On the Robustness of Random Forest Against Untargeted Data Poisoning: An Ensemble-Based Approach

Marco Anisetti, *Senior Member, IEEE*, Claudio A. Ardagna, *Senior Member, IEEE*, Alessandro Balestrucci, Nicola Bena, *Graduate Student Member, IEEE*, Ernesto Damiani, *Senior Member, IEEE*, and Chan Yeob Yeun, *Senior Member, IEEE*

***Abstract*—Machine learning is becoming ubiquitous. From finance to medicine, machine learning models are boosting decision-making processes and even outperforming humans in some tasks. This huge progress in terms of prediction quality does not however find a counterpart in the security of such models and corresponding predictions, where perturbations of fractions of the training set (poisoning) can seriously undermine the model accuracy. Research on poisoning attacks and defenses received increasing attention in the last decade, leading to several promising solutions aiming to increase the robustness of machine learning. Among them, ensemble-based defenses, where different models are trained on portions of the training set and their predictions are then aggregated, provide strong theoretical guarantees at the price of a linear overhead. Surprisingly, ensemble-based defenses, which do not pose any restrictions on the base model, have not been applied to increase the robustness of random forest. The work in this paper aims to fill in this gap by designing and implementing a novel hash-based ensemble approach that protects random forest against untargeted, random poisoning attacks. An extensive experimental evaluation measures the performance of our approach against a variety of attacks, as well as its sustainability in terms of resource consumption and performance, and compares it with a traditional monolithic model based on random forest. A final discussion presents our main findings and compares our approach with existing poisoning defenses targeting random forests.**

***Index Terms*—Ensemble, machine learning, poisoning, random forest, sustainability.**

I. Introduction

With the introduction of deep neural networks in the last decade, machine learning (ML) is now leaving academia and powering an increasing number of applications, from finance [1] to smart grid [2], weather forecast [3], signal processing [4], and medicine [5], [6]. Machine learning models are reportedly performing better than humans in some specific tasks [7], with an increasing adoption even in safety-critical application scenarios.

In this context, it is of paramount importance to properly evaluate and protect the security of ML models. As such, one of the most relevant threat vectors are data, being ML models trained on (very) large datasets. In particular, *poisoning attacks* include attacks carried out at training time by maliciously altering the training set, with the aim of decreasing the overall classification accuracy, or misclassifying some specific inputs when the model is deployed. Poisoning attacks have been reported in several application scenarios, from malware detection [8] to biometrics [9], healthcare [10], and source code completion [11] and against several types of machine learning models, from support vector machines [12] to decision trees [10], random forests [8], and neural networks [13], to name but a few. Solutions counteracting poisoning are vary, and range from improving the poisoned dataset by removing or repairing (suspicious) data points [14], [15], [16], [17], [18], [19] to strengthening the ML model itself, to make it more resistant to poisoning [20], [21], [22]. Among the model strengthening solutions, ensemble is mostly studied in deep learning and image recognition [20], [21], [22], [23]. It consists of training several ML models on different (possibly partially overlapped) partitions of the training set, and then aggregating their predictions in a single one. This solution stands out for its ability to provide a theoretical bound on the correctness of the prediction according to the extent of poisoning. Although simple in design and implementation, it often builds on a large number (hundreds or thousands) of base models (e.g., [21]), leaving open questions on their sustainability.

Manuscript received 26 October 2022; revised 2 June 2023; accepted 11 June 2023. Date of publication 7 July 2023; date of current version 8 December 2023. This work was supported in part by Technology Innovation Institute under Grant 8434000394, in part by the Project MUSA - Multilayered Urban Sustainability Action - Project, funded by the European Union - NextGenerationEU, through the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D innovation ecosystems, set up of territorial leaders in R&D (CUP G43C22001370007, Code ECS00000037), and in part by the Project SERICS (PE00000014) through NRRP MUR Program funded by the EU - NextGenerationEU. Recommended for acceptance by Special Issue On Sustainable Defense and Security Systems. *(Corresponding author: Claudio A. Ardagna.)*

Marco Anisetti, Claudio A. Ardagna, and Nicola Bena are with the Department of Computer Science, Università degli Studi di Milano, 20133 Milan, Italy (e-mail: marco.anisetti@unimi.it; claudio.ardagna@unimi.it; nicola.bena@unimi.it).

Alessandro Balestrucci is with the Consorzio Interuniversitario per l'Informatica, 00185 Rome, Italy (e-mail: alessandro.balestrucci@consorzio-cini.it).

Ernesto Damiani is with the Department of Computer Science, Università degli Studi di Milano, 20126 Milan, Italy, and also with the Khalifa University of Science and Technology, Abu Dhabi 127788, UAE (e-mail: ernesto.damiani@ku.ac.ae).

Chan Yeob Yeun is with the Khalifa University of Science and Technology, Abu Dhabi 127788, UAE (e-mail: chan.yeun@ku.ac.ae).

Digital Object Identifier 10.1109/TSUSC.2023.3293269

Surprisingly, although random forests are one of the most adopted models on tabular datasets [24] and have been deeply studied from several perspectives, such as explainability [25], fairness [26], and sustainability [27], their robustness has been barely analyzed. Existing works focused on robustness against traditional poisoning attacks [13], [28], [29], [30], [31], [32] and defenses that aim to repair poisoned datasets [33], [34] in limited settings. No model strengthening solutions, including ensemble-based defenses, have been proposed.

Our paper aims to fill in these research gaps in a novel scenario that targets a sustainable and scalable robustness approach for random forest, assuming poisoning attacks that can be implemented by attackers with little to none knowledge and resources (Section IV). In particular, we propose a novel hash-based ensemble approach and empirically evaluate its robustness and sustainability against untargeted, random data poisoning attacks to the accuracy of random forest. Our hash-based ensemble is based on hash functions to route data points in the original training set in different partitions used to train different models in the ensemble. Our implementation extends the well-studied ensemble in [21], [23] as follows: i) each model in the ensemble is trained on a disjoint partition of the training set to which data points are assigned according to hashing, and ii) the final prediction is retrieved according to majority voting. Contrary to state of the art, our paper evaluates ensembles of small to moderate size (i.e., up to 21 random forests), targeting sustainability of defense. Throughout fine-grained experiments, we show that even the simplest label flipping attack carried out with no knowledge or strategy can significantly undermine plain random forests' performance, while consistently with results in literature [13], random forests are almost insensitive to other perturbations. In addition, we show that the usage of even the smallest ensemble does protect from label flipping, while providing a sustainable approach in terms of required resources (CPU and RAM) and performance (execution time).

Our contribution is twofold. We first design and develop a sustainable hash-based ensemble approach extending [21], [23] to increase the robustness of random forest against untargeted, random poisoning attacks; according to our knowledge, this is the first defense based on model strengthening that is applied on random forest. We then evaluate the robustness in terms of accuracy variation according to several untargeted poisoning perturbations, and corresponding sustainability comparing the performance and resource demands of our approach and a plain random forest.

The remainder of this paper is organized as follows. Section II discusses the background and state of the art in the context of poisoning attacks and defenses. Section III presents an overview of our approach based on an ensemble of random forests, whose robustness and sustainability is evaluated according to the threat model in Section IV. Section V describes the evaluation process and target datasets, while Section VI details the results of such a process. Section VII discusses the sustainability of our approach. Section VIII discusses our main findings, while Section IX presents a comparison with approaches in literature. Section X draws our concluding remarks.

II. Background and Related Work

The research community has worked hard to strengthen the security of machine learning (ML) models [35], [36], [37], [38] against different categories of attacks that can be classified according to the stage where they occur. On one side, *adversarial attacks* occur at inference time and consist of specially-crafted data points that are routed to the ML model to cause a faulty or wrong inference. Their goal is the misclassification of such data points. On the other side, *poisoning attacks*, the focus of this paper, occur at training time and inject poisoned data points in the dataset. They aim to reduce the accuracy of the model or cause the misclassification of specific data points at inference time.

Poisoning attacks alter the dataset with malicious data points. They are created by perturbing existing data points in terms of *i)* samples or values of the features [39]; *ii)* labels, having the advantage of not creating anomalous, or at least suspicious, data points [12], [40], [41]. Perturbations can be crafted according to a specific goal such as *i)* misclassification of *positive* data points, for instance in spam detection (*targeted poisoning*), requiring sophisticated perturbations such as *feature collision* [42]; or *ii)* accuracy reduction (*untargeted poisoning*).

The latter often corresponds to random perturbations [12] and is the focus of this paper.

Defenses against poisoning attacks can be performed in two main ways: *dataset strengthening* or *model strengthening*; for other approaches, we refer the reader to [38]. Dataset strengthening aims to increase the quality of the dataset by removing or sanitizing poisoned data points, detected with some heuristics. The latter are based on outlier identification [14], [18], [19], [41], [43] and the evaluation of the impact of data points on the ML model [15], [44], [45], to name but a few. Sanitization include *randomized smoothing* and *differential privacy*. In randomized smoothing, each data point is *smoothed* (i.e., its label is replaced) according to its neighbor data points. Smoothing has been initially proposed to counteract inference-time attacks [46], and then adapted to poisoning [17]. Similarly, in differential privacy, noise is added during training such that predictions done by a model trained on the original dataset are indistinguishable from those of a model trained on the corresponding poisoned dataset, up to a certain ϵ [16], [47].

Model strengthening aims to increase the robustness of the ML model by altering the model itself, such that the effect of poisoning is reduced. Among them, we focus on simple yet effective ensemble approaches, where the monolithic ML model is replaced by a (large) ensemble of the same model [20], [21], [22], [23]. This technique, evaluated mostly on neural network-based models, splits the training set in different partitions according to some strategies, and each partition is used as the training set of a model of the ensemble. Intuitively, this reduces the influence of poisoned data points, since each model is trained on a smaller fraction of poisoned data points.

Some of the above techniques, including ensemble [20], [21], [22], [23], randomized smoothing [17], and differential privacy [16], can provide a certifiable guarantee such that the model prediction is correct up to a certain amount of poisoning on the dataset.

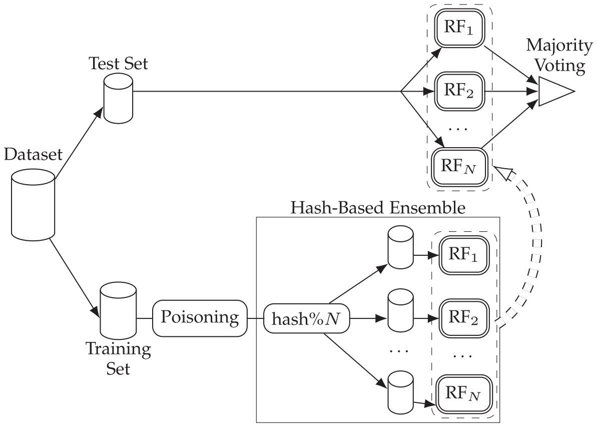


Fig. 1. Overview of our hash-based ensemble approach.

