**PROJECT ARES**

Autonomous Rivalry & Evolution System

*Advanced Cyber Defense Simulation Framework*



Prepared by: Rao Hamd Rizwan  
Department of Cyber Security  
Air University

Contents

[1. Objective 2](#_Toc221050195)

[Visualization of the entire Network 3](#_Toc221050196)

[2. Infrastructure Setup (The Arena) 3](#_Toc221050197)

[2.1. System Architecture 3](#_Toc221050198)

[2.2. Challenges & Solutions 4](#_Toc221050199)

[2.3. Verification 4](#_Toc221050200)

[3. The Nervous System (eBPF Monitoring) 5](#_Toc221050201)

[3.1. Kernel Hooking with Python BCC 5](#_Toc221050202)

[3.2. Advanced Forensic Visibility 5](#_Toc221050203)

[4. The Brain (AI Intelligence & Active Defense) 6](#_Toc221050204)

[4.1. Connecting the Intelligence 6](#_Toc221050205)

[4.2. Hunter-Killer Mode (IPS) 7](#_Toc221050206)

[4.3. Attack Simulation Results 7](#_Toc221050207)

[5. The War Room (Visualization) 8](#_Toc221050208)

[5.1. Dashboard Implementation 8](#_Toc221050209)

[5.2. Real-Time Telemetry 8](#_Toc221050210)

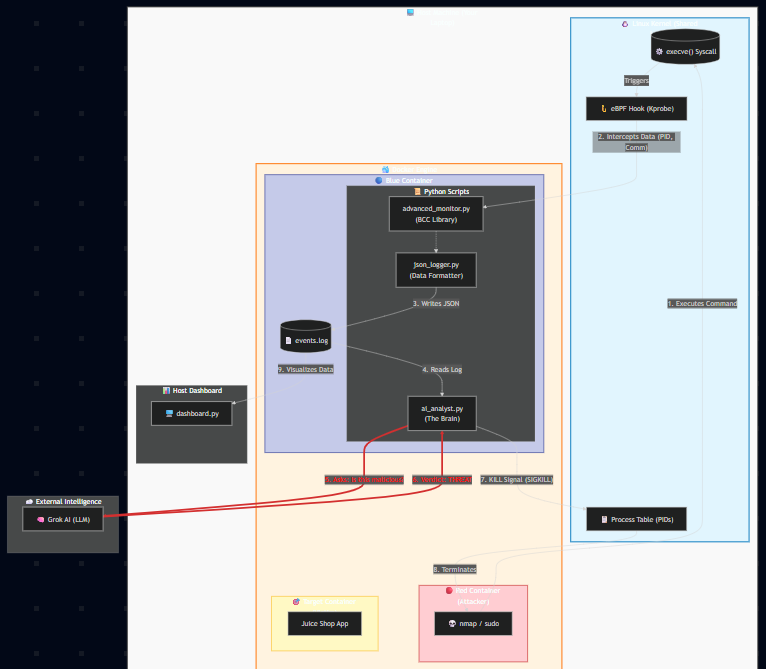
[6. Conclusion 9](#_Toc221050211)

# 

# 1. Objective

The A.R.E.S. project establishes a containerized adversarial network designed to simulate, detect, and neutralize cyber threats in real-time. By leveraging **eBPF (Extended Berkeley Packet Filter)** for kernel-level observability, the system bridges the gap between passive monitoring (IDS) and active defense (IPS). The final implementation integrates an AI "Brain" (Grok LLM) that analyzes system calls and automatically terminates malicious processes, visualized through a live "War Room" dashboard.

## Visualization of the entire Network



# 2. Infrastructure Setup (The Arena)

## 2.1. System Architecture

The foundation of the project is a "Purple Team" environment built on Docker, consisting of three distinct nodes:

* **Red Node (Attacker):** A headless Kali Linux container pre-loaded with offensive tools like Nmap.
* **Blue Node (Defender):** A privileged Ubuntu container acting as the sentinel, capable of monitoring the host kernel.
* **Target Node (Victim):** OWASP Juice Shop, a vulnerable web application used as a sandbox for attacks.

