Xarxes. **Curs 2015-2016**

Pràctica 2: *Aplicació client/servidor per al desenvolupament d'una eina per a la configuració d'un firewall*

1 Objectiu

En aquesta pràctica veureu les comunicacions entre dues aplicacions situades a màquines diferents utilitzant `sockets TCP`, orientats a connexió. Per això, desenvolupareu una parella d'aplicacions client/servidor en C.

2 Enunciat

En aquesta pràctica implementarem dues aplicacions, un client i un servidor per configurar, de forma remota, les regles d'un *firewall* instal.lat en un router.

Concretament configurarem les regles que s'encarreguen de filtrar el tràfic de xarxa que passa a través d'un *router*, és l'anomenat tràfic de **`FORWARD`**. Noteu que aquest no és el tràfic dels paquets que tenen com a destí el *router*, ni dels paquetss que genera el *router*, es tracta dels paquets que passen a través d'ell.

Un firewall consisteix en un mecanisme, software o hardware o combinat, que permet prendre decisions sobre què fer amb el tràfic de xarxa que entra, surt o passa a través d'una màquina.

En el sistema operatiu *GNU-linux* s'utilitza l'eina **iptables** per tractar els datagrames IP que que es generen, travessen o es manipulen en el sistema operatiu. `Iptables` és doncs l'eina que ens permet configurar un *firewall*.

`Iptables` interactua directament amb el módul de `netfilter` del sistema operatiu. Aquest mòdul és el que gestiona directament tot el tràfic de xarxa de la màquina.

Com veureu en l'assignatura de Tecnologies Avançades d'Internet, l'any vinent, les regles del *firewall* s'agrupen en el que anomenem **cadenes**, les quals, a la seva vegada, s'agrupen en taules.

A continuació definirem com és una regla de filtratge del *firewall*.

Una regla de filtratge conté un conjunt de condicions i una acció.

Les accions més utilitzades a l'hora de filtrar paquets són la d'acceptar el paquet, `ACCEPT`, o bé el de descartar-lo, `DROP`.

Nosaltres donarem d'alta regles per descartar paquets. Així doncs l'acció de la regla no ens caldrà especificar-la ja que implícitament sempre serà un `DROP` del paquet.

Les regles de filtratge amb les que treballarem tindran, doncs, els següents camps:

source/destination Indiquem si filtrem per origen, `src`, o per destí, `dest`.

IP o bé NetID

IP IP per la que volem filtrar. Aquesta IP pot ser d'origen o de destí, en funció del que haguem especificat en el camp anterior.

NetID Identificador de xarxa. En aquest cas estarem especificant tot un rang de IP's. És a dir, estarem filtrant tot el tràfic d'una xarxa.

A efectes pràctics treballarem amb una adreça, ja sigui una IP o una NetID.

Netmask En cas que en el camp anterior haguem especificat una `NetID`, en aquest camp especificaríem la màscara de la xarxa. En cas que haguem especificat una IP, com a `Netmask` especificarem el valor `32`, especificant doncs, un `hostID`.

source/destination Indiquem si filtrem per port d'origen, `sport`, o per port de destí, `dport`. Aquest camp és opcional. No l'especificarem en cas que no haguem especificat un port.

port Port pel que filtrem. Aquest port pot ser d'origen o de destí en funció del que haguem especificat en el camp anterior. Aquest camp és opcional. Podem especificar una regla que no filtri per port. En aquest cas especificarem un 0 tant en el camp sport/dport com en el camp port.

Aquests podrien ser exemples de regles de filtratge:

- `src 158.109.79.0/27 dport 32`
  Amb aquesta regla filtrem tot el tràfic que vingui de la xarxa `158.109.79.0/27` i que tingui com a destí el port `32`.
- `dst 158.109.79.65/32 sport 12345`
  Amb aquesta regla filtrem tot el tràfic que tingui com a destí el host `158.109.79.65` i que tingui com a origen el port `12345`.

- `dst 8.8.8.8/32 0 0`
  Amb aquesta regla filtrem tot el tràfic que tingui com a destí el host `8.8.8.8`. Noteu que no especifiquem cap port per això posem un `0` en el camp `sport/dport` i un `0` en el camp `port`.

Així **des de l'aplicació client** podrem donar d'alta noves regles de filtratge, llistar les que ja s'han donat d'alta, modificar-les i eliminar-les.

L'aplicació servidora, **el servidor**, doncs, s'executarà directament en el *router*.

L'aplicació servidora centralitzarà les operacions que fem sobre les regles de filtrat del *router*.

