Laborator 8. Translatarea de adrese

Cunoștințe și abilități ce vor fi dobândite

- Cunoștințe generale legate de translatarea de adrese
- Translatarea de adrese pe Linux
- Translatarea directă (SNAT, MASQUERADE) și port forwarding (DNAT)
- Tunelare SSH

Cheat sheet

- Cheat Sheet

Pregătire infrastructură de laborator

- **Reminder**: avem nevoie de o masina virtuală a laboratorului. Vă rugăm urmăriți pagina aceasta pentru instrucțiuni, apoi reveniți.
- Schimbati utilizatorul curent ca `root` folosind comanda

```
student@host:~$ sudo su
```

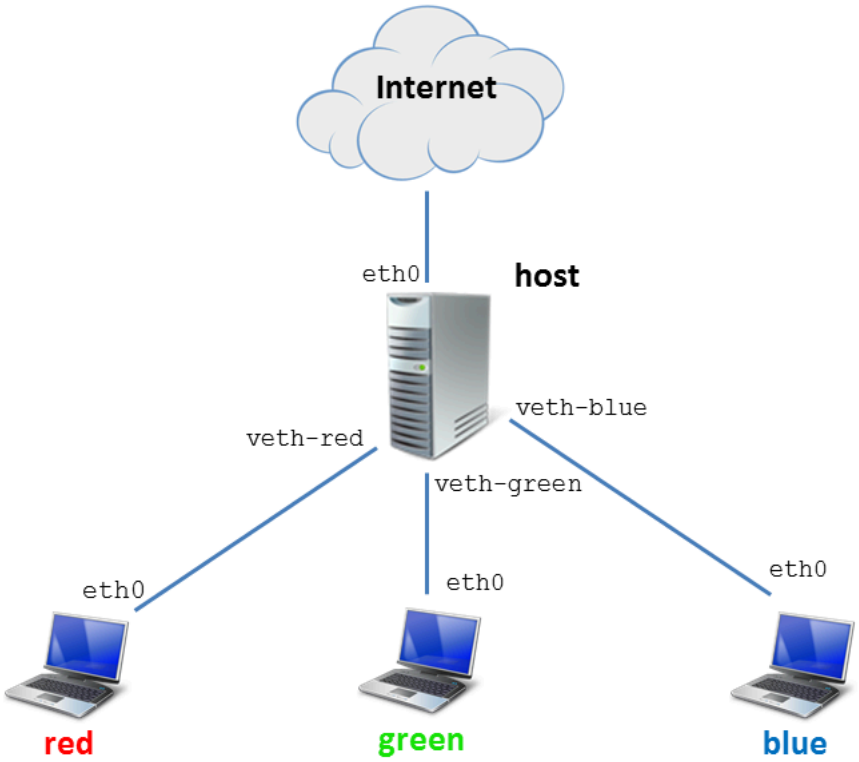
- Pentru a pregăti configurația de laborator, pe mașina virtuală (stația `host`) folosiți comenzile următoare din contul utilizatorului `root` de pe stația `host` (puteți da copy/paste la comenzi în terminal):

```
root@host:~# update_lab --force
root@host:~# start_lab lab-nat
```

- Deschideți trei noi tab-uri în terminal (folosiți combinația de taste `Ctrl+Shift+t`), și conectați-vă, din nou, la mașina virtuală folosind comanda `ssh` de mai sus.
  - De pe cele trei noi tab-uri, conectați-vă la cele trei containere (`red`, `green` și `blue`).
  - Pentru o conectare mai usoara puteti folosi aliasul `go` (ex. `go red`)

În mod implicit folosiți contul `root` pentru conectare pe toate stațiile. Aveți nevoie de drepturi privilegiate pentru configurare. Folosiți contul `student` doar unde vi se cere explicit.

Topologie



Navigare

Laboratorul 8

- 01. [10p] Configurare translatare de adrese (MASQUERADE)
- 02. [10p] Format de pachete la translatare
- 03. [10p] Format de pachete TCP la translatare
- 04. [10p] Configurare incorectă a translatării
- 05. [10p] Port forwarding
- 06. [10p] Extindere port forwarding

- 07. [10p] Format de pachete la port forwarding
- 08. [10p] Port forwarding pentru telnet
- 09. [10p] Configurare persistentă rutare și NAT
- 10. [10p] Tunel SSH invers
- 11. [BONUS - 10p] Translatare selectivă de adrese

Exerciții

Pentru a primi întregul punctaj va trebui ca la finalul laboratorului să ștergeți mașina virtuală pornită și să îi arătați asistentului listarea instanțelor din OpenStack.

