FINER: Enhancing State-of-the-art Classifiers with Feature Attribution to Facilitate Security Analysis

Yiling He
Zhejiang University
yilinghe@zju.edu.cn

Jian Lou
Zhejiang University
jian.lou@zju.edu.cn

Zhan Qin
Zhejiang University
qinzhan@zju.edu.cn

Kui Ren
Zhejiang University
kuiren@zju.edu.cn

ABSTRACT

Deep learning classifiers achieve state-of-the-art performance in various risk detection applications. They explore rich semantic representations and are supposed to automatically discover risk behaviors. However, due to the lack of transparency, the behavioral semantics cannot be conveyed to downstream security experts to reduce their heavy workload in security analysis. Although feature attribution (FA) methods can be used to explain deep learning, the underlying classifier is still blind to what behavior is suspicious, and the generated explanation cannot adapt to downstream tasks, incurring poor explanation fidelity and intelligibility.

In this paper, we propose FINER, the first framework for risk detection classifiers to generate high-fidelity and high-intelligibility explanations. The high-level idea is to gather explanation efforts from model developer, FA designer, and security experts. To improve fidelity, we fine-tune the classifier with an explanation-guided multi-task learning strategy. To improve intelligibility, we engage task knowledge to adjust and ensemble FA methods. Extensive evaluations show that FINER improves explanation quality for risk detection. Moreover, we demonstrate that FINER outperforms a state-of-the-art tool in facilitating malware analysis.

CCS CONCEPTS

• **Security and privacy → Software and application security**.

KEYWORDS

Model Explainability; Malware Analysis; Human-AI Interaction

ACM Reference Format:
Yiling He, Jian Lou, Zhan Qin, and Kui Ren. 2023. FINER: Enhancing State-of-the-art Classifiers with Feature Attribution to Facilitate Security Analysis. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3616599

1 INTRODUCTION

Deep learning (DL) based classifiers have shown great potential in the risk detection phase. They automate large-scale detection and achieve considerable accuracy for different risk types including mobile malware [8, 33, 48, 52], code vulnerability [14, 39, 40], and network intrusion [23, 25]. However, when coming to the security analysis stage, those classifiers fall short as they only produce prediction labels. This problem is severe since security experts need to actively respond to the detected risks. Without knowing the reason for each detection, they would face an extremely difficult task considering the large amount of detected risk [10] and the tedious workload in risk dissection [46].

To explain DL-based decisions, *feature attribution* (FA) methods are promising for being compatible with various model architectures [32]. FA methods work on trained classifiers and assign an importance score for each feature of an individual input. A successful application in image analysis is known as saliency map [4], where important pixels are visually highlighted on an image for human users to inspect. Motivated by the success, prior works try to explain risk detection classifiers with the same approach but find a low explanation *fidelity* [77]. Although a few security-customized FA methods have been proposed to improve fidelity [78], they are task-specific since they add intuitive constraints to a certain domain-general method. For example, LEMNA [27] considers the same black-box setting with LIME [57], but differs from it in handling feature dependency, which is mostly observed for recurrent neural network (RNN) based applications, e.g., function start detection [67]. Unsurprisingly, as validated by our experiments in Section 5, they need more computational cost while the fidelity improvement is limited / does not hold true in other tasks.

Explaining data-driven risk detection classifiers is more challenging. First, the fidelity problem originates from the diversity of classifiers that use particular data representations (as shown in Table 1). For example, to represent a binary program, the extracted features can be a long sequence of opcodes, APIs, or hand-crafted statistics [2]; when encoded into vector space, the data can be of large sizes and have great variance in shape and distribution. Second, an overlooked *intelligibility* problem is caused by the semantic gap between model feature and actionable understanding. As the example in Figure 2, the data-driven malware classifier explores low-level language (i.e., bytecode) that is intrinsically hard to read and individual features (i.e., opcodes) do not serve as the fundamental unit of maliciousness. Under such circumstances, the per-feature explanation style of FA is not insightful to help security experts with security analysis.

In this paper, we address the fidelity and intelligibility issues from the perspective of the explainable risk detection system (ERDS). Specifically, ERDS has an end goal to facilitate security analysis with explanations, and it consists of a data-driven classifier and an FA-based explainer. Our design is based on two intuitions. First, the classifier should be the major agent of ERDS, and high-fidelity explanations depend on its reasonable decision boundary [59]. For instance, risk detection should be correlated with semantic features

CCS '23, November 26–30, 2023, Copenhagen, Denmark.

ACM ISBN 979-8-4007-0050-7/23/11…$15.00
https://doi.org/10.1145/3576915.3616599

instead of artifacts [7], which is a prerequisite for downstream explanation endeavors (e.g., customizing FA to handle feature dependency) to be effective. Second, security experts should be the consumer of ERDS, and intelligibility relies on their desiderata for specific tasks to abstract low-level explanations. For example, since malicious functionalities are consumable explanations for malware analysts [21], ERDS with the opcode feature-based classifier should be adjusted to generate function-level explanations.

We propose a framework, named FINER, to promote data-driven risk detection classifiers into ERDS that generates useful explanations for security analysis. To improve fidelity, we design an explanation-guided data augmentation strategy for risk samples and leverage it in multi-task learning to fine-tune the classifier. To improve intelligibility, we define task-aware explanations on the level of *intelligible component* (IC) and correspondingly adapt a fidelity metric to ensemble different FA methods. Specifically, FINER has an interface to engage with task knowledge, and three modules (namely *explanation-guided model updating*, *task-aware explanation generation*, and *explanation quality measurement*) to build ERDS with a more interpretable classifier and a more adaptable explainer. The decoupled architecture also supports the needs of different stakeholders at different stages of building an ERDS.

We apply FINER on three state-of-the-art risk detection classifiers targeting tasks including Android malware detection [49], Windows malware detection [21], and vulnerability detection [37]. The classifiers are trained on 14K apps, 48K binaries, and 32K gadgets respectively, and the explainers are formed with six representative FA methods including white-box methods [68, 69, 71] and black-box methods [27, 42, 57]. The results show that FINER significantly improves the explanation fidelity of ERDS across all classifiers and explanation scenarios. The updating module is effective to improve model interpretability (from 21.28% to 82.05% depending on the classifier) without accuracy trade-off, and the ensemble module achieves higher explanation fidelity than the baseline (from 10.12% to 17.00% depending on the scenario). We also show that FINER outperforms a state-of-the-art tool in malicious functionality localization task.

Contributions. This paper makes the following contributions.

- We propose a formalization of the ERDS to establish explanation desiderata for security analysis. We propose to address the fidelity and intelligibility problems by explanation-guided model updating and IC-based explanation ensemble.
- We implement the framework FINER to promote data-driven risk detection classifiers into ERDS with high-fidelity and high-intelligibility explanations. FINER can meet the needs of different stakeholders at each stage of building an ERDS, and we make the dataset and code open-source [1].
- We evaluate FINER with three critical risk detection tasks and six representative FA methods, showing that explanation fidelity is improved across all ERDS settings, i.e., different combinations of the classifier and explainer. We also demonstrate that FINER outperforms a state-of-the-art tool in localizing malware functions.

[1] https://github.com/E0HYL/FINER-explain

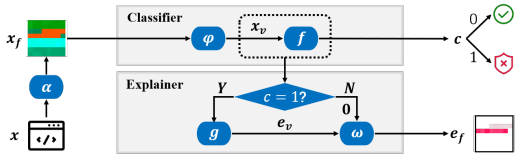


Figure 1: Workflow of ERDS with the formalized notations.

2 EXPLAINABLE RISK DETECTION SYSTEM

In this section, we introduce *explainable risk detection system* (ERDS), where the emerging feature attribution (FA) techniques are used as the *explainer* to interpret the deep learning (DL) detection of the *classifier*. We first define the problem of ERDS and propose a novel formalization. Then, we discuss representative classifiers and explainers from related works and highlight the main parameters in our formalization. Finally, we provide background for understanding the capability of FA explanations.

2.1 Problem Definition

Local Post-hoc Explainability. Explainable risk detection is a subset of the broader field of eXplainable Artificial Intelligence (XAI), which aims to make AI systems more interpretable to humans. As XAI research has progressed in recent years, there is a wide range of techniques designed for explaining different scenarios [22]. To build the vocabulary, we introduce the taxonomy of XAI methods in terms of what can be explained and how explanations can be generated: first, an explanation can indicate how the model decisions are affected by individual instances (*local* methods), or by entire model parameters (*global* methods); second, the explainability can be achieved by building simple yet inherently interpretable models (*intrinsic* methods), or by approximating the decision-making of complex models that are already trained (*post-hoc* methods). Recent advances in the local and post-hoc XAI technique, i.e., FA, have led to the development of ERDS [27, 77]. Different from other techniques, the local post-hoc explainability handles the most common explanation scenario in risk detection that can (1) provide clues for understanding critical risk samples and (2) be compatible with trained state-of-the-art classifiers.

Formalization. Under this context, ERDS is generally expanded from a well-trained risk detection classifier by appending a post-hoc explainer, and the "classification with explanation" workflow for an individual instance can be illustrated as Figure 1. In the following, we propose the formalization of ERDS and its two components, i.e., the classifier and the explainer.

Definition 1 (ERDS). *Given a problem-space instance $x \in \mathcal{Z}$, ERDS first performs feature extraction $\alpha : \mathcal{Z} \to \mathcal{F}$ to embed it into a feature-space representation $x_f \in \mathcal{F}$, and then generates two outputs respectively from its classifier and explainer: a predicted class label $c \in \mathcal{C} = \{0, 1\}$ where the zero value means normal sample and the positive value represents risk sample, and an explanation $e_f \in \mathcal{F}$ indicating why the sample can be identified as a risk.*

Definition 2 (Classifier). *The classification pipeline is defined as $c = f(\varphi(x_f))$, where $\varphi : \mathcal{F} \to \mathcal{V} \subseteq \mathbb{R}^{m \times n}$ generates a m-by-n vector-space representation $x_v \in \mathcal{V}$, such that $\varphi(x_f) = x_v$, and*

Table 1: DL-based risk detection classifiers investigated in related works.

Classifier	Application	Feature Space $\mathcal{F}^*$	Vector Space $\mathcal{V}^\dagger$	Model f
Mimicus+	PDF malware	Static document statistics (based on file structure)	Tabular $(135, 1)$	MLP
Drebin+	Android malware	Static app statistics (based on manifest and bytecode)	Tabular $(\geq 545000, 1)$	MLP
VulDeePecker	Program vulnerability	Static token sequence (from source code)	Pseudo text $(50, 200)$	RNN
DAMD	Android malware	Static opcode sequence (from bytecode)	Pseudo text $(N+, 8)$	CNN
DR-VGG	PE malware	Static ACFG statistics of basic blocks (from bytecode)	Pseudo image $(20000, 18)$	CNN

* Feature-space data representation: the "statistics" is based on several factors chosen by domain experts, according to their experience with risk patterns; since the feature space construction of Mimicus+ and Drebin+ is entirely dependent on hand-crafted patterns, the two classifiers are considered pattern-driven; since VulDeePecker, DAMD, and DR-VGG preserve token, opcode, and basic block, respectively, in feature space without manual selection, the three classifiers are considered as data-driven.

