ECS655U: Security Engineering (2019) Preparation for the Lab of Week 1

How to install and use `Jupyter Notebook` – 30 minutes

Dr. Arman Khouzani

Jan 11, 2019

We will be using Python 3 scripts in labs, mostly in the first half of the module, i.e., crypto-related exercises. This document is to prepare the ground for that.
If you don't know any python, it is fine: no python knowledge is assumed. However, learning basics of python scripting is part of the soft skills of this module as well. It is expected that a CS-degree holder knows a bit of a scripting language besides JavaScript (Python, Ruby, Perl, Tcl, VBScript, PHP, ASP) – and the most popular of the bunch is Python! **For the labs of this module, you need to be logged in to a Linux machine. If your machine is in any other OS, you need to restart and choose Linux from the boot-up menu.**

Jupyter Notebook

`jupyter notebook` is an open-source web-application that provides an interactive environment (in a web browser on your machine) for writing and executing codes for many (interpreted) languages, most notably, `python`. We will be using this as it allows seamless integration of both code and formatted texts.

Installing and configuring `jupyter notebook`

Jupyter Notebook (a descendant of IPython Notebook) comes with many python distributions like `anaconda`, but can also be installed as a "module" (as we will describe later). Python itself comes native with a lot of Linux distributions. The following steps describe how to create a python 3 "virtual environment" and install jupyter in it.

1. Open a `terminal`.

2. In Linux, the command `pwd` (print working directory), shows where your are. If you want this to become part of your "prompt", issue the following:

   ```
   export PS1='\u@\h:\w\$ '
   ```

   which means set the prompt to show:
   `username@host-machine:working-directory $`
   Make sure the names of your directories do not contain any spaces (or special characters, like parentheses or brackets, etc.): virtual environment doesn't like it! (a known bug, unfortunately). If your pathname does contain any of these characters, rename them: e.g. either remove the spaces, or replace them with underlines.

3. Go to your home directory (`cd ~`) or wherever you prefer (e.g. `cd ~/Documents`), and create a folder for our module, and change into that directory:

   ```
   mkdir ECS655U && cd ECS655U
   ```

4. A "virtual environment" provides us a clean copy of python cores and allows us to install any extra packages (or "modules" as they are called in Python) we wish for a specific project. So next, we create a python 3 `virtual environment` using `venv` and label it with whatever name you would like as its argument (e.g. I have chosen `ECS655U_VENV_P3`).

   ```
   python3.4 -m venv ECS655U_VENV_P3
   ```

   The instructions uses `python 3.4` because it is tested and works on ITL machines, but if you are setting things up on your own machine, any `python 3` version would do.

5. Activate your virtual environment:

   ```
   source ECS655U_VENV_P3/bin/activate
   ```

6. Notice the change in the prompt sign, indicating the activation of the environment. You can check the active python version by invoking `which python`, and checking its version by invoking `python -V` (or `python --version`). You should see the version of the python is `3.4.8`.

7. As we said, we can now install any packages we wish in our virtual environment, using a python package manager called `pip` (and `setuptools`). So first, we update our package manager of the python 3 virtual environment we just activated, and then installing `jupyter`:

   ```
   python -m pip install --upgrade pip
   python -m pip install --upgrade setuptools
   python -m pip install jupyter
   ```

Wait for the downloads and installation to finish and the return of the prompt (it should take less than a minute).

8. You are now ready to launch `jupyter notebook`:

   ```
   python -m jupyter notebook
   ```

   The notebook environment should open in your default web browser (e.g. `Firefox`).[1]

   NOTE: Jupyter's root directory becomes wherever you launch it from. That means the folders above it cannot be accessed from jupyter. But any subfolders will be accessible. So if you have been following the instructions so far, the root directory of Jupyter is `ECS655U`.

9. Importantly, in your jupyter notebook, click on the the drop-down `New` menu on the top-right corner: you should see a Python 3 kernel (that of our virtual environment) listed here. If not (e.g., if you only see `python 2`), proceed with the "Troubleshooting" subsection next. Otherwise, you are done with setting up jupyter notebook successfully!

Troubleshooting

If you don't see a python 3 kernel in your jupyter, then we need to manually add it to its available kernels. For that, we first need to install another module called ipykernel. Close your jupyter notebook and return to the terminal that you activated our virtual environment from. Press `Ctrl+C` to kill the jupyter program. Then, enter:

```
python -m pip install ipykernel
```

Next, issue the following command, noting that what you put after `--name` is the name of your virtual environment (the one we activated using `source activate`) and the name you put after `--display-name` is the name that appears inside the jupyter notebook for that kernel (also note the final quotation marks), and all should be in a single line:

```
python -m ipykernel install --user --name ECS655U_VENV_P3
 --display-name "P3"
```

All should be now fine: we can check the list of available kernels of your jupyter by issuing:

```
python -m jupyter kernelspec list
```

You should see the kernel you just added. Once done, launch the jupyter notebook:

```
python -m jupyter notebook
```

[1] If it does not, click on the link that appear in the prompt after running the command (or copy/paste it in your web browser manually).

Using `jupyter notebook`

- Make sure the kernel selected is the python 3 kernel.
- First things first, take for an overview of the jupyter notebook's user interface: click on `Help -> User Interface Tour` from the menu on the top.
- The most important concepts to keep in mind is, things are arranged in cells, and there are two (main) cell types: **Code** and **Markdown**. As you can guess: the "code" cells (by default indicated by In [ ]:) are where you can enter code and run them by clicking the *run cell* button on the top or simply using one of the following two shortkeys:

  Alt-Enter runs the current cell and proceeds to the next cell.

  Ctrl-Enter run the current cell and stay there.

- You can "create", "edit", "save", and "open" other notebook files (which are designated with the extension of `.ipynb`). Again, note that whenever you run a jupyter notebook instance, the root directory becomes the directory that it is launched from (specifically, `~\ECS655U\` in our example). Hence, to access `.ipynb` files that you download from QM+ or elsewhere, you need to save them somewhere inside that directory.
- The variables (and functions) are preserved from a cell to the next. If you want to clear them out and start from a blank state, click on *Kernel* from the above tabs, and click on *Restart & Clear Output*.
- A note and a warning, quoting from here:

  Save notebooks: modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu `File -> Make a copy ...`) and save the modifications on the copy.

  Warning: Pay attention to not open the same notebook document on many tabs: edits on different tabs can overwrite each other! To be safe, make sure you open each notebook document in only one tab. If you accidentally open a notebook twice in two different tabs, just close one of the tabs.

If you need further introduction on the jupyter notebook:

- Download this notebook file (click) and open in jupyter notebook as described above (or just read it online).
- Watch this youtube video (at home!)

Wait for further instructions!