RCOMP - Redes de Computadores (Computer Networks)

2023/2024

Theoretical-practical lesson 06

- Static firewall.
- Cisco IOS standard and extended Access Control Lists.
- IP Named Access Control Lists.

Firewalls

When connected to the internet, systems are exposed to all types of traffic from all over the world. It's a serious security threat. Firewalls have the mission of analysing packets and block any traffic that might represent a threat.

Nevertheless, the golden rule on setting up a firewall is: **let the desired (required) traffic pass through and block all the rest**.

Each end-node (e.g., workstation or server) should run a local firewall, taking special attention to incoming traffic, however, by running a firewall on intermediate nodes, like routers, many nodes can be protected at the same time. Workstations are often out of control from network administrators, still, they can be protected if traffic is blocked before reaching them. We must, however, bear in mind this will not block traffic between local workstations.

Servers can also be protected by firewalls running on routers, in this case all servers should be placed in an isolated network apart from users and workstations. Because we assume all servers are trustable, we can even get rid of local firewalls on each server, that isolated network for servers is usually known as the Demilitarized zone (DMZ).

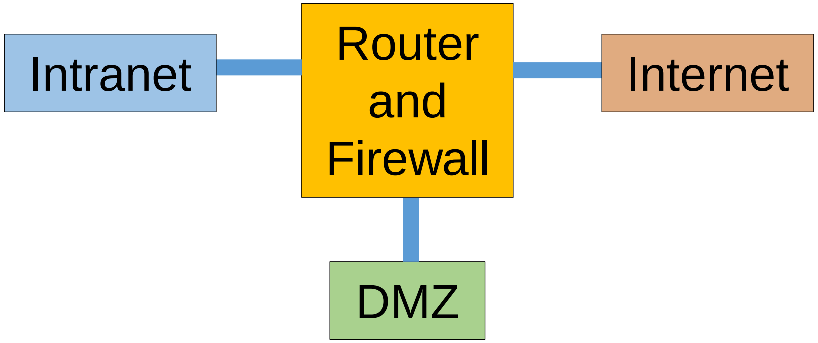
Internet, intranet and DMZ

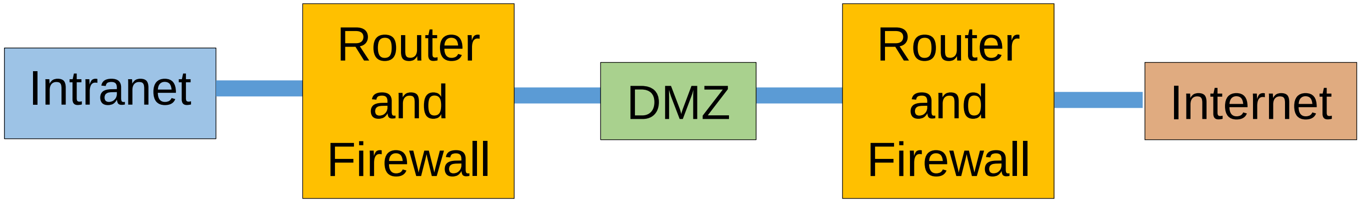
Under the security point of view, at least three network zones can be pointed out:

- The Internet – from where most threats are expected to come from
- The Intranets – local networks for internal users
- The Demilitarized zone (DMZ) – trusted local network, for servers only

Because they comprise different levels of security risk, these zones should be separated by firewalls.

Some typical layouts are:





Static firewalls

There are several different kinds of firewalls, they all have the same objective: let pass only desired traffic. **Static firewalls**, also called **stateless firewalls** or **packet filters** work by analysing each individual packet's layer three and layer four headers. This means the analysis will be on IP source and destination addresses (layer three), and also, ICMP message types, UDP port numbers, and TCP port numbers (layer four). They are called static and stateless because they don't change their behaviour over time.

Stateful or **dynamic firewalls**, on the other hand, track application layer network transactions, while an authorised transaction is ongoing, they allow the required traffic to pass through. They are called dynamic because rules change depending on what's happening.

Some types of attacks can only be prevented by dynamic firewalls, it's the case of DoS (denial-of-service) attacks. A DoS attack consists of flooding with requests a service that is supposed to be accessible to everyone.

A static firewall can't handle DoS attacks because it must allow access to that service for normal use. A dynamic firewall, however, can detect the attack and change the rules to block it.

