This is the peer reviewd version of the followng article:

Hardening Random Forest Cyber Detectors Against Adversarial Attacks / Apruzzese, G., Andreolini, M., Colajanni, M., Marchetti, M.. - In: IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE. - ISSN 2471-285X. - 4:4(2020), pp. 427-439. [10.1109/TETCI.2019.2961157]

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

10/10/2026 12:49

(Article begins on next page)

Hardening Random Forest Cyber Detectors Against Adversarial Attacks

Giovanni Apruzzese, Mauro Andreolini, Michele Colajanni, Mirco Marchetti
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Modena, Italy
{giovanni.apruzzese, mauro.andreolini, michele.colajanni, mirco.marchetti}@unimore.it

***Abstract*—Machine learning algorithms are effective in several applications, but they are not as much successful when applied to intrusion detection in cyber security. Due to the high sensitivity to their training data, cyber detectors based on machine learning are vulnerable to targeted adversarial attacks that involve the perturbation of initial samples. Existing defenses assume unrealistic scenarios; their results are underwhelming in non-adversarial settings; or they can be applied only to machine learning algorithms that perform poorly for cyber security. We present an original methodology for countering adversarial perturbations targeting intrusion detection systems based on random forests. As a practical application, we integrate the proposed defense method in a cyber detector analyzing network traffic. The experimental results on millions of labelled network flows show that the new detector has a twofold value: it outperforms state-of-the-art detectors that are subject to adversarial attacks; it exhibits robust results both in adversarial and non-adversarial scenarios.**

***Index Terms*—Adversarial samples, machine learning, random forest, intrusion detection, flow inspection, botnet**

I. Introduction

THE adoption of machine learning to support security operators is an inevitable trend because of the continuous increment of network traffic and sophistication of the attacks [1]–[3]. Machine learning algorithms are employed with success in an increasing number of areas including image processing, speech and text recognition, social media marketing [4] and, more recently, in cyber security. Indeed, modern Network Intrusion Detection Systems (NIDS) are being increasingly enriched with machine learning (e.g., [1], [5], [6]) and deep learning algorithms (e.g., [6]–[8]). Even some commercial products (e.g. Darktrace or Dragonfly Threat Sensor) integrate detectors based on machine learning. Despite these positive achievements, recent literature (e.g., [9]–[12]) highlights that existing machine learning techniques are vulnerable to the so called *adversarial attacks*. These malicious actions involve the production of samples designed to thwart the machine learning algorithm by inducing outputs favorable to the attacker. Similar vulnerabilities are critical in the cyber security domain because any undetected attack may compromise an entire organization. The problem of adversarial attacks against machine learning detectors is a relevant open issue.

We propose a novel approach for hardening cyber detectors based on machine learning. We focus on the random forest algorithm due to its proven effectiveness for intrusion detection [13]–[18]; however, recent studies also highlight its vulnerability to adversarial perturbations [19]–[21]. Our solution is based on the observation that existing machine learning cyber detectors rely on excessively rigid classification criteria: they are typically trained through *class labels* that separate samples in disjointed categories where each sample may be either malicious or benign. A similar approach cannot work in the cyber domain where each sample may present more vague attributes. For this reason, we leverage the idea of introducing some degree of flexibility in the training data set by using *probability labels*. The intuition is that a model that uses probability labels instead of hard class labels can be more resilient to adversarial perturbations, and can achieve comparable or even superior results even in the absence of attacks. Our methodology has several applications in all fuzzy scenarios characterizing cyber security that involve classifiers based on random forests. As a first test case, in this paper we adopt it for devising botnet detectors based on network flows analyzers.

We validate our approach through a large set of experiments, performed on a set of publicly available and labelled traffic traces containing over 20 million network flows with benign and malicious samples of different malware families. These data sets capture the network behavior of medium-large enterprises and represent an appropriate setting for a realistic evaluation. The experimental results demonstrate that the proposed solution de-

vises a detector with comparable or superior performance than state-of-the-art methods in scenarios that are not subject to adversarial attacks. Moreover, it significantly improves the robustness of random forest models against adversarial attacks. Achieving both results is a fundamental success for real contexts where we cannot anticipate whether a machine learning detector will be subject or not to adversarial attacks. Our promising results have room for further improvements, but we are confident that this paper represents a first important step towards more robust cyber defensive platforms based on machine learning against adversarial attacks.

The remainder of this paper is structured as follows. Section II introduces adversarial attacks and compares our paper against related work. Section III describes the proposed method. Section IV illustrates the scenario and the threat model considered in this paper. Section V presents the methodology and testbeds used for performance evaluation. Section VI discusses the experimental results. Section VII concludes the paper with some final remarks and possible extensions of this work.

II. Related work

The complexity of network attacks and the augment of daily traffic requires security operators to rely on some machine learning support [1], [2]. These methods may detect anomalies and may even reveal attack variants that are not recognizable through signature-based approaches [5], [22]. However, the success of novel defensive methods also induce the formulation of new offensive strategies. Today, the so called *adversarial attacks* represent a major limitation to the adoption of a fully autonomous cyber defence platform. We describe the main characteristics of adversarial attacks, and then compare our proposal with the state-of-the-art.

Adversarial attacks are based on the generation of specific samples that induce a machine learning model to produce an output that is favorable to the attacker. This result is caused by the intrinsic sensitivity of machine learning models to their internal configuration settings [1], [23], [24]. Early examples of adversarial attacks against spam filtering are proposed in [25]–[27]. These papers show that linear classifiers could be tricked by few carefully crafted changes in the text of spam emails without affecting the readability of the spam message. Another interesting example of adversarial attack against neural networks classifiers for image processing is presented in [28], where imperceptible perturbations to images used in the training phase can modify arbitrarily the model's output. Adversarial attacks can be classified through the taxonomy inspired by [29] that considers the following two properties.

Influence determines whether an attack is performed at training-time or test-time.

- *Training-time*: these attacks, also known as poisoning attacks, manipulate the training dataset through the insertion or removal of specific samples, therefore altering the decisions of the trained model.
- *Test-time*: these attacks subvert the behavior of the detector through the injection of specific samples during its operational phase.

Violation denotes the type of security violation, which can affect the availability or integrity of the system.

- *Integrity*: often referred to as evasion attacks, the goal is increasing the false negative rate of the model by introducing malicious samples that are classified as benign.
- *Availability*: these attacks tend to cause overwhelming spikes of false alarms, inducing temporary shutdowns and/or recalibrations of the detector.

There is extensive literature on adversarial perturbations against image processing (e.g., [9], [11], [12], [30]), while few papers consider adversarial attacks from a cyber security perspective (e.g., [18], [19], [21], [29], [31]). Several recent results demonstrate that adversarial attacks can represent a dangerous threat to any defensive system based on machine learning. For example, [10] and [32] consider the case of adversarial samples against PDF malware detectors based on Support Vector Machines (SVM), neural networks, and random forests. Other papers [21], [33], [34] highlight the problem of adversarial evasion for Android malware and spam detectors. Furthermore, the capability of a Generative Adversarial Network to thwart a Domain Generation Algorithm detector based on random forests is evaluated in [35]. More recently, [20] shows the fragility of a flow-based botnet detector relying on random forest against small adversarial perturbations. Although the threats posed by adversarial inputs are clear, the few existing solutions are not immediately applicable to real contexts. For example, [35] proposes to harden the classifier through multiple re-training steps based on adversarial samples. This is an interesting theoretic solution with practical limitations because it requires the creation and continuous management of datasets with realistic adversarial samples. Moreover, [31] suggests to improve the robustness against evasion attacks by not considering the features that can be manipulated by an attacker. The problem of this approach is that it reduces accuracy in normal scenarios as shown in [18], [36]. On the other hand, our proposal is immediately applicable to real contexts as demonstrated by multiple experimental settings.

