CYT245. Midterm Project _Summer2024, 15%

Below you see the description of 2 options for a home project.

To complete your Midterm Project, you need to select ONE of them and work on the selected project only.

Project 1 is Individual.

Project 2 is either Individual or Teamwork (See the difference in scope below).

You are asked to enter student names below:

1. Ishan Aakash Patel	2.
3.	4.

Team leader is _____Ishan_____

Only one submission from the team is expected. It will be done by the current team leader; however this role should be rotated from one teamwork to another. Screen0 should be made on the Leader's computer.

Project 1 – individual, 15%

Objective: Learn AlienVault tools

Sources of information:

OTX User Guide located at

https://cybersecurity.att.com/documentation/resources/pdf/otx-user-guide.pdf

Sample of OTX alert

https://otx.alienvault.com/pulse/642d624ccd3a7cca31c9e252?utm_userid=tato1234&utm_medium=InProduct&utm_source=OTX&utm_content=Email&utm_campaign=new_pulse_from_following

Task 1. Learn Open Threat Exchange (OTX) from AlienVault – 8%

Read OTX User Guide documentation, in particular pay attention to the section Getting Started with OTX. From this reading you learn how to read and understand the content of Alien Vault Alerts.

Do practice on your ability to understand OTX information. Connect to the sample of OTX alert at the link:

https://otx.alienvault.com/pulse/642d624ccd3a7cca31c9e252?

Examine information from the main page. Write comments you can make in regard to Summary Description of the threat.

Then proceed with IOCs' list and detailed analysis:

- Name the different categories and types of IOCs which can be observed in the list. Use more detailed classification of types of IOCs which is given in the Guide (page 31).
- For each category, take a sample of IOC and proceed to details.
- Comment on details, including External Sources, when applicable. Proceed as deep as you can.

Document results of your analysis:

- Main page analysis
- IOCs list analysis
- Samples of IOCs analysis

Use the most professional language in your document.

Task 2. Objective: Exploratory Data Analysis – 7%

Sources of information:

Data-Driven security, Chapter 3, start from "Homing in on a Question". This is in continuing the Assignment 1 work on IP Reputation Database.

In this Part you will study the influence of two characteristics present in AlienVault IP reputation database – that is Risk and Reliability. The method which is described in the book considers the possibility to prioritize records from large data input before they will be loaded to your company SIEM or IPS system for future analysis.

Inside the text you see the Python code you may run to automate and visualize your results. Run the code and make comments on the resulting images. This is the list of Python Listings and Resulting Figures:

- Listing 3.20, Figure 3-9
- Listing 3-23, Figure 3-12
- Listing 3-25, Figure 3-14
- Listing 3-27, Figure 3-16

Run the code, make screenshots and meaningful comments, and include them in your report. You may be in need of making changes in Python code. The sample of customized code is included.

Submission content

0-screen

Task 1 – document

Task 2 – Screenshots and comments

Online submission requirements

Make online submission to BB.

ZIP the script folder and MS Word document with screenshots.

Name the file you will uploading as indicated below. The name must include:

- Course ID (CYT245)
- What is this (e.g. lab1, assignment 1, etc)
- Author's name

Sample: CYT245_Lab1_Peter

Note: submissions that do not follow the requirements will not be accepted

Project 2 – individual or teamwork

Objectives: Practical Usage of MITRE ATT&CK Navigator

Pre-requisites:

- Understanding of the MITRE ATT&CK Matrix tools
- Awareness of ATT&CK Navigator basics
- Certain level of programming skills including Python

Link to locate and download the Navigator:

https://mitre-attack.github.io/attack-navigator/

You can work with either web-based or locally installed version.

In this project you work on samples of practical usage of the Navigator for important Threat Intelligence tasks.

Start with the principal video to capture the idea:

MITRE Practical Use Cases - YouTube

In this video you see explanation of 3 Use Cases where the ATT&CK Matrix and Navigator are used to address common threat intelligence tasks. Your job is to learn the methods which are explained in the video and reproduce them on your own computer.

The Use Cases are:

- Attack Detection
- Attack Prevention
- Threat Modeling

Start with Use Case 1. Attack detection.

Pay attention to the need to integrate ATT&CK scenarios with data sources and sensors in order to make the results actionable. It means that you need the ability to build the modules which then can be integrated into your company protection mechanisms.

Given the complexity of this project, there is a differentiation between individual and teamwork (see below).

Individual work

Use Case 1. Attack Detection – 15%

This use case is split into 3 steps:

Step1 – 5%. Mapping the Matrix Techniques to data sources which can be used to detect the attacks. Learn the method from the following video:

MITRE DeTTECT - Data Source Visibility and Mapping - YouTube

Make screenshots and comment on each step of implementation.

Step 2 – 5%. DeTT&CT : Mapping detection to MITRE ATT&CK

This is presented at

https://blog.nviso.eu/2022/03/09/dettct-mapping-detection-to-mitre-attck/

Here programming work is required. Include the workflow steps, scripts/commands, make screenshots and comments.

Step 3 – 5%. Build the Navigator layer to see the entire detection mechanism. Data source will be integrated into the layer, and JSON object produced.

This activity to some extent is present at previous 2 steps, however it is explained consistently at your principal video:

MITRE Practical Use Cases - YouTube

Learn and run each step on your computer. Make the screenshots and meaningful comments to include into MS Word/PDF document for submission. Your comments should adequately refer to the task of Attack Detection.

The completeness of this task is sufficient to individual work. You may proceed with the submission.

Teamwork option

For 15%:

Complete Use Case 1. Attack Detection.

Work in addition on other 2 Use Cases: Attack Prevention and Threat Modeling from the same principal video.

If you run only the Use Case 1 your score will be **10%.**

Online submission requirements

Make online submission to BB.

ZIP the script folder and MS Word document with screenshots.

Name the file you will uploading as indicated below. The name must include:

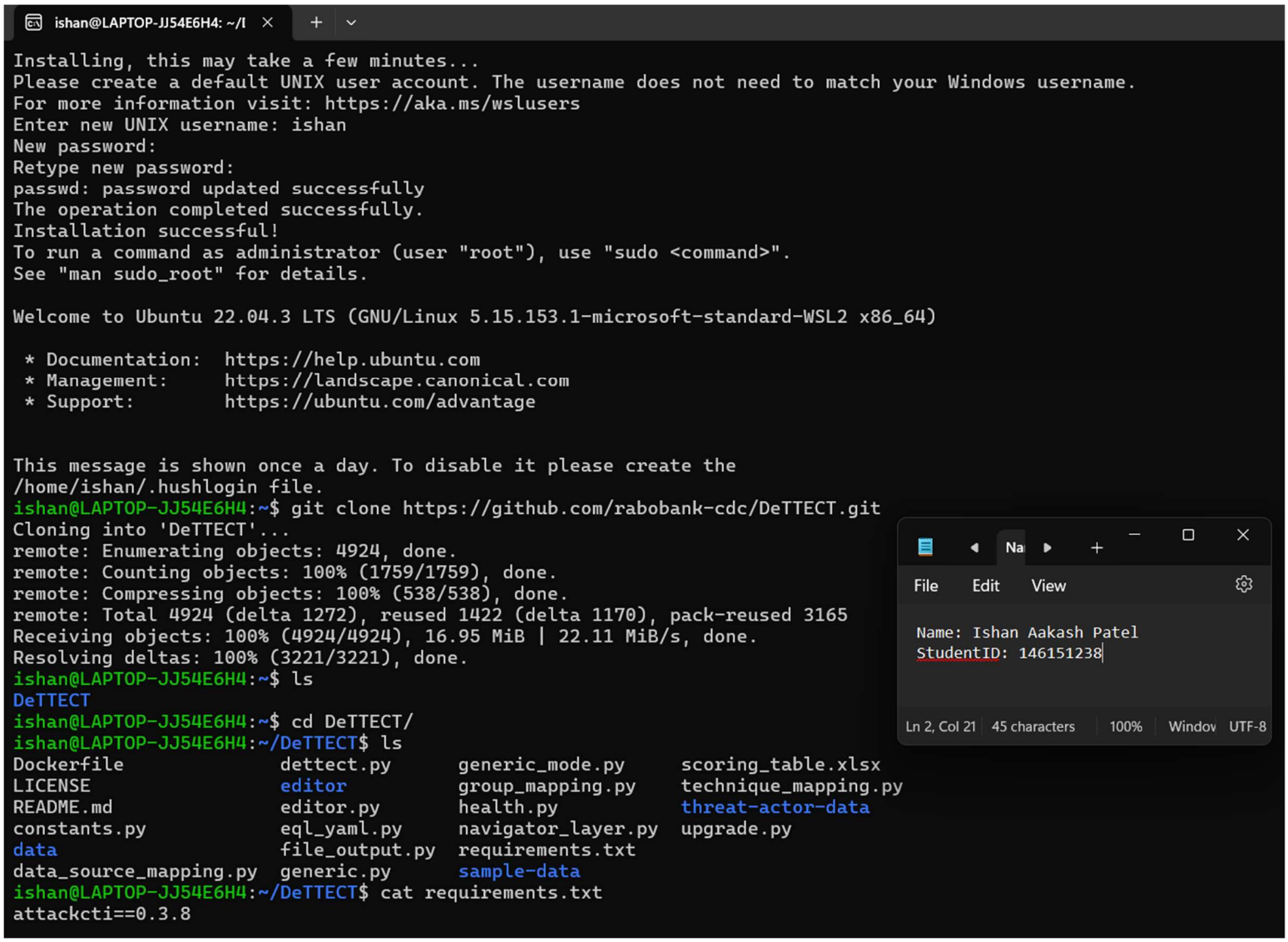
- Course ID (CYT245)
- What is this (e.g. lab1, assignment 1, etc)
- Authors' names

Sample: CYT245_Lab1PeterJohn
Note: submissions that do not follow the requirements will not be accepted

Project – 2
Objectives: Practical Usage of MITRE ATT&CK Navigator

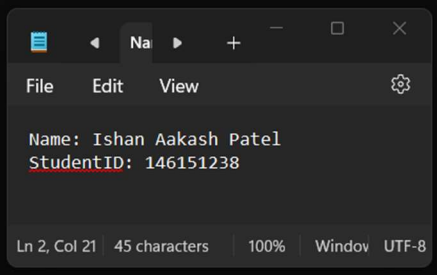
Use Case 1. Attack Detection : DeTT&CT stands for Detect Tactics, Techniques & Combat Threats. This framework has been created at the Cyber Defence Center of Rabobank and is developed and at the time of writing maintained by Marcus Bakker and Ruben Bouman. The purpose of DeTT&CT is to assist blue teams using MITRE ATT&CK to score and compare data log source quality, visibility coverage and detection coverage. By using this framework, blue teams can quickly detect gaps in the detection or visibility coverage and prioritize the ingest of new log sources.

First, I installed WSL in windows, Then I cloned the repository of dettect.