Nevertheless, static firewalls can block many other attack types.

Access Control List (ACL)

In Cisco IOS, a static firewall may be deployed in a router by using access control lists. An ACL is a sequence of rules, each rule contains a **matching criteria** specification and **an action** to perform to every packet matching that criteria. Typical actions are **permit** or **deny**.

Each packet's characteristics are confronted sequentially with each rule in the ACL until a match is found, if so, the corresponding action is performed over the packet (**permit** or **deny**).

With Cisco IOS access control lists, if a packet doesn't match any rule in the ACL, then **the packet is ultimately denied**.

There are several types of ACLs in Cisco IOS, to start with, there are numbered ACLs (identified by numbers) and named ACLs (identified by names), beyond the way to identify, they are very similar. We'll start with numbered ACLs.

Each numbered ACL has a unique number that identifies it (within the device), the number also settles the ACL type.

1 up to 99	– for **standard** access lists
100 up to 199	– for **extended** access lists

Standard access lists

For numbered access lists, one ACL is known to be of the standard type if its identifier is between 1 and 99. These access lists support a single criterion for packet matching, and no others, it's **the packet's source address**.

The **access-list** command is used to create numbered access lists:

access-list number action source-address

number is the ACL identifier (1 to 99)

action is either **permit** or **deny**

source-address is an **address matching criterion** for packets source address

In fact, the **access-list** command adds (appends) one rule to the ACL, **a numbered ACL can't be edited**, commands must be typed in the exact order rules are to be in the ACL.

If a wrong rule is added (entered), there's no other option but to remove the entire ACL (by using the **no access-list number** command), and then type and enter all the commands again, one by one.

ACL addresses matching criterion

A matching criterion for IPv4 addresses can be specified in one of three ways:

host IPv4-ADDRESS matches the exact provided IPv4-ADDRESS

any matches any IPv4 address

And the general form:

IPv4-ADDRESS IPv4-wildcard matches the provided IPv4-ADDRESS, however, **bits that have value one at the IPv4-wildcard are ignored**.

Both IPv4-ADDRESS and IPv4-wildcard are dot-decimal representations of 32 bits. The IPv4-wildcard acts as a mask of bits, for each bit with value one at the IPv4-wildcard, the same bit at the IPv4-ADDRESS is ignored for matching criterion.

Consequently:

host IPv4-ADDRESS is the same as **IPv4-ADDRESS 0.0.0.0**

any is the same as **0.0.0.0 255.255.255.255**

IPv4 wildcards and network masks

To create a matching criterion encompassing an entire IPv4 network we can use the IPv4 network address as IPv4-ADDRESS and the inverted network mask (reversed bits) as IPv4-wildcard.

Examples:

To match any address of network **192.168.50.0/24** we can use **`192.168.50.0 0.0.0.255`**

(24 bits prefix-length means the network mask is 255.255.255.0)

To match all addresses of network **10.0.0.0/8** we can use **`10.0.0.0 0.255.255.255`**

(8 bits prefix-length means the network mask is 255.0.0.0)

To match all addresses of network **170.2.64.0/23** we can use **`170.2.64.0 0.0.1.255`**

(23 bits prefix-length means the network mask is 255.255.254.0)

As it comes clear in the last example, to fully understand which addresses are matched, a binary analysis may be required.

IPv4 wildcards

As seen, wildcards can be used as network masks equivalents, yet wildcards are much more powerful because unlike with network masks, bits one and zero can be alternated.

Examples:

The criterion **`192.4.50.0 0.1.0.255`** matches networks **192.4.50.0/24** and **192.5.50.0/24**

The criterion **`170.5.7.10 0.0.0.64`** matches addresses **170.5.7.10** and **170.5.7.74**

The criterion **`10.1.0.5 0.32.0.64`** matches **10.1.0.5**, **10.33.0.5**, **10.1.0.69**, and **10.33.0.69**

Looking in more detail into this last example:

- because bit with value 32 on the second octet is one, it's ignored for matching purpose, therefore, addresses with value 1 on that octet match the address, but if the value is 33 they also match.
- because bit with value 64 on the fourth octet is one, it's ignored for matching purpose, likewise, addresses with value 5 on that octet match the address, but if the value is 69 they also match.

A rule may be used to check our solution: **the number of matched addresses is always the power of two of the number of one bits in the wildcard**. We can also conclude no wildcard can match an odd number of addresses.

