# Adversarial Attacks Against Deep Learning-based Network Intrusion Detection Systems and Defense Mechanisms

# Adversarial Attacks Against Deep Learning-based Network Intrusion Detection Systems and Defense Mechanisms

Chaoyun Zhang, Xavier Costa-Pérez, *Senior Member, IEEE,* and Paul Patras, *Senior Member, IEEE*

**Abstract**—Neural networks (NNs) are increasingly popular in developing NIDS, yet can prove vulnerable to adversarial examples. Through these, attackers that may be oblivious to the precise mechanics of the targeted NIDS add subtle perturbations to malicious traffic features, with the aim of evading detection and disrupting critical systems. Defending against such adversarial attacks is of high importance, but requires to address daunting challenges. Here, we introduce TIKI-TAKA, a general framework for *(i)* assessing the robustness of state-of-the-art deep learning-based NIDS against adversarial manipulations, and which *(ii)* incorporates defense mechanisms that we propose to increase resistance to attacks employing such evasion techniques. Specifically, we select five cutting-edge adversarial attack types to subvert three popular malicious traffic detectors that employ NNs. We experiment with publicly available datasets and consider both one-to-all and one-to-one classification scenarios, i.e., discriminating illicit vs benign traffic and respectively identifying specific types of anomalous traffic among many observed. The results obtained reveal that attackers can evade NIDS with up to 35.7% success rates, by only altering time-based features of the traffic generated. To counteract these weaknesses, we propose three defense mechanisms: model voting ensembling, ensembling adversarial training, and query detection. We demonstrate that these methods can restore intrusion detection rates to nearly 100% against most types of malicious traffic, and attacks with potentially catastrophic consequences (e.g., botnet) can be thwarted. This confirms the effectiveness of our solutions and makes the case for their adoption when designing robust and reliable deep anomaly detectors.

**Index Terms**—Adversarial Attacks, Network Intrusion Detection Systems, Deep Learning

## 1 INTRODUCTION

NETWORK Intrusion Detection aims at identifying malicious traffic flows, so as to protect computers, networks, servers, and data from attacks, unauthorized access, modification, or destruction [2]. Given the unprecedented growth in the data traffic volume transiting both wired and wireless infrastructure, Network Intrusion Detection (NID) is becoming increasingly important to ensure system/service availability and protect individuals' safety and privacy online. As new cyberattacks proliferate, traditional intrusion detection methodologies that rely on pattern matching (e.g., IP address and port number) and classification are losing effectiveness [3]. In this context, machine learning-based solutions are gaining traction, as they rely increasingly less often on deep packet inspection (hence raising fewer privacy concerns) and may have better generalization abilities.

Stimulated by recent success in areas such as image classification, the limited extent of feature engineering involved, and the decreasing cost of parallel processing hardware, deep learning – a subset of machine learning – is making its way also in the networking domain [4]. This includes NID, where solutions based on Deep Neural Networks (DNNs) yield demonstrably superior detection accuracy (see, e.g., [5], [6]). However, due to their complex structures, DNNs also suffer from limited interpretability, which inevitably raises important questions: *Is deep learning a truly reliable option for NID? Is there any "Achilles' heel" that can be exploited to compromise the expected high detection accuracy of neural network-based NID models?* Answering these questions is crucial to guaranteeing the reliability of Network Intrusion Detection Systems (NIDS).

Unfortunately, DNNs have been proven vulnerable to adversarial examples [7] or backdoor attacks [8] in several applications [9], [10], whereby they can be fooled by subtle perturbations introduced in the input [11], which interfere with the correctness of the inferences made. Since such adversarial manipulations are often extremely difficult to detect, deep learning-based NIDS are also at risk. Attackers, potentially unaware of the properties of an NIDS (i.e., black-box system), could generate adversarial samples by repeatedly changing small subsets of the traffic features, and make 'queries' to the NIDS. After each query, the attacker receives some feedback (e.g., an acknowledgment or lack of any response), which indicates the success or failure of the attack attempt.[1] Based on this feedback, the attacker can adjust the perturbations on selected features (e.g., intervals between consecutive packets) of the traffic, or introduce new ones, without changing its essence, until succeeding in bypassing the NIDS [12], [13]. By this approach, malicious flows

*C. Zhang is with Tencent Lightspeed & Quantum Studios, China. X. Costa-Pérez is jointly affiliated with NEC Labs Europe, Germany; i2CAT Foundation, Spain; and ICREA, Spain. P. Patras is with The University of Edinburgh, UK and with Net AI, UK. C. Zhang was with The University of Edinburgh when this work was conducted.*

*A preliminary version of this paper appeared in the Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop (CCSW'20) [1].*

1. For example, in the LOIC-HTTP DDoS attack HTTTP GET requests are sent towards a target, which would send back an HTTP response with a status code.

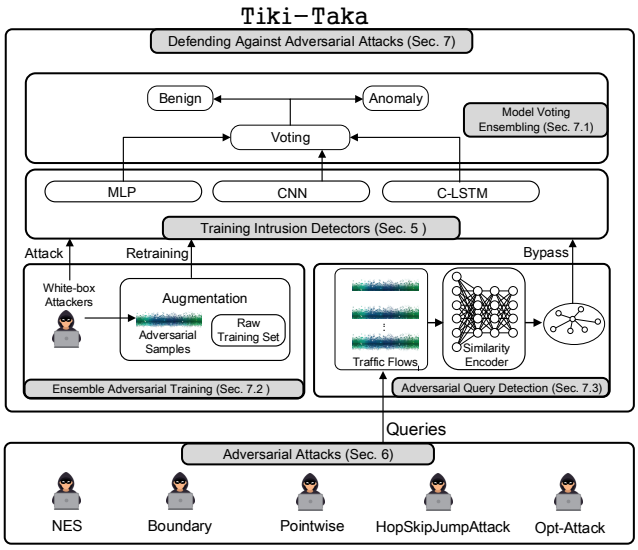


Fig. 1. The TIKI-TAKA framework for crafting and defending against adversarial attacks towards Network Intrusion Detection Systems (NIDS).

could then disguise into benign traffic and compromise their targets [14], while remaining undetected even by NIDS thought to be highly accurate. Cyberwarfare is exploding [15] and such **adversarial strategies offer cost-effective means to jeopardize healthcare systems, electronic voting, banking, industrial automation, and countless more**.

In this paper, we tackle the severe intrusion detection issues faced by classifiers under adversarial attacks. We first scrutinize the robustness of state-of-the-art deep learning NID models against different adversarial mechanisms, considering attacks in practical decision-based settings (i.e., attackers can only infer if the traffic generated was classified as benign or malicious, without knowledge about the exact class to which the traffic was mapped). We test the effectiveness and efficiency of each attack in two detection scenarios: one-to-all and one-to-one, i.e., aiming to discriminate malicious vs. benign traffic, and respectively to identify precise types of attacks. We then propose three solutions to defend against this new class of threats, which effectively reduce the success rate of each attack to a large extent. This enables more robust and reliable NID. In a nutshell we make the following **key contributions**:

**[C1]** We implement three types of DNN architectures based on state-of-the-art NID models, i.e., Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM), and perform NID on the realistic cyber defense dataset CSE-CIC-IDS2018 [16]. The NID models implemented achieve over 98.7% detection accuracy based on a limited set of features, which matches the performance of previously reported NIDS implementations (details in Sec. 5).

**[C2]** To demonstrate the NIDS considered can be evaded, we employ five state-of-the-art attack strategies to generate adversarial samples (i.e., NES, BOUNDARY, HOPSKIPJUMPATTACK, POINTWISE, and OPT-ATTACK), bounding traffic feature manipulations to realistic domain constraints (i.e., leaving unchanged those features that may alter flow semantics). We conduct a comprehensive evaluation on the effectiveness of each adversarial attack and provide an in-depth analysis of their characteristics (see Sec. 6).

**[C3]** We propose three defense mechanisms to strengthen deep learning-based NIDS against adversarial attacks, namely: model voting ensembling, ensembling adversarial training, and query detection. Each defense method can either operate individually or jointly with the others. Experiments show these methods drastically reduce the attack success rates, significantly improving the robustness of the NIDS considered as we bring detection rates close to 100%. Furthermore, the proposed methodology generalizes well to other environments, as our results of testing with a previously unseen dataset reveal (Sec. 7),

We name our general attack–defense framework TIKI-TAKA[2] and illustrate the workflow of our methodology in Fig. 1. To the best of our knowledge, **we are the first to introduce defense mechanisms against adversarial attacks targeting NIDS.**

## 2 RELATED WORK

DNNs are increasingly used for NID purposes, as they help minimize feature engineering efforts and operate with high detection accuracy [2]. However, recent research suggests that there exist loopholes that can degrade the performance of neural NIDS, as perturbation added to their input can trigger traffic misclassification [4], [17]. Thus, defending deep learning-based approaches from adversarial samples becomes a crucial issue for network security.

### 2.1 Deep Learning-based NID

Niyaz *et al.* employ sparse autoencoders for self-taught learning and extract important features from traffic flows [5]. They conduct NID on the NSL-KDD dataset [18] and achieve 98.84% F1 scores. Faker and Dogdu design an MLP to discriminate malicious traffic [19] in the CIC-IDS2017 dataset [16]. Although the model structure is simple, the MLP achieves significantly higher detection rates than Random Forest (RF), Gradient Boosting Tree (GPT), and Support Vector Machine (SVM) structures. Similar conclusions have been reached in [6], where Vinayakumar *et al.* compare the MLP with a large set of traditional machine learning approaches to NID, showing that deep learning yields better performance.

CNN-based approaches have been employed for NID as well [20]. Zhang *et al.* design a two-branch CNN and employ feature fusion, to resolve the class imbalance problem of the dataset used [21]. Their proposal detects a minor class of anomalies with higher accuracy, while being more efficient in terms of execution time. Recurrent Neural Networks (RNNs) are popular candidates for extracting temporal features of traffic flows [22]. Zhang *et al.* perform NID on raw packet-level traffic [23]. They combine CNNs and

2. TIKI-TAKA is a football tactic that encourages short and fast ball passing, and tackling on the spot when losing ball possession. We use this to metaphorize the frequent queries passed to an NIDS in the attack process, with the detector subsequently regaining control through defenses.

LSTMs to extract important spatial and temporal features, achieving higher detection rates than when using each of these components individually.

In our study, we select MLP, CNN, and LSTM as representative models to perform NID, then test them against adversarial attacks, and subsequently augment these models with a set of defense mechanisms that we propose for enhancing their robustness.

### 2.2 Attacking Deep Learning-based NIDS

The majority of existing methods that employ adversarial samples to compromise classifiers target image applications (e.g., [24], [25], [26]). Research on evading deep learning-based NIDS is scarce. Wang *et al.* employ four sets of white-box attack algorithms designed for image classification, to bypass MLP-based intrusion detectors trained on the NSL-KDD dataset [27]. Their experiments suggest that these attack algorithms are transferable to the NID domain and the MLP detectors are vulnerable to adversarial samples. However, attackers may not have access to the neural model underpinning the targeted NIDS, which make such settings more useful to NIDS designers to assess the robustness of their systems [14].

Yang *et al.* generate adversarial samples in black-box settings [28] using three types of approaches, namely surrogate models [29], Zeroth Order Optimization (ZOO) [30], and Generative Adversarial Networks (GANs) [31]. These methods can reduce the performance of MLP-based classifiers, thus becoming a threat to NIDS. Kuppa *et al.* consider a more realistic situation, performing black-box attacks against different deep learning-based detectors in decision-based and query-limited settings [14]. By learning and approximating the distribution of benign/anomalous samples, these methods can evade NIDS with high success rate.

Teuffenbach et al. extend existing adversarial example crafting algorithms by accounting for domain-specific constraints and demonstrate effectiveness in subverting DL-based NIDS [32]. Piplai et al. introduce a Generative Adversarial Network (GAN) based solution to train a traffic classifier and then compromise the trained model with adversarial attacks [33]. Five adversarial attack types are staged against deep learning models in [34] and their robustness is studied following adversarial training, which is one defense approach further considered in [35] as well as in this work.

### 2.3 Defending from Adversarial Samples

There exist a range of strategies for defending deep learning models from adversarial examples. Commonly used methods include Network Distillation [36], Adversarial Training [37], Adversarial Detecting [38], Input Reconstruction [39], Classifier Robustifying [40], Network Verification [41], and an ensemble of them [11], [42], which work either reactively or proactively. Network distillation methods employ a student neural network to learn knowledge from a more complex teacher network. With this approach, the student network generalizes better and becomes more robust to adversarial samples. Adversarial training retrains the neural networks by augmenting the original training set with adversarial samples, such that they can better defend against those inputs with subtle feature perturbations. Input reconstruction reduces the effectiveness of the perturbations by recovering the original input. Classifier robustifying employs various approaches (e.g., model ensembling) to improve the robustness of the original classifier. Network verification uses an additional classifier to identify adversarial samples.

While such approaches can be effective in the computer vision and natural language processing domains, *(i)* work by Carlini *et al.* demonstrates defense mechanisms against adversarial examples in imaging can be defeated by constructing new loss functions [43], while *(ii)* none of these are aimed at defending against adversarial samples in NIDS, which is the problem we tackle in this paper.

## 3 Threat Model

We focus on scenarios where attackers generate adversarial samples by adding small perturbations to the input given to NIDS, thereby aiming to cause misclassification and evade detection of their malicious traffic. As in [44], we denote by $x$ the input to a classifier (i.e., features extracted from flows), an adversarial sample as $x_{\text{adv}} = x + \sigma_x$, and the targeted class as $y_{\text{adv}}$. The objective of the adversarial attacks can be formulated as finding $x_{\text{adv}}$ such that $||x_{\text{adv}} - x||_\infty < \epsilon$ and $x_{\text{adv}}$ is classified as $y_{\text{adv}}$. Here, $\sigma_x$ is the perturbation added to the input and $\epsilon$ limits the perturbation scale.

