Core Infrastructure and BGP Intro exercises

Henrik Kramselund Jereminsen
hlk@zencurity.com

September 9, 2021

The main exercise is the exercise 11 on page 17

The exercises requires your team to have a Debian 9 installed as described in exercise 10

The ones before are introducing a lot of tools that may help in debugging networks. Feel free to jump around. ☺

Contents

1 Bonus: Download Kali Linux Revealed (KLR) Book 10 min 3

2 Bonus: Check your Kali VM, run Kali Linux 30 min 4

3 Bonus: Wireshark and Tcpdump 15 min 5

4 Bonus: Capturing TCP Session packets 10 min 7

5 Bonus: Using ping and traceroute 10 min 9

6 Bonus: DNS and Name Lookups 10 min 11

7 Bonus: Zeek on the web 10min 12

8 Bonus: Zeek DNS capturing domain names 10min 13

9 Bonus: Zeek TLS capturing certificates 10min 15

10 Check your Debian VM 10 min 16

11 VLANs, Routing and RPF - 2h 17

12 Configuration of DHCP server 30min 23

13 Bonus: Configuration of Ubound DNS server 20min 25

14 Bonus: Configuration of BIRD BGP daemon 40min 27

15 How to Configure Mirror Port 10min 30

16 Learn about port security - 10 min 32

17 Bonus: SNMP walk 15min 34

18 Bonus: Monitoring - setup LibreNMS - 30 min 35

19 Bonus: IDS with Zeek and Suricata - 10min 37

20 Bonus: Nping check ports 10 min 38

21 Bonus: Try pcap-diff 15 min 40

22 Bonus: Discover active systems ping sweep 10 min 42

23 Bonus: Execute nmap TCP and UDP port scan 20 min 43

24 Bonus: Perform nmap OS detection 10 min 44

25 Bonus: Logning med syslogd og syslog.conf 10min 45

Preface

This material is prepared for use in *Core Infrastructure and BGP Intro workshop* and was prepared by Henrik Kramselund Jereminsen, http://www.zencurity.com . It describes the networking setup and applications for trainings and workshops where hands-on exercises are needed.

Further a presentation is used which is available as PDF from kramse@Github
Look for core-infrastructure-exercisesin the repo security-courses.

These exercises are expected to be performed in a training setting with network connected systems. The exercises use a number of tools which can be copied and reused after training. A lot is described about setting up your workstation in the repo

https://github.com/kramse/kramse-labs

Prerequisites

This material expect that participants have a working knowledge of TCP/IP from a user perspective. Basic concepts such as web site addresses and email should be known as well as IP-addresses and common protocols like DHCP.

Have fun and learn

Introduction to networking

IP - Internet protocol suite

It is extremely important to have a working knowledge about IP to implement secure and robust infrastructures. Knowing about the alternatives while doing implementation will allow the selection of the best features.

ISO/OSI reference model

A very famous model used for describing networking is the ISO/OSI model of networking which describes layering of network protocols in stacks.

This model divides the problem of communicating into layers which can then solve the problem as smaller individual problems and the solution later combined to provide networking.

Having layering has proven also in real life to be helpful, for instance replacing older hardware technologies with new and more efficient technologies without changing the upper layers.

In the picture the OSI reference model is shown along side with the Internet Protocol suite model which can also be considered to have different layers.

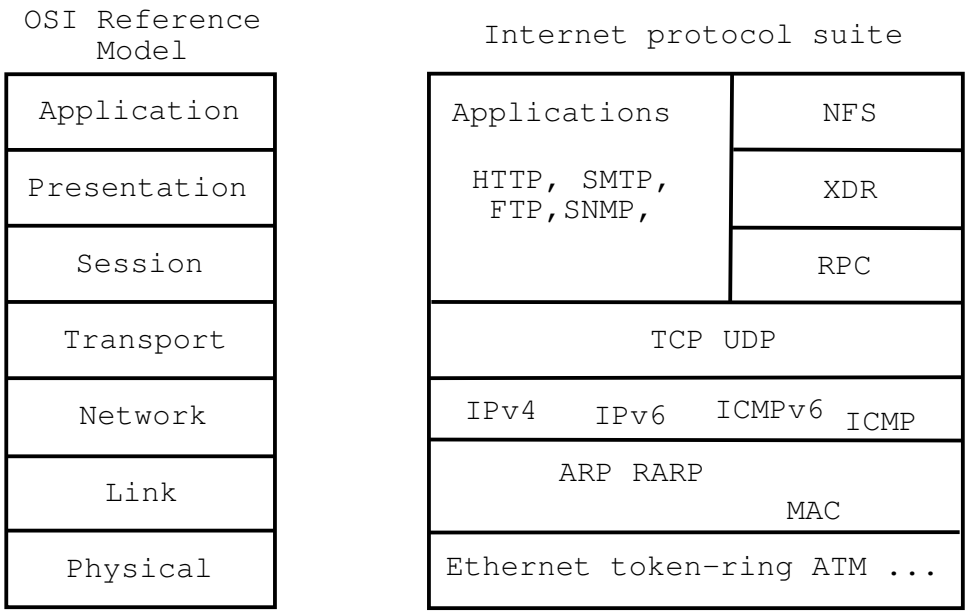


Figure 1: OSI og Internet Protocol suite

Exercise content

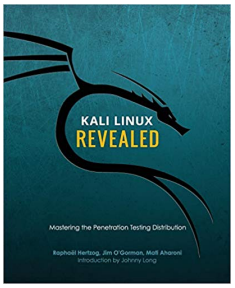
Most exercises follow the same procedure and has the following content:

- **Objective:** What is the exercise about, the objective
- **Purpose:** What is to be the expected outcome and goal of doing this exercise
- **Suggested method:** suggest a way to get started
- **Hints:** one or more hints and tips or even description how to do the actual exercises
- **Solution:** one possible solution is specified
- **Discussion:** Further things to note about the exercises, things to remember and discuss

Please note that the method and contents are similar to real life scenarios and does not detail every step of doing the exercises. Entering commands directly from a book only teaches typing, while the exercises are designed to help you become able to learn and actually research solutions.

Exercise 1

Bonus: Download Kali Linux Revealed (KLR) Book 10 min



Kali Linux Revealed Mastering the Penetration Testing Distribution

Objective:
We need a Kali Linux for running tools during the course. This is open source, and the developers have released a whole book about running Kali Linux.

This is named Kali Linux Revealed (KLR)

Purpose:
We need to install Kali Linux in a few moments, so better have the instructions ready.

Suggested method:
Create folders for educational materials. Go to https://www.kali.org/download-kali-linux-revealed-book/ Read and follow the instructions for downloading the book.

Solution:
When you have a directory structure for download for this course, and the book KLR in PDF you are done.

Discussion:
Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Kali Linux is a free pentesting platform, and probably worth more than $10.000

The book KLR is free, but you can buy/donate, and I recommend it.

