



CYT-250-Threat Investigation

Lab 7-Seeing VERIS in Action

Elaborate by:

Leandro Delgado

Student Number: 114416241



Professor: Tatiana Outkina

CYT250 Lab 7. Seeing VERIS in Action – 4 %

Individual work

Objective: learn the VERIS tools recommended to (1) support the VERIS Community Database (VCDB), and (2) use VCDB for your own investigation of security incidents.

Technical skills:

- Create incident report via VERIS WebApp
- Retrieve data from VERIS VCDB

Pre-requisites

Work with security incidents assume learning of vocabulary which set the basis of information sharing. We also must think about incident reports repository and tools which can be used in practice.



Figure 1.0 Screen

Task 1. Learn how to complete security incidents reports with the help of VERIS WebApp

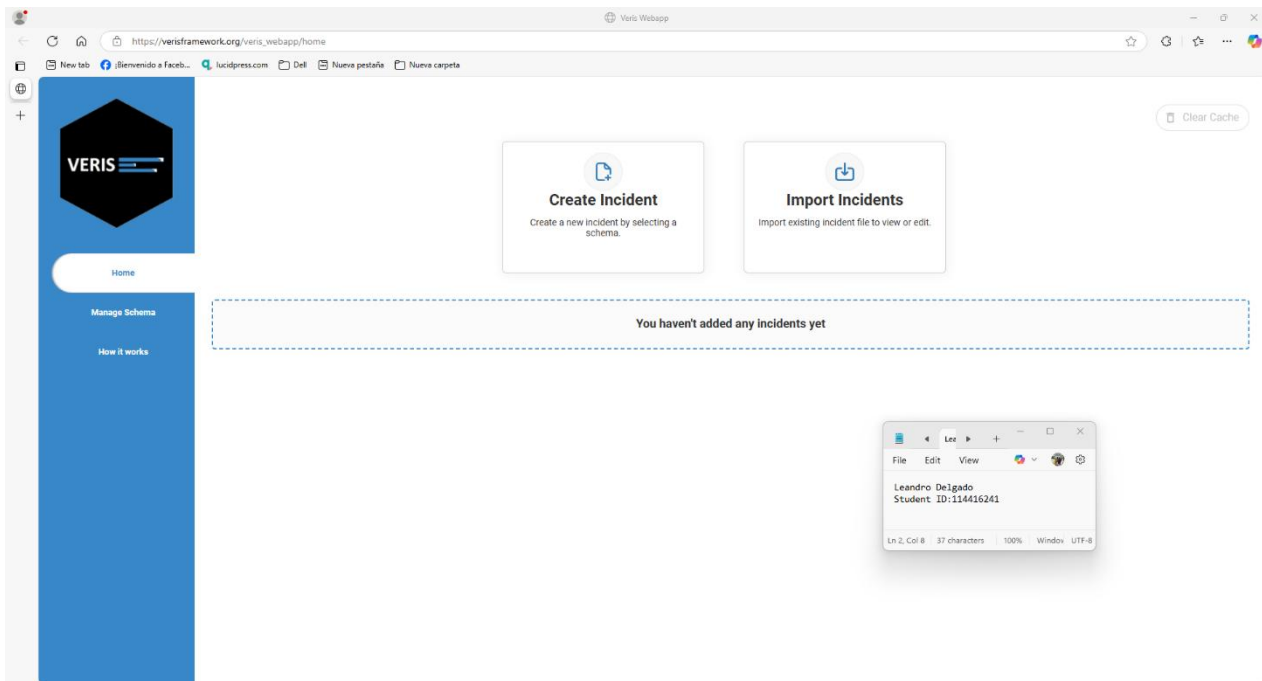


Figure 2. VERIS Web App Home Interface.

