MalProtect: Stateful Defense Against Adversarial Query Attacks in ML-based Malware Detection

Anonymous

Abstract

Machine learning models are known to be vulnerable to adversarial query attacks. In these attacks, queries are iteratively perturbed towards a particular class without any knowledge of the target model besides its output. The prevalence of remotely-hosted ML classification models and Machine-Learning-as-a-Service platforms means that query attacks pose a real threat to the security of these systems. To deal with this, stateful defenses have been proposed to detect query attacks and prevent the generation of adversarial examples by monitoring and analyzing the sequence of queries received by the system. Several stateful defenses have been proposed in recent years. However, these defenses rely solely on similarity or out-of-distribution detection methods that may be effective in other domains. In the malware detection domain, the methods to generate adversarial examples are inherently different, and therefore we find that such detection mechanisms are significantly less effective. Hence, in this paper, we present MalProtect, which is a stateful defense against query attacks in the malware detection domain. MalProtect uses several threat indicators to detect attacks. Our results show that it reduces the evasion rate of adversarial query attacks by 80+% in Android and Windows malware, across a range of attacker scenarios. In the first evaluation of its kind, we show that MalProtect outperforms prior stateful defenses, especially under the peak adversarial threat.

CCS Concepts: • **Computing methodologies → Machine learning**; • **Security and privacy**;

Keywords: adversarial machine learning, malware detection, security

1 Introduction

Machine learning has offered enormous capabilities [6, 27, 46] leading to the large-scale use of Machine-Learning-as-a-Service (MLaaS) (e.g., [4, 29, 91]). This allows users to leverage the power of remotely-hosted ML models by requesting predictions from them [107]. The widespread use of these services means that their reliability and security is paramount, especially considering the threat posed by adversarial machine learning [36, 37, 72, 84, 95], as recent work has shown that these systems are vulnerable to adversarial query attacks [13, 16, 22, 24, 48, 61, 77, 82]. With query attacks, an attacker iteratively perturbs a malware sample based on feedback from the target model that includes its predicted output. Using the feedback, the perturbations to the input sample are tuned across the queries to observe how the target model responds until, eventually, the truly malicious sample is classified as benign (i.e., goodware) [22, 82]. To deal with the adversarial ML threat, several defenses have been proposed based on a variety of approaches (e.g., [21, 75, 78, 83, 87, 95, 98, 99, 102, 103]). However, most of these approaches have been shown to be ineffective against *query* attacks in several domains [16, 22, 48], including malware detection [77, 82, 84].

Recent work has stressed that systems must monitor queries to identify hazards such as adversarial attacks [47]. In fact, ML-based anomaly detection has been regarded as essential to detect the misuse of ML-based systems [17]. To this end, *stateful defenses* have been proposed to protect ML prediction models against query attacks [24] by offering greater system awareness than *stateless* defenses. This is achieved by monitoring queries received by the system. Researchers have hypothesized that sequences of adversarial queries are often abnormally similar to each other, unlike sequences of legitimate queries [24, 51, 60], and often do not fit their distribution [11, 51, 52]. Hence, stateful defenses analyze the sequence of queries made to the system, such as the similarity of feature vectors representing an image. Several stateful defenses have been proposed (e.g., [24, 51, 60, 69, 108]), however, prior to our work, they have not been tested in the malware detection domain, where query attacks can cripple prediction models [82, 84]. The malware detection domain is significantly different from other domains, with more constraints imposed on attackers regarding the discrete representation of feature vectors and the preservation of malicious functionality when generating adversarial examples [14, 84]. Therefore, attackers use techniques that are different from those in other domains to generate adversarial examples [77, 82]. Consequently, stateful defenses that have been applied to other domains may be ineffective when tested in the malware detection domain.

Hence, in this paper, we present *MalProtect*, which is a model-agnostic stateful defense against query attacks in the malware detection domain. Upon receiving a new query, MalProtect employs several threat indicators that conduct an analysis of the queries to predict if there is an attack in progress. Across Android and Windows, we show that MalProtect reduces the evasion rate of query attacks by 80+%, and we compare this to the meager performance of prior stateful defenses across a range of attack scenarios in the first evaluation of its kind. At a high level, we show that MalProtect produces interpretable decisions. We also show that, despite being designed to protect against query attacks, MalProtect affords a degree of protection against other types of attacks, such as transferability attacks. Furthermore, we demonstrate that an adaptive attacker — with complete knowledge of MalProtect — cannot achieve significant evasion. In summary, we make two key contributions:

1. We propose the first stateful defense against adversarial query attacks in the malware detection domain. Our defense, MalProtect, uses several threat indicators to reliably detect query attacks and prevent the generation of adversarial examples.
2. We provide the first evaluation of several existing stateful defenses applied to the ML-based malware detection domain. The results show that MalProtect significantly reduces the evasion rate of query attacks across Android and Windows under a range of threat models and attacker scenarios.

2 Background & Related Work

ML-based Malware Detection. ML-based malware detection has rapidly grown in popularity. It allows for unseen and unknown threats to be discovered and offers performance that surpasses rule-based and signature-based approaches [84]. ML-based malware detection classifiers decide if a query is benign (i.e., goodware) or malware. These models are trained on binary feature vectors representing the presence or absence of features in executables [45, 84]. The quality of the predictions produced is determined by these features, which can include the usage of API calls, the network addresses accessed, or the libraries used [2, 32, 44, 84].

Adversarial ML Attacks. However, ML models (i.e., prediction models) are vulnerable to adversarial ML attacks. Specifically, adversarial examples can be developed by an attacker, which are queries that are designed to evade a classifier. Even without direct access to the target model, an attacker can perturb a malware sample to have it classified as benign and evade the prediction model [95], through transferability attacks and query attacks. With transferability attacks, an estimation of the target model is developed and attacked in anticipation that any generated adversarial examples will *transfer to* the target model [72]. This is based on the transferability property of adversarial examples [95].

Meanwhile, query attacks generate an adversarial example by iteratively perturbing an input sample towards the desired class based on feedback received from the target model until evasion is achieved [22, 84]. This attack can be conducted in a complete black-box manner without any knowledge about the target model besides the predicted outputs. In the image recognition domain, recent work has introduced several query attacks using techniques such as gradient and decision boundary estimation that perturb the continuous feature-space of images [13, 16, 22, 24, 48, 61]. However, in the malware detection domain, the functionality of software executables must be preserved, and discrete feature vectors must be used when generating adversarial examples [45, 82, 84]. For example, a feature representing an API call (e.g., $CreateFile()$) cannot be perturbed continuously by an attack (e.g., $CreateFile()$+0.05); rather, a completely different feature must be used [82] that offers the same functionality. Therefore, software transplantation-based techniques have been proposed to achieve evasion that involve using features from benign samples to perturb a malware sample [77, 82].

Defenses Against Adversarial ML. Many defenses against adversarial attacks have been proposed. These include gradient-based [75, 98], feature-based [99, 103] and randomization-based [87, 102] approaches, as well as techniques based on adversarial training [95, 98]. Most approaches are single-model defenses, where an individual model is made robust [71]. Ensemble defenses and moving target defenses have also been proposed, which use several models with some particular method to return predictions [5, 28, 30, 80, 87, 89, 92]. However, recent studies [10, 19, 78] have shown most of these defenses to be ineffective against query attacks [24, 82, 84]. In fact, as these defenses are *stateless*, they cannot store, monitor, or analyze the sequence of queries received, allowing query attacks to succeed.

Stateful Defenses. To protect ML prediction models against query attacks, *stateful defenses* have been proposed [11, 24, 24, 51, 52, 60, 65]. In essence, these stateful defenses conduct a form of anomaly detection, which has been recommended in recent studies as a method to combat adversarial query attacks [17, 40, 47]. It has been hypothesized that because query attacks iteratively perturb an input sample, they produce a sequence of abnormal queries that have high similarity or are outside the distribution of legitimate queries. Figure 1 shows that stateful defenses retain the sequence of queries received by the system and analyze them using different techniques to detect attacks. This is similar to intrusion detection systems and firewalls that monitor network activity to detect threats [15].

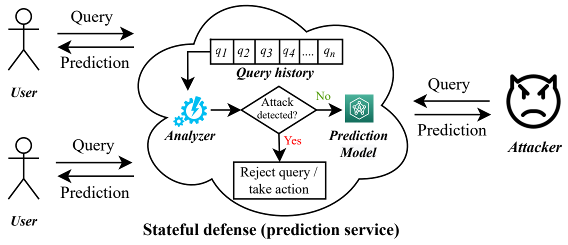


Figure 1. Overview of a stateful defense.

Limitations of Existing Stateful Defenses. A number of stateful defenses have been proposed in recent years [24, 51, 52, 60, 108]. Subsequent work has identified several limitations of these defenses, including insufficient similarity detection mechanisms, poor maintenance of the query history, and weaknesses of account-based detection, among others [47, 60, 69, 108]. More specifically, Chen et al.'s Stateful Detection approach (SD) is an account-based defense that maps a user's query to a feature vector and measures the average distance between it and its k-nearest neighbors. If this is below a threshold, an attack is detected, which resets the query history and cancels the user's account. However, this can be defeated by a Sybil attack, where an attacker uses multiple accounts in the attack [34, 104]. Moreover, inspecting queries per-account limits the detection scope [60]. Alternatively, Li et al. present Blacklight [60], which measures the similarity of queries for all users using the L_2 distance (i.e., Euclidean distance), which is a type of L_p norm [20]. An attack is detected if the Euclidean distance between several queries (represented by hashes) is below the threshold. Other distance-based defenses have been proposed (e.g., [65]), though these can all be evaded when queries are intentionally designed to be dissimilar [47] or through query-blinding attacks [108].

Out-of-distribution (OOD) detectors have also been proposed to combat query attacks. These check whether a query belongs to the distribution of the target model's training data [11, 52]. When OOD queries are detected, Kariyappa et al. propose returning inconsistent labels to those detected OOD queries [52]. However, an attacker may be able to construct adversarial queries that remain within the required distribution [107, 108]. PRADA [51] is a defense against model extraction attacks, which is a related problem for which other stateful defenses have also been proposed as solutions (e.g., [108]). Prior work has found that it can also detect evasion attacks [24]. PRADA is based on the assumption that the L_2 distance among benign queries follows a normal distribution. This is monitored using the Shapiro-Wilk normality test. However, manipulation of the query distribution has been found to evade it [24, 107].

Our defense, MalProtect, is model-agnostic allowing it to be used with any underlying prediction model. To detect attacks, it utilizes techniques beyond basic similarity and OOD detection, such as analyzing the autoencoder loss of queries, examining the distribution of enabled and shared features across queries, and more. We next present the threat model we consider for our work, followed by a detailed description of MalProtect.