Enforcing access lists

Whatever type of access list we are talking about, just creating it has no immediate effect, it must be applied to a specific network interface to filter the incoming or outgoing traffic on that specific interface.

Until an ACL is applied to an interface, all traffic is allowed on that interface, once an ACL is applied, only traffic that matches a **permit** rule on the ACL will be allowed.

An existing ACL is associated to an interface by using the **ip access-group** command, within the interface's configuration:

```
ip access-group ACCESS-LIST-ID in|out
```

Where ACCESS-LIST-ID is the ACL identifier (either a number or a name).

The ACL is either applied to incoming traffic (in) or outgoing traffic (out). Incoming traffic stands for traffic being received from the network on this interface. Outgoing traffic stands for traffic being emitted to the network from this interface. Each traffic direction is independent of the other, for instance, enforcing an ACL on incoming has no effect on outgoing traffic of the interface. Two access lists can't be applied to the same interface in the same direction, the second application removes the first.

Anti spoofing with standard access lists

Standard access lists only analyse packets' source address, nevertheless, that's enough to prevent IP spoofing attacks.

IP spoofing stands for forging the source address of packets, they may serve several illicit purposes, including:

- Misleading static firewalls that allow access to some services only if traffic comes from some allowed source addresses.
- Misleading dynamic firewalls on DoS attacks by making them think the requests flooding the network are coming from different sources.

The network administrator is obliged to block two types of spoofing:

External spoofing: this applies to incoming traffic to a local network, for the sake of the local network's security, any incoming packet with a source address belonging to the local network is spoofing and should be blocked.

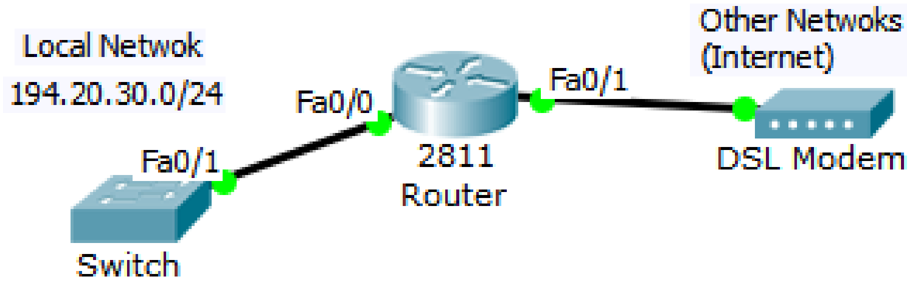
Internal spoofing: this applies to outgoing traffic from a local network, for the sake of other networks' security, any outgoing packet with a source address that doesn't belong to the local network, is spoofing and should be blocked.

Anti spoofing - example

Imagine the following scenario.

The local network is 194.20.30.0/24

We are to block external spoofing coming from the internet and internal spoofing coming from the local network.

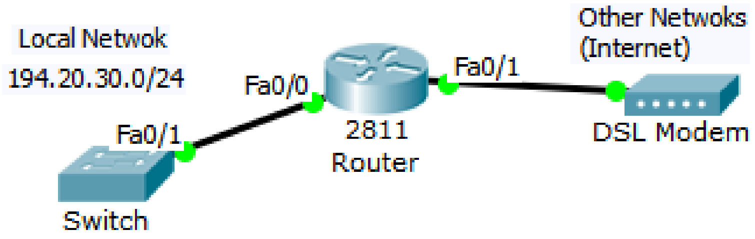


The first concern is: in which interfaces the access lists should be enforced. We could block external spoofing on Fa0/1 incoming traffic, or on Fa0/0 outgoing traffic. We could block internal spoofing on Fa0/0 incoming traffic, or on Fa0/1 outgoing traffic.

The general rule is: if there's no specific advantage in doing otherwise, we should always enforce access lists as close to the traffic source as possible, so usually over incoming traffic.

Following this principle, in this case, we will block external spoofing on Fa0/1 incoming traffic and block internal spoofing on Fa0/0 incoming traffic.

Anti spoofing - example



So, we will define one access list for each spoofing type:

```
no access-list 10
access-list 10 deny 194.20.30.0 0.0.0.255
access-list 10 permit any

no access-list 20
access-list 20 permit 194.20.30.0 0.0.0.255

interface Fa0/0
ip access-group 20 in

interface Fa0/1
ip access-group 10 in
```

Access list 10 says any traffic with source address belonging to the local network should be blocked and all others should pass.