### 3.1 Adversarial Settings

Typical attacks against machine learning models can be categorized into three classes: *(i)* white-box attacks, *(ii)* grey-box attacks, and *(iii)* black-box attacks. White-box attacks assume the malicious actors have access to both the target model's structure and training data, whereas grey-box attacks involve prior knowledge of the training data and only a rough understanding of the target model's structure. Such hypotheses apply in cases where system designers seek to improve the robustness of their NIDS, but less commonly to scenarios with external adversaries. At best, malicious actors could conduct white-box attacks against own models and subsequently seek to transfer these attacks onto victim NIDS. More often, potential hackers are forced to treat a NIDS as a black-box, since the details of a victim system's inner workings remain hidden and the only way in which the NIDS behavior can be learned is through a sequence of queries and the feedback received. This is also the primary practical threat model that we consider in this work, while the defense mechanisms we propose can also fend off adversarial samples adapted from white-box methods, as we reveal.

In general, an attacker may send a traffic flow towards the target network, which will be first examined by a NID model. This is known as a query process. Subsequently, the attacker will receive implicit/explicit feedback from the model, e.g., an ACK packet, which reflects whether the traffic flow was classified as anomalous. Based on the feedback, the attacker can adjust and apply subtle perturbations to the malicious traffic flows, thereby producing adversarial samples that eventually may compromise the effectiveness of the NIDS, which will end up classifying malicious traffic

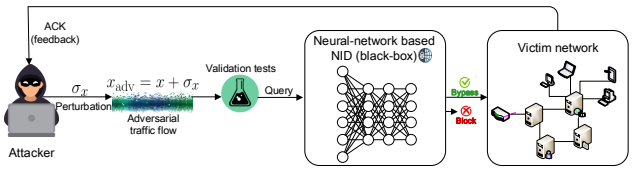


Fig. 2. An illustration of the attack process against machine learning-based NID models.

as benign. On the other hand, the attacker may not have confidence about the exact decision class decided by the NIDS, but whether the traffic was deemed malicious or benign (decision-based attacks). We illustrate this attack process against NID models in Fig. 2.

### 3.2 Domain Constraints

Unlike adversarial attacks against image classifiers, adversarial samples against NIDS must respect certain domain constrains [14], such that the functionality and intactness of the samples is preserved when introducing perturbations $\sigma_x$. This means that *(i)* only a subset of features are amendable; and *(ii)* the features of adversarial samples do not violate the properties inherent to the original samples. To meet these requirements, here we confine consideration to 22 time-based features, to which we add perturbations, as also suggested in [14]. These include *(a)* Forward Inter Arrival Time – the time between two packets sent in the forward direction (mean, min, max, std); *(b)* Backward Inter Arrival Time – the time between two packets sent in reverse direction (mean, min, max, std); *(c)* Active-Idle Time – the amount of time a flow was idle before becoming active, and vice-versa (mean, min, max, std), *(d)* Average number of bytes and packets sent in forward and backward directions in the initial window or/and sub-flows. By this approach, traffic flows in the original dataset do not change their semantics and thus their labels remain unchanged. It is conceivable that an attacker may also attempt to mimic the time features of benign flows and conceal malicious content within payloads (e.g., SQL injection, cross-site scripting, etc.). In such cases, our TIKI-TAKA framework can also accommodate payload-based features extracted, e.g., through word embedding or Text-CNNs [45]. Features outside the subset that may change flow semantics remain unchanged during the attack, which is inline with recent research confirming that adversarial samples can be constructed effectively by perturbing only a small subset of the input features [46].

In addition, we expect that *(i)* the Mean Absolute Percentage Error (MAPE) for each feature $k$ does not exceed 20%, i.e., $100 \cdot |(x^{(k)} - x^{(k)}_{\text{adv}})/x^{(k)}| \leq 20\%$; *(ii)* the perturbed features preserve the mean property (e.g., the mean forward inter arrival time) plus/minus std features do not exceed their corresponding max and min features; *(iii)* the sign of each perturbed sample remains the same as that of the original; and *(iv)* if the std feature is zero, the corresponding mean, max, min and std features remain unchanged in the adversarial sample. Samples that violate these constrains are to be regarded as unsuccessful trials, since they alter the originally intended functionality of the flows. In crafting adversarial attacks for our study, we will also perform validation tests based on these constraints.

TABLE 1
Statistics of the CSE-CIC-IDS2018 dataset employed.

| Flow Type | No. Instances | Ratio |
|---|---|---|
| Benign | 14,097,779 | 83.6861% |
| Bot | 286,191 | 1.6989% |
| DoS attack-SlowHTTPTest | 139,890 | 0.8304% |
| DoS attack-Hulk | 461,912 | 2.7420% |
| Brute Force-XSS | 230 | 0.0014% |
| SQL Injection | 87 | 0.0005% |
| Infiltration | 161,934 | 0.9613% |
| DoS attack-GoldenEye | 41,508 | 0.2464% |
| DoS attack-Slowloris | 10,990 | 0.0652% |
| Brute Force-Web | 611 | 0.0036% |
| FTP-Brute Force | 193,360 | 1.1478% |
| SSH-Brute Force | 187,589 | 1.1136% |
| DDoS attack-LOIC-UDP | 1,730 | 0.0103% |
| DDoS attack-HOIC | 686,012 | 4.0723% |
| DDoS attack-LOIC-HTTP | 576,191 | 3.4203% |
| All of the above attacks | 2,748,235 | 16.3139% |
| Total | 16,846,014 | 100% |

## 4 DATASET

We conduct all experiments (i.e., NID, black-box adversarial attacks, and attacks against NIDS augmented with the proposed defenses) using the publicly available CSE-CIC-IDS2018 dataset [16]. This encompasses 14 types of network intrusion traffic flows along with benign traffic. The attacks can be categorized into seven classes, namely Brute Force, Heartbleed, Botnet, Denial of Service (DoS), Distributed Denial of Service (DDoS), Web attacks, and infiltration. Table 1 summarizes the prevalence of each type of traffic. The infrastructure employed includes 50 machines, which attempt to intrude a victim network consisting of 420 end hosts and 30 servers.

A total of 80 features of the traffic flows are extracted to perform intrusion detection and we filter 65 of them for the purpose of our work. The features selected can be grouped into 8 classes, specifically *(a)* Forward Inter Arrival Time – the time between two packets sent in the forward direction (mean, min, max, std); *(b)* Backward Inter Arrival Time – the time between two packets sent in reverse direction (mean, min, max, std); *(c)* Flow Inter Arrival Time – the time between two packets sent in either direction (mean, min, max, std); *(d)* Active-Idle Time – the amount of time a flow was idle before becoming active (mean, min, max, std) and the amount of time a flow was active before becoming idle (mean, min, max, std); *(e)* Flags based features – the number of times the URG, PSH flags are set, both in the forward and backward direction; *(f)* Flow characteristics – bytes per second, packets per second, flow length (mean, min, max, std) and ratio between number of bytes sent downlink and uplink; *(g)* Packet count with flags FIN, SYN, RST, PUSH, ACK, URG, CWE and ECE; *(h)* Average number of bytes and packets sent in forward/backward directions in the initial window, bulk rate, and sub flows count. Our framework is readily extensible to other types of features, e.g., extracted from payloads [45].

We train all deep learning models, implement and defend against the adversarial attacks using the selected features, instead of parsing raw traffic packets flows, which reduces privacy concerns.

## 5 Training Intrusion Detectors

Training accurate deep network intrusion detectors is the initial important step of our study, as Tiki-Taka builds on the pre-trained NID models. To this end, we employ three well-known deep learning architectures, namely Multilayer Perceptron (MLP) [19], Convolutional Neural Network (CNN) [21], and CNN with Long Short-Term Memory (LSTM) layers, i.e., C-LSTM [23]. These models are frequently used for NID purposes and have achieved notable performance. We illustrate the model architectures in Fig. 3.

The MLP is the most simple deep learning architecture, which employs multiple stacks of fully-connected layers for features extraction. It is particularly suitable for handling traffic flows that have mixture type features and ranges. In our study, we construct an MLP with 3 hidden layers. Each layer has 200 units, except for the last hidden layer, which has 400 units. CNNs have good spatial perception abilities and have demonstrated remarkable precision in NID tasks [21]. In this work, we design a CNN with 10 one-dimensional CNN layers, each equipped with 108 filters, with filter size 5. Lastly, we replicate the C-LSTM employed in [23], with our C-LSTM operating on the features that characterize the traffic, instead of operating on raw flows. The C-LSTM combines CNN and LSTM structures to extract spatial and temporal features separately. Data will be first processed by a CNN with 5 hidden layers, then passed to a 2-layer LSTM for final predictions. Each LSTM layer has 160 units. We perform NID, black-box adversarial attacks, and then defend against them based on these models, as we detail next.

We consider NID in two different scenarios, namely *(i)* one-to-all detection and *(ii)* one-to-one detection. The one-to-all scenario groups all types of attacks into a single 'anomaly' class, which leads to a supervised binary classification problem. In contrast, one-to-one detectors separate each network attack (14 in total) into individual classes, and perform multi-class classification. In our study, the same neural network architectures are employed for both scenarios, except for changes in the final layers, as their number directly depends on the number of classes considered for identification. We train and validate all models using 80% of the dataset and test on 20% of it, as customary. All models are trained via minimizing the cross-entropy loss function through the Adam optimizer [47]. Super-sampling is employed to handle class imbalance between benign and malicious traffic, inherent to the dataset. In particular, we randomly choose samples from the minority class (anomalous), which we duplicate in order to ensure the numbers of benign and anomalous samples are equalized prior to training [48].

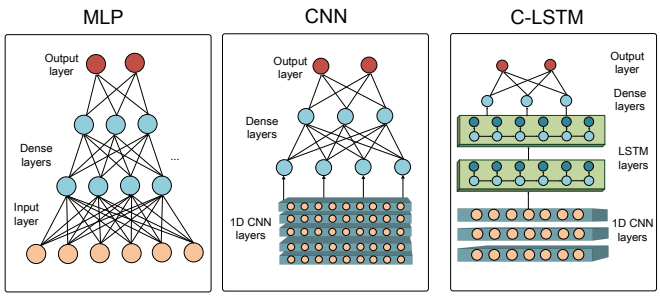


Fig. 3. Architectures of the deep learning-based Network Intrusion Detection (NID) models used in this study.

All models are trained and evaluated on a parallel computing cluster equipped with one or multiple Nvidia TITAN X, Tesla M40 or/and Tesla P100 GPUs. The neural models are implemented in Python using the TensorFlow [49] and TensorLayer [50] packages.

### 5.1 One-to-all NID Performance

We quantify the performance of the NIDS using four metrics, namely accuracy, precision, recall, and F1 score, as shown in Table 2. These metrics are frequently employed for evaluating binary classifiers.

TABLE 2
The detection performance of MLP, CNN, and C-LSTM in the one-to-all scenario.

| | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| MLP | 0.987 | 0.968 | 0.954 | 0.961 |
| CNN | 0.987 | 0.968 | 0.953 | 0.960 |
| C-LSTM | 0.987 | 0.967 | 0.952 | 0.960 |

Observe that all models achieve high detection performance, as all F1 scores are above 0.960. In addition, the three models considered perform similarly, since the difference between the F1 scores attained by each never exceeds 0.01. This matches the performance of state-of-the-art deep learning-based NID solutions, thus the models we use can be considered to be 'reliable'.

### 5.2 One-to-one NID Performance

One-to-one NIDS aim at classifying each traffic flow into 14 types of anomalies and benign. We employ the same neural networks and this time resort to normalized confusion matrices to assess their performance, as shown in Fig. 4. The diagonal elements represent ratios of points for which the predicted label is equal to the true label, while off-diagonal elements indicate misclassification ratios [51]. Therefore, the elements of each row sum to 1. The higher the diagonal values in a confusion matrix, the higher the performance, indicating many correct predictions.

Observe that all NID models achieve high detection accuracy for most types of anomalies, as diagonal values are close to 1. However, taking a closer look at the Brute Force-XSS, SQL Injection, Infiltration, and Brute Force-Web attacks, it appears the NID models tend to misclassify them as 'benign'. In addition, all DNNs face difficulties in dealing with DoS attack-SlowHTTPTest and FTP-Brute Force, as they mix them roughly 50/50. Further, the C-LSTM misclassifies almost all DDoS attack-HOIC traffic as DDoS attack-LOIC-HTTP. This is perhaps less critical, since both attacks belong to DDoS category. Overall, the MLP, CNN, and C-LSTM attain 98.4%, 98.3%, and respectively 98.3% classification accuracy, which matches fairly closely the performance observed in the one-to-all scenario.

In what follows, we demonstrate that **although the NID solutions considered seem reliable in terms of detection accuracy, they can be easily compromised through a se-**

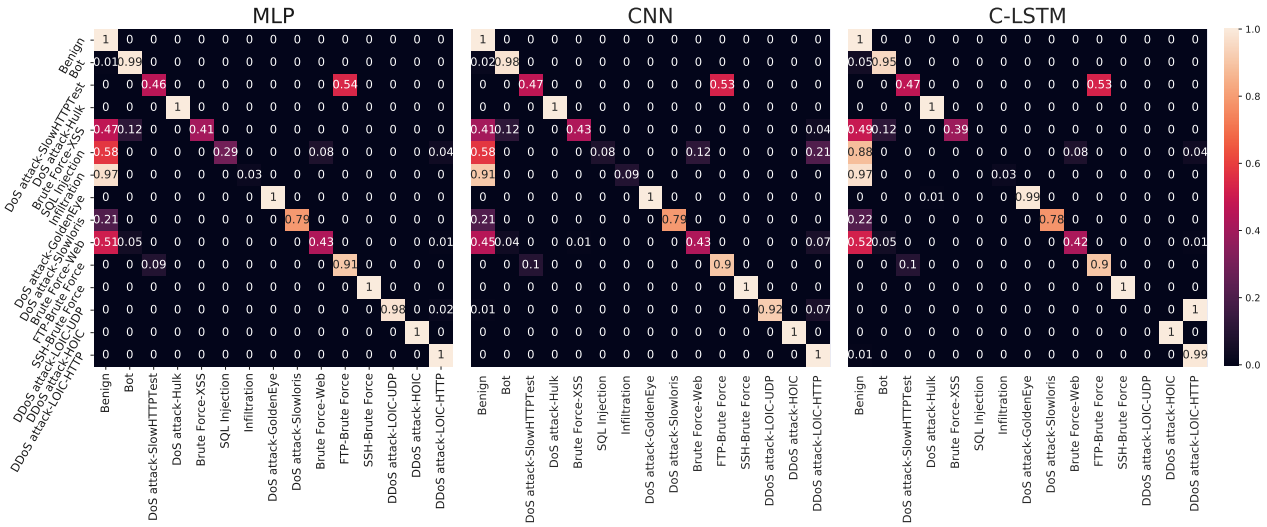


Fig. 4. Confusion matrices of the MLP, CNN, and C-LSTM models for the one-to-one NID.

