

FAME: From Adversarial Feature Vectors to Malware Evasion via SMT

Endre Glocker[✉], József Sándor[✉], and Roland Nagy[✉]

CrySyS Lab, Budapest University of Technology and Economics, Hungary

Abstract—Internet of Things (IoT) malware detection remains a challenging task due to the severe resource constraints of IoT devices, which require lightweight yet effective security solutions. SIMBioTA-ML addresses this challenge by leveraging machine learning (ML) classifiers operating on TLSH-based static features for efficient malware detection. However, like other ML-based systems, SIMBioTA-ML is vulnerable to adversarial evasion attacks. While prior work investigated this problem through ad-hoc binary modifications, it did not provide a systematic or guaranteed approach for generating adversarial examples.

In this paper, we propose FAME (From Adversarial feature vectors to Malware Evasion), a framework for generating adversarial malware samples against SIMBioTA-ML. FAME applies established ML evasion attacks in the feature space and addresses the inverse feature-mapping problem between continuous adversarial feature vectors and corresponding executable binaries. To this end, we formulate the discretization of continuous adversarial feature vectors as a Satisfiability Modulo Theories (SMT) problem, enforcing both TLSH validity constraints and SIMBioTA-ML evasion constraints. Using the target TLSH hash determined by the SMT solver, the original malware binary is then modified accordingly. By combining adversarial ML, constraint solving, and controlled binary modification, FAME enables the scalable and reliable generation of executable adversarial malware samples.

Index Terms—IoT, malware detection evasion, adversarial example, inverse feature-mapping problem

I. INTRODUCTION

The proliferation of Internet of Things (IoT) devices has introduced significant security challenges, with IoT malware emerging as an increasingly urgent threat. Unlike traditional computing environments, IoT devices typically have severe resource constraints in terms of processing power, memory, and energy consumption. Consequently, antivirus solutions designed for traditional computing environments are not suitable for the IoT domain. Malware detection solutions for this domain must be lightweight to remain practical. One promising direction is to leverage binary similarity for detection. In this context, machine learning (ML)-based methods have shown considerable potential, as they can be trained on easily extractable features, such as similarity hashes. By learning from these features, such models are able to detect previously unseen malware that shares characteristics with samples observed during training.

Despite their effectiveness, ML-based malware detectors are inherently susceptible to adversarial evasion attacks [1],

where an adversary takes a malware sample that is correctly classified by a detector and applies carefully crafted perturbations to cause it to be misclassified as benign. The resulting modified sample is referred to as an *adversarial example* (AE). However, generating AEs in the malware domain is considerably more challenging than in domains such as image recognition [2]. A key difficulty lies in the fact that most ML attack methods are defined over continuous feature spaces, whereas executable binaries are inherently discrete and must remain functional after modification. Furthermore, malware detectors do not operate on arbitrary feature vectors; instead, inputs are derived from executable binaries through a feature extraction process. As a result, not every point in the feature space corresponds to a valid input in the problem space, i.e., a realizable executable binary. This discrepancy makes it non-trivial to translate an adversarial feature vector into a concrete modification of the original malware sample. In literature, this challenge is referred to as the *inverse feature-mapping problem*, which stems from the gap between the *feature space* and the *problem space* [2].

In this work, we solve this problem in the context of the IoT malware detection method SIMBioTA-ML [3], which represents both malware and benign samples as feature vectors derived from their TLSH hashes [4]. TLSH is a similarity-preserving digest scheme that enables efficient similarity-based classification. Prior work [5] showed that SIMBioTA-ML is vulnerable to AEs; however, that work relied on ad-hoc modifications of malware binaries, without guarantees of successful evasion.

