Computer Networks *(Redes de Computadores - RCOMP)* – 2023/2024

Laboratory Class 07 (PL07 – 3 hours)

Static firewall. Cisco IOS Access Control Lists. Network Address Translation. Static and dynamic NAT in Cisco IOS. Network Address and Port Translation (NAPT). Packet Tracer activities.

1. Static firewall

Static firewalls permit or deny individual packets based on static rules defined by the network administrator. Other names for this firewall type are **First generation firewall**, **Packet Filter**, **Stateless Firewall, Screening Router** or yet **Network Layer Firewall**.

Criterions for packet matching (and consequent permit or deny) are relative to layer three and four properties: IP source and destination node addresses (IPv4/IPv6), payload type (e.g., ICMP, UDP, TCP), source and destination port numbers (for UDP and TCP payloads only), message types for ICMP payloads, etc.

Static firewalls react the same way to a given packet regardless of context, this makes them inadequate against DoS attacks. The problem is nothing allows it to distinguish between a packet from a licit access to the service and packet from a DoS attack to the service. A different type of firewall is required to handle DoS attacks, they are called dynamic or stateful.

Although they are not able to block some DoS attacks, static firewalls can do a lot on behalf of network security. On the top of the list of attacks that a static firewall can avoid are IP spoofing attacks.

1.1. IP spoofing

IP spoofing means forging the packet source IP address, this can be used for several illicit purposes, including concealing DoS attacks (Distributed DoS) to prevent them from being detected by dynamic firewalls. IP spoofing is also used to mislead static firewalls making them believe a packet is coming from an authorised source node address, and thus, allow access to some restricted services.

The measures to be taken to prevent IP spoofing, depend on whether traffic is coming from known networks, or is going to known networks. **The Local networks' addresses are known, but the internet networks' addresses are unknown.**

To avoid **internal spoofing**: any traffic going to the internet must have source addresses belonging to one of the known local networks.

To avoid **external spoofing**: any traffic coming from the internet must have source addresses not belonging to any of the known local networks.

1.2. Access control

Although end-nodes should have local firewalls, firewalls are most effective in routers because they can be used by network administrators to enforce access rules for the entire infrastructures. Under the security level point of view, internet connected infrastructures should be split in at least three zones:

Intranet - Local users' networks (known networks) – from where some attacks are expected.

Internet – All other networks (unknown addresses networks) – from where most attacks are expected.

Demilitarized Zone (DMZ) – Local isolated and trusted network – no attacks are expected form here.

The DMZ must be isolated from other networks by routers and firewalls, no user workstations are allowed here, only servers. Firewalls must be deployed between security level zones.

The recommended access policy for any firewall is: block all traffic except for explicitly required traffic.

1.3. Selecting the access control deployment place

Packets travel throughout the network infrastructure passing through several routers, access control can be enforced in any of them. A decision must be taken about where (in which router) access control should be enforce for each purpose. For each access policy to be implemented, there will usually be several alternative options regarding the deployment router, there's no absolute rule for a decision because specific environments can lead to different optimal solutions. The best solution is the one that results in fewer rules and also, avoids unnecessary traffic within the infrastructure.

Despite the inexistence of an absolute rule, we can, nevertheless, point out one general rule: block traffic as soon as possible.

The interpretation of this general rule is, block incoming traffic as close to its source as possible. In other words, when it's being received by the closest to the source router, if possible, the router that is directly connected to the source network.

Yet, this is not a golden rule, specific circumstances may turn this not to be the best choice.

2. Static firewall setup in Cisco IOS

In Cisco IOS we setup firewall rules by creating access control lists (ACL) and assigning them to incoming our outgoing traffic in specific router's interfaces.

An access list is a sequence of rules, each rule sets a match criterion and an action (permit or deny). Each individual packet is sequentially confronted with every rule on the ACL until a match is found. If a match is found, the action is executed, and the analysis ends for that packet. Therefore, the rules' order on the ACL is pretty relevant.

If no matching rule is found on the ACL the packet is denied by default.

While until no ACL is deployed all packets are allowed to pass, once an ACL is deployed, only packets matching a permit rule without previously matching a deny rule will be allowed.

2.1. Numbered standard access lists

They are identified by a number between one and 99. The only matching criterion for standard access list rules is the source IP address. To **add (append) a rule** to a numbered standard access list the following command is used:

```
access-list NN ACTION SOURCE-IP-ADDRESS-SPECIFICATION
```

Where NN represents the access list identifier (1 up to 99) and action is either **permit** or **deny**. Numbered access lists are not editable, the only two available operations are adding (appending) a rule and removing the entire ACL (remove all rules) by using the following command:

```
no access-list NN
```

Each added rule will match packets with source addresses conforming to SOURCE-IP-ADDRESS-SPECIFICATION, this can be either:

```
host DOT-DECIMAL-IP-ADDRESS
any
DOT-DECIMAL-IP-ADDRESS DOT-DECIMAL-WILDCARD
```

The first form matches a single IP address (DOT-DECIMAL-IP-ADDRESS), the second, matches any IP address. The last form matches all addresses equal to DOT-DECIMAL-IP-ADDRESS but bits with one value in DOT-DECIMAL-WILDCARD are ignored, this means it only compares bits with zero value in DOT-DECIMAL-WILDCARD.