**quence of perturbations and queries,**[3] **without requiring knowledge about the underlying models**.

## 6 Adversarial Attacks against NIDS

We consider five state-of-the-art black-box attack approaches, which we use to generate adversarial samples and compromise the pretrained deep anomaly detectors discussed in Sec. 5. These include *(i)* Natural Evolution Strategies (NES) [44], *(ii)* BOUNDARY Attack [52], *(iii)* POINTWISE Attack [53], *(iv)* HOPSKIPJUMPATTACK [54], and *(v)* OPT-ATTACK [55], all of which were originally designed to compromise image classifiers. We quantify their performance in terms of different metrics and examine closely the role of different features in the adversarial sample generation process, as well as the decision mechanics of each model.

### 6.1 Black-box Adversarial Attack Methods

We first summarize the operation of each of the adversarial attack techniques we use against NIDS. We note that a plethora of adversarial attack techniques have emerged recently, especially against classifiers in the computer vision and natural language processing (NLP) domains [56], [57], [58]. In our work, we select five representative techniques that have demonstrated state-of-the art performance. Recall that our focus is black-box attacks, which are most realistic in real-world NID scenarios.

**NES [44]** are black-box gradient estimation methods for machine learning models. Estimated gradients can be used for projected gradient descent (as used in white-box attacks) to construct adversarial examples. This approach does not require a surrogate network, thus it is more query-efficient and reliable when crafting adversarial examples. Notably, NES work well in decision-based settings, which makes them suitable for attacks against NID models.

**BOUNDARY Attack [52]** is a method that follows the decision boundary between adversarial and non-adversarial samples via rejection sampling. At each step, it employs constrained i.i.d. samples following a Gaussian distribution, starting from a large perturbation and successively reducing this until successful. This attack is highly flexible and can accommodate a set of adversarial criteria.

**POINTWISE Attack [53]** is a simple decision-based attack method that greedily minimizes the $L_0$-norm between raw and adversarial samples. In image applications, it first introduces salt-and-pepper noise until misclassification, and then repeatedly iterates over each perturbed pixel, resetting it to the initial value if the perturbed image remains adversarial. We implement a similar approach to attack the NID models, but substitute the salt-and-pepper noise with additive Gaussian noise, to better suit network traffic.

**HOPSKIPJUMPATTACK [54]** is a hyperparameter-free, query-efficient attack method, which consists of three main steps: *(i)* estimation of the gradient direction, *(ii)* step-size search via geometric progression, and *(iii)* boundary search via a binary search approach. It is applicable to more complex settings, such as non-differentiable models or discrete input transformations, and achieves competitive performance against several defense mechanisms.

**OPT-ATTACK [55]** projects the decision-based attack into a continuous optimization problem and solves it via randomized zeroth-order gradient update. In particular, a Random Gradient-Free (RGF) method is employed to find appropriate perturbations and converge to stationary points. Since OPT-ATTACK does not rely on gradients, it can attack other non-differentiable classifiers besides neural networks, e.g., Gradient Boosting Decision Trees.

We employ a modified version of the mean absolute percentage error to quantify the deviation between each

3. Arguably previously unseen types of malicious traffic could go undetected by neural models that exhibit remarkable performance, as those consider herein. We note however that our focus is not to verify the ability of such models to detect new types of threats, but rather to show their susceptibility to manipulations of behaviors already learned, and how this can be remedied.

TABLE 3
Statistics of the dataset used to generate adversarial samples.

| Attack Type | Number of Instances | Ratio [%] |
|---|---|---|
| Bot | 5,217 | 10.434 |
| DoS attack-SlowHTTPTest | 5,217 | 10.434 |
| DoS attack-Hulk | 5,217 | 10.434 |
| Brute Force-XSS | 51 | 0.102 |
| SQL Injection | 24 | 0.048 |
| Infiltration | 5,217 | 10.434 |
| DoS attack-GoldenEye | 5,217 | 10.434 |
| DoS attack-Slowloris | 2,475 | 4.950 |
| Brute Force-Web | 117 | 0.234 |
| FTP-Brute Force | 5,217 | 10.434 |
| SSH-Brute Force | 5,217 | 10.434 |
| DDoS attack-LOIC-UDP | 383 | 0.766 |
| DDoS attack-HOIC | 5,217 | 10.434 |
| DDoS attack-LOIC-HTTP | 5,214 | 10.428 |
| Total | 50,000 | 100.000 |

unmodified sample $x$ and its adversarial version $x_{\text{adv}}$, i.e.,

$$\textbf{MAPE}(x, x_{\text{adv}}) = \frac{100\%}{N} \sum_{k=1}^{N} \left| \frac{x^{(k)} - x_{\text{adv}}^{(k)}}{x^{(k)}} \right|, \qquad (1)$$

where $N$ is the total number of perturbed features in $x$ and $x^{(k)}$, $x_{\text{adv}}^{(k)}$ are the $k^{\text{th}}$ features of the original and adversarial samples respectively. Smaller MAPE indicates higher similarity between the raw input $x$ and adversarial sample $x_{\text{adv}}$.

We randomly select 50,000 malicious traffic flows from the test set, to craft adversarial samples. We summarize the statistics of these samples in Table 3.

We quantify the performance of each attack approach using four performance metrics, namely Attack Success Rate (ASR), average benign confidence, MAPE, and average number of queries. The ASR is widely used to assess the effectiveness of adversarial attacks against DNNs [24] and is measured by the ratio between the number successful adversarial samples and the total attack attempts (in our case 50,000). An attack attempt is successful if and only if the underlying algorithm converges, and the adversarial

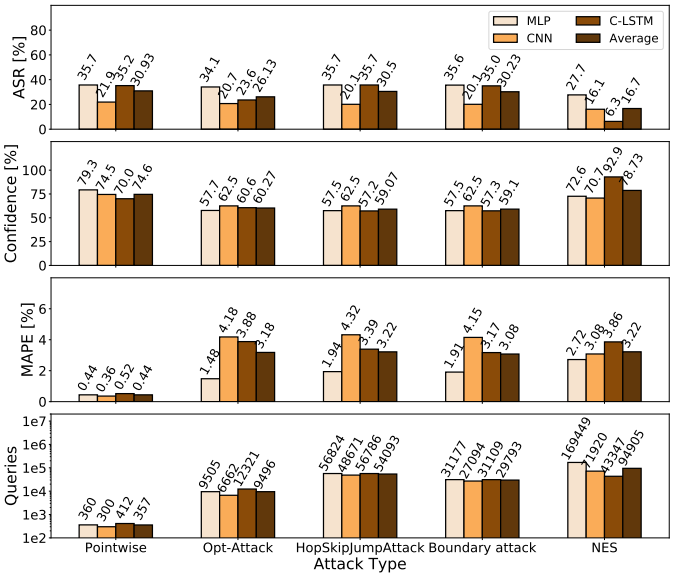


Fig. 5. ASRs, Confidence, MAPE, and number of queries of all attack approaches against the 3 NID models considered, and their average values in the one-to-all scenario.

samples meet the constraints discussed in Sec. 3.2. The average benign confidence denotes the probability that the model predicts an adversarial sample $x_{\text{adv}}$ as benign. In particular, the last layer of the classifiers comes with a softmax function, whose output represents a probability distribution over predicted output classes, which is our interest. Higher confidence implies that the model is more confident about the decision made over a sample. MAPE is defined in Eq. (1) and is computed over 22 features that allow perturbations. Recall that lower MAPE represent higher similarity between the raw and adversarial samples. The number of queries indicates how many attempts an attacker should perform in order to generate a successful adversarial sample. This can be used to measure the efficiency of an attack approach. Higher number of queries might trigger the NIDS, making the attack easier to be detected. Note that the MAPE, benign confidence, and number of queries are averaged over the successful attack attempts for each attack approach and NID model. All attacks are conducted using the original implementations and Python Foolbox [59].

### 6.2 Attack Performance in One-to-all Scenario

We show the performance of all attack approaches in the one-to-all detection scenarios in Fig. 5. Observe that the BOUNDARY, POINTWISE, and HOPSKIPJUMPATTACK obtain similar performance in terms of success rates for all NID models. Worryingly, these approaches can generate adversarial samples with a 30% success rate on average, which makes them a serious threat to deep NIDS. In particular, the POINTWISE attack achieves the highest benign confidence, lowest MAPE, and requires the fewest number of queries. This implies that the POINTWISE attack is highly efficient in generating adversarial samples, and more difficult to defend against. Though less efficient and effective, the NES attack generates adversarial samples with high benign confidence, therefore it appears the best in evading the NID models. On the other hand, the performance of the OPT-ATTACK appears moderate, as it does not stand out in terms of any metric. Turning attention to the horizon of NID models, CNN appears to be the most robust model against black-box attacks in the one-to-all scenario, as it exhibits the lowest ASR (19.78%) among the three. In addition, attackers are required to make larger changes to the raw samples, in order to subvert the CNN, as the average MAPE appears the highest compared to the MLP and C-LSTM. The benign confidence and number of queries are however similar between models.

We delve deeper into the attack performance, by showing the ASR over each type of malicious traffic in Fig. 6. Observe that it is almost impossible for adversarial samples generated for the DoS attack-Hulk and DDoS attack-LOIC-UDP to bypass the NIDs model, as their ASR is close to 0%. On the contrary, adversarial samples for Brute Force-XSS, SQL Injection, Infiltration, Brute Force-Web, and DDoS attack-HOIC appear more likely to evade the NID models. Notably, the robustness and vulnerability to a specific type of attack may vary between models. For example, no adversarial samples of the DoS attack-SlowHTTPTest can bypass CNN, while its ASR with MLP is mostly over 90%. **This offers useful insights to network service providers to defend against specific types of attacks**.

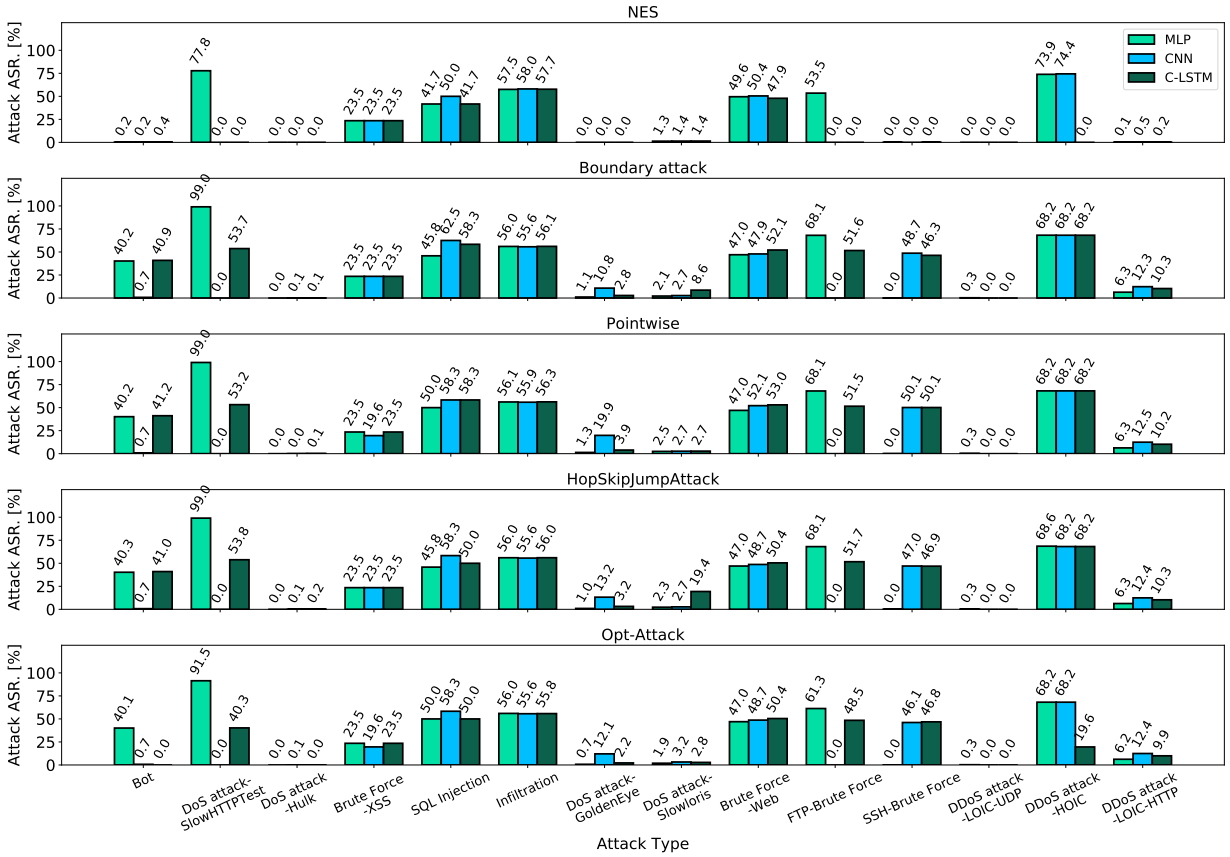


Fig. 6. ASRs over each type of malicious traffic against all NID models in the one-to-all scenario.

