Self-Taught Black-Box Adversarial Attack to Multilayer Network Automation

Xiaoqin Pan[†‡] and Zuqing Zhu[†]
[†]School of Information Science and Technology, University of Science and Technology of China, Hefei, China
[‡]Engineering Technology Center, Southwest University of Science and Technology, Mianyang, China
[†]Email: {zqzhu}@ieee.org

***Abstract*—Although the idea of multilayer network automation (MLy-NA) has gained its initial success due to the advances on software-defined networking (SDN) and machine learning (ML), the security vulnerabilities brought by the reduction of human intervention in network operation should not be ignored. In this work, we study how to mislead the ML-based classifiers for anomaly detection in MLy-NA of packet-over-optical networks, and propose a self-taught black-box adversarial attack (BB-AdA) scheme. Specifically, we design novel algorithms to synthesize and label training data for the substitute classifier used by the BB-AdA. The algorithms not only generate synthetic data to cover all the anomaly types based on a small set of legitimate telemetry data only containing the "Normal" type, but also label the data with minimized queries to the target classifier in MLy-NA. Extensive simulations are conducted with the telemetry data collected from a real-world packet-over-optical network testbed. The results show that with our self-taught BB-AdA, an attacker can interact with an MLy-NA system quietly and efficiently to train itself adaptively, generate well-crafted adversarial samples to mislead the target classifiers in different ML architectures and severely affect their performance on anomaly detection, and disturb the operation of MLy-NA in the hard-to-detect manner.**

***Index Terms*—Machine learning (ML), Multilayer Networks, Network automation, Adversarial samples, Black-box attack.**

I. Introduction

Nowadays, emerging network services have dramatically reshaped metro/core networks [1], and brought new challenges to network control and management (NC&M), especially for anomaly detection and location. The difficulties mainly come from the following three aspects. Firstly, network infrastructures are becoming more heterogeneous with the advances on various physical-layer technologies [2–6], which has complicated the relation between network state and quality-of-service (QoS) metrics. Secondly, the wide usage of virtualization technologies, such as virtual network slicing [7, 8] and network function virtualization (NFV) [9, 10], improves the flexibility of networks at the cost of loosening the correlation between network services and physical devices. Finally, to realize high resource utilization and satisfactory QoS, operators of metro/core networks need to manage multilayer network architectures with intelligent and timely decisions [11].

The aforementioned issues can be mostly addressed by introducing multilayer network automation (MLy-NA), which has been promoted by the symbiosis of software-defined networking (SDN) [12] and machine learning (ML) [13]. Specifically, MLy-NA can be realized by inserting an ML-based data analytics (ML-DA) module into the centralized control plane and establishing the decision loop of "observe-analyze-act" (as shown in the left of Fig. 1): 1) *Observe*: the control plane collects realtime telemetry data regarding network elements in the packet and optical layers, by leveraging the programmable data plane (PDP) [14], 2) *Analyze*: the ML-DA analyzes the telemetry data for accurate anomaly detection/location, and 3) *Act*: the SDN controller makes timely decisions to address the anomalies detected/located by the ML-DA [15]. Therefore, MLy-NA not only makes NC&M more efficient but also reduces unnecessary human intervention in network operation.

Despite the bright future of MLy-NA, we should be careful about the security issues caused by reducing human intervention in network operation, especially when considering to apply MLy-NA in real-world metro/core networks. As shown in Fig. 1, both the telemetry data from data collection agents (DCAs) and analysis results from ML-DA are transmitted through control channels, which can span over geographically-distributed locations and are set up with the protocols (*e.g.*, the transport layer security (TLS)) that are known to be vulnerable to man-in-the-middle attacks [16]. This makes the control channels vulnerable to eavesdropping and tampering. Hence, a malicious party can easily disturb the operation of MLy-NA with adversarial attacks, *i.e.*, misleading ML-DA by sneaking well-craft adversarial samples in its inputs [17].

Therefore, it is relevant to study the potential ways that can be leveraged to launch adversarial attacks for disturbing the operation of MLy-NA and analyze their consequences. Previously, the authors of [18] showed that the traffic prediction by ML could be easily misled to generate incorrect forecasts by the adversarial attacks, which only injected hard-to-detect adversarial traffic samples. Note that, time series prediction is just one kind of tasks that ML-DA performs in MLy-NA, while another kind is classification tasks, which are also frequently used, especially for anomaly detection/location. This motivates us to investigate the adversarial attacks to MLy-NA, considering the classification for anomaly detection/location.

