Deep Reinforcement Adversarial Learning against Botnet Evasion Attacks

Giovanni Apruzzese*, Mauro Andreolini†, Mirco Marchetti‡, Andrea Venturi‡, Michele Colajanni§
*Hilti Chair of Data and Application Security – University of Liechtenstein, Vaduz, Liechtenstein
†Department of Physics, Computer Science and Mathematics – University of Modena, Italy
‡Department of Engineering "Enzo Ferrari" – University of Modena, Italy
§Department of Informatics, Science and Engineering – University of Bologna, Italy
{giovanni.apruzzese, mauro.andreolini, mirco.marchetti, andrea.venturi, michele.colajanni}@unimore.it

***Abstract*—As cybersecurity detectors increasingly rely on machine learning mechanisms, attacks to these defenses escalate as well. Supervised classifiers are prone to adversarial evasion, and existing countermeasures suffer from many limitations. Most solutions degrade performance in the absence of adversarial perturbations; they are unable to face novel attack variants; they are applicable only to specific machine learning algorithms. We propose the first framework that can protect botnet detectors from adversarial attacks through Deep Reinforcement Learning mechanisms. It automatically generates realistic attack samples that can evade detection, and it uses these samples to produce an augmented training set for producing hardened detectors. In such a way, we obtain more resilient detectors that can work even against unforeseen evasion attacks with the great merit of not penalizing their performance in the absence of specific attacks. We validate our proposal through an extensive experimental campaign that considers multiple machine learning algorithms and public datasets. The results highlight the improvements of the proposed solution over the state-of-the-art. Our method paves the way to novel and more robust cybersecurity detectors based on machine learning applied to network traffic analytics.**

***Index Terms*—Adversarial attack, Machine learning, Network Intrusion Detection, Deep Reinforcement Learning, Botnet**

I. Introduction

MACHINE Learning (ML) approaches are being increasingly applied to cybersecurity where data-driven detection algorithms outperform traditional signature-based methods against novel forms of attacks [1], [2]. The problem is that defensive systems have to deal with proactive enemies who are turning their attentions against modern ML detectors. The adopted classifiers are vulnerable to the so called *adversarial evasion attacks* aiming to thwart the ML model through specific malicious samples that can remain undetected (e.g., [3]–[5]). The robustness of cybersecurity detectors is a critical issue because few misclassifications can lead to severe consequences [6], [7]. Existing countermeasures are at an early stage and they suffer from several drawbacks. For example, they are effective only against predictable attack strategies or they can be applied only to specific ML algorithms. Moreover, their detection rate tends to degrade when the system is not subject to evasion strategies [2], [8]–[11].

In the context of adversarial evasion attacks against network intrusion detection systems (NIDS) based on ML, we propose a novel approach that leverages Deep Reinforcement Learning (DRL) to increase the robustness of detectors relying on network flow analyses. Our proposal allows an automatic generation of realistic adversarial samples that preserve their underlying malicious logic and can evade detection with high probability. The detector is hardened by means of an adversarial training procedure based on automatically generated samples [10], [12]. To the best of our knowledge, this paper represents the first proposal that exploits deep reinforcement learning for the purpose of hardening botnet detectors through adversarial training. In our research we consider the real constraints that characterize the cybersecurity domain. They include the necessity of creating adversarial samples through small and feasible modifications, but also the implication that the attacker has limited queries to evade detection. Moreover, adversarial training requires accurate analyses because it may even decrease detection performance in the absence of adversarial attacks (e.g., [13]).

The implementation of the proposed approach produces a framework that can be used to attack state-of-the-art botnet detectors and to defend them against known and novel evasion strategies. With respect to existing works, our framework protects the detectors against unforeseen evasion attempts without compromising the detection rate in the absence of adversarial attacks. Our proposal is applicable to botnet detectors relying on

different machine learning classifiers. Its effectiveness is demonstrated in realistic scenarios represented by multiple datasets of enterprise network flows. An extensive experimental campaign shows the benefits of our method against previous literature in several terms. The samples generated by our method increase the resilience of botnet classifiers against existing and novel evasion attacks through adversarial training. The improved detectors maintain their performance even in the absence of adversarial attacks. By varying the amount of malicious samples to include in the augmented training dataset, we also show that our autonomous solution increases the detection rate by requiring less samples than approaches entailed by manual adversarial training. In the best cases, by adding just 1% of adversarial samples to the training set, we are able to increase the detection rate by nearly 25%.

The remainder of this paper is structured as follows. Section II motivates our paper and describes the threat model. Section III presents the proposed method. Section IV details the experimental settings for the evaluation. Section V discusses the results of the experiments. Section VI compares our paper against related work. Section VII concludes the paper with some final remarks and future research directions.

II. Motivation

The threats posed by botnets are increasing (e.g., [6], [7], [14]) and the difficulties of their detection represent a real menace for modern organizations. Botnet detection is the topic of a large body of literature where traditional methods based on full-packet captures are replaced by recent solutions relying on both supervised and unsupervised ML approaches [2], [9] that analyze network flows [15]–[19]. Unfortunately, the growing popularity of these botnet detectors is arousing the interest of adversaries that plan and produce new evasion attacks [20]. As a consequence, it is important to devise novel defensive approaches that can improve the detector robustness against these evasion attempts.

Adversarial attacks to ML-based detectors aim to generate specific samples that induce the model to produce an incorrect output [8]. These adversarial samples can be introduced during the training phase (so called *poisoning* attacks [21]) or at inference time [22] that is of interest for this paper. Within this latter category, we consider *evasion* attempts where the goal of the attacker is to induce a misclassification of malicious samples. Many papers tackle this issue in image and speech processing domains (e.g., [23]–[27]). Surprisingly, the cybersecurity context is new although it has inherently to deal against intentional attackers. Most studies focus on spam and malware analysis [8], [28]–[30], while there are few results on network intrusion detection which represents the focus of this paper. Attacks against detectors were investigated by the same authors and by other researchers in [4], [5], [10], [31], [32]. Most of these papers highlight that even small adversarial perturbations against machine learning detectors can significantly reduce their detection rates, while a careless insertion of malicious samples may favor detection.

In Figure 1 we outline the typical scenario consisting of an enterprise network with many internal hosts, where at least one machine has been compromised by a botnet malware communicating with a Command and Control system (CnC). The network traffic is inspected by a NIDS that analyzes network flows to identify botnet activities by means of supervised machine learning methods. Our threat model represents a realistic gray-box attack where the adversary has partial information about the defensive system. We consider unrealistic to assume that the attacker knows the precise internal configuration of the model, the full set of features, and the complete training set of the detector as in other papers.

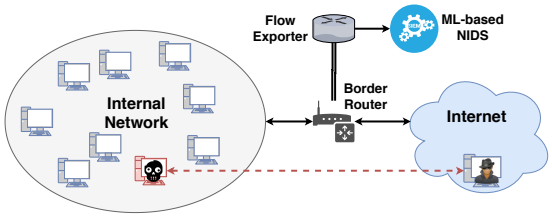


Figure 1: Considered threat scenario.

The attacker can realistically assume that a modern network is monitored by a ML-based NIDS that has been trained on network flows [15], [33]. They do not know the complete set of features, but they can expect that time- and data-related information are included because most flow-based detectors use them as reported by literature on botnet detectors (e.g., [4], [16], [17], [34]–[37]).

We consider detectors that analyze the most important features captured by network flows, reported in Table I, where the gray background denotes the features on which the attacker operates. They can issue commands to the infected machines through the Command and Control infrastructure. The attacker tries to evade detection by slightly modifying the botnet communications. These modifications alter the original flow characteristics and result in adversarial perturbations that may evade detection. For example, typical adversarial samples may present different *durations*, different amounts of exchanged *bytes* or transmitted *packets*. Modifying these metrics affects multiple attributes in Table I because

some of them are derived features. We consider these alterations because we are interested in modifications that do not compromise the underlying logic of the botnet. Similar perturbations can be easily obtained by inserting short communication delays, or adding random junk data in the transmitted packets. These operations require small changes to the source code of the botnet malware variant without compromising its logic [38]. Metamorphic malware operates in a similar way [39].

Table I: Traffic features of the considered detectors.

#	Feature name	Type
1,2	*Source/Dest IP address type*	Bool
3,4	*Source/Dest port*	Num
5	*Flow direction*	Bool
6	*Connection state*	Cat
7	*Duration (seconds)*	Num
8,9	*Source/Dest ToS*	Num
10,11	*Source/Dest bytes*	Num
12	*Total pkts*	Num
13	*Total bytes*	Num
14,15	*Source/Dest port type*	Cat
16	*Bytes per second*	Num
17	*Bytes per packet*	Num
18	*Packets per second*	Num
19	*Ratio of Source/Dest bytes*	Num

III. Proposed method

The proposed approach leverages Deep Reinforcement Learning to generate realistic adversarial samples that preserve their malicious logic and are able to evade detection. These samples are used as a mean for hardening the original detector through adversarial training [12]. The expectation is that the resulting botnet detector achieves better detection rates than its initial version. The method consists of three phases that are represented in Figure 2 and detailed below.

