Stolen Risks of Models with Security Properties

Yue Qin
Indiana University Bloomington
Bloomington, Indiana, USA
qinyue@iu.edu

Zhuoqun Fu
Tsinghua University
Beijing, China
fzq20@mails.tsinghua.edu.cn

Chuyun Deng
Tsinghua University
Beijing, China
dengcy20@mails.tsinghua.edu.cn

Xiaojing Liao
Indiana University Bloomington
Bloomington, Indiana, USA
xliao@indiana.edu

Jia Zhang
Tsinghua University
Beijing, China
zhangjia@cernet.edu.cn

Haixin Duan
Tsinghua University&Zhongguancun Laboratory
Beijing, China
duanhx@tsinghua.edu.cn

ABSTRACT

Verifiable robust machine learning, as a new trend of ML security defense, enforces security properties (e.g., Lipschitzness, Monotonicity) on machine learning models and achieves satisfying accuracy-security trade-off. Such security properties identify a series of evasion strategies of ML security attackers and specify logical constraints on their effects on a classifier (e.g., the classifier is monotonically increasing along some feature dimensions). However, little has been done so far to understand the side effect of those security properties on the model privacy.

In this paper, we aim at better understanding the privacy impacts on security properties of robust ML models. Particularly, we report the first measurement study to investigate the model stolen risks of robust models satisfying four security properties (i.e., LocalInvariance, Lipschitzness, SmallNeighborhood, and Monotonicity). Our findings bring to light the factors that influence model stealing attacks and defense performance on models trained with security properties. In addition, to train an ML model satisfying goals in accuracy, security, and privacy, we propose a novel technique, called BoundaryFuzz, which introduces a privacy property into verifiable robust training frameworks to defend against model stealing attacks on robust models. Experimental results demonstrate the defense effectiveness of BoundaryFuzz.

CCS CONCEPTS

• **Security and privacy**;

KEYWORDS

Adversarial Machine Learning; Model Stealing Attacks and Defenses; Security Properties

ACM Reference Format:
Yue Qin, Zhuoqun Fu, Chuyun Deng, Xiaojing Liao, Jia Zhang, and Haixin Duan. 2023. Stolen Risks of Models with Security Properties . In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 24 pages. https://doi.org/10.1145/3576915.3616653

1 INTRODUCTION

Recent studies have revealed that the deployment and success of Machine Learning (ML) models can be affected by a broad range of attacks that prompt serious security and privacy risks [5, 7, 8, 14, 15, 30, 32, 36, 39, 40, 43, 43, 48, 49]. In the security domain, evasion attacks [5, 14, 39, 40] associated with adversarial examples [14] *mislead* the target models by adding imperceptible perturbations to the inputs. Meanwhile, in the privacy domain, inference attacks [7, 15, 30, 32, 36, 42, 43, 49] allow adversaries to *infer* information about the training data, the parameters, or the decision boundary of a target model. To mitigate the security risks of ML models, the verifiable robust ML [10, 13, 19, 33, 33, 51–53] stands for a new trend of ML security defense. In verifiable robust ML models, the Security Property (SP) is identified to represent a class of evasion strategies that might be available to an adversary and specifies a requirement on their effect on the classifier [10]. As an instance of the SP, the Monotonicity property requires that the classifier is monotonically increasing along some feature dimensions [10]. In the context of health insurance premium prediction, the premium cost is designed to be monotonically increasing along with BMI value or a habit of smoking, as those features imply a worse health status of the insured person. Designed as logical constraints in multiple forms, SP is more flexible and expressive than traditional defenses [31, 33, 33, 41, 44, 51] and can meet various security needs of different applications [10, 13]. However, though SP has demonstrated its success to secure ML models with satisfying accuracy-security trade-off [10, 13], its side effect on the privacy domain is unclear and has not been studied.

In this paper, we investigate privacy impacts on four security properties (i.e., LocalInvariance, Lipschitzness, SmallNeighborhood and Monotonicity) of robust ML models, with a focus on Model Stealing (MS) attacks, where the attacker aims to train a substitute model with similar functionality to the target model [20, 35, 57]. MS attacks arouse severe privacy risks. For instance, stealing a trained model inherently constitutes intellectual property theft as it is often difficult to train an advanced ML model due to the lack of data or computing resources [35]. In the meantime, the success of MS attacks can break the black-box limit of the target model's query interface and thus assists several other ML privacy attacks such as membership inference attack [29, 43] and attribute inference attack [32] that leak sensitive information of individuals. Following [10], our study focuses on five real-world datasets of different scale used in security applications. These datasets involve classification tasks that, when attacked by adversarial evasions, may result in financial loss or sensitive information leakage.

Particularly, we systematically evaluate privacy impacts on four security properties by measuring the success of three typical MS attacks (in five variants) and two state-of-the-art MS defenses on robust models[1]. In our study, we reveal that all robust models trained with the four security properties are more vulnerable to MS attacks than their naturally undefended training (i.e., natural models). This is because a security property of a target model can benefit an MS attacker. More specifically, in MS attacks, the adversary queries the target model to label a dataset (i.e., the substitute training set) and trains a substitute model that approximates the target model's behavior. The main challenge lies in two aspects: 1) the attacker has little training data of the target model, and 2) the query budget (i.e., opportunities to query the target model's prediction interface) is limited. However, a model trained with SPs results in more stable and smooth predictions [10, 13], which enables higher-quality predictions on noisy data inputs. This enhances the querying effectiveness and facilitates the success of MS attacks. Additionally, SP constrains the model's output according to the condition of the model's input, which can further leak information about the target model's decision boundary. Similarly, we found that MS defenses are less effective on robust models. In MS defenses, the most effective methods protect the confidentiality of the model's decision boundary by selectively adding perturbations to the predictions of suspicious inputs [21, 37]. The accuracy-privacy trade-off depends on how well the adversary distinguishes suspicious inputs from benign ones. However, SP reduces the prediction gap between benign and suspicious samples, making the defense against MS more difficult.

Given the above understandings, we further propose MaskedThief, an optimized, semi-supervised MS attack against robust models. In particular, MaskedThief incorporates current MS attacks with an inference strategy. Only a few samples are sent to query the target model, while the predictions on the subsequent samples are inferred based on the predictions of its nearest neighbors. The inference uncertainty can be quantified using Hoeffding's Inequality [16] applied to the SP of the target model. The attacker can thereby determine whether to adopt the inference to label the synthesized sample or to query the target model instead. With such a strategy, only inputs with high uncertainty (i.e., information relative to the decision boundary) are sent to query the target model. This enhances the query effectiveness and enlarges the actual size of the substitute training set under the same query budget. Experimental results show that the inference strategy of MaskedThief improves all evaluated MS attacks.

To mitigate such privacy risk on models trained with security properties, we design a privacy property, BoundaryFuzz, which enforces the model predictions on inputs near the decision boundary to be uncertain. Compared with other state-of-the-art MS defenses, our method has an overwhelming advantage in defending robust models against MS attacks for it is compatible with existing security properties. Perturbation-based defenses, though achieving accuracy-privacy trade-off, can inevitably undermine the security properties of robust models. This is because model predictions are deteriorated by the perturbation after the robust training process which enforces SP. However, our method can fit well into the robust training process to be enforced along with security properties. Experimental results demonstrate the defense effectiveness of BoundaryFuzz, and it results in little performance drop in the classification accuracy as well as the constraint accuracy of SP. To the best of our knowledge, this is the first defense for ML models to satisfy goals in accuracy, security, and privacy altogether.

[1]Robust models and models trained with security properties were used interchangeably in this paper

Contributions. We summarize the contributions as follows:
- We systematically evaluate the privacy impact on four security properties (i.e., LocalInvariance, Lipschitzness, SmallNeighborhood and Monotonicity) of robust ML models, and provide in-depth analysis. Our results reveal that robust models suffer from larger privacy risks, in terms of model stealing, than natural models;
- We propose MaskedThief, an optimized MS attack specific to robust models. Our approach improves upon existing MS attacks by leveraging information in SPs;
- We design BoundaryFuzz, a privacy property that is compatible with security properties for ML models. A robust model equipped with BoundaryFuzz achieves balanced trade-offs between accuracy, security, and privacy. Our code, data, and full-version paper with Appendix are available at [3].

2 BACKGROUND

Security Properties of Robust Models. For a distribution D, a set $\mathbb{A}$ and a security property ϕ, a *verifiable adversarial defense system* trains the parameters θ of an ML model F_θ towards

$$\underset{\theta}{\operatorname{argmax}} \Pr_{\boldsymbol{x}\sim D}[\forall \boldsymbol{z} \in \mathbb{A}.\, \phi(\boldsymbol{x}, \boldsymbol{z}, \theta)]$$

s.t. the probability of F_θ satisfying the *security property* (SP) $\phi(\boldsymbol{x}, \boldsymbol{z}, \theta)$ for $\boldsymbol{z}$ is as large as possible. In this objective, $\boldsymbol{x}$ describes one or more independent data set samples.

In this paper, we consider two state-of-the-art verifiable adversarial defenses, DL2 [13], and LogicEnsemble [10]. These robust training frameworks enforce SPs by translating the property into derivative loss functions and optimizing the objective using gradient-based methods [31, 41]. In particular, DL2 demonstrates its success in training SPs where the inputs are exactly from the training data or are closed to them, a.k.a. local SPs, such as LocalInvariance, Lipschitzness, Segmentation, as elaborated below). LogicEnsemble shows effectiveness on training security properties, whose inputs are not constrained to have a similar distribution to the training data, a.k.a., global SPs, such as Monotonicity, and SmallNeighohood. In this paper, we consider four representative security properties and adopt their original definition and implementation in DL2 and LogicEnsemble. Below we elaborate on the formal definition and the security implication of these properties.

