Fan *et al. Cybersecurity* (2023) 6:35
https://doi.org/10.1186/s42400-023-00171-y

Cybersecurity

RESEARCH

Open Access

FMSA: a meta-learning framework-based fast model stealing attack technique against intelligent network intrusion detection systems

Kaisheng Fan[1], Weizhe Zhang[1,2*], Guangrui Liu[1] and Hui He[1]

Abstract

Intrusion detection systems are increasingly using machine learning. While machine learning has shown excellent performance in identifying malicious traffic, it may increase the risk of privacy leakage. This paper focuses on implementing a model stealing attack on intrusion detection systems. Existing model stealing attacks are hard to implement in practical network environments, as they either need private data of the victim dataset or frequent access to the victim model. In this paper, we propose a novel solution called Fast Model Stealing Attack (FMSA) to address the problem in the field of model stealing attacks. We also highlight the risks of using ML-NIDS in network security. First, meta-learning frameworks are introduced into the model stealing algorithm to clone the victim model in a black-box state. Then, the number of accesses to the target model is used as an optimization term, resulting in minimal queries to achieve model stealing. Finally, adversarial training is used to simulate the data distribution of the target model and achieve the recovery of privacy data. Through experiments on multiple public datasets, compared to existing state-of-the-art algorithms, FMSA reduces the number of accesses to the target model and improves the accuracy of the clone model on the test dataset to 88.9% and the similarity with the target model to 90.1%. We can demonstrate the successful execution of model stealing attacks on the ML-NIDS system even with protective measures in place to limit the number of anomalous queries.

Keywords AI security, Model stealing attack, Network intrusion detection, Meta learning

Introduction

Deep neural networks (DNN) and machine learning (ML) have received much attention. In a variety of fields, like as image classification (Touvron et al. 2021), autonomous driving (Kiran et al. 2021), and natural language processing (Brown et al. 2020), they have achieved significant strides. ML has also proved great potential in security-sensitive areas like network intrusion detection system that uses machine learning techniques (ML-NIDS) (Goryunov et al. 2020). NIDS is used to detect malicious behavioral activities usually generated by malware. ML has improved the accuracy of malware detection, which is better suited to detect sophisticated cyber-attacks than traditional methods. However, deploying these methods might be threatened by attacks (Rüping et al. 2022) against ML models that raise privacy and security risks, especially in security-sensitive areas.

Motivation

Although ML-NIDS can effectively defend against traditional network attacks, the security vulnerabilities of the algorithms are easily ignored. In particular, poisoning

*Correspondence:
Weizhe Zhang
wzzhang@hit.edu.cn
[1] School of Cyberspace Science, Harbin Institute of Technology, Harbin 150001, China
[2] Department of New Networks, Peng Cheng Laboratory, Shenzhen 518055, Guangdong, China

attacks (Truong et al. 2020) can poison the model and give enemies the ability to interfere with model decisions; adversarial samples (Aldahdooh et al. 2022) crafted under malicious perturbation cause models to compute false predictions; model stealing attacks (Orekondy et al. 2019) cause leakage of information about the parameters of models. In addition, various security threats, such as model reverse engineering (Oh et al. 2019), membership inference attacks (Hu et al. 2022), and backdoor attacks (Wang et al. 2019), can lead to severe consequences. Among them, once stolen, AI models, as the core carrier of the technology, will be exposed to risks, which can cause the enterprise or organization with the technology to suffer huge losses. The model stealing attack also will be a stepping stone to subsequent effective attacks (Mahmood et al. 2021) against the target model.

We focus on model stealing attacks against ML-NIDS. With just black-box access to the victim model, the model stealing attack enables the adversary to train a clone model to replicate target prediction capabilities. Training an advanced ML model is often tricky when lacking data or computational resources, and low-cost stealing of trained models constitutes intellectual property theft (Chen et al. 2022a).

Challenge

There are three key obstacles to conducting undetected model stealing attacks against ML-NIDS.

Challenge 1 No access to the victim ML-NIDS training dataset. Most existing works (Juuti et al. 2019; Kesarwani et al. 2018) underrate the effects of model stealing attacks and assume that a significant quantity of training data or supplementary information about the victim is required for a successful theft. It is a challenge to conduct model stealing attacks effectively with little to no access to private data knowledge of the victim model.

Challenge 2 The NIDS has a strict limit on the number of input queries. Unlike previous model stealing attacks have unlimited access to victim models, malicious input queries are detected because of the sensitivity security of NIDS. We need to carry out the model stealing attack with very few queries without being detected by ML-NIDS.

Challenge 3 Existing works do not evaluate the degree to which trained models are at risk from model stealing attacks and underestimate the threat posed by model stealing attacks. The benefit of an ML-NIDS model stealing attack is to obtain a clone model that closely resembles the target NIDS. The adversary may use the replica model's white-box access to launch additional attacks. It is a challenge in our future work on the assessment and defense of model stealing.

Contributions

In this work, we propose a fast model stealing attack method against ML-NIDS. We built on the perspective of meta-learning and few-shot learning (Vanschoren 2018; Wang et al. 2020; Sun et al. 2019), which requires only a small number of samples. The objective of the model in meta-learning is to swiftly learn a new task from a limited quantity of data. Model Agnostic Meta Learning (MAML) (Finn et al. 2017) is an approach to few-shot meta-learning that is commonly utilized. We also introduce an adversarial training framework for the clone and victim models to learn the predictive performance of the victim ML-NIDS. Figure 1 shows the methods necessary to carry out a quick model stealing attack.

Due to the difficulty of acquiring data knowledge from ML-NIDS and the strict restrictions on access queries, few model stealing attacks against ML-NIDS have been successfully executed. Our proposed approach is the first practical model stealing method for ML-NIDS scenarios. We show that only a small amount of auxiliary knowledge and queries are required to make the clone model as similar to the victim as possible. Moreover, we can still craft adversarial samples with stolen models compared with data-free model stealing attacks. Due to the transferability property of adversarial samples, we demonstrate that these samples can evade the detection of the victim ML-NIDS. Briefly stated, our primary contributions include:

Contribution 1 We first introduce a meta-learning framework to model stealing attacks and design an adversarial learning strategy for cloning models and generators. The meta-learning algorithm allows the adversary to converge the clone model after a few gradient updates

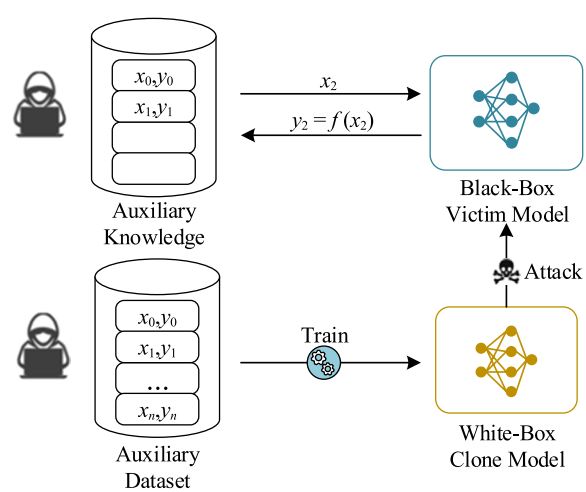

Fig. 1 Model stealing attack using auxiliary knowledge and query samples to the victim model, and subsequently evasion attacks on the victim model based on unlimited access to the white-box clone model

and can successfully steal models with little knowledge of the private data.