A. Preparation

The goal of the first phase is to create a DRL *agent* that is capable of autonomously generating evasive adversarial samples against botnet detectors on the basis of network flows belonging to some botnet b. As anticipated in Section II, modeling a realistic scenario requires to preserve the malware underlying logic. Moreover, the attackers are also constrained to the number of samples that they can submit to the detector to guess the underlying ML logic. This agent learns how to make the best decisions through an autonomous trial-and-error approach [40], [41] where at each agent's choice, which is chosen among the defined *Action Space*, corresponds a *Reward* provided by the *Environment*.

The Environment includes two elements: the *state generator* transforms the input flow sample in a format that is recognized by the agent; the *botnet detector* $\widehat{D}$ leverages a ML classifier trained on a dataset $\mathcal{T}$ of network flows containing legitimate and malicious samples including some pattern of the botnet b. Let $\widehat{D}(\mathcal{T})$ denote this detector.

The Reward depends on the output of the detector to the sample produced by the agent. A correct and an incorrect classification is associated to a positive and a null reward, respectively. The agent continues to modify the sample until it is able to evade detection or after a maximum amount of failures.

The Action Space includes the set of perturbations that an agent can introduce in a malicious sample. Our focus is on flow-based botnet detectors and our goal is to generate samples that an attacker can realistically reproduce. As described in Section II, we limit our Action Space to small increments of few essential traffic flow features (*duration*, *sent bytes*, *received bytes*, *transmitted packets*) and correlated features.

We consider agents based on two deep reinforcement learning algorithms that have been applied in cybersecurity (e.g., [40], [42], [43]): one is based on the off-policy *Double Deep Q-Network*; the other one is based on the on-policy *Deep State–action–reward–state–action*. We consider DRL approaches because they address complex tasks better than basic RL methods. These latter achieve poor performance when the problem requires to evaluate many possibilities in terms of feasible states and related action-space [44]. The function representing the value of an action can be seen as a table that maps all states and all actions to the expected long-term return. In our case, the dimension of this table is large and compiling it requires high computational costs. Approaches based on DRL leverage deep neural networks to estimate (instead of fully creating) the value function. In such a way, they avoid the two-dimensional representation of the value function and devise models with general capabilities that are able to achieve better results [45].

To the best of our knowledge, we are the first authors to consider the 2DQN and Sarsa algorithms to both evade and harden a botnet detector. Other authors in [4] consider Deep Q-Learning but only for offensive purposes.

***Double Deep Q-Network** (2DQN):* was proposed in [46], and it leverages the synergy of the original Double Q-Learning approach with Deep Q-Networks [46]. In these methods, the system uses the same values to both choose and evaluate the effects of an action, thus inducing some over-estimations. In our context, this approach tends to generate samples that deviate significantly from their initial variant. The aim of 2DQN is to decrease such over-estimations. The main intuition is to determine the best decision by splitting the optimization into action

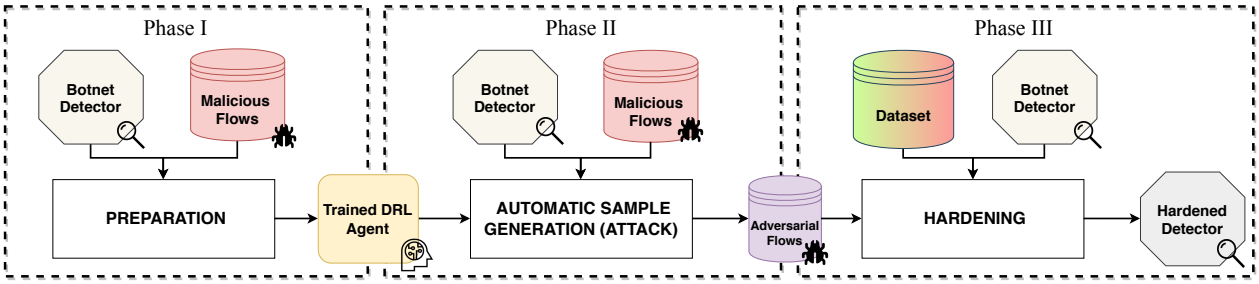


Figure 2: The three phases of the proposed solution.

selection and evaluation by relying on two deep neural networks. The 2DQN algorithm is characterized by the following formula [46]:

$$Y_t^{2DQN} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \theta_t^+); \theta_t^-) \quad (1)$$

where Y_t^{2DQN} is the target function at time t, and $\gamma \in [0, 1]$ is the discount factor used for the reward; R_{t+1} is the immediate reward and S_{t+1} is the resulting next state (at time t+1); θ^+ and θ^- denote the two different deep neural networks; $Q(\cdot, a; \theta_t)$ is the function that regulates the updating procedure at time t: in particular, a and θ are used to denote the vector of action values associated to the θ network. The target network θ^- is used to estimate the value selected by the online network θ^+. In summary, 2DQN adopts a greedy-policy selection method, which results in a training process with fast and low cost iterations. Our expectation is that a similar approach will generate agents that require few queries to evade detection.

Deep State-action-reward-state-action *(Sarsa):* is a DRL algorithm [47] that fosters a risk-adverse strategy. Unlike the greedy approach adopted by 2DQN, here the learner updates its parameters with the action determined by the policy. We consider the deep learning variant of *Sarsa* which relies on a deep neural network θ to estimate the action values. The algorithm is based on the following update equation [47]:

$$Y_t^{Sarsa} = R_{t+1} + \gamma Q(S_{t+1}, a_{t+1}; \theta) - Q(S_t, a_t; \theta) \quad (2)$$

where the same notation of Eq. 1 is adopted. The function that regulates the update procedure does not include the argmax operator because of the more conservative strategy characterizing *Sarsa*. Our expectation is that agents based on Sarsa will try to evade detection by preferring smaller modifications, at the expense of an increased number of iterations.

Our paper investigates the effectiveness of on-policy and off-policy methods, as they belong to complementary DRL paradigms. We report in Figure 3 the main differences that can be summarized as follows. On-policy techniques such as Sarsa adopt a linear approach. They learn to choose the best action by following and improving one policy, which for each state suggests a single action. On the other hand, off-policy techniques such as Q-learning (the precursor of 2DQN) use an exploratory policy that suggests multiple actions to play in each state. These actions are evaluated and chosen by a separate core policy, and the learning procedure aims to improve this second core policy [48]. Training on-policy methods requires more iterations but they are more robust than off-policy algorithms.

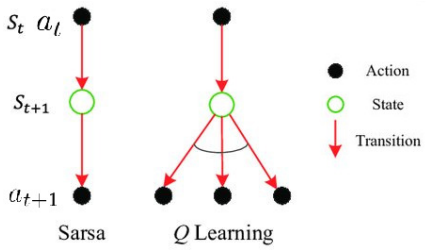


Figure 3: Sarsa and Q-Learning algorithms (source: [48]).

The final output of this first *Preparation* phase is the DRL agent $A(b)$ which is trained to generate adversarial samples on the basis of network flows belonging to the botnet b contained in $\mathcal{T}$.

B. Automatic sample generation (Attack)

In this phase, the trained DRL agent $A(b)$ is used to produce samples that are able to evade a botnet detector $D(\mathcal{T})$ similarly to an adversarial attack scenario. It is important to observe that the targeted detector can even be different from the one adopted for training the agent [21]. The methodology is outlined in Figure 4. The system accepts a malicious flow of botnet b as its input which is sent to the state generator (step ❶) and then forwarded to the agent $A(b)$ (step ❷). After that,

the agent communicates the best action to modify the sample to the state generator (step ❸) that applies the modification and issues the sample to the detector $D(\mathcal{T})$ (step ❹). If the evasion is successful, then the modified sample is saved in a dedicated dataset of adversarial flows (step ❺). Otherwise, the process is re-activated and the sample is further modified until it is able to evade $D(\mathcal{T})$ or until a maximum amount of attempts $\mathcal{Q}_{max}$ is reached. The agent $A(b)$ does not receive any reward in this phase because it has already been trained.

As an example, assume a malicious flow $\mathsf{f} \in b$ in input: the system may opt to increase the *duration* of f thus obtaining the sample f', and submits it to the detector $D(\mathcal{T})$. If the evasion is successful, then f' is added to the dataset of the adversarial samples. Otherwise, the agent further modifies f' by increasing another (or the same) feature and repeats the submission process.

The output of this procedure is a set of adversarial flows $G_D^A(b)$ that are perturbed versions of the botnet b flows that is, they have been altered by the agent $A(b)$ in order to evade the detector $D(\mathcal{T})$. The flows contained in $G_D^A(b)$ can then be used to harden the detector.