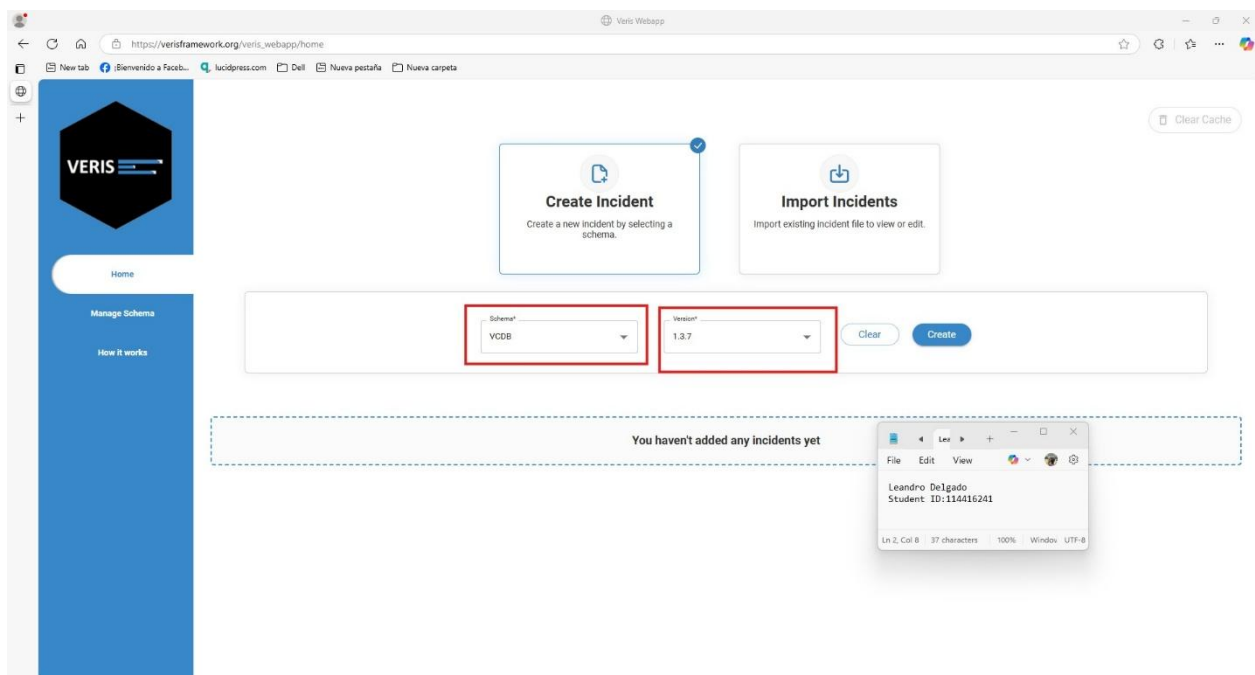


Figure 3. Version selection

I began working in the VERIS WebApp by accessing the main dashboard, where I had the option to create or import incidents. To ensure compatibility with the VERIS

Community Database, I selected the “VCDB” schema and version 1.3.7, setting the foundation to accurately structure and document security incident data.

Figure 4. Dashboard Veris app to prepare the incident documentation.

This screenshot shows the beginning of the incident documentation process using the VERIS WebApp. Here, I accessed the “Incident info” tab, where I began filling out key metadata such as the summary, data source, confidence level, and analyst details. This foundational step ensures that the incident is clearly contextualized and properly structured for further categorization across the VERIS schema.

Figure 5. Incident documentation in the VERIS WebApp.

This screenshot documents the original HIPAA Journal article that reports on a cyber breach experienced by the Hand & Plastic Surgery Centre in Michigan, affecting about twenty thousand patients. The article describes the nature of the breach, how access was gained to the systems, the involvement of forensics experts, and what data may have been accessed. This source serves as the basis of the incident documentation in the VERIS WebApp.

