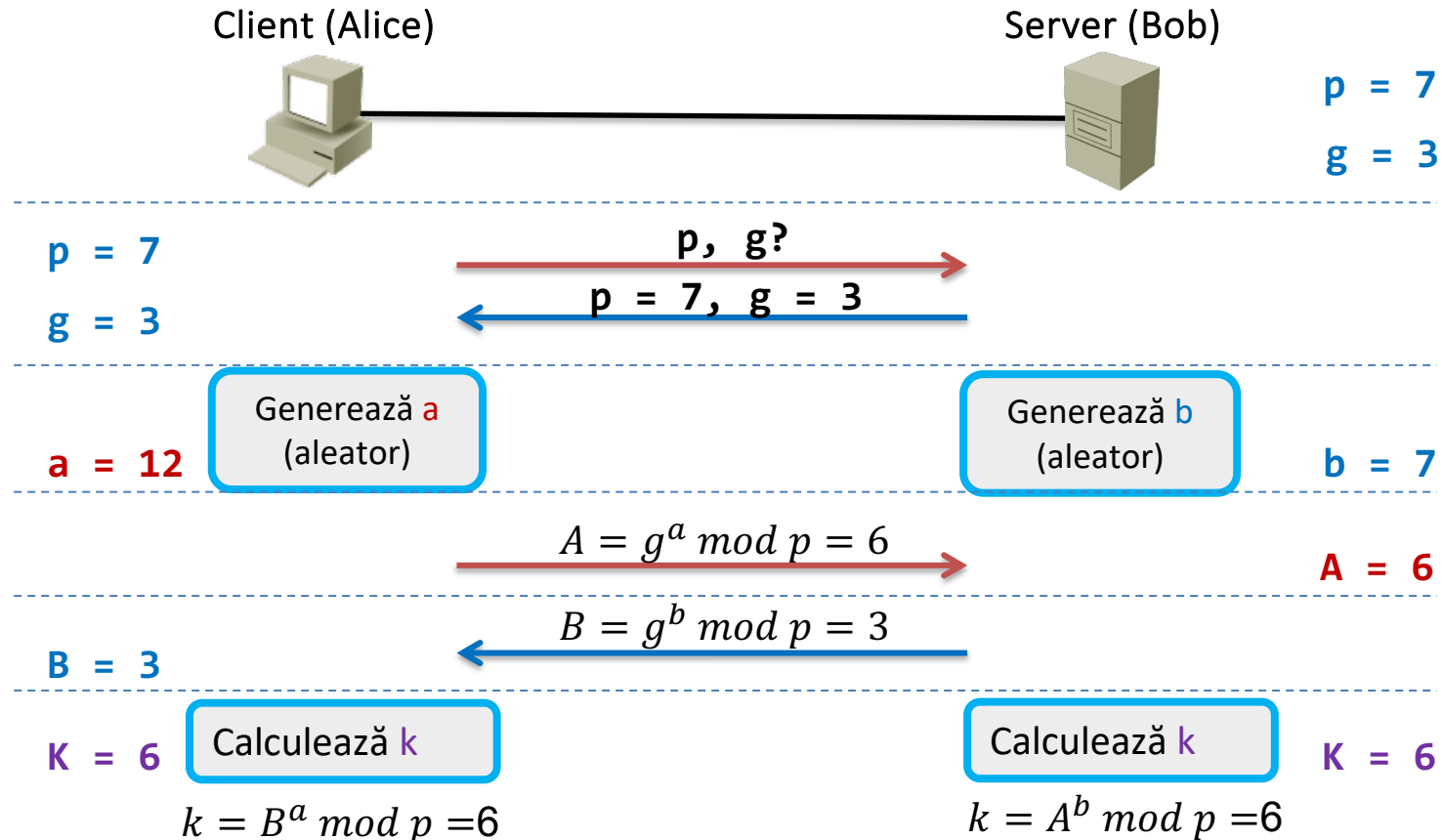
Mesajul ajuns la destinație nu a fost modificat pe parcurs

Funcții SSH – Confidențialitate

- Sunt folosiți algoritmi de criptare simetrică
 - AES: Advanced Encryption Standard
 - 3DES: Triple Data Encryption Standard
 - IDEA, DES, ARCFOUR, BLOWFISH, TSS
- Criptare simetrică = cheie comună
- 3DES este o variantă populară
 - Necesită o cheie comună pe 168, 112 sau 56 de biți
- Nu vrem să trimitem cheia pe canal pentru a nu fi interceptată
 - Trebuie stabilită o cheie comună fără ca aceasta să fie transmisă
 - Soluție: Diffie-Hellman Key Exchange

