AI-Cerberus

The Three-Headed Guardian Against Malicious Code

Version: 1.0.0 | Platform: Windows | Linux | License: MIT | Python: 3.10+

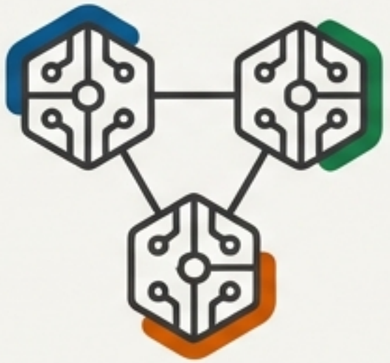
“Guarding your systems from the gates of digital threats”

A Multi-Layered Defense for Modern Threats

AI-Cerberus is an enterprise-grade malware analysis platform. It employs three independent detection engines, powered by artificial intelligence, to provide a comprehensive and resilient defense against malicious code.

Analysis & Formats

- Multi-Format Binary Analysis (PE, ELF, Mach-O)
- Advanced Disassembly (x86, x64, ARM)
- Automated Threat Scoring (0-100 scale)



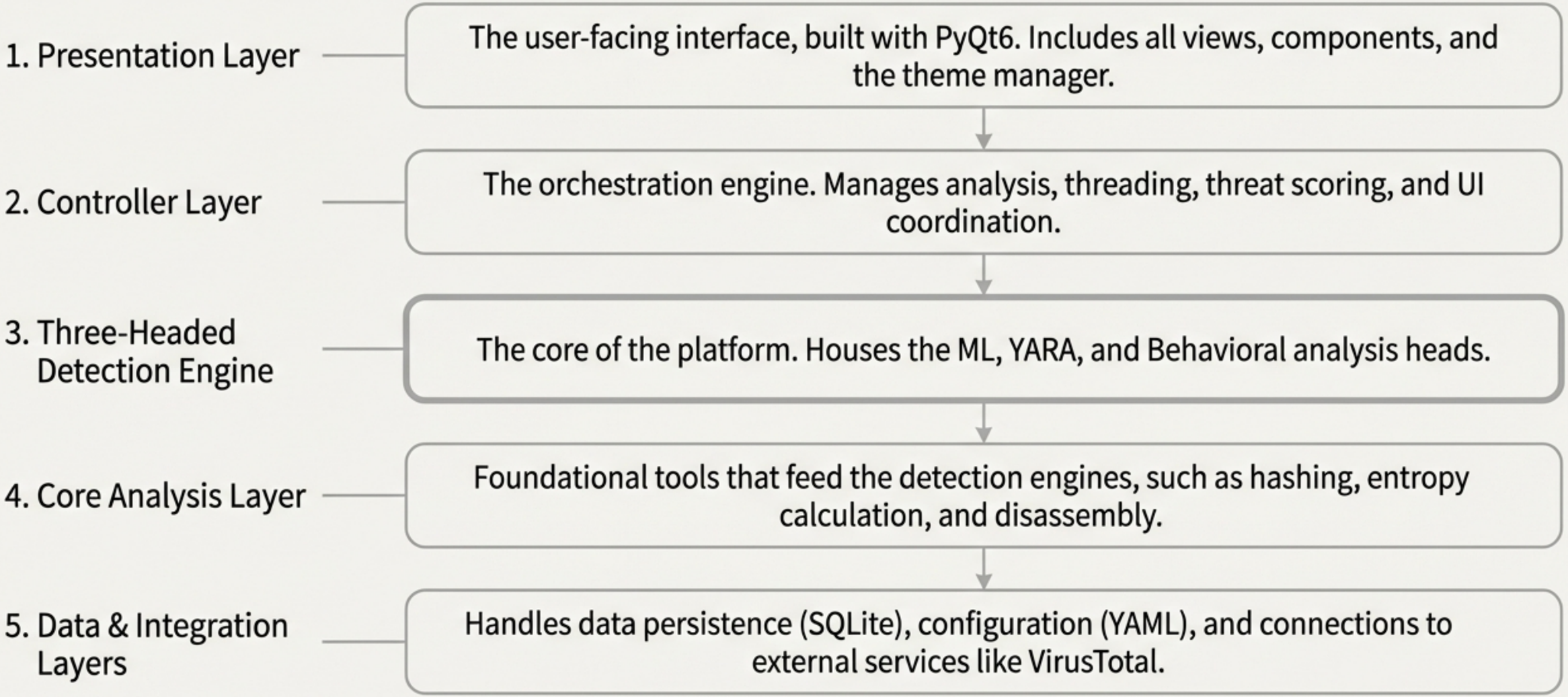
Detection & Engines

- 100+ Machine Learning Features
- Extensible YARA Rule Engine
- 20+ Behavioral Profile Categories

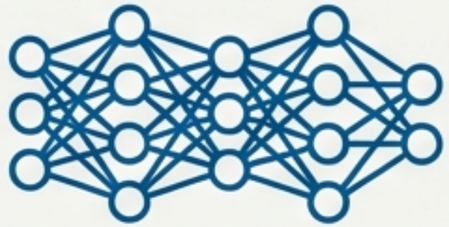
Platform & Integration

- VirusTotal Integration