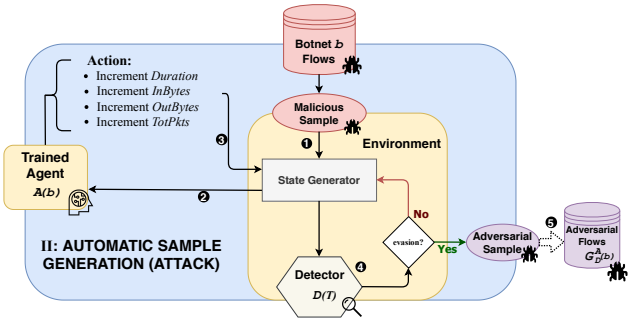


Figure 4: Automatic generation of adversarial samples.

C. Hardening

The final phase leverages the adversarial training paradigm to harden the botnet detector. This goal is achieved by re-training the detector $D(\mathcal{T})$ through an augmented training set $G_D^A(b)$ that includes the adversarial samples generated during the previous phase. A threshold Ψ specifies the percentage of adversarial samples that are introduced in the initial training dataset to generate the augmented dataset. This latter may contain also some additional benign flows to maintain a realistic ratio of legitimate-to-illegitimate samples. We are the first to evaluate the sensitivity of the detector as a function of different Ψ percentages in Section V.

At the end, this phase yields a hardened version of the detector $D(\mathcal{T})$ that is trained on the adversarial samples $G_D^A(b)$ and that we denote as $\overline{D}^A(\mathcal{T})$.

IV. Experimental Settings

We now describe and motivate the experimental environment used for our evaluation. We start by presenting the testbed and the specifics of the target detectors, and then detail the configuration of the proposed framework. Finally, we report the characteristics of the considered adversarial attacks.

A. Datasets

We consider two datasets in the experimental campaign: the `CTU` [49] and the `BOTNET` [50], that contain labelled collections of millions of network flows generated by benign and malicious devices in networks of hundreds of hosts representing modern enterprises. The heterogeneous environments captured by these datasets are appreciated by related literature in botnet detection [4], [34], [35], [51]. The malicious flows in these datasets belong to different botnet families. As recommended by the state-of-the-art [4], [5], [8], [52], [53] we use ensembles of classifiers, in which each classifier is devoted to a specific botnet variant.

We report in Table II some samples of network flows that are included in these datasets with the corresponding features. For the sake of readability, we omit the derived features.

B. Detectors

The botnet detectors are based on two famous machine learning classifiers: *Random Forest* (RF) and *Wide and Deep* (WnD). RF is recognized in literature as one of the most proficient techniques for cyber detection [4], [12], [53], [54].

WnD is a deep learning-based approach proposed by Google [55] that, to the best of our knowledge, has never been evaluated for botnet detection. We consider it due to its appreciable results in other classification contexts [56]. Both algorithms adopt the feature set reported in Table I.

The training procedure of each classifier follows the best practices in related literature [17], [31]: 80% of samples are used for training and 20% for testing; the samples are distributed in a legitimate-to-illegitimate ratio of 20:1. We report the amount of samples used for training and testing in Table III, where five famous botnet families (included in the `CTU` and `BOTNET` datasets) are considered. We exclude from the evaluation the botnets with too few samples in the datasets because they yield under-performing detectors. Each detector consists of an ensemble of five classifiers, each trained on a specific botnet family (see Section IV-A).

Table II: Example of network flows included in the datasets.

S_IP	*D_IP*	*S_Port*	*D_Port*	*Dir*	*State*	*Dur*	*S_ToS*	*D_ToS*	*S_Bytes*	*D_Bytes*	*T_Pkts*	*S_Port_t*	*D_Port_t*
int	ext	43458	80	mo	RA	0.09	0	0	264	0	4	high	known
ext	int	22	10005	bi	CON	192.62	0	0	4968	1413	40	known	high

Table III: Datasets used for training and testing. Total samples: CTU: 3 431 629; BOTNET: 189 352.

Dataset	Family	Overall	Training set		Testing set	
		Malicious	Malicious	Benign	Malicious	Benign
CTU	Neris	80 097	60 071	1 201 455	20 026	400 485
	Rbot	27 509	20 631	412 635	6 878	137 545
	Virut	32 347	24 260	485 205	8 087	161 735
	Menti	2 825	2 118	42 375	707	14 125
	Murlo	1 106	829	16 590	277	5 530
BOTNET	Neris	3 685	2 763	55 275	922	38 425
	Rbot	3 685	2 763	55 275	922	38 425
	Virut	878	658	13 170	220	4 390
	Menti	3 685	2 763	55 275	922	38 425
	Murlo	3 685	2 763	55 275	922	38 425

Combining the two network scenarios (CTU and BOTNET) and the two detectors (RF and WnD), we have four combinations. For example, the detector RF(CTU) consists of an ensemble of 5 random forest classifiers where each is trained on a botnet family of the CTU dataset.

C. Framework

The training of the agents is based on RF because it outperforms other classifiers [5], [51], [54], [57]. When an agent wants to modify a sample, it selects a feature and then increments it by a chosen amount. An important choice of our approach is to generate novel adversarial samples that are realistic and remain consistent with traffic features. For these reasons, we limit the increment of each selected feature to at most two units. For example, an agent can increase the *duration* feature by 1 or 2 seconds or can modify the *Total Pkts* by increasing it of 1 or 2 packets.

Unlike attacks occurring in the *problem space* (e.g., [28]), our agent generates adversarial samples by directly modifying the malicious network flows. As the perturbations are applied directly to the features, the modified samples must not contain values causing inconsistencies [58]–[60]. We address this issue by instructing the agent to check and update all the features whenever a manipulation is performed. (For example, an increase to the *Source bytes* determines an increment of the *Bytes per second*). As the magnitude of the modification is small (at most $+2$ units), we do not need to update the *Duration* because it is realistic and feasible to transmit more data through packets or bytes in the same timeframe.

At the end of the training process, we obtain four agents that are denoted by the corresponding reinforcement learning algorithm. For instance, Sarsa(CTU) is the agent relying on the algorithm Sarsa that is trained to evade the RF(CTU) detector. In the remainder we omit the considered network because all experiments consider agents and detectors that operate on the same network.

The trained agents generate evasive samples against all detectors. The goal is to show that our trained agents are not only effective at attacking the same detector used in the preparation phase, but also against different detectors. For instance, we use 2DQN to attack both RF and WnD by considering three values of $\mathcal{Q}_{max}$=$(1, 5, 80)$. The first two values are used to emulate a realistic attack scenario, compliant with the considered threat model, where the adversary cannot see the output of the detector ($\mathcal{Q}_{max}$=1) or can leverage a limited number of queries ($\mathcal{Q}_{max}$=5). The last value is used for hardening purposes because the defenders can freely access and query the detector. In this attack phase, the agents modify every available malicious sample into an evasive network flow. For each value of $\mathcal{Q}_{max}$, the agents generate two sets of adversarial flows for every detector. For example, we obtain the sets G_{RF}^{2DQN} and G_{RF}^{Sarsa} for the RF detector.

We use the sets of adversarial samples that are produced by our agents to create augmented training sets that improve the robustness of the detectors against evasion attacks. The effects of adversarial training are studied as a function of different amounts of injected samples. Existing proposals on adversarial training simply inject a fixed amount of samples and proceed to measure the results (e.g., [12], [61]). We adopt a more realistic approach that is necessary for devising cybersecurity solutions in real scenarios. For this reason, we consider to inject different percentages of injected malicious samples Ψ=$(1\%, 5\%, 10\%, 20\%, 100\%)$, where Ψ=100% is the baseline as in related literature [61]. For hardening we use the adversarial datasets generated with $\mathcal{Q}_{max}$=1000 which contains the exact amount of samples of the original malicious datasets. We perform the re-training and subsequent re-testing by following the same ratios and splits used for the baseline detectors. After this phase we have eight hardened detectors for every value of Ψ where, for example, the detector RF^{2DQN} is the variant of RF that is hardened through the samples contained in G_{RF}^{2DQN}.

D. Implementation of attacks

Solutions leveraging adversarial training are evaluated against the attack scenarios that resemble the samples used for producing the augmented dataset [8], [12], [21], [61]. These defense strategies are effective if the attacks can be foreseen that is, by manually crafting samples that replicate the predicted evasion attempts. Our proposal aims to overcome a similar limitation and shows that our autonomous approach allows the creation of resilient detectors that are less affected by novel perturbations. In order to achieve a comprehensive evaluation, we consider three gray box evasion attack scenarios that are compatible with the threat model in Section II and are described below.