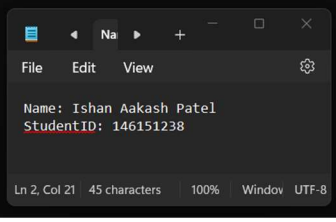


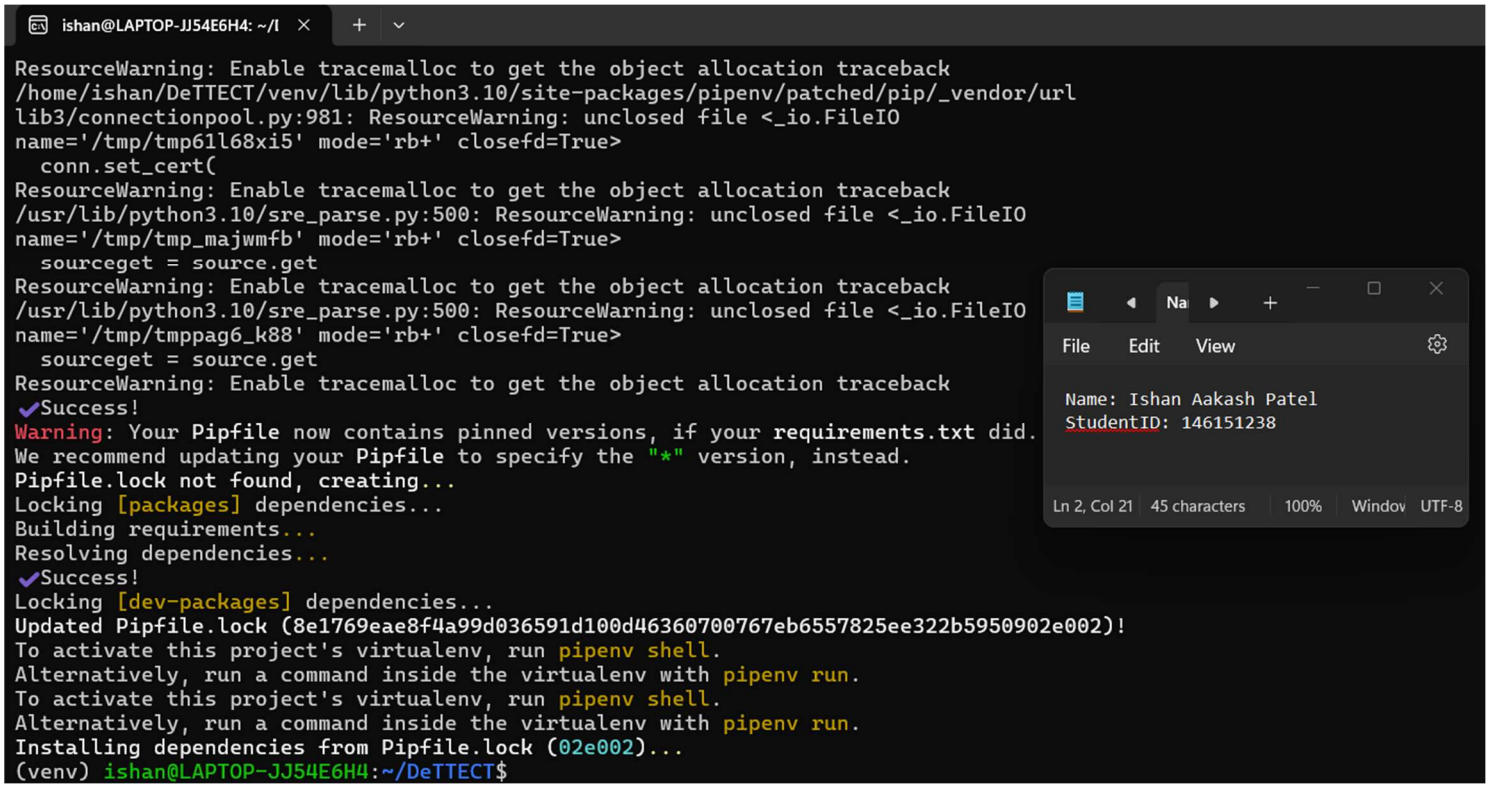
Then I installed pipenv for installing few python libraries which were present requirements.txt.

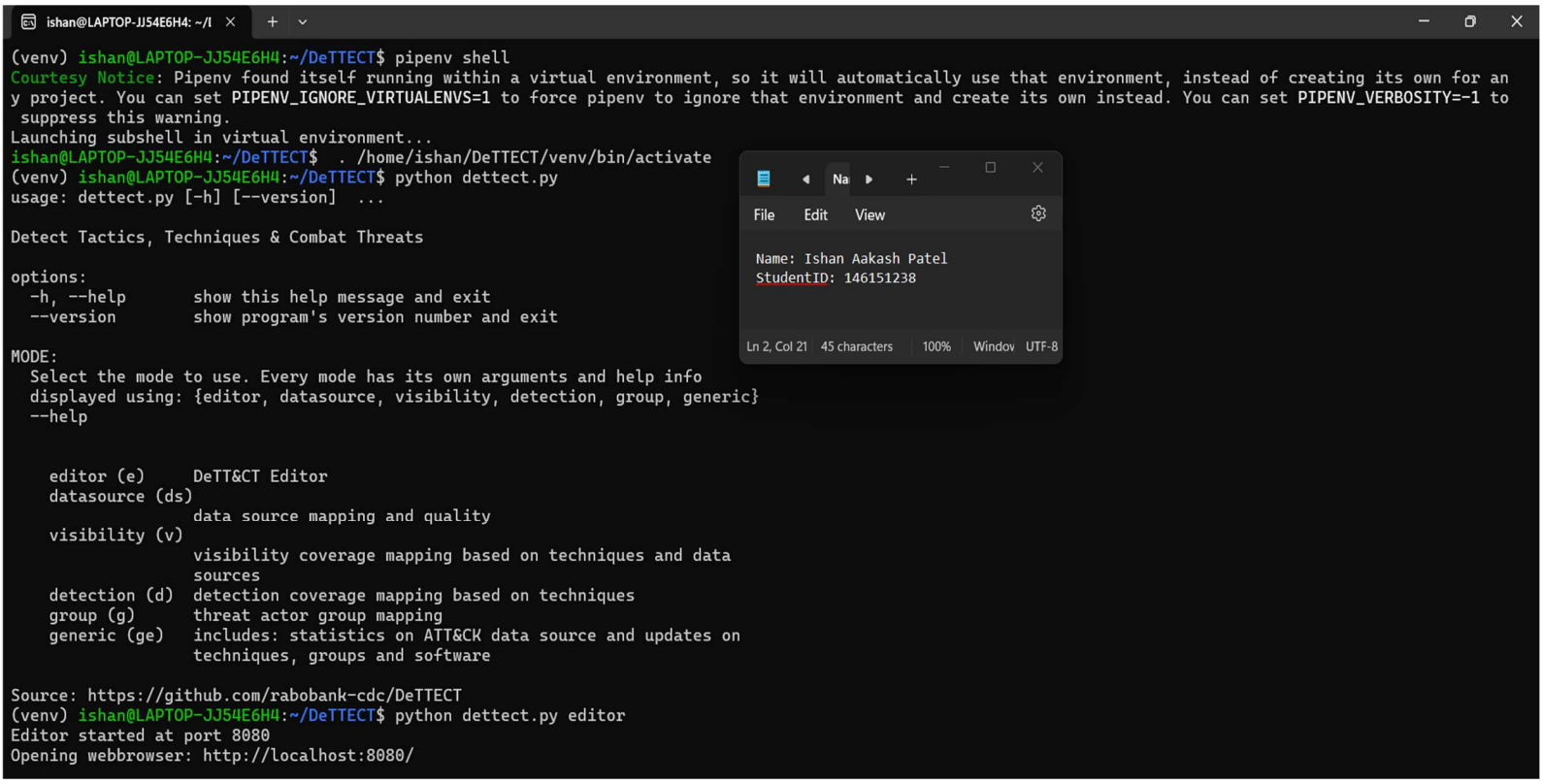
```
ishan@LAPTOP-JJ54E6H4: ~/I
Failing command: /home/ishan/DeTTECT/venv/bin/python3

ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ sudo apt install python3.10-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  javascript-common libexpat1-dev libjs-jquery libjs-sphinxdoc libjs-underscore
  libpython3-dev libpython3.10-dev python3-certifi python3-dev python3-distlib
  python3-filelock python3-pip python3-platformdirs python3-virtualenv
  python3-virtualenv-clone python3-wheel python3-wheel-whl python3.10-dev
  zlib1g-dev
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  python3.10-venv
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 5716 B of archives.
After this operation, 28.7 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 python3.10-venv amd64 3.10.12-1~22.04.3 [5716 B]
Fetched 5716 B in 0s (20.0 kB/s)
Selecting previously unselected package python3.10-venv.
(Reading database ... 31676 files and directories currently installed.)
Preparing to unpack .../python3.10-venv_3.10.12-1~22.04.3_amd64.deb ...
Unpacking python3.10-venv (3.10.12-1~22.04.3) ...
Setting up python3.10-venv (3.10.12-1~22.04.3) ...
ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python3 -m venv venv
ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ source venv/bin/activate
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ pip install pipenv
Collecting pipenv
  Using cached pipenv-2024.0.1-py3-none-any.whl (3.2 MB)
Collecting certifi
  Downloading certifi-2024.6.2-py3-none-any.whl (164 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 164.4/164.4 kB 3.6 MB/s eta 0:00:00
Collecting setuptools>=67
  Using cached setuptools-70.2.0-py3-none-any.whl (930 kB)
Collecting virtualenv>=20.24.2
  Using cached virtualenv-20.26.3-py3-none-any.whl (5.7 MB)
Collecting platformdirs<5,>=3.9.1
  Using cached platformdirs-4.2.2-py3-none-any.whl (18 kB)
Collecting distlib<1,>=0.3.7
  Using cached distlib-0.3.8-py2.py3-none-any.whl (468 kB)
```




```
ishan@LAPTOP-JJ54E6H4: ~/I
Successfully installed certifi-2024.6.2 distlib-0.3.8 filelock-3.15.4 pipenv-2024.0.1 platformdirs-4.2.2 setuptools-70.2.0 virtualenv-20.26.3
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ pipenv install
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environment, instead of creating its own for an
y project. You can set PIPENV_IGNORE_VIRTUALENVS=1 to force pipenv to ignore that environment and create its own instead. You can set PIPENV_VERBOSITY=-1 to
 suppress this warning.
requirements.txt found in /home/ishan/DeTTECT instead of Pipfile! Converting...
/home/ishan/DeTTECT/venv/lib/python3.10/site-packages/pipenv/patched/pip/_vendor/url
lib3/connectionpool.py:981: ResourceWarning: unclosed file <_io.FileIO
name='/tmp/tmppktxg5gr' mode='rb+' closefd=True>
  conn.set_cert(
ResourceWarning: Enable tracemalloc to get the object allocation traceback
/home/ishan/DeTTECT/venv/lib/python3.10/site-packages/pipenv/patched/pip/_vendor/url
lib3/connectionpool.py:981: ResourceWarning: unclosed file <_io.FileIO
name='/tmp/tmp25k4t620' mode='rb+' closefd=True>
  conn.set_cert(
ResourceWarning: Enable tracemalloc to get the object allocation traceback
/home/ishan/DeTTECT/venv/lib/python3.10/site-packages/pipenv/patched/pip/_vendor/url
lib3/connectionpool.py:981: ResourceWarning: unclosed file <_io.FileIO
name='/tmp/tmpyv7501tb' mode='rb+' closefd=True>
  conn.set_cert(
ResourceWarning: Enable tracemalloc to get the object allocation traceback
/home/ishan/DeTTECT/venv/lib/python3.10/site-packages/pipenv/patched/pip/_vendor/url
lib3/connectionpool.py:981: ResourceWarning: unclosed file <_io.FileIO
name='/tmp/tmpo8zwa8_x' mode='rb+' closefd=True>
  conn.set_cert(
ResourceWarning: Enable tracemalloc to get the object allocation traceback
```