To overcome this, we propose FAME, a systematic three-stage pipeline. First, we generate AEs using established evasion attack strategies from the ML literature. However, these methods are typically designed for continuous feature spaces, whereas TLSH-based representations are discrete. Second, we discretize the adversarial feature vector into a valid TLSH hash. Second, we discretize the adversarial feature vector into a valid TLSH hash. For this, we employ a Satisfiability Modulo Theories (SMT) solver to enforce the syntactic and semantic constraints of TLSH while ensuring that the resulting hash remains within the benign decision region of the classifier. Third, we modify the original malware sample such that its TLSH hash matches this target representation, adapting the method proposed in [6]. Overall, this pipeline enables reliable and scalable generation of AEs against SIMBioTA-ML.

This work was supported by the National Research, Development and Innovation Fund of Hungary (NKKP-ADVANCED_25-153325).

The remainder of this paper is organized as follows. Section II highlights related approaches from the literature that address the inverse feature mapping problem. Section III provides the necessary background and describes the building blocks used in FAME. Sections IV and V present the design and implementation details of FAME. Section VI evaluates the AE generation capability of FAME. Finally, Section VII concludes the paper.

II. RELATED WORK

Early adversarial attacks against malware detectors primarily focused on generating adversarial feature vectors without guaranteeing that the resulting perturbations corresponded to valid and functional malware binaries [7]. Existing methods for solving the inverse feature mapping problem remain tightly coupled to specific malware formats and feature extractors, thereby limiting their generality. We therefore highlight several representative works that employ different approaches.

Grosse et al. [8] proposed one of the first practical attacks against the Android malware detector. Their method generated AEs by applying semantics-preserving source-code modifications, ensuring that the resulting applications remained executable yet malicious. Pierazzi et al. [2] formalized the distinction between feature-space and problem-space attacks and showed that arbitrary feature perturbations often do not correspond to realizable malware samples. Their work models attacks as sequences of valid binary transformations that induce desired feature changes while preserving functionality. In the PDF malware domain, Li et al. [9] proposed fvGAN, which generates adversarial feature vectors using a GAN and reconstructs functional malicious PDFs by injecting benign contents and metadata into the document structure. The attack leverages the flexibility of the PDF format to realize feature modifications without breaking exploit execution. For PE malware detectors such as MalConv, Demetrio et al. [10] proposed functionality-preserving black-box optimization techniques that use a genetic algorithm to search over valid PE transformations in order to evade malware detectors while maintaining executable correctness.

III. BACKGROUND

SIMBioTA-ML is an IoT malware detection method designed for resource-constrained devices [3]. It trains an ML model on static features derived from previously observed malware and benign samples, where the static feature is the TLSH hash of the file. Leveraging these TLSH-based representations, it enables effective and efficient malware detection.

TLSH is a similarity digest scheme that maps a file to a 35-byte hash value, where similar files at the binary level produce similar hashes. It processes the input using a 5-byte sliding window with a step size of one byte. For each window, all possible selections of three bytes are considered while avoiding repeated byte triads, and these are mapped into 128 buckets. Based on the resulting bucket counts, TLSH computes the q_1 , q_2 , and q_3 quartiles. The final TLSH hash is then

constructed from several components: one byte `checksum`¹, one byte `lvalue` representing the length of the file on a logarithmic scale, half byte for the `Q1ratio` and half byte for the `Q2ratio`, where $Q1ratio = \lfloor 100 \frac{q_1}{q_3} \rfloor \bmod 16$ and $Q2ratio = \lfloor 100 \frac{q_2}{q_3} \rfloor \bmod 16$. Finally, it includes 128 quarter-byte codes, representing the quantized values of the bucket counts in the range 0-3, where 0 indicates that the bucket value is lower than q_1 , 1 if it is between q_1 and q_2 , 2 if it is between q_2 and q_3 , and 3 indicates that it is higher than q_3 .

In the original SIMBioTA-ML paper, a Random Forest (RF) classifier was used [11]; however, ongoing research suggests that other models may offer advantages over RF. These include linear classifiers such as Logistic Regression (LR) [12] and Support Vector Machines (SVM) [13], as well as other ensemble-based methods, including AdaBoost (AB) [14] and Gradient Boosting (GB) [15]. While LR and SVM base their decisions on linear or kernel-based decision functions, ensemble methods combine the outputs of multiple decision trees to improve predictive performance.

