Next-Generation Firewall + IDS

Built from Scratch

Nirav Patel

Security Research & Development

Inspired by Real-World Enterprise Security Pipelines

Agenda

- **Introduction:** Problem statement and project motivation.
- **System Architecture:** Overview of the modular design and pipeline.
- **The 9-Stage Pipeline:** Deep dive into the logic and file structure.
- **Operations & Visualization:** Dashboarding and performance monitoring.
- **Conclusion:** Future enhancements.

The Problem: Traditional Firewall Limits

- **Static Rule Dependency:** Relying solely on pre-defined lists misses evolving threats.
- **Detection Gap:** Inability to identify complex behavioral attacks or anomalous patterns.
- **Requirement:** Modern networks need intelligent, adaptive defense mechanisms.

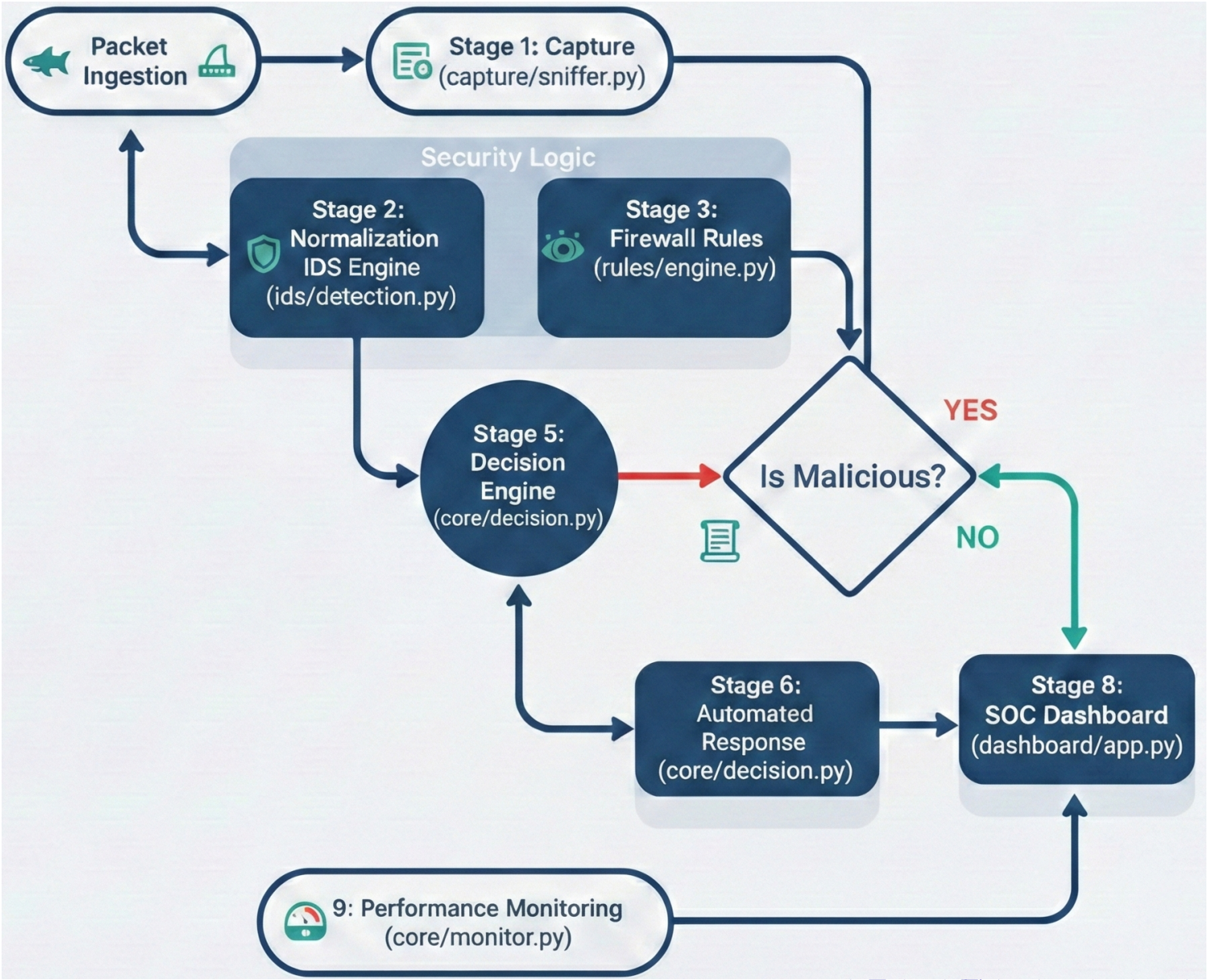
Solution: The 9-Stage Security Pipeline

- **End-to-End Processing:** Flow from packet capture to automated response.
- **Integrated Intelligence:** Combines packet inspection and behavioral detection.
- **Actionable Defense:** Features real-time monitoring and automated blocking.

Project Architecture

- **Modular Design:** Independent layers similar to enterprise-grade systems.
- **Core Components:** Capture Layer, IDS Engine, Rule Engine, and Decision Engine.
- **Operations:** Supported by logging, dashboard, and performance monitoring.

System Architecture Flow



Stage 1 & 2: Capture and Normalization

Stage 1: Packet Capture

- Scapy-based live sniffing using a producer-consumer queue.
- **Access File:** `capture/sniffer_v2.py`

Stage 2: Packet Normalization

- Converts raw binary packets into structured formats (IP, Ports, Protocol).
- **Access File:** `capture/parser.py`

Stage 3 & 4: Policy and Behavior

Stage 3: Firewall Rule Engine

- Static policy enforcement using JSON-stored rules for dynamic updates.
- **Accessed Files:** `rules/engine.py`, `rules/firewall_rules.json`

Stage 4: Intrusion Detection System (IDS)

- Stateful analysis to detect SYN Floods, Port Scans, and Land Attacks.
- **Accessed File:** `ids/detection.py`

Stage 5 & 6: Decision and Logging

Stage 5: Decision Engine

- Acts as the "Correlation Brain" by merging firewall and IDS outputs.
- **Accessed File:** `core/decision.py`

Stage 6: Logging & Forensics

- Records data in multi-format logs to support deep investigation.
- **Accessed Files:** `core/logger.py`, `storage/events.db`

Stage 7 & 8: Automation and SOC

Stage 7: Automated Response

- Executes real-time IP blocking and dynamic rule injection.
- **Accessed File:** `main.py`

Stage 8: SOC Dashboard

- Flask interface providing real-time traffic stats and admin control.
- **Accessed Files:** `dashboard/app.py`

Stage 9: Performance Monitoring

- **Resource Tracking:** Monitors CPU, Memory, and Packet Rate.
- **System Integrity:** Ensures the security overhead does not impact health.
- **Accessed File:** `core/monitor.py`

End-to-End Packet Life Cycle

The Path of a Packet

Capture → Parse → Check → Analyze → Decision → Log → Visualize

- If malicious: Automatically blocked and logged for forensic review.

Future Enhancements

- **AI/ML Integration:** Shifting toward predictive Machine Learning models.
- **Threat Intel:** Integrating global threat feeds for proactive defense.
- **XDR Expansion:** Scaling into distributed sensor networks.

Conclusion

- **Functional Prototype:** A research-grade NGFW ready for deployment testing.

Thank You!