Defensive distillation may work in mitigating adversarial perturbations against image classification [37], but this technique is built and evaluated only on neural network algorithms [38]. Although cyber detectors based on this algorithm exist and can be hardened through the original distillation proposal [39], in cybersecurity scenarios detectors based on random forests outperform those relying on neural networks and other supervised methods [13], [15]–[18], [40]. More recently, [41] evaluates different classifiers for the specific problem of botnet detection and confirms that random forest yields the best results. Finally, [42] proposes a NIDS that inspects network flows through a random forest classifier to identify botnets and obtains outstanding results with detection rates close to 0.99. For this reason, we devise an original formulation of the distillation technique that is specifically aimed at hardening random forest detectors, thus allowing to devise robust defensive schemes for cyber detection based on machine learning. Although a recent work [43] shows that it is possible to evade the defensive distillation, we observe that the considered threat model is unrealistic because it assumes an attacker with complete control of the detector: with similar privileges, attackers can (and most likely will) adopt measures much more invasive and disruptive than those based on adversarial perturbations. Other works on defenses against adversarial samples [44], [45] consider just SVM classifiers applied to malware analysis, which is out of the scope of this paper. We are not aware of other defensive mechanisms against evasion adversarial attacks that are applicable to random forest algorithms for network intrusion detection. Hence, we can conclude that the topic considered in this paper is a promising research theme, which we address through a novel approach that hardens random forest-based detectors through an original defensive distillation method.

III. Proposed method

We propose a novel method that hardens machine learning detectors based on random forest against adversarial attacks. The idea comes from the observation that the excessively rigid classification criteria learned by machine learning algorithms in the training phase are vulnerable to subtle adversarial perturbations. Indeed, existing detectors are trained through class labels that separate samples in disjointed categories where each sample may be either malicious or benign but not both. On the other hand, the cyber domain is more fuzzy, and a sample may present characteristics belonging to different categories. Any rigid classification produced by *hard class labels* may represent an exploitable weakness of cyber detectors in adversarial settings. For this reason, we aim to introduce some degree of flexibility and uncertainty in the training process by using *probability labels* that allow the algorithm to capture additional information between classes such as similarity. The intuition is that a model that uses probability labels instead of hard class labels can be more resilient to adversarial samples, and can achieve comparable or superior results even in the absence of attacks. The main difficulty of a similar approach is that probability labels are not readily available in the cyber domain; hence we devise an original solution built upon the two following phases:

1) generation of probability labels from hard class labels;
2) deployment of a supervised model trained with the generated probability labels to perform the cyber detection.

Fig. 1 shows that this approach considers as its input a *dataset* and its *class labels*. Then, it computes the corresponding *probability labels* (represented in the leftmost box), and uses them to train a *supervised model* that will be integrated in the detector. We apply this method to the random forest machine learning algorithm by leveraging the foundations [46] of the defensive distillation for neural networks [37]. By using the information encoded in the probability labels in the form of probability vectors, generated after training an initial model, it is possible to develop a second "distilled" model that is more robust against adversarial attacks. The entire workflow applied to the random forest algorithm is illustrated in Fig. 2 where each step is denoted by a circled number that is explained in the following subsections. Unlike the original defensive distillation technique, the generation of probability labels and their use for detection is performed through random forest-based models instead of neural networks.

A. Generation of the probability labels

The initial phase is performed through a random forest classifier, the **Condenser**, denoted by $\mathcal{C}$. We first train this classifier (step ❶ in Fig. 2). Then, we leverage the intrinsic property of the random forest algorithm of being an ensemble method, that is, a composition of several decision trees (or estimators), where the final output is generated after evaluating the response of each individual tree. This characteristic allows us to produce the desired probability vectors by considering the percentage of estimators that predicted a specific result (step ❷ in Fig. 2). Formally, let X be a dataset, $|X| \in \mathbb{N}$ the number of samples that constitute X, and $x_i \in X (0 \leq i \leq |X|)$ a sample within this dataset; let Y be the set of hard class labels (in the form of indicator vectors) associated

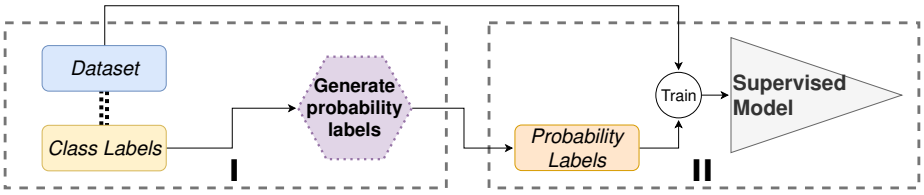


Figure 1: The two phases of the cyber detector.

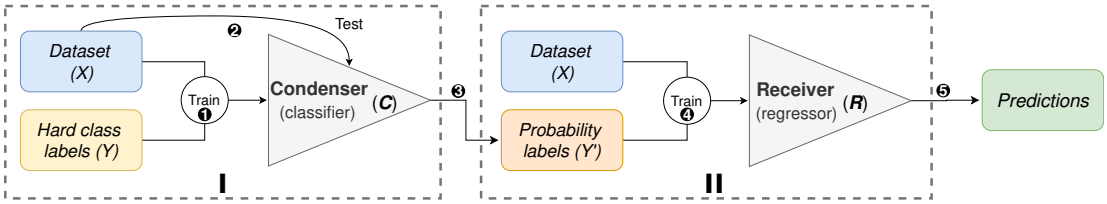


Figure 2: Workflow of the approach where distillation is applied to the random forest algorithm.

to dataset X, and $y_i \in Y$ the label associated to x_i. If $\mathcal{C}$ is a random forest classifier, then $|\mathcal{C}| \in \mathbb{N}$ is the number of estimators that compose $\mathcal{C}$, and $t_j \in \mathcal{C} (0 \leq j \leq |\mathcal{C}|)$ is a tree of classifier $\mathcal{C}$. After training $\mathcal{C}$ by means of X (as training dataset) and of Y (as labels), the set of probability labels Y' that can be obtained from X through $\mathcal{C}$ is:

$$Y' = \left\{ y'_i \mid y'_i = \frac{\sum_{j=1}^{|\mathcal{C}|} t_j^i}{|\mathcal{C}|} \right\}, \tag{1}$$

where y'_i is the probability vector corresponding to sample x_i, and t_j^i denotes the output of tree t_j for sample x_i, which is an indicator vector. As an example, let us consider a random forest classifier consisting of 100 estimators that are trained to solve a binary classification problem (either 0 or 1). Now, let us assume that, for a given sample, 31 estimators predict 0 and produce the indicator vector $(1, 0)$, while the remaining 69 predict 1 and produce the indicator vector $(0, 1)$. In this case, although the final output of the classifier is the indicator vector $(0, 1)$, we generate the binary probability vector $(0.31, 0.69)$ which encodes the output produced by each individual tree. On the other hand, if 69 estimators predict 0 and 31 estimators predict 1, we would obtain the probability vector $(0.69, 0.31)$.

It should be noted that the objective of the Condenser is to generate accurate probability labels but it does not perform detection. As the focus is on the prediction of every individual estimator, and not on the classification results of the whole random forest classifier, the concept of "misclassification" does not strictly apply to this phase. For example, let us consider a binary classification scenario where we train the Condenser and then test it to generate the probability labels: it may be possible that, for a sample associated to the label 1, 69% of the estimators of the Condenser predict a 0. This event cannot be considered a misclassification because the output of the Condenser is a probability (e.g., the probability vector $(0.69, 0.31)$). However, such occurrences may have a detrimental effect in the next phase. To minimize similar risks, we utilize the entire available dataset to both train and test $\mathcal{C}$: this approach would yield the best results as it ensures that each sample is associated to a probability label with the highest degree of confidence.

B. Model deployment

In the second phase, the probability vectors generated by the Condenser (step ❸ in Fig. 2) are used as training labels for a random forest *regressor* that uses those probabilities as its training input (step ❹ in Fig. 2). We define this model as the **Receiver** denoted by $\mathcal{R}$. Since this model performs the actual detection tasks (step ❺ in Fig. 2), we evaluate it against the adversarial inputs. Hence, it is important that this model is trained by following the best practices (as in [24]) to avoid the risk of overfitting. For example, the training and validation sets should be chosen through appropriate splits of the available dataset.

