Laborator 7. Securizarea unui server

Cunoștințe și abilități ce vor fi dobândite

- Cunoștințe generale legate de securizarea rețelei
- Avantajele și dezavantajele formelor de transfer securizat (criptat) și nesecurizat (în clar)
- Folosirea utilitarului `iptables` pentru configurarea unui firewall pe Linux
- Folosirea protocolului SSH și a utilitarelor aferente pentru conectarea la distanță și transferul securizat de fișiere

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.
- 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):

```
student@host:~# update_lab --force
student@host:~# start_lab lab-iptables
```

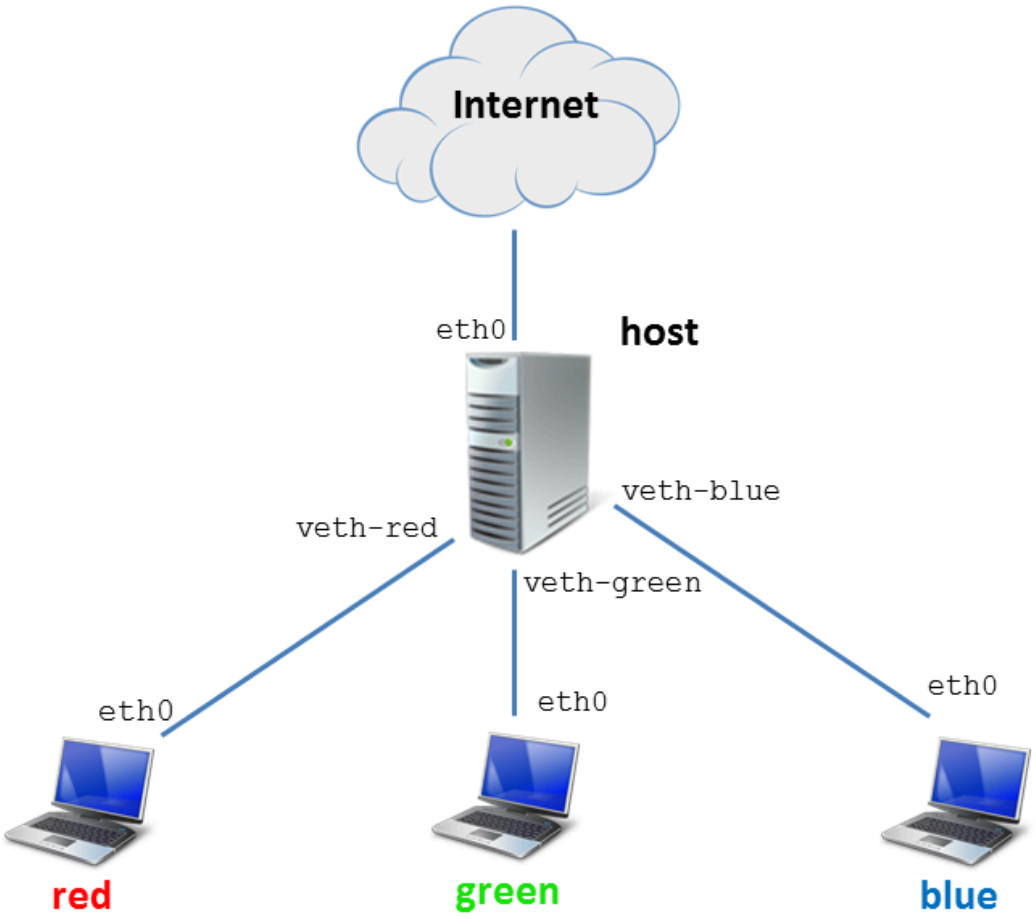
- Schimbati utilizatorul curent ca `root` folosind comanda

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

- 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 7

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.

1. [10p] Conectare SSH folosind cheie publică

Dorim să realizăm o sesiune securizată shell la distanță (*remote shell*).

Pentru exercițiile de conectare SSH, vom urma pașii:

1. Conectare folosind SSH cu chei. Totul este configurat și vedem că merge.
2. Copiere cheie pentru conectare SSH. Cheia există și vom face apoi conectare ca mai sus.
3. Crearea cheie, copiere și conectare SSH.

Urmăm pașii în acest fel pentru a urmări întâi comenzile de conectare, sintaxa SSH și rolul său și apoi să investigăm cum se întâmplă lucrurile în spate.

În cazul în care veți avea un cont pe un server și veți dori conectare SSH cu chei, veți urma doar pasul 3.

