

IT Technology

Assignment 32 - OLA

Source nat on physical SRX



.....

University College

Author

Group 14

Bogdan-Dumitru Buterchi

bdbu37436@edu.ucl.dk

Thursday 10. Marts. 2022

Table of Contents

Contents

Table of Contents	0
Introduction	3
Learning goals	3
Tasks	3
Audience	4
Inventory	4
1. Networking diagram	5
2. Setting up the network.	9
3.Wireshark	11
4.Ping	11
5.Conclusion	12

Introduction

For assignment 32 we to set physical routing between PC3 and PC4, using two routers R2 and R3. Router 3 is the one that generates internet from school. Router 2 (R2) is configured default route and Nat based on the diagram, using Putty, injecting the security policies with the right IP that you can see in the diagram below.

Learning goals

How to build a network that allows Source Natting on a SRX router to be set up and tested. A network suggestion is shown above.

Tasks

- Draw the topology diagram with explanation.
- Configure default route and NAT on the R2 according to the diagram. The router should provide internet access for all possible hosts on networks 192.168.12.0/24 and 192.168.13.0/24. Please note that the 10.56.16.0 is a /22 network and not a /24 network. I.e. the /22 network mask is 255.255.252.0.
- Put the router configuration on GitLab and also the topology diagram. Post a working link here to the configuration and topology. It is obligatory to put relevant comments in the configuration.
- Demonstrate how these programs were possibly used for troubleshooting:
 - a. ping.
 - b. traceroute.
 - c. Wireshark.
- Run Wireshark and the ping command on PC3 or PC4 and show one screenshots/description. Comment on MAC addresses and IP addresses.
- On the SRX run the command:

```
show security flow session nat [brief | extensive | summary]
```

 - a. Only run ping 8.8.8.8 on PC3 and show the SRX output and explain it.
 - b. Also run a web browser and browse to e.g. dr.dk and show the SRX output and explain it.

Audience

Anybody looking to learn how to set up a physical server and connecting the security policies.

