Demystifying the Transferability of Adversarial Attacks in Computer Networks

Ehsan Nowroozi, Member, IEEE, Yassine Mekdad, Member, IEEE, Mohammad Hajian Berenjestanaki, Mauro Conti, Fellow, IEEE, and Abdeslam El Fergougui, Fellow, IEEE

Abstract—Convolutional Neural Networks (CNNs) models are one of the most frequently used deep learning networks, and extensively used in both academia and industry. Recent studies demonstrated that adversarial attacks against such models can maintain their effectiveness even when used on models other than the one targeted by the attacker. This major property is known as transferability, and makes CNNs ill-suited for security applications. In this paper, we provide the first comprehensive study which assesses the robustness of CNN-based models for computer networks against adversarial transferability. Furthermore, we investigate whether the transferability property holds in computer networks applications. In our experiments, we first consider five different attacks: the Iterative Fast Gradient Method (I-FGSM), the Jacobian-based Saliency Map (JSMA), the Limited-memory Broyden Fletcher Goldfarb Shanno BFGS (L-BFGS), the Projected Gradient Descent (PGD), and the DeepFool attack. Then, we perform these attacks against three well-known datasets: the Network-based Detection of IoT (N-BaIoT) dataset, the Domain Generating Algorithms (DGA) dataset, and the RIPE Atlas dataset. Our experimental results show clearly that the transferability happens in specific use cases for the I-FGSM, the JSMA, and the L-BFGS attack. In such scenarios, the attack success rate on the target network range from 63.00% to 100%. Finally, we suggest two shielding strategies to hinder the attack transferability, by considering the Most Powerful Attacks (MFAs), and the mismatch LSTM architecture.

Index Terms—Adversarial examples, attack transferability, machine and deep learning, adversarial machine learning, computer networks, convolutional neural networks, cybersecurity.

I. INTRODUCTION

I N THE 21st century, the evolution of artificial intelligence has demonstrated the tremendous benefit of deep learning algorithms to solve the most challenging problems of our society [1]. For example, applying deep learning techniques in the medical field shows promising results to perfectly detect and classify cancerous tissues through histopathology images [2]. Nowadays, existing Neural Networks (NNs) solutions have achieved human-level performance. In particular, Convolutional Neural Networks (CNNs) models which are among the most popular neural network architectures. Convolutional Neural Networks (CNNs) are increasingly employed in various computer network applications, such as network threat detection, network intrusion detection [3], malware detection in 5G-IoT Healthcare Applications [4], and multimedia forensics to detect several distorted pictures [5]. Prior studies investigated the robustness of Graph Neural Networks against adversarial attacks [6], and demonstrated the possibility of successfully identifying important patterns of adversarial attacks against such networks. In this line of research, the fundamental issue is that even when only partial information of the graph is provided, or the attack is limited to a few perturbations, the classification performance constantly degrades. Furthermore, the attacks apply to a variety of node categorization schemes. As a result, the CNN model outperforms Graph Neural Networks regarding classification performance.

On the other hand, recent research studies have shown that some examples can be precisely computed to mislead machine-learning models [7]. To better illustrate these examples, Grosse et al. [8] proved the possibility of crafting Android malware examples to mislead neural network malware detection models. The authors adapted the algorithm proposed by Papernot et al. [9] to craft the adversarial samples.

To elaborate more, the generation of these examples are so-called adversarial examples and pose real-life security threats [10]. A real-world scenario consists of making a self-driving car misclassify the red light as green. This scenario could be a life-threatening example for autonomous vehicles. The popularity and diversity of the CNN applications make them a perfect target for adversarial examples. The concept of adversarial examples was introduced in 2014 by Szegedy et al. [11]. Later, many studies demonstrated the
use of different adversarial attacks to compromise CNN-based models (e.g., offline handwritten signature verification, synthetic aperture radar image classification, object detection, automatic speech recognition). These attacks pose new security threats for systems built on machine learning algorithms, hindering the model’s functionality in various applications.

It is worth mentioning that building adversarial examples can typically be applied to every machine learning algorithm due to the native characteristics of machine learning architectures. This vulnerability also affects modern deep neural networks. An interesting case in a black-box setting is that two models with different network architectures and trained with different training sets can misclassify the same adversarial example. In this scenario, the adversary trains the source network, and designs the adversarial example, then transfer it to the victim’s network even with a few knowledge about the victim’s model [12]. This substantial property is known as transferability and might disrupt CNN’s operation in various applications. To better visualize this concept, Wu et al. [13] evaluated the adversarial transferability in image classification against three datasets (MNIST, CIFAR-10, and ImageNet) and provided three key factors influencing it: the architecture, the test accuracy, and the model capacity. For the architecture, the similarity between the source and target model significantly enables the transferability property. Regarding the test accuracy, the models with high accuracy manifest high transferability. However, the models with deeper capacity (i.e., higher amount of model parameters) are less transferable when they act as source models. The purpose of our study is to see if and to what degree the transferability of adversarial examples applies in computer networks, following some recent studies demonstrating that adversarial examples jeopardize CNN forensics techniques [14]. From a research point of view, it is highly important to answer this question because the attack transferability would make developing defense strategies much more complex. Furthermore, giving the attacker a complete access to the source network will not prevent fooling the target network. Experimental evidence demonstrated by Tramèr et al. [15] has shown that the transferability of adversarial examples from a source network to a target network is more likely to occur in very large adversarial subspaces (i.e., the space of adversarial examples), where the two models will intersect. Consequently, the subspace dimensionality of two adversarial subspaces correlates with the transferred property.

In this paper, we solely consider two instances of CNN’s deep learning networks, which have lately been employed in computer vision, multimedia forensics, adversarial machine learning, and other fields. We mainly investigate if the attack transferability might occur in security-oriented applications for computer networks. In other words, we verify if an adversarial attack on a Source Network (SN) might be effective against a possibly different and unknown Target Network (TN). Although authors in [12], [14] recently demonstrated that the transferability of adversarial samples holds in computer vision applications and computer forensics. We launched five different attacks against three well-known datasets under black-box settings. Our experimental results demonstrate that the generated adversarial examples are not transferable in most cases. However, just a few cases of attacks are transferable from the SN to the TN. In particular, in the cross model transferability scenario where the same datasets such as N-BaIoT, DGA, and RIPE Atlas datasets are trained on two different network architectures. These adversarial examples achieve good transferability and can perfectly deceive the TN.

A. Contributions

Our contributions are summarized as follows:

- We offer the FAIL models SPRITZ1 and SPRITZ2 that fail against different adversarial attacks, and which provides a basic framework for simulating and analyzing realistic adversaries. Across multiple degrees of adversarial control, we assess the robustness of the networks in terms of attack transferability against machine learning systems.
- We perform five different adversarial attacks with different parameters from low to high to demystify the transferability in computer networks, namely: the I-FGSM attack, the JSMA attack, the L-BFGS attack, the PGD attack, and the DeepFool attack.
- We independently examine the impact of training data and network architecture mismatch on the attacks’ transferability. Furthermore, we systematically investigate the impact of partial adversarial knowledge and control on the robustness of machine learning models against the test-time attack (exploratory evasion-attack).
- We offer two shielding strategies that overcome prior adversarial attacks’ limits. First, by using mismatch classifiers and considering the LSTM architecture as a target network to hinder the adversarial transferability. Second, by fine-tuning with Most Powerful Attacks (MPAs) [16] to improve the security of the target network.

