Enabling Privacy-Preserving Cyber Threat Detection with Federated Learning

Yu Bi[1], Yekai Li[1], Xuan Feng[2], and Xianghang Mi[*,1]

[1]University of Science and Technology of China
[2]Microsoft Research Asia

Abstract—Despite achieving good performance and wide adoption, machine learning based security detection models (e.g., malware classifiers) are subject to concept drift and evasive evolution of attackers, which renders up-to-date threat data as a necessity. However, due to enforcement of various privacy protection regulations (e.g., GDPR), it is becoming increasingly challenging or even prohibitive for security vendors to collect individual-relevant and privacy-sensitive threat datasets, e.g., SMS spam/non-spam messages from mobile devices. To address such obstacles, this study systematically profiles the (in)feasibility of federated learning for privacy-preserving cyber threat detection in terms of effectiveness, byzantine resilience, and efficiency. This is made possible by the build-up of multiple threat datasets and threat detection models, and more importantly, the design of realistic and security-specific experiments.

We evaluate FL on two representative threat detection tasks, namely SMS spam detection and Android malware detection. It shows that FL-trained detection models can achieve a performance that is comparable to centrally trained counterparts. Also, most non-IID data distributions have either minor or negligible impact on the model performance, while a label-based non-IID distribution of a high extent can incur non-negligible fluctuation and delay in FL training. Then, under a realistic threat model, FL turns out to be adversary-resistant to attacks of both data poisoning and model poisoning. Particularly, the attacking impact of a practical data poisoning attack is no more than 0.14% loss in model accuracy. Regarding FL efficiency, a bootstrapping strategy turns out to be effective to mitigate the training delay as observed in label-based non-IID scenarios.

I. Introduction

Machine learning especially deep learning has been increasingly adopted in the cyber security domain, along with good performance achieved in various cyber threat detection tasks, e.g., malware detection [1], [2], spam filtering [3], [4], [5], [6], [7], malicious network traffic detection [8], [9]. However, the power of machine learning is hindered by several security-specific critical factors. First of all, it is becoming more and more challenging for security vendors to collect individual-relevant and privacy-sensitive threat datasets (e.g., SMS spam/non-spam messages, and dynamic runtime logs of Android apps), which are essential for building up robust and up-to-date machine learning models. This is partly due to the enforcement of various regulations on data protection and privacy, such as General Data Protection Regulation (GDPR) [10] and California Consumer Privacy Act (CCPA) [11]. These regulations aim to enhance individuals' control and rights over

*Corresponding author.

their personal data, along with more limitations enforced for data controllers and processors including security vendors. Then, following these privacy regulations, computing platforms (e.g., Android and iOS) have updated their developer policies along with more restrictions on collecting and uploading on-device data [12]. What makes things even worse is that the robustness of cyber threat detection models largely relies on the freshness of the training dataset. This is because of the issue of concept drift [13], [14], i.e., the aging of the detection models as the attackers keep evolving their malicious payloads to escape from the detection radar. Also, the inability of collecting up-to-date threat data has already led to degraded performance for some critical security detection tasks, e.g., SMS spam filtering [15].

Another factor resides in the curse of isolated data islands. A typical security vendor will usually devote much effort to collecting cyber threat data from various sources, and regular organizations (e.g., governments) can also observe cyber threats during their security operation, such as DDoS attacks targeting their network infrastructure or spam emails targeting their employees. In many cases, sharing such threat data across organizations can help build up a more capable threat detection model. However, the curse of isolated data islands is there, as such threat incidents, especially ones observed during security operations could contain sensitive information relevant to either the involved organizations or individuals. Sharing such threat data will likely incur non-negligible risks for the involved parties (e.g., compromising an organization's reputation or business interests).

To address these obstacles, we explore in this study the (in)feasibility of federated learning for privacy-preserving cyber threat detection, which, to the best of our knowledge, is among the first-of-its-kind works (see §II for more discussion). Federated learning (FL), as a distributed learning paradigm, has been well explored in multiple privacy-sensitive domains, e.g., medical imaging understanding [16] and the next-word prediction task in virtual keyboards [17]. In a nutshell, FL allows distributed clients (data owners) to collaboratively train a machine learning model without revealing their local datasets to any parties. Despite achieving promising performance results in some privacy-sensitive machine learning tasks, FL is also known to be vulnerable to adversarial machine learning attacks, especially poisoning attacks [18], [19], [20], [21], [22], [23], [24]. Therefore, this study focuses on the following

research questions. First of all, *how effective can FL be when applied to cyber threat detection tasks?* In another word, compared to centrally-trained counterparts, what performance can FL-trained threat detection models achieve, especially in security-specific realistic settings? Then, compared to non-security prediction models, the FL training process for threat detection models can be subject to more adversarial attacks, i.e., various poisoning attacks. Therefore, the secondary research question is *how adversary-resistant the FL training process can be when evaluated under a realistic threat model.* In addition to effectiveness and adversarial resistance, one more research question we aim to understand is that *how (in)efficient the FL training process can be when applied to cyber threat detection tasks.*

To pursue aforementioned research questions, We have encountered and addressed several challenges. Firstly, we need to decide what threat detection tasks to target. A qualified task should satisfy the following three criteria: 1) It should be a well-known and representative threat detection task; 2) It involves privacy-sensitive scenarios, e.g., collecting data from end users; 3) It may gain benefits through adopting FL. Following these criteria, SMS spam detection and Android malware detection, two binary classification tasks, are chosen, for which, detailed explanations will be given in §III. Briefly, the SMS spam detection task requires collection of privacy-sensitive SMS spam/non-spam messages, which can thus benefit from FL. On the other hand, many Android malware detection techniques rely on dynamic features that are extracted from runtime logs of Android apps. Such runtime logs are critical but costly to collect on the server, but are abundant on end-user devices. However, similar to SMS messages, collecting app runtime logs from end-user devices is privacy-sensitive. Then, given the threat detection tasks, one more issue we encountered is that either the datasets or the classification algorithms evaluated in previous works are outdated. For instance, when training and evaluating the proposed SMS spam detection models, most previous studies used only the UCI SMS spam dataset [25] which was published more than 10 years ago and contains only hundreds of SMS spam messages. Also, although transformer-based language models have achieved state-of-the-art performance in many NLP tasks [26], [27], previous works in SMS spam detection failed to consider such kinds of latest machine learning advancements. On the other hand, most studies in Android malware detection differ in their evaluation datasets, which impedes a direct and objective comparison among these studies. Therefore, we addressed this issue through collecting up-to-date and large-scaled threat datasets as well as considering the latest progress in machine learning when building up threat detection models.

Given threat datasets and machine learning algorithms decided, the third challenge stands out as what FL experiments to conduct so as to evaluate FL in a realistic and security-specific manner. To achieve this goal, multiple novel experiments have been designed and implemented, which take security-specific scenarios and realistic threat models into considera-

tion. Particularly, we observe that FL clients in threat detection tasks tend to follow various non-IID data distributions. For instance, FL clients in the SMS spam detection task can vary a lot in terms of the natural languages of their local SMS messages, while the Android malware detection task may involve a non-IID distribution in terms of malware families across FL clients. Besides, different from non-security prediction tasks, FL clients in threat detection tasks may consistently have samples of one class outweigh samples of the other class, e.g., all FL clients have much more benign SMS messages than spam messages. We name this scenario as *the consistent label imbalance (CLI)* scenario. Therefore, when evaluating the effectiveness of FL, we focus on various non-IID distributions of the threat samples across FL clients, rather than exhaustively exploring FL hyperparameters. Also, when evaluating the adversary resistance of FL, we define a realistic threat model and dedicate our evaluations to practical adversary settings, e.g., the fraction of compromised FL clients M is considered as practical only when $M \leq 5\%$ for data poisoning and $M \leq 1\%$ for model poisoning.

Next, we highlight some key observations and findings as distilled from our FL experiments. First, regarding the effectiveness of FL, we found that cyber threat detection models trained via FL can achieve a performance that is comparable to that of their centrally trained counterparts. For instance, the spam detection model trained via cross-device FL has achieved a recall of 99.03% and a precision of 99.05% , while it is 98.79% and 99.23% respectively for the centrally trained counterpart. Then, in realistic FL deployments, there can be various non-IID data distributions across FL clients. And our evaluations show that non-IID data distributions can lead to high instability during the convergence of the FL training process, while the impact on the model performance is either negligible or minor. Particularly, a high degree of quantity-based non-IID distribution can even yield better model performance and faster training convergence, while a high degree of label-based non-IID distribution has no obvious impact on model performance but can incur notable fluctuations (i.e., unstable convergence) in FL training, especially for the cross-device FL. Then, when it comes to the consistent label imbalance scenario that is specific to cyber threat detection tasks, we observed that CLI towards positive (malicious) samples can incur a non-negligible degradation in performance, while the performance impact of CLI towards negative samples (more practical in cross-device FL) is minor.

Regarding the adversarial resistance of FL (§VI), we observed that data poisoning with a practical fraction of poisoned clients ($\leq 5\%$) has a *negligible* attack impact of up to 0.14% decrease in model accuracy for both the SMS spam detection task and the Android malware detection task. On the other hand, for model poisoning with a practical fraction of poisoned clients ($\leq 1\%$), the attack impact is also *negligible* for Android malware detection as well as being minor for SMS spam detection with up to 1.52% decrease in accuracy. Furthermore, the minor performance impact of practical model poisoning can be addressed through the deployment of robust aggregation

rules (e.g., Trimmed Mean [28] and Multi-Krum [18]). For instance, Trimmed Mean has decreased the attack impact of model poisoning on SMS spam detection to almost zero (0.09%). Then, regarding the efficiency of FL, we found out that non-IID data distributions can incur a significant delay in convergence, to address which, a bootstrapping strategy has been proposed with its effectiveness well demonstrated (§VII).

Our key contributions are two-fold. On one hand, we have systematically evaluated the effectiveness, byzantine resilience, and efficiency of FL for two representative cyber threat detection tasks, which features security-specific and realistic FL experiments. On the other hand, our evaluation has distilled a set of novel findings regarding the pros and cons of FL when applied to privacy-sensitive threat detection tasks, which we believe can benefit future efforts of fostering FL-based privacy-preserving cyber threat detection.

II. BACKGROUND AND RELATED WORKS

Federated learning. As an emerging distributed learning paradigm, federated learning (FL) [29] enables distributed FL clients (i.e., mobile devices) to jointly train a global deep learning model without the necessity of sharing their local privacy-sensitive data. In recent years, FL has been explored in multiple privacy-sensitive areas [17], [30], [16], [31], [32], [33], e.g., next-word prediction and medical image segmentation. Particularly, Hard et al. [17] applied FL to the training of a next-word prediction model that is used in a mobile virtual keyboard. Upon real-world but privacy-sensitive keyboard input data distributed across mobile devices, the FL-enabled next-word prediction model has achieved better performance than the centrally trained counterpart. Besides, FL has also been demonstrated in the task of semantic segmentation of medical image [16], which enables cross-institution model training without sharing the medical images of patients.