Contribution 2 Our method uses the quantity of model access requests as a constraint for successful model stealing attacks for the first time, which is more realistic than the unlimited queries to the victim in previous approaches.

Contribution 3 We combine an adversarial learning model stealing approach with a data synthesis approach approximating the victim data. Our proposed method requires only a small amount of auxiliary knowledge to successfully attack and demonstrate that the clone model facilitates subsequent evasion attacks on the victim ML-NIDS.

Related work

Network intrusion detection system (NIDS)

NIDS detects behaviors that compromise the security of computer systems (Yang et al. 2022). NIDS is generally deployed in the network nodes of the intranet, and all network requests flow through these network nodes. Machine learning, in particular deep learning techniques, has enabled NIDS to detect a variety of cyberthreats. Usually, NIDS is unavailable to the adversary, deployed inside the system, and the adversary can only get its classification results but not internal information. In our work, we take ML-NIDS as an attack target model only with black-box access and limit access to the training data as well as the query quantity.

Model stealing attack

Model stealing attack (Orekondy et al. 2019; Mahmood et al. 2021), also known as model extraction attack, aims to extract a replica of a black-box victim model. The purpose of the model stealing attack is to train a clone model that performs similarly to the target model using the limited information related to the target model. For example, enterprises must spend a lot of time, money, and workforce training ML models. If an adversary can steal the model, it saves lots of computing resources and seriously threatens intellectual property security.

The model stealing attack will also serve as a springboard for more powerful attacks to come, such as adversarial attacks that need access to the target ML model's white-box (Liu et al. 2021). Recent works have proposed three separate types of model stealing attacks, which are categorized based on the attack target: (1) theft of function (Kariyappa et al. 2021), which aims to imitate the target model's output forecasts in a clone model; (2) theft of parameters (Rakin et al. 2022; Tramèr et al. 2016) aims to obtain information such as the intermediate gradient of the model; (3) theft of hyper-parameters (Wang and Gong 2018; Oh et al. 2019) focuses on getting the hyper-parameters involved in the target model's training algorithm's model architecture. In the current landscape, the majority of model function stealing techniques necessitate a substantial number of queries to the target model for gathering sufficient information to construct a replica. In contrast, our approach adopts a few-shot learning method, seeking to achieve model function stealing of a black-box target model within the constraints of limited query instances. By leveraging this methodology, we gain insights into private data information through probability prediction vectors.

Model stealing attacks can also be divided into three categories, depending on the access to the victim's privacy data: (1) Data with partial privacy (Papernot et al. 2017), adversary's predictions of partial private data from the victim are used to train a clone model. Nonetheless, it implies that a certain level of familiarity with the data distribution of the victim model is essential, making it impractical for universal application. (2) Data with privacy-related auxiliary knowledge (Orekondy et al. 2019), the adversary lacks direct access to personal information but can acquire auxiliary knowledge similar to the victim training dataset and comes from a distinct task domain. Unlike the scenario of partial privacy data, privacy-related auxiliary data alone is insufficient to facilitate the training of cloned models. Additional techniques such as data synthesis are required to carry out model stealing attacks. This approach aligns more closely with real-world situations, especially when targeting victims who have implemented privacy protection measures on their private data. (3) Data-free (Roberts et al. 2019; Sanyal et al. 2022), the adversary does not have access to victim knowledge but introduces an adversarial learning framework that eliminates the differences between the output probability vectors of the clone and victim models. This indicates that the adversary is capable of extensively searching the entire feature space in order to replicate the output of the target model, even without any prior knowledge of the private data. However, it needs to consider the real distribution of the data and requires many training steps to converge. Our proposed approach utilizes a small amount of auxiliary knowledge with adversarial learning and, for the first time, combines meta-learning with model stealing attacks, aiming at having the meta-learner automatically learn how to steal, i.e., the meta-learning idea of learning to learn.

Meta-learning and few-shot learning

While performing well on multiple tasks, machine learning and deep learning cannot adapt to new tasks as quickly as humans can based on a minimal amount

of prior knowledge. Inspired by the fast learning ability of humans, researchers want machine learning models to learn quickly with only a limited number of samples for new classes after learning a vast amount of tasks and data, hence the proposal of meta-learning (Tian et al. 2022), or learning to learn. The benefit of employing meta-learning lies in its ability to dynamically adjust to different tasks and datasets by utilizing learning algorithms. This adaptation enhances the efficiency and accuracy of the learning process. In the area of supervised learning, few-shot learning (Wang et al. 2020) is a meta-learning application. It tackles intricate classification and regression problems using a limited number of training samples, addressing challenges commonly encountered in traditional machine learning and deep learning, such as overfitting issues.

In our scenario, we set the meta-stealer to go through several different scenarios of stealing tasks and finally apply it to the target model. Model-agnostic meta-learning (MAML) (Finn et al. 2017) and prototypical networks (Snell et al. 2017) are two more representative few-shot learning algorithms. MAML is designed to teach models how to initialize parameters for a specific task so that they can converge quickly after only a few samples of training, and the underlying assumption of the prototypical network is that there exists a low-latitude embedding space in which the sample distributions of all categories are far from each other. Both methods have advantages in facilitating knowledge transfer across diverse tasks and datasets, as they enhance the adaptability and generalization capability of the learning process. In our study, we adopt the MAML algorithm to guide the training of our meta-learners and incorporate the concept of prototypical networks to extract features from limited auxiliary information, promoting the generation of more representative query samples.

Generative adversarial network

In some generative tasks, such as the generation of images (Bao et al. 2017) and medical cases (Chen et al. 2022b), generative adversarial networks (Goodfellow et al. 2020) have made substantial progress. The generative adversarial network consists of a generator and a discriminator. The discriminator attempts to discriminate between authentic data and fake data generated by the generator in an adversarial game. In contrast, the generator aims to confuse the discriminator. DCGAN (Yang et al. 2019) and WGAN (Gulrajani et al. 2017) are two significant developments in generative adversarial networks. DCGAN is a modified version of GAN that primarily enhances the image generation quality by utilizing deep convolutional neural networks. On the other hand, WGAN is an alternative approach to GAN that employs Wasserstein distance for measuring the discrepancy between the generator and discriminator, effectively addressing issues such as gradient vanishing and mode collapse. In our research, we leverage the advantages of Wasserstein distance and deep convolutional neural networks to enhance the stability and quality of GAN-based sample generation, thus enabling our model stealing attacks.

Method

This section outlines the attack scenario and proposes our model stealing attack method.

Attack scenario

In this paper, we emphasize ML-NIDS, one of the applications of ML in security-sensitive scenarios. The goal of an ML model is to generally map data samples to the appropriate class to which they belong. The input to an ML-NIDS model is a session-based traffic sample. The output is a vector of probabilities called posterior probability, with each dimensional element representing the probability that the input sample belongs to a specific category. ML-NIDS classifies input as the daily benign label or a certain malware label.

