VAPD: An Anomaly Detection Model for PDF Malware Forensics with Adversarial Robustness

Side Liu[1], Jiang Ming[2], Yilin Zhou[1], Jianming Fu[1], Guojun Peng[1]*
[1] *Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University,*
[2] *Department of Computer Science, School of Science and Engineering, Tulane University*

Abstract

Malicious PDFs are a prevalent threat in the modern web security landscape, often used as attack vectors in phishing campaigns and other web application attacks. With the widespread integration of PDF readers in browsers, malicious PDFs exploit vulnerabilities in web applications and browsers, posing significant risks. Despite advances in machine learning for malware detection, existing PDF classifiers struggle with adversarial attacks, where minor modifications to malicious files evade detection and lead to serious consequences like ransomware or data breaches.

In this paper, we propose *VAPD*, an anomaly detection model based on reconstruction with dual forensics objectives: 1) identifying PDF malware through the reconstruction error between input and output, and 2) pinpointing anomalous regions. We strategically leverage the notion that a model exclusively trained on benign samples struggles to reconstruct malicious counterparts, thereby yielding amplified reconstruction errors. We have evaluated VAPD on multiple datasets, including real-world Advanced Persistent Threat samples, achieving an accuracy rate of 99.54% that stands out among existing anomaly detection methods. Moreover, we measure the robustness of VAPD by utilizing four adversarial attack frameworks in both feature and problem spaces. Our findings demonstrate that our model exhibits superior robustness performance when compared to the state-of-the-art work. Notably, we achieve this level of performance at a significantly lower training cost, which is equivalent to only 4.8% of the state-of-the-art work. Additionally, VAPD offers an advanced localization capability that outperform signature-based tools.

1 Introduction

The Portable Document Format (PDF) is one of the most widely used file formats on the web [1], making it an attractive target for cybercriminals due to its flexibility and ubiquity [2–7]. PDFs are the most common type of malicious attachments in phishing emails [6], with approximately 70% evading network defenses and 15% bypassing endpoint protections [8]. The integration of PDF readers into modern browsers has further amplified these risks, enabling attackers to exploit vulnerabilities in web applications and browsers to execute malicious actions, such as cross-site scripting (XSS) or embedding harmful code [9, 10].

Machine learning-based classifiers have demonstrated high accuracy on baseline datasets in detecting PDF malware [11–19]. However, these methods often operate as binary classifiers, merely labeling samples as "benign" or "malicious", without identifying the specific anomalous regions within a PDF. This restricts their utility in forensic analysis, where pinpointing the exact sources of anomalies is critical for rapid threat mitigation. Furthermore, these models are highly susceptible to adversarial attacks, where small, crafted modifications to malicious PDFs can lead to misclassification [17, 20]. Such adversarial samples can bypass detection systems, potentially enabling ransomware attacks or data breaches.

Adversarial training has emerged as a popular approach to enhancing the robustness of machine learning models. While effective, it comes with significant trade-offs [21, 22], including high computational costs, increased false positive rates, and reliance on generating high-quality adversarial samples [23–25]. For instance, Evans et al. [24] retrained a PDF classifier with adversarial samples, resulting in a false positive rate as high as 85%. Similarly, Grosse et al. [23] retrained the Android malware classifier using adversarial malware examples, leading to a false positive rate of 67%. Additionally, adversarial training necessitates the creation of high-quality adversarial malware samples, resulting in high training costs. In a previous study conducted by Lucas et al. [26], it was found that the generation of an adversarial malware variant took an average of approximately 4,424 seconds. Additionally, a single adversarial training cycle required weeks to complete. The recent follow-up work [21] estimated that this adversarial training method for comprehensive training on their dataset

*Corresponding author.

would take as many as 31 years! Chen et al. [17] employed symbolic interval analysis and manual-defined properties for robust training, which circumvented the time-consuming generation of high-quality adversarial samples. However, the computational complexity associated with this training method is relatively high, rendering it suitable only for small-scale neural networks with a few hidden layers [27]. Manually defining training properties poses challenges in applying this approach to diverse security scenarios and complex neural networks. These limitations restrict the scalability and practicality of adversarial training in real-world applications.

Reconstruction-based anomaly detection presents a promising alternative for overcoming these challenges. In cybersecurity, malware and threats are inherently anomalies relative to the dominant benign samples [28]. This approach leverages the relative abundance of benign samples, making it effective in identifying unknown threats without requiring large datasets of diverse malicious samples. Encoder-decoder models, particularly autoencoders (AEs), are widely used for reconstruction-based anomaly detection [29]. Recent studies have demonstrated that AEs not only exhibit inherent resilience to adversarial attacks [30] but also provide a capability of pinpointing specific anomalous regions through localized reconstruction errors [31–33]. This ability to highlight regions with higher reconstruction errors enhances the interpretability and forensic utility of the detection process. However, the potential of reconstruction-based anomaly detection has yet to be fully explored in the domain of PDF malware detection.

To address the limitations of existing methods, we propose VAPD, a novel framework for anomaly detection and forensic analysis based on encoder-decoder architecture and reconstruction. We first redesigned feature extraction approach [15] for VAPD, incorporating a novel path-object mapping table. This enhancement enables quick identification and extraction of anomalous regions within PDFs, a capability absent in previous studies. By providing detailed insights into specific malicious regions, this mechanism allows security analysts to swiftly identify malicious regions, facilitating the rapid generation of malware signatures, updating of defense systems, and prevention of future attacks. In addition, we utilize reconstruction errors as a core signal for identifying malicious PDFs and develop a variational autoencoder (VAE) as reconstructor. Trained exclusively on benign samples, VAPD learns to accurately reconstruct benign features while struggling with malicious or adversarial samples due to their inherent anomalies. This approach effectively eliminates the need for adversarial training, which is computationally expensive and prone to high false positive rates [23, 24]. Furthermore, the integration of the path-object mapping mechanism enhances interpretability by directly linking anomalous reconstruction errors to specific PDF regions, enabling analysts to focus on critical areas with actionable insights.

This paper introduces VAPD as a cost-effective and robust solution for PDF malware forensic analysis. On a baseline dataset, VAPD achieves an accuracy of 99.54%, surpassing existing anomaly detection methods. Its robustness against adversarial attacks is particularly noteworthy. We conducted four types of adversarial attacks, including both white-box and black-box scenarios, covering feature space and sample space. The experimental results demonstrate that VAPD outperforms existing state-of-the-art classifier [17] under all types of attacks. For example, under MalGAN [34], a generative adversarial network-based feature space attack, VAPD achieves a 100% resistance. Against adaptive EvadeML [35, 36], an advanced sample space attack utilizing genetic algorithms, VAPD limits the evasion rate to just 6.01%, compared to over 50% when targeting the state-of-the-art classifier. Notably, we achieve this level of performance at a significantly lower training cost, which is equivalent to only 4.8% of the state-of-the-art work. Additionally, VAPD can locate the anomalous regions of a sample within 0.01 seconds while producing its output, making it a practical choice for large-scale deployments. In localization evaluations, it achieved recall of 81.8% at the document level and 74.5% at the object level, surpassing signature-based analysis tool [37]. These results underscore VAPD's ability to combine accuracy, robustness, and cost-efficiency, offering a promising approach to PDF malware detection and forensic analysis. In a nutshell, we make the following contributions:

- We investigate various anomaly detection techniques for PDF malware detection and introduce VAPD, a novel reconstruction-based model that not only detects malicious PDFs but also automatically identifies anomalous objects. VAPD achieves high accuracy and robustness while maintaining low training overhead.
- We propose a new PDF feature, the path-object mapping table, which, when combined with VAPD's reconstruction capabilities, enables rapid identification and localization of anomalies within PDFs. This capability has not been achieved in previous machine learning-based approaches.
- We rigorously evaluated our system against both adversarial samples and real-world instances of PDF malware. Our experimental results demonstrate that VAPD is highly effective in combating PDF malware.

Open Source We release a prototype of VAPD and datasets to facilitate reproduction and reuse, as all found at Zenodo.

2 Background, Related Work, and Motivation

We first provide background information on PDF format and PDF malware. Later, we review existing work on detecting PDF malware and adversarial attacks, which have prompted

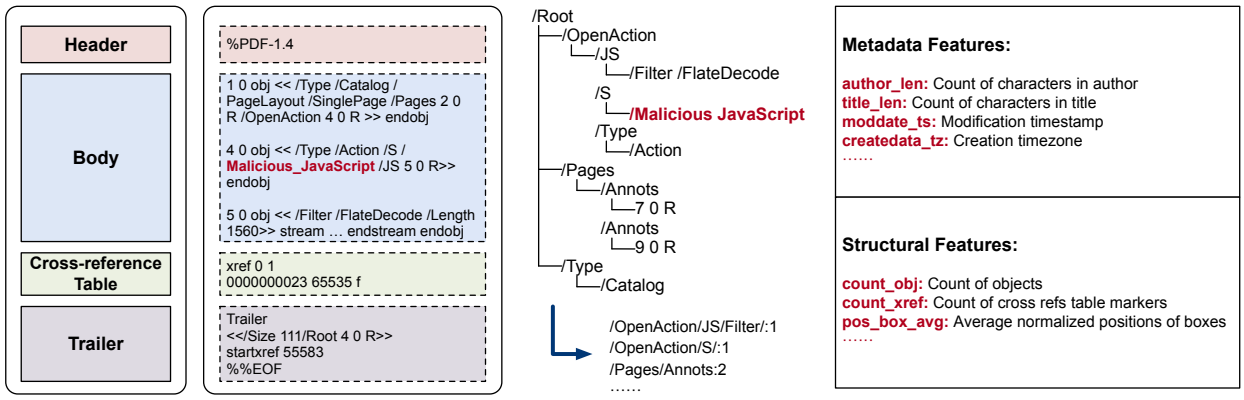


Figure 1: An example of the PDF structure and extracted features: (a) displays the basic format of a PDF; (b) depicts the logical structure of a PDF malware, and the extracted paths serve as features in Hidost [15]; (c) showcases extracted metadata and structural features utilized in PDFrate [12].

our research topic. Finally, we introduce the reconstruction-based anomaly detection and the encoder-decoder architecture, serving as the inspiration for our research methodology.

2.1 PDF Structure and PDF Malware

PDF Structure Portable Document Format (PDF) serves as the predominant attack vector utilized in document-based cyber threats. PDF is a file format originally developed by Adobe in 1992, and it was subsequently standardized as ISO 32000 [38] in 2008. The prevalence and popularity of PDF documents often make them a preferred choice for victims to open over other file types [35]. As shown in Figure 1(a), the structure of a PDF file typically consists of the following four components:

- **Header** The header section of a PDF file is the first line and specifies the PDF version number.
- **Body** The body of a PDF file is where the objects are defined. It contains the pages, graphical content, and most auxiliary information encoded as a series of objects.
- **Cross-reference Table** The cross-reference table is the set of the exact byte offset of each object from the beginning of the file. This enables the reader to locate and access these objects quickly.
- **Trailer** The trailer contains information about the root node of the file and the starting point for file parsing.

PDF Malware Definition PDF malware refers to malicious Portable Document Format files engineered to execute harmful activities upon opening or user interaction. Such activities encompass exploiting PDF readers' vulnerabilities, executing unauthorized code, or delivering additional malware into the victim's system [39–41]. Owing to the extensive adoption of PDFs for document sharing, PDF malware emerges as a notable security menace. It leverages document features, such as JavaScript, ActionScript, and embedded files, to conceal and trigger malicious payloads [42,43]. Previous research has identified three primary methods of exploitation: JavaScript-based attacks, ActionScript-based attacks, and File Embedding [43]. Among these, JavaScript-based attacks are the most prevalent. Taking JavaScript-based attack as an example, malicious JavaScript is typically found under the `/OpenAction` entry, as illustrated in Figure 1(a). `/OpenAction` specifies the value of the action to be executed when the document is opened. In Figure 1(a), it specifies the execution of `4 0 R`, referring to the object with number 4 and version 0, denoted as `4 0 obj`, which declares a JavaScript action. The actual executable content is located at `5 0 R`, i.e., `5 0 obj`, a stream object containing the JavaScript payload. This constructs an execution path:

$$\texttt{1 0 obj} \rightarrow \texttt{4 0 obj} \rightarrow \texttt{5 0 obj}$$

This path ultimately reaches `5 0 obj`, which contains the abnormal object with the malicious payload. Such paths provide a structural trace of how payloads are embedded and triggered, offering insight into the spatial and logical construction of PDF-based threats.