Given a target model $F_\theta : \mathcal{X} \to \mathcal{Y}$ and a sample $(\boldsymbol{x}, y)$, where $\boldsymbol{x} \in \mathcal{X}$ is the input and $y \in \mathcal{Y}$ is the true label, an (ℓ_p, ϵ)*-adversary* generates a perturbed input $\tilde{\boldsymbol{x}} \in \mathcal{B}_{p,\epsilon}(\boldsymbol{x})$ within the perturbation budget of ϵ under ℓ_p-norm distance: $\mathcal{B}_{p,\epsilon}(\boldsymbol{x}) = \{\tilde{\boldsymbol{x}} : ||\boldsymbol{x} - \tilde{\boldsymbol{x}}||_p < \epsilon\}$, where $||\boldsymbol{x}||_p = \left(\sum_{i=1}^{d} |x_i|^p\right)^{\frac{1}{p}}$. Against such an adversary, a robust model will satisfy the following security properties:

Property 1 (LocalInvariance)**.** This property requires that for any input $\boldsymbol{x}$ with a classification result y, inputs in its ϵ neighborhood have a high probability to be predicted as y:

$$\forall \boldsymbol{x} \in \mathcal{X}_{\text{tr}},\ \tilde{\boldsymbol{x}} \in \mathcal{B}_{p,\epsilon}(\boldsymbol{x}), \quad F_\theta(\tilde{\boldsymbol{x}})_y > 1 - \delta \tag{LocalInvariance}$$

This is also the general form of certified adversarial robustness [6] and is similar to the constraints in many previous studies [33, 33, 51]. The training-set-specific variant of LocalInvariance requires that the KL divergence of two closed random inputs from the dataset is small:

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}_{\text{tr}}, ||\boldsymbol{x} - \boldsymbol{x}'||_2 < \epsilon_1 \Rightarrow \text{KL}[F_\theta(\boldsymbol{x})||F_\theta(\boldsymbol{x}')] < \epsilon_2 \tag{LocalInvariance^T}$$

Property 2 (Lipschitzness)**.** This property requires that for the inputs in the neighborhood of random input pairs from the training set, the distance between their outputs is less than the Lipschitz constant L times the distance between the inputs:

$$\begin{gathered}\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}_{\text{tr}},\ \tilde{\boldsymbol{x}} \in \mathcal{B}_{p,\epsilon}(\boldsymbol{x}),\ \tilde{\boldsymbol{x}}' \in \mathcal{B}_{p,\epsilon}(\boldsymbol{x}'),\\ ||F_\theta(\tilde{\boldsymbol{x}}) - F_\theta(\tilde{\boldsymbol{x}}')||_2 < L \cdot ||\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}'||_2\end{gathered} \tag{Lipschitzness}$$

The training-set-specific variant of Lipschitzness requires the same constraint on random input pairs from the training set:

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}_{\text{tr}}, ||F_\theta(\boldsymbol{x}) - F_\theta(\boldsymbol{x}')||_2 < L \cdot ||\boldsymbol{x} - \boldsymbol{x}'||_2 \tag{Lipschitzness^T}$$

Property 3 (SmallNeighborhood)**.** This property requires that for any two neighboring inputs within distance ϵ, the distance between their outputs is less than a constant L times ϵ:

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, d(x, x') < \epsilon \Rightarrow ||F_\theta(\boldsymbol{x}) - F_\theta(\boldsymbol{x}')||_2 < L \cdot \epsilon, \tag{SmallNeighborhood}$$

where $d(x, x') = \max_i \frac{x_i - x_i'}{\sigma_i}$ is the l_∞ norm of the feature values normalized by the standard deviation. This property can be regarded as a relaxed form of Lipschitzness.

Property 4 (Monotonicity)**.** Given a feature j,

$$\begin{aligned}&\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \quad [x_j \le x_j' \land (\forall i \ne j, x_i = x_i')] \Rightarrow\\ &\qquad \text{Increasing} \quad F_\theta(\boldsymbol{x})_y \le F_\theta(\boldsymbol{x}')_y \quad (\text{Monotonicity}^I)\\ &\qquad \text{Decreasing} \quad F_\theta(\boldsymbol{x})_y \ge F_\theta(\boldsymbol{x}')_y \quad (\text{Monotonicity}^D)\end{aligned}$$

This is a domain-specific security property where the model output is positively or negatively correlated with the value of certain features that affect the suspiciousness of the inputs in common sense. Below we present an example to explain this property.

• *An example of the monotonicity property.* In the task of health insurance premium prediction, the goal is to predict whether the premium cost of an insured exceeds a certain value according to his or her personal information. Aside from other features, a higher BMI value or a habit of smoking implies a worse health status of the insured person, and thus may result in a higher cost of medical premium. Thus, the BMI value and the smoking habit are features with monotonicity property in this task.

Property Generalizability. We applied the concept of robustness generalization in [10, 13] as the property generalizability in this study. For instance, DL2 [13] trains properties with low generalizability which sample inputs from training data (e.g., Lipschitzness(T)), or moderate generalizability which sample inputs from the neighborhood of the training data (e.g., Lipschitzness), bounded by l_p norm distance. LogicEnsemble [10] trains properties with high generalizability, sampling inputs from arbitrary feature space.

Model Stealing Attacks and Defenses. MS attacks involve reconstructing the target model's parameters (non-public) [49, 50] or stealing the model functionality [12, 20, 49, 54] (i.e., the decision boundary). In this paper, we focus on the attacks where the adversary aims to train a substitute model which approximates the behavior of the victim model to steal the functionality. Such attacks use data augmentation techniques [20, 39] or sampling strategies [49] to generate a set of data points and query the victim model to label them. The labeled dataset is used to train a substitute model for high prediction accuracy on unseen testing data and high prediction agreement with the victim model. In our study, we evaluate three state-of-the-art model stealing attacks (in five variants) on robust models to investigate the privacy impact on security properties w.r.t the model confidentiality (see Section 3.2).

Existing defenses strategies against model stealing are in two streams: detection-based methods [18, 22, 38, 56, 57] which aim to detect stealing query patterns, and perturbation-based methods [21, 28, 37] which aim to degrade the quality of the predicted posterior via perturbation. Detection-based attacks are not the focus of this paper since they make strong assumptions on the query set, i.e., the distance between queried samples are expected to approximately fit a normal distribution [20], and are only targeted to specific attacks [35, 36]. Contrarily, perturbation-based defenses are more general and effective [21, 35, 37]. In perturbation-based defenses, the defender aims to decrease the performance of the substitute model or to increase the number of queries required by the attack. In this paper, we evaluate two popular perturbation-based model stealing defense methods on robust models to analyze the privacy impact on security properties from the perspective of risk mitigation (§3.3).

3 MEASUREMENT STUDY: STOLEN RISKS OF ROBUST MODELS

In this section, we present our measurement study on stolen risks of robust models. Specifically, we compare the stolen risks of models with and without security properties under three model functionality stealing attacks, i.e., Random Query Attack, Line Search Attack, Jacobian-based Augmentation Attack, and provide qualitative analysis on why models trained with security properties are more *vulnerable* to MS attacks. We also evaluate two popular *defense* methods against MS attacks and explain why such approaches are *less effective* on robust models.

Below we introduce the threat model of the attack, the datasets for applications that are highly sensitive both in security and privacy, and the target models with different security properties to bootstrap our measurement study.

Threat Model. We focus on substitute model training attacks that steal the functionality of the target models under *Label-only* black-box settings. The target model F_T maps a feature vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]$ with d features to the prediction confidence scores $[f_T(\boldsymbol{x})_1, f_T(\boldsymbol{x})_2, \ldots, f_T(\boldsymbol{x})_C]$ for C classes. We denote the normalized confidence scores as $F_T(\boldsymbol{x})$, where $F_T(\boldsymbol{x})_i = \frac{\exp(f_T(\boldsymbol{x})_i)}{\sum_{j=1}^{C} \exp(f_T(\boldsymbol{x})_j)}$ Following [10], we focus on binary classification (i.e., $C = 2$), which can be extended to the multi-class scenario.

The attack goal is to train a substitute model F_S with a decision boundary similar to that of the target model F_T. The attacker knows the category of the target ML model, while its architecture and parameters are unknown. The attacker has access to an auxiliary dataset D_{aux} as a seed dataset, which is either a small part (D^P_{aux}) of or has similar distribution (D^X_{aux}) with the training set of the target model. Given an input $\boldsymbol{x}$, the target model F_T predicts the probability of $\boldsymbol{x}$ being in class k as $F_T(\boldsymbol{x})_k$, and returns the prediction label as $y_T(\boldsymbol{x}) = \text{argmax}_k F_T(\boldsymbol{x})_k$. In addition, similar to previous work [17, 20, 49, 57], we assume the attacker has limited budget (Q) to query the target model.

3.1 Experimental Setup

Datasets The overview of the datasets is shown in Table 1. As in [30], we split each dataset D into a training set D_{tr}, a testing set D_{ts}, and an external auxiliary dataset D^X_{aux} unseen to the models, and randomly select a small part of D_{tr} as a partial seed set D^P_{aux}. We elaborate on the details of each dataset below.

Dataset	$\lvert D\rvert$	$\lvert D_{\text{tr}}\rvert$	$\lvert D_{\text{ts}}\rvert$	$\lvert D_{\text{aux}}\rvert$	d
Medical Insurance	3,275	2,293	932	50	8
Cryptojacking	4,000	2,800	1,150	50	7
Customer Churn	10,126	7,088	2,988	50	19
Spam Account	39,980	27,986	11,944	50	15
Spam URL	359,218	251,488	107,730	50	25

Table 1: Overview of the measurement datasets. d represents the number of features.

- *Medical Insurance* (MI). The Medical insurance dataset [1] records 3,630 user information in a medical insurance company. After deduplication in the pre-processing, the cleaned dataset contains information of 3,275 users, where 384 are positive samples and 2,891 are negative samples.
- *Customer Churn* (CC). The Customer Churn dataset [2] is used to predict whether the customers will continue their credit card service. The dataset consists of 10,126 samples in two categories: 8,500 existing customers and 1,626 attrited customers. In our evaluation, we remove ID-like features such as credit card numbers. We also encode categorical features as ordinal integers and transform all data as data frames using scaler.
- *Spam Account* (SA). The Spam Account dataset collects Twitter spam account information (e.g., account age, number of followings) via a social honeypot [27]. This dataset is balanced with 19,254 randomly-sampled benign accounts and 20,726 randomly-sampled spam accounts.
- *Twitter Spam URL* (SU). The Spam URL dataset [10, 24] has been utilized to train a Twitter spam URL detector, using features related to URL redirection chains and graphs. In our study, we used the dataset introduced in [10], where they re-extracted 25 features based on the description in [24]. This is the largest dataset in our evaluation experiment, which contains 158,704 positive samples and 200,566 negative samples.
- *Cryptojacking* (CJ). The Cryptojacking dataset [23] collects cryptojacking website data from 12 families of mining libraries. This dataset includes 2,004 positive samples and 1,996 negative samples.

Target Models We apply the same target models as in DL2 [13] and LogicEnsemble [10]. Specifically, we train Multi-Layer Perceptron (MLP) [34] with LocalInvariance and Lipschitzness using DL2 and train XGBoost [9] with SmallNeighborhood and Monotonicity using LogicEnsemble. We follow the model architecture and other experimental settings of DL2 and LogicEnsemble while fine-tuning several hyperparameters, such as the learning rate and the number of trees. The hyperparameters used in our implementation can be found in Appendix 9.2. We evaluate the performance of the target models, and the results can be found in Appendix 9. In particular, Table 9 shows the results of natural models. Table 10 shows the test performance and the security property effectiveness, i.e., the constraint accuracy, of robust models trained with LocalInvariance and Lipschitzness. Table 11 shows the test performance of robust models trained with SmallNeighborhood and Monotonicity. Since we train these two properties using LogicEnsemble, which enforces global robustness, their constraint accuracy is equivalent to 100%.
Metrics Aligned with previous MS attacks [12, 17, 20, 49], we measure the *agreement* between predictions of the substitute models and the target models. This is equivalent to measure the accuracy of the predictions of the substitute models when taking the predictions of the target models as groundtruth labels. In addition, we report the *Mean Squared Error* (MSE) between the prediction posteriors of the substitute models and the target models. A smaller MSE indicates that the substitute models have more similar posterior with the victim models at a score-level. We also measure the classification performance of the substitute models. Since some of our datasets are unbalanced (e.g., Medical dataset), we use *F1 score* taking the class with minority samples as positive class instead of accuracy.