In FL, a central server is deployed to instruct client-side local training, aggregate model updates from clients, and evaluate the resulting global model. A typical FL training process consists of multiple rounds of client-side local training and server-side aggregation. For instance, in round r, the aggregation server first select n out of all the available N workers, and pushes the latest global model (model parameters, i.e., θ_g^r) to the selected n workers. Along with the global model are some configurations to specify the local training, e.g., the number of epochs, the batch size, and the learning rate, etc. Then, each selected client c fine-tunes θ_g^r with its local private data using stochastic gradient descent (SGD), and gets the updated model θ_c^r. The difference between the client-received global model θ_g^r and the updated client-specific model θ_c^r is calculated as $\nabla_c^r = \theta_c^r - \theta_g^r$, and will be uploaded to the server for aggregation. In the server-side aggregation, given $\{\nabla_c^r, c \in [n]\}$, an aggregation rule f_{agr} will be applied to get ∇_{agr}^r, which will be used to update the global model through $\theta_g^{r+1} = \theta_g^r + \eta \nabla_{\text{agr}}^r$ with η being the server-side learning rate.

Besides, FL is considered as *cross-device* FL when the clients are resource-constrained and large-scaled, e.g., many thousands of mobile devices. In cross-device FL, the client devices are likely crowd-sourced and can thus be byzantine or even malicious. On the other hand, when FL is applied to the training collaboration among trusted organizations, it is called *cross-silo* FL in which the clients are the participating organizations and thus of a smaller scale (e.g., ≤ 100). Since cross-silo FL tends to be a contracted collaboration across large corporations, it is less likely to have byzantine or malicious clients compared with cross-device FL.

A key element of FL is the aggregation rule (AGR). Among a variety of AGRs [29], [34], [35], [36], [37], the most commonly used one is Federated Averaging (FedAVG) [29], wherein the server updates the global model through dimension-wise and weighted averaging of the model updates. Besides, despite avoiding uploading private client-side data, the uploaded model updates, if leaked to attackers, may still raise various privacy attacks, e.g., gradient inversion attack [38], [39], [40], preference profiling attack [41] and membership inference attack [42]. To mitigate privacy attacks, various secure aggregation protocols [43], [44], [45], [46] have been proposed to ensure the aggregation can be carried out without revealing local model updates.

However, these privacy attacks require the access to model updates of benign FL clients and thus assume that the attackers can either compromise the central server or conduct successful man-in-the-middle (MITM) attacks against the communication between a targeted client and the central server, which we consider as impractical when evaluating the adversarial resistance of FL. Instead, we consider a practical threat model wherein the central FL server is trusted and the communication between FL clients and the central server is free of MITM attacks. Therefore, privacy attacks and secure aggregation rules are not evaluated in this study.

Poisoning attacks against FL. Depending on the attacker's goal, poisoning attacks against FL can be divided into three subcategories – *targeted* FL poisoning, *backdoor* FL poisoning and *untargeted* FL poisoning. The *targeted* FL poisoning attacks [19], [20] target a subset of designated classes and aim to decrease the model's prediction performance only for the classes of interest. Similarly, the *backdoor* FL poisoning attacks [21], [22], [23], [24] are designed to ensure that the corrupted model predict normally for regular samples but fails for samples stamped with a backdoor pattern (e.g., special pixel values in an image), and can thus be considered as a variant of the targeted FL poisoning attack. On the contrary, an *untargeted* FL poisoning attack [18], [21], [47], [48] is intended to degrade the overall prediction accuracy of a model for any input across all the classes. When evaluating adversary resistance of FL for security tasks, we focus on untargeted FL poisoning attacks, as they are more challenging to carry out and can incur great threats to real-world FL deployments [49].

Similar to poisoning attacks against central training, poisoning attacks against FL can be carried out through compromising the local datasets (e.g., label flipping), namely, *data poisoning* attacks [24], [20], [49]. Furthermore, attackers may also directly manipulate the model updates of compromised

devices, and thus poison the resulting model, which is named as the *model poisoning* attack [18], [19], [22], [23], [24], [47], [48], [49]. Existing model poisoning attacks against FL aim to directly manipulate the model updates of compromised devices in a manner that maximizes the attack goal while satisfying various constraints with regards to the fraction of compromised clients and the norms of the manipulated model updates. Representative model poisoning attacks include the *Little is Enough* (LIE) attack [21], MIN-MAX and MIN-SUM [48], static optimization (STAT-OPT) [47] , and projected gradient ascent (PGA) [49]. In this study, we assume a practical threat model featuring that the attacker is AGR-agnostic, therefore, three AGR-agnostic model poisoning algorithms are considered, which include LIE, MIN-MAX, and MIN-SUM.

Defense against FL poisoning attacks. To mitigate poisoning attacks, various robust aggregation rules (AGRs) have been proposed and evaluated. Such robust AGRs can be classified into two groups depending on how they detect and minimize the impact of manipulated model updates (gradients). One group focuses on filtering out malicious gradients either dimension-wise or vector-wise, and concrete examples include Krum [18], Multi-Krum [18], median-based filtering [50], [28], [51], [52], Bulyan [53], ERR/LFR [47], divide-and-conquer (DnC) [48]. The other group of AGRs utilizes various regularization techniques to minimize the adversarial impact of malicious model updates, e.g., Trimmed Mean [28], norm clipping [22]. In this study, when evaluating the effect of robust AGRs, we select from each group a representative robust AGR, namely, Multi-Krum and Trimmed Mean respectively.

Machine learning based SMS spam detection. Spam denotes unsolicited messages that are delivered to victims through various channels, such as the short message service (SMS) and the Email services. Varied by the delivery channel and the message format, spam messages can thus be grouped as SMS spam, Email spam, among others. In this study, we focus on the SMS spam detection considering the following factors. Particularly, it is privacy-invasive to collect SMS messages, regardless of spam or not spam, not to mention uploading SMS messages to the central server for model training and evaluation, which partially explains the scarcity of publicly available SMS spam/non-spam datasets. Besides, SMS messages are short in their length, and the underlying SMS spam campaigns keep evolving across time with more stealthy evasion techniques [15], rendering existing detection systems decay in their performance. All these factors make SMS spam detection a perfect scenario for federated learning.

The detection of SMS spam is typically defined as a binary text classification task. To conquer this task, various machine learning algorithms have been explored, which range from traditional classification algorithms (e.g., naive Bayes, random forest, and support vector machine) [25], [54], [55], to deep neural network architectures such as CNN [3], [4], LSTM [3], [4], BiLSTM [5], and transformer models [6]. Particularly, Roy et al. [3] built up a CNN-based SMS spam detection model which has achieved a state-of-the-art performance when trained and evaluated on the UCI SMS spam dataset [25]. However, almost all these detection models [25], [55], [3], [6], [5] were built upon the UCI SMS spam dataset [25] which, released in 2012, is small-scaled with only 747 English spam messages and likely outdated. To address the scarcity of public SMS spam datasets, Tang et al. [15] proposed *SpamHunter* to collect SMS spam messages as reported by victims on Twitter, which results in the release of 22K multilingual SMS spam messages, while Abayomi-Alli et al. [5] released another English spam dataset coined as *ExAIS_SMS* which was collected from 20 end users. In this study, we have combined all these newly released datasets when building up and evaluating SMS spam detection models in FL settings.

Machine learning based Android malware detection. Malware detection aims to decide whether a given software program is malicious or not, and optionally attribute a malicious program (malware) to relevant malware families. Serving as a key threat detection task, malware detection plays an important role in protecting end users and end devices. As malware programs on different operating systems can have significantly differentiated syntactic and malicious behaviors, the malware detection problem can be further divided into Android malware detection, Windows malware detection, and iOS malware detection, among others. In this study, we choose Android malware detection as the 2nd threat detection task due to several factors elaborated below. Firstly, Android, as an open mobile operating system, has been widely adopted by billions of mobile users, which gives rise to a large client base for FL-enforced malware detection. Also, alike SMS spam detection, Android malware detection also suffers from the scarcity and outdatedness of publicly available datasets [56], [57]. Besides, as malware intelligence is an important intellectual property, it is impractical for security vendors to directly share their proprietary malware datasets. Instead, collaborative learning via FL appears to be promising, which will be comprehensively evaluated in this study.

A long line of studies have thus proposed various machine learning systems for Android malware detection [1], [58], [2], [59], [60], [14], [56], [61], [62], e.g., Drebin [1], Mamadroid [58], and MalDozer [59]. These systems differ in many aspects, especially how detection features are extracted, what machine learning algorithms have been adopted, as well as how the detection system is evaluated and what groundtruth datasets have been applied to the evaluation. Particularly, McLaughlin et al. [2] abstracted an Android app as a sequence of opcodes and utilized an embedding layer to automatically embed the opcode sequence into a fixed-size feature vector, while Gao et al. [61] considered an Android app as a node in a graph and utilizes node embedding techniques to auto-encode an Android app. Also, various classification algorithms have been explored, e.g., random forest [58], SVM [1], [63], CNN [2], [63], [59], [60], LSTM [2], [63], and graph neural networks [61], [62]. Among these algorithms, CNN has achieved the best performance in multiple evaluations [2], [63], and we thus choose the CNN architecture proposed in [2]

as the default Android malware detection model for our FL experiments.

The evaluation of FL on cyber threat detection tasks. Concurrent with our work, Sidhpura et al. [64] evaluated the effectiveness of FL on SMS spam detection. However, their FL experiment settings are not practical as only two FL clients were deployed and the spam dataset under evaluation was outdated. Another study on FL-based SMS spam detection only considered three clients [65]. Also, the authors failed to explore security-specific FL scenarios (e.g., consistent label imbalance settings), not to mention profiling the adversarial resistance of FL for SMS spam detection. Besides, federated SVM was applied in Android malware detection in 2020 [66] while a FL-based Android malware detection framework, namely, FEDriod, was introduced in 2023 [67]. However, both studies considered no more than 7 FL clients for training, emphasizing only the effectiveness rather than efficiency and adversary resistance of federated learning. In 2023, a dynamic weighted federated averaging (DW-FedAvg) strategy was applied to Android malware detection [68]. Besides, Fereidooni, et al. [69] applied the concept of cross-silo FL to threat intelligence sharing across organizations. Still, very few settings were considered when profiling the effectiveness of FL, and no experiments were conducted to understand the adversarial resistance of FL.

Moving forward from these works, when evaluating the effectiveness of FL for cyber threat detection tasks, we have comprehensively considered various non-IID settings and consistent label imbalance scenarios. Furthermore, we have profiled, for the first time, the adversarial resistance of FL, in practical threat settings and for cyber threat detection tasks. Lastly, we have highlighted a set of efficiency issues for FL-based cyber threat detection along with promising solutions proposed and demonstrated.

III. SECURITY CLASSIFICATION TASKS

In this study, we target two representative security classification tasks, namely, SMS spam detection and Android malware detection, for both of which, respective detection models have been reproduced with state of the art (SOTA) performance achieved. Below, we elaborate these detection models, their training/testing datasets, as well as the model performance.

SMS spam detection. As aforementioned (§II), we focus on the *SMS* spam detection because this task is both challenging and privacy-sensitive when compared to other spam detection tasks, e.g., Twitter spam and Email spam.

Datasets. As listed in Table I, our groundtruth dataset is a combination of multiple publicly available SMS spam datasets. The first is the UCI SMS spam dataset [25] which was released in 2012 and consists of only 747 SMS spam messages and 4,827 non-spam messages. To enable an up-to-date evaluation of SMS spam detection, two more recent datasets were further collected, namely, ExAIS [5] and SpamHunter [15]. In total, these three datasets have contributed 26,346 SMS spam messages and 7,717 SMS non-spam messages which are featured

TABLE I
OUR SMS SPAM DATASETS.