2.2. Numbered extended access lists

They are identified by a number between 100 and 199. Again, numbered access lists are not editable, the only two available operations are adding (appending) a rule and removing the entire ACL (remove all rules). But, unlike standard access lists, several other criterions can be now used beyond the packet's source IP address. **Now, a packet matches a rule only if it matches at same time all criterions on the rule**. Required criterions are now the protocol identifier (IP packet payload type), the source address and the destination address:

```
access-list NNN ACTION PROTOCOL SOURCE-IP-ADDRESS DESTINATION-IP-ADDRESS
```

Where NNN represents the access list identifier, now a number between 99 and 199. The PROTOCOL identifier can be either **ip**, which matches any payload type, or a payload identifier (upper layer protocol) like, for instance, **udp**, **tcp** or **icmp**. Both the SOURCE-IP-ADDRESS and DESTINATION-IP-ADDRESS specifications can use any of the three forms mentioned earlier.

Additional optional criterions may be available depending on the matching protocol. For UDP and TCP a source port number criterion may be added to the source IP address and a destination port number criterion may be added to the destination IP address

For the ICMP protocol, a message type identifier criterion may be added. For TCP protocol, the **established** criterion may also be used to match only packets regarding an already established TCP connection.

2.3. Named access lists

Named access lists are identified by names and not numbers, but they work pretty the same as numbered ACLs. The biggest advantage is they can be edited. The command used to create a new, or edit an existing, named access list is:

```
ip access-list standard|extended ACL-NAME
```

This will make CLI go into to a specific configuration level for the named ACL, where rules can be entered, each with a sequence number. In no sequence number is specified for a rule, it's appended, starting with number 10 and with a step of 10. Depending on being standard or extended rules are entered the same way as with numbered access lists, except that the **access-list** command is omitted.

For instance, to insert a permit rule in ACL position 305 we would enter:

```
305 permit ...
```

This will fail if there's already a rule in position 305, we must first remove it:

```
no 305
```

At any time, we can renumber the rules in the named access list:

```
ip access-list resequence ACL-NAME STARTING-NUMBER STEP
```

2.4. Appling defined access lists

Just defining an access list has no immediate practical effect. It must be applied, either to incoming or outgoing traffic of a router's interface. Until one access list is applied to an interface traffic, all traffic is permitted, as soon as we apply one, only traffic permitted by the ACL is allowed. Incoming and outgoing traffics are independent, for instance, applying an access list to incoming traffic does not affect outgoing traffic.

To apply an access list to some interface's traffic we must first enter that interface configuration level:

```
interface INTERFACE-NAME
```

Once in interface configuration level, we can then apply one ACL to incoming traffic and another ACL to outgoing traffic:

```
ip access-group ACL-IDENTIFIER in|out
```

Where ACL-IDENTIFIER is either a number or a name. We may apply one ACL to incoming traffic and another ACL to outgoing traffic. But only one ACL to each traffic type.

3. Practical exercise

Create the Cisco Packet Tracer diagram on Figure 1. This diagram is available for download at Moodle, with already settled IP addresses, and routing configuration: pl07-a.pkt.

