

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=nProduct&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 DeTECT - 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

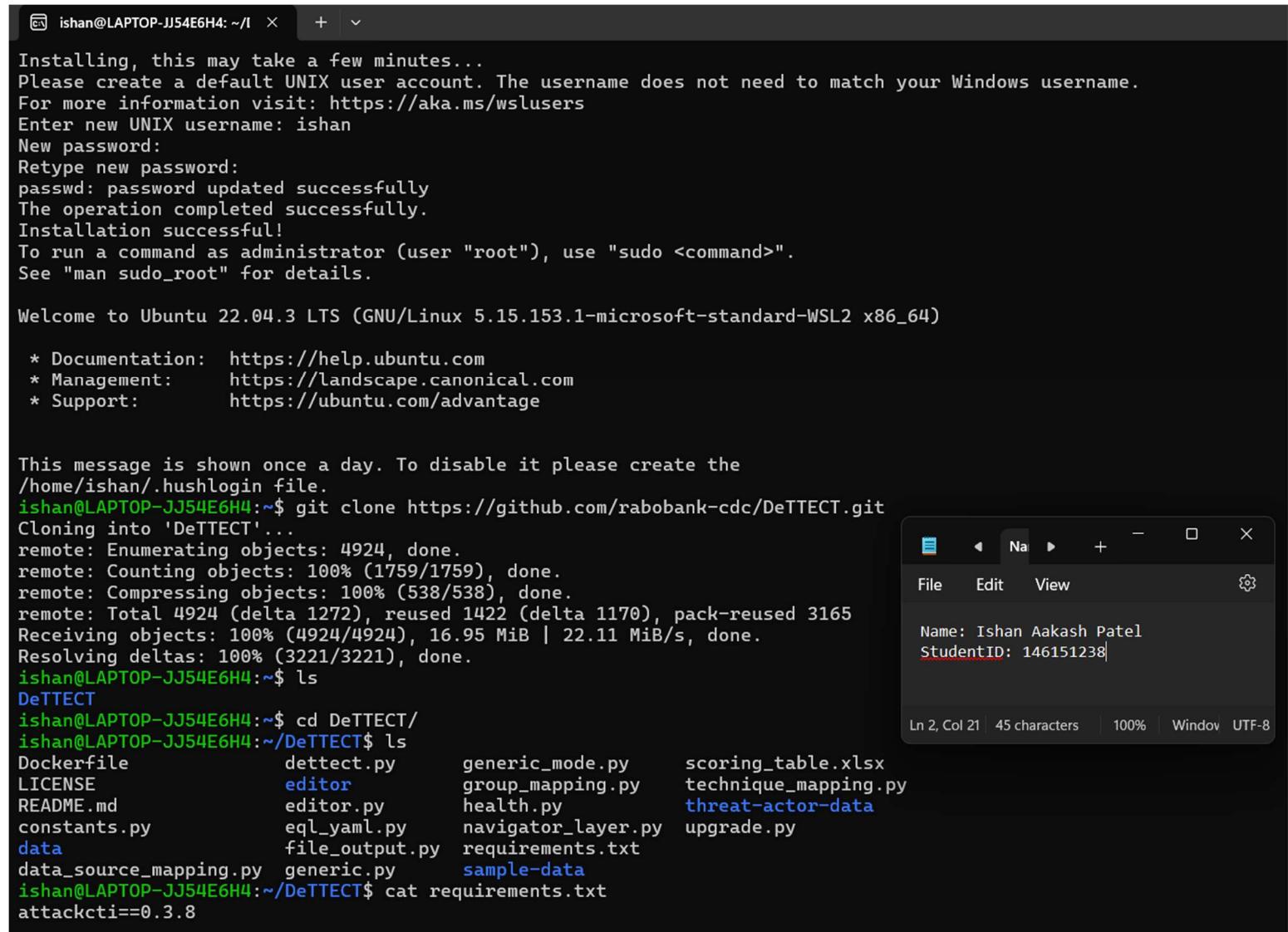
Note: submissions that do not follow the requirements will not be accepted

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.