Datasets	Spam	Non-Spam	Languages	Period
UCI	747	4,827	English	2012
ExAIS	2,350	2,890	English	2015
SpamHunter	23,249	0	Multilingual	2018-2022
Twitter	0	18,629	Multilingual	2018-2022
Total	26,346	26,346	Multilingual	2012-2022

TABLE II
SMS SPAM DETECTION MODELS.

Model	Datasets	Recall	Precision	F1-Score
CNN [3]	Spam-2022 [1]	0.9788	0.9807	0.9797
BERT [26]	Spam-2022	0.9879	0.9923	0.9804

[1] This is a merge of all datasets listed in Table I.

by 75 different natural languages and over 12 diverse spam categories [15]. Furthermore, to facilitate solid performance evaluation, we moved to make this dataset balanced. Specifically, we randomly sampled tweets from the Twitter archive [1] to complement the non-spam SMS messages, which is based upon the assumption that most posts published on Twitter are benign. This assumption is aligned with observations in previous studies [70], [71]. Also, we manually labeled 1,000 sampled tweets and confirmed that 97% of them are benign. In total, we have got a balanced dataset of 52,692 spam/non-spam messages. And we name this dataset as *Spam-2022*.

The model architectures. Based on a comprehensive literature for spam detection [25], [54], [55], [3], [4], [5], [6], two representative neural network architectures are selected for SMS spam detection. One is the CNN model as proposed in [3], while the other is a pre-trained transformer-based language model, namely, the multilingual BERT [26] which has achieved SOTA performance in many NLP tasks [26], [27]. Note that the original CNN model supports only English text as input and SMS messages in other languages should be translated into English in advance. Also, the CNN model has only 7 million parameters, which is much smaller than the multilingual BERT model of 167 million parameters.

The model performance. Given the largest-ever SMS spam groundtruth dataset (Spam-2022), two SMS spam detection models have been trained and tuned with different hyper-parameters explored. During model training, 80% of the spam groundtruth were randomly sampled out for training and validation, while the left 20% were used for testing. Table II lists a direct comparison between the CNN model and the BERT model with regards to their spam detection performance. As we can see, the BERT model outperforms the CNN model by 1.76% in recall and by 0.90% in F1 score. Considering the better performance as well as the multilingual support, the BERT model was chosen as the baseline model in our FL experiments for SMS spam detection.

[1] https://archive.org/details/twitterarchive

Datasets	Malware	Benign	Period
Drebin	4255	0	2010-2012
CIC-AndMal2017	0	1645	2015-2017
Androzoo	0	2610	2022
Total	4255	4255	2010-2022

Android malware detection. For Android malware detection, cross-silo FL may help security vendors collaboratively train an effective Android malware detection without revealing their proprietary malware datasets, while cross-device FL provides a possibility for Android users to contribute to Android malware detection without the need of uploading their local malware/benign Android apps to the server. Although benign Android apps can be easily collected, it is not the case for their realistic runtime logs, e.g., the api call sequence and network traffic flows. Such dynamic features are critical but costly to collect. However, end-user devices have app logs that are both realistic and abundant, but also privacy-sensitive, which renders a good fit for cross-device FL.

Datasets. Following previous practices in Andriod malware detection, a balanced groundtruth dataset has been composed, which consists of 4255 malware samples and 4255 benign samples as listed in Table III. For the malware samples, all were collected from the Drebin dataset [1], one of the most commonly used malware dataset [33], [59], [72]. Specifically, the Drebin dataset contains 5560 malware samples that belong to 179 different malware categories and families. After excluding files with decompiling failures or files with size smaller than 5KB, 4255 were selected out as the malware samples in our groundtruth. Since Drebin doesn't contain benign Android apps, the benign samples were composed from two sources. One is CIC-AndMal2017 [73] and the other is Androzoo [74]. Benign samples in the CIC-AndMal2017 were apps published in Google Play between 2015 and 2017. To guarantee the variety and obtain new samples, we also downloaded some other benign samples published in 2020 from Androzoo. For all benign samples, we queried VirusTotal [75] regarding their maliciousness and have thus confirmed there are no virus alerts for each of the selected benign samples.

The model architecture. We choose the CNN architecture proposed in [2] as the baseline model for Android malware detection, owing to its multiple advantages over other works [1], [58], [59], [60], [14], [56], [61], [62]. First of all, it has achieved a SOTA performance in multiple evaluations. Then, it takes as input only static features of an Android app, specifically, the sequence of opcodes, which can be easily extracted with low computing overhead. On contrary, other Android malware detection methodologies extract features either through dynamic execution of a given app [76], [58] or complicated static analysis [77], both of which are costly and thus infeasible in the FL scenario especially for cross-device FL. What's more, [2] is among the very few works

that have released the source code for training and evaluating the proposed malware detection systems, which makes it time-efficient for us to reproduce their model.

The model performance. Given the model architecture and datasets, a baseline model was then trained on our central server so as to facilitate a direct comparison with future models trained through federated learning. When building up this baseline model, 90% ground truth was used for training, and 10% were held out for model testing. Then, when training the model, the learning rate was set up as $1e^{-4}$ while the batch size was 5. As a result, this baseline model has achieved a precision of 98.60%, a recall of 99.76%, an accuracy of 99.18%, and a F1-score of 99.18%.

IV. THE FL SETTINGS

In this study, we aim to profile the feasibility of FL for the aforementioned two security classification tasks, namely SMS spam detection, and Android malware detection. This is achieved from three perspectives: effectiveness, byzantine resilience (i.e., adversarial resistance), and efficiency. Instead of exhaustively evaluating all possible FL configurations (hyper-parameters), we focus on *realistic* and *security-specific* scenarios which were missed by most previous works but are critical for the real-world FL deployment for cyber threat detection. Below, we first present the default FL settings (e.g., FL framework, hyper-parameters, and the computing environment) that will be adopted across our FL experiments, if not otherwise specified. What is followed is an overview regarding what experiments we have designed and conducted so as to give you the whole picture before going into detailed experimental results, which will be presented in §V for the effectiveness of FL, §VI for the byzantine resilience, and §VII for the efficiency (cost) of FL.

An overview of the FL experiments. Our FL experiments are designed to profile the following questions. First of all, *How effective can FL be when applied to representative security prediction tasks in realistic settings?* Then, given byzantine or even malicious clients that are likely to be present in security prediction scenarios, *how byzantine-resilient can FL be?* Besides, with or without defenses against adversarial attacks, *how efficient can FL be?* Finally, given FL evaluated in terms of effectiveness, byzantine resilience, and efficiency, *how can we address the weaknesses of FL if any?* In other words, what are the best practices when applying FL to security prediction tasks?

Experiments to evaluate FL effectiveness. The first set of FL experiments are designed to evaluate the effectiveness of FL in realistic settings for security prediction tasks. Instead of exhausting FL hyper-parameters, experiments in this group focus on evaluating realistic categories of non-IID (non-independent and identically distributed) data distributions across FL clients. Two of them are the non-IID data distributions of samples in terms of labels or quantity, which have also been evaluated in previous FL works for non-security tasks [78], and we thus name them as label-based non-IID and quantity-based non-

IID. Taking the label-based non-IID for SMS spam detection as an example, one client in the cross-device FL may have a spam/non-spam α ratio of $2:1$ while another may have an inverse α ratio of $1:2$.

Then, a security-specific non-IID scenario is also designed and evaluated. Specifically, in our cross-device FL for spam detection, clients (i.e., real-world end devices, especially mobile devices) tend to consistently have more non-spam samples than spam ones in their local datasets, whereas organization participants in the cross-silo FL are likely to have lots of spam messages but much fewer non-spam ones due to the privacy regulations. This gives rise to a scenario that is different from traditional non-IID data distributions, and we call it *the consistent label imbalance* (CLI) wherein most (if not all) clients consistently have samples of one class outweighing samples of another class. This CLI scenario is also applicable to malware detection, especially for the cross-device FL wherein the participating Android end devices are likely to have more benign apps installed than the malware ones. However, this may not be the case for the cross-silo malware detection as the participating anti-virus vendors are likely to have balanced datasets.

Besides, in the spam detection task, FL clients located in different countries or regions may differ in the language of their local message datasets. We name such kinds of scenarios as *language-based non-IID*. Similarly, Android malware apps may belong to different malware families, and samples belonging to different malware families can differ in their non-IID distribution across clients, which will also be evaluated under the name of *family-based non-IID*. More detailed experiment settings are presented in Section V along with respective results and observations.

Experiments to evaluate FL byzantine resilience. Since the FL clients can be byzantine or even malicious especially in the cross-device FL, the 2nd group of experiments aim to evaluate the byzantine resilience (adversary resistance) of FL in security prediction tasks. To achieve this, we first evaluated a representative data poisoning attack, namely label flipping [49], under different ratios of compromised clients. Also, previous works typically assume the attacker can poison all samples on a client or even generate a poisoned dataset that is much larger than those of benign clients, however, we argue that this is not realistic. This is because most attackers typically have no direct control over a large volume of FL clients, but can only remotely distribute poisoned data samples to FL clients, which means they can only poison a portion instead of all of the local samples. For instance, in the SMS spam detection scenario, attackers may distribute stealthy but true spam messages to some FL clients, and these messages may escape the device owner's annotation and be considered as non-spam during FL training. However, in the meantime, the attacker cannot prevent the same FL client from receiving SMS messages (spam or non-spam) from other sources as well as assigning true labels to these messages. The only chance that an attacker can fully poison a client

TABLE IV
THE DEFAULT FL PARAMETERS.

Parameter	Cross-Device	Cross-Silo
N: total number of FL clients	200	20
n: number of FL clients sampled in each FL round	10% of N (20)	All
Aggregation rule (AGR)	FedAVG	
Local epoch number e	2	
Local batch size β and learning rate e	32, $5e^{-5}$ for SMS spam 5, $1e^{-4}$ for Android malware	
Data distribution across clients	Dirichlet distribution with $\alpha = 1$	
The split ratio of training and testing	9:1	

is when they have full control of the respective client, in which case model poisoning is a better option rather than data poisoning. Also it can be easily detected by the server if an unusually large set of poisoned samples is used by the attacker. Therefore, stepping forward from previous studies, we evaluated different poisoning rates, i.e., the fraction of local samples being poisoned, and we assume that the size of the poisoned dataset is similar to that of a regular dataset of a benign FL client.

Then, when some participating FL clients are fully controlled by the attackers, model poisoning attacks become possible, in addition to data poisoning attacks. Among the SOTA model poisoning attacks, some (e.g., STAT-OPT and DYN-OPT [49]) rely on the knowledge of the server-side AGR algorithm, which we believe is not practical in the real threat model. Among the left ones without such a dependency, three are selected for our experiments, which include LIE [21], MIN-SUM [48], and MIN-MAX [48].

Note that for the above data/model poisoning attacks, we focus on untargeted poisoning, i.e., the attacking goal is to undermine the availability of the resulting model. We choose to evaluate untargeted poisoning rather than targeted or backdoor poisoning because untargeted poisoning is more challenging and has higher adversarial impact on realistic FL security applications. We have also evaluated the defensive effectiveness of two representative robust AGRs including Trimmed Mean [28] and Multi-Krum [18]. More details can be found in Section VI.

