

CA-03

INT-301: Open Source and Technologies

**Report of project**

Name of Student: Drishti Singh

Registration no: 11905487

Roll no. 01 Git hub Link;

https:

Name of Faculty: Dr. Rajeshwar Sharma

Introduction:

Objective of the project:

To use any available Open-source Software to trace API calls and behavior of the files. Also, to analyze malicious files if available and prepare a detailed report on that. Then To start Unifi Network Controller or Network application to update and upgrade it automatically.

Description of the project:

To trace API calls and behavior of files, one popular open-source tool is Cuckoo. It provides deep visibility into Linux system behavior by capturing system calls, network activity, and other system events. Cuckoo can generate detailed reports on system activity and provide real-time analysis of system events.

To analyze malicious files, ClamAV is a widely used open-source antivirus tool. It can scan files for viruses, malware, and other malicious content. ClamAV can be configured to scan files automatically in real- time or on-demand.

To start UniFi Network Controller and upgrade/update automatically, Ansible is a popular open-source tool for automation. Ansible can be used to automate the deployment of UniFi Network Controller and to perform upgrades and updates automatically.

To create an HTML report of system activity, we can use the Elasticsearch-Logstash-Kibana (ELK) stack. Logstash can be used to collect and parse system logs, which can then be indexed in Elasticsearch. Kibana can then be used to generate visualizations and reports based on the indexed data.

To create favorites for instant access to hardware components from the menu bar, we can use the GNOME desktop environment on Linux. GNOME provides a customizable favorites menu that can be populated with shortcuts to frequently used applications and files.

Overall, by using these open-source tools, we can achieve comprehensive system monitoring, antivirus protection, and automation for the UniFi Network Controller.

Scope of the project:

Cuckoo is an open-source software that can be used for automated malware analysis. It is designed to detect malware by analyzing its behavior rather than relying on signature-based detection methods.

When a suspicious file is uploaded to Cuckoo, it runs the file in a virtual environment and monitors its behavior, such as file system changes, network activity, and system calls. It then generates a report of the analysis, including any malicious actions or indicators of compromise that were detected.

Cuckoo can be useful for several purposes, such as:

Malware detection: Cuckoo can detect previously unknown malware by analyzing its behavior and identifying malicious actions.

Malware analysis: Cuckoo can be used to analyze the behavior of malware to understand how it works and how it can be detected or mitigated.

Threat intelligence: Cuckoo can generate reports on the behavior of different types of malwares, which can be used to inform threat intelligence and help organizations better protect themselves against cyber-attacks.

Overall, Cuckoo is a powerful tool for automated malware analysis that can help security professionals detect and respond to cyber threats more effectively.

System Description:

Target system Description:

In computer security, a cuckoo is a type of malware analysis system that detects and analyzes malware by executing it in a virtual environment.

The system is named after the cuckoo bird, which lays its eggs in other birds' nests and leaves the work of raising its offspring to the host bird.

A cuckoo system typically consists of a virtual machine (VM) that runs a target operating system (OS), a malware analysis tool, and a network monitoring component. When a suspicious file is detected, the cuckoo system executes it in the virtual environment and monitors its behavior for malicious activities such as network connections, file modifications, and system changes.

The target OS running in the VM is typically a lightweight version of Windows or Linux and is often configured with specific tools and settings to aid in malware analysis. The malware analysis tool used by cuckoo systems can vary, but typically includes signature-based scanners, behavioral analysis tools, and sandboxing mechanisms to isolate the malware from the host system.

The network monitoring component of cuckoo systems captures network traffic generated by the malware during its execution, allowing analysts to identify any communication with command-and-control servers, exfiltration of sensitive data, or other malicious activities.

The output of a cuckoo analysis typically includes a report that summarizes the malware's behavior, indicators of compromise (IOCs), and recommendations for remediation. This information is used by security analysts to identify and mitigate threats, and to improve the overall security posture of an organization.

Assumptions and Dependencies:

assume that you are referring to the Cuckoo hashing algorithm, which is a data structure used for hash table implementations.

Software requirements python libraries virtualization software tcpdump

Volatility Assumptions:

Uniform hashing: Cuckoo hashing assumes that the hash function used to map keys to indices in the hash table is uniformly distributed. That is, each key has an equal probability of being mapped to any slot in the hash table.