Fig. 1 explains the principle of the black-box adversarial attack (BB-AdA) that can be launched to disturb the ML-based classifiers for MLy-NA [19]. Specifically, the BB-AdA is launched with the following procedure. First, the attacker gets a small set of legitimate telemetry data by eavesdropping the control channels, and build a substitute classifier that will be used to mimic the operation of the target classifier in

978-1-6654-3540-6/22 © 2022 IEEE

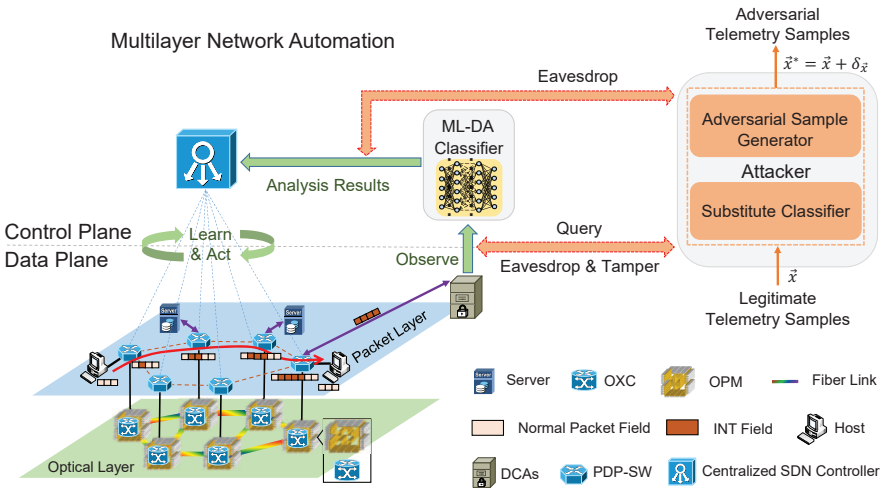


Fig. 1. Multilayer network automation and black-box adversarial attack to it, PDP-SW: programmable data plane switch, OXC: optical cross-connect, OPM: optical performance monitor, DCAs: data collection agents, ML-DA: machine learning based data analytics.

ML-DA. Second, the attacker trains the substitute classifier with a training set that contains both legitimate telemetry data and synthetic data generated locally, where the synthetic data is labeled by first sneaking it in the control channels between DCAs and ML-DA and then tapping the control channel between ML-DA and SDN controller to eavesdrop the classification results from the target classifier. Third, after being trained, the substitute classifier can help the attacker to craft adversarial telemetry samples, which will be injected back to the control channels between DCAs and ML-DA, for misleading the target classifier in ML-DA.

We refer to the attack strategy above as "black-box" because the attacker does not need to have any pre-knowledge about the training/testing data sets or architecture of the target classifier [19]. Note that, comparing with directly tampering the classification results from ML-DA, BB-AdA is more sophisticated and much more difficult to be detected [17, 19]. This justifies the motivation and necessity of studying it thoroughly. Nevertheless, the question "*Can BB-AdA be launched to disturb the operation of the MLy-NA in practical metro/core networks?*" can only be answered after the following two issues have been addressed properly. First, as a network mostly operates in the normal state [20], it will be difficult for the attacker to eavesdrop a set of legitimate telemetry data that contains sufficient anomaly types. Second, when generating and labeling synthetic data, the queries from the attacker to the target classifier should be minimized to avoid being detected.

In this work, we propose a self-taught BB-AdA scheme to address the two aforementioned issues. Specifically, we design novel algorithms to synthesize and label training data for the substitute classifier. The algorithms not only generate synthetic data to cover all the anomaly types based on a small set of legitimate telemetry data only containing the "Normal" type, but also label the data with minimized queries to the target classifier. We conduct simulations with the telemetry data collected from a real-world packet-over-optical network testbed to evaluate our proposal. The results demonstrate that with our proposed self-taught BB-AdA, an attacker can interact with an MLy-NA system quietly and efficiently to train itself adaptively, generate well-crafted adversarial samples to mislead the target classifiers in different ML architectures and severely affect their performance on anomaly detection, and disturb the operation of MLy-NA in the hard-to-detect manner.

The rest of the paper is organized as follows. Section II explains the principle of BB-AdA. We design our self-taught BB-AdA scheme in Section III. Simulation results are shown in Section IV. Finally, Section V summarizes the paper.