Experiments to evaluate FL efficiency and identify best FL practices. We have also characterized FL efficiency through a combination of empirical evaluation and theoretical analysis. Particularly, we investigated how to speed up FL training using model pre-training, especially in scenarios where data imbalance issues significantly slow down model convergence. The full details will be presented in Section VII.

The FL framework. We explored FL frameworks that are both popular and open source, and chose the Flower framework [79] because of its advantages in many aspects. Particularly, it is extensible enough to allow us to customize the FL training process and experiment with different realistic settings such as the presence of model poisoning attacks and

the deployment of robust aggregation rules (AGRs). Besides, it is agnostic to the underlying machine learning frameworks, which facilitates the migration of centrally trained security prediction models to the FL scenario.

The default FL hyper-parameters. Since this study focuses on evaluating realistic and security-specific FL settings, we adopt the same set of FL hyper parameters across experiments, unless specifically noted. Some of these FL hyper parameters are learned from common practices of previous FL works [80], [49], e.g., the number of clients in cross-device FL and cross-silo FL, and the aggregation rule, while others are distilled from our preliminary FL experiments, e.g., the client-side learning rate, the number of client-side epochs, etc. Also, FL training with these default hyper-parameters has achieved a performance comparable to that of centralized training, for both security classification tasks under our study. One thing to note is that the best hyper parameter setting varies across security classification tasks, and identifying such a setting is not our focus, but can be easily achieved by the central FL server in real-world deployments.

As listed in Table IV, the cross-device FL experiments involve the total number of clients $N = 200$ for both security classification tasks. Also, the number of clients sampled in each FL round is configured as $n = 20$. In addition, the well-adopted FedAVG [29] is chosen as the default aggregation rule (AGR) while the maximum of rounds of the FL training is configured as 500. Then, the client-side FL settings consist of the local epochs $e = 2$, a learning rate $l = 5e^{-5}$ for SMS spam detection while $1e^{-4}$ for Android malware detection, and a batch size $\beta = 32$ and 5 similarly. The cross-silo settings differ in two aspects. Since the cross-silo FL is designed for cross-organization collaborative learning, the total number of clients should be much smaller than that of cross-device FL. Also, unlike the cross-device FL, the cross-silo FL considers all clients (e.g., security vendors) to be available in each round. Therefore, we set $n = N = 20$. Also, for both cross-device FL and cross-silo FL, 10% of the samples are held out on the FL server for testing, while the remaining 90% are distributed to the N clients using a Dirichlet distribution [81] with $\alpha = 1$, which gives a quantity-based non-IID distribution of the samples across the FL clients.

The computing environment. Four servers were set up for our FL experiments. Each server runs on Ubuntu 20.04.6 LTS and is equipped with an AMD EPYC 7V13 CPU with 24 CPU cores, 220 GB memory, 1 TB of disk storage, and an NVIDIA A100 GPU with 80 GB of GPU memory.
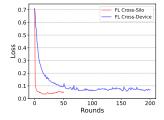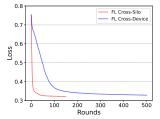
V. THE EFFECTIVENESS OF FL

In this section, we evaluate the effectiveness of FL for security prediction tasks. Below, detailed experiments along with key results and observations are presented.

> **Finding I**: *The FL-trained security detection models can achieve a performance that is comparable to their centrally trained counterparts.*

Task	Model	Precision	Recall	Accuracy
SMS Spam	Central	**0.9923**	0.9879	0.9901
	FL$_{cross-device}$	0.9871	0.9845	0.9858
	FL$_{cross-silo}$	0.9905	**0.9903**	**0.9904**
Android Malware	Central	**0.9907**	0.9976	**0.9941**
	FL$_{cross-device}$	0.9766	0.9936	0.9849
	FL$_{cross-silo}$	0.9838	**1.0**	0.9918

(a) SMS spam classification.   (b) Android malware classification.

Fig. 1. The convergence process of the FL baseline models for SMS spam detection and Android malware classification respectively.

FL baseline performance. Using the default FL settings (Table IV), a FL baseline model is trained for the aforementioned two security classification tasks and two FL scenarios (cross-device and cross-silo). Table V lists the resulting performance statistics of these baseline models along with a comparison to centrally trained counterparts. We can see that the FL baseline models, no matter whether cross-device FL or cross-silo FL, have achieved a performance that is comparable to their centrally trained counterparts. For instance, the spam detection model trained via cross-device FL has achieved a recall of 99.03% and a precision of 99.05%, while it is 98.79% and 99.23% respectively for the centrally trained counterpart. Figure 1 also presents the FL training process for these baseline models, and we can see that the cross-silo FL converges faster than the cross-device FL in terms of the number of rounds, which is expected since cross-silo FL involves all clients in each training round while cross-device FL samples only 10% clients in each training round. In addition, the number of rounds needed to converge varies across security detection tasks. For instance, in cross-device FL training, the SMS spam detection task converges more quickly with less than 100 rounds while it is over 200 rounds for Android malware detection.

> **Finding II**: *A high degree of quantity-based non-IID distribution can yield better model performance and faster training convergence, especially for cross-device FL.*

Quantity-based non-IID distribution. The quantity-based non-IID is defined as the non-IID distribution of data samples across FL clients in terms of the quantity. It is assumed that these client-side local datasets still have the same label-wise distribution as the testing dataset on the FL server. The quantity of the client-side local dataset is abstracted as a

TABLE VI
THE PERFORMANCE OF FL-TRAINED **SPAM DETECTION** MODELS UNDER
VARIOUS QUANTITY-BASED NON-IID DATA DISTRIBUTIONS.

FL	Dirichlet	Precision	Recall
Cross-Device	$\alpha = 0.5$	**0.9893** $\pm$ 0.0037	**0.9865** $\pm$ 0.0024
	$\alpha = 1$	0.9871 $\pm$ 0.0042	0.9845 $\pm$ 0.0036
	$\alpha = 5$	0.9806 $\pm$ 0.0063	0.9755 $\pm$ 0.0037
	$\alpha = 10$	0.9807 $\pm$ 0.0062	0.9682 $\pm$ 0.0046
Cross-Silo	$\alpha = 0.5$	0.9899 $\pm$ 0.0015	**0.9912** $\pm$ 0.0010
	$\alpha = 1$	0.9905 $\pm$ 0.0025	0.9903 $\pm$ 0.0027
	$\alpha = 5$	0.9907 $\pm$ 0.0019	0.9885 $\pm$ 0.0019
	$\alpha = 10$	**0.9914** $\pm$ 0.0015	0.9883 $\pm$ 0.0017

TABLE VII
THE PERFORMANCE OF FL-TRAINED **MALWARE DETECTION** MODELS
UNDER VARIOUS QUANTITY-BASED NON-IID DATA DISTRIBUTION.

FL	Dirichlet	Precision	Recall
Cross-Device	$\alpha = 0.5$	**0.9797** $\pm$ 0.0022	**0.9976** $\pm$ 0.0000
	$\alpha = 1$	0.9766 $\pm$ 0.0007	0.9936 $\pm$ 0.0011
	$\alpha = 5$	0.9588 $\pm$ 0.0009	0.9972 $\pm$ 0.0009
	$\alpha = 10$	0.9538 $\pm$ 0.0028	0.9953 $\pm$ 0.0000
Cross-Silo	$\alpha = 0.5$	**0.9860** $\pm$ 0.0000	0.9967 $\pm$ 0.0012
	$\alpha = 1$	0.9838 $\pm$ 0.0000	**1.0000** $\pm$ 0.0000
	$\alpha = 5$	0.9751 $\pm$ 0.0032	0.9939 $\pm$ 0.0012
	$\alpha = 10$	0.9633 $\pm$ 0.0032	0.9953 $\pm$ 0.0000

Dirichlet distribution, and the extent of non-IID is configured through the parameter alpha $\alpha \in \{0.5, 1, 5, 10\}$ where the larger the α, the less non-IID the respective data distribution is. For instance, in the case of $\alpha = 10$, FL clients are more likely to have same-sized local datasets compared with $\alpha = 1$.

The performance statistics in quantity-based non-IID settings are listed in Table VI for SMS spam detection, and Table VII for Android malware detection. Since the FL training process didn't converge stably in some quantity-based non-IID settings, we report here the average performance of the last 10 rounds of each FL training along with the standard deviation. As we can observe that the higher the quantity-based non-IID of client-side data is, the better performance the resulting FL model can achieve. In other words, as the extent of quantity-based non-IID increases with smaller α,

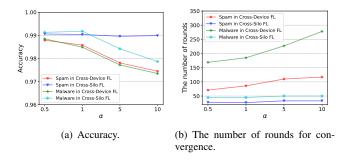(a) Accuracy. (b) The number of rounds for convergence.

Fig. 2. The stats of FL-trained security classification models under various quantity-based non-IID data distributions. The α denotes the parameter of Dirichlet distribution.

the resulting FL model achieves a better performance. For instance, when the extent of non-IID increases from $\alpha = 10$ to $\alpha = 0.5$, the performance of the Android malware detection model has increased by 2.59% (from 95.38% to 97.97%) for the precision and by 0.23% (from 99.53% 99.76%) for the recall. Such performance variance is also illustrated in Figure 2(a).

As presented in Figure 2(b), another benefit of a higher extent of quantity-based non-IID is the fewer number of rounds it takes to converge in FL training. Here, a patience of 10 is configured for spam and 3 for malware, when determining whether the training converges. Given such a benefit of quantity-based non-IID, one possible explanation is that a higher level of quantity-based non-IID leads to a greater variance in the local data size of the FL clients. Consequently, it allows some FL clients to train better local models more quickly. The aggregation of such early but better local model updates leads to faster convergence and better performance. However, one exception is the SMS spam detection model trained through cross-silo FL, where the performance variance among different α does not exceed 0.001. Our explanation is that the average client-side SMS data size of each client selected in cross-silo FL is ten times larger than that in cross-device FL. As a result, regardless of the level of the quantity-based non-IID, the client-side SMS datasets in cross-silo FL are sufficiently large to generate a global model that can quickly converge with good performance. In a nutshell, it can be concluded that in practical scenarios, a higher level of quantity-based non-IID distribution may actually lead to better performance and faster convergence, especially for cross-device FL. However, on contrary, label-based non-IID can incur extra delay in convergence as well as making the training process unstable, as detailed below.
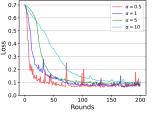
> **Finding III**: *A high degree of label-based non-IID distribution has no obvious impact on model performance, but can incur notable fluctuations (i.e., unstable convergence) in FL training, especially for the cross-device FL.*
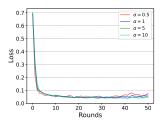
Label-based non-IID distribution. Under the label-based non-IID distribution, client-side local datasets vary widely in the label distribution. We still utilized the Dirichlet function to capture this distribution and experimented with different alpha values ($\alpha \in \{0.5, 1, 5, 10\}$). Specifically, given an α, a Dirichlet sequence of N values is generated to instruct how to distribute all the positive samples (spam samples or malware samples) to the N clients, and another Dirichlet sequence of N values with the same α is generated to specify what fraction of negative samples each FL client should account for.

Table VIII lists the performance stats of the resulting FL-trained models under different α values, and we can see that the label-based non-IID distribution has no obvious impact on the model performance. Nevertheless, the higher the degree of label-based non-IID is, the more fluctuations we can observe during the FL training, which is likely due to client-side overfitting. For instance, in the cross-device FL, a SMS spam model trained with $\alpha = 0.5$ has a standard deviation of

TABLE VIII
The performance of FL-trained models under various
label-based non-IID data distribution.