Poisoning defenses often add a not-negligible resources overhead over training and inference, two procedures that by themselves are significantly resource-intensive. For instance, dataset strengthening techniques require training of additional (un)supervised models [14] or nearest neighbor search [18], while model strengthening techniques require very large ensembles of base models [20], [21], [22], [23], often without a performance analysis.

Poisoning attacks and defenses on random forest, the target of this paper, have been only partially investigated. In terms of attacks, existing work focuses on poisoning attacks that alters either labels [13], [28], [32], [33], [34], [48] or features [29], [30], [31]. The most relevant finding is that random forests are more resilient to poisoning than other types of ML models [28], [32]. In terms of defenses, there exist only two papers studying dataset strengthening solutions [33], [34], while no papers presented solutions based on model strengthening, including ensembles. The approach in this paper departs from traditional solutions where the attacker and the defender can perform sophisticated and resource-intensive attacks and defenses (e.g., [18], [21]); it rather aims to provide a robust, while sustainable, model strengthening defense for random forest against untargeted training-time poisoning of labels and features. To the best of our knowledge, this is the first model strengthening defense applied on random forest; a more detailed comparison with the state of the art can be found in Section IX.

III. Our Approach

Fig. 1 shows an overview of our hash-based ensemble approach that aims to increase the robustness of random forest against poisoning attacks. It first splits the tabular dataset in two parts forming the training set (*Training Set* in Fig. 1) and the test set (*Test Set* in Fig. 1). The training set is then poisoned according to our threat model in Section IV (*Poisoning* in Fig. 1). Based on the work in [21], [23], the training set is split in N disjoint partitions using a hash function (*Hash-Based Ensemble* in Fig. 1), with N being the number of random forest in the ensemble, according to the following steps: *i)* the hash value of each data point of the training set is retrieved according to a given hash function; *ii)* the modulo operator (modulo N) is applied on the corresponding hash value (*hash % N* in Fig. 1); *iii)* each data point is routed to a partition of the training set according to the modulo operator on the corresponding hash value (e.g., data points whose hash value modulo N is 0 are assigned to partition 1); and *iv)* the i-th training set of each random forest RF_i is created by evenly taking data points from each partition, such that the training sets are disjoint, have the same cardinality, and balance the contribution from the different partitions in term of data points.

Each random RF_i is then independently trained on the corresponding i-th training set.

At testing and inference time, data points in the test set are fed to each model RF_1, RF_2, ..., RF_n and the final prediction is retrieved according to majority voting.

Beyond being applied on random forest, our ensemble approach and its evaluation departs from existing works in the literature (Section II) according to the following characteristics.

- *Additional round-robin training set creation:* most hash-based ensemble in literature (e.g., [21], [23]) considers one hash function plus a modulo operator only, except few special cases [22]. We instead propose an additional phase, where data point assignment follows hash and modulo operations to increase diversity and ensure equally-sized training sets.
- *Small number of base models:* existing ensemble-based defenses in neural networks require a large number of partitions and base models (e.g., [21]). We instead consider smaller numbers (up to 21) to increase sustainability, while maintaining a good degree of protection.
- *Tabular datasets for binary classification:* most of attacks and defenses are mostly evaluated in image-based scenarios where image datasets are given as input to the models [21]. We instead consider tabular datasets for binary classification, which are still a significant portion of ML.
- *Untargeted poisoning:* most of defenses are evaluated against targeted poisoning (e.g., [18]), where few specially-crafted data points are injected in the training set. We instead consider a threat model where an attacker with limited knowledge and resources randomly alters the dataset to reduce the accuracy of the resulting model (see Section IV).

We note that our approach has been designed and developed to be sustainable, requiring a low amount of resources, since: *i)* it is based on a limited number of base models in the ensemble; *ii)* it does not involve any additional resource-intensive computations, such as the training of additional models other than the random forests in the ensemble; *iii)* it uses a hash-based data point assignment, with hash functions being notoriously fast and lightweight; and *iv)* it trains N random forests independently on disjoint partitions, that is, the cardinality of the dataset is not increased, while training can be parallelized to reduce training time.

IV. Threat Model

Our threat model considers a novel scenario where attackers need to cope with limited knowledge and resources. The attacker departs from targeted attacks and executes untargeted poisoning to reduce the accuracy of the ML model. To this aim, she randomly alters the dataset up to a predefined budget in terms of the amount of manipulated features and labels. Specifically, the attacker implements different perturbations acting on features (*zeroing, noising, out-of-ranging*) and labels (*label flipping*), each implementing a specific poisoning attack that is tested independently. Each perturbation takes as input a training set, (denoted as D), the percentage of data points and features to alter (denoted as ϵ_p and ϵ_f, respectively), according to the specific perturbation, and returns as output the poisoned training set, (denoted as $\widetilde{D}$). The poisoned training set is then partitioned in disjoint training sets, each used to train a model of the ensemble, according to the evaluation process in Section V-A. We note that each perturbation randomly selects the data points and the corresponding features to poison according to ϵ_p and ϵ_f. In particular, the selected features are the same for every perturbation, to ensure proper comparison.

Let us consider as an example a binary classification task (classes 0 and 1), and a 5-feature data point p with value $\langle 0, 10, 15, 0, 1\rangle_0$, where the subscript indicates the label class and the second feature (10) is the target of poisoning.

Perturbation *zeroing* produces a poisoned training set $\widetilde{D}$, where the selected data points are perturbed by changing the values of the selected features to 0. For instance, the poisoned data point of p has value $\langle 0, \mathbf{0}, 15, 0, 1\rangle_0$.

Perturbation *noising* produces a poisoned training set $\widetilde{D}$, where the selected data points are perturbed by replacing the values of the selected features with a value within the distribution of the same feature in the opposite class. For instance, let us consider the second feature of data point p. It takes value in $[0, 10]$ for class 0, and $[20, 40]$ for class 1. For instance, the poisoned data point of p has value $\langle 0, \mathbf{37}, 15, 0, 1\rangle_0$.

Perturbation *out-of-ranging* produces a poisoned training set $\widetilde{D}$, where the selected data points are perturbed by changing the value of the selected features with values outside their valid range. For instance, the poisoned data point of p has value $\langle 0, -\mathbf{1}, 15, 0, 1\rangle_0$.

Perturbation *label flipping* produces a poisoned training set $\widetilde{D}$, where the selected data points are perturbed by flipping their labels. For instance, the poisoned data point of p has value $\langle 0, 10, 15, 0, 1\rangle_\mathbf{1}$.

We note that the effectiveness of these perturbations strongly depends on the data points *actually* perturbed. For instance, let us consider perturbation *zeroing*. Assuming that the first feature of data point p (with value 0) is selected for poisoning, the corresponding poisoned data point is not altered. We also note that the threat model in this paper follows the general trend of cybersecurity attacks, where the danger mostly comes from unsophisticated yet impactful attacks [49].

V. Evaluation Process

We present the evaluation process and the target datasets at the basis of the experimental results on accuracy degradation in Section VI.

A. Evaluation Process in a Nutshell

The evaluation process in Fig. 2 validates the robustness of our ensemble approach in Section III against poisoning attacks in Section IV. For each attack, we calculate the accuracy loss between the plain (monolithic) model and our ensemble approach trained on both original and poisoned dataset. Fig. 2 shows 4 different paths: *i)* poisoned hash-based ensemble, considering base models in the ensemble trained on partitions of the poisoned dataset, *ii)* non-poisoned hash-based ensemble, considering base models in the ensemble trained on partitions of the original dataset, *iii)* monolithic model *RF* trained on the entire poisoned dataset, *iii)* the monolithic model *RF* trained on the entire original dataset.

Our process takes as input: *i)* the original dataset; *ii)* the number of random forests N composing the ensemble; *iii)* the perturbation type; *iv)* the percentage of data points ϵ_p; and *v)* features ϵ_f to poison.

The evaluation process consists of two activities, namely, training, and testing and evaluation. Activity training is composed of 4 steps as follows.

Step 1. Preparation: It splits the dataset into training set (denoted as D) and test set, that is, *held out*. Test set is left untouched for the rest of process.

Step 2. Poisoning: It applies the selected perturbation to the training set D producing a poisoned training set $\widetilde{D}$, according to the percentages of poisoning received as input.

Step 3. Creation of training sets: It builds the training sets for the monolithic and ensemble models. It first creates N empty sets (*partitions*). Second, given a (original or poisoned) training set, each data point is converted to a string by concatenating the value of each feature. For instance, data point p with value $\langle 0, 10, 15, 0, 1\rangle_0$ becomes `0101501`. Third, the hash of such string is retrieved according to a specific hash algorithm. For instance, the hash value of `0101501` according to MD5 is `adf5c364bc3a61133eb2360f7dd0b8f2` (in hexadecimal). Fourth, the modulo operator (modulo N) is applied to the hash value converted back to a number. The result of this operation indicates to which partition the data point belongs, that is, the result n of modulo corresponds to the $n+1$ partition. Then, this step produces N training sets (subsets of the original training set) by selecting the data points assigned to each partition in a round-robin fashion, such that each training set contains roughly the same amount of data points from each partition. This additional assignment distinguishes our ensemble approach from existing hash-based ensemble approaches (e.g., [21], [22], [23]) and guarantees that each training set has approximately the same size and diversity. The ith training set is then used to train random forest RF_i in the ensemble. We note that the training set of the monolithic model is the entire input training set. We also note that this step is repeated both on the original and poisoned datasets.

Step 4. Training: It trains both the ensemble and monolithic models separately on the original and poisoned training sets created at Step 3. We note that random forests in the ensemble are trained independently one to another.

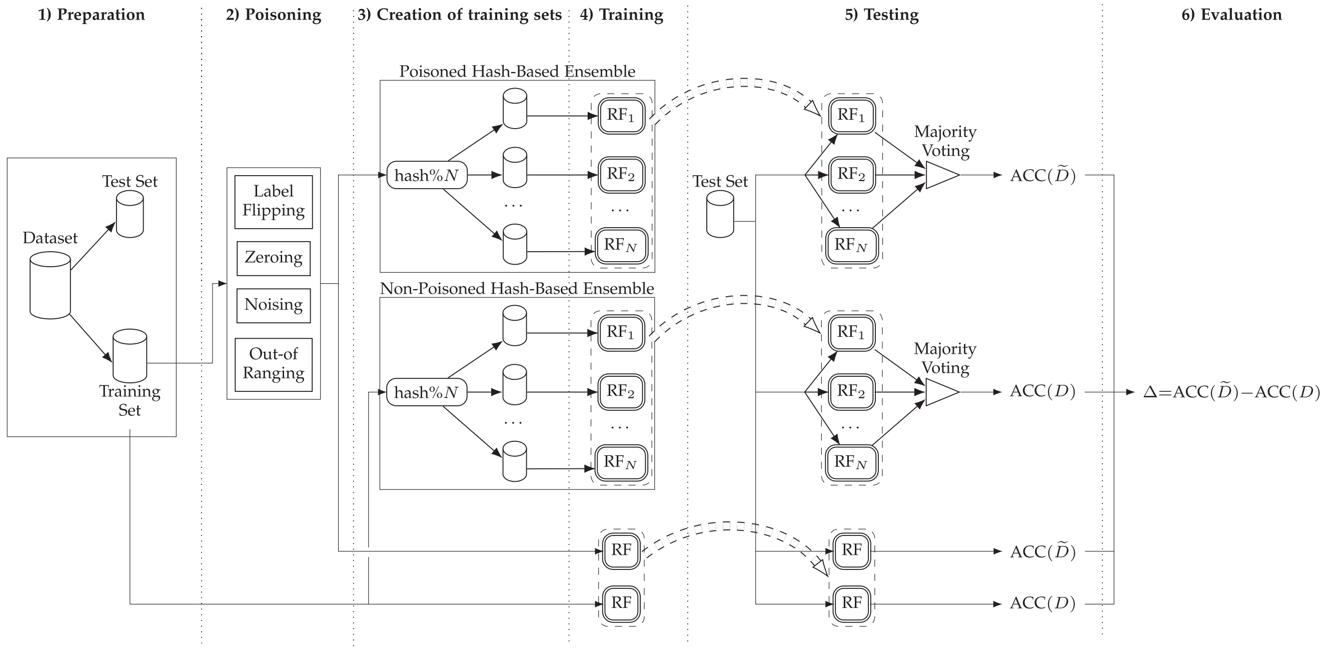


Fig. 2. Evaluation process based on the ensemble approach in Fig. 1.