ishan@LAPTOP-JJ54E6H4: ~/I
ResourceWarning: Enable tracemalloc to get the object allocation traceback
/home/ishan/DeTTECT/venv/lib/python3.10/site-packages/pipenv/patched/pip/_vendor/url
lib3/connectionpool.py:981: ResourceWarning: unclosed file <_io.FileIO
name='/tmp/tmp61l68xi5' mode='rb+' closefd=True>
conn.set_cert(
ResourceWarning: Enable tracemalloc to get the object allocation traceback
/usr/lib/python3.10/sre_parse.py:500: ResourceWarning: unclosed file <_io.FileIO
name='/tmp/tmp_majwmfb' mode='rb+' closefd=True>
sourceget = source.get
ResourceWarning: Enable tracemalloc to get the object allocation traceback
/usr/lib/python3.10/sre_parse.py:500: ResourceWarning: unclosed file <_io.FileIO
name='/tmp/tmppag6_k88' mode='rb+' closefd=True>
sourceget = source.get
ResourceWarning: Enable tracemalloc to get the object allocation traceback
✔Success!
Warning: Your Pipfile now contains pinned versions, if your requirements.txt did.
We recommend updating your Pipfile to specify the "*" version, instead.
Pipfile.lock not found, creating...
Locking [packages] dependencies...
Building requirements...
Resolving dependencies...
✔Success!
Locking [dev-packages] dependencies...
Updated Pipfile.lock (8e1769eae8f4a99d036591d100d46360700767eb6557825ee322b5950902e002)!
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock (02e002)...
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$
File Edit View
Name: Ishan Aakash Patel
StudentID: 146151238
Ln 2, Col 21 45 characters 100% Windov UTF-8


ishan@LAPTOP-JJ54E6H4: ~/I
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ pipenv shell
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environment, instead of creating its own for an
y project. You can set PIPENV_IGNORE_VIRTUALENVS=1 to force pipenv to ignore that environment and create its own instead. You can set PIPENV_VERBOSITY=-1 to
suppress this warning.
Launching subshell in virtual environment...
ishan@LAPTOP-JJ54E6H4:~/DeTTECT$. /home/ishan/DeTTECT/venv/bin/activate
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py
usage: dettect.py [-h] [--version] ...
Detect Tactics, Techniques & Combat Threats
options:
-h, --help show this help message and exit
--version show program's version number and exit
MODE:
Select the mode to use. Every mode has its own arguments and help info
displayed using: {editor, datasource, visibility, detection, group, generic}
--help
editor (e) DeTT&CT Editor
datasource (ds)
data source mapping and quality
visibility (v)
visibility coverage mapping based on techniques and data
sources
detection (d) detection coverage mapping based on techniques
group (g) threat actor group mapping
generic (ge) includes: statistics on ATT&CK data source and updates on
techniques, groups and software
Source: https://github.com/rabobank-cdc/DeTTECT
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py editor
Editor started at port 8080
Opening webbrowser: http://localhost:8080/
File Edit View
Name: Ishan Aakash Patel
StudentID: 146151238
Ln 2, Col 21 45 characters 100% Windov UTF-8

Create the virtual environment:

python3 -m venv venv

Activate the virtual environment:

source venv/bin/activate

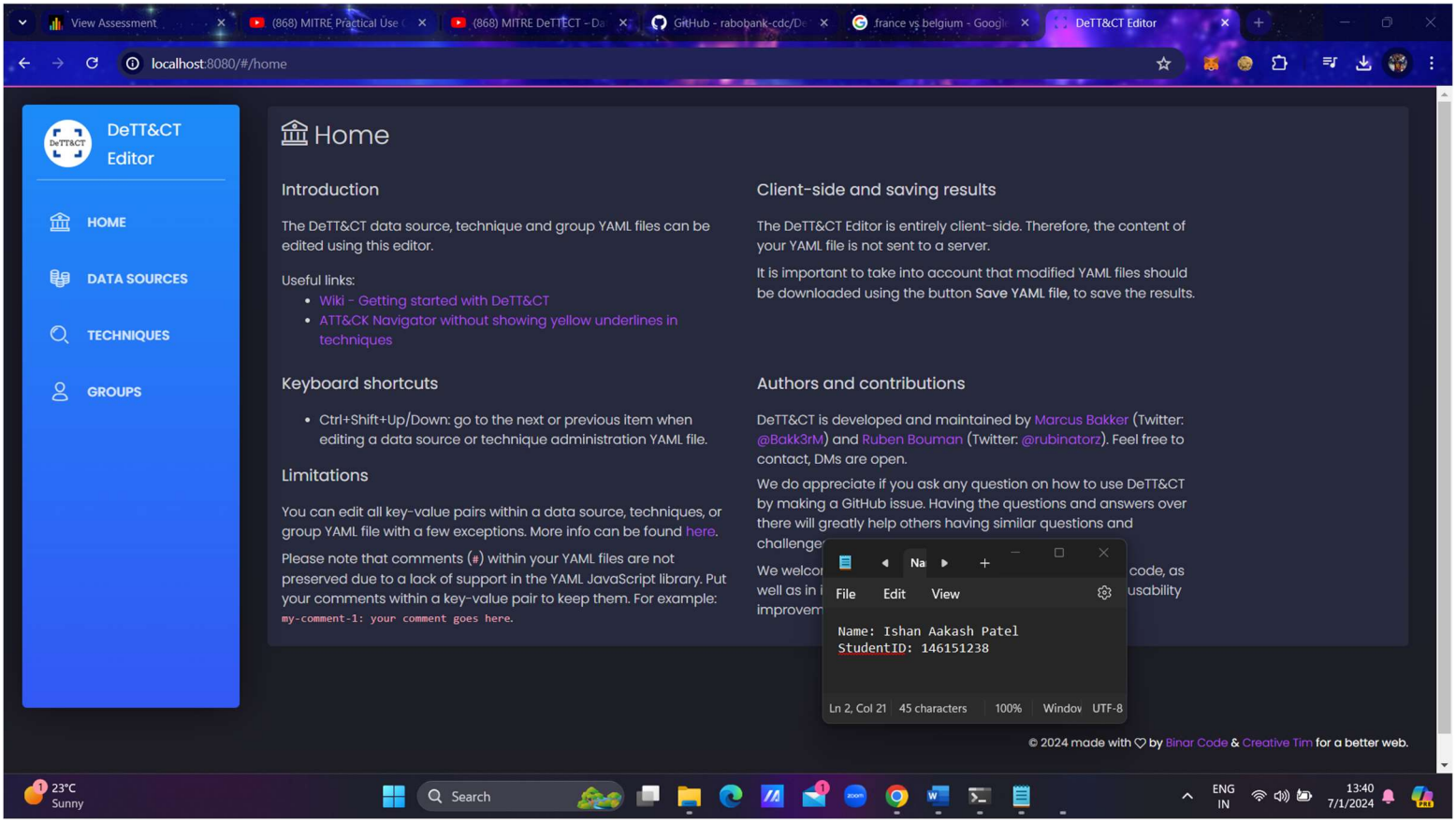
Install pipenv within the virtual environment:

pip install pipenv

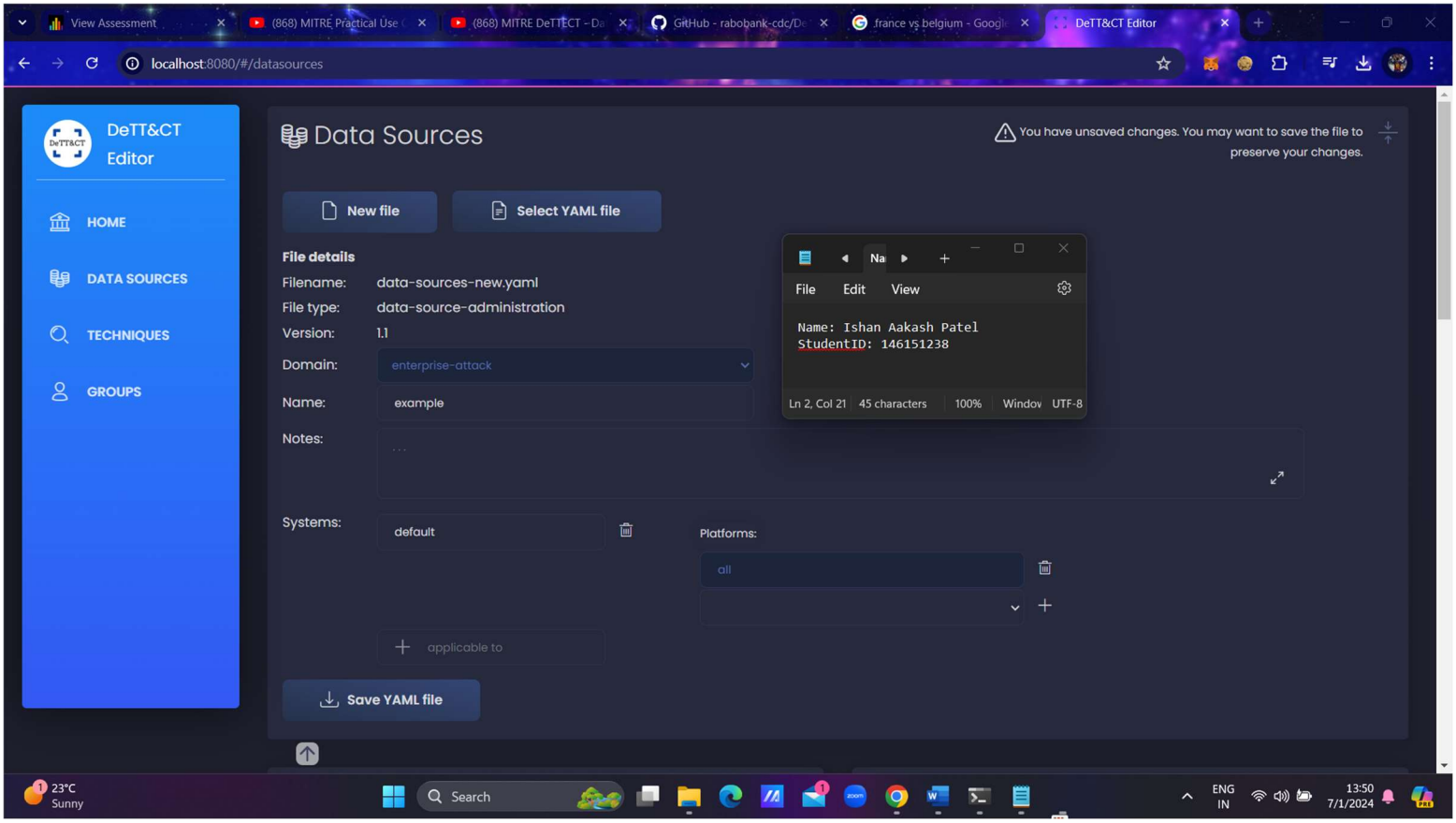
Use pipenv to install your dependencies:

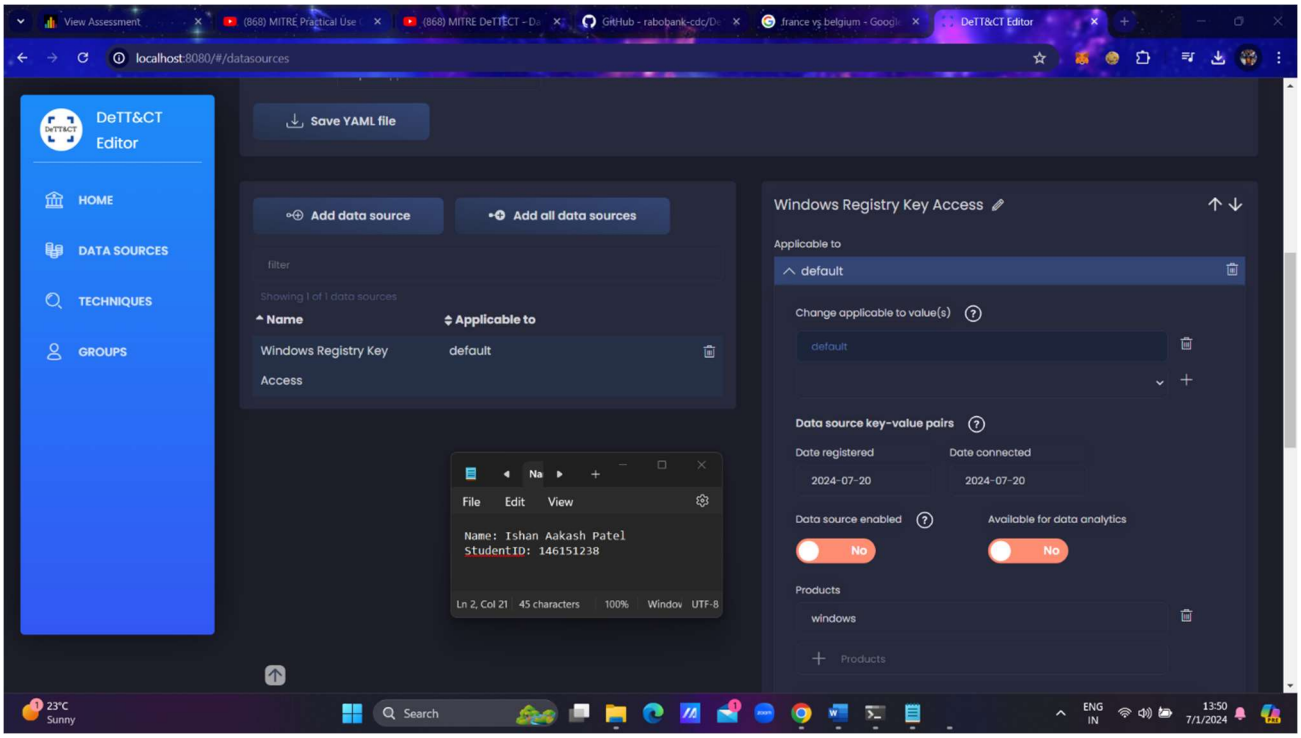
pipenv install

Now go the localhost

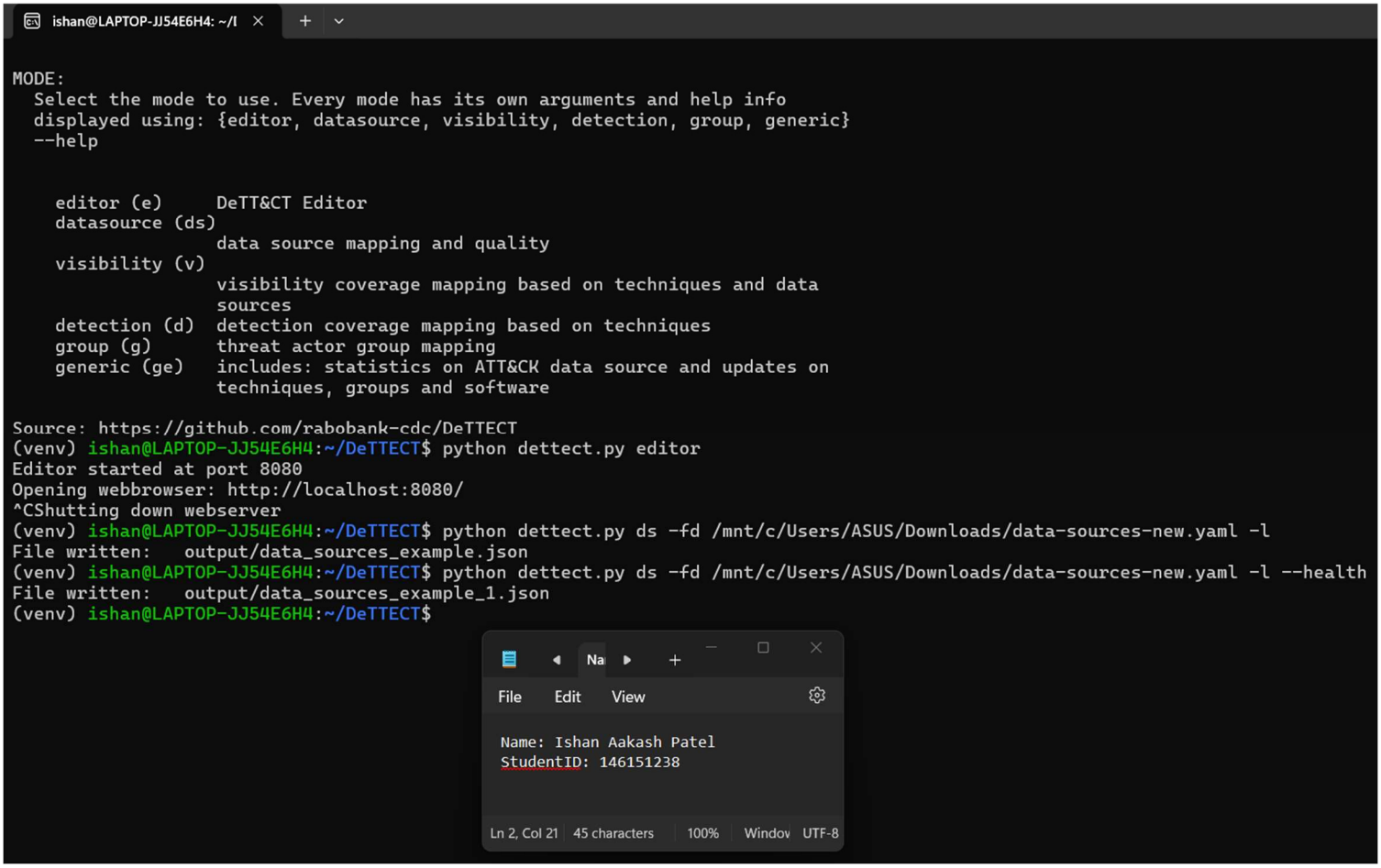


Add Data sources, Techniques, and Groups.