A continuació us detallem la funcionalitat de l'aplicació.

3 Funcionalitat

L'aplicació client mostra a l'usuari, l'administrador de sistemes, un menú per poder gestionar les regles de filtrat del *firewall* d'un *router* d'una xarxa *ethernet*.

Després de cada execució d'una opció de menú el client mostra, de nou, el menú.

A través d'aquest menú l'usuari podrà fer les següents operacions:

Hello És una funcionalitat de test per veure si ens funciona bé la comunicació bidireccional entre les aplicacions client i servidor. El client envia un missatge de tipus `HELLO` i en rebre la resposta del servidor, mostra per pantalla un dels camps del missatge.

Llistar les regles de filtrat del firewall Mostra una llista numerada amb les regles de filtrat del *firewall* que s'han donat d'alta fins al moment.

Un possible resultat de l'execució d'aquesta opció de menú seria:

```
Regles de FORWARD:
1 src 158.109.79.0/27 dport 32
2 dst 158.109.79.65/32 sport 12345
3 dst 8.8.8.8
```

Els números que acompanyen a les regles: `1`, `2`, `3`, són els identificadors de cada una de les regles. Utilitzarem aquests identificadors per modificar la regla o bé eliminar-la.

Mostrarem el llistat de totes les regles que ja tenim donades d'alta per cada operació que realitzi l'usuari.

Donar d'alta una nova regla de filtrat El programa client li demanarà a l'usuari que especifiqui els diferents camps de la regla.

Un possible resultat de l'execució d'aquesta opció de menú seria:

```
Introdueix la regla seguint el format:
src|dst address Netmask [sport|dport] [port]

src 192.168.0.1 32 dport 80

Regles de FORWARD:
1 src 158.109.79.0/27 dport 32
2 dst 158.109.79.65/32 sport 12345
3 dst 8.8.8.8
4 src 192.168.0.1/32 dport 80

Opcions de menú...
```

Amb aquesta execució el client ha donat d'alta la regla: `src 192.168.0.1/32 dport 80`. Recordeu que el port és opcional. Podríem donar d'alta una regla on no especifiquem cap port.

El programa client ha enviat la petició d'alta al servidor i ha mostrat la llista de totes les regles donades d'alta en el servidor incloent-hi la nova, i de nou, mostra totes les opcions del menú.

Modificar una regla de filtrat Per modificar una regla cal especificar els camps de la regla, amb els nous valors i, a més, l'identificador de la regla.

Un possible resultat de l'execució d'aquesta opció de menú seria:

```
Indica l'identificador de la regla a modificar seguint el format:
rule_id src|dst address Netmask [sport|dport] [port]

3 src 192.168.132.4 32 0 0

La regla s'ha modificat amb èxit.

Regles de FORWARD:
1 src 158.109.79.36/27 dport 32
2 dst 158.109.79.65/32 sport 12345
3 src 192.168.132.4/32
4 src 192.168.0.1/32 dport 80
```

En cas de que la regla existeixi i s'hagi pogut modificar, el servidor enviarà un missatge de `OK`.

En cas de que la regla no existeixi el servidor enviarà un missatge d'error.

Eliminar una regla de filtrat Per eliminar una regla només cal especificar l'identificador de la regla que volem eliminar.

En cas que la regla existeixi i s'hagi pogut eliminar, el servidor enviarà un missatge de `OK`.

En cas que la regla no existeixi el servidor enviarà un missatge d'error.

Eliminar totes les regles de filtrat Amb aquesta opció de menú eliminarem totes les regles de filtrat que haguem donat d'alta fins el moment.

El client informarà a l'usuari si l'operació s'ha dut a terme amb èxit o no.

Sortir Quan l'usuari especifiqui aquesta opció de menú el client **tancarà la connexió amb el servidor** i acabarà el programa client.

En les següents seccions us detallem el funcionament del programa client i del programa servidor.

4 Client de configuració del *firewall*

L'administrador de sistemes executarà l'aplicació client per configurar el *firewall*. Se li mostrarà un menú amb totes les operacions disponibles.

A través de la línia de comandes, l'administrador especificarà una opció de menú.

Per llençar l'aplicació client la sintaxi serà la següent:

```
./fwClient -h host [-p port]
```

on

-h host És el host on s'està executant el servidor, p.e: `deic-dc23.uab.es` o bé `158.109.70.227`.

-p port És el port a on s'està executant el servidor. Aquest paràmetre és **opcional**, si no s'especifica pren un valor per defecte.