TABLE I
DATASETS DETAILS

Name	N° data points (N° per class)	N° features	N° data points (Preproc.)	Sparsity (%)	Training set size (N° per class)	Test set size (N° per class)
Musk2 (M2)	6,598 (1,017/5,581)	166	2,034	0.28	1,628 (810/818)	406 (207/199)
Android malware (AM)	18,733 (7,254 /11,479)	1,000	14,508	92.37	11,607 (5,831/5,776)	2,901 (1,423/1,478)
Spambase (SB)	4,061 (2,788/1,813)	57	3,626	77.44	2,901 (1,458/1,443)	725 (355/370)
Diabetic Retinopathy Debrecen (DR)	1,151 (611/540)	19	1,080	10.41	864 (431/433)	216 (109/107)

Activity testing and evaluation is composed of 2 steps as follows.

Step 5. Testing: It evaluates the accuracy of both the ensemble and monolithic models on the test set isolated at at Step 1.

Step 6. Evaluation: It compares poisoned and original models using evaluation metric *delta*, denoted as Δ, as follows.

$$\Delta = \mathrm{ACC}(\widetilde{D}) - \mathrm{ACC}(D), \tag{1}$$

where $\mathrm{ACC}(D)$ is the accuracy retrieved on the original training set and $\mathrm{ACC}(\widetilde{D})$ is the accuracy retrieved on the poisoned training set.

We note that Δ measures the accuracy variation in a model trained on a poisoned training set with regards to the same model trained on the original training set. A negative value of Δ indicates that the model trained on a poisoned dataset decreases in accuracy, a positive value indicates that the model trained on a poisoned dataset increases in accuracy, a value equals to 0 indicates the same accuracy.

B. Target Datasets

We experimentally evaluated the approach in this paper using the datasets in Table I, which significantly differ in cardinality (*N° of data points (N° per class)*), number of features (*N° of features*), and sparsity (*Sparsity (%)*). Table I also describes the details of the datasets after a class balancing preprocessing in terms of cardinality (*N° of data points (Preproc.)*), number of data points randomly selected in the training set and number of data points per class (*Training set size (N° per class)*), and the number of data points randomly selected in the test set and the number of data points per class (*Test set size (N° per class)*).[1] We note that sparsity is retrieved from the dataset after preprocessing, and columns describing the cardinality of each class report the cardinality of the positive class first.

Musk2 (M2) is an open dataset for the identification of musk molecules [50], divided in two classes *musk* and *non-musk*.

[1] Datasets are publicly available at https://github.com/SESARLab/ensemble-random-forest-robustness-against-poisoning

The dataset is collected by including different conformations (shapes) of musk and non-musk molecules. In particular, all the low-energy conformations of 141 initial molecules have been generated and manually annotated.

The dataset consists of 6,598 data points (1,017 musk and 5,581 non musk), organized in 166 features. We built a balanced dataset of 2,034 points by randomly subsampling data points in class non musk. After preprocessing, the dataset exhibits a low sparsity ≈0.28%. We finally split the dataset in a training set of 1,628 data points (810 musks and 818 non musks) and in a test set of 406 data points (207 musks and 199 non musks).

Android malware (AM) is a proprietary dataset for the detection of malware on Android devices. The dataset is collected on Android devices with benign and malign apps installed, by capturing the system calls performed by the apps. Any sequence of three consecutive system calls is a feature, whose value is the number of times such sequence has been called.

The dataset consists of 18,733 data points (7,254 malware and 11,479 non malware), organized in 25,802 features. We then built a balanced dataset of 14,508 points, by randomly subsampling data points in class non malware. We further split the dataset in a training set of 11,607 data points (5,831 malware and 5,776 non malware) and in a test set of 2,901 data points (1,423 malware and 1,478 non malware). Finally, being a dataset with high dimensionality and more features than data points, to avoid overfitting, we reduced the number of features according to *InfoGain* [51], a feature ranking method selecting those features that reduce the *entropy* in the dataset (i.e., the most informative features with regards to the dataset). With this method, we reduced the number of features to 1,000. After preprocessing, the dataset exhibits a high sparsity (≈92.37%).

Spambase (SB) is an open and well-know dataset for spam detection in email body messages [52], [53], [54]. It contains features counting the occurrence of particular words and the length of sequences of consecutive capital letters.

The dataset consists of 4,061 data points (2,788 spam and 1,813 non spam), organized in 57 features. We built a balanced dataset of 3,626 points, by randomly subsampling data points in class spam. After preprocessing, the dataset exhibits a medium-high sparsity but lower than AM (≈77.44%). We finally split the dataset in a training set of 2,901 data points (1,458 spam and 1,433 non spam) and in a test set of 725 instances (355 spam and 370 non spam).

Diabetic Retinopaty Debrecen (DR) is an open dataset for the detection of symptoms of diabetic retinopathy [55]. It contains features extracted from the Messidor image set [56]. All features are numeric and represent either a detected lesion, a descriptive feature of a anatomical part, or an image-level descriptor. The label indicates if an image contains signs of diabetic retinopathy or not.

The dataset consists of 1,151 data points (611 signs of disease and 540 no signs of disease), organized in 19 features. We built a balanced dataset of 1,080 points, by randomly subsampling data points in class signs of disease. After preprocessing, the dataset exhibits a low sparsity but higher than M2 (≈10.41%). We finally split the dataset in a training set of 864 data points (431 signs of disease and 433 no signs of disease) and in a test set of 216 data points (109 signs of disease and 107 no signs of disease).

VI. Experimental Results

We present the accuracy degradation retrieved by our ensemble approach against *label flipping* (Section VI-B), and *zeroing*, *noising*, and *out-of-ranging* (Section VI-C) for datasets M2, AM, SB, and DR in Section V-B. Label flipping is in fact the most effective perturbation substantially affecting the behavior of monolithic model, while *zeroing*, *noising*, *out-of-ranging* do not produce substantial accuracy degradation on it. For readability, we omit the percentage symbol (%) when presenting values of Δ, accuracy, as well as percentage of data points ϵ_p and features ϵ_f to be poisoned.

A. Experimental Settings

Our experiments have been built on the ML library *Weka* [57] version 3.8 running on Java version 8. We executed our process in Section V-A on a VM equipped with 16 vCPUs Intel Core Processor (Broadwell, no TSX) 2.00 GHz and 48 GBs of RAM. The entire process has been executed 5 times averaging accuracy and Δ.

The settings of our experiments varied *i)* the number of random forests N in the ensemble in $\{3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$; *ii)* the perturbations in *zeroing*, *noising*, *out-of-ranging* and *label flipping* (Section IV); *iii)* the percentage of poisoned data points ϵ_p in $[10, 35]$, step 5; and *iv)* the percentage of poisoned features ϵ_f on each data point in $[10, 35]$, step 5. We note that ϵ_f is not applicable to perturbation *label flipping*.

Each combination of these parameters represented an instance of the evaluation process in Section V-A. In addition, we put ourselves in a worst-case scenario using the outdated hash function MD5 to assign data points to partitions, to determine whether our ensemble approach can still provide some protection against poisoning attacks. Finally, we configured random forests according to the well-known practices in the state of the art.[2]

B. Label Flipping

Tables II(a)–(d) show the results retrieved by executing perturbation *label flipping* against datasets M2 (Table II(a)), AM (Table II(b)), SB (Table II(c)), and DR (Table II(d)), varying the percentage of poisoned data points ϵ_p and the number N of random forests in the ensemble. The column with $N = 1$ indicates the monolithic model, while the row with $\epsilon_\mathrm{p} = 0$ (rows with gray background in Tables II(a)–(d)) indicates the accuracy $\mathrm{ACC}(D)$ retrieved from the model trained on the original dataset, that is, the dataset with no poisoned data points. Each cell is divided in two parts. The top-most part reports the Δ in (1), retrieved according to the accuracy of the model trained on the poisoned training set and the one on the original training set. The bottom-most part reports the accuracy $\mathrm{ACC}(\widetilde{D})$ retrieved by the model trained on the poisoned training set.

[2] https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/RandomForest.html

TABLE II
RESULTS OF *LABEL FLIPPING* VARYING NUMBER OF RANDOM FORESTS N AND PERCENTAGE OF POISONED DATA POINTS ϵ_p

		Number of random forests N of the ensemble										
		1	3	5	7	9	11	13	15	17	19	21
Poison. data points ϵ_p (%)	0	0.000 91.872	0.000 91.133	0.000 90.805	0.000 89.901	0.000 90.230	0.000 90.066	0.000 89.984	0.000 89.737	0.000 88.752	0.000 88.423	0.000 88.177
	10	-3.449 88.423	-1.560 89.573	-0.083 90.722	-0.328 89.573	0.493 90.723	0.000 90.066	-0.493 89.491	-0.410 89.327	0.246 88.998	-0.246 88.177	0.821 88.998
	15	-6.650 85.222	-2.874 88.259	-1.314 89.491	-1.149 88.752	-0.247 89.983	-1.232 88.834	-1.560 88.424	0.327 90.066	-3.202 85.550	-0.575 87.849	0.411 88.588
	20	-7.553 84.319	-1.970 89.163	-0.657 90.148	-1.149 88.752	-0.985 89.245	-1.478 88.588	-1.971 88.013	-2.052 87.685	-1.806 86.946	-0.657 87.767	-0.657 87.520
	25	-11.330 80.542	-4.187 86.946	-4.269 86.535	-3.448 86.453	-2.463 87.767	-2.135 87.931	-3.449 86.535	-3.612 86.125	-2.052 86.700	-0.903 87.521	-2.709 85.468
	30	-16.092 75.780	-10.920 80.213	-6.158 84.647	-4.597 85.304	-5.665 84.565	-3.941 86.125	-5.173 84.811	-5.254 84.483	-3.695 85.057	-5.255 83.169	-3.694 84.483
	35	-22.414 69.458	-15.435 75.698	-12.562 78.243	-8.949 80.952	-9.688 80.542	-11.330 78.736	-9.442 80.542	-9.113 80.624	-7.636 81.116	-10.181 78.243	-6.814 81.363