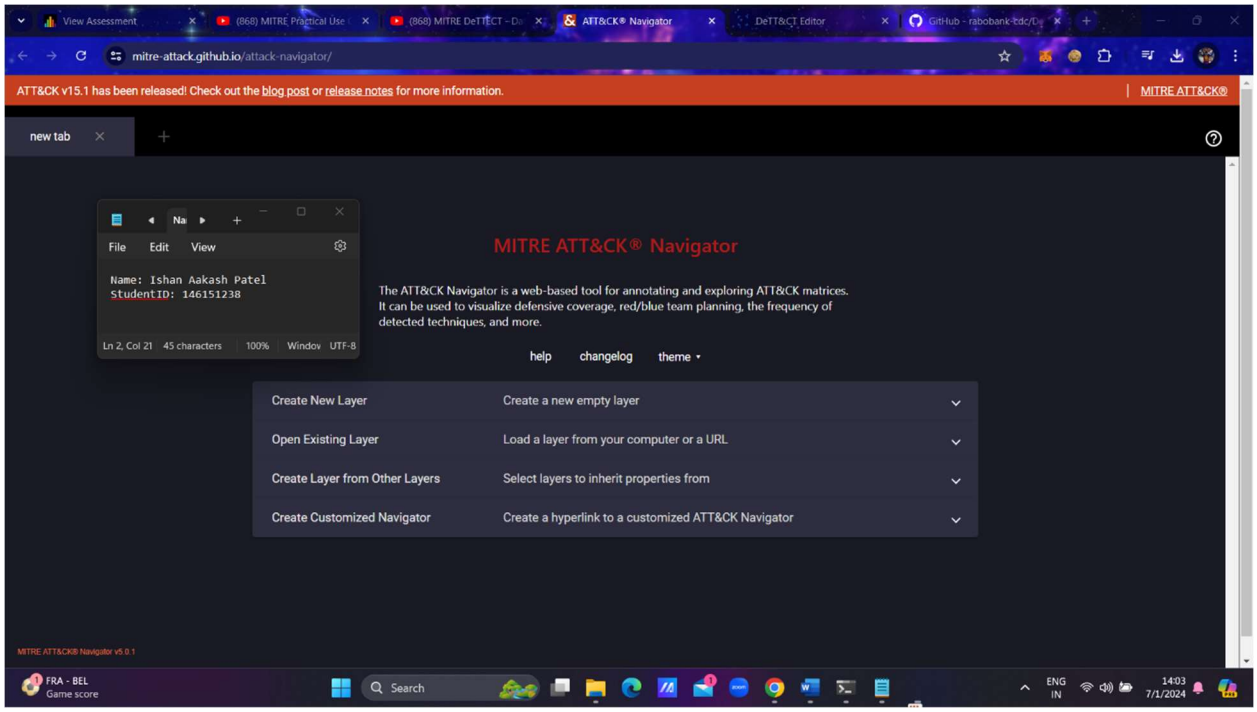


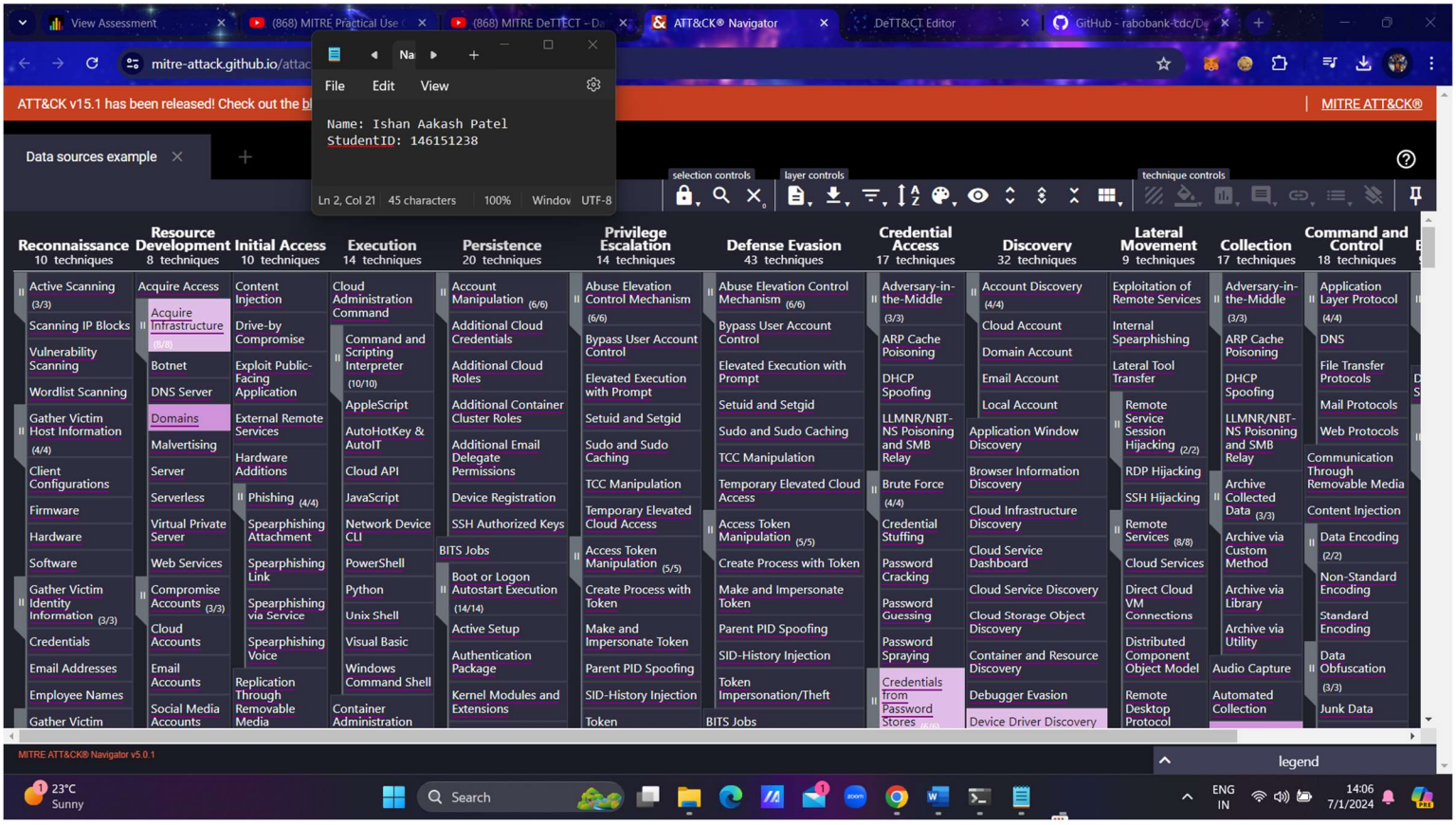


Now after adding the data source, save the yaml file and then convert yaml file .json file.



Now open the json file in MITRE Attack Navigator to analyze the results.



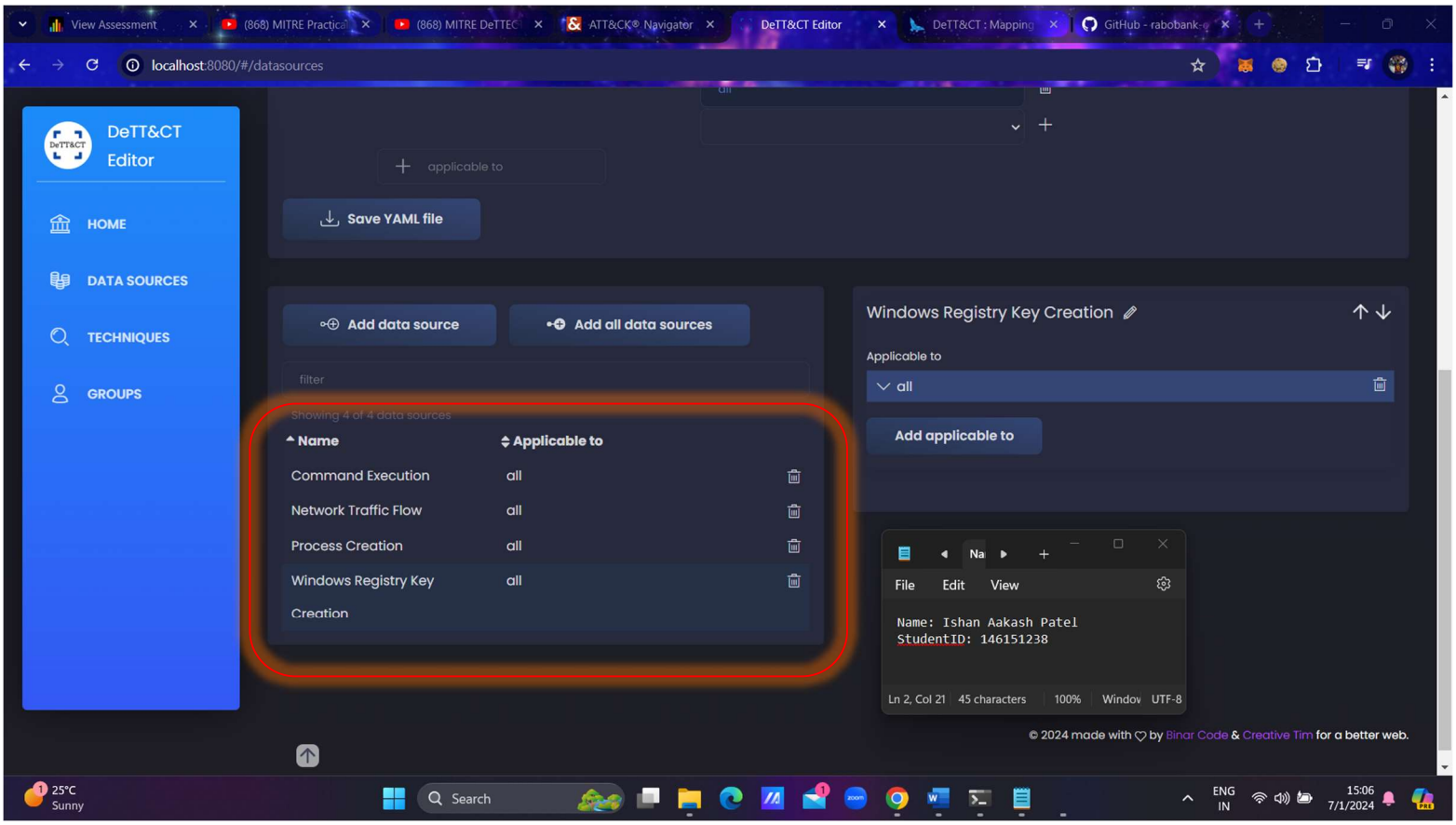


Part – 2

Visibility : Visibility is used within DeTT&CT to indicate if you have sufficient data sources with sufficient quality available to be able to capture evidence for activities associated with ATT&CK techniques. Visibility is necessary to perform incident response, execute hunting investigations and build detections. Within DeTT&CT you can score the visibility coverage per ATT&CK technique. The visibility scores are administered in the technique administration YAML file.

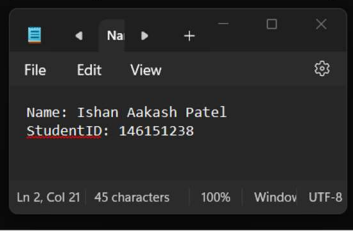
Detection : Only when you have the right data sources with adequate data quality and available to you for data analytics, your visibility can be used to create new detections for ATT&CK techniques. Detections often trigger alerts and are hence followed up on by your blue team. Scoring and administering your detections is also done in the technique administration YAML file.

Here we will take an example – First we will add few data sources.

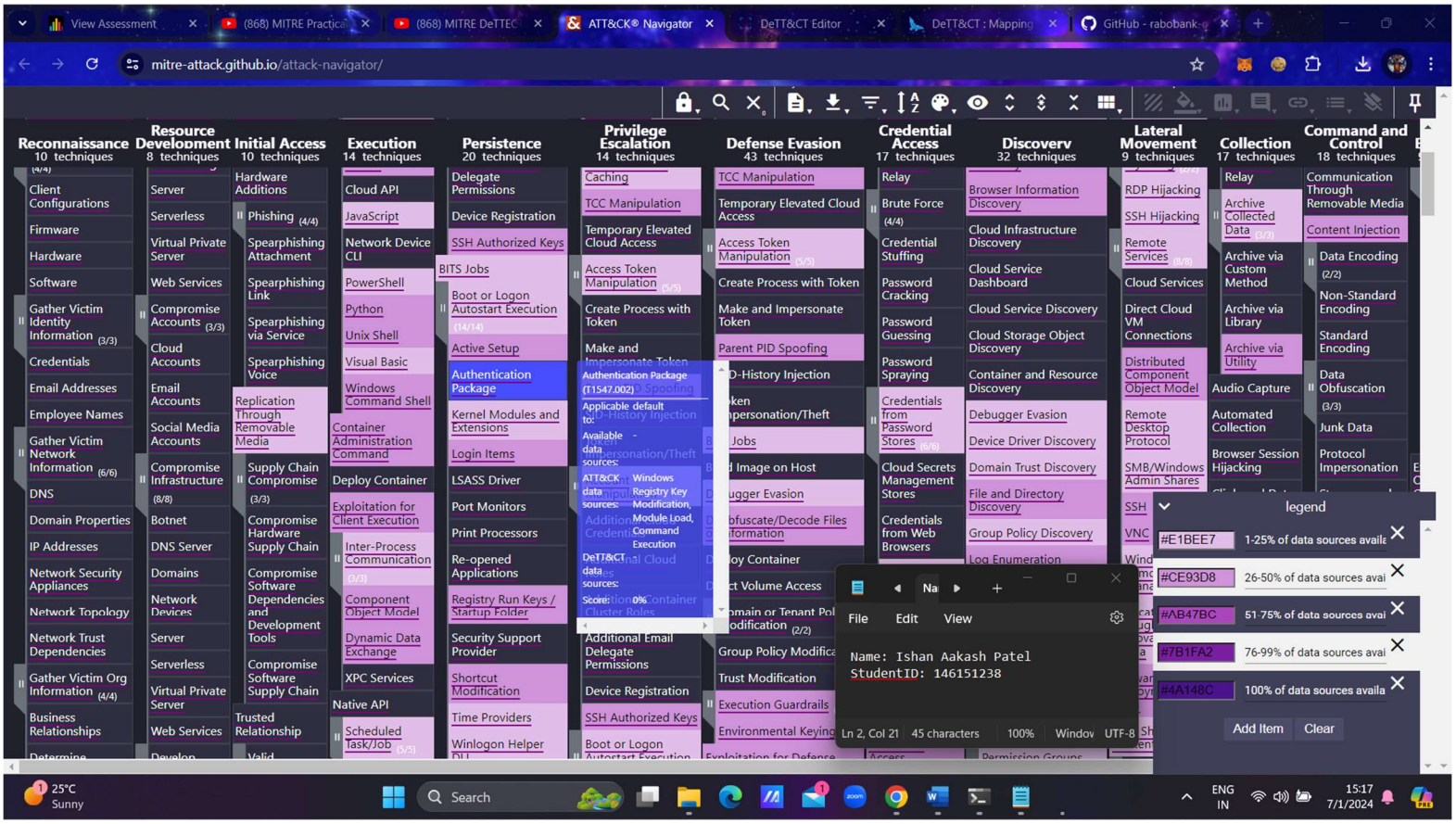


Save the YAML file and convert it to JSON file and open the JSON file in the Attack Navigator.

```
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python detect.py ds -fd /mnt/c/Users/ASUS/Downloads/data-sources-new-1.1.yaml --yaml --yaml-all-techniques --health
File written:   output/techniques-administration-example.yaml
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ cd output/
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$ ls
data_sources_example.json    data_sources_example_3.json
data_sources_example_1.json  techniques-administration-example.yaml
data_sources_example_2.json
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$ cp techniques-administration-example.yaml /mnt/c/Users/ASUS/Downloads/
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$
```



Attack Navigator Output:



Visibility Coverage

Generate the Technique Administration YAML File:

- Use the --yaml argument to generate a technique administration YAML file based on your data source administration file. This will give you visibility scores.
- By default, this command includes only techniques with a visibility score greater than 0.
- To include all ATT&CK techniques relevant to the specified platform(s), add the --yaml-all-techniques argument.