- $\mathbb{E}1$ represents the attacks performed by our framework. This scenario assumes a powerful attacker that is able to query the detector several times without the risk of triggering other defensive mechanisms. We use this scenario to show that our framework is effective even against stronger but less realistic attackers that can see the response of the detector to a malicious flow, by using the machine learning model as an oracle [26]. We evaluate the detectors against attackers that can modify each sample up to five times. In practice, we submit each sample in G_{RF}^{2DQN} (at $\mathcal{Q}_{max} = 5$) to each hardened detector in the same network setting.
- $\mathbb{E}2$ represents an attack strategy that was shown to effectively evade detection [5]. Here, the detector is evaded by manually modifying combinations of up to four features (*duration*, *source bytes*, *destination bytes*, *total packets*) that are altered by fixed amounts. We manually craft adversarial samples from the original malicious flows that mimic a similar attack pattern. Each flow has the values of the considered features that are increased by five amounts that is, $(+1, +2, +5, +10, +30)$. Experiments show that these perturbations produce adversarial flows that are able to evade the considered detectors with high probability. We consider $\mathbb{E}2$ as the basis for producing the set of manually crafted adversarial flows that are denoted by G^{Man}.
- $\mathbb{E}3$ represents attacks that alter the same features of $\mathbb{E}2$ but with intermediate increments. For example, if the samples in $\mathbb{E}2$ increase the duration of the original malicious flows by $(1, 2, 5, 10, 30)$, then the samples in $\mathbb{E}3$ increase the duration by $(1.5, 3.5, 7.5, 20)$. The adversarial samples in $\mathbb{E}3$ represent unforeseen attacks, and the toughest testbed for our approach.

The rationale for considering these three attack scenarios is as follows. As the augmented datasets for adversarial training include samples that mimic the corresponding attack pattern, we can expect that detectors hardened through our framework can achieve better results when the attack types are in $\mathbb{E}1$, and a manual approach should yield better results in the $\mathbb{E}2$ scenario. On the other hand, the adversarial samples in $\mathbb{E}3$ represent unforeseen attacks, hence it is of maximum importance that our hardening approach is effective even in this scenario. The experimental evaluation aims to show that the proposed solution is able to mitigate all these attack scenarios.

V. Evaluation

The experimental campaign has the twofold objective of showing that our framework produces samples that are able to evade detectors with high success rate and few queries, and that the generated samples can be used to harden the detectors against evasion attacks without decreasing their performance in non-adversarial settings. The framework has been implemented in Python3 with the *scikit-learn*, *Keras-RL* and *OpenAI Gym* toolkits. For evaluating the performance in non-adversarial settings we adopt the usual metrics of machine learning studies: *Precision*, *Detection Rate* (DR, or *Recall*), and *F1-score* [51], [53]; on the other hand, for the attack scenarios we consider the Detection Rate of the adversarially manipulated samples.

A. Baseline performance

We initially evaluate the baseline detectors RF and WnD for the two network scenarios in non-adversarial contexts. The results in Table IV show that our baseline detectors achieve values that are comparable to the state of the art [5], [36], [51]. RF slightly outperforms WnD as anticipated by previous studies (e.g., [5], [51], [54]). We observe that the WnD detector in the `BOTNET` scenario obtains poor Precision in the case of the Virut and Menti families. As these deep learning classifiers are affected by high rates of false positives, signaling that these novel methods still present margin for improvements. However, these classifiers reach suitable Recall values against these botnet families (above 0.95) thus implying that they are able to detect most malicious samples, and hence represent a valid baseline for the experimental campaign.

We also present in Figures 5 the ranking of the top-5 most significant features for every baseline detector. Each detector has minor different rankings, but all of them have among the most important features those that the proposed DRL agent will modify to construct the adversarial samples. The only exception is the *destination port_type* of the RF(`CTU`) detector. We also highlight that

Table IV: Performance of the baseline detectors RF and WnD in non-adversarial settings.

Detector		Random Forest (RF)			Wide & Deep (WnD)		
Network	Family	*F1-score*	*Recall*	*Precision*	*F1-score*	*Recall*	*Precision*
CTU	Neris	0.985	0.988	0.983	0.845	0.978	0.743
	Rbot	0.996	0.993	0.998	0.988	0.983	0.992
	Virut	0.987	0.998	0.976	0.934	0.991	0.883
	Menti	0.999	0.998	0.999	0.923	0.960	0.929
	Murlo	0.990	0.986	0.994	0.927	0.973	0.878
	average (*std. dev.*)	0.991 (0.005)	0.993 (0.003)	0.991 (0.008)	0.935 (0.047)	0.980 (0.006)	0.898 (0.082)
BOTNET	Neris	0.995	0.996	0.994	0.916	0.914	0.919
	Rbot	0.999	1.000	0.999	0.986	0.998	0.975
	Virut	0.993	0.992	0.994	0.358	0.951	0.220
	Menti	1.000	1.000	1.000	0.806	0.975	0.686
	Murlo	0.999	0.999	1.000	0.994	0.999	0.989
	average (*std. dev.*)	0.998 (0.002)	0.998 (0.001)	0.998 (0.002)	0.902 (0.135)	0.970 (0.034)	0.864 (0.179)

these rankings are similar to those obtained by related researches on botnet detection through supervised ML algorithms [34]–[37]. For example, in [34] the *Tot_bytes* feature is the second most important feature, as in RF(CTU). The detectors in [36] and in [35] focus on the *Tot_Pkts* feature as in WnD(BOTNET). These results suggest that adversarial attacks aiming to perturb these features are more likely to be effective.

B. Evasion

We evaluate the offensive capabilities of our proposal by using the trained agents to launch evasion attacks against the target detectors. The performance is measured through the *Evasion Rate* (ER). The results are reported in Tables V, where the cells show the ER and *average amount of queries* $\mathcal{Q}_{avg}$ (in parentheses) of the samples generated by our agents with $\mathcal{Q}_{max}$=80. Gray cells report the weighted average across all botnet families of each network. From these tables, we observe that the agents produced by our framework are able to evade the RF detector with very high probability, and they are also effective against the WnD detector where ER exceeds 90%. We also note that, as anticipated in Section III-A, an agent based on 2DQN requires less attempts to evade detection compared to an agent based on Sarsa.

We compare the automatic evasion capabilities of the proposed solution against an attack proposed by the same authors [5], where the adversarial samples were generated by manually increasing the same features by fixed amounts. Those attacks obtained an average evasion rate of 35% on the CTU dataset, and of 40% on the BOTNET dataset, while our evasion rates are above 90% (Tables V).

In realistic scenarios, attackers cannot arbitrarily query the detector or inspect its output without exposing themselves. For this reason, we consider the amount of generated samples that are able to evade detection on the first attempt and in less than five attempts. From the results in Tables VI, we can observe that many samples are capable of evading the detection mechanism immediately (e.g., over 80% and 50% against RF(CTU) and WnD(BOTNET), respectively). Moreover, for the Sarsa agent, the majority requires few attempts in both networks. These results confirm that detectors based on machine learning are highly vulnerable to adversarial attacks.

We compare the offensive capabilities of our proposal against other automatic methods for evading ML detectors [4], [13], [40], [41], [61]–[66]. In Table VII we report the best results for each proposal. Our framework can leverage agents that achieve a high success rate through few queries, while other approaches generate evasive samples that are either less effective or require many more attempts.

Our proposal is also superior if we limit the comparison to the botnet detection scenarios discussed in [4] that consider only the CTU dataset. Successful evasion is reached in 41% of the cases with $\mathcal{Q}_{avg}$=4, while our 2DQN agent achieves an evasion rate of 99% with $\mathcal{Q}_{avg}$=2.4 (Table Va).

The fact that even small and easily achievable alterations affect the most important features of the considered detection models is a critical issue that demands proper countermeasures. The effectiveness of the proposed attacks is likely due to the high importance that the modified features have for the baseline detectors. Perturbations affecting these features are likely to produce samples that evade detection. A similar observation was in [50]. Other studies on evasion attacks [4] do not consider the feature importance of the baseline detectors.

C. Hardened defense

We now evaluate the effects of adversarial training based on the samples generated by our agents. We report the results of the hardened version of RF and WnD that are obtained through re-training on datasets including the generated malicious samples G^{2DQN} and G^{Sarsa}. Let us denote the hardened versions through $\overline{\text{RF}}$ and $\overline{\text{WnD}}$. We initially consider non-adversarial settings where Ψ is set to 100% (see Section IV-C). The results are shown in Tables VIII, which report the average and standard deviation of the metrics for each botnet family and network scenario, and also include the performance of the baseline detectors in the bottom rows (taken from Table IV). By comparing the results of the hardened versions against those of their corresponding baselines, we can appreciate that our method does not degrade performance in the absence of evasion attacks. This is an important improvement with respect to the state of the art.