Exercise 2

Bonus: Check your Kali VM, run Kali Linux 30 min

Objective:
Make sure your virtual machine is in working order.

We need a Kali Linux for running tools during the course.

Purpose:
If your VM is not installed and updated we will run into trouble later.

Suggested method:
Go to https://github.com/kramse/kramse-labs/

Read the instructions for the setup of a Kali VM.

Hints:
If you allocate enough memory and disk you wont have problems.

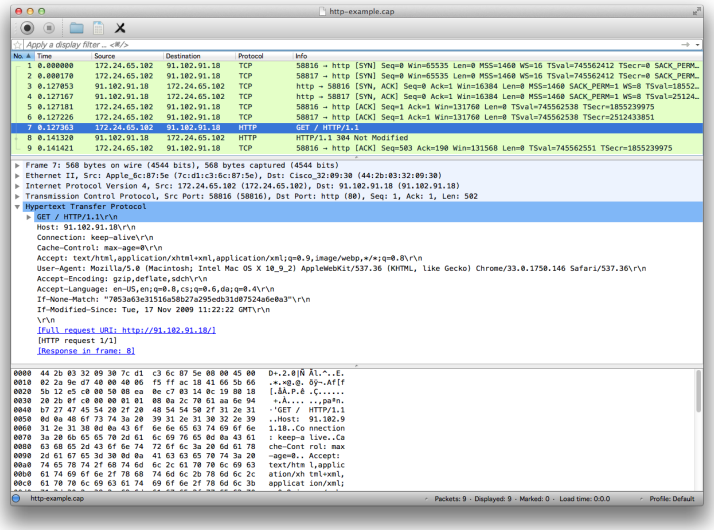
Solution:
When you have a updated virtualisation software and Kali Linux, then we are good.

Discussion:
Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Kali Linux includes many hacker tools and should be known by anyone working in infosec.

Exercise 3

Bonus: Wireshark and Tcpdump 15 min



Objective:
Try the program Wireshark locally your workstation, or tcpdump

You can run Wireshark on your host too, if you want.

Purpose:
Installing Wireshark will allow you to analyse packets and protocols

Tcpdump is a feature included in many operating systems and devices to allow packet capture and saving network traffic into files.

Suggested method:
Run Wireshark or tcpdump from your Kali Linux

The PPA book page 41 describes Your First Packet Capture.

Hints:
PCAP is a packet capture library allowing you to read packets from the network. Tcpdump uses libpcap library to read packet from the network cards and save them. Wireshark is a graphical application to allow you to browse through traffic, packets and protocols.

Both tools are already on your Kali Linux, or do: `apt-get install tcpdump wireshark`

Solution:
When Wireshark is installed sniff some packets. We will be working with both live traffic and saved packets from files in this course.

If you want to capture packets as a non-root user on Debian, then use the command to add a Wireshark group:

```
sudo dpkg-reconfigure wireshark-common
```

and add your user to this:

```
sudo gpasswd -a $USER wireshark
```

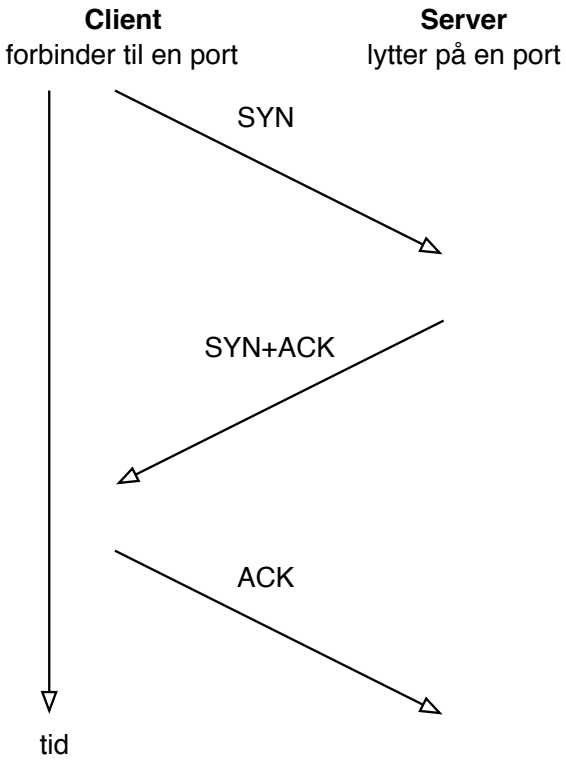
Dont forget to logout/login to pick up this new group.

Discussion:
Wireshark is just an example other packet analyzers exist, some commercial and some open source like Wireshark

We can download a lot of packet traces from around the internet, we might use examples from
https://www.bro.org/community/traces.html

Exercise 4

Bonus: Capturing TCP Session packets 10 min



Objective:
Sniff TCP packets and dissect them using Wireshark

Purpose:
See real network traffic, also know that a lot of information is available and not encrypted.

Note the three way handshake between hosts running TCP. You can either use a browser or command line tools like cURL while capturing

```
curl http://www.zencurity.com
```

Suggested method:
Open Wireshark and start a capture
Then in another window execute the ping program while sniffing

or perform a Telnet connection while capturing data

Hints:
When running on Linux the network cards are usually named eth0 for the first Ethernet and wlan0 for the first Wireless network card. In Windows the names of the network cards are long and if you cannot see which cards to use then try them one by one.

Solution:
When you have collected some TCP sessions you are done.

Discussion: Is it ethical to collect packets from an open wireless network?

Also note the TTL values in packets from different operating systems

Exercise 5

Bonus: Using ping and traceroute 10 min

Objective:
Be able to do initial debugging of network problems using commands ping and traceroute

Purpose:
Being able to verify connectivity is a basic skill.

Suggested method:
Use `ping` and `traceroute` to test your network connection - can be done one Windows and UNIX.

Hints:

```
$ ping 10.0.42.1
PING 10.0.42.1 (10.0.42.1) 56(84) bytes of data.
64 bytes from 10.0.42.1: icmp_seq=1 ttl=62 time=1.02 ms
64 bytes from 10.0.42.1: icmp_seq=2 ttl=62 time=0.998 ms
^C
--- 10.0.42.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.998/1.012/1.027/0.034 ms
```

Dont forget that UNIX ping continues by default, press ctrl-c to break.

Do the same with traceroute.

Solution:
Run both programs to local gateway and some internet address by your own choice.

Discussion:
Note the tool is called tracert on Windows, shortened for some reason.

ICMP is the Internet Control Message Protocol, usually used for errors like host unreachable. The ECHO request ICMP message is the only ICMP message that generates another.

The traceroute programs send packets with low Time To Live (TTL) and receives ICMP messages, unless there is a problem or a firewall/filter. Also used for mapping networks.