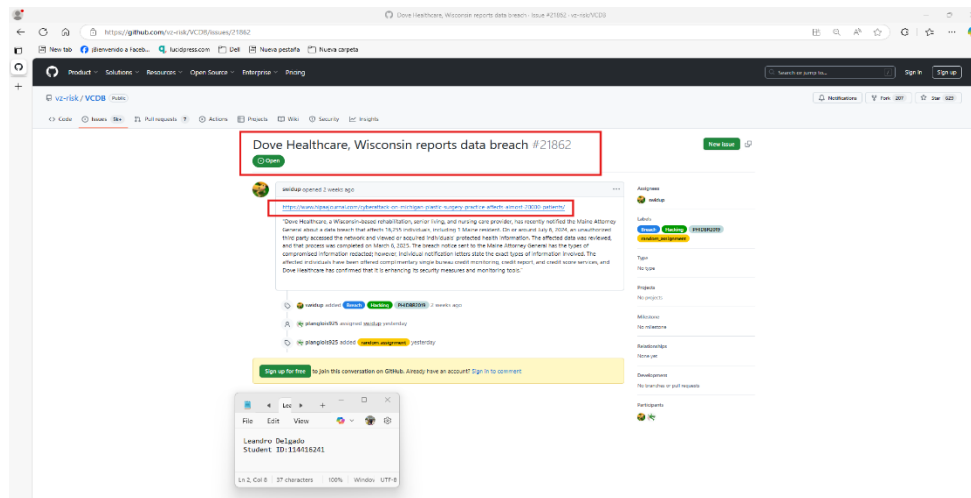


Figure 6. GitHub issue entry

This screenshot shows the official GitHub issue entry (#21862) for the data breach reported by Dove Healthcare in Wisconsin. The incident was added to the VCDB (VERIS Community Database) and references the HIPAA Journal article as its source. This validates the event and ensures its traceability for cybersecurity research and analysis.

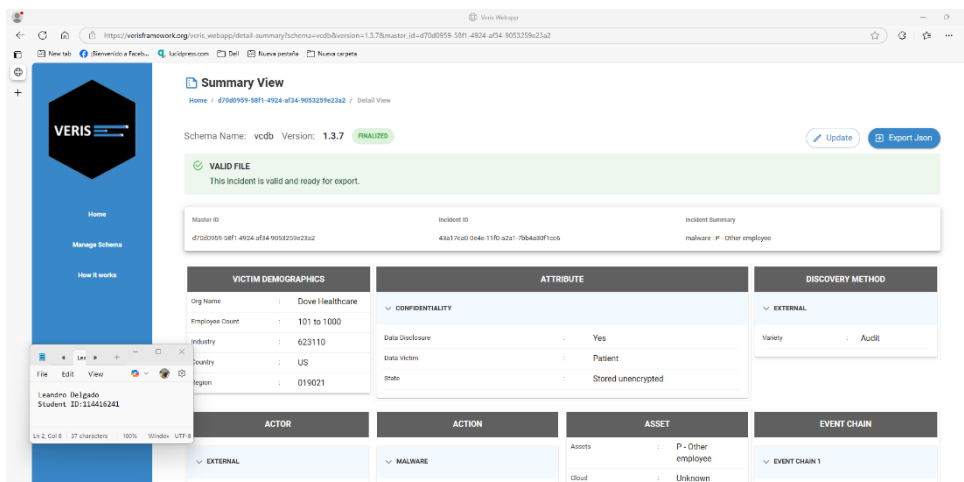


Figure 7. Final summary view confirms the incident has been successfully encoded

This final summary report confirms the incident has been successfully encoded using the VERIS schema (version 1.3.7) and is marked as valid for export. It documents key details such as the organization (Dove Healthcare), data exposure (patient data), discovery method (external audit), and classification of actor, action, asset, and event chain, making it ready for sharing or inclusion in the VCDB.