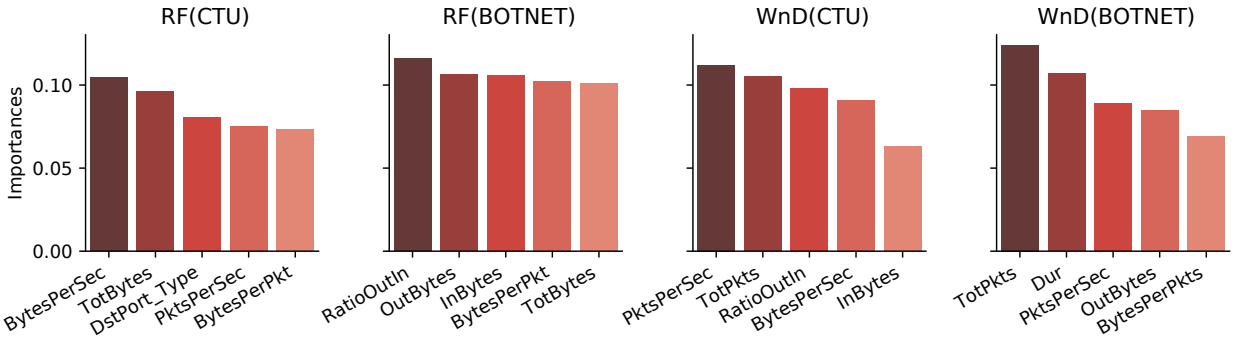


Figure 5: Top-5 important features for the baseline RF and WnD detectors on each network scenario.

Table V: Attack performance of the 2DQN and Sarsa agents for $\mathcal{Q}_{max} = 80$.

Network	CTU						BOTNET					
Malware	Neris	Rbot	Virut	Menti	Murlo	*average*	Neris	Rbot	Virut	Menti	Murlo	*average*
2DQN	97% (4.72)	99% (1.30)	99% (2.43)	100% (1.59)	100% (2.15)	99% (2.43)	95% (13.93)	99% (11.45)	92% (9.32)	99% (8.68)	99% (14.20)	96% (11.51)
Sarsa	97% (4.00)	99% (1.57)	96% (4.06)	99% (2.43)	100% (2.07)	98% (2.82)	90% (20.16)	95% (10.12)	98% (13.09)	100% (6.99)	92% (10.33)	95% (12.13)

(a) *Evasion Rate* (and $\mathcal{Q}_{avg}$) against the baseline RF detectors in both network scenarios.

Network	CTU						BOTNET					
Malware	Neris	Rbot	Virut	Menti	Murlo	*average*	Neris	Rbot	Virut	Menti	Murlo	*average*
2DQN	82%(21.27)	96%(11.42)	80%(25.15)	100%(1.07)	100%(23.50)	91%(16.48)	96%(6.52)	100%(1.66)	98%(2.37)	99%(9.78)	100%(4.55)	98%(4.97)
Sarsa	92%(8.07)	95%(12.42)	70%(30.14)	99%(16.21)	100%(30.78)	91%(19.52)	99%(2.50)	99%(1.81)	98%(2.60)	100%(1.47)	72%(25.68)	93%(6.81)

(b) *Evasion Rate* (and $\mathcal{Q}_{avg}$) against the baseline WnD detectors in both network scenarios.

Table VI: Attack performance of the 2DQN and Sarsa agents for $\mathcal{Q}_{max} = 1$ and $\mathcal{Q}_{max} = 5$.

Network		CTU						BOTNET					
Family		Neris	Rbot	Virut	Menti	Murlo	*average*	Neris	Rbot	Virut	Menti	Murlo	*average*
2DQN	$\mathcal{Q}_{max}$=1	73.3%	86.2%	93.3%	58.6%	7.9%	80.4%	11.7%	2.1%	43.7%	23.5%	15.1%	15.1%
	$\mathcal{Q}_{max}$=5	89.7%	99.7%	96.6%	98.1%	99.3%	93.4%	37.3%	18.7%	72.4%	33.8%	64.9%	40.5%
Sarsa	$\mathcal{Q}_{max}$=1	78.6%	86.1%	89.9%	31.2%	6.7%	81.2%	10.9%	3.2%	41.4%	9.2%	15.9%	12.3%
	$\mathcal{Q}_{max}$=5	91.8%	98.1%	92.7%	93.8%	100%	93.3%	20.9%	57.8%	57.8%	85.5%	67.7%	57.9%

(a) *Evasion Rate* against the baseline RF detectors in both network scenarios.

Network		CTU						BOTNET					
Family		Neris	Rbot	Virut	Menti	Murlo	*average*	Neris	Rbot	Virut	Menti	Murlo	*average*
2DQN	$\mathcal{Q}_{max}$=1	25.9%	2.9%	11.8%	93.4%	12.3%	19.5%	80.8%	86.1%	94.2%	6.7%	43.3%	56.4%
	$\mathcal{Q}_{max}$=5	32.8%	13.1%	11.8%	100%	12.3%	25.4%	86.7%	93.4%	97.5%	85.2%	89.2%	90.3%
Sarsa	$\mathcal{Q}_{max}$=1	85.5%	2.5%	11.1%	8.1%	12.3%	50.2%	90.8%	83.9%	94.6%	90.9%	15.1%	71.5%
	$\mathcal{Q}_{max}$=5	89.7%	12.7%	11.4%	16.3%	15.1%	55.4%	95.5%	98.6%	97.2%	94.4%	64.9%	88.7%

(b) *Evasion Rate* against the baseline WnD detectors in both network scenarios.

Table VII: Comparison of ER and Q_{avg} of our proposal with related approaches.

Framework	ER	Q_{avg}
[4]	41%	4
[13]	95%	12
[40]	16%	✗
[41]	46%	✗
[62]	79%	> 200
[63]	15%	✗
[64]	100%	✗
[61]	52%	✗
[65]	100%	> 100
[66]	100%	$> 40\,000$
Ours	97%	9

We now evaluate the efficacy of our proposal at countering the three considered attack scenarios. We also compare the effects of training performed on the basis of the generated samples with samples manually crafted to replicate the patterns of existing attacks. Moreover, we carry out a sensitivity analysis by varying the percentages of adversarial samples (Ψ parameter) introduced in the training dataset by evaluating the Detection Rate.

We initially analyze the defensive capabilities against the attacks scenario $\mathbb{E}1$. The results are reported in Tables IX, where cells show the average Detection Rate for increasing values of Ψ. The cells with a gray background refer to mechanisms hardened through our proposal, and numbers in bold denote the results obtained by the best approach for the corresponding value of Ψ.

Table VIII: Performance of the hardened detectors in non-adversarial settings.

Network		CTU			BOTNET		
Metric		*F1-score (std. dev.)*	*Recall (std. dev.)*	*Precision (std. dev.)*	*F1-score (std. dev.)*	*Recall (std. dev.)*	*Precision (std. dev.)*
Hardening	2DQN	0.988 (0.009)	0.993 (0.006)	0.984 (0.017)	0.995 (0.005)	0.998 (0.002)	0.992 (0.008)
	Sarsa	0.989 (0.010)	0.993 (0.005)	0.985 (0.017)	0.996 (0.003)	0.999 (0.001)	0.994 (0.006)
Baseline		0.991 (0.005)	0.993 (0.003)	0.991 (0.008)	0.998 (0.002)	0.998 (0.001)	0.998 (0.002)

(a) Results of the hardened $\overline{\text{RF}}$ detectors.

Network		CTU			BOTNET		
Metric		*F1-score (std. dev.)*	*Recall (std. dev.)*	*Precision (std. dev.)*	*F1-score (std. dev.)*	*Recall (std. dev.)*	*Precision (std. dev.)*
Hardening	2DQN	0.928 (0.062)	0.967 (0.030)	0.901 (0.112)	0.896 (0.085)	0.968 (0.045)	0.845 (0.129)
	Sarsa	0.903 (0.086)	0.985 (0.007)	0.848 (0.149)	0.918 (0.069)	0.969 (0.040)	0.883 (0.118)
Baseline		0.935 (0.047)	0.980 (0.006)	0.898 (0.082)	0.902 (0.135)	0.970 (0.034)	0.864 (0.179)

(b) Results of the hardened $\overline{\text{WnD}}$ detectors.

The Detection Rate achieved by the baseline detectors is reported in the caption of each table, which corresponds to setting $\Psi = 0\%$. From Tables IX, we can appreciate that the detectors hardened through our methods significantly improve the capabilities of the baseline detectors. Moreover, they always outperform the results of those manually trained.

We then test the hardened detectors against $\mathbb{E}2$ that represents the manually crafted attacks, and report the results in Tables X. We observe that all proposed approaches improve the baseline detection rate against existing attacks and that for high values of Ψ the manual approach tends to be more effective because these samples exactly match those used to attack the detectors. Our method, however, requires to inject a significantly smaller amount of samples than the manual approach as shown by the column Ψ=1% in Table Xa for the BOTNET network, and in Table Xb for the CTU network. These results show that, to be effective, training through human-crafted samples requires not only to predict all the attack patterns that can be used to evade the detectors, but also the necessity to craft a significant amount of samples with high manual effort.

Finally, we show that our framework is also effective in protecting the detectors against previously unforeseen attacks in scenario $\mathbb{E}3$. The results in Tables XI show that in the case of the hardened RF detector in the BOTNET network scenario (Table XIa) our proposal outperforms the manual approach for the majority of the considered percentages Ψ. The same conclusion is valid for the hardened WnD^{2DQN} in the CTU network (Table XIb). We can observe that hardening a detector through manually crafted samples is effective only for a very large number of injected samples in the order of Ψ=100%. This means that all malicious samples should be generated with a considerable effort.