PDF malware can manifest in varied forms. Some samples exploiting specific vulnerabilities may fail to open due to missing content, while others, such as those using embedded files, may appear fully functional and visually normal. In contrast, benign PDFs typically adhere to correct formats, open reliably, and exhibit relatively stable structural features, with most changes limited to document content. These distinctions motivate our anomaly detection approach, which aims to identify malware by detecting deviations from the structural norms of benign PDFs.

2.2 Representative PDF Malware Classifiers

In the realm of machine learning-based PDF malware detection studies, two prominent works are Hidost [15] and

PDFrate [12]. Numerous studies have been conducted on adversarial attacks against these models. Recently, Chen et al. [17] employed the latest robust training techniques to train PDF malware classifiers. Next, we delve into these three models and discuss their merits and limitations.

Hidost and PDFrate Hidost [15] extracts the shortest structural paths from the parsing tree structure of the PDF, selecting paths appearing in at least 1000 files as the path corpus. It then employs a method similar to the classic Bag-of-Words text model [44], called Bag-of-Paths, to embed these paths. Figure 1(b) shows the structural features used by Hidost. Hidost utilizes a reduced set of 6,087 selected paths as features, derived from an initial pool of over 9 million paths, and employs the support vector machine as its classification model. On the other hand, PDFrate [12] extracts numerical features from the metadata and structure of PDF documents. These include the count of characters in each field, the size and count of each object or stream, the size and location of each box and image, and the count of encrypted objects. Feature examples used by PDFrate are illustrated in Figure 1(c). PDFrate selects 202 features, extracts the raw feature data from the original document using regular expressions, and utilizes a random forest classifier. Unfortunately, both Hidost and PDFrate are susceptible to evasion by adversarial samples.

Robust Training Classifiers Incorporating adversarial sample training undoubtedly improves a classifier's ability to detect adversarial samples. However, this approach often leads to a significant increase in the false positive rate. To address this issue, Chen et al. [17] proposed a novel method that leverages robustness properties to reduce the false positive rate. They first defined five categories of robustness properties based on the structural features of Hidost [15]. Then, they used robustness properties and symbolic interval analysis to robustly train a model (RTM). RTM demonstrated high accuracy and strong robustness. However, RTM experienced a 1.74% increase in the false positive rate, and the training time escalated by a factor of 11 compared to baseline model.

2.3 Adversarial Attacks

Adversarial attacks aim to induce erroneous outputs from machine learning models by subtly modifying the original samples or creating specially crafted ones. In a white-box attack, the attacker has full access to the model's details, including its architecture, parameters, and training set. In contrast, a black-box attack occurs when the attacker has no knowledge of the model's internals and can only interact with it through inputs and outputs. Grey-box attacks fall in between, where the attacker has partial knowledge of the model.

Adversarial attacks have been widely studied in machine learning and deep learning fields [45–47], especially regarding image recognition tasks. Many such studies have also been conducted on PDF classifiers [35, 48, 49]. Mimicus [48] is a framework for mimicry attacks against PDFrate, modifying malicious documents to appear benign. On the other hand, reverse mimicry attack [50] is an evasion technique that embeds malicious content into benign documents. EvadeML [35, 36] is a framework that utilizes genetic programming to identify evasive variants in the problem space, independent of any specific classifier algorithm or knowledge of feature extraction. The framework modifies PDF samples to generate structurally intact and semantically correct adversarial samples. In contrast, MalGAN [34] is a black-box attack framework that generates adversarial samples in the feature space. By modifying the feature values using generative adversarial networks (GANs), MalGAN automates the process of generating adversarial samples.

In §4.5 of this paper, we evaluate the robustness of our model using one white-box attack based on gradient descent, and three black-box attacks: MalGAN, adaptive EvadeML, and reverse mimicry. These attacks are assessed in both the feature space and the sample space.

2.4 Reconstruction-Based Anomaly Detection

Anomaly detection identifies samples that significantly deviate from the majority of the data [51], focusing on recognizing rare or unexpected patterns. In cybersecurity, malware is often considered an anomaly relative to benign programs [28]. Various anomaly detection methods are used, including distance-based approaches, one-class classifiers, and reconstruction-based methods [52].

Reconstruction-based methods leverage the reconstruction process, where data is mapped to a lower-dimensional space and then restored. The difference between the original and reconstructed data, known as the reconstruction error, is larger for anomalies that deviate from typical patterns. This makes reconstruction error an effective metric for anomaly detection [53]. Reconstruction-based methods have proven to be particularly promising in many security scenarios [33, 54]. Our work continues this line of research to advance PDF malware forensics analysis. In addition, we explored other anomaly detection methods, such as KNN, DeepSVDD [55] and AnoGAN [56] in §4. Our findings indicate that the reconstruction-based anomaly detection method significantly outperforms other approaches, and our experimental results substantiate this conclusion.

The encoder-decoder architecture, a prevalent type of reconstruction method, has demonstrated its utility as an effective approach for learning representations [57]. This architecture consists of an encoder and a decoder that work collaboratively: the encoder transforms the input into an intermediate representation in a latent space, and then the decoder generates the output from this intermediate representation. The ultimate objective is to minimize the difference between the input and output, and any significant deviation between them may indicate an anomaly. Training this architecture on normal samples makes the model struggle to reconstruct anomalous

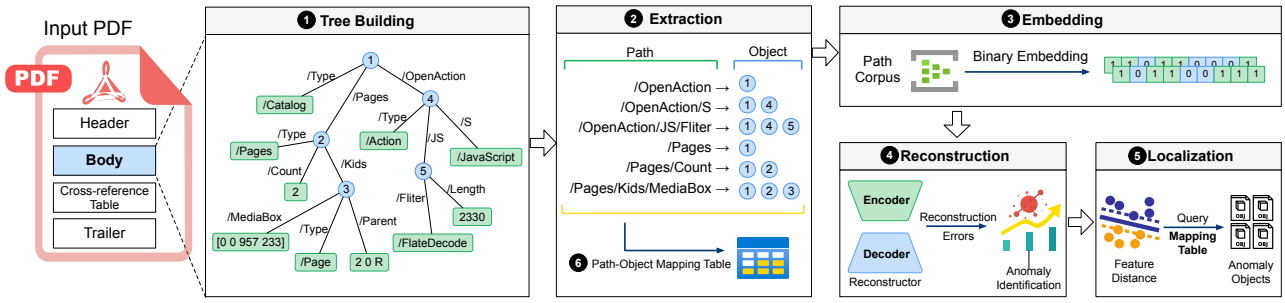


Figure 2: Overview of VAPD's workflow, which consists of three main stages: feature extraction (❶, ❷, ❸), reconstruction and anomaly detection (❹), and localization (❺, ❻).

samples effectively. Anomalies are detected when the reconstruction error exceeds a threshold. The autoencoder (AE) is a widely used model, particularly in medical image diagnosis and anomaly region detection. For example, a recent study [33] showcased using an AE model to pinpoint malware components in a malicious binary file through reconstruction errors. Inspired by this, we developed a variational autoencoder (VAE) that simultaneously performs malicious sample detection and anomaly region localization.

3 Methodology

VAPD aims to detect PDF malware and automatically identify and extract anomalous objects within PDF files, enabling analysts to quickly focus on suspicious components. An overview of VAPD is shown in Figure 2, which consists of three main stages: (1) feature extraction (❶, ❷, ❸ in Figure 2), (2) reconstruction and anomaly detection (❹), and (3) localization (❺, ❻).

In the feature extraction stage, we first construct a structural tree based on the PDF's internal architecture (❶), extract structural paths from the tree (❷), and then vectorize these paths using the Bag-of-Paths approach [15] along with binary embedding techniques (❸). Then, the resulting feature vectors are fed into VAPD for reconstruction. Trained exclusively on benign samples, the reconstructor struggles to accurately reconstruct malicious samples, leading to higher reconstruction errors that serve as a basis for anomaly detection. Finally, for anomalous object localization (❺), we compute anomaly scores by measuring the distance between each feature's reconstructed value and its original value. These scores are then used to infer the regions of the sample that are likely to be anomalous. The corresponding objects are identified by querying the path-object mapping table generated during feature extraction (❻).

3.1 Feature Extraction

We extract paths from the structure tree of PDF objects and employ a path-object reference map as features, where the structural path features are consistent with those used in previous studies [15, 17]. The hierarchical nature of PDF files makes tree structures well-suited for capturing object relationships. Path-based embeddings preserve logical relationships between objects, enabling representation of document structure for anomaly detection and object-level localization. Our feature extraction involves three steps: tree building, structural path and object reference extraction, and embedding.

Tree Building We start by constructing a structural tree based on the hierarchical relationships in the PDF, as shown in Figure 1(b). Since PDF objects often contain indirect references, the tree incorporates these, as depicted in Figure 2 (❶). We redesigned the tree-building implementation of Hidost by assigning a source attribute to each leaf node and edge, indicating the original object. These are marked with blue circles containing object identifiers in Figure 2.

Structural Path and Object Reference Extraction PDF object structures form directed graphs rather than strict trees, often resulting in cyclic graphs due to indirect references. To handle this, we use a heuristic approach [15], recursively enumerating leaf nodes and employing Breadth-First Search (BFS) to traverse paths. Each path starts from the catalog dictionary and ends with an original object type. For example, the paths `/OpenAction/JS/Filter` and `/OpenAction/S/` in Figure 1(b) lead to terminal nodes like `/FlateDecode` and `/JavaScript`.

While traversing, we record the objects each path crosses and create a mapping table linking paths to object identifiers (Stage ❷ of Figure 2). For instance, `/OpenAction/JS/Filter` passes through objects 1, 3, and 5, while `/Pages/Count` traverses objects 1 and 2. This mapping is used to locate anomalous regions (see §3.4).

As the number of structural paths for PDF objects can be unlimited, we computed the paths for all the PDFs in our dataset. For feature selection, we adopted the configuration outlined in [17] and selected the same specific 3,514 paths from the path corpus of the dataset.

Embedding We applied binary embedding to convert the object paths into vectors. If an object path appeared in a PDF file, its feature value was set to 1; otherwise, it was set to 0, resulting in a 3,514-dimensional sparse vector for each sample.

3.2 Model Design

We design an anomaly detection model based on an encoder-decoder architecture, as shown in Figure 3. The encoder transforms an input sample $x_i \in \mathbb{R}^d$ into a lower-dimensional latent vector $z_i \in \mathbb{R}^k$, and the decoder reconstructs the input from this latent representation to produce x_i'. Both encoder and decoder are implemented as neural networks, and the model is trained to minimize reconstruction error. However, due to the limitations of the basic autoencoder's hidden representation, it may struggle to capture complex structures and patterns within the data. Therefore, basic autoencoders may result in higher false positive rate or false negative rates when confronted with complex abnormal patterns. To address this, we adopt a variational autoencoder as the main architecture and augment it with a lightweight latent component, a shallow feedforward network operating on the latent space, to enhance anomaly detection .

A variational autoencoder model, denoted as M, consists of an encoder $E(x)$ and a decoder $D(z)$. Given a dataset $X = \{x_1, x_2, ..., x_n\}$ as input, M assumes the existence of a prior distribution $p(z)$ to describe the distribution of latent variables. Typically, $p(z)$ is assumed to follow a normal distribution $p(z) \sim N(0,1)$. The input X is encoded by $E(x)$ through a series of hidden layers to obtain the mean vector $\mu(x)$ and the standard deviation vector $\sigma(x)$, which parameterize the conditional probability distribution of the latent variable:

$$q(z|x) = N(\mu(x), \sigma(x)^2)$$

The reparameterization technique is employed to sample a latent vector z_i from the latent distribution $q(z|x)$, where i denotes the sample index. It involves sampling a noise vector ε_i from the standard normal distribution $N(0,1)$ and calculates z_i by the following equation:

$$z_i = \mu(x_i) + \sigma(x_i) \odot \varepsilon_i$$

where $\odot$ represents element-wise multiplication.

