Certified Robustness of Learning-based Static Malware Detectors

Anonymous Author(s)

ABSTRACT

Certified defenses are a recent development in adversarial machine learning (ML), which aim to rigorously guarantee the robustness of ML models to adversarial perturbations. A large body of work studies certified defenses in computer vision, where ℓ_p norm-bounded evasion attacks are adopted as a tractable threat model. However, this threat model has known limitations in vision, and is not applicable to other domains—e.g., where inputs may be discrete or subject to complex constraints. Motivated by this gap, we study certified defenses for malware detection, a domain where attacks against ML-based systems are a real and current threat. We consider static malware detection systems that operate on byte-level data. Our certified defense is based on the approach of *randomized smoothing* which we adapt by: (1) replacing the standard Gaussian randomization scheme with a novel deletion randomization scheme that operates on bytes or chunks of an executable; and (2) deriving a certificate that measures robustness to evasion attacks in terms of generalized edit distance. To assess the size of robustness certificates that are achievable while maintaining high accuracy, we conduct experiments on malware datasets using a popular convolutional malware detection model, MalConv. We are able to accurately classify 91% of the inputs while being certifiably robust to *any* adversarial perturbations of edit distance 128 bytes or less. By comparison, an existing certification of up to 128 bytes of substitutions (without insertions or deletions) achieves an accuracy of 78%. In addition, given that robustness certificates are conservative, we evaluate practical robustness to several recently published evasion attacks and, in some cases, find robustness beyond certified guarantees.

CCS CONCEPTS

• **Security and privacy → Logic and verification**; *Malware and its mitigation*; • **Computing methodologies** → *Machine learning*.

KEYWORDS

certified robustness, malware detection, adversarial machine learning

ACM Reference Format:
Anonymous Author(s). 2023. Certified Robustness of Learning-based Static Malware Detectors. In *Proceedings of ACM Conference on Computer and Communications Security (CCS '23).* ACM, New York, NY, USA, 19 pages. https://doi.org/XXXXXXX.XXXXXXX

CCS '23, Month 01–05, 2023, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Machine learning (ML) is impacting many areas of computing thanks to its ability to generalize to complex and unseen data. However, vulnerability of ML models to evasion attacks (a.k.a. adversarial examples) raises concerns about using these models in practice. For example, successful attacks have been demonstrated in general settings [30, 35] and domains such as computer vision [26, 33, 74], natural language [3, 29, 67], and malware detection [20, 21, 40, 42, 52, 60, 76, 77]. While a multitude of defenses have been proposed against evasion attacks, they have historically been broken by stronger attacks. For instance, adversarial training with the Fast Gradient Sign Method [30] and defensive distillation [59] are two defenses that were subsequently found to be ineffective [12, 79]. Six of nine defense papers accepted for presentation at ICLR2018 were defeated months before the conference took place [4]; another tranche of thirteen defenses were circumvented shortly later [78]. Motivated by the arms race between attackers and defenders, a line of work called *certified robustness* has emerged, which aims to *guarantee* that a model is immune to a specified set of attacks [66, 89].

Certified robustness has the greatest prominence in computer vision. The state-of-the-art for ImageNet correctly classifies 71% of a test set, while guaranteeing that the classifications are invariant under ℓ_2-norm bounded attacks of size 127/255 (half maximum pixel intensity) [11]. In computer vision, ℓ_p-norm bounded perturbations are commonly considered as a tractable approximation for visual imperceptibility, despite known limitations. User studies have shown that perturbations with small ℓ_p-norm can be reliably detected through casual inspection, while imperceptible changes can cover large ℓ_p distances [73]. For example, robust defenses can be circumvented by image translation, rotation, blur, and pixelation [28, 48]. Moreover, little is known about certified robustness beyond the ℓ_p threat model, in part because it has had little examination outside computer vision, with few exceptions [37, 58, 67, 92].

To address this gap in certified robustness research, we focus on the static malware detection domain, where evasion attacks are well established. Detecting malicious software (malware) is critical in system security and has advanced considerably over the past couple of decades to keep pace with novel threats, including evasive malware variants and zero-day exploits. ML is starting to play an important role in this advancement. It is now deployed in many commercial systems [7, 39, 54, 80] and remains an active area of research [1, 49, 65, 82]. Despite the apparent advantage of ML in generalizing to novel malware, recent research has shown that ML-based static malware detectors can be evaded by applying adversarial perturbations to malware [20–22, 40, 42, 52, 60, 76, 77]. A variety of perturbations have been considered with different effects at the semantic level, however all of them can be modeled as inserting, deleting and/or substituting bytes. Certifying static ML-based malware detectors within this general threat model—where an attacker can perform byte-level edits—requires advancing certified robustness research. While commercial malware detectors use both static and dynamic analysis [5, 7, 8, 80], published evasion attacks are less developed for this hybrid setting. Overcoming static malware detectors is a realistic goal for adversaries, as they may be used to protect end-user systems [16, 83] and obtaining white-box access may be trivial (e.g., by purchasing a license).

In this paper, we seek to answer three research questions at the nexus of certified robustness and malware detection.

(Q1) *How can certified robustness methods be adapted to the malware detection domain?*

Existing certified robustness methods are designed for models that operate on fixed-dimensional numeric arrays, under the assumption that an attacker can only make perturbations with small, bounded ℓ_p-norm. While these assumptions are relatively accepted for the vision domain, they are fundamentally incompatible with malware detection, where inputs are variable-length byte arrays—the most general representation of an executable. To this end, we consider an attacker that perturbs a file by inserting, deleting or substituting bytes, in place of additive perturbations. We describe this new problem setting for certified robustness in Section 2, before proposing a novel certification mechanism in Section 3. Our certification mechanism, called randomized deletion smoothing (RS-Del), adapts randomized smoothing [17] by replacing Gaussian input randomization with randomized deletions, and may be of independent interest. We customize our mechanism for several threat model variations, and make practical optimizations for computational efficiency.[1]

While certified robustness provides a theoretical framework for measuring the robustness of a model to attacks, it does not provide any guarantees about accuracy. We therefore ask,

(Q2) *What kind of tradeoffs are possible between accuracy and robustness guarantees for malware detection?*

To answer this question, we evaluate our randomized deletion smoothing mechanism on two malware detection datasets using a deep malware detection model [64] in Section 4.2. By varying the aggressiveness of smoothing we examine tradeoffs between robustness certification and accuracy. We find that it is possible to maintain a high accuracy of 91% while guaranteeing robustness to adversarial edits of up to 128 bytes on average, which exceeds edit distances of two published evasion attacks [20, 22]. This suggests potential for operationalizing certifications of static malware detection, in some cases.

It is well-known that certified robustness guarantees are conservative due to model independence or relying on bounds that are not tight in general [25]. Consequently, we ask,

(Q3) *How well does our randomized deletion smoothing mechanism protect against evasion attacks in practice?*

To answer this question, we apply five published evasion attacks [20, 21, 42, 52, 57] against an undefended model and a model employing our randomized deletion smoothing (RS-Del). We find that RS-Del is surprisingly helpful at delivering additional robustness beyond what is guaranteed. That is, even though the distance of attack perturbations is beyond the certified radius of RS-Del, it is still effective at distinguishing malware and benign samples. For example, the attack success rate against RS-Del is 0% compared to 12.9% for an unprotected classifier when up to 17.2KB of a file is perturbed with a displacement attack [52]. This is orders of magnitude larger than the edit distance radius of certification returned by RS-Del. Full attack experimental results are presented in Section 4.3, where we highlight cases where RS-Del's defense capabilities are effective and where further protection is necessary.

[1] We will release an open-source implementation of our mechanisms upon publication.

2 PROBLEM FORMULATION

In this section, we provide background on static malware detection, specify a threat model for evasion attacks on static malware detectors, and introduce certified robustness in the context of static malware detection, where inputs are represented as raw byte arrays.

2.1 Static malware detection

We model a malware detector as a function $f : \mathcal{X} \to \{0, 1\}$ that returns 1 if the input executable file $\mathbf{x} \in \mathcal{X}$ is predicted to be malicious and 0 otherwise. We assume executable files are represented as byte arrays, where $\mathcal{X} = \{0, \ldots, 255\}^\star$ is the space of byte arrays of arbitrary length. For compatibility with randomized smoothing (discussed in Section 3), we assume f is able to make predictions for incomplete files where chunks of bytes have been arbitrarily removed. This assumption can be satisfied by machine learning-based static malware detectors, as demonstrated in our experiments (Section 4). We note that dynamic malware detectors do not satisfy this assumption, since they monitor behavior during execution, which is not generally possible for an incomplete executable file.

2.2 Threat model

We next outline the modeled attacker's goals, capabilities, and information about the detector [6].

2.2.1 Attacker's objective. We consider evasion attacks, where the attacker's objective is to transform an executable file $\mathbf{x}$ so that it is misclassified by a malware detector f. To ensure the attacked file $\bar{\mathbf{x}}$ is useful after evading detection, we require that it is *functionally-equivalent* to the original file $\mathbf{x}$. We focus on evasion attacks that misclassify *malware* as *benign* in our experiments, as these attacks dominate prior work [22]. However, for generality we also consider attacks in the opposite direction—where a *benign* file is misclassified as *malicious*—when outlining our threat model and deriving robustness certificates.

2.2.2 Attacker's capability. We measure the attacker's capability in terms of the number of elementary edits they can make to the original file $\mathbf{x}$. If the attacker is capable of making up to c elementary edits, then they can transform $\mathbf{x}$ into any file in the edit distance ball of radius c centred on $\mathbf{x}$:

$$\mathcal{A}_c(\mathbf{x}) = \{\bar{\mathbf{x}} \in \mathcal{X} : \mathrm{dist}_O(\mathbf{x}, \bar{\mathbf{x}}) \leq c\}. \quad (1)$$

Here $\mathrm{dist}_O(\mathbf{x}, \bar{\mathbf{x}})$ denotes the edit distance from the original file $\mathbf{x}$ to the attacked file $\bar{\mathbf{x}}$ under the set of edit operations (ops) O. Unless otherwise specified, we assume O consists of byte-level deletions (del), insertions (ins) and substitutions (sub), however our analysis covers attackers that are constrained to a subset of these operations as outlined in Table 1. We also consider attackers than perform instruction-level edits in Section 3.3.3.

We note that edit distance is a reasonable proxy for the cost of running evasion attacks that iteratively apply localized functionality-preserving edits (e.g., [20, 52, 57, 60, 76]). For these attacks, the edit distance scales roughly linearly with the number of attack iterations, and therefore the adversary has an incentive to minimize edit distance. While attacks do exist that make large edits of order megabytes in size (e.g., [21]), we believe that an edit distance-constrained threat model is an important step towards realistic threat models for certified malware detection. (To examine the effect of large edits on robustness we include the GAMMA attack [21] in our Section 4.3 experiments.)

Remark 2.1. The set $\mathcal{A}_c(\mathbf{x})$ *overestimates* the capability of an edit distance-constrained attacker, because it may include files that are not functionally equivalent to $\mathbf{x}$. For example, $\mathcal{A}_c(\mathbf{x})$ may include files that are not malicious (assuming $\mathbf{x}$ is malicious) or files that are not valid executables. This poses no problem for certification, since overestimating an attacker's capability merely leads to a stronger certificate than required. Indeed, overestimating the attacker's capability seem necessary, as functionally equivalent files are difficult to specify, let alone analyze.

2.2.3 Malware detector access. We consider attackers with black- or white-box access to the malware detector. In the black-box setting, the attacker may make an unlimited number of queries to the malware detector without observing its internal operation. We permit access to detection confidence scores, which are returned alongside predictions even in the black-box setting. In the white-box setting, the attacker can additionally inspect the malware detector's source code. Such a strong assumption is needed for white-box attacks that compute loss gradients with respect to the detector's internal representations of the input file [42, 52].

2.3 Certified robustness

To provide assurance that a malware detector is robust to evasion attacks, we adapt the concept of *certified robustness* from the machine learning literature. Most existing definitions of certified robustness aim to guarantee that a classifier's prediction is stable, even if the input is perturbed within an ℓ_p neighborhood [17, 43, 44, 89]. This definition is ill-suited for malware detection because it implicitly assumes inputs are fixed-dimension numeric arrays, and that the array values can be perturbed continuously. To adapt the definition, we replace the ℓ_p neighborhood with an edit distance neighborhood concordant with our threat model. This is formalized below.

Definition 2.2. An *edit distance robustness certificate* of radius r for a malware detector f at input file $\mathbf{x}$ is a guarantee that $f(\mathbf{x}) = f(\mathbf{x}')$ for all $\mathbf{x}'$ in the edit distance neighborhood

$$\mathcal{N}_r(\mathbf{x}) = \{\mathbf{x}' \in \mathcal{X} : \mathrm{dist}_O(\mathbf{x}', \mathbf{x}) \leq r\}.$$

To see how this certificate can provide assurance against evasion attacks, consider the following scenario. Suppose an edit-constrained attacker produces an attacked file $\bar{\mathbf{x}} \in \mathcal{A}_c(\mathbf{x})$ based on an original file $\mathbf{x}$. The attacked file is subsequently submitted to a malware detector, which produces an edit distance robustness certificate of radius r. If $r \geq c$ then $\mathbf{x}$ must be in the edit distance neighborhood $\mathcal{N}_r(\bar{\mathbf{x}})$, which implies $f(\mathbf{x}) = f(\bar{\mathbf{x}})$. Hence if the malware detector's prediction is correct for the original file $\mathbf{x}$ it cannot be fooled by the "attacked" file $\bar{\mathbf{x}}$.

When designing certification mechanisms in this paper, we adopt the so-called "conservative" or "sound but incomplete" paradigm [17]. Under this paradigm, a mechanism may *accept* or *decline* to issue an edit distance certificate of a given radius r. If the mechanism *accepts*, the guarantee described in Definition 2.2 must hold, possibly with high probability. On the other hand, if the mechanism *declines*, it makes no statement about whether the guarantee holds.

3 METHODOLOGY

In this section, we address research question **Q1** by adapting the certification approach of *randomized smoothing* to the malware detection domain. To begin, in Section 3.1, we review randomized smoothing and propose a deletion randomization scheme called RS-Del that is aligned with our edit distance threat model. In Section 3.2, we derive a closed form edit distance certificate for RS-Del using lower bounds on the detection confidence. Finally, in Section 3.3, we present practical algorithms for probabilistic certification and discuss how to exploit information from a disassembler to enhance RS-Del, both in terms of the deletion randomization scheme and the semantics of the edit distance certificate.