Availability of the Victim Model. We think about a black-box access configuration. In black-box attacks, adversaries can not have complete information about the victim model, including its architecture and parameters. They are limited to merely querying and receiving predictions from the victim model. Furthermore, due to the sensitivity security of ML-NIDS, There is a hard cap on how many times the adversary may query the model rather than having an unlimited number of queries times.

Auxiliary Dataset. We consider having minimal auxiliary knowledge. Comparatively to having real data from the victim dataset to train a clone model, the adversary can only mimic the distribution of the victim model's training data using limited auxiliary knowledge. In our work, we set to compute a prototype representative (Snell et al. 2017) of each category using each one or few-shot samples by a meta-encoder and then generate samples around the prototype representative using a trained meta-generator.

Attack Object. We assume that a victim model V can accurately categorize malware traffic. Our objective is to train a clone model C to achieve high accuracy on the victim test dataset D_{test} and to approach the categorization capability of the victim model. The adversary does not have access to D_{test}, just only for test evaluation purposes. In our attack method, we attempt to approximate the probability vector of the victim model's output with the output of the clone model, which allows us to recover some useful privacy information.

FMSA

Overview

The FMSA method is a fast model stealing attack based on a meta-learning framework. Unlike existing works, our method employs auxiliary information to synthesize the data rather than a significant quantity of victim training data. After training on a large number of different tasks, models can converge fast after a few gradient updates. The meta-learner may quickly generalize and provide synthetic data that fits the distribution using just a little auxiliary information. In addition, we set up a framework for adversarial learning between the generator and the clone model. Figure 2 depicts the entire framework diagram.

Meta-training and meta-testing are the two key steps that make up the method. In the former phase, the adversary attempts to steal a simulated victim model, while the latter is cloning from an actual victim. We assume the model f_θ is parameterized by meta-learner θ. θ guides the training of f_θ, and the performance of f_θ is fed back to facilitate the training of θ. With just a few gradient steps, the meta-learning architecture may swiftly adapt to a new task.

Let $p(\mathcal{T})$ be a classify task such that a task $\mathcal{T}_i \sim p(\mathcal{T})$ is a collection of examples with labels for the victim model simulated under the current task. To further divide this data, we create a training set $\mathcal{T}_{i_{tr}}$ and a validation set $\mathcal{T}_{i_{val}}$, i.e., $\mathcal{T}_i = \{\mathcal{T}_{i_{tr}}, \mathcal{T}_{i_{val}}\}$. Moreover, we adhere to the fundamental principles of the N-way K-shot issue for the adversary. The adversary can only use ancillary knowledge containing K samples from each of N classes. A generative adversarial network built on the meta-learning framework serves as the model's central component. Our architecture is based on Wasserstein GAN due to its stability properties and can avoid mode collapse issues. It consists of (1) Stealer C for training a clone model by reducing the difference in output with the victim model under the same input, (2) Extractor E for extracting the prototype representation of each class, (3) Generator G using the prototype representative extracted by E as input to generate samples that match that prototype class while maximizing the difference between the victim and the clone model's output, and (4) Discriminator D for discriminating fake data generated by G. We set up meta-learners in each of the four modules.

Training the clone model

We first simulate a well-performed victim model V to train the clone model for task $\mathcal{T}_i$. In order to launch a successful theft, the adversary merely needed to choose a model with enough learning capacity. It means the adversary needs only a basic understanding of the architectural decisions made for the tasks that the victim is solving, not specific knowledge of the victim's architecture (e.g., recurrent neural networks are suitable for natural

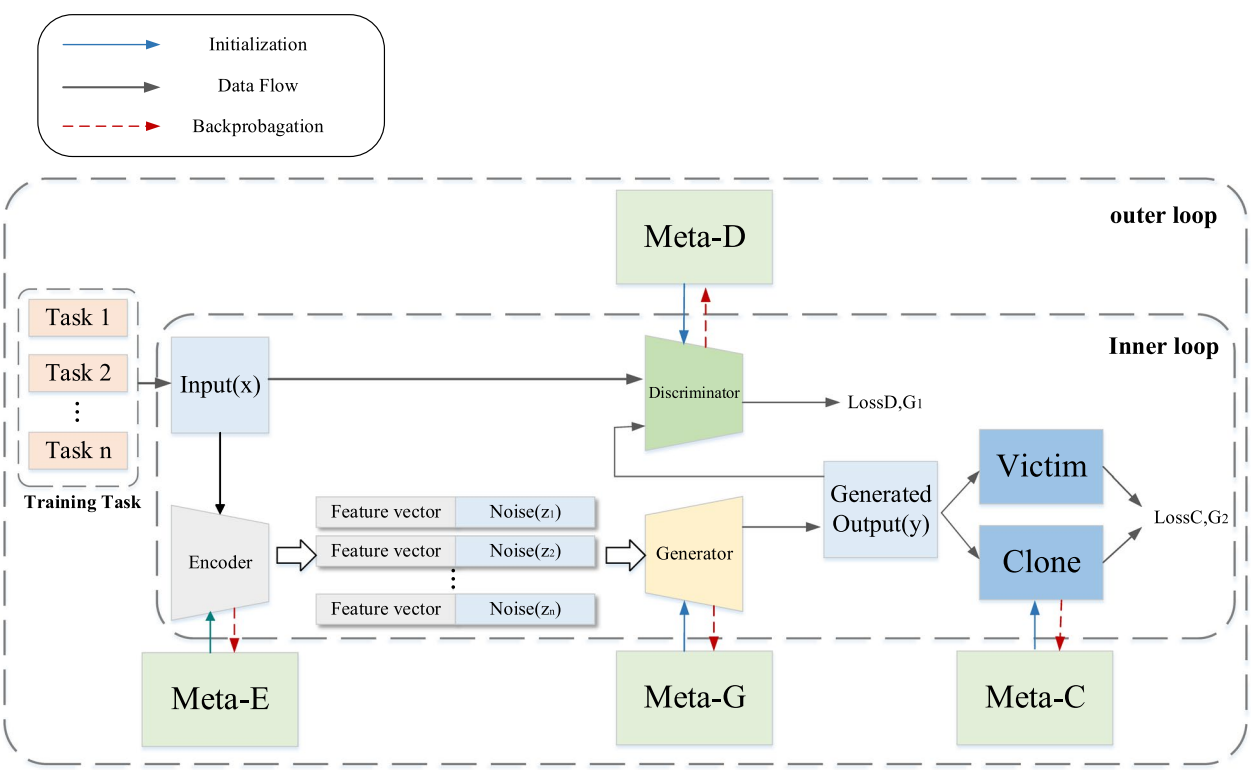

Fig. 2 Fast model stealing attack based on a meta-learning framework

language processing tasks). In our work, we choose DNN as a clone model structure and use the idea of MAML to initialize the clone model for different stealing tasks.