We remark that the Receiver can be seen as a complex multi-output regressor with the challenging task of multi-target regression [47]. However, for the specific scenarios related to cyber detection, it is possible to devise a simpler regressor because the main goal is to analyze network traffic and to identify illegitimate activities. Hence, we can model the case as a binary classification instead of a multi-class problem, in which the algorithm

is required to determine only whether a given sample of traffic is malicious or not. To this purpose, for each data sample, the Condenser needs to generate a single probability value (denoting the likelihood of being a malicious sample) instead of a multi-dimensional probability vector. By considering the binary classification example described in Section III-A, the 31 estimators of the Condenser that predicted a 0 would give the value 0, while the remaining 69 estimators would produce the value 1. Thus, the corresponding probability value for the analyzed sample is 0.69. These probability values are then used as the labels for the Receiver, whose output is another probability value that can be converted into a discrete number through a rounding operation:

$$P(x_i) = \lfloor \mathcal{R}_{x_i} \rceil, \tag{2}$$

where $\mathcal{R}_{x_i}$ is the output of the Receiver $\mathcal{R}$ for the sample x_i, and $P(x_i) \in [0, 1]$ denotes the final prediction of the distilled model.

IV. Application Scenario for the Detector

A realistic scenario where the proposed detector can be applied successfully is represented in Fig. 3, which shows a large enterprise network with many internal hosts and a border router connected to a network flow exporter. The generated flows are inspected by a network intrusion detection system based on machine learning that aims to identify malicious activities (e.g., botnet) by leveraging the random forest algorithm. We assume that an attacker has already established a foothold in the internal network by compromising one or more machines and deploying botnet malware that communicate with a Command and Control (CnC) infrastructure. The attacker model can be described accordingly to the four characteristics described in [10]: goal, knowledge, capabilities, strategy.

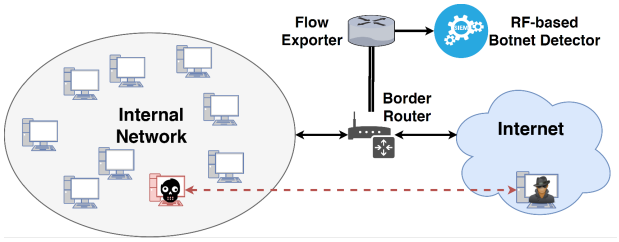


Figure 3: Example of network considered in our use-case.

The main goal of the attacker is to evade detection so that he can maintain access to the internal network, compromise more machines and gather information about adopted defenses [48]. He knows that network communications are monitored by a NIDS based on machine learning. We assume that the attacker can issue commands to the bot through the CnC infrastructure, possibly modifying the underlying network behavior, but he cannot interact with the detector. Although the attacker does not know the specific machine learning algorithm (alongside its parameters and features) used by the NIDS, he can easily guess that the detector is trained over a dataset containing malicious flows generated by the same or a similar malware variant deployed on the infected machines. Hence, he has to devise some countermeasure to evade the botnet detector.

The strategy to avoid detection is through a *targeted exploratory integrity attack* [10] that is performed by inserting tiny modifications in the communications between the bot and its CnC server. These alterations may include slight increases of flow duration, exchanged bytes and exchanged packets. Similar changes can be applied without interfering with the application logic of the bot that can continue to operate as initially designed by the attacker. In such a way, the detector is induced to misclassify the network flows generated by bot communications despite being trained with malicious samples belonging to the botnet variant employed by the attacker.

V. Evaluation Methodology

The evaluation and comparison of machine-based detectors subject to adversarial attacks is a complex procedure. In this section, we describe the methodology of our evaluation by presenting the experimental testbed, the details of the considered random forest models, and the procedure to generate the adversarial samples.

A. Experimental testbed

The experimental evaluation considered in our paper is performed on a public collection of multiple datasets, known as "CTU-13" [49]. The CTU-13 includes network data captured at the Czech Technical University in Prague, and contains labelled network traffic generated by various botnet variants and mixed with normal and background traffic. These flows are captured in a network environment with hundreds of hosts, while the malicious traffic is generated by infecting machines with malware related to several botnet families [49]. Overall, the CTU-13 contains 13 distinct datasets of different botnet activity; each dataset refers to one botnet variant of the 6 considered families: `Neris`, `Rbot`, `Virut`, `Menti`, `Murlo`, `NSIS.ay`. We report the meaningful metrics of each dataset in the CTU-13 collection in Table I, which also includes the botnet-specific piece of malware and the number of infected machines. This Table highlights the massive amount of included data, which can easily represent the network behavior of a medium-to-large

real organization. Nevertheless, we remark that in our evaluation, we prefer not to consider the Sogou botnet because of the limited amount of its malicious samples.

To generate each dataset, the authors first capture the network data in specific packet-capture (PCAP) files, and then convert them into *network flows*. A network flow (or *netflow*) is essentially a sequence of records, each one summarizing a connection between two endpoints (that is, IP addresses). The inspection of network flows allows administrators to easily summarize the information of two endpoints, such as the source and destination of traffic, the class of service, and the size of transmitted data. Network flows are of particular interest for cyber security applications because of the following benefits with respect to full packet captures: lower amount of storage space required; faster analyses; reduced privacy concerns due to the absence of packet-specific payloads [50].

The authors of the CTU-13 convert the raw network packets into network flows by means of Argus, a network audit system. Argus presents a client-server architecture: the server component processes packets (either PCAP files or live packet data) and generates detailed status reports of all the netflows in the packet stream, which are then provided to the dedicated clients. By inspecting the CTU-13, we can assume that the client used by the authors to extract the netflows from each individual PCAP file is `ra`. The output of this conversion process is a CSV file. The final step is the labeling of each individual network flow: indeed, the authors provide an additional "Label" field, which separates legitimate from illegitimate flows. More specifically, benign flows correspond to the *normal* and *background* labels; whereas the *botnet* and *CnC-channel* labels denote malicious samples.

B. Considered detectors

For the evaluation we consider the following detectors based on random forest:

- The **Undistilled** detector, which presents characteristics similar to the random forest classifier model proposed in [51], is used as the baseline for the experiments; a graphical representation of its architecture is provided in Fig. 4.
- The **Distilled** detector represents the main proposal of this paper. It consists of the *Condenser* for generating the probability labels, and of the *Receiver* to perform the detection tasks. This detector is evaluated against the Undistilled detector in adversarial and non-adversarial settings.

Each detector has 6 instances, each one focusing on recognizing a specific malware family of the dataset.

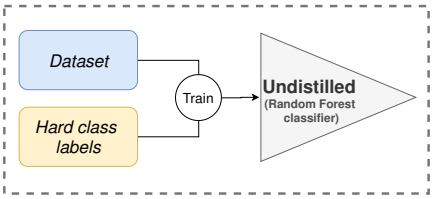


Figure 4: Architecture of the Undistilled detector.

The motivation for this design choice comes from the observation that machine learning techniques yield superior results when they pursue a specific goal rather than aiming to an impossible catch-all solution [6], [51].