Bonus:

Whats the difference between:

- **traceroute** and **traceroute -I**
- NB: traceroute -I is found on UNIX - traceroute using ICMP pakker
- Windows tracert by default uses ICMP
- Unix by default uses UDP, but can use ICMP instead.
- Lots of traceroute-like programs exist for tracing with TCP or other protocols

Exercise 6

Bonus: DNS and Name Lookups 10 min

Objective:
Be able to do DNS lookups from specific DNS server

Purpose:
Try doing DNS lookup using different programs

Suggested method:
Try the following programs:

- nslookup - UNIX and Windows, but not recommended
  `nslookup -q=txt -class=CHAOS version.bind. 0`
- dig - syntax @server domain query-type query-class
  `dig @8.8.8.8 www.example.com`
- host - syntaks host [-l] [-v] [-w] [-r] [-d] [-t querytype] [-a] host [server]
  `host www.example.com 8.8.8.8`

Hints:
Dig is the one used by most DNS admins, I often prefer the host command for the short output.

Solution:
Shown inline, above.

Discussion:
The nslookup program does not use the same method for lookup as the standard lookup libraries, results may differ from what applications see.

What is a zone transfer, can you get one using the host command?

Explain forward and reverse DNS lookup.

Exercise 7

Bonus: Zeek on the web 10min

Objective:
Try Zeek Network Security Monitor - without installing it.

Purpose:
Show a couple of examples of Zeek scripting, the built-in language found in Zeek Network Security Monitor

Suggested method:
Go to http://try.bro.org/ and try a few of the examples.

Hints:
The exercise *The Summary Statistics Framework* can be run with a specifc PCAP.

```
192.168.1.201 did 402 total and 2 unique  requests in the last 6 hours.
```

Solution:
You should read the example *Raising a Notice*. Getting output for certain events may be interesting to you.

Discussion:
Zeek Network Security Monitor is an old/mature tool, but can still be hard to get started using. I would suggest that you always start out using the packages available in your Ubuntu/Debian package repositories.

They work, and will give a first impression of Zeek. If you later want specific features not configured into the binary packet, then install from source.

Also Zeek uses a broctl program to start/stop the tool, and a few config files which we should look at. From a Debian system they can be found in `/etc/bro` :

```
root@NMS-VM:/etc/bro# ls -la
drwxr-xr-x   3 root root  4096 Oct  8 08:36 .
drwxr-xr-x 138 root root 12288 Oct  8 08:36 ..
-rw-r--r--   1 root root  2606 Oct 30  2015 broctl.cfg
-rw-r--r--   1 root root   225 Oct 30  2015 networks.cfg
-rw-r--r--   1 root root   644 Oct 30  2015 node.cfg
drwxr-xr-x   2 root root  4096 Oct  8 08:35 site
```


Exercise 8

Bonus: Zeek DNS capturing domain names 10min

Objective:
We will now start using Zeek on our systems.

Purpose:
Try Zeek with example traffic, and see what happens.

Suggested method packet capture file:
Note: a dollar sign is the Linux prompt, showing the command after

```
$ cd
$ wget http://downloads.digitalcorpora.org/corpora/network-packet-dumps/2008-nitroba/nitroba.pcap
$ mkdir $HOME/bro;cd $HOME/bro; bro -r ../nitroba.pcap
... bro reads the packets
~/bro$ ls
conn.log  dns.log  dpd.log  files.log  http.log  packet_filter.log
sip.log  ssl.log  weird.log  x509.log
$ less *
```

Use :n to jump to the next file in less, go through all of them.

Suggested method Live traffic:
Make sure Zeek is configured as a standalone probe and configured for the right interface. Linux used to use eth0 as the first ethernet interface, but now can use others, like ens192 or enx00249b1b2991.

```
root@NMS-VM:/etc/bro# cat node.cfg
# Example BroControl node configuration.
#
# This example has a standalone node ready to go except for possibly changing
# the sniffing interface.

# This is a complete standalone configuration.  Most likely you will
# only need to change the interface.
[bro]
type=standalone
host=localhost
interface=eth0
...
```

Hints:

There are multiple commands for showing the interfaces and IP addresses on Linux. The old way is using `ifconfig -a` newer systems would use `ip a`

Note: if your system has a dedicated interface for capturing, you need to turn it on, make it available. This can be done manually using `ifconfig eth0 up` **Solution:**
When you either run Zeek using a packet capture or using live traffic

Running with a capture can be done using a command line such as: `bro -r traffic.pcap`

Using broctl to start it would be like this:

```
// install bro first
kunoichi:~ root# broctl
Hint: Run the broctl "deploy" command to get started.

Welcome to BroControl 1.5
Type "help" for help.

[BroControl] > install
creating policy directories ...
installing site policies ...
generating standalone-layout.bro ...
generating local-networks.bro ...
generating broctl-config.bro ...
generating broctl-config.sh ...
...


// back to Broctl and start it
[BroControl] > start
starting bro
// and then
kunoichi:bro root# cd /var/spool/bro/bro
kunoichi:bro root# tail -f dns.log
```

You should be able to spot entries like this:

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       proto   trans_i
     query   qclass  qclass_name     qtype   qtype_name      rcode   rcode_name      AA      TC      RD
1538982372.416180 CD12Dc1SpQm42QW4G3 10.xxx.0.145 57476 10.x.y.141 53 udp 20383 0.045021 www.dr.dk
1 C_INTERNET 1 A 0 NOERROR F F T T 0 www.dr.dk-v1.edgekey.net,e16198.b.akamaiedge.net,2.17.212.93
60.000000,20409.000000,20.000000 F
```

Note: this show ALL the fields captured and dissected by Zeek, there is a nice utility program bro-cut which can select specific fields:

```
root@NMS-VM:/var/spool/bro/bro# cat dns.log | bro-cut -d ts query answers | grep dr.dk
2018-10-08T09:06:12+0200 www.dr.dk www.dr.dk-v1.edgekey.net,e16198.b.akamaiedge.net,2.17.212.93
```

Discussion:
Why is DNS interesting?

Exercise 9

Bonus: Zeek TLS capturing certificates 10min

Objective:
Run more traffic through Zeek, see the various files.

Purpose:
See that even though HTTPS and TLS traffic is encrypted it often show names and other values from the certificates and servers.

Suggested method:
Run Zeek capturing live traffic, start https towards some sites. A lot of common sites today has shifted to HTTPS/TLS.

Hints:
use broctl start and watch the output directory

```
root@NMS-VM:/var/spool/bro/bro# ls *.log
communication.log  dhcp.log files.log known_services.log packet_filter.log  stats.log
stdout.log x509.log conn.log dns.log known_hosts.log loaded_scripts.log  ssl.log
stderr.log weird.log
```

We already looked at `dns.log`, now check `ssl.log` and `x509.log`