Clone model C initialized by meta-learner θ_c is trained using the synthetic samples x that generator G created. The I-dimensional input vector x is used to get the victim model and clone model output probabilities. Measurement of the discrepancy between the victim and clone model is done with the loss function $\mathcal{L}$. The KL divergence is chosen as the loss function. As demonstrated in Eq. 1, the clone model is trained by minimizing $\mathcal{L}$.

$$\begin{aligned} \mathcal{L}^C_{\mathcal{T}_i} &= \mathcal{L}_{KL}(V(x), C(x)) \\ \mathcal{L}_{KL}(V(x), C(x)) &= \sum_{i=1}^{I} C_i(x) \log(\frac{C_i(x)}{V_i(x)}) \end{aligned} \tag{1}$$

Algorithm 1 Overall FMSA Process.

Input:
θ_c:meta-learner for clone model C
θ_e:meta-learner for encoder E
θ_g:meta-learner for generator G
θ_d:meta-learner for discriminator D
Output: Trained C for real ML-NIDS victim
Initialise $\theta_c, \theta_e, \theta_g, \theta_d$
for task $\mathcal{T}_i$ in meta-training tasks $\mathcal{T}$ **do**
  Make a copy of θ_c resulting in f_{θ_c}
  Make a copy of θ_e resulting in f_{θ_e}
  Make a copy of θ_g resulting in f_{θ_g}
  Make a copy of θ_d resulting in f_{θ_d}
  for data in training set $\mathcal{T}_{i_{tr}}$ **do**
    FMSA meta-training algorithm
  end for
  for data in validation set $\mathcal{T}_{i_{val}}$ **do**
    Compute meta-learner losses
    Update parameters of $\theta_c, \theta_e, \theta_g, \theta_d$
  end for
end for
FMSA real-attack algorithm for ML-NIDS victim

Generateing synthetic samples

We use the idea of Prototypical Network. We assume that there exists a low-latitude embedding space in which the sample distributions of all categories are far from each other. We set up encoder E as a prototype extractor. The adversary uses the auxiliary knowledge as input x to E. According to the N-way K-shot setting, we set the auxiliary knowledge as K samples from each class for task $\mathcal{T}_i$. The encoder E projects the input x into a feature vector r, $r = E(x)$. The input to G consists of a randomly sampled noise vector z with a feature vector r from E, $y = G(z, r)$. The output of G is expected to approximate the real sample x, requiring the discriminator D to be involved in working with G to form an adversarial training. The discriminator D is used to predict the Wasserstein distance between the true and fake samples, as shown in Eq. 2.

$$\mathcal{L}^D_{\mathcal{T}_i} = \mathbb{E}_{x \sim P_r}[D(x)] - \mathbb{E}_{y \sim P_g}[D(y)] \tag{2}$$

Where the sample distributions of P_r and P_g correspond to the real and false samples, respectively. G's objective is to reduce the Wasserstein-distance. At the same time, the output y of G is expected to maximize the output difference between the clone model C and the victim model V, constituting another kind of adversarial learning with the clone model. Equation 3 defines the optimization objective of G.

$$\mathcal{L}^G_{\mathcal{T}_i} = \lambda_1 \cdot \mathbb{E}_{y \backsim P_g}[D(y)] + \lambda_2 \cdot \mathcal{L}^C_{\mathcal{T}_i} \tag{3}$$

In contrast to the discriminator D to predict the distance between sample distributions, encoder E wants the input and generator output to be as close as possible, which is consistent with the setting of Prototypical Network, and uses l_2 norm as encoder E 's loss function, as defined in Eq. 4.

$$\mathcal{L}^E_{\mathcal{T}_i} = \sum_{i=1}^{I} ||x_i - y_i|| \tag{4}$$

Algorithm 2 FMSA meta-training algorithm

Input: $f_{\theta_c}, f_{\theta_e}, f_{\theta_g}, f_{\theta_d}$,simulated victim V
simulated dataset $\mathcal{T}_{i_{tr}}$,training iterations N_{iter},
generator iterations N_G
for $i \leftarrow 0$ to N_{iter} **do**
  $x \sim \mathcal{T}_{i_{tr}}$
  for $j \leftarrow 0$ to N_G **do**
    $r = f_{\theta_e}(x)$
    $z \sim \mathcal{N}(0, 1)$
    $y = f_{\theta_g}(r, z)$
    $\mathcal{L}^C_{\mathcal{T}_i} = \mathcal{L}_{KL}(V(y), f_{\theta_c}(y))$
    $\mathcal{L}^E_{\mathcal{T}_i} = \sum_{i=1}^{I} ||x_i - y_i||$
    $f_{\theta_c} \leftarrow f_{\theta_c} - \eta \nabla_{f_{\theta_c}} \mathcal{L}^C_{\mathcal{T}_i}$
    $f_{\theta_e} \leftarrow f_{\theta_e} - \eta \nabla_{f_{\theta_e}} \mathcal{L}^E_{\mathcal{T}_i}$
  end for
  $\mathcal{L}^D_{\mathcal{T}_i} = \mathbb{E}_{x \sim \mathcal{T}_{i_{tr}}}[f_{\theta_d}(x)] - \mathbb{E}_{y \sim P_g}[f_{\theta_d}(y)]$
  $f_{\theta_d} \leftarrow f_{\theta_d} + \eta \nabla_{f_{\theta_d}} \mathcal{L}^D_{\mathcal{T}_i}$
  $\mathcal{L}^G_{\mathcal{T}_i} = \mathbb{E}_{y \backsim P_g}[f_{\theta_d}(y) - \mathcal{L}^C_{\mathcal{T}_i}]$
  $f_{\theta_g} \leftarrow f_{\theta_g} + \eta \nabla_{f_{\theta_g}} \mathcal{L}^G_{\mathcal{T}_i}$
end for

Training meta-learners

We set up meta-learners in each of the four modules, θ_c, θ_e, θ_g, and θ_d represent the parameters of meta-learners of the clone model, encoder, generator, and discriminator, respectively. Following the meta-learning setup, Meta-training and meta-testing both contain inner and outer loops. In the inner loop, each meta-learner directs the

initialization of the corresponding module separately. The overall objective for the inner loop update for task $\mathcal{T}_i$ is:

$$\begin{aligned} &\min_{f_{\theta_c}} \mathcal{L}^C_{\mathcal{T}_i}(f_{\theta_c}), \min_{f_{\theta_e}} \mathcal{L}^E_{\mathcal{T}_i}(f_{\theta_e}) \\ &\max_{f_{\theta_d}} \mathcal{L}^D_{\mathcal{T}_i}(f_{\theta_d}), \max_{f_{\theta_g}} \mathcal{L}^G_{\mathcal{T}_i}(f_{\theta_g}) \end{aligned} \tag{5}$$

where θ'_c, θ'_e, θ'_g, and θ'_d represent the best parameters of four modules for task $\mathcal{T}_i$. In the outer loop, the meta-learner is evaluated for performance based on the best-trained parameters and trained with gradient descent so that the meta-learner can better direct the models' initialization. The overall meta-objective for each meta-learner is:

$$\begin{aligned} \theta &= \theta_c - \eta_1 \nabla_{f_{\theta_c}} \sum_{\mathcal{T}_i \backsim p(\mathcal{T})} \mathcal{L}^C_{\mathcal{T}_i}(f_{\theta'_c}) \\ \theta_e &= \theta_e - \eta_2 \nabla_{f_{\theta_e}} \sum_{\mathcal{T}_i \backsim p(\mathcal{T})} \mathcal{L}^E_{\mathcal{T}_i}(f_{\theta'_e}) \\ \theta_d &= \theta_d + \eta_3 \nabla_{f_{\theta_d}} \sum_{\mathcal{T}_i \backsim p(\mathcal{T})} \mathcal{L}^D_{\mathcal{T}_i}(f_{\theta'_d}) \\ \theta_g &= \theta_g + \eta_4 \nabla_{f_{\theta_g}} \sum_{\mathcal{T}_i \backsim p(\mathcal{T})} \mathcal{L}^G_{\mathcal{T}_i}(f_{\theta'_g}) \end{aligned} \tag{6}$$

The overall implementation pseudocode of FMSA is explained in Algorithm 1. In addition, the FMSA meta-training algorithm and the FMSA real-attack algorithm are explained in Algorithms 2 and 3. The FMSA real attack is equivalent to the meta-testing phase, where meta-learners are no longer updated.

Algorithm 3 FMSA real-attack algorithm

Input: $\theta_c, \theta_e, \theta_g, \theta_d$,ML-NIDS victim V
victim dataset $\mathcal{T}_v$,training iterations N_{iter},
generator iterations N_G,clone iterations N_C,

```
for i ← 0 to N_iter do
    x ~ T_v
    for j ← 0 to N_G do
        r = θ_e(x)
        z ~ N(0,1)
        y = θ_g(r, z)
        L^E_{T_i} = Σ_{i=1}^{I} ||x_i − y_i||
        θ_e ← θ_e − η∇_{θ_e} L^E_{T_i}
    end for
    L^D_{T_i} = E_{x~T_{i_tr}}[θ_d(x)] − E_{y~P_g}[θ_d(y)]
    θ_d ← θ_d + η∇_{θ_d} L^D_{T_i}
    L^G_{T_i} = E_{y~P_g}[θ_d(y) − L^C_{T_i}]
    θ_g ← θ_g + η∇_{θ_g} L^G_{T_i}
    for k ← 0 to N_C do
        L^C_{T_i} = L_{KL}(V(y), θ_c(y))
        θ_c ← θ_c − η∇_{θ_c} L^C_{T_i}
    end for
end for
```

Time complexity analysis

Meta-training phase. In the meta-training phase, the FSMA algorithm consists of two processes: inner loop and outer loop. In the inner loop, the model parameters specific to each task, including the encoder E, discriminator D, generator G, and clone model C, need to be updated using gradient descent optimization methods. This ensures that the clone model closely approximates the victim model simulated for the current task. Assuming the number of gradient descent optimization steps for training in the inner loop is N, the time complexity of each inner loop in FMSA can be expressed as $O(NK)$, where K represents the computational cost of updating the model in each iteration. This factor is crucial and cannot be ignored as it depends on the model's parameter size, the number of training samples per task, and the sample dimensions. In the outer loop, the updated model parameters are used to update the corresponding meta-learner's parameters, enabling it to achieve good generalization performance across all malicious behavior recognition tasks. The overall meta-training time complexity of the FMSA algorithm can be expressed as $O(TNK)$, where T represents the total number of classification tasks set in the outer loop. In our experiments, we set 60,000 classification tasks for malicious and benign traffic, following the N-way K-shot setting of few-shot learning. This allows the meta-learners to reach overall optimality by enabling the model to achieve the best performance within a few gradient descent steps for each sub-task.

Meta-testing phase. In the practical implementation of model stealing attacks, we assume that we have already found the meta-learners with optimal performance. Only a few gradient descent steps are required to make the clone model approximate the victim model to be attacked. At this stage, the meta-testing time complexity of the FMSA algorithm can be expressed as $O(NK)$, where N represents the number of gradient descent optimization steps, and K represents the computational cost of updating the model.

Evaluation

We describe the specifics of our experiments in this section. We first describe the settings of the trials, including the datasets, experimental details, evaluation metrics, and comparison methods. Then, for analysis and debate, the suggested FMSA method is contrasted with different methods. Furthermore, after the model stealing attack, we conduct adversarial attack experiments against victim ML-NIDS to investigate the safety risk caused by our method.

Dataset

There are no acceptable datasets to serve as a benchmark since NIDS is a young subject. We adopt the dataset settings used by Rong et al. (2021). Training datasets of the victim ML-NIDS must adhere to the following criteria to be used in our research: (1) instead of raw network data, they ought to include features derived from the traffic; (2) datasets must contain a large number of different malicious network behavior labels to satisfy the training tasks of the meta-learning. Hence, in order to create our simulated victim datasets, we take into account three real-world datasets. The two data sources for malicious data are MCFP (Stratosphere 2015) and CICIDS2017 (Panigrahi and Borah 2018). We choose USTC-TFC (Wang et al. 2017) as the source for daily benign behavioral data.

CICIDS2017 includes both raw network data and feature data, which was created in a simulated scenario. There is a wide range of attack techniques, including CSRF, Heartbleed, XSS, DDoS, and SYN attacks.

MCFP dataset was produced as part of a CTU (Czech Technical University) research effort that aimed to gather various forms of harmful traffic from real network activities. It contains over forty different specific malware families, most of which carry out malicious behavioral activities, including trojans, denial-of-service attacks, exploit extraction, and botnets.

USTC-TFC dataset is a source of benign samples produced by 21 benign applications and utilized in our investigation. Included in these categories are online social software, web article reading, online shopping, etc.

Experimental details

Experimental details of the FMSA method are described in this section. We executed our code in a computer environment based on the Ubuntu 18.04 operating system, equipped with a single GPU. We utilized CUDA acceleration to enhance the speed of algorithm execution and improve the efficiency of training models. Our code implementation involved the Python PyTorch framework, along with learn2learn, a PyTorch-based meta-learning library. As we update the parameters of the meta-learner in the outer loop based on the best models obtained from the inner loop training, where the initialization of the inner loop's best models is guided by the meta-learner, the parameter update process of the meta-learner essentially involves calculating second-order gradients. The learn2learn meta-learning library assists us in automatically performing second-order gradient computations for updating the meta-learner's parameters or approximating the second-order gradients.

We perform our evaluations by attacking ML-NIDS victim models and employ a 6-layer DNN model with random initialization as the victim model. For clone model C, we also use a 12-layer DNN model with different hyperparameter settings and initialization methods. Any sufficiently complicated DNN may often be utilized as the clone model. Our clone model is trained using an Adam optimizer with a learning rate of 0.008. For prototype feature extractor E, we choose LeNet5 (LeCun et al. 1998) to extract the prototype representation of each class. The feature vectors are rearranged into a grayscale map format as input to E. We employ a generative model G with three CNN layers, and each CNN layer is followed by a batch norm layer. Moreover, the activations are upsampled to guarantee that the outputs produced by G are the appropriate dimensionality for the victim dataset. In addition, we set discriminator D to be the same structure as generator G.