† Vector-space data representation: data modality and the vector-space shape (m, n) in our formalization; for DAMD, m is dynamic that depends on the actual number of opcodes; for other classifiers, the vector-space data are of fixed shape, while a special case is that, for Drebin+, m would increase with the increasing of dataset size [60] and the value reported in the table only applies to an ancient small dataset [85] used in its original paper.

then the trained deep neural network (DNN) $f : \mathcal{V} \to \mathcal{C}$ assigns the representation to the binary class label c.

The classifier deals with binary classification tasks (e.g., whether a computer program is malware or not) and is supervised thanks to the availability of large annotated dataset [73]. Unlike image and text domain, the design of feature space $\mathcal{F}$ involves much domain knowledge and varies among different applications. On the other hand, it is often the case that the classifier would be more accurate/robust if more information is encoded in $\mathcal{F}$ (at the cost of the efficiency of feature extraction) [30].

Definition 3 (Explainer). *For the explanation pipeline, it first calculates a feature importance score e_v with $g : \mathcal{V} \to \mathcal{V}$ if the classifier outputs $c = 1$, or else it would be set to a zero matrix. Next, with $\omega : \mathcal{V} \to \mathcal{F}$, e_f presents the regions of interest (ROI) on x_f corresponding to values in e_v that are above a certain threshold.*

The calculation $g(x_v; \mathcal{M})$ leverages a certain FA algorithm and requires a metadata set $\mathcal{M}$ about the classifier that varies among different algorithms (see Section 2.2). It outputs e_v that has the same shape as x_v, and each element $(e_v)_{ij}$ quantifies the importance of $(x_v)_{ij}$ with respect to $f(x_v)$. The ROI is filtered with a constant threshold $\tau_0 > 0$, which we denote as $\omega(e_v; \tau_0) = \{r_f \in x_f : \|e_v \odot Ind(x_v, \varphi(r_f))\|_1 \geq \tau_0\}$. The indicator function $Ind(x_v, r_v)$ returns a m-by-n binary matrix where the ij-th element would be positive only if $(x_v)_{ij}$ exists in r_v.

The ROI implies risky behaviors and thus would only be non-empty for risky samples. This type of explanations is beneficial for security analysis since it can help experts to quickly identify the threat and develop corresponding prevention strategies. On the other hand, the explanation of normality can have a large overlap between the two classes considering the existence of some common utilities. It will be less distinct for understanding risky behaviors (but worthy of other investigation as in Section 8).

2.2 Diverse Classifiers and Explainers

We describe the classifier and the explainer in related literature [21, 27, 77], with a focus on the design of $\mathcal{F}$ and $\mathcal{V}$, as well as the implementation of $f(\cdot)$ and $g(\cdot)$.

Risk detection Classifiers. We summarize the investigated risk detection classifiers as in Table 1 and have the following findings.

- Applications of Interest: explanations are of high concern for risk detection related with static analysis, especially for malware (including PDF, Android, and PE malware) detection. It is largely due to the high demands that are taken up reverse engineering and the severe negative consequence that a malware a malware spread would bring.planations for these classifiers, ERDS would have the potential to relieve the static analysis process and prevent the unwanted risk spread.
- Different Data Representations: the design of $\mathcal{F}$ and $\mathcal{V}$ varies among the classifiers for encoding different semantic information as different shapes of data. Except for Mimicus+ and Drebin+ that are inherit features from conventional machine learning based methods [8, 70], other advanced risk detection classifiers (i.e., VulDeePecker [37], DAMD [49], and DR-VGG [21]) adopt raw elements (i.e., token, opcode, and basic block) from (disassembled) program code for $\mathcal{F}$. As for $\mathcal{V}$, they encode them as complex text/image data (called pseudo- text/image to highlight the actual difference in data modalities) instead of tabular data.
- Different Model Architectures: MLP, RNN, and CNN are all model types that are used to implement $f(\cdot)$, with the aim of achieving state-of-the-art performance on a particular dataset. While MLP fits tabular data of hand-crafted features, RNN and CNN can handle the data-driven classification tasks on the pseudo text/image data. In particular, specific model architectures often cannot be adopted directly from other domains due to the distinct data shapes and distributions in $\mathcal{V}$.

Feature Attribution Explainers. While various FA methods have been proposed to explain different DNN architectures [18, 64, 84], model-agnostic methods are more suitable for ERDS to handle distinct risk detection classifiers (see examples in Table 2). They can be broadly divided into (1) gradient-based methods that propagate importance signals backward through all neurons of the network, e.g., Gradients, IG, and DeepLIFT, and (2) perturbation-based methods that make feature perturbations while analyzing prediction change, e.g., LIME, LEMNA, and Shapley. As the number of these methods is booming [11, 61], we introduce three factors to consider when selecting candidate explainers:

- Knowledge Assumption: FA methods require different knowledge for the metadata set, and thus some methods would be limited when the explainer cannot have access to certain elements in $\mathcal{M}$. For example, gradient-based methods cannot be applied to black-box scenarios as they require full access to the DNN pipeline [31]. That is, in addition to output layer activation $\mathcal{O}$, they need middle layer activation set $\mathcal{A}$ of the model to accomplish the gradient calculation. Many methods would also involve a baseline set $\mathcal{B}$ to

Table 2: Representative explainers from related works. The involved model metadata (output layer activation $\mathcal{O}$, middle layer activation $\mathcal{A}$, and baseline inputs $\mathcal{B}$) and computational cost are given for understanding the explainers' application scenarios.

Explainer	Metadata $\mathcal{M}$			Computational Cost	
	$\mathcal{O}$	$\mathcal{A}$	$\mathcal{B}$	Propagation	Supplement
Gradients [69]	✓	✓		Forward + Backward	
IG [71]	✓	✓	✓	(Forward + Backward) × (# Interpolations)	
DeepLIFT [68]	✓	✓	✓	(Forward + Layer-wise Backward)*2	
LIME [57]	✓		✓*	Forward × (# Neighbors)	Training LR
LEMNA [27]	✓		✓*	Forward × (# Neighbors)	Training MLR with fussed lasso
Shapley [42]	✓		✓*	Forward × 2^{mn}	Calculating Shapley values

* Implicitly specified baselines for masking features.

provide counterfactual intuition [43]. For gradient-based methods, $\mathcal{B}$ is explicitly specified as a single model input b_v that goes through the same propagation process as the original input x_v. For the perturbation-based method, it is implicitly a background dataset used for feature masking. However, baselines must be carefully chosen with domain knowledge [44] and sometimes are sampled from the training dataset of the model.

- Computational Costs: the cost for explanation cannot be too heavy compared to the classification pipeline. From this perspective, gradient-based methods are usually superior to perturbation-based methods, especially for most risk detection classifiers where the feature size is large. This is because gradient-based methods take a single backward pass for each input data to their approximation rule [5], while perturbation-based methods sample a neighborhood around the instance, perform forward passing for all neighbors, and fit another model for feature influence estimation. Typically, for LIME and LEMNA, training one linear regression model would have $O((mn)^3)$ in time complexity, and LEMNA would be extremely hard to converge with large features; for Shapley, the number of coalitions is related to feature size [80], resulting in $O(2^{mn})$ complexity in forward passing.
- Instance-level Performance: since FA methods rely on general assumptions about the classifier's data and model, their performance is unstable among instances. In other words, they will all fail to handle some corner cases [38], for example, Gradients is found to have a saturation problem when the activation of an input is capped at zero. It is uncertain which method will be the most appropriate until they are evaluated on a specific risk, particularly given the fact that most FA methods are domain-general and primarily cares about data modalities like image and text. LEMNA is a pioneering method proposed for security applications, and it customizes LIME to handle feature dependency and non-linear decision boundary. Intuitively, when working with pseudo-text data and RNN architecture, LEMNA would have a better performance than LIME.

2.3 Understanding Risk Explanations

Semantic Capacity. The semantic capacity of risk explanations depends on its associations to problem-space samples. FA has an intrinsic limitation in that they merely explore the model and do not learn from other resources in the problem space, and thus FA explanations cannot provide risk semantics that are not encoded in the feature space. For instance, if the malware classifier works on tabular features that encode the statistical summary of program patterns (crafted by domain experts), then the information lost during feature extraction (e.g., pattern selection for Drebin [8] and hashing trick for Ember [6]) cannot be captured in explanations. For this type of classifiers, the semantic capacity of FA explanations is relatively low since they only reflect known patterns, which would be too shallow for complex tasks such as malware reverse engineering. To conclude, the higher the level of abstraction in feature extraction α is, the lower the semantic capacity of explanations e_f will be.

Non-trivial Evaluation. Unlike the classification task, evaluating the explanation performance is non-trivial due to the unavailability of labels. It is often impractical to label risk explanations on a large scale since the features are prohibitively long and require much expert knowledge to inspect. For some black-box explainers that use surrogate models (e.g., LR and MLR for LIME and LEMNA), evaluation can be translated into comparing the output probabilities of the surrogate model and the original model, but this approach do not generalize to other FA methods. Therefore, a more general feature deduction-based approach is widely adopted to measure local explanation *fidelity*, which is named descriptive accuracy (DA) [77]. It uses the model prediction of an altered sample where the k most important features are nullified in the feature space. Let τ_k denote the threshold that equals to the k-th largest value in e_v, then with our formalization, the DA metric at k is

$$\begin{aligned} &DA_k(x_v, e_v, f) = f(x'_v)_{[c=1]}, \\ &\text{s. t. } x'_v = x_v \odot (\mathbf{1} - Ind(x_v, r_v)), \text{ where } r_v = \omega(e_v; \tau_k), \end{aligned} \tag{1}$$

and the larger the value drops from $f(x_v)$, the better the explanation is thought to be faithful to the model (i.e., accurate).

3 SCOPE AND MOTIVATION

3.1 Our Research Scope

Goal. We aim to promote ERDS to *assist security experts* in security analysis. The targeted ERDS differ in explanation object (i.e., various classifiers) and application domain (i.e., various risk types). The desired assistance is to reduce human workload by pointing out ROI in risk samples. For example, the ERDS for inspecting malware bytecode should be able to localize malicious functionalities, so that analysts can make less effort in reverse engineering. To this end, ERDS should *accurately* explain why a sample is identified as a risk by the classifier and *comprehensively* convey it to security experts.

Focus. We focus on ERDS that has a high semantic capacity for risk explanations. Thereby, the classifier to be explained are data-driven

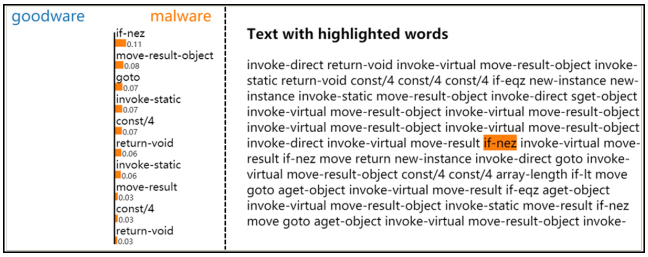


Figure 2: A showcase of the failure of current ERDS: explaining DAMD with the text explainer in LIME toolbox [47]. The selected sample is predicted as malware with 100% confidence by the classifier. For the visualized explanation, the left panel shows the top 10 opcodes that make the sample malicious, and the right panel shows the feature-space opcode sequence with the explanation highlighted. Since the original sequence has a length of 28 245, we display part of the results on the right panel, yet it is representative enough as the explanation is sparse throughout the sequence.

as algorithmic explanations can be mapped back into raw program elements, such as the last three classifiers in Table 1.