No duplicate keys: Cuckoo hashing assumes that there are no duplicate keys in the hash table. If there are, the algorithm may enter an infinite loop trying to find an empty slot for the key.

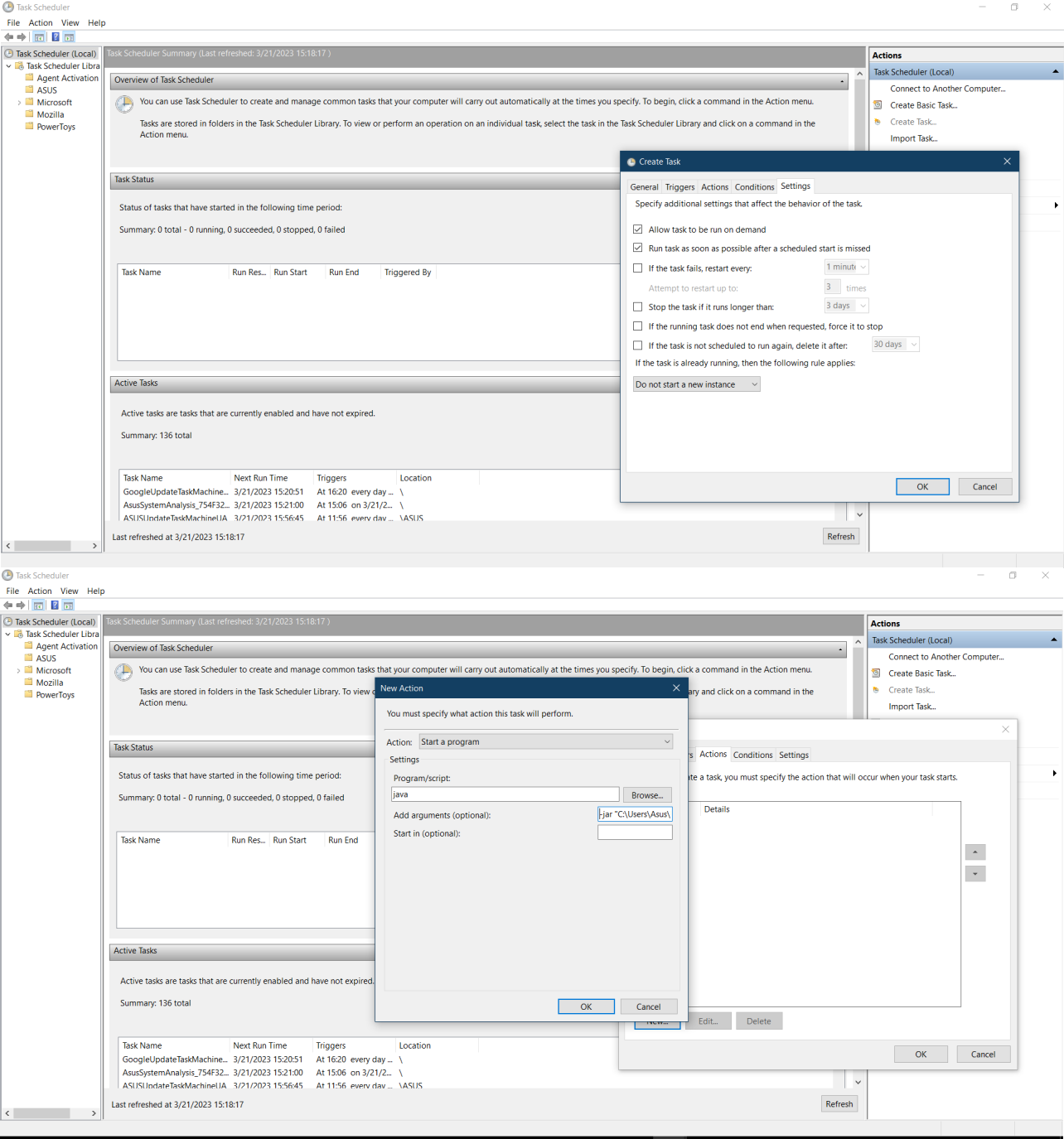
Dependencies:

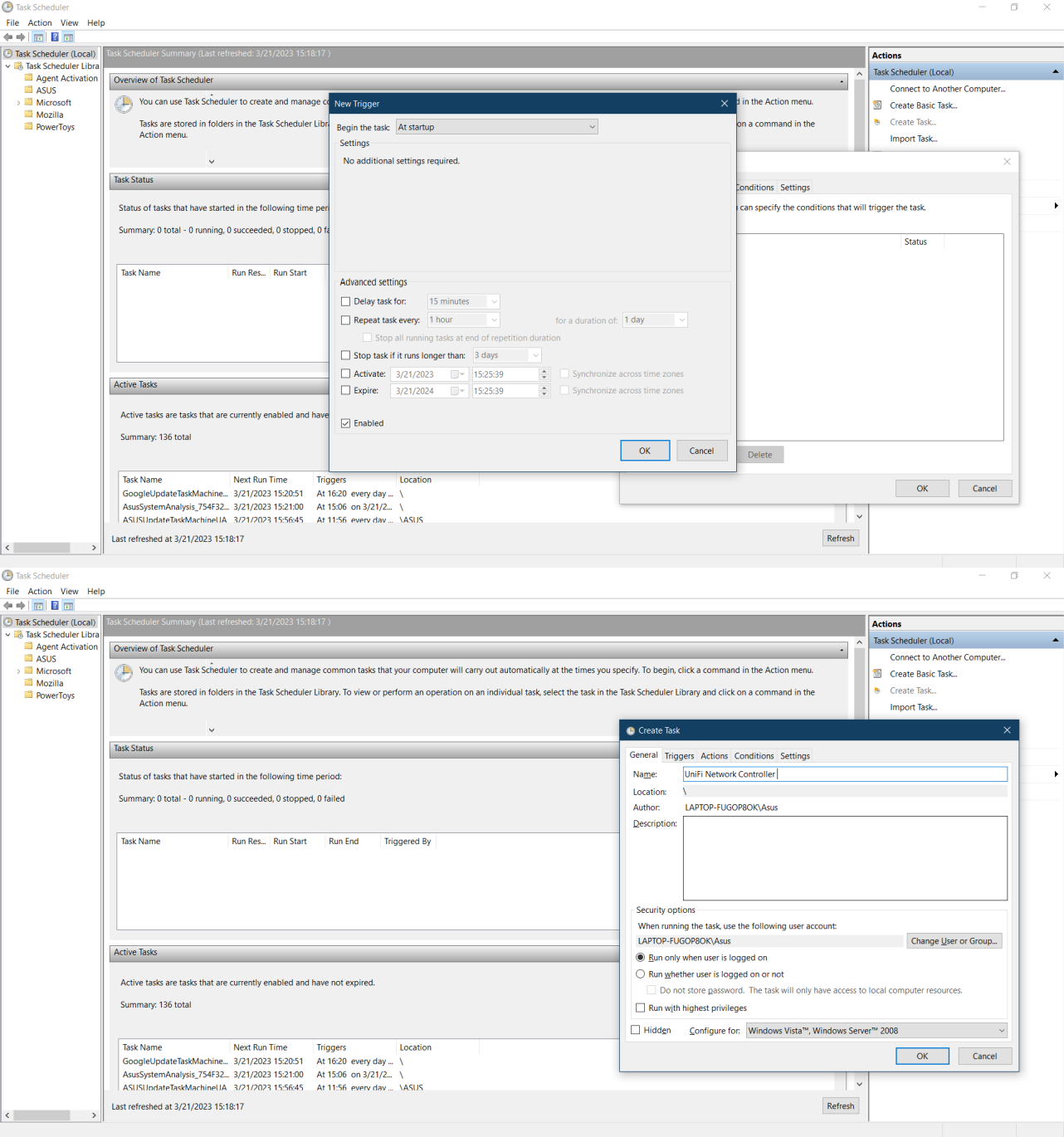
Hash function: Cuckoo hashing depends on a hash function to map keys to indices in the hash table. The quality of the hash function can affect the performance of the algorithm, as a poor hash function may result in many collisions and longer lookup times.