Funcții SSH – Confidențialitate cu DH

- Serverul SSH ține o listă de perechi (p, g) cu proprietăți matematice speciale



Funcții SSH – Autentificare

- Două metode principale de autentificare:
 - Prin parolă
 - Prin chei asimetrice
- Autentificarea are loc **după** stabilirea unui canal criptat cu Diffie-Hellman
- Având în vedere că parola este transmisă printr-un canal criptat, vedeți vreo problemă cu această metodă?
 - **R:** Serverul va decripta parola pentru a valida autentificarea; dacă serverul e compromis, parola va fi descoperită
 - **R:** Parolele sigure sunt greu de ținut minte
- Este preferată folosirea cheilor asimetrice

Chei asimetrice

- Se bazează pe perechi de chei aflate într-o relație matematică:
 - Cheia publică (K^+)
 - Cheia privată (K^-)

- Dându-se un mesaj M , există următoarea relații:

$$K^+(K^-(M)) = M$$

$$K^-(K^+(M)) = M$$

- Cu alte cuvinte, un client poate:
 - A avea configurată pe server cheia sa publică K^+ (de un administrator de exemplu)
 - Cripta un mesaj cu K^-
 - Serverul va putea decripta mesajul cu K^+
- Exemplu de algoritm: RSA

SSH – Autentificare prin cheie privată

1. DH

- Pasul 1:
 - Sesiunea sigură este stabilită prin Diffie-Hellman



SSH – Autentificare prin cheie privată

1. DH

2. Cerere

- Pasul 2:
 - Clientul cere autentificarea cu user-ul **foo**



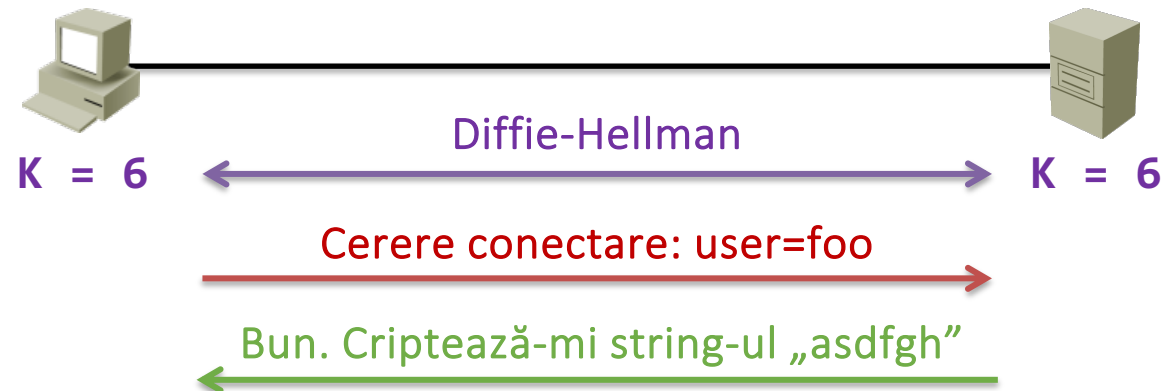
SSH – Autentificare prin cheie privată

1. DH

2. Cerere

3. Challenge

- Pasul 3:
 - Serverul trimite un **challenge**



SSH – Autentificare prin cheie privată

1. DH

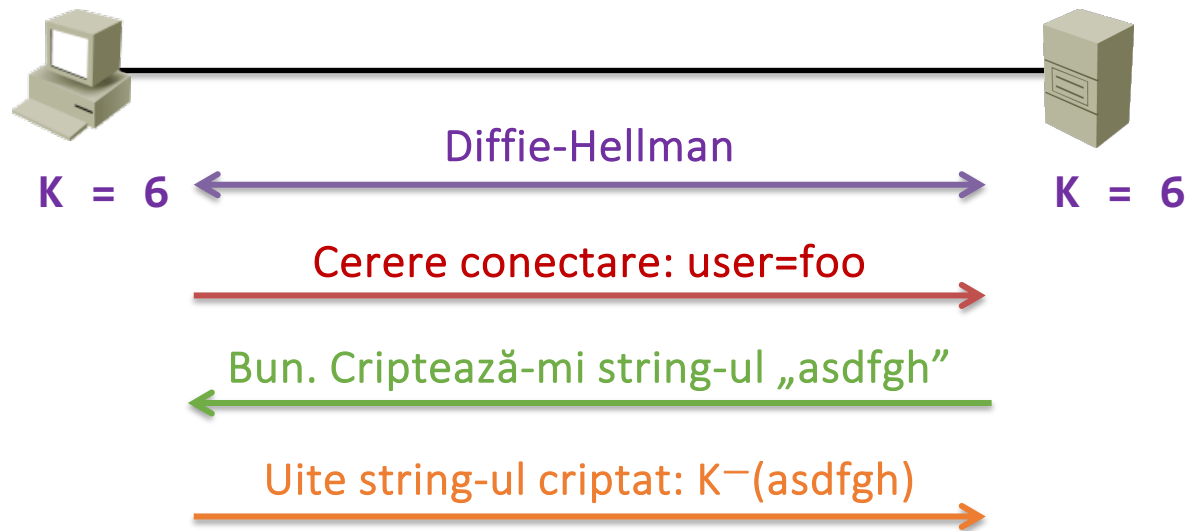
2. Cerere

3. Challenge

4. Authenticator

• Pasul 4:

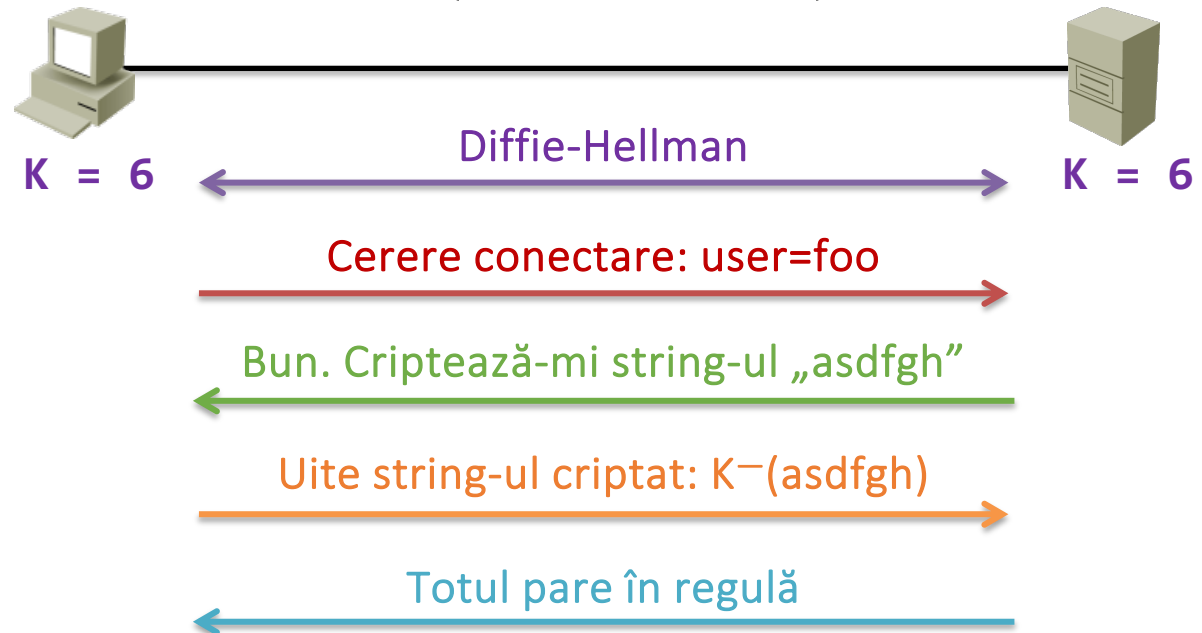
- Clientul criptează challenge-ul cu cheia sa privată
- Răspunsul său poartă numele de **authenticator**



SSH – Autentificare prin cheie privată

- 1. DH
- 2. Cerere
- 3. Challenge
- 4. Authenticator
- 5. Reply

- Pasul 5:
 - Serverul folosește cheia publică preconfigurată a clientului și verifică $K^+(K^-(asdfgh)) = asdfgh$