Access list 20 says only traffic with source address belonging to the local network is allowed.

Access list 20 is applied to Fa0/0 incoming traffic (block internal spoofing) and access list 10 is applied to Fa0/1 incoming traffic (block external spoofing).

Remember, at the end of every access list there's an invisible, although implicit, rule **deny any**.

Before starting the definition of an ACL with **access-list** commands, it's wise to make sure it's empty by issuing the command **no access-list**.

Extended access lists

Numbered extended access lists work the same way as numbered standard access lists, however, they are identified by a number between 100 and 199.

While the packet's source IP address is the only matching criterion on standard access lists, extended access lists also require a protocol criterion and a destination address criterion:

access-list number action protocol source-address destination-address

number is the ACL identifier (100 to 199)

action is either **permit** or **deny**

protocol is layer three or layer four protocol name identifier

source-address is a matching criterion for packets' source address

destination-address is a matching criterion for packets' destination address

The rule only matches a packet if all criteria are meet at the same time.

The protocol name can be, among others: **ip**, **icmp**, **udp** or **tcp**.

Because all these protocols are transported by IP, specifying **ip** as the matching protocol will match any packet's protocol.

ICMP protocol extended access lists

Depending on the matching criterion for the protocol, additional matching criteria are available. For the ICMP protocol:

```
access-list number action icmp source-address destination-address [icmp-type [icmp-code] | icmp-message]
```

We can set criteria either through decimal values for the ICMP message type and code or through a match criterion for the name of the ICMP message.

Some frequently used ICMP message names (**icmp-message**) are **echo** to match ICMP echo requests, and **echo-reply** to match ICMP echo replies.

Specifying **echo** is equivalent to specify **8 0**, because the ICMP echo request is a type 8 message with code 0.

Specifying **echo-reply** is equivalent to specify **0 0**, because the ICMP echo reply is a type 0 message with code 0.

If no ICMP specific criteria are used, then the rule matches any ICMP message.

UDP protocol extended access lists

If the matching protocol is UDP, then we can also define matching criteria based on source and destination UDP port numbers:

access-list number action **udp** source-address **[operator port]** destination-address **[operator port]**

The criterion after source-address is for matching the source port number and the one after destination-address is for matching the destination port number. You may specify only one criterion, both, or none. If no port number is specified it will match any port number.

The **port** argument can either be a port decimal number or a name representing the standard service for that port number.

The **operator** may be one of:

lt	- less than
gt	- greater than
eq	- equal to
neq	- not equal to
range	- within the inclusive range

(in this last case, two space separated port numbers or service names are to be used)

TCP protocol extended access lists

If the matching protocol is TCP we can, as well, define matching criteria based on source and destination port numbers (now for TCP port numbers):

access-list number action **tcp** source-address **[operator port]** destination-address **[operator port] [established]**

Criteria specification for ports is the same as before with UDP. Additionally, one new criterion can be used specifically, and only, for TCP.

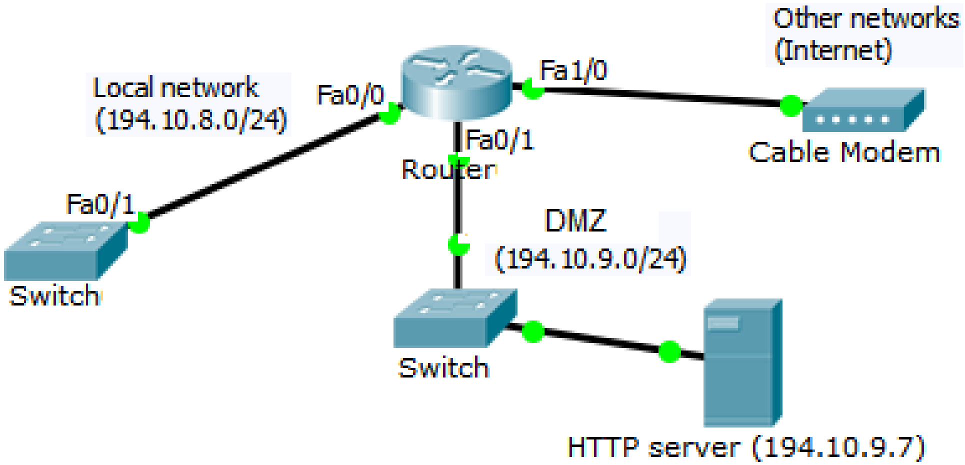
The **established** criterion matches TCP segments with either the ACK bit or the RST bit on. This excludes the initial SYN request segment sent with the purpose of establishing a new TCP connection, therefore, **it will match only TCP traffic regarding already established TCP connections**.