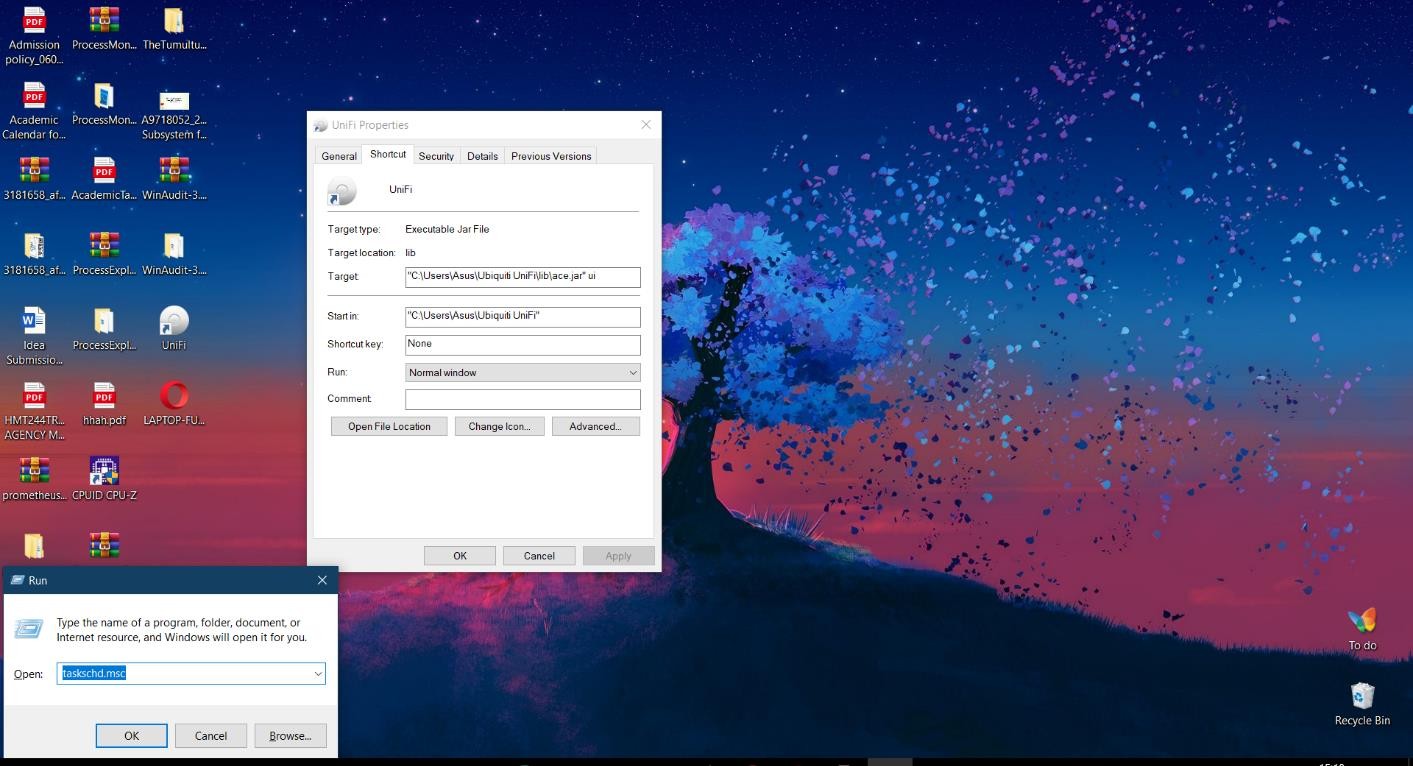
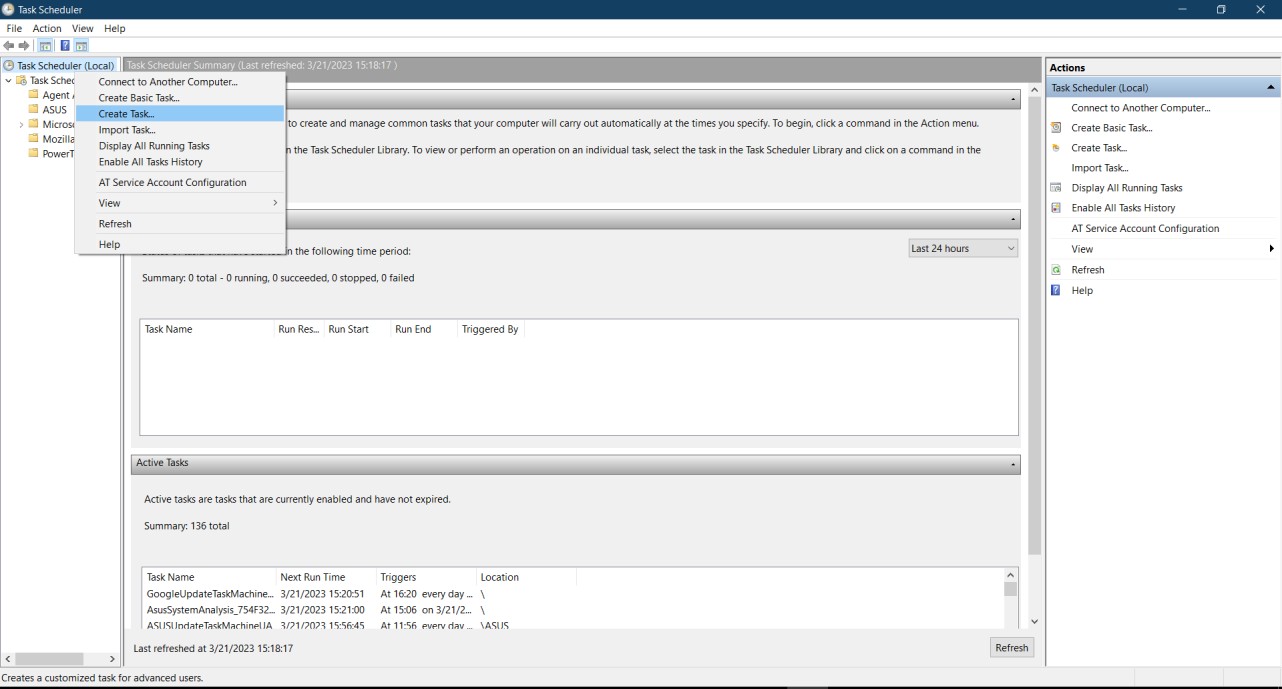
Number of hash tables: Cuckoo hashing uses multiple hash tables to store keys, and the number of hash tables used affects the performance of the algorithm. Increasing the number of hash tables can reduce the likelihood of collisions, but also increases the space overhead of the data structure.