II. Operation Principle

As the attacker does not have any pre-knowledge about the target classifier in ML-DA, we model the target classifier as a black box, denoted as $\mathcal{T}$. The input to $\mathcal{T}$ is a multi-dimensional vector x, in which each dimension corresponds to a type of telemetry data about the multilayer packet-over-optical network, *e.g.*, optical signal-to-noise-ratio (OSNR) of a lightpath, or packet processing latency in a PDP switch. To launch adversarial attacks, the attacker is capable of querying $\mathcal{T}$ with an arbitrary x by eavesdropping and tampering the control channels between DCAs and ML-DA, and collecting the corresponding label $\mathcal{T}(x)$ through tapping the control channel between ML-DA and SDN controller. The adversarial attack is essentially on the output integrity of the target classifier. Specifically, the attacker finds a minimal perturbation δ_x to form an adversarial sample $x^* = x + \delta_x$ to mislead $\mathcal{T}$ to output incorrect classification, and thus δ_x is expressed as

$$\delta_x = \text{argmin}\, ||x^* - x||_p \text{ for } \mathcal{T}(x^*) \neq \mathcal{T}(x), \tag{1}$$

where δ_x is measured with the l_p norm.

As shown in Fig. 1, a key task of the attacker is to train the substitute classifier, denoted as $\mathcal{S}$, to imitate the operation of the target classifier $\mathcal{T}$ in ML-DA. After getting $\mathcal{S}$ trained, the attacker can intercept a legitimate telemetry sample x by tapping on the control channels, and input x to $\mathcal{S}$ to get the minimal perturbation δ_x. Then, it crafts an adversarial sample

x^* according to Eq. (1) with the help of $\mathcal{S}$, and injects x^* back to the control channel between DCA and ML-DA, for misleading $\mathcal{T}$ to output incorrect classification.

III. Design of Self-Taught BB-AdA

In this section, we design the procedure of the self-taught BB-AdA, which includes two major steps: 1) synthesizing the training data set for $\mathcal{S}$, and 2) crafting adversarial samples.

A. Synthesizing Training Data Set

Algorithm 1 explains our proposed procedure of synthesizing the training data set for the substitute classifier $\mathcal{S}$. Before invoking *Algorithm* 1, the attacker needs to hack into the control channels among the DCAs, ML-DA, and SDN controller to get a small set of labeled telemetry samples $\{D, \mathcal{T}(D)\}$, where $\mathcal{T}(D)$ stores the labels of the legitimate samples in D. *Line* 1 is for the initialization, and the usages of the coefficients will be explained later. Then, preprocessing is performed for improving the efficiency of data synthesizing (*Line* 2). Specifically, the attacker calculates the convex hull D^* of the samples in "Normal" type in D with the Quickhull algorithm [21], extracts the samples in other types, *i.e.*, anomaly types (if there is any[1]), and merges them with D^* to get the initial seed set D_0. *Line* 3 constructs the training data set $\{O, \mathcal{T}(O)\}$ for $\mathcal{S}$ with $\{D_0, \mathcal{T}(D_0)\}$.

Next, the while-loop covering *Lines* 4-25 extends the training data set in iterations until the maximum number of queries ($q_{\max}$) or the maximum number of iterations ($i_{\max}$) has been reached. *Line* 5 gets perturbations N_i ($i = 0$ initially) according to the Gaussian distribution $\mathcal{N}(\mu, \sigma)$, and adds them to the samples in seed set D_i for synthesizing new samples in P_i (*i.e.*, $P_i = D_i + N_i$). To minimize the number of queries to the target classifier $\mathcal{T}$, we use *Lines* 6-10 to select the samples in P_i, which are outside of the convex hull set D^*, to insert in set P_i'. This is because only these samples are necessary, *i.e.*, labeling other samples in P_i with $\mathcal{T}$ will not bring any new information for the training of $\mathcal{S}$. The same filtering is applied to D_i to insert the necessary samples in set D_i'. *Line* 11 updates the D_i with D_i'. Then, *Lines* 12-13 query $\mathcal{T}$ with P_i' to label its samples and obtain a set of labeled samples $\{P_i', \mathcal{T}(P_i')\}$, which are then included in the training set $\{O, \mathcal{T}(O)\}$.

To equalize samples in different types, *Lines* 14-19 select the samples whose type has less than γ samples in P_i', put them in the selected set S_i, and merge S_i with D_i to get the seed set for the next iteration $\{D_{i+1}, \mathcal{T}(D_{i+1})\}$. *Line* 20 invokes *Algorithm* 2 to perform the data argument to further extend $\{O, \mathcal{T}(O)\}$ with the selected set $\{S_i, \mathcal{T}(S_i)\}$. Finally, after the data synthesization has been accomplished, *Line* 26 returns the synthetic training data set $\{O, \mathcal{T}(O)\}$ for $\mathcal{S}$.