3 Threat Model

In our work, attackers aim to evade a feature-based ML malware detection classifier so that their malicious queries are misclassified as benign. This is a well-established threat model in this domain [14, 44, 45, 88, 94, 105].

Target Model. The target model is a remotely-hosted malware classification system that predicts whether an input sample belongs to the benign or malware class. We refer to the classification system as the oracle O. The oracle has two principle components: the underlying classifier F (i.e., the prediction model) and the MalProtect component that protects F by analyzing queries to detect attacks (see Section 4 later). To train F, input samples are represented as binary feature vectors using the features that are provided by a dataset. With the features $1...M$, a feature vector X can be constructed for each input sample such that $X \in \{0,1\}^M$, similar to previous work [45, 78, 80]. The presence or absence of a feature i is represented by either 1 or 0 within X. Like prior work on ML-based malware detection [45, 58, 82], the features we employ in our work include, among others, libraries, API calls, permissions, and network addresses; these are provided by the datasets we use (see Section 5 later). With their associated class labels, several feature vectors can be used to train the binary classification model that we refer to as the classifier F. Then, when a user makes a query to O, a prediction is made and returned. For the predicted outputs, we use 1 for the malware class and 0 for the benign class. The oracle O returns completely scoreless feedback [48].

Attacker's Goal. The goal of the attacker is to generate an adversarial example X' from a malware sample X to evade the oracle O and obtain a benign prediction. Suppose $O : X \in \{0,1\}^M$ and we have a function $checkFunc()$ to check the functionality of X. We can summarize this as:

$$checkFunc(X) = checkFunc(X'); O(X) = 1; O(X') = 0 \quad (1)$$

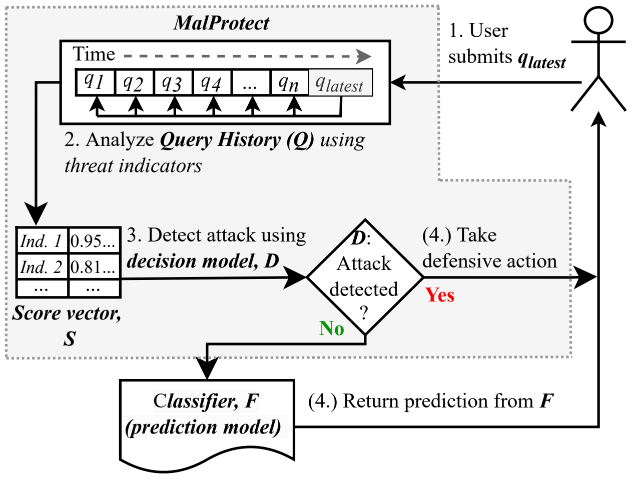
Attacker Capabilities & Knowledge. We model three types of attackers that have been featured in prior work [14, 48, 55, 74, 80]. We consider the *gray-box attacker* who has limited knowledge, has access to the same training data as the classifier F and is aware of the feature representation and the statistical representation of the features across the dataset. However, the attacker is unaware of F's parameters, configurations, and other pertinent information. This represents a scenario similar to when some model data has been leaked. To perform query attacks, this attacker applies perturbations to the feature vector through a software transplantation-based approach [77, 82], guided by their knowledge of the dataset. In contrast, the zero-knowledge *black-box attacker* can only observe the predictions received for their queries and has no information about the target model but does have some information pertaining to the kind of feature extraction performed (e.g., the static analysis that a malware detection classifier may consider). This attacker also uses a software transplantation-based approach, but with considerably less information. Neither attacker is aware that MalProtect is a part of O. We also consider an *adaptive attacker* (*white-box attacker*) who has complete knowledge of the target model and knows that MalProtect is in place. Therefore, this attacker tries to evade all components of the oracle O with a specific, tailored attack.

We assume that all attackers possess ample computing power and resources to submit thousands of queries. Furthermore, unlike previous work [24, 51], we assume that attackers may control multiple user accounts and IP addresses, as in a Sybil attack [34, 104].

4 MalProtect

MalProtect is designed to protect ML models against adversarial query attacks that take place in the malware detection domain's feature-space. Figure 2 shows that MalProtect forms a layer of protection that monitors queries before they reach the prediction model F, which is the ML-based malware detection classifier to be protected. This allows MalProtect to detect adversarial queries that are designed to evade the classifier.

When a user submits a query (Step 1), MalProtect employs several *threat indicators* to analyze the sequence of queries for attack detection (Step 2). This is similar to a system that makes decisions using data from multiple sensors [47]. The threat indicators use different criteria, principally driven by an anomaly detection-based approach. This includes examining aspects such as the autoencoder loss of queries, the distance between queries, the features shared across queries, and whether other characteristics of queries fit within different distributions. Based on the state of the query history, each indicator produces a score that reflects the likelihood of an attack in progress according to that indicator, with adversarial queries expected to cause higher scores. A *decision model* then takes the indicator scores as its input and predicts whether there is an attack in progress (Step 3). If an attack is detected, MalProtect takes some defensive action. If no attack is detected, the prediction model F returns a prediction for the user's query (Step 4). Not only do the scores assist in interpreting how MalProtect produces a particular decision, but as we show later, each threat indicator can be analyzed to determine its influence on the final decision. Next, we provide details about each of MalProtect's core components.



MalProtect
Time
q_1 q_2 q_3 q_4 ... q_n q_{latest}
1. User submits q_{latest}
2. Analyze Query History (Q) using threat indicators
Ind. 1 0.95...
Ind. 2 0.81...
... ...
Score vector, S
3. Detect attack using decision model, D
D: Attack detected ?
(4.) Take defensive action
Yes
No
Classifier, F (prediction model)
(4.) Return prediction from F

Figure 2. Overview of MalProtect.

4.1 Query History

MalProtect analyzes the query history to detect adversarial behavior. When a user requests a prediction for their query q_{latest}, it is appended to the query history Q. Queries can be represented more sparsely in the malware detection domain to reduce storage overheads, which has been cited as an issue in other domains [60]. The depth of analysis performed by MalProtect and the size of Q is controlled by the resources possessed by the defender when MalProtect is deployed, but, as we show later in Section 10, MalProtect needs very little spatial and computational resources to offer a very good performance. Unlike other stateful defenses that reset the query history once they reach capacity (e.g., [24]), MalProtect retains queries as a sliding window to maximize its detection capability. Finally, MalProtect does not discern between queries from different users, as an attacker can conduct a Sybil attack to evade account-based defenses [34, 104].

4.2 Indicators for Analyzing Query History

After adding q_{latest} to Q, MalProtect analyzes the query history using multiple threat indicators to detect attacks through an anomaly detection-based approach. In conventional anomaly detection, thresholds are employed to assess whether a sample meets the criteria for being abnormal. However, with MalProtect, we use a data-driven approach where the weights of each indicator are learned by a decision model to predict an attack. That is, instead of each indicator deciding if an attack is taking place, each indicator produces a score between 0 and 1 that reflects the likelihood of an attack according to its criteria. Scores are then aggregated into a score vector and passed to a decision model that decides if an attack is in progress. This allows for some degree of interpretability too, as we show later, because we can understand both which indicators seem to play the biggest role in the decision model and which indicators produce the highest scores (giving clues about what is anomalous). MalProtect is based on the principle that adversarial queries are expected to cause higher indicator scores as they are anomalous compared to legitimate queries and the training data distribution. To conduct the analysis and produce scores, some indicators use standard anomaly detection techniques, such as the empirical rule [79], to assess whether a query fits the distribution of data. Meanwhile, other indicators use information about the training data to examine how much queries deviate from various measures. We next present each indicator:

Indicator 1. Distance between queries: This indicator assesses the similarity of q_{latest} to other queries in Q, which is the primary method adopted by some stateful defenses (e.g., [24, 60]). A smaller distance between queries is an indication of iterative perturbations during a query attack as an attacker makes modifications to their queries to assess how the target model responds [22, 24]. This indicator therefore examines how anomalous the minimum distance between q_{latest} and other queries in Q is (represented by *minDistQ*). Because the malware detection domain uses binary feature vectors (as opposed to other domains like image recognition), it has been suggested that the L_0 distance is the most appropriate distance measure [20, 78], as it measures the number of features that are different between two feature vectors. Other norms such as L_∞ would always have a distance of 1 if at least one change was made [78].

To assess the potential abnormality of *minDistQ* without using a threshold and also to normalize it between 0 and 1, we put *minDistQ* in the context of the average L_0 distance of all samples in the training data (represented by *avgDistD*). If *minDistQ* is significantly below this, it implies that q_{latest} is similar to another query in Q beyond the norm. The score for this indicator is based on the percentage change between *minDistQ* and *avgDistD*. A smaller percentage change means that there exists a query whose similarity to q_{latest} is significantly lower than the average of the training data (*avgDistD*). So that a higher threat is represented by a higher score value, we negate the result. Thus, the score is calculated as follows:

$$S_1 = -1 \cdot (minDistQ - avgDistD) \ / \ avgDistD \tag{2}$$

Indicator 2. Enabled features shared across queries: This indicator assesses whether q_{latest} shares a significant number of enabled features with another query in Q beyond the norm. The rationale behind this is that an attacker could enable unused features in q_{latest} to intentionally increase the L_0 distance between queries in order to evade basic similarity detection (e.g., while trying to evade Indicator 1). However, some features must *always* remain enabled to preserve the original malicious functionality across a query attack. Thus, queries with the same enabled features would imply similarity specifically in this domain. Therefore, the highest number of enabled features shared between q_{latest} and another query in Q is calculated as *maxSharedQ*.