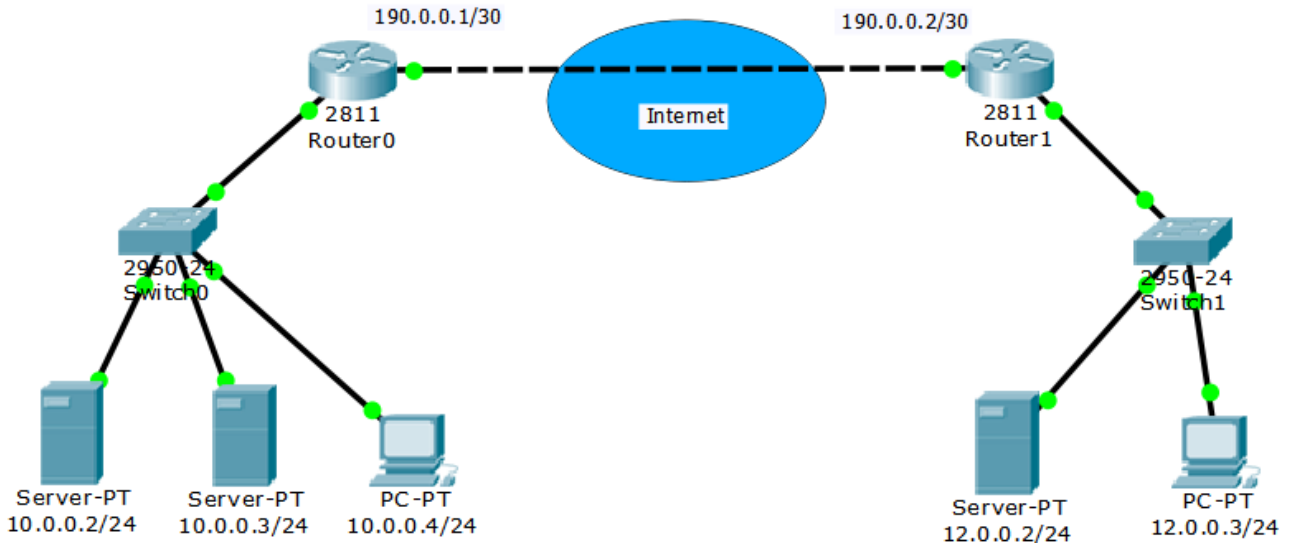

Figure 1 - Networks layout for ACL deployment

Even though for the purpose of this exercise the internet is just a point-to-point connection, we will treat it as being the internet. Therefore, Router 0 is not aware of the right-side network, it just known the default-gateway to use is 190.0.0.2 and everything beyond that is the internet (unknown addresses). As well, Router 1 is not aware of the left side network, it just known the default-gateway to use is 190.0.0.1 and everything beyond that is the internet (unknown addresses).

a) If not already done, settle nodes IP addresses and routing.

Perform ICMP echo request tests to be sure all end nodes can communicate with each other.

b) Now let's check IP spoofing is not being blocked.

In simulation mode.

Select the Add Complex PDU tool and click on PC 12.0.0.3

- Define as destination IP address: 10.0.0.2 (the leftmost server)

- Define as source IP address: 10.0.0.3 (this is IP spoofing)

- Define sequence number as 1, and one shot time as 5 seconds.

Start the simulation.

You will see the request arriving to server 10.0.0.2, without it realizing the source address is forged, therefore, it sends a reply to 10.0.0.3.

Server 10.0.0.3 ultimately discards the received echo reply because it hasn't asked for it in the first place.

The point is, imagine our access policy was blocking any external access to 10.0.0.3, the attacker at 12.0.0.2 has overcome that policy and could for instance perform an echo requests DoS attach to 10.0.0.3.

Regarding the presented scenario, we can highlight **two security faults** over IP spoofing:

First – the right-side network administrator is not preventing **internal spoofing** as he should. He is allowing packets from his internal network outgoing to the internet with forged IP source addresses.

Second – the left side network administrator is failing to protect his own networks from **external spoofing**. He is not blocking incoming packets from the internet with forged source addresses belonging to his internal networks.

c) Fix all IP spoofing issues on this scenario.

Because it's all about packets' IP source addresses, IP spoofing can be prevented by using **standard** access lists, nevertheless, they may have to be later converted to **extended** if additional access policies are to be enforced.

Starting with the left side router administrator (Router 0):

`(config)#no access-list 1`
`(config)#access-list 1 permit 10.0.0.0 0.0.0.255`
 	
`(config)#interface Fa0/1`
`(config-if)#ip access-group 1 in`
	To avoid internal spoofing, we create the numbered standard access list 1. It allows packets with a source address belonging to the internal network (10.0.0.0/24). Remaining traffic is, by default, blocked.
	
	Apply it to incoming traffic on interface Fa0/1.

`(config)#no access-list 2`
`(config)#access-list 2 deny 10.0.0.0 0.0.0.255`
`(config)#access-list 2 permit any`
 	
`(config)#interface Fa0/0`
`(config-if)#ip access-group 2 in`
	To avoid external spoofing, we create the numbered standard access list 2. It blocks any packet with a source address belonging to the internal network (10.0.0.0/24). Remaining traffic must be explicitly permitted, otherwise it would be blocked by default.
	
	Apply it to incoming traffic on interface Fa0/0.

Now, the same principles for the right-side router administrator (Router 1):

(config)#no access-list 10 (config)#access-list 10 permit 12.0.0.0 0.0.0.255 (config)#interface Fa0/1 (config-if)#ip access-group 10 in	To avoid internal spoofing, we create the numbered standard access list 10. It allows packets with a source address belonging to the internal network (12.0.0.0/24). Remaining traffic is, by default, blocked. Apply it to incoming traffic on interface Fa0/1.
(config)#no access-list 20 (config)#access-list 20 deny 12.0.0.0 0.0.0.255 (config)#access-list 20 permit any (config)#interface Fa0/0 (config-if)#ip access-group 20 in	To avoid external spoofing, we create the numbered standard access list 20. It blocks any packet with a source address belonging to the internal network (12.0.0.0/24). Remaining traffic must be explicitly permitted, otherwise it would be blocked by default. Apply it to incoming traffic on interface Fa0/0.

This settles it for IP spoofing blocking.

d) Test again sending IP spoofing ICMP echo request as done in letter b), packets with forged source IP addresses don't even get out to the internet.