Din contul `student` de pe stația `red` conectați-vă la stația `host` prin SSH prin rularea comenzii

```
student@red:~$ ssh student@host
[...]
student@host:~$
```

Observați faptul că v-ați conectat direct, fără parolă, întrucât autentificarea s-a realizat folosind cheie publică. Folosiți combinația de taste `Ctrl+d` sau comanda `exit` pentru a închide sesiunea de shell la distanță.

Cheia publică folosită la conectare este disponibilă în fișierul `~/.ssh/id_rsa.pub`. Pentru vizualizarea cheii rulați, pe stația `red` din contul utilizatorului `student`, comanda

```
student@red:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1[...] student@red
```

Un alt format de comandă SSH de conectare este cel de mai jos. Pe stația `red` rulați comanda

```
student@red:~$ ssh -l student host
[...]
student@host:~$
```

La fel, închideți sesiunea de shell la distanță prin folosirea combinației de taste `Ctrl+d`.

La distanță, cheia publică este stocată, împreună cu alte chei publice folosite pentru conectare, în fișierul `~/.ssh/authorized_keys`. Rulați comanda de mai jos pentru a confirma prezența cheii publice la distanță

```
student@red:~$ ssh -l student host "cat ~/.ssh/authorized_keys"
ssh-rsa AAAAB3N[...] student@blue
ssh-rsa AAAAB3N[...] student@green
ssh-rsa AAAAB3N[...] student@red
ssh-rsa AAAAB3N[...] student@host
```

Se observă că există cheia `student@red` în fișierul de la distanță de pe stația `host`, deci se poate realiza autentificarea SSH pe bază de chei.

2. [10p] Generare cheie publică și autentificare

În contul utilizatorului `corina` de pe stația `blue` generați o pereche cheie publică/cheie privată SSH prin rularea comenzii

```
corina@blue:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/corina/.ssh/id_rsa):
Created directory '/home/corina/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/corina/.ssh/id_rsa.
Your public key has been saved in /home/corina/.ssh/id_rsa.pub.
[...]
```

Folosiți ENTER pentru a folosi căile implicite. Nu folosiți passphrase (adică apăsați ENTER când vi se solicită passphrase-ul). Cheile sunt generate respectiv, în fișierele, `.ssh/id_rsa` (cheia privată) și `.ssh/id_rsa.pub` (cheia publică).

Realizați operațiile necesare pentru a permite autentificarea pe bază de chei din contul utilizatorului `corina` de pe stația `blue` în contul utilizatorului `student` de pe stația `host`. După aceea, verificați faptul că autentificarea se face pe baza de chei.

Aceștia sunt pașii pe care îi veți urma pentru a configura conectare SSH pe bază de chei:

1. generare pereche de chei SSH (folosind comanda `ssh-keygen`)
2. copiere cheie publică în contul de la distanță (~~folosind comanda `ssh-copy-id`~~)
3. conectare la distanță (folosind comanda `ssh`)

Autentificarea cu parolă a fost dezactivată pe `student@host`!

Copiați manual cheia publică a corinei (`/home/corina/.ssh/id_rsa.pub`) în fișierul `/home/student/.ssh/authorized_keys` de pe host.

Atenție: nu ștergeți nimic din fișierul authorized_keys de pe host, altfel riscați să vă tăiați accesul la VM!

3. [10p] Download și upload de director folosind "scp"

Indicație: Pentru download-ul și upload-ul unui director folosiți opțiunea `-r` a comenzii `scp`.

Vom folosi două directoare:

- directorul `assignment/` din directorul home al utilizatorului `student` de pe stația `host`;
- directorul `solution/` din directorul home al utilizatorului `corina` de pe stația `blue`.

Din contul `corina` de pe stația `blue`, **descărcați** directorul `assignment/` **din** directorul home al utilizatorului `student` de pe stația `host`.

Din contul `corina` de pe stația `blue`, **uploadați** directorul `solution/` **în** directorul home al utilizatorului `student` de pe stația `host`.

4. [10p] Copiere fișiere cu diverse protocoale: durată și consum de resurse

Ne propunem să măsurăm timpul de copiere și consumul de resurse pentru un transfer realizat între două stații folosind, pe rând, transfer direct, FTP și SSH. În directorul home al utilizatorului `student` de pe stația `green` există un fișier `file-100M.dat`. Vom transfera acest fișier în directorul home al utilizatorului `student` de pe stația `host`. Vom folosi transfer direct, transfer FTP și transfer SSH. Pentru fiecare caz, stația `host` va rula server-ul, iar stația `green` clientul.