The procedure of the data argumentation for extending the training data set $\{O, \mathcal{T}(O)\}$ with the selected set $\{S_i, \mathcal{T}(S_i)\}$ is shown in *Algorithm* 2. Intuitively, the data argumentation

[1]Here, we consider the cases in which there are samples in anomaly types in D just for not losing the generality, while our proposal can work for the extreme case where all the samples in D are in "Normal" type.

Algorithm 1: Synthesizing Training Data Set for $\mathcal{S}$

Input: maximum iterations $i_{\max}$, maximum queries $q_{\max}$, initial set of labeled samples $\{D, \mathcal{T}(D)\}$.
Output: synthetic training data set $\{O, \mathcal{T}(O)\}$.

1 initialize coefficients $\{\gamma, \delta, \mu, \sigma, \alpha\}$, $q = i = 0$;
2 get initial seed set $\{D_0, \mathcal{T}(D_0)\}$ and convex hull set $\{D^*, \mathcal{T}(D^*)\}$ with preprocessing;
3 $\{O, \mathcal{T}(O)\} = \{D_0, \mathcal{T}(D_0)\}$;
4 **while** $q < q_{\max}$ **do**
5 get Gaussian perturbations N_i with $\mathcal{N}(\mu, \sigma)$ and new synthesized samples $P_i = N_i + D_i$, and set $P_i' = \emptyset$;
6 **for** *each sample* $x \in P_i$ **do**
7 **if** x *is outside of convex hull set* D^* **then**
8 insert x in P_i', get the sample $d \in D_i$ for generating x and insert d in D_i';
9 **end**
10 **end**
11 $D_i = D_i'$;
12 query $\mathcal{T}$ to label samples in P_i', and $q = q + |P_i'|$;
13 $O = O \cup P_i'$, $\mathcal{T}(O) = \mathcal{T}(O) \cup \mathcal{T}(P_i')$;
14 **for** *each data type in* P_i' *according to* $\mathcal{T}(P_i')$ **do**
15 **if** *samples in the type is less than* γ **then**
16 put the samples and labels in S_i and $\mathcal{T}(S_i)$;
17 **end**
18 **end**
19 $D_{i+1} = D_i \cup S_i$, $\mathcal{T}(D_{i+1}) = \mathcal{T}(D_i) \cup \mathcal{T}(S_i)$;
20 invoke *Algorithm* 2 for data augmentation with $\{S_i, \mathcal{T}(S_i)\}$ to update $\{O, \mathcal{T}(O)\}$, A, B and C;
21 $i = i + 1$;
22 **if** $i \geq i_{\max}$ **then**
23 **break**;
24 **end**
25 **end**
26 **return**($\{O, \mathcal{T}(O)\}$);

should synthesize more samples near the boundaries of classification to improve the accuracy of the training of $\mathcal{S}$. Fig. 2 explains the data argumentation based on this idea clearly, *i.e.*, how to generate new samples based on the seed set D_i, current training data set O, and selected set S_i. *Line* 1 of *Algorithm* 2 is to initialize the temporary sets G_1, G_2, and G_3. The for-loop of *Lines* 2-12 checks each sample in $x \in S_i$. First, if its label is different from the sample $x' \in D_i$, which was used to generate it, we insert x' and x in sets G_1 and G_2, respectively (*Lines* 3-6). Second, *Lines* 7-11 select all the samples whose Euclidian distances from x are less than δ, and insert them in G_1. As shown in Fig. 2, we find the blue diamond x' and blue circle y, based on the yellow triangle $x \in S_i$. Then, we get new samples based on G_1 and G_2 with

$$m = x + \eta \cdot (x' - x), \quad x' \in G_1,\ x \in G_2, \tag{2}$$

where η is randomly selected from $[0, 1)$ (*Line* 13). Specifically, Eq. (2) applies the linear interpolation with slope η to x' and x, to get a new sample m [22]. For instance, in Fig. 2, m_1 and m_2 are generated with $\{x, x'\}$ and $\{x, y\}$, respectively.