For each botnet variant, we generate a dedicated training set containing both benign and malicious samples belonging to that family; all instances share the same legitimate-to-illegitimate flow ratio in the training sets. Formally, let D be the set of all the traces of network flows considered in the testbed, and let $D^l \subset D$ and $D^m \subset D$ be the sets of all legitimate and malicious samples in D, respectively (so that $D^l \cup D^m = D$, and $D^l \cap D^m = \emptyset$). Now, let D^b be the set of malicious flows corresponding to the b botnet family, so that $\bigcup_{b=1}^{6} D^b = D^m$. We train each detector's instance corresponding to the b botnet family with samples randomly extracted from D^l and D^b, in a $20 : 1$ ratio. (The randomized extraction of samples is done to reduce the impact of selection bias.) The $20 : 1$ ratio is similar to that in [15], and it is motivated by the fact that in realistic settings the legitimate flows largely outnumber the botnet-generated flows. Other studies use even greater ratios [52]. The instances of the Receiver are trained with 80% of the botnet flows generated by each malware variant, and validated on the remaining 20%. These splits are close to those adopted in [15], [20]. On the other hand, the instances of the Condenser, which generate the probability labels, are trained and tested on the same dataset containing all the malicious flows of the related botnet family. Other details are presented in Section III. These models adopt feature sets that are similar to those adopted in [20] and [51] because they achieve appreciable detection rates. We integrate these features with information about the IANA *port type* for the source and destination hosts, thus obtaining the list summarized in Table II. For completeness, we remark that the code for the experiments is implemented in *Python3* and uses the *scikit-learn* toolkit. Moreover, we report in Table III the meaningful parameter settings of each model, which are chosen through extensive grid search operations. The F parameter denotes the number of features in input, and MSE is the Mean Squared Error.

Table I: Meaningful metrics of the CTU-13 collection. Source: [49].

Dataset	Duration (hrs)	Size (GB)	Packets	Netflows	Malicious Flows	Benign Flows	Botnet	# Bots
1	6.15	52	71 971 482	2 824 637	40 959	2 783 677	Neris	1
2	4.21	60	71 851 300	1 808 122	20 941	1 787 181	Neris	1
3	66.85	121	167 730 395	4 710 638	26 822	4 683 816	Rbot	1
4	4.21	53	62 089 135	1 121 076	1 808	1 119 268	Rbot	1
5	11.63	38	4 481 167	129 832	901	128 931	Virut	1
6	2.18	30	38 764 357	558 919	4 630	554 289	Menti	1
7	0.38	6	7 467 139	114 077	63	114 014	Sogou	1
8	19.5	123	155 207 799	2 954 230	6 126	2 948 104	Murlo	1
9	5.18	94	115 415 321	2 753 884	184 979	2 568 905	Neris	10
10	4.75	73	90 389 782	1 309 791	106 352	1 203 439	Rbot	10
11	0.26	5	6 337 202	107 251	8 164	99 087	Rbot	3
12	1.21	8	13 212 268	325 471	2 168	323 303	NSIS.ay	3
13	16.36	34	50 888 256	1 925 149	39 993	1 885 156	Virut	1

Table II: Features of the random forest models. Source: [20].

#	Feature name	Feature type
1,2	source/destination IP address type	Boolean
3,4	source/destination port	Numerical
5	flow direction	Boolean
6	connection state	Categorical
7	duration (seconds)	Numerical
8	protocol	Categorical
9,10	source/destination ToS	Numerical
11,12	outgoing/incoming bytes	Numerical
13	total transmitted packets	Numerical
14	total transmitted bytes	Numerical
15,16	source/destination port type	Categorical
17	bytes per second	Numerical
18	bytes per packet	Numerical
19	packets per second	Numerical
20	ratio of outgoing/incoming bytes	Numerical

Table III: Parameters of the random forest models.

	Parameter name	Value
Undistilled	Number of estimators	763
	Quality Function	Gini
	Features for best split	$\sqrt{F}$
	Bootstrap	Yes
Condenser	Number of estimators	894
	Quality Function	Gini
	Features for best split	$\sqrt{F}$
	Bootstrap	Yes
Receiver	Number of estimators	1352
	Quality Function	MSE
	Features for best split	$F/2$
	Bootstrap	Yes

C. Generation of adversarial datasets

We produce multiple adversarial datasets by manipulating the botnet netflows D^b through feature modifications. Since the produced adversarial samples are used to evaluate the proposed approach, we consider the portion of botnet netflows from D^b contained in the datasets used for the testing-phase, thus avoiding the submission of samples contained in the training set.

An attacker can evade detection by increasing the flow duration through a small latency; and the number of bytes (or packets) by adding random junk data. All these modifications can be introduced in the network behavior of the bots without altering their underlying logic. To reproduce a similar adversarial attack pattern, we generate adversarial samples by manipulating combinations of up to 4 features, such as the duration of the flows, the total number of transmitted packets, the number of outgoing(Src) or incoming(Dst) bytes. Table IV reports the 15 groups of altered features denoted by G. As an example, adversarial samples belonging to group `1a` alter only the flow *duration*, while those of group `3c` include modifications to the *duration*, *dst_bytes* and *tot_packets* features. The feature manipulation is performed by augmenting each of these groups through 9 increment steps denoted by S; these steps are fixed for all the possible combinations. Hence, for each botnet family, we produce 135 adversarial collections, thus resulting in a total of 810 adversarial datasets (given by 15[groups of altered features] $*$ 9[increment steps] $*$ 6[botnet families]).

Table V reports the relationship between each step and the corresponding feature increments where *Duration* is measured in seconds. As an example, the adversarial datasets obtained through the `VI` step of the group `1b` have the values of their flow outgoing bytes increased by 128. The adversarial datasets obtained through the `II` step of the group `3c` have the values of their flow duration, incoming bytes and total packets increased by 2. There is a greater focus on small increments since they are easier to achieve and they are still able to generate samples that evade detection. The rationale behind the choice of the values shown in Table V is the following: our objective is to generate adversarial malicious samples that are only marginally different from

Table IV: Groups of altered features. Source: [20].

Group (g)	**Altered features**
1a	Duration (in seconds)
1b	Src_bytes
1c	Dst_bytes
1d	Tot_pkts
2a	Duration, Src_bytes
2b	Duration, Dst_bytes
2c	Duration, Tot_pkts
2d	Src_bytes, Tot_pkts
2e	Src_bytes, Dst_bytes
2f	Dst_bytes, Tot_pkts
3a	Duration, Src_bytes, Dst_bytes
3b	Duration, Src_bytes, Tot_pkts
3c	Duration, Dst_bytes, Tot_pkts
3d	Src_bytes, Dst_bytes, Tot_pkts
4a	Duration, Src_bytes, Dst_bytes, Tot_pkts

their original counterparts, as shown in [12]. Although the exact numbers have been selected arbitrarily by adopting the powers of 2 for convenience, our goal is to represent the effects of small, but sensible variations of these features. Furthermore, introducing these small perturbations is a realistic task for the type of attacker considered in this paper. On the other hand, excessive increases higher than those shown in Table V may generate anomalous network flows that can be detected by different defensive mechanisms (e.g., [50]). Moreover, increasing the duration of each flow above 120 seconds may exceed the duration limits of the flow collector [50].

Table V: Increment steps of each feature for generating realistic adversarial samples. Source: [20].

Step (s)	**Duration**	**Src_bytes**	**Dst_bytes**	**Tot_pkts**
I	+1	+1	+1	+1
II	+2	+2	+2	+2
III	+5	+8	+8	+5
IV	+10	+16	+16	+10
V	+15	+64	+64	+15
VI	+30	+128	+128	+20
VII	+45	+256	+256	+30
VIII	+60	+512	+512	+50
IX	+120	+1024	+1024	+100

The generation of the adversarial datasets is described in Algorithm 1, where $\mathcal{A}(\cdot)$ denotes the operator indicating an adversarially manipulated input. We remark the importance of the operation on line 19, because it shows that some features are mutually dependent. For example, for consistency reasons, increasing the flow duration requires to update also the bytes per second and the packets per second.

For the performance evaluation we adopt the typical machine learning metrics: *Precision* (*Prec*), *Detection Rate* (*DR*, or *Recall*), *F1-score*, computed as follows:

$$Prec = \frac{TP}{TP + FP}, \quad (3) \qquad DR = \frac{TP}{TP + FN}, \quad (4)$$

$$F1\text{-}score = 2 * \frac{Precision * DR}{Precision + DR}, \quad (5)$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively. In the remainder of this paper, we consider a positive detection as a malicious sample.