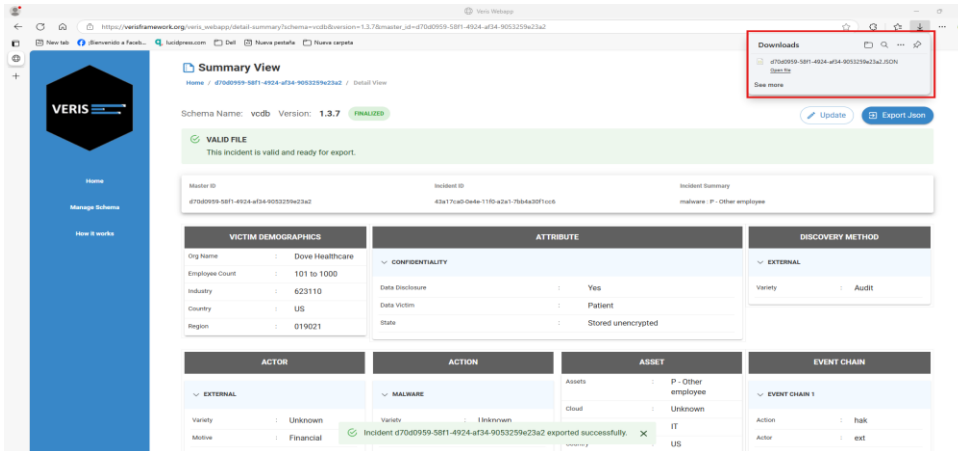


Figure 8. The JSON file has been downloaded

This screenshot captures the successful completion of the VERIS incident creation process. The green "VALID FILE" message confirms that the data entered meets the required standards, and the JSON file has been exported without issues. The download notification in the top-right corner indicates the file is now safely stored and ready for further analysis or submission, marking a smooth and complete workflow.

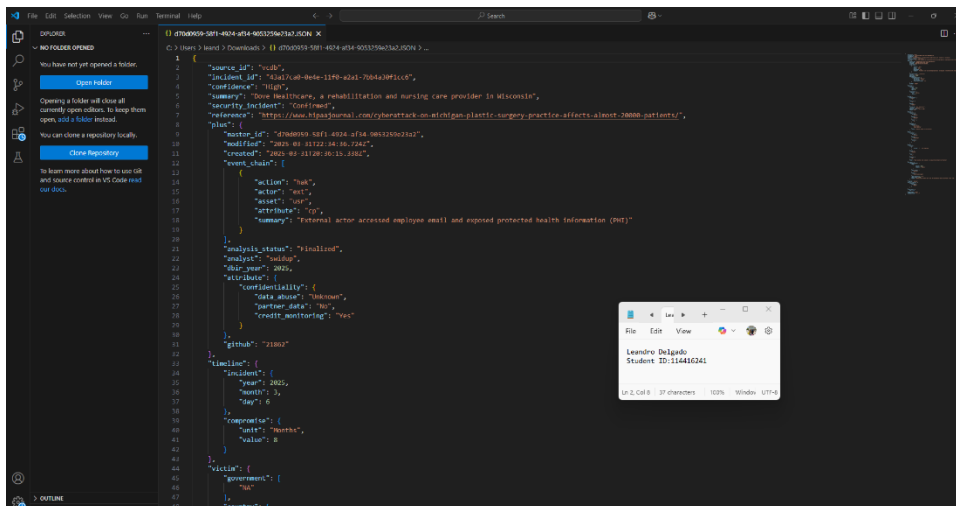
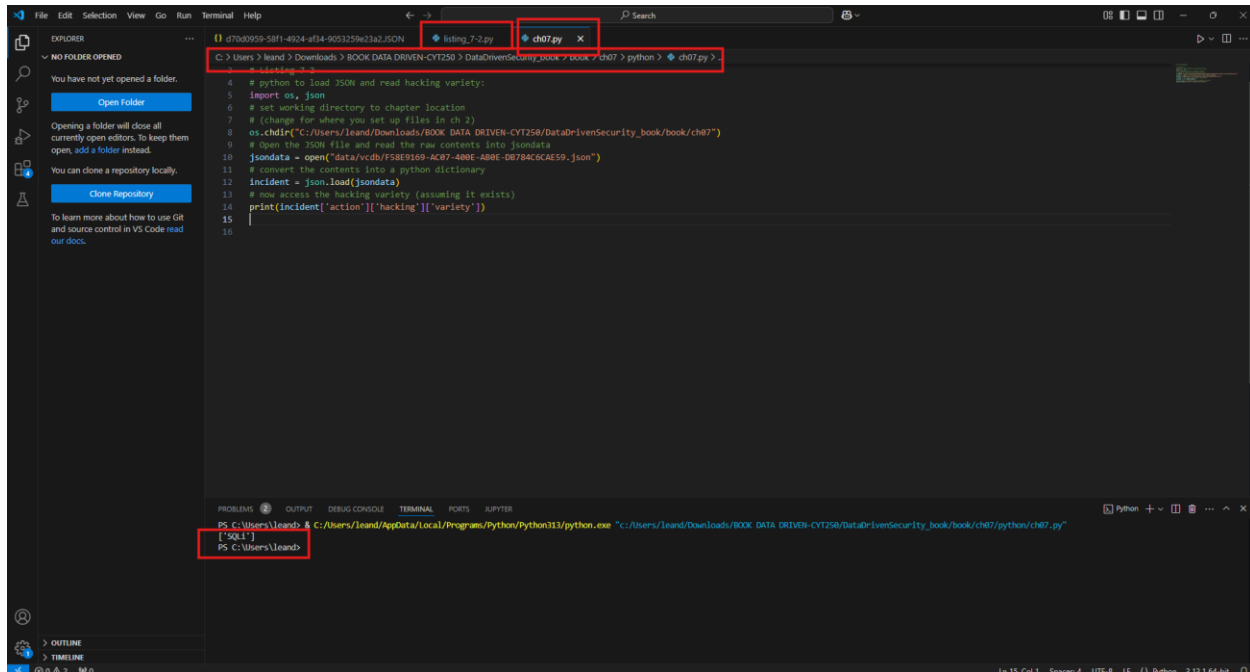