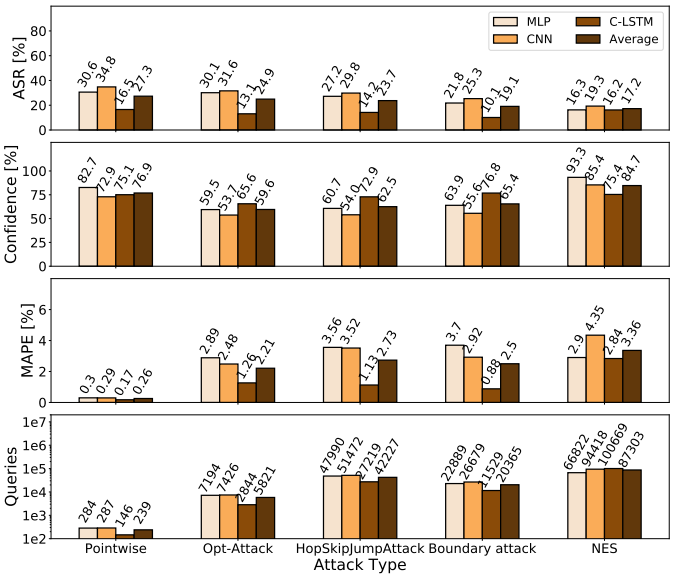


Fig. 7. ASRs, Confidence, MAPE, and number of queries of all attack approaches against the three NID models considered, and their averages, in the one-to-one scenario.

### 6.3 Attack Performance in One-to-one Scenario

We illustrate the statistics of each attack against the different NID models considered, in Fig. 7. Observe that, except for the NES where the performance is similar among the different NID models, the ASR varies among models for all the other attack methods (unlike in the one-to-all scenario discussed in the previous subsection). This is because the models work with large number of classes, which makes it difficult to craft adversarial samples to match the targeted 'benign' label. The POINTWISE method obtains the highest ASR, lowest MAPE, and lower average number of queries. This suggests that this approach is effective and efficient in one-to-one settings. The C-LSTM appears to be the most robust model against adversarial samples, as all attack methods attain the lowest ASR values against this NID model. Although achieving the highest benign confidence with adversarial samples, NES obtain the lowest ASR on average. In general, they also require more queries to craft an adversarial sample.

In Fig. 8, we show the ASR for each type of malicious traffic flow considered, in the same one-to-one scenario. Analyzing these results jointly with Figure 4, observe that anomalies with low detection rate (i.e., Brute Force-XSS, SQL Injection, Infiltration, Brute Force-Web) are easier to be disguised by attackers. This is because the models already have vague decision boundaries for these flow types, thus are easier to be gamed. Attackers obtain the lowest ASR when crafting adversarial samples based on DoS attacks-Hulk, -GoldenEye, -Slowloris, and DDoS attack-LOIC-UDP, as the NID models exhibit high detection rates over these anomalies. Overall, most of attacks achieve similar ASR performance as they obtain in the one-to-all scenario.

### 6.4 Adversarial Samples Analysis

#### 6.4.1 *Cross-Transferability*

We define the transfer ratio $\varepsilon_{m_1}^{m_2} = K_{m_1}^{m_2}/K_{m_1}$ between models $m_1$ and $m_2$, to evaluate the transferability of adversarial

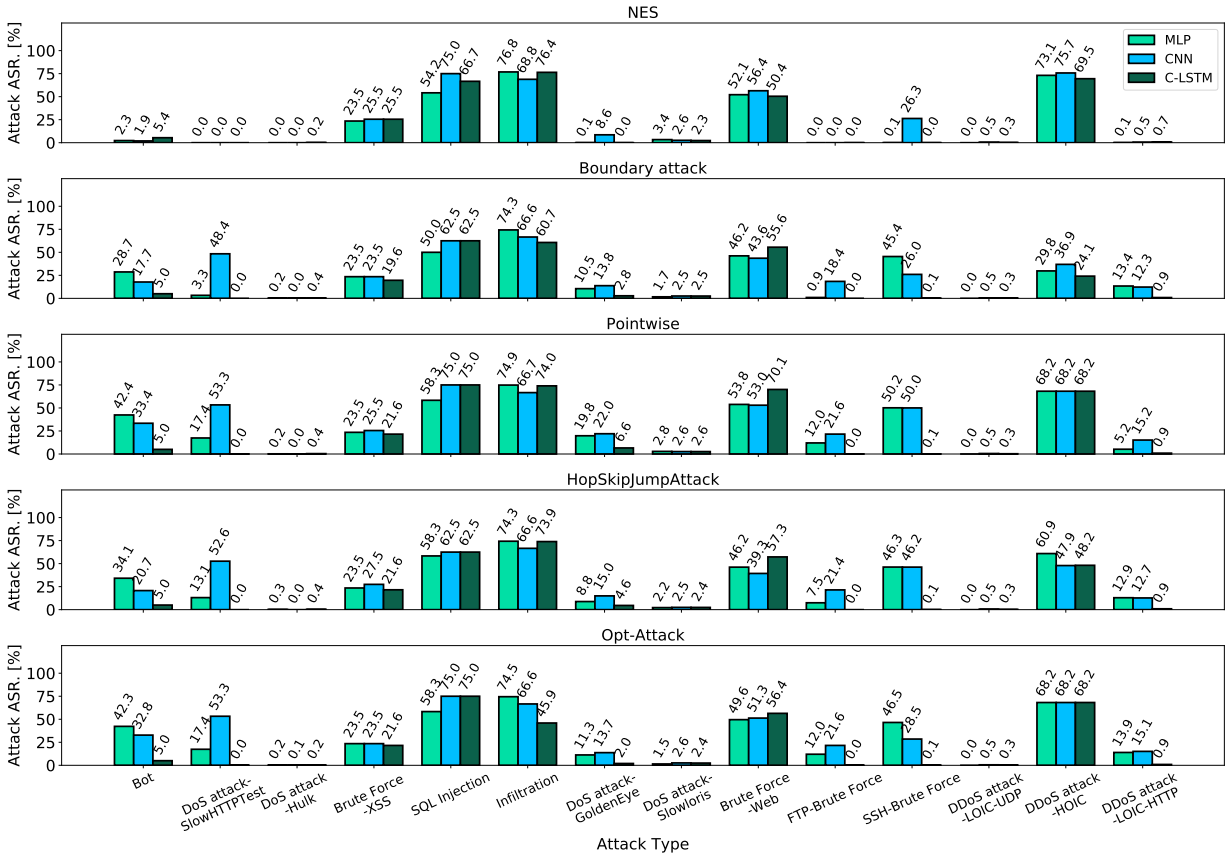


Fig. 8. ASR with different types of attacks against all NID models considered in the one-to-one scenario.

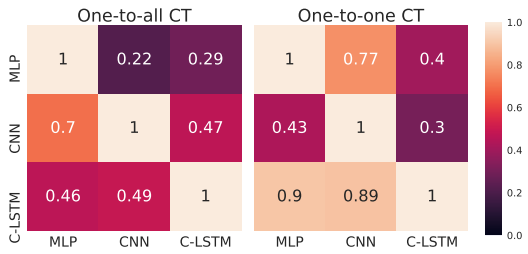


Fig. 9. The cross-transferability (CT) matrix across all NID models.

samples across different NID models. Here, $K_{m_1}$ denote the number of successful adversarial samples crafted for model $m_1$, while $K_{m_1}^{m_2}$ denotes the number of samples among $K_{m_1}$ that can bypass $m_2$ as well. We show the transfer ratios across models as Cross-Transferability (CT) matrices in Fig. 9. An element in the $i^{\text{th}}$ row and $j^{\text{th}}$ column represents the value of $\varepsilon_{m_i}^{m_j}$ (e.g., the value on the first row of the second column indicates that 22% of adversarial samples crafted for MLP can bypass CNN.) Note that $\varepsilon_m^m = 1$ holds for all $m$.

Observe that a large proportion adversarial samples are transferable across NID models for both detection scenarios. This has been also confirmed with adversarial attacks against image classifiers [60], and implies that, even with completely different structures, NID models suffer from similar weaknesses, which attackers can exploit. For example, assume a network service provider (NSP) employs the C-LSTM model to perform one-to-one NID. To compromise the system, attackers manage to craft $K$ successful adversarial malicious traffic flows. After being detected, the NSP changes the NID model to MLP. Nevertheless, 90% of the old adversarial traffic can still bypass the new NID system due to the high cross-transferability.

### 6.4.2 *Feature-wise MAPE*

We delve deeper into the adversarial samples generated, by showing in Fig. 10 the average MAPE of each perturbed feature on all successful attack samples, across all NID models and attack approaches. Observe that for both detection scenarios, the Active/Idle Time (i.e., the time a flow was idle before becoming active and amount of time a flow was active before becoming active) are less affected, as the related features remain almost unchanged in the attack process. In contrast, those **features that characterize the average number of bytes and packets sent in the forward and backward directions in the initial window or/and sub flows, are perturbed more significantly**. This indicates that these features are the most influential in the decision of NID models, and therefore **more likely to be exploited by potential attackers**.

### 6.4.3 *t-SNE Visualization*

We also investigate the inner workings of each NID model, by visualizing the output embedding of their hidden layers, so as to understand better how a neural network 'thinks' of the benign, malicious, and adversarial samples. To this end, we adopt the t-distributed Stochastic Neighbor Embedding (t-SNE) [61] to reduce the dimension of the last hidden layer of each model to 2. In Fig. 11, we plot the t-SNE embedding ($x, y$ axes) of the hidden representations of 10,000 benign samples (blue), 10,000 adversarial samples (green) generated by each attack method, and their corresponding anomalous samples used to craft them (pink), along with their benign confidence ($z$ axis). Note that a sample will be considered benign iff the benign confidence is greater than 0.5 (above the decision plane). Typically, the t-SNE approach

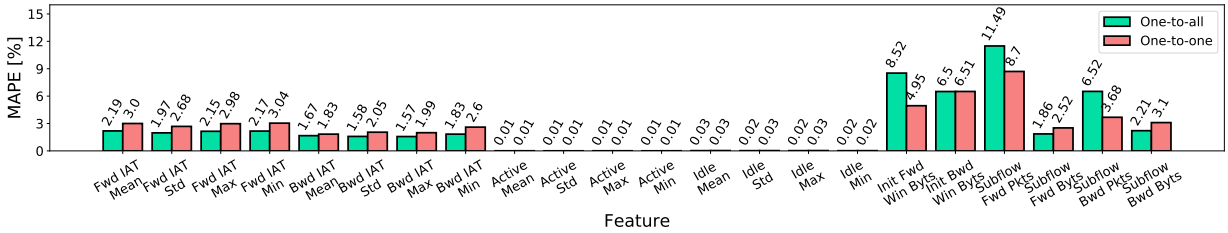


Fig. 10. The MAPE between original and adversarial samples of each feature that allows perturbations.

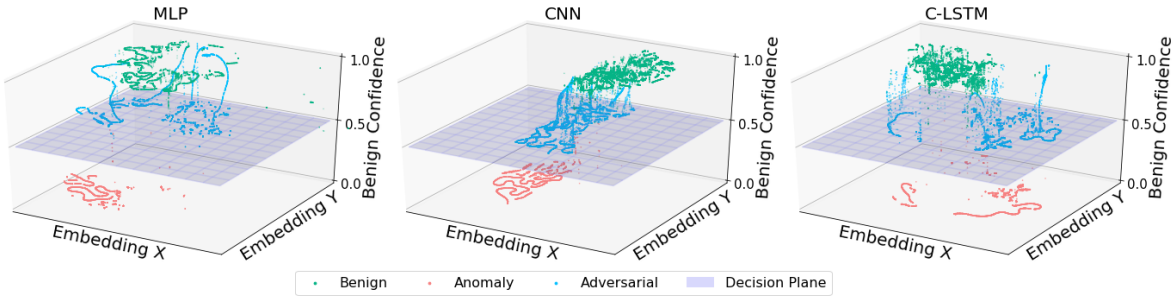


Fig. 11. Two-dimensional ($x, y$ axes) t-SNE embedding of the representations in the last hidden layer, along with the benign confidence ($z$ axis) of each NID model. Data generated using 10,000 benign samples (blue), 10,000 adversarial samples (green) produced by all attack methods, and their corresponding original malicious samples (pink).

organizes data points that have higher similarity into nearby embeddings [62]. It can therefore reflect how the model 'thinks' of the data samples, as similar data representations will be clustered together.

Observe that anomalous samples can be clearly distinguished from benign samples by their t-SNE embeddings for all NID models. The purpose of adversarial attacks is to cause misclassification by bringing malicious samples across the decision boundary. This is reflected in Fig. 11, as the t-SNE embedding of adversarial samples are moved closer to the benign embedding cluster, while they remain anomalies in nature. This successfully confuses the NID models, making the adversarial samples indistinguishable. In addition, adversarial samples with higher benign confidence are closer to the benign embedding cluster. This is especially clear in the t-SNE embedding produced by CNN.