Algorithm 1: Algorithm for generating datasets of adversarial samples.

Input: List of datasets of malicious flows X^m divided in botnet-specific sets X^b; list of altered features groups G; list of feature increment steps S.
Output: List of adversarial datasets $\mathcal{A}(X^m)$.

1 $\mathcal{A}(X^m) \leftarrow$ emptyList();
2 **foreach** *group* $g \in G$ **do**
3 **foreach** *step* $s \in S$ **do**
4 **foreach** *dataset* $X^b \in X^m$ **do**
5 $\mathcal{A}_s^g(X^b) \leftarrow CreateOneDataset(s, g, X^b)$;
6 Insert $\mathcal{A}_s^g(X^b)$ in $\mathcal{A}(X^m)$;
7 **return** $\mathcal{A}(X^m)$
8 `// Function for creating a single adversarial dataset` $\mathcal{A}_s^g(X^b)$ `corresponding to a botnet-specific dataset` X^b`, a specific altered feature group` g`, and a specific increment step` s`.`
9 **Function** *CreateOneDataset*(s, g, X^b)
10 $\mathcal{A}_s^g(X^b) \leftarrow$ emptyList();
11 **foreach** *sample* $x^b \in X^b$ **do**
12 $\mathcal{A}_s^g(x^b) \leftarrow AlterSample(s, g, x^b)$;
13 Insert $\mathcal{A}_s^g(x^b)$ in $\mathcal{A}_s^g(X^b)$;
14 **return** $\mathcal{A}_s^g(X^b)$
15 `// Function for creating a single adversarial sample` $\mathcal{A}_s^g(x^b)$ `corresponding to a botnet-specific sample` x^b`, a specific altered feature group` g`, and a specific increment step` s`.`
16 **Function** *AlterSample*(s, g, x^b)
17 $\mathcal{A}_s^g(x^b) \leftarrow x^b$;
18 Increment features g of $\mathcal{A}_s^g(x^b)$ by s;
19 Update features of $\mathcal{A}_s^g(x^b)$ that depend on g;
20 **return** $\mathcal{A}_s^g(x^b)$

VI. Performance evaluation

We present the results of a large set of experiments with the aim of demonstrating that: (i) the proposed distilled random forest detector achieves comparable or better detection performance than state-of-the-art algorithms in scenarios that are not subject to adversarial inputs; (ii) it significantly improves the robustness of machine learning models against adversarial attacks. Achieving both results is an important outcome for cyber security contexts where we cannot anticipate whether a machine learning detector is subjected or not to adversarial attacks.

We evaluate and compare the performance of distilled and undistilled models in scenarios where samples are not adversarially modified. Then, we assess the effectiveness of the distilled random forest model against adversarial perturbations. Finally, we compare the result of the proposed method against two existing defensive strategies that can be applied to any supervised machine learning algorithm.

A. Evaluation in normal scenarios

We initially generate the probability labels for the Distilled detector by training and testing its Condenser model. Then, we train both the Distilled (through the Receiver) and Undistilled detectors on the same training set (but with appropriate labels), and proceed to evaluate them on the same test set. The results are shown in Table VI, where the columns report the chosen evaluation metrics, and the rows denote the botnet-specific instances of the Undistilled and Distilled detectors; the last row summarizes the results of each detector, which are averaged among all instances. From this table, we observe that the Distilled detector achieves the best results as it obtains higher Precision and F1-scores, and superior detection rates. We stress that the performance of the Distilled is similar to that obtained by state-of-the-art random forest-based botnet detectors [41], [51]. Furthermore, we highlight that our proposal also outperforms the initial defensive distillation technique applied to neural networks in non-adversarial settings, because the distilled neural network model presents a reduced accuracy of ~1.5% when compared to a not-distilled neural network model [37]; this performance drop also affects distilled neural networks for malware classification scenarios [39], which exhibit an increased rate of false alarms. It is important to note that the unusual perfect $Prec$ scores achieved by both models for the `Murlo` botnet and by the Undistilled model for the `Menti` botnet can be motivated as follows: the large majority of the network flows generated by these botnet variants are significantly different from benign traffic, hence the models are able to recognize their malicious samples without generating false positives; however, some instances are still able to evade detection as indicated by the imperfect Recall value. These experiments show that, in the absence of adversarial attacks, our version of the distillation technique applied to random forests yields a detector with similar or superior performance than those that do not adopt a distillation technique. These results are crucial because they refer to a large set of scenarios and demonstrate that random forest-based detectors integrated with distillation are effective even in the absence of adversarial inputs.

Table VI: Baseline vs. Distilled model performance.

Botnet	Detector	F1-Score	Precision	Recall
`Neris`	Undistilled	0.9577	0.9615	0.9540
	Distilled	0.9651	0.9671	0.9632
`Virut`	Undistilled	0.9682	0.9876	0.9496
	Distilled	0.9753	0.9876	0.9633
`Murlo`	Undistilled	0.9932	1	0.9866
	Distilled	0.9968	1	0.9937
`Rbot`	Undistilled	0.9994	0.9999	0.9999
	Distilled	0.9995	0.9999	0.9990
`Menti`	Undistilled	0.9984	1	0.9969
	Distilled	0.9979	0.9997	0.9969
`NSIS.ay`	Undistilled	0.9213	0.9925	0.8596
	Distilled	0.9273	0.9784	0.8812
Average	Undistilled	0.9729	0.9774	0.9684
	Distilled	0.9777	0.9804	0.9751

Since supervised machine learning methods for cyber defense need periodic re-trainings [6], it is important to evaluate the computational cost of the proposed solution. Thus, we measure and report the training times of the considered detectors in Table VII, which compares the time (in seconds) required for training the baseline Undistilled detector (composed of a single random forest classifier) with those required by our method; as the proposed Distilled detector includes both the Condenser and the Receiver, we report the combined training time of these components. Computations are performed on a machine with the following hardware: CPU Intel Core i7-7700HQ, RAM 32GB, and SSD 512GB. We observe that training the Distilled detector requires more effort, because it is composed of two models and, in addition, training a random forest regressor (that is, the Receiver) is more demanding than training a classifier. However, we stress that these operations needs to be executed only periodically. Moreover, by performing the training computations on machines with dedicated hardware it is possible to decrease the absolute training time difference to negligible amounts.

B. Evaluation in adversarial settings

It must be determined whether and to which extent the proposed method is able to address issues related to adversarial attacks. To this purpose, we test the Distilled and the Undistilled detectors against the generated adversarial datasets, and compare their performance. The detection rate is the metric of interest for these analyses. We anticipate that this evaluation highlights a twofold improvement of our proposal: a significant increase in the detection rate; a more stable behavior against different adversarial samples of the same botnet family.

Among the considered 810 adversarial datasets, the Distilled detector clearly outperforms the baseline Undistilled in 759 cases; for the remaining 51 datasets, the

Table VII: Training time of each instance of the detectors.

Botnet	Detector	Time (s)
Neris	Undistilled	75.8
	Distilled	212.5
Virut	Undistilled	16.7
	Distilled	42.7
Murlo	Undistilled	19.8
	Distilled	53.9
Rbot	Undistilled	77.1
	Distilled	210.4
Menti	Undistilled	2.8
	Distilled	8.5
NSIS.ay	Undistilled	1.6
	Distilled	5.7
Average	Undistilled	32.3
	Distilled	87.0

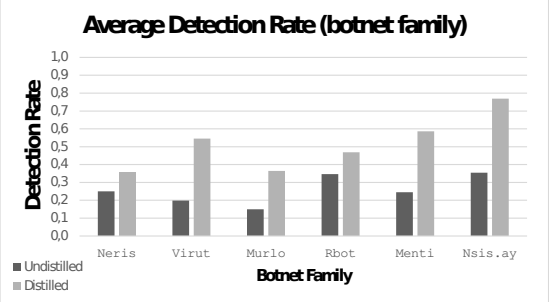


Figure 5: Comparison of the average detection rates on each malware family.

results of the two detectors are close. A comprehensive overview of the effectiveness of the two detectors is presented in Fig. 5, where the black and gray histograms report the detection rates of the Undistilled and Distilled detectors, respectively. Each histogram denotes the average performance of the models applied to each botnet family. There is no doubt that the Distilled is significantly superior to the Undistilled detector, with improvements ranging from 50% to 250%.