Notice, though, that in fact, this does not solve every IP spoofing issue. For local traffic, not passing through any firewall there is yet a flaw.

e) Let's see this flaw.

In simulation mode.

Select the Add Complex PDU tool and click on PC 10.0.0.4

- Define as destination IP address: 10.0.0.2 (the leftmost server)

- Define as source IP address: 10.0.0.3 (this is IP spoofing)

- Define sequence number as 1, and one shot time as 5 seconds.

Start the simulation.

Again, you will see the request arriving to server 10.0.0.3, without it realizing the source address is forged, therefore, it sends a reply to 10.0.0.2.

Solving this would require a separate network with its own firewall for each node, so this is as far as network administrators can go. This must be handled by the node local firewall administrator, not the network administrator.

f) Now let's add additional traffic restrictions.

First let's start by checking some accesses with present configuration. As far as source addresses are not forged, all accesses are allowed. All these servers have an HTTP service, and also a client (Web Browser).

Select a server, then the Desktop tab and finally select Web Browser, enter the server IP address to be accessed by HTTP and you will see the main HTML page displayed. As things stand any node can access any HTTP server.

We are now going to enforce several restrictions for the left side router, the right-side router will remain with no further restrictions beyond spoofing blocking.

Access policy for the left side administrator:

- Allow ICMP echo requests (echo message type) from the internet to server 10.0.0.2 and respective ICMP echo replies (echo-reply message type).
- Allow HTTP (TCP port 80) access from the internet to server 10.0.0.3 and reply traffic back to the internet.
- All other traffic is to be blocked.

Previously defined ACLs for Router 0 will be now useless because defined criterions are not compatible with standard access lists, nevertheless, IP spoofing measures must still be effective. Old access lists can be erased or will simply be ignored when we assign the new ones to interfaces traffic.

Internet incoming traffic on Router 0 (traffic from internet to the local network)

A solution with a numbered extended access list applied to Fa0/0 incoming traffic:

```
(config)#no access-list 100
(config)#access-list 100 deny ip 10.0.0.0 0.0.0.255 any
(config)#access-list 100 permit icmp any host 10.0.0.2 echo
(config)#access-list 100 permit tcp any host 10.0.0.3 eq 80

(config)#interface Fa0/0
(config-if)#ip access-group 100 in
```

Intranet incoming traffic on Router 0 (traffic from the local network to the internet)

A solution with a named extended access list applied to Fa0/1 incoming traffic:

```
(config)#ip access-list extended FROM-LOCAL
(config-ext-nacl)#permit icmp host 10.0.0.2 any echo-reply
(config-ext-nacl)#permit tcp host 10.0.0.3 eq 80 any established

(config)#interface Fa0/1
(config-if)#ip access-group FROM-LOCAL in
```

g) Now let's test it.

Try to establish a relation between each test result and the enforced access lists.

- Send ICMP echo requests from server 10.0.0.2 to servers 10.0.0.3 and 12.0.0.2
- Send ICMP echo requests from server 12.0.0.2 to servers 10.0.0.2 and 10.0.0.3
- Try other ICMP echo requests, including to routers.
- Open the Web Browser in server 10.0.0.2, try opening URLs http://10.0.0.3 and http://12.0.0.2
- Open the Web Browser in server 12.0.0.2, try opening URLs http://10.0.0.2 and http://10.0.0.3
- Open the Web Browser in server 10.0.0.3, try opening URLs http://10.0.0.2 and http://12.0.0.2

4. Network Address Translation

NAT is a technique by which packets' source and destination addresses are automatically changed to achieve several different goals, some of them related to the use of private addresses. NAT is usually implemented in intermediate layer three nodes like routers, and it must be invisible to end-nodes.

Changing packets' source addresses is called SNAT (Source Network Address Translation), changing packets' destination addresses is called DNAT (Destination Network Address Translation). NAT is invisible to end nodes because to a SNAT application in one direction matches a DNAT application in the opposite direction.

Two settings are required to establish the NAT application:

- Addresses equivalency – This defines which address in one side is equivalent to another address on the other side, meaning one will be automatically replaced by the other. NAT addresses equivalencies are usually presented as a **NAT table**.
- It must define the translation direction, to do this, sides need to be identified. We could call them side A and side B, but in Cisco routers, they are called **inside** and **outside**, though these names take no special meaning. Once sides are identified, then the NAT application direction may be established in either two equivalent ways:
    - Instruct the router to apply SNAT when packets travel from inside to outside, this automatically implies DNAT will be applied when packets travel from outside to inside.
    - Instruct the router to apply DNAT when packets travel from inside to outside, this automatically implies SNAT will be applied when packets travel from outside to inside.

4.1. Cisco IOS NAT commands

One thing we must do is set sides. In a router, sides are related to network interfaces, so we must first enter the interface configuration level and then use commands **ip nat inside** or **ip nat outside** to settle to which side does the interface belongs. NAT is applied only to traffic being transferred from an inside interface and an outside interface and vice versa. **Never between same side interfaces**.