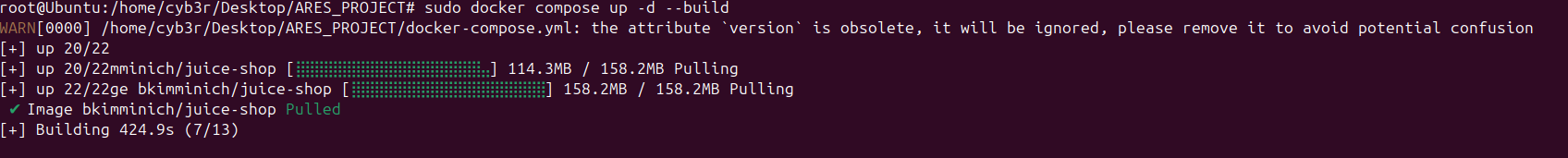
## 2.2. Challenges & Solutions

A key technical challenge arose regarding docker-compose version mismatches on the host Ubuntu system. This was resolved by migrating the entire infrastructure to the Docker Compose Plugin V2 syntax (docker compose), ensuring seamless orchestration.

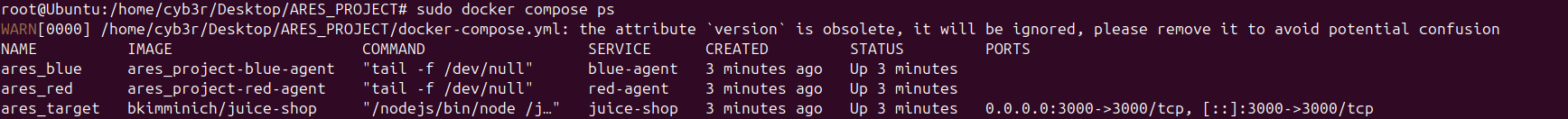
## 2.3. Verification

We verified network isolation and connectivity by confirming the Red Node had <1ms latency to the Target Node and that the Blue Node successfully mounted the host's /sys/kernel/debug directory for tracing.

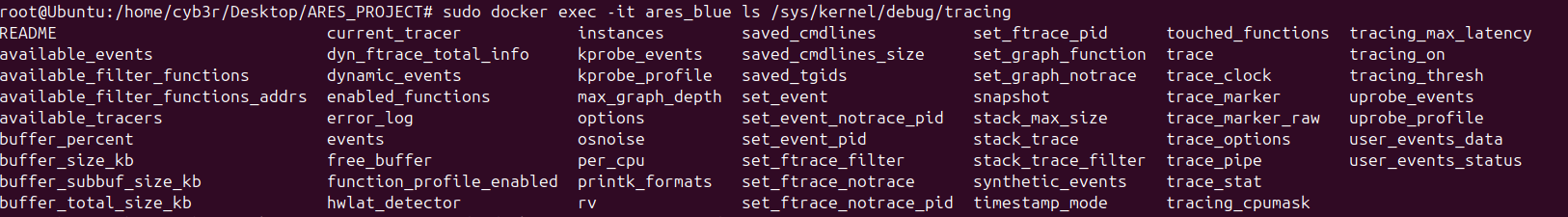
* **Sudo docker compose up -d --build**



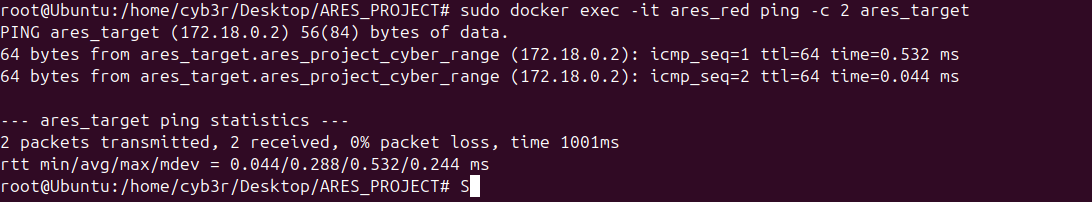
* **Confirming if the ‘arena/docker ’ is online**



