COM402 Tools

Operating Systems

While some exercises, setups and assignments may work on Windows, some others are likely to cause problems. Thus, we strongly recommend you use a Linux distribution (e.g., Ubuntu) or OS X as your host operating system. If your main OS is Windows, then you could install Linux in a dual-boot fashion.

If, however, you decide to use Windows, we advise you to set up a virtual machine (using VirtualBox with Ubuntu 18.04). If you choose setting up VirtualBox, please visit https://www.virtualbox.org/

Note that we will not assist you with Windows setup. Moreover, we know from past years that some assignments involving network setup may not work well with VirtualBox.

Docker

Throughout the semester, we will be using Docker for the exercises, which we introduce here. If you do not know yet what Docker is, please have a quick look at https://www.docker.com/what-docker. Wednesday's session is the right moment to ask for help if you have trouble when installing / using Docker on your machine.

Installation

First, you need to install Docker. Please follow the instructions at the website. Once you have finished the tutorial successfully, go to the next section below. We also advice you to run the post-installation steps. This will allow you to run docker without `sudo`.

First run

For this exercise, we set up a minimalistic docker image which runs an nginx web server. To pull and run the image, execute in a terminal: `docker run -d -p 8090:80 --name tools dedis/com402-tools`

This pulls the image from Dockerhub, an online repository of Docker images, and then runs the image on a container. The options are:

- `-d` is for detached mode. The docker container runs in the background.
- `-p` is for publishing ports of the container. This means that the port 80 on the container will be binded to the port 8090 on the host (your machine).
- `--name` tools means that you can refer to the container as "tools" instead of using hexadecimal identifiers.

You should now be able to access the welcome page by browsing to `localhost:8090`

`docker --help` gives you more information about docker commands. Here are some useful ones:

- List containers: `docker ps -as`
- Stop the container: `docker stop tools`
- Start the container again: `docker start tools`

For more information about images and containers, please see: https://docs.docker.com/storage/storagedriver/

For more information about network rules, please see: https://www.ctl.io/developers/blog/post/docker-networking-rules/

Shared Folder

This part is about setting up a shared folder between your machine and the container. This will be extremely useful for you when you will have to write some scripts that you want to execute inside the container. You can still write in your favorite editor and have the script being executed inside the container.

First, let's delete the previous container, as it is not allowed for security reasons to mount a volume inside the container once created:

```
docker rm -f tools
```

The -f option tells Docker to force the deletion of the container even if it is currently running.

Secondly, create a new directory containing a very simple 'index.html' file such as:

```
<h1> This is my custom index </h1>
```

Finally, run the container again with the shared volume (note the change of port and the new -v flag):

```
docker run -d -p 80:80 --name tools -v <path>:/html dedis/com402-tools
```

`<path>` must point to the local directory you created before, containing the `<index.html>` file **WITHOUT** the ending slash. For example, the path can be

`/User/john/com402/tools` . You need to use the absolute path. To find it, you can use the `pwd` command.

The flag `-v/--volume` has the following syntax: `<local>:<remote>`. * `<local>` refers to the local directory you are mounting * `<remote>` refers to the place where you are mounting your directory, *within the container*

Using the above values, you are using the content of `/User/john/com402/tools`, that will appear in the container at `/html`.

You can now try again to visit the page `localhost/shared/` in your browser and see your own page!

NOTE: For Windows users, you must enable first the directory inside the Docker app located in the system tray. For more information, look at https://rominirani.com/docker-on-windows-mounting-host-directories-d96f3f056a2c

Network Traffic

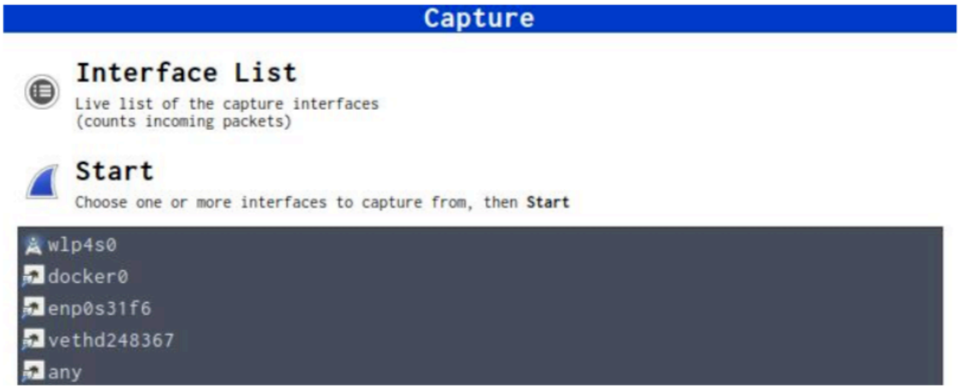
During the semester, we may ask you to monitor the traffic to collect some information about your targets. Or simply, for some of the assignments you may find it useful to take a look at the traffic to better understand what's going on. The most famous tool for traffic monitoring is called Wireshark. We will be using Wireshark to monitor the traffic from the container that we have just set up.