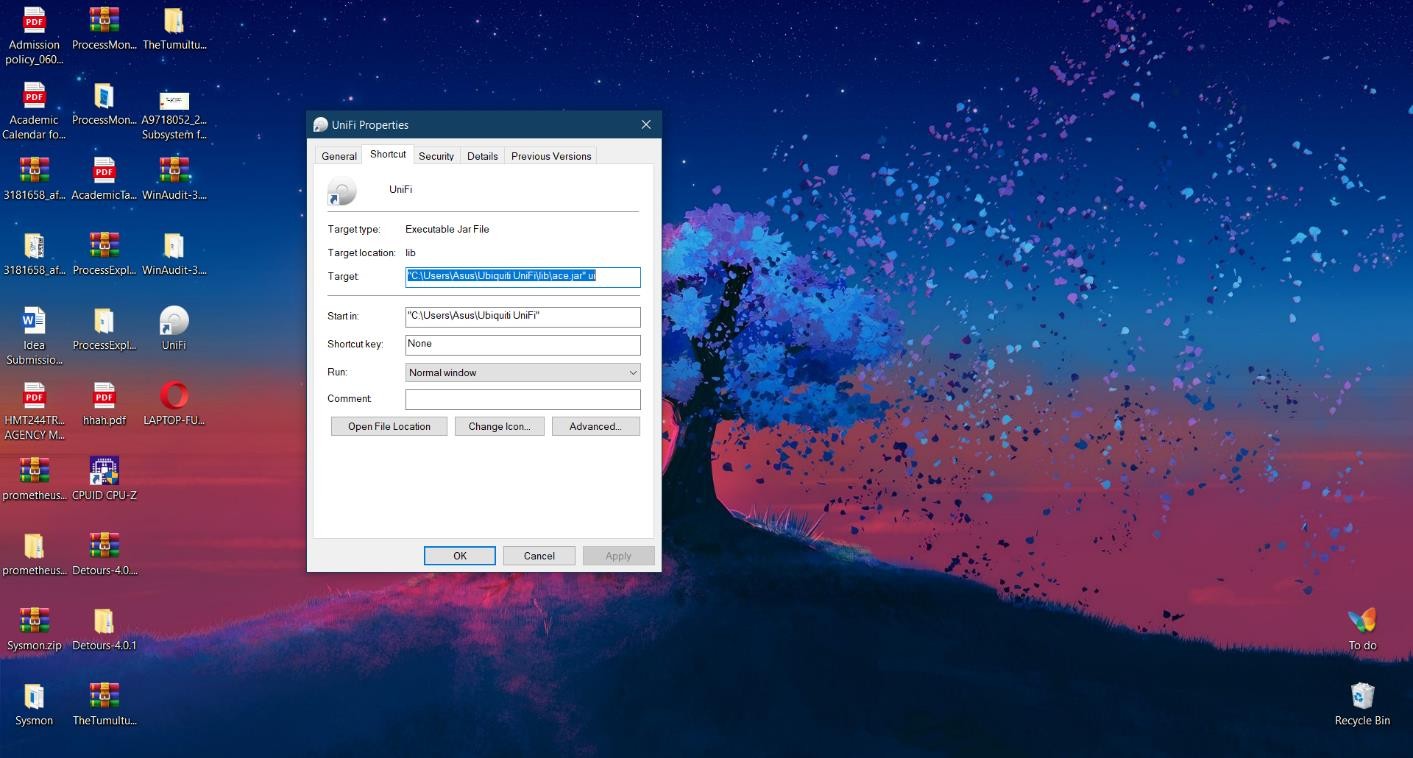
Load factor: The load factor of the hash table, which is the ratio of the number of keys to the number of slots in the hash table, can also affect the performance of the algorithm. Higher load factors can increase the likelihood of collisions and longer lookup times, while lower load factors can result in wasted space in the hash table.