FL	Dirichlet	Precision	Recall
Spam Cross-Device	$\alpha = 0.5$	0.9811 ± 0.0201	0.9792 ± 0.0116
	$\alpha = 1$	0.9773 ± 0.0105	0.9817 ± 0.0063
	$\alpha = 5$	0.9831 ± 0.0049	0.9633 ± 0.0042
	$\alpha = 10$	0.9670 ± 0.0067	0.9680 ± 0.0054
Spam Cross-Silo	$\alpha = 0.5$	0.9920 ± 0.0011	0.9865 ± 0.0022
	$\alpha = 1$	0.9898 ± 0.0016	0.9889 ± 0.0021
	$\alpha = 5$	0.9898 ± 0.0016	0.9866 ± 0.0026
	$\alpha = 10$	0.9913 ± 0.0011	0.9896 ± 0.0015
Malware Cross-Device	$\alpha = 0.5$	0.9611 ± 0.0021	1.000 ± 0.0000
	$\alpha = 1$	0.9689 ± 0.0053	0.9972 ± 0.0009
	$\alpha = 5$	0.9632 ± 0.0027	0.9974 ± 0.0007
	$\alpha = 10$	0.9541 ± 0.0029	0.9984 ± 0.0015
Malware Cross-Silo	$\alpha = 0.5$	0.9757 ± 0.0011	0.9821 ± 0.0041
	$\alpha = 1$	0.9521 ± 0.0024	0.9953 ± 0.0000
	$\alpha = 5$	0.9765 ± 0.0056	0.9953 ± 0.0000
	$\alpha = 10$	0.9607 ± 0.0024	0.9953 ± 0.0000

TABLE IX
The accuracy of security models under CLI scenarios.

Task	Model	PNR		
		0.25	1	4
Spam	Central	0.9865	0.9898	0.9873
	Cross-Device FL	0.9703	0.9857	0.9641
	Cross-Silo FL	0.9857	0.9905	0.9821
Malware	Central	0.9854	0.9963	0.7866
	Cross-Device FL	0.9805	0.9901	0.5000
	Cross-Silo FL	0.9755	0.9894	0.5026

(a) SMS spam in cross-device FL.

(b) SMS spam in cross-silo FL.

(c) Malware in cross-device FL.

(d) Malware in cross-silo FL.

Fig. 3. The FL training processes for label-based non-IID scenarios.

2.01% in its precision for the last 10 rounds of FL training, while it is only 0.67% for $\alpha = 10$. As we can see, this phenomenon is especially obvious for cross-device FL, as further illustrated in Figure 3. We conclude that the label-based non-IID distribution can cause non-negligible instability (fluctuations) for the FL convergence process, especially for cross-device FL involving a large number of clients. Also, another side effect of label-based non-IID is the delay in convergence, which will be further elaborated in §VII.

> **Finding IV**: *The consistent label imbalance (CLI) towards positive (malicious) samples can incur a non-negligible degradation in performance, while the performance impact of CLI towards negative samples (more practical in cross-device FL) is minor.*

Consistent label imbalance (CLI). CLI refers to that FL clients *consistently* have samples of one class outweighing samples of another, e.g., more benign apps than malware, or more non-spam messages than spam. Here, we define PNR, the ratio of positive samples to negative samples, to profile the extent of consistent label imbalance. Specifically, in cross-device FL, clients tend to have more negative samples than positive ones, i.e., $PNR < 1$. On the contrary, FL clients in the cross-silo scenario are more likely to have more positive samples than negative ones, which renders $PNR > 1$. For both scenarios, we have experimented $PNR \in \{0.25, 1, 4\}$.

Given a PNR, the following strategies are designed to distribute samples to FL clients to guarantee that the clients' local datasets satisfy the given PNR. When $PNR \leq 1$, all the negative samples (except for the testing dataset) are first distributed to the N FL clients by following a Dirichlet distribution with $\alpha = 1$, which assigns NS_i negative samples to client i. Then, $\lceil NS_i \times PNR \rceil$ positive samples will be randomly selected from all the positive samples with replacement before being assigned to client i. Similarly, when $PNR > 1$, the positive samples for each client will first be assigned by following the same Dirichlet distribution. Then, given that the number of the positive samples for client i is decided as PN_i, the negative samples for client i will be $NS_i = \lceil PN_i / PNR \rceil$, and they will be randomly sampled from the overall negative samples with replacement.

Table IX presents the accuracy stats of the resulting models under different PNR values. We can see that the balanced setting ($PNR = 1$) has achieved the best performance, across security tasks and FL scenarios. However, when the dataset is not balanced ($PNR \neq 1$), the cases with more positive samples ($PNR > 1$) can incur worse performance when compared to the cases with more negative samples ($PNR < 1$), especially for the cross-device scenario. Particularly, when the $PNR = 4$, the cross-device SMS spam model has achieved an accuracy of 96.41%, which is 2.16% lower than the balanced case and 0.62% lower than $PNR = \frac{1}{4}$. This pattern is more obvious for the malware detection task which achieved an accuracy of 98.05% at $PNR = \frac{1}{4}$ in cross-device FL while the counterpart at $PNR = 4$ didn't even converge (see Figure 4). We thus conclude that the consistent label imbalance towards positive (malicious) samples can incur a non-negligible degradation in performance, while the impact of the CLI towards negative samples is minor. Further investigation shows that
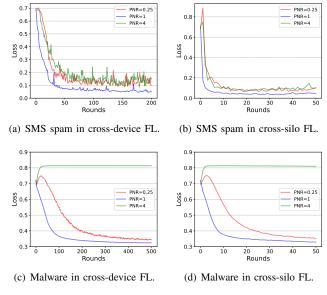
(a) SMS spam in cross-device FL.

(b) SMS spam in cross-silo FL.

(c) Malware in cross-device FL.

(d) Malware in cross-silo FL.

Fig. 4. The FL training processes for the scenarios of consistent label imbalance.

TABLE X
THE PERFORMANCE STATS OF FL-TRAINED SPAM DETECTION MODELS UNDER LANGUAGE-BASED NON-IID DISTRIBUTION.

FL	k	Precision	Recall
Cross-Device	$k = 1$	0.9609 ± 0.0087	0.9889 ± 0.0025
	$k = 2$	0.9726 ± 0.0059	0.9896 ± 0.0022
	$k = 3$	0.9771 ± 0.0101	0.9861 ± 0.0042
	$k = 4$	0.9796 ± 0.0063	0.9891 ± 0.0024
Cross-Silo	$k = 1$	0.9760 ± 0.0037	0.9162 ± 0.0071
	$k = 2$	0.9850 ± 0.0044	0.9920 ± 0.0019
	$k = 3$	0.9840 ± 0.0042	0.9901 ± 0.0017
	$k = 4$	0.9866 ± 0.0044	0.9895 ± 0.0025

(a) Malware in cross-device FL.

(b) Malware in cross-silo FL.

Fig. 5. The FL training processes for the scenarios of family imbalance for malware detection.
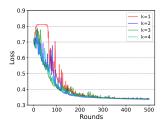
when $PNR > 1$, the FL-trained model tends to generate much more false alarms, which leads to a much lower precision. For instance, when $PNR = 4$, compared with the centrally trained counterparts, FL-trained models have a similar recall but the precision has lowered by almost 4% for the cross-device spam model and 29% for the cross-device malware model. More details can be found in Appendix A-A As FL clients for cyber threat detection tend to have more negative samples than positive ones, we believe the CLI issue won't undermine the effectiveness of FL for cyber threat detection.
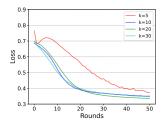
Language-based non-IID distribution for SMS spam detection. When applying FL to SMS spam detection, FL clients, especially those in the cross-device scenario, are more likely to have all their spam/non-spam messages composed in very few natural languages. It is thus interesting and necessary to evaluate the impact of such a language-based non-IID distribution on FL effectiveness. As detailed in §III, samples in our spam/non-spam dataset are of 25 different natural languages. In our language-relevant experiments, each FL client will first be randomly assigned with k different languages, which means this client can only have samples belonging to one of these k languages. Then, samples of language l will be assigned only to the subset of clients that qualify this language requirement, and a Dirichlet distribution with $\alpha = 1$ is used to fulfill the sample assignment. In our experiments, 4 different k values have been evaluated, which are $k \in \{1, 2, 3, 4\}$. The respective performance stats are listed in Table X. In a nutshell, the more languages each FL client is assigned with, the better precision the respective FL-trained model can achieve, while the effect of language-based non-IID on the recall is minor.

Family-based non-IID distribution for Android malware detection. Similar to spam detection, malware detection also has a unique non-IID distribution that is related to malware

families. Specifically, there exists an imbalanced distribution of malware families across FL clients, especially in the scenario of cross-device FL. To profile its potential impact on FL training, we conducted experiments following a setting that is similar to that of aforementioned language-based non-IID experiments. Here, we have 133 distinct malware families. Considering cross-silo FL only has 20 clients in total, in order not to miss out too much families, each cross-silo client needs to select at least $\{k : k > 4\}$ families. The following k values have been explored: $\{5, 10, 20, 30\}$ in cross-silo FL and $\{1, 2, 3, 4\}$ in cross-device FL.

The respective performance stats can be found in Table XI while the training process is illustrated in Figure 5. We can see that, as k gets smaller, the training processing becomes more unstable and converges more slowly. However there is no obvious pattern observed for the model performance, which is different from the language-based non-IID distribution for SMS spam detection.

Finding V: *Non-IID data distribution can lead to high instability during the convergence of the respective FL training process.*

The convergence of FL training in non-IID or imbalanced scenarios. As revealed from above experiments, some non-IID scenarios lead to higher instability for the convergence of the respective FL training process. For instance, in the scenario of label-based non-IID distribution with $\alpha = 0.5$, given the models trained during the last 10 rounds, the standard deviation of their respective accuracy is 0.81% while it is only 0.07% for our baseline model for SMS spam. This observation is further illustrated in Figure 6 showing the FL

FL	k	Precision	Recall
Cross-Device	$k = 1$	0.9617 ± 0.0080	0.9972 ± 0.0027
	$k = 2$	0.9503 ± 0.0053	0.9967 ± 0.0056
	$k = 3$	0.9530 ± 0.0050	0.9967 ± 0.0012
	$k = 4$	0.9615 ± 0.0044	0.9976 ± 0.0000
Cross-Silo	$k = 5$	0.9774 ± 0.0018	0.9548 ± 0.0090
	$k = 10$	0.9472 ± 0.0025	0.9788 ± 0.0033
	$k = 20$	0.9742 ± 0.0009	0.9951 ± 0.0007
	$k = 30$	0.9609 ± 0.0016	0.9875 ± 0.0018

(a) Spam in cross-device FL.

(b) Spam in cross-silo FL.

(c) Malware in cross-device FL.
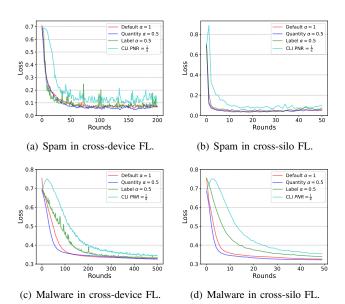
(d) Malware in cross-silo FL.