First, you need to install Wireshark on your platform. If you are on Windows or Mac OS, refer to https://www.wireshark.org/download.html. If you are running a Linux distribution, there is a high chance that wireshark is available through your regular package manager (e.g., apt-get). Once installed, launch Wireshark. You should see the list of your computer's network interfaces (the view depends on your host OS):

or

Select the interface named `docker0`, or `any` if you don't see the docker interface. You can also use the loopback `lo0` interface. Then launch the monitoring by clicking the shark icon:

The new window will show you **every** network packet going through that interface. For this tool demo, we only care about HTTP requests, since we are dealing with a web server. In the filter text box, type http and press `<Enter>`. Now, wireshark only shows the HTTP traffic going through the interface.



Capture

Interface List

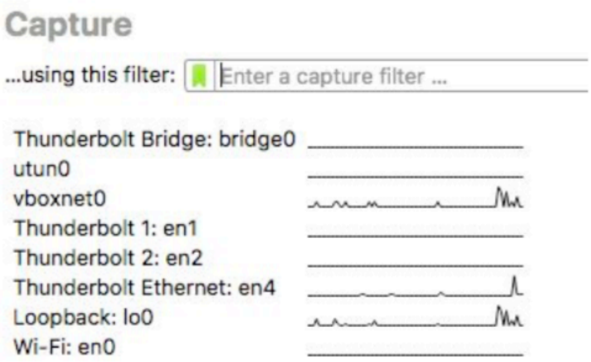
Live list of the capture interfaces
(counts incoming packets)

Start

Choose one or more interfaces to capture from, then **Start**

wlp4s0
docker0
enp0s31f6
vethd248367
any

Figure 1: Wireshark interface 1



Capture

...using this filter: Enter a capture filter ...

Thunderbolt Bridge: bridge0
utun0
vboxnet0
Thunderbolt 1: en1
Thunderbolt 2: en2
Thunderbolt Ethernet: en4
Loopback: lo0
Wi-Fi: en0

Figure 2: Wireshark interface 2

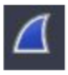
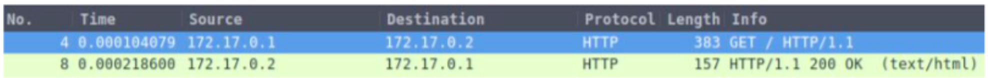


Figure 3: shark icon

Your wireshark is now set up correctly, let's sniff our http requests ! Go to your browser and request again the page localhost. Go back to wireshark, and you should now see a new HTTP request and response looking like this:



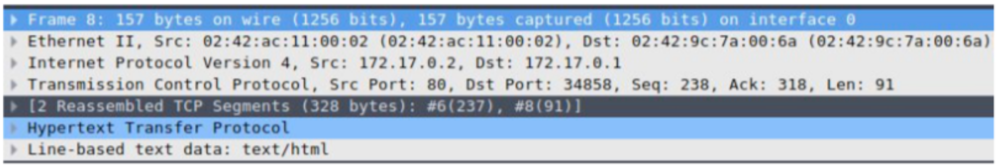
No.	Time	Source	Destination	Protocol	Length	Info
4	0.000104079	172.17.0.1	172.17.0.2	HTTP	383	GET / HTTP/1.1
8	0.000218600	172.17.0.2	172.17.0.1	HTTP	157	HTTP/1.1 200 OK (text/html)

Figure 4: http requests

The first line is the request from your browser to the web server in the container that wants to access the root of the web server, i.e., `index.html`

The second line is the response from the web server to your browser, it contains the content of the file index.html with some metadata, such as the response code 200 meaning the request is correct.