3.1 RS-Del: Randomized deletion smoothing

In robust machine learning, smoothing is a technique that averages a model's output with respect to randomized inputs. It has been applied as a heuristic defense against evasion attacks in the vision domain [10, 50], owing to its ability to reduce a model's sensitivity to noise or fine-scale variations. More recently, it has been shown to achieve certified robustness in a framework known as *randomized smoothing* [17, 43, 47, 71]. Most existing applications of randomized smoothing employ additive Gaussian or Laplace noise when randomizing inputs, yielding ℓ_p robustness certificates. However, these randomization schemes are inappropriate for malware detection, as they erroneously assume input byte values are numeric when they are best treated as categorical, and they erroneously assume input files are the same size, even though file sizes may vary. To address these incompatibilities, we propose *randomized deletion smoothing* (RS-Del) which randomizes inputs by deleting bytes, while yielding edit distance robustness certificates.

3.1.1 Smoothed malware detectors. We begin with a generic formulation of randomized smoothing following Lee et al. [44]. Consider a "base" malware detector f_b and a randomization scheme $\phi \colon \mathcal{X} \to \mathcal{D}(\mathcal{X})$ that maps an input file to a distribution over the space of input files. Let

$$p_y(\mathbf{x}; f_\mathrm{b}) = \Pr_{\mathbf{z} \sim \phi(\mathbf{x})} [f_\mathrm{b}(\mathbf{z}) = y] \tag{2}$$

denote the probability that f_b predicts y for an input file $\mathbf{x}$ randomized according to ϕ (we omit the dependence on f_b where it is clear from context). The *smoothed* malware detector f composed from f_b and ϕ is defined as

$$f(\mathbf{x}) = \operatorname*{arg\,max}_{y \in \{0,1\}} p_y(\mathbf{x}) - \eta_y \tag{3}$$

where $\eta_1 \in (0, 1)$ is a decision threshold and $\eta_0 := 1 - \eta_1$. In words, the smoothed malware detector predicts y if the base detector predicts y with probability $p_y(\mathbf{x})$ exceeding η_y for random inputs drawn from $\phi(\mathbf{x})$.

Remark 3.1. Previous definitions of randomized smoothing do not incorporate a tunable decision threshold η_1 and effectively assume $\eta_1 = \eta_0 = \frac{1}{2}$. A tunable decision threshold is useful for malware detection as a way of controlling false positive and false negative errors. It can be tuned in addition to any decision thresholds associated with the base detector.

3.1.2 Design considerations for ϕ. The behavior of a smoothed malware detector is strongly influenced by the choice of randomization scheme ϕ. When choosing a scheme, we must trade off utility (accuracy) and robustness. Practically, we can improve utility by choosing a scheme that adds less noise to the input, especially noise that would obscure or destroy information relevant to detection. On the other hand, we can improve robustness by choosing a scheme that adds more noise, so that neighboring randomized inputs become indistinguishable to the base detector. More precisely, we would like the statistical distance between $\phi(\mathbf{x})$ and $\phi(\bar{\mathbf{x}})$ to be small for any input files $\mathbf{x}$ and $\bar{\mathbf{x}}$ that are close in edit distance. Secondary to robustness and accuracy considerations, we also consider the efficiency of certification. Deriving a tight computationally efficient certificate may be difficult or impossible for some randomization schemes—in the worst case it may be difficult to outperform certification by brute force search.

3.1.3 Deletion randomization scheme. To satisfy the design considerations, we propose a randomization scheme that edits an input file by deleting bytes. We specify the scheme as a two-stage process. In the first stage, a random edit ϵ is drawn from a distribution $G(\mathbf{x})$ over the space of possible edits to $\mathbf{x}$, denoted $\mathcal{E}(\mathbf{x})$. Since we only consider byte deletions, any edit can be represented as a set of byte indices in $\{1, \ldots, |\mathbf{x}|\}$ that remain post-deletion. Thus $\mathcal{E}(\mathbf{x}) = 2^{\{1,\ldots,|\mathbf{x}|\}}$. We set the distribution $G(\mathbf{x})$ so that each byte is deleted i.i.d. with probability $p_{\mathrm{del}} \in (0, 1)$:

$$\Pr[G(\mathbf{x}) = \epsilon] = \prod_{i=1}^{|\mathbf{x}|} p_{\mathrm{del}}^{[i \notin \epsilon]} (1 - p_{\mathrm{del}})^{[i \in \epsilon]}. \tag{4}$$

In the second stage, the edit ϵ drawn from $G(\mathbf{x})$ is applied to $\mathbf{x}$ to yield a new file:

$$\mathbf{z} = \mathrm{apply}(\mathbf{x}, \epsilon) \coloneqq \left(x_{\epsilon_{(i)}}\right)_{i=1\ldots|\epsilon|}, \tag{5}$$

where $\epsilon_{(i)}$ denotes the i-th smallest element in ϵ. The new file $\mathbf{z}$ is guaranteed to be a subsequence of $\mathbf{x}$. Putting both stages together, the distribution of our randomization scheme ϕ satisfies

$$\Pr[\phi(\mathbf{x}) = \mathbf{z}] = \sum_{\epsilon \in \mathcal{E}(\mathbf{x})} \Pr[G(\mathbf{x}) = \epsilon] \mathbf{1}_{\mathrm{apply}(\mathbf{x},\epsilon)=\mathbf{z}}. \tag{6}$$

Remark 3.2. It may be surprising that our randomization scheme does not use the full set of edit ops O available to the attacker. It is a misconception that smoothing requires perfect alignment between the randomization scheme and the threat model. All that is needed from a robustness perspective, is for the scheme to return distributions that are statistically close for any pair of inputs that are neighboring according to the threat model; this can be achieved solely with deletion. In fact, perfect alignment is known to be suboptimal for some ℓ_p threat models [91]. Our deletion scheme leads to a tractable robustness certificate covering the full set of edit ops (see Section 3.2). Moreover while benefiting robustness, our empirical results show that our deletion scheme has only a minor impact on accuracy (see Section 4.2). Finally, our deletion scheme reduces the size of the input file, which is beneficial for computational efficiency (see Appendix D). This is not true in general for schemes employing insertions/substitutions.

3.2 Edit distance certificate

We now turn to the problem of deriving an edit distance robustness certificate for RS-Del. We specify information the certificate may depend on in Section 3.2.1. We then present the derivation in three parts: Section 3.2.2 provides an outline, Section 3.2.3 derives a lower bound on the probability score of RS-Del and Section 3.2.4 uses the bound to complete the derivation. All proofs are presented in Appendix B.

3.2.1 Information availability. Following prior work [17, 43], we assume limited information about RS-Del is available when computing a certificate. This is to both improve tractability and ensure the certificate does not depend on architectural details of the base detector used with RS-Del. Concretely, let $\bar{\mathbf{x}} \in \mathcal{X}$ be a (possibly adversarial) input file for which we would like to certify the robustness of RS-Del, denoted f. The only information we use when deriving the certificate is: (1) the input file $\bar{\mathbf{x}}$, (2) the prediction of RS-Del $y = f(\bar{\mathbf{x}})$, (3) the probability score of RS-Del for the prediction $\mu_y = p_y(\bar{\mathbf{x}}; f_{\mathrm{b}})$, (4) the decision threshold η_y of RS-Del, and (5) the deletion randomization scheme ϕ specified in (6).

3.2.2 Derivation outline. We derive edit distance robustness certificates aligned with our threat model (see Section 2.2). In doing so, we consider attackers with varying constraints on the edit ops O they can apply. The main results are summarized in Table 1, where we provide the radius r of the certificate as a function of y, μ_y, η_y and p_{del}.

To set the stage for the derivation, recall from Definition 2.2 that an edit distance robustness certificate of radius r can be issued for an input $\bar{\mathbf{x}}$ iff $f(\bar{\mathbf{x}}) = f(\mathbf{x})$ for all $\mathbf{x}$ in the edit distance neighborhood $\mathcal{N}_r(\bar{\mathbf{x}})$. This condition is equivalent to requiring that RS-Del's probability scores for y exceed the detection threshold η_y in the neighborhood, i.e.

$$\min_{\mathbf{x} \in \mathcal{N}_r(\bar{\mathbf{x}})} p_y(\mathbf{x}; f_{\mathrm{b}}) > \eta_y. \tag{7}$$

While it is theoretically possible to solve the minimization problem above, it is technically infeasible due to the size of the neighborhood and the apparent need to resort to brute force search (see Appendix A). We therefore replace the LHS of (7) by a tractable lower bound, noting that if the resulting inequality holds, then (7) holds and we may issue a certificate.

We proceed with the derivation in two steps. In the first step, covered in Section 3.2.3, we replace the objective of the minimization problem $p_y(\mathbf{x}; f_{\mathrm{b}})$ by a lower bound. Then in the second step, covered in Section 3.2.4, we complete the derivation by minimizing the lower bound over the edit distance neighborhood.

3.2.3 Lower bound on the probability scores. We seek a lower bound on the RS-Del's probability score $p_y(\mathbf{x}; f_{\mathrm{b}})$ that satisfies the following requirements: (1) the bound must hold for all $\mathbf{x}$ in

Table 1: Edit distance certificates as a function of the edit ops O the attacker is capable of, the strength of deletion smoothing p_{del}, the confidence μ_y of RS-Del in its prediction y, and the decision threshold η_y.

Edit ops O	{ins}	{del}	{del, ins}	{del, ins, sub}	{sub}	{ins, sub}	{del, sub}
Edit dist. name	Episode [18]	–	LCS	Levenshtein	Hamming	–	–
Certified radius r	$\left\lfloor \frac{\log \frac{1-\mu_y}{1-\eta_y}}{\log p_{\text{del}}} \right\rfloor$	$\left\lfloor \frac{\log \frac{\eta_y}{\mu_y}}{\log p_{\text{del}}} \right\rfloor$	$\left\lfloor \frac{\log \frac{\eta_y}{\mu_y}}{\log p_{\text{del}}} \right\rfloor$	$\left\lfloor \frac{\log(1+\eta_y-\mu_y)}{\log p_{\text{del}}} \right\rfloor$	$\left\lfloor \frac{\log(1+\eta_y-\mu_y)}{\log p_{\text{del}}} \right\rfloor$	$\left\lfloor \frac{\log(1+\eta_y-\mu_y)}{\log p_{\text{del}}} \right\rfloor$	$\left\lfloor \frac{\log(1+\eta_y-\mu_y)}{\log p_{\text{del}}} \right\rfloor$

the edit distance neighborhood $\mathcal{N}_r(\bar{\mathbf{x}})$, and (2) the bound must be independent of the base detector f_b which is assumed unknown.

To begin, we write $p_y(\mathbf{x}; f_\text{b})$ as a sum over the edit space by combining (2) and (6):

$$p_y(\mathbf{x}; f_\text{b}) = \sum_{\epsilon \in \mathcal{E}(\mathbf{x})} s(\epsilon, \mathbf{x}; f_\text{b}), \tag{8}$$

$$\text{with } s(\epsilon, \mathbf{x}; f_\text{b}) = \Pr\left[G(\mathbf{x}) = \epsilon\right] \mathbf{1}_{f_\text{b}(\text{apply}(\mathbf{x},\epsilon)=y)}. \tag{9}$$

We would like to rewrite this in terms of the known probability score at $\bar{\mathbf{x}}$, $\mu_y = p_y(\bar{\mathbf{x}}; f_\text{b}) = \sum_{\bar{\epsilon} \in \mathcal{E}(\bar{\mathbf{x}})} s(\bar{\epsilon}, \bar{\mathbf{x}}; f_\text{b})$. To do so, we identify pairs of edits ϵ to $\mathbf{x}$ and $\bar{\epsilon}$ to $\bar{\mathbf{x}}$ for which the corresponding terms $s(\epsilon, \mathbf{x}; f_\text{b})$ and $s(\bar{\epsilon}, \bar{\mathbf{x}}; f_\text{b})$ are proportional.

Lemma 3.3 (Equivalent edits). *Let $\mathbf{z}^\star$ be a longest common subsequence (LCS) [86] of $\mathbf{x}$ and $\bar{\mathbf{x}}$, and let $\epsilon^\star \in \mathcal{E}(\mathbf{x})$ and $\bar{\epsilon}^\star \in \mathcal{E}(\bar{\mathbf{x}})$ be any edits such that* $\text{apply}(\mathbf{x}, \epsilon^\star) = \text{apply}(\bar{\mathbf{x}}, \bar{\epsilon}^\star) = \mathbf{z}^\star$. *Then there exists a bijection $m : 2^{\epsilon^\star} \to 2^{\bar{\epsilon}^\star}$ such that* $\text{apply}(\mathbf{x}, \epsilon) = \text{apply}(\bar{\mathbf{x}}, \bar{\epsilon})$ *for any $\epsilon \subseteq \epsilon^\star$ and $\bar{\epsilon} = m(\epsilon)$. Furthermore, we have $s(\epsilon, \mathbf{x}; f_\text{b}) = p_{\text{del}}^{|\mathbf{x}|-|\bar{\mathbf{x}}|} s(\bar{\epsilon}, \bar{\mathbf{x}}; f_\text{b})$.*

Applying this proportionality result to all pairs of edits $\epsilon, \bar{\epsilon}$ related under the bijection m yields:

$$\sum_{\epsilon \in 2^{\epsilon^\star}} s(\epsilon, \mathbf{x}; f_\text{b}) = p_{\text{del}}^{|\mathbf{x}|-|\bar{\mathbf{x}}|} \sum_{\bar{\epsilon} \in 2^{\bar{\epsilon}^\star}} s(\bar{\epsilon}, \bar{\mathbf{x}}; f_\text{b}).$$

Thus we can achieve our goal of writing $p_y(\mathbf{x}; f_\text{b})$ in terms of $\mu_y = p_y(\bar{\mathbf{x}}; f_\text{b})$. A simple rearrangement of terms gives:

$$p_y(\mathbf{x}; f_\text{b}) = p_{\text{del}}^{|\mathbf{x}|-|\bar{\mathbf{x}}|} \left(\mu_y - \sum_{\bar{\epsilon} \notin 2^{\bar{\epsilon}^\star}} s(\bar{\epsilon}, \bar{\mathbf{x}}; f_\text{b}) \right) + \sum_{\epsilon \notin 2^{\epsilon^\star}} s(\epsilon, \mathbf{x}; f_\text{b}). \tag{10}$$

This representation is convenient for deriving a lower bound. Specifically, we can drop the sum over $\epsilon \notin 2^{\epsilon^\star}$ and upper-bound the sum over $\epsilon \notin 2^{\bar{\epsilon}^\star}$ to obtain a lower bound that is independent of f_b.