3.2 Model Stealing Attack against Robust Model

Attack Methods and parameter settings We evaluate the following MS attacks on natural and robust models:

- *Random Query Attack* [49], a baseline strategy which samples Q points $x \in \mathcal{X}$ at random following a given distribution. We measure the two distributions of this strategy: *Uniform* distribution and *Gaussian* distribution.
- *Line Search Attack* [49], which samples random points in the feature space and generates additional data using line search between the random points. The line search algorithm generates adaptive queries as the mean between two samples crossing the decision boundary. This attack method does not require a seed dataset.
- *Jacobian-based Augmentation (JBA) Attack* [20, 39, 54], which trains a substitute model using a seed dataset and iteratively generates adversarial examples of the substitute model. It queries the

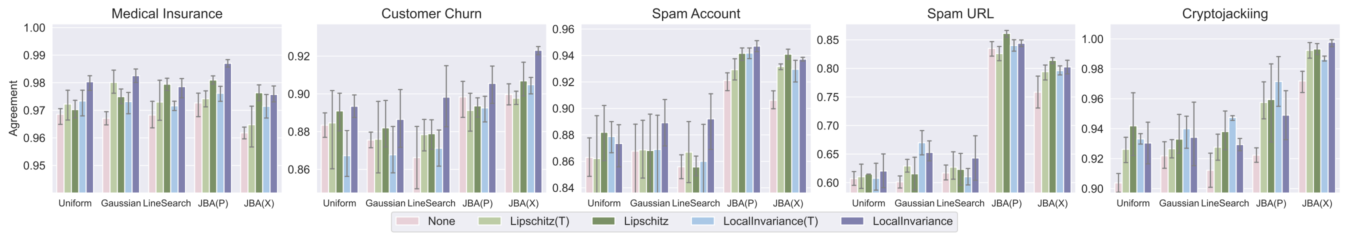


Figure 1: Agreement of MS attacks on MLP models as well as robust MLP with Lipschitzness and LocalInvariance.

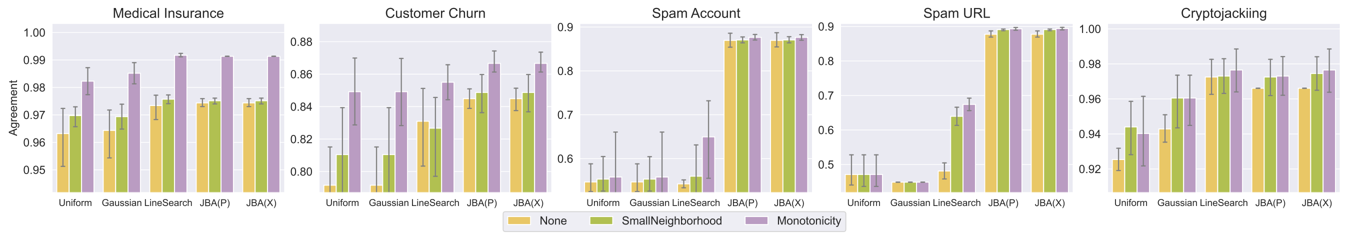


Figure 2: Agreement of MS attacks on XGBoost models as well as robust XGBoost with SmallNeighborhood and Monotonicity.

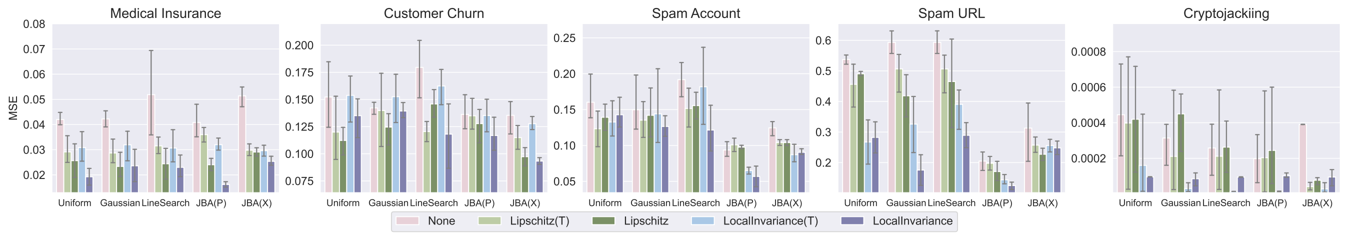


Figure 3: Mean Squared Error of MS attacks on natural MLP as well as robust models with Lipschitzness and LocalInvariance.

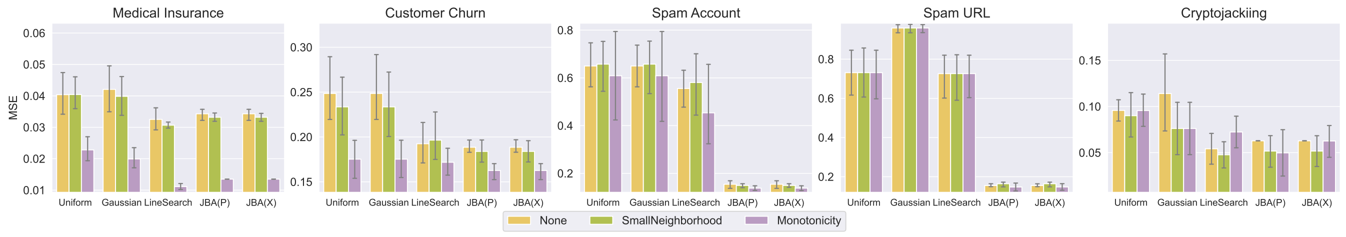
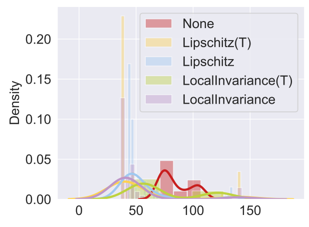


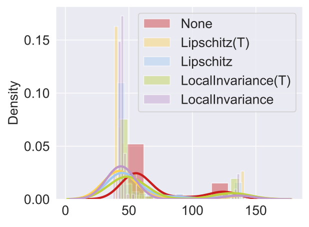
Figure 4: Mean Squared Error of MS attacks on XGBoost models and robust XGBoost with SmallNeighborhood and Monotonicity.

target model to label the adversarial samples as an augmented dataset to retrain the substitute model. The augmentation technique can apply different algorithms to craft adversarial examples. For differentiable models (MLP in this paper), we apply Iterative FGSM (I-FGSM) [14] to generate adversarial examples. This approach adds perturbation to benign samples towards increasing the classification loss iteratively in k steps. For non-differentiable models (XGBoost in this paper), we apply ZOO [8], a black-box adversarial attack against non-differentiable models which approximates the gradient with symmetric difference quotient [26]. We evaluate two variants of Jacobian-based Augmentation Attacks: *JBA(P)* that takes a small part of the training set D^P_{aux} as the seed set and *JBA(X)* that takes an auxiliary dataset D^X_{aux}, which has the same distribution with the training data of the victim model.

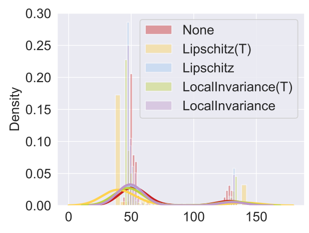
• *Attack Parameter Settings*. For all datasets, we set the size of the seed query set $|D_{\text{aux}}|$ as 50 and the total query budget Q as 200. We set L_∞ Ball for LocalInvariance and Lipschitzness properties as in [13]. We empirically select the *best* hyper-parameters associated with synthetic data generation for each victim model. Specifically, for Random Query attacks, we set a range r from 1 to 5 with step size 0.5 and generate random points from the interval $[-r, r]$. For JBA attacks on Neural Networks, we set the all perturbation steps



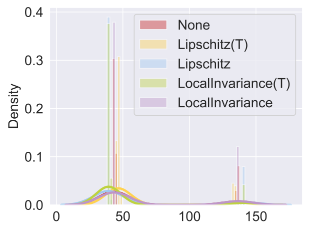
(a) $N = 50$



(b) $N = 100$



(c) $N = 200$



(d) $N = 400$

Figure 5: Distribution of angular deviation θ between gradient on weights.

as 5 and use GridSearch to select the best perturbation budget ϵ from 0.01 to 0.5 for each dataset and victim model. For JBA attacks on XGBoost, we select the best value of the constant h for gradient estimation from $\{0.0001, 0.001, 0.01, 0.1\}$. The augmentation rounds is set as $\log_2 \frac{Q}{|D_{\text{aux}}|} = 2$.

Results We report the agreement (Figure 1,2) and MSE (Figure 3, 4) to compare the effectiveness of MS attacks on natural models versus robust models. The figures show the associated average, the minimum, and the maximum values. Overall, models with security properties (i.e., LocalInvariance, Lipschitzness, SmallNeighborhood, and Monotonicity) show higher agreement and lower MSE in all attacks. Specifically, for neural networks, models trained with LocalInvariance and Lipschitz are more vulnerable (i.e., higher agreement and lower MSE) to MS attacks than models trained with LocalInvariance^T and Lipschitz^T. Both Lipschitzness and SmallNeighborhood are security properties that aim to ensure prediction smoothness, and they have less impact on MS attacks when compared with LocalInvariance or Monotonicity.

We also plot the F1 scores of the substitute models attacking different target models in Appendix Figure 9,10. Note that those F1 scores cannot be directly compared due to variations in the original test performance of the target models.

Insights and Analysis To analyze the underlying reason for the higher performance on stealing robust models, we borrow the concept of *angular deviation* from a defending method, Prediction Poisoning (PP) [37]. Particularly, PP adds targeted noise to the victim model's posteriors which results in a gradient direction of adversary's model to maximize the angular deviation between the original and the poisoned gradient signals. The main idea is to label a poisoned query set which could deteriorate the training process of the substitute model. The angular deviation captures the changes on the gradient of the adversary's loss w.r.t. the model parameters after the perturbation. A large deviation indicates the query set labeled by the perturbed victim model is poisoning for training a substitute model similar to the victim. Therefore, the angular deviation serves as a metric to measure the quality of the labeled query set, taking the original victim's posterior as an optimal gradient direction. PP experimentally demonstrates the effectiveness of this mechanism under the label-only threat model, where the attacker only has access to the top-one labels of the victim's predictions.

Inspired by this work, we adopt the angular deviation to the attack scenario to quantify the potential of a victim model's functionality to be stolen by substitute model training. The main challenge of model stealing lies in the limited knowledge of the training data. Many attack methods make efforts to generate high-quality synthesized samples similar to the original samples and near to the model's decision boundary. Thus, we take the gradient from the loss on benign samples $(\boldsymbol{x}, y)$ as the optimal direction

$$\boldsymbol{u} = -\nabla_w L(F_S(\boldsymbol{x}; w), \boldsymbol{y}) = \sum_k y_k \nabla_w \log F_S(\boldsymbol{x}, w)_k = G^T \boldsymbol{y},$$

and calculate its angular deviation from the gradient from synthesized samples $(\tilde{\boldsymbol{x}}, \tilde{y})$

$$\tilde{\boldsymbol{u}} = -\nabla_w L(F_S(\tilde{\boldsymbol{x}}; w), \tilde{\boldsymbol{y}}) = \sum_k \tilde{y}_k \nabla_w \log F_S(\tilde{\boldsymbol{x}}, w)_k = G^T \tilde{\boldsymbol{y}},$$

$$\theta = \arccos\left(\frac{\boldsymbol{u} \cdot \tilde{\boldsymbol{u}}}{|\boldsymbol{u}| \cdot |\tilde{\boldsymbol{u}}|}\right),$$

where G is the Jacobian matrix of log-likelihood predictions $F(\boldsymbol{x}, w)$ w.r.t. model parameters. A smaller deviation θ implies a higher quality of the synthesized samples, which may result in more successful attacks.