The decoder $D(z)$ decodes the latent variable z_i back to the original data space, reconstructing the input data x_i. The decoding layer is implemented using a neural network with the same network structure as the encoding layer, decoding the latent vector z_i through a series of hidden layers to obtain the reconstructed data's conditional distribution $p(x|z_i)$. Typically, $p(x|z_i)$ is assumed to follow a multivariate Gaussian distribution,

$$p(x|z_i) = N(\mu(x|z_i), \sigma(x|z_i)^2)$$

where $\mu(x|z_i)$ and $\sigma(x|z_i)^2$ are the encoder's mean vector and standard deviation vector. The VAE model M learns parameters by maximizing the marginal log-likelihood between the observed data X and the generated data X'. The marginal log-likelihood can be expressed as follows:

$$\log p(x) = E[\log p(x|z)] - KL[q(z|x)||p(z)]$$

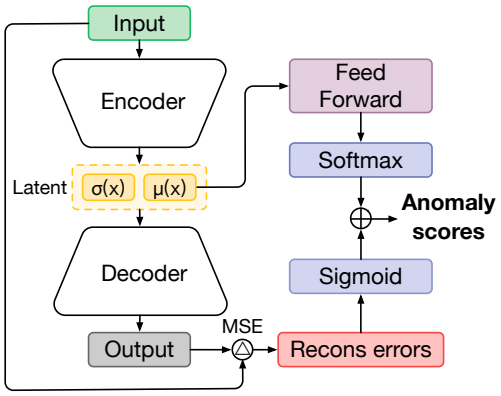


Figure 3: The network architecture of our model primarily consists of a variational autoencoder.

E denotes the expectation, $\log p(x|z)$ represents the log-likelihood of reconstructing data x given latent variable z, and $KL[q(z|x)||p(z)]$ denotes the Kullback-Leibler (KL) divergence between the encoder's output distribution $q(z|x)$ and the prior distribution$p(z)$. We train M on X with the objective of minimizing both the reconstruction loss and the KL divergence loss. The loss function is defined as follows:

$$\begin{aligned} L_{total} &= L_{recon} + L_{KL} \\ &= \sum (x_i - x_i')^2 + \frac{1}{2} \sum \left(-1 + \mu_i^2 + \sigma_i^2 - \log(\sigma_i^2)\right) \end{aligned}$$

While reconstruction error provides a measure of anomaly, it may overlook samples with subtle but consistent structural deviations. To correct these deviations, we introduce a shallow feedforward network operating directly on the latent space. This latent component serves as an auxiliary scoring mechanism that complements reconstruction-based anomaly detection by leveraging the compressed representation learned by the VAE. The input to this network is the mean vector $\mu(x)$ from the latent space of the VAE. The final layer employs a fully connected layer with a softmax activation function for the latent classifier, producing latent scores.

3.3 Anomaly Detection

Our approach is under the intuition that, compared to benign PDFs, PDF malware will contain similar yet unique features. Training M with benign samples makes it hard for M to reconstruct malicious samples. As a result, the features of the reconstructed malicious samples differ conspicuously from their original counterparts. Subsequently, we can use the feature difference between the original feature space and the reconstructed feature space to determine whether the input sample is anomalous.

Reconstruction Error After reconstructing the original feature data by M, we use Mean Squared Error (MSE) to measure the reconstruction errors between the reconstructed and input

samples. We calculate reconstruction error *mse* by:

$$mse = E(x - x')^2$$

If *mse* of an input sample exceeds a specific threshold, the input sample will be identified as anomalous.

Threshold Determination We use the benign sample set X_{benign} from X to train M and obtain a reconstruction error set $\mathcal{E}_{benign}$ for X_{benign}. Then, we use M to predict the reconstruction output for the malicious sample set $X_{malicious}$ from X, calculating the reconstruction error set $\mathcal{E}_{malicious}$. The threshold, $\phi \in \{\mathcal{E}_{benign} + \mathcal{E}_{malicious}\}$, is employed to differentiate whether the input is anomalous. To determine ϕ, we first calculate the True Positive Rate (TPR) and False Positive Rate (FPR) on the reconstruction errors $\mathcal{E}$ of X to plot the ROC curve based on threshold ϕ_i's variation. Then, we calculate the cutoff point using the Youden Index, obtaining the corresponding values of the optimal TPR and optimal FPR. Subsequently, we determine the reconstruction error threshold ϕ associated with this TPR and FPR. The optimal threshold ϕ can be identified by means of the following equation:

$$\phi = g(argmax(TPR - FPR)) \quad (1)$$

$g(x,y)$ represents the mapping from the points on the ROC curve to the set of reconstruction errors $\mathcal{E}$.

Anomaly Scores We use the sigmoid function to compute the anomaly probability s_v, and the calculation formula is as follows:

$$s_v = (1 + e^{-\lambda(mse-\phi)})^{-1}$$

The symbol λ denotes the difference amplification factor. Since the calculated reconstruction errors are rounded to three decimal places, the relative errors, which can be large, may not be significantly apparent when mapped to the probability space. Therefore, a specific coefficient is required to amplify the error difference appropriately. Simultaneously, latent scores denoted as s_l, are derived from the VAE's latent space vector. VAPD integrates both s_v and s_l to determine whether the input sample is anomalous comprehensively. The final anomaly score is calculated using the following equation:

$$s = m \cdot s_v + n \cdot s_l \quad (2)$$

For parameters m and n, we employ the Grid Search method to determine their optimal values [58].

3.4 Anomaly Localization

In real-world security scenarios, simply detecting whether a PDF is malicious is often insufficient. Security analysts typically need to understand which components are responsible for the malicious behavior, which may result from the combined effect of multiple objects, in order to support tasks such as threat attribution and forensics. Object-level localization

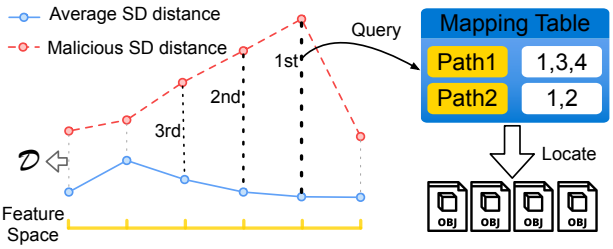


Figure 4: Illustration of how to locate anomalous objects.

narrows the analyst's focus to suspicious components, reducing manual effort and improving interpretability. Our method bridges detection and actionable insights by automatically pinpointing anomalous objects, as illustrated in Figure 4.

In §3.1, we extracted not only structured path features but also path-object reference features. This design choice enables us to pinpoint anomalous paths and their corresponding objects. In the previous subsection, we have trained M on X_{benign}, where for each sample X in X_{benign}, its feature space denotes as $F = \{f_1, f_2, ..., f_d\}$, with d being the input feature dimension. After reconstruction by M, the reconstructed feature space is $F' = \{f'_1, f'_2, ..., f'_d\}$. Consequently, we can calculate the distance for each dimension between the original and reconstructed feature spaces by:

$$f_{\Delta i} = |f_i - f_i|$$

We refer to $f_{\Delta i}$ as the single-dimensional reconstruction feature distance, abbreviated as SD distance. Next, we can obtain the SD distance set for each dimension $F_\Delta = \{f_{\Delta 1}, f_{\Delta 2}, ..., f_{\Delta i}\}$. Thus, we can derive the SD distance matrix of X_{benign}:

$$\mathcal{F}_\Delta = \begin{pmatrix} f_{\Delta 11} & f_{\Delta 12} & \cdots & f_{\Delta 1d} \\ f_{\Delta 21} & f_{\Delta 22} & \cdots & f_{\Delta 2d} \\ \vdots & \vdots & \vdots & \vdots \\ f_{\Delta n1} & f_{\Delta n2} & \cdots & f_{\Delta nd} \end{pmatrix}$$

Later, we calculated the average distance $\overline{f_{\Delta i}}$ for each dimension in the feature space of X_{benign}, denoted as $\mathcal{D}_A$. For a new input sample S, we first calculate its SD distance, denoted as $\mathcal{D}$. Then, we compute the difference between each element in $\mathcal{D}$ and $\mathcal{D}_A$. A larger difference indicates a greater deviation. Hence, we obtain a set of differences, $\mathcal{D}_\Delta$, for the SD distance of sample S relative to $\mathcal{D}_A$. By sorting $\mathcal{D}_\Delta$, we can identify the top k most anomalous features. As we utilized Bag-of-Paths and binary embedding to vectorize the structural path set, we can determine the top k most anomalous structural paths corresponding to features, as shown in Figure 4. Then, based on the path-object references mapping table, we can identify the objects traversed by these anomalous paths, completing the localization of anomalous objects in the PDF. We provide the algorithm for anomaly localization in Appendix A. We also developed an automated tool to identify the top k most anomalous paths and extract the associated objects.

4 Evaluation

We design a set of experiments to measure VAPD's efficacy from five aspects: detection accuracy, reconstruction capability of the original features, robustness against adversarial samples, performance of retrained models, and the capability of locating anomalous regions.

4.1 Experimental Setup

Dataset We collected the following four datasets for evaluation.

D1: Contagio dataset [59]. This dataset has been widely used in previous works [12, 15, 17, 35]. It is well-balanced, comprising 9,109 benign and 11,152 malicious samples. After analysis and deduplication, we ultimately identified 10,344 malicious and 8,994 benign samples. We followed the same dataset partitioning ratio as in [17], where the training set consists of 6,896 malicious and 6,296 benign samples, while the test set comprises 3,448 malicious samples and 2,698 benign samples.

D2: Custom dataset. We extracted regular malicious samples from the CIC-Evasive-PDFMal2022 dataset [60, 61], the latest repository of PDF malware including additional malicious samples collected from VirusTotal [62], and we found 16,323 usable instances. Furthermore, we sourced the latest PDF corpus [63] from the PDF Association, encompassing various PDF documents from recent years, compatible with various PDF readers, including Libre Office, Sumatra PDF, etc. We collected 15,751 benign samples, which, combined with the 16,323 regular malicious samples from the CIC-Evasive-PDFMal2022 dataset, were utilized to craft a tailored dataset for experimental evaluation.

D3: Evasive dataset. The CIC-Evasive-PDFMal2022 dataset includes a subset of evasive samples, which try to evade malware signatures identified within each category. We found 5,554 usable evasive malicious samples in the CIC-Evasive-PDFMal2022 dataset.

D4: Real-world APT Samples. We collected 138 advanced persistent threat (APT) samples from reputable sources such as [64, 65]. These APT samples have been confirmed to be employed by APT groups in actual cyber-attack campaigns and have successfully infiltrated victim environments, posing significant threats to cyberspace.

We ensured that there were no overlapping samples across the three datasets by verifying them with MD5 checksums.

Model Hyperparameters The encoder and decoder of VAPD each consist of two fully connected layers with 256 ReLU-activated neurons. The encoder outputs a 16-dimensional latent vector through a linear layer, while the decoder reconstructs a 3,514-dimensional feature vector using a sigmoid-activated output layer. The latent component is a lightweight feedforward network with two hidden layers of 200 ReLU neurons, followed by a Softmax output layer. We use the Adam optimizer [66] with a learning rate of 0.001, a mini-batch size of 16 and for 40 epochs. Training details and loss curves are provided in Appendix B.

4.2 Baselines

In our study, we compared a total of nine models, as shown in Table 1. The first three are binary classification models: PDFRate and Hidost are classical PDF malware classifiers, while RTM, built on Hidost features, represents a state-of-the-art (SOTA) classifier with strong robustness. The remaining six are anomaly detection models. We included several SOTA anomaly detection models: DeepSVDD [55], which operates in hypersphere space, and AnoGAN [56], a model based on generative adversarial network (GAN). We also used KNN model as baseline. To further evaluate reconstruction-based anomaly detection methods, we included CE-GAN [67], a SOTA Encoder-GAN architecture. Finally, we incorporated two widely used models in reconstruction-based anomaly detection: KPCA and AE. All nine baseline models were implemented to evaluate their performance in detecting PDF malware. Detailed information on the models and their hyperparameters are provided in Appendix C.

Our model and all baseline methods were trained on D1. For the binary classification baselines, all samples in the training set were used during training. For the anomaly detection baselines, only the benign samples from the training set were used for training, while all training samples were used for threshold estimation. We trained our model and baselines on a machine with an Intel Core i7-8700K (3.7 GHz), NVIDIA GeForce GTX 1080Ti, and 32 GB RAM running Ubuntu 20.04. All models were trained on the GPU. The test of PDF malware detection was also performed on the same machine.