Fig. 6. The (in)stability of the FL training processes for the scenarios of extreme non-IID distribution.

training process for various non-IID distributions. We can see the consistent label imbalance with $PNR = \frac{1}{4}$ stands out to suffer from the most severe convergence instability, which is followed by the label-based non-IID with $\alpha = 0.5$ and quantity-based non-IID with $\alpha = 0.5$.

We have also defined a set of metrics so as to quantify the instability of the convergence process across aforementioned non-IID scenarios, which distills the same observation as Figure 6. For more details, please refer to Appendix A-B. To address the instability issue, we proposed and evaluated the bootstrapping strategy which will be detailed in Section VII.

VI. THE BYZANTINE RESILIENCE OF FL

In this section, we move to evaluate the byzantine resilience of FL with regards to data poisoning and model poisoning.

The threat model. When selecting data/model poisoning attacks and evaluating their attack impact, we consider a realistic threat model. Specifically, for both data poisoning and model poisoning, we assume that the attacker has whitebox knowledge of the global model (its parameters, architecture, and output), as this knowledge can be easily inferred from ei-

ther the model updates or the local training process. However, the attacker has no knowledge of the aggregation rule that is enforced by the central server, so any attacks that rely on such knowledge are considered as impractical and are outside the scope of our evaluation.

Also, for both cross-device FL and cross-silo FL, the attackers should have no access to benign FL clients, regardless of data poisoning or model poisoning. Besides, we assume that data poisoning attacks don't involve any effective collusion among poisoned clients as the attackers of data poisoning either have no direct access to the poisoned FL clients or the access is too limited to allow effective collusion. Instead, a model poisoning attacker in the cross-device FL should have compromised one or more FL clients and these compromised FL clients can collude with each other during the FL training process. Note that such a model poisoning attack is considered only possible for the cross-device FL but not the cross-silo FL, as the participants of the cross-silo FL are assumed to be trusted organizations under contracts and their FL servers should be well protected. Regarding to the ratio of poisoned FL clients M, we consider $M \leq 5\%$ as practical and realistic for data poisoning while it is $M \leq \%1$ for model poisoning, which is aligned with recent works on practical poisoning attacks [49]. Then, in terms of the attack goals, the attackers aim to degrade the overall performance of the poisoned model and cause the model to misclassify any given test input.

To summarize, we consider untargeted FL poisoning attacks and the attackers have whitebox knowledge of the global model, but no knowledge of the server-side aggregation rule. Then, a practical data poisoning attack is equipped with very limited access to the poisoned FL clients, and thus does not involve cross-client collusion during FL training, whereas a practical model poisoning attack has full access of the poisoned FL clients to enable cross-client collusion, but is only feasible for the cross-device FL.

A. Data Poisoning Attacks

Finding VI: *Data poisoning with a practical fraction of poisoned clients ($M \leq 5\%$) has a negligible attack impact in terms of accuracy decrease, and even poisoning with an impractically high $M = 40\%$ still results in minor attack impact as long as the per-client sample poisoning rate is realistic (e.g., 25%).*

Data poisoning under the default FL setting. We first evaluated the untargeted data poisoning attacks under the default FL setting (Table IV) where the AGR is the FedAVG and the cross-device data distribution is quantity-based Dirichlet with $\alpha = 1$. Here, we define M as the fraction of *manipulated FL clients* that have their local datasets been poisoned to some extent. Also, for each manipulated FL client, the fraction of local samples being poisoned is defined as the poisoning ratio p. For instance, when $p = 0.5$, each of the manipulated FL clients ($N \times M$) has 50% of its local datasets been manipulated. Then, regarding the manipulation strategy, we consider the static label flipping, which means the manipulated
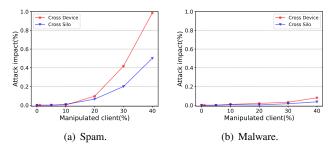
Fig. 7. The attack impact in terms of accuracy decrease for data poisoning attacks of different fractions of manipulated clients M, when $p = 100\%$.

TABLE XII

THE ATTACK IMPACT IN TERMS OF ACCURACY DECREASE FOR **DATA POISONING ATTACKS** AT DIFFERENT FRACTIONS OF MANIPULATED CLIENTS M AND THE SAME LOCAL DATA POISONING RATE $p = 100\%$.

Task	M	Cross-device	Cross-silo
Spam	1%	0.0008 ± 0.0009	-[1]
	5%	0.0014 ± 0.0020	0.0006 ± 0.0009
	10%	0.0032 ± 0.0024	0.0091 ± 0.0178
	20%	0.0968 ± 0.1453	0.0659 ± 0.1214
	30%	0.4183 ± 0.1164	0.2004 ± 0.1573
	40%	0.9858 ± 0.0007	0.5020 ± 0.0153
Malware	1%	0.0002 ± 0.0005	-[1]
	5%	0.0006 ± 0.0008	-0.0001 ± 0.0013
	10%	0.0093 ± 0.0008	0.0062 ± 0.0018
	20%	0.0200 ± 0.0064	0.0034 ± 0.0010
	30%	0.0325 ± 0.0341	0.0165 ± 0.0009
	40%	0.0801 ± 0.0398	0.0364 ± 0.0041

[1] In cross-silo FL, there's no $M = 1\%$ result, since 1 compromised client corresponds to $M = 5\%$.

TABLE XIII

THE ATTACK IMPACT IN TERMS OF ACCURACY DECREASE FOR **DATA POISONING ATTACKS** WITH $M = 40\%$ AND DIFFERENT POISONING RATES p.

Task	p	Cross-device
Spam	25%	0.0081 ± 0.0038
	50%	0.0402 ± 0.0325
	75%	0.4579 ± 0.0505
	100%	0.9858 ± 0.0007
Malware	25%	0.0121 ± 0.0032
	50%	0.0126 ± 0.0107
	75%	0.0154 ± 0.0039
	100%	0.0927 ± 0.0293

TABLE XIV

A COMPARISON AMONG DIFFERENT AGRs IN TERMS OF DEFENDING AGAINST **DATA POISONING ATTACKS** WITH IMPRACTICALLY HIGH M.

Task	M	Trimmed Mean	Multi-Krum	FedAVG
Spam	20%	0.0269 [1]	0.0129	0.0968
	30%	0.0510	0.0239	0.4183
	40%	0.4458	0.2231	0.9858
Malware	20%	0.0169	0.0174	0.0200
	30%	0.0091	0.0173	0.0325
	40%	0.0261	0.0216	0.0801

[1] Each cell denotes the attack impact in terms of the loss of model accuracy.

samples has its payload unchanged but its label flipped. For instance, a spam message under manipulation has its label switched to non-spam but its text content unchanged. Although the settings of different M has been explored in multiple studies [20], [49] on data poisoning, *we are the first to explore the impact of different poisoning rate p*, which is essential to our research goal of evaluating FL in *realistic* scenarios.

We first explored the attack impact of different fractions of manipulated clients $M \in \{1\%, 5\%, 10\%, 20\%, 30\%, 40\%\}$ at a poisoning rate of $p = 100\%$. The attack results are presented in Table XII and Figure 7 while the corresponding FL training processes are illustrated in Appendix B-B. As we can see, when M is in the practical range ($M \leq 5\%$) of data poisoning, the impact of the attack on accuracy degradation is less than 0.2%, and thus negligible. When M is set to a large ratio, e.g., 20%, the attack impact can be significant, 9.68% for spam detection and 2.00% for malware detection, which is consistent with previous results on non-security FL tasks [20], [49]. Also, compared to spam detection, malware detection models appear to be more robust in terms of defending against large but impractical fractions of manipulated clients. For instance, when $M = 30\%$, the data poisoning attack has achieved an accuracy degradation of 41.83% on SMS spam detection while it is only 3.25% for Android malware detection.

As the effect of data poisoning is negligible when $M \leq 5$

and $p = 100\%$, we didn't proceed to evaluate lower p for $M \leq 5$. Instead, we explored the effect of different p at $M = 40\%$, for which $p \in \{25\%, 50\%, 75\%, 100\%\}$ were considered. The attack results are listed in Table XIII. As we can see, even if the M is impractically high at 40%, a practical sample poisoning rate of $p = 25\%$ renders the data poisoning almost ineffective, which decreases the model accuracy by only 0.81% for SMS spam detection and 1.21% for Android malware detection. We also considered the scenario wherein security vendors with large datasets may all have their data poisoned but at a low fraction. Specifically, we further explored the cross-silo FL at $M = 100\%$ and $p = 10\%$, for which, the attack impact turns out to be negligible.

Data poisoning under label-based non-IID data distribution. As aforementioned, under the default FL setting, practical data poisoning attacks have a negligible attack impact in terms of model performance degradation. We thus moved on to explore whether they can work in the label-based non-IID scenario, which is a more realistic FL scenario and may thus give the attackers more poisoning opportunities. Specifically, we consider two practical attack settings, one with $M = 5\%, p = 50\%$ and the other with $M = 5\%, p = 100\%$. We then evaluated their attack impact under different label-based non-IID distributions which vary by the Dirichlet parameter $\alpha = \{0.5, 1, 5, 10\}$. And we observed that data poisoning has no attack impact across these label-based non-IID scenarios and practical poisoning settings, which is consistent with our observation for the default FL settings. Specific data points can be found in Appendix B-A.
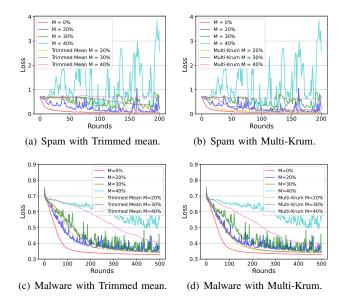
(a) Spam with Trimmed mean.

(b) Spam with Multi-Krum.

(c) Malware with Trimmed mean.

(d) Malware with Multi-Krum.

Fig. 8. The training processes under data poisoning attacks in cross-device FL with or without robust AGRs enforced.

Finding VII: *Given impractically powerful data poisoning attackers (equipped with impractically high M and p), robust AGRs can not only significantly lower the attack impact but also make the training process more smooth.*

Data poisoning under robust AGRs. Despite impractical, some data poisoning attacks with both high M and high p still achieve non-negligible attack impact, so we moved on to evaluate the effectiveness of robust AGRs in terms of defending against such types of impractical data poisoning attacks. Specifically, two representative robust AGRs (Trimmed Mean [28] and Multi-Krum [18]) have been tested in our experiments. Table XIV presents a direct comparison among these AGRs in terms of preventing data poisoning attacks, and we can observe that robust AGRs can effectively prevent data poisoning attacks to some extent. For instance, when $M = 30\%$, the attack on FedAVG resulted in a 41.83% decrease in accuracy for SMS spam detection, but this decreased to 2.39% with the use of Multi-Krum and to 5.10% with Trimmed Mean. Figure 8 also presents a comparison between training processes with robust AGRs and training processes with the default FedAVG. We can see that robust AGRs can not only help mitigate impractical but effective data poisoning attacks, but also smooth the training process, i.e., making the training process converge more stably.

B. Model Poisoning Attacks

Finding VIII: *Model poisoning with a practical fraction of poisoned clients ($M \leq 1\%$) has an either negligible or minor attack impact for cyber threat detection tasks.*

Model poisoning attacks under the default FL setting. As mentioned earlier, when it comes to model poisoning attacks, we consider it is only practical for the cross-device FL but not the cross-silo FL, as it tends to be unlikely for

TABLE XV
THE ATTACK IMPACT IN TERMS OF ACCURACY DECREASE FOR **MODEL POISONING ATTACKS** IN THE DEFAULT FL SETTINGS.

