

## Lab 5: ACLs (Access Lists) con IOS

José M<sup>a</sup> Barceló Ordinas

### 1. ACLs (Access Lists)

Las listas de acceso (ACL) se usan para el filtrado de paquetes en función de ciertos parámetros como pueden ser las direcciones de red origen o destino, los puertos origen o destino, el tipo de protocolo (ip, icmp, tcp, udp, etc). Una de las aplicaciones donde se usan más las listas de acceso es en la seguridad de la red. Con las ACLs se puede bloquear el tráfico no deseado en una interfaz ya sea de salida o de entrada. Sin embargo notad que las ACLs no solo se usan en temas de seguridad, sino que también se usan para filtrar en general paquetes en aplicaciones tan variadas como pueden ser NAT (Network Address Translation), en BGP para filtrar rutas al crear políticas de encaminamiento, etc.

Existen ACLs para distintas pilas de protocolos: TCP/IP, IPX/SPX, Apple, etc. En este laboratorio nos centraremos en las ACLs aplicadas a seguridad en la red para la pila de protocolos TCP/IP. La diferencia con las otras pilas de protocolos está en el rango de ACLs que se pueden generar. Por ejemplo las ACLs entre la 1 y la 199 se usan en TCP/IP, mientras que las comprendidas entre la 800 y la 999 se usan para IPX/SPX, otros rangos se usan para DECnet (300-399), XNS (400-599), AppleTalk (600-699), etc.

Cuando creamos una lista de acceso y la aplicamos a una interfaz de entrada o de salida, estamos creando una secuencia de instrucciones que son chequeadas cada vez que un paquete entra o sale por esa interfaz. Es importante notar varias características de las ACLs.

Primero, que una ACL se aplica a la interfaz ya sea de entrada o de salida. Se pueden crear una ACL para la interfaz de salida y otra distinta para esa interfaz de entrada.

Lo segundo, las ACLs son secuencias de instrucciones que son chequeadas contra el paquete. El orden de las instrucciones es importante, ya que cuando una línea de la secuencia da cierta en el chequeo, se toma una acción y se sale de la ACL, es decir no se continua chequeando para comprobar que haya otra línea de la secuencia que también resulta cierta. Por consiguiente es muy importante diseñar la ACL en la secuencia que nos interese más.

Por ejemplo no es lo mismo estas dos líneas de un ACL:

- Si el paquete es icmp recházalo
- Si el paquete es ip acéptalo

que la secuencia:

- Si el paquete es ip acéptalo
- Si el paquete es icmp recházalo

Suponed que llegara un paquete ICMP. En el primer caso el paquete se rechazaría ya que la primera línea se cumple, el paquete es ICMP. En el segundo caso el paquete ICMP se aceptaría ya que la primera línea también se cumple, con lo cual ya no se comprobaría la segunda.

Otro aspecto importante es que no podemos insertar líneas en la secuencia. Si nos equivocamos al crearla o queremos insertar una línea a a hay que borrar TODA la ACL y volverla a crear.

Finalmente, también MUY IMPORTANTE, la última línea de una lista de acceso NUNCA aparece, es decir existe de forma explícita y siempre es DENIEGO TODO.

Dentro de las listas de acceso TCP/IP hay dos tipos de ACLs

- Listas de acceso IP estándar (1-99)
- Listas de acceso IP extendidas (100-199)

## Lab 5: ACLs (Access Lists) con IOS

### José M<sup>a</sup> Barceló Ordinas

#### 1.1. Wildcard mask

La wildcard mask es una máscara de 32 bits que indica que bits de la dirección IP se tienen que comprobar y cuales no. Si los bits de la máscara están a 0 entonces se comprueban, si están a 1 entonces no se comprueban.