Inventory

- PC 1 and 2
- Physical router
- Putty
- Wireshark
- SRX router
- 7 RJ45 cables

1. Networking diagram

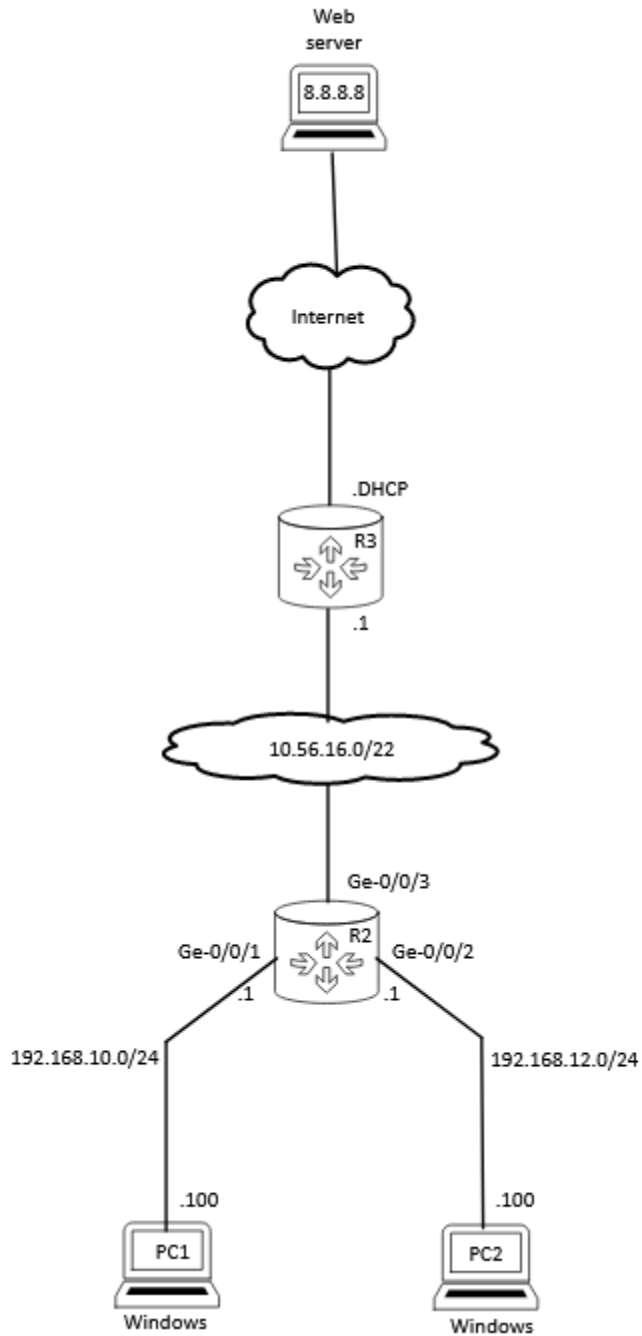


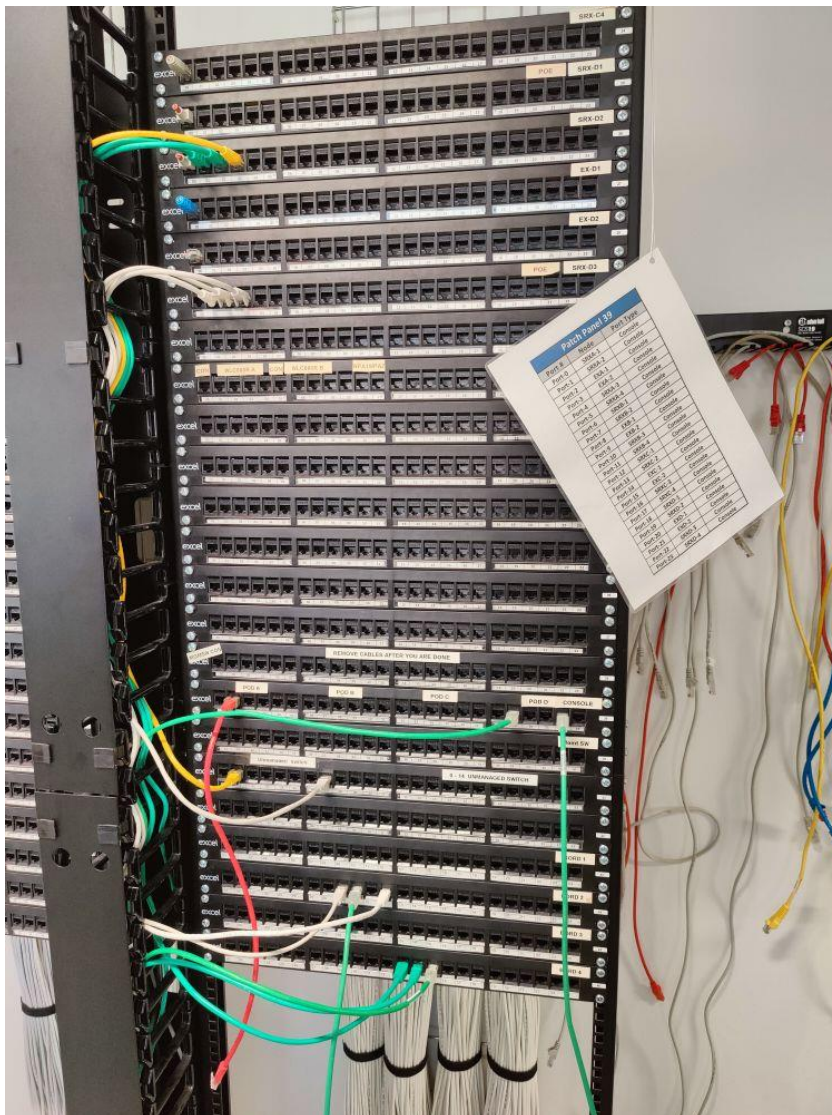
Figure 1: Networking diagram: Source nat on physical SRX

The PC1 and PC2 are connected to the R2 router that allows the sourcing NAT. As we can see in the diagram PC1 and PC2 are on separate LAN's but connected to the same router R2.