Our model stealing attack assumes having N-way K-shot data samples from the victim dataset as the auxiliary knowledge. It indicates that the adversary picks N classes, each with K samples, at random from the total dataset. We presume that the real victim dataset and the simulated meta-training dataset come from the same network environment. Yet, the data classes for simulated and real victims differ. We set the values of N and K to 10 and 5, respectively.

Evaluation metrics

The accuracy rate (ACC), model extraction rate (MER), and attack cost of querying the victim model serve as our primary assessment measures in experiments.

Accuracy rate (ACC) We utilize the accuracy rate to assess how well clone models perform on the victim test set. Victim ACC and Clone ACC denote the prediction accuracy of the victim model and the clone model on the victim test set, respectively. In the FMSA method, a meta-cloner is trained with the K-way N-shot setup for comparison with other methods.

Model extraction rate (MER) To assess the similarity between the clone model and the victim model, Eq. 7 determines the extraction rate. $V(x)$ and $C(x)$ represent the results returned by models for the input sample x, and the function d calculates the Hamming distance.

$$\begin{aligned} \mathrm{MER}(C, V) &= 1 - \sum_{\boldsymbol{x} \in \mathcal{D}_{test}} \frac{C(V(\boldsymbol{x}), V(\boldsymbol{x}))}{|\mathcal{D}_{test}|} \\ d(C(\boldsymbol{x}), V(\boldsymbol{x})) &= \sum_{i=1}^{n} C(x_i) \oplus V(x_i) \end{aligned} \tag{7}$$

Attack cost In addition to the clone model's accuracy and extraction rate, we are also concerned about the cost of training clone models, including the training cost to make clone models converge and the quantity of victim model inquiries. The quantity of query samples is even

more critical in security-sensitive ML-NIDS and needs to be strictly controlled to prevent NIDS alerts. In this paper, we investigate the relationship between the victim model's query volume and the clone model's performance to evaluate the cost of the model stealing attack. It is worth noting that only one of the methods used in the experiments uses real victim data for querying, including our FMSA method, which utilizes synthetic data.

Also, we use FID (Heusel et al. 2017) and IS (Salimans et al. 2016) to evaluate synthetic data. IS evaluates how well a generative model is and whether the generated data is actual and diverse; the more significant the IS value, the better the generative model is. FID indicates the distance between the generated and real feature vectors. The closer the FID distance is, the better the generated model is; i.e., the smaller the FID is, the better the generated model is.

Comparison methods

We choose three methods to compare with the proposed FSMA method depending on the type of training data of the clone model. These three methods represent random noise, partial real data, and adversarial data, respectively.

1. Random (baseline): Randomly produced data may be the only dataset the adversary may use if they are unaware of the victim's training data. As suggested by Roberts et al. (2019), we query the victim model using data randomly sampled from an Ising prior model. This attack acts as a baseline for comparison with our method.
2. Knockoff (Orekondy et al. 2019): It assumes the adversary owns part of the victim's actual dataset and constructs a clone training dataset using this part and prediction results from the victim model.
3. MAZE (Kariyappa et al. 2021): The generator produces adversarial data (not adversarial attack samples) to execute model stealing. However, synthetic samples only reduce the output error of the clone and victim models without any practical meaning. It tends to ignore the real data distribution and leads to catastrophic forgetting and mode collapse (Thanh-Tung and Tran 2020).

Performance evaluation

According to the experimental setting, we assume simulated victim datasets from the same network environment as the real victim dataset. For example, in the case of the conventional Internet, simulated and real victim datasets are both considered as parts of the CICIDS2017. We carry out our method from two representative situations independently to investigate the effectiveness and cost of FMSA against ML-NIDS deployed in various network settings. Table 1 shows the performance of different methods. We compare the results with three separate

Table 1 Comparison of clone accuracy rates and extraction rates obtained from different methods

Dataset	Victim ACC (%)	Attacks	Clone ACC (%)	MER (%)
CICIDS2017	96.71	FMSA	88.92	90.10
		Random	48.64	41.39
		Knockoff	69.15	57.10
		MAZE	80.83	82.50
MCFP	92.36	FMSA	83.85	87.74
		Random	37.28	22.75
		Knockoff	62.73	51.46
		MAZE	75.48	79.47

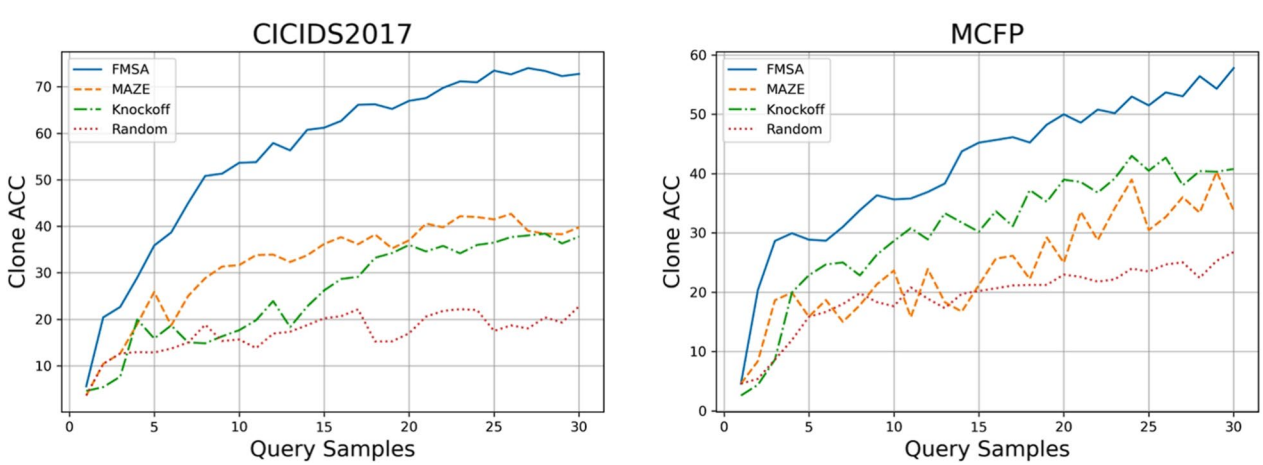

Fig. 3 The clone accuracy rate of FMSA, MAZE, Noise, and KnockoffNets as the query budget varies

attacks to steal the victim model. Furthermore, Fig. 3 demonstrates how the clone model's accuracy varies as the number of query samples rises.