Dați copy-paste la comenzile de mai jos ca să vă asigurați că le scrieți corect. Pentru paste folosiți, în terminal, combinația de taste `Shift+Insert`.

Transfer direct

Pentru transfer direct vom porni un server simplu TCP pe stația `host` și un client pe stația `green`; vom folosi utilitarul `netcat`, prescurtat și `nc`. Rulați pe stația `host`, în contul utilizatorului `student`, comanda

```
student@host:~$ nc -l 12345 > file-100M-nc.dat
```

Comanda deschide un server TCP care ascultă conexiuni pe portul `12345`. Comanda rămâne activă în așteptarea unei conexiuni de la un client.

De pe stația `green`, din contul utilizatorului `student`, transferați fișierul și măsurați timpul de transfer și consumul de resurse prin rularea comenzii[1)]

```
student@green:~$ time cat file-100M.dat | nc -q0 host 12345
```

Comanda de mai sus trimite un fișier comenzii `nc`, iar comanda `nc` are rolul unui client TCP care se conectează la stația `host` pe portul `12345`.

Transfer FTP

Pentru transfer FTP, serverul de FTP este deja pornit pe stația `host`. De pe stația `green`, din contul utilizatorului `student`, transferați fișierul și măsurați timpul de transfer și consumul de resurse prin rularea comenzii

```
student@green:~$ time curl -T file-100M.dat -u student:student ftp://host/file-100M-ftp.dat
```

Transfer SSH

Pentru transfer SSH, serverul de SSH este deja pornit și configurat pe stația `host`. De pe stația `green`, în contul utilizatorului `student`, transferați fișierul și măsurați timpul de transfer și consumul de resurse prin rularea comenzii

```
student@green:~$ time scp file-100M.dat student@host:file-100M-scp.dat
```

Urmăriți timpii de rulare și consumul de memorie și de procesor pentru cele trei cazuri de mai sus. Observați valorile ridicate pentru SSH pentru timpul utilizator (*User time*) și procentul de procesor (*Percent of CPU*). Acesta valori ridicate se datorează componentei de criptare a SSH: această componentă asigură securizarea datelor cu un cost de reducere de performanță.

5. [10p] Trafic criptat și necriptat

Traficul generat de servicii se clasifică în **trafic criptat** și **trafic în clar**. Traficul **în clar (necriptat)** poate fi interpretat și înțeles dacă este capturat. Traficul criptat nu poate fi interpretat în absența cheii de criptare; doar transmițătorul și receptorul cunosc cheia pentru a putea comunica.

Ne propunem să analizăm, din punctul de vedere al criptării traficului, următoarele protocoale/servicii:

- telnet (port TCP 23)
- SSH (port TCP 22)
- FTP (port TCP 21)

Vom folosi `tcpdump`, un utilitar cu ajutorul căruia putem captura pachetele care trec printr-un anumit server, pentru a afișa datele transmise (parolă). Vom folosi topologia de laborator și ne vom conecta de la stația `red` la stația `green` prin intermediul stației `host`.

Pentru a porni procesul de captură, autentificați-vă ca `root` pe stația `host` și rulați comanda

```
root@host:~# tcpdump -vvv -A -i veth-green
```

Pentru comunicație prin telnet, rulați pe stația `red` comanda

```
root@red:~# telnet green
```

În urma rulării comenzii ați realizat o conexiune telnet de la stația `red` la stația `green` prin intermediul stației `host`. La prompt-ul generat de comandă folosiți username-ul `student` și parola `student`; după aceea rulați comanda `ls` și apoi comanda `exit` pentru a închide conexiunea. Observați pe stația `host` captura credențialelor (username și parolă) transmise prin telnet între stațiile `red` și `green` ⇒ traficul telnet între cele două stații a fost **trafic în clar** și a fost capturat pe stația `host`.

Pentru comunicație prin FTP, rulați pe stația `red` comanda

```
root@red:~# ftp green
```

În urma rulării comenzii ați realizat o conexiune FTP de la stația `red` la stația `green` prin intermediul stației `host`. La prompt-ul generat de comandă folosiți username-ul `student` și parola `student`; după aceea rulați comanda `ls` și apoi comanda `quit` pentru a închide conexiunea. Observați pe stația `host` captura credențialelor (username și parolă) transmise prin FTP între stațiile `red` și `green` ⇒ traficul FTP între cele două stații a fost **trafic în clar** și a fost capturat pe stația `host`.

