CYBERSENTINEL
AI-BASED MEMORY FORENSICS ANALYZER

FINAL PROJECT REPORT (THESIS EDITION)

Version:	4.0 (Ultimate Edition)
Date:	December 16, 2025
Author:	Development Team
Institution:	Department of Computer Science & Cybersecurity

Generated: 2025-12-16 05:05

FINAL PROJECT REPORT: CYBERSENTINEL

Date: December 16, 2025

Version: 4.0 (U

---

1. ABSTRACT

This comprehensive technical report delineates the end-to-end lifecycle of "CyberSentinel," a state-of-the-art AI-driven memory forensics system. In an era where "fileless" malware and advanced persistent threats (APTs) bypass traditional perimeter defenses, memory forensics has emerged as the definitive frontier for threat detection. However, the analysis of volatile memory (RAM) remains an esoteric discipline, requiring manual inspection of hexadecimal dumps and kernel structures-a process that is notoriously time-consuming and prone to human error.

CyberSentinel bridges this critical gap by automating the forensic analysis pipeline. Leveraging a dataset of 58,596 memory artifacts extracted via the Volatility Framework (CIC-MalMem-2022), the system employs a sophisticated Ensemble Super Learner architecture. This architecture integrates Random Forest (for high-variance reduction), Logistic Regression (for linear separability), and Multi-Layer Perceptrons (for capturing non-linear feature interactions). The resulting model achieves a statistically validated accuracy of 99.9915%, with a False Positive Rate (FPR) approaching zero.

Beyond binary classification, the system introduces a specialized multi-class taxonomic engine capable of distinguishing between Ransomware (e.g., WannaCry, Shade), Spyware (e.g., No.More.Ransom), and Trojans (e.g., Emotet) with precision. To address the perennial challenge of zero-day exploits, an unsupervised Isolation Forest module provides anomaly detection capabilities, flagging statistical outliers that deviate from benign memory patterns.

All analytical outputs are synthesized into a "Cyberpunk-aesthetic" User Interface (UI), built on Streamlit. This UI is not merely cosmetic; it is designed to reduce cognitive load for forensic analysts through high-contrast visualizations, interactive 3D Principle Component Analysis (PCA) clusters, and immediate visual feedback loops. This report documents every phase of development, from mathematical model formulation to UI/UX design principles, ensuring a reproducible and rigorously validated scientific contribution.

---

2. CHAPTER 1: INTRODUCTION AND THEORETICAL FRAMEWORK

1.1 Background and Motivation

The paradigm of cyber warfare has shifted. Adversaries no longer rely on dropping executable files (`.exe`) onto a disk where they can be scanned by traditional Antivirus (AV) engines. Instead, they exploit "Living off the Land" (LotL) tactics, injecting malicious code directly into the memory space of legitimate processes (e.g., `svchost.exe`, `explorer.exe`). This technique, known as "fileless malware," leaves no physical footprint on the hard drive.

Consequently, Memory Forensics-the act of capturing and analyzing the volatile state of a computer's RAM-has become indispensable. It allows investigators to view running processes, active network connections, and decrypted keys that are otherwise invisible. However, the barriers to entry are high:

1. Complexity

1.2 Problem Statement

Despite the availability of tools like Volatility and Rekall, the interpretation of their output remains manual. An analyst must run plugins like `malfind` or `pslist` and subjectively evaluate thousands of lines of output.

- Scalability Issue: A Security Operations Center (SOC) cannot manually analyze 10,000 endpoints.
- Expertise Gap: There is a global shortage of analysts capable of kernel-level debugging.
- Latency: The "Time to Detect" (TTD) for memory-resident threat often exceeds 200 days.

1.3 Research Objectives

The primary objective is to construct an autonomous system that reduces TTD from days to seconds.

1. Automated
cleaning.

1.4 Project Scope and Limitations

- Scope: The system focuses on post-acquisition analysis. It assumes a memory dump has already been acquired and processed into feature vectors.
- Operating System: The current model is trained on Windows 7/10 kernel structures (the most prevalent targets).
- Limitation: The system categorizes known malware families. Entirely new families with unique behaviors may be flagged only as "Anomalies" rather than specific classes.

---

3. CHAPTER 2: COMPREHENSIVE LITERATURE REVIEW

2.1 The Evolution of Memory Forensics

Memory forensics began as crude "string searching" in crash dumps.

- 2005: DFRWS Challenge: Testing the community's ability to extract artifacts.
- 2007: The Volatility Framework: Walters et al. introduced the concept of traversing the kernel's doubly-linked lists (e.g., `ActiveProcessLinks` in `EPROCESS` structures) to map the system state.
- 2014: "The Art of Memory Forensics": Ligh et al. codified specific heuristics for malware detection, such as checking for `RWX` (Read-Write-Execute) permissions in private memory, a hallmark of code injection.

2.2 Machine Learning in Cybersecurity

The application of ML to security is well-documented but often siloed.

- Schultz et al. (2001): Pioneered data-mining for executable analysis.
- Moskovitch et al. (2008): Applied Decision Trees to worm detection.
- Recent Transformer Models: While BERT and LLMs are gaining traction, they are computationally expensive for tabular data. For tabular forensic data, Tree-based ensembles (XGBoost, Random Forest) remain the State of the Art (SOTA) due to their interpretability and speed.

---

4. CHAPTER 3: SYSTEM ANALYSIS AND ARCHITECTURE

3.1 High-Level Architecture

The system follows a modular "Microservices-like" architecture, ensuring maintainability and scalability.

[FIGURE: SYSTEM ARCHITECTURE DIAGRAM]

1. Data Layer: Consists of raw CSV files (`malmem.csv`) and serialized model artifacts (`.pkl`).

2. Processin
Load) operatio

3.2 Detailed Data Flow

Detailed flow of a single analysis session:

1. Upload: Us

- Tier 1: Ensemble Classifier predicts Benign/Malware.
- Tier 2: If Malware, MLP Multiclass classifier predicts Family.
- Tier 3: Isolation Forest calculates Anomaly Score.

5. Rendering:

3.3 Hardware and Software Specifications

To ensure reproducibility, the development environment is specified below:

Component	Specification	Purpose
CPU	Intel Core i9-13900K	High-throughput data processing
GPU	NVIDIA RTX 4090 (Optional)	Accelerated Neural Network training
RAM	64GB DDR5	Handling large in-memory datasets
Language	Python 3.11.4	Core logic implementation
Framework	Streamlit 1.35.0	Web interface generation
ML Engine	Scikit-Learn 1.4.2	Model training and inference

---

5. CHAPTER 4: METHODOLOGY AND ALGORITHMIC DESIGN

4.1 Dataset Analysis (CIC-MalMem-2022)

The dataset is a curated collection of memory dumps.

- Total Instances: 58,596
- Dimensionality: 57 Columns (55 features, 2 labels)
- Class Balance: Perfectly balanced (50% Benign / 50% Malware)
- Families Included:
- Spyware: 180solutions, Coolwebsearch, Gator, Transponder, CWS.
- Ransomware: Ako, Shade, Conti, Maze, Petya.
- Trojan: Zeus, Emotet, Refleve, Scar, Reconyc.

4.2 Feature Engineering: The Volatility Plugins

We utilize features derived from specific Volatility plugins. Understanding these features is key to understanding the model's power.

- `pslist` (Process List): Inspects the active process list.
- `nproc`: Number of running processes. Malware often tries to hide, or conversely, spawn many workers.
- `avg_threads`: Malware often uses specific threading models for Command & Control (C2) beacons.
- `dlllist` (DLL List): Tracks loaded libraries.
- `avg_dlls_per_proc`: A process with very few DLLs might be a hollowed process (unpacked malware).
- `malfind` (Malware Finder): The most critical plugin.
- `ninjections`: Counts memory segments with `PAGE_EXECUTE_READWRITE` protection that are not backed by a file on disk. This is the "smoking gun" of code injection.
- `commitCharge`: Examples of malware allocating private memory for unpacking payloads.
- `svcscan` (Service Scan): Malware often installs itself as a service for persistence.

4.3 Algorithmic Formulations

4.3.1 Random Forest

A bagging ensemble of Decision Trees.

- Split Criterion: Gini Impurity.

Formula: $Gini(D) = 1 - \sum_{i=1}^{C} (p_i)^2$

Where p_i is the probability of an item belonging to class i.

- Advantage: Immune to feature scaling, handles non-linear data well.

4.3.2 Logistic Regression

Used for its high interpretability and calibration.

- Sigmoid Function:

Formula: $\sigma(z) = \frac{1}{1 + e^{-z}}$

Where $z = w^T x + b$. This outputs a probability between 0 and 1.

4.3.3 Multi-Layer Perceptron (MLP)

A feed-forward artificial neural network.

- Architecture: Input Layer (55 neurons) -> Hidden Layer 1 (100 neurons, ReLU) -> Hidden Layer 2 (50 neurons, ReLU) -> Output Layer (1 neuron, Sigmoid).
- Optimization: Adam Optimizer (Adaptive Moment Estimation).

Formula: $w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$

4.3.4 Isolation Forest (Anomaly Detection)

Instead of profiling normal data, it explicitly isolates anomalies.

- Principle: Anomalies are "few and different," making them easier to isolate in random partitions of the feature space.
- Path Length: Calculating the average path length $E(h(x))$ in the trees. Shorter paths indicate anomalies.

---

6. CHAPTER 5: RESULTS AND EMPIRICAL EVALUATION

5.1 Experimental Setup

- Train/Test Split: 80% Training (46,876 samples), 20% Testing (11,720 samples).
- Cross-Validation: 5-Fold Stratified K-Fold to ensure no overfitting.

5.2 Quantitative Results

5.2.1 Binary Classification (Benign vs. Malware)

The Ensemble model was evaluated against base estimators.

Model	Accuracy	Precision	Recall	F1-Score	Processing Time
**Logistic Regressic	99.88%	99.87%	99.89%	99.88%	2.1s
**Decision Tree*	99.98%	99.98%	99.98%	99.98%	1.5s
**Random Forest	99.99%	100.00%	100.00%	100.00%	14.2s
**MLP (Neural Net	99.99%	100.00%	100.00%	100.00%	45.3s
**Ensemble Votinç	**99.99%**	**100.00%**	**100.00%**	**100.00%**	**62.0s**

5.2.2 Multi-Class Classification (Family Detection)

The system demonstrated equal proficiency in distinguishing malware types.

Family	Precision	Recall	F1-Score	Support
Ransomware	100.00%	99.95%	99.97%	1,954
Spyware	99.95%	100.00%	99.97%	1,953
Trojan	100.00%	100.00%	100.00%	1,953

Note: The support indicates a perfectly balanced test set, validating the dataset curation.

5.3 Confusion Matrix Analysis

A visual representation of the prediction errors.

In the test set

- True Negatives (TN): 5,859
- True Positives (TP): 5,860
- False Positives (FP): 1
- False Negatives (FN): 0

[FIGURE: CONFUSION MATRIX HEATMAP]

The single False Positive (a Benign sample flagged as Malware) was investigated and found to be a process with an unusually high number of injected thread handles, mimicking behavior of the "Zeus" trojan. This suggests the model is highly sensitive to injection-like patterns.

5.4 Feature Importance Analysis

Understanding the "Black Box" is crucial for forensics. We utilized Feature Permutation Importance.

[FIGURE: FEATURE IMPORTANCE CHART]

- Ranking:

1. `malfind.co
to run.

---

7. CHAPTER 6: USER INTERFACE (UI) AND EXPERIENCE (UX)

6.1 Design Philosophy: "Cyberpunk Glitch"

The choice of a "Cyberpunk" aesthetic serves a functional purpose beyond visual appeal.

1. High Cont
(`#00F0FF`) d

6.2 Key UI Components

- System Status Heartbeat: A pulsing indicator that provides immediate visual confirmation of system health.
- 3D PCA Cluster Map: An interactive Plotly chart allowing users to rotate and zoom into the 3-dimensional memory space, visually identifying how malware clusters separate from benign processes.
- Glitch Typography: CSS animations on headers (`@keyframes glitch`) reinforce the high-tech theme.

[FIGURE: DASHBOARD SCREENSHOT]

---

8. CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

The CyberSentinel project stands as a testament to the efficacy of AI in low-level forensics. By automating the extraction and classification of volatile memory artifacts, we have effectively eliminated the "Analysis Bottleneck." The system's near-perfect accuracy, combined with its ability to detect zero-day anomalies and visualize complex data, makes it a formidable tool in the modern SOC arsenal.

7.2 Recommendations for Deployment

- Infrastructure: Deploy on a secure, air-gapped analysis network to prevent malware escape.
- Calibration: Periodically re-train the Anomaly Detection module on "GOLD" images (known benign baselines) of the organization's specific standard operating environment (SOE).

7.3 Future Work

1. Adversarial Robustness: Research into "Evasion Attacks" where malware modifies its memory footprint to fool the AI.

2. Live Memo
capture and a

---

9. APPENDICES

Appendix A: System Configuration Files

The system utilizes a `requirements.txt` for dependency management. Crucially, `scikit-learn` version consistency is enforced to ensure model serialization compatibility.

```
streamlit>=1.28.0
pandas>=2.1.0
scikit-learn>=1.3.0
plotly>=5.17.0
fpdf2>=2.7.5
joblib>=1.3.0
shap>=0.42.1
```

Appendix B: Detailed Model Hyperparameters

- Random Forest: `n_estimators=100`, `criterion='gini'`, `max_depth=None`, `bootstrap=True`.
- MLP: `hidden_layer_sizes=(100, 50)`, `activation='relu'`, `solver='adam'`, `alpha=0.0001`.
- Isolation Forest: `n_estimators=100`, `contamination=0.01` (tuned for low false positives).