(a) Dataset M2

		Number of random forests N of the ensemble										
		1	3	5	7	9	11	13	15	17	19	21
Poison. data points ϵ_p (%)	0	0.000 98.828	0.000 98.541	0.000 98.449	0.000 98.047	0.000 98.070	0.000 97.886	0.000 97.828	0.000 97.679	0.000 97.483	0.000 97.311	0.000 97.104
	10	-2.344 96.484	-0.402 98.139	-0.184 98.265	-0.023 98.024	0.023 98.093	-0.046 97.840	-0.068 97.760	0.000 97.679	-0.034 97.449	-0.069 97.242	0.081 97.185
	15	-4.447 94.381	-0.943 97.598	-0.333 98.116	-0.150 97.897	-0.161 97.909	-0.069 97.817	-0.321 97.507	-0.230 97.449	-0.367 97.116	0.046 97.357	0.092 97.196
	20	-6.641 92.187	-1.862 96.679	-1.092 97.357	-0.356 97.691	-0.219 97.851	-0.265 97.621	-0.310 97.518	-0.138 97.541	-0.137 97.346	-0.241 97.070	0.161 97.265
	25	-11.100 87.728	-4.332 94.209	-1.999 96.450	-1.494 96.553	-1.230 96.840	-0.644 97.242	-0.505 97.323	-0.138 97.541	-0.471 97.012	-0.195 97.116	-0.252 96.852
	30	-16.052 82.776	-7.136 91.405	-4.148 94.301	-2.620 95.427	-1.873 96.197	-1.218 96.668	-1.597 96.231	-0.655 97.024	-0.723 96.760	-0.655 96.656	-0.574 96.530
	35	-22.578 76.250	-14.536 84.005	-8.595 89.854	-6.067 91.980	-4.355 93.715	-3.562 94.324	-3.630 94.198	-2.298 95.381	-1.964 95.519	-1.884 95.427	-0.942 96.162

(b) Dataset AM

		Number of random forests N of the ensemble										
		1	3	5	7	9	11	13	15	17	19	21
Poison. data points ϵ_p (%)	0	0.000 94.897	0.000 93.425	0.000 92.184	0.000 91.632	0.000 91.402	0.000 91.264	0.000 90.942	0.000 90.575	0.000 90.391	0.000 90.483	0.000 90.115
	10	-3.173 91.724	-0.322 93.103	0.873 93.057	0.138 91.770	0.138 91.540	0.138 91.402	0.644 91.586	0.184 90.759	0.046 90.437	0.276 90.759	0.414 90.529
	15	-4.828 90.069	-0.506 92.919	0.644 92.828	0.736 92.368	0.460 91.862	0.046 91.310	0.598 91.540	0.873 91.448	0.781 91.172	0.322 90.805	0.552 90.667
	20	-6.069 88.828	-1.655 91.770	0.184 92.368	0.552 92.184	0.506 91.908	0.782 92.046	0.414 91.356	1.103 91.678	0.414 90.805	1.011 91.494	0.828 90.943
	25	-8.138 86.759	-3.448 89.977	-0.828 91.356	-0.643 90.989	-0.184 91.218	-0.092 91.172	0.368 91.310	0.413 90.988	0.827 91.218	-0.506 89.977	1.195 91.310
	30	-10.759 84.138	-5.839 87.586	-1.931 90.253	-1.241 90.391	-0.597 90.805	-0.414 90.850	0.414 91.356	-0.138 90.437	-0.184 90.207	0.414 90.897	-0.046 90.069
	35	-17.380 77.517	-9.609 83.816	-5.839 86.345	-3.494 88.138	-2.115 89.287	-2.115 89.151	-1.379 89.563	-0.414 90.161	0.184 90.575	-0.552 89.931	0.046 90.161

(c) Dataset SB

		Number of random forests N of the ensemble										
		1	3	5	7	9	11	13	15	17	19	21
Poison. data points ϵ_p (%)	0	0.000 69.908	0.000 68.827	0.000 67.901	0.000 68.519	0.000 66.821	0.000 66.050	0.000 65.587	0.000 68.210	0.000 65.741	0.000 67.439	0.000 67.593
	10	-1.852 68.056	-0.309 68.518	1.389 69.290	-0.926 67.593	-1.389 65.432	0.000 66.050	0.463 66.050	-1.852 66.358	2.161 67.902	-0.926 66.513	-1.389 66.204
	15	-3.241 66.667	-1.697 67.130	-2.469 65.432	-3.241 65.278	0.154 66.975	-0.309 65.741	0.154 65.741	-3.704 64.506	-0.463 65.278	-0.309 67.130	-2.161 65.432
	20	-6.019 63.889	-6.481 62.346	-4.167 63.736	-3.395 65.124	1.543 68.364	0.000 66.050	-1.544 64.043	-2.315 65.895	1.389 67.130	-4.013 63.426	-2.315 65.278
	25	-9.259 60.648	-5.864 62.963	-2.773 65.124	-5.402 63.117	-2.469 64.352	-1.544 64.506	-2.933 62.654	-3.241 64.969	0.309 66.050	-1.389 66.050	-3.550 64.043
	30	-15.741 54.167	-7.870 60.957	-4.012 63.889	-7.562 60.957	-3.549 63.272	-2.624 63.426	-2.624 62.963	-3.858 64.352	-2.932 62.809	-4.630 62.809	-2.624 64.969
	35	-17.593 52.315	-10.185 58.642	-9.414 58.487	-9.723 58.796	-6.482 60.339	-5.093 60.957	-6.019 59.568	-6.173 62.037	-6.173 59.568	-6.328 61.111	-9.260 58.333

(d) Dataset DR

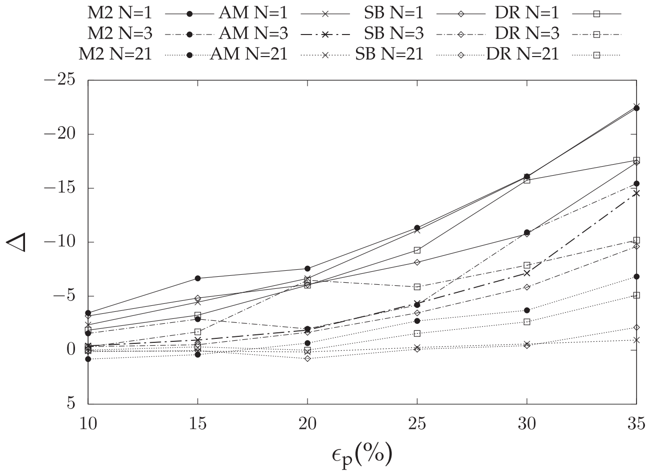


Fig. 3. Results for *label flipping* with monolithic ($N = 1$) and the smallest ($N = 3$) and largest ($N = 21$) ensemble models for datasets M2, AM, SB, and DR.

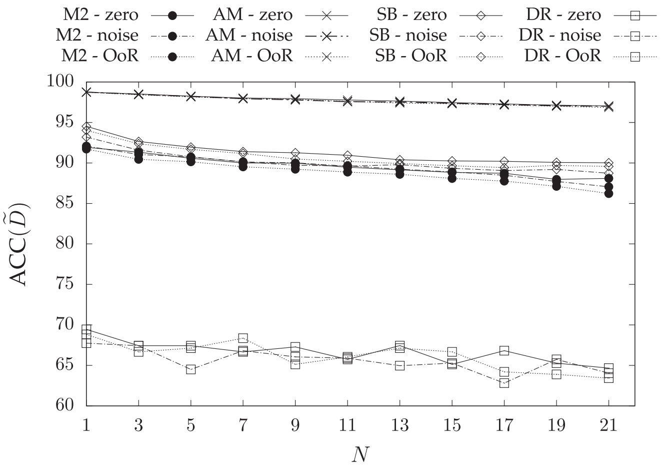


Fig. 4. Results for other attacks averaged over ϵ_p with perturbations *zeroing*, *noising*, and *out-of-ranging* abbreviated as *zero*, *noise*, and *OoR*, respectively.

Our results first show that the accuracy retrieved using the 4 datasets varies significantly. In particular, the monolithic model shows ACC(D) of 91.872 for M2, 98.828 for AM, 94.897 for SB, and 69.908 for DR. These variations in accuracy depends on the diversity of the selected dataset and can be observed in all configurations, reaching the peak with $\epsilon_p = 35$.

Our results additionally show two clear trends. First, as the percentage of poisoned data points increases, the corresponding Δ decreases, that is, the more label flips, the higher the accuracy decrease. This trend can be observed downward column by column. For instance, considering the smallest ensemble $N = 3$. Δ decreases from −1.560 with $\epsilon_p = 10$ to −15.435 with $\epsilon_p = 35$ for M2, from −0.402 to −14.536 for AM, from −0.322 to −9.609 for SP, and from −0.309 to −10.185 for DR. Second, as the number of random forests in the ensemble increases, the corresponding Δ increases, that is, the larger the ensemble, the lesser the accuracy decrease. This trend can be observed rightward row by row. For instance, considering the worst perturbation ($\epsilon_p = 35$), we can see that Δ improves of ≈70% for M2 (increasing from −22.414 with $N = 1$ to −6.814 with $N = 21$), ≈96% for AM (increasing from −22.578 to −0.942), 100% for SB (increasing from −17.380 to 0.046), and ≈53% on DR (increasing from −17.593 to −9.260). It is important to note that these two trend are less pronounced in dataset DR, due to the limited amount of data points in the training set and to the low classification performance of the random forest.

Fig. 3 shows Δ for the monolithic model ($N = 1$) and the smallest ($N = 3$) and largest ($N = 21$) ensembles varying the datasets and the percentage of poisoned data points ϵ_p. As expected, the monolithic model always experience the largest accuracy decrease. The decrease ranges from −3.448 (with $\epsilon_p = 10$) to −22.414 (with $\epsilon_p = 35$) with minimum accuracy of 69.458 for M2; from −2.344 to −22.578 with minimum accuracy 76.520 for AM; from −3.173 to −17.380 with minimum accuracy 77.517 for SB; from −1.852 to −17.593 with minimum accuracy 52.315 for DR. Instead, our ensemble approach shows higher robustness and keeps the accuracy drop under control. This can be noticed even with the smallest ensemble $N = 3$, where $\Delta = -9.225$ with $N = 1$ and $\Delta = -4.740$ with $N = 3$ on average on the 4 datasets. More in detail, when considering dataset M2, Δ increases from −3.448 to −1.560 with $\epsilon_p = 10$, and from −22.414 to −15.435 with $\epsilon_p = 35$ (−6.158 on average with $N = 3$). When considering dataset AM, Δ increases from −2.344 to −0.402 with $\epsilon_p = 10$ and from −22.578 to −14.535 with $\epsilon_p = 35$ (−4.868 on average with $N = 3$). When considering dataset SB, Δ increases from −3.173 to −0.322 with $\epsilon_p = 10$ and from −17.380 to −9.609 with $\epsilon_p = 35$ (−3.563 on average with $N = 3$). When considering dataset DR, Δ increases from −1.852 to −0.309 with $\epsilon_p = 10$ and from −17.593 to −10.185 with $\epsilon_p = 35$ (−5.401 on average with $N = 3$). We can therefore observe that, when the number of random forests N increases, Δ increases too, as Fig. 3 shows.