Figure 9. JSON file opened in Visual Studio Code

This screenshot shows the exported JSON file opened in Visual Studio Code, allowing for a clear, structured view of the finalized VERIS incident. The file contains key details such as source, timeline, incident summary, and threat actor actions. Reviewing it in this format enables precise validation and the opportunity to manually inspect or edit the data for future use or sharing.

Task 2. Learn how to search and retrieve JSON objects from VCDB via Python



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a file named `d7080959-58f1-4924-9052-99c242.json`. The main editor displays a Python script with the following content:

```
1 # python to load JSON and read hacking variety:
2 import os, json
3 # set working directory to chapter location
4 # (change for where you set up files in ch 2)
5 os.chdir("C:/Users/leand/Downloads/BOOK DATA DRIVEN-CYT250/DataDrivenSecurity_book/book/ch07")
6 # open the JSON file and read the raw contents into jsondata
7 jsondata = open("data/vcdb/f58e9169-ac07-480e-ab8e-d6784c5ae59.json")
8 # convert the contents into a python dictionary
9 incident = json.load(jsondata)
10 # now access the hacking variety (assuming it exists)
11 print(incident["action"]['hacking']["variety"])
12
```

The terminal at the bottom shows the command `python ch07.py` being executed, resulting in the output `['SQLi']`.

Figure 10. Running Python Script from Listing 7.2

This Python script retrieves and parses a JSON breach record from the VERIS Community Database (VCDB). It specifically accesses the "hacking" section of the "action" field to extract the variety of attacks used. The output ['SQLi'] confirms that the incident involved a SQL Injection attack, indicating unauthorized access through database query manipulation.