We then use the average number of enabled features shared between samples in the training data (*avgSharedD*) to assess to what extent *maxSharedQ* may be abnormal. This is achieved by calculating the deviation of *maxSharedQ* from this mean, as a percentage change. A higher value indicates that q_{latest} shares an abnormally high number of enabled features with some other query in Q considering the training data's distribution. The score is calculated as follows:

$$S_2 = (maxSharedQ - avgSharedD) \ / \ avgSharedD \tag{3}$$

Indicators 3A & 3B. Number of enabled features: An attacker could rapidly traverse the decision boundary by enabling a substantial number of benign features in q_{latest} at once. An anomalous number of enabled features could indicate an attack attempt. The indicator 3A is the percentage change between the number of enabled features in q_{latest} ($|q_{latest}|$) and the average number of enabled features in training data samples (*avgFeaturesD*). A large percentage change implies that far more features are enabled in q_{latest} compared to *avgFeaturesD*. The score is calculated as follows:

$$S_{3A} = (|q_{latest}| - avgFeaturesD) \ / \ avgFeaturesD \tag{4}$$

The indicator 3A only assesses whether $|q_{latest}|$ is anomalous considering the training data and does not compare with other queries in Q. However, it is also useful to assess whether $|q_{latest}|$ is anomalous considering the distribution of Q regardless of the training data, similar to previous works (e.g., [11, 51, 52]), as most queries are expected to be legitimate. Therefore, we also use the indicator 3B, which assesses to what extent the number of enabled features in q_{latest} is anomalous considering the queries in Q. This uses the empirical rule (i.e., 3-σ rule) [79], which is a standard technique in anomaly detection. The score is calculated as the percentage change from 3 standard deviations of the mean using the following equation, where $\mu_{FeaturesQ}$ is the mean number of enabled features of queries in Q and $\sigma_{FeaturesQ}$ is the standard deviation:

$$C = \mu_{FeaturesQ} + (\sigma_{FeaturesQ} \times 3) \tag{5}$$

$$S_{3B} = (|q_{latest}| - C) \ / \ C \tag{6}$$

Indicators 4A & 4B. Distribution similarity via autoencoder reconstruction loss: An autoencoder is a neural network where the input and output are the same but the hidden layers have fewer neurons [86]. This limits the amount of information that can travel through the model, requiring it to learn a compressed version of the input. Input samples that belong to the distribution of the autoencoder's training data will produce a smaller distance between the input and output representations (known as the *reconstruction loss*) [70]. Thus, an autoencoder trained on legitimate queries will produce a significantly higher reconstruction loss for an adversarial query. Other novelty detection techniques, such as one-class SVMs, have been found to be outperformed by autoencoders in this task in other domains [41].

Hence, indicator 4A's score is the percentage change between the reconstruction loss of q_{latest} ($RecLoss_{q_{latest}}$) and the maximum reconstruction loss observed of training data samples (*maxRecLossD*). A higher percentage change would suggest a significant increase from the maximum reconstruction loss observed for training data samples, and thus an indication that q_{latest} may be adversarial. The score is calculated as follows:

$$S_{4A} = (RecLoss_{q_{latest}} - maxRecLossD) \ / \ maxRecLossD \tag{7}$$

As with Indicator 3B, we also consider the case where q_{latest} is only compared to other queries in Q regardless of the training data distribution. In this way, the indicator 4B uses the empirical rule to assess whether the reconstruction loss of q_{latest} is anomalous considering the queries in Q. The score is calculated as a percentage change from 3 standard deviations using the following equation, where $\mu_{RecLossQ}$ is the mean reconstruction loss of queries in Q and $\sigma_{RecLossQ}$

is the standard deviation:

$$C = \mu_{RecLossQ} + (\sigma_{RecLossQ} \times 3) \tag{8}$$

$$S_{4B} = (RecLoss_{q_{latest}} - C) \;/\; C \tag{9}$$

4.3 Attack Detection

Once scores are produced by each indicator, they are aggregated into the score vector S. The next step is to make a final decision using all the scores. Static techniques such as a weighted sum model or sum of squares model can combine several individual scores [38]. However, a key drawback of these methods is that the weights for each indicator (i.e., how much each indicator contributes to the overall score) must be selected manually. Moreover, static aggregation models cannot detect trends and patterns in data.

Hence, MalProtect uses a *decision model* (e.g., a neural network) that makes a prediction on the score vector. Each threat indicator is a feature of this model, with the predicted output representing whether there is an attack in progress. Importantly, MalProtect allows analysts to understand why it has made a particular decision with its scoring system. Analysts can *further* understand the influence of each threat indicator on the final prediction by examining the global feature importance of the decision model. Later, in Section 5, we present the decision models that we evaluated, and we then discuss how each threat indicator influences the predictions made by these models in Section 8.

4.4 Defensive Action & Prediction

If no attack is detected by the decision model, q_{latest} is passed on to the classifier F. The classifier makes a prediction of whether q_{latest} is benign or malware, which is then returned to the user. That is, if the latest query received is not considered to be part of an attack by MalProtect, the query is then forwarded to the ML classifier trained to decide whether an input sample is benign or malware. Note that the classifier can itself be *hardened* using techniques such as adversarial training (as shown later). MalProtect only acts as a filter, which only lets what it considers to be *legitimate* queries to be passed on to the ML model used for malware classification.

If MalProtect does detect an attack, a defensive action can be taken, such as returning a specific prediction, returning inconsistent labels, notifying system administrators, banning user accounts, or rejecting further user queries. In this paper, if an attack is detected, we take the defensive action of returning a *malware* prediction. This is as if the query had actually been forwarded to the classifier and that had been its output. This means that an attacker would encounter failure throughout the course of the attack, believing that their query is being consistently classified as malware, without necessarily knowing about MalProtect's presence. However, a very interesting line of future work would be to study what defensive actions could be more or less effective depending on the specific setting.

5 Experimental Setup

Datasets. Sampling from the true distribution is a challenging and open problem in many ML-based security applications [7, 9]. Particularly, the lack of publicly accessible, up-to-date datasets in the malware detection domain is a well-known issue that limits the remits of academic research in this domain. To mitigate this, we use three datasets representative of different architectures as well as collection dates and methods that have been used in a variety of studies (e.g., [12, 31, 45, 50, 54, 76, 77, 101]). The datasets we use are AndroZoo about Android malware [3], SLEIPNIR about Windows malware [2], and DREBIN about Android malware [8]. As we show later, the results across each of the datasets are consistent with each other, indicating that MalProtect's performance transcends the characteristics and nuances of the datasets.

The AndroZoo dataset [3] contains Android apps from 2017 to 2018, offering apps from different stores and markets with VirusTotal summary reports for each. Similar to prior work [66, 76, 77], we consider an app malicious if it has 4 or more VirusTotal detections, and benign if it has 0 VirusTotal detections (with apps that have 1-3 detections discarded). The dataset contains $\approx$ 150K recent applications, with 135,859 benign and 15,778 malicious samples. To balance the dataset, we use 15,778 samples from each class. Meanwhile, SLEIPNIR consists of 19,696 benign and 34,994 malware samples. This dataset is derived from Windows API calls in PE files parsed by LIEF [96]. As our work is in the feature-space, we use SLEIPNIR as a representation of Windows out of simplicity due to its feature space, which is binary. This permits a more precise comparison between the Android and Windows datasets. We use 19,696 samples from each class. Meanwhile, DREBIN is based on extracted static features from Android APK files. The dataset contains 123,453 benign and 5,560 malware samples, from which we use 5,560 samples from each class. As in prior work [32, 45], and for completeness, we use a large number of features for each dataset, i.e., 10,000 for AndroZoo, 22,761 for SLEIPNIR, and 58,975 for DREBIN.

Initially, the datasets are partitioned with an 80:20 ratio for training and test data according to the Pareto principle. This training data is partitioned again into training and validation data using the same ratio, producing a 64:16:20 split that has been extensively used in prior work (e.g., [56, 63, 81]). We use the training data to construct the prediction models used in our evaluation. Meanwhile, we use a partition of randomly-chosen malware samples from a subset of the test data as the input samples for the query attacks, with 234 samples for AndroZoo, 230 for SLEIPNIR, and 229 for DREBIN.

We also consider the well-established guidelines for conducting experiments related to malware detection [85]. As the prediction models in our evaluation decide whether an input sample is benign or malicious, we deem it necessary to retain benign samples in the datasets. As our models separate benign inputs from malicious ones, we do not strictly balance datasets over malware families but instead over the positive and negative classes. We randomly sample unique samples from each class to appear in the training and test data without repetition [2, 78, 84].

MalProtect Configurations. We develop two MalProtect configurations to showcase our defense's capabilities. Each configuration uses a different decision model to predict an attack based on the indicator scores. For both configurations, we cap the query history size at 10,000. Later in Section 10, we discuss the overheads related to this.

To build each MalProtect configuration, we follow the steps described in Section 4 for each dataset. For some indicators, we derive certain values from the dataset that are needed to assess queries (e.g., $avgDistD$, which is the average L_0 distance of samples in the training data). Then, to train each decision model, a defender could use real-world attack and score data. However, in its absence, a synthetic dataset can be used. Therefore, we develop a synthetic dataset of 1000+ labelled samples. For this, we initialize the query history to simulate past user activity with a randomly-chosen set of input samples from the training data. We then supply a range of queries (both legitimate and adversarial, as part of attacks) from our training data to have each indicator produce scores. This produces the synthetic dataset, where each threat indicator is a feature and the class labels represent the true state of the system at the time (whether it was under attack (1) or not (0)). As decision models, we then train a *logistic regression model* for one configuration (MalProtect-LR) and a *neural network* for another (MalProtect-NN) (see Appendix A for architectures) to predict if an attack is in progress when given a score vector. Theoretically, both models should offer accurate predictions [35].

Other Stateful Defenses. We compare MalProtect with three other stateful defenses that have been applied in other domains to detect query attacks. The L_0 defense is based on similarity detection using L_p norms (e.g., [60]); PRADA [51]; and Stateful Detection (SD) [24]. Although we keep the implementations of these stateful defenses as close to the original as possible (e.g., procedures and parameters), techniques applicable to other domains (e.g., encoding queries to generate a feature vector) are not applicable to the malware detection domain. We provide the technical details of each stateful defense in Appendix B. Recall that stateful defenses themselves do not provide predictions; they protect prediction models.

Prediction Models (Non-stateful Defenses). In our evaluation, we use six *non-stateful* defenses as the prediction models. Rather than using vanilla models, we use single-model defenses, ensemble defenses, and moving target defenses as the prediction models to demonstrate their vulnerability to attacks despite being defenses in their own right. Each prediction model is constructed according to the procedures outlined in their original papers using the training data for each dataset (see Appendix C for information about architectures). For single-model defenses, we test a neural network with defensive distillation (NN-DD) [75]. We use several white-box attacks to develop adversarial examples for a set of vanilla models (see later subsection), which are used to adversarially-train a neural network (NN-AT). We adversarially-train with a quantity of adversarial examples that is 25% of the size of the training data [98]. For ensemble defenses, we test majority voting and veto voting [26, 80, 80, 89, 100, 106]. For moving target defenses, we evaluate Morphence [5] and StratDef [80]. Each prediction model is evaluated against attacks as-is; that is, in a non-stateful setup. Then, each prediction model is combined with each stateful defense, allowing us to evaluate different setups.