4.3 Evaluation 1 — Detection Accuracy

We evaluate the performance of VAPD and baseline models in detecting PDF malware. All models are trained on D1 and tested on D1, D2, D3, and D4, with results summarized in Table 1. Each dataset targets a specific evaluation aspect: D1 serves as a benchmark for comparison with prior work; D2 includes diverse recent samples to assess generalization; D3 focuses on detecting evasive samples; and D4 consists of real-world APT samples to test practical effectiveness.

Results on D1 and D2 Among the three binary classification models, PDFRate achieved the best performance on the D1 test set. Although its accuracy slightly exceeded that of RTM, it demonstrated significantly weaker adversarial robustness and lacked localization capability, which will be discussed in later subsections. On the D2 dataset, all three binary classifiers showed a noticeable performance drop, with Hidost declining the least. This is primarily due to its substantially

Table 1: Comparative evaluation results on the D1 test, D2 custom, D3 evasive and D4 APT.

Type	Model	Training Time (m)	D1 Test Acc (%)	D1 Test Rec (%)	D1 Test Prec (%)	D1 Test F1(%)	D2 Custom Acc (%)	D2 Custom Rec (%)	D2 Custom Prec (%)	D2 Custom F1(%)	D3 Evasive DR(%)	D4 APT DR(%)	**Adversarial Robustness**	**Locating Capability**
Binary Classification	PDFRate	1	99.95	99.97	99.94	99.96	92.17	97.34	88.44	92.68	95.28	64.49	weak	✗
	Hidost	1	98.86	98.43	99.53	98.98	94.50	95.59	93.73	94.65	92.56	65.22	weak	✗
	RTM	84	99.15	99.97	98.54	99.25	91.60	98.17	87.00	92.25	93.95	89.86	medium	✗
Anomaly Detection	KNN	1	91.03	94.52	90.01	92.21	67.77	88.27	63.11	73.60	87.83	83.87	-	✗
	DeepSVDD	1	95.07	91.68	99.50	95.43	79.97	89.74	75.47	81.99	88.17	66.45	weak	✗
	AnoGAN	21	55.68	98.17	55.99	71.31	58.66	99.98	55.12	71.04	-	-	-	✗
	KPCA	1	71.69	99.97	66.47	79.85	58.88	99.96	55.31	71.22	-	-	-	✗
	CE-GAN	1	88.80	92.63	85.26	88.80	82.12	90.50	77.92	83.74	86.60	76.09	weak	✗
	AE	1	83.01	82.08	86.92	84.43	68.44	85.53	64.27	73.39	84.12	83.87	-	✗
	VAPD (Ours)	4	99.54	99.85	99.34	99.60	97.96	98.80	97.24	**98.02**	**96.69**	**100**	**strong**	✓

higher precision compared to the other two, indicating better generalization and lower false positive rates on benign samples.

Among the six anomaly detection baselines, AnoGAN was the only model that completely failed on the PDF malware detection task, with a recall nearly 1.7 times its precision, indicating a very high false positive rate. The other five anomaly detection models also underperformed compared to binary classifiers on both D1 and D2, with a significant drop in performance on D2. This can be attributed to the fact that binary classifiers are trained on balanced datasets containing both benign and malicious samples, allowing them to learn more comprehensive and discriminative features. In contrast, anomaly detection models are trained solely on benign data and use malicious samples only for threshold estimation, limiting their performance in standard in balanced detection scenarios.

VAPD, however, achieved results comparable to binary classifiers on D1 and outperformed both Hidost and RTM. Moreover, it achieved the best performance on D2. While its performance slightly declined from D1 to D2, the drop was the smallest among all baseline models, demonstrating superior generalization to unseen PDF samples.

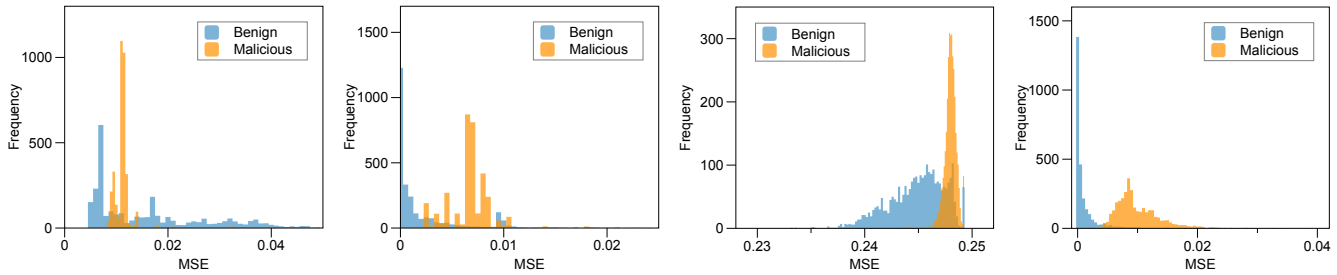
Results on D3 and D4 We continued to evaluate our model and baseline models on D3 and D4. Since D3 and D4 consist entirely of malicious samples, we use Detection Rate (DR) as the evaluation metric, and the results are shown in Table 1. Due to the extremely high recall and low precision of AnoGAN and KPCA on the D2 dataset, which typically indicating a very high false positive rates, and these models tend to classify the majority of benign samples as malicious. As a result, evaluating their performance on D3 and D4, which consist entirely of malicious samples, becomes meaningless.

The binary classification models achieved slightly higher detection rates than the anomaly detection baselines on the D3 dataset. However, their performance dropped significantly on D4, especially for PDFRate and Hidost, indicating limited generalization to previously unseen malicious samples. As for anomaly detection models, GAN-based method proved ineffective, as demonstrated by the poor performance of AnoGAN. Models based on distance metrics, such as KNN and DeepSVDD, showed moderate performance but were far from optimal. Similarly, reconstruction-based approaches, including KPCA, CE-GAN, and AE, performed poorly on this task, and CE-GAN showed relatively effective performance, indicating its superior ability to learn the characteristics of benign samples compared to KPCA and AE.

Results Analysis In the PDF malware detection task, anomaly detection methods based on distance metrics and GANs, such as KNN, DeepSVDD, and AnoGAN, demonstrated poor performance. Even when alternative features like those from PDFRate were used as input, these models failed to achieve satisfactory results. Reconstruction-based models such as KPCA and AE showed relatively better performance when using PDFRate features. However, since PDFRate features do not capture structural information, models relying on them cannot support object-level localization, even if they succeed in malware classification. Detailed results are presented in Table 7 of Appendix D. Among all reconstruction-based models, only VAPD achieved consistently strong performance, outperforming all other baselines in both detection accuracy and localization capability. In the next subsection, we evaluate VAPD's reconstruction ability to understand why it surpasses other reconstruction-based approaches.

4.4 Evaluation 2 — Feature Reconstruction

Among the six baseline anomaly detection models, KPCA, CE-GAN, and AE rely on reconstruction for anomaly detection. We evaluate their reconstruction performance alongside VAPD. All four models were trained on benign samples from the D1 training set, and their reconstruction error distributions on benign and malicious samples in the D1 test set are shown in Figure 5. Among the four models, KPCA exhibited the weakest reconstruction performance. It failed to effectively learn the characteristics of benign samples and to distinguish them from anomalies. We provide the reconstruction errors of KPCA on the D1 training set in Appendix E. Upon closer inspection, it is evident that the reconstruction error distributions of malicious samples in KPCA-based method align closely with the trends observed in the test set (which includes

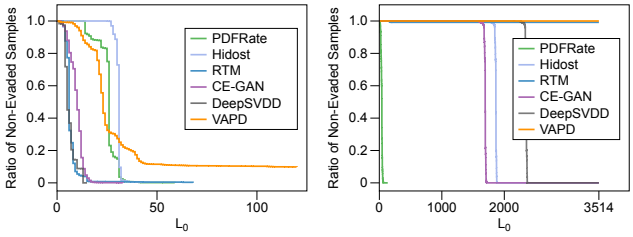


(a) KPCA (b) AE (c) CE-GAN (d) VAPD

Figure 5: Histogram distributions of reconstruction errors on the D1 test set for the KPCA, AE, CE-GAN, and VAPD.

Table 2: Summary of adversarial attack settings.

Attack Method	Type	Target Space
Gradient Descent	White-Box	Feature
MalGAN	Black-Box	Feature
Adaptive EvadeML	Black-Box	Sample
Reverse Mimicry	Black-Box	Sample



(a) Gradient Descent (b) MalGAN

Figure 6: Ratio of unevaded samples under gradient descent attack and MalGAN attacks.

both benign and malicious samples). This implies that KPCA struggles to correctly reconstruct data beyond the training set, considering all data outside the training set as anomalous. This phenomenon explains why KPCA exhibited very high recall, but low precision in the detection task discussed in the previous subsection.

The method based on AE and CE-GAN exhibited some discriminative capability on the D1 test, but there is still a considerable overlap, as shown in Figure 5(c). This indicates a significant proportion of samples that they cannot correctly distinguish. VAPD outperformed the best among the other three reconstruction methods, demonstrating excellent discriminative ability on the D1 test. As illustrated in Figure 5(d), the reconstruction errors for the majority of benign samples are negligible, with values concentrated near zero.

To directly observe the differences before and after reconstruction through VAPD, we employed the t-SNE algorithm using scikit-learn package [68] to visualize the high-dimensional feature space in a two-dimensional plot, as depicted in Figure 13 in Appendix F. Upon examining these two figures, we can observe that the reconstruction errors of malicious and benign samples exhibit a small overlap. We attribute this to the enhanced robustness of the model, which has come at the expense of a slight increase in false positive rate.

4.5 Evaluation 3 — Adversarial Attacks

We first conducted adversarial evaluations on the models from Table 1 with F1 scores above 80% on both D1 and D2, using four adversarial techniques: gradient descent [69], MalGAN [34], adaptive EvadeML [36], and reverse mimicry [50]. Other models, such as KNN, AnoGAN, KPCA, and AE, were excluded from this evaluation due to their excessively high false positive rates, which render adversarial testing results less meaningful. We discuss this limitation at the end of this subsection. The summary of adversarial attack settings are shown in Table 2 and detailed parameters are provided in Appendix G.

Gradient Descent Attack Gradient descent attack is a type of white-box attack, where the attacker has complete knowledge of the target model and access to gradient information. By perturbing the input feature values, the attacker uses the model's gradient information to guide the direction of the perturbations, as the gradient indicates the direction in which the loss function increases most rapidly. We did not impose any constraints on the perturbation distance, making it unbounded. The attack stops once all evasion instances are found or after 200,000 iterations, consistent with the attack setup in [17].

We applied gradient descent attacks to five baseline models from Table 1. Since PDFRate uses a random forest classifier by default, which is non-differentiable and thus incompatible with standard gradient-based methods, we employed a finite-difference approximation to estimate gradients and perform the attack indirectly. The attacks were conducted on the D1 test set, and the results are presented in Figure 6(a). The curves show the proportion of non-evaded samples at varying L_0 distances, where L_0 denotes the number of feature modifications required to achieve evasion. A higher L_0 indicates that the attack must modify more features to succeed.

As shown in the figure, all baseline models except VAPD were fully evaded. RTM was completely evaded at $L_0 = 68$, which is the highest perturbation level observed among the

baseline models. This suggests that evading RTM requires a higher modification cost compared to other baselines. In contrast, VAPD retained 9.6% of samples that could not be evaded, and its curve exhibited the slowest decline. Unlike the other models, which dropped sharply toward zero, VAPD maintained better resistance throughout. These results indicate that VAPD demonstrates the strongest robustness against gradient-based attacks. VAPD uses regularization constraints to improve generalization, reducing overfitting to specific feature distributions. As a result, even with unbounded gradient perturbations, VAPD maintains robustness by requiring significantly larger and less feasible perturbations to evade detection.

MalGAN MalGAN is a type of black-box attack, where the attacker has no knowledge of the target model's details and can only access the model's output (beign or mailicous) without query limit. MalGAN employs a generative adversarial network (GAN) which consists of a generator and a substitute detector. Adversarial samples are crafted by appending a segment of noise data to the end of the input feature vector. To approximate the behavior of the black-box classifier and provide gradient information back to the generator, a substitute detector is utilized.