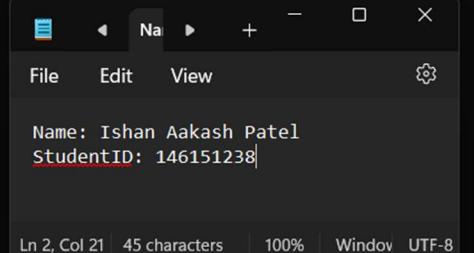


```
ishan@LAPTOP-JJ54E6H4: ~/l + 
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: ishan
New password:
Retype new password:
passwd: password updated successfully
The operation completed successfully.
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This message is shown once a day. To disable it please create the
/home/ishan/.hushlogin file.
ishan@LAPTOP-JJ54E6H4:~$ git clone https://github.com/rabobank-cdc/DeTTECT.git
Cloning into 'DeTTECT'...
remote: Enumerating objects: 4924, done.
remote: Counting objects: 100% (1759/1759), done.
remote: Compressing objects: 100% (538/538), done.
remote: Total 4924 (delta 1272), reused 1422 (delta 1170), pack-reused 3165
Receiving objects: 100% (4924/4924), 16.95 MiB | 22.11 MiB/s, done.
Resolving deltas: 100% (3221/3221), done.
ishan@LAPTOP-JJ54E6H4:~$ ls
DeTTECT
ishan@LAPTOP-JJ54E6H4:~$ cd DeTTECT/
ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ ls
Dockerfile          dettect.py      generic_mode.py    scoring_table.xlsx
LICENSE            editor         group_mapping.py  technique_mapping.py
README.md          editor.py     health.py       threat-actor-data
constants.py       eql_yaml.py   navigator_layer.py upgrade.py
data               file_output.py requirements.txt
data_source_mapping.py generic.py   sample-data
ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ cat requirements.txt
attackcti==0.3.8
```



Name: Ishan Aakash Patel
StudentID: 146151238

Ln 2, Col 21 | 45 characters | 100% | Window | UTF-8

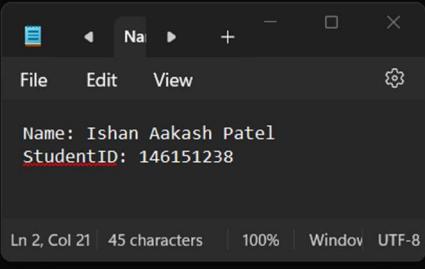
Then I installed pipenv for installing few python libraries which were present requirements.txt.

```

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)

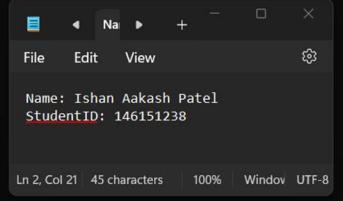
```



```

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/tmpktxg5gr' 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:~/l + v
ResourceWarning: Enable tracemalloc to get the object allocation traceback
/home/ishan/DeTECT/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:~/DeTECT$
```

Name: Ishan Aakash Patel
StudentID: 146151238

Ln 2, Col 21 45 characters | 100% | Window UTF-8

```

ishan@LAPTOP-JJ54E6H4:~/l + v
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTECT$ 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:~/DeTECT$ . /home/ishan/DeTECT/venv/bin/activate
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTECT$ python detetect.py
usage: detetect.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/DeTECT
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTECT$ python detetect.py editor
Editor started at port 8080
Opening webbrowser: http://localhost:8080/
```

Name: Ishan Aakash Patel
StudentID: 146151238

Ln 2, Col 21 45 characters | 100% | Window 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

The screenshot shows a web browser window with the URL `localhost:8080/#/home`. The main content is the **DeTT&CT Editor** homepage. On the left, there's a sidebar with navigation links: **HOME**, **DATA SOURCES**, **TECHNIQUES**, and **GROUPS**. The main area has several sections:

- Introduction**: Describes the editor for editing YAML files.
- Client-side and saving results**: Notes that the editor is client-side and files are saved locally.
- Authors and contributions**: Mentions developers [Marcus Bakker](#) and [Ruben Bouman](#).
- Keyboard shortcuts**: Lists `Ctrl+Shift+Up/Down` for navigating YAML files.
- Limitations**: Notes that comments (`#`) are not preserved in the YAML file.

A code editor window is open on the right, showing the following text:

```
Name: Ishan Aakash Patel
StudentID: 146151238
```

At the bottom, there's a footer with copyright information and system status icons.

Add Data sources, Techniques, and Groups.