A wide range of ML evasion attacks have been proposed in the literature to generate AEs in the feature space. In this work, we apply such attacks to the models used within SIMBioTA-ML. Given a trained classifier, these methods aim to minimally perturb an input sample so that it crosses the decision boundary and is misclassified, thereby producing an AE. We consider both black-box and white-box attack settings. Black-box attacks assume no access to the internal structure or parameters of the target model; instead, the attacker can only query the model and observe its outputs, such as predicted labels or confidence scores. These attacks are model-agnostic and can therefore be applied to all classifiers considered in this work. Specifically, we use HopSkipJump [16], Zeroth Order Optimization (ZOO) [17], and the Boundary Attack [18]. In contrast, white-box attacks assume full knowledge of the target model, including its architecture and parameters. Within this category, we distinguish between gradient-based methods and attacks tailored to tree-based models. Gradient-based attacks leverage the gradient of the loss function with respect to the input to identify perturbations that induce misclassification; in our setting, these are applied to SVM and LR models. The methods considered include Fast Gradient Sign Method (FGSM) [1], Basic Iterative Method (BIM) [19], Projected Gradient Descent (PGD) [20], Carlini & Wagner (C&W) attack [21], and DeepFool attack [22]. For the decision tree-based models such as RF, AB, and GB, we employ two specialized attacks that exploit their structure [23], [24].

In practice, we cannot feed adversarial TLSH digests directly to the detector. We must modify the original malware binary to produce the desired adversarial TLSH hash. The TLSH anti-whitelisting attack proposed in [6] enables exactly this: it exploits weaknesses in the TLSH similarity digest scheme to modify executable binaries so that their TLSH hash matches an arbitrary target hash, achieving complete hash collisions. The attack appends specially crafted data to

¹SIMBioTA-ML does not use the checksum field as a feature.

the end of the file, using a greedy byte generation technique that incrementally selects bytes to manipulate bucket counts toward target ranges derived from the desired hash. The attack proceeds in phases: first, achieving the required bucket-count relations to match the Q1 and Q2 ratio fields; then padding with neutral patterns to match the log-length (file-size encoding); and finally adjusting the checksum through a breadth-first search for appropriate trailing bytes.

IV. DESIGN OF FAME

We consider a white-box threat model in which the attacker has full access to the trained classifier, including its parameters². The attacker’s objective is to evade detection by modifying a malware sample such that it is misclassified as benign, while preserving its original functionality. To achieve this, the attacker operates at the binary level, introducing carefully crafted modifications that result in a corresponding adversarial representation in the feature space.

We now describe the overall attack procedure and its individual steps from the perspective of an adversary. First, we select a malware sample that is correctly classified as malicious and apply one of the evasion attacks described in Section III. This yields a perturbed feature vector corresponding to an adversarial feature vector. The AEs generated in the continuous feature space do not directly correspond to valid TLSH hashes. Since TLSH is defined over 131 discrete features, a naive discretization approach – enumerating all possible floor/ceiling combinations – would require exploring 2^{131} candidates, which is computationally infeasible.

To address this, we instead project the continuous AEs onto valid TLSH hashes that are still classified as benign by SIMBioTA-ML. This discretization is formulated as a constraint satisfaction problem and solved using an SMT solver. Conceptually, the procedure searches over the vertices of the integer unit hypercube surrounding the AE in feature space. The search is subject to two classes of constraints:

- **TLSH constraints:** The solution must satisfy the syntactic and semantic constraints of a valid TLSH hash.
- **Classification constraints:** The discretized point must remain within the benign decision region of the classifier.