* **Blue container can see the host container**



* **Red team connectivity**

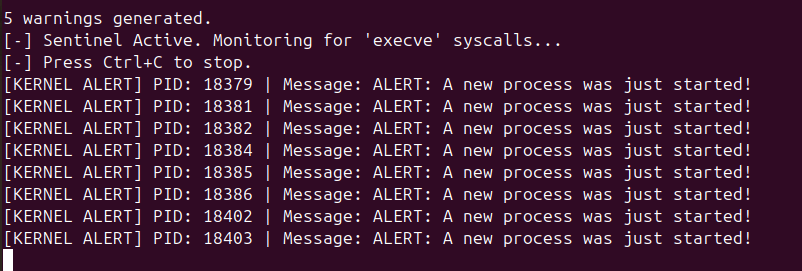


# 3. The Nervous System (eBPF Monitoring)

## 3.1. Kernel Hooking with Python BCC

The core innovation of A.R.E.S. is the use of eBPF to hook the execve system call. Unlike traditional antivirus software that scans files, this approach monitors behavior at the kernel level.

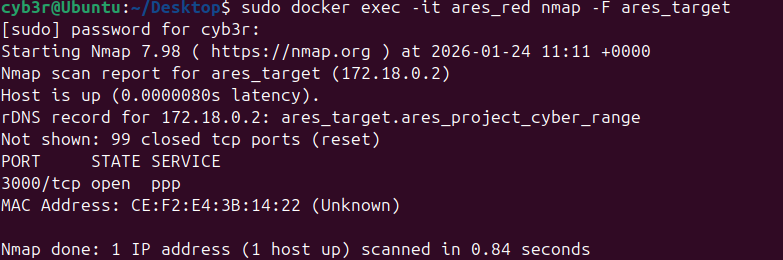
We developed a script named process\_monitor.py which successfully intercepted host system calls from inside the Blue Agent container. This effectively gave the container "ears" to hear every process starting on the machine.

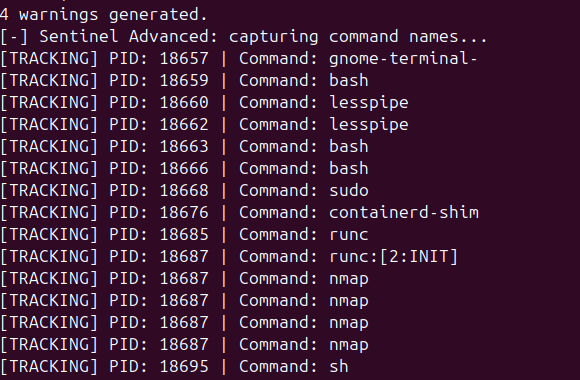


## 3.2. Advanced Forensic Visibility

To identify specific threats, we upgraded the monitor to advanced\_monitor.py using C structures to capture the **Process ID (PID)** and **Command Name (comm)**.

We validated "Cross-Container Forensic Visibility" by launching an Nmap scan from the Red Container. The Blue Agent successfully detected the attack passing through the Host Kernel, logging the specific command used.