Compared with explaining pattern-driven classifiers (see discussion in Appendix), we work on a more *challenging* problem due to the complexity of the feature space and the model design. However, this allows ERDS to benefit more *practical* scenarios, e.g., automatic detection of malicious components (instead of pointing out predefined properties) in malware.

Assumption. We assume that problem-space data samples (e.g., code chunks, Android applications, PE binaries) can be correctly embedded into feature space [35]. That is, problems such as code obfuscation and packing can be solved with existing tools [17] so that the feature extraction α functions well. We also assume that inputs and models are benign for the explainer, while deliberate attacks can be avoided with existing defense methods [81, 82].

3.2 Why Current ERDS is Insufficient

Existing works build ERDS by stacking an off-the-shelf classifier with a specific explainer at hand. Through this approach, the generated explanations usually fail to be accurate or comprehensive, making them useless for security analysis. For example, as the explanation illustrated in Figure 2, the sparsely dispersed opcodes hardly provide analysts with insights about malware behaviors. The failure can be attributed to the following two causes.

Ill-suited Explainer for Classifier. Due to the post-hoc property, the explainer makes some general assumptions about the classifier, leading to unavoidable approximation error for ERDS. Firstly, domain-general FA methods are obviously not suitable for explaining risk detection classifiers. As these methods are designed intuitively for image and text domain, the clear differences from ERDS in dataset distribution, data modality, and model architecture can lead to poor explanation performance. Secondly, although a few security-customized methods have been proposed, they require much higher computational costs and still do not fit ERDS well. For one thing, these methods make efforts to increase the complexity of the approximation model [45]. For another, they rely on observations about security applications to intuitively adapt the approximation assumption, which usually does not summarize risk detection classifiers. For example, the statement that RNN and MLP are more widely adopted might apply to function start detection and tabular feature-based malware detection [27] but does not hold true for advanced risk detection. To be fair, since there exist distinct types of security applications, making a general approximation would be extremely hard.

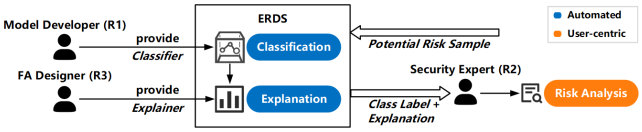


Figure 3: Different Stakeholders involved in building ERDS.

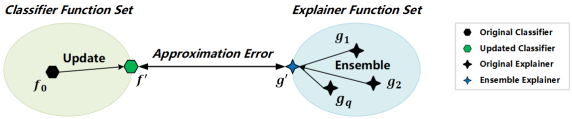


Figure 4: Insights for minimizing the approximation error.

Explanation on Low-level Features. Current ERDS follows FA to provide feature-space explanations, and little user perspective is involved to adapt the explanation level. However, there exists a gap between model features and user understanding, especially for data-driven risk detection classifiers that work on a low abstraction level. As for the example in Figure 2, it is the opcode-level where discrete opcodes require high skills to read and cannot serve as a unit of malicious behavior. However, existing FA applications in security are not fully aware of this issue, as much attention is paid to facilitating internal use cases like model debugging [54, 65]. For rare external use cases that aim to facilitate human analysis, they study pattern-based classifiers where the feature-space elements are originally simple for understanding (e.g., manually abstracted PDF file features for Mimicus+) yet with limited semantic capacity. As a matter of fact, the proper explanation level is task-specific, requiring ERDS to engage with domain knowledge.

To conclude, in order to generate useful explanations for security analysis, building ERDS should be a task-aware process with respect to its internal components and external users. The key challenges lie in (1) handling the mismatch between classifier and explainer; (2) handling the intelligibility gap between classifier and human.

4 FINER FRAMEWORK

4.1 Insights Behind Our Design

To address the above challenges, our solutions are (1) fidelity patch: updating the targeted classifier and ensembling available explainers; (2) intelligibility redesign: engaging with users and generating abstracted explanations. The high-level idea is to improve the explainability of ERDS with joint efforts from different stakeholders. In the following, we first identify the stakeholders with their abilities and expectations. Then, we respectively introduce the insights behind the two solutions.

Identifying Stakeholders. As shown in Figure 3, the three main stakeholders differ in their roles in ERDS. We describe them as:

- Model Developer (R1) who develops the classifier to accomplish a risk detection product. They have knowledge about training

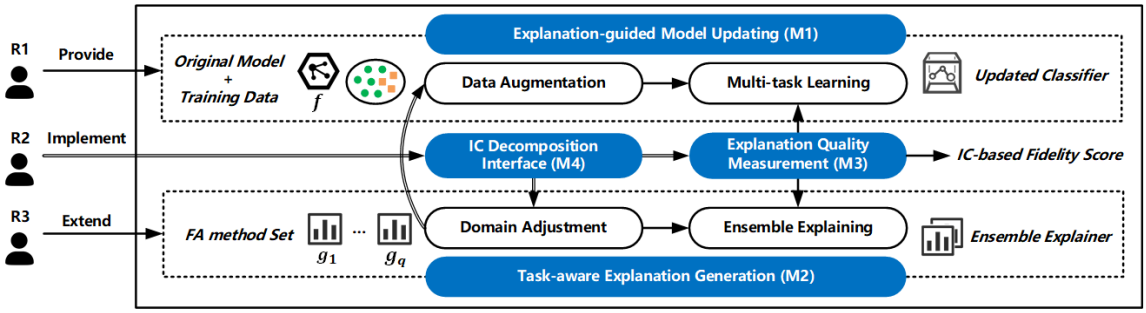


Figure 5: The overall architecture of FINER.

data and certain DL skills. For them, making the model easier to be explained would make the product more trustworthy.

- Security Expert (R2) who is the end user of the system. They are typically non-data scientists but with high domain skills. They want to leverage the automated explanation to reduce their manual work in security analysis, e.g., malware reverse engineering.
- FA Designer (R3) who design FA algorithms to encourage the development of XAI [26]. They are often data scientists motivated by creating a "right to explanation" [15], but care little about the application domain of the classification and the explanation.

Note that different roles can be played by the same identity. Telling them apart actually helps with establishing clear desiderata [12]. For example, under a typical scenario where R1 and R2 belong to the same security company, supporting black-box setting would no longer be desired for R3. We highlight that our design path is different from related works: we investigate how to maximize the contributions that can be made by different stakeholders instead of leaving all the explanation burden to R3.

Minimizing Approximation Error. The essence of improving explanation accuracy is to minimize the approximation error between the exact function $f(\cdot)$ and the approximate function $g(\cdot)$. As illustrated in Figure 4, we find two design points in ERDS:

- Fine-tuning the classifier for post-hoc explanation task. As model training can be regarded as looking for the best in a function set, the definition of the best should be revised when enforcing post-hoc model explainability. Therefore, we would like to update the model parameters of $f(\cdot)$ with multi-task learning. Practitioners who have full access to the classifier and training data, e.g., R2, should be responsible to do this step.
- Ensembling an FA set for local explanations. Given the diversity of FA algorithms and the uncertainty of local performance, it would be helpful to balance different attributions dynamically for different inputs. With this aim, we want to leverage a FA set $\mathcal{G} = \{g_i; i \in \mathbb{N}\}$ and weight the outputs locally. With this design, explanation performance can be improved whenever new FA methods are extended into the system by R3.

Raising Abstraction Level. To make a risk explanation comprehensive, we move the explanation target of ERDS to a high abstraction level that attends R2's understanding. For example, explaining malware with malicious functions instead of discrete opcode features. We define such an explanation as domain-space explanation $e_d \in \mathcal{D}$, which indicates the ROI on *intelligible components* (ICs). To generate and evaluate e_d, our intuitions are

- Adding an IC decomposition interface. The definition of the IC varies among security analysis tasks. Thus, we want to leverage the skills of R2 to achieve an appropriate abstraction level. Considering their abilities, we hide the algorithmic details and design the interface that accepts an IC decomposition function $h : \mathcal{Z} \to \mathcal{D}$. This function should generate the domain-space risk representation $x_d = h(x)$, an IC set of which the size can be variable depending on specific instances (e.g., the unfixed number of functions in malware).
- Adjusting data operations with IC. To move the explanation target of ERDS from $\mathcal{F}$ to $\mathcal{D}$, we switch all feature-space data operations into the domain space. Typically, both the feature perturbation used by $g(\cdot)$ and the feature deduction used by DA should nullify domain-space regions $r_d \in \mathcal{D}$. We also constrain that the nullifying should be meaningful with IC in whatever space, for which we introduce the function $mask(x, r; \mathcal{B})$. It would replace the specified region r on x, i.e., $Ind(x, r)$, with regions sampled from non-risk baseline instances $\mathcal{B}$ (e.g., benign software).

4.2 Overview

Based on the key insights, we propose a framework called FINER to enhance the explanation performance of ERDS.

Architecture. Figure 5 shows the architecture of FINER. It includes two major modules (*M1: explanation-guided model updating* and *M2: task-aware explanation generation*) to minimize approximation error and two auxiliary modules (*M3: explanation quality measurement* and *M4: IC decomposition Interface*) to raise abstraction level. For classification, M1 accepts the DNN model and training data of the original classifier to produce an updated classifier that has high model interpretability. For explanation, M2 uses an extensible FA algorithm set, incorporates application-specific knowledge about IC, and combines multiple results to get an ensemble explainer that generates optimal explanations. Being invoked by the major modules, M3 and M4 constitute IC-based fidelity metrics, which are used in M1 to measure the explanation task for updating and in M2 to assign explanation weights for ensembling.

Roadmap. The remainder of this section proceeds as follows. Section 4.3 (M1) presents the introduction and optimization of the explanation task in fine-tuning; Section 4.4 (M2) introduces the adjustment of data operations and the local weights for different explainers; Section 4.5 (M3) provides the explanation evaluation approaches in the context of ERDS with the new framework. Section 4.6 addresses deployment problems such as user instructions

for the interface (M4) and the applicable scope and functionalities of the framework.

4.3 Explanation-guided Model Updating

This module fine-tunes $f(\cdot)$ with multi-task learning, where the model jointly learns how to classify and how to explain. For the explanation task, the label-based regularization [59] would be impractical considering the challenge of labeling risk explanations, e.g., identifying malicious functions for the whole training dataset. To overcome this problem, we adopt the idea from self-supervised learning [16], which novelly introduces explanations into data augmentation and then regularizes model predictions of the augmented samples. Specifically, given a training batch of data $\mathbf{X}$ and labels $\mathbf{Y}$, three new sets of data are constructed from positive risk samples with a surrogate explainer $g_{train}(\cdot)$. To update model parameters θ, the prediction of the original sample set and the three constructed sample sets are treated as different tasks with different aims.