Next, we update A, B, and C with G_3, G_1, and G_2,

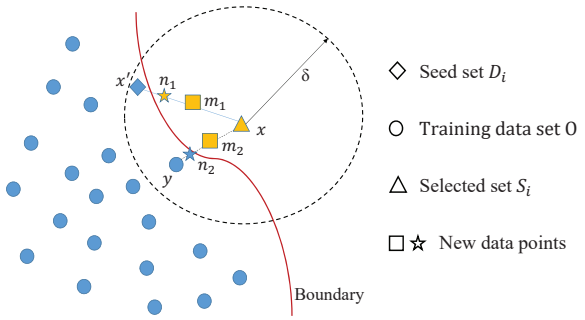


Fig. 2. Example on data argumentation of the training data set for $\mathcal{S}$.

respectively (*Line* 14), query $\mathcal{T}$ to label new samples in G_3 (*Line* 15), and update the training set $\{O, \mathcal{T}(O)\}$ to include $\{G_3, \mathcal{T}(G_3)\}$ (*Line* 16). To balance the number of samples in each type in $\{O, \mathcal{T}(O)\}$, samples in the α largest types are removed from A (*Line* 17). Then, *Lines* 18-19 apply the same logic of *Lines* 3-6 to get B' and C' based on A, B, and C, and *Line* 20 operates similarly as *Line* 13 to generate new samples in A' based on B' and C', for pushing the new samples closer to the boundaries of classification. For example, in Fig. 2, new samples n_1 and n_2 are generated with $\{m_1, x'\}$ and $\{m_2, y\}$, respectively. Finally, the new samples in A' are labeled in *Line* 21, and inserted in the training set $\{O, \mathcal{T}(O)\}$ (*Lines* 22-23).

Algorithm 2: Data Augmentation for Training Data Set

1 $G_1 = G_2 = G_3 = \emptyset$;
2 **for** *each sample* $x \in S_i$ **do**
3 get the sample $x' \in D_i$ for generating x;
4 **if** $\mathcal{T}(x') \neq \mathcal{T}(x)$ **then**
5 insert x' and x in G_1 and G_2, respectively;
6 **end**
7 **for** *each sample* $y \in O_i$ **do**
8 **if** $||x - y||_2 < \delta$ **then**
9 insert y and x in G_1 and G_2, respectively;
10 **end**
11 **end**
12 **end**
13 generate new samples with Eq. (2) based on G_1 and G_2, and put the results in G_3;
14 $A = A \cup G_3$, $B = B \cup G_1$, $C = C \cup G_2$;
15 query $\mathcal{T}$ to label samples in G_3, and $q = q + |G_3|$;
16 $O = O \cup G_3$, $\mathcal{T}(O) = \mathcal{T}(O) \cup \mathcal{T}(G_3)$;
17 remove samples in the α largest types from A, and also update B and C accordingly;
18 select the samples in B and C whose labels are different from those of samples generated based on them in A;
19 put the selected samples in B', and $C' = A$;
20 generate new samples with Eq. (2) based on B' and C', and put the results in A';
21 query $\mathcal{T}$ to label samples in A', and $q = q + |A'|$;
22 $A = A'$, $\mathcal{T}(A) = \mathcal{T}(A')$, $B = B'$, $C = C'$;
23 $O = O \cup A'$, $\mathcal{T}(O) = \mathcal{T}(O) \cup \mathcal{T}(A')$;

B. Crafting Adversarial Samples

After obtaining the training data set $\{O, \mathcal{T}(O)\}$ for the substitute classifier $\mathcal{S}$, the attacker selects a proper architecture to build $\mathcal{S}$, and trains it for imitating the operation of the target classifier $\mathcal{T}$ in ML-DA. Specifically, $\mathcal{S}$ is trained with the classic back-propagation and gradient-descent algorithm [23]. With the trained substitute classifier, the attacker can leverage existing algorithms (*e.g.*, FGSM [24] and DeepFool [25]) to craft adversarial samples for BB-AdA accordingly [19].

IV. Performance Evaluation

In this section, we perform simulations with the telemetry data collected from a real-world packet-over-optical network testbed to evaluate our self-taught BB-AdA scheme.

A. Telemetry Data Set

We collect telemetry data from a small but real packet-over-optical network testbed [26], where the optical layer consists of bandwidth-variable wavelength-selective switches (BV-WSS') and optical line systems (OLS'), and the packet layer is built with client hosts, PDP switches (PDP-SWs), and DCAs (as shown in Fig. 1). Each BV-WSS is commercially-available in the configuration of 1×9, and it can switch optical spectrum at a granularity of 12.5 GHz according to flexible grids [27]. The OLS' are also commercial products, each of which includes a pair of bandwidth-variable transponders (BV-Ts) on nodes and in-line erbium-doped fiber amplifiers (EDFAs) on the fiber links between them. Each BV-T supports line-rates ranging in $[100, 400]$ Gbps. The MLy-NA system for the packet-over-optical network collects telemetry data regarding both the optical and packet layers, and then leverages ML-DA to analyze the data for anomaly detection/location.

