Towards Automated Generation of Adversarial Malware Samples with a Genetic Algorithm

József Sándor, Bence Kovács, and Levente Buttyán

Abstract: Machine-learning-based malware detection poses a particular challenge for IoT devices with limited resources. SIMBIoTA-ML has proven effective at identifying variants of known IoT malware families, but its robustness against evasion attacks is insufficient. This paper presents GAME (Genetic Algorithm for Malware Evasion), a framework that automatically generates adversarially crafted malware samples using a Genetic Algorithm (GA). Instead of patching ELF executables at the byte level, GAME introduces format-preserving modification strategies whose parameters are evolved by the GA to find optimal evasion settings for a given binary. GAME reliably generates evasion strategies whose adversarial examples successfully deceive SIMBIoTA-ML, making them well-suited for adversarial training to make the detector more robust.

Keywords: IoT Malware Detection, Adversarial Examples, Genetic Algorithm

1 Introduction

The proliferation of Internet of Things (IoT) devices has dramatically expanded the cybersecurity attack surface. Unlike traditional IT infrastructure, IoT environments are characterized by heterogeneous architectures and severe resource constraints, making them attractive targets for large-scale botnet operations. The devastating impact of this paradigm shift was demonstrated by the Mirai botnet and sustained by subsequent polymorphic families such as Qbot and SpyBot.

The industry has moved beyond rigid signature-based detection toward similarity-based and machine learning (ML) approaches. SIMBIoTA-ML [2] is a prime example: an effective ML-based malware detector for IoT devices with lightweight resource requirements. However, new detection techniques are not necessarily robust against deliberately crafted adversarial examples (AEs), i.e., modified malware samples that evade detetcion by a specific anti-malware system while preserving full malicious functionality.

Adversarial training is an approach to make ML-based malware detectors more robust against evasion attacks by using AEs too in their training. This requires high-volume, diverse, and valid AE datasets. The central problem addressed in this paper is the design of an automated framework capable of producing such datasets without human intervention. Our approach is to use a Genetic Algorithm (GA) to traverse the solution space, where the evolutionary process is grounded in rigorously defined modification primitives – atomic, format-preserving operations that allow the algorithm to alter a binary's structural and statistical footprint without disrupting execution flow. Our framework, called GAME, reliably generates AE creation strategies, which produce AEs that deceive SIMBIoTA-ML. Hence, ultimately, GAME can be used in the adversarial training of ML-based detectors to make them more robust.

2 Background

SIMBIoTA-ML [2] is a lightweight machine learning-based malware detector for embedded IoT devices. It trains a Random Forest classifier using feature vectors extracted from TLSH hashes [1]. SIMBIoTA-ML reports a malware detection rate of approximately 95% while maintaining a low false positive rate. Prior adversarial strategies against SIMBIoTA-ML include the Chunker (appending chunks of the malware to itself to distort its statistical profile) and Disguiser (appending a benign file to drag the feature vector toward the benign cluster) [3]. While

adversarial training against these known strategies improves robustness, it inherently relies on the defender anticipating the attacker's logic. This paper argues that fixed strategies are insufficient proxies for determined attackers and proposes a GA to discover novel, unforeseen evasion techniques.

The **Executable and Linkable Format (ELF)** is the standard binary format for Linux-based systems, covering the vast majority of IoT firmware running on ARM, MIPS, and PowerPC architectures. The malware samples targeted in this work are ELF binaries as well. To evade SIMBIoTA-ML detection, these binaries must be modified in a way that alters their TLSH hash values, while preserving full execution functionality – a non-trivial constraint, as malformed binaries are immediately rejected by the operating system's loader. To ensure all generated AEs remain valid and functional, GAME uses the LIEF (Library to Instrument Executable Formats) library, which handles the low-level structural bookkeeping of ELF manipulation automatically.

Genetic Algorithms are stochastic search heuristics inspired by natural selection, designed for optimization over vast, discrete, non-differentiable search spaces [4]. In this work, we employ a GA in which the population consists of modification strategies (genotypes) rather than raw binary files [5]. Each strategy is evaluated by instantiating it as a set of concrete modified binaries (phenotypes). The evolutionary operators (mutation, crossover) act upon the genotypes, and the fitness evaluation is performed on the phenotypes.

