## netkit lab

**network address translation**

<table>
<thead>
<tr>
<th>Version</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Massimo Rimondini</td>
</tr>
<tr>
<td><strong>E-mail</strong></td>
<td><a href="mailto:contact@netkit.org">contact@netkit.org</a></td>
</tr>
<tr>
<td><strong>Web</strong></td>
<td><a href="http://www.netkit.org/">http://www.netkit.org/</a></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A simple lab showing the operation of a NAT gateway using static NAT</td>
</tr>
</tbody>
</table>
copyright notice

- All the pages/slides in this presentation, including but not limited to, images, photos, animations, videos, sounds, music, and text (hereby referred to as “material”) are protected by copyright.

- This material, with the exception of some multimedia elements licensed by other organizations, is property of the authors and/or organizations appearing in the first slide.

- This material, or its parts, can be reproduced and used for didactical purposes within universities and schools, provided that this happens for non-profit purposes.

- Information contained in this material cannot be used within network design projects or other products of any kind.

- Any other use is prohibited, unless explicitly authorized by the authors on the basis of an explicit agreement.

- The authors assume no responsibility about this material and provide this material “as is”, with no implicit or explicit warranty about the correctness and completeness of its contents, which may be subject to changes.

- This copyright notice must always be redistributed together with the material, or its portions.
nat types

- rfc 3489 enumerates four “treatments observed in implementations”:
  - full cone
  - restricted cone
  - port restricted cone
  - symmetric
- obsoleted by rfc 5389, because “many NATs did not fit cleanly into the types defined there”
- anyway...
nat types (from rfc 3489) full cone nat

- requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)
nat types (from rfc 3489)
full cone nat

requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)
nat types (from rfc 3489)

full cone nat

- requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- any external host can send a packet to the internal host, by using its mapped external address
nat types (from rfc 3489)
full cone nat

- requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- any external host can send a packet to the internal host, by using its mapped external address
nat types (from rfc 3489)

full cone nat

- requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- any external host can send a packet to the internal host, by using its mapped external address
nat types (from rfc 3489) (address) restricted cone nat

- requests from the same internal \((I_{IP},I_{port})\) are mapped to the same external \((E_{IP},E_{port})\)
nat types (from rfc 3489)  
(address) restricted cone nat

requests from the same internal \((I_{IP},I_{port})\) are mapped to the same external \((E_{IP},E_{port})\)
nat types (from rfc 3489) (address) restricted cone nat

- requests from the same internal $({I}_{IP},{I}_{port})$ are mapped to the same external $({E}_{IP},{E}_{port})$

- any external host $({H}_{IP},{*})$ can send a packet to the internal host, only if the latter has previously sent a packet to $({H}_{IP},{*})$
nat types (from rfc 3489) (address) restricted cone nat

- requests from the same internal \((I_{IP},I_{port})\) are mapped to the same external \((E_{IP},E_{port})\)

- any external host \((H_{IP},*)\) can send a packet to the internal host, only if the latter has previously sent a packet to \((H_{IP},*)\)
nat types (from rfc 3489) (address) restricted cone nat

- requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- any external host \((H_{IP}, *)\) can send a packet to the internal host, only if the latter has previously sent a packet to \((H_{IP}, *)\)
nat types (from rfc 3489)
port restricted cone nat

- requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)
nat types (from rfc 3489) 
port restricted cone nat

requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)
nat types (from rfc 3489)
port restricted cone nat

- requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- any external host \((H_{IP}, H_{port})\) can send a packet to the internal host, only if the latter has previously sent a packet to \((H_{IP}, H_{port})\)
nat types (from rfc 3489)
port restricted cone nat

- requests from the same internal \((I_{IP}, I_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- any external host \((H_{IP}, H_{port})\) can send a packet to the internal host, only if the latter has previously sent a packet to \((H_{IP}, H_{port})\)
nat types (from rfc 3489)
port restricted cone nat

- requests from the same internal \((I_{IP},I_{port})\) are mapped to the same external \((E_{IP},E_{port})\)

- any external host \((H_{IP},H_{port})\) can send a packet to the internal host, only if the latter has previously sent a packet to \((H_{IP},H_{port})\)
nat types (from rfc 3489)  
port restricted cone nat

- requests from the same internal \((I_{IP},I_{port})\) are mapped to the same external \((E_{IP},E_{port})\)

- any external host \((H_{IP},H_{port})\) can send a packet to the internal host, only if the latter has previously sent a packet to \((H_{IP},H_{port})\)
nat types

symmetric nat

- requests from the same internal \((I_{IP}, I_{port})\) to the same
destination \((D_{IP}, D_{port})\) are mapped to the same external
\((E_{IP}, E_{port})\)
**nat types**

**symmetric nat**

- requests from the same internal \((I_{IP}, I_{port})\) to the same destination \((D_{IP}, D_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- packets from the same \((I_{IP}, I_{port})\) but with a different destination are mapped to a different \((E_{IP}, E_{port})\)
nat types
symmetric nat

- requests from the same internal \((I_{IP}, I_{port})\) to the same destination \((D_{IP}, D_{port})\) are mapped to the same external \((E_{IP}, E_{port})\).