Performing Query Attacks. When generating an adversarial example in the malware detection domain, there are specific constraints that must be considered. Primarily, the malicious functionality of the original malware sample must remain intact, but additionally, the feature vectors must remain discrete (i.e., consist of 0s and 1s) as they represent the presence or absence of each feature. Query attacks designed for other domains (e.g., [22]) do not consider these constraints and therefore produce ineffective adversarial examples for this domain [82] (see Appendix G). In the malware detection domain, however, query attacks can use software transplantation-based techniques to perturb the features of a malware sample using features from benign "donor" samples. In our evaluation, we modify query attack strategies that have been proven successful in this domain [82] (see Appendix F for full details). Prior to starting an attack, we initialize the query history to simulate past user activity (as explained before). Then, using malware samples from our test set, we apply each attack strategy accordingly under each threat model to perturb malware feature vectors. We use the parameter n_{max} to govern the maximum number of allowed queries permitted, where the transplantation of features continues until the target model is evaded, n_{max} is reached, or the donor features are exhausted. We apply a procedure to preserve the functionality of the original malware sample in the feature-space. For this, the valid perturbations for each dataset are determined by consulting industry documentation and prior work [1, 2, 54, 57, 58, 77, 80]. If a feature has been modified invalidly — that is, the functionality has not been preserved in the feature-space — it is restored to its original value. This is to offer a lower bound of functionality preservation within the feature-space, similar to prior work [45, 58, 88].

AndroZoo and DREBIN permit feature addition and removal (see Appendix E). For SLEIPNIR, only feature addition is possible because of the processing performed by LIEF to extract features when originally developing the dataset. While we remain in the feature-space (like recent work e.g., [58, 88]), the perturbations could be translated to the problem-space using techniques from previous work (e.g., [77, 82]). For example, feature addition can be achieved by adding dead-code or by using opaque predicates [67, 77]. Feature removal — which is more complex — can be performed by rewriting the dexcode, encrypting API calls and network addresses (removing features but retaining functionality).

Performing Other Attacks. Prior work has evaluated a *query blinding attack* against stateful defenses using techniques specifically for the image recognition domain [24, 60, 108] (e.g., modifying contrast, brightness, or rotation). These techniques cannot be applied directly to the malware detection domain as it does not use continuous features and functionality must be preserved. Therefore, we present an adaptive attack instead, where we examine how MalProtect performs against a *white-box attacker* with full knowledge of how MalProtect operates, aiming to evade both MalProtect and the underlying classifier (see Section 8 later).

We also generate adversarial examples through a transferability attack: to adversarially-train models as a defender; and, to evaluate defenses further under different system conditions (see Section 9 later). For this, we construct a set of vanilla models using the training data (see Appendix D for architectures). Then, we apply a range of white-box attacks against these vanilla models (BIM [53], Decision Tree attack [43], FGSM [33, 42], JSMA [73] and SVM attack [43]). As these attacks are not designed for the malware detection domain, they perturb features without considering the constraints of this domain resulting in continuous values in the feature vector. Therefore, we apply a procedure to preserve the functionality of the original malware sample in the feature-space and discretize the generated feature vectors, as these are core constraints in this domain. Once the attack is conducted, each value in the feature vector is discretized (i.e., if the value in the feature vector is < 0.5, it is set to 0, else it is set to 1). Then, if a feature has been modified invalidly, it is restored to its original value (similar to the process for query attacks) so that its malicious functionality is preserved in the feature-space.

6 Black-box Query Attack Results

In this scenario, the attacker's only capability is that they have access to the predicted output of the oracle. We conduct the black-box query attack against each *non-stateful* defense as is, to establish the baseline performance of the prediction models. We then evaluate and compare each combination of stateful defense and prediction model, including both MalProtect configurations.

Figure 3 shows the evasion rate of the black-box query attack versus n_{max}. Against non-stateful defenses (that is, the prediction models as they are), the black-box query attack can achieve a 100% evasion rate in some cases, with the average evasion rate sitting at 70+% across the datasets. These results clearly demonstrate that the non-stateful defenses do not provide adequate protection in this attack scenario, despite being defenses in their own right. Out of the prediction models, only veto voting limits the effectiveness of the attack to some extent, with the attack achieving a maximum evasion rate of $\approx 50\%$ across the datasets. Despite this, around one in two to three queries can still achieve evasion against this model.

Meanwhile, our stateful defense, MalProtect, significantly decreases the attack success, with peak reductions in the evasion rates for AndroZoo, SLEIPNIR, and DREBIN of 84%, 96% and 98%, respectively. This highlights the benefits (and necessity) of a stateful defense for this domain. Comparing the two MalProtect configurations, MalProtect-LR and MalProtect-NN exhibit comparable performance, with both configurations able to detect attacks after 5, 7, and 2 queries for AndroZoo, SLEIPNIR, and DREBIN, respectively. This level of performance is achieved without compromising other metrics, such as the accuracy or false positive rate on legitimate queries (as we show in Section 9 later).

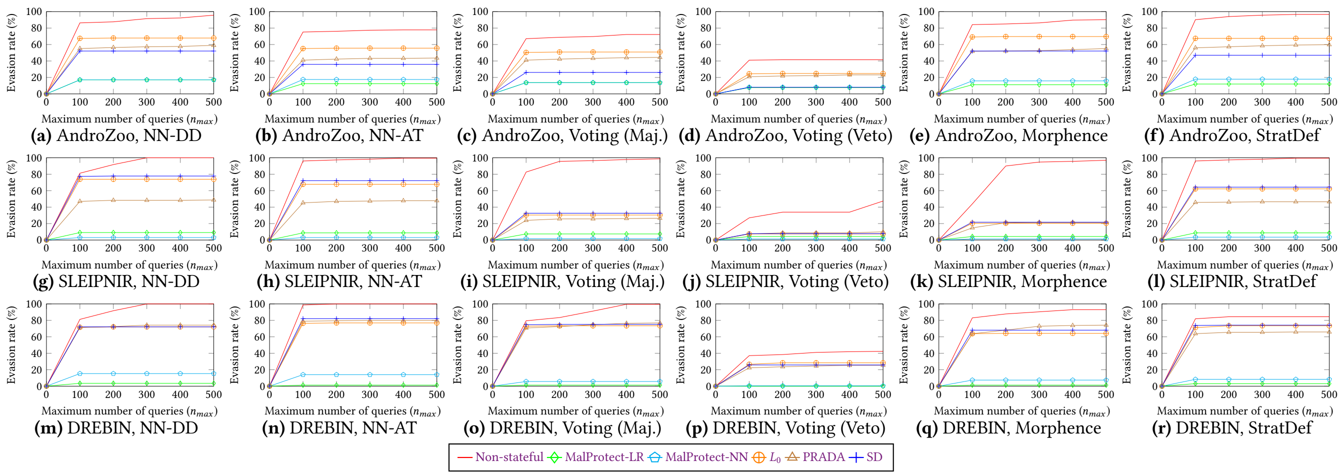


Figure 3. Evasion rate vs. n_{max} of black-box query attack against non-stateful defenses (prediction models) and each combination of prediction model and stateful defense.

Meanwhile, the other stateful defenses that we evaluate are significantly less effective against this attack. For AndroZoo, L_0 offers the mildest performance in terms of adversarial robustness, while SD offers slightly improved performance, but only marginally. In the attack's weakest performance against veto voting, we find that SD performs similarly to MalProtect for AndroZoo and SLEIPNIR, but its performance is much inferior in other cases (with other prediction models) where MalProtect demonstrates robustness. For SLEIPNIR, PRADA offers better performance than the prior stateful defenses, though only marginally in most cases, while L_0 and SD seem similar in their defensive robustness. Meanwhile, for DREBIN, we observe that the three prior stateful defenses exhibit similar performance in all cases. That is, the black-box query attack can cripple most configurations involving prior stateful defenses, as they behave similarly to the non-stateful defenses. Overall, the black-box query attack can achieve a 60+% evasion rate in most cases against prior stateful defenses, compared with a peak evasion rate of ≈ 18% for MalProtect. MalProtect provides demonstrably better protection against the black-box query attack.

7 Gray-box Query Attack Results

Prior work has stressed the importance of evaluating defenses against stronger adversaries [19]. Therefore, we next evaluate MalProtect's performance against the gray-box query attack and compare it with other stateful defenses in that attack scenario. Under this scenario, the attacker has access to the training data of the prediction models and has more information about the features. The gray-box attack strategy therefore utilizes information about the frequency of features in benign samples to determine the order of transplantation, resulting in a more effective attack. As before, we conduct the attack against each non-stateful defense to establish a baseline performance, and then against each combination of stateful defense and prediction model.

Figure 4 shows the evasion rate of the gray-box query attack against all combinations of stateful defenses and prediction models, versus n_{max}. Against the non-stateful defenses, this stronger attack achieves significantly greater evasion across all the datasets, with 100% evasion rate in many cases. Perhaps the greatest difference from the black-box query attack can be observed in the results against veto voting, which was the least evaded non-stateful defense in Section 6. Whereas the black-box query attack peaked at ≈ 40% against this model (without any stateful defense), the gray-box query attack achieves 60 + % evasion rate.

As far as protection by stateful defenses is concerned, MalProtect performs the best. In the best case for the configurations, it reduces the average evasion rate across all models from a peak of 84% to 13% for AndroZoo, 94% to 5% for SLEIPNIR, and 91% to 6% for DREBIN, while only taking ≈ 6, 6, and 2 queries to detect an attack for AndroZoo, SLEIPNIR, and DREBIN, respectively. Like the black-box query attack, MalProtect-LR and MalProtect-NN offer similar performance, with MalProtect-NN slightly outperforming MalProtect-LR in some cases and vice-versa. In fact, the gray-box query attack only achieves a maximum evasion rate of ≈ 20% against MalProtect across the board.

