Configuration Manual

MSc Research Project
MSc. Cybersecurity

Daniel Ruane
Student ID: 22195980

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet
School of Computing

Student Name: Daniel Ruane
…. ………………………………………………………………………………………………………

Student ID: 22195980
………………………………………………………………………………………………………..……

Programme: ……MSc of Cybersecurity…………… **Year:** ………2024…………..

Module: ……Research Project…………………………………………………………………….………

Supervisor: ……Ross Spelman……………………………………………………………………….………

Submission Due Date: ……12th August 2024…………………………………………………….………

Project Title: … Internet Of Things Device Security……………………………….………

Word Count: ………………1189…………… **Page Count**……………8…………….……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……………Daniel Ruane………………………………………………………

Date: …………6th August 2024…………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	

Penalty Applied (if applicable):	

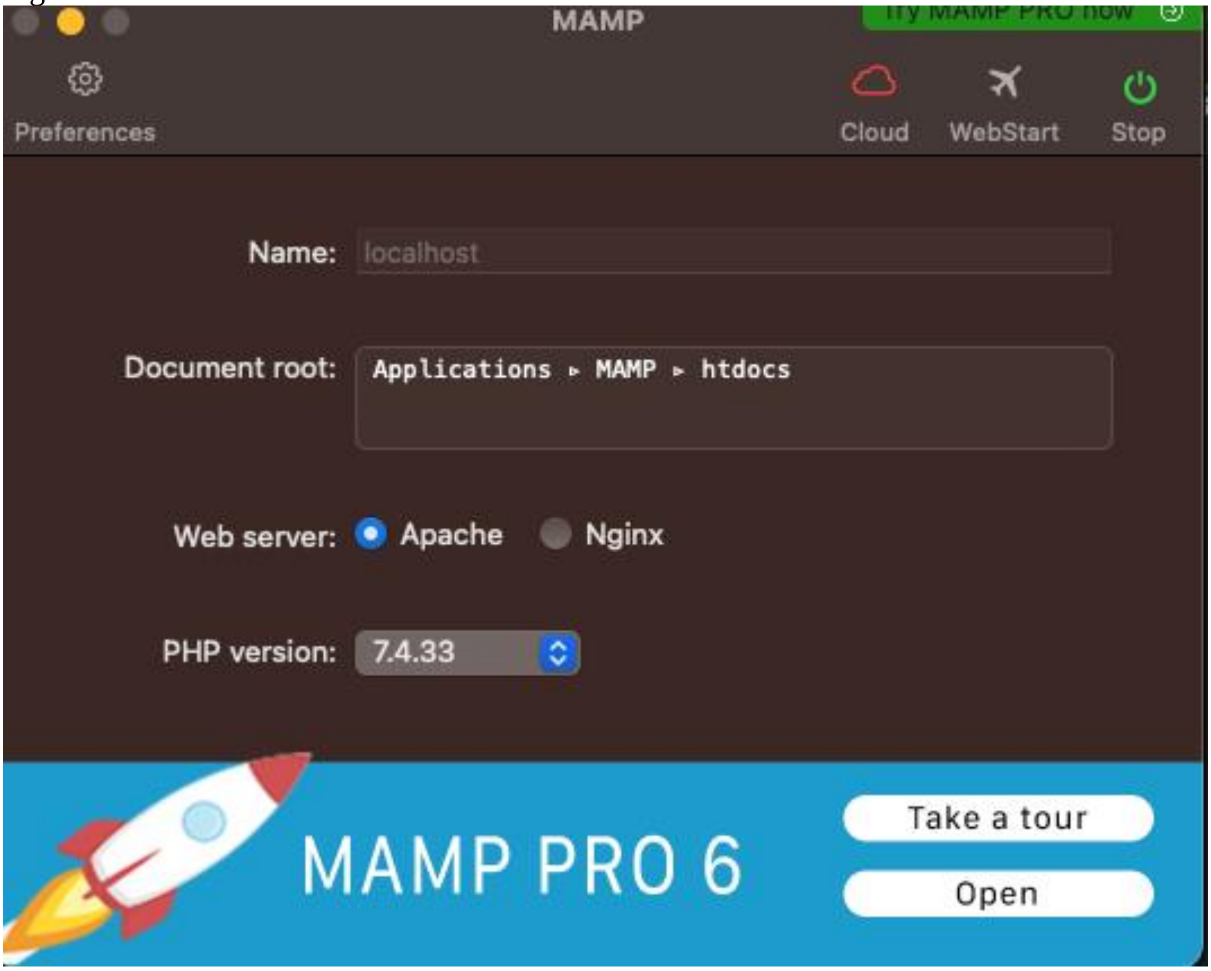
Configuration Manual

Daniel Ruane
Student ID: 22195980

1 Building The Staging Environment

Step 1. The first step to building the staging environment on a Mac is to download and install MAMP version 6 here https://downloads.mamp.info/MAMP-PRO/macOS/MAMP-PRO/MAMP-MAMP-PRO-6.9-Intel-x86.pkg MAMP is the MacOS tool that installs all of the tools that you will need to run Php, MySQL, HTML and CSS. (See fig 1.0)