B. Organization

We outline the rest of our paper as follows. In Section II, we provide the background and preliminaries on the datasets and adversarial attacks used in our experiments. Then, we overview the related works that discussed the transferability property in Section III. Afterward, in Section IV, we provide our methodology and the experimental setup. Section V presents the experimental results followed by a discussion evaluating the transferability in computer networks, and presents two recent approaches for hindering adversarial transferability. Finally, we conclude our paper and provide promising future research work in Section VI.

II. BACKGROUND

In this section, we provide the background on the datasets and adversarial attacks used in our experiments. First, we present the N-BaIoT [17], the DGA [18], and RIPE Atlas [19] datasets. Afterward, we discuss the adversarial attacks performed to evaluate the transferability property in CNN-based models on computer networks. Table I lists all the abbreviations and notations used in our paper.
A. Datasets

To evaluate the effectiveness of the transferability property in computer networks, we considered comprehensive real-world datasets that contain malicious and benign data in computer networks. It is worth mentioning the importance of relying on the high quality of datasets to achieve high training accuracy [20]. In our experiments, we train the SN and the TN on three different datasets, namely: the N-BaIoT, the DGA, and the RIPE Atlas datasets. Additionally, these datasets are well-known and have lately been used for various cybersecurity purposes. The N-BaIoT dataset has been used to recognize botnet attacks in IoT devices employing deep auto-encoders [21], detection methods in the Internet of Things [22], and federated learning for detecting attacks in the Internet of Things [23]. Besides, the DGA dataset is one of the most widely used datasets in detecting and classifying DGA botnets, and their families [24]. It presents a framework to identify Domain Generating Algorithms (DGAs) in real-world network traffic [18]. The RIPE Atlas dataset has been employed in JurisNN [25], network devices monitor and alerts [26], and so forth. In what follows, we provide a detailed description of each one of the considered datasets.

1) N-BaIoT Datasets: The N-BaIoT datasets are built upon the collection of real benign and malicious traffic using port mirroring techniques and gathered from nine different commercial IoT devices. The N-BaIoT datasets contain 115 features and around seven million instances. With the increase of the IoT cyber attacks, which turns out the Internet of Things into the Internet of vulnerabilities, the adversaries rely on using botnets to exploit such vulnerabilities [27]. The generation of malicious IoT traffic for the N-BaIoT datasets is performed by compromising nine commercial devices by two well-known IoT-based botnets: The BASHLITE botnet and the Mirai botnet. These botnets leverage the use of DDoS (Distributed Denial of Service) attack resulting in the unavailability of the IoT devices. The BASHLITE botnet is a well-known IoT botnet that infects Linux-based IoT devices. With its capability of launching DDoS attacks, BASHLITE infected up to 1 million IoT commercial devices, mostly DVRs and Web-connected video cameras [28]. The Mirai botnet is one of the most popular IoT botnets that infected thousands of devices used in the past decades. The release of its source code shows that Mirai botnet consists of (1) automatic scanning for vulnerable IoT devices, (2) uploading the malware into the vulnerable IoT device to perform the DDoS attack. The cyber attacks performed by the two authentic botnets in the IoT network cover flooding attacks (e.g., TCP-SYN flooding attack, UDP flooding attack), sending spam data, and scanning attacks.

The collection of malicious traffic generated by such attacks supports the adoption of N-BaIoT datasets in several applications. In [21], the authors provided a deep learning network-based anomaly detection method for IoT botnets. Their experimental results show that the use of deep autoencoders techniques can accurately detect IoT botnets. Another work proposed a privacy-preserving framework for malware detection in IoT devices [23], the authors applied federated learning mechanisms for training and evaluating supervised and unsupervised learning models while protecting the privacy of sensitive shared data. We believe that we can also consider deep autoencoders approach utilized for IoT botnet detection to investigate the transferability property.

2) DGA Datasets: The DGA datasets comprise legitimate and malicious DNs (Domain Names). The clean datasets are the Alexa (1M) records of top websites [18], and the malicious DNs are gathered from more than 25 different families of malware (e.g., Kraken, nymaim, shiotob, and alureon) that exploit the DGA algorithm. These algorithms are embedded into the malware and principally designed to generate many malicious DNs automatically. In addition, DGA algorithms are resistant against blacklisting and sinkholing techniques, which detect and block malicious DNs. Integrating DGA algorithms into the malware enables C&C (Command and Control) servers for botnets to perform potential DDoS attacks. Given the high quality and practical applications of DGA datasets, prior works in the literature have considered the DGA datasets. In [29], the authors developed an approach to detect DGA-based malware. This approach enables the identification of a single DGA domain and relies on using a set of lexical evidence and RF (Random forest) algorithms to profile DGA-based malware. A similar work investigated the use of DL algorithms to detect DGAs [30], the authors leveraged the RNN based architectures to detect potentially malicious DGAs, and their evaluation on real-world dataset demonstrated the high performance of RNN based DGA classifiers.

3) RIPE Atlas Datasets: The Internet of Things (IoT) concept has acquired much interest in past years. One approach to characterize IoT networks is through a large collection of Internet-aware detectors and controllers, including the electrical switches that turn the central air conditioning device on or off. The RIPE Atlas study assesses information about the Network through Internet-aware sensors [19]. RIPE Atlas could also be used to continually check the accessibility of networks or hosts from hundreds of locations across the world.
Additionally, it can perform ad hoc connection tests to analyze and resolve identified network issues and test the availability of DNS servers. In these datasets, we considered feature-engineered RAW data by decoding the buffers returned in DNS queries, and returning the DNS measurement results [31]. Initially, the RIPE dataset aim to identify the application’s type based on the flow data. These datasets are tested in two different network architectures: the Recurrent Neural Networks (RNN), and the Convolutional Neural Networks (CNN). We mention that some transformation steps have been performed to prepare the datasets [32], and the network is trained to classify the flow sequence for several applications. Although the accuracy of both training and testing data reached 80%, the model can perfectly predict the majority of flow sequences, and the error rate is significantly reduced by feeding additional data into the system.

B. Adversarial Attacks

Much attention has been received recently concerning the possibility of adversarial examples to fool the CNN-based models. In the existing literature, the adversary can perform the adversarial attack under three different settings: (i) white-box, (ii) gray-box, and (iii) black-box attacks. (i) In the white-box setting, the adversary has a Perfect Knowledge (PK) on the TN by adopting a particular categorization for adversarial examples [10]. (ii) In the gray-box setting, the adversary has a Limited Knowledge (LK) on the TN [33]. (iii) In the black-box setting, the attacker does not have access to the internal details of the TN, which is a more feasible and complicated scenario than the previous ones; as a result, the adversary uses multiple queries to obtain such internal details. Therefore, we are motivated to consider the black-box scenario given its feasibility in real-world applications. In our study, we propose and test a generalized technique for black-box threats on ML that takes advantages of adversarial transferability.

