Visual Analytics of Control-Flow Graphs with Node-Importance Classification

Supasan Muanjit
Advisor Asst. Prof. Dr. Jakapan Suaboot

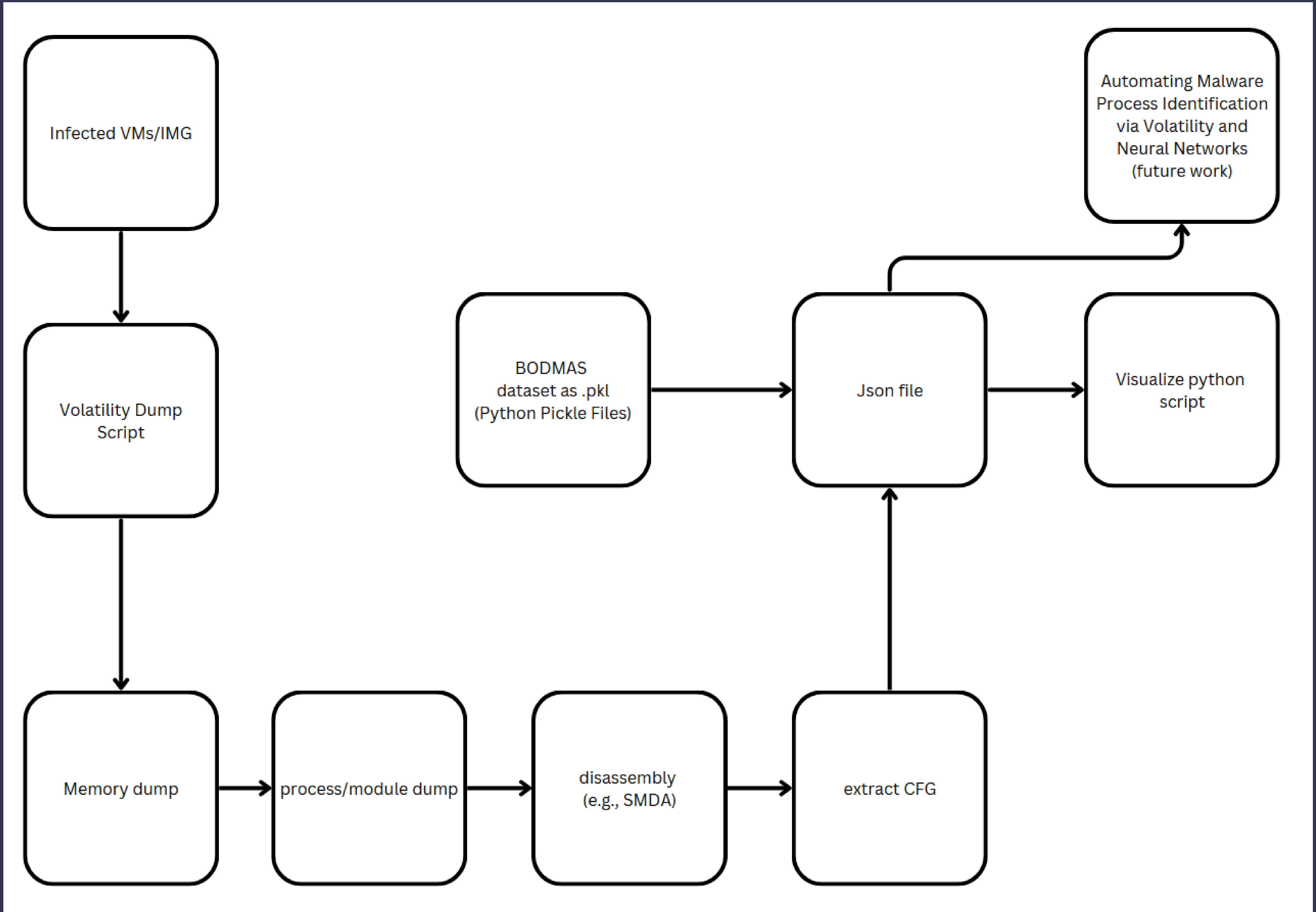
Project Overview

- Analysis and visualization of malware behavior from pre process data
- Use of Python libraries (pandas, matplotlib, NetworkX, pyvis)
- Objective: Identify malicious process indicators visually and structurally

Objectives

- Extract process metadata from memory dumps using Volatility Framework
- Construct Control Flow Graphs (CFGs)
- Provide interactive visualization for malware detection
- Lay foundation for machine learning integration