Finally, when the number of random forests N is greater than 9, Δ improves significantly on all the datasets. This is clear with datasets AM and SB, where Δ are higher than −1 in almost all configurations. A similar trend can be observed in dataset M2 for N=9, though the impact of poisoning (ϵ_p) on the model accuracy is higher: $\Delta = -3.202$ in the worst case with $\epsilon_p \leq 20$, $\Delta = -11.330$ with $\epsilon_p > 20$.

Overall, our results show that plain random forests are sensitive to *label flipping*, but its effect can be easily counteracted using our ensemble approach. In particular, ACC($\widetilde{D}$)≤ACC(D)±1.438 on average with our ensemble approach, ACC($\widetilde{D}$)≤ACC(D)±9.225 with the monolithic model.

C. Other Attacks

Fig. 4 shows the accuracy degradation for perturbations *zeroing*, *noising*, and *out-of-ranging* in Section IV varying the datasets. Δ does not show any major trends and is not presented in Fig. 4, being −0.503 on average (with $\sigma \approx 0.925$).

Our results show two clear trends opposed to the trends retrieved with label flipping. First, *zeroing*, *noising*, and *out-of-ranging* marginally affect the monolithic model, with Δ= −0.401 on average (Δ= −0.158 for *zeroing*, Δ= −0.507 for *noising*, and Δ= −0.539 for *out-of-ranging*). As a consequence,

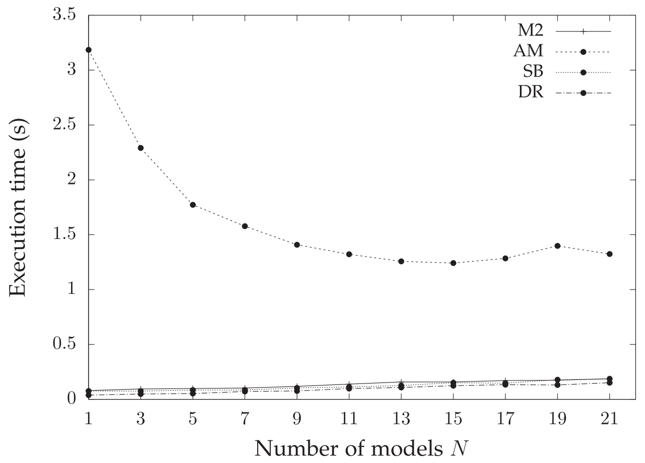


Fig. 5. Execution time for model training varying N and datasets M2, AM, SB, and DR.

there are no major improvements in Δ when using our ensemble approach, with $\Delta = -0.514$ on average ($\Delta = -0.219$ for *zeroing*, $\Delta = -0.562$ for *noising*, and $\Delta = -0.761$ for *out-of-ranging*).

Second, as depicted in Fig. 4, the accuracy decreases as the number of random forests N increases, but with a relatively small average difference of 2.755 between $N = 3$ and $N = 21$. This decrease is higher for M2 in all perturbations with a decrease of 3.818 (from 90.941 with $N = 3$ to 87.123 with $N = 21$, on average), followed by DR with a decrease of 2.898 (from 68.038 with $N = 3$ to 65.140 with $N = 21$, on average), SB with a decrease of 2.802 (from 92.631 with $N = 3$ to 89.829 with $N = 21$, on average), and finally AM with a decrease of 1.501 (from 98.469 with $N = 3$ to 96.968 with $N = 21$, on average).

Overall, our results show that monolithic models are significantly less sensitive to perturbations *zeroing*, *noising*, and *out-of-ranging* than label flipping, with Δ always larger than -0.492. In particular, $\text{ACC}(\hat{D}) \leq \text{ACC}(D) \pm 0.503$ on average.

VII. Sustainability of Our Approach

We evaluated the sustainability of our ensemble approach measuring the execution time (Section VII-B) and resource consumption (Section VII-C).

A. Settings

We recall that our experiments have been executed on a VM equipped with 16 vCPUs Intel Core Processor (Broadwell, no TSX) 2.00 GHz and 48 GBs of RAM. Our experiments varied the number of models in the ensemble in $N \in \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$, the percentage of used data points (i.e., dataset cardinality) in $|D| \in \{10, 25, 50, 75, 100\}$, and the percentage of features in $|f| \in \{10, 25, 50, 75, 100\}$.

We note that, as shown in Fig. 5, the execution time for model training (step 4 in Section V-A) is negligible (less than 0.2 s) when datasets M2, SB, and DR are used, while it is over 2 s in the worst case when dataset AM is used. The execution time of training sets creation (step 3 in Section V-A) is constant for each dataset and does not depend on the number of models N (e.g., 2.46 s, with $\sigma = \pm 10$ms, in the worst case for dataset AM). For these reasons, we evaluated the sustainability of our approach using dataset AM in Table I (worst case scenario). Execution time was evaluated considering only step 4 in Section V-A, while resource consumption considered steps 3 and 4 in Section V-A.

The experiments have been repeated 5 times and the reported results correspond to the average over repetitions.

B. Execution Time

We measured the impact of the training process (step 4 in Section V-A) on the execution time of our approach by varying the number of models N in the ensemble, the dataset cardinality ($|D|$), and the feature cardinality ($|f|$).

Execution Time Varying N: Fig. 5 shows the training execution time according to the number of models N of the ensemble. The execution time is affected by two main dimensions: *i)* the size of the datasets, *ii)* the parallelization approach used to create the training sets and train the ensemble. We recall that $N{=}1$ refers to the monolithic model, while $3 \leq N \leq 21$ to our ensemble approach.

Fig. 5 shows that our approach is sustainable regardless the value of N, and its execution time is 2.29 s in the worst case for $N = 3$ and dataset AM. Such dataset shows a decreasing trend, where the training time is inversely proportional to the number of models N. As such, the monolithic model requires the highest training time (3.18 s). These results are due to: *i)* the capability of our implementation to train the base models in the ensemble in parallel, thus exploiting all cores (16) of the VMs used for the experiments; *ii)* the cardinality of each single training set, which decreases ($|D|/N$) as the number of models N in the ensemble increases. The decreasing trend is more evident for $N{<}9$ in Fig. 5, reaching a time-stability with $N = 9$. With $N \geq 17$, the trend slightly grows because all 16 cores are used, and the thread scheduling affects the performance.

Fig. 5 also shows that the trend followed by datasets M2, SB, and DR is significantly different from the one followed by dataset AM. This is mostly due to the cardinality of the datasets, which is particularly low for dataset M2 (2,034 data points), SB (3,626) and DR (1,080). As a consequence, training cannot fully benefit from parallelism, and training time increases as the number of models N increases, being 0.187 s in the worst case for $N = 21$ on dataset SB.

Execution Time Varying $|D|$: Fig. 6 shows the training execution time varying the cardinality of dataset Android malware. Specifically, we randomly subsampled the dataset to achieve a percentage of data points in $\{10, 25, 50, 75, 100\}$ with respect to the original dataset, and measured the execution time with $N \in \{1, 3, 11, 21\}$. Our results show a sustainable trend that linearly increases for all N with a worst case of 3.246 s obtained with the monolithic model. We note that, up to 25% of the dataset, the training time is almost similar for all N. For higher percentages, the execution time for $N = 1$ and $N = 3$ starts raising more steeply. Comparing $N = 1$ and $N = 3$, we can already appreciate a good gain when our ensemble approach is used, becoming more evident with larger values of N.

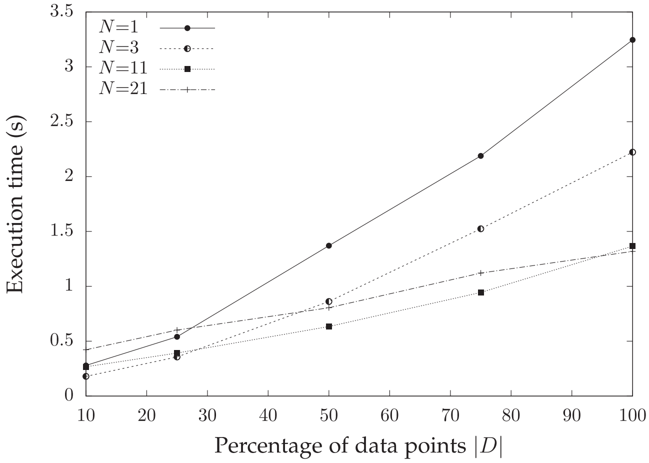


Fig. 6. Execution time varying dataset cardinality.

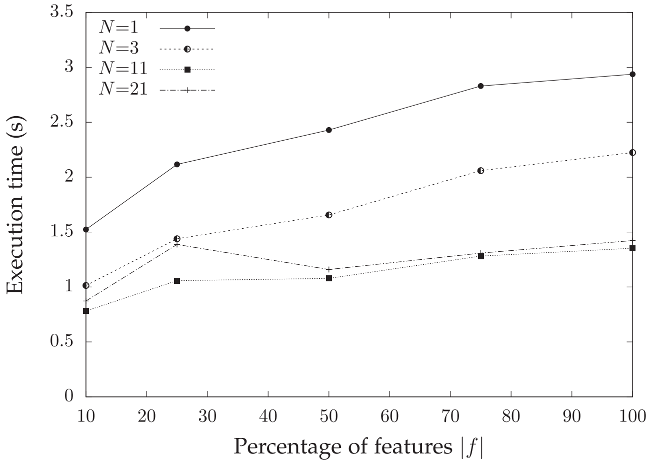


Fig. 7. Execution time varying number of features.

Execution Time Varying $|f|$: Fig. 7 shows the training execution time varying the number of features in dataset AM. Specifically, we randomly subsampled the dataset to achieve a percentage of features in $\{10, 25, 50, 75, 100\}$ with respect to the original dataset, and measured the execution time with $N \in \{1, 3, 11, 21\}$. Our results show a sustainable trend that increases for all N following the equation of a parabola with a worst case of 2.937 s obtained, also in this case, with the monolithic model. As expected, the execution time grows proportionally to the number of features regardless N, while the slope of the corresponding curves are gradually lower as N increases.

C. Resource Consumption

We measured the resource consumption in terms of CPU and RAM usage during activity training (i.e., steps 3–4 in Section V-A), by wrapping the execution of each individual experimental setting in Section VII-A with Linux tool `time`.[3] We measured the CPU user time and the maximum amount of allocated memory; we executed each individual setting 5 times averaging its results. We note that the resource consumption for activity testing and evaluation (i.e., steps 5–6 in Section V-A), including the execution of our approach on a single data point in the test set and the calculation of Δ, is negligible.