We provide a more detailed comparison of the two detectors by considering the impact on detection rates of different altered features. The results are reported in Fig. 6, where the x-axis denotes the group of altered features, and every histogram is generated by averaging the detection rates achieved by each instance of the detectors for all increment steps. From this figure, we can observe that the Distilled achieves superior detection rates for all the groups. The improvements for the groups `2a`, `2b` and `3a` are the most significant, as they allow the Distilled to retain a detection rate that is much higher than that of the Undistilled model. Moreover, the results for group `1a` show that the Distilled detector is almost unaffected by alterations of the flow duration. On the other hand, adversarial alterations involving multiple features have a high impact on the performance of both detectors, as these modifications cause the malicious test samples to be considerably different than those used to train each model. Nevertheless, it is appreciable that, even in these tough circumstances, the Distilled is able to correctly identify more than twice the amount of malicious flows with respect to the Undistilled detector.

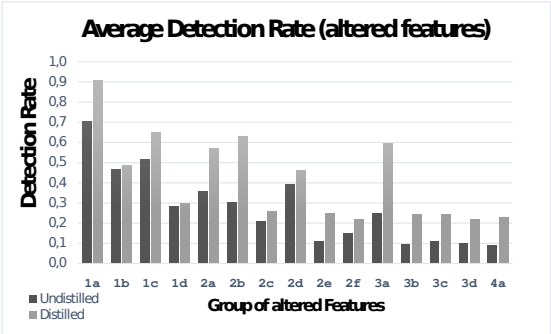


Figure 6: Comparison of the average detection rates for each group of altered features.

We also evaluate the detection rates of the two detectors for variable increment steps. The results are presented in Fig. 7, where the x-axis represents the increment steps and the histograms are generated by averaging the performance over all groups of altered features. We note that not only the Distilled outperforms the Undistilled model, but that it is much more resilient against samples that greatly differ from their original malicious version. Indeed, the detection rates for the `VIII` and `IX` steps are close to 50%; whereas the 15% detection rate of the Undistilled model is unacceptably low. This figure also shows that the Distilled presents a more stable behavior against adversarial samples that are obtained through different increment steps: its detection rates are between 46% and 61%, against the much broader 11% to 45% range of the Undistilled model. From Fig. 7 and Fig. 6, we observe that greater perturbations correspond to the lowest detection rates; however, we remark that such modifications may generate alerts from other defensive mechanisms (as explained in Section V). Furthermore, we highlight that adversarial attacks are more effective and more difficult to detect when they are carried out through adversarial samples that are as close as possible to original samples.

We investigate the increased stability of our proposal through the fine grained comparisons in Figs. 8, where the lines denote the detection rate (averaged for all botnet families) of the two models for four fixed groups of

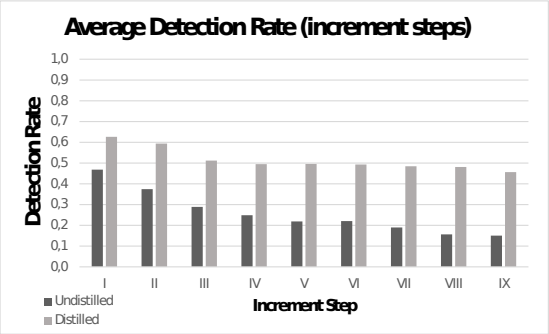


Figure 7: Comparison of the average detection rates for each increment step.

altered features (reported on top of each figure) and variable increment steps. The x-axis denotes the increment steps, and the y-axis the detection rate. The black and the gray line refers to the Undistilled and the Distilled model, respectively. In order to appreciate the improved stability of the performance, we include in Figs. 9 the boxplots related to the results of Figs. 8. These boxplots highlight that the Distilled detector is not affected by sudden performance drops, thus indicating that it is able to maintain its performance even against adversarial inputs that are different from the scenarios considered in this paper. The increased resilience of the Distilled detector is motivated by the fact that its Receiver model adopts a more robust set of feature importances when compared to the Undistilled model. In other words, a random forest model makes a prediction by comparing the features of a sample with the feature importances learned during its training phase: the probability labels used to train the Receiver produce a random forest model with a set of feature importances having a higher degree of flexibility than that of the Undistilled classifier, which adopts hard class labels. As a consequence, an adversary can significantly alter the detection results of the Undistilled model through tiny alterations of the features, while the Distilled detector is capable of withstanding even perturbations of high magnitude. For example, let us consider two cases: in Fig. 8b the adversary modifies only one feature (*Dst_Bytes*); in Fig. 8d the adversary changes three features (*Duration*, *Src_Bytes* and *Dst_Bytes*). In the former case, the two detectors have comparable performance for the first increment steps because the manipulated feature (*Dst_Bytes*) has high and similar importance for both models. In the latter instance, when alterations concern even incoming bytes and flow duration, the detection rates of the Undistilled model are unacceptably low (below 15%).

The improved resilience of our method is confirmed by comparing the detection rates of the two detectors for fixed botnet families. The results and corresponding boxplots are presented in Fig. 10 and Fig. 11, respectively. The name of the considered botnet family is reported on top of each figure. Overall, these figures confirm the superior detection capabilities and improved stability of the Distilled model.

C. Comparison with existing defensive strategies

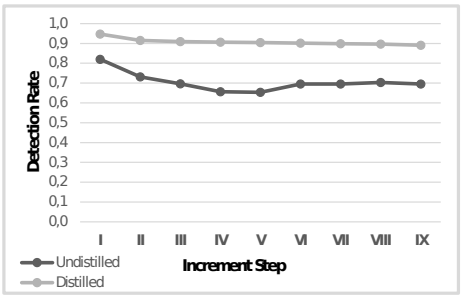
We compare the effectiveness of our proposal against two known countermeasures against evasion adversarial attacks that have been proposed in the literature [18], [31], [34], [35], and that can be applied to any supervised machine learning algorithm: *adversarial retraining* and *feature removal*. To this purpose, we perform our experiments by following the same procedures described in [18], due to the common characteristics shared by the considered adversarial scenarios and employed datasets. Hence, for the case of *adversarial retraining* we generate a "hardened" Undistilled detector by re-training it after introducing a small (10%) portion of the generated adversarial samples into the corresponding training sets, and then measure its detection rate on the same adversarial datasets used in our previous experiments for both the normal and adversarial scenarios. The results of this evaluation are presented in Table VIII which shows the (averaged) Recall obtained by the re-trained Undistilled detector, the proposed Distilled detector, and the baseline Undistilled detector that we include for completeness.

Table VIII: Comparison with adversarial retraining.

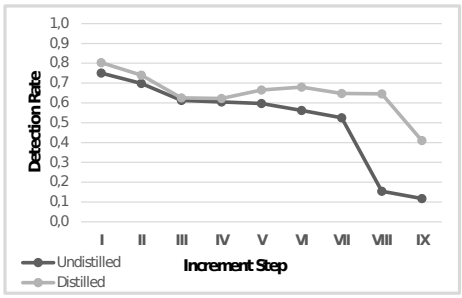
Detector Type	Recall (normal)	Recall (adversarial)
Undistilled (retrained)	0.9695	0.4987
Undistilled (baseline)	0.9684	0.2573
Distilled	0.9751	0.5152

With regards to *feature removal*, we develop a different Undistilled detector by training it on the same dataset used in our previous experiments but without considering the features that we modified to generate our adversarial samples (that is, *Tot_Pkts*, *Duration*, *Dst_Bytes*, *Src_Bytes*), and then test it on the datasets used in Section VI-A; this is motivated by the fact that *feature removal* countermeasures, despite being resilient against adversarial attacks targeting the removed features, are known to generate excessive false alarms. The evaluation results are shown in Table IX, which compares the (average) Precision, Recall and F1-score of the Undistilled detector (after excluding the features) with those obtained by the Distilled and the baseline Undistilled detector.