În cadrul exercițiilor din laboratoarele de Linux vom folosi topologia de mai sus.

01. [10p] Configurare translatare de adrese (MASQUERADE)

Epuizarea adreselor IPv4 în cadrul rețelelor curente a condus la folosirea de adrese IP din clase private (de ex. `192.168.0.0/24`). Pe lângă comunicația dintre stațiile unei rețele, dorim și accesul la Internet al acestora. De aceea s-a introdus conceptul de translatare de adrese (NAT: *Network Address Translation*) prin care mai multe stații dispun de acces la Internet folosind aceeași adresă IP rutabilă: adresa gateway-ului. Activarea translatării de adrese (NAT) pe gateway conduce la înlocuirea perechii <adresă IP sursă, port sursă> (aparținând stației) cu perechea <adresa IP gateway, port disponibil>.

Configurarea NAT pe Linux se realizează tot prin intermediul comenzii `iptables`, la fel ca în cazul configurării firewall-ului. Dacă pentru configurarea firewall-ului foloseam tabela `filter` (tabela implicită a `iptables`), pentru configurarea translatării de adrese vom folosi tabela `nat`.

Astfel, pentru a activa NAT pe un server Linux executăm comanda

```
root@host:~# iptables -t nat -A POSTROUTING -j MASQUERADE
```

În comanda de mai sus:

- `-t` specifică tabela pe care se aplică regula, în cazul nostru tabela `nat`.
- `-A` înseamnă adăugarea unei reguli la sfârșitul listei de reguli.
- `POSTROUTING` se referă la momentul când va fi realizat procesul de translatare de adrese: după rutare.
  - În nomenclatura `iptables` acesta se numește și **lanț (chain)**.
  - Exemple de alte lanțuri: "INPUT", "OUTPUT", "FORWARD", "PREROUTING" [https://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO-3.html].
- `-j` este acțiunea ce va fi luată, iar în acest caz este `MASQUERADE` (acțiune simplă de translatare de adrese).

Pentru a verifica și valida regula, afișăm intrările din lanțul `POSTROUTING` din tabela `nat` folosind comanda

```
root@host:~# iptables -t nat -L POSTROUTING -n -v
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Vrem să verificăm configurarea corectă a NAT. Pentru acesta vom trimite de pe stația `red` un pachet către `8.8.8.8`. Pachetul va trece prin gateway (adică stația `host`) și va fi translatat. Pe stația `red` rulăm comanda

```
root@red:~# ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 192.168.1.2 icmp_seq=1 Destination Host Unreachable
From 192.168.1.2 icmp_seq=2 Destination Host Unreachable

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 999ms
```

Observăm că nu există conectivitate de la stația `red` către adresa IP `8.8.8.8`. Consultăm întreaga tabelă `nat`:

```
root@host:~# iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 2 packets, 168 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Observăm că pachetele ajung în lanțul `PREROUTING` (înainte de rutare), dar nu ajung în lanțul `POSTROUTING` (după rutare). Ne gândim că este posibil să fie o problemă cu rutarea pe gateway. Verificăm dacă rutarea este activată:

```
root@host:~# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

Într-adevăr, rutarea nu este activată. Pentru a activa rutarea pe stația `host` rulăm comanda

```
root@host:~# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Intrăm din nou pe stația `red` și folosim `ping` pentru a testa conectivitatea la adresa IP `8.8.8.8`:

```
root@red:~# ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=61 time=92.9 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=61 time=81.2 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 81.272/87.094/92.917/5.829 ms
```

Acum există conectivitate, lucru certificat și de prezența unor pachete în lista prelucrată pe lanțul `POSTROUTING`:

```
root@host:~# iptables -t nat -L POSTROUTING -n -v
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
```

```
 pkts bytes target     prot opt in     out     source               destination
    2   168 MASQUERADE  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

02. [10p] Format de pachete la translatare

Ne propunem să analizăm antetul IP al pachetelor ce sunt generate de stațiile `red`, `green` și `blue` și au ca destinație o rețea din Internet. Pentru acest lucru vom folosi utilitarul de captură `tcpdump`.

Pe stația `red` pornim comanda `ping` către `8.8.8.8.`:

```
root@red:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=127 time=42.0 ms
[...]
```

Atunci când rulăm comanda `tcpdump` secvența de urmat este:

1. Se rulează comanda `tcpdump` cu opțiunile aferente într-un terminal, activând astfel captura de pachete. Utilitarul `tcpdump` așteaptă acum transmiterea de pachete pe interfețele pe care ascultă.
2. Într-un alt terminal se rulează o comandă specifică unui client de rețea care generează trafic.
3. Se revine în terminalul în care rulează comanda `tcpdump` și se urmăresc pachetele capturate.
4. Când nu mai este nevoie de utilitarul `tcpdump` se întrerupe captura de pachete folosind combinația de taste `Ctrl+c`.

Pentru a urmări traficul, pe stația `host` rulăm comanda

```
root@host:~# tcpdump -n -i eth0 ip dst host 8.8.8.8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:59:20.976707 IP host > 8.8.8.8: ICMP echo request, id 625, seq 6, length 64
12:59:21.977708 IP host > 8.8.8.8: ICMP echo request, id 625, seq 7, length 64
```

După rularea comenzii de mai sus, probabil vi se va bloca terminalul. În mașina virtuală VMware apare o mini-fereastră care vă indică intrarea plăcii de rețea în modul *promiscuous*. Apăsați pe butonul `OK` al ferestrei pentru a debloca mașina virtuală și, astfel, terminalul.

Observăm că adresa IP sursă este `host.local` chiar dacă stația `red` este cea care execută comanda `ping` și generează pachetele de tip `ICMP echo request`.

Pentru a vedea pachetele așa cum sunt generate inițial, rulăm comandă `tcpdump` pe interfața `veth-red` în loc de `eth0`:

```
root@host:~# tcpdump -n -i veth-red ip dst host 8.8.8.8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth-red, link-type EN10MB (Ethernet), capture size 65535 bytes
13:01:12.557692 IP red > 8.8.8.8: ICMP echo request, id 626, seq 6, length 64
13:01:13.559726 IP red > 8.8.8.8: ICMP echo request, id 626, seq 7, length 64
```

Observăm că pe interfața `veth-red` adresa IP sursă este adresa stației `red`, adică așa cum este generat de la început pachetul. Motivul este reprezentat de faptul că pachetele ce intră pe intefața `veth-red` sunt capturate înainte de rutare, iar procesul NAT este aplicat **după rutare** (vezi 01. [15p] Observare porturi deschise pe o stație.

Pentru a vedea modul în care se translatează traficul capturăm traficul pe toate interfețele (cele de interes sunt `veth-red` și `eth0`)