Task	M	LIE	MIN-MAX	MIN-SUM
Spam	1%	0.0003	0.0152	0.0130
	5%	0.0016	0.0177	0.0166
	10%	0.0014	0.0175	0.0164
	20%	0.0019	0.0180	0.0168
	30%	0.0014	0.0197	0.0241
	40%	0.0026	0.0201	0.0199
Malware	1%	0.0002	0.0014	-0.0022
	5%	0.0054	-0.0007	0.0073
	10%	-0.0038	0.0049	-0.0039
	20%	-0.0007	0.0042	0.0049
	30%	0.0048	0.0039	0.0053
	40%	0.0065	0.0015	0.0176

an attacker to compromise an organization participating in the cross-silo FL. When evaluating model poisoning, different fractions of compromised clients were experimented and they are $M \in \{1\%, 5\%, 10\%, 20\%, 30\%, 40\%\}$. However, unlike the data poisoning where $M \leq 5\%$ are considered as practical, only $M \leq 1\%$ is considered as practical since model poisoning requires a higher threat bar than data poisoning in the real world. Regarding to the specific model poisoning algorithms, we consider three representative ones: LIE (little is enough) [21], MIN-MAX [48], and MIN-SUM [48], all of which are AGR-agnostic and thus qualify our threat model.

Across these algorithms, we assume the attackers have no access to benign FL clients (and their updates), but full access to the original benign updates of the compromised FL clients. In a nutshell, the LIE attack adds to each dimension of the model updates of the compromised clients, a perturbation that is small enough to not be detected and pruned as outliers by robust AGRs. Unlike LIE, MIN-MAX and MIN-SUM formulate the perturbation generation as an optimization problem wherein the resulting malicious model update is upper bounded by its distance to benign updates of compromised FL clients. Particularly, In MIN-MAX, the distance of the resulting malicious model update to any benign updates is upper bounded by the maximum distance between any pair of benign updates, whereas MIN-SUM requires the sum of squared distance between the malicious update and all benign updates should be no more than that of all pairs of benign updates. For more details regarding these attack algorithms, please refer to the original papers [21], [48].

Table XV lists the attack impact of these algorithms in terms of the decrease in model accuracy. We can see that, regardless of the attack techniques, the practical model poisoning attack ($M \leq 1\%$) has a negligible impact on accuracy loss (less than 0.2%). One exception resides in the attacks using MIN-MAX and MIN-SUM against the spam detection model, which have achieved a loss in accuracy by 1.52% and 1.30% respectively, which is still minor and thus acceptable, considering the extra benefits of FL. Then, even if the M is set at an impractically high value, e.g., 30%, the attack impact is still very minor for

TABLE XVI
THE ATTACK IMPACT OF PRACTICAL **MODEL POISONING ATTACKS**
UNDER LABEL-BASED NON-IID DATA DISTRIBUTION.

Task	M	Non-IID	LIE	MIN-MAX	MIN-SUM
Spam	1%	0.5	−0.0025	0.0131	0.0107
		1	−0.0008	0.0199	0.0287
		5	−0.0004	0.0267	0.0282
		10	−0.0019	0.0250	0.0223
	5%	0.5	−0.0041	0.0213	0.0240
		1	−0.0007	0.0338	0.0329
		5	0.0015	0.0485	0.0451
		10	−0.0018	0.0433	0.0407
Malware	1%	0.5	0.0031	0.0086	0.0033
		1	0.0015	0.0072	0.0054
		5	0.0027	−0.0026	−0.0071
		10	−0.0022	0.0053	0.0032
	5%	0.5	0.0045	−0.0060	−0.0011
		1	0.0055	0.0067	0.0052
		5	−0.0052	−0.0025	−0.0022
		10	−0.0040	0.0015	−0.0031

TABLE XVII
THE EFFECT OF DIFFERENT AGRS IN TERMS OF DEFENDING AGAINST
MODEL POISONING ATTACKS. ACROSS THESE EXPERIMENTS, THE FL
CLIENTS ARE SUBJECT TO A LABEL-BASED NON-IID DIRICHLET
DISTRIBUTION WITH $\alpha = 0.5$.

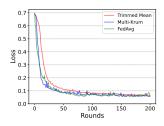| Setting	AGR	MIN-MAX	MIN-SUM
Spam with $M = 1\%$	FedAVG	0.0131	0.0107
	Trimmed Mean	0.0014	0.0009
	Multi-Krum	0.0028	0.0077

Android malware detection (with the highest impact achieved by MIN-SUM at 0.53%). We also see some negative but minor values for the attack impact, which were caused by the inherent randomness of our experiments and suggest the respective model poisoning attacks have no impact on the model performance.
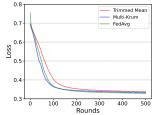
Model poisoning attacks under label-based non-IID distribution. As FL clients in realistic security training scenarios tend to be subject to label-based non-IID data distributions, we then moved on to evaluate model poisoning attacks under different label-based non-IID data distributions. Again, the Dirichlet distribution is used to profile the extent of label-based non-IID and various α values were considered: $\{0.5, 1, 5, 10\}$. Table XVI presents the attack impact of model poisoning under different attack settings of $M \in \{1\%, 5\%\}$. Similar to the default FL scenario, practical model poisoning attacks ($M \leq 1\%$) under label-based non-IID scenarios have incurred a very constrained impact on security detection tasks. Specifically, on one hand, none of the three model poisoning attacks have any impact on the Android malware detection task. On the other hand, for SMS spam detection, the MIN-MAX attack has led to the maximum decrease of accuracy by 2.50%, which however is still not much.

Defending against effective model poisoning attacks via robust AGRs. As some practical model poisoning attacks have resulted in a minor performance degradation for the SMS

TABLE XVIII
THE PERFORMANCE OF THREAT DETECTION MODELS THAT ARE
FL-TRAINED WITH DIFFERENT AGRS.

Task	Scenario	Trimmed Mean	Multi-Krum	FedAVG
Spam	Cross-device	97.47%	98.03%	98.32%
Malware	Cross-device	97.34%	97.74%	98.49%

(a) Spam in cross-device FL. (b) Malware in cross-device FL.

Fig. 9. The FL training processes of different AGRs in the default cross-device FL settings.

spam detection task, we further explored the effect of robust AGRs on defending against such model poisoning attacks. Particularly, two robust AGRs (Trimmed Mean and Multi-Krum) were selected under our experiments. As shown in Table XVII, both Trimmed Mean and Multi-krum work well in terms of decreasing the attack impact. Particularly, Trimmed Mean has decreased the attack impact of MIN-MAX from 1.31% to 0.14% while the attack impact of MIN-SUM has decreased from 1.07% to 0.09%.

Model poisoning with impractically high fractions of compromised FL clients under label-based non-IID distribution. Furthermore, we also evaluated impractical fractions of compromised FL clients with $M \in \{10\%, 20\%, 30\%, 40\%\}$, and the detailed results are presented in Appendix C. The key messages we can distill from these results are two-fold. On one hand, model poisoning attacks under impractically high M values can lead to not only significant performance degradation, but also even convergence failures. For instance, when the fraction of compromised FL clients is 40%, the MIN-SUM model poisoning attack can degrade the SMS spam detection by 7.56% in accuracy as well as causing convergence failures for the Android malware detection task. On the other hand, the defensive effect of robust AGRs varies significantly across different M settings and security prediction tasks, which calls for task-specific and scenario-specific prior evaluations before deploying robust AGRs. Particularly, for some attacking scenarios (e.g., MIN-SUM attack at $M = 40\%$), robust AGRs can decrease the attack impact from over 45% to only 1%, while in some other attacking settings, they can have no defensive effect or even increase the attack impact.

VII. THE EFFICIENCY OF FL

In this section, we summarize the efficiency issues as observed in our FL experiments, and highlight a few promising strategies to address these issues.

TABLE XIX
THE CONVERGENCE OF ROBUST AGRs FOR THE DEFAULT FL SETTINGS.

Task	Scenario	Trimmed Mean		Multi-Krum	
		Rounds[1]	Time[2]	Rounds	Time
Spam	Cross-device	218.18%	195.61%	101.52%	121.58%
Malware	Cross-device	122.58%	118.73%	79.03%	75.75%

[1] The ratio of the number of FL rounds over that of FedAVG.
[2] The ratio of time cost for the respective robust AGR over that of FedAVG.

> **Finding IX**: *When enabled by default for FL training, Multi-Krum, a robust AGR, has a minor overhead for the model performance and the convergence time.*

The (in)efficiency of robust AGRs. As revealed by our byzantine resilience evaluation, robust AGRs can help mitigate attacks of both model poisoning and data poisoning to some extent in cross-device FL. However, it is unclear whether such a benefit makes it worth to enable robust AGRs by default in a routine FL training process, since model poisoning attacks may not always be present but efficiency matters a lot in FL training. We thus moved to profile the overheads of aforementioned two robust AGRs along with a direct comparison with FedAVG under the default FL settings. Table XVIII presents the accuracy stats of the resulting models and we can see enabling robust AGRs have a minor performance impact.

We then looked into the training processes and compared different AGRs with regards to the training time and the number of convergence rounds. We ran each settings for three times and obtained the average. As shown in Figure 9 and Table XIX, Trimmed Mean has led to both a *slow* training process and a notable increase in training time. When compared with FedAVG, it increased the number of cross-device FL training rounds by 118.18% and 22.58% for SMS spam detection and Android malware detection respectively. Similarly, the time cost also increased by 95.61% and 18.73%. On the other hand, the overheads of Multi-Krum appear to be minor across the two security prediction tasks. For SMS spam detection, Multi-Krum has increased the time of cross-device FL training by only 21.58% while barely increasing the rounds needed to converge. However, for Android malware detection, it has made the training process more efficient along with fewer training rounds and training time. A likely explanation is that our SMS spam detection model is much larger than the Android malware detection model, and the large-volume parameters can make it costly to calculate Euclidean distance between model updates, which however is necessary in Multi-Krum when selecting out qualified model updates. Based on these results, we conclude that Multi-Krum can be enabled by default in cross-device FL training so as to defend possible poisoning attacks while incurring few or even no overheads, especially when the model under training is of a small size.

> **Finding X**: *A label-based non-IID data distribution can incur a significant delay in convergence, which however can be effectively mitigated through the bootstrapping strategy.*
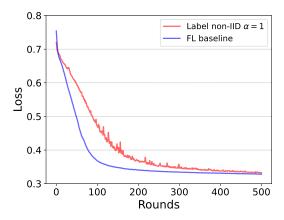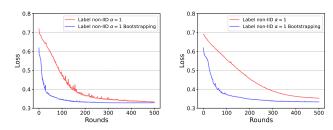

Fig. 10. The convergence processes of the baseline model (quantity-based non-IID with $\alpha = 1$) and the model under label-based non-IID distribution ($\alpha = 1$). Both models are for Android malware detection in cross-device FL.

The convergence delay due to label-based non-IID data distribution. What is also observed is that extreme label-based non-IID data distributions can lead to notable convergence delay in cross-device FL. For instance, as illustrated in Figure 10 for the task of Android malware detection, the label-based non-IID distribution with $\alpha = 1$ has increased the cross-device FL training process by 200 more rounds when compared to the training process under the default FL setting.