Fig 1.0 MAMP – LAMP stack for MacOS



Name: localhost

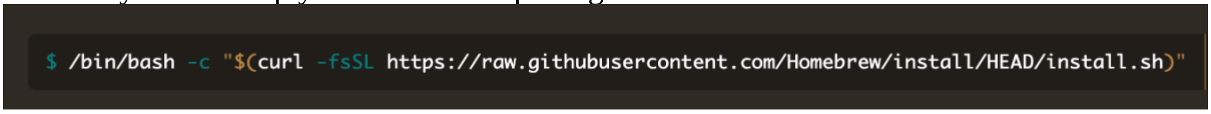
Document root: Applications ▸ MAMP ▸ htdocs

Web server: ● Apache ○ Nginx

PHP version: 7.4.33

Step 2. The next step to setup the staging environment is to install the tools required to run the Python simulation file. This requires Python3 so it’s best to start using a package manager from the get-go.

The best package manager for MacOS is Homebrew. To install Homebrew run from terminal the following command: /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)" Once homebrew is installed you are setup you install more packages.



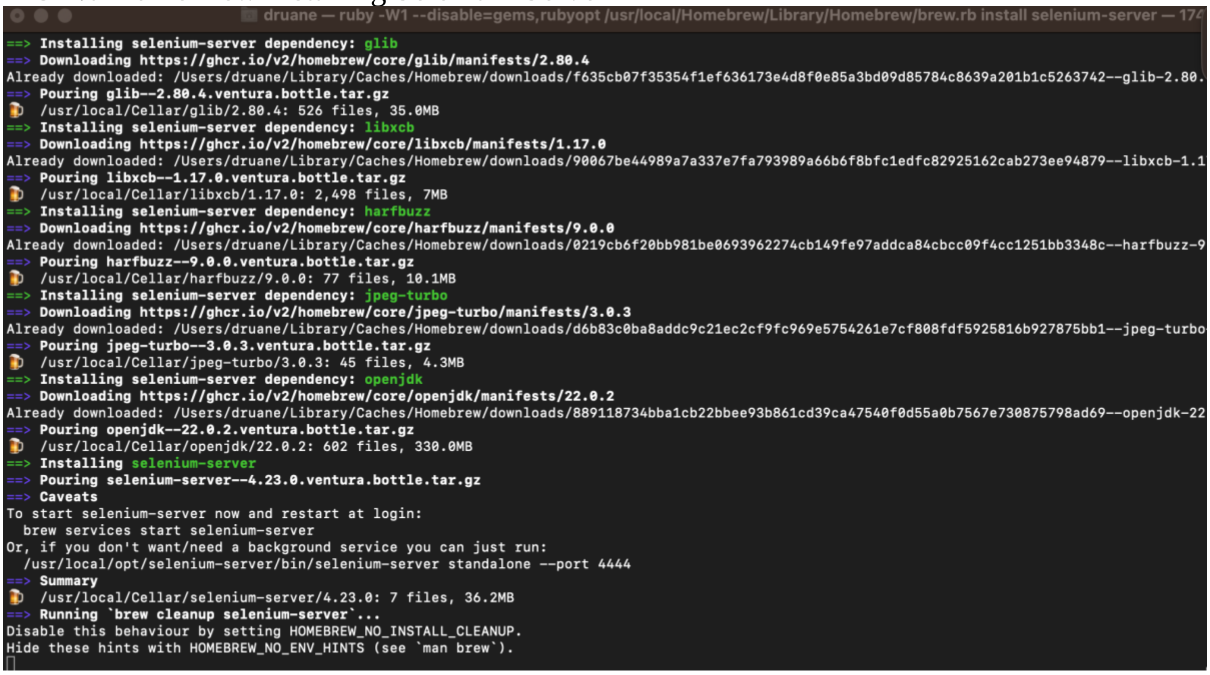
```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Step 3. You can now install Python3 if it’s not already installed by running brew install python

Step 4. After Pythion installs it’s a good idea to upgrade pip, this is a Python lib stroke package manager. You can do this by running pip install – upgrade pip

Step 5. After Python3 and Pip are up and running its time to instal the Selenium Server. Do this by running brew install selenium-server (See Fig 2)

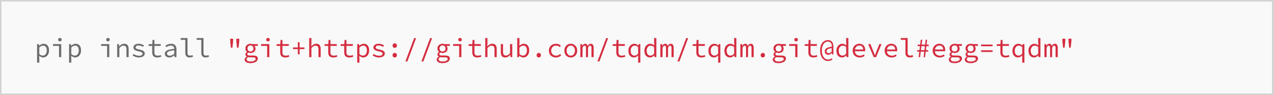
FIG 2.0 HomeBrew Installing Selenium Server



druane — ruby -W1 --disable=gems,rubyopt /usr/local/Homebrew/Library/Homebrew/brew.rb install selenium-server — 174