Once sides are settled, we can enforce NAT by using (in general configuration level) one of the following commands:

ip nat inside source ...	Apply SNAT to packets traveling from inside to outside. (Implies DNAT will be applied to packets traveling from outside to inside)
ip nat outside source ...	Apply SNAT to packets traveling from outside to inside (Implies DNAT will be applied to packets traveling from inside to outside)

4.2. Static NAT

Static NAT means the NAT table is static and manually defined by the administrator, there is a static equivalency between the inside address and the outside address. Both the inside and outside addresses are permanently allocated for this purpose.

Imagine the setup in Figure 2.

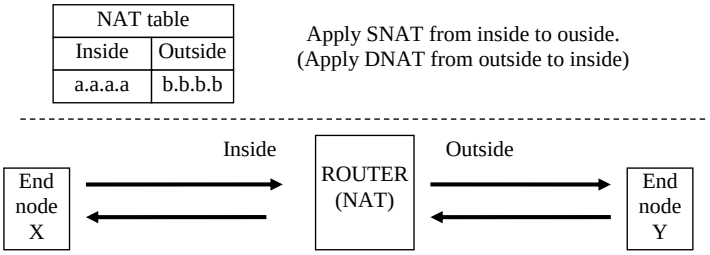

Figure 2 - Static NAT

We have a NAT table and sides identified, we also have defined the SNAT application direction, and consequently the DNAT application in the opposite direction.

- Now imagine end-node X has address a.a.a.a and sends a packet to end-node Y, when the packet travels through the router (inside to outside) the packet's **source address** is translated from a.a.a.a to b.b.b.b, thus, end-node Y is going to receive a packet with source address b.b.b.b.

- Now imaging end node Y sends a packet to address b.b.b.b (may be a reply to the previous packet), when the packet travels through the router (outside to inside) the packet's **destination address** is translated from b.b.b.b to a.a.a.a, thus, it's going to be delivered to node X.

From this setup we can see some NAT features. In this configuration, inside nodes have their source addresses hidden from outside nodes. Also, indirect access to inside nodes can be provided if outside nodes send requests to the outside address.

These features turn to be very useful if inside is a private network and outside a public network (internet). Inside nodes using private addresses will be able to access internet services because, due to SNAT, when requests reach the internet servers they will be coming from a public address (b.b.b.b on the earlier scenario). Server replies, of course, are sent to the public address, but then, DNAT ensures they will reach the real source node with the inside private address.

We can also have servers on the inside private network, then outside internet clients must send requests to the public address (b.b.b.b on the above scenario), DNAT will redirect them to the inside private server address and the server reply will have SNAT applied to hide the private source address.

We can deploy static NAT in a Cisco router by using the command:

```
ip nat inside source static IP-ADDRESS-1 IP-ADDRESS-2
```

This means packets with source address IP-ADDRESS-1 traveling from an inside interface to an outside interface will have the source address changed to IP-ADDRESS-2, also, packets with destination address IP-ADDRESS-2 traveling from an outside interface to an inside interface will have the destination address changed to IP-ADDRESS-1.

The command **ip nat outside source static IP-ADDRESS-2 IP-ADDRESS-1** is therefore equivalent to the previous command.

If there are two different commands to do the same thing, you may ask why they are required. Remember one interface may be inside or outside, not both. We may want to apply SNAT to some traffic in one direction and SNAT to other traffic in the opposite direction, then the two different commands will be required:

```
ip nat inside source static IP-ADDRESS-1 IP-ADDRESS-2
ip nat outside source static IP-ADDRESS-3 IP-ADDRESS-4
```

4.3. Dynamic NAT

Dynamic NAT means letting the router automatically create the NAT table entries as requests come. However, because we don't want to expose private nodes to external access, dynamic NAT entries store information about protocols and remote nodes being accessed (Table 1).

Table 1 - Example dynamic NAT table

Dynamic NAT table		
Inside	**Outside**	**Remote**
192.168.10.5 – ICMP:2234	**190.10.7.8 – ICMP:10226**	**120.1.72.235 – ICMP:10226**
192.168.10.200 – UDP:2288	**190.10.7.9 – UDP:45611**	**182.67.20.1 – UDP:34**
172.17.0.235 – TCP:45002	**190.10.7.6 – TCP:25516**	**120.50.7.3 – TCP:80**

Picking, for instance, the **second line**, it exists because the inside node 192.168.10.200 has sent a UDP packet to port number 34 of node address 182.67.20.1. The line will allow that node to reply back (to 190.10.7.9), nevertheless, only a UDP packet from node 182.67.20.1, port 34 sent to address 190.10.7.9 port 45611 will be allowed. This avoids exposing inside node 192.168.10.200 to other external traffic, something static NAT could not prevent.

Dynamic NAT configuration in Cisco routers requires some additional items:

- **An addresses' pool**. This is a set of outside IP addresses that are made available for dynamic NAT management and are to be assign to inside nodes as needed.
- **An ACL** to match (permit) packets to which we want NAT to be applied.