Theorem 3.4 (Lower bound). *Let $\mathbf{z}^\star$ be a longest common subsequence of $\mathbf{x}$ and $\bar{\mathbf{x}}$, and assume $\mu_y = p_y(\bar{\mathbf{x}}; f_\text{b})$. Then*

$$p_y(\mathbf{x}; f_\text{b}) \geq \text{lb}(\mathbf{x}; \bar{\mathbf{x}}, \mu_y) = p_{\text{del}}^{|\mathbf{x}|-|\bar{\mathbf{x}}|} \left(\mu_y - 1 + p_{\text{del}}^{|\bar{\mathbf{x}}|-|\mathbf{z}^\star|} \right). \tag{11}$$

3.2.4 Edit distance certificate. To complete the derivation we minimize the lower bound in (11) over the edit distance neighborhood:

$$\rho(\bar{\mathbf{x}}; \mu_y) = \min_{\mathbf{x} \in \mathcal{N}_r(\bar{\mathbf{x}})} \text{lb}(\mathbf{x}; \bar{\mathbf{x}}, \mu_y). \tag{12}$$

Recall that we are interested in general edit distance neighborhoods, where the edit ops O used to define the edit distance may be constrained—e.g., deletions may not be allowed in the threat model of the attacker. As a step towards solving the minimization problem, it is therefore useful to express $\text{lb}(\mathbf{x}; \bar{\mathbf{x}}, \mu_y)$ in terms of the edit ops, as shown below.

Corollary 3.5. *Suppose there exists an edit path from $\mathbf{x}$ to $\bar{\mathbf{x}}$ that consists of n_{sub} substitutions, n_{ins} insertions and n_{del} deletions such that $n_{\text{sub}} + n_{\text{ins}} + n_{\text{del}} = \text{dist}_O(\mathbf{x}, \bar{\mathbf{x}})$ and $n_{\text{sub}}, n_{\text{ins}}, n_{\text{del}} \geq 0$. Then*

$$\text{lb}_y(\mathbf{x}; \bar{\mathbf{x}}, \mu_y) = p_{\text{del}}^{n_{\text{del}}-n_{\text{ins}}} \left(\mu_y - 1 + p_{\text{del}}^{n_{\text{sub}}+n_{\text{ins}}} \right).$$

This parameterization of the lower bound enables us to re-express (12) as an optimization problem over counts of edit ops:

$$\rho(\bar{\mathbf{x}}; \mu_y) = \min_{n_{\text{sub}}, n_{\text{ins}}, n_{\text{del}} \in C_r} p_{\text{del}}^{n_{\text{del}}-n_{\text{ins}}} \left(\mu_y - 1 + p_{\text{del}}^{n_{\text{sub}}+n_{\text{ins}}} \right), \tag{13}$$

where C_r encodes constraints on the set of counts. If the edit ops O are unconstrained so that insertions, deletions and substitutions are all allowed, then the edit distance is known as the *Levenshtein distance* and C_r consists of sets of counts that sum to r. We solve the minimization problem for this case below.

Theorem 3.6 (Levenshtein distance certificate). *Suppose an RS-Del malware detector f predicts y with probability score μ_y for input file $\bar{\mathbf{x}}$. Then a lower bound on the malware detector's probability score within the Levenshtein distance neighborhood $\mathcal{N}_r(\bar{\mathbf{x}})$ is $\rho(\bar{\mathbf{x}}; \mu_y) = \mu_y - 1 + p_{\text{del}}^r$. It follows that the largest radius at which we can issue a Levenshtein distance robustness certificate is*

$$r = \left\lfloor \frac{\log\left(1 + \eta_y - \mu_y\right)}{\log p_{\text{del}}} \right\rfloor.$$

This is known as the certified radius.

It is straightforward to adapt this result to account for constraints on the edit ops O. Results for all combinations of edit ops are provided in Table 1.

3.3 Practical considerations

Before we can implement RS-Del and the certificate developed in the previous sections, we must consider several practical issues. First, given that evaluating RS-Del exactly requires brute force enumeration over all possible deletions, we present a sampling-based approximation in Section 3.3.1 and adapt the robustness certificate to account for sampling error. Second, in Section 3.3.2, we show how to train a base malware detector for use RS-Del. And third, in Section 3.3.3, we discuss how to use information from a disassembler to improve the semantics of the threat model and certificates. In particular, we advocate for associating groups of bytes with instructions, and constraining the threat model to operate on such bytes as one unit. This can prune invalid instructions from the attacker's action space, leading to larger certified regions. We note that the sampling-based approximation and input randomization training described in Sections 3.3.1 and 3.3.2 follow standard practice in the randomized smoothing literature [17].

3.3.1 Approximating RS-Del via sampling. To evaluate and certify RS-Del at input $\bar{\mathbf{x}}$, we must compute the probability scores $\mu_y = p_y(\bar{\mathbf{x}})$ defined in (2). Computing the scores *exactly* is infeasible, since it is necessary to enumerate all subsequences of $\bar{\mathbf{x}}$ and their probabilities under $\phi(\bar{\mathbf{x}})$, which takes exponential time in $|\bar{\mathbf{x}}|$. We therefore follow standard practice in randomized smoothing [17] and *approximate* the score μ_y using a sample of randomized inputs $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ drawn i.i.d. from $\phi(\bar{\mathbf{x}})$:

$$\hat{\mu}_y = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{f_b(\mathbf{z}_i)=y}. \tag{14}$$

We can estimate a lower bound on μ_y to account for sampling error by applying a binomial test:

$$\underline{\mu}_y = \mathtt{LowerConfBound}\left(\textstyle\sum_{i=1}^{n} \mathbf{1}_{f_b(\mathbf{z}_i)=y}, n, \alpha\right) \tag{15}$$

where $\mathtt{LowerConfBound}(k, n, \alpha)$ returns a one-tailed $1-\alpha$ confidence interval for μ_y given a sample $k \sim \text{Binomial}(n, \mu_y)$. We use this lower bound in Algorithm 1 to compute a certificate that holds with probability $1-\alpha$. The validity of Algorithm 1 relies on a generalization of Theorem 3.6 and Table 1 to the probabilistic setting as asserted below.

Corollary 3.7. *Suppose Algorithm 1 predicts y with certified radius r. Then an edit distance robustness certificate of radius r holds at $\bar{\mathbf{x}}$ with probability $1-\alpha$.*

We now make a few comments about the design and usage of Algorithm 1. We emphasize that independent samples are used for estimating RS-Del's prediction (lines 1–2) and bounding RS-Del's confidence in that prediction (line 3). This avoids the need to perform a correction for multiple hypothesis tests. In our experiments, we use a smaller number of samples for prediction $n_{\text{pred}} = 1000$ since we observe low variance, and a higher number of samples for bounding the confidence $n_{\text{bound}} = 4000$, since doing so may potentially improve the radius. If the lower bound on the probability score for the predicted class $\underline{\mu}_{\hat{y}}$ is less than the class-specific threshold $\eta_{\hat{y}}$, we require that RS-Del abstains from making a prediction since it would not be robust (line 6).

3.3.2 Training RS-Del. Though RS-Del is theoretically compatible with any base detector, it will generally perform poorly for conventionally trained base detectors. This is because the distribution of randomized inputs is likely to be very different from the distribution of non-randomized inputs encountered during training. To mitigate this issue, we follow standard practice in randomized smoothing and train the base detector on randomized inputs [43]. In other words, when iterating over batches of files during training, we apply the deletion randomization scheme to the files before passing them to the base detector, ensuring that the randomization varies each time a file is encountered. We ensure the p_{del} parameter for the deletion randomization scheme is set to the same value during training and testing. Note that our certifications are sound irrespective of how the base model is trained—the present training process aims to improve utility.

Algorithm 1: Probabilistic certification for RS-Del

input : input file $\bar{\mathbf{x}}$, base detector f_b, deletion randomization scheme ϕ with probability p_{del}, decision threshold η_1, significance level α, number of samples used for prediction n_{pred} and lower bound n_{bound}, edit ops O

output: prediction and certified radius

```
1 Compute μ̂_y using Eq. (14) and n_pred randomized inputs
2 Estimate prediction ŷ ← arg max_{y∈{0,1}} μ̂_y − η_y
3 Compute μ_ŷ (lower) using Eq. (15) and n_bound randomized inputs
4 Compute certified radius r using Table 1
5 if μ_ŷ (lower) ≥ η_ŷ then return prediction ŷ, radius r
6 else return ABSTAIN
```

3.3.3 Exploiting semantics from disassembly. Though the most elementary representation of an executable is as a sequence of bytes, it ignores the semantics of the program. If decompiling an executable is feasible, we can associate bytes with their corresponding machine instructions, data, or other structures. By exploiting such information, we can group bytes corresponding to an instruction, and regard them as a single token in the input sequence. This grouping of bytes extends to our definition of the edit distance threat model (we consider edits at the token level) and to our deletion randomization scheme (we delete at the token-level). There are a few advantages to this approach: Deletion at the token-level fully preserves the instruction semantics and improves the base detector's performance. We also obtained a tighter threat model by eliminating inputs containing invalid instructions. As a result, an instruction-level certificate covers a larger set of possible adversarial examples than a byte-level certificate of the same radius. An illustration of this concept can be found in Figure 1.

4 EVALUATION

We now empirically evaluate our proposed method RS-Del on two malware datasets to address the remaining research questions. To answer **Q2** on tradeoffs between malware detection accuracy and robustness guarantees, we vary the aggressiveness of smoothing, and provide comparisons with a non-smoothed baseline and an alternative randomized smoothing method [46] that applies to a constrained version of our threat model. Towards **Q3** on the practical robustness of RS-Del, Section 4.3 reports success rates of five published evasion attacks against RS-Del and a non-smoothed baseline. Computational efficiency and training convergence are reported in Appendix D.

4.1 Experimental setup

We next detail the experimental setup, including data sources, machine learning models for malware detection, and parameters for randomized smoothing and certification.

4.1.1 Datasets. Though our methods are compatible with executable files of any format, in our experiments we focus on the *Portable Executable (PE) format* [53], since datasets, malware detection models and adversarial attacks are more extensively available. Moreover, PE format is the standard for executables, object files and

Original input file

File offset	Byte rep.	Disassembly rep.
00000000	77	NI
00000001	90	NI
00000002	144	NI
⋮	⋮	⋮
00000400	85	push ebp
00000401	139	mov ebp, esp
00000402	236	
00000403	131	sub esp, 5Ch
00000404	236	
00000405	92	
⋮	⋮	⋮

Byte-level deletion

File offset	Byte rep.
00000000	77
00000002	144
⋮	⋮
00000400	85
00000403	131
00000404	236
⋮	⋮

Instruction-level deletion

File offset	Disassembly rep.
00000001	NI
⋮	⋮
00000400	push ebp
00000401 00000402	mov ebp, esp
⋮	⋮

Figure 1: Illustration of byte-level and instruction-level threat models. RS-Del (Section 3.3.1) is approximated by aggregating predictions of the base detector on byte-level (middle) or instruction-level (right) randomized inputs. *Left:* An executable file prior to input randomization. The byte array representation is shown in the 2nd column and partial output from the disassembler (Ghidra [56]) is shown in the 3rd column. Bytes that do not correspond to machine instructions are marked `NI`. Shading represents bytes (light gray) or instructions (dark gray) that are deleted in the randomized inputs to the right. *Middle:* A sample randomized input under the byte-level threat model, where semantic information from the disassembler is ignored. This may result in individual instructions being partially deleted. *Right:* A sample randomized input under the instruction-level threat model, where bytes corresponding to an instruction (or non-instruction `NI`) are treated as a single unit.

shared libraries in Microsoft Windows and is an attractive target for malware authors. We use two PE datasets which are summarized in Table 2 and described below.

Sleipnir2. This dataset attempts to replicate data used in past work [2], which was not published with raw samples. We reconstructed the raw malicious samples by retrieving them from VirusShare [84] using the provided hashes. Since we were unable to reconstruct the raw benign samples, we followed established protocols [41, 42, 72] to collect new benign samples. Specifically, we set up a Windows 7 virtual machine with over 300 packages installed using Chocolatey package manager [75]. We then extracted PE files from the virtual machine, which were assumed benign[2], and subsampled them to match the number of malicious samples. The dataset is randomly split into training, validation and test sets with a ratio of 60%, 20% and 20% respectively.

VTFeed. This dataset was first used in recent attacks on end-to-end ML-based malware detectors [52]. It was collected from VirusTotal—a commercial threat intelligence service—by sampling PE files from the live feed over a period of two weeks in 2020. Labels for the files were derived from the 68 antivirus (AV) products aggregated on VirusTotal at the time of collection. Files were labeled *malicious* if they were flagged malicious by 40 or more of the AV products, they were labeled *benign* if they were not flagged malicious by any of the AV products, and any remaining files were excluded. Following Lucas et al. [52], the dataset is randomly split into training, validation and test sets with a ratio of 80%, 10%, and 10% respectively.

We note that VTFeed came with strict terms of use, which prevented us from loading it on our high performance computing (HPC) cluster. As a result, we use Sleipnir2 for comprehensive experiments (e.g., varying p_{del}) on the HPC cluster, and VTFeed for a smaller selection of experiments run on a local server.

[2]Chocolatey packages are validated against VirusTotal [14].

Table 2: Summary of datasets.

Dataset	Label	Number of samples		
		Train	Validation	Test
Sleipnir2	Benign	20 948	7 012	6 999
	Malicious	20 768	6 892	6 905
VTFeed	Benign	111 258	13 961	13 926
	Malicious	111 395	13 870	13 906

4.1.2 Malware detectors. We experiment with static malware detectors based on a neural network model called MalConv [64]. MalConv was one of the first *end-to-end* models proposed for malware detection—i.e., it learns to classify directly from raw byte sequences, rather than relying on manually engineered features. Architecturally, it composes a learnable embedding layer with a shallow convolutional network. A large window size and stride of 500 bytes are employed to facilitate scaling to long byte sequences. Though MalConv is compatible with arbitrarily long byte sequences in principle, we truncate all inputs to 2MB to support training efficiency. We use the original parameter settings and training procedure [64], except where specified in Appendix E.