In our work, we performed five adversarial attacks under black-box settings, where we do not have access to the underlying deployed model: the I-FGSM attack [34], the JSMA attack [9], the L-BFGS attack [11], the PGD attack [35], and the DeepFool attack [36]. In our experiments, we considered the Foolbox library [37] to implement all adversarial attacks.

1) The I-FGSM Attack: The I-FGSM attack represents the iterative version of the FGSM (Fast Gradient Sign Method) [38]. The FGSM attack is one of the most well-known adversarial examples. It consists of targeting neural networks, which fail the network and potential misclassification of an input. The Fast Gradient Sign Method is a gradient-based attack that adjusts the input data to maximize the loss. Given an input \(X\) of a model with \(\theta\) parameters and a ground truth label \(Y\). We compute the adversarial example \(Adv_X\) of the FGSM attack as follows:

\[
Adv_X = X + \epsilon \times \text{sign} \left( \nabla_X J(\theta, X, Y) \right)
\]

where \(\epsilon\) is the normalized attack strength factor, and \(J\) is the cross-entropy cost function used to train the model. We mention that \(\epsilon\) is the parameter controlling the perturbations and should be considered minimal for the success of the adversarial example. Given that the FGSM attack often fails in some scenarios, the iterative version of the FGSM attack, namely the I-FGSM attack [34], guarantees finer perturbations and could be described as a straightforward extension of the FGSM. More specifically, the I-FGSM attack considers iterating the FGSM method with smaller steps on the way to the gradient sign. For the I-FGSM attack, we express the adversarial example for each \(i + 1\) iteration such that \(Adv_X^i = X\) as follows:

\[
Adv_X^{i+1} = Adv_X^i + \epsilon \times \text{sign} \left( \nabla_X J(\theta, Adv_X^i, Y) \right).
\]

2) The JSMA Attack: Papernot et al. [9] introduced the JSMA attack to reduce the number of perturbations comparing to the I-FGSM attack. Therefore, the construction of JSMA-based adversarial examples is more suitable than I-FGSM for targeted misclassification and harder to detect. However, the JSMA algorithm requires a high computational cost. The JSMA attack is a gradient-based method that uses a forward derivative approach and adversarial saliency maps. Instead of a backward propagation like the I-FGSM attack, the forward derivative is an iterative procedure that relies on forward propagation. Given an \(N\)-dimensional input \(X\) of a model, the adversary starts by computing a Jacobian matrix of the \(M\)-dimensional classifier \(F\) learned by the model during the training phase. The Jacobian matrix is given by:

\[
J_F(X) = \frac{\partial F(X)}{\partial X} = \left[ \frac{\partial F_i(X)}{\partial x_j} \right]_{i=1..M, j=1..N}.
\]

Afterward, the adversary produces adversarial samples by constructing the adversarial saliency maps \(S\) based on the forward derivative. These maps identify specific input features where the adversary enables the perturbations needed for the desired output. Thus, building the adversarial sample space. For a target class \(t\), we define the adversarial saliency map by the following equation:

\[
S(X, t)[i] = \begin{cases} 0, & \text{if } \frac{\partial F_i}{\partial x_j}(X) < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_i}{\partial x_j}(X) > 0, \\
\frac{\partial F_i}{\partial x_j}(X) & \left| \sum_{j \neq t} \frac{\partial F_i}{\partial x_j}(X) \right|, \text{otherwise.} \end{cases}
\]

In summary, JSMA is a greedy repeated process that generates a saliency map at each iteration through forwarding propagation, indicating which pixels contributed the most to classification. A large value in this map, for instance, indicates that changing that pixel significantly increases the likelihood of selecting the incorrect class. The pixels are then changed individually, depending on this map, by a relevant number \(\theta\) (\(\theta\) refers to the range of possible pixel changes).

3) The L-BFGS Attack: The L-BFGS attack has been proposed by Szegedy et al. [11]. It is a non-linear gradient-based numerical optimization method and aims to reduce the perturbations for a given input. The L-BFGS attack generates the adversarial examples by optimizing the input \(X\) and maximizing the prediction error. The L-BFGS attack is a practical box-constrained optimization method to generate adversarial samples. However, it is time-consuming and computationally demanding. For an input \(X\), The formal expression of the optimization problem of the L-BFGS attack is given by:

\[
\min_{Adv_X} ||X - Adv_X||^2 \text{ subject to: } F(X + r) = l.
\]
Such that $F$ is the classifier, $l$ is the target label, and $r$ is the minimizer. However, the optimization problem depicted in Eq. (5) is computationally expensive and considered as a hard problem. An approximate solution solves this problem by using a box-constrained L-BFGS [11], and is expressed as follows:

$$\min_{Adv} \|X - AdvX\|_2^2 - J(\theta, X, Y). \quad (6)$$

In this approximation, the objective is to find the minimum scalar $c > 0$ and for which the minimizer $r$ of the Eq. (5) is satisfied.

4) The PGD Attack: The PGD attack [35] is a first-order adversarial attack, i.e., consists of using the local first-order information about the targeted network. It is an iterative version of the FGSM, such as the I-FGSM. Except that in the PGD attack, for each iteration $i + 1$, the input $X$ is updated according to the following rule:

$$Adv_{X}^{i+1} = \Pi_{X+S}(Adv_{X}^{i} + \epsilon* \text{sign} (\nabla_X J(\theta, X, Y))) \quad (7)$$

where $\Pi$ is the projection operator that keeps $Adv_{X}^{i+1}$ within a set of perturbation range $S$. We note that the PGD attack looks for the perturbations that maximize $J(\theta, X, Y)$ while considering the $L^\infty$ distortion.

5) The DeepFool Attack: Regarding the DeepFool attack [36], it fools the binary and multiclass classifiers by finding the closest distance from the input $X$ to the other side of the classification boundary. For any given classifier, the minimal perturbation $r$ to generate an adversarial example by the DeepFool attack is formally expressed by:

$$\delta(X, F) = \min_r \|r\|_2 \text{ subject to: } F(X + r) \neq F(X). \quad (8)$$

Such that $\delta$ describes the robustness of $F$ at the input $X$. We mention that the state-of-the-art classifiers are weak against the DeepFool attack. Therefore, the DeepFool present a reference to test and evaluate the robustness and the performance of existing classifiers. The N-BaIoT datasets are built open the collection of real and benign malicious traffic in pcap using nine gathered.

### III. Related Work

The early studies of adversarial examples revealed the transferability of adversarial attacks [11], [38]. Recently, initial efforts have been devoted to explore this property in different areas (e.g., multimedia forensics, natural language processing).

In multimedia forensics applications, the use of machine learning techniques offers promising solutions for security analysts. However, adversarial image forensics significantly hinder such possibility [7]. Barni et al. [39] investigated the transferability of adversarial examples in CNN-based models. The authors launched the I-FGSM and the JSMA attacks against two different datasets and argued that adversarial examples are non-transferable under black-box settings. In the same context, another work explored the transferability and vulnerability of CNN-based methods for camera model investigations [14]. Later, the authors in [40] proposed an input transformation-based attack framework to enhance the transferability of adversarial attacks.

In Natural Language Processing (NLP) applications, Wallace et al. [41] investigated the transferability in black-box Machine Translation (MT) systems through gradient-based attacks. In their proposed approach, the authors built an imitation model similar to the victim model. Next, they produced transferable adversarial examples to the production systems.