Specifically, the overall telemetry data set contains $\sim 95,000$ samples, each of which contains six dimensions, corresponding to OSNR, power-level, chromatic dispersion regarding the optical layer, and packet forwarding latency and bandwidth usage on ports about the packet layer, respectively. Then, each sample can be labeled to denote its anomaly type, which can be "Normal", "High Power", "Low Power", "Degraded OSNR", "WSS-Left-Shift", "WSS-Right-Shift", "High Delay", "Packet Congestion", "Packet Loss", or "Switch Misconfiguration". We build the target classifier $\mathcal{T}$ based on a deep neural network (DNN), put 90% and 10% of the samples in the telemetry data set in its training and testing sets, respectively, and train $\mathcal{T}$ to get a classification accuracy of 99.99% on the testing set.

B. Performance of Self-Taught BB-AdA

We use the BB-AdA scheme developed in our previous work [19] as the benchmark, which requires that the initial legitimate telemetry data eavesdropped by the attacker to include data in all the anomaly types, and does not optimize the queries from the attacker to the target classifier. We first assume that the initial seed set D_0 contains 770 samples, which cover all the 10 anomaly types (*i.e.*, $|D_0| = 770$ and $|\mathcal{T}(D_0)| = 10$), and for simplicity, "Normal" is also treated as an anomaly type in the following discussions. Similar as the target classifier, the substitute classifier $\mathcal{S}$ is also built with DNN. In the simulations, the attacker uses the benchmark and our self-taught BB-AdA (*i.e.*, *Algorithm* 1) to synthesize the training

data set, and trains the substitute classifier $\mathcal{S}$ with $i_{\max} = 15$ and $q_{\max} = \{4000, 5000, 6000, 7000, 8000\}$. To guarantee sufficient statistical accuracy, we average the results from 10 independent runs to get each data point in the simulations.

The accuracies of the substitute classifiers on the testing set, which are trained with different BB-AdA schemes, are listed in Table I. As expected, the accuracies of the substitute classifiers trained with both algorithms increase with the number of allowed queries (q_{max}), because more queries to the target classifier $\mathcal{T}$ can obtain a larger training data set $\{O, \mathcal{T}(O)\}$ to improve the performance of the substitute classifier. Meanwhile, we notice that our self-taught BB-AdA with *Algorithm* 1 achieves much higher accuracies than the benchmark in all the cases. Specifically, the substitute classifier trained with the training data set from *Algorithm* 1 achieves more than 99% accuracy with only $6,000$ queries, while the one trained with the benchmark cannot realize more than 95% accuracy even with $8,000$ queries. This suggests that the data synthesization of our self-taught BB-AdA is much more efficient and intelligent, such that $\mathcal{S}$ can be trained more effectively to mimic the operation of $\mathcal{T}$ precisely.

TABLE I
PERFORMANCE OF ALGORITHMS

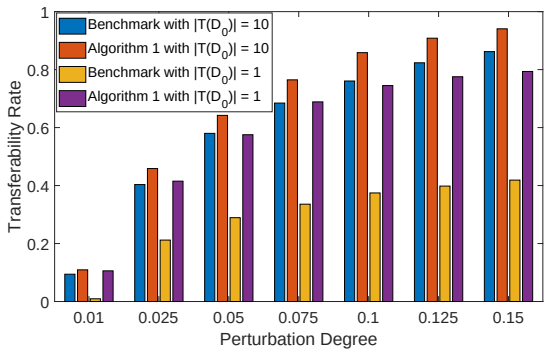
Accuracy on Testing Set					
$q_{\max}$	4,000	5,000	6,000	7,000	8,000
Benchmark	78.78%	87.59%	89.89%	90.69%	94.01%
Algorithm 1	97.57%	97.95%	99.02%	99.07%	99.07%

To further verify the performance of our proposal, the simulations also consider the cases in which the attacker has to start with an initial seed set D_0 that only contains data in "Normal" type (*i.e.*, $D_0 = 500$ and $|\mathcal{T}(D_0)| = 1$). Then, the substitute classifiers used by the attacker are still based on DNN, and they are trained by using the benchmark and *Algorithm* 1 with $q_{\max} = 8,000$. We respectively obtain the classification accuracies of the substitute classifiers as 45.92% and 88.30% on the testing set, which still confirms the effectiveness of our self-taught BB-AdA.