Figure 1 illustrates the idea of the discretization process in a simplified two-dimensional setting, where each axis corresponds to a TLSH feature. Starting from an original malware feature vector, the ML-based evasion attack produces a continuous adversarial feature vector, denoted as AE_{ML} , which lies across the decision boundary in the benign region. Since TLSH features are discrete, we consider the vertices of the unit hypercube surrounding AE_{ML} ; in two dimensions, this reduces to a unit square representing candidate adversarial TLSH hashes (AE_{C1} – AE_{C4}). Among these candidates, only a subset may satisfy both the structural constraints of valid TLSH hashes and the requirement of remaining within the

benign decision region. In the depicted example, AE_{C3} is the only candidate that fulfills both conditions and thus constitutes a valid adversarial TLSH hash. In contrast, AE_{C1} and AE_{C2} fall back into the malware decision region, while AE_{C4} violates TLSH validity constraints.

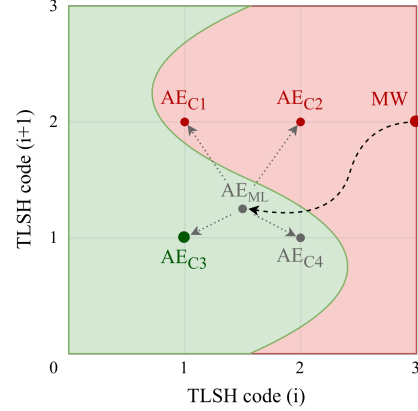


Fig. 1. Discretization of a continuous adversarial feature vector AE_{ML} in two dimensions. Perturbations are added to the original malware (MW) feature vector to obtain AE_{ML} . The vertices of the unit square surrounding AE_{ML} represent candidate TLSH hashes; among them, only AE_{C3} is both valid and classified as benign.

The SMT solver returns *Satisfiable* if a valid adversarial TLSH hash is found, *Unsatisfiable* if no such solution exists, and *Unknown* if the solver exceeds the allotted time limit. If the SMT solver returns *Satisfiable*, we proceed by modifying the original malware binary such that it yields the corresponding adversarial TLSH hash. To achieve this, we employ the TLSH anti-whitelisting attack proposed in [6], as described in Section III.

V. SMT ENCODING

We formulate the continuous feature vector discretization problem as a constraint satisfaction task and solve it using an SMT solver, specifically Z3 [25]. For every feature, we consider both its floor and ceiling values, resulting in a discrete set that represents all candidate TLSH feature vectors in the local neighborhood of the adversarial point. Accordingly, we construct a Z3 solver instance with a configurable timeout, where each feature is represented as an integer variable constrained to lie within the given bounds. In addition to the constraints enforcing discrete feature values, we incorporate both TLSH validity constraints and model decision constraints into the solver, as detailed in the following two subsections.

A. TLSH constraints

To ensure that the discretized feature vector corresponds to a valid TLSH hash, we impose a set of syntactic and semantic constraints on its components. First, all 131 features are required to take integer values within their respective valid ranges. The `lvalue`, which encodes the file size, is constrained based on the original malware sample. Since the anti-whitelisting attack can only increase the file size, the

²A white-box threat model is a realistic assumption, since an attacker may obtain an IoT device – either by gaining physical access to an already deployed one or by purchasing one – thereby gaining full access to the ML model it contains.

lvalue of the adversarial TLSH must be greater than or equal to that of the original malware, while remaining within the valid range [0,169]. The Q1ratio and Q2ratio fields are each constrained to lie between 0 and 15.

The remaining 128 features correspond to the bucket codes, each taking a value in the range [0,3]. These codes indicate how the corresponding bucket counts compare to the quartiles and, therefore, must satisfy additional distribution constraints. Specifically, at least 32 codes must take the value 0, at least 64 codes must be either 0 or 1, and at least 96 codes must be in the set {0,1,2}. These constraints ensure consistency with the quartile-based quantization underlying TLSH.

B. Classification constraints

We encode each trained model by translating its decision function into an SMT formula using its learned parameters. We ensure that any satisfying assignment of the SMT formula corresponds to a feature vector classified as benign by the corresponding model.