```
root@NMS-VM:/var/spool/bro/bro# grep dr.dk ssl.log
1538983060.546122 CtKYZ625cq3m3jUz9k 10.xxx.0.145 49932 2.17.212.93 443 TLSv12 TLS_ECDHE_RSA_WITH_AES_2
secp256r1 www.dr.dk F -h2 T FzmZCt3o9EYcmNaxIi,FKXcmxQHT3znDDMSj (empty) CN=*.dr.dk,O=DR,L=Copenhagen S
CN=GlobalSign Organization Validation CA - SHA256 - G2,O=GlobalSign nv-sa,C=BE --ok
1538983060.674217 CLjZo51fzuTcvPT0lg 200xxxxb:89b0:5cbf 49933 2a02:26f0:2400:2a1::3f46 443 TLSv12
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 secp256r1 asset.dr.dk F -h2 TFEpW9a1IFe6NTUZNpb,FwV50B4CHIwF1CPlu
(empty) CN=*.dr.dk,O=DR,L=Copenhagen S,ST=Copenhagen,C=DK CN=GlobalSign Organization Validation CA - SH
sa,C=BE --ok
```

Solution:
When you have multiple log files with data from Zeek, and have looked into some of them. You are welcome to ask questions and look into more files.

Discussion:
How can you hide that you are going to HTTPS sites?

Hint: VPN

Exercise 10

Check your Debian VM 10 min

Objective:
Make sure your virtual Debian 9 machine is in working order.

We need a Debian 9 Linux for running the router during the course.

Only one is needed per team that want to do the exercises.

Purpose:
If your VM is not installed and updated we will run into trouble later.

Suggested method:
Go to https://github.com/kramse/kramse-labs/

Read the instructions for the setup of a Debian VM.

Hints:
Sometimes having a router with a GUI can help, but today we do it the most basic way. This ensure we have a good understanding of precisely which parts we have changed.

Also Tcpdump can show what packets are sent and received.

Solution:
When you have a updated virtualisation software and Kali Linux, then we are good.

Discussion:
Linux is free and everywhere. The tools we will run in this course are made for Unix, so they run great on Linux.

Exercise 11

VLANs, Routing and RPF - 2h

Objective:
Configure a network and connect it to the local internet exchange. We will learn about IP routing, and internet exchanges. We will also be able to configure blocking and firewall rules.

Note: it is recommended to do this in teams with two laptops. One laptop stays connected to the local wifi with internet access, for searching. The other one runs a Debian server.

The one running a Debian server will be configured as a router! YMMV but having an USB Ethernet adapter that can be dedicated and inserted into the VM will probably be the most efficient and ensure that VLAN tagging - required - are working.

You MAY try using the built-in Ethernet in your laptop for this purpose and use *bridging* to connect it to the network.

Note: the network does NOT need NAT! You can use NAT on your router, but it is recommended NOT to. If you enable NAT on your router it will be harder to connect back into your network - need to use features like port forwarding.

Purpose:
The network built for this training allows you to configure network devices without affecting production networks. You can make all the mistakes and no manager will ask you to write incident reports.

Your team will have a prefix of IPv4 /16, example 10.14.0.0/16. The same ID 14 in this example is used for other purposes too.

All your devices must end up in this network.

There are multiple networks including the other teams, all are in 10/8.

The uplink expects your connection to the rest of the networks are done with a switch using the internet exchange model with a peering network of 10.10.10.0/24. The core network will initially have a static route pointing all of your prefix to your peering address.

Your router will need an interface in this range, to be able to communicate to the internet and the other networks.

Suggested method:

Follow the instructions below for each part. Use your team and ask questions - learn.

There are some Ansible scripts in the Github repository `https://github.com/kramse/kramse-labs` in the directory core-net-lab.

Recommend doing a git clone FIRST, before changing network configuration of the Virtual Machine.

As always there might be some tuning and learning to do on the way.

This is why I have included a lot of bonus exercises in this exercise book.

Enable routing

Make sure routing is actually enabled on your device! Sysctl will show the current setting

Use your favourite editor and activate the line in /etc/sysctl.conf:

```
user@debian-9-lab:~$ grep forward /etc/sysctl.conf
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Note: requires a reboot to pick up this change, or set using sysctl.

Changing it can be done using sysctl or echo:

```
# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1

# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Check using:

```
# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

also install the `vlan` package using: `apt-get install vlan`

This can of course be automated using Ansible, with this:

```
  # Set ip forwarding on in /proc and verify token value with the sysctl command
```



```
  - sysctl:
      name: net.ipv4.ip_forward
      value: '1'
      sysctl_set: yes

  # Set ip forwarding on in /proc and in the sysctl file and reload if necessary
  - sysctl:
      name: net.ipv4.ip_forward
      value: '1'
      sysctl_set: yes
      state: present
      reload: yes
```

Network Cards

Before configuring your Debian as a router, make sure to remove / deactivate any network cards to your virtualisation host

You should only have the USB Ethernet configured, as a USB device in the virtual machine. Name may vary - on my server it became: `enx00249b1b2991`

So your configuration should be similar to this:

```
root@NMS-VM:~# ifconfig  -a
enx00249b1b2991: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 00:24:9b:1b:29:91  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1355  bytes 556600 (543.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1355  bytes 556600 (543.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Initial steps to get started

Start configuring the switch you have. We can collect logs about network traffic based on generic netflow, specific types DNS/TLS/protocols, and traffic matching advanced rulesets.

Recommend resetting it first time!

also during configuration, remember to **save config** once in a while.

1. Cable a laptop to any port, except the first one - port 1/0/1
2. Take control of the switch, default settings are in place, see label on bottom and use Ethernet preferably
3. Add VLAN with ID: 2770 Name: KramsIX
4. Configure the Uplink port with VLAN tag: 2770
5. Configure the first port with VLAN tag: 2770
   If you keep the port in VLAN 1 you should be able to connect to the switch management via this port using tagged or untagged, as you prefer ☺
6. Configure VLAN interface on your router with the peering IP in the peering LAN 10.10.10.0/24 - if you team ID is 14, then your peering IP is 10.10.10.14/24
7. Ping 10.10.10.1 from your router
8. Add 10.10.10.1 as your default gateway
   Dont worry about

When your router can ping 10.10.10.1, and possibly the outside world - congratulate yourself! You have just used VLAN tagging successfully!

If your other laptop can also ping your router from the training network, awesome!

Tools to use for debugging, traceroute, nping, arp

PS The "peering LAN" uses tags to avoid the multiple switches with the same initial configuration to interfere. Only allowing VLAN tagged packets ensure only "configured devices" are let in. Real internet exchanges have lots of rules about which devices you may connect and which traffic is allowed.

Your network config should look something like this:

```
$ cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto enx00249b1b2991
iface enx00249b1b2991 inet static
  address 192.168.0.2
  netmask 255.255.255.0

auto vlan2770
iface vlan2770 inet static
vlan-raw-device enx00249b1b2991
address 10.10.10.14
netmask 255.255.255.0
  gateway 10.10.10.1
```

Note: you should probably also add an IP for the management of the router as shown. If you do NOT activate the base network card, the VLAN sub interfaces will not come online!

Implement your VLAN and IP plan

Now that your router is connected continue by adding your VLANs and IPs.

1. Add your VLANs using the names from your plan.
   If you have no plan then use VLAN 100 LAN, VLAN 200 Wifi
2. Add ports to your VLAN - may be done while defining them too.
   You may have a VLAN 100 which is intended for LAN traffic then add this as untagged on a port and tagged to the router port 1/0/1
3. Add your VLAN interfaces and IPs to the router
   Again, if VLAN 100 is the LAN, add an interface to router with this tag and IP 10.14.100.1 - some prefer the first for routers, others the last .254

4. When adding the interface and port for the Wifi - make the port towards the AP untagged in your VLAN initially. The wifi APs are configured with only one SSID and will just bridged this traffic to the Ethernet port.
   Using VLAN tagging for the wifi, multiple SSIDs require reconfiguration of the wifi AP - you can do this though.

Check using laptop that you can ping your router, before trying to ping other parts of the network. From the router you should also provide DHCP service - to avoid the clients needing a static IP.

If you are new to routing it may be recommended to have no firewall rules initially. If the routing parts work without trouble then enable firewall.

Advanced variants

1. Remove the default route from your router, replace by a BGP connection to the router 10.10.10.1 using BGP with default settings, no password
   When the BGP connection is working, ask trainer to remove static route
2. Reverse Path Forwarding (RPF) check can be added using Access Control Lists ACLs
3. Run a DMZ with a web server and block everything except 80/tcp and 443/tcp

Hints:
Use your team, search the internet, ask for help.

This is a complex routing exercise, if you are new to routing, take it easy.

Solution:
When you have a network that allows you to surf from inside your own network and one that allows traffic from the training network into your network - you are done.

Discussion:
How much in this exercise is similar to real networks?

I would say a lot. The exercise emulates a large part of the technologies used in real networks, including layer 3 routing having a peering internet exchange and a transit provider. Also the lab networks allows for playing with layer 2 features such as port security settings.

Exercise 12

Configuration of DHCP server 30min

Objective:
Configure DHCPD

Purpose:
Getting DHCPD configured ensures that normal people can connect devices easily.

Suggested method:
Look into dhcpd.conf and configure address pool for clients.

Hints:
DHCPD is the DHCP daemon, helping spirit.

It is configured using a small config file: `/etc/dhcp/dhcpd.conf`

Solution:
You should hopefully end up with a running configuration which allows clients to get an IP and configuration settings.

Example running config `/etc/dhcp/dhcpd.conf`:

```
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#

# option definitions common to all supported networks...
option domain-name "kramse.org";
option domain-name-servers 91.239.100, 89.233.43.71 ;

# These are from https://blog.uncensoreddns.org/
# 91.239.100.100  2001:67c:28a4::
# 89.233.43.71  2a01:3a0:53:53::

default-lease-time 600;
max-lease-time 7200;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
#log-facility local7;

# This is a very basic subnet declaration.

subnet 10.14.0.0 netmask 255.255.255.0 {
  range 10.14.0.10 10.14.0.200;
  option routers 10.14.0.1;
  #  option domain-name-servers ns1.internal.example.org;
  #  option domain-name "internal.example.org";
}
```

Discussion:
This service is needed in almost any network.

With IPv4 DHCPD is the recommended way.

With IPv6 sometimes a mix of stateless autoconfiguration and DHCPDv6 is used.

Exercise 13

Bonus: Configuration of Ubound DNS server 20min

Objective:
Learn how to run your own resolver.

Purpose:
DNS is used by all clients today. So we can configure our own DNS resolver, DNS server.

Suggested method:
Install Unbound which is the software from `https://nlnetlabs.nl/projects/unbound/about/`

Hints:
Use the package system:

```
sudo apt install unbound
```

Make sure it is enabled for boot:

```
root@debian-9-lab:~# systemctl enable unbound
Synchronizing state of unbound.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable unbound
```

Solution:
By default it seems the Debian Unbound has very little configuration, but allows lookup from localhost:

```
root@debian-9-lab:~# host www.kramse.org
www.kramse.org has address 185.129.60.130
www.kramse.org has IPv6 address 2a06:d380:0:3065::80
```

It would be required to add a little more, starting with access control:

```
# The server clause sets the main parameters.
server:
        interface: 0.0.0.0
        access-control: 127.0.0.0/8 allow
        access-control: 10.0.0.0/16 allow
        access-control: 192.168.0.0/16 allow
        access-control: 10.0.45.0/24 allow
        access-control: 8.0.0.0/8 allow
```

This allows local clients access to resolving names.

Discussion:
Unbound is very feature complete and can perform DNSSEC checks etc. We wont be able to configure and use all these features today.

Exercise 14

Bonus: Configuration of BIRD BGP daemon 40min

Objective:
Learn how to run your BGP service, to announce your network.

Purpose:
Border Gateway Protocol (BGP) is a very simple yet complicated technology.

```
https://en.wikipedia.org/wiki/Border_Gateway_Protocol
```

Today just realize:

- BGP allows you to tell a network about your IP addresses, named IP prefixes - similar to subnets, like 10.14.0.0/16
- BGP allows you to receive network prefixes and add them to your local route table.

The internet is built using BGP and in the core networks, Tier 1 networks there is no default route - no 0.0.0.0 route.

Suggested method:
Install BIRD which is the software from `https://bird.network.cz/`

Hints:
Use the package system:

```
sudo apt install bird
```

Make sure it is enabled for boot:

```
root@debian-9-lab:~# systemctl enable bird
```

Solution:
When you install BIRD is has a minimal configuration.

You should first make sure it runs

NOTE: This material expect the version 1.6 of BIRD, so use the documentation from:
`https://bird.network.cz/?get_doc&f=bird.html&v=16`

Method:

- Make sure process runs, syntax is correct - check logs
- Make sure it connects to the other party, BGP is 179/TCP
- Make sure you are sending and receiving the right prefixes
- Make sure filters are in place - not needed her, but for production setups!

Many times a router will NOT announce if the route/prefix is not active in the routing tables on the local system. Making what is called a null route is also recommended, so start out by null-routing your whole network locally.

This also prevents a lot of routing loops!

The end result might be similar to this:

```
# This is a minimal configuration file, which allows the bird daemon to start
# but will not cause anything else to happen.
#
# Please refer to the documentation in the bird-doc package or BIRD User's
# Guide on http://bird.network.cz/ for more information on configuring BIRD and
# adding routing protocols.