```
==> Installing selenium-server dependency: glib
==> Downloading https://ghcr.io/v2/homebrew/core/glib/manifests/2.80.4
Already downloaded: /Users/druane/Library/Caches/Homebrew/downloads/f635cb07f35354f1ef636173e4d8f0e85a3bd09d85784c8639a201b1c5263742--glib-2.80.
==> Pouring glib--2.80.4.ventura.bottle.tar.gz
🍺  /usr/local/Cellar/glib/2.80.4: 526 files, 35.0MB
==> Installing selenium-server dependency: libxcb
==> Downloading https://ghcr.io/v2/homebrew/core/libxcb/manifests/1.17.0
Already downloaded: /Users/druane/Library/Caches/Homebrew/downloads/90067be44989a7a337e7fa793989a66b6f8bfc1edfc82925162cab273ee94879--libxcb-1.1
==> Pouring libxcb--1.17.0.ventura.bottle.tar.gz
🍺  /usr/local/Cellar/libxcb/1.17.0: 2,498 files, 7MB
==> Installing selenium-server dependency: harfbuzz
==> Downloading https://ghcr.io/v2/homebrew/core/harfbuzz/manifests/9.0.0
Already downloaded: /Users/druane/Library/Caches/Homebrew/downloads/0219cb6f20bb981be0693962274cb149fe97addca84cbcc09f4cc1251bb3348c--harfbuzz-9
==> Pouring harfbuzz--9.0.0.ventura.bottle.tar.gz
🍺  /usr/local/Cellar/harfbuzz/9.0.0: 77 files, 10.1MB
==> Installing selenium-server dependency: jpeg-turbo
==> Downloading https://ghcr.io/v2/homebrew/core/jpeg-turbo/manifests/3.0.3
Already downloaded: /Users/druane/Library/Caches/Homebrew/downloads/d6b83c0ba8addc9c21ec2cf9fc969e5754261e7cf808fdf5925816b927875bb1--jpeg-turbo
==> Pouring jpeg-turbo--3.0.3.ventura.bottle.tar.gz
🍺  /usr/local/Cellar/jpeg-turbo/3.0.3: 45 files, 4.3MB
==> Installing selenium-server dependency: openjdk
==> Downloading https://ghcr.io/v2/homebrew/core/openjdk/manifests/22.0.2
Already downloaded: /Users/druane/Library/Caches/Homebrew/downloads/889118734bba1cb22bbee93b861cd39ca47540f0d55a0b7567e730875798ad69--openjdk-22
==> Pouring openjdk--22.0.2.ventura.bottle.tar.gz
🍺  /usr/local/Cellar/openjdk/22.0.2: 602 files, 330.0MB
==> Installing selenium-server
==> Pouring selenium-server--4.23.0.ventura.bottle.tar.gz
==> Caveats
To start selenium-server now and restart at login:
  brew services start selenium-server
Or, if you don't want/need a background service you can just run:
  /usr/local/opt/selenium-server/bin/selenium-server standalone --port 4444
==> Summary
🍺  /usr/local/Cellar/selenium-server/4.23.0: 7 files, 36.2MB
==> Running `brew cleanup selenium-server`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
```

Step 6. After Selenium server is installed it’s time to install tqdm and Pillow via pip. To install TQDM (A python library to develop the CLI progress bar) run:



```
pip install "git+https://github.com/tqdm/tqdm.git@devel#egg=tqdm"
```

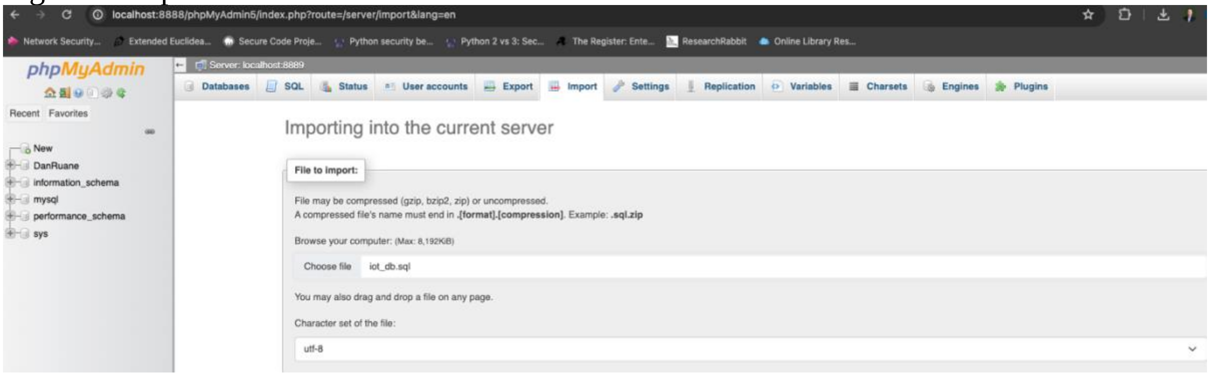
To install pillow (AKA PIL, an image library for python3) just run the command: pip install pillow

Step 7. After all the required libraries are installed, we can now proceed to render the simulation IoT Device Login page. Firstly, take all the files that are in the folder

WWW_HTDocs. You will find this after unzipping the project file and place them in the MAMP HTDocs directory. This is usually located at /applications/MAMP/htdocs

Step 8. Next, start the MAMP application and navigate to http://localhost:8888/MAMP/?language=English. Here you can click on tools in the navigation menu and click phpMyAdmin from the drop-down menu. From here you will be greeting with the phpMyAdmin application where you can generate and run your MySQL Db that contains the IoT Device login page password and username. Click import on the phpMyAdmin interface menu and select the file iot_db.sql from the unzipped project folder. (See Fig 3.0)

Fig 3.0 – Import IoT Password and User DB



Importing into the current server

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: (Max: 8,192KiB)

Choose file iot_db.sql

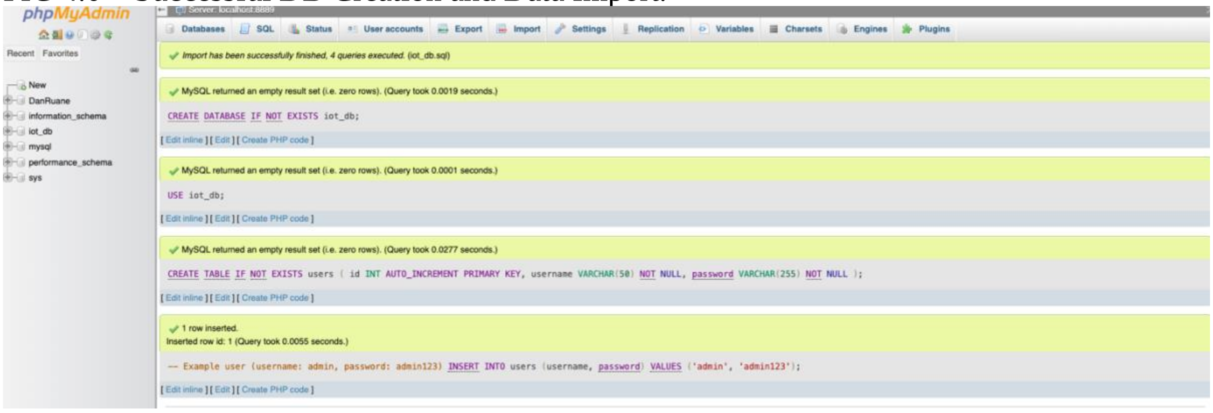
You may also drag and drop a file on any page.

Character set of the file:

utf-8

After the import has run and the new IoT Db has been built you should see the following screen (See Fig 4.0)

FIG 4.0 – Successful DB Creation and Data Import.



✓ *Import has been successfully finished, 4 queries executed.* (iot_db.sql)

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0019 seconds.)

```
CREATE DATABASE IF NOT EXISTS iot_db;
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

```
USE iot_db;
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0277 seconds.)