LR labels inputs by the sign of a weighted sum of features plus a bias term. We enforce this sum to be less than or equal to zero as a linear inequality in the solver, keeping the feature vector on the benign side of the decision boundary. For SVM, we use a polynomial kernel to implicitly map features into a higher-dimensional space where they become linearly separable. Here, the classifier’s decision value is a weighted sum of kernel evaluations between the input and the support vectors, plus a bias term. We similarly constrain this value to be less than or equal to zero in the solver to keep the feature vector on the benign side.

RF, AB, and GB models process the input feature vector by traversing every tree from the root to a leaf. At each internal node, a feature-based split condition is evaluated, and the path traversal continues along the corresponding branch until a leaf node is reached, which determines the prediction of that tree. The final classification is then obtained by aggregating the outputs of all trees in the ensemble. For every tree, we encode each root-to-leaf path leading to a benign leaf as a conjunction (AND) of split conditions, because all inequalities along the path must hold simultaneously for a sample to reach that leaf. The benign decision of a tree is then represented as the disjunction (OR) over all such benign paths, since satisfying any one of these paths is sufficient to reach a benign leaf.

To obtain the final model decision, the tree-level outcomes are aggregated according to the respective ensemble strategy, which differs across the three models considered. In the case of RF, which uses majority voting, we enforce that a sufficient number of trees reach a benign leaf, i.e., more than half of the trees must vote for the benign class. AB uses weighted voting across the ensemble, where each tree contributes to the final decision according to its learned weight. We encode this mechanism by enforcing that the weighted sum of trees predicting the benign class exceeds half of the total weight of all trees, ensuring that the overall ensemble decision is benign. GB makes a prediction by accumulating contributions from a sequence of trees, rather than relying on voting. It starts with

an initial score and then iteratively refines this value by adding the output of each tree in the ensemble. For each tree, the input traverses to a leaf, whose value provides a small real-valued contribution to the overall score. The final prediction is obtained by summing the initial score and all these incremental contributions. If this aggregated score is less than or equal to zero, the sample is classified as benign.

VI. EVALUATION

In this section, we evaluate the success of FAME by measuring how often originally detected malware can be transformed into evasive malware and, in cases of successful evasion, the required increase in file size.

Our experiments use ARM and MIPS executable binaries, with 2,000 malware and 2,000 benign samples per architecture. The malware samples come from the publicly available CUBE-MALIOT-2021 dataset³, and the benign samples were extracted from IoT firmwares. We evaluate all experiments separately for ARM and MIPS architectures. To ensure the reliability of our results, all measurements are repeated 10 times and averaged. We use a 50%–50% split for the training and test sets. The trained models achieve strong baseline performance on the test set, with accuracy generally ranging between 97% and 99%. Our goal is to modify the true positive (TP) samples using the method of FAME to evade detection.

The method proceeds as follows. For each TP sample, we first compute its TLSH hash and extract the corresponding feature vector. We then apply ML attacks to add perturbations to this feature vector. Next, the resulting continuous adversarial feature vectors are converted back into valid TLSH hashes using an SMT solver. Finally, an anti-whitelisting attack is employed to modify the original malware sample so that it matches the target TLSH hash. In the following, we analyze how many samples are lost during each phase of the process.

The first phase (Ph-1) of FAME is to generate AEs in the continuous feature space. For this, we use the state-of-the-art Adversarial Robustness Toolbox (ART) framework [26]. We implemented two white-box attacks against decision tree-based models, which we refer to as Tree-based-1 [23] and Tree-based-2 [24]. All other attacks (listed in Section III) were already available in ART. Black-box attacks are applied uniformly across all models. In contrast, model-specific white-box attacks are used against the corresponding models, as they are designed to exploit the internal structure of their respective architectures. The first (ARM) and fourth (MIPS) columns of Table I show the number of adversarial feature vectors generated by these attacks for linear models, while Table II shows the corresponding results for decision tree-based models.