2.The networking setup

Devices	Port Table	SRX Interfaces
PC1	Port 13	Ge-0/0/1
PC2	Port 14	Ge-0/0/2
Router2 (SRX console port)	COM4	Serial port

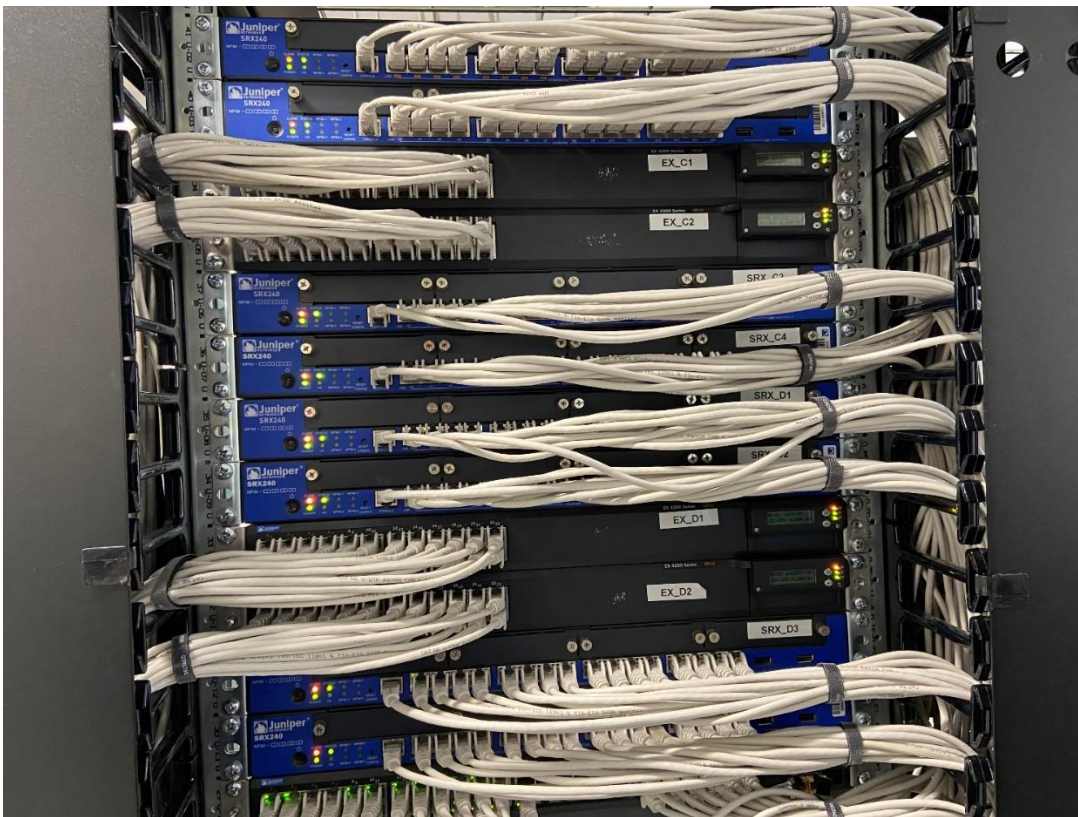
Table 1 – the physical network setup



Picture 1- physical network setup between our router 2 and router 3 (school). The wires for connecting the default route and Nat that we are using are with yellow and green.



Picture 2



Picture 3 – In this picture is an image with the school server room, we are using SRX_D2

The cabling was done first and after the SRX was set up using Putty.

2. Setting up the network.

To see how to set up a network with two subnets and one router, have a look at assignment 30¹

For configuring Putty we inject the code from Figure 2. If this is now the first time you set up the router, after you log in, enter “*cli*” and after “*edit*”, for entering edit mode, press “*load override terminal*”, this will allow you to inject the new code for configuring pc1 and pc2.