Pentru comunicație prin SSH, rulați pe stația `red` comanda

```
root@red:~# ssh -l student green
```

În urma rulării comenzii ați realizat o conexiune SSH de la stația `red` la stația `green` prin intermediul stației `host`. În sesiunea de shell deschisă la distanță, rulați comanda `ls` și apoi comanda `exit` pentru a închide conexiunea. Observați că pe stația `host`, `tcpdump` nu afișează informații în clar, traficul SSH între cele două stații fiind **criptat** și transmis într-un format binar.

Pe stația `host`, pentru a opri comanda `tcpdump` folosiți combinația de taste `Ctrl+c`.

Traficul telnet și FTP este trafic necriptat (în clar), în timp ce traficul SSH este trafic criptat.

6. [10p] Blocare servicii necriptate

`iptables` este un utilitar Linux care oferă și rol de firewall software. `iptables` folosește suportul nucleului pentru a intercepta pachete de rețea în diverse puncte ale trecerii acestora prin nucleu și a efectua acțiuni asupra acestora. Astfel de acțiuni sunt:

- acceptarea pachetului (`ACCEPT`)
- respingerea pachetului (`REJECT`)
- aruncarea pachetului (`DROP`), similar cu respingerea dar nu se transmite nici o notificare de respingere către cel care a transmis pachetul inițial

O diagramă a drumului urmat de un pachet de rețea în nucleu este aici [https://commons.wikimedia.org/wiki/File:Diagrama_linux_netfilter_iptables.png].

Comanda `iptables` înseamnă lucrul cu reguli de filtrare de la nivelul nucleului. În mod obișnuit se va preciza:

- tipul de operație pe regulă (adăugare, ștergere, înlocuire, inserare)
- punctul din nucleul în care trebuie să se găsească pachetul pentru a se aplica regula
- regula în sine

De exemplu, comanda de mai jos are semnificația descrisă în continuare:

```
iptables -A FORWARD -d green -p tcp --dport telnet -j REJECT
```

- `-A`: se adaugă regulă (este vorba de *append*, se adaugă la finalul listei de reguli);
- `FORWARD`: regula se aplică pachetelor care vor fi rutate; alte variante sunt `INPUT` (pachetele primite direct de sistem) și `OUTPUT` (pachetele care pleacă de la sistem);
- `-d green`: sunt selectate pachetele care au ca destinație adresa stației `green`;
- `-p tcp`: pachetele selectate sunt pachete TCP;
- `--dport telnet`: portul TCP destinație este portul specific protocolului telnet (adică portul `23`, identificat din fișierul `/etc/services`)
- `-j REJECT`: pachetul este respins

În tabela de filtrare aferentă `iptables` vom avea, așadar, o listă de reguli care sunt parcurse secvențial. Partea `-A FORWARD` identifică lanțul de reguli, partea `-d green -p tcp --dport telnet` este partea de **match** (ce pachete fac match pe regulă), iar partea `-j REJECT` este partea de **acțiune** (ce face regula cu pachetul).

După cum ați observat la punctul anterior, traficul pentru protocoalele telnet și FTP este trafic în clar, necriptat, putându-se afla cu ușurință credențialele unui anumit cont și comenzile rulate.

Ne propunem să blocăm accesul de la stația `red` către stația `green` pentru aceste servicii, configurând ruterul dintre cele două stații, adică stația `host`. Practic vom configura pe stația `host` opțiuni de firewall cu ajutorul utilitarului `iptables`.

Autentificați-vă prin SSH ca `root` pe stația `host`. Pentru a bloca accesul la serviciul telnet (port 23) destinat stației `green`, rulați pe stația `host` comanda de mai jos. Comanda adaugă regula `iptables` corespunzătoare.

```
root@host:~# iptables -A FORWARD -d green -p tcp --dport telnet -j REJECT
```

Pentru a verifica adăugarea regulii de mai sus, rulați pe stația `host` comanda

```
root@host:~# iptables -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp  --  anywhere             green                tcp dpt:telnet reject-with icmp-port-unreachable
```

Pentru a afișa informații și despre pachetele prelucrate și interfețele folosite, rulați pe stația `host` comanda

```
root@host:~# iptables -L FORWARD -v
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     tcp  --  any    any     anywhere             green                tcp dpt:telnet reject-with icmp-port-unreachable
```

Pentru a afișa informații în format numeric (pentru nume de stații și nume de porturi), rulați pe stația `host` comanda

```
root@host:~# iptables -L FORWARD -v -n
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     tcp  --  *      *       0.0.0.0/0            192.168.2.2          tcp dpt:23 reject-with icmp-port-unreachable
```

De acum înainte recomandăm folosirea acestor opțiuni (`-v -n`) pentru listarea regulilor `iptables`.