Figure 5 shows the distribution of the angular deviation on benign samples and synthesized samples of Medical Insurance dataset. Here, we randomly select N samples from the testing set as benign samples and N samples from the augmented set in JBA(P) attack as synthesized samples. Following [37], we randomly initialize the substitute model and train it with batch size 1. At each training step, we record the gradient direction from the loss on the sample newly added to the training data. Finally, we calculate the pair-wise cosine similarities between the gradient direction from benign samples and synthesized samples, map them to the angular deviations using the inverse trigonometric functions and plot their distribution.

The results in Figure 5a show that with a small number of queries ($N = 50$), the resulted angular deviations from robust victim models are prevalently smaller than those resulted from natural victim models (e.g., red line with None property). This indicates that **robust models are capable to label synthesized dataset with higher quality for more successful MS attacks under a limited query budget.** One reason might be that the difference between predictions on benign samples and synthesized samples from robust models are constrained by the security properties, which results in more similar gradient directions and smaller angular deviations. In addition, we observe that such advantage decays with the increase of the number of queries (see Figure 5b, 5c,5d). Hence, the difference in the quality of the query sets could be compensated by a larger size of the query sets.

Dataset	Target Model	Prediction Poisoning					Adaptive Misinformation				
		Uniform	Gaussian	LineSearch	JBA(P)	JBA(X)	Uniform	Gaussian	LineSearch	JBA(P)	JBA(X)
Medical Insurance	MLP	**2.45**	**2.88**	**4.58**	**3.02**	5.39	**6.54**	5.07	5.47	2.43	3.07
	Lipschitzness(T)	0.13	2.32	3.8	3.5	8.33	1.94	**7.26**	3.91	**3.29**	3.95
	Lipschitzness	0.92	2.23	2.85	0.18	4.42	2.48	7.06	3.12	3.04	0.25
	LocalInvariance(T)	0.55	2.00	3.47	1.12	**8.5**	3.07	4.11	**5.68**	0.71	**8.07**
	LocalInvariance	0.06	0.12	0.09	0.93	0.77	0.37	0.33	0.73	0.15	0.11
	XGB	**0.94**	**0.50**	**2.03**	**4.74**	**4.74**	**6.52**	**2.30**	**6.86**	**4.50**	**3.70**
	SmallNeighborhood	0.10	0.34	1.17	3.97	3.89	2.2	0.89	5.03	2.70	2.70
	Monotonicity	0.36	0.38	0.27	4.12	3.71	1.81	0.07	2.76	3.26	3.16
Curstomer Churn	MLP	**6.80**	4.63	**9.98**	6.51	0.92	**5.71**	2.13	**5.90**	4.27	**4.08**
	Lipschitzness(T)	5.87	5.39	0.17	3.73	**8.56**	5.07	**3.73**	2.38	**4.86**	2.57
	Lipschitzness	4.4	0.32	3.77	4.86	6.92	5.47	1.39	3.11	4.18	3.91
	LocalInvariance(T)	4.35	**6.43**	5.76	**8.5**	0.07	4.29	0.04	2	0.91	0.48
	LocalInvariance	4.99	2.18	6.67	5.97	3.15	4.31	3.53	0.13	4.65	0.35
	XGB	**12.46**	**12.46**	**4.57**	**5.23**	**5.23**	**8.97**	**3.64**	**12.14**	**2.03**	**2.13**
	SmallNeighborhood	2.41	2.41	4.27	5.18	5.18	1.1	1.56	11.73	1.02	0.48
	Monotonicity	0.95	0.95	3.29	4.98	4.98	5.59	3.28	7.24	0.97	1.52
Spam Account	MLP	1.54	2.12	**3.75**	0.91	0.87	**2.43**	0.46	**6.28**	1.42	0.45
	Lipschitzness(T)	2.00	1.20	2.27	0.64	1.13	1.56	1.12	0.72	0.00	0.50
	Lipschitzness	0.05	0.49	2.75	0.54	0.99	0.39	0.04	4.08	1.70	0.82
	LocalInvariance(T)	**3.36**	**3.18**	1.69	**1.14**	**1.93**	2.18	**1.71**	2.19	**1.57**	1.61
	LocalInvariance	0.54	1.00	2.05	1.12	0.62	0.93	0.49	2.08	0.24	**2.2**
	XGB	**1.87**	**2.44**	**3.21**	**0.81**	**1.64**	**2.45**	**1.71**	**4.38**	**1.96**	**1.09**
	SmallNeighborhood	0.95	1.15	0.24	0.38	0.38	0.93	0.83	1.77	0.50	0.30
	Monotonicity	1.76	0.38	1.41	0.63	0.63	1.28	0.17	2.13	0.96	0.38
Spam URL	MLP	**3.43**	**3.01**	**2.67**	1.11	1.56	**2.79**	**1.84**	**3.12**	2.29	1.33
	Lipschitzness(T)	2.12	1.43	1.79	1.34	1.87	1.97	1.57	1.67	0.97	1.15
	Lipschitzness	0.87	0.94	1.73	0.95	1.69	0.82	0.31	3.75	2.10	1.26
	LocalInvariance(T)	3.13	2.96	1.77	**1.43**	**2.01**	2.35	1.69	2.26	**1.64**	**1.65**
	LocalInvariance	0.94	1.12	1.95	1.12	0.98	1.65	0.87	2.14	0.67	1.22
	XGB	**1.95**	**2.56**	**2.31**	**1.31**	**1.94**	**2.25**	**1.89**	**3.01**	**1.67**	**1.49**
	SmallNeighborhood	1.53	1.09	0.84	0.63	0.92	1.10	0.56	1.32	0.91	0.24
	Monotonicity	1.72	0.74	1.65	0.81	0.73	1.35	0.56	2.01	1.21	0.45
Cryptojacking	MLP	1.75	2.46	**3.15**	1.34	1.21	**2.43**	0.97	**3.24**	1.64	0.81
	Lipschitzness(T)	2.06	1.72	2.35	0.74	**1.89**	1.26	1.64	1.25	0.78	**2.34**
	Lipschitzness	0.32	0.73	2.42	0.73	1.46	0.89	0.04	3.08	2.14	1.21
	LocalInvariance(T)	**3.02**	**2.64**	1.25	**1.02**	1.63	2.05	**1.71**	2.32	**1.13**	1.35
	LocalInvariance	1.43	1.35	2.65	1.54	0.97	1.36	0.69	1.97	0.64	1.35
	XGB	**1.68**	**2.14**	**2.23**	**1.11**	**1.47**	**2.23**	**1.98**	**2.98**	**2.36**	**0.97**
	SmallNeighborhood	0.82	1.04	1.78	0.64	0.38	1.01	0.95	1.92	0.65	0.72
	Monotonicity	0.89	0.83	1.94	0.93	0.67	1.18	0.43	2.15	1.59	0.72

Table 2: Decrease in substitute models' F1, compared to the attack performance without defense.

3.3 Defense Degradation on Robust Models

Defense Methods and Parameter Settings In our study, we evaluate the following state-of-the-art defense methods against MS attacks on robust models:

• *Prediction Poisoning (PP)* [37]: the defender aims to decrease the accuracy of the stolen model by adding targeted noise to the posteriors of the victim model by $F'_T(x) = (1-\alpha)F_T(x) + \alpha\eta$, where η is an extreme point selected from one-hot representation of each class label, which maximizes the angular deviation between the original and the poisoned gradient of the surrogate model. The perturbation factor α is a hyper-parameter which controls the weight of the perturbation.

• *Adaptive Misinformation (AM)* [21]: the defender trains a misinformation function $\hat{F}$ by minimizing the reverse cross entropy loss and adaptively inject the misinformation into model prediction:

$$F'_T(x) \leftarrow (1-\alpha) \cdot F_T(x) + \alpha \cdot \hat{F}_T(x),$$

where $\alpha = \frac{1}{1+\exp\left\{v \cdot [\max_i F_T(x)_i - \tau]\right\}}$, v and τ are pre-defined hyper-parameters. This allows the defender to selectively modify the predictions of OOD (out-of-distribution, i.e., adversarial) queries while preserving the correctness of the ID (in-distribution, i.e., benign) queries. The defense method is motivated by the observations that the predictions on OOD queries and ID queries vary in Maximum Softmax Probability (i.e., the prediction confidence score).

• *Defense Parameter Settings.* For v in Adaptive Misinformation defense, we use the same value as in [21]. For τ in Adaptive Misinformation defense and α in Prediction Poisoning defense, we select the best values for each target model, respectively. The best value of the hyper-parameter is determined as the one that results in a largest degree of perturbation with at most 1% decrease in model accuracy. For example, for selecting the best value for α, we start from setting its value as 0.01 and test the decrease in accuracy after the perturbation. In each step, we increase the value of α by 0.01 until a decrease in accuracy reaches 1%. We showcase the original accuracy, the accuracy after defending perturbation and the decrease in accuracy of Medical Insurance dataset in Table 12.

Results Table 2 shows the decrease in F1 score of the substitute models, where the target models are defensed by Prediction Poisoning and Adaptive Misinformation. The results are averaged on five individual tests. Overall, both defending methods show higher effectiveness on naturally trained models, where the substitute models have a larger F1 decrease. We also observe that the training set constraints (i.e., Lipschitzness^T, LocalInvariance^T) might result in higher F1 decrease, compared with natural models.

Specifically, for Medical Insurance dataset, LocalInvariance^T model has the largest F1 decrease in both defense methods against JBA(X) attack, where the F1 decreases of Lipschitzness^T model are also larger than naturally trained model. For Customer Churn dataset, LocalInvariance^T model has the largest F1 decrease in PP defense against Gaussian and JBA(P) attacks, and Lipschitzness^T model has the largest F1 decrease in PP defense against JBA(X) attacks and MA defense against Gaussian and JBA(P) attacks. For Spam Account dataset, LocalInvariance^T model has the largest F1 decrease in PP defense against Uniform, Gaussian and both JBA attacks, and in AM defense against Gaussian and JBA(P) attacks.

Insights and Analysis To further analyze the defending effectiveness, we compare the agreement between the predictions made by defended and undefended models on ID and OOD samples, respectively. This measures how similar predictions a target model can make after it is defended against model stealing, compared to its undefended counterparts. Table 3 shows the results on Medical Insurance dataset. For each defense method, the third column (i.e., Gap) is the difference between the prediction agreements on ID samples and OOd samples. A larger gap indicates that the defended model is more capable to make correct predictions on ID samples and misleading predictions on OOD samples, which implies higher defending capability in terms of the trade-off between privacy and accuracy. As shown in Table 3 and Table 12, with similar accuracy decrease (≈ 1%) resulted by the perturbation, both defending methods show significantly higher agreement gaps on naturally trained models than robust models. The underlying reasons are two folds. First, **the robust training process makes the predictions on ID samples more sensitive to the perturbations.** This limits the degree of the perturbation that could be added to the robust models' predictions, given the requirement to preserve model usability. Second, the **robust training enforces the victim models to make similar predictions between ID and OOD samples**. This makes it difficult for the defending algorithm to distinguish OOD samples from ID samples, which disables the selective modification on the predictions of OOD samples.