Once we have our basic network, we can edit the configuration to only allow traffic going from PC1 to PC2, we do this by putting each machine in its own security zone.

After that we had to define our policy, we want to permit traffic from the PC 1 zone to the PC 2 zone.

This is how the policies are going to look when defined in Figure 2.

```

security {
  nat {
    /* NAT changes the source address of egress IP packets */
    source {
      rule-set trust-to-untrust {
        from zone trust;
        to zone untrust;
        rule rule-any-to-any {
          match {
            source-address 0.0.0.0/0;
            destination-address 0.0.0.0/0;
          }
          then {
            source-nat {
              /* Use egress interface source IP address */
              interface;
            }
          }
        }
      }
    }
  }
}
policies {
  from-zone trust to-zone trust {
    policy default-permit {
      match {
        source-address any;
        destination-address any;
        application any;
      }
      then {
        permit;
      }
    }
  }
  from-zone untrust to-zone trust {
    policy default-permit {
      match {
        source-address any;
        destination-address any;
        application any;
      }
      then {
        permit;
      }
    }
  }
  from-zone trust to-zone untrust {
    policy internet-access {
      match {
        source-address any;
        destination-address any;
        application any;
      }
      then {
        permit;
      }
    }
  }
}
zones {
  security-zone trust {
    interfaces {
      ge-0/0/1.0 {
        host-inbound-traffic {
          system-services {
            ping;
          }
        }
      }
      ge-0/0/2.0 {
        host-inbound-traffic {
          system-services {
            ping;
          }
        }
      }
    }
  }
  security-zone untrust {
    interfaces {
      ge-0/0/3.0 {
        host-inbound-traffic {
          system-services {
            ping;
          }
        }
      }
    }
  }
}
}

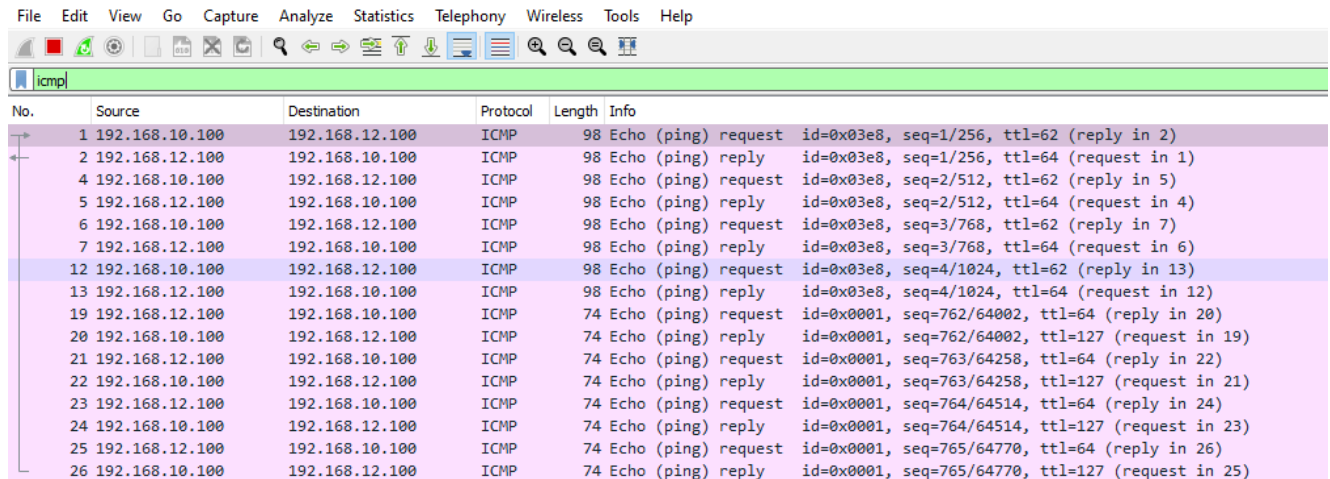
```

Figure 2: security policies for assignment 32

Full config here: https://gitlab.com/emil.privat/group_14/-/tree/main/Networking

3. Wireshark

Using Wireshark, we were able to troubleshoot to test the connectivity. By pinging with Wireshark, we can match the IP address from the one in the diagram in figure 1 with PC1 and PC2.