In Table XII we compare the results of the proposed approach against those of existing defensive mechanisms to evasion attacks. In this table, we report the performance of the best classifier as reported by each paper before and after the hardening procedure in adversarial and non-adversarial scenarios. (The authors in [67] use a custom metric denoted as *resistance*.) We note that some approaches [13], [61], [68] do not evaluate the performance of the hardened detector in non-adversarial settings. Addressing this lack is a contribution of this paper. The method in [69] is used only to enhance the detector but it does not consider adversarial scenarios. Other approaches (denoted with a gray background in Table XII) are affected by significant performance degradation in the absence of adversarial attacks [29], [67], [70]–[72]. This problem does not affect our proposal and the approach presented in [12]. Nevertheless, its initial performance in non-adversarial settings is poor (*F1-score* of only 0.69), and the improvement in adversarial scenarios is considerably smaller than the results achieved by our method (2% against 30%).

VI. Related work

The literature has demonstrated that even small adversarial perturbations can impact severely the performance of detectors based on machine learning models, but the solutions are still at an early stage [2], [3], [9], [11], [24]. Existing countermeasures conform to either the *security-by-design* or the *security-by-obscurity* paradigms [8]. Here, we focus on the former group because of the unreliability of security-by-obscurity defensive strategies [73]. Security-by-design strategies against evasion attacks can be divided into three groups [10]: *feature manipulation*, *defensive distillation*, and *adversarial training*.

Several studies have shown that approaches leveraging altered feature sets may be effective at mitigating [8], [67] or even nullifying [11] attacks that involve the manipulation of the involved features. However, training the model on different sets of features may cause significant performance degradation in the absence of adversarial attacks [12], [72]. The same drawback also affects countermeasures based on defensive distillation. As evidenced in [29], [70], these approaches tend to increase the false positive rate. Furthermore, they are tailored to algorithms based on neural networks that are not the best choice in network intrusion detection [2], [4], [12], [53], [54].

Adversarial training aims to harden the detector through an augmented dataset containing samples with

Table IX: Performance against $\mathbb{E}1$ of the hardened detectors for varying Ψ values.

Network	**CTU**					**BOTNET**				
Ψ	1%	5%	10%	20%	100%	1%	5%	10%	20%	100%
2DQN	**0.948**	**0.973**	**0.991**	**0.978**	**0.965**	**0.845**	**0.971**	**0.995**	**0.984**	**0.964**
Sarsa	0.919	0.954	0.966	0.959	0.953	0.679	0.834	0.965	0.967	0.944
Man	0.523	0.772	0.779	0.958	0.604	0.309	0.761	0.978	0.956	0.807

(a) Average *DR* of the hardened $\overline{\text{RF}}$ detectors against $\mathbb{E}1$. Baseline *DR* on $\mathbb{E}1$: RF(CTU)=0.065, RF(BOTNET)=0.591.

Network	**CTU**					**BOTNET**				
Ψ	1%	5%	10%	20%	100%	1%	5%	10%	20%	100%
2DQN	**0.901**	0.906	**0.908**	**0.924**	**0.933**	**0.786**	**0.813**	0.919	**0.921**	**0.924**
Sarsa	0.877	**0.908**	0.853	0.907	0.911	0.581	0.657	**0.921**	0.824	0.631
Man	0.718	0.717	0.760	0.877	0.729	0.329	0.812	0.903	0.812	0.830

(b) Average *DR* of the hardened $\overline{\text{WnD}}$ detectors against $\mathbb{E}1$. Baseline *DR* on $\mathbb{E}1$: WnD(CTU)=0.456, WnD(BOTNET)=0.097.

Table X: Performance against $\mathbb{E}2$ of the hardened detectors for varying Ψ values.

Network	**CTU**					**BOTNET**				
Ψ	1%	5%	10%	20%	100%	1%	5%	10%	20%	100%
2DQN	**0.681**	**0.689**	0.684	0.692	0.712	**0.858**	**0.886**	**0.898**	**0.906**	**0.914**
Sarsa	0.657	0.680	**0.689**	**0.738**	0.710	0.848	0.878	0.893	0.901	0.908
Man	0.466	0.589	0.681	0.709	**0.789**	0.677	0.784	0.821	0.859	0.884

(a) Average *DR* of the hardened $\overline{\text{RF}}$ detectors against $\mathbb{E}2$. Baseline *DR* on $\mathbb{E}2$: RF(CTU)=0.327, RF(BOTNET)=0.679.

Network	**CTU**					**BOTNET**				
Ψ	1%	5%	10%	20%	100%	1%	5%	10%	20%	100%
2DQN	**0.748**	**0.753**	**0.755**	**0.756**	0.801	0.475	0.570	0.564	0.601	0.761
Sarsa	0.704	0.682	0.694	0.703	0.853	**0.531**	0.532	0.567	0.582	0.681
Man	0.587	0.656	0.674	0.717	**0.937**	0.457	**0.623**	**0.672**	**0.712**	**0.881**

(b) Average *DR* of the hardened $\overline{\text{WnD}}$ detectors against $\mathbb{E}2$. Baseline *DR* on $\mathbb{E}2$: WnD(CTU)=0.514, WnD(BOTNET)=0.413.

Table XI: Performance against $\mathbb{E}3$ of the hardened detectors for varying Ψ values.

Network	**CTU**					**BOTNET**				
Ψ	1%	5%	10%	20%	100%	1%	5%	10%	20%	100%
2DQN	**0.717**	0.674	0.658	0.671	0.677	**0.731**	**0.766**	**0.784**	0.791	**0.851**
Sarsa	0.713	**0.686**	**0.669**	0.671	0.679	0.714	0.754	0.775	0.788	0.834
Man	0.396	0.545	0.660	**0.718**	**0.765**	0.552	0.705	0.756	**0.807**	0.391

(a) Average *DR* of the hardened $\overline{\text{RF}}$ detectors against $\mathbb{E}3$. Baseline *DR* on $\mathbb{E}3$: RF(CTU)=0.253, RF(BOTNET)=0.526.

Network	**CTU**					**BOTNET**				
Ψ	1%	5%	10%	20%	100%	1%	5%	10%	20%	100%
2DQN	**0.728**	**0.738**	**0.741**	**0.742**	0.805	0.461	0.543	0.544	0.566	0.719
Sarsa	0.664	0.656	0.672	0.676	0.822	**0.503**	0.504	0.531	0.551	0.648
Man	0.549	0.650	0.671	0.719	**0.941**	0.422	**0.598**	**0.652**	**0.698**	**0.876**

(b) Average *DR* of the hardened $\overline{\text{WnD}}$ detectors against $\mathbb{E}3$. Baseline *DR* on $\mathbb{E}3$: WnD(CTU)=0.496, WnD(BOTNET)=0.408.

adversarial perturbations. This approach comes with two challenges: obtaining appropriate adversarial samples and planning the re-training operations. It may be possible to manually craft samples that reflect realistic attacks, but similar methods are time consuming; they can only protect against predictable attacks that comply to the generated samples. Although adversarial training works even in non-adversarial contexts, the results in [13] suggest the need of studying the effects of the augmentation process. To the best of our knowledge, we are the first to evaluate the impact of adversarial training by varying the amounts of injected samples in adversarial and non-adversarial botnet detection scenarios.

Reinforcement learning is often associated with adversarial machine learning in different contexts and goals. These mechanisms can be a target of adversarial attacks [74], but also as a means to conceive attacks [4], [13], [40], [41], [62]–[64], and as a countermeasure to

Table XII: Performance comparison against evasion attacks.

Framework	Non-adversarial settings		Adversarial settings	
	Initial	*Hardened*	*Initial*	*Hardened*
[4]	*Acc*: 0.99	✗	*DR*: 0.59	✗
[13]	*F1*: 0.89	✗	*DR*: 0.56	*DR*: 0.84
[41]	*AUC*: 0.96	✗	*DR*: 0.54	✗
[62]	*Acc*: 0.88	✗	*DR*: 0.31	✗
[63]	*AUC*: 0.96	✗	*DR*: 0.85	✗
[64]	*Acc*: 0.99	✗	*Acc*: 0.15	✗
[61]	*AUC*: 0.97	✗	*DR*: 0.68	*DR*: 0.70
[29]	*Acc*: 0.98	*Acc*: 0.94	*DR*: 0.35	*DR*: 0.61
[67]	*Acc*: 0.96	*Acc*: 0.93	*Res*: 0.36	*Res*: 0.62
[11]	*F1*: 0.96	*F1*: 0.82	*DR*: 0.34	*DR*: 0.61
[12]	*F1*: 0.69	*F1*: 0.71	*F1*: 0.63	*F1*: 0.65
[70]	*Acc*: 0.88	*Acc*: 0.78	*DR*: 0.02	*DR*: 0.03
[72]	*Acc*: 0.96	*Acc*: 0.93	*Acc*: 0.70	*Acc*: 0.77
[68]	*Acc*: 0.99	✗	*Acc*: 0.68	*Acc*: 0.84
[69]	*F1*: 68.5	*Acc*: 70.5	✗	✗
[71]	*Acc*: 91.9	*Acc*: 90.8	*DR*: 0.03	*DR*: 0.31
Ours	*F1*: 0.99	*F1*: 0.99	*DR*: 0.58	*DR*: 0.88

these threats through adversarial training [13], [40], [61].