If you click on the response, you should now see in the lower window something that looks like:



```
▸ Frame 8: 157 bytes on wire (1256 bits), 157 bytes captured (1256 bits) on interface 0
▸ Ethernet II, Src: 02:42:ac:11:00:02 (02:42:ac:11:00:02), Dst: 02:42:9c:7a:00:6a (02:42:9c:7a:00:6a)
▸ Internet Protocol Version 4, Src: 172.17.0.2, Dst: 172.17.0.1
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 34858, Seq: 238, Ack: 318, Len: 91
▸ [2 Reassembled TCP Segments (328 bytes): #6(237), #8(91)]
▸ Hypertext Transfer Protocol
▸ Line-based text data: text/html
```

Figure 5: http request details

This window shows the packet contents including headers, which means you can look in detail into the structure and contents of a packet. For instance, this is useful if you programatically generate packets and want to see whether their structure is what you intended. In this example, you can see there is a "Hypertext Transfer Protocol" line which abbreviates to HTTP. You can get more information about the HTTP response by clicking on the arrows on the left.

Now go back to the localhost page. Can you see the hidden message?

For more information about HTTP, we refer you to the wikipedia article. There is also a tremendous amount of information about HTTP on the web, but don't hesitate to ask us questions if you have any!

Congrats, you finished the first setup!

Python and MySQL

Python is an easy-to-learn scripting language that can be used to assess security of a system. Python has a huge community and a tremendous amount of libraries in all kind of domains. Throughout the semester, we will be using the Python programming language to attack or defend some systems. MySQL is the most used database system in the world and has a long running experience. It is therefore also the most attacked by hackers.

As a warm-up task, we will code a small python script to connect to a running MySQL instance, retrieve the grades of some students and add new students!

Setup

Python is already installed inside the Docker container so you will be able to code with your own editor using a shared volume. First, let's delete our previous container with:

```
docker rm -f tools
```

Then let's mount it again by creating another shared volume. You might wonder why not just reusing the volume set up with nginx from the previous exercises. The answer is that it is always best to keep things separated. For example, if this was a real life situation, you would likely expose your python script to the whole Internet!

```
docker run -d --name tools -v <hostpath>:/scripts dedis/com402-tools
```

You can now create the script `scrapper.py` from the skeleton below and save it in your <hostpath> folder.

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import pymysql.cursors

# Connect to the database
connection = pymysql.connect(host='localhost',
                             user='monty',
                             password='python',
                             db='students',
                             cursorclass=pymysql.cursors.DictCursor)

try:
    # connection is not autocommit by default. So you must commit to save
    # your changes.
    connection.commit()

    with connection.cursor() as cursor:
        # Read a single record
        sql = "SELECT * FROM `grades`"
        cursor.execute(sql)
        result = cursor.fetchall()
        print(result)

        # Create a new record
        sql = "INSERT INTO `grades` (`name`, `grade`) VALUES (%s, %s)"
        cursor.execute(sql, ('bryan', '6'))

        # Read a single record
        sql = "SELECT * FROM `grades`"
        cursor.execute(sql)
        result = cursor.fetchall()

        found = False
        for row in result:
            if row["name"] == "bryan" and int(row["grade"]) == 6:
                print("Victory is sweetest when you’ve known defeat.")
                found = True
        if not found:
            print("I can only show you the door. You're the one that has to walk through it."

finally:
    connection.close()
```

Warm-up Scripting

If you look at the Python script, you will see it uses the library pymysql. It is a library used to communicate with a running MySql instance. You can look up the documentation

You can test the result by running the script in the container:

```
docker exec -it tools python3 /scripts/scrapper.py
```

The flag `-i` means *interactive*, (you will keep STDIN open even if not attached), and the flag `-t` means *TTY* (you will allocate a pseudo-TTY). While they are not strictly necessary for this example, they don't hurt and are often useful (try `docker exec -it tools python3`, and then try to remove one flag)

The database may take a few seconds to start; if you have an error `Can't connect to MySQL server on 'localhost'`, wait about 10 seconds and try again.

If all is well, you should see:

```
[{'name': 'Spock', 'grade': 3}, {'name': 'Kirk', 'grade': 5}]

Victory is sweetest when you’ve known defeat.
```

Congrats! :)

EPFL VPN

On rare occasions, the EPFL VPN is required to access some resources. It's best to make sure early on that you have a working connection. Please follow the procedure at https://www.epfl.ch/campus/services/ressources-informatiques/network-services-reseau/