Although we can specify port numbers matching criteria for both source ports and destination ports, they are most useful for destination ports. Servers provide network services in standard fixed number ports, if we want to allow clients access to specific servers, we can use the destination port criterion. Clients on the other hand use dynamically assigned port numbers.

Port numbers can be replaced by the service name, for instance the name **www** represent number 80 and the name **domain** represents number 53.

Extended access lists application example

We now have this scenario:



We still want to block spoofing, but now there's a HTTP server on a DMZ. The DMZ is trusted, so no spoofing will be coming from there.

The DMZ must be protected, access to the HTTP server is the only to be allowed.

```
no access-list 100
access-list 100 permit tcp 194.10.8.0 0.0.0.255 host 194.10.9.7 eq 80
access-list 100 deny ip any 194.10.9.0 0.0.0.255
access-list 100 permit ip 194.10.8.0 0.0.0.255 any

no access-list 120
access-list 120 deny ip 194.10.8.0 0.0.1.255 any
access-list 120 permit tcp any host 194.10.9.7 eq 80
access-list 120 deny ip any 194.10.9.0 0.0.0.255
access-list 120 permit ip any any

interface Fa0/0
ip access-group 100 in

interface Fa1/0
ip access-group 120 in
```

Named access lists

Named access lists have one major advantage over numbered access lists, they can be edited. A named access list can be created, and later edited, with the **ip access-list** command:

ip access-list standard|extended ACCESS-LIST-NAME

This will make CLI enter in **named access list configuration mode**, either (config-std-nacl) for a standard access list or (config-ext-nacl) for an extended access list.

In **named access list configuration mode**, we add rules by using the same syntax as with numbered access list, except that the access-list command is omitted:

[SEQUENCE-NUMBER] permit|deny MATCH-CRITERIA-SPECIFICATION

The MATCH-CRITERIA-SPECIFICATION is exactly the same as for numbered access-list, likewise it will differ depending on being standard or extended.

Every rule in the named access list has a sequence number, if none is specified when adding, the rule it will be appended (added to the end. The first rule to be added will have sequence number 10 and next rules will be added with a 10 increment. Sequence numbers can be from 1 up to 2147483647.

If the SEQUENCE-NUMBER is specified the rule will be placed in that position, <u>if the position is already occupied the command fails</u>, it must be removed first.

Named access lists

We can remove a rule by using the command:

no SEQUENCE-NUMBER

On a real router, not in Packet Trace, you can also redefine the sequence numbers of rules in a named ACL:

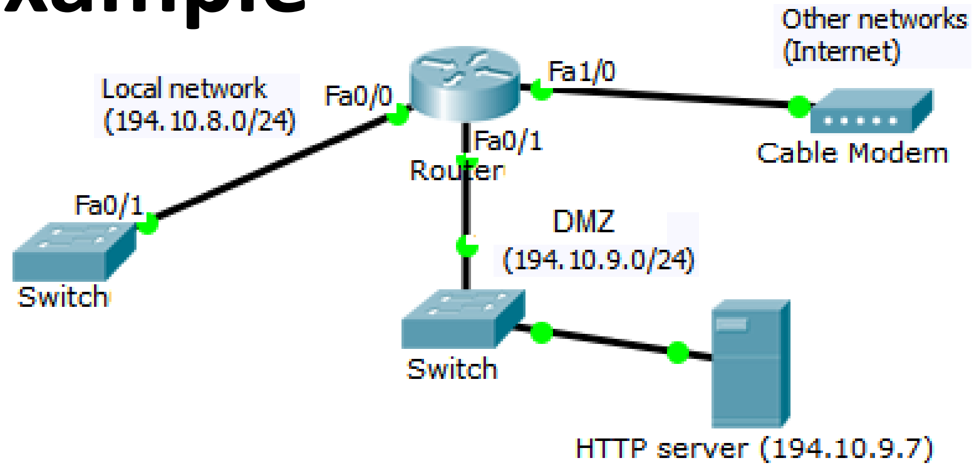
ip access-list resequence ACCESS-LIST-NAME starting-sequence-number increment-value