Pentru a verifica blocarea traficului telnet către `green`, rulați pe stația `red` comanda

```
telnet green
```

Vă apare un mesaj de forma

```
Trying 192.168.2.2...
telnet: Unable to connect to remote host: Connection refused
```

semnificând faptul că se încearcă realizarea conexiunii dar conexiunea este respinsă

Pentru a vedea că regula de blocare a funcționat, rulați din nou pe stația `host` comanda

```
root@host:~# iptables -L FORWARD -v -n
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   120 REJECT     tcp  --  *      *       0.0.0.0/0            192.168.2.2          tcp dpt:23 reject-with icmp-port-unreachable
```

Observați, în output-ul comenzii, că există acum valori diferite de 0 în coloana `pkts` și `bytes`, semn că au fost pachete prelucrate de această regulă, deci blocate.

Pentru a verifica funcționarea în continuare a altor conexiuni (diferite de telnet) de la `red` la `green`, rulați pe stația `red` comenzile

```
ftp green
ssh -l student green
```

Dorim să blocăm și celălalt serviciu necriptat, FTP. Adăugați o regulă `iptables` similară pentru a bloca, pe stația `host`, traficul FTP destinat stației `green`. După adăugarea regulii folosiți `iptables -L FORWARD -n -v` pentru a valida adăugarea regulii.

Pentru această regulă puteți transmite argumentul `21` opțiunii `--dport` sau chiar numele `ftp`. Asocierea între port (număr) și protocol (nume) se găsește în fișierul `/etc/services`.

De pe stația `red` verificați blocarea traficului FTP către stația green folosind comanda

```
ftp green
```

7. [10p] Blocare SSH

Ne propunem ca stația `green` să nu fie acesibilă nici prin SSH. Pentru aceasta adăugați pe stația `host` o regulă `iptables` care va bloca traficul aferent serviciului SSH (portul 22).

Verificați adăugarea regulii `iptables` și apoi verificați de pe stația `red` blocarea traficului SSH către stația `green`.

8. [10p] Permitere trafic SSH

În acest moment, traficul SSH către stația `green` este blocat.

Dorim să permitem traficul SSH **de la** stația `red` **către** stația `green`. Adăugați o regulă corespunzătoare pe stația `host`.

După ce ați adăugat regula, încercați realizarea unei conexiuni SSH de la stația `red` la stația `green`. Observați că nu se realizează conexiunea.

Afișați lista de reguli `iptables` de pe stația `host`. De ce nu a reușit conexiunea? Țineți cont de ordinea regulilor afișate; sunt parcurse secvențial.

Pentru rezolvarea problemei ștergeți regula `iptables` introdusă anterior și **inserați** regula pe stația `host`. Pentru inserare folosiți opțiunea `-I` a comenzii `iptables`. Verificați că acum conexiunea SSH între `red` și `green` va fi realizată.

Pentru a șterge o regulă puteți folosi opțiunea `-D`.

Pentru a insera o regulă folosiți opțiunea `-I` urmată de numele lanțului (`INPUT`, `OUTPUT` sau `FORWARD`), urmată de indexul poziției unde doriți plasată regulă (1, 2, 3, …) și apoi urmată de specificarea regulii.

9. [10p] Ștergere reguli adăugate

Pentru a permite tot traficul către stația `green`, ștergeți pe stația `host` toate regulile `iptables` din lanțul `FORWARD`. Folosiți opțiunea `-F` (*flush*) a comenzii `iptables`. Practic revenim la situația inițială, fără reguli `iptables` pe stația `host`. Folosiți comanda `iptables -L FORWARD -n -v` pentru a valida ștergerea regulilor din lanțul `FORWARD`.

După ștergerea regulilor verificați funcționarea serviciilor telnet, FTP, SSH prin conectare de la stația `red` la stația `green`.

10. [10p] Capturi de trafic

Tcpdump este un utilitar din linia de comandă a Linux-ului, care se ocupă de captura și analiza pachetelor de rețea la nivel de interfață. Este deseori folosit pentru trobleshooting sau ca tool de securitate. Este versatil, oferă filtre și poate fi folosit într-o varietate de cazuri. Fiind un utilitar de linie de comandă, cel mai des se folosește în sistemele care nu au GUI, pentru a colecta date, care apoi pot fi mutate și vizualizate cu Wireshark [https://www.wireshark.org/].