Target Model	Prediction Poisoning			Adaptive Misinformation		
	ID	OOD	Gap	ID	OOD	Gap
MLP	96.42	81.65	**14.76**	96.30	82.52	**13.78**
Lipschitzness(T)	96.81	91.26	5.55	98.23	95.55	5.68
Lipschitzness	96.97	91.02	5.94	97.95	94.61	3.35
LocalInvariance(T)	95.63	86.73	8.90	97.52	93.82	3.70
LocalInvariance	96.89	92.13	4.76	98.66	97.56	1.10
XGB	98.14	92.56	**5.58**	98.60	95.16	**3.44**
SmallNeighborhood	98.14	94.42	3.72	98.60	96.09	2.51
Monotonicity	99.07	97.21	1.86	99.26	98.14	1.12

Table 3: Agreement between predictions made by undefended and defended models on in-distribution (ID) and out-of-distribution (OOD) samples of Medical Insurance dataset.

3.4 Discussion

Choice of Datasets. In this paper, we investigate the privacy impact on security properties of robust models (trained by the framework DL2 [13] and LogicEnsemble [10]), and discuss the factors associated with security properties, such as generalizability. To the best of our knowledge, LogicEnsemble [10] is currently the only robust training framework to enable global security properties (i.e., highest generalizabilty), while this framework cannot be adapted to image datasets. To evaluate robust model trained by LogicEnsemble, our study select datasets following [10]. We acknowledge the importance of conducting measurement studies on image datasets regarding privacy impacts on security properties and MS attacks. However, we leave this task as future work.

Theoretical Analysis. In our paper, we conduct a measurement study on stolen risks of robust models. While our study provides valuable insights into the practical implications of security properties on model stealing, we recognize the need for an in-depth theoretical analysis of the stolen risks of robust models. The analysis can be bootstrapped by formulating a definition for the risk of model stealing, followed by analyzing linear models trained robustly, similar to the attack analysis in StealML [49], and use the results to deliver preliminary theoretical insights into the impact of security properties as well as other factors on the risk of model stealing. We will leave this task as future work.

4 OPTIMIZED MODEL STEALING ATTACK AGAINST ROBUST MODELS

Based on the above observations, in this section, we propose an optimized MS attack against models with security properties. Our attack incorporates an inference strategy based on the security property into existing MS attacks, which enlarges the volume of the substitute training set under the same query budget. We elaborate on the attack procedure in §4.1 and experimental results in §4.2

4.1 Methodology

MaskedThief applies a semi-supervised learning approach to save queries, enabling the attacker to label more augmented data under the fixed query budget. Specifically, the attacker first queries

Algorithm 1: MaskedThief: Semi-supervised MSA

```
Input: Prediction API $M_T$ and Property $\varphi$ of the victim
       model $F_T$; Query Budget $Q$; Seed Query Set
       $X_{\text{seed}} = \{x_1, x_2, \ldots, x_m\}$; $n$ training epochs for each
       augmentation round; N training epochs after all
       rounds
Output: Substitute Model $F_S$
$X_T \leftarrow X_{\text{seed}}$; $Y_T \leftarrow M_T(X_T)$ ;$D_T \leftarrow \{(X_T, Y_T)\}$
Initialize $F_S$ and train $n$ epochs on $D_T$
while Number of queries < $Q - m$ do
    for $x$ in $X_T$ do
        if $|D_T| > B$ then
            break
        $x' \leftarrow$ Synthesize$(x, F_S, M_T, \mathcal{L}, t, \alpha)$
        $X_T \leftarrow X_T \bigcup \{x'\}$
        if Infer$(x', X_T, Y_T, \psi)$ then
            $Y_T \leftarrow Y_T \bigcup \{$Infer$(c, x', X_T, Y_{T,c}, \varphi)\}$
        else
            $Y_T \leftarrow Y_T \bigcup \{M_T(x')\}$
        $D_T \leftarrow \{(X_T, Y_T)\}$
        Reinitialize $F_S$ and train $n$ epochs on $D_T$
Reinitialize $F_S$ and train $n$ epochs on $D_T$ epochs
return $F_S$
```

Table 4: Deviation of approximation for binary classification.

φ	$\xi_\varphi(\boldsymbol{x}, \boldsymbol{x}')$
Lipschitzness	$\frac{L}{\sqrt{2}} \cdot \lvert\lvert \boldsymbol{x} - \boldsymbol{x}' \rvert\rvert_p$
LocalInvariance	$\delta \cdot \exp\{\max(\lvert\lvert \boldsymbol{x} - \boldsymbol{x}' \rvert\rvert_p - \epsilon, 0) \cdot \lambda\}$
SmallNeighborhood	$\frac{\epsilon \cdot L}{\sqrt{2}} \cdot \exp\{\max(\lvert\lvert \boldsymbol{x} - \boldsymbol{x}' \rvert\rvert_p - \epsilon, 0) \cdot \lambda\}$
Monotonicity(I)	$(1 - F_T(\boldsymbol{x})_c) \cdot \exp\{\max(x_j - x'_j, \sum_{i \neq j} \lvert x_i - x'_i \rvert, 0) \cdot \lambda\}$
Monotonicity(D)	$(1 - F_T(\boldsymbol{x})_c) \cdot \exp\{\max(x'_j - x_j, \sum_{i \neq j} \lvert x_i - x'_i \rvert, 0) \cdot \lambda\}$

the target model F_T with a seed set to train a substitute model F_S and synthesizes *additional samples* by random sampling or data augmentation. For each additional sample, the attacker infers F_T's prediction on it along with a prediction uncertainty, according to its distance to the preceding queries. Synthesized samples with high inference certainty (i.e., low information w.r.t decision boundary) are masked from the query dataset, as the attacker directly uses the inferred predictions as their labels. The attack procedure is shown in Algorithm 1. For each additional query, we calculate an intersected confidence interval (CI), with the width determined by the security property. The validity of the CI border (i.e., left endpoint ≤ right endpoint) serves as a measure of uncertainty, while the median serves as the inference result. During this process, the security property serves as a bridge that connects the response of the additional query with the information already known by the attacker, such as the responses of previous queries or the distances between the additional query and previous queries. This connection guides the inference process in semi-supervised learning, providing a theoretical guarantee of inference accuracy. In other words, the security property helps the attacker to make better use of the available information, allowing them to make more accurate inferences from the limited labeled data.

More specifically, given an additional sample x', the attacker calculates a confidence interval $\text{CI}(x')$ with respect to a given confidence level η according to the prediction on the K-nearest neighbors of x', and infers the posterior $F_T(x')$. The inference process is shown in Algorithm 2. Below we elaborate on how the adversary infers the target model's prediction on the augmented samples and how this works on models with security properties. As in [47], we assume the response is conditionally Bernoulli distributed given the values of the features: $\mathbb{1}[y(\boldsymbol{x}) = c] | \boldsymbol{x} \sim \text{Bernoulli}(F_T(\boldsymbol{x})_c)$.

Algorithm 2: Infer: posterior of target model

```
Input: Label $c$, Augmented sample $x'$;,Previous query
       inputs $X_T$ and predictions $Y_{T,c}$, Model Property $\varphi$;
       Hyperparameters: Maximum number of neighbors
       $K$, Confidence Level $\eta$, Marginal of Label $p$;
Output: Inferred prediction $y'$ on label $c$
$l \leftarrow 0; r \leftarrow 1$ ;
for $k$ in $\{1, 2, \ldots K\}$ do
    $\mu_k \leftarrow 0$; $\sigma_k \leftarrow 0$
    $X_k \leftarrow k$ nearest neighbors of $x'$ in $X_T$
    $Y_{k,c} \leftarrow$ Predictions of $x \in X_k$ on label c in order
    $D_k \leftarrow \{(x_i, y_i) | x_i \in X_k, y_i \in Y_{k,c}\}$
    for $(x_i, y_i)$ in $D_k$ do
        $\mu_k \leftarrow \mu_k + y_i$
        $\sigma_k \leftarrow \sigma_k + \xi(x_i, x')$
    $\mu_k \leftarrow \frac{\mu_k}{k}$; $\sigma \leftarrow \frac{\sigma_k}{k} + \sqrt{\frac{1}{2k} \ln(\frac{2k}{1-\eta})}$
    $l \leftarrow \max(l, \mu_k - \sigma_k)$; $r \leftarrow \min(r, \mu_k + \sigma_k)$
if $r < l$ then
    return False
else
    if $\frac{l+r}{2} > p$ then
        return 1
    else
        return 0
```

Let $\boldsymbol{x}' \in \mathcal{B}_{p,\epsilon}(\boldsymbol{x})$ denote a synthesized sample of $\boldsymbol{x}$. For any $k \in \{1, 2, \ldots, K\}$, let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ be the k nearest neighbors of $\boldsymbol{x}'$ in the labeled set. Also, for each $i \in \{1, 2, \ldots, k\}$, let $y_{i,c}$ be the response (independently) sampled from $\text{Bernoulli}(F_T(\boldsymbol{x}_i)_c)$, where $\Pr(y_{i,c} = 1) = F_T(\boldsymbol{x}_i)_c$ and $\Pr(y_{i,c} = 0) = 1 - F_T(\boldsymbol{x}_i)_c$. By Hoeffding's Inequality [16], for any failure probability $\alpha \in (0, 1)$ we have that

$$\Pr\left\{\left|\frac{1}{k}\sum_{i=1}^{k} F_T(\boldsymbol{x}_i)_c - \frac{1}{k}\sum_{i=1}^{k} y_{i,c}\right| \leq \sqrt{\frac{1}{2k}\ln(2/\alpha)}\right\} \geq 1 - \alpha. \quad (1)$$

On the other hand, we let $\xi_\varphi(\boldsymbol{x}', \boldsymbol{x}_i)$ denote the upper bound on the deviation term $|F_T(\boldsymbol{x}')_c - F_T(\boldsymbol{x}_i)_c|$ based on the security property φ of the target model. The deviation $\xi_\phi(\boldsymbol{x}, \boldsymbol{x}')$ with regard to different security property φ is listed in Table 4. Now we have that

$$\left| F_T(\boldsymbol{x}')_c - \frac{1}{k}\sum_{i=1}^{k} F_T(\boldsymbol{x}_i)_c \right| \le \frac{1}{k}\sum_{i=1}^{k} \xi_\varphi(\boldsymbol{x}', \boldsymbol{x}_i). \qquad (2)$$

Combining Equations (1,2), we have that with probability at least $1-\alpha$, it holds that

$$F_T(\boldsymbol{x}')_c \in [\ell_{k,c}, r_{k,c}] \overset{\text{def}}{=} \frac{1}{k}\sum_{i=1}^{k} y_{i,c} \pm \left(\sqrt{\frac{1}{2k}\ln(2/\alpha)} + \frac{1}{k}\sum_{i=1}^{k} \xi_\varphi(\boldsymbol{x}', \boldsymbol{x}_i) \right), \qquad (3)$$

where we use $a \pm b$ to denote the interval $[a-b, a+b]$. We further explain the derivation of Equations (1, 3) in Appendix 9.4 and Equation 2 in Appendix 9.3.

Now, based on the k nearest neighbor of $\boldsymbol{x}'$, we have derived a confidence interval $[\ell_k, r_k]$ for $F_T(\boldsymbol{x}')_c$ with confidence level $(1-\alpha)$. Setting $\alpha = (1-\eta)/K$ and via a union bound over all $k \in \{1, 2, \ldots, K\}$, we have that

$$\Pr[F_T(\boldsymbol{x}')_c \in \cap_{k=1}^{K} [\ell_k, r_k]] \ge \eta.$$

Therefore, we set $\mathrm{CI}(\boldsymbol{x}') = \cap_{k=1}^{K} [\ell_k, r_k]$ to be our final confidence interval for $F_T(\boldsymbol{x}')_c$ with confidence level η.