In Computer Vision (CV) applications, it is worth mentioning that deep learning-based adversarial attacks have become a significant concern for the artificial intelligence community [42]. The transferability of adversarial examples has also been investigated in computer vision tasks [43]. In particular, the authors in [43] proposed adversarial examples to enhance the cross-task transferability in classification, object detection models, explicit content detection, and Optical Character Recognition (OCR).

In different application scenarios, Suciu et al. [44] implemented a framework to evaluate realistic adversarial examples on image classification, android malware detection, Twitter-based exploit prediction, and data breach prediction. Then, proposed evasion and poisoning attacks against machine learning systems. They prove that a recent evasion attack is less effective in the absence of broad transferability. The authors explored the transferability of such attacks and provided generalized factors that influence this property. In a similar work, Demontis et al. [45] evaluated the transferability of poisoning and evasion attacks against three datasets that are associated with different applications, including android malware detection, face recognition, and handwritten digit recognition. The authors demonstrated that the transferability property strongly depends on three metrics: the complexity of the target model, the variance of the loss function, and the gradient alignment between the surrogate and target models. However, the considered metrics are not well proven in security-oriented applications and might add restrictions on the threat model.

In Table II, we present a comparison that summarizes our work and current works on the transferability in machine and deep learning applications. To the best of our knowledge, most covered research in this area did not consider the transferability property in computer networks. To that end, we aim to evaluate the robustness of transferable examples in CNN-based models for computer networks by performing a comprehensive set of experiments under black-box settings.

### IV. Methodology

To evaluate the transferability property of adversarial attacks against CNN-based models in IoT networks, which are trained with the N-BaIoT [21], DGA [18], and RIPE Atlas [19] datasets. We considered two network architectures namely SPRITZ1 and SPRITZ2, and five different adversarial attacks (as described in Section II). Given the datasets and the network architectures, we built six classes of networks as described in Table III. In what follows, we present the architectural details of the network and the experiments carried out during our investigation.

In ML and DL models, the adversary can influence the amount of training and testing data depending on his
TABLE II
DIFFERENCE BETWEEN OUR WORK AND EXISTING WORKS ON THE TRANSFERABILITY IN MACHINE AND DEEP LEARNING APPLICATIONS

| Ref  | Application Domain      | Adversarial Attacks       | Considered Datasets       | Model Settings  | Transferability Results                          |
|------|-------------------------|---------------------------|---------------------------|----------------|-----------------------------------------------|
| [39] | Multimedia forensics    | -ISMA, and IDPSM          | -RAISS [46], and VISION [47] | -Black box    | -No transferability between the SN and the TN |
| [14] | Multimedia forensics    | -FPGM, and PID            | -VISION [47]              | -White box    | -No transferability between the SN and the TN |
| [40] | Image processing        | -Input transformation    | -ImageNet [48]            | -Black box    | -The transferability happens in white and black box settings |
| [41] | Natural Language Processing (NLP) | -Gradient-based uprising     | -EPSLY [49], and Europal [50] | -Black box    | -The adversarial transferability happens on an intermediate convolution layer |
| [42] | Computer Vision (CV)    | -Dispersion reduction     | -ImageNet [48], NSF2 Data | -Black box    | -The adversarial transferability happens on an intermediate convolution layer |
| [43] | Different applications  | -The PAE, attack model    | -Droid Android malware detector [51], Scorer [52], and CCOcDx-Test [53] | -Black box    | -Bypass the SN by considering the transferability property |
| [44] | Different applications  | -Gradient-based evasion    | -MNIST [54], and Droidin [55] | -Black box    | -Different metrics that impact attacks’ transferability |
| [45] | Different applications  | -Gradient-based evasion    | -MNIST [54], and Droidin [55] | -Black box    | -Different metrics that impact attacks’ transferability |
| Our work | Computer Networks     | -ISMA, IDPSM, LBPGS, PID, and DeepPool | -N-BaIoT [19], DGA [18], and RIPE Atlas [19] | -Black box    | -The adversarial transferability happens for the ISMA, IDPSM, and LBPGS attacks |

TABLE III
GENERAL DESCRIPTION OF THE CONSIDERED NETWORKS WITH THEIR RESPECTIVE DATASETS

| Networks | Datasets | N-BaIoT | DGA | RIPE |
|----------|----------|---------|-----|------|
| SPRITZ1  | N-BaIoT  | SPRTIZ1 |     | SPRTIZ1 |
| SPRITZ2  | N-BaIoT  | SPRTIZ2 |     | SPRTIZ2 |

capability. As a result, the attacker’s capability is relevant to the impact of the attack. In the exploratory-evasion attack scenario, the adversary’s capability is limited to alterations to the testing data, while changes to the training data are not permitted. In a causative or poisoning attack scenario, the attack might disrupt the training phase; these attacks are sometimes referred to as poisoning attacks. The majority of the computer network attacks developed so far come within the group of exploratory attacks. In this study, we examined the exploratory evasion scenario [57] in which the adversary’s capabilities are limited to the alterations of the testing data, while the changes on the training data are not permitted. In the case of causative attacks, the attack might disrupt the training process, such as poisoning attacks that are commonly used to describe these types of attacks. The majority of the planned computer network attacks in ML and DL belong to the first scenario. These test time attacks operate by modifying the target sample and pushing it beyond the decision boundary of the model without changing the training process or the decision boundary itself, as shown in Figure 1.

A. Network Architecture

In our experiments, we considered the network architecture provided by [5]. It is a CNN-based network that includes an input layer, convolutional layers, max-pooling layers, and one fully connected layer. The network is characterized by several convolutional layers before the first max pooling layer to enable at a large scale a good reception for each neuron. Additionally, we maintain the maximum spatial information by setting the stride to 1. Accordingly, we deployed the SPRITZ1 and SPRITZ2 networks with different types of datasets (ref. Table III). It is worth mentioning that we are considering the deep network given its difficulty to compromise compared to the shallow network. Moreover, we rely on such network architecture because it achieves a high accuracy during the training phase. Both networks are configured once as the SN and once as the TN. The architectural details of SPRITZ1 and SPRITZ2 networks are given below.

1) SPRITZ1 Network: The configuration of SPRITZ1 is a shallow network that consists of nine convolutional layers, two max-pooling layers, and one fully connected layer. In Figure 2, we illustrate the pipeline of the SPRITZ1 architecture. For all the convolutional layers, we use the kernel size of $3 \times 3$ and stride 1. In addition, we applied the max-pooling with the kernel size of $2 \times 2$ and stride 2. We note that by halving the number of features maps in the final convolutional layer, we reduce the number of parameters and consider only one fully-connected layer.

2) SPRITZ2 Network: In the SPRITZ2 network, we increased the number of convolutional layers to fifteen layers...
comparing to SPRITZ1. As depicted in Figure 3, the SPRITZ2 network is deeper than SPRITZ1 network. However, we respected the same settings as the SPRITZ1 network. In other words, we used one fully connected layer and two max-pooling layers with the kernel size of $2 \times 2$ and stride 1. For all the convolutional layers, we considered the kernel size as $3 \times 3$ and stride 1.