Si executem el client amb la crida:

```
# ./fwClient -h deic-dc15.uab.es -p 8311
```

el client ha de mostrar el menú amb les diferents operacions disponibles:

```
0. Hello
1. Llistar les regles filtrat
2. Afegir una regla de filtrat
3. Modificar una regla de filtrat
4. Eliminar una regla de filtrat
5. Eliminar totes les regles de filtrat.
6. Sortir

Opció?
1

Firewall FORWARD rules:
-----------------------
src 1.1.1.1/16
dst 2.2.2.2/32 dport 80
```

Un altre exemple:

```
0. Hello
1. Llistar les regles filtrat
2. Afegir una regla de filtrat
3. Modificar una regla de filtrat
4. Eliminar una regla de filtrat
5. Eliminar totes les regles de filtrat.
6. Sortir

Opció?
2

Introdueix la regla seguint el format:
src|dst address Netmask [sport|dport] [port]

dst 192.2.0.0/16 0 0
```

Un cop l'usuari selecciona una opció de menú, se li envia un missatge al servidor en funció de l'opció de menú seleccionada i, si cal,també se li envíen paràmetres. Llavors **s'espera una resposta del servidor**.

El client mostra per pantalla la resposta del servidor.

A continuació, torna a mostrar el menú per a que l'administrador pugui continuar configurant el *firewall*.

En la següent secció indicarem com funcionarà el programa servidor.

5 Servidor de configuració del *firewall*

Aquesta és la funcionalitat que ha d'implementar el vostre servidor:

1. Tenim un servidor **passiu** que espera, mitjançant un `socket TCP`, rebre connexions dels clients.
2. Quan el servidor rep una connexió, l'accepta i obté un **altre `socket TCP`** que utilitzarà per atendre aquesta petició.
3. El servidor crea en memòria una estructura de dades per emmagatzemar totes les regles de filtrat que s'aniran donant d'alta, modificant i eliminant.
4. A mesura que l'usuari va configurant el *firewall*, el servidor va aplicant els canvis en memòria.
5. Un cop el client enviï el missatge de `sortir de l'aplicació`, el servidor tancarà la connexió establerta amb aquest client.

El servidor s'invocarà de la següent manera:

```
./fwServer [-p port]
```

on

-p port Indica el port on s'està executant el servidor. Aquest paràmetre és opcional. Si no s'especifica pren un valor per defecte seguint les indicacions descrites més a vall.

El servidor ha d'utilitzar un port conegut per a que el client el pugui localitzar. Per aquest motiu tindrem un port per defecte, que calcularem segons la següent fórmula:

$8000 + (100 * Grup) + Subgrup$ (GrupA $\rightarrow$ 1, B $\rightarrow$ 2, C $\rightarrow$ 3,...).

Per exemple:

Si sóm el grup `x-c11`: grup C $\rightarrow$ 3, port= $8000 + 100 * 3 + 11 = 8311$.

Recordeu que el port a utilitzar es pot canviar en executar el servidor, tot utilitzant el paràmetre `-p`. En aquest cas, el servidor farà servir el port especificat en comptes del port per defecte. Llavors el client s'haurà d'invocar tot especificant, explícitament, el port on es troba el socket del servidor.

6 Com generar els executables: `fwClient` i `fwServer`

Per a generar els fitxers que es puguin executar: `fwClient` i `fwServer`, cal que compilem el codi i generem els executables.

Per compilar el codi utilitzarem l'eina `make`.

Aquesta eina utilitza un fitxer de configuració: `Makefile`, on hi han els paràmetres de configuració per a la compilació i generació dels executables, tant del client com del servidor.

Aquest fitxer ja el tindreu en el vostre compte de pràctiques, en el directori de la pràctica.

Per a generar els fitxers `fwServer` i `fwClient`, servidor i client respectivament, executarem la següent comanda:

```
make
```

7 Concurrència

Quan el servidor rep una connexió d'un client, passa a executar les comandes que el client li demana. Si mentrestant altres clients s'hi volen **connectar**, els **encua** fins que no s'hagi tancat la connexió que té establerta amb el client. Aquest tancament no es durà a terme fins que el client no li enviï el missatge de **sortir**. Tenim doncs un servidor **iteratiu**.

Imagineu-vos el següent context: El vostre servidor és el *firewall* d'una xarxa molt gran i per tant teniu moltes regles de filtrat per administrar. Per a fer aquesta feina teiu uns quants administradors de sistemes.

En aquesta situació, té sentit pensar que poden haver diferents administradors de sistema que es vulguin connectar alhora al servidor per a gestionar el *firewall*.