Defining an address pool:

```
ip nat pool POOL-NAME IP-ADDRESS-1 IP-ADDRESS-2 netmask NETWORK-MASK
```

This defines a pool named POOL-NAME with addresses starting at IP-ADDRESS-1 up to IP-ADDRESS-2 and the given network mask.

Dynamic NAT can then be deployed by:

```
ip nat inside source list ACL-IDENTIFIER pool POOL-NAME
```

Out of the configuration level, we can see the current NAT translations with the ip nat translations command line (Figure 3).

```
Router#show ip nat translations
Pro  Inside global     Inside local       Outside local      Outside global
icmp 12.0.0.101:2      192.168.0.3:2      170.0.0.3:2        170.0.0.3:2
---  12.0.0.5          192.168.0.4        ---                ---
tcp 12.0.0.100:1025    192.168.0.2:1025   170.0.0.2:80       170.0.0.2:80

Router#
```

Figure 3 - ip nat translations command

In this example's output, the second line is from static NAT, others are from dynamic NAT. You may be puzzled why four columns of addresses exist.

Local means addresses to be used inside, **Global** refers to addresses to be used outside. If local is the same as global, then there is no translation on that side.

For instance, the first line in the above sample has one **Inside global** and a different **Inside local**, this means there is a translation for the inside address, however, **Outside local** and **Outside global** are the same, this means there is no translation for the outside address.

Yet, as we can see in Figure 4 it's possible to have translation for both inside addresses and outside addresses at the same time.

```
Router#show ip nat translations
Pro  Inside global     Inside local       Outside local      Outside global
icmp 12.0.0.101:1      192.168.0.3:1      170.0.0.2:1        170.0.0.2:1
icmp 12.0.0.101:2      192.168.0.3:2      170.0.0.2:2        170.0.0.2:2
---  12.0.0.5          192.168.0.4        ---                ---
tcp 12.0.0.100:1025    192.168.0.2:1025   170.0.0.2:80       170.0.0.2:80
---  ---               ---                192.168.0.5        12.0.0.2

Router#
```

Figure 4 - NAT translations inside and outside

5. Practical exercise

Create or download from Moodle (pl07-b.pkt) the Cisco Packet Tracer diagram in Figure 5. The downloaded version has already settled IP addresses and routing information.

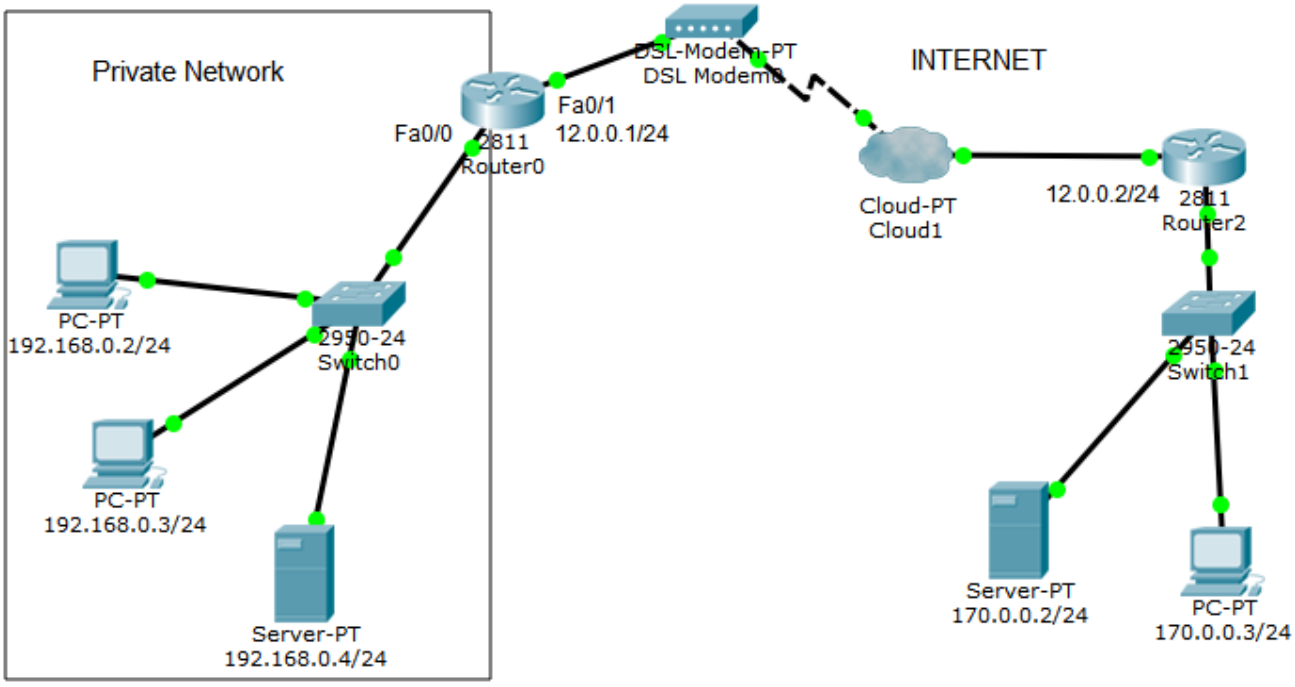

Figure 5 - Networks layout with a private network

The left side network is private, therefore, is not known to the internet, namely Router 2.

a) If not done already, setup IP addresses and routing.