4.2 Evaluation

Implementation and Settings. Given an additional sample, the attacker determines to query the target model or to make an inference according to the validity of the confidence interval derived in §4.1. Specifically, the attacker considers a confidence interval as invalid if the left end point of the confidence interval is greater than or equals the right end point, or the length of the interval exceeds a threshold. In this circumstance, the attacker queries the target model to label the sample. Otherwise, the attacker makes an inference on the sample by comparing the end points of the confidence interval to a threshold. In our implementation, for all experiments we set $\lambda = 2$ and $K = 50$ and use the marginal distribution of the class in the labeled set as the threshold. The adversary leverages GridSearch [25] on the labeled set to compute the best attack parameters (i.e., failure probability α and property parameter L, δ, or ϵ) for each dataset.

Baseline. MixMatch [4] is a holistic semi-supervised learning (SSL) algorithm that guesses low-entropy labels for data-augmented unlabeled examples and mixes labeled and unlabeled data using MixUp [55]. To integrate MixMatch with MS attacks, we use the predictions of the victim model as the labels to be guessed, and the substitute model as the target of SSL. Specifically, we first query the target model to label the seed dataset and the synthesized data, and then update the substitute model as described in § 3.2 until we reached the query budget Q. After obtaining a labeled dataset X and a trained substitute model, we continued to synthesize Q samples as the "equal-size unlabeled set" U in MixMatch. Then we applied MixMatch to guess the labels, compute the combined loss L for SSL, and used L to update the substitute model. We use the same data augmentation technique on MaskedThief and MixMatch. As in [4], we set $T = 0.5$ and $K = 200$, and tune λ and α on a per-dataset basis. More specifically, we select the best value of α from 0.1 to 1 with a step size 0.05 and the best value of λ from 10 to 100 with a step size 10.

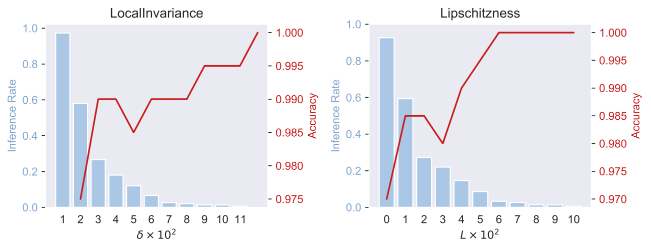


Figure 6: The inference rate and the accuracy of the substitute training set varying the attack parameter.

Results. Overall, MaskedThief can enhance the performance of all attack methods, as well as outperforming MixMatch, against robust models on all five datasets. Table 5 shows the effectiveness of MaskedThief on Spam Account dataset and Spam URL dataset, respectively, by presenting the F1 scores of the substitute models, compared with MixMatch and the original MS attacks. The experiment results of other datasets are shown in Appendix Table 13.

Specifically, we observe that for most robust models, MaskedThief has better performance on attack methods relying on randomly generated samples (i.e., Uniform, Gaussian, and LineSearch) than JBA. Moreover, for security properties in the same form, MaskedThief shows better performance on general properties than on properties specific to the training set. For example, MaskedThief increases 1.27%, 2.29%, 1.01%, 0.24%, and 0.36% F1 score of model with Lipschizness and increases 1.01%, 1.22%, 0.51%, 0.18%, and 0.06% F1 score of model with Lipschizness^T. The results indicate that attackers can leverage more information from security properties that have higher generalizability for MS attacks.

To understand how MaskedThief improves MS attacks against robust models, we plot changes in the proportion of samples being inferred by the attacker (i.e., inference rate) as well as the accuracy of the substitute training set (i.e., dataset used to train the substitute model) by varying the attack parameters. Figure 6 shows the results of LocalInvariance property and Lipschitzness property on Medical Insurance dataset, where MaskedThief applies inference to enhance JBA(X) attack. Specifically, we fix $\alpha = 0.1$ and vary the values of the parameters of security properties (i.e., δ for LocalInvariance model and L for Lipschitzness model). The labeling accuracy of the substitute training set through this semi-supervised strategy achieves almost 99% when the attacker labels 60% additional samples by inference instead of queries. Even when the attacker replaces 90% queries with inference, the accuracy still achieves 97%. Thus, this semi-supervised strategy can save a large portion of queries with high accuracy for the adversary to label the synthesized dataset. Under the same query budget, the adversary is rendered more opportunities to query the model with uncertain samples, which provides more information about the decision boundary of the target model.

Target Model	Attack Approach	Spam Account Dataset					Spam URL Dataset				
		Uniform	Gaussian	LineSearch	JBA(P)	JBA(X)	Uniform	Gaussian	LineSearch	JBA(P)	JBA(X)
Lipschitzness(T)	Original	84.81	84.87	85.21	89.54	90.17	63.2	64.12	65.44	81.48	75.74
	MixMatch	85	85.43	85.4	89.67	90.21	63.75	64.75	65.47	81.8	75.92
	MaskedThief	**85.22**	**85.84**	**85.43**	**89.88**	**90.45**	**63.99**	**65**	**66.07**	**82.45**	**76.41**
Lipschitzness	Original	83.57	85.01	84.37	90.81	90.57	63.52	65.37	67.15	82.06	76.43
	MixMatch	84.1	86.2	84.95	91.02	91.05	64.15	66.21	67.88	82.75	76.9
	MaskedThief	**84.57**	**87.08**	**85.08**	**91.52**	**91.47**	**64.77**	**66.68**	**68.77**	**83.11**	**77.27**
LocalInvariance(T)	Original	86.03	85.61	84.65	91.28	90.39	66.39	69.99	65.96	81.99	74.4
	MixMatch	80.01	86.01	84.99	91.32	90.56	66.83	70.6	66.75	82.01	74.13
	MaskedThief	**86.3**	**86.46**	**85.54**	**91.62**	**90.82**	**67.31**	**71.19**	**66.93**	**82.42**	**74.85**
LocalInvariance	Original	85.38	86.78	87.64	91.33	90.71	65.64	72.33	67.39	83.12	76.04
	MixMatch	85.9	88.34	87.88	91.94	91.53	66.89	74.9	68.37	83.78	76.86
	MaskedThief	**86.28**	**89.98**	**88.16**	**92.29**	**91.86**	**67.94**	**75.58**	**69.59**	**84.3**	**76.99**
SmallNeighborhood	Original	69.78	69.78	68.37	85.59	85.59	63.07	62.01	63.53	88.06	88.06
	MixMatch	69.9	69.9	69.9	85.95	85.95	63.88	62.59	63.78	88.61	88.61
	MaskedThief	**71.38**	**71.18**	**70.06**	**86.75**	**86.71**	**64.47**	**63.31**	**64.57**	**88.8**	**88.68**
Monotonicity	Original	69.6	69.6	74.69	87.45	87.45	63.07	62.12	65.36	88.47	88.6
	MixMatch	70.1	70.1	74.82	87.6	87.6	**63.87**	**63.24**	65.91	88.9	88.9
	MaskedThief	**70.44**	**70.36**	**74.9**	**87.83**	**87.68**	**63.87**	**63.24**	**66.12**	**89.21**	**89.22**

Table 5: Substitute Moddel F1 (%) on Spam Account Dataset and Spam URL Dataset.

4.3 Discussion

Sensitivity to the choice of dataset. To understand the sensitivity of MaskedThief to the choice of dataset, we conducted the experiments to measure attack improvement variance of MaskedThief within a dataset and across datasets. Specifically, for each dataset, we randomly select different seed datasets to evaluate attack improvement, and repeat the experiment five times. Figure 7,8 in Appendix 9.6 show the distribution of attack improvement of MaskedThief on each dataset and across five datasets, when compared with the original MS attacks (i.e., Uniform, Gaussia, LineSearch, JBA(X), JBA(P)). Experiment results show that attack improvement variance within a dataset and across five datasets are all small, with a standard deviation of 0.0008 and 0.0018, respectively, on average. This indicated the performance of MaskedThief is not sensitive to the choice of dataset.

Attack assumption. In MaskedThief, the adversary needs to know the form of the security properties, while the exact values of the property parameters can be unknown. The values of property parameters provide a guarantee of inference accuracy in our theoretical analysis in §4.1, but they are not necessarily known by the adversary in practice, as mentioned in §4.2. Particularly, our experiments use GridSearch [25] to select the best attack parameters. The results in Table 5 do not rely on the values of property parameters.

5 ROBUST MODEL WITH PRIVACY PROPERTY

In this section, we introduce BoundaryFuzz, a privacy property to defend against MS attacks on robust models. The state-of-the-art model stealing defense methods mitigate the stolen risks of models by selectively adding perturbations to model outputs. Although such methods achieve satisfying accuracy-privacy trade-off, it is difficult to maintain the security properties of the defended model. This is because that the inputs of the security properties can be selected by the defender to add perturbations. In this case, the constraints on their corresponding outputs will be undermined. In contrast, we propose a privacy property which fits well into the process of robust training and can be enforced together with the security properties. Such a defense strategy causes a moderate performance drop in both accuracy and security while protecting the confidentiality of model decision boundary.

5.1 Methodology

Our key idea is to protect the query responses near the decision boundary against most MS attacks. We apply the definition of *Boundary-Sensitive Zone* and *Sensitive Query* in [58]. The boundary-sensitive zone of model f_θ is the feature space adjacent to the decision boundary $Z_\Delta = \{x \in R^d | \text{dist}(\boldsymbol{x}, f_\theta) < \Delta)\}$, where Δ is a positive constant around 0, $\text{dist}(\cdot)$ measures the distance between a feature vector $\boldsymbol{x}$ and the decision boundary. A query $\boldsymbol{x}$ is determined sensitive to the model f_θ iff. $\exists x' \in \mathcal{B}_{p,\Delta}(\boldsymbol{x})$, such that $\boldsymbol{x}$ and $\boldsymbol{x}'$ lie on the different sides of the decision boundary of f_θ. In this paper, we use $\text{argmax}_{c \in C} f_\theta(\boldsymbol{x})_c \neq \text{argmax}_c f_\theta(\boldsymbol{x}')_c$ rather than $y(\boldsymbol{x}) \neq y(\boldsymbol{x}')$ to represent that $\boldsymbol{x}$ and $\boldsymbol{x}'$ are on the different sides of the decision boundary. [58] proved that $\exists \Delta_i \in \Delta \cdot I$, $y(x \pm \Delta_i) \neq y(x)$ is a sufficient condition for x to be a sensitive query, where I is the identity matrix, Δ_i is the projected interval on some dimension i, x and $x \pm \Delta_i$ are identical in dimensions other than i and differ Δ in dimension i. Following this, we formally define our ε-Soft Boundary Privacy (ε-SBP) and privacy-preserving constraint, BoundaryFuzz, against MS attacks, and explain how BoundaryFuzz leads to ε-SBP.