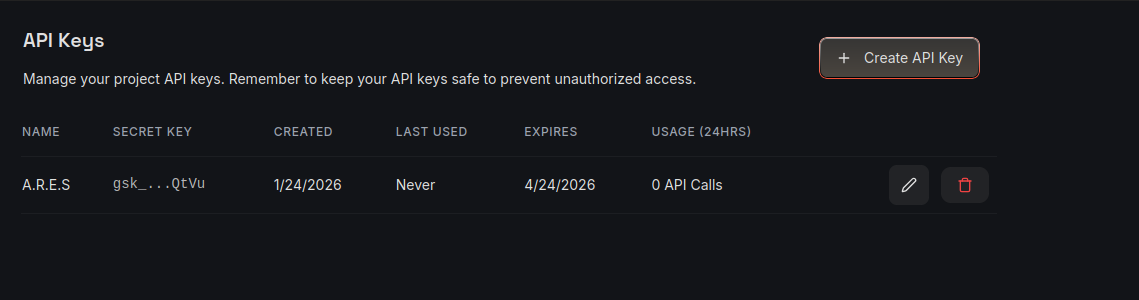


# 4. The Brain (AI Intelligence & Active Defense)

## 4.1. Connecting the Intelligence

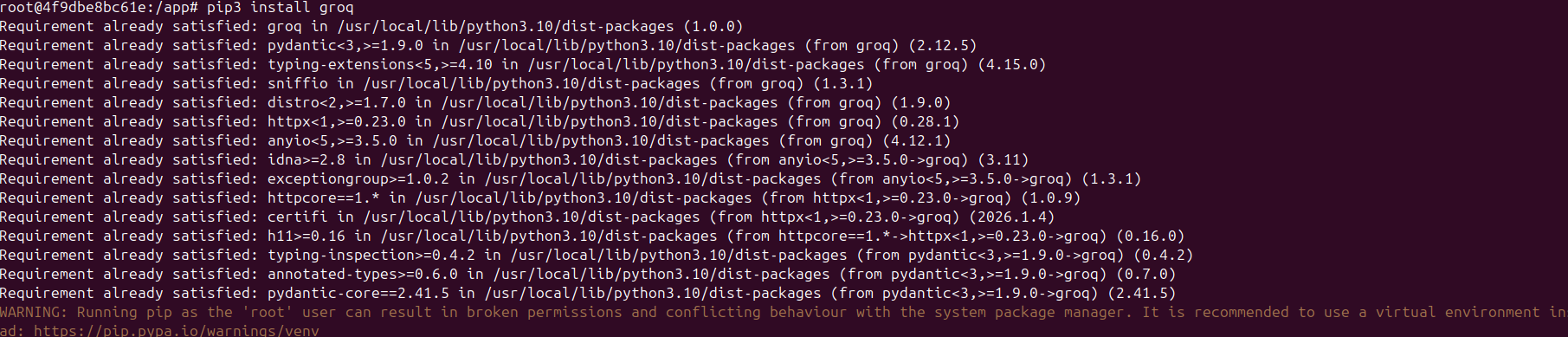
We integrated the **Grok API** to serve as the intelligence engine. The Python script ai\_analyst.py reads the real-time logs and queries the LLM to determine if a command is malicious.