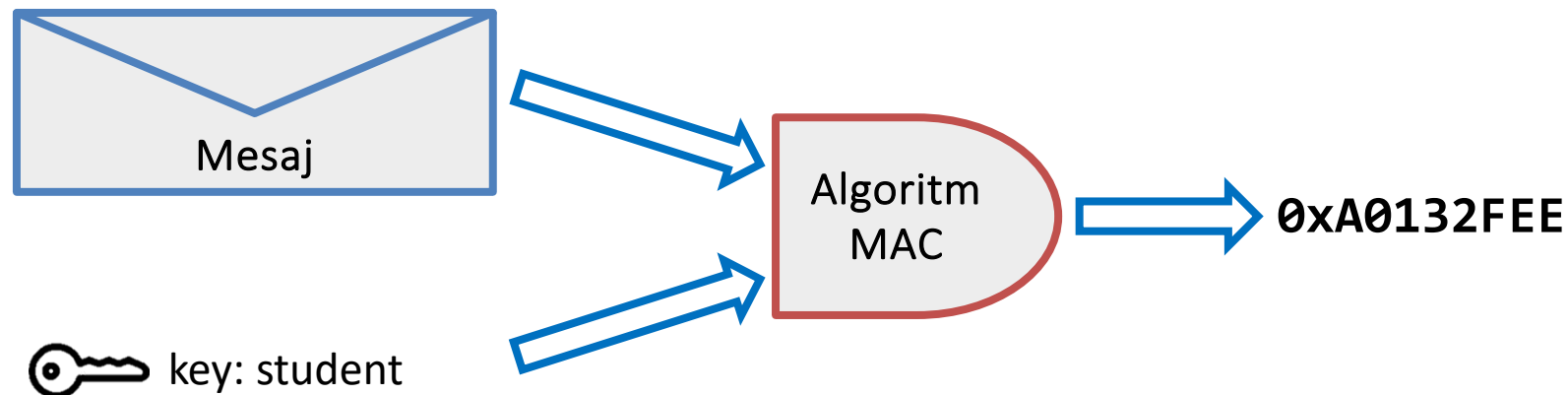
SSH – Autentificare prin cheie privată



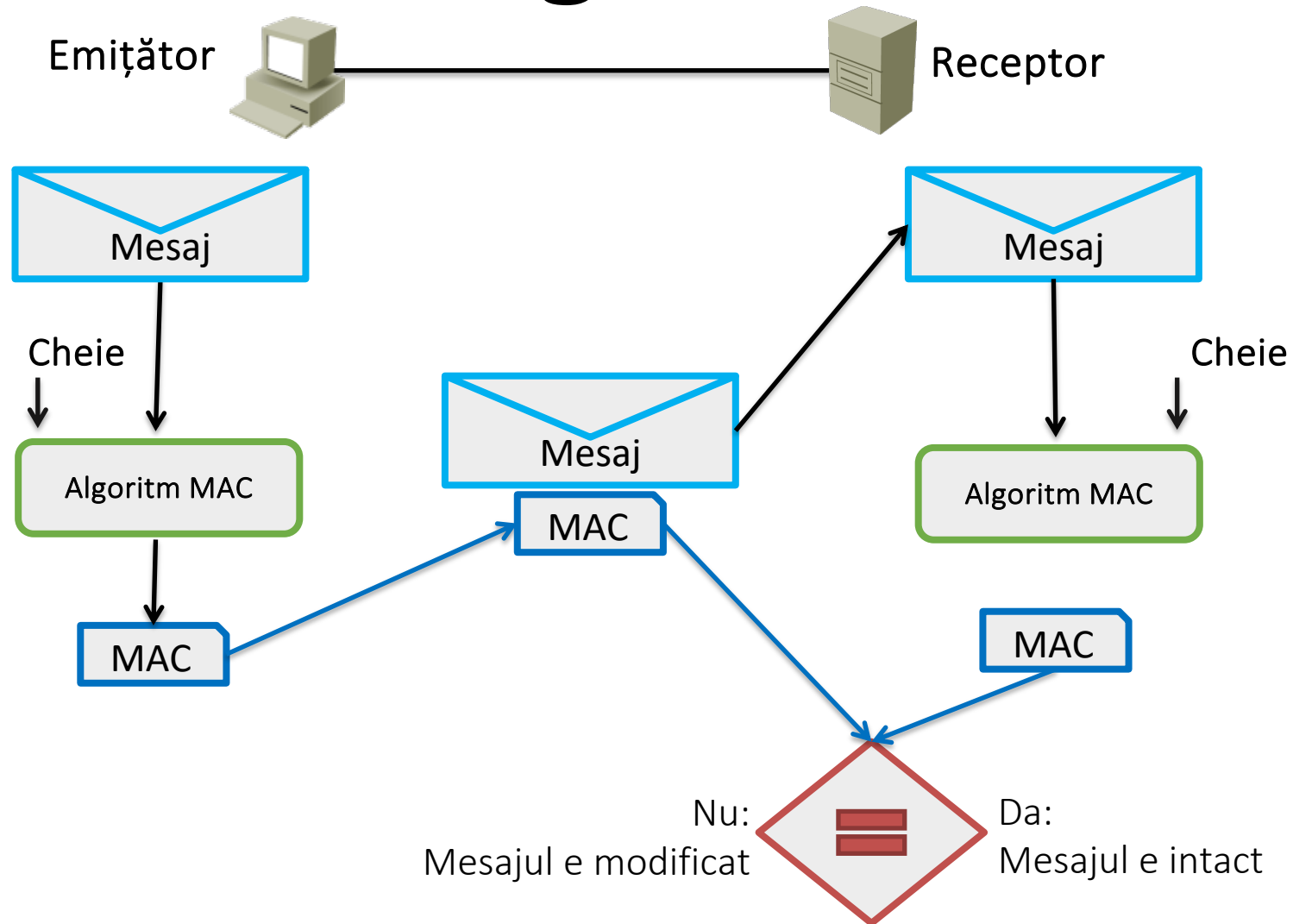
- Doar **challenge-ul** este criptat cu cheia privată
 - Cu alte cuvinte, este folosită strict pentru operațiile de autentificare
- Motivul este eficiența:
 - Cheile asimetrice sunt ineficiente în operațiile de criptare/decriptare
 - Impactul criptării asimetrice a întregului trafic este mult prea mare