**B. Experimental Setup**

To build our models $N^{N-BaIoT}_{SPRITZ1}$ and $N^{N-BaIoT}_{SPRITZ2}$, for each class, we considered 58000 patches for training, 20000 for validation, and 20000 for testing, per class. The input patch size is set to $64 \times 64$ in all the cases. Regarding $N^{DGA}_{SPRITZ1}$ and $N^{DGA}_{SPRITZ2}$, we considered 2000 patches for training and 500 patches for validation and testing, per class. Concerning $N^{RIPE}_{SPRITZ1}$ and $N^{RIPE}_{SPRITZ2}$, we employed 60000 patches for training, 20000 for validation, and 20000 for testing, per class, that we think these number of features enough for a CNN for generalizing. By using small patches, we increase the depth of the network with the same parameters. Consequently, the detection accuracy is increased when aggregating the patch scores. We implemented our CNN, and LSTM networks for training and testing in TensorFlow via the Keras API [59] with Python. We ran our experiments using Intel Core i7-10750H, NVIDIA GeForce RTX 2060 with 6GB GDDR6 GPU, DDR4 16GB*2 (3200MHz), Linux Ubuntu 20.04. We made the simulation Python code publicly available at Github [60]. The number of training epochs is set to 20. We used Adam solver with a learning rate of $10^{-4}$ and a momentum of 0.99. The batch size for training and validation on N-BaIoT, DGA, and RIPE is set to 64, and the test batch size is set to 100.

**C. Empirical Study**

In the experiments, we launched five different adversarial attacks against six classes of networks (ref. Table III) to highlight the transferability in a large domain. Concretely, for each two different classes of networks, namely the SN (Source Network) and the TN (Target Network), we analyzed the transferability property of adversarial attacks from the SN to the TN. According to Barni et al. [39] and Papernot et al. [12], we split our experiments into three scenarios of transferability. These scenarios are related to the type of mismatch between the SN and the TN (i.e., half mismatch and complete mismatch). We provide in Figure 4 the general scheme of the transferability property in computer networks.

**V. RESULTS AND DISCUSSION**

In this section, we present the results of our experiments; then, we suggest two defense mechanisms against adversarial transferability. Afterward, we provide a discussion to evaluate the performance of the transferability property in computer networks.

**A. Experimental Results**

The test accuracy achieved by SPRITZ1 and SPRITZ2 on the test set N-BaIoT, DGA, and RIPE, detection task is: 99.89% for $N^{N-BaIoT}_{SPRITZ1}$, $N^{N-BaIoT}_{SPRITZ2}$, $N^{DGA}_{SPRITZ1}$, $N^{DGA}_{SPRITZ2}$, and $N^{RIPE}_{SPRITZ2}$. We
achieved the test accuracy 99.50% for $N_{\text{SPRITZ1}}^{DGA}$, and 99.20% for $N_{\text{SPRITZ1}}^{RIPE}$.

For the I-FGSM attack, we fixed the number of steps to 10, and we considered the normalized strength factor $\varepsilon = 0.1, 0.01, \text{and} 0.001$, where the average PSNR is above 40 dB. Regarding the JSMA attack, the model parameters $\theta$ are set to 0.1 and 0.01.

The results presented in the following correspond to the average ASR on SN, the average ASR on TN, the average PSNR on 250 samples, the average $L_1$ distortion, and the average maximum absolute distortion. During the experiments, we consider the adversarial attack transferability successful if the average ASR threshold is above 50%. We highlight the transferable attacks in bold for each scenario.

Cross Training Transferability: We illustrate and report in Table V five case scenarios in the cross-training transferability. For the SPRITZ1 network, and given both datasets interchangeably across the SN and the TN. Although the average ASR on the SN is successful (except for the JSMA attack that fails on the SN $N_{\text{SPRITZ1}}^{N-BaloT}$ with the parameter $\theta = 0.01$), the transferability property is not satisfied on the TN. More specifically, for all the considered five adversarial attacks, the average ASR on the TN is almost null. Similarly, for the SPRITZ2 network, the adversarial attacks cannot be transferred between the SN and the TN. In particular, we remark that the average ASR on the TN remains almost null. Moreover, the JSMA attack fails on the SN $N_{\text{SPRITZ2}}^{N-BaloT}$ and $N_{\text{SPRITZ2}}^{DGA}$ when considering the parameter $\theta = 0.01$. In summary, we observe that the adversarial attacks generated are not transferable from the SN to the TN, and in all the cases, the adversary cannot fool the TN. With respect to the training dataset, it is worth mentioning that, for a given computer network, where the SN and TN share the same architecture and are trained on two different datasets, the cross-training transferability is asymmetric. In other words, when the datasets are mismatched but the architectures are matched, the I-FGSM, JSMA, LBFGS, PGD, and DeepFool attacks are less transferable in the cross-training transferability. One plausible explanation is that while the attacks only affect a small percentage of the samples, the attacked model tends to overfit more. It’s also worth noting that the transferability of training datasets for a specific task isn’t symmetric. This implies that, mostly in computer networks, the baseline dataset may influence the features learned by the network in some way and to some amount. We considered another dataset, RIPE Atlas, as a TN (SPRITZ1) for a few experiments to better evaluate the transferability performance. When using RIPE Atlas dataset, we have found that attacking on SN is still not enough to deceive TN.

Cross Model Transferability: We consider five case scenarios in the cross model transferability, and we describe the experimental results in Table VI. When training the same datasets on two different networks (SPRITZ1 and SPRITZ2), we can see that the average ASR on the SN is successful (except for the JSMA attack that fails on the SN $N_{\text{SPRITZ1}}^{N-BaloT}$ with the parameter $\theta = 0.01$). Moreover, we remark that some adversarial attacks exhibit a transferability between the SN and the TN, such as the JSMA attack, the I-FGSM attack, and the LBFGS attack. For the N-BaloT datasets, the JSMA attack enables a strong transferability from the SN to the TN when considering the SNs $N_{\text{SPRITZ1}}^{N-BaloT}$ and $N_{\text{SPRITZ2}}^{N-BaloT}$ with the parameter $\theta = 0.1$ and $\theta = 0.01$, respectively. The experimental results show that the average ASR on the TN is 1 for $N_{\text{SPRITZ1}}^{N-BaloT}$ as a SN and 0.9960 for $N_{\text{SPRITZ2}}^{N-BaloT}$ as a SN. Regarding the DGA datasets, the average ASR on the SN is successful (except for the JSMA attack that fails on the SN $N_{\text{SPRITZ2}}^{DGA}$ with the parameter $\theta = 0.01$), and we achieve the transferability for both the I-FGSM attack and the LBFGS attack. Considering the $N_{\text{SPRITZ2}}^{DGA}$ as a SN, the transferability happens for the I-FGSM attack with both parameters $\varepsilon = 0.01$ and $\varepsilon = 0.001$, and their average ASR on the TN is 1. Additionally, the LBFGS attack with the default parameter is transferable given the $N_{\text{SPRITZ2}}^{DGA}$ as a SN, and the average ASR on the TN is 0.63. Differently from the previous scenario, the cross model transferability demonstrates five different cases in computer networks where the adversary can perfectly fool the TN. Therefore, the absence of transferability in the JSMA, IFGSM, and LBFGS attacks scenarios in cross-modal transferability is considerably more significant. As a result, we may expect shallow and deep architectures to obtain similar peculiar features in some scenarios, thus enhancing the attack transferability. For additional tests, we used the RIPE dataset in this setting while TN was trained on it. We noticed that non-transferability exists in this scenario as well, when RIPE dataset is employed.