```
CREATE TABLE IF NOT EXISTS users ( id INT AUTO_INCREMENT PRIMARY KEY, username VARCHAR(50) NOT NULL, password VARCHAR(255) NOT NULL );
```

[Edit inline] [Edit] [Create PHP code]

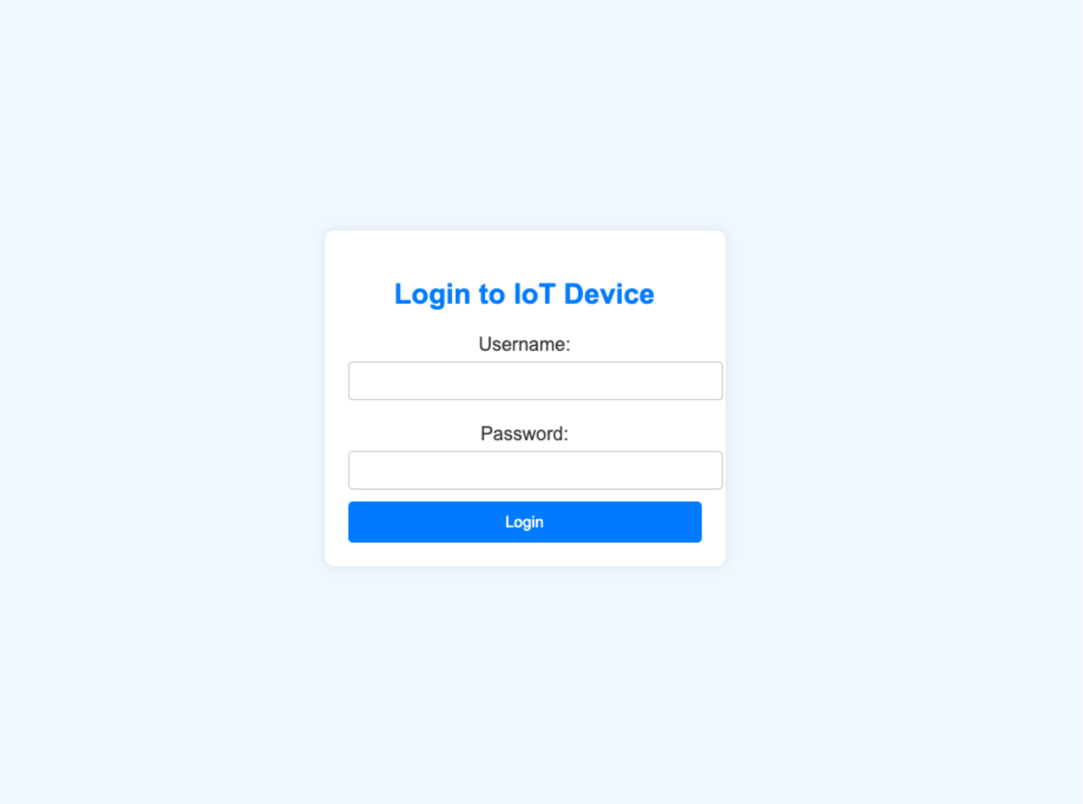
✓ 1 row inserted.
Inserted row id: 1 (Query took 0.0055 seconds.)