CPU Usage: Fig. 8(a), (c), and (e) show the CPU user time. We note that CPU user time is the sum of the time each CPU core spent within the process in the user space. Having implemented a parallel approach, CPU user time is higher than real execution time in Section VII-B. CPU user time is affected mostly by the dimensions (data points and features) of the dataset, growing at worst linearly as the percentage of data points and features increases, varying between ≈9 s and ≈71 s. Fig. 8(e) shows that our ensemble approach increases the CPU user time with respect to the monolithic model, from ≈25 s to ≈45 s in the worst case with $N = 3$. CPU user time remains however stable when N increases, showing that the size of the ensemble and therefore its protection does not substantially affect its sustainability.

Memory Usage: Fig. 8(b), (d), and (f) show the maximum amount of allocated memory for the process. They exhibit similar patterns to the ones of CPU user time, being affected mostly by the dimensions of the dataset and growing at worst linearly with them. It varies between ≈204 MB and ≈4.2 GB. The same is true also for memory consumption varying N, with a significant increase from monolithic model to our ensemble approach, from ≈1 GB to ≈3.75 GB in the worst case. Memory consumption remains however stable when N increases.

In summary, consumed resources are well below the available resources. Despite our ensemble approach introduces an additional overhead with respect to the monolithic model, it can be easily optimized thanks to its vertical and horizontal scalability.

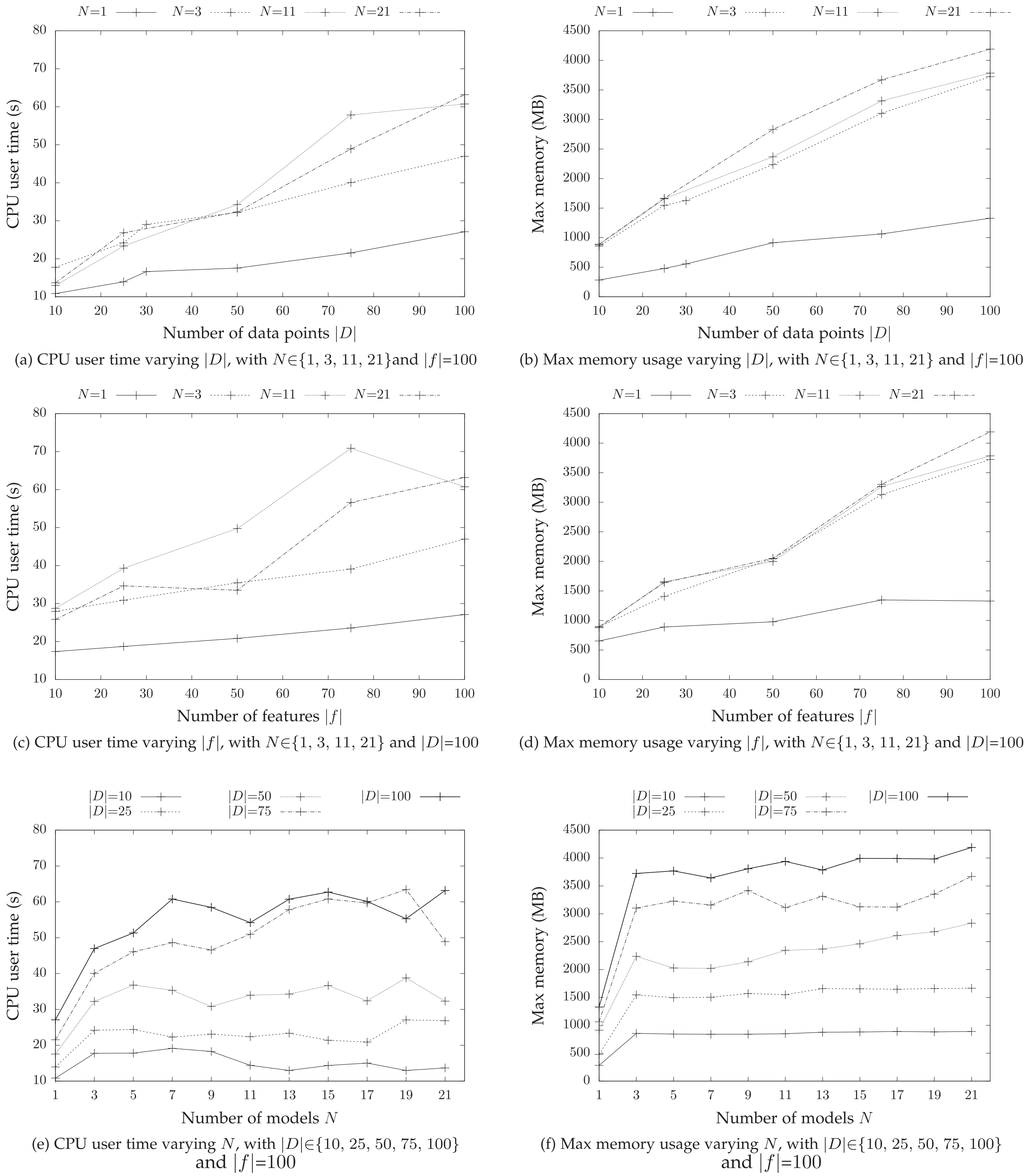
VIII. Discussion

Our main research question focused on investigating the behavior of random forests against poisoning attacks in a scenario where resources are limited on both sides. We evaluated the monolithic model (i.e., ensemble of decision trees) and our ensemble approach (i.e., ensemble of random forests) varying the type (i.e., perturbations) and impact (i.e., ϵ_p and ϵ_f) of poisoning. Our main findings are as follows.

F1 *Monolithic model is highly sensitive to label flipping only:* Finding F1 can be observed in Fig. 3, where the monolithic models ($N = 1$) are significantly worse than our ensemble approach. We also note that Δ of the monolithic models is -9.225 on average under flipping, while it is -0.503 on average under *zeroing*, *noising* and *out-of-ranging*, with a difference of 94.56%. In addition, the accuracy decrease caused by label flipping is proportional with the percentage ϵ_p of poisoned data points. This can be noted by comparing ACC$(\widetilde{D})$ and Δ on the datasets in Table II(a)–(d), where ACC$(\widetilde{D})$ and Δ progressively worsen as ϵ_p increases.

F2 *The effectiveness of perturbations depends also on the characteristics of the dataset and of the ensemble:* Finding F2 can be observed for label flipping by comparing downward the right-hand side of Table II(a)–(d). Being M2 and

[3] https://www.man7.org/linux/man-pages/man1/time.1.html



(a) CPU user time varying $|D|$, with $N\in\{1, 3, 11, 21\}$and $|f|$=100

(b) Max memory usage varying $|D|$, with $N\in\{1, 3, 11, 21\}$ and $|f|$=100

(c) CPU user time varying $|f|$, with $N\in\{1, 3, 11, 21\}$ and $|D|$=100

(d) Max memory usage varying $|f|$, with $N\in\{1, 3, 11, 21\}$ and $|D|$=100

(e) CPU user time varying N, with $|D|\in\{10, 25, 50, 75, 100\}$ and $|f|$=100

(f) Max memory usage varying N, with $|D|\in\{10, 25, 50, 75, 100\}$ and $|f|$=100

Fig. 8. CPU user time and maximum memory on dataset AM, with 14,508 data points and 1,1000 features, varying the number of models (N), percentage of data points ($|D|$), and percentage of features ($|f|$).

DR smaller than AM and SB in terms of cardinality and sparsity, Δ worsens more rapidly as the percentage of poisoned data points increases. In addition, Δ also has a smaller improvement on M2 and DR as N increases, because the cardinality of the individual partitions and training sets is increasingly reduced. Finding F2 can be observed for other perturbations by comparing the trends in Fig. 4. In this case, accuracy worsens at a higher rate in M2 and SB (smaller than AM) as N increases, with out-of-ranging being the most effective perturbation. This worsening trend can be observed also in DR despite its ACC(D) being significantly lower.

F3 *Ensemble of random forests is an adequate protection from untargeted label flipping:* Finding F3 can be observed by comparing the increase of Δ of our ensemble approach with regards to the monolithic model in Table II(a)–(d). For instance, considering dataset AM, accuracy becomes $>$96 with at least $N = 9$ random forests in our approach for ϵ_{p} $\leq$30 of poisoned data points, and keeps increasing slightly with N. Instead, for ϵ_{p} $>$30 of poisoned data points, our approach starts suffering of not-negligible accuracy decreases, being $<$96 in virtually all cases. We note that, for $N > 15$, this decrease can be still considered negligible, being accuracy $\geq$95. In general, the improvement provided by our ensemble approach is significant, as summarized in Fig. 3, where the highest lines corresponding to the monolithic model are always significantly worse than those of our ensemble approach.

F4 *Ensemble of decision trees is an adequate protection from untargeted perturbations zeroing, noising, out-of-ranging:* Finding F4 is a direct consequence of F1 and can be observed by comparing the accuracy of monolithic and ensemble models in Fig. 4. Perturbations *zeroing*, *noising*, and *out-of-ranging* introduce a minor accuracy decrease, which largely fails to make the monolithic model unusable in practice. Accuracy variation of the poisoned monolithic model with regards to the original accuracy is in fact always less than 4.784. This implies that our ensemble approach is redundant in this scenario, and explains the accuracy decrease we observed as N increases. In practice, our approach only reduces the cardinality of the training set of each base model from $|D|$ to $|D|/N$, hence affecting classification accuracy. This tradeoff is advantageous in label flipping, when the accuracy decrease caused by the smaller training set is balanced by containing the accuracy decrease caused by poisoning. It is detrimental for other perturbations where the accuracy decrease caused by poisoning is negligible.

F5 *Our ensemble approach is sustainable.* Finding F5 can be observed by analyzing the growth of resources consumption when N increases. Consumed resources (CPU and RAM) stay mostly stable and are affected by the dimensions (number of data points and features) of the dataset. In addition, our ensemble largely benefits from parallelism. This is noticeable by comparing CPU user time in Fig. 8(a), (c), and (e) with real execution time in Figs. 5, 6, and 7: high CPU user time still corresponds to low execution time. This means that size and therefore robustness of our ensemble approach can be tuned according to the scenario, incurring in a *constant* overhead.

From the above findings, we can conclude that random forest (a native ensemble algorithm) provides an empirically-strong robustness against *zeroing*, *noising*, *out-of-ranging* attacks. When label flipping is considered, an ensemble of random forests is needed to ensure robustness. The size of the ensemble must however be carefully balanced to avoid accuracy decrease due to an oversized ensemble approach. In short, even untargeted poisoning attacks requiring little to none knowledge and resources on the attacker side can be dangerous and untractable with dataset strengthening defenses [58]; these attacks can be rather counteracted with a sustainable model strengthening defense.

