**This document will show you how to use the module 2 of the bug injector**

**Setup joern**

First, add the image neepl/joern to your docker : https://hub.docker.com/r/neepl/joern/

Create a folder where you will put your code, here it will be joern9

Put the code that you want to analyze in it

Launch the following command (adapt the path to your folder)
*sudo docker run --name joern-construct -v /path/to/your/folder:/code -p 7474:7474 -p 7687:7687 --rm -w /code -it neepl/joern java -jar /joern/bin/joern.jar .*
You should have the following result :

```
julien@julien-Spin-SP513-51:~$ sudo docker run --name joern-construct -v /home/julien/PI/joern9:/code
 -p 7474:7474 -p 7687:7687 --rm -w /code -it neepl/joern java -jar /joern/bin/joern.jar .
[sudo] password for julien:
Warning: your JVM has a maximum heap size of less than 2 Gig. You may need to import large code bases
 in batches.
If you have additional memory, you may want to allow your JVM to access it by using the -Xmx flag.
./test_fopen.c
```

The database for joern is now created, we will create the container that we will work with.

Launch the following command to create the container :
*sudo docker run --name joern-run -v /path/to/your/folder:/code -p 7474:7474 -p 7687:7687 -it neepl/joern /var/lib/neo4j/bin/neo4j console*
You should see the launch of neo4j and when you have the following 2 lines, the container is successfully launched and fonctionnal :

```
2019-02-26 14:44:46.706+0000 INFO  [API] Server started on: http://0.0.0.0:7474/
2019-02-26 14:44:46.706+0000 INFO  [API] Remote interface ready and available at [http://0.0.0.0:7474
/]
```

You can now stop the container with CTRL+C

You have now a working container with joern and neo4j running on it, you can start and stop it as usual.

Some commands :
Launch the container :
*sudo docker start joern-run*
Enter in the container :
*sudo docker exec -it joern-run /bin/bash*
Stop the container :
*sudo docker stop joern-run*

**Use the joern UC variables detection script**

First you should put the python script in your “/code” folder.

You should also put the output.json from the clang UC variables detection script in your “/code” folder.
As a reminder, the “/code” folder is the folder where you did put the C code where you want to inject vulnerabilities.

You can now (in the joern container) launch the command “*python script.py*” in the “/code” folder.

It will generate all the outputs for the next step.