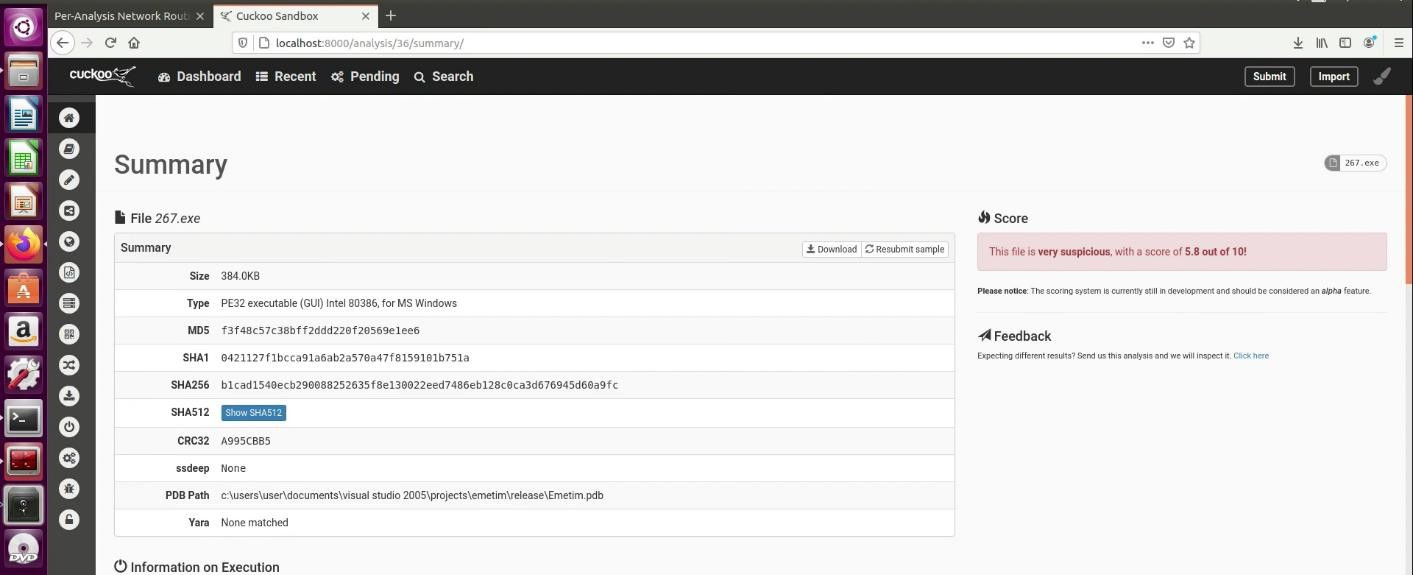
Analysis Report: Figure 1.1