## 7 Defending Against Adversarial Attacks

Defense mechanisms against adversarial attacks should improve the robustness of deep learning models to adversarial samples, such that they become less likely to be compromised and the ASR of different attacks is reduced. In general, countermeasures for adversarial examples can be categorized into two types [11]: *(i)* **Reactive** – detecting adversarial examples after DNNs have been trained; and *(ii)* **Proactive** – improving the robustness of DNN models against adversarial examples. In this paper, we propose three different defense mechanisms, and combine them to counteract the adversarial samples generated by the black-box attack methods discussed in the previous section. These defense mechanisms include:

1) **Model Voting Ensembling (Proactive):** Ensembling pretrained MLP, CNN, and C-LSTM using a voting mechanism, to construct stronger models that are less susceptible to misclassification of adversarial samples;
2) **Ensemble Adversarial Training (Proactive):** Augmenting the training dataset with adversarial samples, and retraining the NID models, thereby reinforcing their capabilities against adversarial samples;
3) **Adversarial Query Detection (Reactive):** Detecting the query process in the black-box attack process, to blacklist the attacker's IP address before they may succeed.

In what follows, we detail the proposed defense mechanisms, and demonstrate their effectiveness.

### 7.1 Model Voting Ensembling

The experiments we reported in Sec. 6 suggest that an attacker can successfully compromise a NID model with up to 35% ASR. However, only a small set of adversarial samples can bypass all three NID models simultaneously. This motivates us to construct a new ensembling model [63], [64] by combining all of these structures, to strengthen the barrier against potential attacks. Specifically, for each input traffic flow, we gather the decisions of all NID models individually, and make the classification using a voting process. A flow will be classified as 'benign' if all models reach consensus, i.e., all of them classify it as 'benign'. Otherwise, the traffic flow will be regarded as an 'anomaly'. We recognize several advantages of using such model voting ensembling as means of defense:

1) In order to construct a successful adversarial sample, attackers need to defeat all NID models simultaneously, which is much harder than compromising a single one;
2) The voting mechanism makes the entire model non-differentiable, thus attack approaches that rely on model gradient estimation (e.g., NES) will be obstructed;
3) The voting mechanism is easy to implement, as it does not require to re-train the original NID models.

The proposed model voting ensembling method is a proactive approach, as it improves the robustness of the pretrained models against adversarial samples. We show the NID performance of the ensembling model for the one-to-all scenario in Table 4 and the confusion matrix for the one-

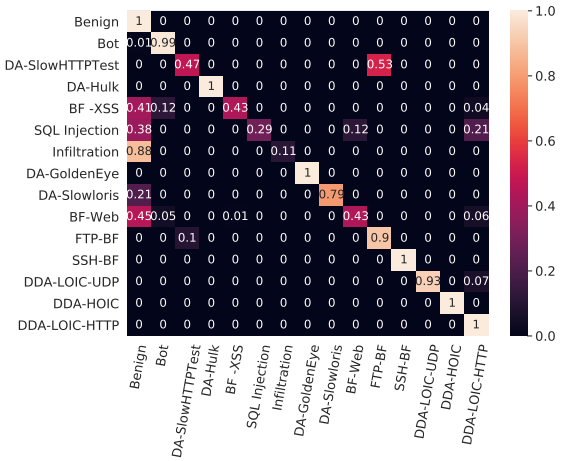


Fig. 12. The confusion matrix of the ensembling model in the one-to-one detection scenario.

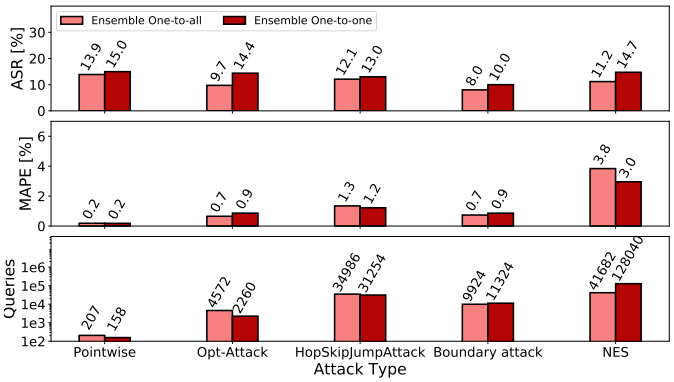


Fig. 13. ASRs, MAPE, and number of queries statistics of all attack approaches against ensembling models in one-to-all/-one scenarios.

to-one scenario in Fig. 12. Revisiting Table 2 and Figure 4, observe that **the ensembling model obtains very close performance compared to its individual components in both detection scenarios, while achieving lower false negative rates**, since it requires consensus to make the 'benign' classification decision.

TABLE 4
One-to-all NID performance of ensembling model.

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| Ensembling | 0.987 | 0.964 | 0.954 | 0.959 |

We re-run the same five black-box attacks considered previously over the same set of 50,000 malicious samples and show statistics of their performance in Fig. 13. Note that the benign confidence measure is abandoned, since the outputs of ensembling models are no longer probabilities. Jointly analyzing these results with Fig. 7 and Fig. 5, observe that the ASR of each attack approach against the ensembling model has dropped relatively to when attacking each of the model's component (i.e., MLP, CNN, and LSTM). In the best case, the BOUNDARY approach obtains 20.1% ASR in attacking the CNN-based NIDS in the one-to-all scenario, while its success rate is merely 8.0% when attacking the ensembling model. Regarding the one-to-one scenario, the reduction in ASR is also substantial. On average, the ensembling models lead to 17.12% and 9.02% drop of ASR for one-to-all and one-to-one scenarios respectively. This indicates that the voting ensembling mechanism is an effective defense approach. Turning attention to MAPE, observe that adversarial samples crafted against ensembling models yield low MAPE, which suggests that **this defense mechanism applies hidden and tighter constraints to the adversarial samples, to prevent them from deviating excessively from the raw input samples, which in turn improves detection**.

We also show in Figs 14 and 15 the ASR on a malicious traffic type basis, when crafting adversarial samples against the ensembling model, for both scenarios. Observe that the voting ensembling mechanism successfully defends 9 type of adversarial samples, i.e., Bot, DoS attack-SlowHTTPTest, DoS attack-Hulk, DoS attack-GoldenEye, DoS attack-Slowloris, FTP-BruteForce, SSH-Bruteforce, DDoS attack-LOIC-UDP and DDoS attack-LOIC-HTTP, as their ASR is virtually 0%. **For attacks such as botnet this is critical, since any success could have catastrophic effects, which our ensembling technique thwarts.** For other types of malicious traffic, the ASR also drops by varying degrees, but not as significant, which calls for further defenses, as we show next.

### 7.2 Ensemble Adversarial Training (EAT)

As discussed in Sec. 3, white-box strategies are not commonly accessible to external adversaries seeking to compromise NIDS, as the training data, model structures and parameters are opaque. However, recent literature confirms that adversarial samples are adaptable across different attack methods and victim models [37], [65], [66]. Therefore, from the defenders' points of view, adversarial samples generated using white-box attacks can be exploited to improve the robustness of NID models, so as to defend against potential adversarial samples. Therefore, we employ the Ensemble Adversarial Training (EAT) as an additional defense approach [37], which augments the training data with adversarial examples generated by white-box attacks crafted on other static pre-trained NID models. Subsequently, the original NID models are reinforced by re-training on the augmented dataset. We expect that, with the proposed re-training, the NID models learn to classify adversarial samples better and thus become more resilient to attacks. The principle behid EAT is illustrated in Fig. 16.

#### *7.2.1 Reinforcing NID with White-box Adversarial Samples*

We randomly select 250,000 malicious flows to generate adversarial samples using three state-of-the-art white-box attack approaches: Fast Gradient Sign Method [67], Iterative Attack (I-FGSM) [68] and Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [69]. The FGSM-based approaches perform one step gradient update along the direction the gradient at each feature that allows perturbations, and introduce noise following that direction. I-FGSM extends the FGSM by running a finer optimization for multiple iterations to generate a valid adversarial sample. MI-FGSM introduces a momentum term into the iterative process of I-FGSM, which helps stabilizing the update directions and escaping from poor local maxima. This leads to more transferable adversarial samples. We show statistics of malicious traffic samples used for white-box attacks, number of successful adversarial samples, and their ratios

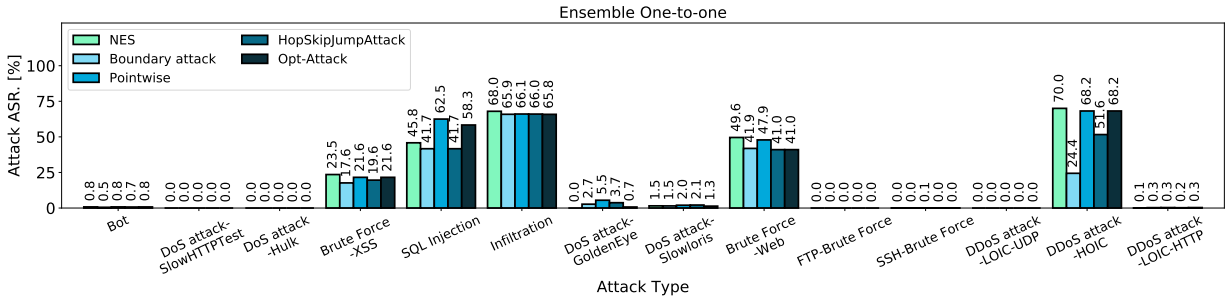


Fig. 14. ASRs of each type of attack against the model voting ensemble technique in the one-to-one scenario.

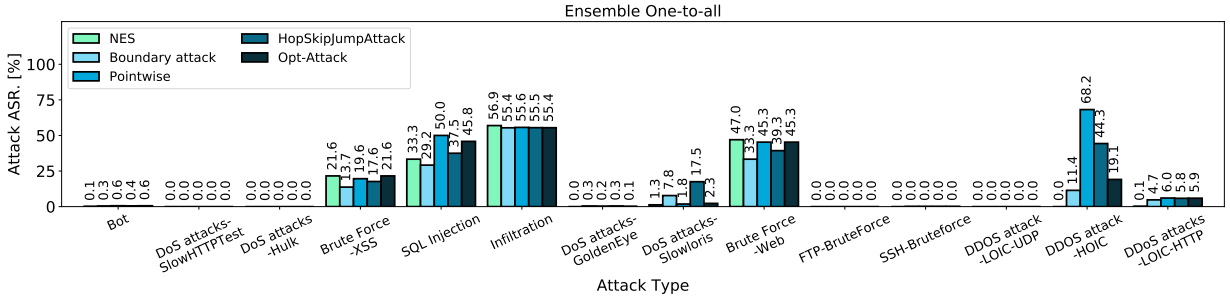


Fig. 15. ASRs of each type of attack against the model voting ensemble technique in the one-to-all scenario.

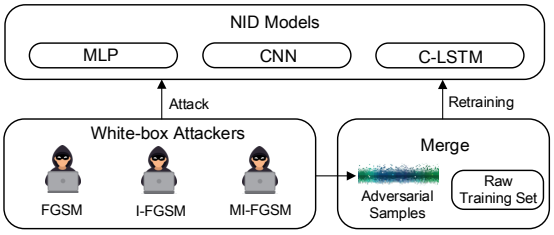


Fig. 16. An illustration of the EAT defense approach.

TABLE 5
Statistics of the malicious traffic flows used to generate adversarial samples (shared set)/total number of adversarial samples successfully generated by all methods for EAT, and the ratios of each attack with respect to the total, using white-box attacks

| Attack Type | Number of Instances | Ratio [%] |
|---|---|---|
| Bot | 26,601/426,755 | 10.640/12.356 |
| DoS attack-SlowHTTPTest | 26,601/317,338 | 10.640/9.188 |
| DoS attack-Hulk | 26,601/311,740 | 10.640/9.026 |
| Brute Force-XSS | 179/2,530 | 0.072/0.073 |
| SQL Injection | 63/1,083 | 0.025/0.031 |
| Infiltration | 26,601/473,116 | 10.640/13.698 |
| DoS attack-GoldenEye | 26,601/349,907 | 10.640/10.131 |
| DoS attack-Slowloris | 8,515/112,142 | 3.406/3.247 |
| Brute Force-Web | 494/8,430 | 0.198/0.244 |
| FTP-Brute Force | 26,601/287,768 | 10.640/8.332 |
| SSH-Brute Force | 26,601/404,402 | 10.640/11.708 |
| DDoS attack-LOIC-UDP | 1,347/2,423 | 0.539/0.070 |
| DDoS attack-HOIC | 26,601/401,923 | 10.640/11.637 |
| DDoS attack-LOIC-HTTP | 26,594/354,402 | 10.638/10.261 |
| Total | 250,000/3,453,959 | 100/100 |

between each type and their fraction of the totals, in Table 5. Note that the adversarial sample numbers are summed over all white-box attacks, all models, and both detection scenarios.

Due to the *information asymmetry* between attackers and defenders, the defenders do not have knowledge about which features will be perturbed for attack purposes. We

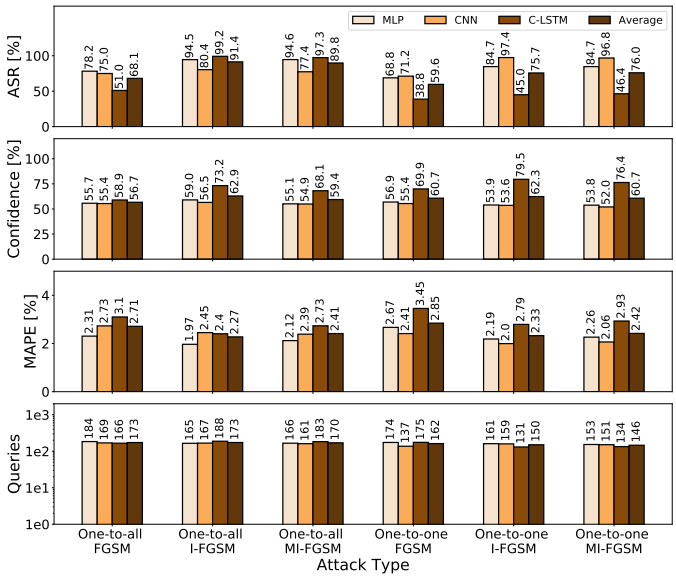


Fig. 17. ASRs, Confidence, MAPE, and number of queries statistics of all white-box attack approaches against the three NID models considered in this study, and their average values for both NID scenarios.

therefore relax the feature constraints (see Sec. 3.2) for perturbations in the white-box setting. However, the constraints over MAPE ($\leq$ 20%) are retained, to restrict the scale of the perturbations and ensure the semantics and labels of the original flow samples do not change. Note that the adversarial samples generated by white-box attacks are not necessarily valid traffic flows, as they are only employed for training purposes. We gather successful adversarial samples generated by all white-box attack methods (i.e., FGSM, I-FGSM, and MI-FGSM), crafted with all NID models (i.e., MLP, CNN, and C-LSTM) in both detection scenarios (i.e., one-to-all and one-to-one) and combine these with the original training data, to build an augmented dataset for EAT.