We can see the present sequence numbers of a named ACL by exiting the configuration mode (**end** command) and running the command:

show ip access-lists ACCESS-LIST-NAME

Named access lists example

The same scenario and objectives as before, now with named access lists.



```
(config)#ip access-list extended LOCAL
(config-ext-nacl)#permit tcp 194.10.8.0 0.0.0.255 host 194.10.9.7 eq 80
(config-ext-nacl)#deny ip any 194.10.9.0 0.0.0.255
(config-ext-nacl)#permit ip 194.10.8.0 0.0.0.255 any
(config-ext-nacl)#exit
(config)#ip access-list extended INTERNET
(config-ext-nacl)#deny ip 194.10.8.0 0.0.1.255 any
(config-ext-nacl)#permit tcp any host 194.10.9.7 eq 80
(config-ext-nacl)#deny ip any 194.10.9.0 0.0.0.255
(config-ext-nacl)#permit ip any any
(config-ext-nacl)#exit
(config)#interface Fa0/0
(config-if)#ip access-group LOCAL in
(config-if)#exit
(config)#interface Fa1/0
(config-if)#ip access-group INTERNET in
(config-if)#exit
(config)#
```

Additional notes regarding anti spoofing

We must be aware there are some cases where network administrators are powerless regarding IP spoofing:

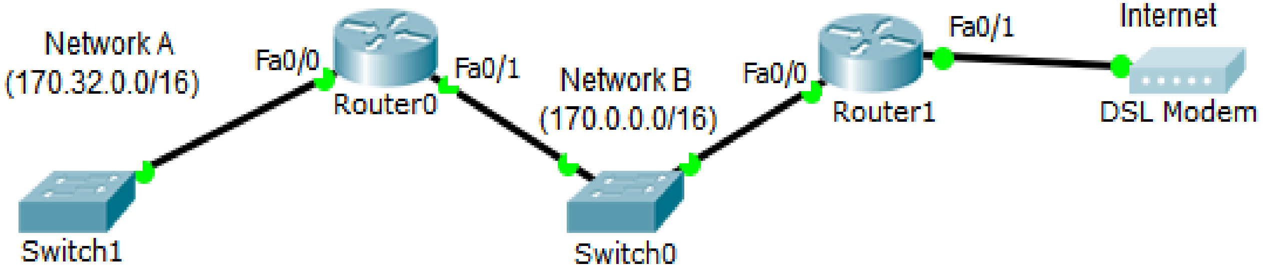
Internet incoming traffic – except for the local networks addresses, we can’t tell if one source address is spoofing or not, all we can do is block packets with local networks’ source addresses. This is why it’s so important for each network to play its role and block internal spoofing.

Network traffic within local networks – traffic between nodes in the same IPv4 network is not handled by any router and thus escapes the administrator’s control. To be able to do something about this we would have to use layer three switches. A layer three switch operates as a layer two switch but understands IPv4 and access control lists can be enforced to block local IP spoofing within the network.

Local transit networks – transit networks are used for traffic passing, this means some traffic travelling through them is neither originated by them nor destined to them. Obviously, this is an issue for anti-spoofing rules because we can no longer enforce that every incoming packet from that network must have a source address belonging to the network.

Example

Network B is a transit network, so anti-spoofing isn’t going to be perfect here.



```
                  Router0
no access-list 10
access-list 10 permit 170.32.0.0 0.0.255.255

no access-list 20
access-list 20 deny 170.32.0.0 0.0.255.255
access-list 20 permit any

interface Fa0/0
ip access-group 10 in

interface Fa0/1
ip access-group 20 in
```

```
                  Router1
no access-list 10
access-list 10 permit 170.0.0.0 0.32.255.255

no access-list 20
access-list 20 deny 170.0.0.0 0.32.255.255
access-list 20 permit any

interface Fa0/0
ip access-group 10 in

interface Fa0/1
ip access-group 20 in
```

We can see there are two spoofing issues about transit Network B:

- Network B is able to send packets to the internet with forged source addresses belonging to Network A
- Network B is able to send packets to Network A with any forged source address (except source addresses belonging to Network A).