- Batch Processing & Plugin Architecture
- Professional Dark Theme PyQt6 UI

The Anatomy of the Guardian



Three Independent Heads, One Unified Verdict



The ML Head

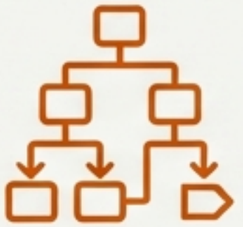
Engine: Machine Learning

Function: Utilizes neural networks and ensemble classifiers (Random Forest, Gradient Boosting) to identify malicious patterns based on over 100 extracted features.

The YARA Head

Engine: Signature Scanning

Function: Employs pattern-based matching with a comprehensive set of YARA rules to detect known malware families, packers, and suspicious code snippets.



The Behavioral Head

Engine: Behavioral Analysis

Function: Analyzes runtime behavior by profiling API calls and system interactions to identify malicious intent even in unknown or obfuscated samples.

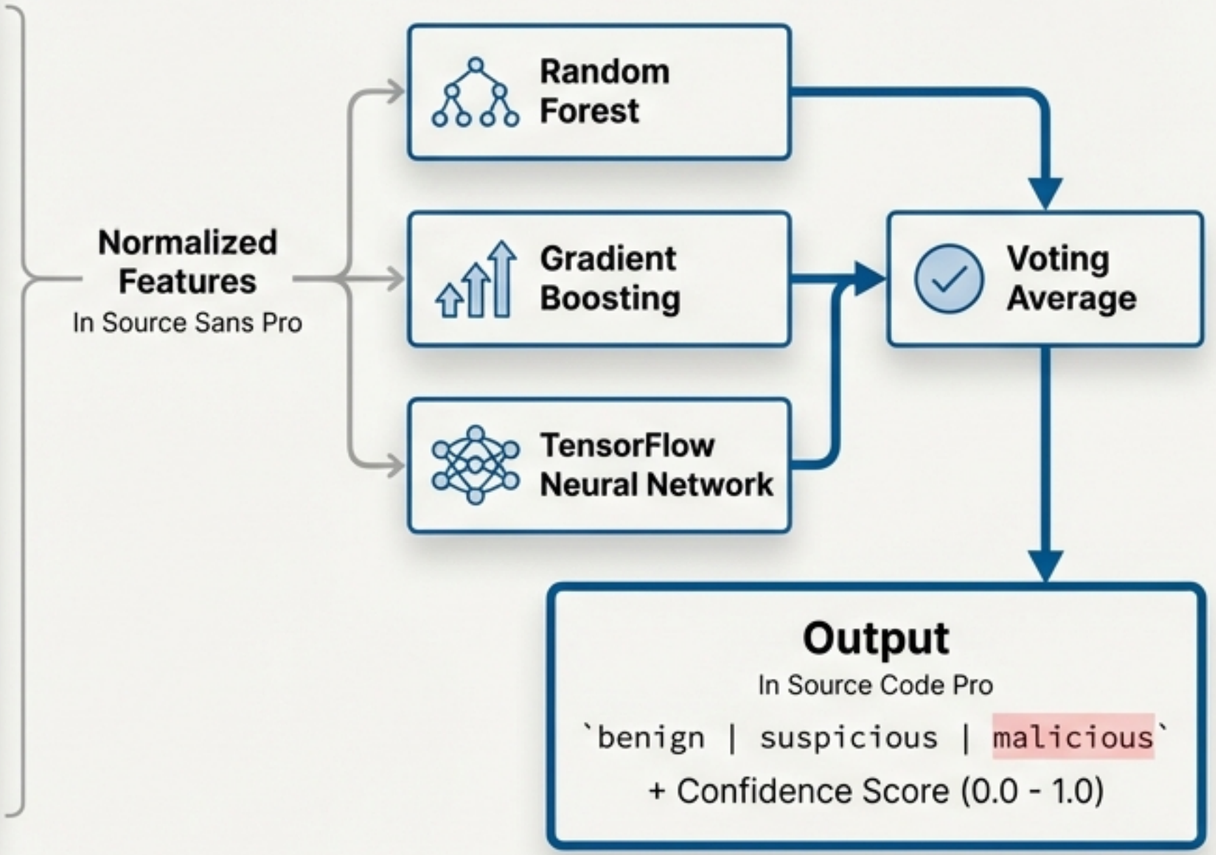
Head 1: The ML Engine's Predictive Power

Phase 1: Comprehensive Feature Extraction

A list of key feature categories extracted from the binary (100+ total):

- **File Metadata** (size, type, timestamps)
- **Entropy** (overall, per-section, variance)
- **PE Headers** (architecture, subsystem, entry point)
- **Imports** (by category: networking, crypto, anti-debug)
- **Strings** (URL/IP counts, registry paths)
- **Byte Statistics & Packer Indicators**

Phase 2: The Ensemble Classifier



Heads 2 & 3: Signature and Behavioral Sentinel

YARA Signature Scanning

Pattern-Based Threat Identification

Scans files against a robust library of YARA rules categorized for precision.

`malware/`
Source Code Pro

Generic malware signatures

`packers/`
Source Code Pro

Packer detection rules

`suspicious/`
Source Code Pro

Suspicious behavior patterns

Behavioral Analysis

Unmasking Malicious Intent

Identifies high-risk actions by analyzing the code's intended API calls.

Process Injection

Persistence Mechanisms

Anti-Debugging / Anti-VM

Network C2 Communication

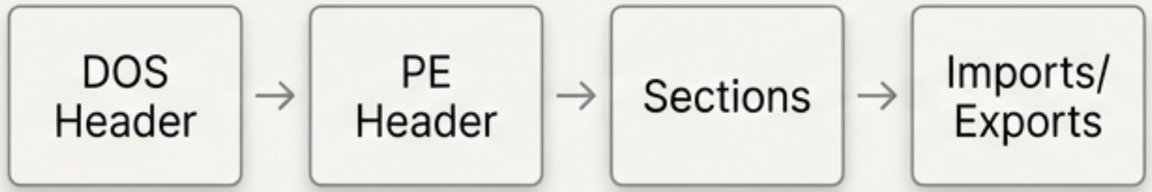
Cryptographic Activity

Privilege Escalation

Deconstructing the Binary: Foundational Analysis

PE File Analysis

Flow Infographic



Anomaly Detection

✓ Executable + Writable sections
✓ Unusual section names (.upx, .themida)
✓ Entry point outside the code section
✓ High entropy (packed/encrypted)

Disassembly Engine

Supported Architectures

x86 x64 ARM ARM64

Suspicious Pattern Detection

- Dangerous API calls
- Self-modifying code
- Direct syscalls and XOR loops

```
.text:60401000  push    ebp
.text:60401001  mov     ebp, esp
.text:60401603  sub     esp, 20h
.text:60401606  push    0                       ; lpProcessAttributes
.text:60401608  push    0                       ; lpThreadAttributes
.text:6040160A  push    0                       ; bInheritHandles
.text:6040160C  push    0                       ; dwCreationFlags
.text:6040160E  push    0                       ; lpEnvironment
.text:60401610  push    0                       ; lpCurrentDirectory
.text:60401612  push    offset lpStartupInfo    ; lpStartupInfo
.text:60401617  push    offset lpProcessInfor ; lpProcessInformation
.text:6040161C  push    offset szCommandLine    ; lpCommandLine
.text:60401621  push    0                       ; lpApplicationName
.text:60401623  call    CreateProcessA          ; Call to CreateProcessA
```

Visual Highlighting: Color-coding indicates risk levels (Critical, High, Medium).

The Cerberus Verdict: A Unified Threat Score

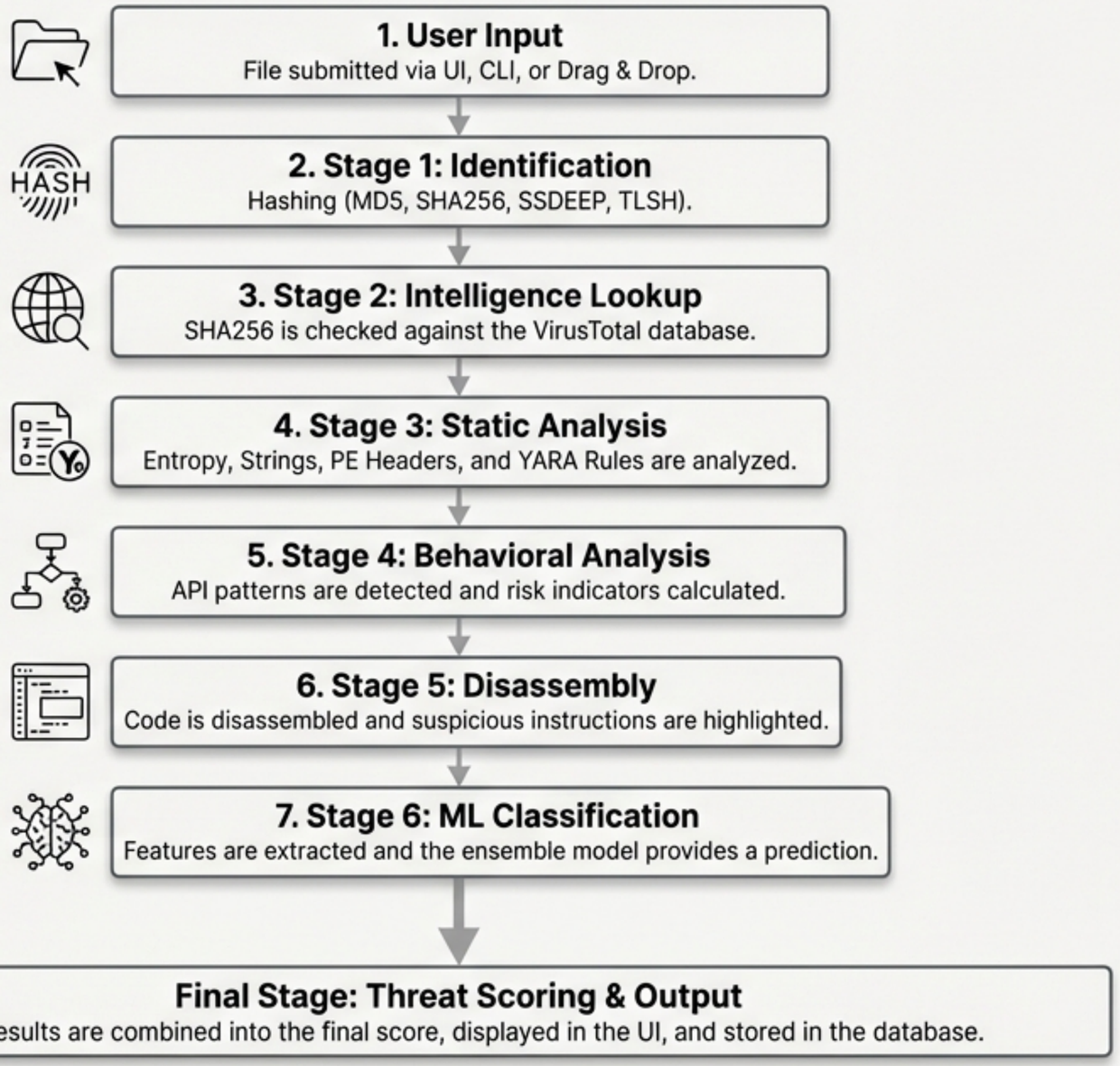
Scoring Components

Component	Max Points	Calculation
Behavioral Indicators	70 points	`behavior_score × 0.7`
YARA Matches	25+ points	`critical=25, high=15, ...`
ML Classification	20 points	`malicious=20 × confidence`
VirusTotal Detections	20 points	`(detections/total) × 20`
Entropy Analysis	15 points	`packed=10, encrypted=15`
Suspicious Strings & PE Anomalies	10 points each	

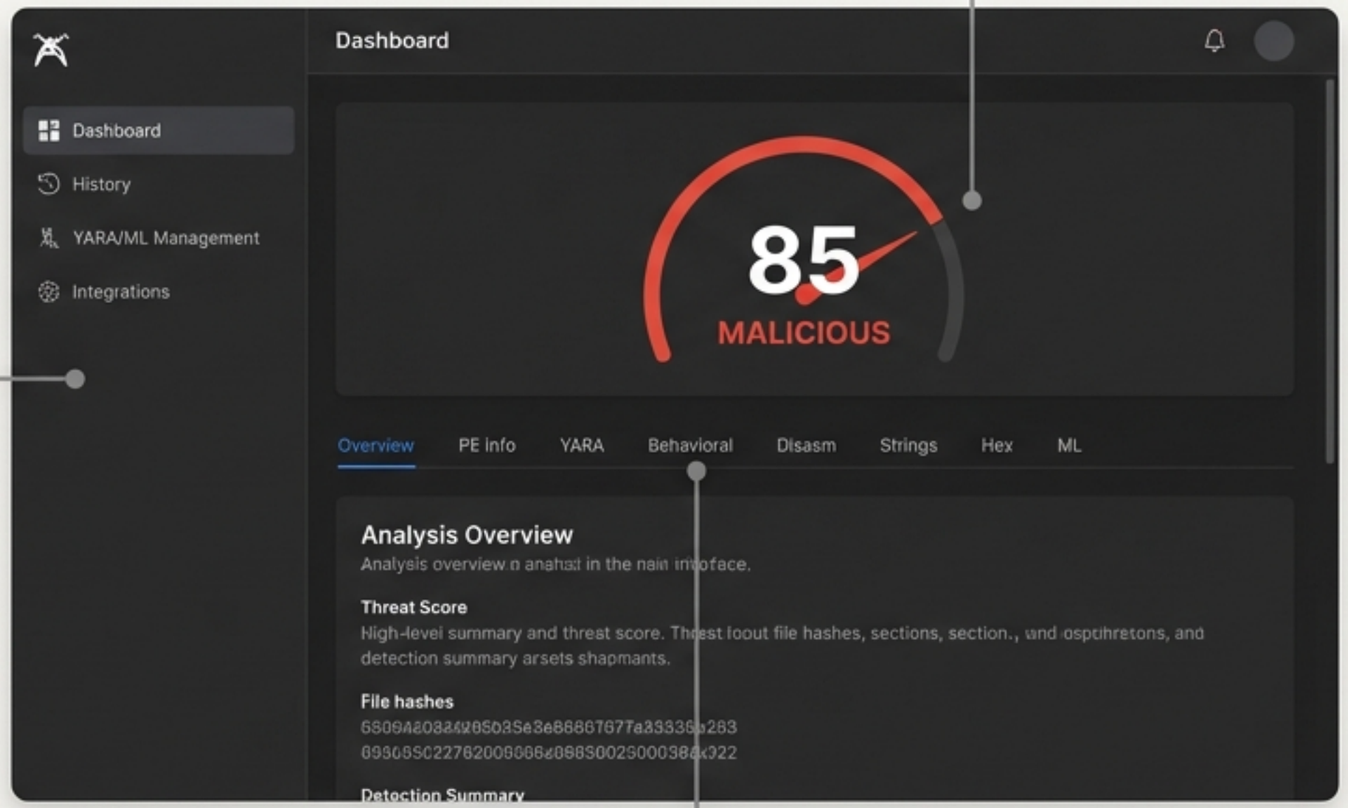
Classification Thresholds

- ≥ 70: MALICIOUS
- 40 - 69: SUSPICIOUS
- < 40: BENIGN

A File's Journey: The 7 Stages of Analysis



The Analyst's Cockpit: The AI-Cerberus UI



Detailed Analysis at Your Fingertips

Overview
High-level summary and threat score.

PE Info
Headers, sections, and imports.

YARA & Behavioral
Matched rules and detected behaviors.

Disasm
Interactive disassembly view.

Strings & Hex
Raw data exploration.

ML
Classification details and confidence score.

Unleash the Guardian: Installation & Quick Start

4-Step Installation

1.**Clone Repository**

```
git clone https://github.com/yourusername/ai-cerberus.git
```

2. **Create Virtual Env**

```
python -m venv venv && source venv/bin/activate
```

3. **Install Dependencies**

```
pip install -r requirements.txt
```

4. **Run Application**

```
python main.py
```

Get Analyzing in Seconds

GUI Launch

`python main.py` and use the file dialog.

Drag & Drop

Simply drag a file onto the application window.

Direct Analysis (CLI)

```
python main.py /path/to/file.exe
```

Batch Analysis (CLI)

```
python main.py --batch /path/to/samples/
```

Taming the Beast: Configuration & Extensibility

Total Control with `config.yaml`

```yaml
# config.yaml

analysis:
  max_file_size: 10MB
  timeout: 300
  max_instructions: 1000000

ml:
  confidence_threshold: 0.85
  model_parameters:
    batch_size: 64
    epochs: 10

integrations:
  virustotal_api_key: "YOUR_API_KEY"
  hybrid_analysis_api_key: "YOUR_API_KEY"

logging:
  level: "INFO"
  format: "%(asctime)s - %(levelname)s - %(m
  log_file_paths:
    - "logs/ai-cerberus.log"
```

Highlighted Configurable Options:

- analysis: `max_file_size`, `timeout`, `max_instructions`
- ml: `confidence_threshold`, model parameters
- `integrations: API keys for VirusTotal, Hybrid Analysis
- `logging: `level`, `format`, log file paths

Extend Capabilities with Plugins

Easily add custom analyzers or integrate new tools.

```python
# plugins/my_plugin.py
from src.plugins import BasePlugin

class MyPlugin(BasePlugin):
    name = "My Custom Plugin"
    description = "Custom analysis plugin"
    def analyze(self, file_path, data, results):
        analyze(self, file_path, data, results)
        # Custom analysis logic
        custom_result = self.custom_analysis(data)
        # cstom plugin logic
        results["my_plugin"] = custom_result
        return results
```

A Developer's Toolkit: Core API Reference

Core Analyzers

```
from src.core import PEAnalyzer,
YaraEngine

# PE analysis
pe = PEAnalyzer()
pe_results = pe.analyze(file_
path, data)
print(pe_results.imports)


# YARA scanning
yara = YaraEngine()
matches = yara.analyze(file_
path, data)
```

ML Classification

```
from src.ml import
MalwareClassifier

classifier = MalwareClassifier()

result = classifier.classify(
file_path, data)
print(result.classification,
result.confidence)
```

Database Operations

```
from src.database import
get_repository

repo = get_repository()

# Query for all malicious
samples
samples =
repo.get_samples_by_classificati
on("malicious")
```

“In Greek mythology, Cerberus guards the gates of the Underworld, preventing the dead from leaving. In cybersecurity, AI-Cerberus guards your systems, preventing malware from entering.”

Project: AI-Cerberus
GitHub: github.com/yourusername/ai-cerberus
Contribution: We welcome contributions! Please see `CONTRIBUTING.md`.
License: MIT