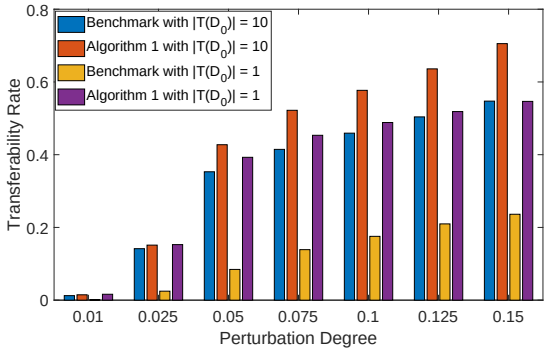
Next, we evaluate the performance of the BB-AdA schemes that use the trained substitute classifiers to craft adversarial samples. We first introduce the following performance metric.

Definition 1: We define the **transferability rate** of BB-AdA as the ratio of misclassification by the target classifier $\mathcal{T}$ on the samples tampered by the attacker, which can be used to evaluate the effectiveness of adversarial attacks.

Fig. 3 compares the transferability rates of the BB-AdA schemes that use the substitute classifiers trained with the benchmark and *Algorithm* 1, when the adversarial samples are crafted with FGSM and DeepFool. Here, the "perturbation degree" refers to δ_x in Eq. 1. It can be seen that our self-taught BB-AdA outperforms the benchmark in all the cases. Meanwhile, we notice that the transferability rate increases with $|\mathcal{T}(D_0)|$ for both schemes, but it increases more significantly when the substitute classifier is trained with the benchmark. This suggests that our self-taught BB-AdA is much less sensitive to the pre-knowledge of anomaly types in the initial telemetry data set, and thus justifies that our proposed scheme does have the capability of self-teaching. Moreover, for the extreme cases with $|\mathcal{T}(D_0)| = 1$ (*i.e.*, the initial telemetry data set only contains data in "Normal" type) in Fig. 3(a), our proposal can achieve more than 0.5 transferability rate (*i.e.*, the BB-AdA will make the target classifier in ML-DA to misclassify 50% of telemetry samples) when the perturbation degree is only 0.05. This confirms the effectiveness and practicalness of *Algorithm* 1, because the BB-AdA with it can effectively mislead the target classifier in ML-DA, when the attacker has to start with an initial telemetry data set that only contains data in "Normal" type and the maximum data tampering for generating the adversarial samples cannot exceed 0.05.



(a) FGSM



(b) DeepFool

Fig. 3. Transferability rate of BB-AdA schemes.

C. Generalization of Self-Taught BB-AdA

Finally, we conduct simulations to verify that our proposal can affect the target classifiers, which are built with various structures. Specifically, we use the logistic regression (LR), decision tree (DT) and support vector machine (SVM) to architect the target classifier $\mathcal{T}$, train them in the same way as in Section IV-A. Then, with the initial seed set (*i.e.*, $D_0 = 500$ and $|\mathcal{T}(D_0)| = 1$), we use *Algorithm* 1 to synthesize the training data set and train the substitute classifiers in DNN accordingly. When $\mathcal{T}$ is architected with LR, DT and SVM, we respectively obtain the classification accuracies of the substitute classifiers in DNN on the testing set as 76.40%, 82.55%, and 77.34%. Then, we craft adversarial samples with DeepFool, *i.e.*, the one performs worse in Fig. 3 for showing the worst-case scenario of our proposal's performance.

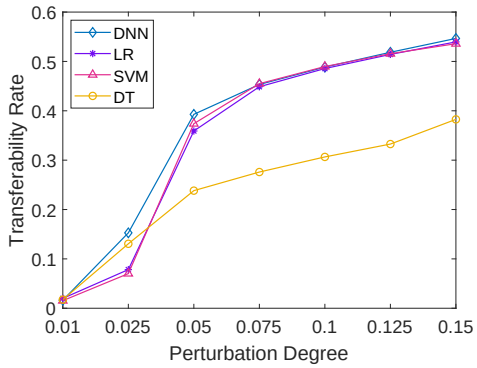


Fig. 4. Transferability rate of self-taught BB-AdA with DeepFool.

We plot the results on transferability rate in Fig. 4, which suggests that the target classifiers in different structures can all be severely misled by our self-taught BB-AdA, even though the substitute classifier is always built with DNN. Moreover, we can see that our proposal performs similarly in Fig. 4, except for the case in which $\mathcal{T}$ is architected with DT. This confirms the generalization of our proposed self-taught BB-AdA, *i.e.*, its attacking performance does depends much on the architecture of the target classifier. Specifically, if we limit the perturbation degree as 0.05, the transferability rate induced by our self-taught BB-AdA is still higher than 20% when the target classifier is in DT (*i.e.*, the worst case).