Cross Model and Training Transferability: We consider three case scenarios, and present in Table VII our evaluation on two different networks (SPRITZ1 and SPRITZ2) that are trained on three different datasets (N-BaloT, DGA, and RIPE Atlas). As it can be seen, the average ASR on the SN is successful (except for the JSMA attack that fails on the SN $N_{\text{SPRITZ1}}^{N-BaloT}$ with the parameter $\theta = 0.01$, and the SN $N_{\text{SPRITZ2}}^{DGA}$ with the parameter $\theta = 0.01$). We notice that none of the five adversarial attacks performed in our experiments are transferable from the SN to the TN, and the average ASR on the TN is almost null. Therefore, the feasibility of the cross model and training transferability in computer networks is unlikely hard to happen. Table VII shows that, in this scenario, the attack success rate decreases, as we have a full mismatch in networks and datasets. For additional experiments, we used the RIPE as a mismatch dataset for TN. In a complete mismatch setting, we conclude that SN and TN are non-transferable. Figure 5 shows a visual representation of adversarial attacks that transfer from SN to TN.

B. Anti-Exploratory Evasion Defense Mechanisms

The outcomes of our study reveal that the adversarial samples are often non-transferable, contrary to what we observe in standard classification tasks, such as multimedia forensics and similar applications. This finding is important because we can leverage the limitation of transferability to hinder the attack. For instance, an LK scenario could be imposed somehow, as with standard ML techniques, to prevent adversarial examples. However, we identified various adversarial attacks in different
scenarios that enable the transferability property, meaning that the adversarial attacks can entirely mislead the TN. In this case, we suggest two current defense mechanisms that prevent adversarial transferability. The first strategy consists of fine-tuning the classifier using the Most Powerful Attacks (MPAs) whenever the transferability happens for a given adversarial attack [16]. Consequently, we improve the security of the TN against different adversarial attacks only for the attacks that transfer between the SN and TN (refer to Table VI). We apply the fine-tuning technique on the TN, and we report our

| SN                  | TN                  | Accuracy          | Attack Type          | PSNR  | $L_1$ dist | Max. dist | ASR (SN) | ASR (TN) |
|---------------------|---------------------|-------------------|----------------------|-------|------------|-----------|----------|----------|
| N-BaIoT SPRITZ1     | N-DGA SPRITZ1       | SN = 98.90%, TN = 100% | I-FGSM, $\epsilon = 0.1$ | 18.65 | 22.07      | 65.12     | 1.0000   | 0.0480   |
| N-BaIoT SPRITZ1     | N-DGA SPRITZ1       | SN = 98.90%, TN = 100% | I-FGSM, $\epsilon = 0.001$ | 18.93 | 21.26      | 62.71     | 1.0000   | 0.0000   |
| N-BaIoT SPRITZ1     | N-DGA SPRITZ1       | SN = 98.90%, TN = 100% | JSMA, $\theta = 0.1$ | 19.06 | 20.94      | 61.76     | 1.0000   | 0.0000   |
| N-BaIoT SPRITZ1     | N-DGA SPRITZ1       | SN = 98.90%, TN = 100% | JSMA, $\theta = 0.01$ | 18.49 | 7.29       | 178.50    | 1.0000   | 0.1100   |
| N-BaIoT SPRITZ1     | N-DGA SPRITZ1       | SN = 98.90%, TN = 100% | LBFGS, default parameter | 22.57 | 9.98       | 106.35    | 1.0000   | 0.0000   |
| N-BaIoT SPRITZ1     | N-DGA SPRITZ1       | SN = 98.90%, TN = 100% | PGD, default parameter | 19.59 | 19.71      | 54.00     | 0.9890   | 0.0020   |
| N-BaIoT SPRITZ1     | N-DGA SPRITZ1       | SN = 98.90%, TN = 100% | DeepFool, default parameter | 20.90 | 12.11      | 142.85    | 0.9500   | 0.0400   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 98.90%, TN = 100% | I-FGSM, $\epsilon = 0.1$ | 31.4618 | 4.81     | 17.20     | 1.0000   | 0.0050   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 100%, TN = 98.90% | I-FGSM, $\epsilon = 0.001$ | 32.51 | 4.21       | 15.92     | 1.0000   | 0.0000   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 100%, TN = 98.90% | JSMA, $\theta = 0.1$ | 31.23 | 1.47       | 62.50     | 1.0000   | 0.2000   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 100%, TN = 98.90% | JSMA, $\theta = 0.01$ | 35.95 | 1.077      | 17.85     | 1.0000   | 0.0010   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 98.90%, TN = 100% | LBFGS, default parameter | 37.19 | 1.41       | 23.38     | 1.0000   | 0.0000   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 98.90%, TN = 100% | PGD, default parameter | 33.75 | 3.58       | 13.93     | 1.0000   | 0.0000   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 98.90%, TN = 100% | DeepFool, default parameter | 36.98 | 2.20       | 38.08     | 1.0000   | 0.0300   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 98.90%, TN = 100% | I-FGSM, $\epsilon = 0.1$ | 17.98 | 22.67      | 58.06     | 1.0000   | 0.0360   |
| N-DGA SPRITZ1       | N-BaIoT SPRITZ1     | SN = 98.90%, TN = 100% | I-FGSM, $\epsilon = 0.001$ | 18.64 | 20.96      | 53.47     | 1.0000   | 0.0000   |

Table V: Experimental Results for Cross Training Transferability on SPRITZ1 and SPRITZ2 Architecture (N-BaIoT and DGA Datasets Are Used Interchangeably in the SN and the TN, While RIPE Atlas Dataset Is Used Only in the TN).
Another strategy to prevent adversarial transferability relies on using the LSTM architecture as the TN rather than the CNN, namely as mismatch architecture. Therefore, the TN is different from the classification structure of the original CNN. In this strategy, the attacker has less attack information than the previous system with no knowledge of the TN architecture. We verify this approach in a Cross-Model setting by considering two adversarial attacks enabling the transferability: the JSMA attack with a parameter $\theta = 0.01$ and the LBFGS attack with a default parameter.