Using MalConv as a basis, we consider three malware detectors as described below.

NS. This detector corresponds to a non-smoothed (NS) MalConv model. It serves as a non-certified, non-robust baseline—i.e., no specific techniques are employed to improve robustness to evasion attacks and certification is not supported.

RS-Abn. This detector implements randomized smoothing using MalConv as a base detector, with an ablation randomization scheme proposed by Levine and Feizi [46]. It serves as a certified robust baseline, albeit for a more restricted threat model than the one we propose in Section 2.2. Specifically, it supports robustness certification for the Hamming distance threat model, where an attacker's capability is measured by the number of substituted bytes. Since Levine and Feizi's formulation was for images, several modifications are required to support malware detection. We adapt the encoding of ablated (masked) values by introducing a special mask token; we add support for variable-length inputs by ablating a fraction p_{ab} of the input values rather than a constant number; and we apply gradient clipping when learning parameters in the embedding layer to improve convergence (see Appendix D). We consider variants of this detector for different values of the ablation probability p_{ab}.

RS-Del. This is our proposed detector: it implements randomized smoothing using MalConv as a base detector, with our proposed deletion randomization scheme. It supports robustness certification for the generalized edit distance threat model. We consider variants of this detector for different values of the deletion probability p_{del} and the detection threshold η_1.

4.1.3 Controlling false positive rates. Malware detectors are typically tuned to have a low false positive rate (FPR) (e.g., less than 0.1–1%) since producing too many false alarms is a nuisance to users.[3] To make all detectors comparable, we report results by calibrating the FPR to be 0.5% on the test set for Section 4.2 and 0.5% on the validation set for Section 4.3 unless otherwise noted. This tuning is done by adjusting the decision threshold on the probability at which the base detector (MalConv) predicts a file to be malicious.

4.2 Accuracy and Certification of RS-Del

In this section, we address research question **Q2** by evaluating the robustness guarantees and malware detection accuracy of RS-Del. We consider two instantiations of the edit distance threat model. First, in Section 4.2.1, we consider the Levenshtein distance threat model, where the attacker's elementary edits are unconstrained and may include deletions, insertions and substitutions. Then, in Section 4.2.2, we consider the more restricted Hamming distance threat model, where an attacker is only able to perform substitutions. We summarize our findings in Section 4.2.3. Overall, we find RS-Del generates robust predictions with minimal impact on model accuracy for the Levenshtein distance threat model, and outperforms RS-Abn [46] for the Hamming distance threat model.

We report the following quantities in our evaluation:

- *Certified radius (CR).* The radius of the largest edit distance robustness certificate (see Definition 2.2) that can be issued for a given input file, malware detector and certification method. Note that this is a conservative measure of robustness since it is *tied to the certification method.* The *median CR* is computed on the test set.
- *Certified accuracy* [17, 43], also known as *verified-robust accuracy* [45, 90], evaluates robustness certificates and accuracy simultaneously with respect to a test set. It is defined as the fraction of files in the test set $\mathbb{D}$ for which the malware detector f's prediction is correct *and* certified robust at radius r:

$$\textsc{CertAcc}_r(\mathbb{D}) = \sum_{(\mathbf{x},y)\in\mathbb{D}} \frac{\mathbf{1}_{f(\mathbf{x})=y}\mathbf{1}_{\mathrm{CR}(\mathbf{x})\geq r}}{|\mathbb{D}|} \tag{16}$$

  where $\mathrm{CR}(\mathbf{x})$ denotes the certified radius for input $\mathbf{x}$ returned by the certification method.
- *Clean accuracy.* The fraction of files in the test set for which the malware detector's prediction is correct.

We briefly mention default parameter settings for experiments in this section. When approximating the smoothed malware detectors (RS-Del and RS-Abn) we sample $n_{\text{pred}} = 1000$ randomized inputs for prediction and $n_{\text{bound}} = 4000$ randomized inputs for certification, while setting the significance level α to 0.05. Unless otherwise specified, we set the decision thresholds for the smoothed detectors so that $\eta_0 = \eta_1 = 0.5$. The decision thresholds for the base detectors are tuned to yield a false positive rate of 0.5%. We note that the entire test set is used when reporting metrics and summary statistics in this section.

4.2.1 Levenshtein distance threat model. We first present results for the Levenshtein distance threat model, where the attacker's elementary edits are unconstrained ($O = \{\text{del}, \text{ins}, \text{sub}\}$). We vary three parameters associated with RS-Del: the deletion probability p_{del}, the decision threshold of the smoothed detector η_1, and the elementary token (bytes versus instructions). We use NS as a baseline as there are no prior certified defenses for this threat model to our knowledge.

Certified accuracy. Figure 2 plots the certified accuracy of RS-Del as a function of the radius r on the Sleipnir2 dataset for several values of p_{del} using byte-level Levenshtein distance. The corresponding plot for instruction-level Levenshtein distance exhibits similar behavior, and is presented in Figure 4 of Appendix C. We observe that the curves for larger values of p_{del} approximately dominate the curves for smaller values of p_{del}, for $p_{\text{del}} \leq 99.5\%$ (i.e., the accuracy is higher or close for all radii). This suggests that the robustness of RS-Del can be improved without sacrificing accuracy by increasing p_{del} up to 99.5%. However, for the larger value $p_{\text{del}} = 99.9\%$, we observe a drop in certified accuracy of around 10% for smaller radii and an increase for larger radii.

It is interesting to relate these certification results to published evasion attacks. Figure 2 shows that we can achieve a certified accuracy in excess of 90% at a Levenshtein distance radius of 128 bytes when $p_{\text{del}} = 99.5\%$. This radius is larger than the median Levenshtein distance of two attacks that manipulate headers of PE files [20, 57] (see Table 4). We can therefore provide reasonable robustness guarantees against these two attacks. However, a radius of 128 bytes is orders of magnitude smaller than the median Levenshtein distances of other published attacks which range from tens of KB [42, 52] to several MB [21] (see Table 4). While some of these attacks arguably fall outside an edit distance constrained threat model, we consider them in our empirical evaluation of robustness in Section 4.3.

Clean accuracy and abstention rates. We report clean accuracy and abstention rates in Table 7 of Appendix C, and summarize

[3] https://www.av-comparatives.org/testmethod/false-alarm-tests/

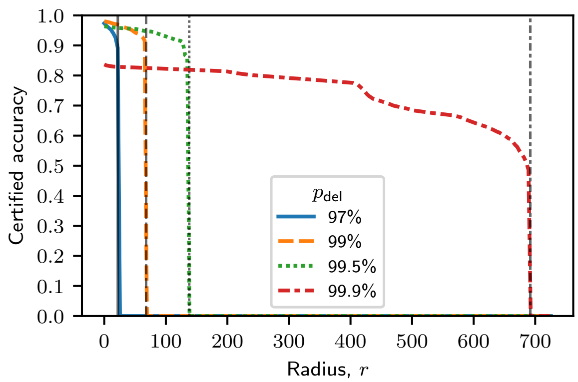

Figure 2: Certified accuracy for RS-Del as a function of the certificate radius (horizontal axis) and byte deletion probability p_{del} (colored line styles). The results are plotted for the Sleipnir2 test set under the byte-level Levenshtein distance threat model (with $O = \{\text{del}, \text{ins}, \text{sub}\}$). The grey vertical lines represent the best achievable certified radius for RS-Del (setting $\mu_y = 1$ in the expressions in Table 1). It is apparent that p_{del} controls a robustness/accuracy tradeoff. Note that in this setting, a non-smoothed, non-certified detector (NS) achieves a clean accuracy of 98%.

trends here. For the Sleipnir2 dataset, we find that clean accuracy is relatively constant for p_{del} in the range 90-99.5%, but drops by more than 10% at $p_{\text{del}} = 99.9\%$. This is in line with the trends observed for certified accuracy. We note that the clean accuracy of RS-Del (excluding $p_{\text{del}} = 99.9\%$) is at most 3% lower than the NS baseline for Sleipnir2 and at most 7% lower than the NS baseline for VTFeed.

Accuracy under high deletion. It may be surprising that RS-Del can maintain high accuracy even when deletion is aggressive. We offer some possible explanations. First, we note that even with a high deletion probability of $p_{\text{del}} = 99.9\%$, the smoothed detector accesses almost all of the file in expectation, as it aggregates $n_{\text{pred}} = 1000$ predictions from the base detector each of which accesses a random 0.1% of the file in expectation. Second, we posit that malware detection may be "easy" for RS-Del on these datasets. This could be due to the presence of signals that are robust to deletion (e.g., file size or byte frequencies) or redundancy of signals (i.e., if a signal is deleted in one place it may be seen elsewhere).

Decision threshold. Table 3 provides error rates and robustness metrics for several values of the decision threshold η_1, using byte-level Levenshtein distance with $p_{\text{del}} = 99.5\%$ (see Appendix C for plots of the certified true positive and true negative rates). When varying η_1, we also vary the decision threshold of the base detector to achieve a target false positive rate (FPR) of 0.5%. Looking at the table, we see that η_1 has minimal impact on the false negative rate (FNR), which is stable around 7%. However, there is a significant impact on the median CR (and theoretical upper bound), as reported separately for each class. The median CR is balanced for both the malicious and benign class when $\eta_1 = 50\%$, but favours

Table 3: Impact of the smoothed decision threshold η_1 on false negative error rate (FNR) and median certified radius (CR) for malicious and benign files. The false positive rate (FPR) is set to a target value of 0.5% by varying the decision threshold of the base classifier. The results are reported for Sleipnir2 with $p_{\text{del}} = 99.5\%$ using byte-level Levenshtein distance. "UB" refers to an upper bound on the median CR for a best case smoothed detector (based on Table 1 with $\mu_y = 1$).

			Median CR (UB)	
η_1 (%)	FNR (%)	FPR (%)	Malicious	Benign
50	6.8	0.5	137 (138)	137 (138)
25	6.9	0.5	275 (276)	57 (57)
10	6.8	0.5	455 (459)	20 (21)
5	6.6	0.5	578 (597)	10 (10)
1	7.1	0.5	582 (918)	1 (2)
0.5	6.9	0.5	506 (1057)	0 (0)

the malicious class as η_1 is decreased. For instance when $\eta_1 = 5\%$ a significantly larger median CR is possible for malicious files (137 to 578) at the expense of the median CR for benign files (137 to 10). This asymmetry in the class-specific CR is a feature of the theory—that is, in addition to controlling a tradeoff between error rates of each class, η_1 also controls a tradeoff between the CR for each class (see Table 1).

4.2.2 Hamming distance threat model. We now turn to the more restricted Hamming distance threat model, where the attacker is limited to performing substitutions only ($O = \{\text{sub}\}$). We choose to evaluate this threat model as it is covered in previous work on randomized smoothing, called *randomized ablation* [46] (abbreviated RS-Abn), and can serve as a baseline for comparison with our method. Recall that we adapt RS-Abn to malware detection by introducing a parameter called p_{ab}, which is the fraction of bytes that are "ablated" (replaced by a special masked value) (see Section 4.1.2). This parameter is analogous to p_{del} in RS-Del, except that the number of ablated bytes is deterministic in RS-Abn, whereas the number of deleted bytes is random in RS-Del. We compare RS-Del and RS-Abn for varying values of p_{del} and p_{ab} using the Sleipnir2 dataset and byte-level Hamming distance.

Certified accuracy. Figure 3 plots the certified accuracy of RS-Del and RS-Abn for three values of p_{del} and p_{ab}. We observe that the certified accuracy is uniformly larger for our proposed method RS-Del than for RS-Abn when $p_{\text{del}} = p_{\text{ab}}$. The superior certification performance of RS-Del is somewhat surprising given it is not optimized for the Hamming distance threat model. One possible explanation relates to the learning difficulty of RS-Abn compared with RS-Del. Specifically, we find that stochastic gradient descent is slower to converge for RS-Abn despite our attempts to improve convergence (see Appendix D).

Recall, that RS-Del provides certificates for any of the threat models in Table 1—in addition to the Hamming distance certificate—without needing to modify the randomization scheme.

Tightness. RS-Abn is provably tight, in the sense that it is not possible to issue a larger Hamming distance certificate unless more

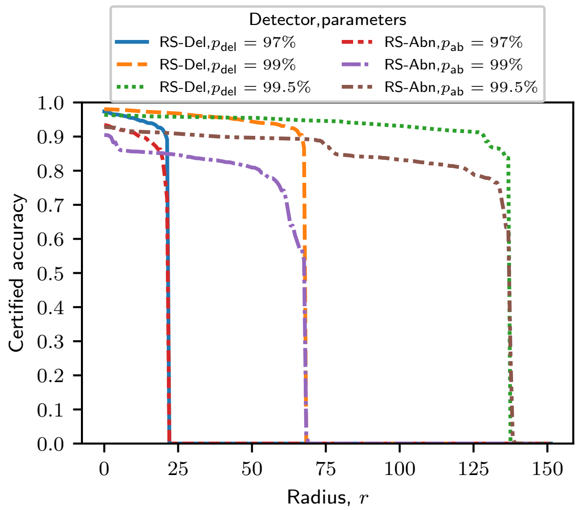

Figure 3: Certified accuracy comparison for RS-Del (our method) and RS-Abn [46] under the Hamming distance threat model for the Sleipnir2 dataset. Note that our method is not optimized for the Hamming distance threat model.

information is made available to the certification mechanism or the ablation randomization scheme is changed. This tightness result for RS-Abn, together with the empirical results in Figure 3, indicate that RS-Del produces certificates which are tight or close to tight in practice, at least for the Hamming distance threat model. This is an interesting observation, since it is unclear how to derive a tight, computationally tractable certificate for RS-Del.

4.2.3 Summary. Our evaluation shows that RS-Del provides non-trivial robustness guarantees with a low impact on accuracy. The certified radii we observe are close to the best radii theoretically achievable using our mechanism. For the Levenshtein byte-level edit distance threat model, we obtain radii of a few hundred bytes in size, which can certifiably defend against attacks that edit headers of PE files [20, 22, 57]. However, certifying robustness against more powerful attacks that modify thousands or millions of bytes remains an open challenge. By varying the detection threshold, we show that certification can be performed asymmetrically for benign and malicious files. This can boost the certified radii of malicious files by a factor of 4 in some cases. While there are no prior methods to use as baselines for the Levenshtein distance threat model, our comparisons with RS-Abn [46] for the Hamming distance threat model show that RS-Del outperforms RS-Abn in terms of both accuracy and robustness.