Definition 1 (ε-Soft Boundary Privacy (ε-SBP))**.** A classifier F_θ achieves ε-soft boundary privacy, if and only if for any three queries $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ in the boundary-sensitive zone Z_Δ, the following

SP of Target Model	Defense	Test Performance		CA (%)	Decrease in Substitute Model's F1 (%)				
		Acc (%)	F1 (%)		Uniform	Gaussian	LineSearch	JBA(P)	JBA(X)
Lipschitznessness(T)	PredictionPoisoning	96.15	82.14	–.–	0.64	3.48	1.05	2.27	**4.32**
	AdaptiveMisinformation	94.99	75.70	–.–	2.66	3.02	1.88	**2.66**	1.31
	BoundaryFuzz	**96.24**	**84.46**	**94.52**	**6.6**	**5.44**	**3.29**	0.36	3.36
Lipschitznessness	PredictionPoisoning	96.23	83.76	–.–	3.80	7.44	3.00	1.19	6.40
	AdaptiveMisinformation	94.99	78.97	–.–	4.27	6.84	2.24	1.25	7.20
	BoundaryFuzz	**96.34**	**84.55**	**95.17**	**8.80**	**9.80**	**3.85**	**2.52**	**7.58**
LocalInvariance(T)	PredictionPoisoning	96.23	83.71	–.–	1.94	1.29	0.20	4.03	4.49
	AdaptiveMisinformation	94.89	76.65	–.–	**2.41**	**3.96**	0.94	**5.71**	**5.03**
	BoundaryFuzz	**96.24**	**84.08**	**95.38**	1.44	3.83	**3.02**	5.29	4.74
LocalInvariance	PredictionPoisoning	96.15	83.19	–.–	0.70	1.11	1.53	2.97	1.81
	AdaptiveMisinformation	94.89	78.01	–.–	0.60	1.18	0.64	5.69	3.82
	BoundaryFuzz	**96.24**	**84.46**	**94.13**	**2.11**	**2.75**	**2.53**	**8.12**	**8.14**
SmallNeighborhood	PredictionPoisoning	**98.60**	**90.91**	–.–	1.35	1.69	0.61	1.01	0.41
	AdaptiveMisinformation	97.67	83.87	–.–	1.35	1.69	0.61	1.01	0.41
	BoundaryFuzz	97.10	87.24	**100.00**	**2.61**	**2.35**	**1.24**	**7.71**	**4.91**
Monotonicity	PredictionPoisoning	**98.60**	**89.66**	–.–	1.53	0.72	0.43	1.30	1.30
	AdaptiveMisinformation	97.67	81.48	–.–	1.53	0.72	0.43	1.30	1.30
	BoundaryFuzz	97.01	87.24	**100.00**	**2.61**	**2.35**	**1.24**	**7.71**	**4.94**

Table 6: Model Stealing Defense effectiveness on Medical Insurance Dataset. CA stands for the constraint accuracy of the privacy property.

inequality always holds for the label outputs $y(\boldsymbol{x}_1)$, $y(\boldsymbol{x}_2)$, and $y(\boldsymbol{x}_3)$.

$$e^{-\varepsilon} \leq \frac{\Pr[y(\boldsymbol{x}_1), y(\boldsymbol{x}_2) | \boldsymbol{x}_1 \parallel \boldsymbol{x}_2]}{\Pr[y(\boldsymbol{x}_1), y(\boldsymbol{x}_3) | \boldsymbol{x}_1 \nparallel \boldsymbol{x}_3]} \leq e^{\varepsilon},$$

where $\boldsymbol{x} \parallel \boldsymbol{x}'$ is short for $\text{argmax}_c f_\theta(\boldsymbol{x})_c = \text{argmax}_c f_\theta(\boldsymbol{x}')_c$ and $\boldsymbol{x} \nparallel \boldsymbol{x}'$ is short for $\text{argmax}_c f_\theta(\boldsymbol{x})_c \neq \text{argmax}_c f_\theta(\boldsymbol{x}')_c$. ε-SBP guarantees that the model's responses on two sensitive queries are indistinguishable for the adversary to determine the true decision boundary.

Property 5 (BoundaryFuzz).

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \underset{c}{\text{argmax}}\, f_\theta(\boldsymbol{x})_c \neq \underset{c}{\text{argmax}}\, f_\theta(\boldsymbol{x} + \Delta \cdot I)_c$$
$$\Rightarrow ||F_\theta(\boldsymbol{x})||_2 < \sqrt{\frac{e^\varepsilon}{e^\varepsilon + 1}}$$

Theorem 1. *Models with* BoundaryFuzz *property satisfy ε-SBP when C=2.*

We detail the proof of Theorem 3 in Appendix 9.7. This can be generalized to cases where $C > 2$ (i.e., multi-label classification) as shown in Appendix 9.8.

5.2 Evaluation

We apply DL2 [13] and LogicEnsemble [10] to train BoundaryFuzz along with security properties and evaluate the defense effectiveness against MaskedThief.

Implementation. To train BoundaryFuzz along with LocalInvariance and Lipschitzness, we rewrite BoundaryFuzzfollowing the definition of the security properties in DL2 [13] :

$$\forall \boldsymbol{x} \in \mathcal{X}_{\text{tr}}, \forall \tilde{\boldsymbol{x}} \in \mathcal{B}_{p,\epsilon}\boldsymbol{x} \quad \underset{c}{\text{argmax}}\, f_\theta(\tilde{\boldsymbol{x}})_c \neq \underset{c}{\text{argmax}}\, f_\theta(\tilde{\boldsymbol{x}} + \Delta \cdot I)_c$$
$$\Rightarrow ||F_\theta(\tilde{\boldsymbol{x}})||_2 < \sqrt{\frac{e^\varepsilon}{e^\varepsilon + 1}}$$

where $\forall x \in \mathcal{X}_{tr}, \forall \tilde{x} \in \mathcal{B}_{p,\epsilon}(x)$ indicates moderate generalizability.

To train BoundaryFuzz along with SmallNeighborhood and Monotonicity, we rewrite BoundaryFuzzfollowing the definition of the security properties in LogicEnsemble [10] :

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \quad \underset{c}{\text{argmax}}\, f_\theta(\boldsymbol{x})_c \neq \underset{c}{\text{argmax}}\, f_\theta(\boldsymbol{x}' + \Delta \cdot I)_c$$
$$\Rightarrow ||F_\theta(\boldsymbol{x})||_2 < \sqrt{\frac{e^\varepsilon}{e^\varepsilon + 1}} \wedge ||F_\theta(\boldsymbol{x}')||_2 < \sqrt{\frac{e^\varepsilon}{e^\varepsilon + 1}}$$

where $\forall x, x' \in \mathcal{X}$ indicates high generalizability. Note that BoundaryFuzz cannot be trained with low generalizability because it cannot be enforced on the training data.

Settings. For security properties, we use the same constraint parameters as in §3. We select the best values for the parameters in BoundaryFuzz to train with each security property on each dataset. The best value of the parameter is determined as the one that involves the largest proportion of samples near the decision boundary and forces the largest degree of uncertainty in their predictions. In our implementation, we first set $\Delta = 0.1, \varepsilon = 0.001$ and train BoundaryFuzz with each security property on the target models. If all properties are successfully trained, we double the value of Δ to involve more suspicious samples to the privacy constraint. Otherwise, we reduce the value of Δ by half to loosen the constraint. If the properties cannot be trained until Δ reaches 0.01, we increase the value of ε to further loose the constraint. In particular,

Target Model	Defense	Test Performance		CA (%)	Decrease in Substitute Model's F1 (%)				
		Acc (%)	F1 (%)		Uniform	Gaussian	LineSearch	JBA(P)	JBA(X)
Lipschitzness(T)	PredictionPoisoning	97.60	97.12	–.–	2.37	1.64	**3.95**	0.77	0.64
	AdaptiveMisinformation	97.86	97.63	–.–	2.66	1.87	2.92	0.35	0.57
	BoundaryFuzz	**98.23**	**98.01**	**96.24**	**3.54**	**4.31**	3.89	**2.75**	**2.42**
Lipschitzness	PredictionPoisoning	96.36	96.12	–.–	1.02	0.94	2.25	1.23	0.94
	AdaptiveMisinformation	96.43	96.25	–.–	0.98	1.58	1.45	1.34	1.29
	BoundaryFuzz	**97.02**	**96.53**	**98.56**	**3.25**	**4.31**	**4.67**	**3.21**	**3.35**
LocalInvariance(T)	PredictionPoisoning	92.68	93.07	–.–	1.05	1.79	0.19	1.01	1.43
	AdaptiveMisinformation	97.88	97.45	–.–	167	2.31	0.99	**2.92**	**3.04**
	BoundaryFuzz	**98.39**	**98.19**	**97.50**	**3.32**	**2.96**	**2.03**	2.37	2.94
LocalInvariance	PredictionPoisoning	97.66	97.25	–.–	1.46	4.77	4.12	2.09	1.94
	AdaptiveMisinformation	92.47	92.95	–.–	2.01	3.37	3.12	1.98	2.64
	BoundaryFuzz	**98.12**	**97.84**	**99.89**	**4.14**	**5.23**	**4.58**	**3.41**	**3.29**
SmallNeighborhood	PredictionPoisoning	97.57	97.21	–.–	1.94	3.05	5.49	1.63	1.78
	AdaptiveMisinformation	97.43	97.08	–.–	2.14	3.17	5.6	1.34	1.99
	BoundaryFuzz	**98.06**	**97.82**	**100.00**	**4.25**	**4.12**	**5.80**	**2.31**	**3.16**
Monotonicity	PredictionPoisoning	96.21	95.98	–.–	1.65	3.44	2.70	3.69	2.85
	AdaptiveMisinformation	90.23	90.58	–.–	1.86	3.13	2.32	2.15	1.16
	BoundaryFuzz	**94.45**	**94.10**	**100.00**	**4.50**	**4.75**	**5.21**	**4.39**	**4.77**

Table 7: Model Stealing Defense effectiveness on Spam URL Dataset. CA stands for the constraint accuracy of the privacy property.

we multiply the value of ε by 10 each time until it reaches 0.1. For LocalInvariance and Lipschitzness, we tune the value of ϵ using the same method as for Δ while starting from 1 with step size 1. The specific values of the constraint parameters are detailed in Appendix 9.2.

Results. Similar to §3, we evaluate the decrease in the substitute model's test F1 when applying the defense. All results are averaged on five independent tests. We also measure the test accuracy and test F1 of the defended target model to compare the performance drop in classification caused by different defense methods.

Results on the Medical Insurance dataset are presented in Table 6. In most cases, our method leads to the highest F1 decrease of substitute models. Specifically, for models with Lipschitzness, LocalInvariance, SmallNeighborhood, and Monotonicity, BoundaryFuzz outperforms other defense methods against all attacks. For models with Lipshitzness^T, AdaptiveMisinformation shows the best performance against JBA(P) and PredictionPoisoning shows the best performance against JBA(X). For models with LocalInvariance^T, AdaptiveMisinformation shows the best performance against Random Query attacks (i.e., Uniform and Gaussian), and Jacobian-based Augmentation attacks (i.e., JBA(P) and JBA(X)). We speculate that BoundaryFuzz is less effective on LocalInvariance^T and Lipshitzness^T for the security constraints and privacy-preserving constraints in this setting are both trained only on the training data. In this case, most synthesized samples are exclusive to the constraints.