In this case, the attack on the source network fails and we cannot transfer it to the target network. Therefore, the numerical values of the average PSNR, the average $L_1$ dist, and the average maximum distortion are not possible.
Table VII
Experimental Results for Cross Model and Training Transferability (the architectures and datasets are used interchangeably in the SN an the TN, except for RIPE Atlas Dataset which is used only in the TN)

| SN          | TN          | Accuracy | Attack Type     | PSNR | L₁ dist | Max. dist | ASR (SN) | ASR (TN) |
|-------------|-------------|----------|-----------------|------|---------|-----------|----------|----------|
| N-BaIoT     | DGA        | SN = 98.80% , TN = 100% | I-FGSM,  ε = 0.1 | 18.65 | 22.07   | 64.78     | 1.0000   | 0.0000   |
| SPRTZ1      | DGA        | SN = 98.80% , TN = 100% | I-FGSM,  ε = 0.01 | 18.93 | 21.26   | 62.71     | 1.0000   | 0.0000   |
| N-BaIoT     | DGA        | SN = 98.80% , TN = 100% | I-FGSM,  ε = 0.001 | 19.06 | 20.93   | 61.72     | 1.0000   | 0.0000   |
| SPRTZ1      | DGA        | SN = 98.80% , TN = 100% | JSMA,  θ = 0.1   | 18.49 | 7.29    | 178.50    | 1.0000   | 0.0000   |
| N-BaIoT     | DGA        | SN = 98.80% , TN = 100% | JSMA,  θ = 0.01  | -    | -       | -         | -        | -        |
| SPRTZ1      | DGA        | SN = 98.80% , TN = 100% | LBFGS, default parameter | 22.57 | 10.46   | 110.67    | 0.9910   | 0.0000   |
| N-BaIoT     | DGA        | SN = 98.80% , TN = 100% | DeepPool, default parameter | 20.87 | 12.14   | 143.19    | 0.9676   | 0.0324   |
| SPRTZ1      | DGA        | SN = 99.60% , TN = 99.60% | I-FGSM,  ε = 0.1 | 28.49 | 6.03    | 24.15     | 1.0000   | 0.0000   |
| N-BaIoT     | DGA        | SN = 99.60% , TN = 99.60% | I-FGSM,  ε = 0.01 | 28.62 | 5.99    | 23.25     | 1.0000   | 0.0000   |
| SPRTZ1      | DGA        | SN = 99.60% , TN = 99.60% | I-FGSM,  ε = 0.001 | 28.92 | 5.76    | 22.64     | 1.0000   | 0.0000   |
| N-BaIoT     | DGA        | SN = 99.60% , TN = 99.60% | JSMA,  θ = 0.1   | 31.17 | 1.28    | 56.56     | 1.0000   | 0.0000   |
| SPRTZ1      | DGA        | SN = 99.60% , TN = 99.60% | JSMA,  θ = 0.01  | -    | -       | -         | -        | -        |
| N-BaIoT     | DGA        | SN = 99.60% , TN = 99.60% | LBFGS, default parameter | 34.22 | 1.40    | 33.93     | 0.9860   | 0.0040   |
| SPRTZ1      | DGA        | SN = 99.60% , TN = 99.60% | PGD, default parameter | 29.24 | 5.99    | 22.24     | 1.0000   | 0.0000   |
| N-BaIoT     | DGA        | SN = 99.60% , TN = 99.60% | DeepPool, default parameter | 31.02 | 2.59    | 53.42     | 1.0000   | 0.0104   |
| SPRTZ1      | DGA        | SN = 99.80% , TN = 99.80% | I-FGSM,  ε = 0.1 | 18.65 | 22.07   | 64.78     | 1.0000   | 0.0000   |
| N-BaIoT     | RIPE       | SN = 98.80% , TN = 99.80% | JSMA,  θ = 0.1   | 18.49 | 7.29    | 178.50    | 1.0000   | 0.0000   |
| SPRTZ1      | RIPE       | SN = 98.80% , TN = 99.80% | LBFGS, default parameter | 22.57 | 10.46   | 110.67    | 0.9910   | 0.0000   |
| N-BaIoT     | RIPE       | SN = 98.80% , TN = 99.80% | PGD, default parameter | 19.59 | 19.71   | 54.00     | 1.0000   | 0.0000   |
| SPRTZ1      | RIPE       | SN = 98.80% , TN = 99.80% | DeepPool, default parameter | 20.87 | 12.14   | 143.19    | 0.9676   | 0.0200   |

FAIL*: In this case, the attack on the source network fails and we cannot transfer it to the target network. Therefore, the numerical values of the average PSNR, the average L₁ dist, and the average maximum distortion are not possible.

C. Discussion
Combining all the experimental results, we demonstrate that among five different adversarial attacks, three of them are transferable from the SN to the TN, which are: the JSMA attack, the I-FGSM attack, and the LBFGS attack. Therefore, given a CNN-based model in computer network, specific cases of generated adversarial examples can perfectly fool another network model even with different architectures. In particular, we found that both models in the SN and TN enable the same misclassification; hence, the transferability property. To that end, the adversary can leverage a surrogate model with different architectures and classes of algorithms to approximate the adversarial example for the TN. We note that in the cross model transferability, the features learned by the network might influence somehow to some degree by the underlying dataset, and this aspect requires additional research.

Moreover, we performed multiple adversarial attacks on the SN for 250 manipulated examples to evaluate their...
computational cost. On the other hand, after detecting adversarial attacks on SN, we remark that the processing testing time for TN is relatively fast. In Table IX, we report our results on the execution running attack time on SN in seconds. This study, as we expected, will also address the issue of usability in botnet attacks. To elaborate more, the authors in [61] suggested a solution for botnet attack detection in the IoT environment using CNN. As a result, most ML and DL methods are subject to various adversarial attacks if we have a classification algorithm for botnet detection. Additionally, the experts in the research area demonstrate that an attack in the SN might not be enough to deceive the TN, based on our studies and other research in computer networks.

### VI. Conclusion and Future Work

In this paper, we thoroughly dissected the transferability of adversarial examples in computer networks. We investigated three transferability scenarios: the cross training, the cross model, and the cross model and training. Considering three well-known datasets (N-BaIoT, DGA, and RIPE Atlas), we found that the transferability happens in specific cases where the adversary can perfectly transfer the examples from the SN to the TN. Although in other cases, such as the Cross Model and Training transferability, it is difficult to transfer the adversarial examples from the SN to the TN. The adversary still leverages the existing cases where the transferability happens. Due to the black-box nature of deep neural networks, our understanding is very narrow regarding the decisions enabling the transferability between the CNN-based models. This knowledge gap is a critical issue, especially for security-oriented applications in computer networks. Therefore, additional research is required to understand what are the main factors responsible for the transferability property.

According to our experiments, the JSMA and I-FGSM attacks are more transferable than other adversarial attacks, particularly in the cross model transferability. One probable rationale is that when JSMA and I-FGSM attacks alter particular features (those to which the network is more vulnerable), it is likely to overfit the attacked model. It is also worth noting that the transferability is not symmetric in terms of the training datasets for a particular task. These results indicate that the underlying dataset may influence the network’s learned features in some way and to some extent in computer network applications. Thus, we need to investigate this area in future work to understand why computer networks are less vulnerable to attack transferability. Our future work will focus on improving the power of adversarial attacks to increase their transferability. However, this is not a straightforward process, considering the complexity of the decision boundary learned by CNNs or even other deep neural networks.