3 Approach and Implementation

The framework operates as a closed-loop evolutionary system. Given a functional, malicious ELF binary as input, it iteratively evolves a population of modification strategies to evade the detection of SIMBIoTA-ML. The system is black-box: it requires no knowledge of the detector's internal state, relies solely on the prediction label. Each generational cycle consists of the following phases:

1. Population Creation: N strategies are initialized randomly in the first generation to ensure broad search space coverage.
2. Phenotype Evaluation: Each strategy is executed multiple times; each run produces a distinct modified ELF binary. The strategy's fitness is the average fitness across its phenotypes, ensuring statistical robustness.
3. Selection: Tournament selection with elitism preserves the top-k strategies unchanged.
4. Crossover and Mutation: Selected parents are recombined and subjected to mutation – structural changes (swapping injection primitives or data sources) and parameter-level tuning.
5. Re-population: The next generation is composed of elites, fresh random strategies, and offspring.

The fundamental unit of evolution is the Gene, representing one atomic modification. A complete strategy is defined as an ordered sequence of these genes, allowing for potential chaining of modifications. Formally, each gene G is defined as:

$$G = \langle f_{slot}, f_{data}, \mathbf{P}_{dist}, \mathbf{P}_{meta} \rangle \tag{1}$$

where f_{slot} selects the injection primitive (append, padding overwrite, or segment injection), f_{data} selects the payload data generation function, $\mathbf{P}_{dist}$ encodes the statistical parameters of the data generator (e.g., mean), and $\mathbf{P}_{meta}$ encodes global constraints (e.g., payload size).

Three format-preserving **modification primitives** (f_{slot}) are implemented using the `LIEF` library:

- Data Appending: Appends a generated payload to the physical end of the ELF file. Requires no structural parsing; highly effective but increases file size.
- Padding Overwriting: Exploits memory alignment gaps between ELF segments by overwriting unused padding bytes with generated data, incurring zero file size overhead.
- Segment Injection: Instantiates a new `PT_LOAD` segment and relies on `LIEF`'s builder to automatically recalculate virtual addresses and header offsets, integrating the payload into the binary's memory map.

To defeat the hash calculation mechanism of the TLSH, injected payloads must exhibit a statistical texture similar to legitimate binaries. Two categories of **data generation** (f_{data}) are implemented:

- Distribution-Based Generation: Uses Inverse Transform Sampling over Normal, Gamma, and Weibull distributions to synthesize bytes whose frequency profiles mimic machine code or compressed data regions. Distribution parameters are evolved by the GA.
- File-Based Generation: Transplants byte sequences from benign ELF binaries using sequential sampling (preserving local structure) or random block sampling (preserving the global histogram while destroying local patterns).

The **fitness function** balances evasion effectiveness against stealth, thus modeling the attacker's objectives. The primary driver is the TLSH difference score, reflecting the core evasion goal. Secondary stealth constraints include size overhead and entropy deviation. Stealth constraints penalize naive strategies, biasing the algorithm toward discovering sophisticated techniques. Critically, if the detector classifies the phenotype as malicious, the fitness score is harshly reduced, making detection evasion the dominant objective.

4 Experiments and Results

For the experiments, we used ARM binaries: 2,000 malware samples[1] and 2,000 benign files extracted from firmware images. These samples served as the training data for SIMBIoTA-ML. After training, we applied GAME to a subset of the malware samples to generate evasion strategies against the detector.

The experimental results show a consistent and rapid convergence pattern (see Figure 1). The initial random population (first generation) typically exhibited low fitness, with most individuals falling below the detection threshold. Within 5 to 10 generations, the average population fitness consistently spiked. By generation 32, over 90% of evolved samples successfully bypassed SIMBIoTA-ML.

Injecting a small number of fresh random strategies per generation consistently produced earlier breakthroughs in detection evasion, particularly for samples that proved difficult to evade. Without fresh strategies, the population could stagnate for many generations without crossing the detection threshold. Mutation rate analysis revealed a trade-off: low mutation rates sufficed when the initial population already contained strong candidates, enabling stable hill-climbing convergence. Higher mutation rates were necessary for weaker initial populations to ensure adequate exploration, but increased the risk of losing good strategies. This trade-off is managed by the elitism mechanism, which guarantees that the best solutions are never discarded.