```
root@host:~# tcpdump -n -i any ip dst host 8.8.8.8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 65535 bytes
10:23:07.632412 IP red > 8.8.8.8: ICMP echo request, id 707, seq 237, length 64
10:23:07.632430 IP host > 8.8.8.8: ICMP echo request, id 707, seq 237, length 64
10:23:08.633936 IP red > 8.8.8.8: ICMP echo request, id 707, seq 238, length 64
10:23:08.633954 IP host > 8.8.8.8: ICMP echo request, id 707, seq 238, length 64
```

În lista de mai sus observăm atât pachetele care sunt captuarate pe interfața `veth-red` (generate de stația `red`) cât și cele capturate pe interfața `eth0` (translatate de stația `host`).

Capturați și pachetele de reply, care au ca **sursă** adresa `8.8.8.8`. Folosiți șirul de argumente `ip src host 8.8.8.8` pentru `tcpdump`.

Repetați testele de mai sus pentru stația `green`.

Întrucât latența DNS poate fi semnificativă, recomandăm folosirea opțiunii `-n` a utilitarului `tcpdump`, opțiune care dezactivează rezolvarea DNS. În exemplele de mai sus, nu am folosit opțiunea `-n` a utilitarului `tcpdump` pentru a identifica mai ușor numele stațiilor implicate în conversație.

03. [10p] Format de pachete TCP la translatare

Dorim să urmărim conținutul pachetelor TCP înainte și după translatarea de adrese. În plus față de exercițiul de mai sus, vrem să urmărim și porturile. Pentru aceasta vom genera pachete TCP (HTTP) folosind `wget` de pe stația `red` și vom captura pachetele folosind `tcpdump` pe stația `host`.

Pe stația `host` capturați pe toate interfețele pachetele HTTP care au ca adresă destinație numele `cs.pub.ro`.

Va trebui să folosiți argumentele `ip dst host cs.pub.ro` și `tcp dst port 80` la `tcpdump`. Legați argumentele prin șirul `and`. Urmăriți și cheatsheet-ul de aici [https://cdn.comparitech.com/wp-content/uploads/2019/06/tcpdump-cheat-sheet-1.pdf].

Folosiți opțiunea `-n` a `tcpdump` ca să afișați în format rezultatele în format numeric (adresă IP și port). Vă va fi mai ușor să urmăriți pachetele și translatarea de adrese.

Pe stația `red` folosiți `wget` pentru a obține pagina de la `cs.pub.ro`.

Urmăriți translatarea adresei IP sursă și a portului sursă în cadrul capturii de pachete realizată cu `tcpdump` pe `host`.

O să existe o latență (2-3 secunde) la afișarea pachetelor capturate cu `tcpdump` datorată rezolvării DNS. Puteți folosi opțiunea `-n` a `tcpdump` pentru a dezactivarea rezolvarea DNS și pentru a elimina latența.

Observați în cadrul capturii că portul sursă, ales de stația `red` este același cu cel folosit în urma translatării de stația `host`. În general implementările de NAT vor păstra portul în urma translatării. În cazul rar în care acel port este ocupat pe stația `host` (posibil datorită unei alte translatării) se va aloca un alt port.

Pe stația `host` capturați pe toate interfețele pachetele HTTP care au ca adresă numele `cs.pub.ro`; nu contează dacă este sursă sau destinație, capturăm și pachetele de interogare și cele de răspuns; la fel, portul poate fi sursă sau destinație; folosiți `ip host` în loc de `ip dst host` și `tcp port` în loc de `tcp dst port`. Pe stația `red` folosiți, din nou, `wget` pentru a obține pagina de la `cs.pub.ro`. Urmăriți translatarea adresei IP sursă și a portului sursă pentru pachetele de interogare și translatarea adresei IP destinație și a portului destinație pentru pachetele de răspuns.

04. [10p] Configurare incorectă a translatării

Comanda folosită mai sus pentru translatare, la 01. [15p] Observare porturi deschise pe o stație, are neajunsuri pe care le vom identifica mai jos.

Pe stația `green` capturăm traficul pe interfața `green-eth0` folosind comanda `tcpdump`:

```
root@green:~# tcpdump -i green-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on green-eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
```

Pe stația `red` executăm comanda `ping` către stația `green`:

```
root@red:~# ping -c 2 green
PING green (192.168.2.2) 56(84) bytes of data.
64 bytes from green (192.168.2.2): icmp_req=1 ttl=63 time=0.155 ms
64 bytes from green (192.168.2.2): icmp_req=2 ttl=63 time=0.086 ms
[...]
```

În acest moment, pe stația `green` avem captura pachetelor în output-ul comenzii `tcpdump`:

```
11:18:54.535064 IP host > green: ICMP echo request, id 633, seq 17, length 64
11:18:54.535092 IP green > host: ICMP echo reply, id 633, seq 17, length 64
```

Observăm că adresa sursă a pachetelor de tip `ICMP echo request` este stația `host`, nu stația `red`. Acest lucru se întâmplă din cauză că atunci când am activat NAT nu am specificat pentru ce tip de trafic să aplice politica de translatare. Astfel stația `host` aplică politica de NAT pentru tot traficul care o tranzitează. Dorim să aplicăm politica doar pentru pachetele ce **ies** în Internet, adică cele ale căror **interfață de ieșire** este interfața `eth0`. Vom șterge vechea regulă de NAT și vom adăuga o regulă corectă.

Pentru ștergerea vechii reguli de NAT rulăm pe stația `host` comanda

```
root@host:~# iptables -t nat -D POSTROUTING -j MASQUERADE
```

Verificăm faptul că nu mai există comanda

```
root@host:~# iptables -t nat -L POSTROUTING -n -v
Chain POSTROUTING (policy ACCEPT 1 packets, 328 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Adăugăm din nou regula de NAT, dar de data aceasta vom specifica să aplice politica de NAT pachetelor ce ies prin interfața `eth0` a stației `host`:

```
root@host:~# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Verificați că regula a fost inserată:

```
root@host:~# iptables -t nat -L POSTROUTING -n -v
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
```

Observăm prezența interfației `eth0` în coloana `out`. Înainte apărea caracterul `*`, adică orice interfață.

Repetați testul de la începutul exercițiului și arătați că pachetele trimise de la `red` către `green` au adresele IP sursă/destinație nemodificate (nu mai fac trecerea prin NAT), în vreme ce pachetele ce ies în Internet sunt translatate.

05. [10p] Port forwarding

La exercițiile de până acum am folosit NAT pentru a permite stațiilor cu adrese private dintr-o rețea locală să acceseze Internetul. NAT poate fi folosit și pentru a permite unei stații din rețeaua locală să fie accesată din Internet, adică să se inițieze conexiuni din Internet. Acest proces poartă numele de *port forwarding*.

Dorim să putem accesa prin SSH, din Internet, stația `red`. Acest lucru nu este posibil în mod implicit întrucât stația `red` are adresă IP privată. Soluția este “deschiderea unui port” pe gateway (adică stația `host`) și redirectarea acestui port (*port forwarding*) către portul aferent serviciului SSH (portul TCP `22`) de pe stația `red`.

Vom aplica pe stația `host` o regulă prin care redirectăm traficul ce vine către `host` pe portul `10022` către portul `22` (SSH) al stației `red` (adresa IP `192.168.1.2`):

```
root@host:~# iptables -t nat -A PREROUTING -p tcp --dport 10022 -j DNAT --to-destination 192.168.1.2:22
```

Verificăm aplicarea regulii prin consultarea lanțului `PREROUTING` din tabela NAT:

```
root@host:~# iptables -t nat -L PREROUTING -n -v
Chain PREROUTING (policy ACCEPT 1 packets, 474 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:10022 to:192.168.1.2:22
```

Pentru a verifica rezultatul de mai sus, de pe `fep.grid.pub.ro` deschidem un nou terminal și ne conectăm prin SSH folosind portul `10022` la stația `host`:

```
mihai.carabas@fep:~$ ssh -l student $ADRESA_IP_MV -p 10022
root@192.168.138.129's password:
[...]
root@red:~#
```

unde `$ADRESA_IP_MV` este adresa IP a masinii virtuale obținută din dashboard-ul OpenStack.

> Daca rulați VMware local, `$ADRESA_IP_MV` este adresa interfeței `eth0` a stației `host`.

Observăm că în urma autentificării ne găsim pe stația `red`. *Port forwarding* a funcționat.

Folosim comanda de conectare SSH la stația `host` de pe stația `green`:

```
root@green:~# ssh -l student host -p 10022
[...]
root@red:~#
```

Observăm că și de pe stația `green` am accesat stația `red` prin *port forwarding*. Dorim să limităm *port forwarding* doar pentru conexiuni de la stațiile din Internet. Pentru aceasta trebuie să actualizăm regula de *port forwarding*.

Ștergeți regula de *port forwarding* și adăugați o regulă nouă care să permită **doar** stațiilor din Internet să acceseze prin *port forwarding* stația `red`.

Aplicați regula doar pentru pachetele ce **sosesc** pe interfața `eth0`. Adică `eth0` este interfața de **intrare**. Folosiți opțiunea `-i` a `iptables` pentru precizarea interfeței de **intrare**.

Realizați apoi din nou conectare SSH pe portul `10022` al stației `host` de pe `fep.grid.pub.ro` și de pe stația `green`. Dacă ați configurat corect, nu va merge conexiunea SSH de pe stația `green` dar va merge în continuare de pe `fep.grid.pub.ro`.

06. [10p] Extindere port forwarding

Dorim să accesăm din Internet/exterior stațiile `green` și `blue` prin SSH folosindu-ne de stația `host`. Vom folosi:

- portul `20022` de pe stația `host` pentru a face *port forwarding* pe portul 22 al stației `green`;
- portul `30022` de pe stația `host` pentru a face *port forwarding* pe portul 22 al stației `blue`.

Pe modelul exercițiului anterior, realizați configurarea necesară pentru această extindere de port forwarding. Verificați prin conectare de pe `fep.grid.pub.ro` (echivalentul Internet-ului/exteriorului) pe porturile respectiv `20022` și `30022` de pe stația `host`.

Pentru a afla adresele IP ale stațiilor `green` și `blue` rulați comanda de afișare a configurării interfeței `eth0` pe fiecare stație. Verificați, pentru siguranța, faptul că există conectivitate între ele și stația `host`.

07. [10p] Format de pachete la port forwarding

La exercițiul 03. [10p] Accesare serviciu web folosind wget am observat cum se modifică adresa IP sursă și portul sursă în cazul translatării de adrese. La port forwarding vom observa cum se modifică adresa IP destinație (de aici și numele DNAT: `Destination NAT`).

Vom captura traficul SSH inițiat din exterior către stația `red`, prin intermediul portului `10022` al stației `host`. Este vorba de traficul înainte de *port forwarding*. Pentru aceasta, pe stația `host` vom folosi comanda