4.3 Empirical robustness to attacks

In this section, we address research question **Q3** by empirically evaluating the robustness of RS-Del to several published evasion attacks. By doing so, we aim to provide a more complete picture of robustness, as our certificates are conservative and may *underestimate* robustness to real attacks, which are subject to additional constraints (e.g., maintaining executability, preserving a malicious payload, etc.). We introduce the attacks in Section 4.3.1, provide details of the experimental setup in Section 4.3.2 and discuss the results in Section 4.3.3.

4.3.1 Attacks. We consider five recently published attacks designed for evading static PE malware detectors as summarized in Table 4. The attacks cover a variety of edit distance magnitudes from tens of bytes to millions of bytes. While attacks that edit millions of bytes arguably fall outside of our edit distance-constrained threat model, we include one such attack (GAMMA) to test the limits of our methodology. We note that four of the five attacks are able to operate in a black-box setting and can therefore be applied directly to RS-Del. However most of the white-box attacks, including Slack, are designed to attack neural network-base detectors with an initial embedding layer. It is not obvious how to apply these attacks to RS-Del, as the deletion sampling deviates from the assumed architecture, and makes computing gradients difficult. As an alternative, we therefore transfer the white-box Slack attack from NS to RS-Del.

4.3.2 Experimental setup. Since some of the attacks take hours to run per file, we use smaller evaluation sets containing malware subsampled from the test sets in Table 2. The evaluation set for Sleipnir2 consists of 500 files, and the one for VTFeed consists of 100 files (matching [52]). We note that our evaluation sets are comparable in size to prior work [40, 42, 77]. For each evaluation set, we report attack success rates against malware detectors trained on the same dataset.

Since all attacks employ greedy optimization with randomization, they may fail on some runs, but succeed on others. We therefore repeat each attack 5 times per file and use the best performing attacked file in our evaluation. We define the attack success rate as the proportion of files initially detected as malicious for which at least one of the 5 attack repeats is successful at evading detection. Lower attack success rates correlate with improved robustness against attacks. We permit all attacks to run for up to 200 attack iterations of the internal optimizer. Early stopping is enabled for those attacks that support it (Disp, Slack, GAMMA), which means the attack terminates as soon as the malware detector's prediction flips from malicious to benign.

Where possible, we run *direct attacks* against RS-Del and compare success rates against NS as a baseline. We also consider *transfer attacks* from NS to RS-Del as an important variation to the threat model, where an attacker has limited access to the target RS-Del during attack optimization. When running direct attacks against RS-Del, we use a reduced number of samples ($n_{\text{pred}} = 100$) to make the computational cost of the attacks more manageable. For both direct and transfer attacks against RS-Del, we set $p_{\text{del}} = 97\%$ and use bytes as the elementary tokens for smoothing.

4.3.3 Results. Table 5 presents results for transfer attacks from NS to RS-Del. The results for direct attacks against RS-Del are presented in Table 6. Almost all of the attacks transfer poorly to RS-Del. In most cases the attack success rates drop to zero or single digit percentages. This may be evidence of increased robustness, towards **Q3**. Among the attacks, GAMMA is an exception: included to test the limits of RS-Del, GAMMA produces attacks with edit distances several orders of magnitude greater than the certifications.

Table 4: Evasion attacks used in our evaluation. The *attack distance* refers to the median Levenshtein distance computed on a set of 500 attacked files from the Sleipnir2 test set. We use a closed source implementation of Disp and open source implementations of the remaining attacks based on `secml-malware` [19].

Attack	Supported settings	Attack distance	Optimizer	Description
Disp [52]	White-box, black-box	17.2 KB	Gradient-guided	Disassembles the PE file and displaces chunks of code to a new section, replacing the original code with semantic nops.
Slack [42]	White-box	34.7 KB	Fast Gradient Sign Method [30]	Replaces non-functional bytes in slack regions or the overlay of the PE file with adversarially-crafted noise.
HDOS [20]	White-box, black-box	17.0 B	Genetic algorithm	Manipulates bytes in the DOS header of the PE file which are not used in modern Windows.
HField [57]	White-box, black-box	58.0 B	Genetic algorithm	Manipulates fields in the header of the PE file (debug information, section names, checksum, etc.) which do not impact functionality.
GAMMA [21]	Black-box	2.10 MB	Genetic algorithm	Appends sections extracted from benign files to the end of a malicious PE file and modifies the header accordingly.

Table 5: Success rates of attacks transferred from NS to RS-Del in white- and black-box settings.

Setting	Attack	Dataset	Success rate (%) NS	RS-Del
White-box	Disp [52]	Sleipnir2	73.8	0.414
		VTFeed	94.1	0.0
	Slack [42]	Sleipnir2	57.9	2.90
		VTFeed	96.0	1.01
Black-box	HDOS [20]	Sleipnir2	0.0	0.0
		VTFeed	0.0	0.0
	HField [57]	Sleipnir2	0.607	0.0
		VTFeed	0.990	0.0
	Disp [52]	Sleipnir2	0.607	0.0
		VTFeed	10.9	0.0
	GAMMA [21]	Sleipnir2	99.2	99.6
		VTFeed	76.2	100.0

Table 6: Success rates of direct black-box attacks against RS-Del and the NS baseline.

Attack	Dataset	Success rate (%) NS	RS-Del
HDOS [20]	Sleipnir2	0.0	0.0
	VTFeed	0.0	0.0
HField [57]	Sleipnir2	0.607	0.0
	VTFeed	0.990	0.0
Disp [52]	Sleipnir2	0.809	0.0
	VTFeed	10.9	0.0
GAMMA [21]	Sleipnir2	99.2	54.1
	VTFeed	76.2	100.0

We hypothesize that GAMMA adds so much benign content that it overwhelms the malicious signal—enough to cross the decision boundary—akin to a good word attack [51]. We find that HDOS and HField are ineffective for both RS-Del and the baseline NS. Both attacks change up to 58 bytes in the header, and tend to fall within our certifications.

5 RELATED WORK

Since Goodfellow et al. [30] reported the vulnerability of neural networks to adversarial examples, numerous mitigations have been proposed, with most aiming for best response against a particular attack [85]. By contrast, certified robustness aims to guarantee a model's output does not change under adversarial perturbations. The literature predominantly considers *norm-bounded perturbations* in computer vision under ℓ_p norms. *Deterministic approaches* to certification [24, 31, 45, 55, 66, 81, 88, 88, 89, 94] achieve this objective by computing outer bounds on a model's possible outputs under perturbation. Such approaches apply to specific network architectures, by employing convex relaxation or exploiting piecewise-linear structures, limiting their adaptation to generic base models and new domains. As an alternative, *randomized smoothing* [17, 43, 44, 46] provides high-probability guarantees with flexible certification mechanisms. Randomized smoothing is agnostic to the inner workings of the model, and only uses API access to inference: randomness is introduced to model inputs, followed by aggregation of corresponding predictions. In this paper we adapt randomized smoothing to malware detection. Section 3 develops a novel smoothing mechanism based on random deletions, and offers practical recommendations on tractable Monte Carlo approximation, effective model training, and improved performance through operating at the level of instructions.

Despite the rich body of research and useful abstraction, general ℓ_p*-norm-bounded* threat models are inadequate for many problems including perturbations to executable files considered in this work. Even in computer vision, ℓ_p-norm bounded defenses can be circumvented by image translation, rotation, blur, and other human-imperceptible transformations that induce extremely large

ℓ_p distances. One technique to address this issue is to re-parametrize the norm-bounded distance in terms of image transformation parameters [28, 32, 48]. In other words, instead of certifying instances to be robust against ℓ_p perturbations, one may consider ℓ_p distance in terms of transformation parameters. Natural language, on the other hand, faces a different issue: while the general ℓ_p threat model covers adversarial word substitution [67], it is too broad and covers many actual (non-adversarial) examples as well. For example, "He loves cat" and "He hates cat" are both 1 word Hamming distance away from "He likes cat", but are semantically different. A certificate of radius 1 will force a wrong prediction for at least one neighbor. To address this, Jia et al. [37] and Ye et al. [92] constrain the threat model to synonyms only.

Similar to natural language, adversarial examples encountered in malware are constrained by the semantics of the platform and instruction specifications. However, in this paper we go beyond the word substitution threat model of previous work [67], as consideration of insertions and deletions is necessary in malware detection. Such edits are not captured by the ℓ_p threat model: there is no fixed file size, and even when edits are size-preserving, a few edits may lead to large ℓ_p distances. Arguably, our edit distance threat model and RS-Del mechanism are of independent interest to natural language also.

Several empirical defense methods have been proposed to improve robustness of ML classifiers [23, 63]. Íncer Romeo et al. [36] compose manually crafted Boolean features with a classifier that is constrained to be monotonically increasing with respect to selected inputs. This approach permits a combination of (potentially vulnerable) learned behavior with domain knowledge, and thereby aims to mitigate adversarial examples. Demontis et al. [23] show that the sensitivity of linear support vector machines to adversarial perturbations can be reduced by training with ℓ_∞ regularization of weights. In another work, Quiring et al. [63] take advantage of heuristic-based semantic gap detectors and an ensemble of feature classifiers to improve empirical robustness. Compared to our work on certified adversarial defenses, these approaches do not provide formal guarantees.

Binary normalization [9, 15, 61, 87] was originally proposed to defend against polymorphic/metamorphic malware, and can also be seen as a mitigation to certain adversarial examples. It attempts to sanitize binary obfuscation techniques by mapping malware to a canonical form before running a detection algorithm. However, binary normalization cannot fully mitigate attacks like Disp (see Table 4), as deducing opaque and evasive predicates are NP-hard problems [52].

Dynamic analysis can provide additional insights for malware detection. In particular, it can record a program's behavior while executing it in a sandbox (e.g., collecting a call graph or network traffic) [38, 62, 68, 93, 95]. Though detectors built on top of dynamic analysis can be more difficult to evade, as the attacker needs to obfuscate the program's behavior, they are still susceptible to adversarial perturbations. For example, an attacker may insert API calls to obfuscate a malware's behavior [27, 34, 69, 70]. Applying RS-Del to certify detectors that operate on call sequences [95] or more general dynamic features would be an interesting future direction.

6 CONCLUSION

In this paper we study certified robustness of machine learned malware detectors. There has been relatively little research on certification outside the computer vision domain, where threats are modeled as ℓ_p-norm bounded perturbations. By contrast, malware detection is a highly adversarial setting where evasion is the rule not the exception and where the action space does not preserve length—ruling out the ℓ_p-norm altogether. We organize our study across three research questions **Q1**–**Q3**.

We address **Q1** on the feasibility of certified robustness for malware detection, by identifying an appropriate edit distance threat model and designing a randomized smoothing-based certification mechanism. Our threat model covers adversaries that can make substitution, deletion or insertion perturbations, and is likely of independent interest beyond the malware domain. Our novel randomized smoothing mechanism called RS-Del, can produce certified guarantees within this threat model (and several variations) using only API access to the malware detector.

We respond to **Q2** on the size of certified guarantees and costs to accuracy by carefully evaluating RS-Del on two malware datasets using a recent static deep malware detector [64]. Besides providing certified radii that are close to the best achievable using our theoretical analysis, we find that RS-Del can certify a radius as large as 128 bytes (in Levenshtein distance) without significant loss in detection accuracy. A certificate of this size covers in excess of 10^{606} files in the proximity of a 10KB input file.

In recognition that certifications are necessarily conservative, for **Q3** we examine RS-Del in the presence of five recently published attacks. We find that while empirical robustness is not absolute, it does extend beyond the certified radius, when attacks are of a modest size, i.e., where the edit distance threat model is appropriate.

Our results suggest a number of directions for future work. It would be interesting to adapt RS-Del to malware detectors with dynamic analysis, e.g., using recorded call sequences [95]. Where certifications may naturally define regions in feature space [28, 32, 48], it would be most helpful to relate guarantees to natural actions of an attacker. Operationalizing certifications has so far eluded systems in computer vision, but may be more forthcoming in malware detection where both automation and human analysts are prevalent. Finally, certifying robustness against attacks that modify thousands or millions of bytes remains an open challenge.

REFERENCES

[1] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. 2020. When Malware is Packin' Heat; Limits of Machine Learning Classifiers Based on Static Analysis Features. In *Proceedings of Symposium on Network and Distributed System Security (NDSS)*. The Internet Society. https://doi.org/10.14722/ndss.2020.24310

[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 (S&PW)*. IEEE, 76–82. https://doi.org/10.1109/SPW.2018.00020

[3] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating Natural Language Adversarial Examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2890–2896. https://doi.org/10.18653/v1/D18-1316

[4] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of the 35th International Conference on Machine*

Learning (Proceedings of Machine Learning Research, Vol. 80). PMLR, 274–283. https://proceedings.mlr.press/v80/athalye18a.html

[5] Avast Software. [n. d.]. Malware detection and blocking. Retrieved 2022-12-22 from https://www.avast.com/en-us/technology/malware-detection-and-blocking

[6] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. 2006. Can machine learning be secure?. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. Association for Computing Machinery, 16–25. https://doi.org/10.1145/1128817.1128824

[7] Blackberry Limited. 2022. Cylance AI from Blackberry. Retrieved 2022-11-25 from https://www.blackberry.com/us/en/products/cylance-endpoint-security/cylance-ai

[8] Broadcom. 2022. How does Symantec Endpoint Protection use advanced machine learning? Retrieved 2022-12-22 from https://techdocs.broadcom.com/us/en/symantec-security-software/endpoint-security-and-management/endpoint-protection/all/Using-policies-to-manage-security/preventing-and-handling-virus-and-spyware-attacks-v40739565-d49e172/how-does-use-advanced-machine-learning-v120625733-d47e275.html

[9] Danilo Bruschi, Lorenzo Martignoni, and Mattia Monga. 2007. Code Normalization for Self-Mutating Malware. *IEEE Security & Privacy* 5, 2 (2007), 46–54. https://doi.org/10.1109/MSP.2007.31

[10] Xiaoyu Cao and Neil Zhenqiang Gong. 2017. Mitigating Evasion Attacks to Deep Neural Networks via Region-Based Classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC)*. Association for Computing Machinery, New York, NY, USA, 278–287. https://doi.org/10.1145/3134600.3134606

[11] Nicholas Carlini, Florian Tramer, Krishnamurthy, Dvijotham, and J. Zico Kolter. 2022. (Certified!!) Adversarial Robustness for Free! https://doi.org/10.48550/ARXIV.2206.10550

[12] Nicholas Carlini and David Wagner. 2016. Defensive Distillation is Not Robust to Adversarial Examples. https://doi.org/10.48550/ARXIV.1607.04311

[13] Panagiotis Charalampopoulos, Solon P. Pissis, Jakub Radoszewski, Tomasz Waleń, and Wiktor Zuba. 2020. Unary Words Have the Smallest Levenshtein k-Neighbourhoods. In *31st Annual Symposium on Combinatorial Pattern Matching (CPM 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 161)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 10:1–10:12. https://doi.org/10.4230/LIPIcs.CPM.2020.10

[14] Chocolatey Software. [n. d.]. Chocolately Software Docs | Security. Retrieved 2022-12-22 from https://docs.chocolatey.org/en-us/information/security

[15] Mihai Christodorescu, Johannes Kinder, Somesh Jha, Stefan Katzenbeisser, and Helmut Veith. 2005. *Malware Normalization*. Technical Report TR1539. Department of Computer Sciences, University of Wisconsin-Madison.