Data Augmentation. With the current model state fixed, the ROI for a batch would be $\mathbf{R} = \omega(g_{train}(\mathbf{X}); \tau_k)$. Since instances that are predicted as risks by the model have non-zero ROI, we utilize them to create new tasks. Those positive instances can be divided into true positives (TPs) and false positives (FPs). For TPs, the ROI is associated with true risky behaviors, while for FPs, it indicates why the model regards them as outliers from the training data. Therefore, the three new data sets are constructed as follows.

- Sanitized risk set: ROI is nullified in TPs that

$$\mathbf{X_{san}} = \{mask(\mathbf{X}_i, \mathbf{R}_i); f_\theta(\mathbf{X}_i) = 1, \mathbf{Y}_i = 1\}. \tag{2}$$

- Variant risk set: non-ROI in TPs is exchanged that

$$\mathbf{X_{var}} = \{mask(\mathbf{X}_i, \mathbf{X}_i - \mathbf{R}_i); f_\theta(\mathbf{X}_i) = 1, \mathbf{Y}_i = 1\}. \tag{3}$$

- Counter example set: ROI or non-ROI is nullified in FPs that

$$\begin{aligned}\mathbf{X_{cou}} = &\{mask(\mathbf{X}_i, \mathbf{R}_i); f_\theta(\mathbf{X}_i) = 1, \mathbf{Y}_i = 0\} \\ &\cup \{mask(\mathbf{X}_i, \mathbf{X}_i - \mathbf{R}_i); f_\theta(\mathbf{X}_i) = 1, \mathbf{Y}_i = 0\}.\end{aligned} \tag{4}$$

For the function $mask(\cdot)$, the baselines $\mathcal{B}$ are omitted (as well as in the rest of the paper) for brevity, and they would be true negatives among the data batch in the context of fine-tuning. For the surrogate explainer, $g_{train}(\cdot)$ can be implemented with any FA method since all metadata about the classifier is available; we use Gradients that naturally constrain the decision boundary and has the least computational cost. We explain the intuition of the three augmentation approaches with malware classification: the generation of $\mathbf{X_{san}}$ can be regarded as malicious code removal, the generation of $\mathbf{X_{var}}$ can be considered as piggybacking malicious payload into goodware, and the generation of $\mathbf{X_{cou}}$ mimics the process of implementing different benign utilities.

Multi-task Learning. Given the original sample set and the three new sample sets constructed by explanation-guided data augmentation, our aim is to ensure high accuracy on the original classification task and to achieve high fidelity on additional explanation tasks. Specifically, for the explanation tasks, we want to make the prediction probability of the positive class drop for sanitized risk set while holding for the other two, compared with their origins (TPs for $\mathbf{X_{san}}$ and $\mathbf{X_{var}}$, and FPs for $\mathbf{X_{cou}}$). Therefore, multi-task learning would be formulated as the following optimization problem

$$\begin{aligned}&\underset{\theta^*}{\operatorname{argmin}}\ \mathcal{L}_0(\mathbf{X}, f_\theta, \mathbf{Y}) + \lambda_1\mathcal{L}_1(\mathbf{X_{san}}, \mathbf{X}, f_\theta; \mathbf{Y} = \mathbf{1}) \\ &\quad + \lambda_2\mathcal{L}_2(\mathbf{X_{var}}, \mathbf{X}, f_\theta; \mathbf{Y} = \mathbf{1}) + \lambda_3\mathcal{L}_2(\mathbf{X_{cou}}, \mathbf{X}, f_\theta; \mathbf{Y} = \mathbf{0}), \\ &\text{s. t. } \theta^* \subseteq \theta,\end{aligned} \tag{5}$$

where the first item $\mathcal{L}_0$ is the loss function of the original classifier that deals with supervised binary classification (typically cross entropy loss); the last three items measure the relative "drop" and "hold" with $\mathcal{L}_1$ and $\mathcal{L}_2$, and they are respectively balanced by weight coefficients λ_1, λ_2, and λ_3; the constraint means the model parameters are partially unfrozen as θ^*.

Nevertheless, the optimization objective is difficult to minimize due to the measurement of relative values. For one thing, the two items compared are both related to θ^*. For another, the results can have high variance among different instances considering the bias brought by both predictions and explanations. Therefore, as the origins of a constructed sample set belong to the same class, we simplify the problem by classifying sanitized samples as the opposite class while others as their original classes. The last three items in Equation 5 are replaced with:

$$\lambda_1\mathcal{L}_0(\mathbf{X_{san}}, f_\theta, \mathbf{0}) + \lambda_2\mathcal{L}_0(\mathbf{X_{var}}, f_\theta, \mathbf{1}) + \lambda_3\mathcal{L}_0(\mathbf{X_{cou}}, f_\theta, \mathbf{0}). \tag{6}$$

Then, we can use any standard gradient-based optimization method to solve it [34]. To understand the three modified items better, the intuition is that the first two constrain the model's decision-making process with more boundary values and the last one broadens the model's horizon with unexplored normal spaces.

Algorithm 1: Task-aware explanation generation

Data: input x, classifier metadata $\mathcal{M}$, explainer set $\langle \mathcal{G}, \omega \rangle$; transformation functions (h, α, φ), workload expectation k
Result: domain-space explanation e_d.

```
Get I, I with get_ic_indicator(x, h, α, φ)        ▷ IC Indicator
Init w to [0]_{|G|×1}                             ▷ Explainer Weight
Init explanations E to [0]_{|G|×|I|}, current index i to 0
for each FA method g ∈ G do
    if g is black-box then                        ▷ Domain Adjustment
        Get g_I by modifying data operations in g with I
        Get IC attributions e_v' ∈ R^{|I|} with g_I
    else
        Get importance score e_v with g
        Get IC attributions e_v' ∈ R^{|I|} with e_v and I
    Assign normalize(e_v') to E[i]                ▷ Ensemble Explaining
    Get score s by inputting (x_v, e_v', f, k, I) to M3
    Assign s to w[i]; Add 1 to i
Get e_v' ∈ R^{|I|} with E^T · normalize(w)
Get e_d with ω(e_v'; τ_k) defined on I
return e_d
```

4.4 Task-aware Explanation Generation

The explanation generation process with FINER is described in Algorithm 1, which can be divided into two steps. First, domain adjustment customizes individual FA methods in $\mathcal{G}$ to generate

Table 3: Data operations inside different FA methods.

Explainer	Data	Rule
Gradients	$\mathbf{x}_v, \mathcal{A}, \mathcal{O}$	Partial gradients
IG	$\mathbf{x}_v \sim b_v, \mathcal{A}, \mathcal{O}$	Accumulated gradients
DeepLIFT	$\mathbf{x}_v, b_v, \mathcal{A}, \mathcal{O}$	Discrete gradients
LIME	$neighbor(x), \mathcal{O}$	Model coefficients
LEMNA	$neighbor(x), \mathcal{O}$	Model coefficients
Shapley	$\{coalition(x_i); i \leq \|x\|\}, \mathcal{O}$	Averaged differences

IC-based attributions (Lines 5-10). Then, the ensemble explaining leverages the IC-based local fidelity scores to weight different attributions (Lines 11-15).

Compared with the original explanation pipeline in Definition 3, the newly involved data include the transformation functions and the workload expectation. First, the transformation functions (h, α, φ) are used to generate the local *IC indicator* $(\mathbf{I}, \mathcal{I})$ where $\mathbf{I} \in \mathcal{V}$ and $\mathcal{I} \in \mathcal{D}$, which supports domain-space data operations on vector-space inputs. Specifically, the ij-th element of the vector-space indicator object $\mathbf{I}$ maps to the IC index in the domain-space indicator object $\mathcal{I}$ that $(x_v)_{ij}$ belongs to; then when $x \in \mathcal{V}$ and $r \in \mathcal{D}$, the implementation of $Ind(x, r)$ would be checking whether $(\mathbf{I})_{ij}$ exists in r denoted with IC indexes, which would be leveraged by data operations such as the $mask(\cdot)$ function. To highlight the cross-domain data operations, we will add the subscript $\mathbf{I}$ to related functions in this section. Second, the workload expectation $k \in \mathbb{N}^+$ determines the number of ICs that should be nullified for fidelity measurement, and then the local *explainer weight* $\mathbf{w} \in \mathbb{R}^{|\mathcal{G}|}$ can be generated for ensembling.

Domain adjustment. To generate the domain-space attribution vector that has a size of $|\mathcal{I}|$ (Lines 5-10), the adjustment for FA algorithms are divided into two cases in terms of the data operation used in their approximation approaches. As shown in Table 3, gradient-based methods explicitly work in $\mathcal{V}$ where gradients should be calculated for inputs that can propagate in the model neurons, but for perturbation-based methods, vector-space representations are not strictly required by the approximation rules. That is, for LIME and LEMNA, the regression models can accept vectors of arbitrary length and the output size complies with $x_n \in neighbor(x)$; for Shapley, the cooperative game theory uses arithmetic operations that work on any set of coalition predictions and generates attributions with a length of $|\mathbf{X}_c|$ where $\mathbf{X}_c = \{coalition(x_i); i \leq |x|\}$. Therefore, for perturbation-based methods, we switch the neighborhood sampling function and the coalition enumeration function into $\mathcal{D}$ (Line 7), i.e., $neighbor_{\mathbf{I}}(\cdot)$ and $coalition_{\mathbf{I}}(\cdot)$, by (1) performing random sampling and subset permutation with $\mathcal{I}$ to get the IC indexes that should be nullified and (2) sending those indexes to $mask_{\mathbf{I}}(\cdot)$ as the second parameter. For gradient-based methods, we generate the i-th IC attribution with $\|e_v \odot Ind(x_v, \mathbf{I}_{[i]})\|_1$ (Line 10). Note that the step of querying $\mathcal{O}$ (with x_n and $\mathbf{X}_c$) still involves vector-space representations, and that is when we should translate these operations into vector space with the indicator matrix.

Ensemble Explaining. Given multiple IC attributions $e_{v'} \in \mathcal{S} \subset \mathbb{R}^{|\mathcal{I}|}$, we generate their weights $\mathbf{w}$ (Lines 11-13) by evaluating explanation fidelity with the module M3. To balance the effect of different IC attributions $e_{v'}$, we need a metric that indicates higher IC-based fidelity with larger values, for which we define the model prediction drop (MPD) at k as

$$\begin{aligned} &MPD_k(x_v, e_{v'}, f) = \left(f(x_v) - f(x'_v)\right)_{[c=1]}, \\ &\text{s. t. } x'_v = mask_{\mathbf{I}}(x_v, r_d) \text{ where } r_d = \omega(e_{v'}; \tau_k). \end{aligned} \tag{7}$$

Besides the high-is-better property, MPD is different from DA in (1) refined meaning: describing the relative drop by establishing the connection to original predictions (that are different among instances), and (2) modified nullifying: translating IC deduction into vector space instead of setting separate elements to zero (as in Equation 1). Additionally, there are two normalizing steps that act on the IC attribution $e_{v'}$ and the explainer weight $\mathbf{w}$, respectively: the first reduces the variance of attribution distributions among different explainers (Line 11); the second adjusts the weight vector to sum up to one after clipping negative values (Line 14).