The results in Figure 6(b) indicate that MalGAN successfully evaded all models except RTM and VAPD. RTM demonstrated significant robustness against MalGAN, with only 0.8% of test samples successfully evading detection, leaving 99.2% of the test samples unevaded. In contrast, VAPD exhibited complete robustness to MalGAN, achieving 100% resistance to its adversarial attempts. Note that PDFRate was fully evaded at a relatively small L_0, which does not necessarily mean it is more easily bypassed by MalGAN. This is primarily due to its lower feature dimensionality, with only 135 features compared to 3514 used by the other models. Detailed results of MalGAN attacks based on PDFRate's feature space are provided in Appendix D.

MalGAN's success relies on approximating the target model using a substitute detector to generate adversarial samples. However, the complexity of VAPD's anomaly detection, which evaluates discrepancies between reconstructed and original features, makes it inherently non-trivial to approximate. This results in the failure of MalGAN to generate effective adversarial samples against VAPD, as the attack is unable to exploit gradient-based vulnerabilities that are often present in traditional classifiers.

Adaptive EvadeML EvadeML is a black-box attack framework specifically designed for PDF malware detectors. It enables attackers to access the confidence scores of target detectors to guide the iterative generation of adversarial examples using a genetic algorithm. Chen et al. [17] developed a stronger adaptive evolutionary attack based on EvadeML, which is not constrained by robustness features. In this study, we adopt the configuration of the adaptive EvadeML. Consistent with prior research [17, 35], we use 500 seed samples with a population size of 48, 20 evolutionary rounds, and a mutation rate of 0.1. The validity of the adversarial samples within the population is verified using Cuckoo Sandbox [70]. Adversarial examples are generated through an evolutionary process starting from the seed samples. These samples are subsequently executed in a sandbox environment to generate network communication indicators. To ensure the persistence of malicious functionalities, an adversarial sample is considered legitimate if it produces the same network communication indicators as its corresponding seed sample within the sandbox. The presence of identical communication indicators demonstrates the preservation of malicious functionalities.

Table 3: Adversarial samples detection accuracy (robustness accuracy) under adaptive EvadeML attack and reverse mimicry attack.

Model	Robustness Accuracy	
	Adaptive EvadeML	Reverse Mimicry
PDFRate	0	0.14
Hidost	0	0
RTM	0.44	0.53
CE-GAN	0.02	0
DeepSVDD	0	0
VAPD	**0.94**	**0.89**

Given the time-consuming nature of dynamic sandbox validation, the attack tests on baseline models and VAPD were conducted over four weeks. The results are presented in Table 3. The results show that PDFRate, Hidost, and DeepSVDD were completely evaded by the adaptive EvadeML attack. CE-GAN was also nearly fully evaded, with a robustness accuracy of only 0.02. RTM performed moderately, with 44% of seed samples failing to generate adversarial examples capable of evading RTM. In contrast, VAPD demonstrated the strongest resistance, with 94% of samples remaining non-evaded. VAPD's resistance lies in its feature representation and reliance on anomaly detection. The reconstruction process captures the holistic characteristics of benign samples, making it difficult for the evolutionary algorithm to identify modifications that both evade detection and maintain functional consistency.

Reverse Mimicry Reverse mimicry manipulates the sample space directly by injecting malicious payloads into benign samples, thereby generating viable adversarial examples designed to evade classification boundaries. This black-box attack is independent of specific detectors or features and does not require repeated interactions or iterations with the target detector, making it straightforward to execute. We used the 500 seed samples from the previous adaptive EvadeML evaluation and applied the same Cuckoo sandbox configuration to validate the legitimacy of the adversarial samples. Evasion samples generated by adaptive EvadeML and reverse mimicry from exploit-labeled seeds retained identical beacon

Table 4: Comprehensive results of retrained PDFRate, Hidost, DeepSVDD, and CE-GAN models with evasive samples.

Retrained Model	D1 Test		D2 Custom		Robustness Accuracy		
	Accuracy (%)	FPR (%)	Accuracy (%)	FPR (%)	Gradient Descent	MalGAN	Reverse Mimicry
PDFRate	99.89$\downarrow_{0.06}$	0.26 $\uparrow_{0.22}$	72.08$\downarrow_{20.09}$	56.55$\uparrow_{43.46}$	0.004$\uparrow_{0.004}$	0.007$\uparrow_{0.007}$	0.869$\uparrow_{0.729}$
Hidost	95.73$\downarrow_{3.13}$	7.52 $\uparrow_{6.93}$	93.24$\downarrow_{1.26}$	9.46 $\uparrow_{2.84}$	0.001$\uparrow_{0.001}$	0.001$\uparrow_{0.001}$	0
DeepSVDD	97.95$\uparrow_{2.88}$	4.04 $\uparrow_{3.44}$	78.82$\downarrow_{1.15}$	41.57$\uparrow_{11.34}$	0	0	0
CE-GAN	88.27$\downarrow_{0.53}$	21.39$\uparrow_{0.93}$	82.80$\uparrow_{0.68}$	28.63$\uparrow_{1.99}$	0	0	0
VAPD	99.54	0.85	97.96	2.92	**0.098**	**1**	**0.894**

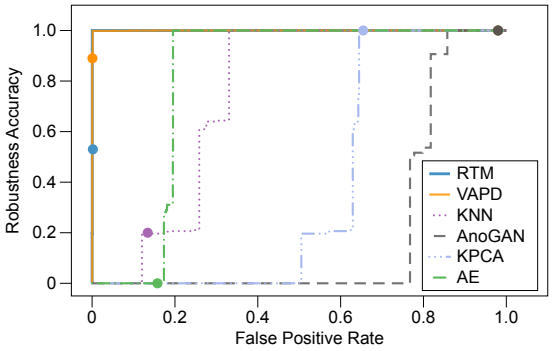


Figure 7: Curve of false positive rate versus robustness accuracy, with each point representing the FPR-accuracy achieved by the model in the detection task.

behaviors in sandbox tests and received the same exploit tags from VirusTotal, confirming the preservation of the exploit structure.

The experimental results are shown in Table 3. Hidost, CE-GAN and DeepSVDD were fully evaded, showing no resistance to the reverse mimicry attack. PDFRate and RTM detected 14% and 53% of the adversarial samples respectively, while VAPD detected 89%. Among the evaluated models, VAPD demonstrated the strongest resistance to reverse mimicry attacks. VAPD's robustness stems from its reconstruction-based detection strategy, which is sensitive to structural and behavioral deviations introduced by the embedded payloads. Unlike decision-boundary-based methods, which can be tricked by slight shifts in feature values, VAPD evaluates anomalies at a granular level through reconstruction errors. These errors highlight inconsistencies in the overall feature distribution, allowing the model to effectively detect manipulations.

Adversarial Results Analysis Overall, VAPD demonstrates stronger adversarial robustness compared to current SOTA robust model RTM. Notably, these improvements are achieved at a significantly reduced training cost. Specifically, VAPD's training time of 4 minutes is merely 4.8% of that required by RTM (84 minutes) under the same GPU specifications.

Additional, as mentioned at the beginning of this subsection, we did not conduct adversarial evaluations on KNN, AnoGAN, KPCA, and AE due to their excessively high false positive rates (FPR), which render them nearly unusable in real-world scenarios. To further explore the relationship between FPR and adversarial robustness, we use adversarial samples generated by a practical and easily executed attack, reverse mimicry, to analyze how these four models behave under varying FPR levels. The results are shown in Figure 7.

As shown in Figure 7, when the FPR is below 10%, all four baseline anomaly detection models exhibit a robustness accuracy of 0. AE achieves a robustness accuracy of 1 only when its FPR reaches approximately 20%, which is already unacceptably high in real-world applications. For KPCA and AnoGAN, an FPR exceeding 70% is required to achieve perfect robustness, which is clearly impractical and renders the models unsuitable for deployment. In comparison, RTM achieves its current level of robustness with an FPR of 1.89%, while VAPD achieves the same with a significantly lower FPR of just 0.85%.

4.6 Evaluation 4 — Retrain with Evasive Samples

We retrained the baseline models evaluated in §4.5 using the evasive samples from the D3, while keeping the same parameters and experimental settings. The two binary classifiers, PDFRate and Hidost, incorporated evasive samples directly during training. In contrast, the two anomaly detection models, DeepSVDD and CE-GAN, used evasive samples during threshold calibration. This was done to investigate whether incorporating evasive samples during training can effectively improve the models' adversarial robustness. The results are shown in Table 4.

All four retrained baseline models experienced varying degrees of decline in both accuracy and FPR. Notably, on the D2 dataset, the retrained PDFRate exhibited an FPR as high as 56.55%, marking a 43.46% increase compared to its original version without evasive sample training. Although the robustness accuracy of PDFRate against the reverse mimicry attack improved significantly to 0.869, this came at the cost of an unacceptably high FPR, severely limiting its practical usability. Moreover, the improvement against gradient-based attacks and MalGAN was negligible. For Hidost, the improvement in robustness after retraining was also minimal. However, its performance degradation on the D2 custom dataset was

Table 5: Localization recall at document-level and object-level.

Top K	CE-GAN		VAPD		PeePDF	
	Doc	Obj	Doc	Obj	Doc	Obj
K=1	0	0	0.609	0.281	0.760	0.641
K=3	0	0	0.622	0.469		
K=5	0	0	0.764	0.679		
K=7	0	0	**0.818**	**0.745**		

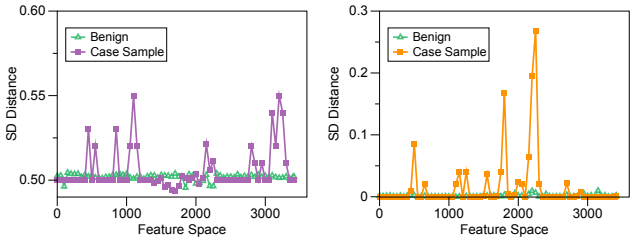
much smaller than that of PDFRate, suggesting that Hidost's structural path features are more stable and less sensitive to changes in training data. The remaining two anomaly detection baselines, DeepSVDD and CE-GAN, not only showed performance drops on both D1 and D2 after retraining, but also failed to achieve any meaningful gains in robustness. Retraining baseline models with evasive samples does not lead to significant improvements in adversarial robustness and, in all cases, yields weaker performance than VAPD, which requires no special training.

4.7 Evaluation 5 — Localization

We evaluated localization using seed samples from adaptive EvadeML and reverse mimicry attacks, which were manually labeled in prior work [35] to indicate malicious payloads within specific objects. As the first study to incorporate object-level localization into machine learning-based PDF malware detection, no established benchmarks are available. To support evaluation, we manually analyzed all test samples and found that each contained between 1 and 12 anomalous objects, with an average of 2.7 per sample.

We evaluated the localization capability at both the document-level and object-level. Document-level localization is considered successful if at least one true anomalous object is identified, while object-level evaluation computes the recall over all malicious objects per sample, offering a more fine-grained and challenging assessment. We also compared VAPD's performance with the latest community version of the signature-based analysis tool PeePDF [37].

Localization Results and Analysis We selected the top 1, 3, 5, and 7 most anomalous paths and mapped them to their corresponding objects, as shown in Table 5. Although CE-GAN is also reconstruction-based, it performed poorly in localization. To investigate the cause, we analyzed a randomly selected sample (details in Appendix H) and compared its SD distance distributions under CE-GAN and VAPD with those of benign samples from D1. As shown in Figure 8, VAPD and CE-GAN identify different top anomalous paths, with CE-GAN showing a divergent error pattern. This indicates CE-GAN fails to capture and reconstruct the anomalous regions effectively, leading to poor localization performance. This limitation stems from CE-GAN's selective reconstruction mechanism, which focuses only on partial regions of the input and lacks awareness of the full structural context. In contrast, VAPD leverages object traversal paths to capture the logical flow of the document and accurately localize anomalies. Furthermore, VAPD is highly efficient, completing localization in approximately 0.01 seconds per sample.



(a) CE-GAN (b) VAPD

Figure 8: Distribution of SD distances of the case sample computed by CE-GAN (a) and VAPD (b).