For the other stateful defenses, we observe trends similar to those seen in the results for the black-box query attack. For AndroZoo, SD outperforms L_0 and PRADA, with average evasion rates of 57.5%, 46.9%, and 37.5% against L_0, PRADA, and SD, respectively. For SLEIPNIR, PRADA outperforms L_0 and SD for the majority of prediction models, a trend that was also observed in the black-box query attack results. Here, the average evasion rates of the attack sit at 43.8%, 19.3%, and 50.3% for L_0, PRADA, and SD, respectively. Although PRADA offers some robustness, it underperforms when other metrics are also considered (as we show later in Section 9). For DREBIN, the prior stateful defenses perform

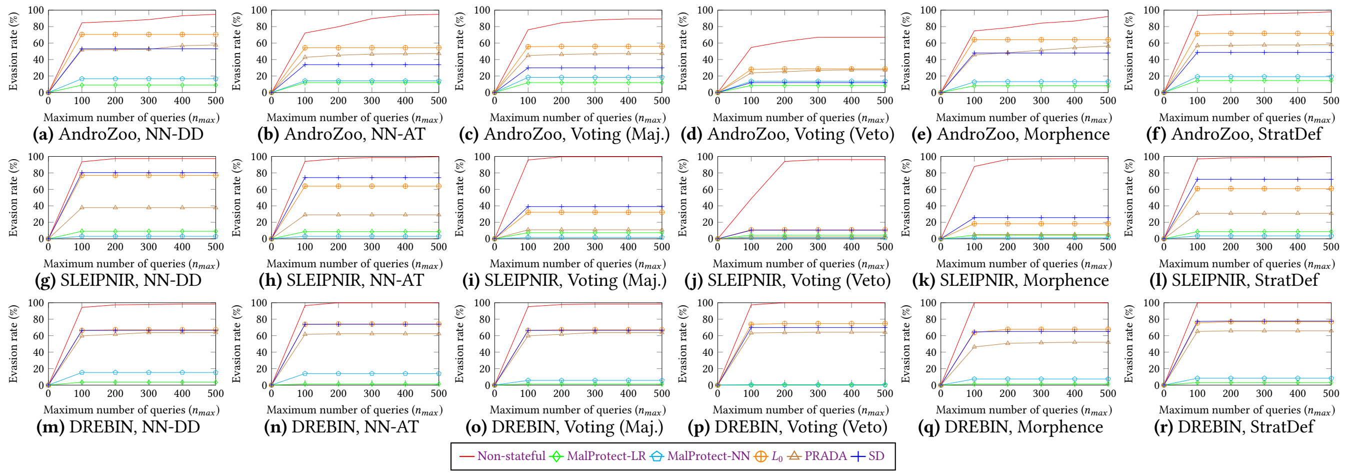


Figure 4. Evasion rate vs. n_{max} of gray-box query attack against non-stateful defenses (prediction models) and each combination of prediction model and stateful defense.

similarly with more balanced performance. That is, they perform equally poorly, with the gray-box query attack able to achieve average evasion rates of 71.3%, 62%, and 69.9% for L_0, PRADA, and SD, respectively, across all configurations. Interestingly, once again, PRADA (minimally) surpasses L_0 and SD in terms of robustness, though the poor general performance of these prior stateful defenses seen for both black-box and gray-box results reaffirms that relying on a single similarity or OOD detection mechanism is inadequate for this domain. In the malware detection domain, attackers use different techniques to generate adversarial examples that often include replacing features rather than making small perturbations to them that can be detected.

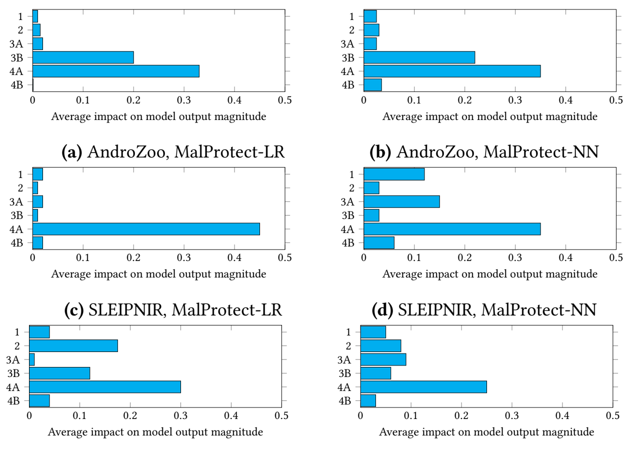
Further evaluating the evasion rate versus n_{max}, 100 or fewer queries are enough to achieve attack success in most cases. Query attacks in domains using continuous features (e.g., image recognition) may require substantially more queries [22] to achieve attack success compared with domains that use discrete features. This is because perturbations in a discrete feature-space (e.g., 0 to 1) have a greater effect on the final prediction, requiring fewer of them to accomplish evasion than perturbations made per query in a continuous feature-space (e.g., +0.01).

8 Interpretability & Adaptive Attack

Interpretability. As ML is being applied more to the cybersecurity domain, a key challenge is the interpretability of predictions made by models [68]. As an initial step towards addressing this in stateful defenses, MalProtect produces interpretable decisions. By observing the scores produced by each threat indicator, analysts can better understand how MalProtect made a particular decision, as higher scores for an indicator give clues about what is anomalous about queries. In addition, the *influence* of each threat indicator on MalProtect's decision can be examined by analyzing the *global feature importance* in MalProtect's decision model. Recall that each threat indicator is a feature of the decision model. Therefore, we can assess the global feature importance using SHAP [62], which is a widely-used framework for interpreting ML predictions. Each feature of the decision model is given an importance value by SHAP at a global level, which produces the data observable in Figure 5.

Figure 5 shows that the indicator 4A has the greatest influence on MalProtect's predicted output across all decision models and datasets. This indicates that a significantly high autoencoder loss is more likely to affect the predicted output. The importance of the other indicators is more balanced. The indicators based on similarity detection seem less influential, which is interesting because other stateful defenses rely solely on similarity detection to detect attacks. In malware detection, not only do our results show that attackers can evade stateful defenses relying solely on such schemes, but we further show that MalProtect rightly considers those indicators as less important.

However, a capable attacker could leverage this information to evade MalProtect. In an adaptive attack (i.e., white-box attack) scenario [19, 97], the attacker has complete knowledge of MalProtect and, thus, knowledge of how much each indicator contributes to the final decision. With this information, an attacker could craft an adversarial example that evades both MalProtect and the underlying prediction model. We next examine how such an attack could be performed.



(a) AndroZoo, MalProtect-LR **(b)** AndroZoo, MalProtect-NN

(c) SLEIPNIR, MalProtect-LR **(d)** SLEIPNIR, MalProtect-NN

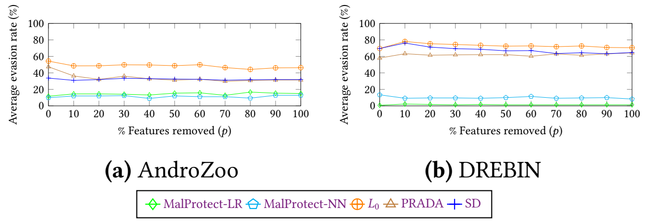
(e) DREBIN, MalProtect-LR **(f)** DREBIN, MalProtect-NN

Figure 5. Influence of MalProtect threat indicators.

Attack Rationale. In the adaptive attack, the attacker knows precisely how MalProtect operates, its internal workings, and other pertinent information related to the threat indicators. At a high level, to evade MalProtect's detection system, adversarial queries must appear as legitimate as possible and convince the indicators that the queries are distinct yet *normal*. Additionally, adversarial queries must remain within the distribution of legitimate queries. For example, we know that the indicator 4A is the most influential in the final prediction, and therefore it would be essential to ensure that the autoencoder loss is not abnormally high. In essence, queries must remain as far apart from each other, while retaining their original malicious functionality and appearing legitimate. Importantly, of course, any generated adversarial example must also evade the underlying prediction model; otherwise, such an attack would be of no use. The attacker must bypass all components of the target model to be successful [6].

Adaptive Attack Strategy. We modify the gray-box attack strategy to produce the adaptive attack strategy (see Appendix F). Firstly, we limit the number of features that can be perturbed in a single iteration of the attack so as not to exhaust possible perturbations early on. This frees up perturbations to be used in later iterations of the attack if earlier queries cannot achieve evasion. While this may increase the query distance and make queries appear less anomalous, other indicators (e.g., indicator 4A) may still be able to trigger attack detection. To deal with this, the adaptive attack strategy *removes* a proportion (p) of features at each iteration. This means that queries will be more distant, with fewer shared and enabled features, while conforming to the distribution of legitimate queries and the training data. For example, a *combination of features* that may cause an increase in the autoencoder loss (such that it appears anomalous) may be removed. Only the AndroZoo and DREBIN datasets support the removal of features while preserving functionality within the feature-space. Therefore, the adaptive attack strategy can only be applied to these datasets.

Results. Figure 6 shows the evasion rate of the adaptive attack against the stateful defenses (averaged across the maximum number of queries permitted, up to 500) versus the percentage of features removed (p). Successful adversarial examples evade each stateful defense as well as the underlying prediction model, which is NN-AT.



(a) AndroZoo **(b)** DREBIN

Figure 6. Average evasion rate vs. p for adaptive attack. The average is across the maximum number of queries permitted.

For both datasets, the adaptive attack fails to achieve significant increases in the evasion rate against either MalProtect configuration. In fact, in some cases, the average evasion rate decreases as p increases. This is likely because the (benign) features selected to cross the decision boundary are in fact removed. The adaptive attack causes other stateful defenses to exhibit similar performance to the gray-box query attack, with an average evasion rate of 72.9%, 61.9%, and 67.7% for L_0, PRADA, and SD, respectively, with a consistent evasion rate across different values of p. This is expected, as the adaptive attack is specifically designed to target MalProtect and not the other stateful defenses.

9 Beyond Adversarial Robustness

We also evaluate the stateful defenses with metrics beyond the evasion rate. This is imperative, as the original task of classifying benign and malicious queries must be done properly too. For example, accuracy is important across all domains. Additionally, particularly in malware detection, the FPR must remain low [45, 59, 93, 105] lest a system be deployed that frustratingly flags legitimate queries as malicious, affecting the quality of service.

Procedure. We evaluate the stateful defenses under different system conditions by querying them with varying degrees of adversity. For this, we generate adversarial examples through a transferability attack using substitute models trained on the same training data as the target models. We construct four vanilla models (see Appendix D for architectures) and apply a range of attacks against these models (see Section 5) to generate 1000+ adversarial examples with the aim of having them transfer to the target models. Each stateful defense is then evaluated using NN-AT as the prediction model. For this, each stateful defense is queried 1000+ times with values of $0.1 \leq k \leq 0.9$, with k representing the adversarial intensity. For example, $k = 0.1$ means that 10% of queries are adversarial examples, while the remainder is an equal number of benign and non-adversarial malware samples from our test set. We avoid the instances where there are no adversarial queries ($k = 0$) and when all queries are adversarial ($k = 1$) as it is unlikely that a defense is deployed in such environments. Where necessary, the query history is initialized with randomly selected samples from the training data.