* **Creating an API key on grok**



* **Installing brain library**

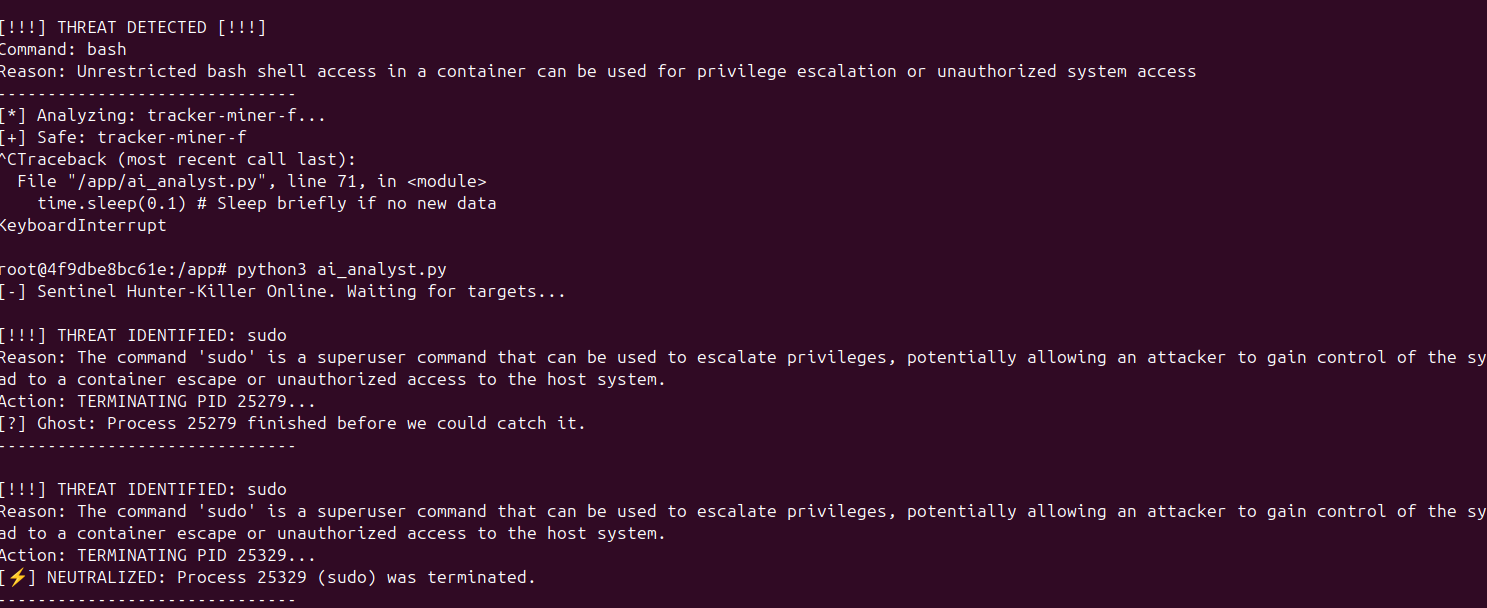
Pip3 install grok



Then make python script of ai\_analyst.py   
and paste inside the API key

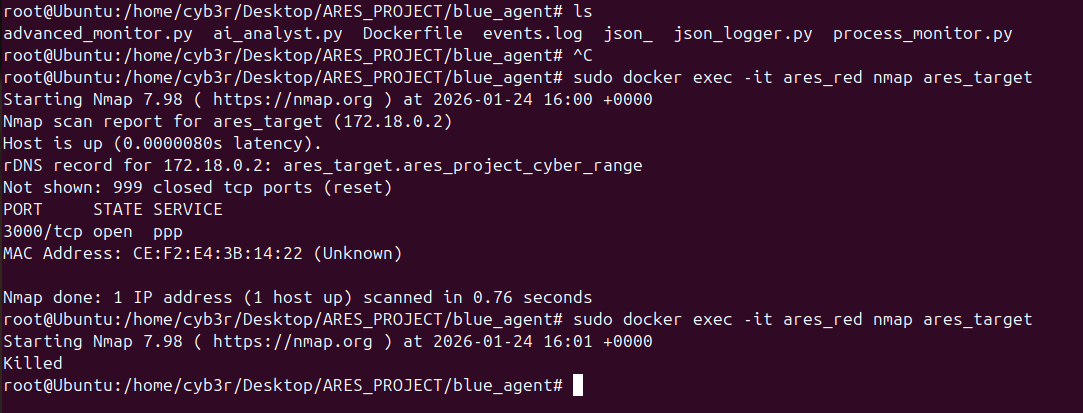
## 4.2. Hunter-Killer Mode (IPS)

The system was upgraded from a passive detector to an active **Intrusion Prevention System (IPS)**. Upon detecting a threat (e.g., sudo usage or unauthorized nmap scanning), the AI Analyst automatically issues a kill signal to the specific PID.



## **4.3. Attack Simulation Results**

During a live test, the Red Agent attempted to scan the target. The A.R.E.S. Blue Agent detected the anomaly and terminated the process instantly. The attacker's terminal showed the message "Killed" immediately after execution.