According to the results, FMSA achieves advanced performance and outperforms the comparative methods. Though the adversarial learning of the generator and the clone model in our approach is similar to MAZE, FMSA allows the generator to learn the distribution of real victim data simultaneously. Since the generator's goal in our task is to generate feature samples rather than complex data such as images, the generator can converge quickly after a few training iterations. For the CICIDS2017 dataset, the clone model trained using the FMSA algorithm achieved an accuracy of 88.92% and a similarity of 90.10% to the victim model on the test set. However, as the network environment becomes more complex and the diversity of malicious traffic increases, the accuracy and similarity of the clone model on the MCFP dataset dropped to 83.85% and 87.74%, respectively. It is noteworthy that the decrease in accuracy for both the clone model and the victim model is approximately equal, while the decrease in the model extraction rate is smaller. This indirectly reflects that the optimization goal of the clone model is to fit the target model's classification ability rather than solely optimizing the classification task.

FMSA also achieves high accuracy with a small number of queries to the victim model compared to other methods, which require lots of queries to perform as well as our method. As shown in Fig. 3, with only 10 query samples, FMSA's cloning model can achieve a cloning ACC that far exceeds that of other methods. After undergoing 30 iterations of queries, the cloned models trained by other model stealing algorithms are far from convergence. While other algorithms such as MAZE and random noise do not rely on the victim dataset, they still require a large number of queries and attempts to search the entire feature space to successfully imitate the victim model. With the introduction of meta-learning, clone models can learn and converge fast with a few samples after learning many different tasks. Although such a model stealing attack comes with a high training cost, it is very effective for ML-NIDS with a limited number of queries.

Data assessment

In this section, we use two metrics to assess synthetic data quality In Table 2, we compare the average IS and FID values across the five different categories of data: (1) the victim's training dataset; (2) random noise; (3) auxiliary dataset utilized in Knockoff attack; (4) synthetic data generated by MAZE, and (5) synthetic data generated by our method FMSA.

The real victim training dataset most accurately depicts the input space of the five data sets. The auxiliary dataset utilized in the Knockoff attack also achieves high quality and diversity, as it is derived from the real dataset. On the other hand, the random noise dataset performs the worst, obtaining the lowest IS value and the highest FID value. The synthetic data generated by MAZE also obtained similar quality scores as the noise data since the primary purpose of the synthetic data is to widen the gap between the outputs of the victim and clone models, favoring the training of the clone model with no real data distribution. Our method FMSA produces synthetic data that maximizes the difference between the output of the clone and victim models while being supervised by a discriminator that forces the learning of the distribution of the real victim data, which obtains a quality score only slightly worse than the auxiliary data set.

Further adversarial attacks

This section shows that our FMSA method will pose a severe security risk to ML-NIDS. Deep learning models are susceptible to adversarial example attacks, in which adding tiny adjustments imperceptible to the human eye to the original samples can successfully misclassify the samples (Aldahdooh et al. 2022). We, as adversaries, obtain clone models through black-box model stealing attacks and use the clone models to generate adversarial samples. We exploit the adversarial samples' transferability against the black box victim model. We use the PGD approach (Madry et al. 2017), an iterative adversarial attack on a white-box set. We performed the PGD attack against the clone and victim models, with the success rates shown in Table 3. We compare three different attack scenarios: (1) attack against the white-box victim model, (2) attack against the white-box clone model, and () attack

Table 2 Qualiity analysis of data using IS and FID

Dataset	Victim	Auxiliary		Random		MAZE		FMSA	
	$\mathcal{D}_{train}$	$\mathcal{D}_{aux}$		$\mathcal{D}_{rand}$		$\mathcal{D}_{syn}$		$\mathcal{D}_{syn}$	
	IS	IS	FID	IS	FID	IS	FID	IS	FID
CICIDS2017	6.32	4.63	19.28	1.85	86.50	2.08	84.80	4.22	26.32
MCFP	5.64	4.16	34.23	1.32	128.21	1.77	95.16	3.45	48.10

Table 3 Success rate of adversarial attacks

Dataset	Attack success rate (%)		
	White-box victim model	White-box clone model	Black-box victim model
CICIDS2017	98.95	95.60	94.69
MCFP	96.07	92.31	87.15

against the black-box victim model. We discover that the success rate of the black-box adversarial attack is comparable to that of the white-box, demonstrating the effectiveness of the clone model in translating the adversarial example from the victim model. It highlights the dangers of model stealing attacks, which could be a security risk for systems like ML-NIDS that are sensitive to security.

Conclusion

This study proposes FMSA, a low-query, high-accuracy model stealing attack against the victim model. FMSA is the first model stealing attack based on meta-learning and sets ML-NIDS as attack targets. Since the NIDS system will strictly protect the private data and detect abnormal queries, it is almost impossible for the adversary to access the target dataset and perform a large number of queries. We propose two different sets of adversarial learning in FMSA, namely, adversarial learning of generators and discriminators, and adversarial learning of generators and clone models. This allows the clone model to use auxiliary knowledge to learn the distribution of the victim dataset while reducing the difference in output with the victim model. FMSA achieves the highest accuracy and model similarity compared to other techniques on the CICIDS2017 and MCFP datasets and can achieve high accuracy with only a few queries. Nevertheless, the FMSA method entails a substantial training cost, which might conflict with the initial objective of model stealing to replicate victim models at a minimal expense. However, to overcome the security measures of ML-NIDS, a compromise solution is necessary. Future research endeavors should concentrate on developing methods that enable the cloning of ML-NIDS victim models with enhanced training efficiency and reduced costs. In summary, the proposed FMSA method presented in this study offers a novel solution in the field of model stealing attacks, while also highlighting the risks associated with employing ML-NIDS for safeguarding network security. Despite the high accuracy and sensitivity of these systems, our research demonstrates that effective model stealing attacks can still be conducted by adversaries, even in the presence of protective measures.

Abbreviations

FMSA	Fast model stealing attack
ML	Machine learning
DNN	Deep neural network
NIDS	Network intrusion detection system
ML-NIDS	Network intrusion detection system using machine learning techniques
MAML	Model agnostic meta learning
ACC	Accuracy rate
MER	Model extraction rate
IS	Inception score
FID	Frechet inception distance

Acknowledgements
This work was supported in part by the Joint Funds of the National Natural Science Foundation of China (Grant No. U22A2036), the Fundamental Research Funds for the Central Universities (HIT.OCEF.2021007), the National Key Research and Development Program of China (2020YFB1406902), and the Key-Area Research and Development Program of Guangdong Province (2020B0101360001).

Author contributions
All authors read and approved the final manuscript.

Funding
This work was supported by Grant Nos. U22A2036, HIT.OCEF.2021007, 2020YFB1406902, and 2020B0101360001.

Availability of data and materials
Dataset used in paper is publicly available in Panigrahi and Borah (2018), Stratosphere (2015), Wang et al. (2017).