Funcții SSH – Integritate

- Funcție realizată prin MAC
 - Message Authentication Code
 - Este de fapt un hash cu cheie

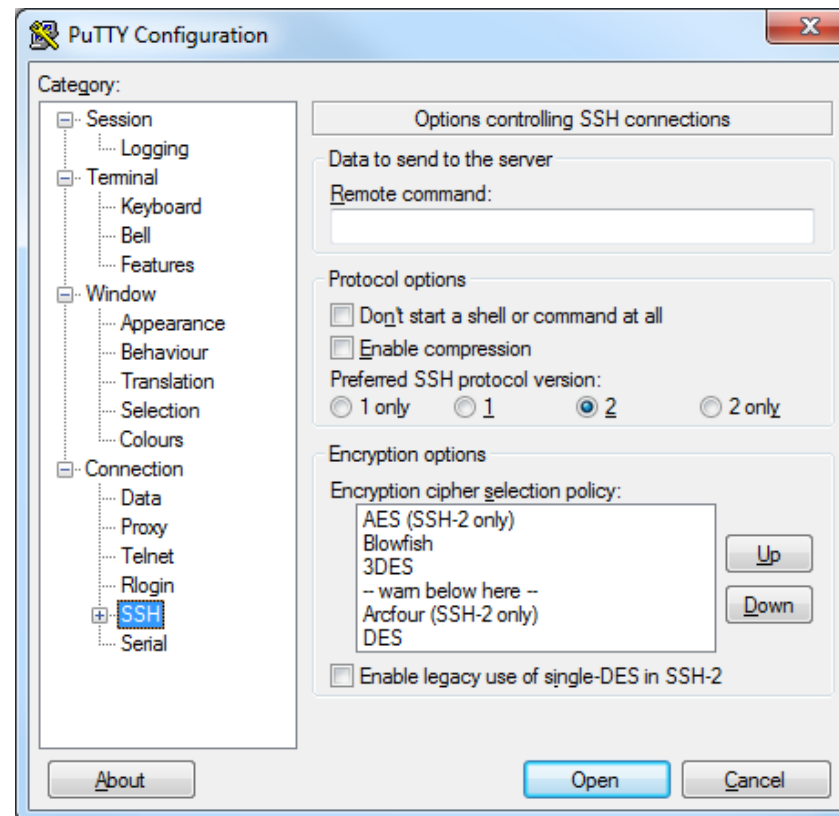


Funcții SSH – Integritate

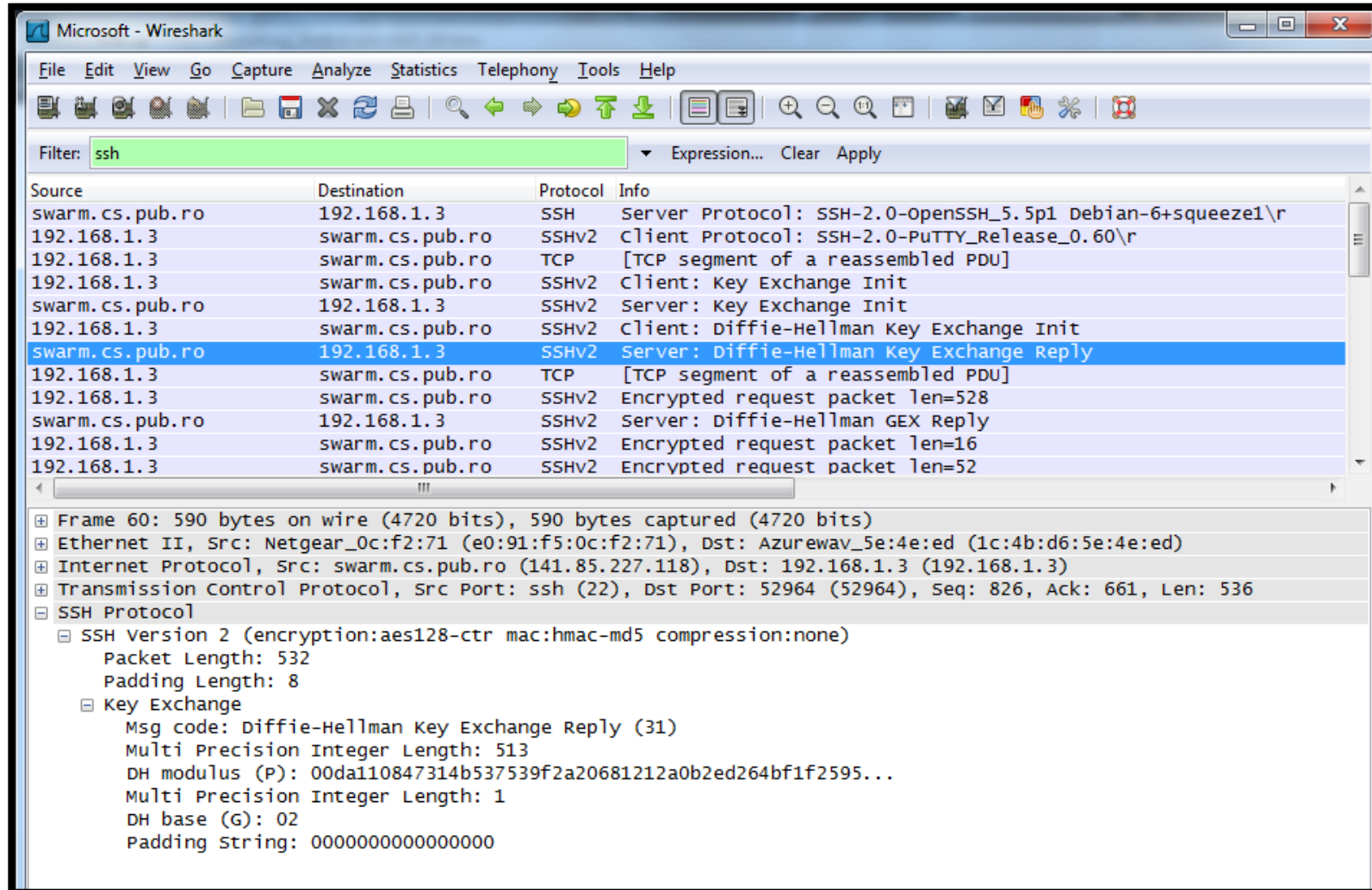


SSH – Exemplu client

- Putty este un client ssh pentru Windows, disponibil sub licența MIT



SSH – Captură trafic



Microsoft - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

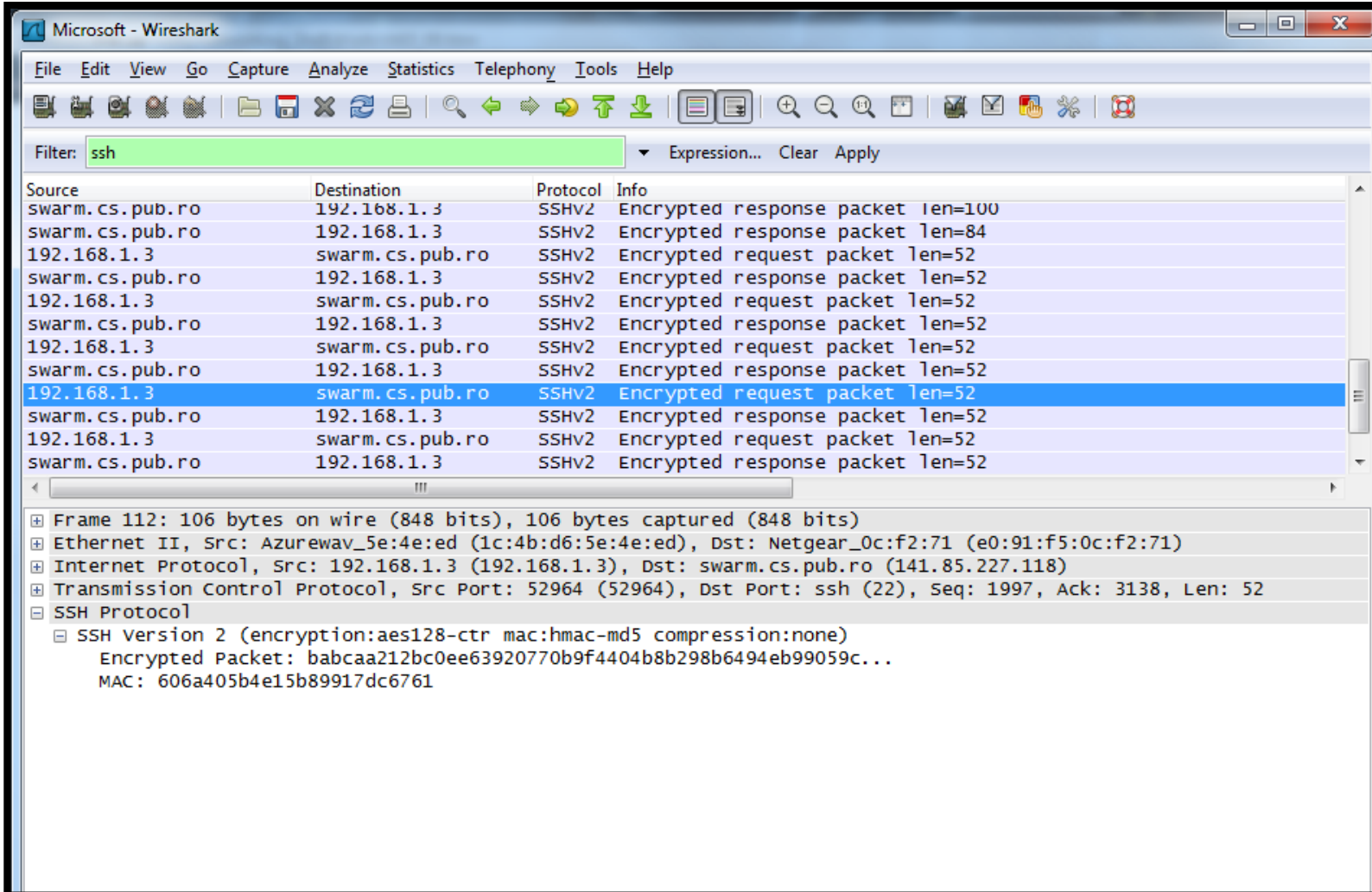
Filter: ssh

| Source | Destination | Protocol | Info |
|-----------------|-----------------|----------|--|
| swarm.cs.pub.ro | 192.168.1.3 | SSH | Server Protocol: SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze1\r |
| 192.168.1.3 | swarm.cs.pub.ro | SSHv2 | Client Protocol: SSH-2.0-PuTTY_Release_0.60\r |
| 192.168.1.3 | swarm.cs.pub.ro | TCP | [TCP segment of a reassembled PDU] |
| 192.168.1.3 | swarm.cs.pub.ro | SSHv2 | Client: Key Exchange Init |
| swarm.cs.pub.ro | 192.168.1.3 | SSHv2 | Server: Key Exchange Init |