TABLE III
COMPARISON WITH RELATED WORK ON DATA POISONING ATTACKS AND DEFENSES AGAINST RANDOM FOREST

Ref.	Poisoning type		Defense type		Threat model			Domain
	Features	Labels	Dataset	Model	Type	Select.	Pert.	
[13]	✗	✓	✗	✗	¬T	¬S	¬S	Single
[33]	✗	✓	✓	✗	¬T	S	S	Single
[28]	✗	✓	✗	✗	T	S	S	Single
[29]	✓	✗	✗	✗	¬T	¬S	S	Single
[30]	✓	✗	✗	✗	¬T	¬S	S	Single
[31]	✓	✗	✗	✗	¬T	¬S	S	Single
[32]	✗	✓	✗	✗	¬T	¬S,S	¬S,S	Multi
[34]	✗	✓	✓	✗	¬T	S	S	Single
[48]	✗	✓	✗	✗	¬T	¬S	¬S	Single
This	✓	✓	✗	✓	¬T	¬S	S	Multi

Threat model type (*Type*): Untargeted $\neg T$ or targeted T; Selection strategy (*Select.*): Random$\neg S$ or with a specific strategy S; Perturbation strategy (*Pert.*): (Random $\neg S$ or with a specific strategy S)

Finally, we note that the results in this paper, although novel, are in line with other work on ML robustness, for instance [13], [20], claiming that *i)* if the amount of poisoning is reasonable, an ensemble strategy can reduce the influence of poisoned data points to the resulting model, and *ii)* random forests are, in some cases, more robust than other models (e.g., naive bayes, neural networks) [13], [28], [32].

IX. COMPARISON WITH THE STATE OF THE ART

Random forest, being one of the most popular algorithms for tabular datasets, has been studied from different angles including: *i)* explainability [25], [59], [60], [61], [62], [63]; *ii)* fairness [26], [64], [65], [66]; *iii)* sustainability [27], [67], [68], [69], [70]; and *iv)* robustness [13], [28], [29], [30], [31], [32], [33], [34], [48].

In terms of robustness, different solutions have been defined on random forest, though none of them can be easily experimentally compared against the ensemble approach in this paper. Table III shows how these solutions compares with our approach in terms of *Poisoning type*, the portion of the data points (features or label) affected by poisoning; *Defense type*, the type of defense, either dataset or model strengthening; *threat model*, the model of the attacker capabilities. The latter is composed of three main dimensions: *i)* the threat model type (untargeted $\neg T$ or targeted T), denoted as *Type*; *ii)* the strategy used to select the data points to be poisoned (random $\neg S$ or with a specific strategy S), denoted as *Select.*; and *iii)* the strategy used to poison the selected data points (random $\neg S$ or with a specific strategy S), denoted as *Pert.*

Most of the surveyed approaches have considered poisoning attacks (no defenses) against standard random forests in a single domain [13], [28], [29], [30], [31], [48], with few papers evaluating different datasets from different domains [32]. In addition, most evaluations consider attacks affecting labels only [13], [28], [32], [33], [34], [48]. Few defenses have been proposed and evaluated, whose threat model implements label flipping attack against data points selected according to a specific strategy. Taheri et al. [33] presented two dataset strengthening defenses in the domain of Android malware detection, based on healing suspicious data points according to label propagation and clustering. Shahid et al. [34] proposed a dataset strengthening defense in the domain of human activity recognition from sensors data. The latter is based on the approach by Paudice et al. [41], where a clustering model trained on a trusted dataset is used to heal suspicious data points. We also note that there exist some approaches where random forest is part of the defense such as in [33], [71].

The ensemble defense in this paper (last row in Table III) departed from assumptions and configurations in the state of the art, making a comparative experimental evaluation meaningless. First, our threat model considers untargeted attacks, where data points are selected according to a random strategy and poisoning affects both labels and features. Second, our approach strengthens the ML model rather than the training set. Third, our approach focused on significantly different domains and datasets, rather than on a single domain with one [34] or more [33] datasets.

X. Conclusions

Machine learning models play an increasingly vital role in the digital services we interact with. As a consequence, the need for properly securing such models from attacks is a key issue being investigated by the research community. This paper aimed to shed new light on the usage of ensembles as a means of protecting random forests against poisoning attacks. While ensembles have been already proposed in the context of certified protection in the domain of image recognition, little has been done in the context of random forests. Throughout fine-grained experiments, we show that label flipping, even if performed with no strategy, is a very dangerous type of perturbation, significantly degrading the performance of plain random forests. A simple yet effective and sustainable countermeasure consists in training models on disjoint training sets, then aggregating their predictions with majority voting. Other perturbations are less effective, and random forest is already an effective countermeasure. The paper leaves space for future work. First, we plan to enrich our set of perturbations with targeted attacks, including backdoor poisoning. Second, we plan to fine-tune the hyperparameters of random forest to find relevant trends with respect to the considered threat model and datasets. Third, we plan to develop a complete benchmark considering the new perturbations and different hash function.

References

[1] F. Rundo, F. Trenta, A. L. di Stallo, and S. Battiato, “Machine learning for quantitative finance applications: A. survey,” *Appl. Sci.*, vol. 9, no. 24, 2019.

[2] E. Hossain, I. Khan, F. Un-Noor, S. S. Sikander, and M. S. H. Sunny, “Application of Big Data and machine learning in smart grid, and associated security concerns: A. review,” *IEEE Access*, vol. 7, pp. 13960–13988, 2019.

[3] R. Chen, W. Zhang, and X. Wang, “Machine learning in tropical cyclone forecast modeling: A review,” *Atmosphere*, vol. 11, no. 7, 2020.

[4] C. Mio and G. Gianini, “Signal reconstruction by means of embedding, clustering and autoencoder ensembles,” in *Proc. IEEE Symp. Comput. Commun.*, Barcelona, Spain, 2019, pp. 1–6.

[5] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, “Machine learning applications in cancer prognosis and prediction,” *Comput. Struct. Biotechnol. J.*, vol. 13, pp. 8–17, 2015.

[6] P. Katrakazas, A. Ballas, M. Anisetti, and I. Spais, “An artificial intelligence outlook for colorectal cancer screening,” in *Proc. IEEE 8th Int. Conf. Big Data Comput. Serv. Appl.*, San Francisco, CA, USA, 2022, pp. 66–72.

[7] J. G. Richens, C. M. Lee, and S. Johri, “Improving the accuracy of medical diagnosis with causal machine learning,” *Nature Commun.*, vol. 11, no. 1, Aug. 2020.

[8] J. Y. Chang and E. G. Im, “Data poisoning attack on random forest classification model,” in *Proc. Int. Conf. Smart Media Appl.*, 2020.

[9] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” 2017, *arXiv: 1712.05526*.

[10] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, “Systematic poisoning attacks on and defenses for machine learning in healthcare,” *IEEE J. Biomed. Health Inform.*, vol. 19, no. 6, pp. 1893–1905, Nov. 2015.

[11] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, “You autocomplete me: Poisoning vulnerabilities in neural code completion,” in *Proc. USENIX Secur.*, 2021, pp. 1559–1575.

[12] B. Biggio, B. Nelson, and P. Laskov, “Support vector machines under adversarial label noise,” in *Proc. Asian Conf. Mach. Learn.*, 2011, pp. 97–112.

[13] C. Dunn, N. Moustafa, and B. Turnbull, “Robustness evaluations of sustainable machine learning models against data poisoning attacks in the Internet of Things,” *Sustainability*, vol. 12, no. 16, 2020.

[14] C. Frederickson, M. Moore, G. Dawson, and R. Polikar, “Attack strength versus detectability dilemma in adversarial machine learning,” in *Proc. Int. Joint Conf. Neural Netw.*, 2018, pp. 1–8.

[15] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart, “Sever: A robust meta-algorithm for stochastic optimization,” in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, 2019, pp. 1596–1606.

[16] Y. Ma, X. Zhu, and J. Hsu, “Data poisoning against differentially-private learners: Attacks and defenses,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macao, China, 2019, pp. 4732–4738.

[17] E. Rosenfeld, E. Winston, P. Ravikumar, and Z. Kolter, “Certified robustness to label-flipping attacks via randomized smoothing,” in *Proc. 37th Int. Conf. Mach. Learn.*, Virtual, 2020, pp. 8230–8241.

[18] N. Peri et al., “Deep k-NN defense against clean-label data poisoning attacks,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 50–55.

[19] P. W. Koh, J. Steinhardt, and P. Liang, “Stronger data poisoning attacks break data sanitization defenses,” *Mach. Learn.*, vol. 111, no. 1, pp. 1–47, 2021.

[20] J. Jia, X. Cao, and N. Z. Gong, “Intrinsic certified robustness of bagging against data poisoning attacks,” in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 7961–7969.

[21] A. Levine and S. Feizi, “Deep partition aggregation: Provable defenses against general poisoning attacks,” in *Proc. Int. Conf. Learn. Representations*, Vienna, Austria, 2021.

[22] W. Wang, A. Levine, and S. Feizi, “Improved certified defenses against data poisoning with (Deterministic) finite aggregation,” in *Proc. Int. Conf. Mach. Learn.*, Baltimore, MD, USA, 2022, pp. 22769–22783.

[23] R. Chen, Z. Li, J. Li, J. Yan, and C. Wu, “On collective robustness of bagging against data poisoning,” in *Proc. Int. Conf. Mach. Learn.*, Baltimore, MD, USA, 2022, pp. 3299–3319.

[24] L. Grinsztajn and G. V. Edouard Oyallon, “Why do tree-based models still outperform deep learning on typical tabular data?,” in *Proc. Adv. Neural Inf. Process. Syst.*, New Orleans, LA, USA, 2022, pp. 507–520.

[25] M. T. Ribeiro, S. Singh, and C. Guestrin, ““why should I trust you?”: Explaining the predictions of any classifier,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, San Francisco, CA, USA, 2016, pp. 1135–1144.

[26] Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, “MAAT: A novel ensemble approach to addressing fairness and performance bugs for machine learning software,” in *Proc. 30th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Singapore, 2022, pp. 1122–1134.

[27] A. Prasad, S. Rajendra, K. Rajan, R. Govindarajan, and U. Bondhugula, “Treebeard: An optimizing compiler for decision tree based ML inference,” in *Proc. IEEE/ACM 55th Int. Symp. Microarchitecture*, Chicago, IL, USA, 2022, pp. 494–511.

[28] H. Zhang, N. Cheng, Y. Zhang, and Z. Li, “Label flipping attacks against naive bayes on spam filtering systems,” *Appl. Intell.*, vol. 51, no. 7, pp. 4503–4514, 2021.

[29] L. Verde, F. Marulli, and S. Marrone, “Exploring the impact of data poisoning attacks on machine learning model reliability,” *Procedia Comput. Sci.*, vol. 192, pp. 2624–2632, 2021.

[30] K. Talty, J. Stockdale, and N. D. Bastian, “A sensitivity analysis of poisoning and evasion attacks in network intrusion detection system machine learning models,” in *Proc. IEEE Mil. Commun. Conf.*, San Diego, CA, USA, 2021, pp. 1011–1016.

[31] A. Prud–Homme and B. Kantarci, “Poisoning attack anticipation in mobile crowdsensing: A competitive learning-based study,” in *Proc. 3rd ACM Workshop Wireless Secur. Mach. Learn.*, Abu Dhabi, UAE, 2021, pp. 73–78.