[16] Cisco Systems, Inc. [n. d.]. ClamAV: Creating signatures for ClamAV. Retrieved 2022-12-22 from https://docs.clamav.net/manual/Signatures.html

[17] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 1310–1320. https://proceedings.mlr.press/v97/cohen19c.html

[18] Gautam Das, Rudolf Fleischer, Leszek Gasieniec, Dimitris Gunopulos, and Juha Kärkkäinen. 1997. Episode matching. In *Combinatorial Pattern Matching*. Springer Berlin Heidelberg, Berlin, Heidelberg, 12–27.

[19] Luca Demetrio and Battista Biggio. 2021. secml-malware: Pentesting Windows Malware Classifiers with Adversarial EXEmples in Python. https://doi.org/10.48550/ARXIV.2104.12848

[20] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. 2019. Explaining Vulnerabilities of Deep Learning to Adversarial Malware Binaries. https://doi.org/10.48550/ARXIV.1901.03583

[21] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. 2021. Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware. *IEEE Transactions on Information Forensics and Security* 16 (2021), 3469–3478. https://doi.org/10.1109/TIFS.2021.3082330

[22] Luca Demetrio, Scott E. Coull, Battista Biggio, Giovanni Lagorio, Alessandro Armando, and Fabio Roli. 2021. Adversarial EXEmples: A Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection. *ACM Trans. Priv. Secur.* 24, 4, Article 27 (Sept. 2021). https://doi.org/10.1145/3473039

[23] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. 2019. Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection. *IEEE Transactions on Dependable and Secure Computing* 16, 4 (2019), 711–724. https://doi.org/10.1109/TDSC.2017.2700270

[24] Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. 2018. Training verified learners with learned verifiers. https://doi.org/10.48550/ARXIV.1805.10265

[25] Krishnamurthy (Dj) Dvijotham, Jamie Hayes, Borja Balle, Zico Kolter, Chongli Qin, András György, Kai Xiao, Sven Gowal, and Pushmeet Kohli. 2020. A Framework for Robustness Certification of Smoothed Classifiers using f-Divergences. In *8th International Conference on Learning Representations (ICLR)*. OpenReview.net. https://openreview.net/forum?id=SJlKrkSFPH

[26] 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 *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1625–1634. https://doi.org/10.1109/CVPR.2018.00175

[27] Fenil Fadadu, Anand Handa, Nitesh Kumar, and Sandeep Kumar Shukla. 2020. Evading API Call Sequence Based Malware Classifiers. In *Information and Communications Security*. Springer, Cham, 18–33. https://doi.org/10.1007/978-3-030-41579-2_2

[28] Marc Fischer, Maximilian Baader, and Martin Vechev. 2020. Certified Defense to Image Transformations via Randomized Smoothing. In *Advances in Neural Information Processing Systems (NeurIPS, Vol. 33)*. Curran Associates, Inc., 8404–8417.

[29] Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based Adversarial Examples for Text Classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 6174–6181. https://doi.org/10.18653/v1/2020.emnlp-main.498

[30] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations (ICLR)*. http://arxiv.org/abs/1412.6572

[31] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. 2018. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. https://doi.org/10.48550/ARXIV.1810.12715

[32] Zhongkai Hao, Chengyang Ying, Yinpeng Dong, Hang Su, Jian Song, and Jun Zhu. 2022. GSmooth: Certified Robustness against Semantic Transformations via Generalized Randomized Smoothing. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, 8465–8483. https://proceedings.mlr.press/v162/hao22c.html

[33] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. 2021. Natural Adversarial Examples. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 15262–15271. https://doi.org/10.1109/CVPR46437.2021.01501

[34] Weiwei Hu and Ying Tan. 2018. Black-Box Attacks against RNN Based Malware Detection Algorithms. In *The Workshops of the The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI Workshops)*. AAAI Press, 245–251. https://aaai.org/ocs/index.php/WS/AAAIW18/paper/view/16594

[35] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. 2011. Adversarial Machine Learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence (AISec)*. Association for Computing Machinery, New York, NY, USA, 43–58. https://doi.org/10.1145/2046684.2046692

[36] Íñigo Íncer Romeo, Michael Theodorides, Sadia Afroz, and David Wagner. 2018. Adversarially Robust Malware Detection Using Monotonic Classification. In *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics (IWSPA)*. Association for Computing Machinery, New York, NY, USA, 54–63. https://doi.org/10.1145/3180445.3180449

[37] Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified Robustness to Adversarial Word Substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 4129–4142. https://doi.org/10.18653/v1/D19-1423

[38] Haodi Jiang, Turki Turki, and Jason T. L. Wang. 2018. DLGraph: Malware Detection Using Deep Learning and Graph Embedding. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 1029–1033. https://doi.org/10.1109/ICMLA.2018.00168

[39] Kaspersky Lab. 2021. Machine Learning for Malware Detection. https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf

[40] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. 2018. Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables. In *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 533–537. https://doi.org/10.23919/EUSIPCO.2018.8553214

[41] J. Zico Kolter and Marcus A. Maloof. 2006. Learning to Detect and Classify Malicious Executables in the Wild. *Journal of Machine Learning Research* 7, 99 (2006), 2721–2744. http://jmlr.org/papers/v7/kolter06a.html

[42] Felix Kreuk, Assi Barak, Shir Aviv-Reuven, Moran Baruch, Benny Pinkas, and Joseph Keshet. 2018. Deceiving End-to-End Deep Learning Malware Detectors using Adversarial Examples. https://doi.org/10.48550/ARXIV.1802.04528

[43] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified Robustness to Adversarial Examples with Differential Privacy. In *2019 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 656–672. https://doi.org/10.1109/SP.2019.00044

[44] Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi Jaakkola. 2019. Tight Certificates of Adversarial Robustness for Randomly Smoothed Classifiers. In

Advances in Neural Information Processing Systems (NeurIPS, Vol. 32). Curran Associates, Inc., 4910–4921.

[45] Klas Leino, Zifan Wang, and Matt Fredrikson. 2021. Globally-Robust Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, 6212–6222. https://proceedings.mlr.press/v139/leino21a.html

[46] Alexander Levine and Soheil Feizi. 2020. Robustness Certificates for Sparse Adversarial Attacks by Randomized Ablation. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (2020), 4585–4593. https://doi.org/10.1609/aaai.v34i04.5888

[47] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. 2019. Certified Adversarial Robustness with Additive Noise. In *Advances in Neural Information Processing Systems (NeurIPS, Vol. 32)*. Curran Associates, Inc., 9459–9469.

[48] Linyi Li, Maurice Weber, Xiaojun Xu, Luka Rimanic, Bhavya Kailkhura, Tao Xie, Ce Zhang, and Bo Li. 2021. TSS: Transformation-Specific Smoothing for Robustness Certification. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, New York, NY, USA, 535–557. https://doi.org/10.1145/3460120.3485258

[49] Kaijun Liu, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, and Haifeng Liu. 2020. A Review of Android Malware Detection Approaches Based on Machine Learning. *IEEE Access* 8 (2020), 124579–124607. https://doi.org/10.1109/ACCESS.2020.3006143

[50] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards Robust Neural Networks via Random Self-ensemble. In *Computer Vision – ECCV 2018*. Springer, 381–397. https://doi.org/10.1007/978-3-030-01234-2_23

[51] Daniel Lowd and Christopher Meek. 2005. Good Word Attacks on Statistical Spam Filters.. In *Second Conference on Email and Anti-Spam (CEAS)*.

[52] Keane Lucas, Mahmood Sharif, Lujo Bauer, Michael K. Reiter, and Saurabh Shintre. 2021. Malware Makeover: Breaking ML-Based Static Analysis by Modifying Executable Bytes. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. Association for Computing Machinery, New York, NY, USA, 744–758. https://doi.org/10.1145/3433210.3453086

[53] Microsoft Corporation. 2022. PE Format. Retrieved 2022-09-14 from https://docs.microsoft.com/en-us/windows/win32/debug/pe-format

[54] Microsoft Defender Security Research Team. 2019. New machine learning model sifts through the good to unearth the bad in evasive malware. Retrieved Accessed: 2022-11-28 from https://www.microsoft.com/en-us/security/blog/2019/07/25/new-machine-learning-model-sifts-through-the-good-to-unearth-the-bad-in-evasive-malware/

[55] Matthew Mirman, Timon Gehr, and Martin Vechev. 2018. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 3578–3586. https://proceedings.mlr.press/v80/mirman18b.html

[56] National Security Agency. [n. d.]. Ghidra (version 10.1.5). Retrieved 2022-10-23 from https://www.nsa.gov/ghidra

[57] Dario Nisi, Mariano Graziano, Yanick Fratantonio, and Davide Balzarotti. 2021. Lost in the Loader: The Many Faces of the Windows PE File Format. In *Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. Association for Computing Machinery, New York, NY, USA, 177–192. https://doi.org/10.1145/3471621.3471848

[58] Raphael Olivier and Bhiksha Raj. 2021. Sequential Randomized Smoothing for Adversarially Robust Speech Recognition. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 6372–6386. https://doi.org/10.18653/v1/2021.emnlp-main.514

[59] 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 (S&P)*. IEEE, 582–597. https://doi.org/10.1109/SP.2016.41

[60] Daniel Park, Haidar Khan, and Bülent Yener. 2019. Generation & Evaluation of Adversarial Examples for Malware Obfuscation. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 1283–1290. https://doi.org/10.1109/ICMLA.2019.00210

[61] Frédéric Perriot. 2003. Defeating Polymorphism Through Code Optimization. In *Proceedings of the 2003 Virus Bulletin Conference (VB2003)*. Virus Bulletin Ltd., 142–159.

[62] Yong Qiao, Yuexiang Yang, Lin Ji, and Jie He. 2013. Analyzing Malware by Abstracting the Frequent Itemsets in API Call Sequences. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 265–270. https://doi.org/10.1109/TrustCom.2013.36

[63] Erwin Quiring, Lukas Pirch, Michael Reimsbach, Daniel Arp, and Konrad Rieck. 2020. Against All Odds: Winning the Defense Challenge in an Evasion Competition with Diversification. https://doi.org/10.48550/ARXIV.2010.09569

[64] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K. Nicholas. 2018. Malware Detection by Eating a Whole EXE. In *The Workshops of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI Workshops)*. AAAI Press, 268–276. https://aaai.org/ocs/index.php/WS/AAAIW18/paper/view/16422

[65] Edward Raff, William Fleshman, Richard Zak, Hyrum S. Anderson, Bobby Filar, and Mark McLean. 2021. Classifying Sequences of Extreme Length with Constant Memory Applied to Malware Detection. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 11 (2021), 9386–9394. https://doi.org/10.1609/aaai.v35i11.17131

[66] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Certified Defenses against Adversarial Examples. In *6th International Conference on Learning Representations (ICLR)*. OpenReview.net. https://openreview.net/forum?id=Bys4ob-Rb

[67] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1085–1097. https://doi.org/10.18653/v1/P19-1103

[68] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. 2008. Learning and Classification of Malware Behavior. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer Berlin Heidelberg, Berlin, Heidelberg, 108–125.

[69] 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 (ACSAC)*. Association for Computing Machinery, New York, NY, USA, 611–626. https://doi.org/10.1145/3427228.3427230

[70] 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 *Research in Attacks, Intrusions, and Defenses*. Springer, Cham, 490–510.

[71] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. 2019. Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers. In *Advances in Neural Information Processing Systems (NeurIPS, Vol. 32)*. Curran Associates, Inc., 11289–11300.

[72] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo. 2001. Data mining methods for detection of new malicious executables. In *2001 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 38–49. https://doi.org/10.1109/SECPRI.2001.924286

[73] Mahmood Sharif, Lujo Bauer, and Michael K. Reiter. 2018. On the Suitability of Lp-Norms for Creating and Preventing Adversarial Examples. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 1605–1613. https://doi.org/10.1109/CVPRW.2018.00211

[74] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, New York, NY, USA, 1528–1540. https://doi.org/10.1145/2976749.2978392

[75] Chocolatey Software. [n. d.]. Chocolatey Software. Retrieved 2022-10-11 from https://chocolatey.org/

[76] Wei Song, Xuezixiang Li, Sadia Afroz, Deepali Garg, Dmitry Kuznetsov, and Heng Yin. 2022. MAB-Malware: A Reinforcement Learning Framework for Blackbox Generation of Adversarial Malware. In *Proceedings of the 2022 ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. Association for Computing Machinery, New York, NY, USA, 990–1003. https://doi.org/10.1145/3488932.3497768

[77] Octavian Suciu, Scott E. Coull, and Jeffrey Johns. 2019. Exploring Adversarial Examples in Malware Detection. In *2019 IEEE Security and Privacy Workshops (S&PW)*. IEEE, 8–14. https://doi.org/10.1109/SPW.2019.00015

[78] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems (NeurIPS, Vol. 33)*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). Curran Associates, Inc., 1633–1645.

[79] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble Adversarial Training: Attacks and Defenses. In *6th International Conference on Learning Representations (ICLR)*. OpenReview.net. https://openreview.net/forum?id=rkZvSe-RZ

[80] Spark Tsao. 2019. Faster and More Accurate Malware Detection Through Predictive Machine Learning. Retrieved 2022-11-25 from https://www.trendmicro.com/vinfo/pl/security/news/security-technology/faster-and-more-accurate-malware-detection-through-predictive-machine-learning-correlating-static-and-behavioral-features

[81] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. 2018. Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS, Vol. 31)*. Curran Associates Inc., 6542–6551.

