Secure Programming

Lab 1 – Installation

Introduction

Today's lab will walk you through the installation of the tools you will need to complete the labs throughout the semester. We will be using a Web application built on Python using the Django framework. Don't worry too much if you are not too familiar with Python (or Django!) as you will be walked through each step. As I stated in the lecture, the emphasis for the module is on Security and not Software Development, though we do need to delve into some code in order to fix the application! However, generally the code fixes are small, and you will not be required to write complete code classes or a full Web application. Before we install the tools, let's look at the application and the Django framework.

Vulnerable Web Application

The Web application we will be using is a purposely built vulnerable application called "Coffeeshop". During the labs, we will look at various exploits against this application and then we will secure the code against those exploits. We will be using the same application each week, so by the end of the semester, your application will be secure against the most common attacks.

Django

Django is a high-level, open-source Python web framework that encourages rapid development and clean, pragmatic design. It was designed to help developers take applications from concept to completion as quickly as possible. Django emphasizes **reusability**, **pluggability of components**, and **don't repeat yourself (DRY)** principles. Again, I emphasise not to be concerned about understanding every aspect of Django and Python. We will be using a pre-configured Docker container, with everything already installed and will be administered through Vagrant. If you would like to learn more about Django, there is a great set of tutorials here.

Universal Installation

We are using Docker containers as these provide a way to install on any platform. Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.

Infrastructure Installation (all platforms)

1. Docker Desktop
2. Vagrant
3. Git
4. HTTP Toolkit
5. Windows Terminal (Windows users only) – this is not strictly needed but will allow for enhanced commands that the regular command line is not capable of.
6. Chrome and Firefox Web Browsers (if you don't already have them installed). We will use these for labs.

Download and install all the above tools for **your operating System**.

Install the coffeeshop app

When you have Git installed, start a command prompt (Windows Terminal, Mac Terminal, etc.), change to the directory where you would like to install the code, and type:

git clone https://github.com/stephen-oshaughnessy/django-coffeeshop.git

This will download the coffeeshop application. Later in the semester, we will also use another application called *csthirdparty*, which is also located inside the **Django-coffeeshop** root folder. The *csthirdparty* application is used for labs that require cross-site testing.

The installed directory tree is shown in Figure 1 below.

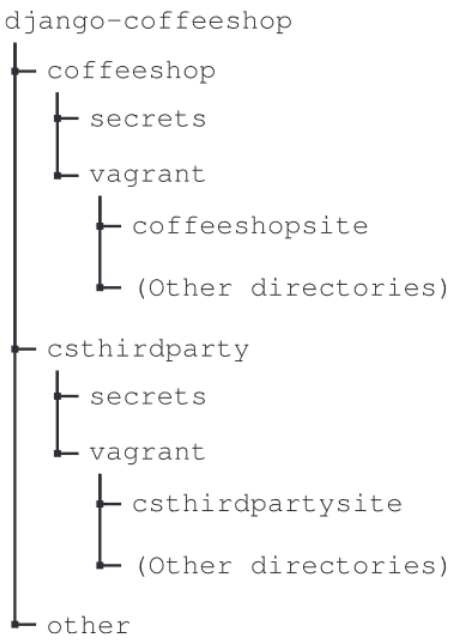


Figure 1: coffeeshop directory tree

Installation via Vagrant

Vagrant is an open-source tool used to build, configure, and manage reproducible and portable development environments, specifically virtual machines (VMs). It is primarily designed to work with virtualization tools like VirtualBox, VMware, and cloud platforms like AWS and containers like Docker.

Vagrant allows developers to manage these environments in a consistent and easily reproducible way by using simple configuration files, called Vagrantfiles. This ensures that the development environment is the same across all developers' machines, preventing the "*it works on my machine*" problem.