4.5 Explanation Quality Measurement

We have introduced the internal usage of M3 in the above sections. The explanation fidelity measurements involved in the two major modules are summarized as: (for M1) the loss item $\mathcal{L}_1$ in Equation 6 that measures how well the *classifier* can predict sanitized samples as the flipped class; (for M2) the MPD in Equation 7 that quantifies how closely an *explainer* can approximate the model prediction locally. In this section, we introduce two scenarios where M3 can be used as a standalone module.

- *Model interpretability* evaluation: to select the model function for a certain risk detection classifier trained with different configurations. It performs IC deduction on a test dataset X with a deduction percentile $p \in \mathbb{R}$ (where the threshold is denoted as τ_p) and a candidate FA algorithm set $\mathcal{G}$. To calculate the model interpretability score, samples are altered with explanations from all possible explainers, and the metric named average model prediction (AMP) is defined as $\frac{1}{|\mathcal{G}||X|} \sum_{i=1}^{|\mathcal{G}|} \sum_{j=1}^{|X|} f(x'_j; x_j, g_i, \tau_p)$.
- *Global fidelity* evaluation: to prioritize better explainers in $\mathcal{G}$ when the task-specific workload $k \in \mathbb{N}^+$ (e.g., the number of functions in the explanation that is acceptable for analyst to read) is known. For the test dataset X, it firstly filters out the samples with an IC size no greater than k, resulting in $X_k = \{x; |\mathcal{I}_x| > k, x \in X\}$. Then, the score for each (ensemble) explainer is the average MPD on remained samples, i.e., $\frac{1}{|X_k|} \sum MPD \circ g(X_k; \tau_k)$.

Different from feature deduction, IC deduction should take the variant IC size into concern. Thus, the percentile-based approach is used for model interpretability evaluation where the downstream explanation task is unknown and all test samples are considered as explanation targets; the traditional number-based approach is used for global fidelity evaluation with the aim of reducing the human workload on samples that are tough for analysis. Typically, users should sample the percentiles or numbers within an acceptable workload range for their evaluation trials (see examples in Figure 7 and Figure 8), to find the best model function or explainer set.

4.6 Framework Deployment

Interface Instructions. As declared in Section 4.1, the interface M4 should receive task-specific implementations of the function $h(\cdot)$, which are essential for intelligibility improvement on those

```
/* Interface Description */
interface IC_Decomposition {
    /* if data-driven classifier,
    better override for intelligibility improvement */
    default public <T> T[] $h$(T $x$) {
        // return $alpha(x)$;
    }
    /* better override for fidelity measurement */
    default public int set_$k$() {
        // return a default (range of) number(s);
    }
    /* directly called by M2 and M3
    to implement the domain-space data operations */
    default <T> int[] get_ic_indicator(T $x$, Callable
     <T> $alpha$, Callable<T> $varphi$) {
        // return the indicator: use the parameters
     and the member function $h$ (Algorithm 2);
    }
    /* queried by M3; then used by M1 and M2 */
    default int get_$k$() {
        // return the output of set_$k$;
    }
}
```

Listing 1: Description of the IC decomposition interface

data-driven classifiers. To implement it, R2 typically needs to provide external tools that can extract ICs from certain types of risks. For example, if IC is defined as the functionality for malware analysis, then to identify function boundaries, Androguard [19] and BinaryNinja [1] can be utilized for App and PE binaries processed by DAMD and DR-VGG, respectively; for vulnerability analysis with VulDeePecker, if IC is defined as tokens including APIs, variables, and operators, then a lexical analysis tool would serve for the decomposition. Precisely defining the IC is beyond the scope of the present paper, and it depends on the context in which R2 wants for their analysis tasks. To be more specific, the local feature space can be abstracted into different collections, and for some collections that are confirmed to be unimportant by experts, they can also be merged together in $h(\cdot)$. As $h(\cdot)$ that indicates the definition of IC, k that depends on the complexity of IC analysis is also task-specific. For instance, inspecting k tokens in source code (e.g., for VulDeePecker) would be much easier than inspecting the same amount of functions in disassembled languages (e.g., for DAMD). To this end, the complete description of the interface can be given as Listing 1.

Applicable Scope. Technically, FINER is a general framework that applies to the formalized classifier and explainer whatever the application is. The main modules would function properly with and without the involvement of R2, i.e., the methods of `$h$` and `set_$k$` in Listing 1 can have default implementations. Nevertheless, if features should be abstracted into different collections for intelligibility improvement, the feature engineering methods should work on independent problem-space objects so that it will apply to the individual IC decomposed by $h(\cdot)$. This is the actual case for the data-driven classifiers that we focus on, e.g., with features extracted from individual tokens, opcodes, and basic blocks. Note that for classifiers with the feature extraction α that is a piece-wise function, for example, defined on different sections of PE files (e.g., `.text`, `.data`, and `.bss`), the proposed IC abstraction method would still work if the α over any interval is data-driven.

Functionality. With FINER, an enhanced ERDS would be made up of the updated classifier and the ensemble explainer, where $f(\cdot)$ and $g(\cdot)$ are optimized towards high-quality explanations for given tasks. Besides the class label and the explanation, it outputs quantitative scores for end users to perceive the explanation quality of ERDS and to determine how much they can rely on the explanations during security analysis. Besides working jointly, different stakeholders can benefit from the decoupled architecture of FINER to accomplish different use cases on their own. For example, R1 can leverage M1 to release a highly interpretable model without the leakage of training data and the concern about downstream explanation methods; R3 can utilize M2 to refine the performance of FA algorithms while task-specific explanation desiderata (abstraction level and maximum workload) is left for R2 to specify.

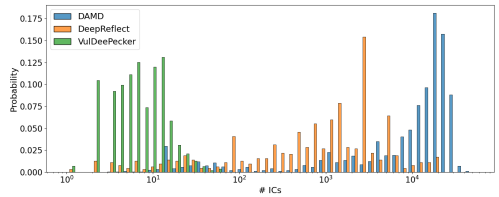


Figure 6: The distribution of IC size for each dataset.

5 SYSTEMATICAL EVALUATION

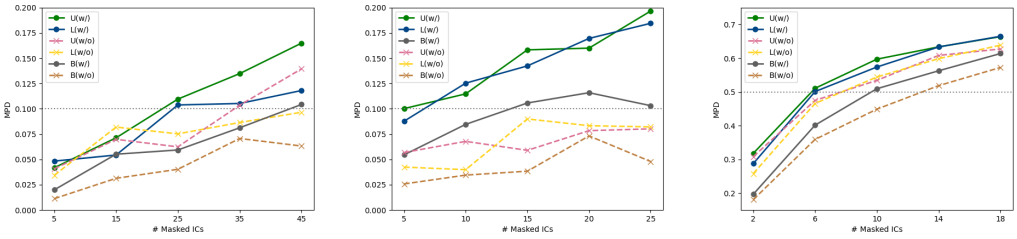
In this section, we quantitatively evaluate the performance of FINER in different ERDS applications. The performance is systematically analyzed in terms of model intelligibility, model accuracy, explanation fidelity, and system efficiency.

5.1 Experimental Setup

Classifier and Dataset. We study the three data-driven classifiers described in Table 1. For DR-VGG, we use the ACFG+ features and the simplified VGG19 model proposed in [21]. For DAMD and VulDeePecker, we use the same implementation of model architectures as [77]. The dataset for vulnerability detection and PE malware detection all follow their original papers, which consist of 23, 307/36, 396 benign/malicious PE files and 29, 313/10, 444 safe/vulnerable code gadgets, respectively. For Android malware detection, since the original dataset is too small and too old, we make efforts to collect a new dataset, consisting of 12, 807/4, 742 benign/malicious Android applications between the year 2017 and 2019. We split each dataset into training and testing sets with 80/20 ratio.

Candidate Explainers. We use all the explainers listed in Table 2. We implement Gradients and IG in accordance to their original papers, and the number of interpolations is set to 64 for IG. We make use of the SHAP toolbox [42] for DeepLIFT and Shapley, where the specific class is called DeepExplainer and PartitionExplainer. For LIME, we use the open-source code from its authors, and for LEMNA, we override LIME's implementation by replacing the surrogate model, where the regression problem with the fussed lasso is solved with CVXPY package [20]. The number of neighborhood samples for these two explainers is set to 1, 000 and the number of mixture models in LEMNA is set to 3.

Enhancement with FINER. For the definition and extraction of IC, we follow the example instructions in Section 4.6. To determine the ROI during data augmentation, we choose the threshold τ_p for each sample according to the top percentile p of its IC attributions, which is 0.2% for DAMD, 2.5% for DR-VGG, and 10% for VulDeePecker. The different choices refer to the IC size distributions of the dataset as in Figure 6: for the two malware classifiers with large IC size,



(a) DAMD (b) DR-VGG (c) VulDeePecker

Figure 7: IC deduction test for ERDS with (w/) and without (w/o) FINER: model prediction drop (MPD) measured at different number of masked ICs. The higher MPD means higher explanation fidelity of the system.

Table 4: Explanation fidelity improvement of M1 for separate explainers: MPD evaluated at a constant value of k, which is 25 for the two malware detection systems and 10 for the vulnerability detection system.

Explainer	DAMD			DR-VGG			VulDeePecker		
	w/o	w/	Improves	w/o	w/	Improves	w/o	w/	Improves
Gradients	0.0985	0.1193	21.09%	0.0770	0.1395	81.03%	0.3553	0.4309	21.28%
IG	**0.1298**	**0.1588**	22.35%	**0.1025**	**0.1865**	82.05%	0.3945	0.4752	20.44%
DeepLIFT	0.1176	0.1320	12.27%	0.0858	0.1514	76.44%	**0.5497**	**0.5719**	4.04%
LIME	**0.1188**	**0.1297**	9.16%	0.0729	0.1077	47.75%	0.3446	0.3596	4.36%
LEMNA	0.0917	0.0981	6.99%	**0.0775**	**0.1111**	43.47%	0.2484	0.2714	9.28%
Shapley	0.1188	0.1266	6.62%	0.0626	0.0916	46.49%	**0.4856**	**0.5003**	3.04%

[1] w/o: the original classifier trained without FINER.
[2] w/: the new classifier updated with FINER.

Table 5: Classifier accuracy before and after model updating.

Metric	DAMD		DR-VGG		VulDeePecker	
	w/o	w/	w/o	w/	w/o	w/
Accuracy	0.983	0.984	0.888	0.904	0.918	0.920
Precision	0.966	0.972	0.888	0.895	0.879	0.876
Recall	0.971	0.968	0.889	0.915	0.797	0.809
F1-score	0.974	0.978	0.888	0.900	0.836	0.841

[1] w/o: the original classifier trained without FINER.
[2] w/: the new classifier updated with FINER.

the percentile results in around 25 functions on average (close to the highlighted function numbers in [21]), and for the vulnerability classifier, it ensures 1 token is returned naturally on average (for the cases where $l * r < 1$, we enforce one IC in the ROI).