# Change this into your BIRD router ID. It's a world-wide unique identification
# of your router, usually one of router's IPv4 addresses.
router id 10.14.0.1;

# The Device protocol is not a real routing protocol. It doesn't generate any
# routes and it only serves as a module for getting information about network
# interfaces from the kernel.
protocol device {
}

# The Kernel protocol is not a real routing protocol. Instead of communicating
# with other routers in the network, it performs synchronization of BIRD's
# routing tables with the OS kernel.
protocol kernel {
        metric 64;        # Use explicit kernel route metric to avoid collisions
                          # with non-BIRD routes in the kernel routing table
  import all;
  export all;        # Actually insert routes into the kernel routing table
}

protocol bgp {
        description "My BGP uplink";
        local as 65014;
        neighbor 10.10.10.1 as 65010;
        multihop;
        next hop self;                # Disable next hop processing and always advertise our local address as nexthop
#        source address 198.51.100.14;        # What local address we use for the TCP connection
#        password "secret";        # Password used for MD5 authentication
        export where source=RTS_STATIC;
#        export filter {
#                if source = RTS_STATIC then {
#                        bgp_community = -empty-; bgp_community = add(bgp_community,(65000,5678));
#                        bgp_origin = 0;
#                        bgp_community = -empty-; bgp_community.add((65000,5678));
#                        if (65000,64501) ~ bgp_community then
#                                bgp_community.add((0, 1));
#                        if bgp_path ~ [= 65000 =] then
#                                bgp_path.prepend(65000);
#                        accept;
#                }
#                reject;
#        };
}
```

or This

```
log stderr all;
router id 10.9.8.20;
debug protocols all;

filter rfc1918 {
        if net ~ 192.168.0.0/16 then accept;
        if net ~ 172.16.0.0/12 then accept;
        if net ~ 10.0.0.0/8 then accept;
        else reject;
}

protocol kernel {
        export filter rfc1918;
}

protocol device {}

protocol static {
        route 10.20.0.0/16 via 10.9.8.20;
}

filter not_default {
        if net = 0.0.0.0/0 then reject;
        else accept;
}

protocol bgp {
        description "Camp-IX";
        local as 65520;
        neighbor 10.9.8.1 as 65512;
        export all;
        import filter not_default;
}
```

Discussion:
The BIRD project aims to develop a fully functional dynamic IP routing daemon primarily targeted on (but not limited to) Linux, FreeBSD and other UNIX-like systems and distributed under the GNU General Public License. - source: `https://bird.network.cz/`

BIRD is very feature complete and can perform filtering checks etc. We wont be able to configure and use all these features today.

Exercise 15

How to Configure Mirror Port 10min

Objective:
Mirror ports are a way to copy traffic to Suricata and other devices - for analyzing it. We will go through the steps on a Juniper switch to show how. Most switches which are configurable have this possibility.

Purpose:
We want to capture traffic for multiple systems, so we select an appropriate port and copy the traffic. In our setup, we select the uplink port to the internet/router.

It is also possible to buy passive taps, like a fiber splitter, which then takes part of the signal, and is only observable if you look for signal strength on the physical layer.

Suggested method:
We will configure a mirror port on a Juniper EX2200-C running Junos.

```
root@ex2200-c# show ethernet-switching-options | display set
set ethernet-switching-options analyzer mirror01 input ingress interface ge-0/1/1.0
set ethernet-switching-options analyzer mirror01 input egress interface ge-0/1/1.0
set ethernet-switching-options analyzer mirror01 output interface ge-0/1/0.0
set ethernet-switching-options storm-control interface all
```

Then configure source ports, the ones in current use and destination to the selected high port.

If using the TP-Link T1500G-10PS then this link should describe the process:
`https://www.tp-link.com/en/configuration-guides/mirroring_traffic/?configurationId=18210`

Which describe:

1. Choose the menu MAINTENANCE > Mirroring
2. Select Edit for the Mirror Session 1
3. In the Destination Port Config section, specify a destination port for the mirroring session, and click Apply
4. In the Source Interfaces Config section, specify the source interfaces and click Apply

Using the command line would be similar to this:


```
Switch#configure
Switch(config)#monitor session 1 destination interface gigabitEthernet 1/0/7
Switch(config)#monitor session 1 source interface gigabitEthernet 1/0/1-4 both
Switch(config)#monitor session 1 source cpu 1 both
```

The method is very similar across vendors, `monitor session` is often the same command as shown above.

Hints:
Selecting a port away from the existing ones allow easy configuration of the source to be like, Source ports 1-4 and destination 7 - and easily expanded when port 5 and 6 are activated.

When checking your own devices this is often called SPAN ports, Mirror ports or similar.

`https://en.wikipedia.org/wiki/Port_mirroring`

Cisco has called this Switched Port Analyzer (SPAN) or Remote Switched Port Analyzer (RSPAN), so many will refer to them as SPAN-ports.

Solution:
When we can see the traffic from the network, we have the port configured - and can run any tool we like. Note: specialized capture cards can often be configured to spread the load of incoming packets onto separate CPU cores for performance. Capturing 100G and more can also be done using switches like the example found on the Zeek web site using an Arista switch 7150.

Cisco also has a feature named RSPAN

> Remote SPAN (RSPAN): An extension of SPAN called remote SPAN or RSPAN. RSPAN allows you to monitor traffic from source ports distributed over multiple switches, which means that you can centralize your network capture devices. RSPAN works by mirroring the traffic from the source ports of an RSPAN session onto a VLAN that is dedicated for the RSPAN session. This VLAN is then trunked to other switches, allowing the RSPAN session traffic to be transported across multiple switches. On the switch that contains the destination port for the session, traffic from the RSPAN session VLAN is simply mirrored out the destination port.

Source: https://community.cisco.com/t5/networking-documents/understanding-span-rspan-and-erspan/ta-p/3144951

Discussion:
When is it ethical to capture traffic?

Exercise 16

Learn about port security - 10 min

Objective:
See the options available for port security on the local switch

Purpose:
Even small cheap switches like the ones used in the training have options that can help protect a network. Such as:

- Max Learned Number of MAC

  Specify the maximum number of MAC addresses that can be learned on the port. When the learned MAC address number reaches the limit, the port will stop learning. It ranges from 0 to 64. The default value is 64.

- Exceed Max Learned Trap

  Enable Exceed Max Learned, and when the maximum number of learned MAC addresses on the specified port is exceeded, a notification will be generated and sent to the management host.

- Learn Address Mode

  Select the learn mode of the MAC addresses on the port. Three modes are provided:

  Delete on Timeout: The switch will delete the MAC addresses that are not used or updated within the aging time. It is the default setting.

  Delete on Reboot: The learned MAC addresses are out of the influence of the aging time and can only be deleted manually. The learned entries will be cleared after the switch is rebooted.

  Permanent: The learned MAC addresses are out of the influence of the aging time and can only be deleted manually. The learned entries will be saved even the switch is rebooted.

- Status when exceeded
  Drop: When the number of learned MAC addresses reaches the limit, the port will stop learning and discard the packets with the MAC addresses that have not been learned.

  Forward: When the number of learned MAC addresses reaches the limit, the port will stop learning but send the packets with the MAC addresses that have not been learned.

Suggested method:
Visit the settings page, configure port security and try running the `macof` mac overflow tool from a laptop connected on that port.

Hints:
There may be a configuration guide at:
`https://www.tp-link.com/en/configuration-guides/configuring_port_security/?configurationId=18227`

Solution:
When you have seen the setting, either on the available devices, or searched the internet and found the options available on your favourite device, you are done.

Discussion:
Port security is often neglected, so a small cabling issue in a single device in an office may take down the whole office location network.

Think about port security when installing new networks, and turn it on immediately before allowing people to connect, as most managers are wary about enabling it afterwards.

Exercise 17

Bonus: SNMP walk 15min

Objective:
Run SNMP walk on a switch, `snmpwalk` see information from device.

Lots of basic information helps defenders - AND attackers.

Purpose:
SNMP is a default management protocol used in almost any network.

Suggested method:
Enable SNMPv2 with a community of "public".

Run snmpwalk using community string public. Command is: `snmpwalk -v 2c -c public system`

Hints:
Community strings private and public used to be default for network devices. Today professionel devices have no SNMP configured by default, but lots of networks install the community "public" anyway.

Solution:
When you have done snmpwalk for at least one device.

Discussion:
Which part of the information is most relevant to defenders, and attackers.

Also remember on internal LAN segments, use Nmap:

```
snmp-10.x.y.0.gnmap: nmap -sV -A -p 161 -sU --script=snmp-info -oA snmp-10xy 10.x.y.0/19
snmpscan: nmap -sU -p 161 -oA snmpscan --script=snmp-interfaces -iL targets
```


Exercise 18

Bonus: Monitoring - setup LibreNMS - 30 min

Objective:
See and work with LibreNMS - for a little while.

Purpose:
Show that initial setup of LibreNMS is quite easy, and immediately gives some results.

Suggested method:
Choose installation method from:
https://docs.librenms.org/Installation/

Prebuilt docker and VM images will be fastest.

Hints:
As they themselves write on the web page https://www.librenms.org/

> LibreNMS is a fully featured network monitoring system that provides a wealth of features and device support.

If you feel this is too much work, ask for login to the central one used for the core network.

Solution:
When you have played around with LibreNMS you are done.

It is recommended to look at:

- How basic information about devices are presented, from devices when added - and nothing more.
  See how to add a device and add your own. https://docs.librenms.org/Support/Adding-a-Device/
- How SNMP location is used to categorize devices and provice maps, see https://docs.librenms.org/Extensions/World-Map/
- How protocols like LLDP allow LibreNMS to make maps, see https://docs.librenms.org/Extensions/Network-Map/
- How port description can be used for describing ports, https://docs.librenms.org/Extensions/Interface-Description-Parsing/

Most of this happens with very little effort. Just configure devices consistently and they will be presented nicely.

Discussion:
Have you configured port description? Try it!

Real networks have policies for port description settings.

Exercise 19

Bonus: IDS with Zeek and Suricata - 10min

Objective:
Adding an IDS to your network will help a lot with insights into traffic patterns, and bad traffic.

I recommend the Suricata and Zeek software and use it to monitor your traffic.

Purpose:
Discuss the concept of IDS and difference between Suricata and Zeek.

Suggested method:
Configure a mirror port - select a high numbered port not in use.

Use a laptop on the port to verify you get copies of packets on this port.

Then configure a Suricata or Zeek, maybe this course materials can help:
`https://github.com/kramse/security-courses/tree/master/courses/networking/suricatazeek-worksho`

Hints:

Solution:
When your team has configured a mirror port and seen traffic using your IDS or just tcpdump you are done.

Discussion:
How would you monitor a larger network?

Exercise 20

Bonus: Nping check ports 10 min

Objective:
Show the use of Nping tool for checking ports through a network

Purpose:
Nping can check if probes can reach through a network, reporting success of failure. Allows very specific packets to be sent.

Suggested method:
Run the command using a common port like Web HTTP:

```
root@KaliVM:~# nping --tcp -p 80 www.zencurity.com

Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2018-09-07 19:06 CEST
SENT (0.0300s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (0.0353s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=49674 iplen=44  seq=3654597698 win=16384 <mss
SENT (1.0305s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (1.0391s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=50237 iplen=44  seq=2347926491 win=16384 <mss
SENT (2.0325s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (2.0724s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=9842 iplen=44  seq=2355974413 win=16384 <mss 1
SENT (3.0340s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (3.0387s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=1836 iplen=44  seq=3230085295 win=16384 <mss 1
SENT (4.0362s) TCP 10.137.0.24:3805 > 185.129.60.130:80 S ttl=64 id=18933 iplen=40  seq=2984847972 win=1480
RCVD (4.0549s) TCP 185.129.60.130:80 > 10.137.0.24:3805 SA ttl=56 id=62226 iplen=44  seq=3033492220 win=16384 <mss

Max rtt: 40.044ms | Min rtt: 4.677ms | Avg rtt: 15.398ms
Raw packets sent: 5 (200B) | Rcvd: 5 (220B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 4.07 seconds
```

Hints:
A lot of options are similar to Nmap

Solution:
When you have tried it towards an open port, a closed port and an IP/port that is filtered you are done.

Discussion:
A colleague of ours had problems sending specific IPsec packets through a provider. Using a tool like Nping it is possible to show what happens, or where things are blocked.

Things like changing the TTL may provoke ICMP messages, like this:

```
root@KaliVM:~# nping --tcp -p 80 --ttl 3 www.zencurity.com

Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2018-09-07 19:08 CEST
SENT (0.0303s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (0.0331s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=28456 iplen=7
SENT (1.0314s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (1.0337s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=28550 iplen=7
SENT (2.0330s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (2.0364s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=28589 iplen=7
SENT (3.0346s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (3.0733s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=29403 iplen=7
SENT (4.0366s) TCP 10.137.0.24:37244 > 185.129.60.130:80 S ttl=3 id=60780 iplen=40  seq=1997801125 win=1480
RCVD (4.0558s) ICMP [10.50.43.225 > 10.137.0.24 TTL=0 during transit (type=11/code=0) ] IP [ttl=62 id=30235 iplen=7

Max rtt: 38.574ms | Min rtt: 2.248ms | Avg rtt: 13.143ms
Raw packets sent: 5 (200B) | Rcvd: 5 (360B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 4.07 seconds
```


Exercise 21

Bonus: Try pcap-diff 15 min

Objective:
Try both getting an utility tool from Github and running an actual useful tool for comparing packet captures.

Purpose:
Being able to get tools and scripts from Github makes you more effective.

The tool we need today is https://github.com/isginf/pcap-diff **Suggested method:**
Git clone the repository, follow instructions for running a packet diff.

Try saving a few packets in a packet capture, then using tcpdump read and write a subset - so you end up with two packet captures:

```
sudo tcpdump -w icmp-dump.cap
// run ping in another window, which probably creates ARP packets
// Check using tcpdump
sudo tcpdump -r icmp-dump.cap arp
reading from file icmp-dump.cap, link-type EN10MB (Ethernet)
10:06:18.077055 ARP, Request who-has 10.137.0.22 tell 10.137.0.6, length 28
10:06:18.077064 ARP, Reply 10.137.0.22 is-at 00:16:3e:5e:6c:00 (oui Unknown), length 28
10:06:24.776987 ARP, Request who-has 10.137.0.6 tell 10.137.0.22, length 28
10:06:24.777107 ARP, Reply 10.137.0.6 is-at fe:ff:ff:ff:ff:ff (oui Unknown), length 28
// Write the dump - but without the ARP packets:
sudo tcpdump -r icmp-dump.cap -w icmp-dump-no-arp.cap not arp
```

With these pcaps you should be able to do:

```
sudo pip install scapy
git clone https://github.com/isginf/pcap-diff.git
cd pcap-diff/

$ python pcap_diff.py -i ../icmp-dump.cap -i ../icmp-dump-no-arp.cap -o diff.cap
Reading file ../icmp-dump.cap:
Found 23 packets

Reading file ../icmp-dump-no-arp.cap:
Found 19 packets

Diffing packets:

Found 2 different packets

Writing diff.cap
// Try reading the output packet diff:

$ sudo tcpdump -r diff.cap
```



```
reading from file diff.cap, link-type EN10MB (Ethernet)
10:06:24.777107 ARP, Reply 10.137.0.6 is-at fe:ff:ff:ff:ff:ff (oui Unknown), length 28
10:06:24.776987 ARP, Request who-has 10.137.0.6 tell 10.137.0.22, length 28
```

Note: I ran these on a Debian, so I needed the sudo, if you run this on Kali there is no need to use sudo.

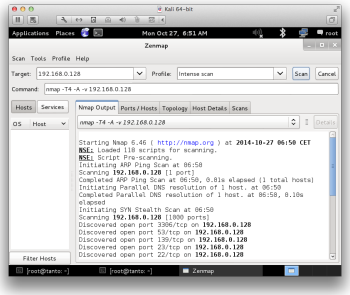
Hints:
Git is one of the most popular software development tools, and Github is a very popular site for sharing open source tools.

Solution:
When you or your team mate has a running pcap-diff then you are done

Discussion:
I often find that 90% of my tasks can be done using existing open source tools.

Exercise 22

Bonus: Discover active systems ping sweep 10 min



Objective:
Use nmap to discover active systems

Purpose:
Know how to use nmap to scan networks for active systems.

Suggested method:
Try different scans,

- Ping sweep to find active systems
- Port sweeps to find active systems with specific ports

Hints:
Try nmap in sweep mode - and you may run this from Zenmap

Solution:
Use the command below as examples:

- Ping sweep `nmap -sP 10.0.45.*`
- Port sweeps `nmap -p 80 10.0.45.*`

Discussion:
Quick scans quickly reveal interesting hosts, ports and services

Also now make sure you understand difference between single host scan 10.0.45.123/32, a whole subnet /24 250 hosts 10.0.45.0/24 and other more advanced targeteting like 10.0.45.0/25 and 10.0.45.1-10

Exercise 23

Bonus: Execute nmap TCP and UDP port scan 20 min

Objective:
Use nmap to discover important open ports on active systems

Purpose:
Finding open ports will allow you to find vulnerabilities on these ports.

Suggested method:
Use `nmap -p 1-1024 server` to scan the first 1024 TCP ports and use Nmap without ports. What is scanned then?

Try to use `nmap -sU` to scan using UDP ports, not really possible if a firewall is in place.

If a firewall blocks ICMP you might need to add `-Pn` to make nmap scan even if there are no Ping responses

Hints:
Sample command: `nmap -Pn -sU -p1-1024 server` UDP port scanning 1024 ports without doing a Ping first

Solution:
Discover some active systems and most inex:basicDebianVMteresting ports, which are 1-1024 and the built-in list of popular ports.

Discussion:
There is a lot of documentation about the nmap portscanner, even a book by the author of nmap. Make sure to visit http://www.nmap.org

TCP and UDP is very different when scanning. TCP is connection/flow oriented and requires a handshake which is very easy to identify. UDP does not have a handshake and most applications will not respond to probes from nmap. If there is no firewall the operating system will respond to UDP probes on closed ports - and the ones that do not respond must be open.

When doing UDP scan on the internet you will almost never get a response, so you cannot tell open (not responding services) from blocked ports (firewall drop packets). Instead try using specific service programs for the services, sample program could be `nsping` which sends DNS packets, and will often get a response from a DNS server running on UDP port 53.

Exercise 24

Bonus: Perform nmap OS detection 10 min

Objective:
Use nmap OS detection and see if you can guess the brand of devices on the network

Purpose:
Getting the operating system of a system will allow you to focus your next attacks.

Suggested method:
Look at the list of active systems, or do a ping sweep.

Then add the OS detection using the option `-O`

Better to use -A all the time, includes even more scripts and advanced stuff See the next exercise.

Hints:
The nmap can send a lot of packets that will get different responses, depending on the operating system. TCP/IP is implemented using various constants chosen by the implementors, they have chosen different standard packet TTL etc.

Solution:
Use a command like `nmap -O -p1-100 10.0.45.45` or `nmap -A -p1-100 10.0.45.45`

Discussion:
nmap OS detection is not a full proof way of knowing the actual operating system, but in most cases in can detect the family and in some cases it can identify the exact patch level of the system.

Exercise 25

Bonus: Logning med syslogd og syslog.conf 10min

Objective:
See how server syslog is configured on regular Unix/Linux

Purpose:
The main idea of this exercise is to understand how easy network connected systems can send log data.

Suggested method:
Log into your local Linux systems or network devices, see how syslog is configured.

Hints:
Look in the config file, may be in /etc/syslog or /etc/syslog-ng/syslog-ng.conf

Sample output from old-skool syslogd

```
*.err;kern.debug;auth.notice;authpriv.none;mail.crit    /dev/console
*.notice;auth,authpriv,cron,ftp,kern,lpr,mail,user.none /var/log/messages
kern.debug;user.info;syslog.info                        /var/log/messages
auth.info                                               /var/log/authlog
authpriv.debug                                          /var/log/secure
...
# Uncomment to log to a central host named "loghost".
#*.notice;auth,authpriv,cron,ftp,kern,lpr,mail,user.none        @loghost
#kern.debug,user.info,syslog.info                               @loghost
#auth.info,authpriv.debug,daemon.info                           @loghost
```

Solution:
When you understand how to configure syslog from a couple of devices and has looked up which protocol and port it uses. (default is 514/udp)

Discussion:
There are syslog senders for Windows too.