```
root@host:~# tcpdump -n -i eth0 tcp dst port 10022 -w /home/student/portfwd_eth0_output.pcap
```

Pe un alt terminal, tot pe stația `host` capturăm traficul **după** *port forwarding*, pe interfața `veth-red` către portul SSH (22) al stației red. Pe stația `host` folosim comanda

```
root@host:~# tcpdump -n -i veth-red tcp dst port 22 -w /home/student/portfwd_veth-red_output.pcap
```

Pentru a genera trafic, de pe `fep.grid.pub.ro` realizăm o conexiune SSH la stația `host` pe portul `10022`, conexiune ce va fi redirectată la portul `22` al stației `red`:

```
mihai.carabas@fep:~$ ssh -l student $ADRESA_IP_VM -p 10022
root@host.local's password:
[...]
root@red:~#
```

Pentru a vizualiza capturile salvate în cele două fișiere copiați-le la voi local (pe calculatorul vostru) folosind `scp` și deschideți-le cu utilitarul `wireshark`.

```
$USER_FEP@fep:~$ scp -i ~/.ssh/openstack.key student@$ADRESA_IP_VM:portfwd_eth0_output.pcap ./   # Dacă fișierul pf_eth0_output.pcap se află în directorul /home/stude
$USER_FEP@fep:~$ scp -i ~/.ssh/openstack.key student@$ADRESA_IP_VM:portfwd_veth-red_output.pcap ./   # Dacă fișierul pf_veth-red_output.pcap se află în directorul /ho

student@lab-EG207-EG208:~$ scp $USER_FEP@fep.grid.pub.ro:portfwd_eth0_output.pcap ./  # Dacă fișierul portfwd_eth0_output.pcap se află în home-ul utilizatorului de pe
student@lab-EG207-EG208:~$ scp $USER_FEP@fep.grid.pub.ro:portfwd_veth-red_output.pcap ./  # Dacă fișierul portfwd_veth-red_output.pcap se află în home-ul utilizatorul
```

Exemplu instalare și pornire `Wireshark` (pe sisteme Debian):

```
student@mjolnir:~$ apt-get install wireshark
student@mjolnir:~$ wireshark portfwd_eth0_output.pcap &
student@mjolnir:~$ wireshark portfwd_veth-red_output.pcap &
```

În capturile realizate de comenzile de mai sus, observăm cum se translatează adresa IP și portul destinație din perechea `<$ADRESA_IP_VM, 10022>` în perechea `<192.168.1.2, 22>`.

08. [10p] Port forwarding pentru telnet

În cadrul exercițiilor anterioare am activat *port forwarding* pentru serviciul SSH. Dorim ca stațiile `red`, `green` și `blue` să fie accesibile și prin `telnet` din Internet astfel:

- `red` să poată fi accesat folosind portul 10023
- `green` să poată fi accesat folosind portul 20023
- `blue` să poată fi accesat folosind portul 30023

Realizați configurațiile necesare pentru a permite *port forwarding* pentru `telnet` așa cum este descris mai sus.

Serviciul `telnet` ascultă în mod implicit conexiuni pe portul `23`. Puteți observa acest lucru cu ajutorul uneia dintre comenzile de mai jos:

```
user@host:~$ grep -w 'telnet' /etc/services
telnet          23/tcp
tfido           60177/tcp                       # fidonet EMSI over telnet

user@host:~$ getent services telnet
telnet                23/tcp
```

Exercițiul este academic. Nu este indicată folosirea `telnet` cu atât mai puțin *port forwarding* peste `telnet` din rațiuni de securitate.

Testați de pe `fep.grid.pub.ro` folosind comanda `telnet`:

```
mihai.carabas@fep:~# telnet <adresa-ip-vm> <port-redirectat>
```

unde `<adresa-ip-vm>` este adresa IP a mașinii virtuale vizibilă de pe fep iar `<port-redirectat>` este portul care face redirectarea pe stația `host` (`10023` sau `20023` sau `30023`).

Există o latență de circa 10-15 secunde din momentul conectării până la apariția promptului `telnet`. Așteptați să apară promptul și apoi autentificați-vă cu username `student` și parolă `student` pentru a valida conectivitatea.

09. [10p] Configurare persistentă rutare și NAT

Configurările realizate până în acest moment sunt temporare. La repornirea stației `host` regulile de NAT se pierd. Dorim să configurăm în mod persistent rutarea și regulile NAT.

Pentru a configura în mod persistent rutarea pe stația `host`, editățm fișierul `/etc/sysctl.conf` și decomentăm linia

```
net.ipv4.ip_forward=1
```

Astfel la fiecarea pornire a sistemului, rutarea va fi activată.

Rutarea nu se va activa în mod automat după ce decomentați linia din fișier. Trebuie să resetați sistemul sau să rulați comanda

```
sysctl -p
```

pentru a aplica modificările din fișierul de configurare `/etc/sysctl.conf`.

Pentru a salva toate regulile `iptables` introduse în sistem se poate folosi comanda `iptables-save`. În mod implicit aceste reguli sunt afișate pe consolă. Noi le vom salva în `/etc/iptables-rules` folosind comanda:

```
root@host:~# iptables-save > /etc/iptables-rules
```

Regulile pot fi restaurate folosind comanda `iptables-restore < /etc/iptables-rules`. Așadar trebuie să configurăm sistemul să execute această comanda la pornire. Modul cel mai uzual pentru a face acest lucru este adăugarea acestei comenzi în fișierul de configurare al interfeței:

```
root@host:~# cat /etc/network/interfaces
[...]
# The primary network interface
auto eth0
iface eth0 inet dhcp
        up iptables-restore < /etc/iptables-rules
```

După ce ați făcut toate configurările, reporniți stația `host`:

```
Dashboard -> Drop Down in dreptul masinii virtuale -> Soft Reboot Instance
```

În urma repornirii stației `host`, va trebui sa rerulam scripturile de pregatire a infrastructurii pentru a recrea containerele. Vom vedea aplicată rutarea și regulile de NAT: (in cazul in care rutarea nu este activata, inseamna ca a fost suprascrisa de catre scripturile de pregatire a infrastructurii si va trebui s-o activati din nou)

```
root@host:~# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
root@host:~# iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 30 packets, 5694 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:10022 to:192.168.1.2:22
    0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:20022 to:192.168.2.2:22
    0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:30022 to:192.168.3.2:22
    0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:10023 to:192.168.1.2:23
    0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:20023 to:192.168.2.2:23
    0     0 DNAT       tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:30023 to:192.168.3.2:23

Chain INPUT (policy ACCEPT 2 packets, 534 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 16 packets, 958 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 15 packets, 918 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    40 MASQUERADE  all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
```

10. [10p] Tunel SSH invers

Sunt situații în care dorim ca o stație să fie accesibilă din exterior, dar este dificil să obținem un port în gateway (pentru *port forwarding*): fie din rațiuni administrative, fie din motive de securitate. Pentru a permite totuși o conexiune din exterior, se poate crea un tunel SSH invers. Tunel SSH invers înseamnă următorii pași:

- se creează o conexiune SSH de la o stație cu adresă IP privată la un server extern cu adresă IP publică,
- se deschide un port pe serverul cu adresă IP publică
- traficul către acel port este tunelat prin conexiunea SSH către stația cu adresă IP privată

Dorim să permitem conectarea la stația `red` din Internet pe SSH. Vom crea o conexiune SSH "directă" de la stația `red` la contul vostru de pe `fep.grid.pub.ro`. Prin această conexiune veți tunela invers traficul SSH către `red` permițând conexiunea din exterior.

Pentru început, de pe stația `red` creăm o conexiune SSH directă către contul de pe `fep.grid.pub.ro` (identic cu cel de pe `cs.curs.pub.ro`):