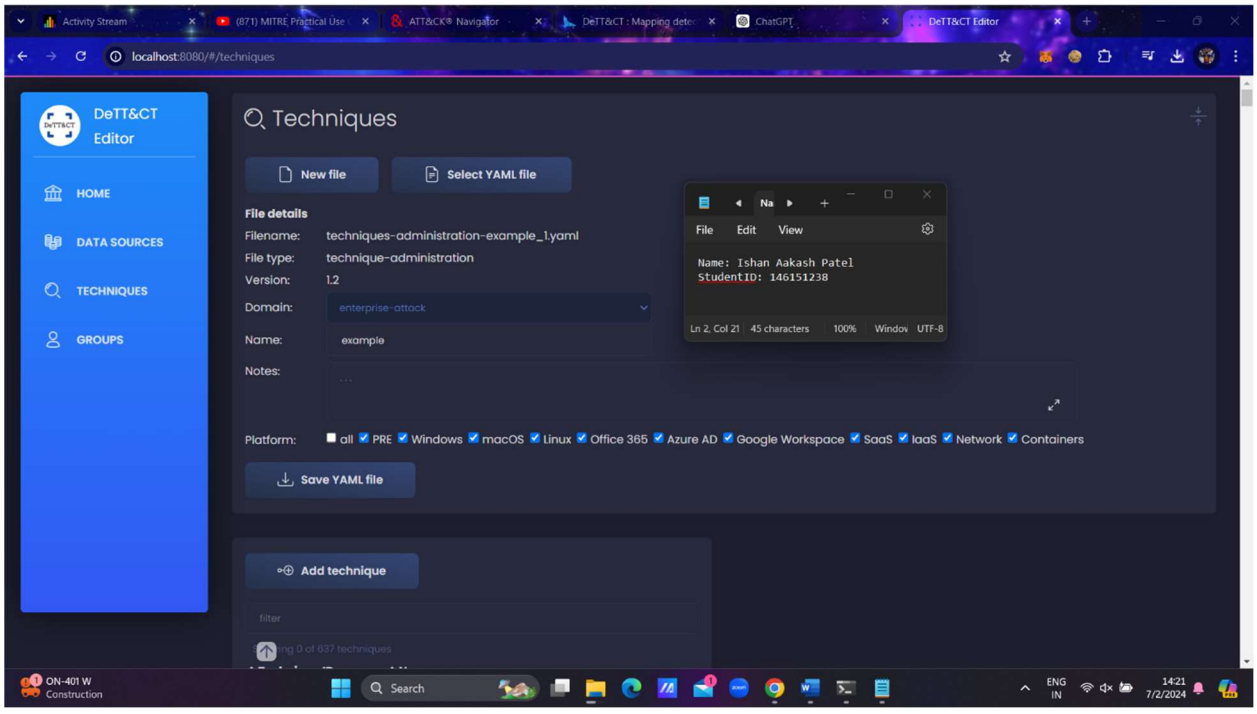
```
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py ds -fd /mnt/c/Users/ASUS/Downloads/new.yaml --yaml --yaml-all-techniques
File written:   output/techniques-administration-example_1.yaml
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ cd output/
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$ cp techniques-administration-example_1.yaml /mnt/c/Users/ASUS/Downloads/
```

Relevant Command Flags:

- ds: Select data source mode.
- -fd: Path to the data source administration YAML file.
- --yaml: Generate a technique administration YAML file with visibility scores based on the number of available data sources.
- --yaml-all-techniques: Includes all ATT&CK techniques applicable to the specified platform(s) in the generated YAML file (requires the --yaml argument).

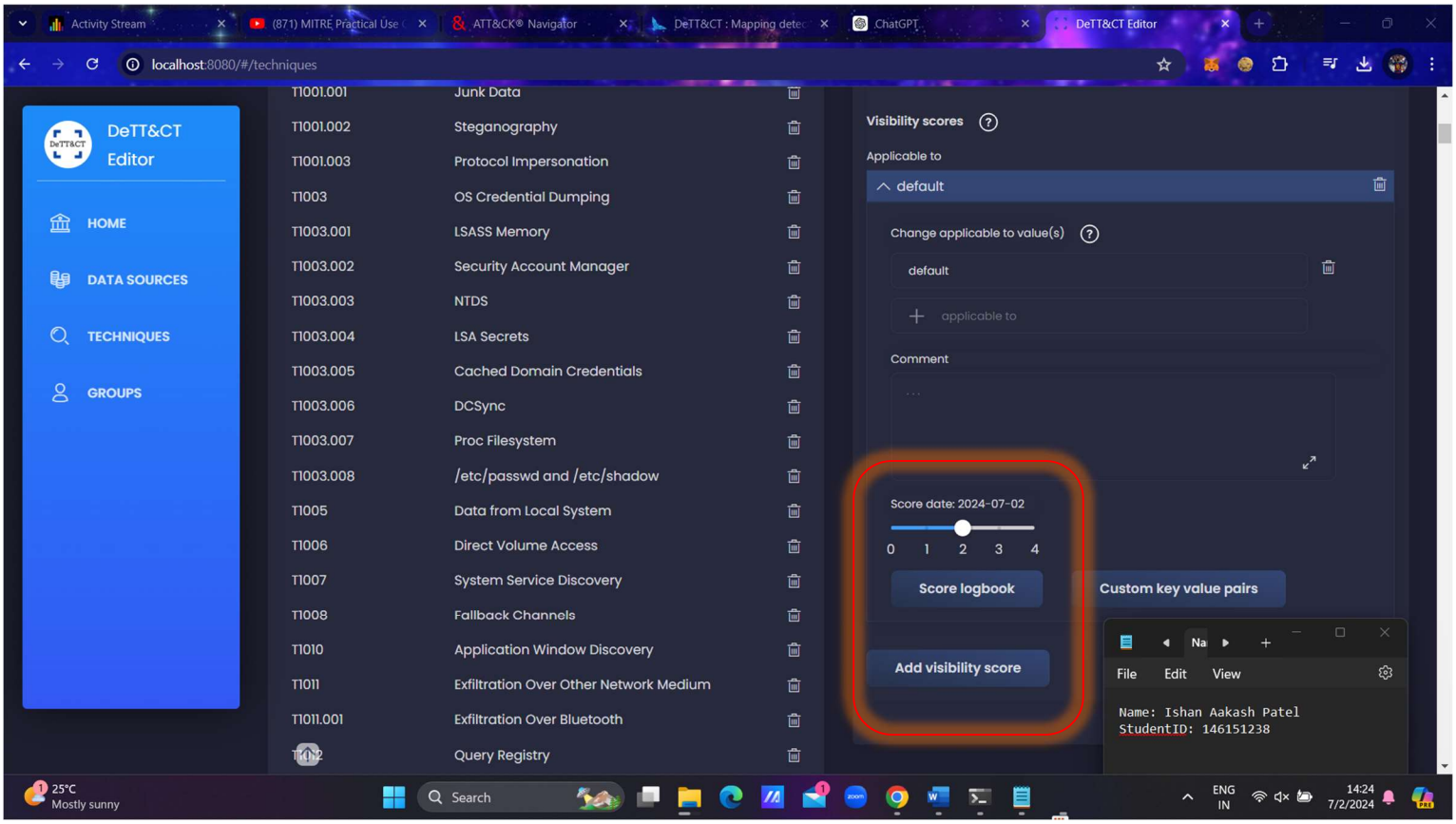
Adjust Visibility Scores:

- Within the resulting YAML file, adjust the visibility score for each technique based on expert knowledge or the quality of a particular data source.



Editing the YAML File:

- For easy editing of the technique administration YAML file, load it using DeTT&CT Editor.



To visualize the visibility scores within an ATT&CK Navigator layer, you can use the following command:

python dettect.py v -ft /mnt/c/Users/ASUS/Downloads/techniques-administration-example_1.yaml -l

This command will generate a file that can be loaded into ATT&CK Navigator to visualize the visibility scores. Here are the relevant parameters and flags for this command:

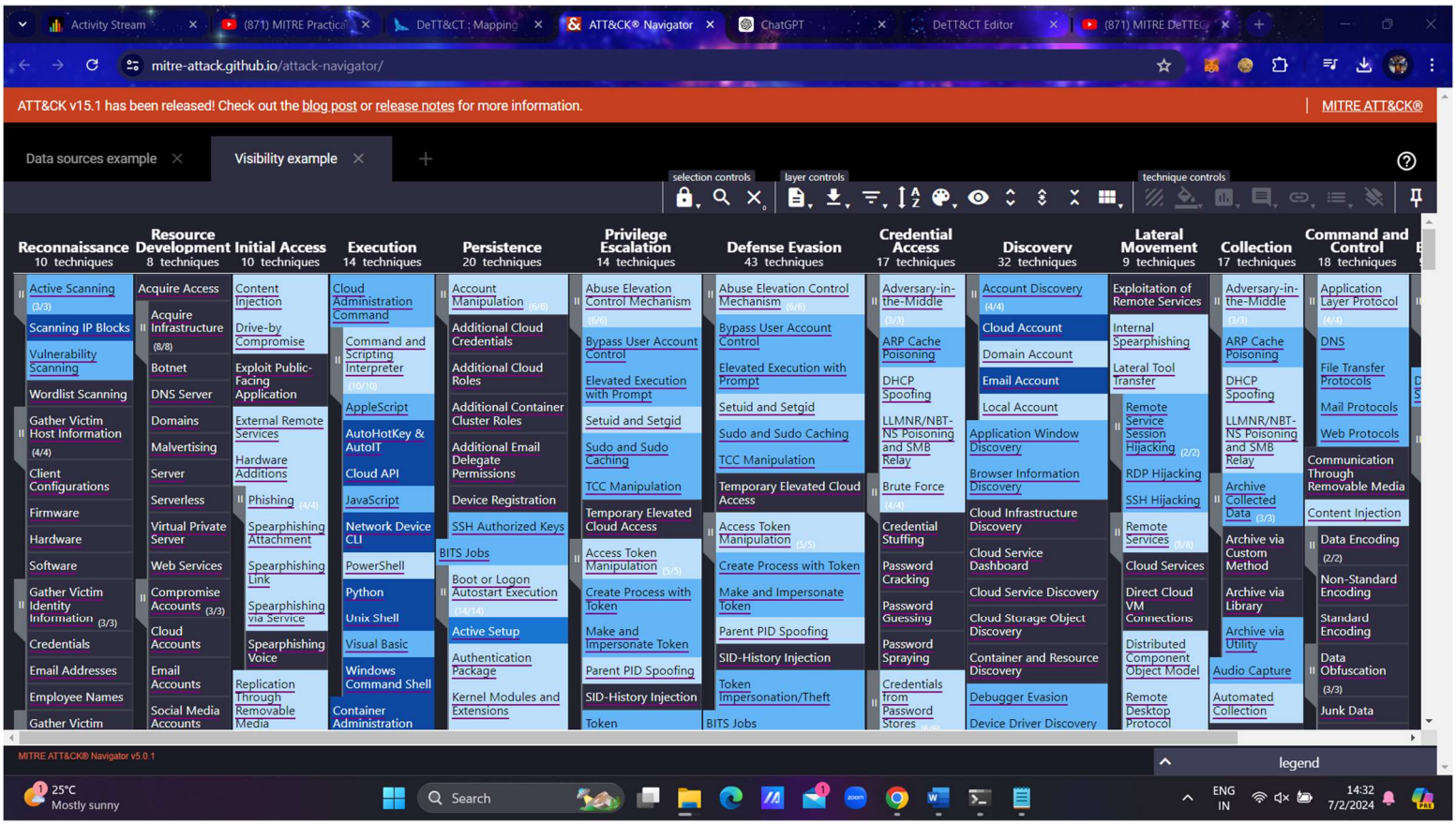
- v: Visibility coverage mapping based on techniques and data sources.
- -ft: Path to the technique administration YAML file.
- -l: Generate a data source layer for the ATT&CK Navigator.