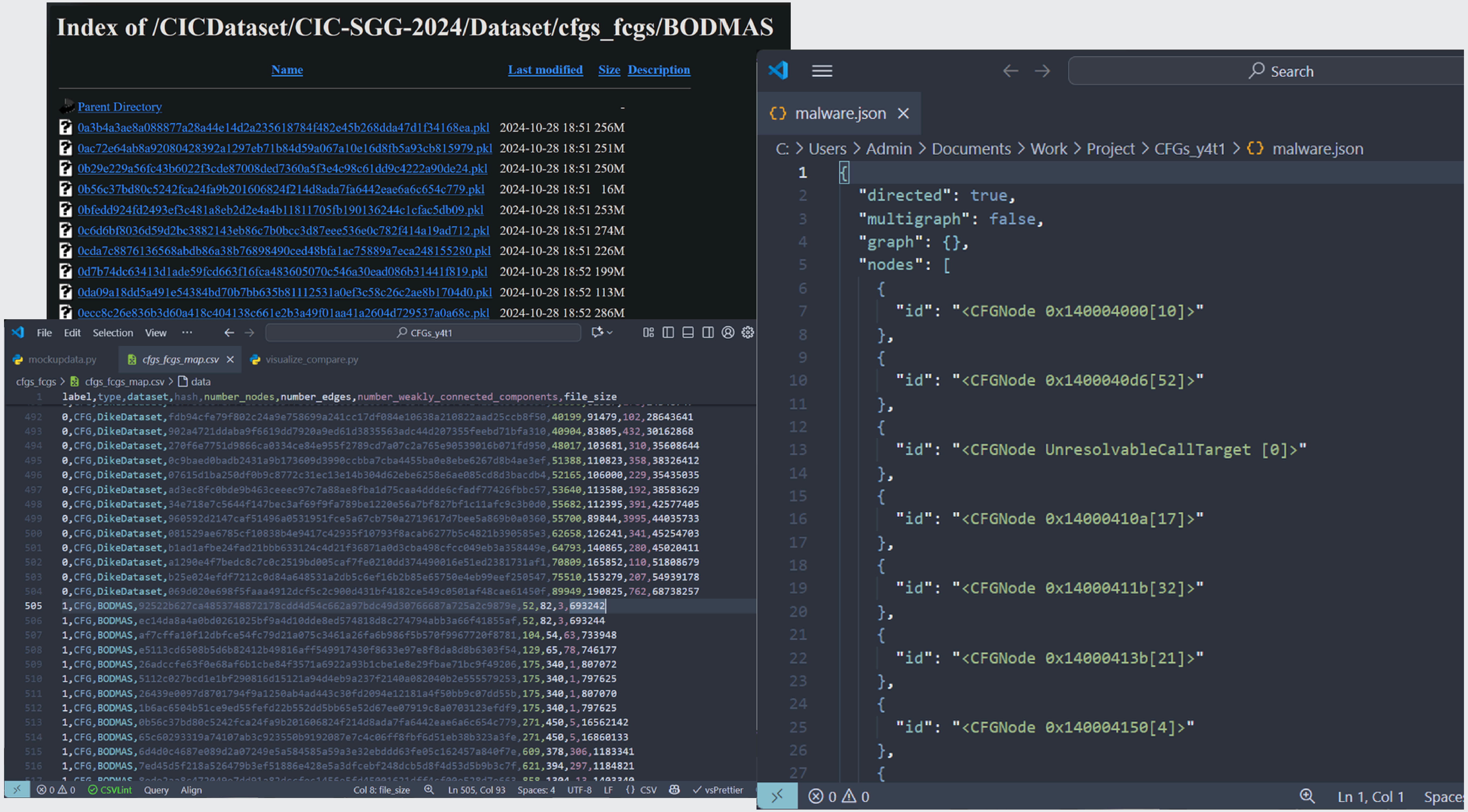
Methodology


Infected VMs/IMG
Volatility Dump Script
Memory dump
process/module dump
disassembly (e.g., SMDA)
extract CFG
BODMAS dataset as .pkl (Python Pickle Files)
Json file
Visualize python script
Automating Malware Process Identification via Volatility and Neural Networks (future work)

- **Memory Acquisition** Infected VMs or images are processed with Volatility to obtain raw memory dumps.
- **Process/Module Extraction** From the memory dumps, individual processes or modules suspected of being malware are isolated and dumped.
- **Disassembly** Tools such as SMDA (or similar) are used on the module dumps to disassemble executable code, allowing for static analysis.
- **CFG Extraction** Disassembled code is analyzed to extract the Control Flow Graph (CFG), which represents the execution paths within the malware's code.
- **Data Transformation** The resulting data—including CFGs—is structured into JSON files, enabling easier downstream processing and visualization.
- **Dataset Integration** Alternatively, pre-existing datasets such as BODMAS can be loaded in Python pickle (.pkl) format and converted into the same JSON structure for consistency.
- **Visualization and Automation** The JSON data can be visualized using Python scripts. The final goal is to automate malware process identification using machine learning techniques, with Volatility and neural networks suggested as future work.