- packets from the same \((I_{IP}, I_{port})\) but with a different destination are mapped to a different \((E_{IP}, E_{port})\).
**symmetric nat**

- requests from the same internal \((I_{IP}, I_{port})\) to the same destination \((D_{IP}, D_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- packets from the same \((I_{IP}, I_{port})\) but with a different destination are mapped to a different \((E_{IP}, E_{port})\)
  
  only replies can be sent back to \((I_{IP}, I_{port})\)
nat types

symmetric nat

- requests from the same internal \((I_{IP},I_{port})\) to the same destination \((D_{IP},D_{port})\) are mapped to the same external \((E_{IP},E_{port})\)

- packets from the same \((I_{IP},I_{port})\) but with a different destination are mapped to a different \((E_{IP},E_{port})\)
  only replies can be sent back to \((I_{IP},I_{port})\)
nat types

symmetric nat

- requests from the same internal \((I_{IP}, I_{port})\) to the same destination \((D_{IP}, D_{port})\) are mapped to the same external \((E_{IP}, E_{port})\)

- packets from the same \((I_{IP}, I_{port})\) but with a different destination are mapped to a different \((E_{IP}, E_{port})\)
  only replies can be sent back to \((I_{IP}, I_{port})\)
lab description – hosts

- really have no setup
- just a default route pointing to the gateway

host1.startup

ifconfig eth0 10.0.0.2 netmask 255.255.255.0 up
route add default gw 10.0.0.1

host2.startup

ifconfig eth0 10.0.0.3 netmask 255.255.255.0 up
route add default gw 10.0.0.1
lab description – servers

- “servers” just because they have a public address
- have an empty configuration (really!)
- automatically launch a sniffer to easily see the traffic

These statements simply split the machine’s terminal to keep the sniffer handy.

```bash
server1.startup
ifconfig eth0 193.204.161.65 up
touch /hostlab/$HOSTNAME.ready
cd /root
screen -c /root/screenrc
```

```bash
server2.startup
ifconfig eth0 193.204.161.103 up
touch /hostlab/$HOSTNAME.ready
cd /root
screen -c /root/screenrc
```
lab description – gateway

- uses iptables to implement the various nat types
- by default, implements a full cone nat
- nat type can be switched by using `set_nat_type.sh`

```bash
root@gateway:/root#c./set_nat_type.sh
Usage: ./set_nat_type.sh nat_type

where nat_type is one of the following, self-explaining, NAT types:
  f)ullcone
  r)estricted
  p)ortrestricted
  s)ymmetric

root@gateway:/root# ```
lab description – gateway

- iptables rules implementing the nat are shown in real time on the gateway
lab description – gateway

- **gateway**’s external interface (**eth1**) is assigned multiple IP addresses by using aliases
- these address make up the pool of public addresses to which private addresses are mapped

```
gateway:~# ifconfig | grep -A 1 eth1
eth1      Link encap:Ethernet  HWaddr de:c3:bf:f8:7a:a
           inet addr:193.204.161.14  Bcast:193.204.161.255  Mask:255.255.255.0

eth1:1   Link encap:Ethernet  HWaddr de:c3:bf:f8:7a:a
          inet addr:193.204.161.15  Bcast:193.204.161.255  Mask:255.255.255.0

eth1:2   Link encap:Ethernet  HWaddr de:c3:bf:f8:7a:a
          inet addr:193.204.161.16  Bcast:193.204.161.255  Mask:255.255.255.0

eth1:3   Link encap:Ethernet  HWaddr de:c3:bf:f8:7a:a
          inet addr:193.204.161.17  Bcast:193.204.161.255  Mask:255.255.255.0
```
experiment 1 – address mapping

- let’s experiment the mapping from a private to a public address (we disregard ports for the moment)
- go on host1 and ping 193.204.161.65 (server1)