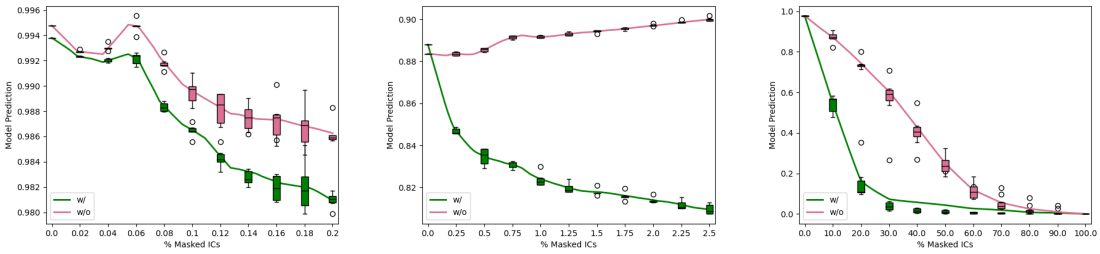
5.2 System Performance Analysis

We first perform end-to-end performance analysis, and then conduct ablation experiments to demonstrate the effectiveness of the two modules, i.e., model updating and explainer ensembling. For each of the three applications, we compare FINER with the baseline system, where the classifier is the original one without updating, and the explainer uses a naive ensembling strategy where multiple explanations are summed up. We also consider three explanation scenarios: (1) Black-box where the three perturbation-based explainers that require no knowledge about $\mathcal{A}$ are used in ensembling; (2) Low-cost where the three gradient-based explainers with lower computational costs are used in ensembling; (3) Unlimited where all the explainers are used in ensembling. Throughout the experiments, we use the evaluation strategies and metrics defined in the measurement module (Section 4.5) to evaluate model interpretability and explanation fidelity.

5.2.1 End-to-end Performance. We show the effectiveness of FINER by performing the global fidelity evaluation with different IC deduction values. The results are illustrated with the k-MPD curve in Figure 7. As in the figure, FINER enhances the explanation fidelity of ERDS in all three applications under different explanation scenarios. We observe that the solid curve always has a higher average MPD score than the dashed curve in its corresponding scenario. The number of ICs that needs to be nullified to achieve an MPD of 0.1 is 22/35, 25/45, 44/∞ for the DAMD system with and without FINER in different scenarios, which means at least 37.1% security analysis workload can be reduced for R2. The enhancement with FINER is the most obvious on DR-VGG, where all solid curves are above the dashed curves. We see that even black-box ensembling of FINER performs better than all the ensembling scenarios without FINER. For VulDeePecker that works on short high-level features, the enhancement is stable in that solid curves are above their corresponding dashed curves at all values of masked IC numbers. For an MPD score at 0.5 where all prediction labels can be flipped, the number of saved manual workloads reaches 24.3% on average.

5.2.2 Effectiveness of Model Updating. To study the effectiveness of model updating (the module M1), we show that model interpretability can be enhanced while there is no trade-off paid to the classification performance. For model interpretability, we provide the r-AMP curve and show the influence on individual explainers at a fixed workload expectation. For classification performance, we use four traditional evaluation metrics.

Model interpretability Enhancement. As the r-AMP curve in Figure 8, there is a clear enhancement for FINER on all classifiers that the green curves drop more sharply than the pink ones. We notice an abnormality that the pink curve for DR-VGG goes upward,



(a) DAMD (b) DR-VGG (c) VulDeePecker

Figure 8: IC deduction test for classifiers with (w/) and without (w/o) M1 updating: AMP measured at different percentiles of masked ICs. The values on the curve are the average risk probabilities of different tests with the six candidate explainers, where their individual values are shown on the box plots. The more AMP drops, the higher the model interpretability is.

Table 6: Explanation fidelity improvement of M2. The experimental setting is consistent with that in Table 4 but the classifier is fixed to the one w/ model updating.

Scenario	DAMD			DR-VGG			VulDeePecker		
	w/o	w/	Improves	w/o	w/	Improves	w/o	w/	Improves
Black-box	0.1235	0.1362	10.30%	0.1309	0.1544	17.99%	0.4068	0.5169	22.72%
Low-cost	0.1525	0.1687	10.58%	**0.1603**	0.1839	14.69%	**0.5571**	0.5895	5.10%
Unlimited	**0.1630**	**0.1784**	9.45%	0.1600	**0.2084**	30.23%	0.5428	**0.5916**	9.28%

[1] w/o: the ensemble explaining without FINER.
[2] w/: the ensemble explaining with FINER.

which means the model without FINER can hardly be explained by a single FA method and most of its classification decisions are based on artifacts. Table 4 shows how the model interpretability acts on the system explanation fidelity for downstream explanation without ensembling. We can see that all FA methods can output higher fidelity explanations when the classifier is updated with M1. The improvement range is 6.62% ∼ 22.35%, 43.47% ∼ 82.05%, 3.04% ∼ 21.28%, respectively, on DAMD, DR-VGG, and VulDeePecker.

Impact on Classification Performance. As shown in Table 5, the common belief that there is a trade-off between model interpretability and classification accuracy do not exist with FINER. For all three classifiers, the accuracy and F1-score metric even slightly improve. Therefore, we can infer that our explanation-guided training strategy actually helps to adjust the model decision boundary, automatically patching some classification mistakes around it.

5.2.3 Effectiveness of Explainer Ensembling. With the classifier fixed to the updated one, we study the effectiveness of our ensembling method in M2 by comparing it with the baseline ensembling.

Explanation Fidelity Enhancement. As the MPD results in Table 6, though both ensembling methods are effective (the best MPD is higher than any explanation produced by a single FA), ensembling w/ FINER clearly has more advancement. Firstly, the explanation fidelity with FINER ensembling is better than the baseline ensembling in all scenarios, with 10.12% to 17.00% higher in MPD. It is interesting to find that, while model updating makes more improvement on gradient-based methods, ensembling is more beneficial to the underprivileged perturbation-based methods. We also observe that, while low-cost ensembling can achieve comparable (and more often even better) results as unlimited ensembling for the baseline method, FINER gathers information more effectively that unlimited explaining is always better than low-cost ensembling, where the greatest advantage is 13.32%. To discuss it system-wide, FINER makes 9.45% ∼ 10.58%, 14.69% ∼ 30.23%, 5.10% ∼ 22.72% improvement on each system. The biggest progress is still on DR-VGG, the pseudo-image-based malware classifier with spatial information whose original model is harder to be explained. For another malware classifier, the improvement is relatively smaller due to the large IC size that has an order of magnitude of 4. For VulDeePecker, although the smaller feature space is intrinsically easier for explanation, FINER is still helpful in that even black-box explaining can achieve an MPD greater than 0.5.

5.3 Analysis of IC Abstraction and Efficiency

5.3.1 Effectiveness of IC Abstraction. The main purpose of IC abstraction is to enhance explanation intelligibility for analysts, which we will introduce case studies and human subjective studies to show the effectiveness. In this subsection, we perform quantitative evaluation to show the influence on the sub-module named domain adjustment. Specifically, since IC is at the same level as the feature for VulDeePecker, we study the classifiers of DAMD and DR-VGG. For ablation experiments, we use the updated classifier and observe individual FA methods.

Explanation Cost Reduction. Firstly, the IC-based adjustment reduces much explanation cost for perturbation-based FA methods. We randomly sample 20 risk instances from the test dataset of DAMD and DR-VGG and analyze the average time used for explaining. As shown in Table 7, for Shapley, explaining the two classifiers with a large feature space can hardly be practical without IC adjustment; for LIME and LEMNA, the time consumption is largely reduced by FINER from 53.43% to 98.58%.

Explanation Fidelity Improvement. Secondly, for gradient-based FA methods where it is practical to calculate feature-level explanations on the whole test dataset, we introduce a naive baseline to observe the MPD change. Specifically, the baseline method uses the standard explanation algorithm to identify a list of important

Table 7: Explanation time for black-box explainers w/ and w/o IC abstraction.

	DAMD			DR-VGG		
	w/o	w/	Red.	w/o	w/	Red.
LIME	4.5E+4	4.7E+3	89.60%	1.3E+3	7.3E+1	94.38%
LEMNA	5.4E+4	7.7E+2	98.58%	3.9E+3	1.8E+3	53.43%
Shapley	N/A	2.1E+3	N/A	N/A	2.3E+2	N/A

[1] w/o: the explainer without IC-based adjustment.
[2] w/: the explainer with IC-based adjustment.
[3] N/A: run out of memory due to the large feature sizes.

Table 8: Gradient-based explainers without IC abstraction. To compare with adjusted explainers, Dec. is the percentage decrease of MPD, and IS measures the explanation similarity.

	DAMD		DR-VGG	
	Dec.	IS	Dec.	IS
Gradients-	56.26%	0.3083	21.16%	0.7841
IG-	45.07%	0.4567	45.68%	0.5239
DeepLIFT-	64.32%	0.3107	25.13%	0.5157

features and picks up ICs that have the most selected features. As shown in Table 8, the mean percentage decrease of MPD (compared with Table 4) is 55.22% and 30.66% for the two systems, showing the abstraction in FINER is effective for improving the fidelity of IC-based explanations. To understand the difference between explanations generated by the baseline and FINER, we also provide the intersection size (IS [77]) in the table. The average IS value of 0.48 suggests that the two explanations are largely different, where less than half of the selected ICs are in common.

5.3.2 System Efficiency Analysis. The major overhead introduced by FINER is in model updating. However, this step costs much less than retraining a new model (intrinsically interpretable but less accurate) and can be performed offline.In our experiments, updating respectively takes 3, 31, and 43 epochs for DAMD, DR-VGG, and VulDeePecker. Actually, FINER can make ERDS more efficient in two aspects. Firstly, as discussed for IC abstraction (Table 7), FINER helps with explanation cost reduction for the black-box scenario, especially when $\mathcal{V}$ has high dimensionality. Secondly, for the low-cost scenario, a less time-consuming FA method on the updated classifier can achieve equal/higher explanation fidelity than a more complex method on the classifier without FINER. For example, the reduced time can be estimated at 0.76 and 3.90 seconds per sample for DAMD and DR-VGG, respectively.

6 TASK EVALUATION WITH GROUND TRUTH

So far, we have validated the fidelity enhancement of FINER on different risk detection systems. Since there exists little ground truth for risk explanation, we perform a case study to show how FINER benefits downstream tasks. In this section, we investigate the effectiveness of FINER on a ground truth dataset for malicious PE functionality localization. We compare DR-VGG enhanced by FINER with two tools proposed in existing work.

Baseline and Dataset. The baselines include (1) DeepReflect: a state-of-the-art tool that uses an unsupervised learning model [58] and identifies malicious components with localized mean-squared-error; (2) DR-IG: the SHAP model in [21] which is FA-based that uses IG to explain the supervised VGG classifier. The ground truth dataset has three malware samples named Rbot, Pegasus, and Carbanak, where each of them respectively has 1, 6, and 6 payload files that are independently classified. The malicious functions inside each file is statically identified by security experts, which makes human-ground explanation labels.