Steps to Visualize Visibility Scores:

1. **Generate the ATT&CK Navigator Layer:**
2. **Load the File into ATT&CK Navigator:**
    - Open ATT&CK Navigator.

- Load the resulting file generated from the command into the Navigator to visualize the visibility scores.

```
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py v -ft /mnt/c/Users/ASUS/Downloads/techniques-administration-ex
ample_1.yaml -l
File written:   output/visibility_example.json
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$
```



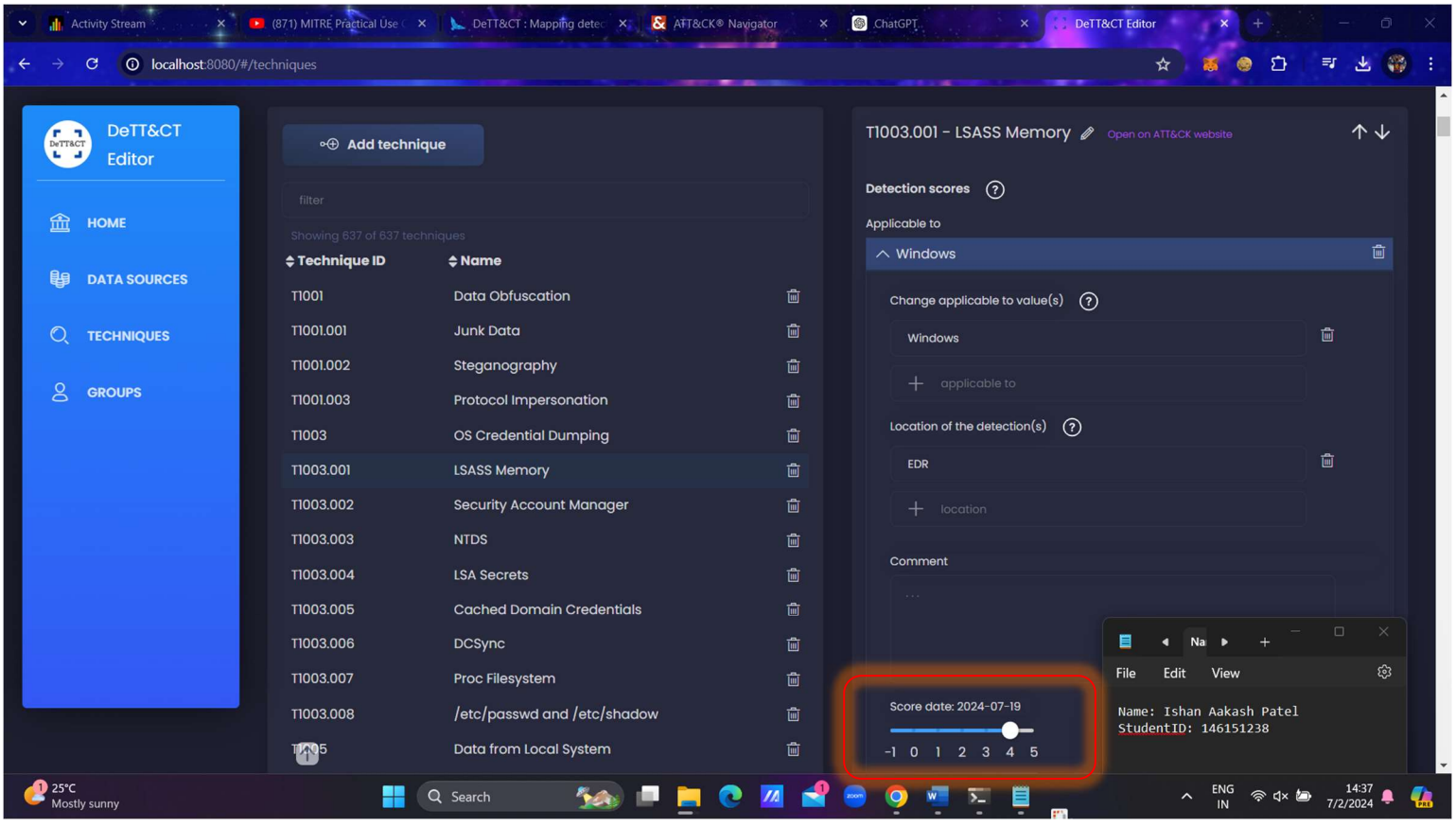
Detection Coverage

After listing our data sources and understanding our visibility, we need to assess our detection capabilities. This involves understanding where we have detection, the level of detection, and identifying any gaps.

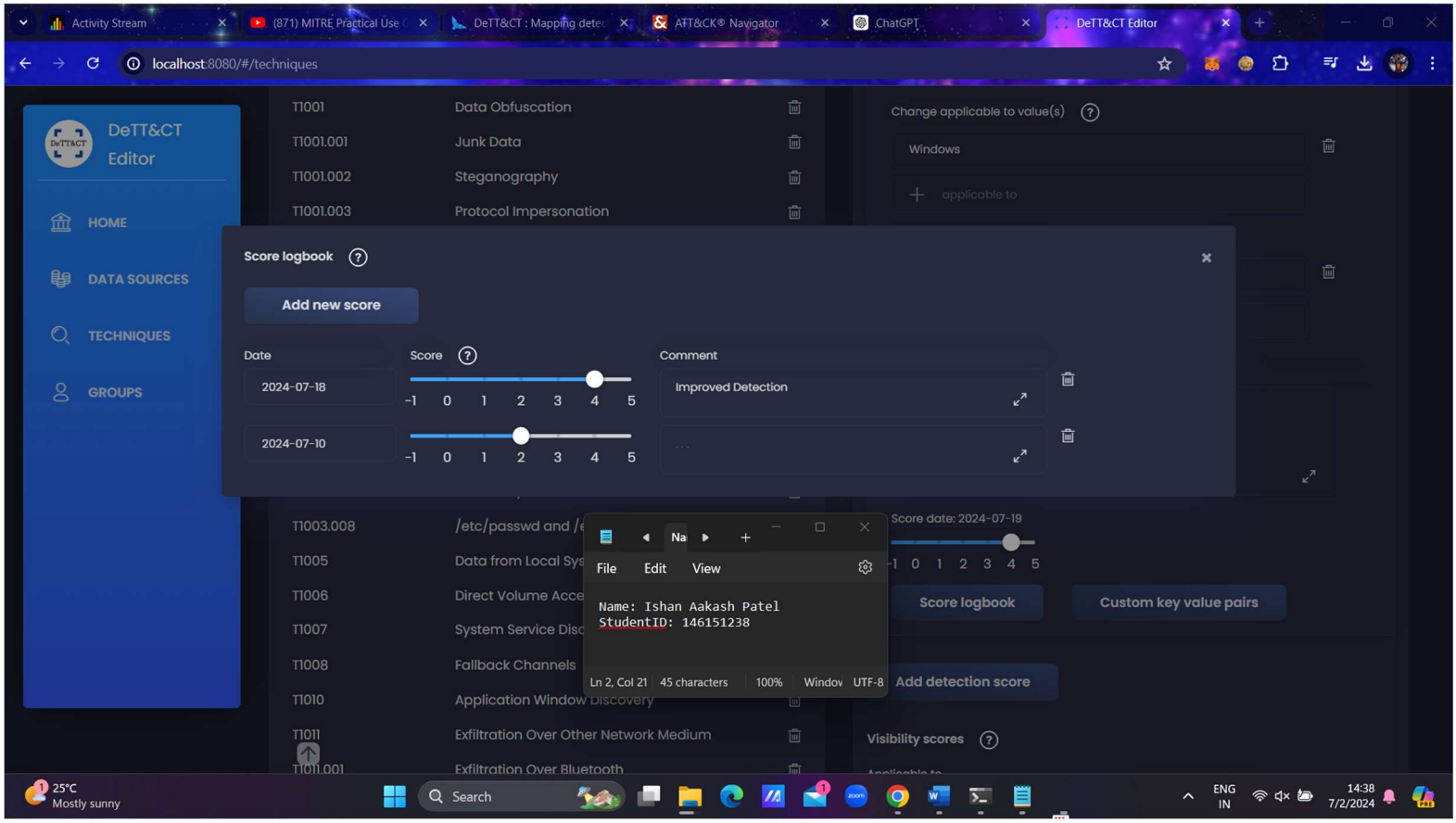
Using the same YAML data source administration file we used for visibility coverage, we can manage our detection levels by recording the following details:

1. **Type of System:**
    - Specify the type of system(s) the detection applies to, such as Windows endpoints, Windows servers, Linux servers, crown jewels, etc.
2. **Location of Detection:**
    - Indicate where the detection resides. This could be an event ID, the name of a detection rule/use case, SIEM, or a specific product name.
3. **Additional Information:**
    - Include any relevant comments.
    - Record the date when the detection was implemented or improved.
    - Assign a detection score.
    - Add further information using key-value pairs as needed.

To facilitate detailed scoring of your detections for different types of systems, you can select multiple detections per technique in the YAML file using the applicable_to property.



Set Score Logbook



To generate a layer file for the ATT&CK Navigator based on the technique administration file, you can use the following command:

python dettect.py d -ft /mnt/c/Users/ASUS/Downloads/techniques-administration-example-all.yaml -l

Here are the relevant parameters and flags for this command:

- d: Detection coverage mapping based on techniques.
- -ft: Path to the technique administration YAML file.
- -l: Generate a data source layer for the ATT&CK Navigator.

In your case, the command should look like this:

python dettect.py d -ft /mnt/c/Users/ASUS/Downloads/techniques-administration-example-all.yaml -l

Steps to Generate and Load the Layer File:

1. **Generate the Layer File:**
    - Run the command above.

- This will generate a file named detection_example.json in the output directory.