The screenshot shows the DeTT&CT Editor interface on a Windows desktop. The left sidebar has icons for HOME, DATA SOURCES, TECHNIQUES, and GROUPS. The main area is titled "Data Sources". It shows a file named "data-sources-new.yaml" with a "data-source-administration" type and version 1.1. The "Domain" is set to "enterprise-attack", and the "Name" is "example". A note field contains "...". Below these fields are "Systems" (set to "default") and "Platforms" (set to "all"). A modal window is open, displaying a text editor with the following content:

```
Name: Ishan Aakash Patel
StudentID: 146151238
```

The status bar at the bottom shows the date as 7/1/2024 and the time as 13:50. The taskbar includes icons for File Explorer, Edge, and various system tools.

This screenshot shows the same DeTT&CT Editor interface after adding a new data source. The main area now displays the "Add data source" and "Add all data sources" buttons. A data source named "Windows Registry Key" is listed under "Applicable to" with "default" selected. A modal window shows the same text editor content as before. The right side of the screen shows the "Windows Registry Key Access" configuration panel, which includes fields for "Date registered" (2024-07-20), "Date connected" (2024-07-20), "Data source enabled" (No), and "Available for data analytics" (No). The "Products" section lists "windows". The taskbar and system status are identical to the previous screenshot.

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

```
ishan@LAPTOP-JJ54E6H4:~/l + ^

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/
^CShutting down webserver
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py ds -fd /mnt/c/Users/ASUS/Downloads/data-sources-new.yaml -l
File written: output/data_sources_example.json
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py ds -fd /mnt/c/Users/ASUS/Downloads/data-sources-new.yaml -l --health
File written: output/data_sources_example_1.json
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$
```

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

The screenshot shows the MITRE ATT&CK Navigator web application. A context menu is open in the top right corner, listing options for creating new layers:

- Create New Layer: Create a new empty layer
- Open Existing Layer: Load a layer from your computer or a URL
- Create Layer from Other Layers: Select layers to inherit properties from
- Create Customized Navigator: Create a hyperlink to a customized ATT&CK Navigator

The main interface displays the ATT&CK Navigator logo and a brief description of its purpose. The status bar at the bottom indicates "MITRE ATT&CK® Navigator v5.0.1".

The screenshot shows a detailed view of the ATT&CK matrix in the MITRE ATT&CK Navigator. The matrix is organized into categories and sub-categories, with specific techniques listed under each. The categories include:

- Reconnaissance**: 10 techniques
- Resource Acquisition**: 8 techniques
- Initial Access**: 10 techniques
- Execution**: 14 techniques
- Persistence**: 20 techniques
- Privilege Escalation**: 14 techniques
- Defense Evasion**: 43 techniques
- Credential Access**: 17 techniques
- Discovery**: 32 techniques
- Lateral Movement**: 9 techniques
- Collection**: 17 techniques
- Command and Control**: 18 techniques

Each technique is represented by a small card with a title and a count of sub-techniques. The interface includes various controls and filters at the top and bottom.

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.

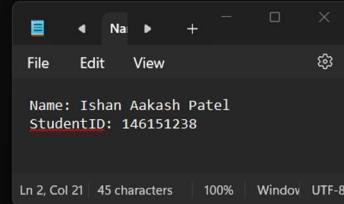
Here we will take an example – First we will add few data sources.

The screenshot shows the DeTT&CT Editor interface. On the left sidebar, there are four main navigation items: HOME, DATA SOURCES, TECHNIQUES, and GROUPS. The DATA SOURCES item is selected. The main content area displays a list of data sources. At the top of this list is a header row with columns for 'Name' and 'Applicable to'. Below this header, there are four data source entries, each with a small trash icon to its right. A red box highlights this entire list. To the right of the list, there is a modal window titled 'Windows Registry Key Creation' with a 'Save YAML file' button at the top. The modal contains fields for 'Name' (Ishan Aakash Patel) and 'StudentID' (146151238). The bottom right corner of the screen shows a status bar with various icons and text, including the date and time (7/1/2024, 15:06).

| Name | Applicable to |
|-------------------------------|---------------|
| Command Execution | all |
| Network Traffic Flow | all |
| Process Creation | all |
| Windows Registry Key Creation | all |

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 dettect.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:

| Technique Category | Technique Sub-Categories | Scoring Breakdown |
|--------------------|---|---|
| Persistence | Cloud API, JavaScript, Network Device CLI, PowerShell, Python, Unix Shell, Visual Basic, Windows Command Shell, BITS Jobs | #E1BEE7 (1-25%), #CE93D8 (26-50%), #AB47BC (51-75%), #7B1FA2 (76-99%), #4A148C (100%) |

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.