Quantitative Results. We follow the evaluation practice of prior work and use the receiver operating characteristic (ROC) curve to illustrate how the attribution scores match the explanation labels. As shown in Figure 10, FINER outperforms DeepReflect and DR-IG on all malware samples. Through the comparison with DeepReflect, we show that FINER can achieve state-of-the-art performance in malicious functionality detection. By comparing FINER with DR-IG, we prove that our explanation enhancement strategy is still effective in human-grounded evaluation. To take a close look, the local AUC scores for each classification unit is presented in Table 9. We observe that FINER has a rather stable performance among payloads with different sizes and maliciousness ratios. The average increase in AUC is 17.75% and 43.79% when compared with DeepReflect and DR-IG, respectively. We also look into the two exceptional cases where DeepReflect works slightly better, and we find that FINER actually does not fail in the top-k recommendation contest. For Pegasus#idd, both methods return 3 correct predictions out of their top 10 results, and for Carbanak#rdp, FINER actually returns the singleton malicious function quicker (at 11th) than DeepReflect (at 41th). As DeepReflect is an unsupervised tool, users can train it from the beginning when no malware labels are available. However, FINER handles existing state-of-the-art classifiers, and given the fact that binary malware detection is available from many resources [3, 73], it is superior to DeepReflect for higher explanation accuracy and fewer training efforts.

Explanation Visualization. To understand how FINER assists the malware analysis task, we show an example of visualized explanations in Figure 9. As illustrated in the figure, the intelligible explanation is presented as highlighted functions for analysts to inspect the corresponding bytecode addresses. For this example, the malicious payload steals confidential logon data (e.g., Kerberos tickets, WDigest/TsPkg/SSP passwords) through Local Security Authority Subsystem Service (LSASS) memory dumping [9]. Comparing the two explanations, we find that FINER is not only more accurate than DeepReflect, i.e., more malicious functions are identified, but also has much higher confidence in those correct identification, i.e., malicious functions are highlighted with darker colors. Specifically, the two functions that implement the main logic to decrypt and dump data are exactly in FINER's top two recommendations, which means analysts can waste no effort and quickly get an overview of the malicious behavior.

7 RELATED WORK

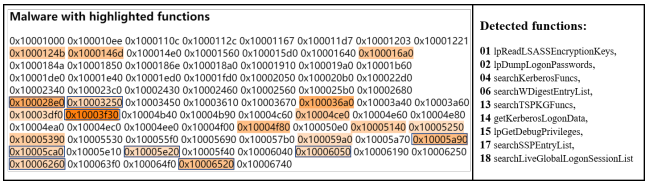
Besides what has been introduced in Section 2, we discuss other related works from three aspects.

Intrinsic and Global XAI Methods. Intrinsic XAI methods are typically achieved by self-interpretable models, such as linear models, decision trees [51], and falling rule lists [74]. Since these methods are less complex, they usually have a lower level of accuracy, making the common belief that model interpretability is at the cost

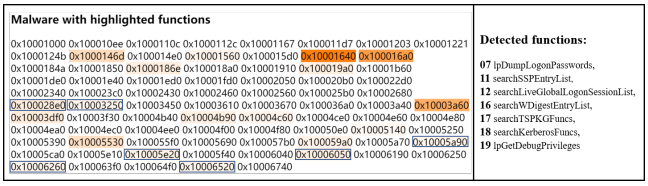
Table 9: Local IC statistics and AUC scores for each malware payload of the ground truth malware samples.

	Rbot	Pegasus						Carbanak					
	#icr	#log	#rep	#idd	#net	#exec	#rse	#auto	#rdp	#cmd	#cve	#bot	#d/l
Mal./Total*	92/440	13/81	15/97	4/98	10/98	6/69	1/93	1/66	1/107	6/804	2/25	44/999	2/234
DeepReflect	0.8429	0.7839	0.7780	**0.9016**	0.6750	0.8942	0.8913	0.6226	**0.9569**	0.7712	0.8016	0.6087	0.6000
DR-IG	0.7887	0.8431	0.7170	0.6029	0.6522	0.7187	0.6772	0.5020	0.8208	0.2759	0.6564	0.5652	0.9692
FINER	**0.8865**	**0.8778**	**0.8390**	0.8761	**0.9814**	**0.9603**	**0.9130**	**0.9231**	0.9057	**0.8358**	**0.9130**	**0.7568**	**0.9828**

* The number of malicious functions vs. the total number of functions inside a payload.



(a) FINER

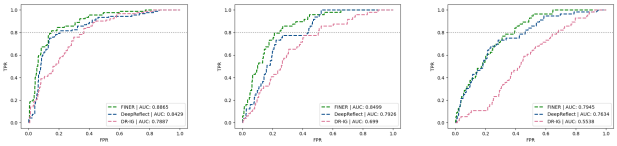


(b) DeepReflect

Figure 9: Function-level malware explanations for Pegasus#log. Important functions are highlighted on the left with their bytecode addresses, and for those match ground truth (marked with blue boxes), the importance rankings and function names are listed on the right.

of accuracy [62]. Global XAI methods explain the overall working mechanism of models with structures or parameters, and a typical example is the attention weights [72]. We choose the post-hoc and instance-level FA methods as a plug-and-play toolset of ERDS. Note that although we update the model, the internal architecture is not changed, and our experiments (as in Section 5.2.2) validate that the interpretability-accuracy trade-off does not exist in FINER.

Regularizing Model with Explanations. Recently, a few studies suggest that training models with explanation constraints help with post-hoc explaining. Some works [36, 59] assume that explanation annotations be available and use them to supervise model gradients, but the assumption can hardly be expected to hold in risk detection scenarios. Others [53, 56, 75, 83] leverage self-supervised learning to regularize unlabelled explanations, which can be divided into two groups: the first focuses on a contrastive setting [63] where explanations are used to generate positive and negative examples for pre-training encoder networks, thus not applicable to off-the-shelf classifiers; the second constrains particular explainer outputs by utilizing some structural priors in data, e.g., [53] imposes consistency on the Grad-CAM heatmap [64] before and after image composition, which do not generalize to different data domain or downstream explainers. Our fine-tuning method also takes inspiration from self-supervised learning. We propose explanation-guided data augmentation in the context of risk detection and novelly introduce the explanation task as classifying augmented samples, making FINER applicable and adaptable to ERDS.



(a) Rbot (b) Pegasus (c) Carbanak

Figure 10: Global ROC curves for the three malware samples with human-annotated explanation labels.

Explanation Applications in Security. Several works have been proposed to leverage explanations for other security purposes, such as vetting malware tags [55], selecting concept drift samples [28, 79], and guiding fairness testing [24, 76]. We focus on the explanation application to provide user assistance in security analysis, and these works are orthogonal to ours, which can be adopted together to develop more powerful security systems.

8 DISCUSSION

Below, we discuss some design choices and potential future work.

New Explanation Methods. We formalize a novel explanation mechanism to tackle the challenging problems of fidelity and intelligibility, where existing FA methods are adaptably used as a toolset. A key observation is that much effort has been made to develop FA methods but their performance in explaining different applications remains uncertain [11]. As our framework is flexible to accommodate new methods, for specific security applications, greater fidelity improvement can be achieved by absorbing more dedicated explanation methods [27].

Normal Sample Explanations. We focus on identifying risky components from abnormal samples due to their importance in risk response. In practical security analysis, experts want to use their limited energy to understand and prevent critical risks [29]. Explaining normal samples would be more helpful for other targets such as model debugging and shift adaptation [13, 28].

Attacks on ERDS. To simplify our main design goal of handling explanation fidelity and intelligibility, we focus on the benign application scenario as explained in Section 3.1. The robustness of ERDS against attacks is an important and complex security aspect. A motivated adversary can perform evasion/backdoor attacks on both the classifier and the explainer [65, 82], which calls for different defense considerations. Future work can investigate the performance of ERDS against these attacks and may benefit from updating the classifier with a trade-off between robustness and classification/explanation accuracy.

Human Subject Evaluation. We use several human-annotated samples and collect feedback from security experts to evaluate the usefulness of malware explanations. Systematic human subject evaluation is challenging for ERDS. Firstly, human annotation requires

domain knowledge and is time-consuming. Secondly, due to the multidisciplinary of ERDS [41], there is little consensus on the participation of humans. Future work can strengthen the evaluation by leveraging user prediction of model output/failure [66] and designing post-study questionnaire [50] for more security applications.

9 CONCLUSION

The black-box property of deep learning-based risk detection classifiers has been a long-standing issue. This paper investigates the emerging field of XAI and proposes an explanation framework, named FINER, to promote these classifiers into explainable risk detection systems. FINER considers "what to explain" in the context of security analysis, "whom to explain for'' as three main stakeholders, and "how to explain" in terms of fidelity and intelligibility. Evaluation results show that FINER is effective to generate high-fidelity intelligible component-based explanations for different security analysis tasks. We hope that this framework and published code can inspire more research efforts on explainable security.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by the National K&D Program of China (No.2020AAA0107705) and the National Natural Science Foundation of China (U20A20178, 62206207).