Table 6: Localization recall against adversarial samples.

Localization Level	VAPD	PeePDF
Document	1	1
Object	**0.869**	0.629

VAPD performs global anomaly detection before narrowing down to specific paths. At small K values (e.g., $K = 1$), localization can be challenging because each sample may contain multiple anomalous objects with varying structures. Selecting only the top-1 path risks missing the true source of malicious behavior, especially when anomaly signals are sparsely distributed across objects. As K increases, coverage improves, leading to higher recall at both the document and object levels. At $K = 5$, VAPD outperforms PeePDF. However, setting K too high results in over-identification, which reduces interpretability and practical usability. Therefore, the choice of K should be adapted based on the specific input PDF size and analysis requirements.

We conduct a further analysis of the case sample in Figure 8, and detailed localiazation results are presented in Table 8 in Appendix H. VAPD identifies obj1, obj3, obj12, and obj15 as anomalous (with $K = 5$), whereas PeePDF detects only obj1 and obj3. The key difference lies in obj12, a generic stream object defined as ≪`/Filter /FlateDecode /Length 15 0 R`≫ that does not match any known signatures but contains the actual malicious script. When the file is opened, the `/OpenAction` entry automatically triggers the execution of JavaScript contained in obj12, which subsequently initiates a series of malicious actions. This reveals that PeePDF, constrained by predefined signatures, often fails to identify the actual payload-carrying objects, instead flagging only preceding references.

Localization on Adversarial Samples We also evaluated localization for adversarial samples generated by the reverse mimicry attack. VAPD achieved a localization recall of 0.869 for object-level detection when K = 3, outperforming PeePDF.

The results are shown in Table 6. This demonstrates that our localization capability remains robust even under adversarial attacks, ensuring reliable detection of anomalies. In reverse mimicry attacks, malicious payloads such as JavaScript are injected into benign PDFs to craft a PDF malware, requiring the addition of trigger objects like `/OpenAction` and `/JS`, along with a stream object containing the code. This creates execution paths similar to those described in §2.1, ultimately reaching the stream object that contains payload. Although the overall feature representation of such adversarial samples closely resembles that of benign PDFs, the malicious payload is typically introduced as a localized structural anomaly. While this difference is subtle, it remains detectable by VAPD as an anomaly. In real-world scenarios of PDF malware detection and response, this dual robustness in both detection and localization offers meaningful interpretability and operational value, making VAPD a practical tool for security analysts.

4.8 Ablation Study

We performed an ablation study to elucidate the influence of two key components within our reconstruction framework: the main VAE architecture and the latent component. We first retained only the VAE component, achieving an F1 score of 96.14% on the D1 test. When using only the latent component, the F1 score on the D1 test was 99%. The combination of both modules resulted in an impressive F1 score of 99.6%. We also conducted adversarial attack experiments using each component separately. Under the same attack settings in §4.5, the VAE component we implemented demonstrated 100%, 98.2%, and 100% resistance against MalGAN, adaptive EvadeML, and reverse mimicry attacks, respectively. In contrast, the sole use of the latent component was rapidly evaded. The VAE component ensures adversarial robustness, while the introduction of the latent classifier enhances the model's regular performance, notably reducing the false positive rate from 8.19% to 0.85%. These findings indicate that both components are essential for achieving optimal performance.

5 Discussion & Conclusion

Catastrophic Forgetting Catastrophic forgetting [71, 72] describes the phenomenon where a model, when trained on a new task, forgets knowledge from previous tasks, resulting in a significant decline in performance. In the context of malware detection, detectors are typically tailored to specific types of malware. This is because different malware types, such as PE malware and PDF malware, have fundamentally different structures and features, including distinct file formats and feature engineering requirements. As a result, a detector trained for one type of malware is rarely retrained to handle other types or adapt to new tasks. Consequently, malware detectors are inherently less prone to catastrophic forgetting.

Concept Drift A primary challenge in malware detection, as highlighted in [73], is concept drift [74–76]. As malware evolves, machine learning models trained on outdated datasets often suffer reduced effectiveness. In contrast, VAPD's reliance on training exclusively with benign samples provides a distinct advantage, as benign PDFs tend to exhibit greater stability over time compared to malicious ones. This makes our approach less vulnerable to concept drift. Our experimental results support this assertion. The model trained on benign samples from D1, a dataset released in 2013, achieved the highest performance on D2, which contains the latest PDF samples from 2022. Additionally, VAPD outperformed other methods when tested on real-world APT samples collected over the past three years.

Limitations & Future Work Our study is not without limitations. First, VAPD still exhibits a small false positive rate in PDF malware detection. This is primarily because VAPD's performance relies on the feature representation of input data, and it occasionally misinterprets complex benign PDFs, leading to inaccurate detection. Second, there is considerable potential for improving VAPD's localization performance. As the reconstruction process is based on global features of the sample, slight deviations in localized reconstruction errors can result in misalignment during anomaly localization. Furthermore, certain obfuscation techniques (e.g., encoded keywords) may introduce feature parsing inaccuracies, leading to incorrect feature extraction in the current approach. Addressing these issues in future work is imperative. We plan to enhance the feature extraction process to improve tolerance for malformed files and design mechanisms to decode obfuscated keywords, such as converting `/J#53` back to `/JS`. Additionally, we aim to further refine the model's sensitivity to local features to minimize localization errors.

Our experimental results indicate that reconstruction-based anomaly detection methods hold promise for PDF malware detection and forensics. Beyond generative models like VAEs, GANs [77], another type of generative model, have also shown potential. We believe that exploring anomaly detection methods based on GAN reconstruction represents an intriguing research direction.

Conclusion In this paper, we introduce VAPD, a reconstruction-based anomaly detection model specifically designed for PDF malware forensic analysis. Trained solely on benign samples, VAPD leverages robust reconstruction to detect malicious instances via reconstruction error analysis. We evaluate the performance of VAPD across various datasets and adversarial samples. The results demonstrate that VAPD achieves high detection accuracy for malicious samples and exhibits strong robustness against adversarial attacks. Additionally, VAPD shows effective localization capabilities in identifying anomalous regions, accurately pinpointing the specific anomalous objects. The experimental results highlight VAPD's overall effectiveness in analyzing, detecting, and localizing malicious PDF files.

Acknowledgment

We thank our anonymous shepherd, and all anonymous reviewers for their valuable comments to improve this paper. This work is supported by the National Nature Science Foundation of China under Grant No.62172308, 62272351, 61972297, and 62172144. Jiang Ming is supported by NSF grants 2312185 & 2417055, Google Research Scholar Award, and Tulane COR Fellowships.

Ethics Considerations

Our research focuses on PDF malware detection and forensics with the aim of enhancing security measures and improving antivirus capabilities. Our main objective is to protect users from malicious activities and make a positive contribution to the field of computer security. Firstly, the data utilized in our study are sourced from publicly available datasets, and no private or sensitive user data were accessed or used. We have ensured that all data sources comply with ethical standards and do not infringe on the privacy rights of individuals or organizations. Secondly, our research does not involve any direct interaction with users or systems that could lead to economic loss or psychological distress. The focus is on improving detection mechanisms, with the ultimate goal of reducing harm by preventing malware infections. Lastly, our research does not involve testing on real-time systems or networks that could disrupt services or compromise security. All evaluations of our detection methods were conducted in a controlled environment, ensuring that no unintended consequences arise.

Open Science Policy

We release a prototype of VAPD and evaluation datasets to facilitate reproduction and reuse, as all are found at Zenodo.

References

[1] Duff Johnson. PDF's Popularity Online. https://pdfa.org/pdfs-popularity-online/, 2021.

[2] Ken Chang and Ying-Dar Lin. Advanced Persistent Threat: Malicious Code Hidden in PDF Documents. Technical report, National Chiao Tung University, September 2014.

[3] Nir Nissim, Aviad Cohen, Chanan Glezer, and Yuval Elovici. Detection of malicious PDF files and directions for enhancements: A state-of-the art survey. *Computers & Security*, 48, 2015.

[4] Meng Xu and Taesoo Kim. PlatPal: Detecting Malicious Documents with Platform Diversity. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security '17)*, 2017.

[5] Trend Micro. Macro Malware: Here's What You Need to Know in 2016. http://tinyurl.com/ysh4pb8h, 2016.

[6] Palo Alto Networks. 2023 Unit 42 Network Threat Trends Research Report. http://tinyurl.com/4266h2hn, 2023.

[7] VIPRE Security Group David Bloxberg, Senior Global Marketing Manager. PDFs: Why They Are Such a Popular Attack Vector. https://safesendsoftware.com/pdf-exploit-popular-attack-vector/, 2024.

[8] CISA. Phishing Infographic. https://www.cisa.gov/sites/default/files/2023-02/phishing-infographic-508c.pdf, December 2022.

[9] Adeline Zhang. Chrome PDF File Parsing 0-Day Vulnerability Threat Alert. https://nsfocusglobal.com/chrome-pdf-file-parsing-0-day-vulnerability-threat-alert/, 2019.

[10] SOCRadar. CVE-2024-4367 in PDF.js Allows JavaScript Execution, Potentially Affecting Millions of Websites: Update Now. https://socradar.io/cve-2024-4367-in-pdf-js-allows-javascript-execution-potentially-affecting-millions-of-websites-update-now/, 2024.

[11] Pavel Laskov and Nedim Šrndić. Static Detection of Malicious JavaScript-Bearing PDF Documents. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC '11)*, 2011.

[12] Charles Smutz and Angelos Stavrou. Malicious PDF Detection Using Metadata and Structural Features. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12)*, 2012.

[13] Chad Smutz and Angelos Stavrou. When a Tree Falls: Using Diversity in Ensemble Classifiers to Identify Evasion in Malware Detectors. In *Proceedings of the 23rd Network and Distributed System Security Symposium (NDSS '16)*, 2016.

[14] Davide Maiorca, Davide Ariu, Igino Corona, and Giorgio Giacinto. A Structural and Content-ased Approach for a Precise and Robust Detection of Malicious PDF Files. In *Proceedings of the 2015 International Conference on Information Systems Security and Privacy (ICISSP '15)*, 2015.

[15] Nedim Šrndic and Pavel Laskov. Detection of Malicious PDF Files Based on Hierarchical Document Structure. In *Proceedings of the 20th Network and Distributed System Security Symposium (NDSS '13)*, 2013.

[16] Nedim Šrndić and Pavel Laskov. Hidost: A Static Machine-Learning-Based Detector of Malicious Files. *EURASIP Journal on Information Security*, 2016:1–20, 2016.

[17] Yizheng Chen, Shiqi Wang, Dongdong She, and Suman Jana. On Training Robust PDF Malware Classifiers. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security '20)*, 2020.

[18] Suleiman Y. Yerima, Abul Bashar, and Ghazanfar Latif. Malicious PDF detection Based on Machine Learning with Enhanced Feature Set. In *Proceedings of 14th International Conference on Computational Intelligence and Communication Networks (CICN '22)*, 2022.

[19] G.M. Sakhawat Hossain, Kaushik Deb, Helge Janicke, and Iqbal H. Sarker. PDF Malware Detection: Towards Machine Learning Modeling with Explainability Analysis. *IEEE Access*, 2024.

[20] Curtis Carmony, Mu Zhang, Xunchao Hu, Abhishek Vasisht Bhaskar, and Heng Yin. Extract Me If You Can: Abusing PDF Parsers in Malware Detectors. In *Proceedings of the 23rd Network and Distributed System Security Symposium (NDSS '16)*, 2016.

[21] Keane Lucas, Samruddhi Pai, Weiran Lin, Lujo Bauer, Michael K Reiter, and Mahmood Sharif. Adversarial Training for Raw-Binary Malware Classifiers. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security '23)*, 2023.

[22] Deqiang Li, Qianmu Li, Yanfang Ye, and Shouhuai Xu. Arms Race in Adversarial Malware Detection: A Survey. *ACM Computing Surveys (CSUR)*, 55(1):1–35, 2021.

[23] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial Examples for Malware Detection. In *Proceedings of the 22nd European Symposium on Research in Computer Security (ESORICS '17)*, 2017.

[24] David Evans, Weilin Xu, and Yanjun Qi. Adversarial Machine Learning: Are We Playing the Wrong Game? https://www.cs.virginia.edu/~evans/talks/cispa2017/, July 2017.