Results. Figure 7 shows the performance of each stateful defense across different metrics versus k. For AndroZoo and SLEIPNIR, most stateful defenses remain largely unaffected as k increases. In particular, both MalProtect configurations exhibit stable performance across both of these datasets across all metrics. Meanwhile, other stateful defenses such as L_0, PRADA, and SD offer relatively consistent accuracy for these datasets as k increases. This is likely because the adversarial queries are less effective as they are generated through transferability attacks. Recall that these attacks are designed for other domains and that perturbations may be reversed if features are modified in a manner that affects functionality, thereby limiting attack success. For DREBIN, we observe the general decline of the non-stateful NN-AT model as well as all stateful defenses besides MalProtect, whose accuracy does not dip as the adversarial intensity increases. L_0, PRADA, and SD perform worse, only exhibiting their peak accuracy at the lowest value of k. These results show that MalProtect exhibits robustness towards transferability attacks, as it can offer 98% accuracy as k reaches 0.9, where the majority of queries are adversarial examples.

MalProtect's strong performance is also reflected in the F1 and AUC metrics. As k increases, its performance tends to remain stable across the datasets similar to the accuracy. Conversely, other stateful defenses, such as PRADA and SD, suffer a decline in their performance with similar trends to the accuracy visible across the datasets.

Finally, and in addition to the other metrics, in the malware detection domain, a low false positive rate (FPR) is essential to ensure a reliable service for users [25, 45, 59, 93, 105]. As we expect and can confirm, stateful defenses exhibit a higher FPR than non-stateful models due to the sensitivity of their detection mechanisms, which leads them to incorrectly flag some legitimate queries as adversarial. MalProtect generally offers a low FPR, as it uses several indicators to predict an attack. Across the datasets, L_0 is the only defense that offers comparable performance to MalProtect, likely as it does not have an overly-sensitive detection mechanism (though as we have seen, it does not work well at defending against query attacks in this domain). For AndroZoo, the FPR of both MalProtect configurations is comparable to the non-stateful NN-AT model, with only 9% for both MalProtect configurations (versus 5% for the non-stateful defense).

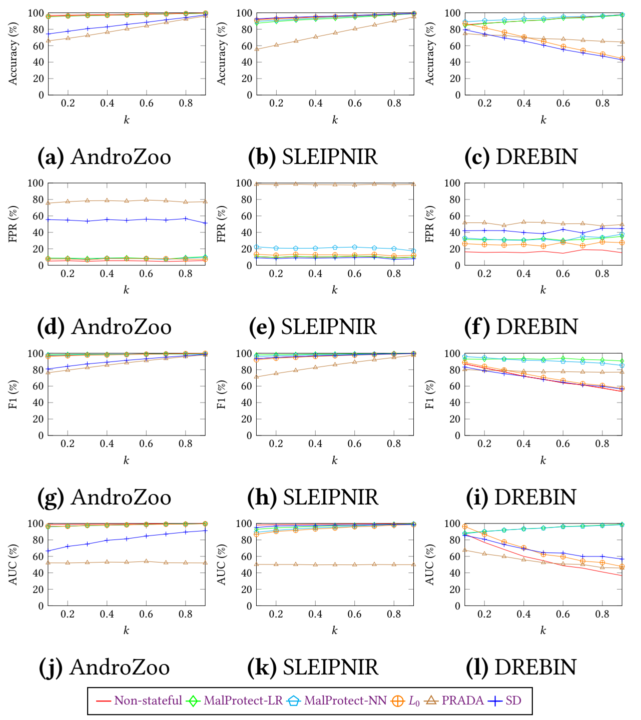


Figure 7. Accuracy, FPR, F1 & AUC vs. k for each stateful defense.

Regarding other stateful defenses, PRADA exhibits a significantly high FPR for SLEIPNIR, similar to AndroZoo. This is closely resembled by SD, with its high FPR for the other two datasets. For DREBIN, PRADA and SD exhibit an average FPR of 40+% across all values of k — this is a trend similar to the AndroZoo dataset. Considering different standard ML metrics, MalProtect clearly offers more reliable performance under an altogether different attack scenario.

10 Overheads of MalProtect

We have shown MalProtect's ability to defend prediction models against adversarial query attacks. However, assessing the storage and time costs of deploying MalProtect is also crucial. MalProtect's detection capacity and performance are proportional to the defender's available resources; with additional resources, MalProtect can store and analyze more queries in Q rapidly and precisely to detect attacks.

Storage. The storage cost scales linearly with the size of Q. In our experiments, the size of Q is capped at 10,000, which only consumes ≈ 0.08MB storage for all the datasets. MalProtect's storage costs are therefore negligible considering modern resources.

Prediction Time. There is latency associated with analyzing the query history and returning the prediction to the user. For MalProtect, the worst-case prediction time — which is when it must compare a query with all of Q — scales linearly with the size of the query history. In our experiments where $|Q|$ is capped at 10,000, the worst-case prediction time sits at ≈ 0.4 seconds for AndroZoo and SLEIPNIR, and ≈ 0.6 seconds for DREBIN. For MalProtect-LR and MalProtect-NN, the analysis stage is identical, while obtaining the prediction from the decision model is near-instant, which produces the same overall cost. While other stateful defenses such as L_0, PRADA, and SD sometimes have lower worst-case prediction times (≈ 0.03-0.6 seconds across the datasets), MalProtect takes more time to perform an in-depth analysis of the query history, which provides more protection against attacks. As we have shown, other stateful defenses are more susceptible to attacks when applied to this domain.

Optimization. MalProtect can be optimized with GPUs. We find that running MalProtect on an NVIDIA A100 GPU reduces the worst-case prediction time to ≈ 0.21-0.41 seconds across the datasets when $|Q|$ is capped at 10,000 queries.

11 Conclusion

In this paper, we presented MalProtect, which is the first stateful defense for adversarial query attacks in the ML-based malware detection domain. As we have shown, ML prediction models and defenses exhibit significant vulnerability to query attacks in this domain. Prior stateful defenses that have been applied to other domains provide little protection here either. Meanwhile, our defense, MalProtect, does not rely solely on similarity or out-of-distribution detection, as these prior defenses do. Instead, several threat indicators and a decision model are used to detect attacks more effectively. Our evaluation has shown that MalProtect performs well against attacks under various scenarios and also offers more reliable predictions for non-adversarial queries than prior stateful defenses. Furthermore, MalProtect displays resilience even against adaptive attackers.

As future work, we aim to explore how additional indicators could be added to increase protection, such as cyber-threat intelligence [90, 109]. Moreover, there is an open research direction regarding *concept drift*. This relates to the constant evolution of malware, which makes it difficult to detect unseen behavior [18, 39, 50], leading to unsustainable models. Prior work has suggested retraining a model regularly [49, 64]. In the case of MalProtect, the defender may need to regularly evaluate the indicators. One potential way to do this would be to couple MalProtect with a detection framework for detecting when such modifications are necessary [50].

References

[1] Zainab Abaid, Mohamed Ali Kaafar, and Sanjay Jha. 2017. Quantifying the impact of adversarial evasion attacks on machine learning based android malware classifiers. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*. 1–10. https://doi.org/10.1109/NCA.2017.8171381

[2] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, and Una-May O'Reilly. 2018. Adversarial deep learning for robust detection of binary encoded malware. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 76–82.

[3] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International*

Conference on Mining Software Repositories (Austin, Texas) *(MSR '16)*. ACM, New York, NY, USA, 468–471. https://doi.org/10.1145/2901739.2903508

[4] Inc. Amazon Web Services. [n. d.]. *Machine Learning on AWS*. https://aws.amazon.com/machine-learning/

[5] Abderrahmen Amich and Birhanu Eshete. 2021. Morphence: Moving Target Defense Against Adversarial Examples. In *Annual Computer Security Applications Conference* (Virtual Event, USA) *(ACSAC)*. Association for Computing Machinery, New York, NY, USA, 61–75. https://doi.org/10.1145/3485832.3485899

[6] Giovanni Apruzzese, Hyrum S. Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin A. Roundy. 2023. "Real Attackers Don't Compute Gradients": Bridging the Gap between Adversarial ML Research and Practice. In *Proceedings of the 1st IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*.

[7] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and don'ts of machine learning in computer security. In *Proc. of the USENIX Security Symposium*.

[8] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. 2014. Drebin: Effective and explainable detection of android malware in your pocket.. In *Ndss*, Vol. 14. 23–26.

[9] Omer Aslan Aslan and Refik Samet. 2020. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* 8 (2020), 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

[10] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*. PMLR, 274–283.

[11] Buse Gul Atli, Sebastian Szyller, Mika Juuti, Samuel Marchal, and N Asokan. 2020. Extraction of complex dnn models: Real threat or boogeyman?. In *International Workshop on Engineering Dependable and Secure Machine Learning Systems*. Springer, 42–57.

[12] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. 2022. Transcending transcend: Revisiting malware classification in the presence of concept drift. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 805–823.

[13] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. 2018. Practical Black-box Attacks on Deep Neural Networks using Efficient Query Mechanisms. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[14] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 387–402.

[15] R. Braden, D. Clark, S. Crocker, and C. Huitema. 1994. RFC1636: Report of IAB Workshop on Security in the Internet Architecture - February 8-10, 1994.

[16] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2017. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248* (2017).

[17] Miles Brundage, Shahar Avin, Jack Clark, Helen Toner, Peter Eckersley, Ben Garfinkel, Allan Dafoe, Paul Scharre, Thomas Zeitzoff, Bobby Filar, et al. 2018. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228* (2018).

[18] Haipeng Cai and John Jenkins. 2018. Towards Sustainable Android Malware Detection. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 350–351. https://doi.org/10.1145/3183440.3195004

[19] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705* (2019).

[20] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. IEEE, 39–57.

[21] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069* (2018).

[22] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. 2020. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1277–1294.

[23] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*. 15–26.

[24] Steven Chen, Nicholas Carlini, and David Wagner. 2020. Stateful detection of black-box adversarial attacks. In *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*. 30–39.

[25] Yizheng Chen, Shiqi Wang, Dongdong She, and Suman Jana. 2020. On training robust {PDF} malware classifiers. In *29th USENIX Security Symposium (USENIX Security 20)*. 2343–2360.

[26] Minhao Cheng, Cho-Jui Hsieh, Inderjit Dhillon, et al. 2020. Voting based ensemble improves robustness of defensive models. *arXiv preprint arXiv:2011.14031* (2020).

[27] Clarence Chio and David Freeman. 2018. *Machine learning and security: Protecting systems with data and algorithms.* " O'Reilly Media, Inc.".

[28] Jin-Hee Cho, Dilli P Sharma, Hooman Alavizadeh, Seunghyun Yoon, Noam Ben-Asher, Terrence J Moore, Dong Seong Kim, Hyuk Lim, and Frederica F Nelson. 2020. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials* 22, 1 (2020), 709–745.