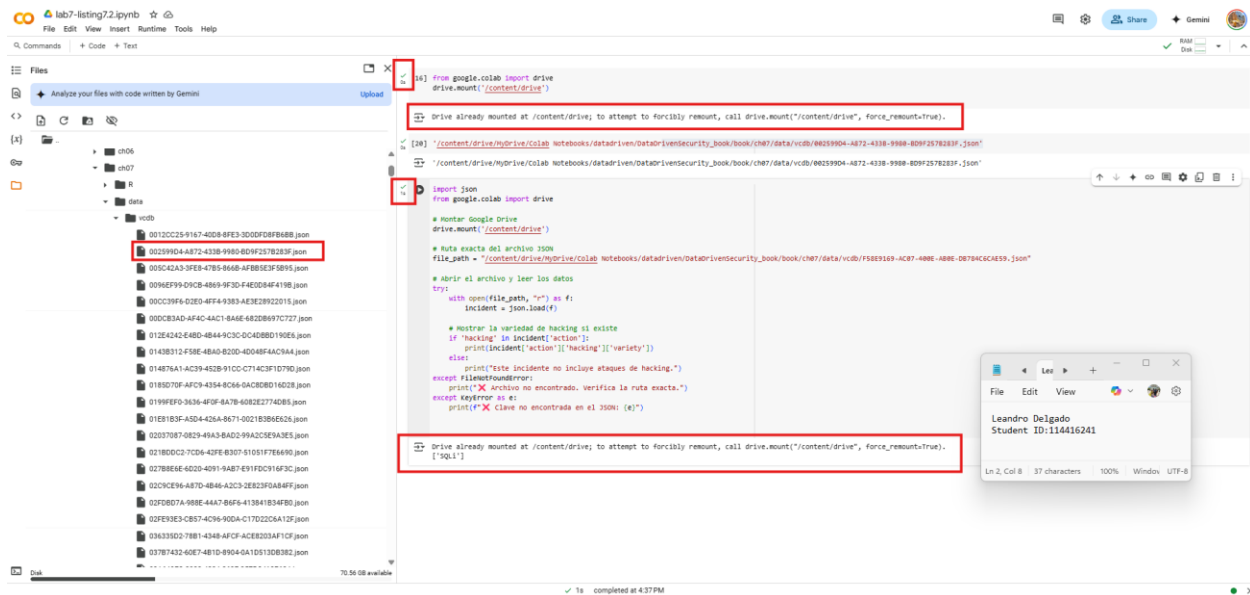


Figure 11. Verification of Python Script on Google Colab.

In this notebook, I replicated the JSON parsing and analysis process from Visual Studio Code to Google Colab to verify consistency across platforms. The goal was to ensure that the "hacking" vector—specifically 'SQLi'—was identified the same way regardless of the environment. This cross-platform validation confirms that the code behaves predictably and that the VCDB data can be analyzed reliably in both local and cloud-based workflows.

Additionally, working in Google Colab simplifies file navigation through Drive integration and enables easier sharing, especially in academic or team settings. It also reduces setup overhead, making it ideal for quick experimentation with large datasets like VCDB.