In phase two (Ph-2), the SMT solver attempts to find a valid TLSH hash close to the adversarial feature vector produced in Ph-1. Since the SMT solver may return either Unsatisfiable or Unknown⁴, the number of samples

³<https://github.com/CrySyS/cube-maliot-2021>

⁴We used a 60-second timeout per sample for the SMT solver in order to keep the total runtime of the evaluation feasible across the full dataset.

TABLE I
NUMBER OF EVASIVE SAMPLES ACROSS THE THREE PHASES FOR
DIFFERENT LINEAR CLASSIFIER-ATTACK COMBINATIONS.

Model	Attack	Phases-ARM			Phases-MIPS		
		Ph-1	Ph-2	Ph-3	Ph-1	Ph-2	Ph-3
LR	BIM	993	993	993	990	986	986
	Boundary	992	975	919	990	972	934
	C&W	941	941	940	988	988	988
	DeepFool	953	953	953	944	944	944
	FGSM	992	992	992	988	987	987
	HopSkipJump	992	986	939	989	985	959
	PGD	992	992	992	988	988	988
	ZOO	982	982	970	988	988	978
SVM	BIM	996	996	996	992	992	992
	Boundary	996	927	893	988	834	815
	C&W	995	995	995	992	991	989
	DeepFool	59	59	59	241	241	241
	FGSM	996	996	996	992	991	990
	HopSkipJump	996	983	976	992	981	979
	PGD	996	996	996	992	992	992
	ZOO	0	0	0	0	0	0

TABLE II
NUMBER OF EVASIVE SAMPLES ACROSS THE THREE PHASES FOR
DIFFERENT TREE-BASED CLASSIFIER-ATTACK COMBINATIONS.

Model	Attack	Phases-ARM			Phases-MIPS		
		Ph-1	Ph-2	Ph-3	Ph-1	Ph-2	Ph-3
AB	Boundary	977	956	656	979	957	792
	HopSkipJump	976	914	527	978	963	750
	Tree-based-1	978	978	932	979	979	978
	Tree-based-2	978	978	699	980	980	773
	ZOO	917	664	607	976	721	396
GB	Boundary	997	978	500	996	976	831
	HopSkipJump	997	968	458	997	983	667
	Tree-based-1	978	949	353	989	962	326
	Tree-based-2	915	906	293	866	864	88
	ZOO	726	719	266	792	789	212
RF	Boundary	994	974	908	991	974	942
	HopSkipJump	993	979	816	992	990	928
	Tree-based-1	988	966	943	984	965	896
	Tree-based-2	956	954	626	957	957	624
	ZOO	994	887	399	828	827	652

reported for Ph-2 in Tables I and II is less than or equal to the corresponding number in the previous column.

In the last phase (Ph-3), we execute the anti-whitelisting attack on the original malware using the target TLSH hash found in Ph-2. Here as well, the number of samples reported for Ph-3 in Tables I and II is less than or equal to the corresponding numbers in Ph-2. The reason for this reduction is the `lvalue` of the target TLSH: if it is smaller than the `lvalue` of the original malware's TLSH, the anti-whitelisting attack fails, since it can only increase the file size.

Figures 2 and 3 report the relative size increase of the generated AEs compared to the original malware for linear and tree-based models, respectively. Each box represents the size increase ratio for a specific model-attack combination in Ph-3. Generally, MIPS samples require a larger size increase than ARM samples. Moreover, there is significant variance across model-attack combinations.