[29] Google Cloud. [n. d.]. *Google Cloud*. https://cloud.google.com/

[30] Prithviraj Dasgupta and Joseph Collins. 2019. A survey of game theoretic approaches for adversarial machine learning in cybersecurity tasks. *AI Magazine* 40, 2 (2019), 31–43.

[31] Dauodi et al. 2022. A Deep Dive Inside DREBIN: An Explorative Analysis beyond Android Malware Detection Scores. *ACM Trans. Priv. Secur.* 25, 2, Article 13 (may 2022), 28 pages. https://doi.org/10.1145/3503463

[32] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. 2017. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Transactions on Dependable and Secure Computing* (2017).

[33] Jasjeet Dhaliwal and Saurabh Shintre. 2018. Gradient similarity: An explainable approach to detect adversarial attacks against deep learning. *arXiv preprint arXiv:1806.10707* (2018).

[34] John R. Douceur. 2002. The Sybil Attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS '01)*. Springer-Verlag, Berlin, Heidelberg, 251–260.

[35] Stephan Dreiseitl and Lucila Ohno-Machado. 2002. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics* 35, 5-6 (2002), 352–359.

[36] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. 2017. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945* 2, 3 (2017), 4.

[37] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song.

2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[38] Peter C. Fishburn. 1967. Letter to the Editor—Additive Utilities with Incomplete Product Sets: Application to Priorities and Assignments. *Operations Research* 15, 3 (1967), 537–542. https://doi.org/10.1287/opre.15.3.537 arXiv:https://doi.org/10.1287/opre.15.3.537

[39] Xiaoqin Fu and Haipeng Cai. 2019. On the Deterioration of Learning-Based Malware Detectors for Android. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. 272–273. https://doi.org/10.1109/ICSE-Companion.2019.00110

[40] Joseph Gardiner and Shishir Nagaraja. 2016. On the Security of Machine Learning in Malware C&C Detection: A Survey. *ACM Comput. Surv.* 49, 3, Article 59 (dec 2016), 39 pages. https://doi.org/10.1145/3003816

[41] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. 2019. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1705–1714.

[42] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[43] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[44] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. 2016. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435* (2016).

[45] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial examples for malware detection. In *European symposium on research in computer security*. Springer, 62–79.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.

[47] Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. 2021. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916* (2021).

[48] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*. PMLR, 2137–2146.

[49] Imam et al. 2019. A Survey of Attacks Against Twitter Spam Detectors in an Adversarial Environment. *Robotics* 8, 3 (2019). https://doi.org/10.3390/robotics8030050

[50] Jordaney et al. 2017. Transcend: Detecting Concept Drift in Malware Classification Models. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 625–642. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/jordaney

[51] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. 2019. PRADA: protecting against DNN model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 512–527.

[52] Sanjay Kariyappa and Moinuddin K. Qureshi. 2020. Defending Against Model Stealing Attacks With Adaptive Misinformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[53] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. 2016. Adversarial examples in the physical world.

[54] Raphael Labaca-Castro, Luis Muñoz-González, Feargus Pendlebury, Gabi Dreo Rodosek, Fabio Pierazzi, and Lorenzo Cavallaro. 2021. Universal adversarial perturbations for malware. *arXiv preprint arXiv:2102.06747* (2021).

[55] Pavel Laskov et al. 2014. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE symposium on security and privacy*. IEEE, 197–211.

[56] Aoxue Li, Tiange Luo, Tao Xiang, Weiran Huang, and Liwei Wang. 2019. Few-shot learning with global class representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9715–9724.

[57] Deqiang Li, Qianmu Li, Yanfang Ye, and Shouhuai Xu. 2020. Enhancing Deep Neural Networks Against Adversarial Malware Examples. *arXiv preprint arXiv:2004.07919* (2020).

[58] Deqiang Li, Qianmu Li, Yanfang Ye, and Shouhuai Xu. 2021. A framework for enhancing deep neural networks against adversarial malware. *IEEE Transactions on Network Science and Engineering* 8, 1 (2021), 736–750.

[59] Deqiang Li, Qianmu Li, Yanfang (Fanny) Ye, and Shouhuai Xu. 2021. Arms Race in Adversarial Malware Detection: A Survey. *ACM Comput. Surv.* 55, 1, Article 15 (nov 2021), 35 pages. https://doi.org/10.1145/3484491

[60] Huiying Li, Shawn Shan, Emily Wenger, Jiayun Zhang, Haitao Zheng, and Ben Y. Zhao. 2022. Blacklight: Scalable Defense for Neural Networks against Query-Based Black-Box Attacks. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/usenixsecurity22/presentation/li-huiying

[61] Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. 2020. Qeba: Query-efficient boundary-based blackbox attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1221–1230.

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

[63] Jiawei Ma, Hanchen Xie, Guangxing Han, Shih-Fu Chang, Aram Galstyan, and Wael Abd-Almageed. 2021. Partner-Assisted Learning for Few-Shot Image Classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10573–10582.

[64] Federico Maggi, William Robertson, Christopher Kruegel, and Giovanni Vigna. 2009. Protecting a Moving Target: Addressing Web Application Concept Drift. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection* (Saint-Malo, France) *(RAID '09)*. Springer-Verlag, Berlin, Heidelberg, 21–40. https://doi.org/10.1007/978-3-642-04342-0_2

[65] Mohammad Maghsoudimehrabani, Amin Azmoodeh, Ali Dehghantanha, Behrouz Zolfaghari, and Gautam Srivastava. 2022. Proactive Detection of Query-Based Adversarial Scenarios in NLP Systems. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security* (Los Angeles, CA, USA) *(AISec'22)*. Association for Computing Machinery, New York, NY, USA, 103–113. https://doi.org/10.1145/3560830.3563727

[66] Brad Miller, Alex Kantchelian, Michael Carl Tschantz, Sadia Afroz, Rekha Bachwani, Riyaz Faizullabhoy, Ling Huang, Vaishaal Shankar, Tony Wu, George Yiu, et al. 2016. Reviewer integration and performance measurement for malware detection. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 122–141.

[67] Andreas Moser, Christopher Kruegel, and Engin Kirda. 2007. Limits of static analysis for malware detection. In *Twenty-third annual computer security applications conference (ACSAC 2007)*. IEEE, 421–430.

[68] Azqa Nadeem, Daniël Vos, Clinton Cao, Luca Pajola, Simon Dieck, Robert Baumgartner, and Sicco Verwer. 2022. Sok: Explainable machine learning for computer security applications. *arXiv preprint arXiv:2208.10605* (2022).

[69] Soham Pal, Yash Gupta, Aditya Kanade, and Shirish Shevade. 2021. Stateful Detection of Model Extraction Attacks. *arXiv preprint arXiv:2107.05166* (2021).

[70] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. 2021. Deep Learning for Anomaly Detection: A Review. *ACM Comput. Surv.* 54, 2, Article 38 (mar 2021), 38 pages. https://doi.org/10.1145/3439950

[71] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. 2019. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*. PMLR, 4970–4979.

[72] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z.B. Celik, and A. Swami. 2018. Practical black-box attacks against machine learning. *Proceedings of the 2017 ACM on AsiA framework for enhancing deep neural networks against adversaria conference on computer and communications security* (2018), 506–519.

[73] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 372–387.

[74] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. 2018. SoK: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 399–414.

[75] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 582–597.

[76] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *Proceedings of the 28th USENIX Conference on Security Symposium* (Santa Clara, CA, USA) *(SEC'19)*. USENIX Association, USA, 729–746.

[77] Pierazzi et al. 2020. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 1308–1325. https://doi.org/10.1109/SP40000.2020.00073

[78] R. Podschwadt and H. Takabi. 2019. On Effectiveness of Adversarial Examples and Defenses for Malware Classification. *International Conference on Security and Privacy in Communication Systems* (2019), 380–393.

[79] Friedrich Pukelsheim. 1994. The Three Sigma Rule. *The American Statistician* 48, 2 (1994), 88–91. http://www.jstor.org/stable/2684253

[80] Aqib Rashid and Jose Such. 2022. StratDef: a strategic defense against adversarial attacks in malware detection. *arXiv preprint arXiv:2202.07568* (2022).

[81] Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning. (2016).

[82] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2020. Query-efficient black-box attack against sequence-based malware classifiers. In *Annual Computer Security Applications Conference*. 611–626.

[83] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2021. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–36.

[84] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. 2018. Generic black-box end-to-end attack against state of the art API call based malware classifiers. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 490–510.

[85] Christian Rossow, Christian J. Dietrich, Chris Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos, and Maarten van Steen. 2012. Prudent Practices for Designing Malware Experiments: Status Quo and Outlook. In *2012 IEEE Symposium on Security and Privacy*. 65–79. https://doi.org/10.1109/SP.2012.14

[86] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

[87] Sailik Sengupta, Tathagata Chakraborti, and Subbarao Kambhampati. 2018. MTDeep: boosting the security of deep neural nets against adversarial attacks with moving target defense. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence.*

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

[89] Raja Khurram Shahzad and Niklas Lavesson. 2013. Comparative analysis of voting schemes for ensemble-based malware detection. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 4, 1 (2013), 98–117.

[90] Xiaokui Shu, Frederico Araujo, Douglas L Schales, Marc Ph Stoecklin, Jiyong Jang, Heqing Huang, and Josyula R Rao. 2018. Threat intelligence computing. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1883–1898.

[91] Hispasec Sistemas. [n. d.]. *VirusTotal*. https://www.virustotal.com/

[92] Qun Song, Zhenyu Yan, and Rui Tan. 2019. Moving target defense for deep visual sensing against adversarial examples. *arXiv preprint arXiv:1905.13148* (2019).

[93] Jack W Stokes, De Wang, Mady Marinescu, Marc Marino, and Brian Bussone. 2017. Attack and defense of dynamic analysis-based, adversarial neural malware classification models. *arXiv preprint arXiv:1712.05919* (2017).

[94] Octavian Suciu, Scott E Coull, and Jeffrey Johns. 2019. Exploring adversarial examples in malware detection. In *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 8–14.

[95] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2014).

[96] Romain Thomas. 2017. LIEF - Library to Instrument Executable Formats. https://lief.quarkslab.com/.

[97] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems* 33 (2020), 1633–1645.

[98] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204* (2017).

[99] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, Xinyu Xing, C Lee Giles, and Xue Liu. 2016. Random feature nullification for adversary resistant deep architecture. *arXiv preprint arXiv:1610.01239* (2016).

[100] Weiwei Wang, Xinli Xiong, Songhe Wang, and Jingye Zhang. 2020. MTDNNF: Building the Security Framework for Deep Neural Network by Moving Target Defense. In *2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence*. 1–1.