[32] F. A. Yerlikaya and Ş. Bahtiyar, “Data poisoning attacks against machine learning algorithms,” *Expert Syst. Appl.*, vol. 208, 2022, Art. no. 118101.

[33] R. Taheri, R. Javidan, M. Shojafar, Z. Pooranian, A. Miri, and M. Conti, “On defending against label flipping attacks on malware detection systems,” *Neural Comput. Appl.*, vol. 32, no. 18, pp. 14781–14800, Sep. 2020.

[34] A. R. Shahid, A. Imteaj, P. Y. Wu, D. A. Igoche, and T. Alam, “Label flipping data poisoning attack against wearable human activity recognition system,” 2022, *arXiv:2208.08433*.

[35] E. Damiani and C. A. Ardagna, “Certified machine-learning models,” in *Proc. Int. Conf. Curr. Trends Theory Pract. Inform.*, Limassol, Cyprus, 2020, pp. 3–15.

[36] M. Anisetti, C. A. Ardagna, E. Damiani, and P. G. Panero, “A methodology for non-functional property evaluation of machine learning models,” in *Proc. 12th Int. Conf. Manage. Digit. EcoSystems*, Abu Dhabi, UAE, 2020, pp. 38–45.

[37] L. Mauri and E. Damiani, “Estimating degradation of machine learning data assets,” *J. Data Inf. Qual.*, vol. 14, no. 2, pp. 1–15, Dec. 2021.

[38] A. E. Cinà et al., “Wild patterns reloaded: A survey of machine learning security against training data poisoning,” *ACM Comput. Surv.*, 2023.

[39] C. Szegedy et al., “Intriguing properties of neural networks,” in *Proc. Int. Conf. Learn. Representations*, Banff, Canada, 2014.

[40] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” in *Proc. Int. Conf. Learn. Representations*, Toulon, France, 2017.

[41] A. Paudice, L. Muñoz-González, and E. C. Lupu, “Label sanitization against label flipping poisoning attacks,” in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases Workshops*, Dublin, Ireland, 2018, pp. 5–15.

[42] A. Shafahi et al., “Poison frogs! targeted clean-label poisoning attacks on neural networks,” in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, Montréal, Canada, 2018, pp. 6106–6116.

[43] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, “Casting out demons: Sanitizing training data for anomaly sensors,” in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, 2008, pp. 81–95.

[44] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, “The security of machine learning,” *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.

[45] A. Prasad, A. S. Suggala, S. Balakrishnan, and P. Ravikumar, “Robust estimation via robust gradient estimation,” *J. Roy. Stat. Soc.: Ser. B.*, vol. 82, no. 3, pp. 601–627, 2020.

[46] B. Biggio et al., “Evasion attacks against machine learning at test time,” in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, Prague, Czech Republic, 2013, pp. 387–402.

[47] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, “On the effectiveness of mitigating data poisoning attacks with gradient shaping,” 2020, *arXiv: 2002.11497*.

[48] K. Aryal, M. Gupta, and M. Abdelsalam, “Analysis of label-flip poisoning attack on machine learning based malware detector,” in *Proc. IEEE Int. Conf. Big Data*, Osaka, Japan, 2022, pp. 4236–4245.

[49] Eurpean Union Agency for Cybersecurity, “ENISA threat landscape 2022,” Eurpean Union Agency Cybersecurity, Tech. Rep., 2022.

[50] D. Dua and C. Graff, “UCI machine learning repository,” 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[51] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[52] D. R. Hush, C. Scovel, and I. Steinwart, “Stability of unstable learning algorithms,” *Mach. Learn.*, vol. 67, no. 3, pp. 197–206, 2007.

[53] Y. Wang and I. H. Witten, “Modeling for optimal probability prediction,” in *Proc. Int. Conf. Mach. Learn.*, Sidney, Australia, 2002, pp. 650–657.

[54] C. Dimitrakakis and S. Bengio, “Online policy adaptation for ensemble classifiers,” in *Proc. Eur. Symp. Artif. Neural Netw.*, Bruges, Belgium, 2004.

[55] B. Antal and A. Hajdu, “An ensemble-based system for automatic screening of diabetic retinopathy,” *Knowl. Based Syst.*, vol. 60, pp. 20–27, 2014.

[56] E. Decencière et al., “Feedback on a publicly distributed image database: The messidor database,” *Image Anal. Stereol.*, vol. 33, no. 3, pp. 231–234, 2014.

[57] M. A. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: An update,” *SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[58] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, “Detection of adversarial training examples in poisoning attacks through anomaly detection,” 2018, *arXiv: 1802.03041*.

[59] Y. Lou, R. Caruana, and J. Gehrke, “Intelligible models for classification and regression,” in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Beijing, China, 2012, pp. 150–158.

[60] H. Chipman, E. George, and McCulloch, “Making sense of a forest of trees,” *Comput. Sci. Statist.*, 1998.

[61] H. Deng, “Interpreting tree ensembles with intrees,” *Int. J. Data Sci. Anal.*, vol. 7, no. 4, pp. 277–287, Jun. 2019.

[62] R. R. Fernández, I. Martín de Diego, V. Aceña, A. Fernández-Isabel, and J. M. Moguerza, “Random forest explainability using counterfactual sets,” *Inf. Fusion*, vol. 63, pp. 196–207, 2020.

[63] F. Gossen and B. Steffen, “Algebraic aggregation of random forests: Towards explainability and rapid evaluation,” *Int. J. Softw. Tools Technol. Transfer*, 2021.

[64] W. Zhang, A. Bifet, X. Zhang, J. C. Weiss, and W. Nejdl, “FARF: A fair and adaptive random forests classifier,” in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2021, pp. 245–256.

[65] P. J. Kenfack, A. M. Khan, S. A. Kazmi, R. Hussain, A. Oracevic, and A. M. Khattak, “Impact of model ensemble on the fairness of classifiers in machine learning,” in *Proc. Int. Conf. Appl. Artif. Intell.*, Halden, Norway, 2021, pp. 1–6.

[66] U. Gohar, S. Biswas, and H. Rajan, “Towards understanding fairness and its composition in ensemble machine learning,” in *Proc. IEEE/ACM Int. Conf. Softw. Eng.*, Melbourne, Australia, 2023.

[67] J. Browne, D. Mhembere, T. M. Tomita, J. T. Vogelstein, and R. Burns, “Forest packing: Fast parallel, decision forests,” in *Proc. SDM 2019*, Calgary, Canada, 2019.

[68] A. H. Peterson and T. R. Martinzed, “Reducing decision tree ensemble size using parallel decision dags,” *Int. J. Artif. Intell. Tools*, vol. 18, no. 4, pp. 613–620, 2009.

[69] A. Joly, F. Schnitzler, P. Geurts, and L. Wehenkel, “L1-based compression of random forest models,” in *Proc. ESANN*, Bruges, Belgium, 2012.

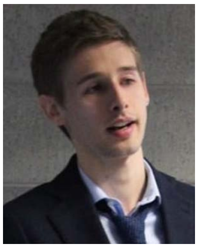
[70] A. Painsky and S. Rosset, “Lossless compression of random forests,” *J. Comput. Sci. Technol.*, vol. 34, no. 2, pp. 494–506, 2019.

[71] Y. Ding, L. Wang, H. Zhang, J. Yi, D. Fan, and B. Gong, “Defending against adversarial attacks using random forest,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, Long Beach, CA, USA, 2019, pp. 105–114.

Marco Anisetti (Senior Member, IEEE) is full professor with the Università degli Studi di Milano. His research interests are in the area of computational intelligence, and its application to the design and evaluation of complex systems. He has been investigating innovative solutions in the area of cloud security assurance evaluation. In this area he defined a new scheme for continuous and incremental cloud security certification, based on distributed assurance evaluation architecture.

Claudio A. Ardagna (Senior Member, IEEE) is full professor with the Università degli Studi di Milano, the director of the CINI National Lab on Big Data, and co-founder of Moon Cloud srl. His research interests are in the area of cloud-edge security and assurance, and data science. He has published more than 140 contributions in international journals, conference/workshop proceedings, and chapters in international books. He has been visiting professor with Université Jean Moulin Lyon 3 and visiting Researcher with Beijing University of Posts and Telecommunications, Khalifa University, George Mason University. He is member of the Steering Committee of *IEEE Transactions on Cloud Computing*, member of the editorial board of *IEEE Transactions on Cloud Computing* and *IEEE Transactions on Services Computing*, and secretary of the *IEEE Technical Committee on Services Computing*.

Alessandro Balestrucci received the PhD degree in computer science from the Gran Sasso Science Institute and qualification of IT engineer, Ca'Foscari University of Venice. He is a postdoctoral researcher with CINI. His research interests are in the area of security of intelligent systems and applied artificial intelligence to disinformation prevention and social network analysis. He has participated/is participating to national and European projects (H2020) cooperating with IMT Lucca, CNR, and University of Milan.



Nicola Bena (Graduate Student Member, IEEE) is currently working toward the PhD degree with the Università degli Studi di Milano. His research interests are in the area of security of modern distributed systems with particular reference to certification, assurance, and risk management techniques. He has participated/is participating to several national and European projects, including H2020 Project CONCORDIA, one of the four European projects aimed to establish the European Cybersecurity Competence Network. He has been visiting scholar at Khalifa University and INSA Lyon.

Ernesto Damiani (Senior Member, IEEE) is full professor with the Department of Computer Science, Università degli Studi di Milano, where he leads the Secure Service-oriented Architectures Research (SESAR) Laboratory. He is also the Founding Director of the Center for Cyber-Physical Systems, Khalifa University, United Arab Emirates. He received an Honorary Doctorate from Institute National des Sciences Appliquées de Lyon, France, in 2017, for his contributions to research and teaching on Big Data analytics. He serves as editor in Chief for *IEEE Transactions on Services Computing*. His research interests include cybersecurity, Big Data, and cloud/edge processing, and he has published more than 680 peer-reviewed articles and books. He is a distinguished scientist of ACM and was a recipient of the 2017 Stephen Yau Award.

Chan Yeob Yeun (Senior Member, IEEE) received the MSc and PhD degrees in information security from Royal Holloway, University of London, in 1996 and 2000, respectively. After the PhD degree, he joined Toshiba TRL, Bristol, U.K., and later, he became the vice president with the Mobile Handset Research and Development Center, LG Electronics, Seoul, South Korea, in 2005. He was responsible for developing mobile TV technologies and related security. He left LG Electronics, in 2007, and joined ICU (merged with KAIST), South Korea, until August 2008, and then the Khalifa University of Science and Technology, in September 2008. He is currently a researcher in cybersecurity, including the IoT/USN security, cyber-physical system security, cloud/fog security, and cryptographic techniques, an associate professor with the Department of Electrical Engineering and Computer Science, and the Cybersecurity Leader of the Center for Cyber-Physical Systems (C2PS). He also enjoys lecturing for MSc degree in cyber security and PhD degree in engineering courses with Khalifa University. He has published more than 140 journal articles and conference papers, nine book chapters, and ten international patent applications. He also works on the editorial board of multiple international journals and on the steering committee of international conferences.