```
host1:~# ping 193.204.161.65
PING 193.204.161.65 (193.204.161.65) 56(84) bytes of data.
64 bytes from 193.204.161.65: icmp_seq=1 ttl=63 time=0.329 ms
64 bytes from 193.204.161.65: icmp_seq=2 ttl=63 time=0.338 ms
64 bytes from 193.204.161.65: icmp_seq=3 ttl=63 time=0.352 ms
```

- do the same from host2
experiment 1 – address mapping

- let’s experiment the mapping from a private to a public address (we disregard ports for the moment)
- go on host1 and ping 193.204.161.65 (server1)

interesting note:
despite server1 having no clue in its routing table about how to get to host1 (10.0.0.2) or to a default route, server1 still sends ICMP replies back to host1

can you tell why the ping works?
experiment 1 – address mapping

- let’s experiment the mapping from a private to a public address (we disregard ports for the moment)
- go on host1 and ping 193.204.161.65 (server1)

```
host1:~# ping 193.204.161.65
PING 193.204.161.65 (193.204.161.65) 56(84) bytes of data.
64 bytes from 193.204.161.65: icmp_seq=1 ttl=63 time=0.329 ms
64 bytes from 193.204.161.65: icmp_seq=2 ttl=63 time=0.338 ms
64 bytes from 193.204.161.65: icmp_seq=3 ttl=63 time=0.352 ms
```

- do the same from host2
- by looking at the ready-to-use sniffer on server1, we can learn that
  - host1 is mapped to the external IP address 193.204.161.14
  - host2 is mapped to the external IP address 193.204.161.15
**experiment 1 – address mapping**

- **gateway** is set up to not alter ports
- therefore, traffic from \((\text{host}_1, I_{\text{port}})\) is always mapped to \((193.204.161.14, I_{\text{port}})\), and traffic from \((\text{host}_2, I_{\text{port}})\) is always mapped to \((193.204.161.15, I_{\text{port}})\)
- check it, e.g., by using `nc 193.204.161.65 999 -p 100` from **host1** and **host2**, using different source port numbers

**attempt to establish a TCP connection to 193.204.161.65 port 999, using source port 100**
experiment 1 – address mapping

- **conclusion**
  - **host1** and **host2** are correctly mapped to an external public IP address
  - **host1** and **host2** are *always* mapped to the *same* address
  - port numbers are never changed, obeying nat rules for traffic from the private to the public network

- **exercise:** you can perform the same check for other nat types (but for the symmetric one)
experiment 2 – full cone

- Let's experiment full cone nat
- **netcat (nc)**: A simple utility to transfer stdin/stdout over a TCP connection
- Start nc in server mode on **host1**
  ```
  host1:~# nc -l -p 100
  listening mode
  port to listen on
  ```
- From **server2**, use nc to connect to the server running on **host1**
  ```
  root@server2:/root# nc 193.204.161.14 100
  host and port to connect to
  ```
experiment 2 – full cone

- type any text on server2’s terminal and see it appearing on host1’s terminal

```
host1:~# nc -l -p 100
```

```
server2
root@server2:/root# nc 193.204.161.14 100
```
experiment 2 – full cone

- type any text on server2’s terminal and see it appearing on host1’s terminal

```
host1:
host1:~# nc -l -p 100

server2:
root@server2:/root# nc 193.204.161.14 100
hello there
```
experiment 2 – full cone

- type any text on server2’s terminal and see it appearing on host1’s terminal

```
host1:
host1:~# nc -l -p 100
hello there
```

```
server2:
root@server2:/root# nc 193.204.161.14 100
hello there
```
experiment 2 – full cone

- conclusion & exercises
  - **server2** can send traffic to **host1** even if **host1** has never sent traffic to **server2** before
    (yes, we have sent some pings before, but restart the lab if you don’t trust 😊)
  - repeat the check for traffic from **server2** to **host2**, from **server1** to **host1**, and from **server1** to **host2**, varying the port numbers
experiment 3 – restricted

- Let’s experiment restricted nat
- Change the nat type on **gateway** to “restricted”

```bash
$ ./set_nat_type.sh restricted
```

- You should see iptables rules on **gateway** change
(yet another) interesting note:

- the behavior of (port) restricted nat does not fit well the connection-oriented semantic of iptables rules
- as a consequence, (port) restricted nat is well known to be very hard to implement in Linux
- in the lab, (port) restricted nat is implemented by some hacks
  - iptables rules are dynamically added according to the traffic that traverses the gateway
  - rules automatically expire after 30s
experiment 3 – restricted

- start two instances of `nc` in server mode on `host1`, each listening on a different port

```
host1:~# nc -l -p 100 & nc -l -p 200 &
[1] 522
[2] 523
```

- from `server2`, attempt to connect to one of the `nc` instances on `host1`