Default gateway for Router 0 is 12.0.0.2, however, Router 2 will not have a default gateway, it knowns Router 0 (12.0.0.1) but it's not aware there is a network behind it, because it's a private network.

Please, before carrying on, save this configuration to a separate file for later use.

b) Check IPv4 connectivity with ICMP echo requests.

The left side network fails to communicate with right side network and vice versa. This is because the left side network is unknown and, therefore, unreachable. In simulation mode we can see requests from left side (private) are reaching the right side (public), however, replies are not reaching the private network.

c) Apply static NAT on Router 0 to allow access from the internet to server 192.168.0.4

This is one thing static NAT is good for. First, we must set NAT sides:

```
(config)#interface Fa0/0
(config-if)#ip nat inside
(config)#interface Fa0/1
(config-if)#ip nat outside
```

Now we can set static NAT:

```
(config)#ip nat inside source static 192.168.0.4 12.0.0.10
```

d) Check if it works.

Go to PC 170.0.0.3 and open the Web Browser, to access server 192.168.0.4 we must now enter URL http://12.0.0.10

This static NAT configuration has another effect, now server 192.168.0.4 can access to the internet. Check that by sending ICMP echo requests from it to internet nodes.

e) Apply dynamic NAT on Router 0 to allow private nodes access to the internet.

We need an address pool with outside addresses:

```
(config)#ip nat pool MYPOOL 12.0.0.100 12.0.0.150 netmask 255.255.255.0
```

We also need an ACL to match traffic to be translated:

```
(config)#no access-list 10
(config)#access-list 10 permit 192.168.0.0 0.0.0.255
```

Now we can settle dynamic NAT:

```
(config)#ip nat inside source list 10 pool MYPOOL
```

f) See dynamic NAT working

Use the Inspect tool to display Router 0 NAT table. Keep it visible.

Now, send ICMP echo requests from PCs 192.168.0.2 and 192.168.0.3 to internet nodes.

Open the Web Browsers in PCs 192.168.0.2 and 192.168.0.3 and access to URL http://170.0.0.2

6. Network Address and Port Translation

For protocols that use port numbers (UDP and TCP), NAPT can be used to handle with port numbers and not only IP addresses. It works the same way as NAT.

For static NAT commands are:

ip nat inside source static udp IP-ADDRESS-1 PORT-1 IP-ADDRESS-2 PORT-2
This will apply SNAT to UDP packets coming from address IP-ADDRESS-1 with source port number PORT-1, traveling from inside to outside, changing the packet source address to IP-ADDRESS-2 and changing the source port number to PORT-2. Likewise, it will apply DNAT to UDP packets sent to address IP-ADDRESS-2 with destination port number PORT-2, traveling from outside to inside, changing the packet destination address to IP-ADDRESS-1 and the destination port to PORT-1-

For the TCP protocols it's the same:

ip nat inside source static tcp IP-ADDRESS-1 PORT-1 IP-ADDRESS-2 PORT-2

7. Dynamic NAT with overload

Overload means using the same inside global address for several different inside local addresses, this is possible because port numbers are used to distinguish traffic belonging to different inside local addresses.

If only one inside global address is to be used, and it's the address already assigned to the outside router's interface, then we may skip the address pool definition and use the following command:

```
ip nat inside source list ACL-IDENTIFIER interface INTERFACE-NAME overload
```

This will apply SNAT to packets traveling from inside to outside that match access list ACL-IDENTIFIER, changing the packet source address to the IP address of interface INTERFACE-NAME. Possibly source port (for UDP or TCP) or message ID (for ICMP) will also be changed so that later reply traffic can have DNAT applied correctly.

In Figure 6 we can see the command output showing a NAT table in this overload scenario, we can see there is only one inside global address being overloaded with several inside local addresses.

If we pay a close attention to the last two lines, we can see dynamic NAT was forced to change inside global source port numbers. If it did not so, there would be lines with same values for Inside global, Outside local and Outside global. Then the router wouldn't be able to tell which DNAT translation it was supposed apply.

```
Router#show ip nat translations
Pro  Inside global     Inside local       Outside local      Outside global
icmp 12.0.0.1:1        192.168.0.3:1      170.0.0.3:1        170.0.0.3:1
icmp 12.0.0.1:2        192.168.0.3:2      170.0.0.2:2        170.0.0.2:2
tcp 12.0.0.1:1024      192.168.0.4:1025   170.0.0.2:80       170.0.0.2:80
tcp 12.0.0.1:1025      192.168.0.2:1025   170.0.0.2:80       170.0.0.2:80
tcp 12.0.0.1:1026      192.168.0.2:1026   170.0.0.2:80       170.0.0.2:80
tcp 12.0.0.1:1027      192.168.0.3:1025   170.0.0.2:80       170.0.0.2:80
tcp 12.0.0.1:1028      192.168.0.2:1027   170.0.0.2:80       170.0.0.2:80

Router#
```