REFERENCES

[1] Vector 35. 2023. Binary Ninja. https://binary.ninja/.
[2] Yousra Aafer, Wenliang Du, and Heng Yin. 2013. Droidapiminer: Mining api-level features for robust malware detection in android. In *International conference on security and privacy in communication systems*. Springer, 86–103.
[3] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories* (Austin, Texas) *(MSR '16)*. ACM, 468–471. https://doi.org/10.1145/2901739.2903508
[4] Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. 2020. Evaluating saliency map explanations for convolutional neural networks: a user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. 275–285.
[5] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2017. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104* (2017).
[6] Hyrum S Anderson and Phil Roth. 2018. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637* (2018).
[7] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and don'ts of machine learning in computer security. In *Proc. of the USENIX Security Symposium*.
[8] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. 2014. Drebin: Effective and explainable detection of android malware in your pocket. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, Vol. 14. 23–26.
[9] MITRE ATT&CK. 2020. OS Credential Dumping: LSASS Memory. https://attack.mitre.org/techniques/T1003/001/.
[10] AV-ATLAS. 2023. New malware. https://portal.av-atlas.org/.
[11] Mohamed Karim Belaid, Eyke Hüllermeier, Maximilian Rabus, and Ralf Krestel. 2022. Do We Need Another Explainable AI Method? Toward Unifying Post-hoc XAI Evaluation Methods into an Interactive and Multi-dimensional Benchmark. *arXiv preprint arXiv:2207.14160* (2022).
[12] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. 2020. Explainable machine learning in deployment. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 648–657.
[13] Michael Cao, Sahar Badihi, Khaled Ahmed, Peiyu Xiong, and Julia Rubin. 2020. On benign features in malware detection. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 1234–1238.
[14] Sicong Cao, Xiaobing Sun, Lili Bo, Ying Wei, and Bin Li. 2021. Bgnn4vd: constructing bidirectional graph neural-network for vulnerability detection. *Information and Software Technology* 136 (2021), 106576.
[15] Bryan Casey, Ashkon Farhangi, and Roland Vogl. 2019. Rethinking Explainable Machines: The GDPR's' Right to Explanation'Debate and the Rise of Algorithmic Audits in Enterprise. *Berkeley Tech. LJ* 34 (2019), 143.
[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.
[17] Binlin Cheng, Jiang Ming, Jianmin Fu, Guojun Peng, Ting Chen, Xiaosong Zhang, and Jean-Yves Marion. 2018. Towards paving the way for large-scale windows malware analysis: Generic binary unpacking with orders-of-magnitude performance boost. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 395–411.
[18] Piotr Dabkowski and Yarin Gal. 2017. Real time image saliency for black box classifiers. *Advances in neural information processing systems* 30 (2017).
[19] Anthony Desnos. 2023. Androguard. https://github.com/androguard/androguard.
[20] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research* 17, 1 (2016), 2909–2913.
[21] Evan Downing, Yisroel Mirsky, Kyuhong Park, and Wenke Lee. 2021. DeepReflect: Discovering Malicious Functionality through Binary Reconstruction. In *30th USENIX Security Symposium (USENIX Security 21)*.
[22] Mengnan Du, Ninghao Liu, and Xia Hu. 2019. Techniques for interpretable machine learning. *Commun. ACM* 63, 1 (2019), 68–77.
[23] Wisam Elmasry, Akhan Akbulut, and Abdul Halim Zaim. 2020. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. *Computer Networks* 168 (2020), 107042.
[24] Ming Fan, Wenying Wei, Wuxia Jin, Zijiang Yang, and Ting Liu. 2022. Explanation-guided fairness testing through genetic algorithm. In *Proceedings of the 44th International Conference on Software Engineering*. 871–882.
[25] Sunanda Gamage and Jagath Samarabandu. 2020. Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications* 169 (2020), 102767.
[26] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. 2019. XAI—Explainable artificial intelligence. *Science robotics* 4, 37 (2019), eaay7120.
[27] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 364–379.
[28] Dongqi Han, Zhiliang Wang, Wenqi Chen, Kai Wang, Rui Yu, Su Wang, Han Zhang, Zhihua Wang, Minghui Jin, Jiahai Yang, et al. 2023. Anomaly Detection in the Open World: Normality Shift Detection, Explanation, and Adaptation. (2023).
[29] Dongqi Han, Zhiliang Wang, Wenqi Chen, Ying Zhong, Su Wang, Han Zhang, Jiahai Yang, Xingang Shi, and Xia Yin. 2021. DeepAID: Interpreting and Improving Deep Learning-Based Anomaly Detection in Security Applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 3197–3217. https://doi.org/10.1145/3460120.3484589
[30] Yiling He, Yiping Liu, Lei Wu, Ziqi Yang, Kui Ren, and Zhan Qin. 2022. MsDroid: Identifying Malicious Snippets for Android Malware Detection. *IEEE Transactions on Dependable and Secure Computing* (2022).
[31] Jeya Vikranth Jeyakumar, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani Srivastava. 2020. How can i explain this to you? an empirical study of deep neural network explanation methods. *Advances in Neural Information Processing Systems* 33 (2020), 4211–4222.
[32] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*. PMLR, 2668–2677.
[33] TaeGuen Kim, BooJoong Kang, Mina Rho, Sakir Sezer, and Eul Gyu Im. 2018. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security* 14, 3 (2018), 773–788.
[34] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[35] Deguang Kong and Guanhua Yan. 2013. Discriminant malware distance learning on structural information for automated malware classification. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1357–1365.
[36] Kunpeng Li, Ziyan Wu, Kuan-Chuan Peng, Jan Ernst, and Yun Fu. 2018. Tell me where to look: Guided attention inference network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9215–9223.
[37] Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. 2018. Vuldeepecker: A deep learning-based system for vulnerability detection. *arXiv preprint arXiv:1801.01681* (2018).

[38] Q Vera Liao and Kush R Varshney. 2021. Human-centered explainable ai (xai): From algorithms to user experiences. *arXiv preprint arXiv:2110.10790* (2021).

[39] Guanjun Lin, Sheng Wen, Qing-Long Han, Jun Zhang, and Yang Xiang. 2020. Software vulnerability detection using deep neural networks: a survey. *Proc. IEEE* 108, 10 (2020), 1825–1848.

[40] Shigang Liu, Guanjun Lin, Qing-Long Han, Sheng Wen, Jun Zhang, and Yang Xiang. 2019. DeepBalance: Deep-learning and fuzzy oversampling for vulnerability detection. *IEEE Transactions on Fuzzy Systems* 28, 7 (2019), 1329–1343.

[41] Pedro Lopes, Eduardo Silva, Cristiana Braga, Tiago Oliveira, and Luís Rosado. 2022. XAI Systems Evaluation: A Review of Human and Computer-Centred Methods. *Applied Sciences* 12, 19 (2022), 9423.

[42] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[43] Daniel D Lundstrom, Tianjian Huang, and Meisam Razaviyayn. 2022. A rigorous study of integrated gradients method and extensions to internal neuron attributions. In *International Conference on Machine Learning*. PMLR, 14485–14508.

[44] Antonios Mamalakis, Elizabeth A Barnes, and Imme Ebert-Uphoff. 2022. Carefully choose the baseline: Lessons learned from applying XAI attribution methods for regression tasks in geoscience. *Artificial Intelligence for the Earth Systems* (2022), 1–18.

[45] Shraddha Mane and Dattaraj Rao. 2021. Explaining Network Intrusion Detection System Using Explainable AI Framework. https://doi.org/10.48550/ARXIV.2103.07110

[46] Alessandro Mantovani, Simone Aonzo, Yanick Fratantonio, and Davide Balzarotti. 2022. RE-Mind: a First Look Inside the Mind of a Reverse Engineer. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, Kevin R. B. Butler and Kurt Thomas (Eds.). USENIX Association, 2727–2745.

[47] marcotcr. 2023. LIME. https://github.com/marcotcr/lime.

[48] E Mariconti, L Onwuzurike, P Andriotis, E De Cristofaro, G Ross, and G Stringhini. 2017. MamaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*.

[49] Niall McLaughlin, Jesus Martinez del Rincon, BooJoong Kang, Suleiman Yerima, Paul Miller, Sakir Sezer, Yeganeh Safaei, Erik Trickel, Ziming Zhao, Adam Doupé, et al. 2017. Deep android malware detection. In *Proceedings of the seventh ACM on conference on data and application security and privacy*. 301–308.

[50] Mahsan Nourani, Samia Kabir, Sina Mohseni, and Eric D Ragan. 2019. The effects of meaningful and meaningless explanations on trust and perceived system accuracy in intelligent systems. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, Vol. 7. 97–105.

[51] Sebastian Nowozin, Carsten Rother, Shai Bagon, Toby Sharp, Bangpeng Yao, and Pushmeet Kohli. 2011. Decision tree fields. In *2011 International Conference on Computer Vision*. IEEE, 1668–1675.

[52] Islam Obaidat, Meera Sridhar, Khue M Pham, and Phu H Phung. 2022. Jadeite: A novel image-behavior-based approach for java malware detection using deep learning. *Computers & Security* 113 (2022), 102547.

[53] Vipin Pillai and Hamed Pirsiavash. 2021. Explainable models with consistent interpretations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 2431–2439.

[54] Lukas Pirch, Alexander Warnecke, Christian Wressnegger, and Konrad Rieck. 2021. TagVet: Vetting Malware Tags Using Explainable Machine Learning *(EuroSec '21)*. Association for Computing Machinery, New York, NY, USA, 7 pages. https://doi.org/10.1145/3447852.3458719

[55] Lukas Pirch, Alexander Warnecke, Christian Wressnegger, and Konrad Rieck. 2021. Tagvet: Vetting malware tags using explainable machine learning. In *Proceedings of the 14th European Workshop on Systems Security*. 34–40.

[56] Gregory Plumb, Maruan Al-Shedivat, Ángel Alexander Cabrera, Adam Perer, Eric Xing, and Ameet Talwalkar. 2020. Regularizing black-box models for improved interpretability. *Advances in Neural Information Processing Systems* 33 (2020), 10526–10536.

[57] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. 'Why should i trust you?' Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[58] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[59] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2662–2670.

[60] Sankardas Roy, Jordan DeLoach, Yuping Li, Nic Herndon, Doina Caragea, Xinming Ou, Venkatesh Prasad Ranganath, Hongmin Li, and Nicolais Guevara. 2015. Experimental study with real-world data for android app security analysis using machine learning. In *Proceedings of the 31st Annual Computer Security Applications Conference*. 81–90.

[61] Waddah Saeed and Christian Omlin. 2021. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *arXiv preprint arXiv:2111.06420* (2021).

[62] Sagar Samtani, Hsinchun Chen, Murat Kantarcioglu, and Bhavani Thuraisingham. 2022. Explainable Artificial Intelligence for Cyber Threat Intelligence (XAI-CTI). *IEEE Transactions on Dependable and Secure Computing* 19, 04 (2022), 2149–2150.

[63] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[64] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[65] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. {Explanation-Guided} Backdoor Poisoning Attacks Against Malware Classifiers. In *30th USENIX Security Symposium (USENIX Security 21)*. 1487–1504.

[66] Hua Shen and Ting-Hao Huang. 2020. How useful are the machine-generated interpretations to general users? A human evaluation on guessing the incorrectly predicted labels. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, Vol. 8. 168–172.

[67] Eui Chul Richard Shin, Dawn Song, and Reza Moazzezi. 2015. Recognizing functions in binaries with neural networks. In *24th USENIX security symposium (USENIX Security 15)*. 611–626.

[68] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*. PMLR, 3145–3153.

[69] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In Workshop at International Conference on Learning Representations*. Citeseer.

[70] Charles Smutz and Angelos Stavrou. 2012. Malicious PDF detection using metadata and structural features. In *Proceedings of the 28th annual computer security applications conference*. 239–248.

[71] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*. PMLR, 3319–3328.

[72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[73] VirusTotal. 2023. VirusTotal. https://virustotal.com/.

[74] Fulton Wang and Cynthia Rudin. 2015. Falling rule lists. In *Artificial intelligence and statistics*. PMLR, 1013–1022.

[75] Lei Wang, Ee-Peng Lim, Zhiwei Liu, and Tianxiang Zhao. 2022. Explanation guided contrastive learning for sequential recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2017–2027.

[76] Zichong Wang, Yang Zhou, Meikang Qiu, Israat Haque, Laura Brown, Yi He, Jianwu Wang, David Lo, and Wenbin Zhang. 2023. Towards Fair Machine Learning Software: Understanding and Addressing Model Bias Through Counterfactual Thinking. *arXiv preprint arXiv:2302.08018* (2023).

[77] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. 2020. Evaluating explanation methods for deep learning in security. In *2020 IEEE european symposium on security and privacy (EuroS&P)*. IEEE, 158–174.

[78] Feng Wei, Hongda Li, Ziming Zhao, and Hongxin Hu. 2023. XNIDS: Explaining Deep Learning-based Network Intrusion Detection Systems for Active Intrusion Responses. (2023).

[79] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. 2021. CADE: Detecting and Explaining Concept Drift Samples for Security Applications. In *30th USENIX Security Symposium (USENIX Security 21)*.

[80] Chih-Kuan Yeh, Kuan-Yun Lee, Frederick Liu, and Pradeep Ravikumar. 2022. Threading the Needle of On and Off-Manifold Value Functions for Shapley Explanations. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1485–1502.

[81] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. 2017. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 39–49.

[82] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. 2020. Interpretable deep learning under fire. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*.

[83] Zhibo Zhang, Jongseong Jang, Chiheb Trabelsi, Ruiwen Li, Scott Sanner, Yeonjeong Jeong, and Dongsub Shim. 2021. ExCon: Explanation-driven supervised contrastive learning for image classification. *arXiv preprint arXiv:2111.14271* (2021).

[84] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2921–2929.

[85] Yajin Zhou and Xuxian Jiang. 2012. Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy*. IEEE, 95–109.