In order to start the installation via Vagrant, you must first change directory (cd) to the vagrant directory located in the repo you have just cloned, e.g.,

```
cd .\django-coffeeshop\coffeeshop\vagrant\
```

- In the vagrant directory, type:

```
vagrant up --provider=docker
```

This should start the installation process. This will take some time to install the container and all dependencies. The “vagrant up” command boots a Vagrant VM. If it has not already been built, it builds it first. The first build can take some time as it must download the Ubuntu Linux OS,install the various packages, configure the web server and database. You should see an output similar to the one below:

```
PS C:\Users\Stephen OShaughnessy\django-coffeeshop\coffeeshop\vagrant> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/focal64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/focal64' version '20240821.0.0' is up to date...
==> default: Setting the name of the VM: coffeeshop
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> default: Forwarding ports...
    default: 8100 (guest) => 8100 (host) (adapter 1)
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
```

:

:

```
    default: After this operation, 90.1 kB of additional disk space will be used.
    default: Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 swapspace amd64 1.14-1 [23.7 kB]
    default: dpkg-preconfigure: unable to re-open stdin: No such file or directory
    default: Fetched 23.7 kB in 0s (191 kB/s)
    default: Selecting previously unselected package swapspace.
(Reading database ... 93957 files and directories currently installed.)
    default: Preparing to unpack .../swapspace_1.14-1_amd64.deb ...
    default: Unpacking swapspace (1.14-1) ...
    default: Setting up swapspace (1.14-1) ...
    default: Created symlink /etc/systemd/system/multi-user.target.wants/swapspace.service → /lib/systemd/system/swapspa
ce.service.
    default: Processing triggers for man-db (2.9.1-1) ...
    default: Processing triggers for systemd (245.4-4ubuntu3.23) ...
==> default: Running provisioner: shell...
    default: Running: C:/Users/STEPHE~1/AppData/Local/Temp/vagrant-shell20240910-5612-1y5ouz.sh
    default: Invoking 'systemctl start apache2'.
    default: Use 'systemctl status apache2' for more info.
PS C:\Users\Stephen OShaughnessy\django-coffeeshop\coffeeshop\vagrant>
```

The CoffeeShop runs on **http://localhost:8080**. If the build has gone successfully, open a Web browser on your host machine and type http://localhost:8080 and you should see a Web page similar to Figure 2 below.