| 192.168.1.3 | swarm.cs.pub.ro | SSHv2 | Client: Diffie-Hellman Key Exchange Init |
| swarm.cs.pub.ro | 192.168.1.3 | SSHv2 | Server: Diffie-Hellman Key Exchange Reply |
| 192.168.1.3 | swarm.cs.pub.ro | TCP | [TCP segment of a reassembled PDU] |
| 192.168.1.3 | swarm.cs.pub.ro | SSHv2 | Encrypted request packet len=528 |
| swarm.cs.pub.ro | 192.168.1.3 | SSHv2 | server: Diffie-Hellman GEX Reply |
| 192.168.1.3 | swarm.cs.pub.ro | SSHv2 | Encrypted request packet len=16 |
| 192.168.1.3 | swarm.cs.pub.ro | SSHv2 | Encrypted request packet len=52 |

Frame 60: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits)

- Ethernet II, Src: Netgear_0c:f2:71 (e0:91:f5:0c:f2:71), Dst: Azurewav_5e:4e:ed (1c:4b:d6:5e:4e:ed)
- Internet Protocol, Src: swarm.cs.pub.ro (141.85.227.118), Dst: 192.168.1.3 (192.168.1.3)
- Transmission Control Protocol, Src Port: ssh (22), Dst Port: 52964 (52964), Seq: 826, Ack: 661, Len: 536
- SSH Protocol
 - SSH Version 2 (encryption:aes128-ctr mac:hmac-md5 compression:none)
 - Packet Length: 532