In addition, we observe that BoundaryFuzz causes a moderate drop in the classification performance. For Lipschitzness^T, Lipschitzness, LocalInvariance^T, and LocalInvariance, models defended by BoundaryFuzz achieve the highest test accuracy and f1. Instead of directly perturbing model outputs, BoundaryFuzz searches for samples near the decision boundary through robust training methods and enforcing their predictions to be uncertain, which might result in less performance drop. However, BoundaryFuzz causes the maximum performance drop for SmallNeighborhood and Monotonicity. We consider the reason as training global security properties and global privacy-preserving properties simultaneously is more difficult. However, the performance drop of BoundaryFuzz is still acceptable compared to other defense methods. The defense on the Credit Card dataset and the Spam Account dataset show similar results, which are presented in Table 14 and Table 7 in Appendix 9.10. In summary, our defense method can easily fit into robust training methods to achieve a promising trade-off between accuracy, security, and privacy. To evaluate the impact of BoundaryFuzz on the original security properties, we present the constraint accuracy of the security properties after incorporating BoundaryFuzz into the robust training process. Specifically, as LogicEnsemble trains models with global security properties [10], the constraint accuracy of SmallNeighborhood and Monotonicity is both 100%. The constraint accuracy of LocalInvariance and Lipschitzness is shown in Table 8.

Property/Dataset	MI	CC	SA	SU	CJ
Lipschitzness(T)	82.98	93.86	74.04	80.64	78.06
Lipschitzness	89.72	90.03	62.23	87.12	82.14
LocalInvariance(T)	85.4	90.1	95.93	97.2	100
LocalInvariance	91.55	95.27	96.39	98.21	100

Table 8: Constraint accuracy (%) of security properties trained with BoundaryFuzz.

6 RELATED WORKS

Previous works have investigated the influencing factors of MS attacks. Steal-ML [49] proposed equation-solving and path-finding attacks and evaluated their sensitivity to the degree of the precision (i.e., rounding decimals) of the target model's predictions. PRADA [20] and SEAT [57] evaluated Jacobian-based attacks [39] and LineSearch attacks [49] and demonstrated that: 1) Jacobian-based attacks are the most effective method when the attacker has access to a small set of samples from the original training set, and 2) more natural seed samples results in higher prediction agreement. In addition, [20] claimed the necessity of natural seed samples to train a substitute model with a high agreement. It also concluded that cross-validated hyperparameters outperform heuristics or the same settings of the target model, and similar architecture complexity between the target and the substitute model yields high predictive performance. ML-Doctor [30] performed a systematic evaluation on four privacy attacks (i.e., membership inference, model inversion attribute inference, and model stealing) and investigated the common factors that influence their performance. It observed that data complexity is negatively associated with model stealing performance. Also, using a part of training data as seed samples results in worse model stealing performance than using an external natural dataset which has a similar distribution with the training data. The author explained this as the query results of training data lead to more confident posteriors which contain less information for the adversary to exploit.

The closest work to our study are [11, 29, 46], which investigated privacy impacts on model robustness. However, all of them focus on membership inference (MI) attacks instead of MS attack. Song et al. [46] revealed that adversarially robust models can be more vulnerable to MI since the smooth predictions around training examples may not generalize well to testing examples. Choquette-Choo et al. [11] proposed label-only MI by exploiting the robustness of a model's predicted labels under perturbations to obtain a fine-grained membership signal. Following [11], Li et al. [29] relaxed the assumptions of the adversary's knowledge (e.g., training algorithm, shadow dataset, etc.) in [11] and proposed a threshold-choosing method for boundary attack. In [11, 29], the attacker can identify the membership of the sample in a label-only scenario due to the train-test gap between model robustness around training and testing points. All three work indicate that robust models suffer from additional membership inference risks due to *poor robustness generalization*. Experiments in [11, 29, 46] demonstrate such risks can be mitigated using existing regularization techniques, such as domain-adaptation-based regularization [45], which can improve robustness generalizability.

Different from [11, 29, 46], in this paper, we explore the model stealing risks of robust models enforcing security properties. Our results reveal that models trained with security properties, especially those with *high robustness generalizability* instead of poor generalizability in MI attacks, suffer from larger model stealing risks. In addition, our study first-time proposes a privacy property which is compatible with security properties for ML models to mitigate such risks.

7 CONCLUSION

In this paper, we investigate privacy impacts on four security properties (i.e., LocalInvariance, Lipschitzness, SmallNeighborhood and Monotonicity) of robust ML models on five real-world security datasets. By measuring the success of three typical MS attacks (in five variants) and two state-of-the-art MS defenses on robust models trained with four security properties, we find that all robust models are more vulnerable to MS attacks than their natural models. Even worse, MS defenses are less effective on those robust models. This is because the security property of the target model results in more stable and smooth predictions, which enhances the query effectiveness of MS attackers and hinders MS defenders from distinguishing suspicious samples from benign ones. Given the above findings, we propose MaskedThief, an optimized MS attack specific to robust models, which improves existing MS attacks by exploiting information in SPs. To mitigate such privacy risk on models trained with security properties, we design a privacy property, BoundaryFuzz, which enforces the model predictions on inputs near the decision boundary to be uncertain. Experimental results demonstrate the defense effectiveness of BoundaryFuzz, which sheds lights on ML models satisfying goals in accuracy, security, and privacy altogether.

8 ACKNOWLEDGMENT

We appreciate Dr. Yizheng Chen and Dr. Tianhao Wang for their valuable discussions and feedback. We thank the anonymous reviewers for their constructive suggestions. This work is supported in part by the NSF CNS-1850725, the Indiana University Institute for Advanced Study (IAS) and the Grant Thornton Institute.

REFERENCES

[1] 2019. Medical Insurance Dataset. https://www.kaggle.com/datasets/rajgupta2019/medical-insurance-dataset.

[2] 2022. Customer Churn Dataset. https://www.kaggle.com/code/yejiseoung/building-gradient-boosting-pipeline-0-99-roc-auc/data.

[3] 2023. Code, dataset, and full version paper. https://sites.google.com/view/maskedthief/home.

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems* 32 (2019).

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

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

[7] Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. 2021. Property inference from poisoning. *arXiv preprint arXiv:2101.11073* (2021).

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

[9] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2* 1, 4 (2015), 1–4.

[10] Yizheng Chen, Shiqi Wang, Yue Qin, Xiaojing Liao, Suman Jana, and David Wagner. 2021. Learning security classifiers with verified global robustness properties. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 477–494.

[11] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International Conference on Machine Learning*. PMLR, 1964–1974.

[12] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. 2018. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[13] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. 2019. Dl2: Training and querying neural networks with logic. In *International Conference on Machine Learning*. PMLR, 1931–1941.

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

[15] Xinlei He, Hongbin Liu, Neil Zhenqiang Gong, and Yang Zhang. 2022. Semi-Leak: Membership Inference Attacks Against Semi-supervised Learning. In *European Conference on Computer Vision*. Springer, 365–381.

[16] Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*. Springer, 409–426.

[17] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2020. High accuracy and high fidelity extraction of neural networks. In *29th USENIX security symposium (USENIX Security 20)*. 1345–1362.

[18] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. 2020. Entangled watermarks as a defense against model extraction. *arXiv preprint arXiv:2002.12200* (2020).

[19] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, Hongbin Liu, and Neil Zhenqiang Gong. 2020. Almost tight L0-norm certified robustness of top-k predictions against adversarial perturbations. *arXiv preprint arXiv:2011.07633* (2020).

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

[21] Sanjay Kariyappa and Moinuddin K Qureshi. 2020. Defending against model stealing attacks with adaptive misinformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 770–778.

[22] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. 2018. Model extraction warning in mlaas paradigm. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 371–380.

[23] Amin Kharraz, Zane Ma, Paul Murley, Charles Lever, Joshua Mason, Andrew Miller, Nikita Borisov, Manos Antonakakis, and Michael Bailey. 2019. Outguard: Detecting in-browser covert cryptocurrency mining in the wild. In *The World Wide Web Conference*. 840–852.

[24] Heeyoung Kwon, Mirza Basim Baig, and Leman Akoglu. 2017. A domain-agnostic approach to spam-url detection via redirects. In *Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part II 21*. Springer, 220–232.

[25] Steven M LaValle, Michael S Branicky, and Stephen R Lindemann. 2004. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research* 23, 7-8 (2004), 673–692.

[26] Peter D Lax and Maria Shea Terrell. 2014. *Calculus with applications*. Springer.

[27] K Lee, BD Eoff, and J Caverlee. 2011. A long-term study of content polluters on twitter. *ICWSM, seven months with the devils* (2011).

[28] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. 2019. Defending against neural network model stealing attacks using deceptive perturbations. In *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 43–49.

[29] Zheng Li and Yang Zhang. 2021. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 880–895.

[30] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. 2022. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, Kevin R. B. Butler and Kurt Thomas (Eds.). USENIX Association, 4525–4542. https://www.usenix.org/conference/usenixsecurity22/presentation/liu-yugeng

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[32] Shagufta Mehnaz, Ninghui Li, and Elisa Bertino. 2020. Black-box model inversion attribute inference attacks on classification models. *arXiv preprint arXiv:2012.03404* (2020).

[33] Matthew Mirman, Timon Gehr, and Martin Vechev. 2018. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*. PMLR, 3578–3586.

[34] Fionn Murtagh. 1991. Multilayer perceptrons for classification and regression. *Neurocomputing* 2, 5-6 (1991), 183–197.

[35] Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. 2022. I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences. *arXiv preprint arXiv:2206.08451* (2022).

[36] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4954–4963.

[37] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2020. Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=SyevYxHtDB

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

[39] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.

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

[41] Gabriel Ryan, Justin Wong, Jianan Yao, Ronghui Gu, and Suman Jana. 2019. CLN2INV: learning loop invariants with continuous logic networks. *arXiv preprint arXiv:1909.11542* (2019).

[42] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246* (2018).

[43] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[44] Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. 2017. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571* (2017).

[45] Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. 2018. Improving the generalization of adversarial training with domain adaptation. *arXiv preprint arXiv:1810.00740* (2018).

[46] Liwei Song, Reza Shokri, and Prateek Mittal. 2019. Privacy Risks of Securing Machine Learning Models against Adversarial Examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM, 241–257. https://doi.org/10.1145/3319535.3354211

[47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[49] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*. 601–618.

[50] Binghui Wang and Neil Zhenqiang Gong. 2018. Stealing hyperparameters in machine learning. In *2018 IEEE symposium on security and privacy (SP)*. IEEE, 36–52.

[51] Eric Wong and Zico Kolter. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*. PMLR, 5286–5295.

[52] Xiaojun Xu, Linyi Li, and Bo Li. 2022. LOT: Layer-wise Orthogonal Training on Improving l2 Certified Robustness. *NeurIPS* (2022).

[53] Zhuolin Yang, Zhikuan Zhao, Boxin Wang, Jiawei Zhang, Linyi Li, Hengzhi Pei, Bojan Karlaš, Ji Liu, Heng Guo, Ce Zhang, and Bo Li. 2022. Improving Certified Robustness via Statistical Learning with Logical Reasoning. *NeurIPS* (2022).

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

[55] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[56] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 159–172.

[57] Zhanyuan Zhang, Yizheng Chen, and David Wagner. 2021. Seat: Similarity encoder by adversarial training for detecting model extraction attack queries. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*. 37–48.

[58] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. 2019. Bdpl: A boundary differentially private layer against machine learning model extraction attacks. In *European Symposium on Research in Computer Security*. Springer, 66–83.