We show the performance of each white-box attack in Fig. 17. Observe that since the NID models are transparent,

TABLE 6
Performance of MLP, CNN, C-LSTM, and ensembling model after EAT in the one-to-all scenario.

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| MLP | 0.987 | 0.968 | 0.953 | 0.960 |
| CNN | 0.986 | 0.959 | 0.954 | 0.956 |
| C-LSTM | 0.985 | 0.953 | 0.955 | 0.954 |
| Ensembling | 0.983 | 0.943 | 0.956 | 0.949 |

TABLE 7
Ratio of adversarial samples that bypass each NID model after EAT.

| Scenario | MLP | CNN | C-LSTM | Ensembling |
|---|---|---|---|---|
| One-to-all | 40.04% | 53.15% | 48.43% | 81.54% |
| One-to-one | 42.45% | 38.06% | 38.26% | 43.31% |

and looser constraints are applied to adversarial samples, the ASR for all white-box attacks is significantly higher than their black-box counterparts. White-box attacks also require fewer queries to generate adversarial samples. Fortunately, attackers normally do not have access to the NID models. The ASR when crafting each type of anomaly is shown in Fig. 18 in the – this is close to 100% for most anomalies.

#### 7.2.2 NID Performance of Post-EAT Models

In Table 6, we report the detection performance on the same test set after EAT, for the one-to-all scenario. Compared to NID models prior to the EAT (See Tables 2 and 4), the detection performance of the newly trained models has dropped slightly in terms of accuracy, precision, and F1 score. However, the recall rate of each model has improved. This indicates that the models are prone to classifying some ambiguous samples as anomalies, which results in higher false positive and lower false negative rates. Similar phenomena are also observed in the one-to-one scenario. The accuracy for the MLP, CNN, C-LSTM, and the ensembling model appears worse than what was achieved prior to EAT. However, by taking a closer look at their confusion matrices in Fig. 19, **post-EAT models achieve high detection rates on most anomalies they failed to detect previously** (i.e., Brute Force-XSS, SQL Injection, and Brute Force-Web). This suggests that EAT has improved the robustness of NID models, making them more sensitive to anomalous traffic that is difficult to classify.

#### 7.2.3 Robustness to Old Adversarial Samples

In Table 7, we further show the ratio of adversarial samples crafted from the models before EAT, which can compromise the corresponding post-EAT models. Observe that EAT also makes each model more resilient to old adversarial samples, as those ratios are significantly below 100%. In particular, only 38.06% of adversarial samples crafted from the old C-LSTM can bypass the Ensemble Adversarial Trained C-LSTM. This means that EAT enables each model to learn to characterize adversarial samples generated using white-box attacks, and therefore fixes some 'bugs' present in the old setting.

This effect is confirmed by their t-SNE embedding. In Fig. 20, we show the two-dimensional ($x, y$ axis) t-SNE embedding of the representations in the last hidden layer along with the benign confidence ($z$ axis) of each NID models after the EAT, as similar to Fig. 11. Note that we employ the same set of samples in Fig. 11 to generate the new Fig. 20. Observe that after the EAT, some old adversarial samples are rejected by the new models (purple), as they are below the decision boundary. It appears that the EAT pushes certain adversarial samples away from the benign clusters, such that they become more separable. Even though some adversarial samples still escape, their benign confidence becomes lower compared to what was observed in Fig. 11. These means that the new models are more suspicious of these new data samples after the EAT. One can raise the decision boundary to filter such samples.

#### 7.2.4 The Effect of EAT

In Fig. 21, we show the ASR for each attack after EAT ($ASR_{EAT}$, bars in the upper part of the plot), and the ASR reduction compared to the case before EAT was applied (ASR - $ASR_{EAT}$, bars in the lower part) for the one-to-all scenario. In the figure, positive numbers below the x-axis indicate that the $ASR_{EAT}$ has dropped after EAT was employed. We observe that the ASR of each attack drops when directed against most of the NID models, which means EAT successfully improves their robustness.

On the other hand, we also observe that the $ASR_{EAT}$ of NES increases by 12.5% when crafting from LSTM in the one-to-all scenario. This also weakens the ensembling model, as the $ASR_{EAT}$ increases accordingly. Nevertheless, the **EAT remains an effective defense approach, as it reinforces each NID model and blocks black-box attack attempts in most of the cases.**

Similarly, in Fig. 22, we show the ASR for each attack after EAT ($ASR_{EAT}$, bars in the upper part of the plot), and the ASR reduction compared to the case before EAT was applied (ASR - $ASR_{EAT}$, bars in the lower part) for the one-to-one NID scenario. We observe that ASR of each attack drops for most of the models. This means that **EAT successfully improves the robustness of each model, making them more difficult to be compromised**. On average, the ASR drops to 6.70% and 5.78% for one-to-one and one-to-all NID scenarios, respectively. **This is particularly beneficial to practically mitigating DoS and brute-force type attacks.** On the other hand, EAT is not a silver bullet for all the cases. For example, the $ASR_{EAT}$ of NES increases when crafting from C-LSTM in the one-to-all scenario, as seen in Fig. 21.

### 7.3 Adversarial Query Detection

Recall that all black-box attack methods rely on continuous queries to the victim model and feedback received. Based on the feedback, the attackers learn to adjust the perturbations added to the input, so as to compromise detection. The scale of perturbations is usually small, so that they to do not change the essence of the original input. Therefore, the queries in the same attack round are typically with high similarity. **This inherent similarity between queries can be harnessed to detect an attack**. Therefore, we explore query detection [70] as the final defense mechanism. Once queries have been discovered, the NIDS can blacklist the attackers' IP addresses, to prevent potential threats.

Specifically, for each IP address, we construct a buffer with size $B$ to store the features of the traffic flows originating from that address in a pre-defined period. To reduce the dimension of the features saved and model the similarity degree between flows, we employ a deep similarity

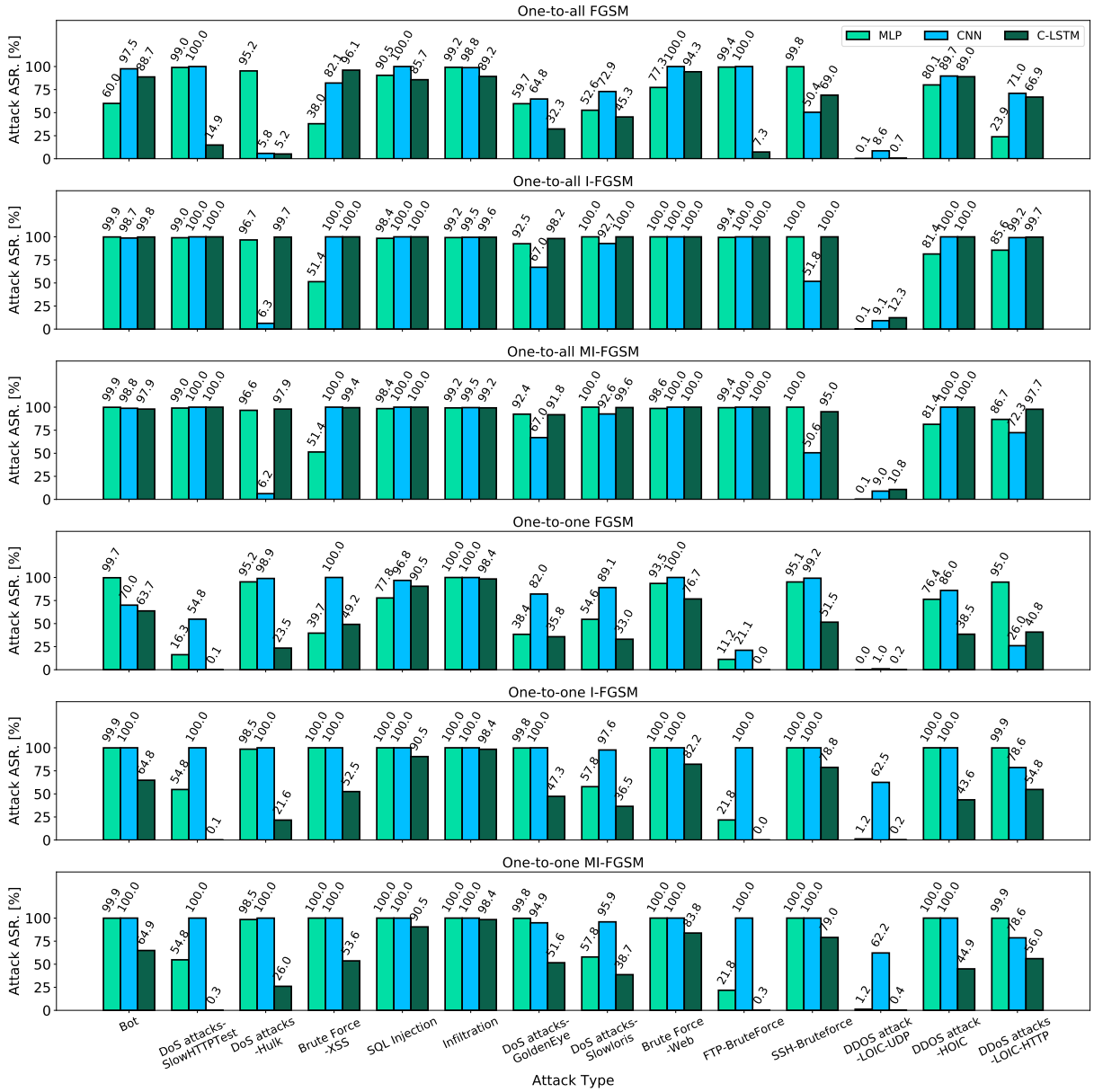


Fig. 18. ASRs over each type of white-box attack against all models for both NID scenarios.

encoder (DSE) [71], encoding similar traffic flows in a lower-dimensional space with shorter $l_2$ distance. More precisely, for each new flow $x$ sent from a given IP address, we compute the pairwise distance between the embedding of this flow and others in the buffer, calculating the $k$ nearest neighbor average distance $d_x^k$. If $d_x^k$ is lower than a threshold $\tau$, i.e., $d_x^k < \tau$, this suggests that that IP address has sent an excessive number of similar traffic flows, which can be considered as queries in an ongoing attack. When this happens, the IP address can be blacklisted and thus the potential threat eliminated. We show the underlying principle of the query detection mechanism in Fig. 23, which bears $O(1)$ complexity as it only depends on the buffer size.

After an attack is detected, the buffer associated to the specific IP address can be cleared. In addition, when query detection suggests a potentially malicious actor, their IP address can be banned either immediately, or after subsequent queries, as suggested in [71]. This can minimize an attacker's knowledge of the time when their attack was detected, therefore reducing the probability of compromising the query detection mechanism.

### 7.3.1 *Deep Similarity Encoder*

The core component of the query detection-based defense mechanism is the deep similarity encoder (DSE) [71], which is a neural network that reduces the dimension of the input. After embedding by a DSE, dissimilar flows will be far from each other in the encoded space, while similar queries will be close. Thus, queries and traffic flows become more distinguishable.

For the DSE, we employ a CNN similar to that in Fig. 3, only replacing the last layer with 3 units. This means that the embedding of each traffic flow is a 3-dimensional vector. We denote $e_i = \text{DSE}(x_i)$ as the embedding of the input sample $x_i$. The DSE can be trained via minimizing the following

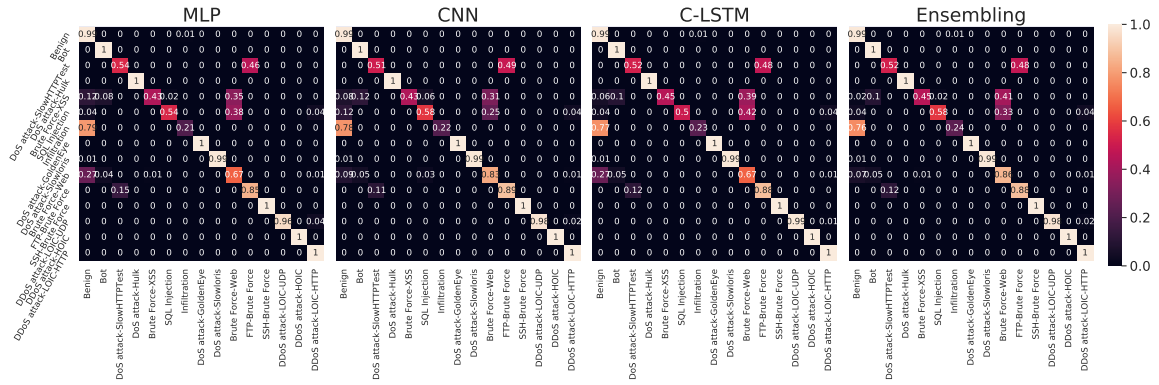


Fig. 19. Confusion matrices of MLP, CNN, C-LSTM, and ensembling model in one-to-one NID, after EAT.