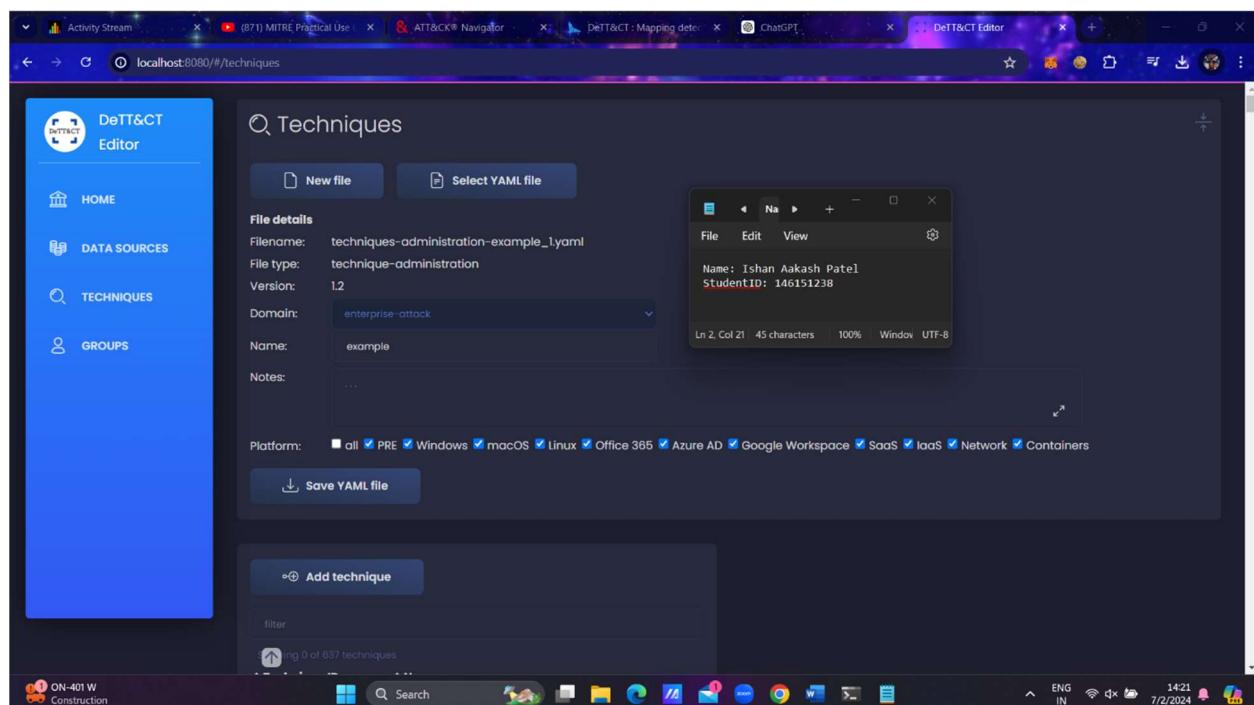
```
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python dettect.py ds -fd /mnt/c/Users/ASUS/Downloads/new.yaml --yaml --yaml-all-techniques
File written: output/techniques-administration-example_1.yaml
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ cd output/
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$ cp techniques-administration-example_1.yaml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

The screenshot shows the DeTT&CT Editor interface. On the left, there's a sidebar with icons for HOME, DATA SOURCES, TECHNIQUES, and GROUPS. The main area displays a list of ATT&CK techniques under the heading 'Techniques'. The list includes entries like T1001.001 (Junk Data), T1001.002 (Steganography), and T1001.003 (Protocol Impersonation). To the right of the list is a 'Visibility scores' panel. This panel has a 'default' section where a score of 2 is set for the date 2024-07-02. There are buttons for 'Score logbook' and 'Add visibility score'. A red box highlights this section. Below the main interface is a taskbar with various application icons and system status indicators.

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:**
 - o 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-example_1.yaml -l
File written:    output/visibility_example.json
(venv) ishan@LAPTOP-JJ54E6H4:~/DeTTECT$
```

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

ATT&CK v15.1 has been released! Check out the [blog post](#) or [release notes](#) for more information.