En aquest context, doncs, ens caldria tenir un **servidor concurrent** per a fer la tasca el més eficient possible.

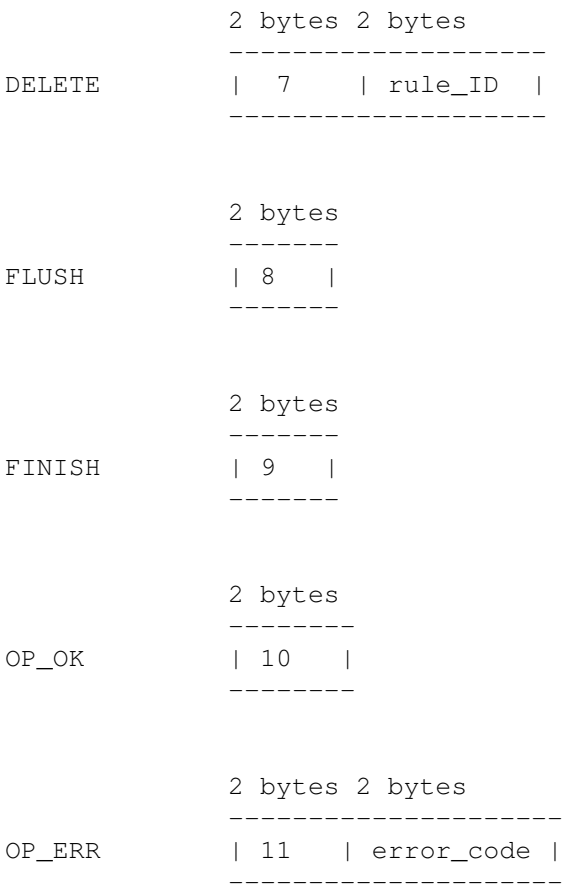
Modifiqueu doncs el servidor per a que sigui **concurrent** i pugui atendre simultàniament a diversos usuaris (administradors de sistemes) que s'hi connectin. Penseu que, en el nostre cas, seran diferents administradors que administraran el *firewall*.

Per a simplificar la pràctica no tractarem inconsistències generades per la modificació concurrent de les regles.

8 Format dels missatges

Entre l'aplicació client i l'aplicació servidora es manté un enviament bidireccional de missatges. És a dir, **implementen un protocol de comunicació**. A continuació us especifiquem quin ha de ser el format dels missatges. **El fet que no seguiu aquest format implicarà que les vostres aplicacions client/servidor no seran compatibles entre els altres grups de pràctiques i per tant hi haurà una penalització en la nota final**.

```
Type          Opcode
              2 bytes
              ----------
HELLO         | 1      |
              ----------

              2 bytes  11 bytes       1 byte
              -----------------------------
HELLO_RP      |  2     | Hello World | 0   |
              -----------------------------

              2 bytes
              ----------
LIST          |  3     |
              ----------


              2 bytes 2 bytes      n bytes
              ----------------------------------------
RULES         |  4    | num_rules  | [rule_espec] *    |
              ----------------------------------------
on
rule_espec:   4 bytes     2 bytes   2 bytes    2 bytes        2 bytes
              ------------------------------------------------------------
              | net_ID/IP | src/dst | net_mask | sport/dport  | port |
              ------------------------------------------------------------


              2 bytes 12 bytes
              --------------------------
ADD           |  5    | [rule_espec]   |
              --------------------------


              2 bytes 2 bytes   12 bytes
              ---------------------------------
CHANGE        |  6    | rule_ID | [rule_espec] |
              ---------------------------------
```



A continuació us mostrem una taula amb el significat de cada un dels missatges:

codi	Tipus	Missatge
1	HELLO	`Hello Request`: Petició de *Hello World*.
2	HELLO_RP	`Hello Response`: Resposta a la petició `HELLO`. **Camps**: * Missatge "*Hello World*" * Caracter de final de cadena: \0
3	LIST	`List Request`: Petició per llistar les regles de filtrat del *firewall*.
4	RULES	`List Response`: Resposta a la petició de llistat de les regles del *firewall*. **Camps**: * Número total de regles. Per a cada regla: * Flag d'origen, `src`: 0, o be de destí, `dst`: 1. * Identificador de xarxa o una IP. * Màscara de xarxa. * Flag d'origen, `src`: 0, o be de destí, `dst`: 1. En el cas que no s'hagi especificat cap port, aquest camp prendrà el valor 0. * Port, o bé el valor 0, en cas de que no calgui especificar cap port.
5	ADD	`Add rule request`: Missatge per donar d'alta una regla. **Camps**: * Regla de filtrat que volem afegir. El format d'aquestes regles està descrit en el missatge `RULES`.
6	CHANGE	Missatge per modificar una regla. **Camps**: * Identificador de la regla a modificar. * L'especificació de la nova regla.
7	DELETE	Missatge per eliminar una regla. **Camps**: * Identificador de la regla a eliminar.
8	FLUSH	Petició per eliminar totes les regles del *firewall*.
9	FINISH	Missatge per indicar que l'usuari ha tancat l'aplicació client.
10	OP_OK	Missatge per indicar que l'operació s'ha dut a terme amb èxit.
11	OP_ERR	Missatge per indicar que s'ha produït un error. **Camps**: * El codi d'error del missatge.