### Table VIII

| Parameter       | 1-FGSM, $\varepsilon = 0.01$ | IFGSM, $\varepsilon = 0.001$ | LBFGS, Default |
|-----------------|-------------------------------|-------------------------------|----------------|
| $\theta = 0.01$ | 0.0000                        | 0.0100                        | 0.2010         |
| $\varepsilon = 0.01$ |                               | 0.0000                        | 0.0000         |
| $\varepsilon = 0.001$ |                               | 0.0000                        | 0.0000         |

### Table IX

| SN        | Attack Type     | Dataset1   | Dataset2 | Dataset3 | Exec. seconds (Dataset1) | Exec. seconds (Dataset2) | Exec. seconds (Dataset3) |
|-----------|-----------------|------------|----------|----------|--------------------------|--------------------------|--------------------------|
| SPRITZ1   | I-FGSM, $\varepsilon = 0.1$ | N-BaIoT    | DGA      | RIPE     | 127.2                    | 127.8                    | 130.8                    |
| SPRITZ1   | L-FGSM, $\varepsilon = 0.01$ | N-BaIoT    | DGA      | RIPE     | 1229.4                   | 1294.2                   | —                        |
| SPRITZ1   | L-FGSM, $\varepsilon = 0.001$ | N-BaIoT    | DGA      | RIPE     | 12238.8                  | 12766.2                  | —                        |
| SPRITZ1   | JSMA, $\theta = 0.1$     | N-BaIoT    | DGA      | RIPE     | 1824.6                   | 625.2                    | 1840.2                   |
| SPRITZ2   | L-FGSM, $\varepsilon = 0.1$ | N-BaIoT    | DGA      | RIPE     | 265.2                    | 2694.2                   | —                        |
| SPRITZ2   | I-FGSM, $\varepsilon = 0.01$ | N-BaIoT    | DGA      | RIPE     | 2636.4                   | 2698.8                   | —                        |
| SPRITZ2   | I-FGSM, $\varepsilon = 0.001$ | N-BaIoT    | DGA      | RIPE     | 26306.16                 | 26818.14                 | —                        |
| SPRITZ2   | JSMA, $\theta = 0.1$     | N-BaIoT    | DGA      | RIPE     | 1954.8                   | 1231.8                   | —                        |
| SPRITZ2   | LBFGS, default parameter | N-BaIoT    | DGA      | RIPE     | 4723.8                   | 3115.8                   | —                        |
| SPRITZ2   | PGD, default parameter  | N-BaIoT    | DGA      | RIPE     | 2715.2                   | 2016.5                   | —                        |
| SPRITZ2   | DeepPool, default parameter | N-BaIoT    | DGA      | RIPE     | 18.6                     | 22.2                     | —                        |

### References

[1] S. Dong, P. Wang, and K. Abbas, “A survey on deep learning and its applications,” *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100379.
[53] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, “DREBIN: Effective and explainable detection of android malware in your pocket,” in Proc. NDSS, vol. 14, 2014, pp. 23–26.

[54] C. Sabottke, O. Suciu, and T. Dumitras, “Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits,” in Proc. 24th USENIX Security Symp., 2015, pp. 1041–1056.

[55] Y. Liu et al., “Cloudy with a chance of breach: Forecasting cyber security incidents,” in Proc. 24th USENIX Security Symp., 2015, pp. 1009–1024.

[56] L. Deng, “The MNIST database of handwritten digit images for machine learning research,” IEEE Signal Process. Mag., vol. 29, no. 6, pp. 141–142, Nov. 2012.

[57] E. Nowroozi, M. Barni, and B. Tondi, “Machine learning techniques for image forensics in adversarial setting,” Ph.D. dissertation, Dept. Inf. Eng. Math., Univ. Siena, Siena, Italy, 2020.

[58] O. Suciu, R. Marginean, Y. Kaya, H. Daume, and T. Dumitras, “When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks,” in Proc. 27th USENIX Security Symp. (USENIX Security), Baltimore, MD, USA, Aug. 2018, pp. 1299–1316.

[59] F. Chollet. “Keras.” 2015. [Online]. Available: https://github.com/fchollet/keras.

[60] E. Nowroozi. “Transferability Forensic Networks.” 2019. [Online]. Available: https://github.com/ehsannowroozi/TransferabilityForensicNetworks.

[61] K. N. Karaca and A. Çetin, “Botnet attack detection using convolutional neural networks in the IoT environment,” in Proc. Int. Conf. Innovat. Intell. Syst. Appl. (INISTA), 2021, pp. 1–6.

Ehsan Nowroozi (Member, IEEE) received the Doctoral degree from the University of Siena, Italy. He is a Postdoctoral Researcher with the Faculty of Engineering and Natural Sciences and the Center of Excellence in Data Analytics (VERIM), Sabanci University’s, Istanbul, Turkey. He works also as an external collaborator with the Security and Privacy Research Group, University of Padua, Italy.

Mauro Conti (Fellow, IEEE) received the Ph.D. degree from the Sapienza University of Rome, Italy. He is a Full Professor with the University of Padua, Italy, and an Affiliate Professor with the TU Delft and University of Washington, Seattle. He was a Postdoctoral Researcher with Vrije Universiteit Amsterdam, The Netherlands. In 2011, he joined as an Assistant Professor with the University of Padua, where he became an Associate Professor in 2015, and a Full Professor in 2018. He has been a Visiting Researcher with GMU, UCLA, UCI, TU Darmstadt, UF, and FIU. His research is also funded by companies, including Cisco, Intel, and Huawei. He has been awarded with a Marie Curie Fellowship by the European Commission and a Fellowship by the German DAAD.

Mohammad Hajian Berenjestanaki received the bachelor’s degree in electrical engineering, electronics from the University of Mazandaran, Iran, and the master’s degree in electrical engineering, communications, cryptography from the University of Tehran in 2016. He is currently a Research Assistant with the SPRITZ Security and Privacy Research Group, University of Padua, Italy. His areas of interests include networks security and cryptography, deep learning and machine learning, block-chain, and distributed ledger technology. He conducts his new researches in applied machine learning in Cybersecurity.

Mauro Conti (Fellow, IEEE) received the Ph.D. degree from the Sapienza University of Rome, Italy. He is a Full Professor with the University of Padua, Italy, and an Affiliate Professor with the TU Delft and University of Washington, Seattle. He was a Postdoctoral Researcher with Vrije Universiteit Amsterdam, The Netherlands. In 2011, he joined as an Assistant Professor with the University of Padua, where he became an Associate Professor in 2015, and a Full Professor in 2018. He has been a Visiting Researcher with GMU, UCLA, UCI, TU Darmstadt, UF, and FIU. His research is also funded by companies, including Cisco, Intel, and Huawei. He has been awarded with a Marie Curie Fellowship by the European Commission and a Fellowship by the German DAAD.

Abdeslam El Fergougui received the Ph.D. degree in computer science from the Mohammed V University in Rabat, Morocco. He is currently a Professor with the Department of Computer science, Faculty of sciences Meknes, Moulay Ismail University. Furthermore, he assured several advanced training workshops in its field of action. He also assured several training courses in several Moroccan and international universities. He has managed several national and international projects. His current research interests include network, sensor network, cryptography, and security. He is an Active Member of University Agency of Francophonie, where he provides training at the international level. He is a member of the reading committee of several journals and conferences in his field.

Yasine Mekdad (Member, IEEE) received the M.Sc. degree in cryptography and information security from the Mohammed V University of Rabat, Morocco. He works with the Computer Networks and Systems Laboratory, Moulay Ismail University of Meknes, Morocco. He holds a Guest Researcher position with the SPRITZ Research Group, University of Padua, Italy. He is currently working as a Research Scholar with the Cyber-Physical Systems Security Lab, Florida International University, Miami, FL, USA. His research interest principally cover security and privacy problems in the Internet of Things, Industrial Internet-of-Things, and cyber–physical systems.