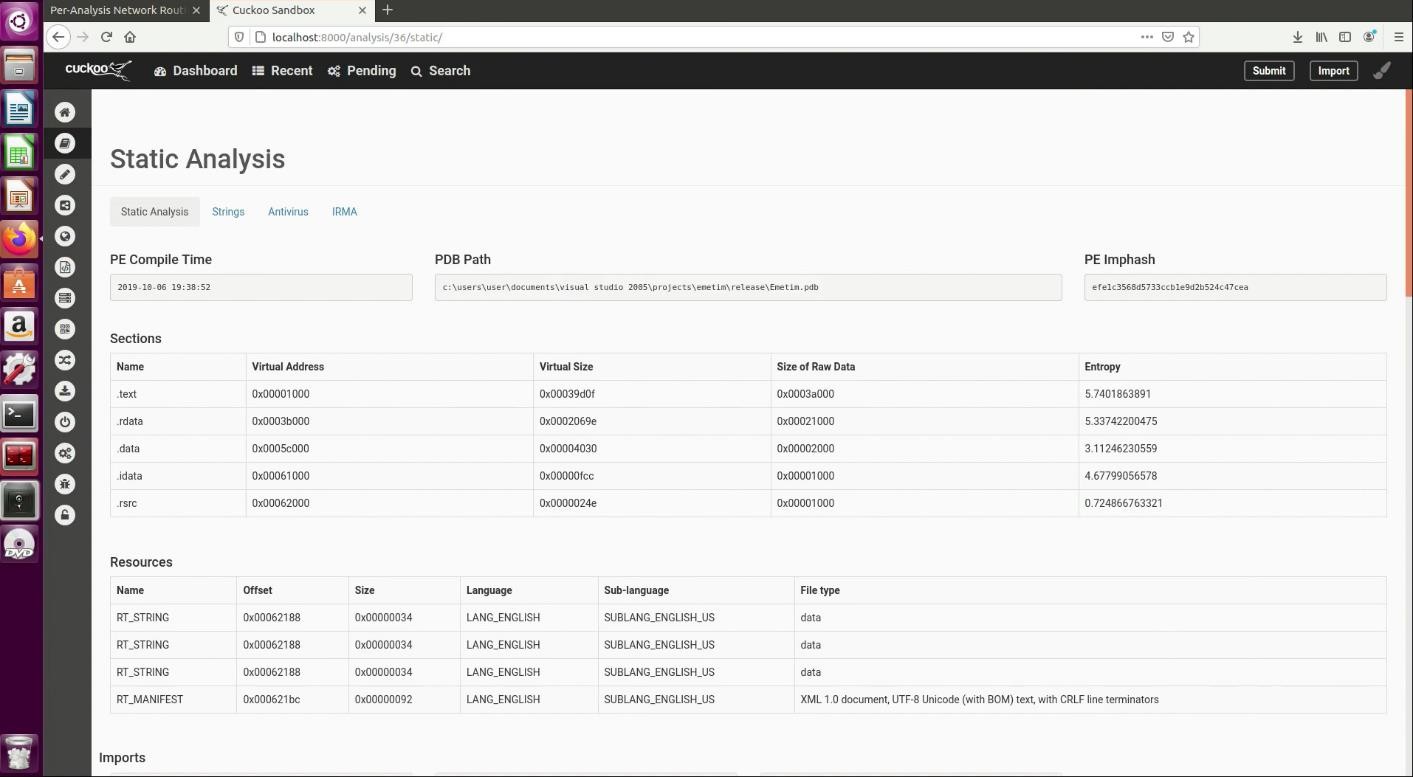


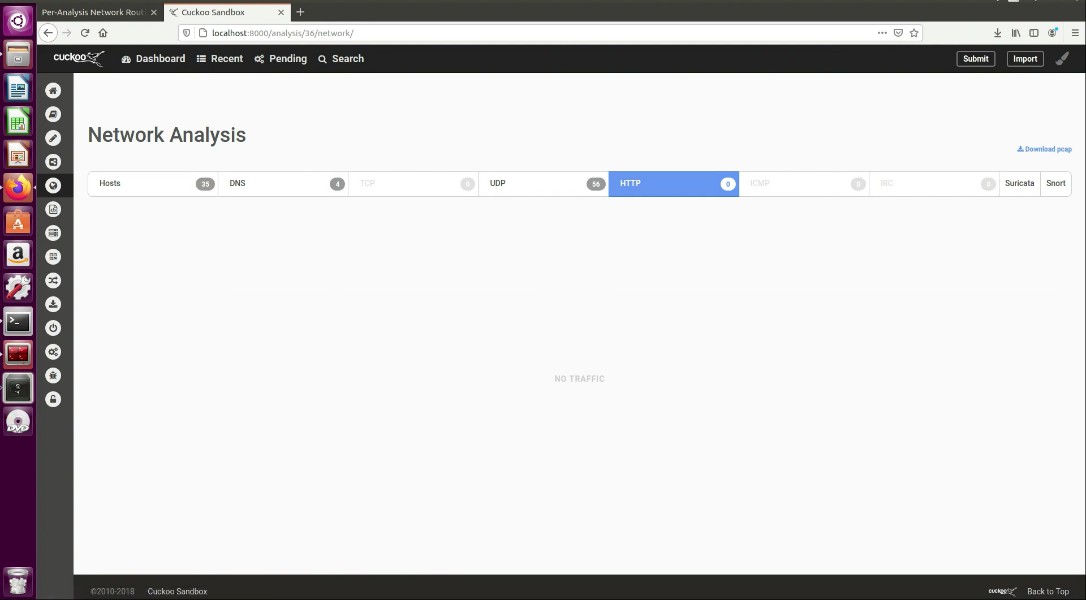