The image shows a Wireshark packet capture of ICMP ping traffic. The packet list table is as follows:

No.	Source	Destination	Protocol	Length	Info
1	192.168.10.100	192.168.12.100	ICMP	98	Echo (ping) request id=0x03e8, seq=1/256, ttl=62 (reply in 2)
2	192.168.12.100	192.168.10.100	ICMP	98	Echo (ping) reply id=0x03e8, seq=1/256, ttl=64 (request in 1)
4	192.168.10.100	192.168.12.100	ICMP	98	Echo (ping) request id=0x03e8, seq=2/512, ttl=62 (reply in 5)
5	192.168.12.100	192.168.10.100	ICMP	98	Echo (ping) reply id=0x03e8, seq=2/512, ttl=64 (request in 4)
6	192.168.10.100	192.168.12.100	ICMP	98	Echo (ping) request id=0x03e8, seq=3/768, ttl=62 (reply in 7)
7	192.168.12.100	192.168.10.100	ICMP	98	Echo (ping) reply id=0x03e8, seq=3/768, ttl=64 (request in 6)
12	192.168.10.100	192.168.12.100	ICMP	98	Echo (ping) request id=0x03e8, seq=4/1024, ttl=62 (reply in 13)
13	192.168.12.100	192.168.10.100	ICMP	98	Echo (ping) reply id=0x03e8, seq=4/1024, ttl=64 (request in 12)
19	192.168.12.100	192.168.10.100	ICMP	74	Echo (ping) request id=0x0001, seq=762/64002, ttl=64 (reply in 20)
20	192.168.10.100	192.168.12.100	ICMP	74	Echo (ping) reply id=0x0001, seq=762/64002, ttl=127 (request in 19)
21	192.168.12.100	192.168.10.100	ICMP	74	Echo (ping) request id=0x0001, seq=763/64258, ttl=64 (reply in 22)
22	192.168.10.100	192.168.12.100	ICMP	74	Echo (ping) reply id=0x0001, seq=763/64258, ttl=127 (request in 21)
23	192.168.12.100	192.168.10.100	ICMP	74	Echo (ping) request id=0x0001, seq=764/64514, ttl=64 (reply in 24)
24	192.168.10.100	192.168.12.100	ICMP	74	Echo (ping) reply id=0x0001, seq=764/64514, ttl=127 (request in 23)
25	192.168.12.100	192.168.10.100	ICMP	74	Echo (ping) request id=0x0001, seq=765/64770, ttl=64 (reply in 26)
26	192.168.10.100	192.168.12.100	ICMP	74	Echo (ping) reply id=0x0001, seq=765/64770, ttl=127 (request in 25)

Figure 3: ping between pc1 and pc2

4. Ping

For testing the connectivity and troubleshooting purposes we used the command line to ping PC2 and PC1.

```
Pinging 192.168.12.1 with 32 bytes of data:
Reply from 192.168.12.1: bytes=32 time<1ms TTL=64
Reply from 192.168.12.1: bytes=32 time<1ms TTL=64
Reply from 192.168.12.1: bytes=32 time<1ms TTL=64
Reply from 192.168.12.1: bytes=32 time<1ms TTL=64
```

Figure 4: ping between pc2 and pc1

For testing the internet we also pinged 8.8.8.8

```
$ ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=8ms TTL=107
Reply from 8.8.8.8: bytes=32 time=8ms TTL=107
Reply from 8.8.8.8: bytes=32 time=8ms TTL=107
Reply from 8.8.8.8: bytes=32 time=8ms TTL=107
```

Figure 5: ping 8.8.8.8

5. Conclusion

For this assignment we learned how to work with Putty for connecting physically. We set it up physically and showed how we made it worked. If we want we can expand the system by connecting more computers to the subnets.

¹ Reference to assignment 10



: Erhvervsakademi og
: Professionshøjskole

