ZeroDetect: Boundary-Aware Generative Modeling for Zero-Day Malware Detection

Aryan Singh
Poolesville High School
Poolesville, MD, USA

Mark Stamp
Department of Computer Science
San Jose State University
San Jose, California, USA

***Abstract*—Zero-day malware poses a critical challenge to modern cybersecurity systems because it exploits previously unseen vulnerabilities and evades signature-based detection. While machine learning approaches have improved malware classification accuracy, many models degrade significantly under mutation and obfuscation, which are real-world effects of zero-day malware. This work introduces ZeroDetect, a malware detection framework leveraging boundary-aware generative modeling. ZeroDetect improves robustness against zero-day threats through boundary-aware training and controlled perturbations. Two datasets are aggregated for the training and evaluation of the model, each having a large number of malware and benign samples. An autoencoder is trained on the malware samples to learn compact latent representations capturing malicious characteristics, which are then fed into a denoising diffusion implicit model (DDIM) for synthetic malware generation. An XGBoost classifier is trained on a dataset augmented with synthetic malware for strengthened classification. A boundary-aware perturbation mechanism is then applied in latent space to simulate realistic mutation, enabling evaluation on robustness. Under progressively increasing latent perturbation strength, the baseline classifier's recall declines from 0.989 to 0.514, whereas ZeroDetect maintains recall above 0.91 at extreme perturbation levels, which demonstrates how ZeroDetect improves generalization under simulated zero-day conditions. By reframing malware detection as a robustness problem, ZeroDetect provides a stronger approach to defending against evolving cyber threats.**

***Index Terms*—Network Intrusion Detection, Diffusion Models, Adversarial Detection, OOD Detection**

I. Introduction

Cybersecurity is becoming an ever-increasing problem for the world's digital operations. At the heart of that problem is malicious software, also known as malware, which is any software that harms or otherwise adversely affects a computer. Malware attacks have been increasing in both frequency and severity as the evolution of malware has outpaced traditional detection systems, with malware authors constantly creating novel malware to avoid detection called zero-day malware.

Zero-day malware refers to malicious software from previously unseen families with behavior that has not been previously observed. These threats rely on exploiting innovative methods of attacking a system using vulnerabilities that haven't been seen by security teams. Detecting such threats requires more than memorizing known patterns, it necessitates understanding the underlying semantics of malicious execution.

Traditional signature-based malware detectors fail on zero-day threats due to signatures not being able to anticipate unseen variants. As a result, malware detection systems are becoming increasingly reliant on machine learning models trained on large datasets of known malicious and benign software. While these systems achieve high accuracy on standard benchmarks, the underlying assumption for these detection systems is that future malware will resemble previously observed samples, which is flawed. Real-world attackers actively design malware to evade existing detection mechanisms, and zero-day malware is the epitome of this as it is novel malware from previously unseen families or malware designed with novel obfuscation strategies.

The detection of zero-day malware should be reframed as a robustness problem, as the insidious nature of zero-day malware comes as a result of evasion techniques and the overall evolution of malware. Therefore, the gap in current research is the absence of evaluation against a simulated evolution of malware strains.

ZeroDetect addresses this gap by creating a malware detection framework that specifically targets boundary-challenging samples and measures the performance of a detection model under increasing amounts of adversarial perturbations. This makes the detection model resistant to evasion techniques and therefore more effective in detecting zero-day threats.

While purely computational for now, the implementation of ZeroDetect's improved zero-day detection can strengthen cybersecurity protection for individuals, companies, and critical infrastructure systems. This ultimately reduces economic loss, lowers downtime, and prevents breaches of sensitive information.

II. Background

A. Autoencoders

Autoencoders are neural architectures designed to learn compact representations of input data by minimizing reconstruction error between the original input and its decoded output. Formally, an encoder f_θ maps input x to a latent vector z, while a decoder g_ϕ reconstructs x' from z. By constraining the dimensionality of z, the model is forced to capture salient structure in the data.

Autoencoders learn compressed representations by minimizing the reconstruction loss function:

$$L(\theta, \varphi) = \frac{1}{n}\sum_{i=1}^{n}(x^i - f_\theta(g_\varphi(x^i)))^2 \tag{1}$$

where f_θ is the encoder and g_φ is the decoder.

In malware analysis, autoencoders have been employed for dimensionality reduction, anomaly detection, and feature learning. Their ability to extract abstract representations makes them well-suited for modeling execution behavior, particularly when raw features are high-dimensional or sparse.

In ZeroDetect, we use an autoencoder to learn latent embeddings of opcode-derived feature vectors. Rather than directly classifying raw n-grams, we first compress them into a lower-dimensional space that captures shared characteristics across malware families. This latent representation serves as the foundation for downstream classification and robustness analysis.

B. XGBoost Classifier

To perform final classification, we employ XGBoost, a gradient-boosted decision tree ensemble known for its strong performance on structured data. XGBoost iteratively builds trees to minimize a regularized objective function, offering robustness to noise and the ability to model nonlinear interactions between features.

Given the structured nature of our latent representations, XGBoost provides an effective balance between expressiveness and interpretability. It also enables rapid experimentation and feature importance analysis, which aids in understanding model behavior under perturbation.