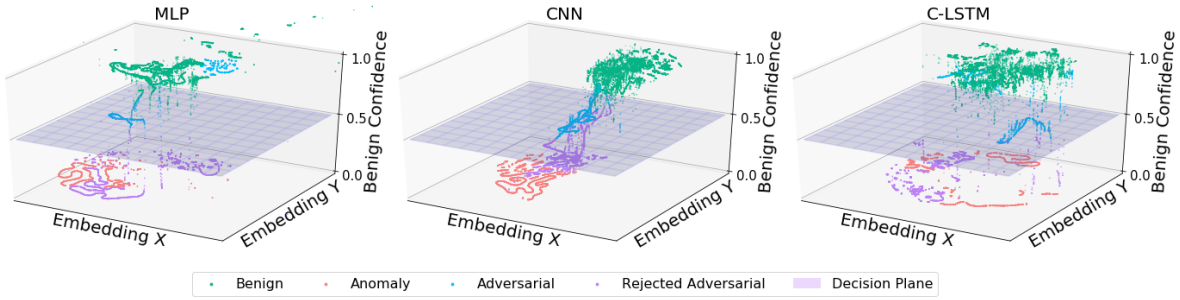


Fig. 20. Two-dimensional ($x, y$ axis) t-SNE embedding of the representations in the last hidden layer along with the benign confidence ($z$ axis) of each NID models after EAT. Data generated using 10,000 benign samples (blue), 10,000 adversarial samples produced by all attack methods that successfully bypass the model (green) and are rejected by the model (purple), with their corresponding malicious samples they craft from (pink).

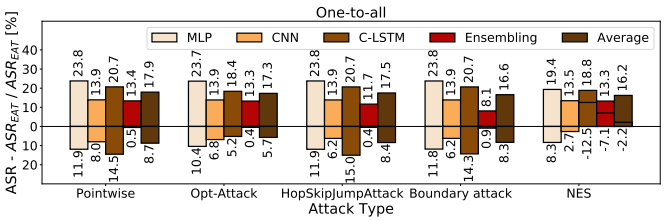


Fig. 21. ASR of each attack after EAT (bars above x axis), and ASR reduction compared to when no defense is applied (bars below x axis) in the one-to-all NID scenario.

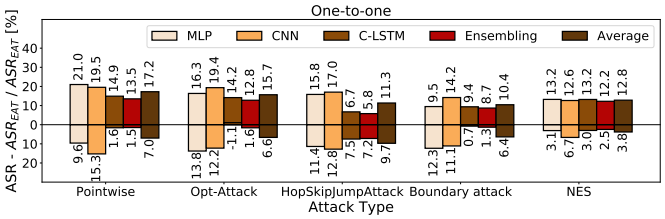


Fig. 22. ASR of each attack after EAT (bars above x axis), and ASR reduction compared to when no further defense is applied (bars below x axis) for the one-to-one NID scenario.

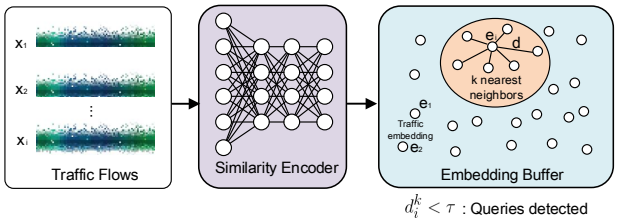


Fig. 23. An illustration of the query detection defense mechanism using a deep similarity encoder.

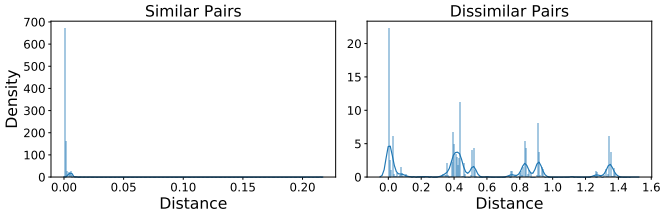


Fig. 24. Histograms of $l_2$ distances of DSE embeddings between similar flow pairs (left) and dissimilar flow pairs (right) generated using the training set.

contrastive loss function:

$$L(x_i, \tilde{x}_i, x_m, x_n; \theta) = ||e_i - \tilde{e}_i||_2^2 + \max(0, \varpi^2 - ||e_m - e_n||_2^2). \quad (2)$$

Here, $x_i, \tilde{x}_i$ are a pair of similar traffic flows, while $x_m, x_n$ are traffic flows that are dissimilar. $\theta$ is the trainable parameter set of the DSE, and $\varpi$ is a constant penalty, which regularizes the scale of $||e_m - e_n||_2^2$. We choose $\varpi = 0.5$ in our experiments. The first term of the function ensures that the $l_2$ distances of the similar traffic flows are minimized, while the second term guarantees that distances of dissimilar traffic pairs are maximized but limited to $\varpi$.

We train the DSE using the same training set sampled from the CSE-CIC-IDS2018 dataset as used by other NID models. For the purpose of training, we construct $\tilde{x}_i$ by adding Gaussian noise $\sigma_i \sim N(0, \alpha|x_i|)$ to each sample $x_i$, i.e., $\tilde{x}_i = x_i + \sigma_i$. Here, $\alpha$ controls the standard deviation of the Gaussian noise and we choose $\alpha = 0.15$. $x_m, x_n$ are sampled from a training set distinct from $x_i$. After training, we use the full training set to randomly generate 13,153,902 pairs of similar and dissimilar flows. The distributions of the

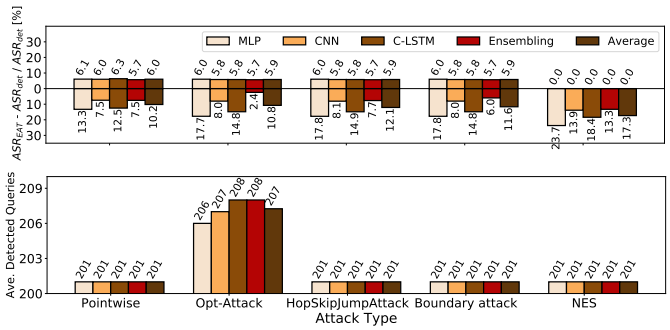


Fig. 25. Performance statistics of the query detection defense in the one-to-all scenario. Top: ASR of each attack after query detection (bars above x axis) and ASR reduction compared to when the query detection is removed (bars below x axis). Bottom: avg. number of queries when the detector is triggered.

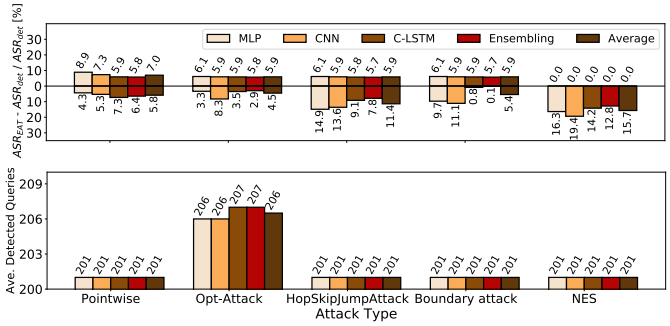


Fig. 26. Performance statistics of query detection defense in the one-to-one scenario. Top: ASR of each attack after query detection (bars above x axis) and ASR reduction compared to when the query detection is removed (below x axis). Bottom: average number of queries when the query detector is triggered.

$l_2$ distances of their embedding by histograms are shown in Fig. 24. Observe that most of of the $l_2$ distances between similar flows pairs are close to 0, while they have multiple peaks away from 0 for dissimilar flows pairs which are farther from the origin. This indicates that the DSE successfully learns the similarity between traffic flows, and therefore can operate effectively for the query detection purpose.

### 7.3.2 Hyper-parameters Selection

There are three important hyper-parameters to be configured for query detection, namely *(i)* the detection threshold $\tau$; *(ii)* the number of neighbors $k$ used for detection; and *(iii)* the size of the buffer $B$, which stores the traffic flows sent from the same IP address. These parameters will significantly affect the performance of the query detection. First, we select $\tau = 0.00157$, since 10% of dissimilar pairs and 86.4% of similar pairs in the training set are below this threshold. This provides an appropriate decision boundary to discriminate normal traffic flows and attack queries. The values of $k$ and $B$ affect the robustness of the detection and also the computational and storage cost of the NIDS. We select $B = 500$ and $k = 200$, as these numbers allow efficient detection and yield 0 false positive rates when operating with traffic streams simulated with the entire training set.

### 7.3.3 Query Detection Defense Performance

In Fig. 25, we show the ASR of each attack after the query detection (bars above the x-axis in the top sub-plot), the ASR reduction compared to when query detection is not employed (bars below the x-axis in the top sub-plot), and the average number of queries (bottom) when the attack is detected, for each attack method in the one-to-all scenario. Observe that the ASR has dropped significantly after the query detection was employed. In particular, the ASR of NES stays at 0 for all models and the detection rates therefore become 100%. Similarly, in Fig. 26 we show performance statistics of the query detection defense mechanism in the one-to-one NID scenario. Again, note the ASR of NES reaches 0 for all models, and the detection rate becomes 100%.

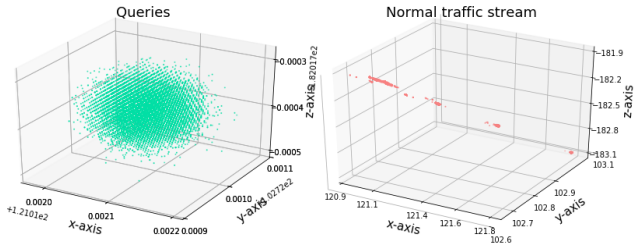


Fig. 27. Traffic sample embeddings generated by the DES. Left: sample embeddings of query process of HopSkipJumpAttack crafted from MLP NID model. Right: samples of routine traffic flows emulated with the training set.

On average, the query detection defense reduces the ASR by 8.56% and 12.38% in the one-one-to one and one-to-all scenario, respectively. Effectively, **the majority of the adversarial attack are detected during their query process.**

Taking a closer look at the average number of detected queries, we observe that NES, Boundary, Pointwise, and HopSkipJumpAttack attack attempts are detected at their 201$^{st}$ query. Recall that the $k$ neighborhood size selected for query detection is 200, hence the detection alarm will only be triggered when the buffer has more than 200 samples. This means that the attack is detected immediately after the buffer has $k$ neighbor samples. Regarding the Opt-Attack attack, this is detected always within 208 queries. This is due to the initial phase of the attack, when it injects a few benign traffic flows to learn the direction of perturbation to be added to the adversarial samples. These samples are normally dissimilar, which slightly increases the detection time. Note that, despite the efficiency of the query detection mechanism, a larger buffer size ($B = 500$) is still needed for tolerance, as the attacks may fill the buffer with queries (similar samples) and garbled traffic (dissimilar samples) alternately, to compromise the defense.

## 7.4 Effectiveness of DSE for Query Detection

To evaluate the effectiveness of our Deep Similarity Encoder (DSE), in Fig. 27 we show the DSE-embedding of the 35,215 queries of a shot of HopSkipJumpAttack crafted from the MLP model, and the DSE-embedding of 35,215 benign samples. These benign samples can be viewed as a stream of routine traffic in real life. Observe that the embeddings of the query set congregate in a fairly small region, and are close to each other. In contrast, the embeddings of the normal traffic appear more dispersed and separable. This further proves that **our DSE can effectively learn the similarity between traffic samples**.

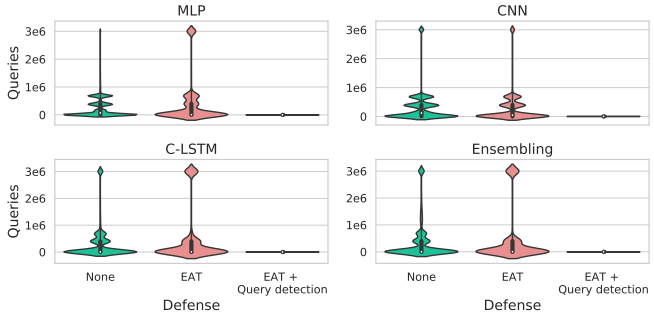


Fig. 28. Violin plots of the query distribution of all attacks in one-to-one NID scenarios.

Overall, **by combining the model voting ensembling, EAT, and query detection mechanisms, our proposal can successfully prevent five mainstream black-box adversarial attacks from compromising deep learning-based NIDS**.

### 7.5 Zooming in on Infiltration Traffic

To understand why infiltration traffic escapes detection, in Fig. 28 we examine the distribution of the number of queries of all successful attack attempts at each defense stage. We observe that while EAT does not change the query distribution significantly, most of the adversarial attacks bypassing the query detection only require 1 query. This means that the original traffic is already misclassified. The confusion matrices in Fig. 19 further confirm this, as the Infiltration traffic yields high misclassification rate.

Note that we removed three features (i.e., flow duration, total time between two packets sent in the forward and backward direction) that may be affected by perturbations during training and evaluation. The total time between two packets sent in the backward direction is however essential for identifying the infiltration traffic, according to our experiments. Once this feature is added back, the detection rate increases to over 97% and our defense mechanisms operate well in this case. We leave this as future work, which requires further study.

### 7.6 Effectiveness in Different Landscapes

To further demonstrate the effectiveness of our defense mechanism, we re-stage the TIKI-TAKA flow (i.e., NID models training, 5 adversarial attacks, and 3 defense mechanism) on a different dataset, namely CICIDS2017 [16]. The CICIDS2017 dataset is the predecessor of the CSE-CIC-IDS2018 dataset [16] used so far in our experiments, where similar benign/abnormal traffic flows were collected and similar features were extracted.

We employ the aforementioned black-box attacks to craft adversarial samples on 10,000 malicious traffic flows. After applying all the defense mechanism proposed, we obtain the ASR for each type of attack in the CICIDS2017 dataset as shown in Fig. 29. Observe that except for Infiltration, the ASR for all attacks is 0% for all models. We show the pre-EAT and post-EAT NID performance of all models considered in Table 8. Observe that all NID models obtain excellent performance, as they achieve over 98% F1 scores. After EAT, their performance is further improved, which demonstrates that the EAT can effectively improve the robustness of NID models. This complies with the performance obtained on the CSE-CIC-IDS2018 dataset, demonstrating that **our defense methods can generalize well and are therefore reliable.**

TABLE 8
The performance of MLP, CNN, C-LSTM, and the ensembling model pre-/post-EAT on the CICIDS2017 dataset in the one-to-all scenario.