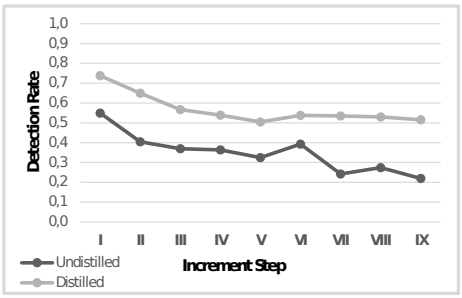
By observing Table VIII, we note that our proposal exhibits a higher detection rate in both scenarios. At



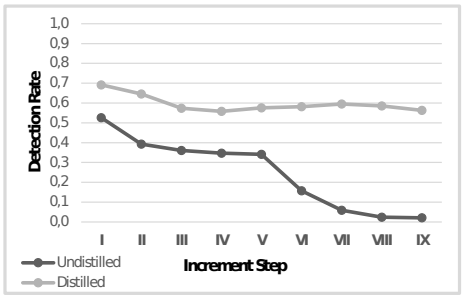
(a) Group `1a`: *Duration*.



(b) Group `1c`: *Dst_Bytes*.

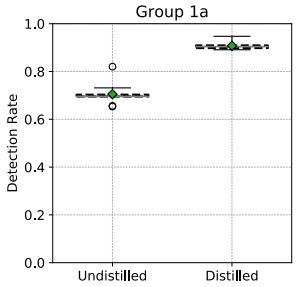


(c) Group `2a`: *Duration & Src_Bytes*.

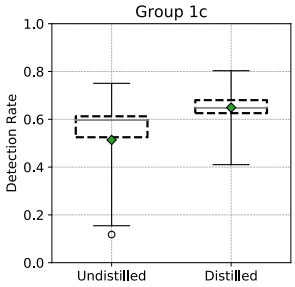


(d) Group `3a`: *Duration & Src_Bytes & Dst_Bytes*.

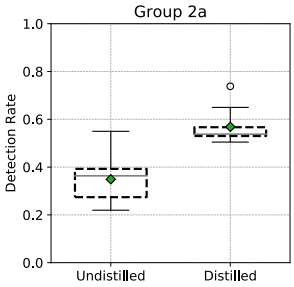
Figure 8: Comparison of the detection rates on the adversarial datasets generated by all malware families.



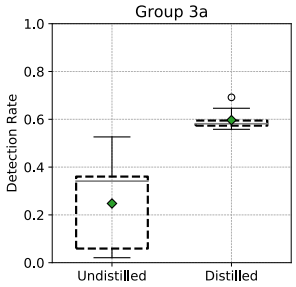
(a) Boxplot for Fig. 8a.



(b) Boxplot for Fig. 8b.



(c) Boxplot for Fig. 8c.



(d) Boxplot for Fig. 8d.

Figure 9: Boxplot visualization of the results in Figs. 8.

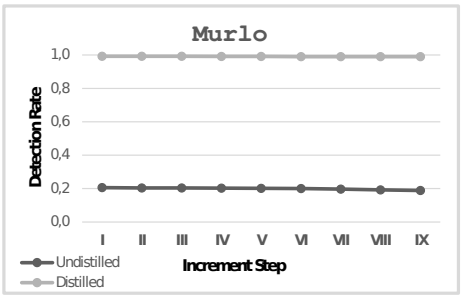
Table IX: Comparison with feature removal.

Detector Type	F1-Score	Precision	Recall
Undistilled (feature removal)	0.8728	0.8497	0.8974
Undistilled (baseline)	0.9729	0.9774	0.9684
Distilled	0.9777	0.9804	0.9751

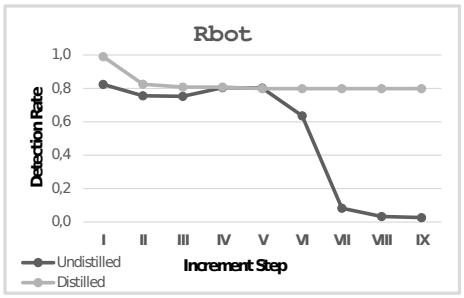
the same time, concerning Table IX, we appreciate that the Distilled detector achieves significantly better results. Indeed, we highlight that the proposed distillation method is not affected by the issues that characterize similar countermeasures: *feature removal* strategies generate unacceptable rates of false positives, whereas *adversarial retraining* requires to constantly update the training set with all the possible variations of samples that can be modified by the attacker (as explained in Section II).

By taking into account all these analyses and evaluations, we can draw the following main conclusions.

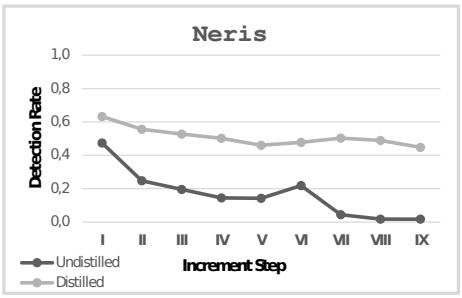
- Current state-of-the-art detection models based on machine learning have features that are too sensitive to the possibile manipulation of an attacker.
- The proposed variation of the defensive distillation technique can be used to devise random forest detectors that: achieve same or better detection performance than existing algorithms in scenarios that are not subject to adversarial inputs; exhibit improved robustness and stability against adversar-



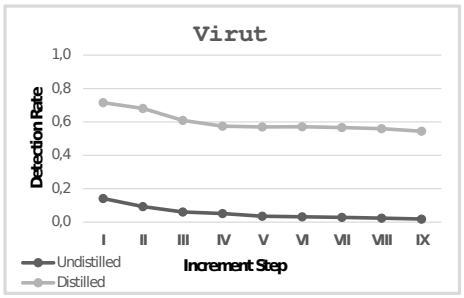
(a) Group `1a`: *Duration*



(b) Group `2b`: *Duration & Dst_Bytes*

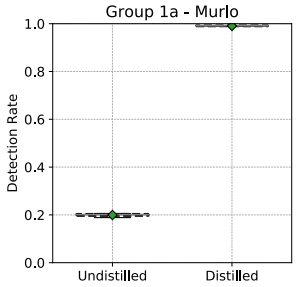


(c) Group `3a`: *Duration & Src_Bytes & Dst_Bytes*

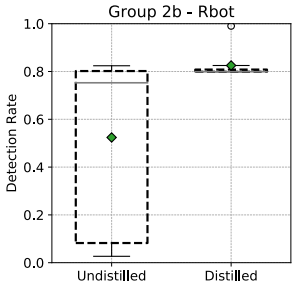


(d) Group `4a`: *Duration & Src_Bytes & Dst_Bytes & Tot_Pkts*

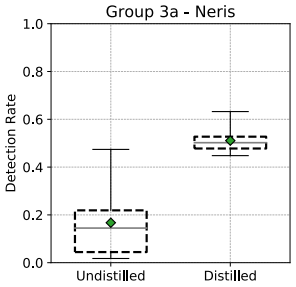
Figure 10: Comparison of the detection rates on the adversarial samples generated by specific malware families.



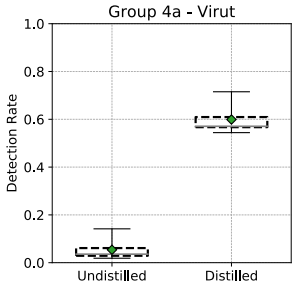
(a) Boxplot for Fig. 10a.



(b) Boxplot for Fig. 10b.



(c) Boxplot for Fig. 10c.



(d) Boxplot for Fig. 10d.