Por ejemplo si queremos que un paquete que entra se compruebe si pertenece al host con dirección IP 145.34.5.6, queremos que se comprueben todos los bits de la dirección IP. Eso significa que la wildcard mask sería 0.0.0.0. En este caso se suele sustituir la tupla `@IP WildcardMask` por `host @IP`. Por ejemplo la tupla `145.34.5.6 0.0.0.0` se puede expresar como `host 145.34.5.6`.

Sino quisiéramos que no se comprobasen ninguno pondríamos una wildcard mask de 255.255.255.255. en este caso se suele sustituir la tupla `@IP WildcardMask` por `any`. Por ejemplo la tupla `145.34.5.6 255.255.255.255` se puede expresar como `any`.

También podemos expresar redes. Por ejemplo para comprobar todos los paquetes de hosts que vengan de la red 145.34.5.0/24. Eso significa que tenemos que comprobar todos los paquetes cuyos primeros 24 bits coincidan con los de la dirección de red. Luego la wildcard mask debería ser 0.0.0.255.

#### 1.2. ACLs estándar

Las ACLs estándar solo usan las direcciones origen para hacer la comprobación. Las listas de acceso estándar tienen números (`acl#`) comprendidos entre el 1 y el 99. El comando tiene el siguiente formato:

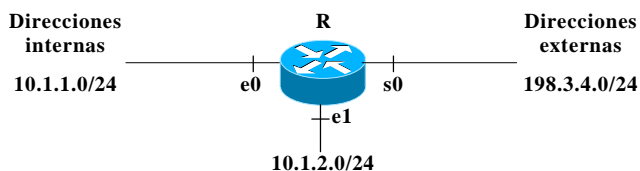
```
access-list acl# {deny|permit} {@IP_source WildcardMask | host @IP_source | any}
ip access-group acl# {in |out}
```

El primer comando, `access-list`, crea la lista de acceso con número `acl#` y con condición deniego o permito sobre la dirección IP origen especificada con la correspondiente wildcard mask. Recordad que la última línea de una ACL nunca aparece pero siempre es `access-list acl# deny any`.

El segundo comando, `access-group`, asigna la lista de acceso `acl#` sobre el protocolo IP sobre la interfaz de entrada o de salida donde se ejecuta dicho comando.

Ejemplo:

Queremos denegar en la interfaz `s0` de salida cualquier paquete IP que provenga de la red 10.1.1.0/24.



```
R# configure terminal
R(config)# access-list 1 deny 10.1.1.0 0.0.0.255
R(config)# access-list 1 permit any
R(config)# interface s0
R(config-if)# ip access-group 1 out
R(config-if)# exit
R# show access-lists
```

Primero creamos la lista de acceso con número igual a 1 y denegamos todo el tráfico que venga de la red 10.1.1.0/24. Como la última línea sería denegar todo lo demás (e.g.; la red 10.1.2.0/24), permitimos el resto

## Lab 5: ACLs (Access Lists) con IOS

### José M<sup>a</sup> Barceló Ordinas

de direcciones. Aplicamos esta ACL sobre la interfaz de salida s0 porque si lo hiciésemos sobre la e0 de entrada entonces bloquearíamos los ICMPs de la red 10.1.1.0/24 hacia la red 10.1.2.0/24.

#### 1.2.1. ACLs extendidas

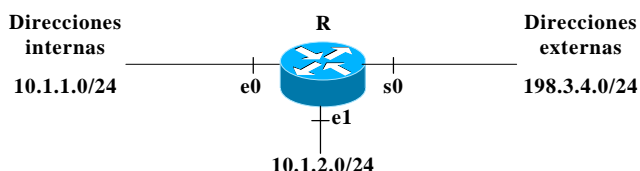
Las ACLs extendidas permiten usar tanto las direcciones origen como destino para hacer la comprobación. Además permiten especificar el protocolo sobre el que se quiere hacer la comprobación y en el caso de que sea TCP o UDP especificar el puerto destino. Las listas de acceso extendidas tienen números (*acl#*) comprendidos entre el 100 y el 199. El comando tiene el siguiente formato:

```
access-list acl# {deny|permit} protocol {@IP_source WildcardMask | host @IP_source  
| any} {@IP_dest WildcardMask | host @IP_dest | any} {operador port}  
ip access-group acl# {in |out}
```