V. Conclusion

In this paper, we proposed a self-taught BB-AdA scheme targeting on the ML-based classifiers for anomaly detection in MLy-NA. Specifically, our proposal eavesdropped and tampered legitimate telemetry samples to craft and inject adversarial samples adaptively, for disturbing the operation of ML-based classifiers on anomaly detection and in turn misleading MLy-NA to make incorrect NC&M decisions. We designed novel algorithms to synthesize and label training data for the substitute classifier used by the BB-AdA. With the telemetry data collected from a real-world packet-over-optical network testbed, we demonstrated that with our self-taught BB-AdA scheme, an attacker could interact with an MLy-NA system quietly and efficiently to train itself adaptively, generate well-crafted adversarial samples to mislead the target classifiers in different ML structures to severely affect their performance on anomaly detection, and disturb the operation of MLy-NA in the hard-to-detect manner. Hence, our self-taught BB-AdA could launch effective attacks without much pre-knowledge about the target MLy-NA, justifying its capability of self-teaching.

Acknowledgments

This work was supported by NSFC project 61871357 and Fundamental Fund for Central Universities (WK3500000006).

References

[1] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[2] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[3] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[4] W. Shi, Z. Zhu, M. Zhang, and N. Ansari, "On the effect of bandwidth fragmentation on blocking probability in elastic optical networks," *IEEE Trans. Commun.*, vol. 61, pp. 2970–2978, Jul. 2013.

[5] P. Marsch *et al.*, "5G radio access network architecture: Design guidelines and key considerations," *IEEE Commun. Mag.*, vol. 54, pp. 24–32, Nov. 2016.

[6] M. Zhang *et al.*, "Bandwidth defragmentation in dynamic elastic optical networks with minimum traffic disruptions," in *Proc. of ICC 2013*, pp. 3894–3898, Jun. 2013.

[7] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[8] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.

[9] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *Proc. of GLOBECOM 2016*, pp. 1–6, Dec. 2016.

[10] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[11] S. Tang, J. Kong, B. Niu, and Z. Zhu, "Programmable multilayer INT: An enabler for AI-assisted network automation," *IEEE Commun. Mag.*, vol. 58, pp. 26–32, Jan. 2020.

[12] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 12–20, Mar./Apr. 2017.

[13] X. Chen *et al.*, "DeepRMSA: A deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks," *J. Lightw. Technol.*, vol. 37, pp. 4155–4163, Aug. 2019.

[14] B. Niu *et al.*, "Visualize your IP-over-optical network in realtime: A P4-based flexible multilayer in-band network telemetry (ML-INT) system," *IEEE Access*, vol. 7, pp. 82 413–82 423, Aug. 2019.

[15] S. Liu *et al.*, "Highly-efficient and automatic spectrum inspection based on AutoEncoder and semi-supervised learning for anomaly detection in EONs," *J. Lightw. Technol.*, vol. 39, pp. 1243–1254, Mar. 2021.

[16] M. Brinkmann *et al.*, "ALPACA: Application layer protocol confusion-analyzing and mitigating cracks in TLS authentication," in *Proc. of SSYM 2021*, pp. 4293–4310, Aug. 2021.

[17] N. Papernot *et al.*, "The limitations of deep learning in adversarial settings," in *Proc. of Euro S&P 2016*, pp. 372–387, Mar. 2016.

[18] M. Wang, H. Lu, S. Liu, and Z. Zhu, "How to mislead AI-assisted network automation in SD-IPoEONs: A comparison study of DRL- and GAN-based approaches," *J. Lightw. Technol.*, vol. 38, pp. 5574–5585, Oct. 2020.

[19] X. Pan, H. Yang, Z. Xu, and Z. Zhu, "Adversarial analysis of ML-based anomaly detection in multi-layer network automation," *submitted to J. Lightw. Technol.*, 2022.

[20] X. Chen *et al.*, "Self-taught anomaly detection with hybrid unsupervised/supervised machine learning in optical networks," *J. Lightw. Technol.*, vol. 37, pp. 1742–1749, Apr. 2019.

[21] C. Barber, D. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, Dec. 1996.

[22] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[24] I. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," *arXiv:1412.6572*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6572.

[25] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. of CVPR 2016*, pp. 2574–2582, Jun. 2016.

[26] X. Pan *et al.*, "Privacy-preserving multilayer in-band network telemetry and data analytics: For safety, please do not report plaintext data," *J. Lightw. Technol.*, vol. 38, pp. 5855–5866, Nov. 2020.

[27] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.