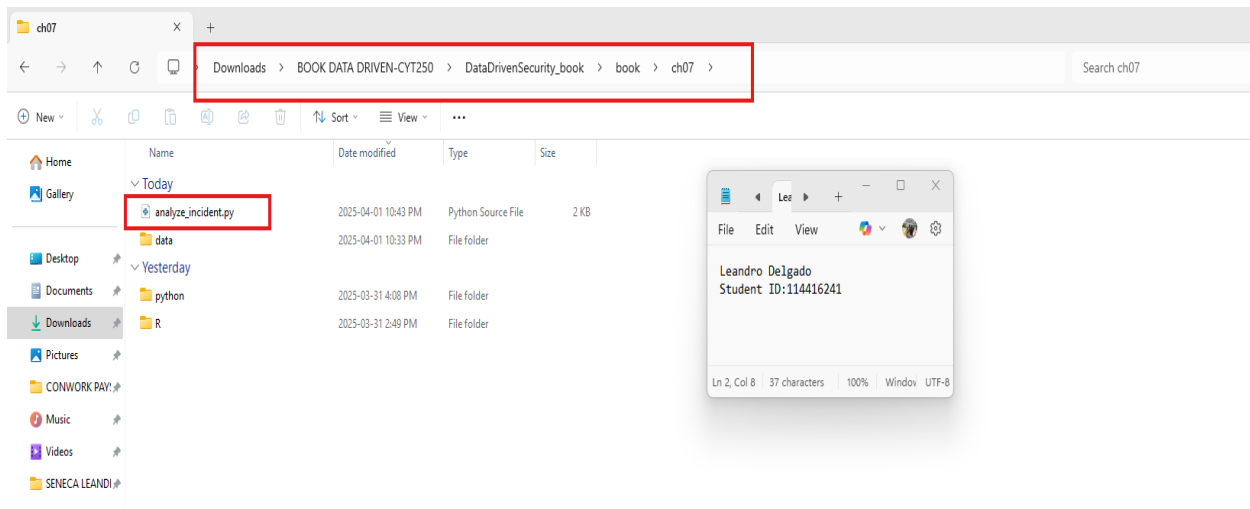


Figure 12. Py script successfully saved

This screenshot confirms that the `analyze_incident.py` script is correctly saved in the `ch07` folder of the project directory, following the structure suggested in the Data-Driven Security book. This setup ensures the script can easily access the JSON data located inside the `data` subfolder. My student information is also included for identification purposes.

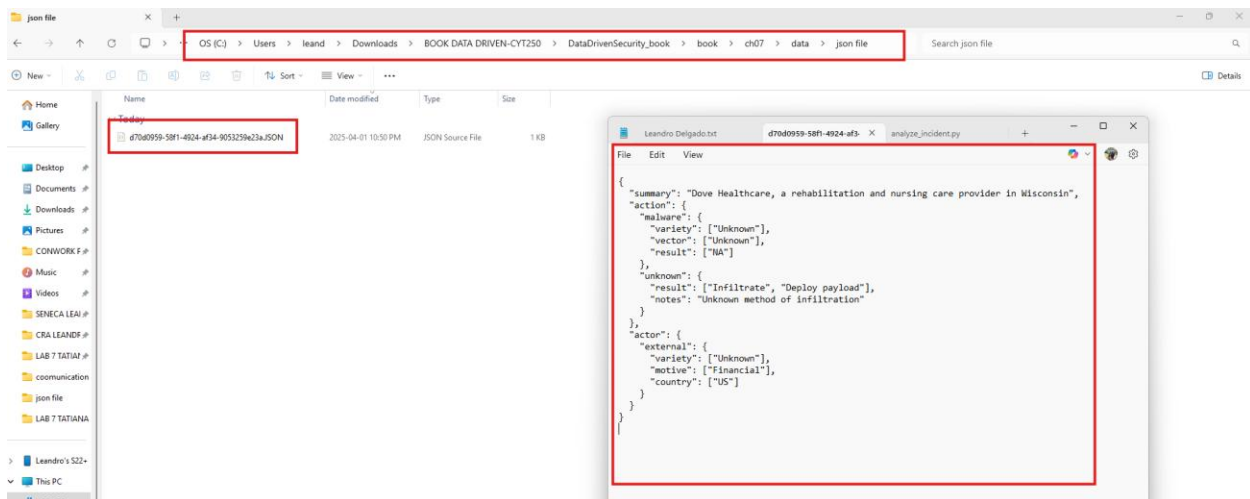
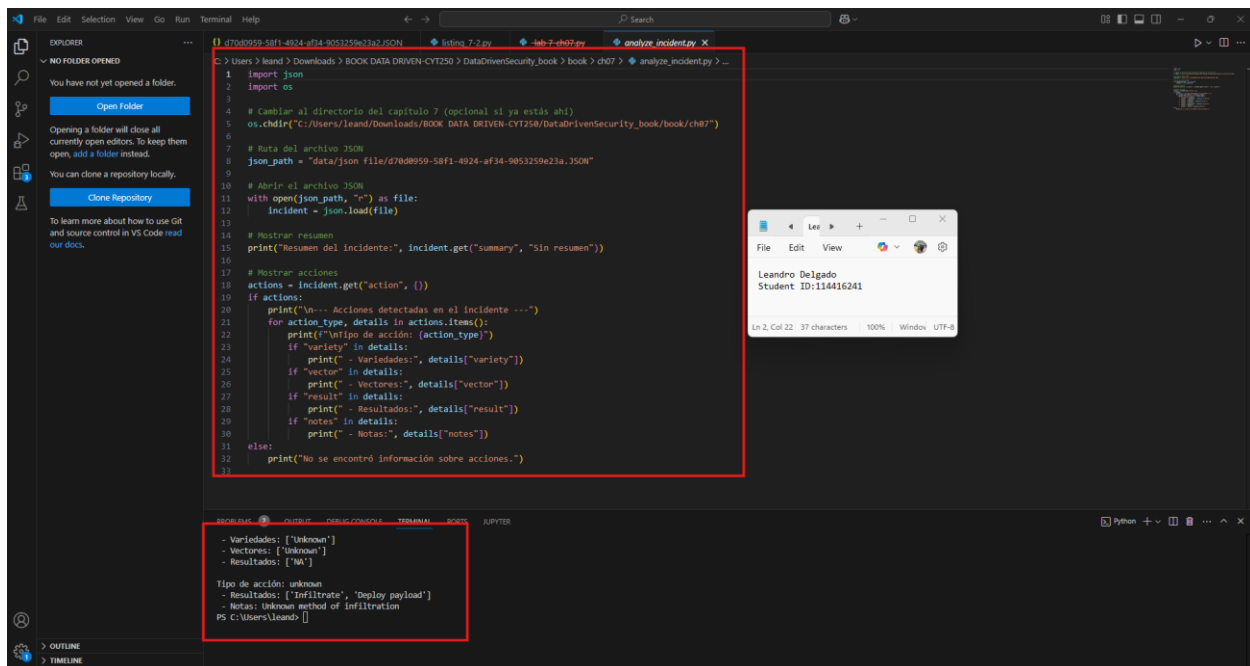