MITRE ATT&CK®

Data sources example Visibility example +

selection controls layer controls technique controls

| Reconnaissance 10 techniques | Resource Development 8 techniques | Initial Access 10 techniques | Execution 14 techniques | Persistence 20 techniques | Privilege Escalation 14 techniques | Defense Evasion 43 techniques | Credential Access 17 techniques | Discovery 32 techniques | Lateral Movement 9 techniques | Collection 17 techniques | Command and Control 18 techniques |
|--|--------------------------------------|-------------------------------------|---|------------------------------------|---|---|--------------------------------------|--|----------------------------------|--------------------------------------|---------------------------------------|
| Active Scanning (8/8) | Acquire Access | Content Injection | Cloud Administration Command | Account Manipulation (6/6) | Abuse Elevation Control Mechanism (6/6) | Abuse Elevation Control Mechanism (6/6) | Adversary-in-the-Middle (8/9) | Account Discovery (4/4) | Exploitation of Remote Services | Adversary-in-the-Middle (8/9) | Application Layer Protocol (4/4) |
| Scanning IP Blocks | Acquire Infrastructure (8/8) | Drive-by Compromise | Command and Scripting Interpreter (9/9) | Additional Cloud Credentials | Bypass User Account Control | Bypass User Account Control | ARP Cache Poisoning | Cloud Account | Internal Spearphishing | ARP Cache Poisoning | DNS |
| Vulnerability Scanning | Botnet | Exploit Public-Facing Application | Additional Cloud Roles | Additional Container Cluster Roles | Elevated Execution with Prompt | Elevated Execution with Prompt | DHCP Spoofing | Domain Account | Lateral Tool Transfer | DHCP Spoofing | File Transfer protocols |
| Wordlist Scanning | DNS Server | External Remote Services | Additional Email Delegate Permissions | Additional Container Cluster Roles | Setuid and Setgid | Setuid and Setgid | LLMNR/NBT-NS Poisoning and SMB Relay | Email Account | Local Account | LLMNR/NBT-NS Poisoning and SMB Relay | Mail Protocols |
| Gather Victim Host Information (4/4) | Malvertising | Hardware Additions | Device Registration | SSH Authorized Keys | Sudo and Sudo Caching | Sudo and Sudo Caching | Application Window Discovery | Remote Service Session Hijacking (2/2) | RDP Hijacking | SSH Hijacking | Web Protocols |
| Client Configurations | Server | Phishing (4/4) | Network Device CLI | BITS Jobs | TCC Manipulation | TCC Manipulation | Brute Force (4/4) | Cloud Infrastructure Discovery | Cloud Infrastructure Discovery | Archive Collected Data (8/8) | Communication Through Removable Media |
| Firmware | Serverless | Spearphishing Attachment | PowerShell | Temporary Elevated Cloud Access | Temporary Elevated Cloud Access | Access Token Manipulation (3/3) | Credential Stuffing | Cloud Service Dashboard | Cloud Service Discovery | Archive via Custom Method | Content Injection |
| Hardware | Virtual Private Server | Spearphishing Link | Python | Access Token Manipulation (3/3) | Create Process with Token | Create Process with Token | Password Cracking | Cloud Storage Object Discovery | Cloud Storage Object Discovery | Archive via Library | Data Encoding (2/2) |
| Software | Web Services | Spearphishing via Service | Unix Shell | Autostart Execution (4/4) | Make and Impersonate Token | Make and Impersonate Token | Password Guessing | Container and Resource Discovery | Container and Resource Discovery | Archive via Utility | Non-Standard Encoding |
| Gather Victim Identity Information (3/3) | Compromise Accounts (3/3) | Spearphishing Voice | Visual Basic | Active Setup | Parent PID Spoofing | Parent PID Spoofing | Password Spraying | Debugger Evasion | Debugger Evasion | Automated Collection | Data Obfuscation (3/3) |
| Credentials | Cloud Accounts | Replication Through Removable Media | Windows Command Shell | Authentication Package | SID-History Injection | SID-History Injection | Token Impersonation/Theft | Device Driver Discovery | Device Driver Discovery | Junk Data | |
| Email Addresses | Email Accounts | Container Administration | Kernel Modules and Extensions | Token | BITS Jobs | | Credentials from Password Stores | | | | |
| Employee Names | Social Media Accounts | | | | | | | | | | |
| Gather Victim | | | | | | | | | | | |

MITRE ATT&CK® Navigator v5.0.1

25°C Mostly sunny

Search           

ENG IN 14:32 7/2/2024

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:**
 - o 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:**
 - o 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:**
 - o Include any relevant comments.
 - o Record the date when the detection was implemented or improved.
 - o Assign a detection score.
 - o 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.

