

COM402 - Tools

Friendly warning: 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 (e.g., using Virtual Box) that runs a Linux distribution. Still, *we will not assist you with a Windows setup*. Moreover, we know from past years that some assignments involving network setup may not work well with Virtual Box.

1. 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>. Friday's session is the right moment to ask for help if you have trouble when installing / using Docker on your machine.

Note: You need a working Internet connection to follow this exercise.

a. Installation

First, you need to install Docker. Please follow the instructions at <https://docs.docker.com/install/>. Once you have finished the tutorial successfully, go to the next section below.

b. 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/engine/userguide/storagedriver/imagesandcontainers/>.

For more information about network rules, please see:

<https://www.ctl.io/developers/blog/post/docker-networking-rules/>.

c. 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, for example:

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

Finally, run the container again with the shared volume:

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

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

```
/User/mickey/com402/tools
```

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#.lw9j78q3d>.

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

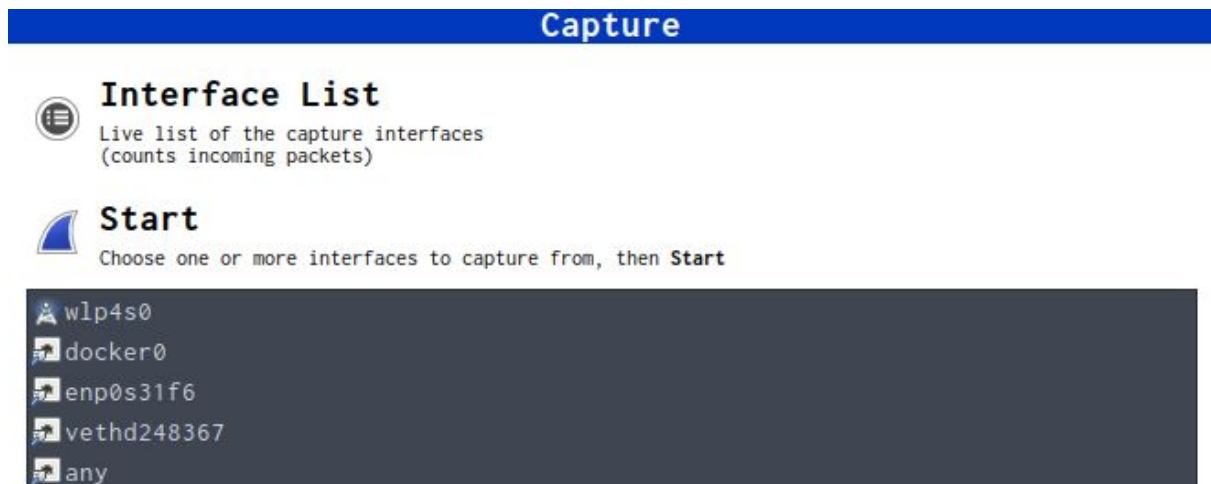
d. Network traffic

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

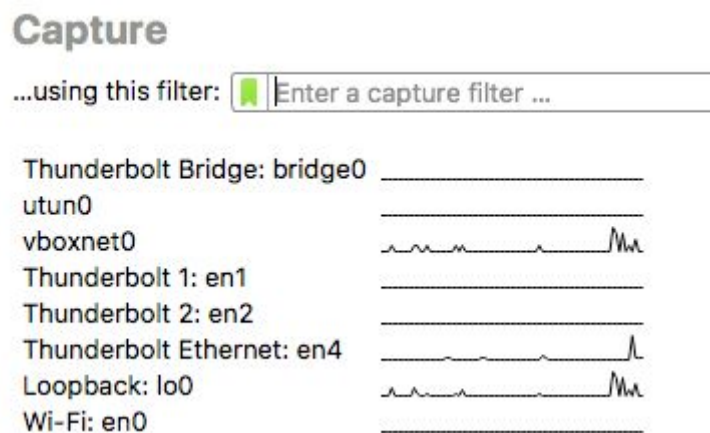
the traffic to better understand what's going on. The most famous tool for traffic monitoring is called Wireshark. We will be using Wireshark to monitor the traffic from the container that we have just set up.

First, you need to install Wireshark on your platform. If you are on Windows or Mac OS, refer to <https://www.wireshark.org/#download>. 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 then press <Enter>. Now, Wireshark only shows the HTTP traffic going through the interface.

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)

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` file 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

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 programmatically 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.

Challenge. Now, for the last part, you must discover a secret inside the HTTP response that you can't directly see on your browser!

For more information about HTTP, we refer you to the Wikipedia article https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. 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!

2. 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!

a. 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 re-using 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 download the skeleton script `scraper.py` from [Moodle](#) (under 01-Course Introduction) and save it in your `<hostpath>` folder. Note that you need to be logged in to Moodle.

b. Warm-up scripting

If you look at the Python script, you will see it uses the library `pymysql`. It's a library used to communicate with a running MySql instance. You can look up the documentation at <https://github.com/PyMySQL/PyMySQL>.

Challenge. Fill in the blanks!

Once you think you have finished, you can test the result by running the script in the container:

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

If all is well, you should see:

```
[{'name': 'Spock', 'grade': 3}, {'name': 'Kirk', 'grade': 5}]  
Victory is sweetest when you've known defeat.
```

Congrats! :)

3. EPFL VPN

Throughout the semester, for the assignments, you'll issue requests to our internal web server. This web server is only accessible from within the EPFL network and through VPN (that way you can also work from home). Because last year some students encountered issues when trying to work from home, it's best to make sure early on that you have a working setup.

First, you should make sure you have a working VPN connection. For installation, please look at: <http://epnet.epfl.ch/AnyConnect-VPN-Clients>

Now, try to reach `http://com402.epfl.ch` from the internal EPFL network **and** from the VPN (e.g., test it from home).