Figure 13. JSON file with valid content

This screenshot confirms that the JSON file `d70d0959-58f1-4924-af34-9053259e23a.JSON` has been correctly saved in the proper directory and contains a valid JSON structure. The file includes key elements such as a summary, action types with varieties and vectors, and actor information, making it ready for processing with the Python script.



```
1 import json
2
3
4 # Cambiar al directorio del capítulo 7 (opcional si ya estás ahí)
5 os.chdir("C:/Users/leand/Downloads/BOOK DATA DRIVEN-CYT250/DataDrivenSecurity_book/book/ch07")
6
7 # Ruta del archivo JSON
8 json_path = "data/json file/d7080959-58f1-4924-af34-9853250e23a.JSON"
9
10 # Abrir el archivo JSON
11 with open(json_path, "r") as file:
12     incident = json.load(file)
13
14 # Mostrar resumen
15 print("Resumen del incidente:", incident.get("summary", "Sin resumen"))
16
17 # Mostrar acciones
18 actions = incident.get("action", {})
19 if actions:
20     print("Se detectaron acciones en el incidente:")
21     for action_type, details in actions.items():
22         print(f"Tipo de acción: {action_type}")
23         if "variety" in details:
24             print(f"  - Variedades: {details['variety']}")
25         if "vector" in details:
26             print(f"  - Vectores: {details['vector']}")
27         if "result" in details:
28             print(f"  - Resultados: {details['result']}")
29         if "notes" in details:
30             print(f"  - Notas: {details['notes']}")
31     else:
32         print("No se encontró información sobre acciones.")
33
```

```
- Variedades: ['Unknown']
- Vectores: ['Unknown']
- Resultados: ['NA']

Tipo de acción: unknown
- Resultados: ['Infiltrate', 'Deploy payload']
- Notas: Unknown method of infiltration
PS C:\Users\leand>
```

Figure 13.Script execution with output in Visual Studio Code

This screenshot shows the complete `analyze_incident.py` script within Visual Studio Code, alongside the console output. The script successfully reads and parses the JSON file, then prints the incident summary and detailed action types. This confirms the logic works as expected and that the JSON data is properly structured. My student ID is also shown for identification.

Summary

The key lesson learned in this lab is the importance of understanding how to navigate, parse, and validate cybersecurity incident data using structured formats like VERIS. By working hands-on with JSON files and using tools like Google Colab and Visual Studio Code, I gained practical experience in file handling, data extraction, and identifying breach patterns such as SQL injection. This exercise also reinforced how attention to file paths, schemas, and precise formatting is essential in real-world data analysis and threat intelligence workflows.