[82] R. Vinayakumar, Mamoun Alazab, K. P. Soman, Prabaharan Poornachandran, and Sitalakshmi Venkatraman. 2019. Robust Intelligent Malware Detection Using Deep Learning. *IEEE Access* 7 (2019), 46717–46738. https://doi.org/10.1109/ACCESS.2019.2906934

[83] VIPRE. [n. d.]. VIPRE Android Security. Retrieved 2022-12-22 from https://vipre.com/home/vipre-android-security/
[84] VirusShare.com. [n. d.]. VirusShare.com. Retrieved 2022-10-11 from https://virusshare.com/
[85] Yevgeniy Vorobeychik and Murat Kantarcioglu. 2018. *Adversarial Machine Learning*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00861ED1V01Y201806AIM039
[86] Robert A. Wagner and Michael J. Fischer. 1974. The String-to-String Correction Problem. *J. ACM* 21, 1 (Jan. 1974), 168–173. https://doi.org/10.1145/321796.321811
[87] Andrew Walenstein, Rachit Mathur, Mohamed R. Chouchane, and Arun Lakhotia. 2006. Normalizing Metamorphic Malware Using Term Rewriting. In *2006 Sixth IEEE International Workshop on Source Code Analysis and Manipulation*. IEEE, 75–84. https://doi.org/10.1109/SCAM.2006.20
[88] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 5276–5285. https://proceedings.mlr.press/v80/weng18a.html
[89] Eric Wong and Zico Kolter. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 5286–5295. https://proceedings.mlr.press/v80/wong18a.html
[90] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. 2018. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems (NeurIPS, Vol. 31)*. Curran Associates, Inc., 8410–8419.
[91] Greg Yang, Tony Duan, J. Edward Hu, Hadi Salman, Ilya Razenshteyn, and Jerry Li. 2020. Randomized Smoothing of All Shapes and Sizes. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 10693–10705. https://proceedings.mlr.press/v119/yang20c.html
[92] Mao Ye, Chengyue Gong, and Qiang Liu. 2020. SAFER: A Structure-free Approach for Certified Robustness to Adversarial Word Substitutions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 3465–3475. https://doi.org/10.18653/v1/2020.acl-main.317
[93] Yanfang Ye, Lifei Chen, Dingding Wang, Tao Li, Qingshan Jiang, and Min Zhao. 2008. SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging. *Journal in Computer Virology* 5, 4 (26 Nov. 2008), 283. https://doi.org/10.1007/s11416-008-0108-y
[94] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. 2020. Towards Stable and Efficient Training of Verifiably Robust Neural Networks. In *8th International Conference on Learning Representations (ICLR)*. OpenReview.net. https://openreview.net/forum?id=Skxuk1rFwB
[95] Zhaoqi Zhang, Panpan Qi, and Wei Wang. 2020. Dynamic Malware Analysis with Feature Engineering and Feature Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 1 (2020), 1210–1217. https://doi.org/10.1609/aaai.v34i01.5474

A BRUTE-FORCE EDIT DISTANCE CERTIFICATION

In this appendix, we show that an edit distance certification mechanism based on brute-force search is computationally infeasible. Suppose we are interested in issuing an edit distance certificate at radius r for a malware detector f at input file $\mathbf{x}$. Recall from Definition 2.2 that in order to issue a certificate, we must show there exists no adversarial file $\mathbf{x}'$ within the edit distance neighborhood $\mathcal{N}_r(\mathbf{x})$ that would change f's prediction. This problem can theoretically be solved in a brute-force manner, by querying f for all inputs in $\mathcal{N}_r(\mathbf{x})$. In the best case, this would take time linear in $|\mathcal{N}_r(\mathbf{x})|$, assuming f responds to queries in constant time. However the following lower bound [13], shows that the size of the edit distance neighborhood is too large even in the best case:

$$|\mathcal{N}_r(\mathbf{x})| \geq \sum_{i=0}^{r} 255^i \sum_{j=i-r}^{r} \binom{|\mathbf{x}|+j}{i}.$$

For example, brute-force certification for a small file of size $|\mathbf{x}| = 10\text{KB}$ and certificate radius $r = 10$ would require $\mathcal{N}_r(\mathbf{x}) \geq 10^{58}$ queries to f. In contrast, our probabilistic certification mechanism (Algorithm 1) makes $n_{\text{pred}} + n_{\text{bound}}$ queries to f, and we can provide high probability guarantees when the number of queries is of order 10^3 or 10^4.

B PROOFS FOR SECTION 3.2

In this appendix, we provide proofs of the theoretical results stated in Section 3.2.

B.1 Proof of Lemma 3.3

Let $r_S \colon S \to \{1, \ldots, |S|\}$ be a bijection that returns the *rank* of an element in an ordered set S. Let $\dot{r}_S \colon 2^S \to 2^{\{1,\ldots,|S|\}}$ be an elementwise extension of r_S that returns a *set of ranks* for an ordered set of elements—i.e., $\dot{r}_S(U) = \{ r_S(i) : i \in U \}$ for $U \subseteq S$. We claim $m(\epsilon) = \dot{r}_{\bar{\epsilon}^\star}^{-1}(\dot{r}_{\epsilon^\star}(\epsilon))$ is a bijection that satisfies the required property.

To prove the claim, we note that m is a bijection from $2^{\epsilon^\star}$ to $2^{\bar{\epsilon}^\star}$ since it is a composition of bijections $\dot{r}_{\epsilon^\star} : 2^{\epsilon^\star} \to 2^{\{1,\ldots,l\}}$ and $\dot{r}_{\bar{\epsilon}^\star}^{-1} : 2^{\{1,\ldots,l\}} \to 2^{\bar{\epsilon}^\star}$ where $l = |\epsilon^\star| = |\bar{\epsilon}^\star|$. Next, we observe that $\dot{r}_{\epsilon^\star}(\epsilon)$ relabels indices in ϵ so they have the same effect when applied to $\mathbf{z}^\star$ as ϵ on $\mathbf{x}$ (this also holds for $\dot{r}_{\bar{\epsilon}^\star}$ and $\bar{\epsilon}$). Thus

$$\begin{aligned}
\operatorname{apply}(\mathbf{x}, \epsilon) &= \operatorname{apply}(\mathbf{z}^\star, \dot{r}_{\epsilon^\star}(\epsilon)) \\
&= \operatorname{apply}(\mathbf{z}^\star, \dot{r}_{\bar{\epsilon}^\star}(\dot{r}_{\bar{\epsilon}}^{-1}(\dot{r}_{\epsilon^\star}(\epsilon)))) \\
&= \operatorname{apply}(\bar{\mathbf{x}}, m(\epsilon))
\end{aligned}$$

as required. To prove the final statement, we use (4), (5) and (9) to write

$$\begin{aligned}
\frac{s(\epsilon, \mathbf{x}; f_{\mathrm{b}})}{s(\bar{\epsilon}, \bar{\mathbf{x}}; f_{\mathrm{b}})} &= \frac{\mathbf{1}_{f_{\mathrm{b}}(\operatorname{apply}(\mathbf{x},\epsilon))=y} p_{\mathrm{del}}^{|\mathbf{x}|-|\epsilon|}(1-p_{\mathrm{del}})^{|\epsilon|}}{\mathbf{1}_{f_{\mathrm{b}}(\operatorname{apply}(\bar{\mathbf{x}},\bar{\epsilon}))=y} p_{\mathrm{del}}^{|\bar{\mathbf{x}}|-|\bar{\epsilon}|}(1-p_{\mathrm{del}})^{|\bar{\epsilon}|}} \\
&= \frac{p_{\mathrm{del}}^{|\mathbf{x}|-|\mathbf{z}|}(1-p_{\mathrm{del}})^{|\mathbf{z}|}\mathbf{1}_{f_{\mathrm{b}}(\mathbf{z})=y}}{p_{\mathrm{del}}^{|\bar{\mathbf{x}}|-|\mathbf{z}|}(1-p_{\mathrm{del}})^{|\mathbf{z}|}\mathbf{1}_{f_{\mathrm{b}}(\mathbf{z})=y}} \\
&= p_{\mathrm{del}}^{|\mathbf{x}|-|\bar{\mathbf{x}}|},
\end{aligned}$$

where the second last line follows from the fact that $\operatorname{apply}(\mathbf{x}, \epsilon) = \operatorname{apply}(\bar{\mathbf{x}}, \bar{\epsilon}) = \mathbf{z}$.

B.2 Proof of Theorem 3.4

Let $\epsilon^\star$ and $\bar{\epsilon}^\star$ be defined as in Lemma 3.3. We derive an upper bound on the sum over $\bar{\epsilon} \in 2^{\bar{\epsilon}^\star}$ that appears in (10). Observe that

$$\begin{aligned}
\sum_{\bar{\epsilon} \notin 2^{\bar{\epsilon}^\star}} s(\bar{\epsilon}, \bar{\mathbf{x}}; f_{\mathrm{b}}) &\leq \sum_{\bar{\epsilon} \notin 2^{\bar{\epsilon}^\star}} \Pr\left[G(\bar{\mathbf{x}}) = \bar{\epsilon}\right] \\
&= 1 - \sum_{\bar{\epsilon} \in 2^{\bar{\epsilon}^\star}} \Pr\left[G(\bar{\mathbf{x}}) = \bar{\epsilon}\right] \\
&= 1 - p_{\mathrm{del}}^{|\bar{\mathbf{x}}|-|\bar{\epsilon}^\star|} \sum_{|\bar{\epsilon}|=0}^{|\bar{\epsilon}^\star|} \binom{|\bar{\epsilon}^\star|}{|\bar{\epsilon}|} p_{\mathrm{del}}^{|\bar{\epsilon}^\star|-|\bar{\epsilon}|}(1-p_{\mathrm{del}})^{|\bar{\epsilon}|} \\
&= 1 - p_{\mathrm{del}}^{|\bar{\mathbf{x}}|-|\bar{\epsilon}^\star|}, \qquad (17)
\end{aligned}$$

where the first line follows from the inequality $\mathbf{1}_{f_{\mathrm{b}}(\operatorname{apply}(\bar{\mathbf{x}},\bar{\epsilon})=y)} \leq 1$; the second line follows from the law of total probability; the third line follows by constraining the indices $\{1, \ldots, |\bar{\mathbf{x}}|\} \backslash \epsilon^\star$ to be deleted; and the last line follows from the normalization of the binomial distribution. Putting (17) and $\sum_{\epsilon \in 2^{\epsilon^\star}} s(\epsilon, \mathbf{x}; f_b) \geq 0$ in (10) then gives the required result.

B.3 Proof of Corollary 3.5

Since the length of $\bar{\mathbf{x}}$ can only be changed by inserting or deleting bytes in $\mathbf{x}$, we have $|\bar{\mathbf{x}}| - |\mathbf{x}| = n_{\text{ins}} - n_{\text{del}}$. We also observe that $\mathbf{x}$ can be transformed into $\bar{\mathbf{x}}$ using the longest common subsequence $\mathbf{z}^\star$ as an intermediary. Specifically, $n_{\text{del}} + n_{\text{sub}}$ bytes can be deleted from $\mathbf{x}$ to yield $\mathbf{z}^\star$, then $n_{\text{ins}} + n_{\text{sub}}$ bytes can be inserted in $\mathbf{z}^\star$ to yield $\bar{\mathbf{x}}$. This implies $|\bar{\mathbf{x}}| - |\mathbf{z}^\star| = n_{\text{ins}} + n_{\text{sub}}$. Substituting the above identities in (11) gives the required result.

B.4 Proof of Theorem 3.6

Eliminating n_{sub} from (13) using the constraint $n_{\text{sub}} = r - n_{\text{del}} - n_{\text{ins}}$, we obtain a minimization problem in two variables:

$$\begin{aligned} \min_{n_{\text{ins}}, n_{\text{del}} \in \mathbb{N}_0} \quad & \psi(n_{\text{ins}}, n_{\text{del}}) \\ \text{s.t.} \quad & 0 \leq n_{\text{ins}} + n_{\text{del}} \leq r \end{aligned}$$

where $\psi(n_{\text{ins}}, n_{\text{del}}) = p_{\text{del}}^{n_{\text{del}} - n_{\text{ins}}} \left(\mu_y - 1 + p_{\text{del}}^{r - n_{\text{del}}}\right)$. Observe that ψ is monotonically increasing in n_{ins} and n_{del}:

$$\begin{aligned} \frac{\psi(n_{\text{ins}} + 1, n_{\text{del}})}{\psi(n_{\text{ins}}, n_{\text{del}})} &= \frac{1}{p_{\text{del}}} \geq 1 \\ \frac{\psi(n_{\text{ins}}, n_{\text{del}} + 1)}{\psi(n_{\text{ins}}, n_{\text{del}})} &= \frac{(\mu_y - 1) p_{\text{del}}^{n_{\text{del}} + 1} + p_{\text{del}}^r}{(\mu_y - 1) p_{\text{del}}^{n_{\text{del}}} + p_{\text{del}}^r} \geq 1, \end{aligned}$$

where the second inequality follows since we only consider r and μ_y such that the numerator and denominator are positive. Thus the minimizer is $(n_{\text{ins}}^\star, n_{\text{del}}^\star, n_{\text{sub}}^\star) = (0, 0, r)$ and we find $\rho(\bar{\mathbf{x}}; \mu_y) = \mu_y - 1 + p_{\text{del}}^r$. The expression for the largest certified radius follows by solving $\rho(\bar{\mathbf{x}}; \mu_y) > \eta_y$ for non-negative integer r.

B.5 Proof of Corollary 3.7

Recall that Corollary 3.5 gives the following lower bound on the smoothed detector's score at $\mathbf{x}$:

$$\text{lb}_y(\mathbf{x}; \bar{\mathbf{x}}, \mu_y) = p_{\text{del}}^{n_{\text{del}} - n_{\text{ins}}} \left(\mu_y - 1 + p_{\text{del}}^{n_{\text{sub}} + n_{\text{ins}}}\right).$$

Observe that we can replace μ_y by a lower bound $\underline{\mu}_y$ that holds with probability $1 - \alpha$ (as is done in lines 3–4 of Algorithm 1) and obtain a looser lower bound $\text{lb}_y(\mathbf{x}; \bar{\mathbf{x}}, \underline{\mu}_y) \leq \text{lb}_y(\mathbf{x}; \bar{\mathbf{x}}, \mu_y)$ that holds with probability $1 - \alpha$. Crucially, this looser lower bound has the same functional form, so all results depending on Corollary 3.5, namely Theorem 3.6 and Table 1, continue to hold albeit with probability $1 - \alpha$.