C. Diffusion Models

Diffusion models have recently gained attention for their ability to model complex distributions via iterative noise injection and denoising processes. Although not directly used in ZeroDetect's current pipeline, diffusion-based robustness research motivates our perturbation strategy by emphasizing how distributional changes can be amplified through controlled transformations.

Our work adopts a similar philosophical stance: zero-day malware should be evaluated through progressive distortion rather than static test splits. While ZeroDetect currently implements perturbations directly in feature space, diffusion-based approaches represent a promising future extension for generative robustness testing.

III. Related Works

Machine learning–based malware detection has explored a wide range of feature modalities, including raw bytes [1], API call graphs [8], control-flow graphs [9], and opcode sequences [10]. Opcode-based methods offer a middle ground between low-level bytes and high-level semantics, capturing execution behavior while remaining computationally tractable.

Deep learning approaches such as CNNs and RNNs have demonstrated strong performance on benchmark datasets [2,3], but their reliance on static distributions limits real-world applicability. Several studies have highlighted how minor obfuscations can significantly degrade detection accuracy [11].

Zero-day detection is often framed as out-of-distribution (OOD) detection or novelty detection [12]. Techniques such as reconstruction error, entropy scoring, and energy-based methods have been proposed, yet many struggle to distinguish between benign anomalies and truly malicious behavior.

More recent work emphasizes robustness to adversarial perturbations [13], though most evaluations remain confined to synthetic attacks rather than realistic malware evolution. Crucially, existing literature rarely measures how detection performance degrades as malware progressively mutates—a gap ZeroDetect directly addresses.

IV. Prodecures

A. Dataset Preparation

Collect malware and benign ".exe" samples from MalwareBazaar and public repositories. Create a bash script that uses the "objdump" linux program to disassemble every ".exe" file into a ".txt" file containing the opcode sequence of the program. Load the opcode database into Google Colab and tokenize each opcode sequence using scikit-learn's HashingVectorizer. Tokenize into word tokens and produce hashing n-gram features ($ngram = (1, 3), n_{features} = 2^{14}$) to avoid vocabulary leakage.

For each tokenized opcode sequence, compute semantic features like sequence length, entropy, unique ratio, and top-5 average frequency. Partition this dataset with opcode feature vectors into a training set, a validation set, and a zero-day test set with samples of a malware family not in the training or validation set in order to perform a family-holdout evaluation.

B. Latent Representation Learning

Train a deep autoencoder on malware opcode feature vectors to learn compact latent embeddings. Use pytorch DataLoader, mixed precision training, and implement early-stopping criteria. Make sure the model is optimized using reconstruction loss to ensure that the latent space captures meaningful structural information.

Extract latent representations from the bottleneck layer into latent space Z_{mal} and train a denoising diffusion model that gravitates towards the decision boundary within this latent space to learn the data distribution and generate synthetic, boundary-challenging samples. Ensure the synthetic latents are virtually indistinguishable from real latents by checking for an accuracy greater than 99% and a UMAP plot that shows synthetic malware points overlapping real malware.

C. Classifier Training

Build several training conditions:

- Baseline: XGBoost on real training data only
- Augmented: XGBoost on real + synthetic boundary malware
- Mixed: Real + synthetic + real benign

Evaluate on held-out families (family-holdout design) and on perturbed test sets. Establish baseline and proposed performance comparisons. Use the classification metrics of accuracy, precision, recall, F1-score, and ROC-AUC, which should be computed for all models.

D. Robustness Evaluation

Apply systematic perturbations to hold-out malware. Inject random smoothed gaussian noise in the latent space at varying rates with an α parameter. Measure evaluation metrics vs perturbation strength (e.g. recall vs α) for all models.

V. Data Analysis

Performance metrics include accuracy, ROC-AUC, confusion matrices, and recall under increasing perturbation intensity. Robustness curves are generated by measuring recall as a function of mutation parameter α.

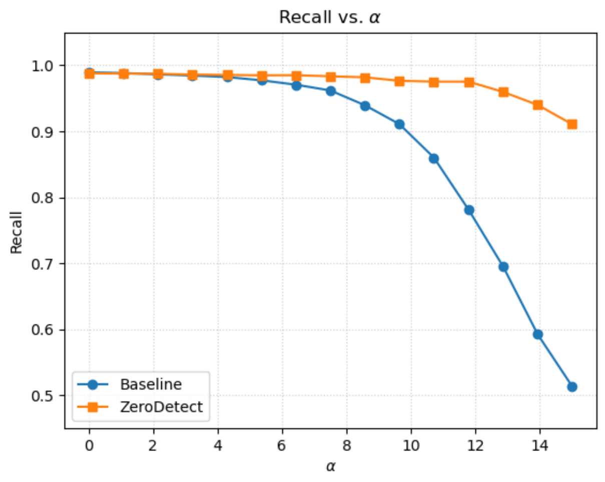

Fig. 1. Recall vs. α comparison between baseline and ZeroDetect. Noticable dropoff in baseline detection performance.

Comparative analysis focuses on degradation rate. For example, baseline recall decreased from approximately 0.989 at $\alpha = 0$ to 0.513 at $\alpha = 15$, whereas the perturbation-aware model maintained recall above 0.91 under the same extreme conditions. The slope of degradation serves as the primary robustness metric.

Latent space structure is analyzed using UMAP visualization to examine separation between benign samples, real malware, and perturbed zero-day variants. Statistical comparisons assess whether robustness improvements are significant.