Figure 6 - Dynamic NAT with overload

8. Practical exercise

We will now redo the previous exercise, using dynamic NAT with overload (NAPT).
(Use the saved configuration, or download it again)

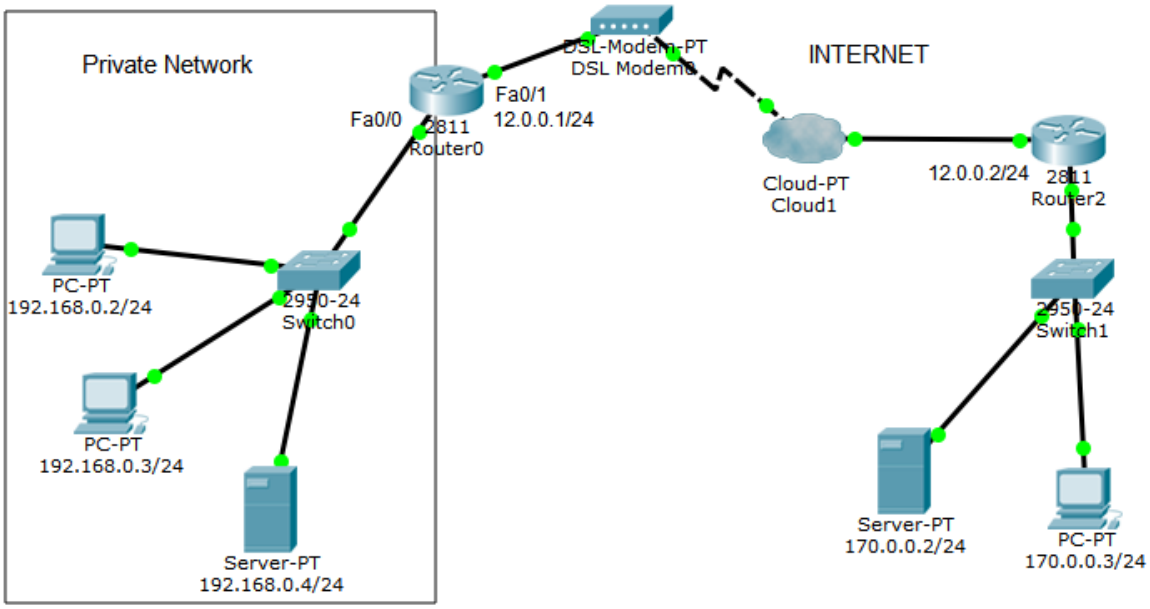

Figure 7 - Networks layout with a private network

a) Check IPv4 connectivity with ICMP echo requests.

Because left side network is private and right-side public (internet), the left side network fails to communicate with right side network and vice versa.

b) Use static NAPT on Router 0 to allow internet access to the 192.168.0.4 HTTP server. The service must be available at address 12.0.0.1, port number 8080.

First, we settle NAT sides:

```
(config)#interface Fa0/0
(config-if)#ip nat inside
(config)#interface Fa0/1
(config-if)#ip nat outside
```

Now we can set static NAPT (remember HTTP runs over TCP and, by default, uses port number 80):

```
(config)#ip nat inside source static tcp 192.168.0.4 80 12.0.0.1 8080
```

c) Check the created static NAPT configuration.

Go to PC 170.0.0.3 and open a Web Browser, then enter URL **http://12.0.0.1:8080**. This is Router 0 outside interface, the port number must be explicitly declared because it's not the standard port number for the HTTP service.

Notice that 192.168.0.4 server is not as exposed to external access as before, only TCP traffic to port number 8080 of address 12.0.0.1 will go through. This also means that, unlike before, this server is not able to access internet addresses. But we will fix that next.

d) Apply dynamic NAT with overload to allow private nodes access to the internet

Creating and address pool, is now pointless because we will use only one public address. Instead of declaring a pool we can directly refer the router's interface.

Though, an ACL to match traffic to be translated is required:

```
(config)#no access-list 10
(config)#access-list 10 permit 192.168.0.0 0.0.0.255
```

Now we can set dynamic NAT with overload:

```
(config)#ip nat inside source list 10 interface Fa0/1 overload
```

Because we chose not to create an addresses' pool, we use the interface parameter to identify the single address to be used and overloaded.

e) See dynamic NAT with overload working

Use the Inspect tool to display Router 0 NAT table. Keep it visible.

Now, send ICMP echo requests from PCs 192.168.0.2 and 192.168.0.3 to internet nodes.

Open the Web Browsers in PCs 192.168.0.2 and 192.168.0.3 and access to URL http://170.0.0.2