We observe that SVM is the most robust model in terms of size increase, as all attacks result in significant file size growth. Moreover, the ZOO and DeepFool attacks either fail to generate AEs or only partially succeed. This suggests

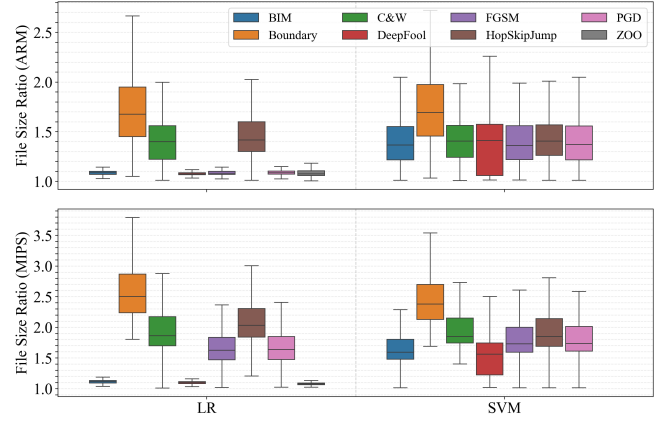


Fig. 2. Size increase induced by the anti-whitelisting attack on malware samples to evade the detection of LR and SVM.

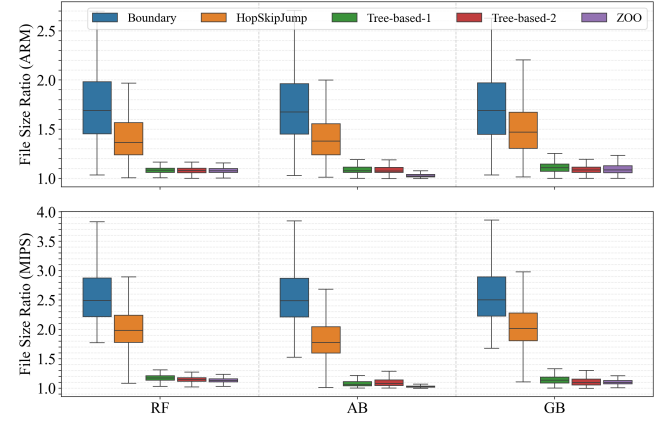


Fig. 3. Size increase induced by the anti-whitelisting attack on malware samples to evade the detection of tree-based models.

that the SVM decision boundary is particularly resistant to the perturbation directions these attacks explore. For LR, the BIM, DeepFool, and ZOO attacks yield the smallest file size increases across both the ARM and MIPS datasets. In the case of tree-based classifiers, one clear trend is that the Boundary and HopSkipJump consistently require larger increases in file size, which is not surprising since they are black-box attacks. Although the ZOO attack is also a black-box method and results in a relatively small increase in size, it generates fewer AEs than Boundary and HopSkipJump. In contrast, targeted white-box attacks against tree-based classifiers generally require smaller increases in file size.

The reader may ask why FAME is preferable to simply modifying a malware sample such that its TLSH hash matches that of a known legitimate binary, such as GCC. First, the most effective attacks produced by FAME require a smaller increase in binary size than forging the TLSH hash of GCC using the method proposed in [6]. Second, TLSH hashes of widely distributed legitimate programs are often known a priori. Consequently, if a received file shares the same

TLSH hash as GCC, a defender can identify the discrepancy by comparing cryptographic hashes such as SHA-256 against the legitimate binary. In contrast, FAME generates previously unseen TLSH hashes that lie within the benign decision region of the classifier, making such heuristic verification approaches considerably less effective.

As we stated earlier, FAME targets the SIMBIOta-ML malware detector specifically. However, in this section, we discuss how AEs generated by FAME behave against other malware detection approaches. FAME is based on attacks against TLSH; thus, other malware detectors that use TLSH might be affected. We performed experiments, where we compared the TLSH hashes of AEs and the original malware samples, and found that the larger the amount of added bytes is, the larger the TLSH difference. Thus, against solutions like SIMBIOta [27], FAME might be effective. Against other malware detectors, FAME's efficiency boils down to what kind of features are used and how they are extracted. E.g., against malware detectors using different similarity digests, the adversarial modifications are probably irrelevant, while dynamic-analysis-based solutions would certainly not be affected.