The screenshot shows the DeTT&CT Editor application running in a browser. On the left, there's a sidebar with navigation links: HOME, DATA SOURCES, TECHNIQUES (which is currently selected), and GROUPS. The main area displays a list of techniques under the heading "Techniques". A modal window is open for the technique "T1003.001 - LSASS Memory". The modal includes fields for "Detection scores" (a slider set to 4.5 with a note "Score date: 2024-07-19"), "Applicable to" (set to "Windows"), "Location of the detection(s)" (set to "EDR"), and a "Comment" section. A small red box highlights the "Score date" field. In the bottom right corner of the modal, there's a small window showing student information: "Name: Ishan Aakash Patel" and "StudentID: 146151238". The status bar at the bottom of the screen shows the date and time as "7/2/2024 14:37".

Set Score Logbook

The screenshot shows the DeTT&CT Editor interface. On the left sidebar, there are links for Activity Stream, (871) MITRE Practical Use Cases, DeTT&CT: Mapping detection, ATT&CK® Navigator, ChatGPT, and DeTT&CT Editor. The main area displays a list of techniques under 'TECHNIQUES': T1001 (Data Obfuscation), T1001.001 (Junk Data), T1001.002 (Steganography), and T1001.003 (Protocol Impersonation). A 'Score logbook' modal is open, showing two entries: one for 2024-07-18 with a score of 4 and a comment 'Improved Detection', and another for 2024-07-10 with a score of 2 and a comment '...'. Below the modal, a context menu is open over technique T1003.008, showing options like 'Score logbook' and 'Custom key value pairs'. The bottom status bar shows weather (25°C, Mostly sunny), search, and system icons.

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/
```

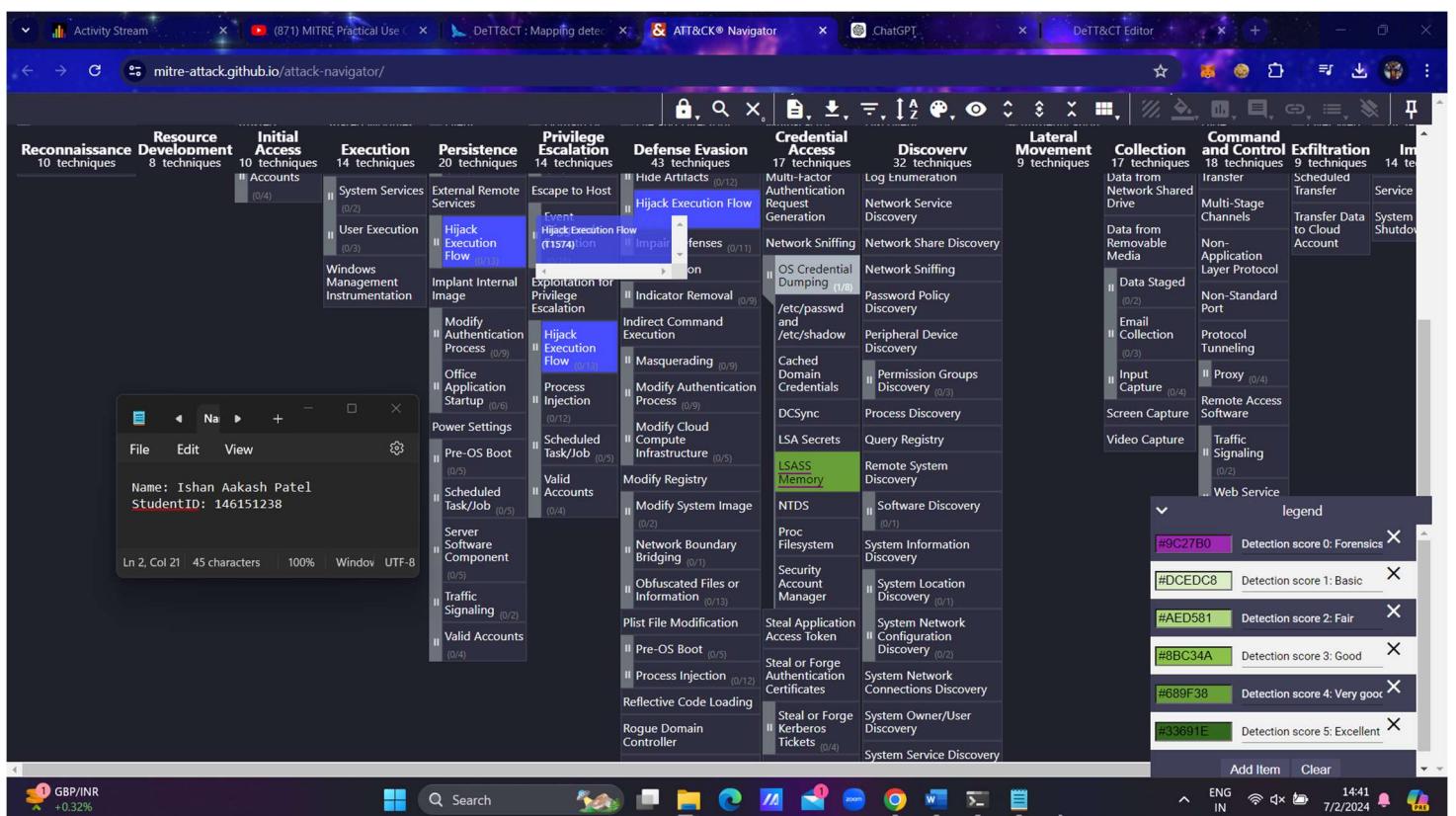
- Copy the generated file to your Downloads directory:

```
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 :

The screenshot shows the ATT&CK Navigator interface with a search result for 'LSASS Memory'. The search results are displayed in a grid format under the 'Discovery' category. The 'LSASS Memory' technique is highlighted with a red box. Below the grid, there is a preview window showing a text file with the following content:

```
Name: Ishan Aakash Patel
StudentID: 146151238
```

At the bottom of the interface, there is a taskbar with various icons and a system status bar indicating 'ENG IN' and the date '7/2/2024'.

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 detect.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.

```
ishan@LAPTOP-JJ54E6H4:~/DeTTECT/threat-actor-data$ cd ..
ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ python3 dettect.py g -g sample-data/groups.yaml -o sample-data/techniques-administrative-endpoints.yaml -t detection
File written: output/attack_red-team-(scenario-1)-overlay_detection.json
ishan@LAPTOP-JJ54E6H4:~/DeTTECT$ cd output/
ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$ cp attack_red-team-(scenario-1)-overlay_detection.json /mnt/c/Users/ASUS/Downloads/
-bash: syntax error near unexpected token `('
ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$ cp "attack_red-team-(scenario-1)-overlay_detection.json" /mnt/c/Users/ASUS/Downloads/
ishan@LAPTOP-JJ54E6H4:~/DeTTECT/output$
```

Output:

The screenshot shows a complex web-based interface for security analysis. At the top, there are multiple tabs including 'Activity Stream', 'mitre-attack.github.io/attack-navigator/' (the active tab), 'DeTT&CT : Mapping detected...', 'ATT&CK® Navigator', 'ChatGPT', and 'DeTT&CT Editor'. The main content area is a large grid of colored boxes representing different MITRE ATT&CK techniques. The grid is organized into columns for Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, and Impact. Each column contains several sub-categories and specific techniques. A context menu is open over the 'Application Access Token' technique in the 'Collection' section, listing options like 'Exploit Application', 'File Transfer Protocols', 'Mail Protocols', 'Archive via Library', 'Archive via Utility', 'Junk Data', 'Steganography', and 'Scheduled Transfer'. Below the grid, a terminal window displays the user's profile: Name: Ishan Aakash Patel and StudentID: 146151238. The bottom of the screen shows a system status bar with weather information (24°C, mostly sunny), a search bar, and various system icons.

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:

- o For detection coverage:

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

- 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.