[25] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in Adversarially Robust Deep Learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML '20)*, 2020.

[26] Keane Lucas, Mahmood Sharif, Lujo Bauer, Michael K Reiter, and Saurabh Shintre. Malware Makeover: Breaking ML-based Static Analysis by Modifying Executable Bytes. In *Proceedings of the 16th ACM Asia Conference on Computer and Communications Security (ASIA CCS '21)*, 2021.

[27] Shouling Ji, Tianyu Du, Shuiguang Deng, Peng Cheng, Jie Shi, Min Yang, and Bo Li. Robustness Certification Research on Deep Learning Models: A Survey. *Chinese Journal of Computers*, 45:190–206, 2022.

[28] Yizheng Chen, Zhoujie Ding, and David Wagner. Continuous Learning for Android Malware Detection. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security '23)*, 2023.

[29] Chong Zhou and Randy C Paffenroth. Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '17)*, 2017.

[30] Jianwen Tian, Kefan Qiu, Debin Gao, Zhi Wang, Xiaohui Kuang, and Gang Zhao. Sparsity Brings Vulnerabilities: Exploring New Metrics in Backdoor Attacks. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security '23)*, 2023.

[31] Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical Approach to Asynchronous Multivariate Time Series Anomaly Detection and Localization. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '21)*, 2021.

[32] David Zimmerer, Fabian Isensee, Jens Petersen, Simon Kohl, and Klaus Maier-Hein. Unsupervised Anomaly Localization Using Variational Auto-Encoders. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference*, 2019.

[33] Evan Downing, Yisroel Mirsky, Kyuhong Park, and Wenke Lee. DeepReflect: Discovering Malicious Functionality Through Binary Reconstruction. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security '21)*, 2021.

[34] Weiwei Hu and Ying Tan. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. In *Proceedings of the 7th International Conference on Data Mining and Big Data (DMBD '22)*, 2022.

[35] Weilin Xu, Yanjun Qi, and David Evans. Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers. In *Proceedings of the 23rd Network and Distributed Systems Symposium (NDSS '16)*, 2016.

[36] David Evans. Is Robust Machine Learning Possible? https://evademl.org/, 2022.

[37] peepdf - PDF Analysis Tool. https://eternal-todo.com/tools/peepdf-pdf-analysis-tool.

[38] ISO. Iso 32000-1:2008 - document management — portable document format — part 1: Pdf 1.7. https://www.iso.org/standard/51502.html, July 2008.

[39] Aviram Shmueli. SentinelOne Detects New Malicious PDF File. http://tinyurl.com/yszazpvy, June 2018.

[40] Kai Lu. Analysis of CVE-2016-4203 - Adobe Acrobat and Reader CoolType Handling Heap Overflow Vulnerability. http://tinyurl.com/4kmp9jb4, July 2016.

[41] Vinh Lam. Ransomware Exploits: Detecting and Exploiting CVE-2008-2992 in Adobe Acrobat Reader. http://tinyurl.com/mv8s7m6p, April 2017.

[42] Didier Stevens. Malicious PDF Documents Explained. *IEEE Security & Privacy*, 9(1):80–82, 2011.

[43] Davide Maiorca, Battista Biggio, and Giorgio Giacinto. Towards Adversarial Malware Detection: Lessons Learned from PDF-based Attacks. *ACM Computing Surveys (CSUR)*, 52(4):1–36, 2019.

[44] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding Bag-of-Words Model: A Statistical Framework. *International Journal of Machine Learning and Cybernetics*, 1:43–52, 2010.

[45] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial Attacks on Neural Network Policies. In *Proceedings of the 5th International Conference on Learning Representations Workshop*, 2017.

[46] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting Adversarial Attacks with Momentum. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '18)*, 2018.

[47] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR '18)*, 2018.

[48] Nedim Šrndić and Pavel Laskov. Practical Evasion of a Learning-Based Classifier: A Case Study. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP '14)*, 2014.

[49] Hung Dang, Yue Huang, and Ee-Chien Chang. Evading Classifiers by Morphing in the Dark. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*, 2017.

[50] Davide Maiorca, Igino Corona, and Giorgio Giacinto. Looking at the Bag is not Enough to Find the Bomb: An Evasion of Structural Methods for Malicious PDF Files Detection. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIA CCS '13)*, 2013.

[51] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3), July 2009.

[52] Durgesh Samariya and Amit Thakkar. A Comprehensive Survey of Anomaly Detection Algorithms. *Annals of Data Science*, 10(3), June 2023.

[53] Charu C. Aggarwal. *Outlier Analysis*. Springer, 2017.

[54] Clement Fung, Shreya Srinarasi, Keane Lucas, Hay Bryan Phee, and Lujo Bauer. Perspectives from a Comprehensive Evaluation of Reconstruction-based Anomaly Detection in Industrial Control Systems. In *Proceedings of the 2022 European Symposium on Research in Computer Security (ESORICS '22)*, 2022.

[55] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep One-Class Classification. In *Proceedings of the 35th International Conference on Machine Learning (ICML '18)*, 2018.

[56] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *Proceedings of the 22nd International Conference on Information Processing in Medical Imaging (IPMI '17)*, 2017.

[57] Yingshui Tan, Baihong Jin, Alexander Nettekoven, Yuxin Chen, Yisong Yue, Ufuk Topcu, and Alberto Sangiovanni-Vincentelli. An Encoder-Decoder Based Approach for Anomaly Detection with Application in Additive Manufacturing. In *Proceedings of the 18th IEEE International Conference on Machine Learning and Applications (ICMLA '19)*, 2019.

[58] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. *arXiv preprint arXiv:1912.06059*, 2019.

[59] Mila Parkour. 16,800 Clean and 11,960 Malicious Files for Signature Testing and Research. http://tinyurl.com/2s36ytve, March 2013.

[60] Maryam Issakhani, Princy Victor, Ali Tekeoglu, and Arash Habibi Lashkari. PDF Malware Detection based on Stacking Learning. In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP '22)*, 2022.

[61] CIC-Evasive-PDFMal2022. https://www.unb.ca/cic/datasets/pdfmal-2022.html.

[62] VirusTotal. https://www.virustotal.com.

[63] Peter Wyatts. A New Stressful PDF Corpus. https://pdfa.org/a-new-stressful-pdf-corpus/, 2020.

[64] Frederick Barr-Smith, Xabier Ugarte-Pedrero, Mariano Graziano, Riccardo Spolaor, and Ivan Martinovic. Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (S&P '21)*, 2021.

[65] APT Malware Dataset. https://github.com/cyber-research/APTMalware.

[66] Zijun Zhang. Improved Adam Optimizer for Deep Neural Networks. In *Proceedings of the 26th IEEE/ACM International Symposium on Quality of Service (IWQoS '28)*, 2018.

[67] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context Encoders: Feature Learning by Inpainting. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CPVR '16)*, 2016.

[68] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85), 2011.

[69] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019.

[70] Cuckoo Sandbox. https://github.com/cuckoosandbox.

[71] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[72] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring Catastrophic Forgetting in Neural Networks. In *Proceedings of the 2018 AAAI conference on Artificial Intelligence (AAAI '18)*, 2018.

[73] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security '19)*, 2019.

[74] Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting Concept Drift in Malware Classification Models. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security '17)*, 2017.

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

[76] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. Transcending Transcend: Revisiting Malware Classification in the Presence of Concept Drift. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (S&P '22)*, 2022.

[77] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *Communications of the ACM*, 63(11), 2020.

[78] Yue Zhao, Zain Nasrullah, and Zheng Li. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

[79] Liang Tong, Bo Li, Chen Hajaj, Chaowei Xiao, Ning Zhang, and Yevgeniy Vorobeychik. Improving Robustness of ML Classifiers Against Realizable Evasion Attacks Using Conserved Features. In *Proceedings of 28th USENIX Security Symposium (USENIX Security '19)*, 2019.

A Anomaly Localization Algorithm

In practice, there can be a shift in the reconstruction of the feature space, meaning that paths corresponding to features with large SD distances may not necessarily exist in the original feature space. Therefore, we need a path-object mapping table to exclude these values. Algorithm 1, after computing Pk, checks whether each anomalous path truly exists by referencing the path-object mapping table.

Algorithm 1 Anomalous Objects Localization

Input: Input sample $\mathcal{S}$; Average SD distance $\mathcal{D}_A$.
Output: Anomalous path-object sets $(\mathbb{P}, \mathbb{O})$

$\mathcal{D} \leftarrow GetSDDistance(\mathcal{S})$
$\mathcal{M} \leftarrow GetPathObjectMap(\mathcal{S})$
$\mathcal{D}_\Delta \leftarrow |\mathcal{D} - \mathcal{D}_A|$
$P_k \leftarrow SelectTopKPath(\mathcal{D}_\Delta, k)$
for $i = 0 \rightarrow length(P_k)$ **do**
    if $P_k[i] \in \mathcal{M}$ **then**
        $O \leftarrow QueryMap(P_k[i], \mathcal{M})$
        $(\mathbb{P}, \mathbb{O}) \leftarrow AddPathObject((P_k[i], O))$
    end if
end for
return $(\mathbb{P}, \mathbb{O})$

B Additional Details of VAPD

We implemented VAPD using TensorFlow and Keras. The VAE component consists of four fully connected layers, each with 256 neurons: the first two layers form the encoder, and the last two form the decoder. The latent space is a 16-dimensional vector. On top of the latent layer, we added a two-layer feedforward network, where each layer has 200 neurons. This network takes the 16-dimensional latent vector as input and produces a softmax score as output. The latent component serves as a complementary scoring mechanism that enhances detection by capturing semantic deviations not fully reflected in reconstruction errors. We trained VAPD for 40 epochs, during which the loss began to converge gradually after epoch 20, as shown in Figure 9. The final anomaly detection threshold used in VAPD is set to 4.43826128e-2.

C Baseline Models Parameters

We compared a total of nine baseline models. Among them, the first three are binary classifiers, while the remaining six

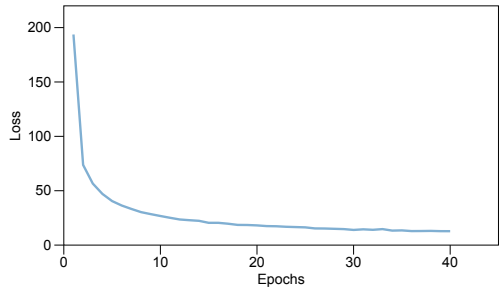


Figure 9: Loss curve of VAPD.

are anomaly detection models that have not previously been applied to PDF malware detection. As a result, no standard hyperparameter settings or tuning guidelines exist for these models in this context. We used the default settings for KNN, matched KPCA's intermediate dimensionality to VAPD's latent space, and adopted the same encoder–decoder structure for AE as in VAPD. For DeepSVDD, AnoGAN, and CE-GAN, we followed the original papers' configurations, adjusting only the input format.

PDFRate and Hidost We adopted the original models from the PDFRate and Hidost papers. PDFRate uses a random forest classifier, while Hidost employs an SVM classifier. In our experiments, we used the same classifier configurations and parameters as specified in the respective original papers.

RTM Chen et al. [17] first built a neural network with two hidden fully connected layers, each with 200 neurons activated by ReLU, and a final layer of Softmax activation. Then, Chen et al. used the robustness properties and symbolic interval analysis method to adversarially retrain a model (RTM) based on neural network.

KNN K-Nearest Neighbors (KNN) is a classic classification-based learning algorithm. If the majority of the k-nearest samples of a given sample in the feature space belong to a particular class, then that sample is also assigned to that class. We used PyOD [78] to implement a base KNN with neighbors of 5 and leaf size of 30.

DeepSVDD Ruff et al. [55] proposed Deep Support Vector Data Description (DeepSVDD), a deep anomaly detection method that trains a neural network to map data into a hypersphere where normal samples are enclosed and anomalies fall outside. We implemented DeepSVDD using two fully connected hidden layers with 200 neurons each, ReLU activation, and a Sigmoid-activated output layer. The model was trained using the Adam optimizer with a learning rate of 0.001.