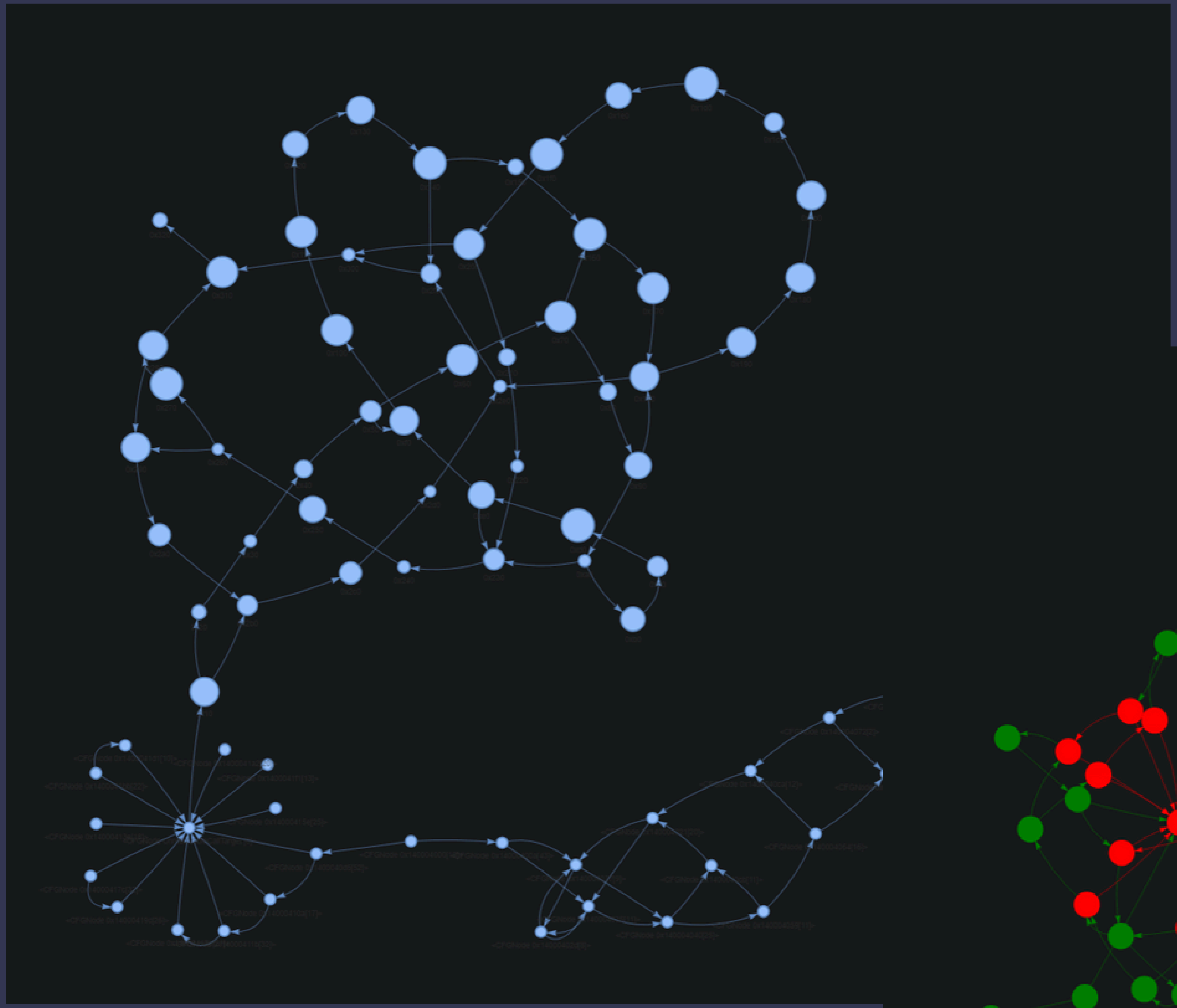
Data Exploration

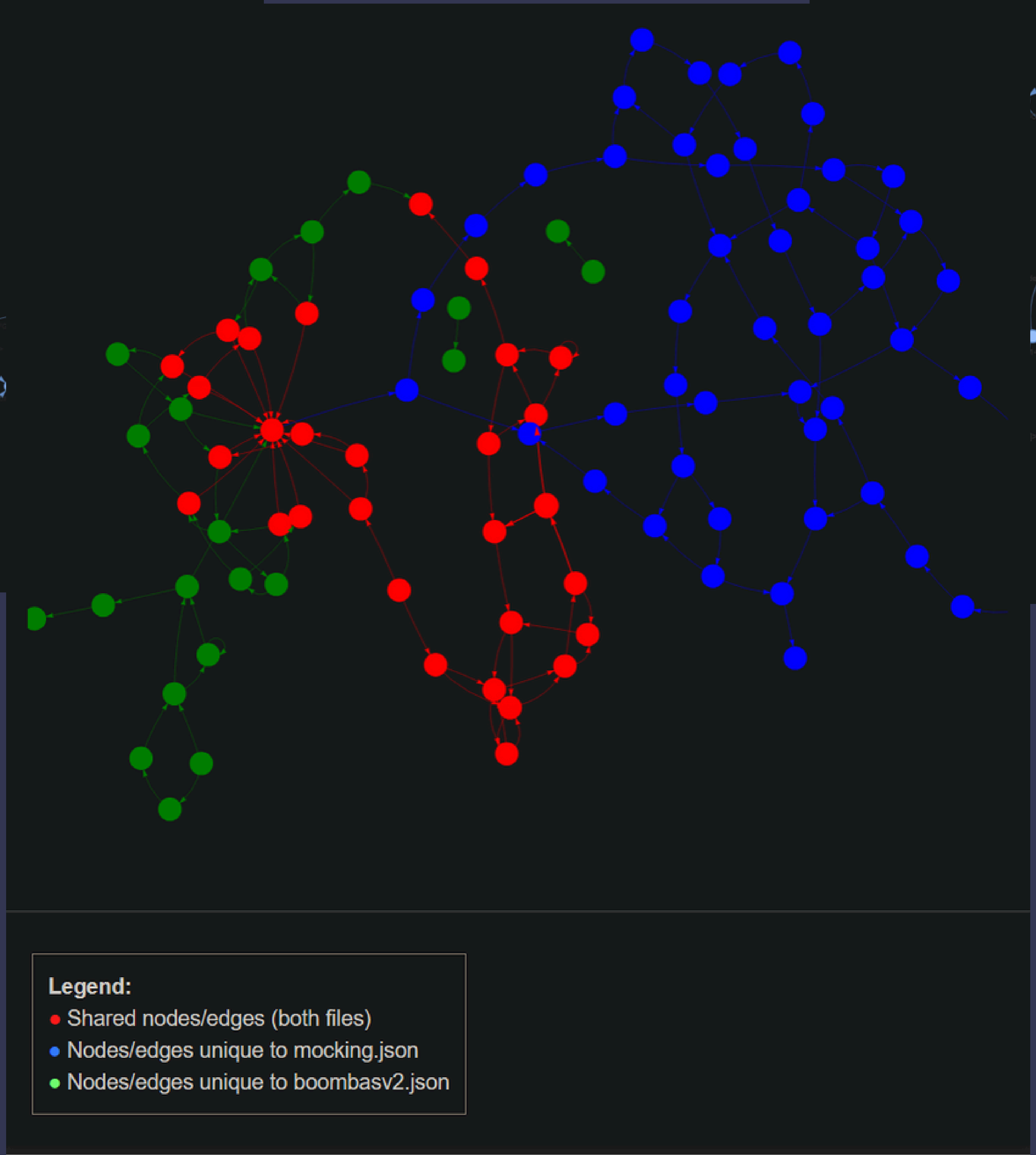
Dataset Overview

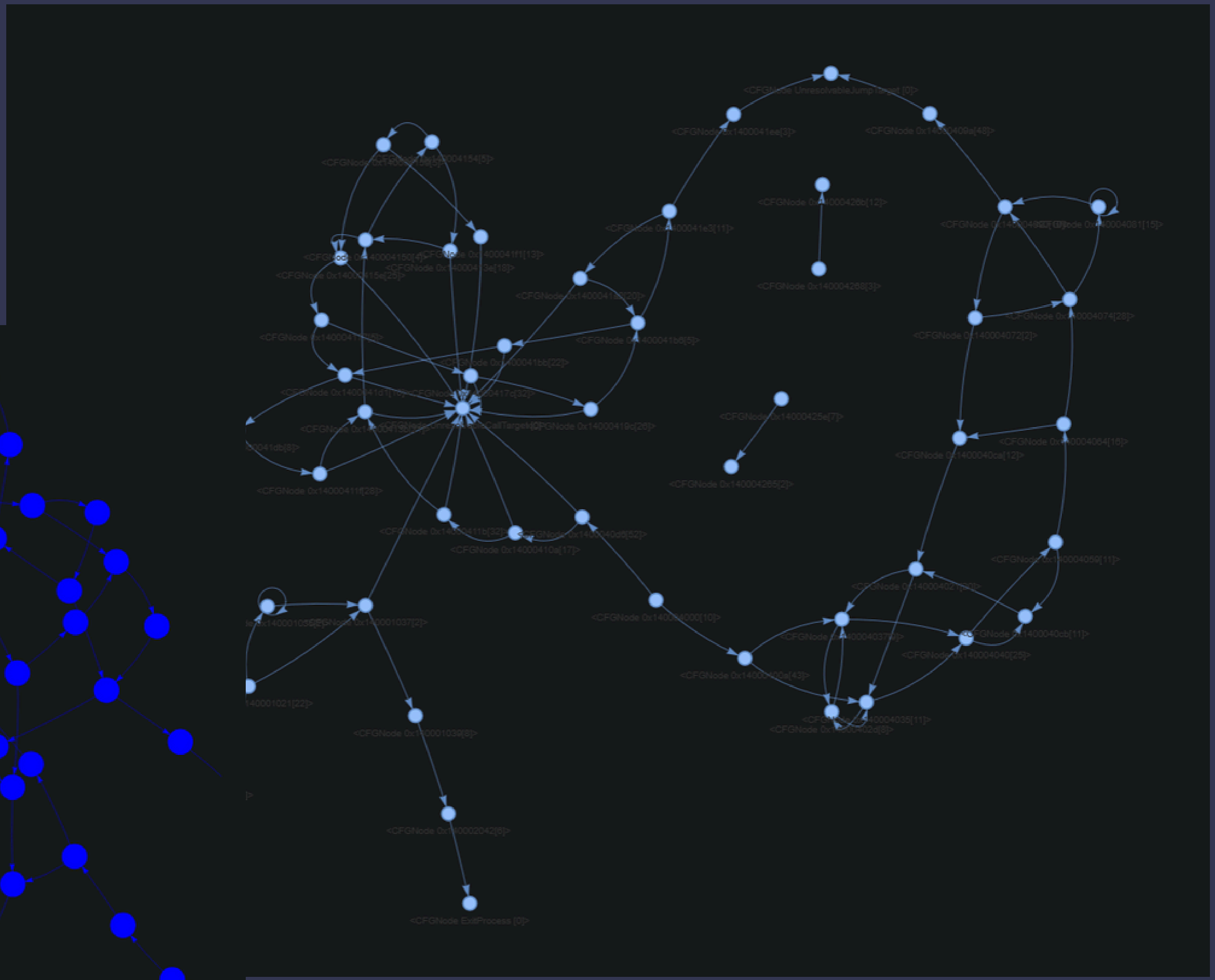


Dataset from BODMAS

Data Virtualization







*Each node's name identifies a specific execution block or function within the analyzed process

*The naming allows forensic analysts to trace which section of code is being executed, whether it's benign or potentially malicious, supporting targeted investigation and cross-referencing with known malware signatures

Demonstration

Summary

What has been Done

Constructed Control Flow Graphs (CFGs) representing malware execution paths using NetworkX.

Created interactive visualizations of CFGs with pyvis, showing shared and unique code paths with color coding.

Implemented basic heuristic anomaly detection to highlight suspicious nodes and edges.

What can improve

Do web version so it can be easy to accese

Reference

[1] A. Huseinović and S. Ribić, "Virtual machine memory forensics," 2014.
[2] A. Afreen, M. Aslam and S. Ahmed, "Analysis of Fileless Malware and its Evasive Behavior," in 2020 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 2020.
[3] M. Ficco, "Malware Analysis by Combining Multiple Detectors and Observation Windows," IEEE Transactions on Computers, vol. 71, pp. 1276 - 1290, 2021.
[4] Akash Thakar; Rakesh Singh Kunwar; Hemang Thakar; Kapil Kumar; Chintan Patel, "Enhanced Malware Detection Method Using Baseline Comparison in Memory Forensics," in 2023 IEEE 11th Region 10 Humanitarian Technology Conference (R10-HTC), Rajkot, India, 2023.

Q&A