```
-- Example user (username: admin, password: admin123) INSERT INTO users (username, password) VALUES ('admin', 'admin123');
```

[Edit inline] [Edit] [Create PHP code]

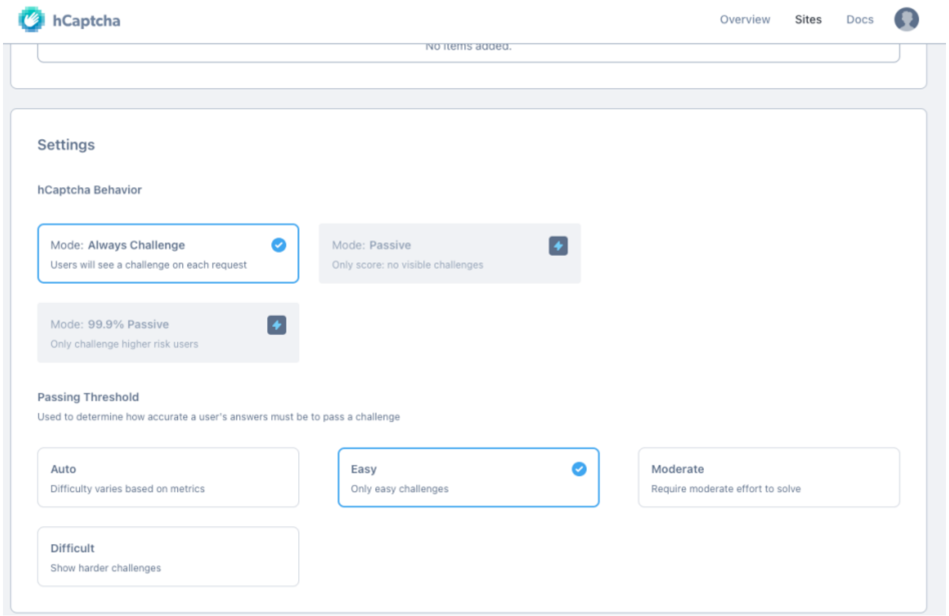
Step 9. You should now be able to surf to the simulation IoT Login page here http://127.0.0.1:8888/index.html. Please ensure to use the localhost IP or the hCaptcha will have trouble displaying as it doesn't like to run on localhost. (See fig 5.0)

FIG 5.0 IoT Simulation Login Page – No Security Measures



Step 10. Setup an account on hCaptcha to get your site and secret keys and select free account https://www.hcaptcha.com/pricing (See fig 6.0). You are now ready to start the Mirai Botnet attack simulation.

FIG 6.0 hCaptcha Account API options



Settings

hCaptcha Behavior

- ☑ Mode: Always Challenge — Users will see a challenge on each request
- Mode: Passive — Only score: no visible challenges
- Mode: 99.9% Passive — Only challenge higher risk users

Passing Threshold

Used to determine how accurate a user's answers must be to pass a challenge

- Auto — Difficulty varies based on metrics
- ☑ Easy — Only easy challenges
- Moderate — Require moderate effort to solve
- Difficult — Show harder challenges

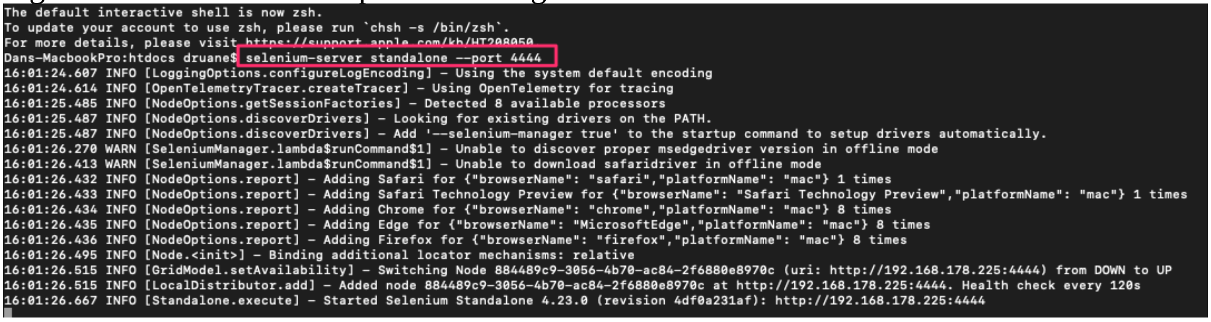
2 Running Experiment 1, 2 and 3

To start the experiment number one, go to the htdocs folder and ensure that there are three index.html files. The first will be called index.html the second will be called index2.html and the third will be index3.html. For each experiment one, two and three. The index file that corresponds to that experiment number must be renamed to index.html. So if we are running experiment 3 we rename index3.html to index.html and rename the old index.html to index2.html as that was the last experiment we ran.

After ensuring that that is the case, we can continue to run Experiment/Simulation 1

Step 1. Enable and run the selenium webserver and chrome driver by running the following command from the htdocs folder: selenium-server standalone – port 4444 (See Fig 7.0)

Fig 7.0 – Selenium Sever Up and Running



