ITRI615 - Computer Security
Project Documentation

Affaan Muhammad - 33016763

Joshua Esterhuizen - 30285976

Contents

Section 1

Installation and setup

1.1 Project files

The project files can be found on the following GitHub link:
https://github.com/AM-ops/SecurityProject

This was our main code repository. We both have been updating the code as we went along and added details and bug fixes to the project.

To copy the code to your own machine, follow the following steps:

1. Make sure Git is installed. If not it can be downloaded from here:
   https://git-scm.com/
2. Create an empty directory where the code can be copied to
3. Run the following command:

   ```
   git clone https://github.com/AM-ops/SecurityProject.git
   ```

1.2 Virtual Environment

There are multiple advantages of using virtual environments when creating software. The primary reason being we create a layer of separation and abstraction between our host machine's files and our software project.

We made use of a Python virtual environment which was handled by Anaconda. This can be downloaded from the following link:
https://www.anaconda.com/products/individual

1.2.1 Creating a virtual environment

Once Anaconda was installed the following commands were run in the terminal to create a virtual environment called `myDjangoEnv`.

```
conda create --name myDjangoEnv
```

Depending on the version of Anaconda installed you might have to use a leading underscore on Windows machines. The same will apply for commands further down. Below is a demonstration.

```
_conda create --name myDjangoEnv
```

1.2.2 Listing virtual environments

To list all virtual environments on your host machine run the following command.

```
conda info --envs
```

or

```
conda env list
```

1.2.3 Deleting a virtual environment

To delete a virtual environment run the following commands.

```
conda remove --name <name_of_virtual_environment> --all
```

or

```
conda env remove --name <name_of_virtual_environment>
```

1.2.4 Activating and deactivating virtual environments

To activate an environment run the following commands for Windows.

```
conda activate <name_of_virtual_environment>
```

For Linux and MacOS the command is as follows.

```
source activate <name_of_virtual_environment>
```

Once the environment is activated your terminal should change. By default, the active environment, is shown in parentheses () or brackets [] at the beginning of your command prompt as shown below.

```
(<name_of_virtual_environment>) >_
```

Depending on your version of Anaconda to deactivate your environment the commands for Windows is.

```
deactivate
```

or

```
conda deactivate
```

For Linux and MacOS the command will be

```
source deactivate
```

1.2.5 Listing Packages installed

To list all the packages you have installed in an environment there are two methods of listing them. First, if the environment is not activated run the following.

```
conda list -n <name_of_virtual_environment>
```

Secondly, if the environment is activated, then simply run the following.

```
conda list
```

1.2.6 Using pip

Due to the fact that Python is being used for the project it is always necessary to make sure `pip` is installed and functioning. If it is not then run the following commands.

```
conda install -n <name_of_virtual_environment> pip
```

1.3 Frameworks and other packages

1.3.1 Django

The primary framework used for development in this project was `Django`. This is a python based Web framework. The documentation for it can be found here:
https://docs.djangoproject.com/en/3.2/

1.3.2 Bootstrap

Bootstrap is Cascading Style Sheets (CSS) Framework which allows for simple, elegant, and responsive Graphical User Interfaces to be developed for the Web. The documentation for it can be found here:
https://getbootstrap.com/docs/5.0/getting-started/introduction/

For a more seamless integration of Bootstrap with the `Django` Framework an additional package called `django-crispy-forms` was also installed. Its documentation ca be found here:
https://django-crispy-forms.readthedocs.io/en/latest/

1.3.3 Miscellaneous

For typesetting of this documentation, LaTeX was utilised. Additionally, a LaTeX package called `minted` was used to typeset code in this documentation. Its homepage is located at: https://www.ctan.org/pkg/minted
To typeset directory structures in a tree-like manner the LaTeX package `dirtree` was used. Its homepage can be found at: https://ctan.org/pkg/dirtree

Lastly, to typeset code within the HTML pages of our project the JavaScript library called `Rainbow` was implemented. The GitHub link for that is located at:
https://github.com/ccampbell/rainbow

Section 2

Programming of artefact

2.1 Development Tools

2.1.1 Operating Systems

The primary systems on which development was done was Linux and Windows 10. The same systems where utilised for testing and bug fixing purposes.