```
root@red:~# ssh -l $USERNAME fep.grid.pub.ro -R 100XY:localhost:22
[...]
$USERNAME@fep.grid.pub.ro's password:
Last login: Mon Sep  9 09:55:37 2013 from 141.85.225.214
[$USERNAME@fep-62-2 ~]$ ss -tln | grep 100XY
tcp        0      0 127.0.0.1:100XY             0.0.0.0:*                   LISTEN
tcp        0      0 ::1:100XY                   :::*                        LISTEN
```

unde $USERNAME este numele contului de pe `fep.grid.pub.ro`/`cs.curs.pub.ro`, iar XY este un identificator numeric unic al stației din laborator (dacă nu aveți pe monitorul stației din laborator un identificator discutați cu asistentul). Introduceți parola corespunzătoare contului vostru de pe `fep.grid.pub.ro`/`cs.curs.pub.ro`.

Observăm că se deschide portul `100XY` pe `fep.grid.pub.ro`. Acest port va fi folosit pentru tunelul invers care ne va duce pe stația `red`.

Argumentul `-R 100XY:localhost:22` transmis comenzii `ssh` are următoarea semnificație:

- `100XY` - portul de pe server pe care se vor asculta conexiuni care vor fi tunelate invers;
- `localhost` - stația către care vor fi trimise pachetele primite prin tunel (chiar stația locală);
- `22` - portul către care vor fi trimise pachetele primite prin tunel (adică portul SSH local).

După ce alți colegi rulează comenzile similare vă veți putea conecta pe stația `red` a acestora folosind portul deschis de ei; discutați cu alți colegi ca să vă spună ce port au deschis. Pentru aceasta, veți folosi stația fizică și vă veți conecta pe contul de pe `fep.grid.pub.ro`.

Pe `fep.grid.pub.ro` rulează un load balancer pentru SSH. Din acest motiv este posibil să intrați pe "celălalt" sistem. Verificați că numele de host din prompt corespunde cu cel pe care l-ați folosit inițial la crearea tunelului: adică ambele să fie `fep-62-2` sau `fep-62-1`. Pentru siguranță folosim și comanda `ss`:

```
[$USERNAME_COLEG@fep-62-2 ~]$ ss -tln | grep 100ZT
tcp        0      0 127.0.0.1:100ZT             0.0.0.0:*                   LISTEN
tcp        0      0 ::1:100ZT                   :::*                        LISTEN
```

În output-ul comenzii de mai sus `100ZT` este portul deschis de coleg.

După conectarea colegului la `fep`, realizăm o conexiune prin tunelul SSH pe portul `100ZT` deschis de coleg:

```
[$USERNAME_COLEG@fep-62-2 ~]$ ssh -l student localhost -p 100ZT
Warning: Permanently added '[localhost]:100ZT' (RSA) to the list of known hosts.
root@localhost's password:
[...]
root@red:~#
```

În cadrul conexiunii ni se cere parola utilizatorului `root` de pe stația `red` (adică parola `student`).

În final ați accesat stația `red` a colegului, o stația cu adresă IP privată inaccesibilă din exterior. Pentru aceasta ați folosit o altă stație (care ar putea fi din altă rețea) pentru a se conecta la `fep.grid.pub.ro` și de acolo la `red` prin intermediul tunelului invers creat inițial.

Creați un fișier pe stația `red` proaspăt accesată, iar colegul poate verifica faptul că, într-adevăr, ați accesat container-ul său prin tunelul SSH invers pe care l-a creat anterior.

11. [BONUS - 10p] Translatare selectivă de adrese

Pentru situația în care adresa IP exterioară a gateway-ului este adresă statică (nu dinamică) se recomandă folosirea acțiunii `SNAT` în loc de `MASQUERADE` la translatarea de adrese. Pe lângă aceasta, SNAT are avantajul precizării unui spațiu de adrese care să fie noile adrese (cele substituite) și a porturilor.

Informații despre folosirea `SNAT` și diferența dintre `SNAT` și `MASQUERADE` găsiți aici [http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO-6.html].

Ștergeți regula anterioară de tip `MASQUERADE` și folosiți reguli de tip `SNAT` pentru a face următoarele translatări:

- conexiunile TCP către Internet de la stația `red` să iasă prin porturi din spațiul `45000-50000`;
- conexiunile TCP către Internet de la stația `green` să iasă prin porturi din spațiul `50000-55000`;
- conexiunile TCP către Internet de la stația `blue` să iasă prin porturi din spațiul `55000-60000`;
- toate celelalte conexiuni să fie realizate în mod obișnuit, având ca adresă sursă adresa IP de exterior a stației `host` (această regulă trebuie să fie ultima).

Verificați translatarea corectă inițiind conexiuni HTTP (folosind `wget`) către `cs.pub.ro` de pe fiecare dintre cele trei stații de tip container (`red`, `green` și `blue`) și folosind `tcpdump` pe stația `host` pentru a captura traficul aferent.

Navigare