Printre opțiunile de `tcpdump`, avem:

- `-i`: interfața pe care să asculte
- `-p`: portul destinație: filtrare după portul destinație al pachetelor
- `-v`: verbosity level
- `-w`: fișierul în care se salvează datele

Pentru a vedea grafic ce pachete ajung la mașina `red`, trebuie să capturați traficul generat către `red` într-un fișier și apoi trebuie copiat fișierul pe mașina fizică, pentru a-l analiza cu Wireshark.

Pașii care trebuie urmați sunt:

1. Porniți `tcpdump` pe interfața `veth-red` de pe `host` cu opțiunea de salvare a output-ului într-un fișier.
2. Generați trafic către `red` de pe oricare dintre celelalte mașini. Puteți folosi orice fel de trafic (ex. ping / ssh / telnet).
3. Folosiți `scp` pentru a copia fișierul de output de pe mașina `host` pe `fep.grid.pub.ro` și apoi pe mașina locală.
4. Deschideți fișierul cu Wireshark.

Ce fel de pachete ați analizat?

11. [BONUS - 10p] Blocare acces din Internet la rețeaua locală

Pentru simularea unei situații reale, vom considera stația `red` ca fiind o stație din rețeaua locală (LAN), iar stația `green` o stație din Internet, conectate între ele prin stația `host`, pe post de gateway.

Dorim să blocăm traficul TCP **inițiat** din Internet către rețeaua locală, adică traficul TCP inițiat **de la** stația `green` **către** stația `red`. Traficul inițiat de stația `red` către stația `green` precum și traficul de răspuns de la stația `green` către stația `red` va fi permis.

Ca exemplu practic, de pe stația `red` vor putea fi inițiate conexiuni SSH către stația `green` folosind comanda

```
ssh green
```

, dar de pe stația `green` **nu** vor putea fi inițiate conexiuni SSH către stația `red` folosind comanda

```
ssh red
```

Configurați `iptables` pe stația `host` pentru a realiza acest lucru.

Folosiți modulul `state` al `iptables`. Accesați pagina de manual `iptables` și căutați după `--state`. Pe distribuțiile mai noi accesați pagina de manual `iptables-extensions`. Pagina de manual este instalată și accesibilă pe stația `host`, **nu** pe stațiile `red`, `green` sau `blue`.

12. [BONUS - 10p] Transfer sincronizat de fișiere folosind "rsync" peste SSH

rsync [http://rsync.samba.org/] este un utilitar și protocol care permite transfer incremental. Este foarte potrivit pentru situații de backup, când se transferă doar actualizările sau pentru mentenanța unui mirror, la fel transferându-se de la serverul central doar noile date. Altfel de operații poartă numele de “sincronizare” (*syncing*).

Dorim să facem backup prin rsync al directorului `proiecte/` din directorul home al utilizatorului `ana` de pe stația `host` în contul `bogdan` de pe stația `blue`. Atunci când vom actualiza conținutul directorului `proiecte/`, procesul de backup va transfera doar modificările.

Folosiți `rsync` și protocolul SSH pentru a realiza acest lucru. Se dorește ca autentificarea să se realizeze pe bază de chei, fără parolă.

Navigare

- 2. [10p] Generare cheie publică și autentificare
- 3. [10p] Download și upload de director folosind "scp"
- 4. [10p] Copiere fișiere cu diverse protocoale: durată și consum de resurse
- 5. [10p] Trafic criptat și necriptat
- 6. [10p] Blocare servicii necriptate
- 7. [10p] Blocare SSH
- 8. [10p] Permitere trafic SSH
- 9. [10p] Ștergere reguli adăugate
- 10. [10p] Capturi de trafic
- 11. [BONUS - 10p] Blocare acces din Internet la rețeaua locală
- 12. [BONUS - 10p] Transfer sincronizat de fișiere folosind "rsync" peste SSH

[1)] În cazul unui transfer prin rețea folosind `netcat` nu se face verificare de integritate (*integrity check*). Există riscul (mic, dar există) ca fișierul să nu fie transferat corespunzător. De aceea e bine să verificați integritatea acestuia prin calcularea sumei de control (*checksum*) folosind, de exemplu, utilitarul `sha512sum`.