Few papers consider cyber security problems related to network intrusion detection which is the focus of our proposal. The authors in [33] and [69], [75] propose methods to generate flows for training NIDS, but they do not evaluate their performance in adversarial scenarios. The method in [13] based on reinforcement learning tends to degrade the baseline performance of the detector. The papers in [40], [63], [65] consider reinforcement learning agents that operate on binary malware detectors, while we operate on network traffic. The proposal in [61] focuses on hardening detectors of Domain Generation Algorithms but does not consider the performance in non-adversarial scenarios. The approach in [4] does not evaluate adversarial training and operates on packet captures, while we focus on network flows that nowadays are preferred by modern detectors [15].

The primary focus of most proposals related to the generation of attack samples is just on the rate of successful evasions [26], [41], [61], [64] and not on the number of required queries issued to the target detector. Neglecting this characteristic is unrealistic because attackers can only perform limited amounts of queries if they want to avoid detection. For example, the methods presented in [62] and [66] allow their agents to submit hundreds or even thousands of queries. Even the proposal in [4], [65] achieves evasion through dozens of attempts against the target detector. Unlike these papers, we propose the first framework based on deep reinforcement learning that hardens existing flow botnet detectors in realistic scenarios, even against attackers that are capable of issuing some queries to the considered detectors.

Other papers on adversarial attacks leverage Generative Adversarial Networks (GANs). Such approaches require two components: a generator, aiming to generate realistic samples from the original data; and a discriminator, which decides whether the output of the generator appears similar to legitimate traffic [13]. GANs are similar to DRL approaches, but these latter have the advantage of controlling more precisely the sample generating procedure. In practice, DRL methods allow to define the detailed action space that is used by the model to create the samples [43]. On the other hand, in GANs the generator network does not consent such a fine-grained control. We consider DRL to be more suitable to simulate realistic attacks that involve precise and small modifications because excessive or improper modifications may trigger detection of defensive mechanisms.

VII. Conclusions

Modern detectors based on machine learning classifiers are increasingly able at identifying malicious network traffic, but they can be exploited by adversarial samples that allow attackers to evade detection. Existing solutions are affected by several drawbacks that do not guarantee a reliable defense. We address the issue of evasion attacks against flow-based botnet detectors by proposing the first defensive approach that relies on deep reinforcement learning to mitigate adversarial perturbations against network intrusion detection systems based on machine learning. We consider the characteristics of a realistic cybersecurity scenario: small and feasible perturbations to the input samples; high degree of evasion with a limited number of queries; assessments of several configurations for defensive purposes. The implementation of our proposal results in a framework that autonomously generates evasive samples against a target botnet detector, and then uses these samples for hardening the detector through adversarial training. An extensive experimental campaign replicating realistic network scenarios of modern organizations shows the quality of our proposal over state-of-the-art methods for multiple reasons. It increases the detection rate against known and novel evasion attacks; it does not degrade performance in non-adversarial settings; the procedure of malicious sample generation can bypass detection through few queries issued to the detector. Our study may pave the way to future researches aiming to face evasion attacks by devising robust detectors that preserve their performance regardless of the presence of adversarial perturbations.

REFERENCES

[1] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1285–1298.

[2] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2016.

[3] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Sok: Security and privacy in machine learning," in *Proc. IEEE Europ. Symp. Secur. Privacy*, Apr. 2018, pp. 399–414.

[4] D. Wu, B. Fang, J. Wang, Q. Liu, and X. Cui, "Evading machine learning botnet detection models via deep reinforcement learning," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[5] G. Apruzzese, M. Colajanni, and M. Marchetti, "Evaluating the effectiveness of adversarial attacks against botnet detectors," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, Oct. 2019, pp. 1–8.

[6] H. Kettani and P. Wainwright, "On the top threats to cyber systems," in *Proc. IEEE Int. Conf. Inf. Comp. Tech.*, Mar. 2019, pp. 175–179.

[7] G. Banga, "Why is cybersecurity not a human-scale problem anymore?" *Commun. ACM*, vol. 63, no. 4, p. 30–34, Mar. 2020.

[8] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Elsevier Pattern Recogn.*, vol. 84, pp. 317–331, 2018.

[9] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware C&C detection: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, p. 59, 2016.

[10] N. Martins, J. M. Cruz, T. Cruz, and P. H. Abreu, "Adversarial machine learning applied to intrusion and malware scenarios: a systematic review," *IEEE Access*, 2020.

[11] G. Apruzzese, M. Colajanni, L. Ferretti, and M. Marchetti, "Addressing adversarial attacks against security systems based on machine learning," in *Proc. IEEE Int. Conf. Cyber Conflicts*, May 2019, pp. 1–18.

[12] S. Calzavara, C. Lucchese, and G. Tolomei, "Adversarial training of gradient-boosted decision trees," in *Proc. ACM Int. Conf. Inf. Knowledge Manag*, 2019, pp. 2429–2432.

[13] M. Usama, M. Asim, S. Latif, J. Qadir *et al.*, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *Proc. Int. IEEE Conf. Wireless Commun. Mobile Comput.*, 2019, pp. 78–83.

[14] "Checkpoint 2020 security report," https://pages.checkpoint.com/cyber-security-report-2020.html, Accessed in March 2020.

[15] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Elsevier Computers & Security*, vol. 70, pp. 238–254, 2017.

[16] A. Pektaş and T. Acarman, "Deep learning to detect botnet via network flow summaries," *Springer Neural Comput. Appl.*, pp. 1–13, 2018.

[17] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *Proc. IEEE Int. Conf. Comput., Netw. and Commun.*, Feb. 2014, pp. 797–801.

[18] S. Nõmm and H. Bahşi, "Unsupervised anomaly based botnet detection in iot networks," in *Proc. IEEE Int. Conf. Machin. Learn. Appl.*, 2018, pp. 1048–1053.

[19] S. Lagraa, J. François, A. Lahmadi, M. Miner, C. Hammerschmidt, and R. State, "Botgm: Unsupervised graph mining to detect botnets in traffic flows," in *Proc. IEEE Conf. Cyber Secur. Netw.*, 2017, pp. 1–8.

[20] Z. Qiu, D. J. Miller, and G. Kesidis, "Flow based botnet detection through semi-supervised active learning," in *Proc. IEEE Int. Conf. Acoustics, Speech, Sign. Process.*, 2017, pp. 2387–2391.

[21] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli, "Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks," in *Proc. USENIX Secur. Symp.)*, 2019, pp. 321–338.

[22] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint Europ. Conf. Mach. Learn. and Knowl. Discov. Databases*. Springer, Sept. 2013, pp. 387–402.

[23] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, 2019.

[24] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Europ. Symp. Secur. Privacy*, Mar. 2016, pp. 372–387.

[25] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *Proc. IEEE Secur. Privacy Workshops*. IEEE, 2018, pp. 1–7.

[26] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. ACM Workshop Artif. Intel. Secur.*, 2017, pp. 15–26.

[27] D. Jakubovitz and R. Giryes, "Improving DNN robustness to adversarial attacks using jacobian regularization," in *Proc. Europ. Conf. Comp. Vision*, 2018, pp. 514–529.

[28] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ml attacks in the problem space," in *IEEE Symp. Secur. Privacy*, 2020.

[29] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *Proc. Springer Europ. Sympo. Res. Comput. Secur.*, 2017, pp. 62–79.

[30] P. Laskov *et al.*, "Practical evasion of a learning-based classifier: A case study," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 197–211.

[31] G. Apruzzese and M. Colajanni, "Evading botnet detectors based on flows and random forest with adversarial samples," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, Oct. 2018, pp. 1–8.

[32] D. J. Miller, Z. Xiang, and G. Kesidis, "Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks," *Proc. IEEE*, vol. 108, pp. 402–433, 2020.

[33] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, 2019.

[34] Z. M. Algelal, E. A. Ghanialdhaher, D. N. Abdul-Wadood *et al.*, "Botnet detection using ensemble classifiers of network flow," *IAES Int. J. Electr. Comput. Eng.*, vol. 10, no. 3, p. 2543, 2020.

[35] I. Letteri, G. Della Penna, and P. Caianiello, "Feature selection strategies for http botnet traffic detection," in *Proc. IEEE Europ. Symp. Secur. Priv.*, 2019, pp. 202–210.

[36] B. Abraham, A. Mandya, R. Bapat, F. Alali, D. E. Brown, and M. Veeraraghavan, "A comparison of machine learning approaches to detect botnet traffic," in *Proc. IEEE Int. Conf. Neur. Netw.*, 2018, pp. 1–8.