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| MLP | 0.993/0.996 | 0.966/0.987 | 0.997/0.994 | 0.981/0.991 |
| CNN | 0.997/0.998 | 0.988/0.993 | 0.997/0.996 | 0.992/0.994 |
| C-LSTM | 0.996/0.998 | 0.984/0.990 | 0.998/0.999 | 0.991/0.994 |
| Ensemble | 0.992/0.997 | 0.963/0.984 | 0.999/0.999 | 0.980/0.992 |

On the other hand, we observe that the defense performance on this dataset is slightly superior to that on the CSE-CIC-IDS2018 traffic. This is reflected by the detection accuracy and the higher F1 scores of each NID models with the CICIDS2017 data. The reason is that the traffic patterns are less diverse in this latter dataset, thus it is easier for NID models to learn.

## 8 CONCLUSIONS

In this paper, we introduced TIKI-TAKA, a framework for defending against adversarial attacks on deep learning-based NIDS. We trained three state-of-the-art deep learning models (MLP, CNN, and C-LSTM) on publicly available datasets, then employed 5 classes of decision-based adversarial attacks to compromise the neural models. Experiments show that despite having high detection rates, deep learning-based NIDS are vulnerable to adversarial samples. To strengthen NIDS against such threats, we proposed three defense methods: model voting ensembling, ensembling adversarial training, and query detection. To our knowledge, these are the first defense mechanisms to be proposed against adversarial attacks targeting NIDS. Their combined use can reduce success rates of all attacks considered, bringing detection close to 100% on most malicious traffic and fending off particularly critical malicious traffic such as botnet and DoS.

## REFERENCES

[1] C. Zhang, X. Costa-Perez, and P. Patras, "Tiki-Taka: Attacking and Defending Deep Learning-Based Intrusion Detection Systems," in *Proc. ACM SIGSAC Conference on Cloud Computing Security Workshop*, ser. CCSW'20, 2020.

[2] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, 2015.

[3] Q. Zhou and D. Pezaros, "Evaluation of machine learning classifiers for zero-day intrusion detection–an analysis on CIC-AWS-2018 dataset," *arXiv preprint arXiv:1905.03685*, 2019.

[4] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Comms Surveys & Tutorials*, 2019.

[5] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. EAI Intl Conference on Bio-inspired Information and Communications Technologies*, 2016, pp. 21–26.

[6] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, 2019.

[7] M. Nasr *et al.*, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *IEEE S&P*, 2019.

[8] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *ACM CCS*, 2019.

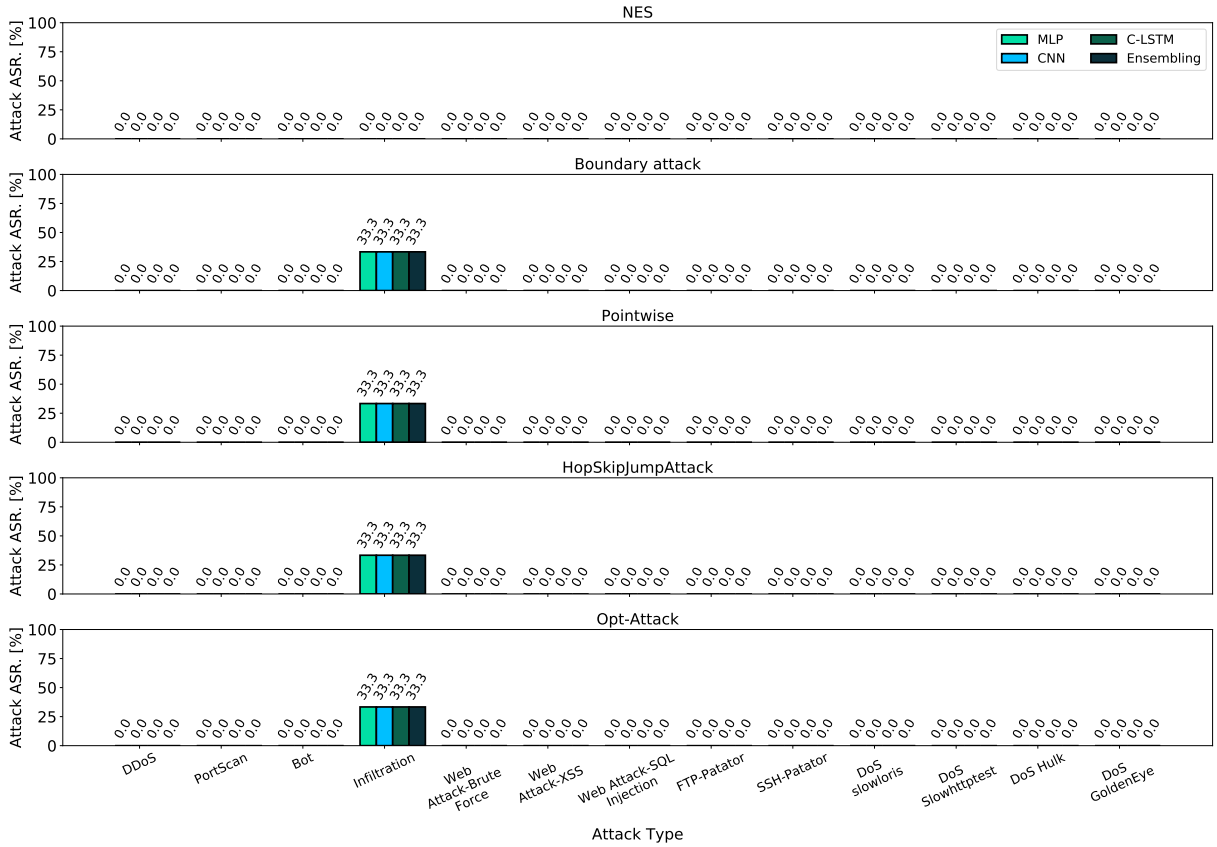


Fig. 29. ASRs over each type of attack after the EAT and query detection against all models in the one-to-all scenario on the IDS 2017 dataset.


[9] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitras, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *USENIX Security*, 2019.

[10] K. T. Co, L. Muñoz González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep convolutional networks," in *ACM CCS*, 2019.

[11] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[12] O. Ibitoye, R. Abou-Khamis, A. Matrawy, and M. O. Shafiq, "The threat of adversarial attacks on machine learning in network security–a survey," *arXiv preprint arXiv:1911.02621*, 2019.

[13] M. Usama, J. Qadir, A. Al-Fuqaha, and M. Hamdi, "The adversarial machine learning conundrum: Can the insecurity of ml become the Achilles' heel of cognitive networks?" *arXiv:1906.00679*, 2019.

[14] A. Kuppa, S. Grzonkowski, M. R. Asghar, and N.-A. Le-Khac, "Black box attacks on deep anomaly detectors," in *Proc. ACM International Conference on Availability, Reliability and Security*, 2019.

[15] Forbes, "Cyberwarfare will explode in 2020 (because it's cheap, easy and effective)," Jan 2020.

[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018.

[17] D. Heaven, "Why deep-learning AIs are so easy to fool," *Nature*, vol. 574, pp. 163–166, 2019.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.

[19] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques," in *Proc. ACM Southeast Conference*, 2019.

[20] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *IEEE International Conference on Advances in Computing, Communications and Informatics*, 2017.

[21] Y. Zhang, X. Chen, D. Guo, M. Song, Y. Teng, and X. Wang, "PCCN: Parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows," *IEEE Access*, vol. 7, pp. 119 904–119 916, 2019.

[22] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[23] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network intrusion detection: Based on deep hierarchical network and original flow data," *IEEE Access*, vol. 7, pp. 37 004–37 016, 2019.

[24] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong, "NATTACK: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks," in *ICML*, 2019.

[25] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *ICML*, 2019.

[26] S. Moon *et al.*, "Parsimonious black-box adversarial attacks via efficient combinatorial optimization," in *ICML*, 2019.

[27] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38 367–38 384, 2018.

[28] K. Yang, J. Liu, C. Zhang, and Y. Fang, "Adversarial examples against the deep learning based network intrusion detection systems," in *IEEE MILCOM*, 2018.

[29] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE S&P*, 2017.

[30] P.-Y. Chen *et al.*, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. ACM WS Artificial Intelligence and Security*, 2017.

[31] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[32] M. Teuffenbach, E. Piatkowska, and P. Smith, "Subverting network intrusion detection: Crafting adversarial examples accounting for domain-specific constraints," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 2020, pp. 301–320.

[33] A. Piplai, S. S. L. Chukkapalli, and A. Joshi, "Nattack! adversarial attacks to bypass a gan based classifier trained to detect network intrusion," in *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, 2020, pp. 49–54.

[34] R. A. Khamis and A. Matrawy, "Evaluation of adversarial training on different types of neural networks in deep learning-based idss,"

in *International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, pp. 1–6.
[35] I. Debicha, T. Debatty, J.-M. Dricot, and W. Mees, "Adversarial training for deep learning-based intrusion detection systems," *arXiv preprint arXiv:2104.09852*, 2021.
[36] N. Papernot *et al.*, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE S&P*, 2016.
[37] F. Tramèr *et al.*, "Ensemble adversarial training: Attacks and defenses," in *ICLR*, 2018.
[38] J. Lu, T. Issaranon, and D. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *IEEE ICCV*, 2017.
[39] P. Samangouei *et al.*, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *ICML*, 2018.
[40] M. Abbasi and C. Gagné, "Robustness to adversarial examples through an ensemble of specialists," in *Workshop in ICLR*, 2017.
[41] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Intl Conference on Computer Aided Verification*, 2017.
[42] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *ACM CCS*, 2017.
[43] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. ACM Workshop on Artificial Intelligence and Security*, 2017.
[44] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *ICML*, 2018.
[45] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Comm. Netw.*, 2018.
[46] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans Evolutionary Computation*, 2019.
[47] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.
[48] H. He and Y. Ma, *Imbalanced learning: foundations, algorithms, and applications*. Wiley-IEEE Press, 2013.
[49] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis *et al.*, "TensorFlow: A system for large-scale machine learning," in *OSDI*, 2016.
[50] H. Dong, A. Supratak, L. Mai, F. Liu, A. Oehmichen, S. Yu, and Y. Guo, "TensorLayer: A versatile library for efficient deep learning development," in *Proc. ACM Multimedia*, 2017.
[51] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," 2011.
[52] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *ICLR*, 2018.
[53] L. Schott *et al.*, "Towards the first adversarially robust neural network model on MNIST," in *ICLR*, 2019.
[54] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *IEEE S&P*, 2020.
[55] M. Cheng, T. Le, P.-Y. Chen, H. Zhang, J. Yi, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in *ICML*, 2019.
[56] D. Chou and M. Jiang, "Data-driven network intrusion detection: A taxonomy of challenges and methods," *arXiv preprint arXiv:2009.07352*, 2020.
[57] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.
[58] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," vol. 11, no. 3, 2020.
[59] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A Python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017.
[60] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *ICLR*, 2014.
[61] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, 2008.
[62] C. Zhang, R. Li, W. Kim, D. Yoon, and P. Patras, "Driver behavior recognition via interwoven deep convolutional neural nets with multi-stream inputs," *IEEE Access*, vol. 8, 2020.
[63] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *MobiHoc*, 2018.
[64] C. Zhang, X. Ouyang, and P. Patras, "ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network," in *ACM CoNEXT*, 2017.
[65] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary. 2015."
[66] Y. Wu, D. Bamman, and S. Russell, "Adversarial training for relation extraction," in *Proc. Empirical Methods in NLP*, 2017.
[67] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.
[68] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *ICLR WS*, 2017.
[69] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *IEEE CVPR*, 2018.
[70] S. Chen, N. Carlini, and D. Wagner, "Stateful detection of black-box adversarial attacks," *arXiv preprint arXiv:1907.05587*, 2019.
[71] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 98, 2015.

**Chaoyun Zhang** is currently a senior researcher at Tencent Lightspeed & Quantum Studios. He obtained PhD and MSc degress from the University of Edinburgh, with a focus on machine learning and mobile networking. He also obtained a BSc degree from the School of Electronic Information and Communications at Huazhong University of Science and Technology, China. His current research interests include the application of game AI and data mining.

**Xavier Costa-Pérez** is ICREA Research Professor, Scientific Director at the i2Cat Research Center and Head of 5G Networks R&D at NEC Laboratories Europe. His team generates research results which are regularly published at top scientific venues, produces innovations which have received several awards for successful technology transfers, participates in major European Commission R&D collaborative projects and contributes to standardization bodies such as 3GPP, ETSI NFV, ETSI MEC and IETF. He has served on the Organizing Committees of several conferences, published papers of high impact and holds tenths of granted patents. Xavier received his Ph.D. degree in Telecommunications from the Polytechnic University of Catalonia (UPC) in Barcelona and was the recipient of a national award for his Ph.D. thesis.

**Paul Patras** is an Associate Professor in the School of Informatics at the University of Edinburgh, where he leads the Mobile Intelligence Lab – a multi-disciplinary team that pursues research at the intersection of network engineering and artificial intelligence, to improve the analysis, resilience, and management of next generation mobile systems. He is also a co-founder and CEO of Net AI, a pioneering university spinout specializing in AI-driven network analytics. He has served on the organizing committee on several conferences and workshops in his field, and advised the ITU-T Focus Group on Machine Learning for Future Networks including 5G. Paul holds M.Sc. and Ph.D. degrees from Universidad Carlos III de Madrid (UC3M) and he was the recipient of a prestigious Chancellor's Fellowship awarded by the University of Edinburgh.