Declarations

Competing interests
The authors declare that they have no competing interests

Received: 15 March 2023 Accepted: 14 June 2023
Published online: 04 August 2023

References

Aldahdooh A, Hamidouche W, Fezza SA, Déforges O (2022) Adversarial example detection for dnn models: a review and experimental comparison. Artif Intell Rev 55(6):4403–4462
Bao J, Chen D, Wen F, Li H, Hua G (2017) Cvae-gan: fine-grained image generation through asymmetric training. In: Proceedings of the IEEE international conference on computer vision, pp 2745–2754
Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al (2020) Language models are few-shot learners. Adv Neural Inf Process Syst 33:1877–1901
Chen J, Wang J, Peng T, Sun Y, Cheng P, Ji S, Ma X, Li B, Song D (2022a) Copy, right? A testing framework for copyright protection of deep learning models. In: 2022 IEEE symposium on security and privacy (SP), pp 824–841
Chen Y, Yang X-H, Wei Z, Heidari AA, Zheng N, Li Z, Chen H, Hu H, Zhou Q, Guan Q (2022b) Generative adversarial networks in medical image augmentation: a review. Comput Biol Med 144:105382
Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning, pp 1126–1135
Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2020) Generative adversarial networks. Commun ACM 63(11):139–144

Goryunov MN, Matskevich AG, Rybolovlev DA (2020) Synthesis of a machine learning model for detecting computer attacks based on the cicids2017 dataset. Proc Inst Syst Program RAS 32(5):81–94
Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017) Improved training of Wasserstein Gans. Adv Neural Inf Process Syst 30:5767–5777
Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2017) Gans trained by a two time-scale update rule converge to a local nash equilibrium. Adv Neural Inf Processing Syst 30:6629–6640
Hu H, Salcic Z, Sun L, Dobbie G, Yu PS, Zhang X (2022) Membership inference attacks on machine learning: a survey. ACM Comput Surv (CSUR) 54(11s):1–37
Juuti M, Szyller S, Marchal S, Asokan N (2019) Prada: protecting against DNN model stealing attacks. In: 2019 IEEE European symposium on security and privacy (EuroS &P), pp 512–527
Kariyappa S, Prakash A, Qureshi MK (2021) Maze: data-free model stealing attack using zeroth-order gradient estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13814–13823
Kesarwani M, Mukhoty B, Arya V, Mehta S (2018) Model extraction warning in mlaas paradigm. In: Proceedings of the 34th annual computer security applications conference, pp 371–380
Kiran BR, Sobh I, Talpaert V, Mannion P, Al Sallab AA, Yogamani S, Pérez P (2021) Deep reinforcement learning for autonomous driving: a survey. IEEE Trans Intell Transp Syst 23(6):4909–4926
LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
Liu B, Ding M, Shaham S, Rahayu W, Farokhi F, Lin Z (2021) When machine learning meets privacy: a survey and outlook. ACM Comput Surv 54(2):1–36
Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2017) Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083
Mahmood K, Mahmood R, Van Dijk M (2021) On the robustness of vision transformers to adversarial examples. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 7838–7847
Oh SJ, Schiele B, Fritz M (2019) Towards reverse-engineering black-box neural networks. Explain AI Interpret Explain Vis Deep Learn, 121–144
Orekondy T, Schiele B, Fritz M (2019) Knockoff nets: Stealing functionality of black-box models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4954–4963
Panigrahi R, Borah S (2018) A detailed analysis of cicids2017 dataset for designing intrusion detection systems. Int J Eng Technol 7(3.24):479–482
Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security, pp 506–519
Rakin AS, Chowdhuryy MHI, Yao F, Fan D (2022) Deepsteal: Advanced model extractions leveraging efficient weight stealing in memories. In: 2022 IEEE symposium on security and privacy (SP), pp 1157–1174
Roberts N, Prabhu VU, McAteer M (2019) Model weight theft with just noise inputs: the curious case of the petulant attacker. arXiv preprint arXiv:1912.08987
Rong C, Gou G, Hou C, Li Z, Xiong G, Guo L (2021) Umvd-fsl: unseen malware variants detection using few-shot learning. In: 2021 international joint conference on neural networks (IJCNN), pp 1–8
Rüping S, Schulz E, Sicking J, Wirtz T, Akila M, Gannamaneni S, Mock M, Poretschkin M, Rosenzweig J, Abrecht S et al (2022) Inspect, understand, overcome: a survey of practical methods for AI safety. Deep Neural Networks and Data for Automated Driving: Robustness, Uncertainty Quantification, and Insights Towards Safety 3
Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X (2016) Improved techniques for training gans. Adv Neural Inf Processing Syst 29:2234–2242
Sanyal S, Addepalli S, Babu RV (2022) Towards data-free model stealing in a hard label setting. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 15284–15293
Snell J, Swersky K, Zemel R (2017) Prototypical networks for few-shot learning. Adv Neural Inf Process Syst 30
Stratosphere: stratosphere laboratory datasets (2015). https://www.stratosphereips.org/datasets-overview Accessed 13 Mar 2020
Sun Q, Liu Y, Chua T-S, Schiele B (2019) Meta-transfer learning for few-shot learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 403–412
Thanh-Tung H, Tran T (2020) Catastrophic forgetting and mode collapse in gans. In: 2020 international joint conference on neural networks (ijcnn), pp 1–10
Tian Y, Zhao X, Huang W (2022) Meta-learning approaches for learning-to-learn in deep learning: a survey. Neurocomputing 494:203–223
Touvron H, Cord M, Sablayrolles A, Synnaeve G, Jégou H (2021) Going deeper with image transformers. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 32–42
Tramèr F, Zhang F, Juels A, Reiter MK, Ristenpart T (2016) Stealing machine learning models via prediction APIS. In: USENIX security symposium vol 16, pp 601–618
Truong L, Jones C, Hutchinson B, August A, Praggastis B, Jasper R, Nichols N, Tuor A (2020) Systematic evaluation of backdoor data poisoning attacks on image classifiers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 788–789
Vanschoren J (2018) Meta-learning: a survey. arXiv preprint arXiv:1810.03548
Wang Y, Yao Q, Kwok JT, Ni LM (2020) Generalizing from a few examples: a survey on few-shot learning. ACM Comput Surv 53(3):1–34
Wang B, Gong NZ (2018) Stealing hyperparameters in machine learning. In: 2018 IEEE symposium on security and privacy (SP), pp 36–52
Wang B, Yao Y, Shan S, Li H, Viswanath B, Zheng H, Zhao BY (2019) Neural cleanse: identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE symposium on security and privacy (SP), pp 707–723
Wang W, Zhu M, Zeng X, Ye X, Sheng Y (2017) Malware traffic classification using convolutional neural network for representation learning. In: 2017 international conference on information networking (ICOIN), pp 712–717
Yang J, Li T, Liang G, He W, Zhao Y (2019) A simple recurrent unit model based intrusion detection system with dcgan. IEEE Access 7:83286–83296
Yang Z, Liu X, Li T, Wu D, Wang J, Zhao Y, Han H (2022) A systematic literature review of methods and datasets for anomaly-based network intrusion detection. Comput Sec 116:102675

Publisher's Note

Weizhe Zhang (Senior Member, IEEE) received B.Eng, M.Eng and Ph.D. degree of Engineering in computer science and technology in 1999, 2001 and 2006 respectively from Harbin Institute of Technology. He is currently a professor in the School of Computer Science and Technology at Harbin Institute of Technology, China, and director in the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China. His research interests are primarily in parallel computing, distributed computing, cloud and grid computing, and computer network. He has published more than 100 academic papers in journals, books, and conference proceedings.