Codis d'error (err_code):

codi	Error
1	Regla de filtrat inexistent

9 Observacions

- Per enviar els missatges amb les operacions i les respostes entre el client i el servidor, utilitzarem un **array de bytes** (de caracters) de tamany fix. Per exemple de **1024 bytes**.

- Conversions que haureu de tenir en compte:

  De format xarxa a format host: `ntohs` Si rebeu un short per la xarxa, per treballar amb ell, l'haureu de convertir a format host amb la funció:

  ```
  uint16_t ntohs(uint16_t netshort);
  ```

  De format host a format xarxa: `htons` Si esteu treballant amb un short i el voleu enviar en el missatge per la xarxa caldrà que el convertiu a format xarxa amb la funció:

  ```
  uint16_t htons(uint16_t hostshort);
  ```

- A l'hora d'omplir o llegir els camps d'aquest array heu d'utilitzar les següents funcions i/o macros:

  Per omplir els camps del missatge –

  - `#define stshort(sval, addr) ( *((short *)(addr))=htons(sval) )`
    Amb aquesta macro **desarem un short** (dos bytes) corresponent al paràmetre `sval` en la posició de memòria apuntada pel paràmetre `addr`.
  - `int inet_aton(const char *cp, struct in_addr *inp)`
    Aquesta funció tracta una adreça internet expressada com una cadena de caracters en format *dotted quad* (paràmetre cp) i la converteix a long (format xarxa). Desa aquest long en el paràmetre inp.
    Us caldrà utilitzar alguna altra funció de C per desar aquest long en la posició que calgui del missatge que voleu enviar.

  Per llegir els camps del missatge –

  - `#define ldshort(addr) ( ntohs(*((short *)(addr)) ) )`
    Aquesta macro **retorna el short** convertit en format host corresponent als dos bytes en espai de memòria apuntat pel punter addr.
  - `char *inet_ntoa(struct in_addr in)`
    Aquesta funció converteix l'adreça internet especificada com a paràmetre 'in' (en format xarxa) en una cadena de caracters en format *dotted quad*.

- Per a la correcció de la pràctica és important que el codi font es trobi a la carpeta:

  ```
  $HOME/entregues/sockets/final
  ```

  ja que serà aquesta carpeta la que es copiarà el dia de l'entrega.

- **Entrega parcial**: La setmana del **18 d'abril**, al principi de la sessió, es farà l'entrega d'una part de la funcionalitat. Aquesta entrega està valorada amb **1 punt** i, és opcional. En cas que no feu l'entrega només podreu optar a un 9 de nota de pràctiques.

  Cal que deseu el codi del control en el directori:

  `$HOME/entregues/sockets/control`

  una hora abans de la sessió de pràctiques.

10 Proposta de planificació de la pràctica per setmanes

Abril	4	`1.` Implementar els missatges `HELLO` i `HELLORP`. El client mostra per pantalla el camp "Hello world". `2.` Implementar els missatges `LIST` i `FINISH`.
	11	`3.` Implementar els missatges `RULES` i `ADD`. `4.` Implementar el missatge `CHANGE`.
	18	**Entrega parcial** dels punts anteriors: `1, 2 i 3`. `5.` Modificar el codi per a que el servidor sigui concurrent. `6.` Implementar el missatge `DELETE`.
(No hi ha classe)	25	7. Implementar el missatge `FLUSH`.
Maig	**2**	**Entrega final de la pràctica.**

11 Condicions de lliurament

Llegiu-vos atentament l'enunciat abans del lliurament i verifiqueu que la vostra pràctica té implementat tot el que es demana.

El lliurament de la pràctica es farà la setmana del **02-05-2016**. Es copiarà la vostra carpeta

$HOME/entregues/sockets/final

una hora abans de la vostra sessió de pràctiques.