Figure 11: Boxplot visualization of the results in Figs. 10.

ial attacks; are not affected by the limitations of existing countermeasures.

- Although our proposal is an important result towards the reduction of the impact of adversarial inputs against machine learning detectors, it represents just a first step. There is still space for researches that aim to further improve the detection rates.

VII. Conclusions

Adversarial attacks represent a prominent and dangerous menace to organizations that rely on machine learning cyber detectors. We observe that existing approaches are based on classification criteria that are too rigid for the highly variable cyber security domain. The intuition is that by developing more flexible models it is possible to counter the manipulation of malicious samples. For this reason, we present an original method that limits the impact of adversarial perturbations by leveraging the defensive distillation technique. We consider the random forest algorithm due to its superior performance in cybersecurity detection tasks. An extensive campaign of experimental evaluations demonstrates the effectiveness of the proposed method, which achieves a twofold advantage over the state-of-the-art: in scenarios subject to adversarially manipulated inputs, it improves

the detection rate up to 250%; in scenarios that are not subject to adversarial attacks, it achieves a similar or superior accuracy than existing techniques. This latter achievement is of particular importance because existing approaches that aim to counter adversarial attacks are often subject to a reduced performance in non-adversarial settings. Despite these promising results, our method presents room for further improvements. The proposed approach represents an original contribution to design robust detectors with high detection rates and strong enough against adversarial attacks.

REFERENCES

[1] 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.

[2] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," *Artif. Intell. Review*, vol. 29, no. 1, pp. 63–92, 2008.

[3] 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.

[4] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[5] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 305–316.

[6] 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.

[7] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of recurrent neural networks for botnet detection behavior," in *Proc. IEEE Biennal Congress of Argentina*, Jun. 2016, pp. 1–6.

[8] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. IEEE Int. Conf. Platform Techn. Service*, 2016, pp. 1–5.

[9] 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.

[10] 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.

[11] 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.

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

[13] S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection," in *Proc. IEEE Int. Conf. Smart Tech. and Manag. Comp., Commun., Controls, Energy and Materials*, May 2015, pp. 89–95.

[14] 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.

[15] 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.

[16] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, p. 48, 2018.

[17] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Elsevier Inf. Sci.*, vol. 378, pp. 484–497, 2017.

[18] 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.

[19] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on gan," *arXiv:1702.05983*, 2017.

[20] 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.

[21] Z. Abaid, M. A. Kaafar, and S. Jha, "Quantifying the impact of adversarial evasion attacks on machine learning based android malware classifiers," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, Oct. 2017, pp. 1–10.

[22] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, "Zero-day malware detection based on supervised learning algorithms of api call signatures," in *Proc. Austraalasian Conf. Data Mining*, vol. 121, 2011, pp. 171–182.

[23] M. Mannino, Y. Yang, and Y. Ryu, "Classification algorithm sensitivity to training data with non representative attribute noise," *Elsevier Decis. Support Syst.*, vol. 46, no. 3, pp. 743–751, 2009.

[24] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[25] N. Dalvi, P. Domingos, S. Sanghai, D. Verma *et al.*, "Adversarial classification," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Aug. 2004, pp. 99–108.

[26] D. Lowd and C. Meek, "Adversarial learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2005, pp. 641–647.

[27] Y. Zhou, Z. Jorgensen, and M. Inge, "Combating good word attacks on statistical spam filters with multiple instance learning," in *Proc. IEEE Int. Conf. Tools Artif. Intell.*, vol. 2, Oct. 2007, pp. 298–305.

[28] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv:1312.6199*, 2013.

[29] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proc. ACM Workshop Secur. and Artif. Intell.*, Oct. 2011, pp. 43–58.

[30] I. Jeun, Y. Lee, and D. Won, "A practical study on advanced persistent threats," in *Comput. Appl. Secur., Control, Syst. Eng.* Springer, 2012, pp. 144–152.

[31] 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.

[32] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proc. Symp. Netw. Distrib. Syst.*, Feb. 2016, pp. 21–24.

[33] A. Demontis, P. Russu, B. Biggio, G. Fumera, and F. Roli, "On security and sparsity of linear classifiers for adversarial settings," in *Proc. Joint. Int. Workshops Statist. Tech. Pattern Recognit. and Struct. Syntactic Pattern Recognit.* Springer, Nov. 2016, pp. 322–332.

[34] F. Zhang, P. P. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 766–777, 2016.

[35] 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.

[36] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Trans. Depend. Sec. Comput.*, 2017.

[37] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy*, May 2016, pp. 582–597.

[38] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *AAAI Conf. Artif. Intell.*, Apr. 2018.

[39] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv:1606.04435*, 2016.

[40] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. IEEE Int. Conf. Inf. Netw.* IEEE, 2017, pp. 712–717.

[41] 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. Joint Conf. Neur. Netw.*, Jul. 2018, pp. 1–8.

[42] M. Stevanovic and J. M. Pedersen, "Detecting bots using multilevel traffic analysis," *Int. J. Cyber Situational Awareness*, vol. 1, no. 1, 2016.

[43] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.

[44] B. Biggio, G. Fumera, and F. Roli, "Pattern recognition systems under attack: Design issues and research challenges," *Int. J. Pattern Recogn. Artif. Intel.*, vol. 28, no. 07, p. 1460002, 2014.

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

[46] S. J. Pan, Q. Yang *et al.*, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[47] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

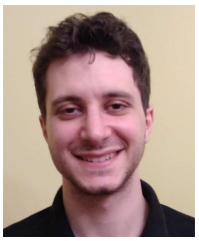
[48] M. Marchetti, F. Pierazzi, M. Colajanni, and A. Guido, "Analysis of high volumes of network traffic for advanced persistent threat detection," *Elsevier Comput. Netw.*, vol. 109, pp. 127–141, 2016.

[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] F. Pierazzi, G. Apruzzese, M. Colajanni, A. Guido, and M. Marchetti, "Scalable architecture for online prioritisation of cyber threats," in *Proc. IEEE Int. Conf. Cyber Conflicts*, May 2017, pp. 1–18.

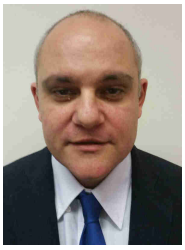
[51] M. Stevanovic and J. M. Pedersen, "An analysis of network traffic classification for botnet detection," in *Proc. IEEE Int. Conf. Cyber Situat. Awar., Data Analyt., Assessment*, Jun. 2015, pp. 1–8.

[52] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," *Comput. Elect. Eng.*, vol. 50, pp. 91–101, 2016.



Giovanni Apruzzese is a PhD Candidate at the International Doctorate School in Information and Communication Technologies (ICT), at the University of Modena and Reggio Emilia, Italy. He received the Master's Degree in Computer Engineering summa cum laude from the same University in 2016. In 2019 he spent 6 months as a visiting scholar at Dartmouth College, NH, USA, under the supervision of prof. VS Subrahmanian. His research interests involve all aspects of big data security analytics with a focus on phishing and network intrusion detection, including machine learning, time series analysis, graph analytics. He is also actively evaluating and researching solutions against adversarial attacks.

Homepage: http://weblab.ing.unimo.it/people/apruzzese



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 (based on a best-effort service or on guaranteed levels of performance).

Homepage: http://weblab.ing.unimo.it/people/andreolini

Michele Colajanni is full professor in computer engineering at the University of Modena and Reggio Emilia since 2000. He received the Master degree in computer science from the University of Pisa, and the Ph.D. degree in computer engineering from the University of Roma in 1992. He manages the Interdepartmental Research Center on Security and Safety (CRIS), and the Master in "Information Security: Technology and Law". His research interests include security of large scale systems, performance and prediction models, Web and cloud systems.

Homepage: http://weblab.ing.unimo.it/people/colajanni

Mirco Marchetti received the Ph.D. degree in Information and Communication Technologies in 2009. He is currently a Researcher 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.

Homepage: https://weblab.ing.unimore.it/people/marchetti