 - Padding Length: 8
 - Key Exchange
 - Msg code: Diffie-Hellman Key Exchange Reply (31)
 - Multi Precision Integer Length: 513
 - DH modulus (P): 00da110847314b537539f2a20681212a0b2ed264bf1f2595...
 - Multi Precision Integer Length: 1
 - DH base (G): 02
 - Padding String: 0000000000000000

SSH – Captură trafic



Microsoft - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: ssh Expression... Clear Apply

| Source | Destination | Protocol | Info |
|-----------------|-----------------|----------|-----------------------------------|
| swarm.cs.pub.ro | 192.168.1.3 | SSHV2 | Encrypted response packet len=100 |
| swarm.cs.pub.ro | 192.168.1.3 | SSHV2 | Encrypted response packet len=84 |
| 192.168.1.3 | swarm.cs.pub.ro | SSHV2 | Encrypted request packet len=52 |
| swarm.cs.pub.ro | 192.168.1.3 | SSHV2 | Encrypted response packet len=52 |
| 192.168.1.3 | swarm.cs.pub.ro | SSHV2 | Encrypted request packet len=52 |
| swarm.cs.pub.ro | 192.168.1.3 | SSHV2 | Encrypted response packet len=52 |
| 192.168.1.3 | swarm.cs.pub.ro | SSHV2 | Encrypted request packet len=52 |
| swarm.cs.pub.ro | 192.168.1.3 | SSHV2 | Encrypted response packet len=52 |
| 192.168.1.3 | swarm.cs.pub.ro | SSHV2 | Encrypted request packet len=52 |
| swarm.cs.pub.ro | 192.168.1.3 | SSHV2 | Encrypted response packet len=52 |
| 192.168.1.3 | swarm.cs.pub.ro | SSHV2 | Encrypted request packet len=52 |
| swarm.cs.pub.ro | 192.168.1.3 | SSHV2 | Encrypted response packet len=52 |

Frame 112: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)

- Ethernet II, Src: Azurewav_5e:4e:ed (1c:4b:d6:5e:4e:ed), Dst: Netgear_0c:f2:71 (e0:91:f5:0c:f2:71)
- Internet Protocol, Src: 192.168.1.3 (192.168.1.3), Dst: swarm.cs.pub.ro (141.85.227.118)
- Transmission Control Protocol, Src Port: 52964 (52964), Dst Port: ssh (22), Seq: 1997, Ack: 3138, Len: 52
- SSH Protocol
 - SSH Version 2 (encryption:aes128-ctr mac:hmac-md5 compression:none)
 - Encrypted Packet: babcaa212bc0ee63920770b9f4404b8b298b6494eb99059c...
 - MAC: 606a405b4e15b89917dc6761