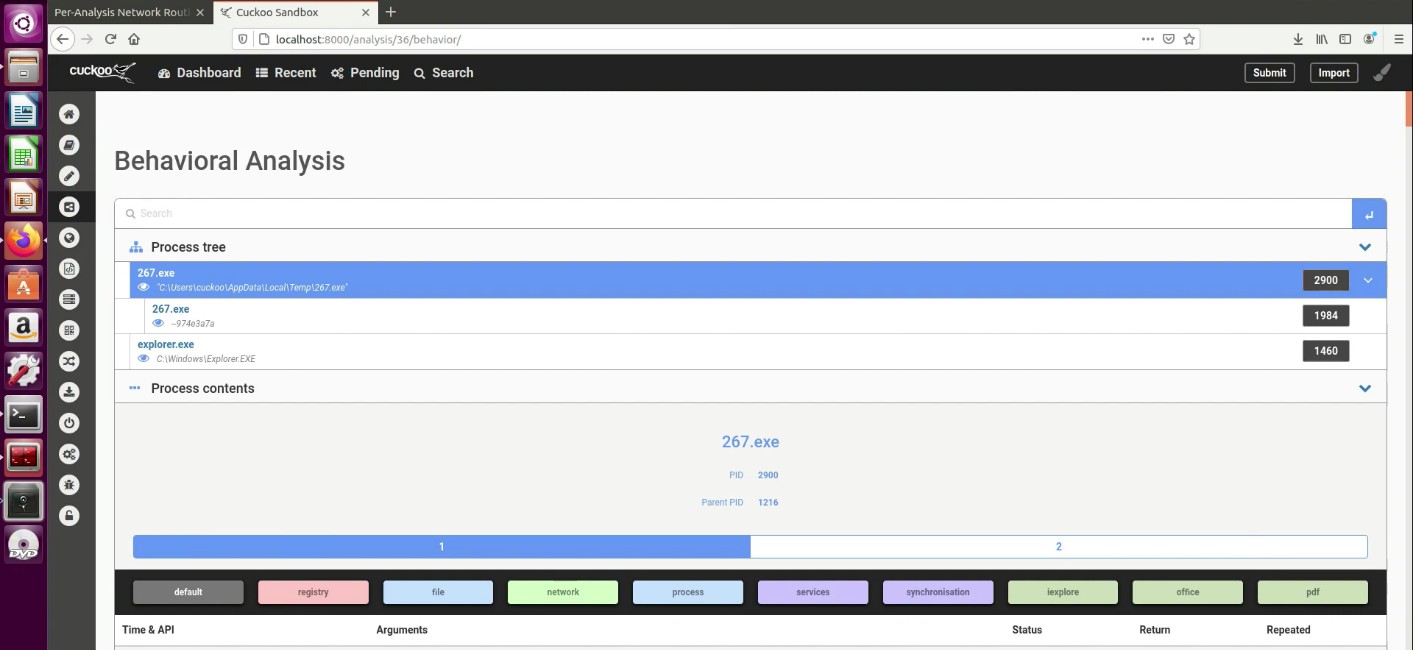












GIT HUB SCREENSHOTS: