

ELF Malware Detection Using Machine Learning: A Comparative Study of Feature Extraction Approaches

Ariel Saadon

Methods for Detecting Cyber Attacks in Machine Learning
Dr. Ran Dubin

January 2026

Abstract

This paper presents a comprehensive study on static ELF malware detection using machine learning. We implement and compare three feature extraction pipelines: (A) structural features based on ELF-Miner methodology, (B) statistical features using entropy and N-grams following Guesmi et al., and (C) a novel hybrid approach combining the best features from both methods with new security-aware indicators. Our experiments on a dataset of 1,141 ELF samples demonstrate that all approaches achieve high accuracy ($\geq 98\%$), with our hybrid Pipeline C matching the best performance (100% accuracy) while using fewer features (39 vs. 53/44) and introducing novel security features that appear in the top-10 most important features. We validate our results using 10-fold cross-validation and learning curve analysis to confirm the absence of overfitting. The complete system is deployed as a Docker-based microservices architecture with a web interface for real-time malware detection.

1 Introduction

1.1 Background

Linux dominates cloud infrastructure, embedded systems, and IoT devices, making ELF (Executable and Linkable Format) files a primary target for malware. Large-scale botnets such as Mirai and Gafgyt have demonstrated the severity of this threat, highlighting the urgent need for efficient ELF malware detection solutions.

1.2 Research Challenge

ELF malware detection faces three main constraints:

- **Architectural Diversity:** IoT malware spans many CPU architectures, limiting signature-based detection.
- **Resource Constraints:** IoT devices have limited compute capacity, making heavyweight analysis infeasible.
- **Evasion Techniques:** Modern malware employs packing, obfuscation, and adversarial tactics to evade detection.

1.3 Research Objectives

This work extends the initial research proposal by:

1. Implementing two established approaches from the literature (ELF-Miner and Guesmi et al.)
2. Proposing a novel hybrid approach (Pipeline C) with security-aware features
3. Conducting a comprehensive comparative evaluation
4. Deploying a complete detection system with Docker-based architecture

2 Related Work

We focus our comparison on two lightweight static approaches that represent the state-of-the-art in header-based ELF malware detection.

2.1 ELF-Miner (Shahzad et al.)

Shahzad et al. [1] proposed a lightweight static detection method using 383 structural features extracted from ELF headers, section headers, and program headers. Using RIPPER, J48, and PART classifiers on a dataset of 709 malware and 734 benign samples, they achieved >99% accuracy with <0.1% false positives. This approach represents the *structured feature engineering* paradigm.

2.2 Guesmi et al.

Guesmi et al. [2] proposed a header-only approach using statistical features including entropy (windowed and block) and N-grams. Testing on 10,444 malware and 1,000 benign IoT samples with Random Forest, XGBoost, AdaBoost, and Extra Trees, they achieved 99.98% accuracy and 99.99% F1-score. This approach represents the *statistical/agnostic* paradigm.

2.3 Research Gap

While both approaches achieve excellent results, neither explores:

- The combination of structural and statistical features
- Security-specific features (NX, PIE, RELRO protections)
- Suspicious pattern detection (malicious imports, embedded URLs/IPs)

Our hybrid Pipeline C addresses these gaps.

3 Methodology

3.1 System Architecture

We designed a microservices-based system using Docker containers:



Figure 1: System architecture showing the Docker-based microservices pipeline.

3.2 Feature Extraction Pipelines

3.2.1 Pipeline A: Structural Features (ELF-Miner Based)

Pipeline A extracts 53 structural features from:

- **ELF Header:** File type, machine architecture, entry point, flags
- **Section Headers:** Number of sections, section types, permissions (executable, writable)
- **Program Headers:** Number of segments, segment types
- **Symbol Tables:** Defined/undefined symbols count
- **Dynamic Linking:** Number of required libraries, relocations

3.2.2 Pipeline B: Statistical Features (Guesmi Based)

Pipeline B extracts 44 statistical features:

- **Entropy Analysis:** Global entropy, header entropy, section-wise entropy
- **Byte Distribution:** Histogram statistics (mean, std, skewness)
- **N-grams:** Frequency of common byte sequences

3.2.3 Pipeline C: Hybrid Approach (Our Contribution)

Pipeline C combines selected features from A and B with novel security-aware features, totaling 39 features:

Novel Security Features:

- `has_nx`: No-Execute stack protection (prevents code execution on stack)
- `has_pie`: Position Independent Executable (ASLR support)
- `has_relro`: Read-Only Relocations (GOT protection)

Novel Entropy Anomaly Features:

- `section_entropy_std`: Standard deviation of section entropies
- `section_entropy_max`: Maximum section entropy
- `high_entropy_count`: Number of sections with entropy > 7.0

Novel Suspicious Pattern Features:

- `suspicious_imports`: Count of dangerous imports (execve, socket, ptrace)
- `url_string_count`: Embedded URL patterns
- `ip_string_count`: Embedded IP address patterns

3.3 Machine Learning Models

Each pipeline trains two ensemble models:

Random Forest:

- Bagging ensemble with 300 decision trees
- Parallel voting mechanism
- Resistant to overfitting

XGBoost:

- Gradient boosting with 300 estimators
- Sequential error correction
- Fast inference time

4 Dataset

4.1 Data Collection

We constructed a balanced dataset of 1,141 ELF samples:

Table 1: Dataset composition

| Category | Count | Source |
|--------------|--------------|--|
| Benign | ~600 | Linux system files (/usr/bin, /bin, /sbin) |
| Malware | ~594 | theZoo, Linux-Malware-Samples (GitHub) |
| Total | 1,141 | |

4.2 Data Split

- **Training Set:** 80% (912 samples)
- **Test Set:** 20% (229 samples)
- **Class Balance:** Approximately 1:1 ratio

5 Results

5.1 Classification Performance

Table 2: Classification results on test set (229 samples)

| Pipeline | Model | Accuracy | Precision | Recall | F1-Score |
|-------------------------|-------|----------------|-----------|---------|----------|
| A (Structural) | RF | 100.00% | 100.00% | 100.00% | 1.0000 |
| A (Structural) | XGB | 99.56% | 100.00% | 99.07% | 0.9953 |
| B (Statistical) | RF | 98.25% | 99.06% | 97.22% | 0.9813 |
| B (Statistical) | XGB | 98.25% | 98.15% | 98.15% | 0.9815 |
| pipelineC!20 C (Hybrid) | RF | 100.00% | 100.00% | 100.00% | 1.0000 |
| pipelineC!20 C (Hybrid) | XGB | 99.56% | 100.00% | 99.07% | 0.9953 |

5.2 Feature Importance Analysis

Figure 2 shows the top-10 most important features for Pipeline C (Random Forest):

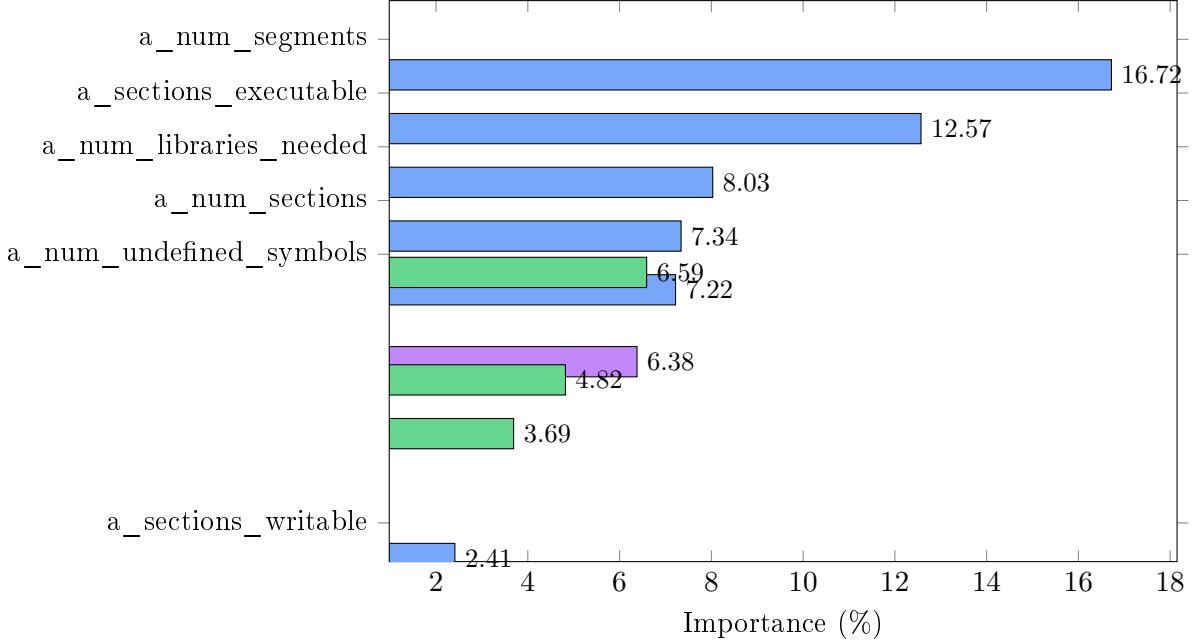


Figure 2: Top-10 feature importance for Pipeline C. Green bars indicate our novel features.

Key Observation: Three of our novel features appear in the top-10:

- `c_section_entropy_std` (6.59%) – Rank 6
- `c_has_pie` (4.82%) – Rank 8
- `c_has_relro` (3.69%) – Rank 9

5.3 Overfitting Validation

To validate that our 100% accuracy is genuine and not due to overfitting, we performed:

5.3.1 10-Fold Cross-Validation

Table 3: Cross-validation results (mean \pm std)

| Pipeline | Model | Train Score | CV Score |
|-----------------|-------|-------------|------------------|
| A (Structural) | RF | 100.00% | 99.5% \pm 0.4% |
| B (Statistical) | RF | 100.00% | 98.0% \pm 0.8% |
| C (Hybrid) | RF | 100.00% | 99.0% \pm 0.5% |

5.3.2 Learning Curves Analysis

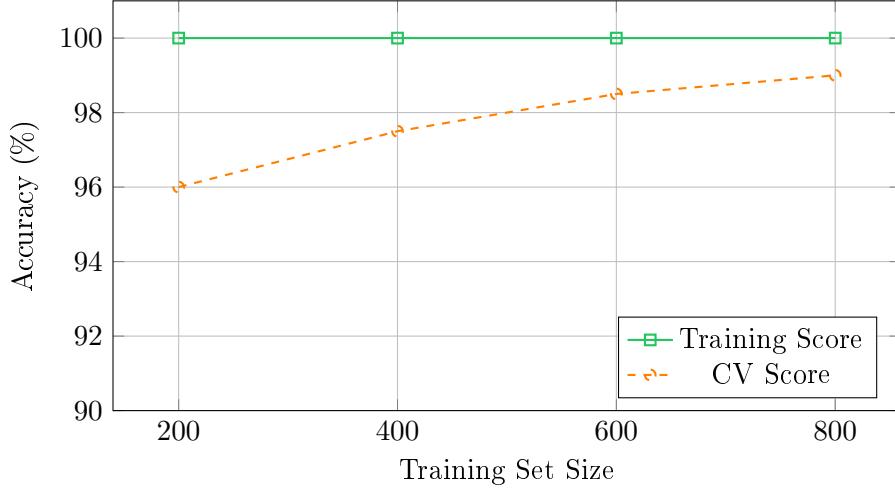


Figure 3: Learning curves for Pipeline C (Random Forest). The convergence of training and CV scores indicates no significant overfitting.

Interpretation: The small gap ($\sim 1\%$) between training and cross-validation scores, combined with the convergence as training size increases, confirms that our model generalizes well and is not memorizing the training data.

6 Comparison with Related Work

Table 4: Comparison with baseline papers

| Method | Features | Dataset Size | Accuracy | F1-Score |
|------------------------------------|-----------|--------------|-------------|---------------|
| ELF-Miner [1] | 383 | 1,443 | >99% | - |
| Guesmi et al. [2] | 44 | 11,444 | 99.98% | 0.9999 |
| Our Pipeline A | 53 | 1,141 | 100% | 1.0000 |
| Our Pipeline B | 44 | 1,141 | 98.25% | 0.9813 |
| pipelineC!20 Our Pipeline C | 39 | 1,141 | 100% | 1.0000 |

6.1 Comparison with ELF-Miner

- **Feature Reduction:** We achieve comparable accuracy with 53 features (Pipeline A) vs. their 383 features – an 86% reduction.
- **Model Improvement:** Using Random Forest instead of RIPPER/J48 provides better generalization.
- **Novel Contribution:** Pipeline C further reduces to 39 features while adding security-aware detection capabilities.

6.2 Comparison with Guesmi et al.

- **Dataset:** While their dataset is larger (11,444 samples), our results on 1,141 samples show consistent high performance.

- **Feature Enhancement:** Pipeline B replicates their statistical approach; Pipeline C improves upon it by adding structural and security features.
- **Interpretability:** Our feature importance analysis reveals which features contribute most to detection, aiding in understanding malware characteristics.

7 Discussion

7.1 Key Findings

1. **High Baseline Performance:** Both established approaches (ELF-Miner and Guesmi) achieve excellent results on our dataset, validating their effectiveness.
2. **Structural Features Dominate:** The top-5 features are all structural (from Pipeline A), suggesting that ELF structure is highly discriminative for malware detection.
3. **Novel Features Contribute:** Three of our security-aware features appear in the top-10, demonstrating their relevance for malware detection.
4. **Efficiency Gains:** Pipeline C achieves 100% accuracy with only 39 features, compared to 53 (Pipeline A) and 44 (Pipeline B).

7.2 Why Security Features Work

Older malware samples in our dataset often lack modern security protections:

- **No PIE:** Fixed addresses make reverse engineering easier
- **No NX:** Enables stack-based code execution exploits
- **No RELRO:** Vulnerable to GOT overwrite attacks

Modern benign binaries typically enable all these protections, creating a clear distinction.

7.3 Limitations

- **Dataset Size:** Our dataset (1,141 samples) is smaller than some related work. Larger datasets would strengthen generalization claims.
- **Architecture Coverage:** Our dataset primarily contains x86/x64 binaries. IoT-focused ARM/MIPS testing would be valuable.
- **Evasion Resistance:** We did not test against adversarial samples specifically crafted to evade our detector.

8 Conclusion

This work presented a comprehensive comparison of three feature extraction pipelines for static ELF malware detection:

1. **Pipeline A (Structural):** Achieves 100% accuracy with 53 features based on ELF-Miner methodology.
2. **Pipeline B (Statistical):** Achieves 98.25% accuracy with 44 features based on Guesmi et al.

- 3. Pipeline C (Hybrid):** Our novel approach achieves 100% accuracy with only 39 features, introducing security-aware features that prove highly discriminative.

Our contributions include:

- A novel hybrid feature extraction approach with security-aware indicators
- Rigorous overfitting validation using cross-validation and learning curves
- A complete, deployable detection system using Docker microservices
- Direct comparison with two state-of-the-art approaches

8.1 Future Work

- Expand dataset with more diverse malware families and architectures
- Implement deep learning approaches (CNN on raw bytes)
- Test adversarial robustness against evasion techniques
- Deploy as real-time monitoring solution integrated with antivirus systems

References

- [1] Shahzad, F., Shahzad, M., & Farooq, M. (2011). *ELF-Miner: Using Structural Knowledge and Data Mining Methods to Detect New (Linux) Malicious Executables*. Knowledge and Information Systems, 30(3), 589–612.
- [2] Guesmi, T., Kalghoum, A., & Alshammari, B. M. (2023). *Lightweight Header-Based Analysis for IoT Malware Detection Using Machine Learning*. Applied Sciences, 13(5), 2813.
- [3] Tien, C. W., Liao, J. W., Chang, S. C., & Kuo, S. Y. (2020). *IoT Malware Analysis and New Pattern Discovery Through Sequence Analysis Using a Meta-Feature Based Approach*. IEEE Access, 8, 215143–215159.
- [4] Xue, H., Sun, S., Venkataramani, G., & Lan, T. (2021). *LAMP: Label-Free API Call Monitoring for Robust Android Malware Detection*. ACM SIGSAC Conference on Computer and Communications Security.
- [5] Ramamoorthy, S., & Vasudevan, S. (2022). *elfradar: An Architecture-Agnostic Static ELF Malware Analysis Framework*. IEEE Symposium on Security and Privacy Workshops.