2.1.2 IDEs

For the purposing of coding the following two Integrated Development Environments were used:

1. Atom. It can be downloaded from: `https://atom.io/`
2. Visual Studio Code, also known as VSCode. It can be downloaded from here: `https://code.visualstudio.com/`

2.1.3 Database Management Tools

For the purposes of database management, TablePlus was the main software we utilised. It was used to see if our `Django` models and cryptographic schemes were correctly implemented. TablePlus can be downloaded from: `https://tableplus.com/`

2.1.4 Hosting

Due to a number of constraints we landed up running our project locally. The server was `localhost` and the port number was `8000`. Therefore the link where we ran our project was: `http://127.0.0.1:8000`

2.2 Prerequisites

2.2.1 Project and Package Initialisation

From here on we will refer to the working directory as the directory where the `manage.py` *file is located. This file is created when the project is setup*

Before our `Django` project can be created we have to install all the packages mentioned above in Section 1.3.1 and 1.3.2. A text file called `requirements.txt` was created which lists the 3 packages we need to install as shown below:

```
django
django-crispy-forms
bootstrap4
```

Thereafter the following command was run in the working directory.

```
pip install -r requirements.txt
```

To start a `Django` project called `SecProj` the following command was run:

```
django-admin startproject SecProj
```

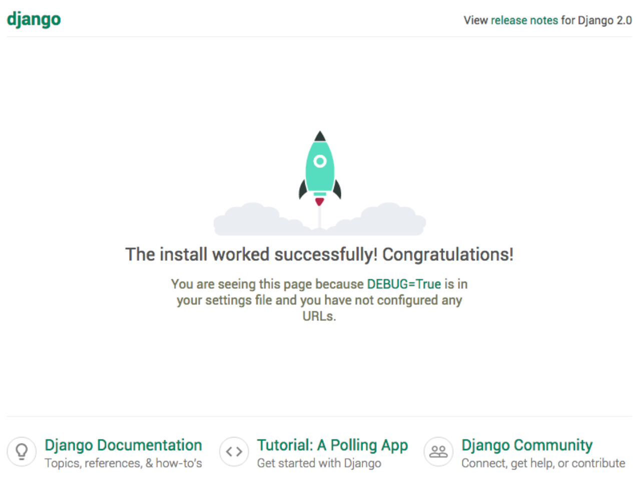
Your directory should look like the following:

```
/
├── manage.py
└── SecProj
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    ├── asgi.py
    └── wsgi.py
```

To verify that your `Django` project is working run the following command in your working directory:

```
python manage.py runserver
```

You should see the following if you open the link: `http://127.0.0.1:8000`


django
View release notes for Django 2.0
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Figure 2.1: Default Homepage of a new Django Project

Now that your `Django` project is up and running it is time to create a `Django` 'App' within this project. This App is where we implemented our cryptographic schemes and the bulk of our project. We called our App `SecApp` and the command to run in your working directory is as follows:

```
python manage.py startapp SecApp
```

Your directory should now look like the following:

```
/
├── manage.py
├── SecProj
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── asgi.py
│   └── wsgi.py
└── SecApp
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── migrations
    │   └── __init__.py
    ├── models.py
    ├── tests.py
    └── views.py
```

2.2.2 Settings and Admin

The following changes were added to the `settings.py` file. The whole file is not shown below.

```python
from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR = os.path.join(BASE_DIR, 'templates')

# Application definition

INSTALLED_APPS = [
    'django.contrib.staticfiles',
    'bootstrap4',
    'SecProj',
    'SecApp',
    'accounts',
    'crispy_forms',
]
CRISPY_TEMPLATE_PACK = 'bootstrap4'

ROOT_URLCONF = 'SecProj.urls'
TEMPLATES = [{'DIRS': [TEMPLATE_DIR]}]

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIR = [os.path.join(BASE_DIR,'static')]
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]
LOGIN_REDIRECT_URL = 'success'
LOGOUT_REDIRECT_URL = 'thanks'
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

To create a superuser to handle `Django` administration run the following commands and follow the prompts:

```
python manage.py createsuperuser
```

The link for `Django` admininstration for the project is: http://127.0.0.1:8000/admin

2.3 Models

2.4 Views

2.5 Templates

Section 3

User manual

Section 4

Reflection

Section 5

Sources

https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html
https://www.udemy.com/course/python-and-django-full-stack-web-developer-bootcamp/
https://docs.djangoproject.com/en/3.2/