

Configuration Manual

MSc Research Project
Cybersecurity

Simon Lowry
Student ID: x21168938

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland
MSc Project Submission Sheet



School of Computing

Student Name: Simon Lowry

Student ID: x21168938

Programme : MSCCYBE_JANOL_O **Year :** 2 of 2.

Module: Research Project

Lecturer: Michael Pantridge

Submission Due Date: 14/12/2023

Project Title: Augmenting Privacy Through Metadata Wiping

Word Count: 1736 words **Page Count:** 9

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: *Simon Lowry*

Date:

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Simon Lowry
Student ID: x21168938

1 Introduction

MetadataWiper is a privacy tool for the wiping of metadata from 4 different file types. These are some of the most commonly used file types across the globe and are as follows: .jpg, .docx, .pdf & .xlsx. The remainder of the document will outline the languages used and their respective versions, the system on which this research was conducted, the steps necessary to setup both the backend and frontend of the application and get it running. It will also detail what it looks like for successful completion of a flow of operations on the various file types from the user interface perspective and also what errors can occur and how they are displayed on the UI. Finally, then there will be some additional useful information for running the application that goes beyond what's listed.

2 System configuration

The specifications of the system throughout the research are as follows:

- Operating system: Windows 11
- Processor: Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz, 1896 Mhz, 4 Core(s), 8 Logical Processor(s)
- System Compatibility: 64-bit
- Hard Disk: 475 GB SSD
- RAM: 8GB

Pycharm was used as the IDE for the python code and visual studio code for the ReactJs.

3 Running the Application

Languages used in the Application and their Versions

The languages required for running the application are Python and ReactJS. The python version that the project was run at was 3.8.3

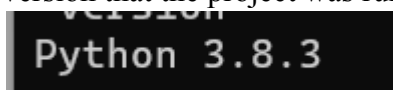
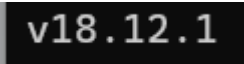
A screenshot of a terminal window with a dark background. The text 'Python 3.8.3' is displayed in a light-colored font.

Figure 1. Output from command run: `python -version`

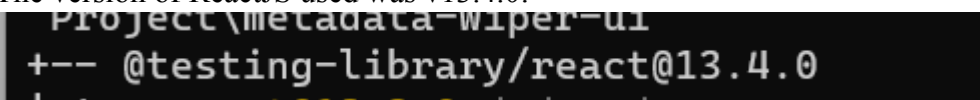
React requires node to be installed as well prior to react. Node is currently at: v18.12.1



```
v18.12.1
```

Figure 2. Output from command run: `node -v`


The version of ReactJS used was v13.4.0:



```
Project\metadata-wiper-ui  
+-- @testing-library/react@13.4.0
```

Figure 3. Output from command run: `npm list react`

A `properties.py` file must be added to the `metadataWiperBackend`. This has been kept separate from the repo for privacy reasons. Please add the following properties to the file:



```
FILE_DIRECTORY = 'C:/YOUR_PATH/Project/metadataWiperBackend/media/files/'  
VT_API_KEY = '123456ABC_YOUR_API_KEY_HERE'  
VT_API = 'https://www.virustotal.com/api/v3/files/'
```

Figure 4. `properties.py` file with required properties to run backend server

Replace `YOUR_PATH` with wherever you have placed the project for the `FILE_DIRECTORY` property. This will serve as the directory for files which are uploaded to the backend server for metadata wiping.

Beyond that, the other two properties are for running the virus total check which occurs after upload. VirusTotal requires an API key in order to run. To generate your own API key please sign up to the Virus Total community here:
<https://www.virustotal.com/gui/join-us>

When you have completed the signup, you will see your API key here:

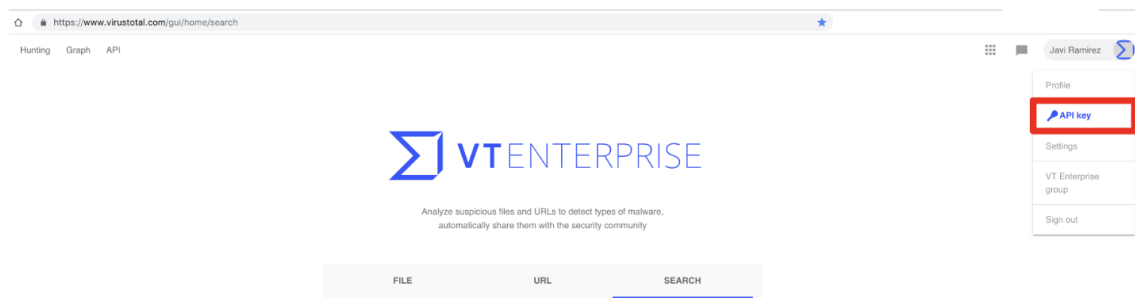



Figure 5. UI for obtaining API key on virus total

It will be available in the menu that pops up when you click on your name and then click on the option API Key.

Alternatively, if either of the markers want to use my API key, please reach out to me via email or microsoft teams and that will be sent onto you.

When you have the API key, simply place within single quotes and replace what's currently there:



```
'123456ABC_YOUR_API_KEY_HERE'
```

Virus Total checks to see if any of the antivirus solutions have marked an uploaded file as malicious. If 5 or more of the AV solutions have marked the uploaded file as malicious, the

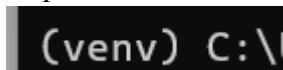
file is rejected by the application, immediately deleted and the message is output to the screen.

Next we look at setting up the python backend server to run and then move onto the ReactJS user interface.

An important way to mitigate against python package vulnerabilities is to run Python in a virtualized environment. Thus, if a given package is vulnerable, this reduces the likely impact and protects other projects and the main host environment. Django does also offer a way of doing this itself with the following commands:

```
source ./django_env/bin/activate
```

This needs to be run on the command line while in the directory denoted as metadataWiperBackend. This will start up the virtualized environment which is already created and enclosed on the repo. You should now see (venv) in the terminal prior to the output of the current directory like so:



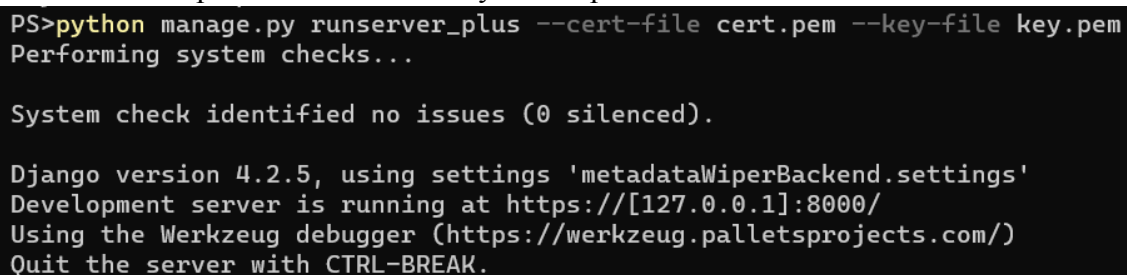
```
(venv) C:\
```

Figure 6. Denoted marker of virtual environment running on terminal

Then to start the backend server run the following command:

```
python manage.py runserver_plus --cert-file cert.pem --key-file key.pem
```

If this has been performed successfully the output on the command line should look like:



```
PS>python manage.py runserver_plus --cert-file cert.pem --key-file key.pem
Performing system checks...

System check identified no issues (0 silenced).

Django version 4.2.5, using settings 'metadataWiperBackend.settings'
Development server is running at https://[127.0.0.1]:8000/
Using the Werkzeug debugger (https://werkzeug.palletsprojects.com/)
Quit the server with CTRL-BREAK.
```

Figure 7. Output from backend server while running on terminal

Here we can see the backend development server is up and running at 127.0.0.1:8000. Next we look to start the user interface. Open up another terminal to run the user interface operating. Change directory to inside of the folder denoted as metadata-wiper-ui. In this directory run the following command:

```
npm start
```

This should open up the user interface for the application at localhost:

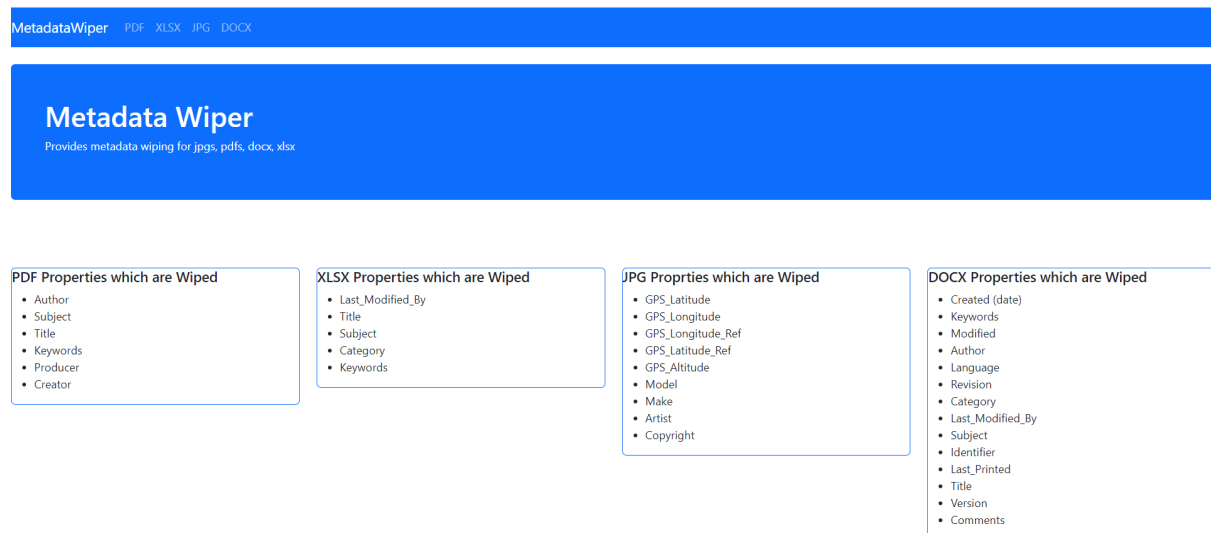


Figure 8. User interface and home screen for metadata wiper web application

This is how the UI homepage currently looks. The navbar shows the four different file types available as options to perform metadata wiping on. The UI for each of these pages is uniform and kept as simple as possible. No unnecessary bloat and employing economy of mechanism here as well.

Also, the terminal will output the following:

```

Compiled successfully!

You can now view metadata-wiper-ui in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully

```

Figure 9. Output on terminal for successful startup of react UI for metadata wiper frontend

Next we look at some of the functionality. An example of what happens when you navigate to the JPEG option on the navbar is shown below:

MetadataWiper PDF XLSX JPG DOCX

JPG Metadata Wiper

Wipes the metadata of JPG files

Enter JPG file here:

Choose File No file chosen

Submit

JPG Properties which are Wiped

- GPS_Latitude
- GPS_Longitude
- GPS_Longitude_Ref
- GPS_Latitude_Ref
- GPS_Altitude
- Model
- Make
- Artist
- Copyright

Figure 10. JPG UI on metadata wiper

A simple UI presents the option to enter a file of .jpg type for performing metadata wiping.

Next, enter a file and hit the submit button. The UI is configured here to help with the selection of .jpg files and will only display jpg files when you select the “Choose File” button. This does not prevent the user from sending other file types through other means but is merely a convenience and guide for the correct and expected file type for each operation.

Note: If you hit submit without selecting a file, you will prompted to do so:

Enter JPG file here:

Choose File No file chosen

Submit

! Please select a file.

Figure 11. UI prompt message for when no file has been entered for attempting file submission

Upon successful completion of metadata wiping the UI will present the following success message in repent:

Enter JPG file here:

Choose File Canon_40D_...HAS_GPS.jpg

Submit

Metadata successfully removed from file. Downloading modified file.

Figure 12. Success message upon completion of metadata wiping for jpg files.

The file will automatically download and be in your Downloads folder on a windows systems. The same message will be displayed for each file type upon completion of metadata wiping.

Each of the metadata wiper services for every file type are preceded in their operations by a file name validator. This examines both the filename and the file type provided. It ensures that the expected filetype is what's expected. All files with more than one dot/period are rejected to protect against double barrel file names masking the real file type. File names over 30 characters are rejected. File sizes can not be longer than 20mb. File names can only contain alphanumeric characters, underscores, hyphens or spaces.