2. **Copy the File to Your Downloads Directory:**
   - Navigate to the output directory:

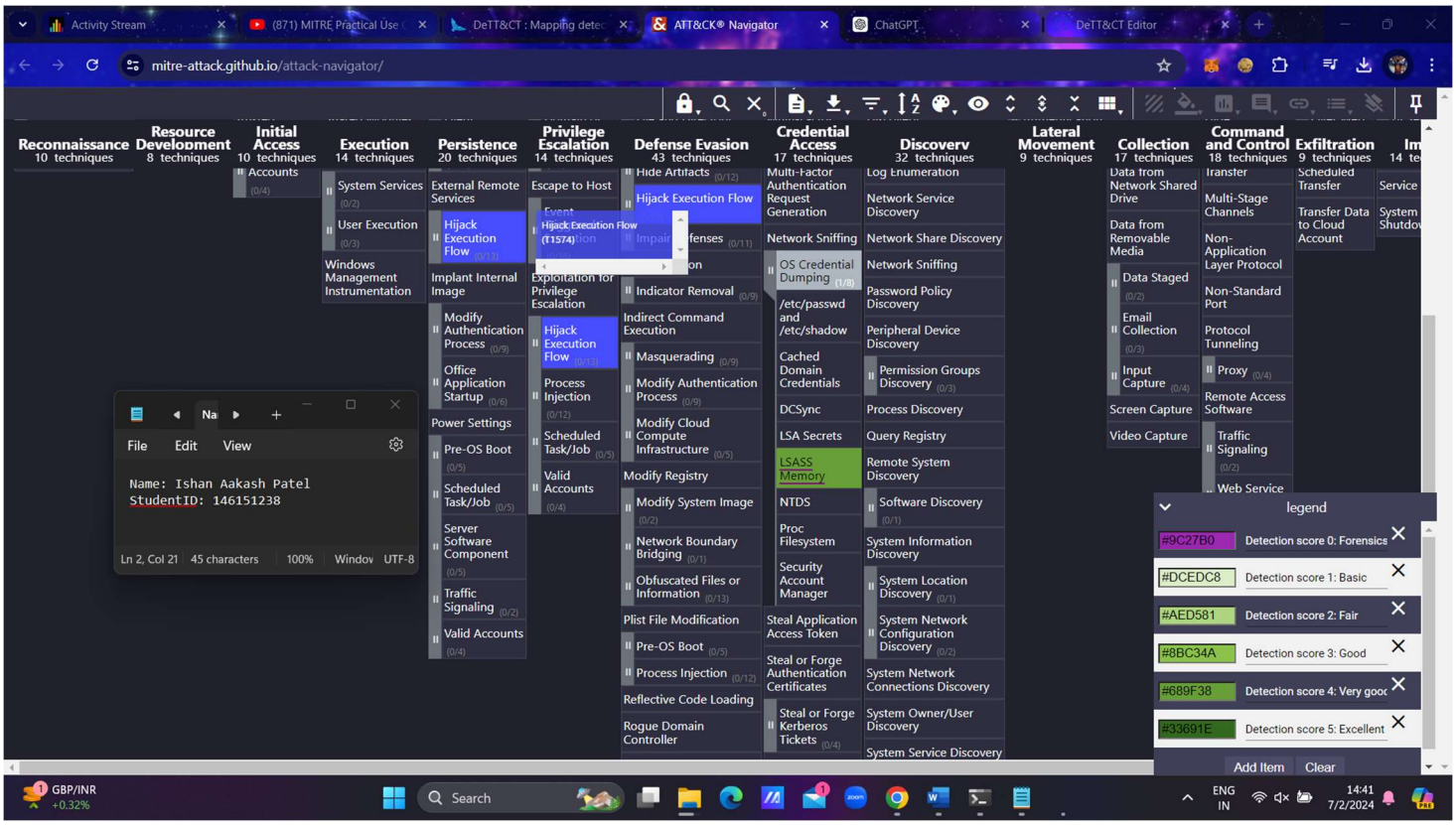
     cd output/

   - Copy the generated file to your Downloads directory:

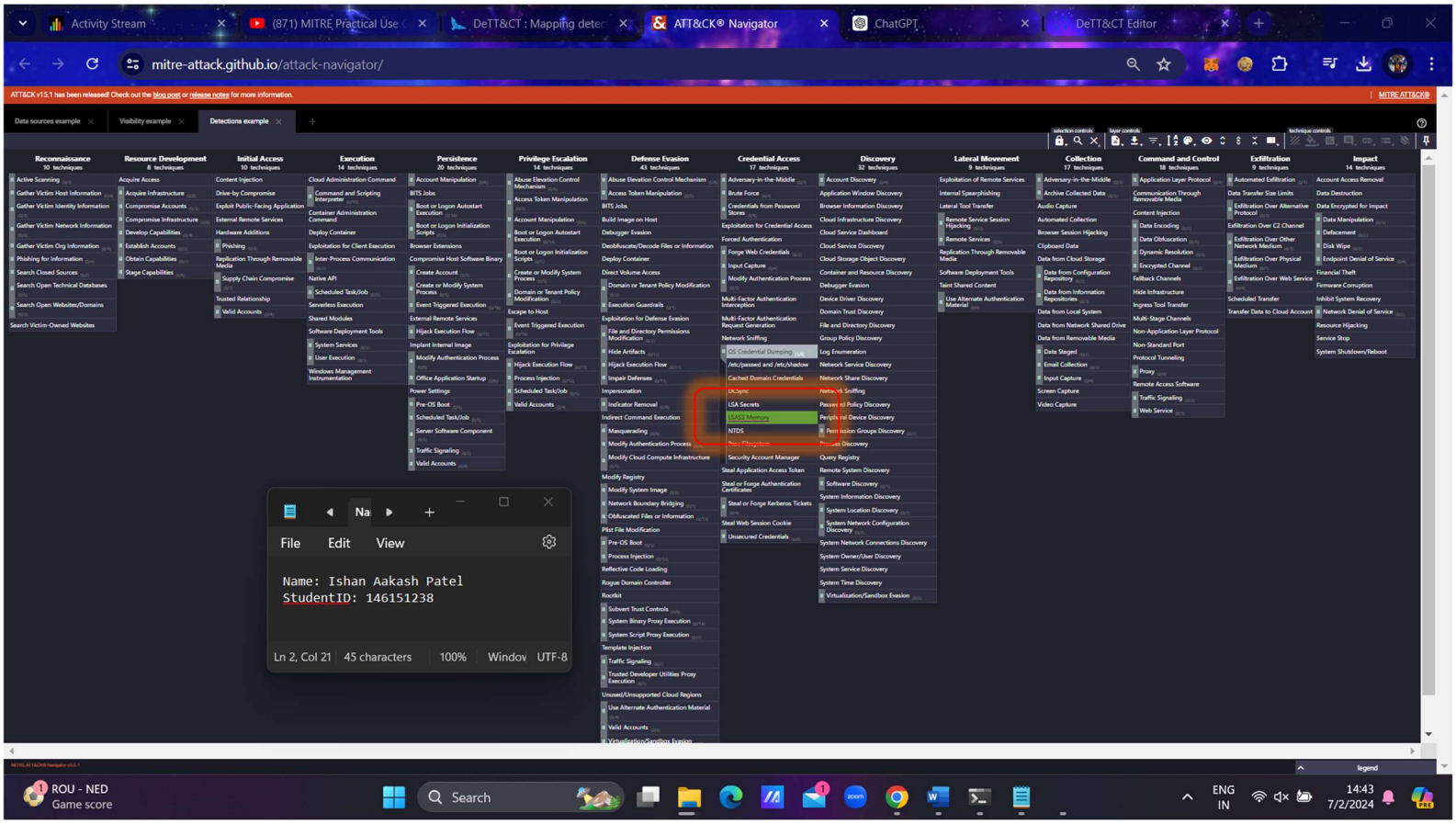
     cp detection_example.json /mnt/c/Users/ASUS/Downloads/

3. **Load the Layer File into ATT&CK Navigator:**
   - Open ATT&CK Navigator.
   - Load the detection_example.json file from your Downloads directory to visualize the detection scores.

```
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py d -ft /mnt/c/Users/ASUS/Downloads/detection.yaml -l
File written:   output/detection_example.json
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ cd output/
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$ cp detection_example.json /mnt/c/Users/ASUS/Downloads/
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$
```



Output :



Gap Analysis Against Threat Actor Group

To identify gaps in your detection capabilities, you can compare your detection layer with your threat analysis layer or a layer generated for a specific red team exercise.

Adversary Emulation:

When performing adversary emulation, the red team defines a set of techniques that mimic the behavior of a known threat to the organization. This scope is typically represented by generating an ATT&CK matrix layer.

By comparing this layer with your detection layer, you can easily spot any gaps in your detection coverage. This analysis helps ensure that your defenses are aligned with real-world threats and can effectively detect and respond to the techniques used by adversaries.

To compare a threat actor group layer with your detection or visibility coverage overlay, you can use the following command, adjusted to your paths:

python3 dettect.py g -g /mnt/c/Users/ASUS/Downloads/groups.yaml -o /mnt/c/Users/ASUS/Downloads/techniques-administration-example-all.yaml -t detection

Here are the relevant parameters and flags for this command:

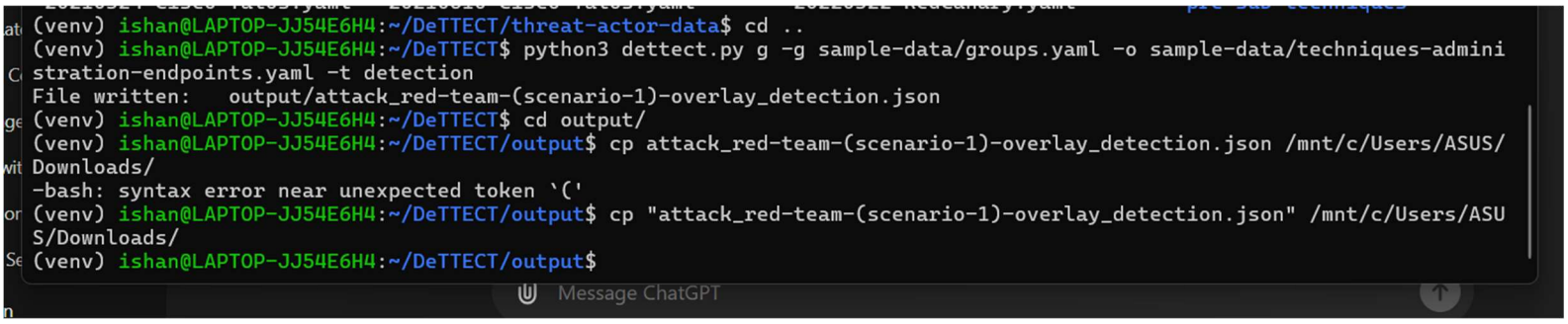
- g: Threat actor group mapping.
- -g: Specify the ATT&CK Groups to include. You can provide a YAML file with a custom group.
- -o: Specify what to overlay on the group(s). To overlay visibility or detection, provide the technique administration YAML file.
- -t {group,visibility,detection}: Specify the type of overlay. You can choose between group, visibility, or detection (default = group).

Steps to Generate and Compare Layers:

1. **Generate the Comparison Layer:**
   - Run the command above to generate the layer that highlights the differences.
2. **Copy the Generated File to Your Downloads Directory:**
   - Copy the generated file to your Downloads directory:

     cp "attack_red-team-(scenario-1)-overlay_detection.json" /mnt/c/Users/ASUS/Downloads/

3. **Load the Layer File into ATT&CK Navigator:**
   - Open ATT&CK Navigator.
   - Load the attack_red-team-(scenario-1)-overlay_detection.json file from your Downloads directory to visualize the differences.



Output:

Comparison between Visibility and Detection

To generate a layer that compares your visibility and detection coverage, you can use the following commands, adjusted to your paths. This will provide an overview of the techniques where you have visibility or detection.

Steps to Generate and Compare Layers:

1. **Generate the Comparison Layer:**
   - For detection coverage:

     python dettect.py d -ft sample-data/techniques-administration-example-all.yaml -o

   - For visibility coverage:

     python dettect.py v -ft sample-data/techniques-administration-example-endpoints.yaml -o

2. **Copy the Generated File to Your Downloads Directory:**

- Copy the generated file to your Downloads directory:

  cp visibility_and_detection_example.json /mnt/c/Users/ASUS/Downloads/

3. **Load the Layer File into ATT&CK Navigator:**
   - Open ATT&CK Navigator.
   - Load the generated JSON file from your Downloads directory to visualize the differences.

```
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py d -ft sample-data/techniques-administration-endpoints.yaml -o
File written:   output/visibility_and_detection_example.json
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ cd output/
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$ cp visibility_and_detection_example.json /mnt/c/Users/ASUS/Downloads/
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$
```

Output :

I believe step 3 is completed in step 2, and also I am attaching the json files with this pdf file.