VII. CONCLUSION

We proposed FAME, a framework capable of generating AEs at scale while inherently solving the inverse feature-mapping problem in the context of SIMBIOta-ML. We evaluated FAME using five ML models and ten evasion attacks from literature, demonstrating the effectiveness of the proposed approach across a diverse set of classifiers and attack strategies. Future work will focus on improving the robustness of SIMBIOta-ML against AEs generated by FAME. In particular, the generated AEs could be leveraged for defense mechanisms such as adversarial training to improve the resilience of the models against evasion attacks. We expect that, after adversarial training, the ML attack will succeed less often, the SMT solver will fail more frequently, and, in the cases where evasion still succeeds, the required file size increase will be larger.

ACKNOWLEDGMENT

We thank Dorottya Futóné Papp for the idea of using SMT for the discretization problem, Gábor Fuchs for providing the implementation of the anti-whitelist attack, Gergely Ács for his insights on ML evasion attacks, and Levente Buttyán for his valuable feedback on this paper.

REFERENCES

- [1] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv 1412.6572*, 2014.
- [2] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing Properties of Adversarial ML Attacks in the Problem Space," in *IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1332–1349.
- [3] D. Papp, G. Ács, R. Nagy, and L. Buttyán, "Simbiota-ml: Light-weight, machine learning-based malware detection for embedded iot devices," in *International Conference on Internet of Things, Big Data and Security (IoTbDS)*, 2022.
- [4] J. Oliver, C. Cheng, and Y. Chen, "Tlsh – a locality sensitive hash," in *Fourth Cybercrime and Trustworthy Computing Workshop*, 2013, pp. 7–13.
- [5] J. Sándor, R. Nagy, and L. Buttyán, "Increasing the robustness of a machine learning-based iot malware detection method with adversarial training," in *Proceedings of the 2023 ACM Workshop on Wireless Security and Machine Learning*, ser. WiseML'23, 2023, p. 3–8.
- [6] G. Fuchs, R. Nagy, and L. Buttyán, "Targeted attacks against the tlsh similarity digest scheme," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 8922–8935, 2025.
- [7] K. Aryal, M. Gupta, M. Abdelsalam, P. Kunwar, and B. Thuraisingham, "A survey on adversarial attacks for malware analysis," *IEEE Access*, vol. 13, pp. 428–459, 2025.
- [8] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European Symposium on Research in Computer Security*, 2017, pp. 62–79.
- [9] Y. Li, Y. Wang, Y. Wang, L. Ke, and Y. an Tan, "A feature-vector generative adversarial network for evading pdf malware classifiers," *Information Sciences*, vol. 523, pp. 38–48, 2020.
- [10] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, "Functionality-preserving black-box optimization of adversarial windows malware," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3469–3478, 2021.
- [11] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.
- [12] J. Berkson, "Application of the logistic function to bio-assay," *Journal of the American Statistical Association*, vol. 39, no. 227, pp. 357–365, 1944.
- [13] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92, 1992, p. 144–152.
- [14] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [15] J. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, 11 2000.
- [16] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1277–1294.
- [17] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17, 2017, p. 15–26.
- [18] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *International Conference on Learning Representations*, 2018.
- [19] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *ArXiv*, vol. abs/1607.02533, 2016.
- [20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.
- [21] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.
- [22] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2574–2582.
- [23] C. Koball and Y. Wang, "Evasion attacks on tree-based machine learning models," in *IEEE International Conference on Consumer Electronics (ICCE)*, 2025, pp. 1–6.
- [24] C. Koball, Y. Wang, V. Vaidyan, and J. Hastings, "Assessing evasion attacks on tree-based machine learning models: Supervised vs. unsupervised approaches," in *IEEE International Conference on Consumer Electronics (ICCE)*, 2025, pp. 1–6.
- [25] L. de Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340.
- [26] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba et al., "Adversarial robustness toolbox v1.2.0," *CoRR*, 2018.
- [27] C. Tamás, D. Papp, and L. Buttyán, "Simbiota: Similarity-based malware detection on iot devices," in *6th International Conference on Internet of Things, Big Data and Security (IoTbDS)*, 2021.