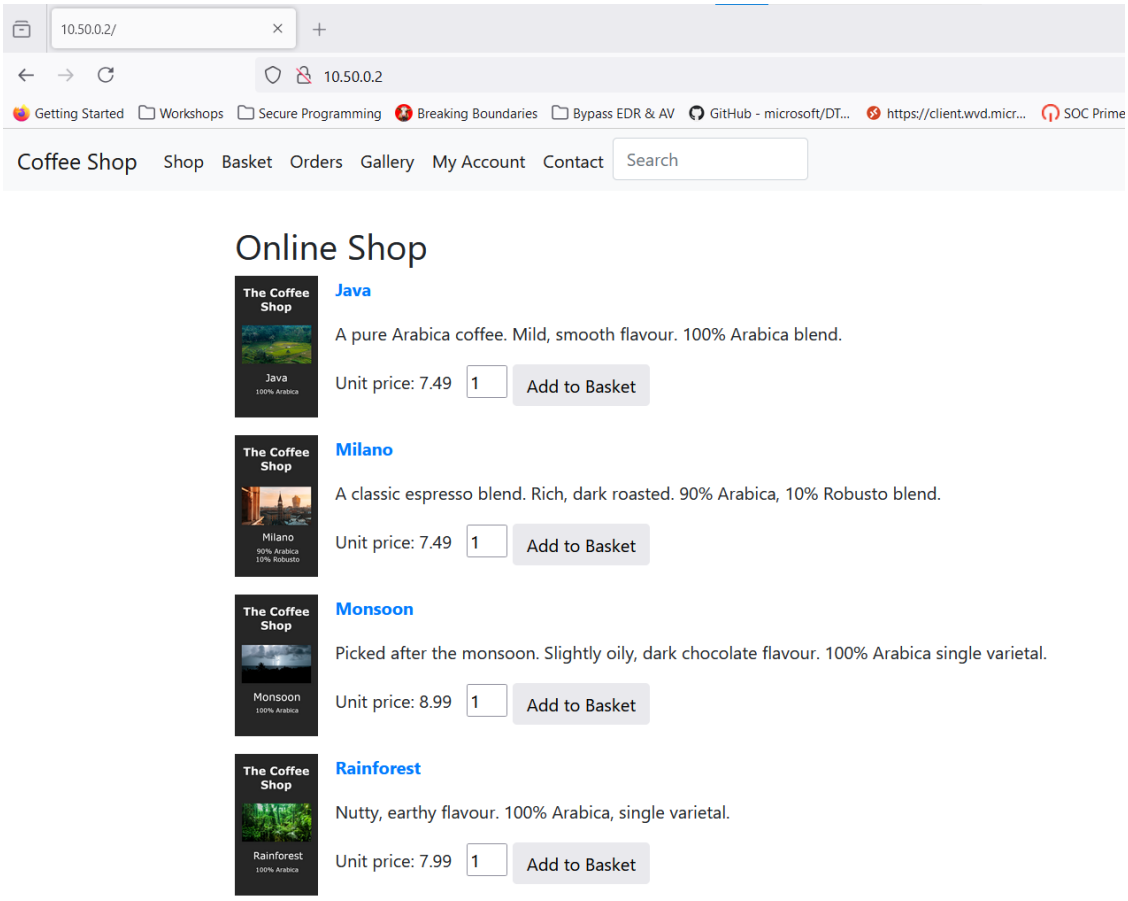
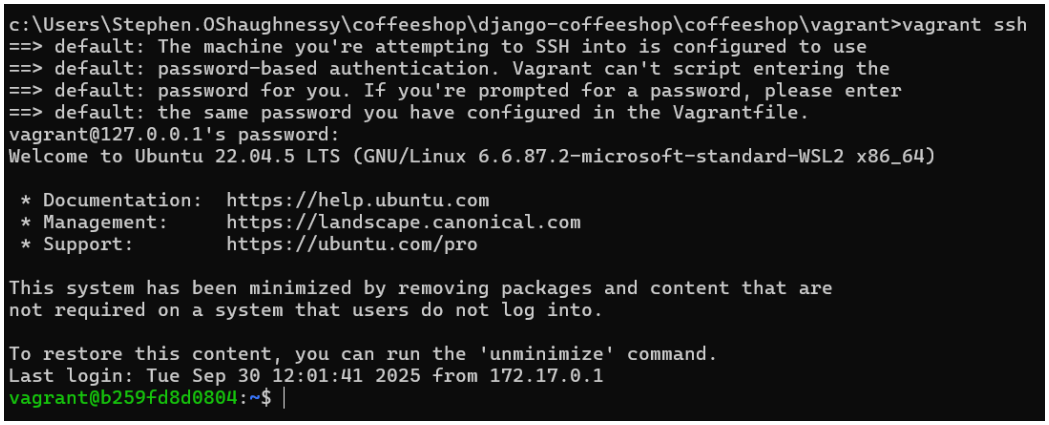


Figure 2: coffeeshop home page

You can run the following check on the container to make sure the database is running and reachable:

- From the vagrant folder on the host, type **vagrant ssh** - this will connect you to the backend container via ssh:



Note: the ssh password is: **vagrant**

Once connected, run the following:

```
sudo -u www-data python3 /vagrant/coffeeshopsite/manage.py shell -c “from coffeeshop.models import Product; print(‘Products:’, Product.objects.count())”
```

Expected output:

```
vagrant@b259fd8d0804:/$ sudo -u www-data python3 /vagrant/coffeeshopsite/manage.py shell -c "
from coffeeshop.models import Product; print('Products:', Product.objects.count())"
Products: 4
```