AnoGAN Schlegl et al. [56] proposed AnoGAN, which detects anomalies by measuring reconstruction loss from a GAN generator trained on normal data. Our implementation uses TensorFlow. The generator includes a fully connected layer ($128 \times 15 \times 15$) and two convolutional layers (kernel size $(5,5)$; 64 and 1 filters). The discriminator has two convolutional layers (64 and 128 filters) and a final dense layer. LeakyReLU is used in hidden layers, Sigmoid in the output, and Adam with a 0.001 learning rate for optimization.

KPCA Kernel Principal Components Analysis (KPCA) is frequently utilized for reducing data dimensionality, enabling reconstruction-based anomaly detection. This process involves mapping data to a lower-dimensional feature space using the KPCA algorithm, followed by reconstructing it back to the original space. We implemented a Kernel Principal Component Analysis (KPCA) algorithm using scikit-learn. We used default RBF as the kernel function, and the dimensionality of the intermediate layer is set to 16.

CE-GAN Pathak et al. [67] proposed Context Encoders, combining an encoder-decoder structure with GANs for image inpainting, demonstrating strong generative capabilities. This architecture can also support reconstruction-based anomaly detection. We reimplemented CE-GAN in TensorFlow for PDF malware detection. The generator is a deep autoencoder with five convolutional layers (neurons: [32, 64, 128, 128, 64]), and the discriminator has three layers (neurons: [64, 128, 256]). LeakyReLU is used for hidden layers, with tanh and Sigmoid activations for generator and discriminator outputs, respectively.

AE We implemented an AE with an encoder and decoder, each composed of two fully connected layers comprising 256 neuronsand a latent dimension of 16. ReLU was used as the activation function, Adam as the optimizer, and the learning rate was set to 0.001. The network architecture, including the number of layers and neurons, is identical to that of VAPD.

D Retrain VAPD with PDFrate Features

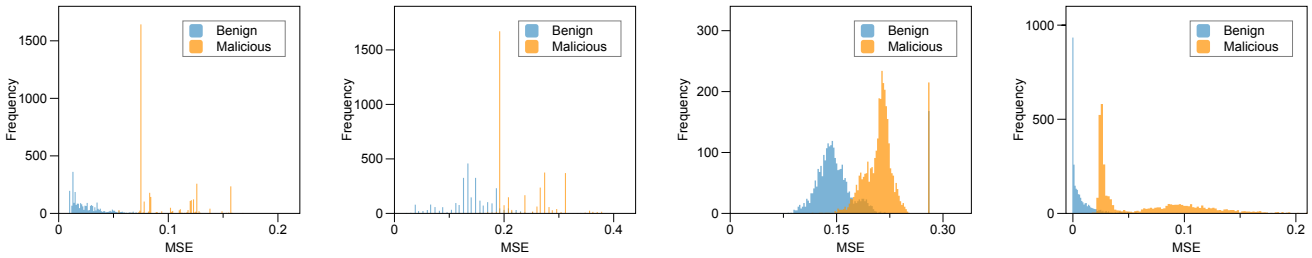
We retrained the VAPD model using the metadata and structural features mentioned in Figure 1 (c), which are also used in PDFrate [12]. PDFRate employs a total of 202 features, but only 135 of them are publicly documented. Similar to prior works [79], we utilize a binarized variant of PDFRate as features that assign 0 when the feature value is 0 and 1 when the feature value is non-zero.

Detection Accuracy We evaluated VAPD and five baseline models using PDFRate features on D1, D2, and D3 datasets, as shown in Table 7. KPCA and AE perform better with PDFRate features than with structural paths. VAPD consistently outperforms all five baselines when using PDFRate features.

Feature Reconstruction We evaluated the reconstruction performance of KPCA, AE, CE-GAN, and VAPD using PDFRate features, following the same settings as in §4. Figure 10 shows the reconstruction error distributions on the D1 test set. All models demonstrate clear discriminative ability, though CE-GAN shows the largest overlap between benign and malicious samples, indicating weaker reconstruction performance. Interestingly, KPCA and AE, which underperformed on structural path features, show improved results on PDFRate features.

Table 7: Comparative evaluation results on the D1 test, D2 custom and D3 evasive.

Model	D1 Test				D2 Custom				D3 Evasive
	Acc (%)	Rec (%)	Prec (%)	F1(%)	Acc (%)	Rec (%)	Prec (%)	F1(%)	DR(%)
KNN PDFRate	45.72	9.72	60.04	16.72	26.13	12.03	17.38	14.21	14.67
DeepSVDD PDFRate	51.51	20.19	75.32	31.84	51.90	19.89	58.03	29.63	19.73
KPCA PDFRate	97.25	98.84	96.35	97.58	65.26	98.72	59.58	74.31	98.25
AE PDFRate	95.51	98.55	93.76	96.10	79.85	97.71	72.37	83.15	95.53
CE-GAN PDFRate	90.13	95.21	88.16	91.55	62.44	94.76	58.03	71.98	93.50
VAPD PDFRate	99.30	99.97	98.79	**99.38**	98.38	99.43	97.44	**98.42**	**98.49**



(a) KPCA PDFRate (b) AE PDFRate (c) CE-GAN PDFRate (d) VAPD PDFRate

Figure 10: Distributions of reconstruction errors of KPCA, AE, CE-GAN, and VAPD with PDFRate.

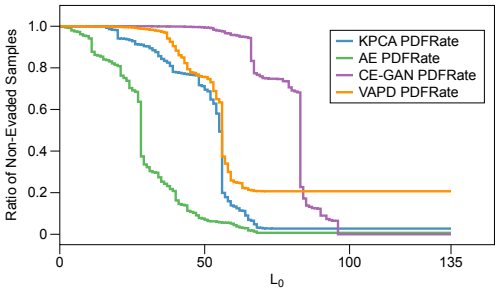


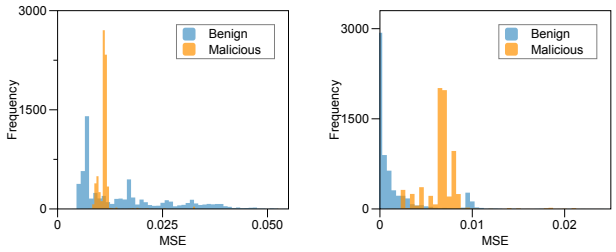
Figure 11: Result of MalGAN attacks.

VAPD maintains strong reconstruction performance, demonstrating its adaptability across feature types. However, as PDFRate features are purely statistical, they cannot support object-level localization.

Robustness We used MalGAN to evaluate the robustness of VAPD, PDFRate, and baseline models with PDFRate features. MalGAN successfully achieved full evasion against AE and CE-GAN. Among the test samples, 770 could not evade VAPD, while 101 failed to evade KPCA. Robustness was further assessed using ERA, as shown in Figure 11. Although CE-GAN required a higher L_0 distance to be evaded than VAPD, it was ultimately bypassed. These results confirm that VAPD remains the most robust model even when using PDFRate features.

E Reconstruction Error Distributions on D1 Training Set

Figure 12 illustrates the distribution of reconstruction errors on the D1 training set for KPCA and AE.



(a) KPCA (b) AE

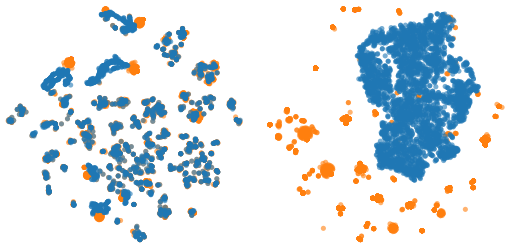
Figure 12: Distributions of reconstruction errors of KPCA and AE on D1 training set.

F Visualization of Reconstruction

Figure 13(a) shows substantial overlap for benign samples before and after reconstruction, indicating minimal reconstruction errors. Conversely, significant dissimilarities are noticeable between the original and reconstructed features of malicious samples, as illustrated in Figure 13(b). This underscores the boundaries between the two categories.

G Adversarial Attack Parameters

Gradient Descent We adopted the unrestricted gradient descent attack setup from [17], allowing perturbation of any binary feature. The attack leverages model output gradients to guide perturbations: features with positive gradients increase classification confidence when set to 1, while those with negative gradients reduce it.



(a) (b)

Figure 13: The feature distributions of benign samples (a) and malicious samples (b) mapped onto a two-dimensional plane after reconstruction by VAPD. Blue and orange dots represent the features mapped onto the two-dimensional plane before and after reconstruction, respectively.

Given the binary feature space, the attack flips bits, 0 to 1 for insertion and 1 to 0 for deletion. Perturbation candidates are selected using both random and greedy strategies. The greedy strategy inserts the zero-valued feature with the highest positive gradient and deletes the one-valued feature with the most negative gradient (by absolute value). We set the maximum number of iterations to 200,000, following [17], to approximate the strongest possible attack.

MalGAN MalGAN employs an substitute discriminator to mimic the original black-box detection model, while the generator produces malicious samples. During training, the substitute discriminator minimizes the loss function below to better approximate the black-box model:

$$L_D = -E_{x \in BB_{Benign}} log(1 - D_{\theta_d}(x)) - E_{x \in BB_{Malware}} log D_{\theta_d}(x)$$

Here, $D_{\theta_d}(x)$ represents the predicted probability of sample x being malicious, BB_{Benign} denotes benign samples identified by the original black-box detection model, and $BB_{Malware}$ represents malware detected by the original model.

The generator, on the other hand, minimizes the following loss function during training to reduce the substitute discriminator's probability of detecting malicious sample:

$$L_G = E_{m \in S_{Malware}, z \sim P_{uniform[0,1)}} log D_{\theta_d}(G_{\theta_g}(m,z))$$

The goal is to generate adversarial samples that deceive the substitute discriminator while resembling real malware. The discriminator's input dimension is 3514 (the feature dimension described in §3.1), with one hidden fully connected layer of 256 neurons and Sigmoid activation. The generator receives the original features and a 100-dimensional noise vector, passes them through a hidden layer with 256 neurons and Sigmoid activation, and outputs a vector matching the original feature dimension (3514). MalGAN uses the Adam optimizer with a learning rate of 0.001, a batch size of 128, and is trained for 50 epochs.

Adaptive EvadeML Compared to the original EvadeML [35], adaptive EvadeML introduces two additional mutation strategies: (1) relocating exploit-triggering paths to alternative execution points, and (2) performing more insertions and deletions under different subtrees of the structural path tree [17]. This attack requires setting up a Cuckoo sandbox environment consisting of a cluster of virtual machines. During dynamic execution, the sandbox captures network communication indicators to verify whether the mutated samples still exhibit malicious behavior.

The mutation setup uses a population size of 48 and evolves over 20 generations, consistent with the genetic algorithm parameters used in [17,35]. After each generation, the mutated samples are submitted to the Cuckoo sandbox for validation. Only samples that remain valid and executable are retained. The $fitness\_score$ of each sample is then calculated using the following formula:

$$fitness\_score = \log(P_{benign}) - \log(P_{malicious})$$

Here, P_{benign} represents the predicted probability that the sample is benign, and $P_{malicious}$ represents the probability that the sample is malicious. A fitness score greater than 0 indicates a successful evasion. If no successful evasions occur in a given generation, the mutant with the highest fitness score is carried forward to the next generation for further evolution.

Reverse Mimicry The reverse mimicry attack operates similarly to JSinject [50], where adversarial samples are generated by injecting malicious payloads into benign PDF files. We utilized the malicious payload dataset provided in [17] for the injection process. The resulting samples were submitted to the previously deployed Cuckoo sandbox for validation, and only those deemed valid and executable were retained.

Table 8: Localization results for case sample.

Top 5 anomalous paths	Related objects
`Pages/Resources/ProcSet` ✗	-
`OpenAction/S`	obj1, obj3
`OpenAction/JS`	obj1, obj3
`OpenAction/JS/Filter`	obj1, obj3, obj12
`OpenAction/JS/Length`	obj1, obj3, obj12, obj15

H Case Study

We extracted the five most anomalous paths and their corresponding objects from the case samples, as detailed in Tables 8. For case sample (MD5: *7abb55433c54f05ad17f677d58eadbf1a26a3612*), the malicious activity involves executing embedded JavaScript code along the `OpenAction/JS/Filter` path, leading to `obj12`, where the script is stored. Using our path-object mapping, we identified a spurious path (marked with ✗ in Table 8) resulting from a reconstruction shift in feature space (Appendix A).