C ADDITIONAL RESULTS FOR EFFECTIVENESS OF CERTIFICATION

In this appendix, we present supplementary results for Section 4.2, covering accuracy and robustness guarantees of our method (RS-Del).

Table 7 reports clean accuracy for RS-Del and the non-certified NS baseline. It also reports abstention rates for RS-Del, the median certified radius (CR), and the median certified radius normalized by file size (NCR). We find that clean accuracy for Sleipnir2 follows similar trends as certified accuracy: it is relatively stable as the deletion probability increases to $p_{\text{del}} = 99.5\%$, but suffers a significant drop at $p_{\text{del}} = 99.9\%$. We observe minimal differences in the results for instruction (Insn) and byte-level (Byte) smoothing, but note that the effective CR is larger for instruction smoothing, since each token may contain several bytes.

Table 7: Clean accuracy and robustness metrics for RS-Del as a function of the dataset (Sleipnir2 and VTFeed), deletion probability p_{del} and elementary token (bytes and instructions). All metrics are computed on the test set. Here "abstn. rate" refers to the fraction of test instances for which RS-Del abstains (line 6 in Algorithm 1), and "UB" refers to an upper bound on the median CR for a best case smoothed detector (based on Table 1 with $\mu_y = 1$). A good tradeoff is achieved when $p_{\text{del}} = 99.5\%$ for both the byte- and instruction-level threat models (highlighted in bold face below).

Detector	Params *token*, p_{del}	Clean accuracy (Abstn. rate) %	Median CR (UB)	Median NCR %
Sleipnir2				
NS		98.9 –	– –	–
RS-Del	Byte, 90%	97.1 (0.2)	6 (6)	0.0023
	Byte, 95%	97.8 (0.0)	13 (13)	0.0052
	Byte, 97%	97.4 (0.1)	22 (22)	0.0093
	Byte, 99%	98.1 (0.1)	68 (68)	0.0262
	Byte, 99.5%	**96.5 (0.2)**	**137 (138)**	**0.0555**
	Byte, 99.9%	83.7 (3.4)	688 (692)	0.2269
	Insn, 90%	97.9 (0.1)	6 (6)	0.0026
	Insn, 95%	97.8 (0.1)	13 (13)	0.0056
	Insn, 97%	98.3 (0.0)	22 (22)	0.0095
	Insn, 99%	97.6 (0.1)	68 (68)	0.0292
	Insn, 99.5%	**96.8 (0.2)**	**137 (138)**	**0.0589**
	Insn, 99.9%	86.1 (0.2)	689 (692)	0.2982
VTFeed				
NS		98.9 –	– –	–
RS-Del	Byte, 97%	92.1 (0.9)	22 (22)	0.033

Figure 4 plots the certified accuracy of RS-Del on the Sleipnir2 dataset using instruction-level Levenshtein distance. It is an analogue of Figure 2, which plots certified accuracy for byte-level Levenshtein distance. We observe similar trends in both plots and refer the reader to the discussion in Section 4.2.1. We note that the instruction-level variant of RS-Del arguably provides stronger guarantees, since the effective radius for instruction-level Levenshtein distance is larger than for byte-level Levenshtein distance.

Figure 5 plots the certified true positive rate (TPR) and true negative rate (TNR) of RS-Del on the Sleipnir2 dataset for several values of the decision threshold η_1. The certified TPR and TNR can be interpreted as class-specific analogues of the certified accuracy. Concretely, the certified TPR (TNR) at radius r is the fraction of malicious (benign) instances in the test set for which the malware detector's prediction is correct *and* certified robust at radius r. The certified TPR and TNR jointly measure accuracy and robustness and complement the metrics reported in Table 3. Looking at Figure 5,

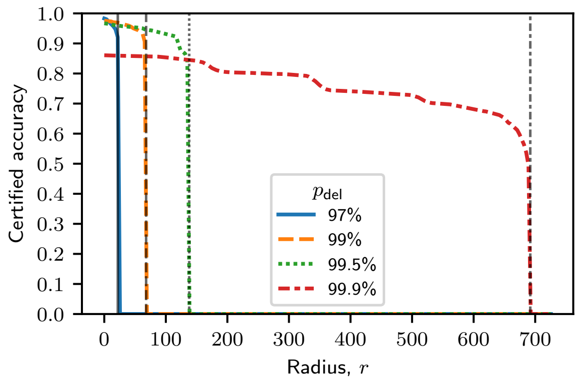

Figure 4: Certified accuracy of RS-Del as a function of the certificate radius (horizontal axis) and token deletion probability p_{del} (colored line styles). The results are plotted for the Sleipnir2 test set under the instruction-level Levenshtein distance threat model (with $O = \{\text{del}, \text{ins}, \text{sub}\}$). It is apparent that p_{del} controls a robustness/accuracy tradeoff. The grey vertical lines represent the best achievable certified radius for RS-Del (setting $\mu_y = 1$ in Table 1). Note that in this setting, a non-smoothed, non-certified detector (NS) achieves a clean accuracy of 98%.

we see that the certified TNR curves drop more rapidly to zero than the certified TPR curves as η_1 decreases. This is in line with comments made in Section 4.2—that decreasing η_1 sacrifices the certified radii of benign instances to increase the certified radii of malicious instances. We note that the curves for $\eta_1 = \frac{1}{2}$ correspond to the same setting as the certified accuracy curve in Figure 2 (with $p_{\text{del}} = 99.5\%$).

Table 8 provides raw certified accuracy data for two of the cases plotted in Figure 3 (RS-Del at $p_{\text{del}} = 99.5\%$ and RS-Abn at $p_{\text{ab}} = 99.5\%$). We find that RS-Del outperforms RS-Abn for all radii up to 138, which is the largest possible certified radius achievable for the Hamming distance threat model using our method (see Table 1). This is notable given our method is not specifically designed for the Hamming distance threat model.

D EFFICIENCY OF RANDOMIZED SMOOTHING

In this appendix, we discuss the training and computational efficiency of RS-Del and provide comparisons with RS-Abn.

Computational efficiency. Table 9 provides wall clock times for training and prediction of smoothed detectors. The prediction times are further decomposed into subtasks: input randomization and prediction for the base detector. All times are recorded on a desktop PC fitted with an AMD Ryzen 7 5800X CPU and an NVIDIA RTX3090 GPU. We execute training and prediction for the base MalConv model on the GPU, and input randomization on the CPU. We use a single PyTorch process, noting that times may be improved by enabling parallel processing for input randomization.

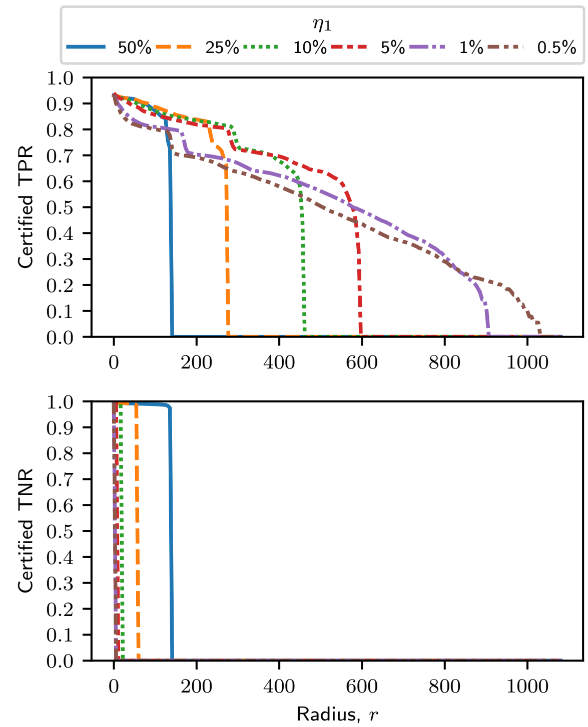

Figure 5: Certified true positive rate (TPR) and true negative rate (TNR) of RS-Del as a function of the certificate radius r (horizontal axis) and the decision threshold η_1 (colored line styles). The results are plotted for the Sleipnir2 test set with $p_{\text{del}} = 99.5\%$ under the byte-level Levenshtein distance threat model (with $O = \{\text{del}, \text{ins}, \text{sub}\}$). It is apparent that η_1 controls a tradeoff in the certified radius between the malicious (measured by TPR) and benign (measured by TNR) classes. Note that in this setting, a non-smoothed, non-certified detector (NS) achieves a clean TPR and TNR of 98.2% and 99.5% respectively.

We now make some observations about the results. First, we note that training is an order of magnitude faster for RS-Del compared with RS-Abn. This is due to the deletion randomization scheme we propose for RS-Del, which drastically reduces the length of inputs, thereby reducing the time taken to perform forward and backward passes for the base detector. On the contrary, the ablation randomization scheme for RS-Abn does not alter the length of inputs, so it does not have a performance advantage in this respect. Second, we note that there is no significant difference in the prediction time for the two detectors. While the time taken to pass the randomized inputs through the base detector is an order of magnitude faster for RS-Del, it does not have an impact on the total prediction time, as input randomization dominates.

Training efficiency. Training curves for the base MalConv detectors used in RS-Del and RS-Abn are provided in Figure 6 for the Sleipnir2 dataset. RS-Del is trained using stochastic gradient

Table 8: Raw certified accuracy data used in Figure 3. Here we provide data for RS-Del with $p_{\text{del}} = 99.5\%$ (our method) and RS-Abn with $p_{\text{ab}} = 99.5\%$ [46]. Note that the certificates are for the Hamming distance threat model.

	Certified accuracy (%)	
Radius	RS-Del	RS-Abn
110	92.33	82.26
112	92.19	82.01
114	92.07	81.90
116	91.89	81.67
118	91.79	81.39
120	91.68	81.17
122	91.59	80.82
124	91.53	79.96
126	91.35	78.70
128	91.09	78.29
130	88.43	77.86
132	86.78	77.13
134	86.39	75.32
136	85.03	68.02
138	–	25.84
140	–	0.09
142	–	0.03
144	–	0.01

Table 9: Comparison of runtime efficiency for RS-Del (our method) and RS-Abn [46]. The first column of wall times measures the time taken to train the base detector MalConv for one epoch on Sleipnir2. The second and third columns of wall times decompose the time to make a prediction for the smoothed detector for a 1MB input file. The second column measures the time taken to apply the randomization scheme $n_{\text{pred}} = 1000$ times and the third column measures the time taken pass the randomized inputs through the base detector.

		Wall time (s)		
			Predict	
Detector	Parameters	Train 1 epoch	Randomize input	Base predict
RS-Del	$p_{\text{del}} = 0.9$, Byte	354	10.42	0.070
RS-Del	$p_{\text{del}} = 0.9$, Insn	494	20.16	0.068
RS-Abn [46]	$p_{\text{ab}} = 0.9$	1692	15.29	0.352
RS-Del	$p_{\text{del}} = 0.99$, Byte	329	8.79	0.043
RS-Del	$p_{\text{del}} = 0.99$, Insn	544	18.62	0.043
RS-Abn [46]	$p_{\text{ab}} = 0.99$	1788	15.60	0.352

descent following standard parameters settings for MalConv [64]. Due to convergence issues for RS-Abn, we adapted training to incorporate gradient clipping when updating the embedding layer. This addresses imbalance in the gradients arising from the dominance of masked (ablated) values in the randomized inputs. However, even with this fix, we observe slower convergence to a higher loss value for RS-Abn than for RS-Del.

Combining the results of Table 9 and Figure 6, we conclude that RS-Abn beats RS-Abn in terms of training efficiency as it requires both fewer epochs to converge and takes less time per epoch.

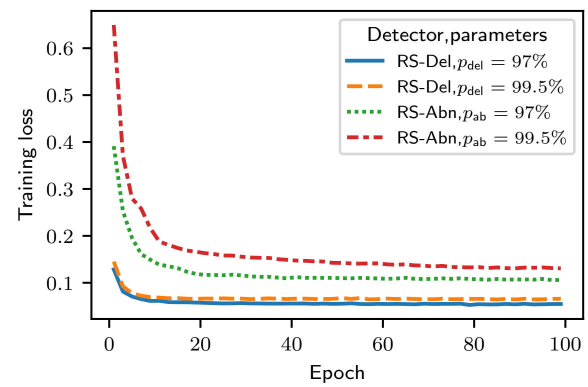

Figure 6: Training curves for RS-Del (our method) using byte-level deletion and RS-Abn [46] for the Sleipnir2 dataset.

E PARAMETER SETTINGS FOR MALCONV

In this appendix, we specify the parameter settings and training procedure for MalConv, which is used as a standalone malware detector in NS, and as a base malware detector for RS-Del and RS-Abn. Table 10 summarizes our setup, which is consistent across all three detectors except where specified. We follow the authors of MalConv [64] when setting parameters for the model and the optimizer, however we set a larger maximum input size of 2MiB to accommodate larger files without clipping. Due to differences in available GPU memory for the Sleipnir2 and VTFeed experiments, we use a larger batch size for VTFeed than for Sleipnir2. We also set a higher limit on the maximum number of epochs for VTFeed, as it is a larger dataset, although the NS and RS-Del detectors converge within 50 epochs for both datasets. To stabilize training for the randomized smoothed malware detectors (RS-Del and RS-Abn), we modify the randomization schemes during *training only* to ensure at least 500 raw bytes are preserved. This may limit the number of deletions for RS-Del and the number of ablated (masked) bytes for RS-Abn. For RS-Abn, we clip the gradients for the embedding layer to improve convergence (see Appendix D).

Table 10: Parameter settings for the MalConv model, optimizer and training procedure. The parameter settings are consistent across all detectors (NS, RS-Del, RS-Abn) except where specified.

MalConv hyperparameters

Max input size	2097152
Embedding size	8
Window size	500
Channels	128

Optimizer

Optimizer	`torch.optim.SGD`
Learning rate	0.01
Momentum	0.9
Weight decay	0.001

Training

Batch size	24 (Sleipnir2), 32 (VTFeed)
Max. epoch	50 (Sleipnir2), 100 (VTFeed)
Min. preserved bytes	500 (RS-Del, RS-Abn), NA (NS)
Embedding gradient clipping	0.5 (RS-Abn), ∞ (RS-Del, NS)
Early stopping	If validation loss does not improve after 10 epochs