```
root@server2:/root# nc 193.204.161.14 100
(UNKNOWN) [193.204.161.14] 100 (?) : Connection refused
root@server2:/root# nc 193.204.161.14 200
(UNKNOWN) [193.204.161.14] 200 (?) : Connection refused
```
experiment 3 – restricted

- the connection correctly fails until host1 sends some TCP traffic to server2
- let’s try to send TCP traffic (a connection attempt is enough) to server1 instead

```
root@server2:/root# nc 193.204.161.14 100
(UNKNOWN) [193.204.161.14] 100 (?) : Connection refused
root@server2:/root# nc 193.204.161.14 200
(UNKNOWN) [193.204.161.14] 200 (?) : Connection refused
```

- the connection from server2 to host1 still does not work

```
host1:~# nc 193.204.161.65 999
(UNKNOWN) [193.204.161.65] 999 (?) : Connection refused
```

```
experiment 3 – restricted

- now let’s send TCP traffic to server2
  ```
  host1:
  host1:~# nc 193.204.161.103 999
  (UNKNOWN) [193.204.161.103] 999 (?) : Connection refused
  ```

- gateway automatically sets a rule for the nat
- traffic from server2 (any port) to any of host1’s ports is now possible, and messages are delivered by nc
  ```
  server2:
  root@server2:/root# nc 193.204.161.14 100
  test message
  ^C
  root@server2:/root# nc 193.204.161.14 200
  another test message
  ^C
  ```

- after 30 seconds, it is again impossible to initiate a connection from server2 to host1
experiment 3 – restricted

- conclusion & exercises
  - traffic can be sent from an external host to an internal one only if the internal host has sent a packet to the external host beforehand
  - check by considering other source-destination pairs and changing port numbers
experiment 4 – port restricted

- let’s experiment port restricted nat
- change the nat type on `gateway` to “portrestricted”

```
gateway:~# ./set_nat_type.sh portrestricted
gateway:~# 
```

- you should see iptables rules on `gateway` change
experiment 4 – port restricted

- start `nc` in server mode (and in background) on `host1`

```
host1:~# nc -l -p 100 &
host1:~#   
```

- attempt to contact it from `server2`

```
server2

root@server2:/root# nc 193.204.161.14 100
(UNKNOWN) [193.204.161.14] 100 (?) : Connection refused
```
experiment 4 – port restricted

- send some TCP traffic (a connection attempt is enough) from host1 to server2 and try again

```bash
host1:~# nc 193.204.161.103 999
(UNKNOWN) [193.204.161.103] 999 (?) : Connection refused
```

```bash
root@server2:/root# nc 193.204.161.14 100
(UNKNOWN) [193.204.161.14] 100 (?) : Connection refused
```
experiment 4 – port restricted

- the connection still fails, because the nat is enforcing a restriction also on the port
- if host1 sends a packet to server2’s port 999, then server2 can only initiate connections to host1 using source port 999
- let’s try again...

```
host1:~# nc 193.204.161.103 999
(UNKNOWN) [193.204.161.103] 999 (?) : Connection refused
```

```
server2
root@server2:/root# nc 193.204.161.14 100 -p 999
this is a test message
^C
```

- now it works!
experiment 4 – port restricted

- conclusion & exercises
  - an external host can initiate connections to an internal host only if it has been contacted by the internal host beforehand, and using as source port the port on which it has been contacted
  - try again using other combinations of source-destination hosts and ports
  - what happens if server2 attempts to connect to a different port on host1 (using always 999 as source port)?
experiment 5 – symmetric

- let’s check that packets from the same source host to different destinations are mapped to different public addresses
- go on host1 and ping 193.204.161.65 (server1), then 193.204.161.103 (server2)
- do the same from host2
- looking at the sniffer on server1 and server2 it can be seen that
  - host1 is mapped to 193.204.161.14 when contacting server1
  - host1 is mapped to 193.204.161.15 when contacting server2
  - host2 is mapped to 193.204.161.16 when contacting server1
  - host2 is mapped to 193.204.161.17 when contacting server2
- this implements the “first half” of the symmetric nat behavior
experiment 5 – symmetric

- The "second half" of the symmetric NAT behavior is to refuse unsolicited traffic entering the private network (i.e., only replies to packets that have been originated in the private network are sent back to the private network).

- To check this:
  - Use **nc** to try to make a connection from **host1** to **server1**: this should work.
  - Use **nc** to try to make a connection from **server1** to **host1**.
experiment 5 – symmetric

- conclusion & exercises
  - there is no way to initiate a connection from an external host to an internal one, because unsolicited traffic to private hosts is always rejected
  - check it by attempting different combinations of connections between the hosts and the servers