[37] M. Alauthaman, N. Aslam, L. Zhang, R. Alasem, and M. A. Hossain, "A p2p botnet detection scheme based on decision tree and adaptive multilayer neural networks," *Springer Neural Computing and Applications*, vol. 29, no. 11, pp. 991–1004, 2018.

[38] C. Xiang, F. Binxing, Y. Lihua, L. Xiaoyi, and Z. Tianning, "Andbot: towards advanced mobile botnets," in *Proc. USENIX Conf. Large-scale Exploits and Emergent Threats*, 2011, pp. 11–11.

[39] S. Cesare, Y. Xiang, and W. Zhou, "Malwise—an effective and efficient classification system for packed and polymorphic malware," *IEEE T. Comput.*, vol. 62, no. 6, pp. 1193–1206, 2012.

[40] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to evade static pe machine learning malware models via reinforcement learning," *arXiv:1801.08917*, 2018.

[41] Z. Fang, J. Wang, B. Li, S. Wu, Y. Zhou, and H. Huang, "Evading anti-malware engines with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 48 867–48 879, 2019.

[42] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *arXiv:1906.05799*, 2019.

[43] K. Malialis and D. Kudenko, "Distributed response to network intrusions using multiagent reinforcement learning," *Elsevier Eng. Appl. Artif. Int.*, vol. 41, pp. 270–284, 2015.

[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[45] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Int. Conf. Machin. Learn.*, 2016, pp. 1928–1937.

[46] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Int.*, 2016, pp. 2094–2100.

[47] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on sarsa," *IEEE Access*, vol. 8, pp. 54 074–54 084, 2020.

[48] H. Jiang, R. Gui, Z. Chen, L. Wu, J. Dang, and J. Zhou, "An improved sarsa reinforcement learning algorithm for wireless communication systems," *IEEE Access*, vol. 7, pp. 115 418–115 427, 2019.

[49] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Elsevier Comput. Secur.*, vol. 45, pp. 100–123, 2014.

[50] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. Comm. Netw. Secur.*, 10 2014.

[51] M. Stevanovic and J. M. Pedersen, "An analysis of network traffic classification for botnet detection," in *Proc. IEEE Int. Conf. Cyber Situat. Awar., Data Analyt., Assessment*, Jun. 2015, pp. 1–8.

[52] B. Biggio, I. Corona, Z.-M. He, P. P. Chan, G. Giacinto, D. S. Yeung, and F. Roli, "One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time," in *Proc. Springer Int. Workshop Multiple Classifier Syst.*, 2015, pp. 168–180.

[53] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cybersecurity," in *Proc. IEEE Int. Conf. Cyber Conflicts*, May 2018, pp. 371–390.

[54] O. Fajana, G. Owenson, and M. Cocea, "Torbot stalker: Detecting tor botnets through intelligent circuit data analysis," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, Oct. 2018, pp. 1–8.

[55] A. Karasaridis, B. Rexroad, D. A. Hoeflin *et al.*, "Wide-scale botnet detection and characterization." *HotBots*, vol. 7, pp. 7–7, 2007.

[56] Z. Li, Z. Qin, and P. Shen, "Intrusion detection via wide and deep model," in *Proc. Springer Int. Conf. Artif. Neural Netw.*, 2019, pp. 717–730.

[57] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *Proc. IEEE Int. Symp. Intel. Syst. Inf.*, 2017, pp. 277–282.

[58] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin, "Practical traffic-space adversarial attacks on learning-based nidss," *arXiv preprint arXiv:2005.07519*, 2020.

[59] A. Chernikova and A. Oprea, "Fence: Feasible evasion attacks on neural networks in constrained environments," *arXiv preprint arXiv:1909.10480*, 2020.

[60] S. Sen, E. Aydogan, and I. A. Aysan, "Coevolution of mobile malware and anti-malware," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2563–2574, 2018.

[61] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop Artif. Intell. Secur.*, Oct. 2016, pp. 13–21.

[62] Y. Senzaki, S. Ohata, and K. Matsuura, "Simple black-box adversarial examples generation with very few queries," *IEICE Transactions on Information and Systems*, vol. 103, no. 2, pp. 212–221, 2020.

[63] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, "Evading machine learning malware detection," *Black Hat*, 2017.

[64] J. Zhang, Q. Yan, and M. Wang, "Evasion attacks based on wasserstein generative adversarial network," in *Proc. IEEE Conf. Comput., Commun. and IoT Appl.* IEEE, 2019, pp. 454–459.

[65] H. Dang, Y. Huang, and E.-C. Chang, "Evading classifiers by morphing in the dark," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 119–133.

[66] Y. Ren, Q. Zhou, Z. Wang, T. Wu, G. Wu, and K.-K. R. Choo, "Query-efficient label-only attacks against black-box machine learning models," *Elsevier Computers & Security*, vol. 90, p. 101698, 2020.

[67] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 1145–1153.

[68] Y. Ji, B. Bowman, and H. H. Huang, "Securing malware cognitive systems against adversarial attacks," in *Proc. IEEE Int. Conf. Cognitive Comput.*, 2019, pp. 1–9.

[69] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *Proc. IEEE Int. Conf. Artif. Int. and Big Data*, 2018, pp. 228–234.

[70] M. Soll, T. Hinz, S. Magg, and S. Wermter, "Evaluating defensive distillation for defending text processing neural networks against adversarial examples," in *Proc. Springer Int. Conf. Artif. Neur. Netw.*, 2019, pp. 685–696.

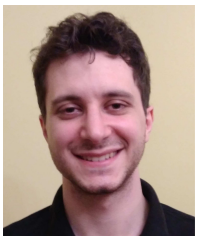
[71] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *Proc. IEEE Secur. Privacy Workshops*, 2018, pp. 76–82.

[72] S. Calzavara, C. Lucchese, F. Marcuzzi, and S. Orlando, "Feature partitioning for robust tree ensembles and their certification in adversarial scenarios," *arXiv:2004.03295*, 2020.

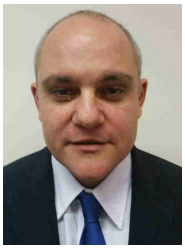
[73] D. Pavlovic, "Gaming security by obscurity," in *Proc. ACM New Secur. Paradigms Workshop*. ACM, 2011, pp. 125–140.

[74] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *Proc. Springer Int. Conf. Machin. Learn. Data Mining Pattern Recogn.*, 2017, pp. 262–275.

[75] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "Enhancing network intrusion detection classifiers using supervised adversarial training," *Springer Journal of Supercomputing*, pp. 1–30, 2019.

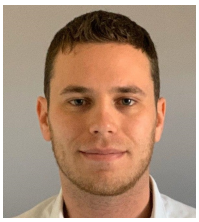


Giovanni Apruzzese is a Post-Doctoral researcher within the Hilti Chair of Data and Application Security at the University of Liechtenstein since 2020. He received the PhD Degree and the Master's Degree in Computer Engineering (summa cum laude) in 2020 and 2016 respectively at the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Italy. In 2019 he spent 6 months as a Visiting Researcher at Dartmouth College (Hanover, NH, USA) under the supervision of Prof. VS Subrahmanian. His research interests involve all aspects of big data security analytics with a focus on machine learning, and his main expertise lies in the analysis of Network Intrusions, Phishing, and Adversarial Attacks.



Mauro Andreolini is currently an Assistant Professor at the Department of Physics, Computer Science and Mathematics of the University of Modena and Reggio Emilia, Italy. He received his Master Degree (summa cum laude) at the University of Roma, Tor Vergata in January, 2001 and his PhD in May, 2005 from the same institution. His research focuses on design, evaluation and security of distributed and cloud-based systems, malware analysis and secure software design.

Mirco Marchetti received the Ph.D. degree in Information and Communication Technologies in 2009. He is currently an Associate Professor with the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Italy. His research interests include all aspects of system and network security, security for cyber physical systems, automotive security, cryptography applied to cloud security, and outsourced data and services.



Andrea Venturi is a PhD student at the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Italy. From the same institution, he received the Master's Degree in Computer Engineering *summa cum laude* in 2020 with a thesis on cybersecurity analytics, and the Bachelor's Degree in Computer Science in 2017, with a thesis on data analysis for scalable and distributed networked systems. His research interests are on machine and deep learning applications for cybersecurity.

Michele Colajanni is Full Professor in computer engineering at the University of Bologna. He received the Master degree from the University of Pisa, and the Ph.D. degree from the University of Rome. He was researcher at the University of Rome, and visiting researcher at the IBM Research Center, Yorktown Heights in 1996. From 1998 to 2020, he was with the Department of Engineering "nzo Ferrari" at the University of Modena and Reggio Emilia. He founded the Interdepartment Research Center on Security and Safety (CRIS), and the Cyber Academy on cybersecurity training. His research interests include cybersecurity, performance and prediction models, cloud systems.