Enter JPG file here:

Choose File 1238297586...55b03_w(.jpg)

Submit

Error: filenames must only contain alphanumeric characters, underscores, hyphens and a single dot/period.

Here we have a file which does not meet the filename requirements and contains a bracket character and is thus rejected. As mentioned, these are the same requirements applied across each of the metadata wiper services and will produce the same error messages.

Additional Information

The APIs for the backend are as follows:

Path	Parameter Name	Allowed Parameter Type
api/jpeg/	image	.jpg (file)
api/pdf/	pdf_file	.pdf (file)
api/docx/	docx_file	.docx (file)
api/xlsx/	xlsx_file	.xlsx (file)

Any of these can be appended to 127.0.0.1:8000/ and used to send http requests whether that be by a curl command or through the software tool postman. They are all POST APIs and they all only accept one parameter.

Here's a sample request made via postman on the api/pdf/ endpoint:

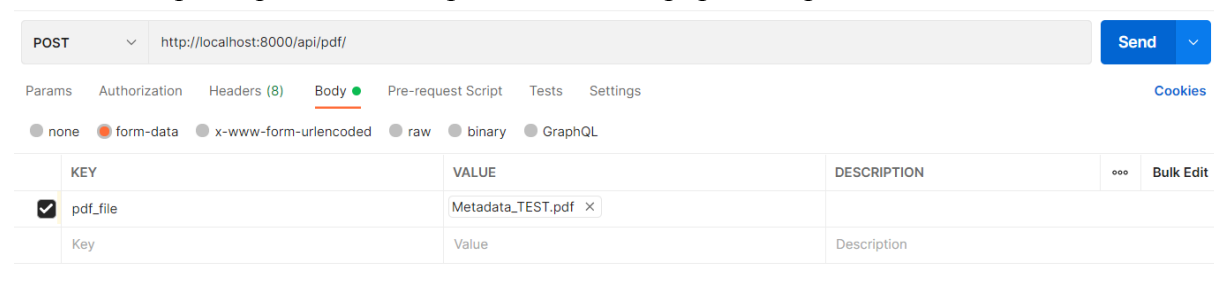


Figure 13. Shows a sample request on postman to api/pdf endpoint

As shown here, the parameter is set to key value of pdf_file and the value is set to a file and that file is called Metadata_TEST.pdf. Please enter your own pdf here to use this API and strip it's metadata.

Upon successful completion of the request, a 200 OK will be returned along with with the pdf file itself and it's metadata stripped as shown below:



Figure 14. Shows the output of a successful request to metadata wiper /api/pdf API

If there's an attempt to make a request to any of these APIs and the backend server is not running, the response will look like this:

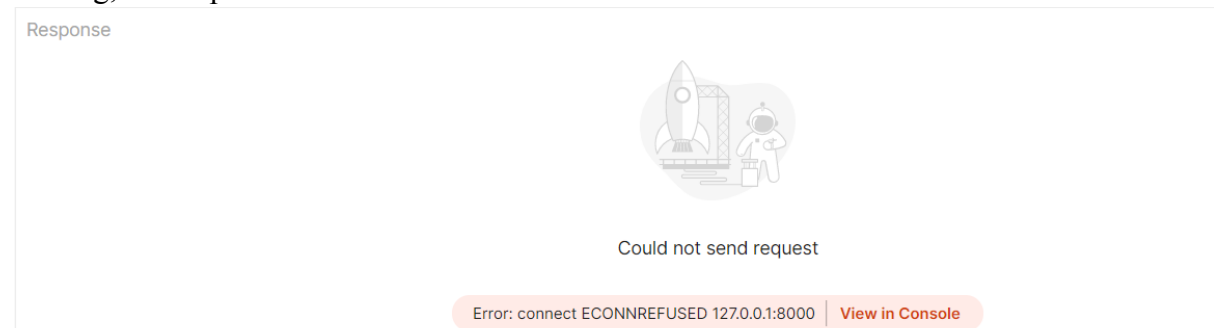


Figure 15. Displays a failure to connect to the backend

It will reply with an ECONNREFUSED on the backend address of 127.0.0.1:8000.