# 5. The War Room (Visualization)

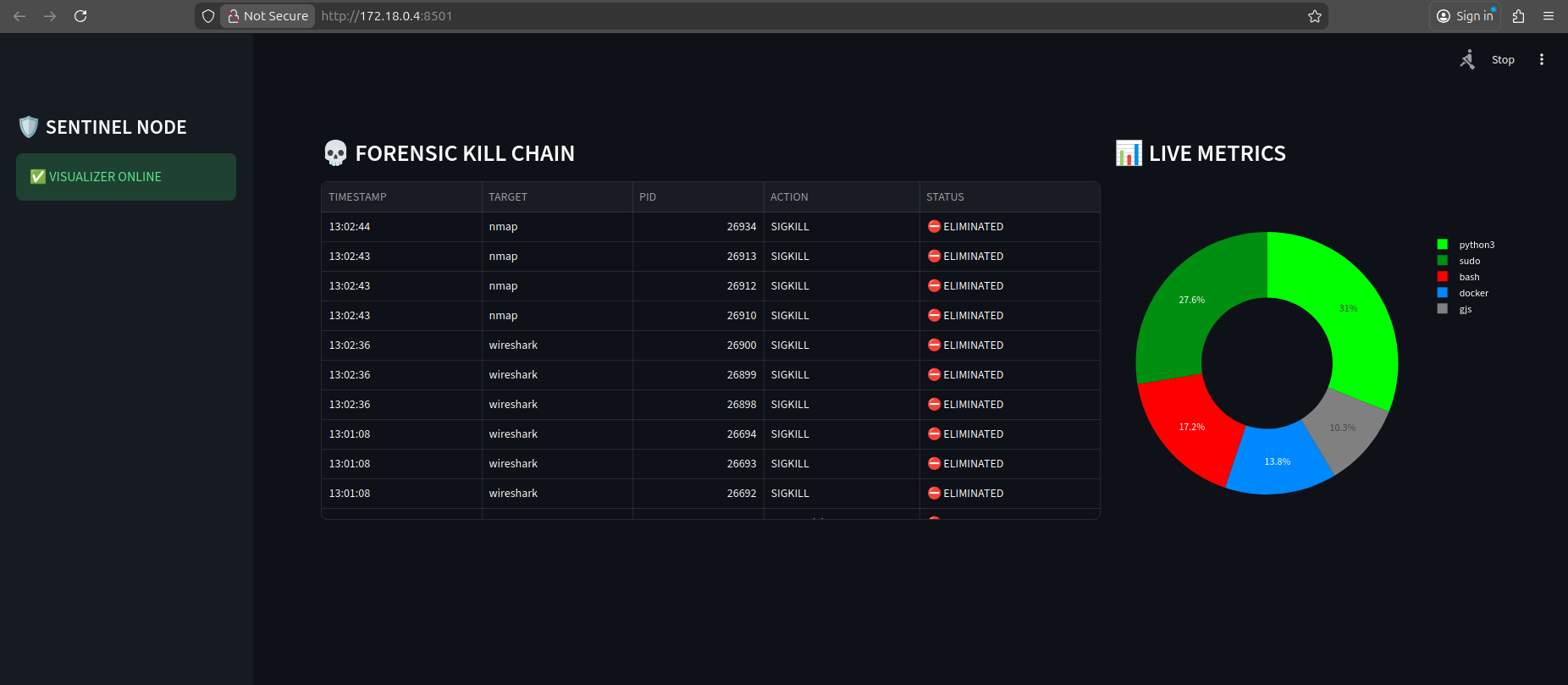
## 5.1. Dashboard Implementation

To visualize the cyber battlefield, we deployed a **Streamlit** dashboard on the host machine. A Python virtual environment was created to manage dependencies and prevent conflicts with the host OS.

## 5.2. Real-Time Telemetry

The dashboard provides a "God’s Eye View" of the system, featuring:

* **Threat Vector Analysis:** A pie chart displaying the distribution of attacks (e.g., Bash infiltration vs. Nmap scanning).
* **Neutralization Log:** A ledger of every malicious process terminated by the AI.
* **System Integrity:** Real-time counters of total events processed and threats neutralized.

****

# 6. Conclusion

Project A.R.E.S. successfully demonstrated the power of combining eBPF observability with AI-driven decision-making. By moving security monitoring into the kernel and automating response via LLMs, we achieved a defense mechanism that is both faster and more intelligent than human operators. The system successfully detected cross-container attacks and neutralized them in milliseconds, fulfilling the primary objective of an autonomous Active Defense system.