El primer comando, *access-list*, crea la lista de acceso extendida con número *acl#* y con condición deniego o permiso sobre la dirección IP origen y/o destino especificadas con las correspondientes wildcard masks. *Protocol* puede ser *ip*, *icmp*, *tcp*, *udp*, etc. *Operador* puede ser *{lt,gt,eq,neq}* (less than, greater than, equal, non equal) y *port* es un puerto TCP o UDP. Recordad que la última línea de una ACL nunca aparece pero siempre es "*access-list acl# deny any any*".

Ejemplo:

Queremos denegar en la interfaz s0 de salida cualquier paquete ICMP que provenga de la red 10.1.1.0/24 y el acceso a cualquier puerto telnet (puerto 23) por parte de un host de esa red.



```
R# configure terminal
R(config)# access-list 101 deny icmp 10.1.1.0 0.0.0.255 any
R(config)# access-list 101 deny tcp 10.1.1.0 0.0.0.255 any eq 23
R(config)# access-list 101 permit any any
R(config)# interface s0
R(config-if)# ip access-group 101 out
R(config-if)# exit
R# show access-lists
```

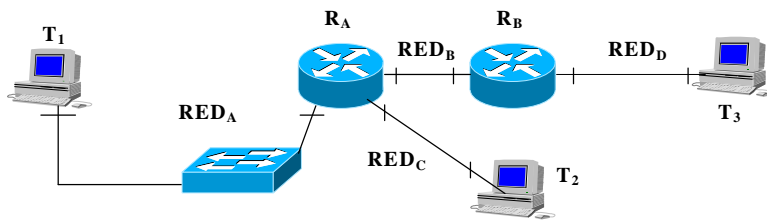
Primero creamos la lista de acceso extendida 101, denegando el acceso de paquetes ICMP, segundo otra línea denegando el acceso a cualquier host con puerto 23, finalmente permitimos cualquier otro tipo de tráfico. A continuación aplicamos la lista de acceso a la interfaz de salida s0.

## 2. Sesión de laboratorio

La topología del laboratorio es la siguiente: 2 terminales conectados al switch del aula del laboratorio formando la Red<sub>A</sub> y un terminal conectado al router formando la Red<sub>B</sub>.

## Lab 5: ACLs (Access Lists) con IOS

José M<sup>a</sup> Barceló Ordinas



La Red<sub>A</sub> es la 10.0.X.0/24, la Red<sub>B</sub> es la 192.5.Y.0/30, la Red<sub>C</sub> es la 192.6.X.0/24, donde X es el número de PC asignado al terminal T<sub>x</sub> e Y es igual a una de las X escogidas. la Red<sub>D</sub> es la 142.2.X.0/24 donde X es el número de PC asignado al terminal T<sub>3</sub>.

El router R<sub>A</sub> emula el router de salida de la red principal y el router R<sub>B</sub> emula el router del ISP.

- Configura todos los equipos para que haya conectividad entre ellos (direcciones IP y routing con RIPv2)

Queremos proteger la red interna de intrusos. Los servidores públicos estarán situados en la Red<sub>C</sub> (e.g.; el terminal T<sub>2</sub>) mientras que la Red<sub>A</sub> es privada.

- Diseña las listas de acceso necesarias para que terminales externos solo puedan acceder a servicios Web y FTP de la red pública, no puedan acceder a la zona privada y los terminales privados tengan acceso a Internet y a los servidores Web y FTP de la zona pública.
- Decide donde has de poner las listas de acceso y configura los routers. Puedes poner tantas listas de acceso como creas necesario, pero has de limitarlas al mínimo posible.