```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Dans-MacbookPro:htdocs druane$ selenium-server standalone --port 4444
16:01:24.607 INFO [LoggingOptions.configureLogEncoding] - Using the system default encoding
16:01:24.614 INFO [OpenTelemetryTracer.createTracer] - Using OpenTelemetry for tracing
16:01:25.485 INFO [NodeOptions.getSessionFactories] - Detected 8 available processors
16:01:25.487 INFO [NodeOptions.discoverDrivers] - Looking for existing drivers on the PATH.
16:01:25.487 INFO [NodeOptions.discoverDrivers] - Add '--selenium-manager true' to the startup command to setup drivers automatically.
16:01:26.270 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper msedgedriver version in offline mode
16:01:26.413 WARN [SeleniumManager.lambda$runCommand$1] - Unable to download safaridriver in offline mode
16:01:26.432 INFO [NodeOptions.report] - Adding Safari for {"browserName": "safari","platformName": "mac"} 1 times
16:01:26.433 INFO [NodeOptions.report] - Adding Safari Technology Preview for {"browserName": "Safari Technology Preview","platformName": "mac"} 1 times
16:01:26.434 INFO [NodeOptions.report] - Adding Chrome for {"browserName": "chrome","platformName": "mac"} 8 times
16:01:26.435 INFO [NodeOptions.report] - Adding Edge for {"browserName": "MicrosoftEdge","platformName": "mac"} 8 times
16:01:26.436 INFO [NodeOptions.report] - Adding Firefox for {"browserName": "firefox","platformName": "mac"} 8 times
16:01:26.495 INFO [Node.<init>] - Binding additional locator mechanisms: relative
16:01:26.515 INFO [GridModel.setAvailability] - Switching Node 884489c9-3056-4b70-ac84-2f6880e8970c (uri: http://192.168.178.225:4444) from DOWN to UP
16:01:26.515 INFO [LocalDistributor.add] - Added node 884489c9-3056-4b70-ac84-2f6880e8970c at http://192.168.178.225:4444. Health check every 120s
16:01:26.667 INFO [Standalone.execute] - Started Selenium Standalone 4.23.0 (revision 4df0a231af): http://192.168.178.225:4444
```

Step 2. Setup your virtual environment by running python -m venv myenv
Step 3. Enable and activate the virtual environment by running source myenv/bin/activate
Step 4. Ensure that pip has installed Selenium Webdriver manager by running pip install selenium webdriver-manager