Padding Overwrite and Segment Injection consistently dominated the top fitness tier. Since padding overwrite adds zero file size overhead, it achieves near-perfect scores on the stealth

[1] `https://github.com/CrySyS/cube-maliot-2021`

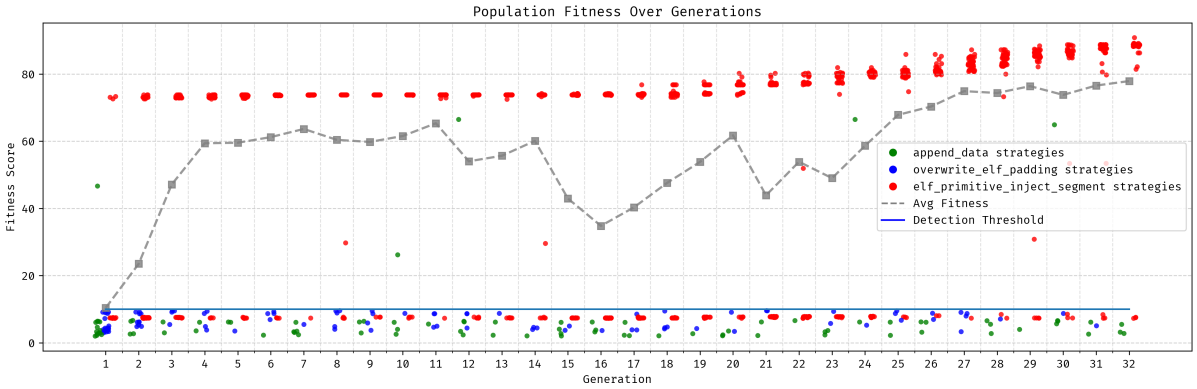

Figure 1: Evolution of strategies for creating AEs in case of a Mirai malware variant.

components of the fitness function. Segment injection proved similarly robust. The Append Data strategy is effective at evasion but penalized by size overhead. The evolution of data distribution parameters resulted in payloads closely matching the entropy of the original code, effectively masking modifications from entropy-based heuristics. Successful AEs typically exhibited a file size increase of less than 5-10%, significantly lower than naive appending approaches.

5 Conclusion

This paper presented an automated evolutionary framework for generating AEs targeting ML-based IoT malware detectors, such as SIMBIoTA-ML. By defining a semantic genetic representation of modification strategies and implementing format-preserving binary manipulation primitives, the system reliably evolves evasion strategies without human intervention. Our experiments confirmed that the adversarial search space for ELF binaries is vast, with the GA consistently achieving over 90% evasion rates within 32 generations. Stealth-oriented primitives – particularly padding overwrite – proved dominant due to their zero file size overhead. The primary defensive application of this work, also our future work, is adversarial training: integrating the GAME framework into the training loop of malware detectors to expose their model to novel AEs, thus increasing their robustness.

References

[1] J. Oliver, C. Cheng, and Y. Chen, "TLSH - A Locality Sensitive Hash," in Proc. 4th Cybercrime and Trustworthy Computing Workshop, IEEE, 2013.

[2] D. Papp, G. Acs, R. Nagy, and L. Buttyan, "SIMBIoTA-ML: Light-Weight, Machine Learning-Based Malware Detection for Embedded IoT Devices," in Proc. IoTBDS, 2022.

[3] J. Sandor, R. Nagy, and L. Buttyan, "Increasing the Robustness of a Machine Learning-Based IoT Malware Detection Method with Adversarial Training," in Proc. ACM Workshop on Wireless Security and Machine Learning, 2023.

[4] J. R. Koza, "Genetic Programming as a Means for Programming Computers by Natural Selection," Statistics and Computing, 1994.

[5] F. Wang et al., "Binary Black-Box Adversarial Attacks with Evolutionary Learning against IoT Malware Detection," Wireless Commun. Mobile Comput., 2021.