[101] Yizhen Wang, Mohannad Alhanahnah, Xiaozhu Meng, Ke Wang, Mihai Christodorescu, and Somesh Jha. [n. d.]. Robust Learning against Relational Adversaries. In *Advances in Neural Information Processing Systems*.

[102] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2017. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991* (2017).

[103] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).

[104] Ziang Yan, Yiwen Guo, Jian Liang, and Changshui Zhang. 2021. Policy-Driven Attack: Learning to Query for Hard-label Black-box Adversarial Examples. In *International Conference on Learning Representations*. https://openreview.net/forum?id=pzpytjk3Xb2

[105] Wei Yang, Deguang Kong, Tao Xie, and Carl A Gunter. 2017. Malware detection in adversarial settings: Exploiting feature evolutions

and confusions in android apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. 288–302.

[106] Suleiman Y Yerima and Sakir Sezer. 2018. Droidfusion: A novel multilevel classifier fusion approach for android malware detection. *IEEE transactions on cybernetics* 49, 2 (2018), 453–466.

[107] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. 2020. CloudLeak: Large-Scale Deep Learning Models Stealing Through Adversarial Examples.. In *NDSS*.

[108] Zhanyuan Zhang, Yizheng Chen, and David Wagner. 2021. SEAT: Similarity Encoder by Adversarial Training for Detecting Model Extraction Attack Queries. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security* (Virtual Event, Republic of Korea) *(AISec '21)*. Association for Computing Machinery, New York, NY, USA, 37–48. https://doi.org/10.1145/3474369.3486863

[109] Ziyun Zhu and Tudor Dumitras. 2018. Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 458–472.

A MalProtect

Model	Parameters
Logistic Regression (MalProtect-LR)	No configurable parameters.
Neural Network (MalProtect-NN)	4 fully-connected layers (128 (Relu), 64 (Relu), 32 (Relu), 2 (Softmax))

Table 1. Decision models for MalProtect configurations.

B Other Stateful Defenses

L_0 defense. The L_0 defense measures the similarity between feature vectors using the L_0 distance. This is akin to similarity detection schemes that use L_p norms in other domains (e.g., [60]). As binary feature vectors are used in the malware detection domain, L_0 is the most appropriate measure of distance between two such feature vectors, X and X' [20, 78]. It measures the number of instances such that $X_i \neq X'_i$. For the L_0 defense, an attack is detected if there are queries with an L_0 distance of less than 10 as lower thresholds may easily miss attacks [24, 51, 60].

PRADA [51]. For PRADA, we use $\delta = 0.9$ for the threshold, as per the original paper [51].

Stateful Detection (SD) [24]. For SD, the threshold for detecting an attack is derived by calculating the k-neighbor distance for the 0.1 percentile of the training set. At prediction-time, if the mean distance of the k-nearest queries falls below the calculated threshold, an attack is detected. For this, we use $k = 50$ (as per the original paper). The calculated thresholds for detection are as follows:

AndroZoo	SLEIPNIR	DREBIN
94.64749019607844	44.350758426966294	34.37145577151924

Table 2. Thresholds for detection for SD as per calculations based on original paper.

C Prediction Models

We use six non-stateful defenses as the prediction models. These are evaluated without any stateful protection and then evaluated in combination with each stateful defense.

Defense	Configuration/parameters/setup
Defensive distillation [75]	Neural Network (128 (Relu), 64 (Relu), 32 (Relu), 2 (Softmax) with defensive distillation applied.
Ensemble adversarial training [80, 98] (NN-AT)	Neural Network (128 (Relu), 64 (Relu), 32 (Relu), 2 (Softmax)). Adversarially-trained up to 25% size of training data.
Morphence [5]	$n = 4$, $p = 3$, $Q_{max} = 1000$.
StratDef [80]	Variety-GT using same models as original paper. Assumed strong attacker and $\alpha = 1$.
Voting (Majority & Veto)	Using same models as StratDef.

Table 3. Architectures of prediction models.

D Vanilla Models

The following vanilla models are used in some instances (e.g., to generate adversarial examples, see Section 5).

Model	Parameters
Decision Tree	max_depth=5, min_samples_leaf=1
Neural Network	4 fully-connected layers (128 (Relu), 64 (Relu), 32 (Relu), 2 (Softmax))
Random Forest	max_depth=100
Support Vector Machine	LinearSVC with probability enabled

Table 4. Architectures of vanilla models.

E Permitted Perturbations for AndroZoo and DREBIN

The AndroZoo [3] and DREBIN [8] datasets are based on the Android platform. Both datasets can be divided into eight feature families comprised of extracted static features such as permissions, API calls, hardware requests, and URL requests. Based on their family, features may be addable or removable during attacks to traverse the decision boundary, according to industry documentation and prior work (e.g., [1, 2, 54, 57, 58, 77, 80]). However, it is imperative to preserve malicious functionality in the feature-space as a core constraint in this domain. For example, attacks cannot remove features from the manifest file nor intent filter, and component names must be consistently named. Therefore, Table 5 summarizes the permitted perturbations for each feature family for these datasets. This is used to determine whether the perturbations performed by an attack are valid. For example, if a feature belonging to the S1 family is removed by an attack, then its original value is restored as it is not permitted to be removed (see Section 5).

	Feature families	Addition	Removal
manifest	S1 Hardware	✓	✗
	S2 Requested permissions	✓	✗
	S3 Application components	✓	✓
	S4 Intents	✓	✗
dexcode	S5 Restricted API Calls	✓	✓
	S6 Used permission	✗	✗
	S7 Suspicious API calls	✓	✓
	S8 Network addresses	✓	✓

Table 5. Permitted perturbations for Android datasets. These are determined by consulting industry documentation and prior work [1, 2, 54, 57, 58, 77, 80].

F Query Attack Strategies

We modify attack strategies from prior work on adversarial attacks in malware detection [82]. These are based on a software transplantation approach where an attacker makes perturbations based on benign samples. We modify prior attack strategies by transplanting multiple features per iteration (rather than one perturbation per iteration). The differences between the black-box and gray-box strategies lie in how perturbations are applied. The black-box attack selects the features to perturb in a randomized manner, while the gray-box attack perturbs features based on their frequency in benign samples in a heuristically-driven approach.

Algorithm 1 Black-box query attack: for oracle O, malware sample X, set of (randomly-ordered) benign features F, maximum permitted queries n_{max}.

```
Input: $O$, $X$, $F$, $n_{max}$
1:  $X' \leftarrow X, n \leftarrow 0$
2:  while $O(X') = 1$ & $n < n_{max}$ & $n < F.length$ do
3:      $X' \leftarrow AddFeature(X', F[n])$
4:      $r \leftarrow RandomInteger(0, Length(F))$
5:      $F_r \leftarrow ChooseRandomFeatures(r, F)$
6:      $X' \leftarrow AddFeatures(X', F_r)$
7:      $X' \leftarrow ValidatePerturbations(X, X')$
8:      $n \leftarrow n + 1$
9:      if $O(X') = 0$ then return Success
10: return Failure
```

For the adaptive attack against MalProtect, the number of features that is added in a single perturbation is capped, and from the *removable features* of the query, $p\%$ of features are removed. This is to make queries as distinct from each other as possible while remaining within the general distribution of other queries.

Algorithm 2 Gray-box query attack: for oracle O, malware sample X, vector of sorted benign features $\vec{s}$, maximum permitted queries n_{max}.

```
Input: $O$, $X$, $\vec{s}$, $n_{max}$
1:  $X' \leftarrow X, n \leftarrow 0$
2:  while $O(X') = 1$ & $n < n_{max}$ & $n < \vec{s}.length$ do
3:      $X' \leftarrow AddFeature(X', \vec{s}[n])$
4:      $r \leftarrow RandomInteger(0, Length(\vec{s}))$
5:      $F_r \leftarrow ChooseRandomFeatures(r, \vec{s})$
6:      $X' \leftarrow AddFeatures(X', F_r)$
7:      $X' \leftarrow ValidatePerturbations(X, X')$
8:      $n \leftarrow n + 1$
9:      if $O(X') = 0$ then return Success
10: return Failure
```

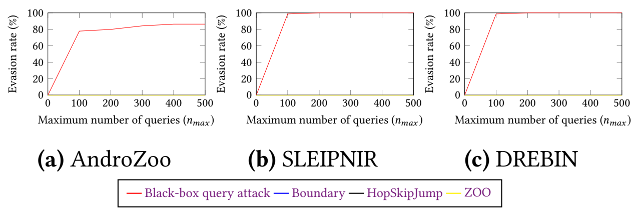
Algorithm 3 Adaptive attack: for oracle O, malware sample X, vector of sorted benign features $\vec{s}$, maximum permitted queries n_{max}, p percentage of features to remove, m maximum features to add in each iteration.

```
Input: $O$, $X$, $\vec{s}$, $n_{max}$, $m$
1:  $X' \leftarrow X, n \leftarrow 0$
2:  while $O(X') = 1$ & $n < n_{max}$ & $n < \vec{s}.length$ do
3:      $X' \leftarrow AddFeature(X', \vec{s}[n])$
4:      $r \leftarrow RandomInteger(0, m)$
5:      $F_r \leftarrow ChooseRandomFeatures(r, \vec{s})$
6:      $X' \leftarrow AddFeatures(X', F_r)$
7:      $X' \leftarrow RemoveFeatures(X', p)$
8:      $X' \leftarrow ValidatePerturbations(X, X')$
9:      $n \leftarrow n + 1$
10:     if $O(X') = 0$ then return Success
11: return Failure
```

G Evaluating Other Query Attacks

We also test the Boundary [16], HopSkipJump [22], and ZOO [23] query attacks. We demonstrate their inability to generate adversarial examples in this domain, as these attacks are designed for other domains and do not consider the constraints of the malware detection domain (functionality preservation and discretization of features). We use each attack to generate adversarial examples against the NN-AT model. For this, once the feature vector of a malware sample has been perturbed, we discretize the feature vectors and evaluate whether the perturbations are permitted for each dataset. Any invalid perturbations are reversed.

Figure 8 shows that these attacks are unable to achieve evasion *at all*. It is clear that the perturbations used to cross the decision boundary are reversed. That is, invalid perturbations are consistently made that must be discretized and reversed to ensure functionality preservation. In this same figure, we also show that our black-box query attack strategy achieves 80 + % evasion rate across the datasets.



(a) AndroZoo **(b)** SLEIPNIR **(c)** DREBIN

Figure 8. Evasion rate of additional query attacks against NN-AT model vs. n_{max}.