Step 5. Ensure that pip has installed PIL by running pip install pillow

Step 6. Ensure that pip has installed tqdm by running pip install tqdm

Step 6. Close any programs that you do not need running as the next part of the experiment will require as much processing power as your Mac can handle.

Step 7. Start the Mirai Botnet simulation by running the python script using the selenium libraries from the htdocs folder: python3 mirai_simulation_selenium.py

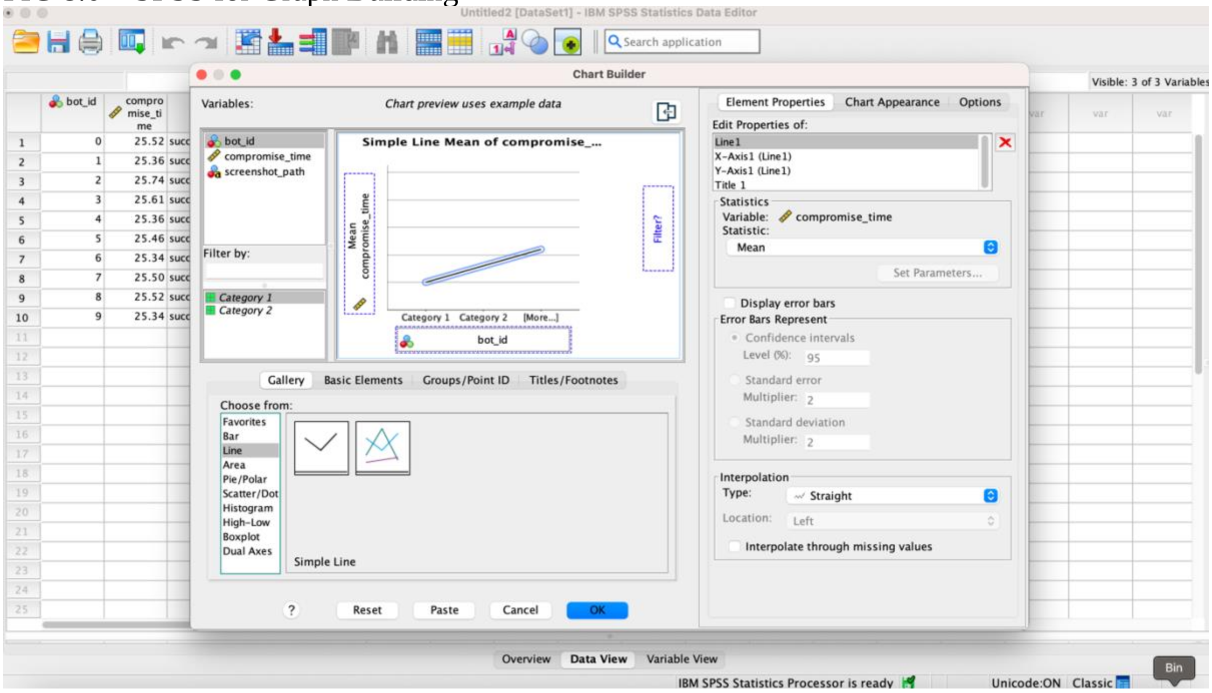
Step 8. The botnet attack simulation will begin, and you may notice 10 Chrome browser windows open and close rapidly. This is the selenium webserver and chrome driver running the brute force attacks in a multithreaded manner with the chrome browser attacking the IoT Login form on this page http://127.0.0.1:8888/index.html

The following second and third experiment can use steps 1-7 to run each experiment. Just ensure to change the index file for each experiment and use the numbered index.html that corresponds with the experiment number.

3 Analysing The Results

To analyse the results, I have used SPSS to generate graphs and present the data in a clear and simple way. An example of the graphs and corresponding data is below. (See fig 8.0 + 9.0)

FIG 8.0 – SPSS for Graph Building



Untitled2 [DataSet1] - IBM SPSS Statistics Data Editor

	bot_id	compromise_time	
1	0	25.52	succ
2	1	25.36	succ
3	2	25.74	succ
4	3	25.61	succ
5	4	25.36	succ
6	5	25.46	succ
7	6	25.34	succ
8	7	25.50	succ
9	8	25.52	succ
10	9	25.34	succ

Chart Builder

Variables: bot_id, compromise_time, screenshot_path

Chart preview uses example data

Simple Line Mean of compromise_...

Choose from: Line — Simple Line

Edit Properties of: Line1, X-Axis1 (Line1), Y-Axis1 (Line1), Title 1

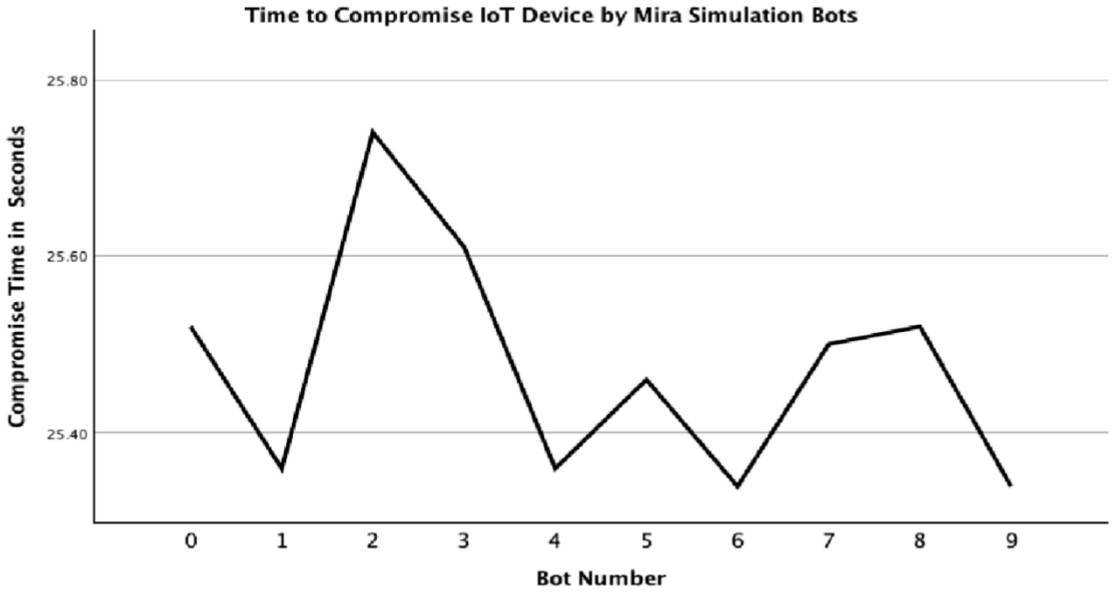
Statistics
Variable: compromise_time
Statistic: Mean

□ Display error bars

Error Bars Represent
● Confidence intervals — Level (%): 95
○ Standard error — Multiplier: 2
○ Standard deviation — Multiplier: 2

Interpolation
Type: Straight
Location: Left
□ Interpolate through missing values

Fig 9.0 Output from SPSS


Time to Compromise IoT Device by Mira Simulation Bots
Compromise Time in Seconds
Bot Number