Mitigating the convergence delay through bootstrapping. We consider a bootstrapping strategy to address the aforementioned additional delays caused by non-IID data distributions. In a nutshell, our bootstrapping strategy first pre-trains an initial model using server-side publicly available datasets (e.g., open source Android malware datasets), and then fine-tunes the pre-trained model using local datasets from FL clients. Similar ideas have been widely considered in various fields of FL, e.g, personalised FL [82].

To evaluate the effectiveness of this bootstrapping strategy, we first trained a CNN-based Android malware detection model using the Genome Android malware dataset in the central server to simulate the pre-training step. This pre-trained model achieved 98.67% in accuracy, 99.23% in precision and 98.47% in recall when evaluated on the test part of the Genome dataset at the small cost of training for 30 rounds. We then tested the model on our held-out testing dataset that is default for the Android malware detection. It gained only 68.71% in accuracy, 95.95% in precision and 39.06% in recall. We then considered this pre-trained but weak model as the initial global model in the FL scenario, and applied FL training to it, using a label-based non-IID distribution with $\alpha = 1$. In addition to trying the default FL setting of 200 clients, we also tested a setting with 500 clients. The convergence process is shown in Figure 11, along with a comparison to their respective non-bootstrapped counterparts. We can see that the bootstrapping settings have effectively improved not only the convergence speed but also the stability in label-based non-IID scenarios. These results suggest the effectiveness of using bootstrapping

(a) Cross-device FL with 200 clients. (b) Cross-device FL with 500 clients.

Fig. 11. The convergence processes for Android malware detection models trained with or without bootstrapping. Label non-IID $\alpha = 1$ denotes the label-based non-IID distribution with $\alpha = 1$.

in FL to speed up and stabilize the convergence of FL training, especially for extreme non-IID data distributions.

VIII. Discussion

Limitations and Future works. In this study, we focus on evaluating the feasibility of FL for two representative and privacy-sensitive security prediction tasks, with regards to effectiveness, byzantine robustness, and efficiency. As discussed above, security tasks can still vary to a notable extent in their resilience to byzantine clients and extreme non-IID data distribution. Therefore, it is worth further evaluating more representative security prediction tasks in FL scenarios, so as to identify the case-by-case deployment strategies customized for each given security prediction task. On the other hand, cyber threat detection tasks can share many common properties, e.g., the threat model, the behavior patterns of the attacker, etc. We thus believe our findings and observations can likely be generalized to tasks sharing similar properties, e.g., the detection of Email/Tweet spam, and classification of malware of other platforms.

Furthermore, for both tasks of SMS spam classification and Android malware detection, we composed historical datasets from multiple sources, due to the unavailability of a threat dataset that is both up-to-date and balanced. However, as these historical datasets are either out-of-dated or imbalanced, the composed dataset may still have a different data distribution when compared with the real-world and up-to-date data. However, this issue appears to be unavoidable and its impact on our FL evaluations may deserve further investigation.

Besides, when evaluating the byzantine resilience, we focus on untargeted poisoning attacks since it is more challenging and can be more destructive when compared with targeted poisoning attacks. However, targeted poisoning attacks may still occur for FL-based security prediction tasks and we leave it as our future works to further explore the byzantine resilience of FL against targeted poisoning attacks. Then, when it comes to the efficiency of FL, we focus on the computational aspect, and leave it as a future work to evaluate the communication efficiency of FL-based security prediction tasks, especially in realistic deployment settings.

Data and code release. We released the source code for reproducing all aforementioned FL experiments along with the respective datasets. The repository link is https://github.com/ChaseSecurity/Fostering_Cyber_Threat_Detection_Through_FL.

IX. Concluding Remarks

In this study, we have extensively explored the feasibility of applying federated learning to cyber threat detection tasks. Our evaluations have demonstrated that FL is both effective and byzantine-resilient when applied to representative threat detection tasks, and its efficiency issues can be well addressed through a bootstrapping strategy. To conclude, FL is a promising option for enabling privacy-preserving cyber threat detection.

References

[1] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in Ndss, vol. 14, 2014, pp. 23–26.

[2] N. McLaughlin, J. M. Del Rincon, B. J. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupe, and G. J. Ahn, "Deep android malware detection," CODASPY 2017 - Proceedings of the 7th ACM Conference on Data and Application Security and Privacy, pp. 301–308, 2017.

[3] P. K. Roy, J. P. Singh, and S. Banerjee, "Deep learning to filter sms spam," Future Generation Computer Systems, vol. 102, pp. 524–533, 2020.

[4] A. Ghourabi, M. A. Mahmood, and Q. M. Alzubi, "A hybrid cnn-lstm model for sms spam detection in arabic and english messages," Future Internet, vol. 12, no. 9, p. 156, 2020.

[5] O. Abayomi-Alli, S. Misra, and A. Abayomi-Alli, "A deep learning method for automatic sms spam classification: Performance of learning algorithms on indigenous dataset," Concurrency and Computation: Practice and Experience, p. e6989, 2022.

[6] X. Liu, H. Lu, and A. Nayak, "A spam transformer model for sms spam detection," IEEE Access, vol. 9, pp. 80 253–80 263, 2021.

[7] T. Wu, S. Liu, J. Zhang, and Y. Xiang, "Twitter spam detection based on deep learning," in Proceedings of the australasian computer science week multiconference, 2017, pp. 1–8.

[8] N. Lyamin, D. Kleyko, Q. Delooz, and A. Vinel, "Ai-based malicious network traffic detection in vanets," IEEE Network, vol. 32, no. 6, pp. 15–21, 2018.

[9] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "Corrauc: A malicious bot-iot traffic detection method in iot network using machine-learning techniques," IEEE Internet of Things Journal, vol. 8, no. 5, pp. 3242–3254, 2020.

[10] C. J. Hoofnagle, B. Van Der Sloot, and F. Z. Borgesius, "The european union general data protection regulation: what it is and what it means," Information & Communications Technology Law, vol. 28, no. 1, pp. 65–98, 2019.

[11] E. L. Harding, J. J. Vanto, R. Clark, L. Hannah Ji, and S. C. Ainsworth, "Understanding the scope and impact of the california consumer privacy act of 2018," Journal of Data Protection & Privacy, vol. 2, no. 3, pp. 234–253, 2019.

[12] "Privacy, deception and device abuse," https://support.google.com/googleplay/android-developer/topic/9877467.

[13] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in 26th USENIX security symposium (USENIX security 17), 2017, pp. 625–642.

[14] K. Xu, Y. Li, R. Deng, K. Chen, and J. Xu, "Droidevolver: Self-evolving android malware detection system," in 2019 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2019, pp. 47–62.

[15] S. Tang, X. Mi, Y. Li, X. Wang, and K. Chen, "Clues in tweets: Twitter-guided discovery and analysis of sms spam," in Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 2751–2764.

[16] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4. Springer, 2019, pp. 92–104.

[17] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," arXiv preprint arXiv:1811.03604, 2018.

[18] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf

[19] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in International Conference on Machine Learning. PMLR, 2019, pp. 634–643.

[20] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25. Springer, 2020, pp. 480–501.

[21] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," Advances in Neural Information Processing Systems, vol. 32, 2019.

[22] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" arXiv preprint arXiv:1911.07963, 2019.

[23] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in International Conference on Artificial Intelligence and Statistics. PMLR, 2020, pp. 2938–2948.

[24] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," Advances in Neural Information Processing Systems, vol. 33, pp. 16 070–16 084, 2020.

[25] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of sms spam filtering: new collection and results," in Proceedings of the 11th ACM symposium on Document engineering, 2011, pp. 259–262.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[27] M. Koroteev, "Bert: a review of applications in natural language processing and understanding," arXiv preprint arXiv:2103.11943, 2021.

[28] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in Proceedings of the 35th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5650–5659. [Online]. Available: https://proceedings.mlr.press/v80/yin18a.html

[29] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, vol. 54, feb 2017. [Online]. Available: http://arxiv.org/abs/1602.05629

[30] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," International journal of medical informatics, vol. 112, pp. 59–67, 2018.

[31] I. Dayan, H. R. Roth, A. Zhong, A. Harouni, A. Gentili, A. Z. Abidin, A. Liu, A. B. Costa, B. J. Wood, C.-S. Tsai et al., "Federated learning for predicting clinical outcomes in patients with covid-19," Nature medicine, vol. 27, no. 10, pp. 1735–1743, 2021.

[32] T. Zhang, C. He, T. Ma, L. Gao, M. Ma, and S. Avestimehr, "Federated learning for internet of things," in Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, 2021, pp. 413–419.

[33] E. M. Campos, P. F. Saura, A. González-Vidal, J. L. Hernández-Ramos, J. B. Bernabé, G. Baldini, and A. Skarmeta, "Evaluating Federated Learning for intrusion detection in Internet of Things: Review and challenges," Computer Networks, vol. 203, no. November 2021, p.

108661, feb 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1389128621005405

[34] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," 2018. [Online]. Available: http://arxiv.org/abs/1812.06127

[35] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in Proceedings of the 37th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5132–5143. [Online]. Available: https://proceedings.mlr.press/v119/karimireddy20a.html

[36] D. Dimitriadis, K. Kumatani, R. Gmyr, Y. Gaur, and S. E. Eskimez, "Federated transfer learning with dynamic gradient aggregation," arXiv preprint arXiv:2008.02452, 2020.

[37] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. Vincent Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," Advances in Neural Information Processing Systems, vol. 2020-Decem, pp. 1–34, 2020.

[38] A. Hatamizadeh, H. Yin, P. Molchanov, A. Myronenko, W. Li, P. Dogra, A. Feng, M. G. Flores, J. Kautz, D. Xu, and H. R. Roth, "Do Gradient Inversion Attacks Make Federated Learning Unsafe?" IEEE Transactions on Medical Imaging, vol. 42, no. 7, pp. 2044–2056, 2023.

[39] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" Advances in Neural Information Processing Systems, vol. 33, pp. 16 937–16 947, 2020.

[40] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," Advances in neural information processing systems, vol. 32, 2019.

[41] C. Zhou, Y. Gao, A. Fu, K. Chen, Z. Dai, Z. Zhang, M. Xue, and Y. Zhang, "PPA: Preference Profiling Attack Against Federated Learning," no. March, feb 2022. [Online]. Available: http://arxiv.org/abs/2202.04856

[42] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 739–753.

[43] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1175–1191.

[44] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 1253–1269.

[45] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," IEEE Journal on Selected Areas in Information Theory, vol. 2, no. 1, pp. 479–489, 2021.

[46] Y. Guo, A. Polychroniadou, E. Shi, D. Byrd, and T. Balch, "Microfedml: Privacy preserving federated learning for small weights," Cryptology ePrint Archive, 2022.

[47] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning," Proceedings of the 29th USENIX Security Symposium, pp. 1623–1640, nov 2019. [Online]. Available: http://arxiv.org/abs/1911.11815

[48] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in NDSS, 2021.

[49] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in 2022 IEEE Symposium on Security and Privacy (SP). IEEE, 2022, pp. 1354–1371.

[50] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 1, no. 2, pp. 1–25, 2017.

[51] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," arXiv preprint arXiv:1802.10116, 2018.

[52] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," IEEE Transactions on Signal Processing, vol. 70, pp. 1142–1154, 2022.

18

[53] R. Guerraoui, S. Rouault et al., "The hidden vulnerability of distributed learning in byzantium," in International Conference on Machine Learning. PMLR, 2018, pp. 3521–3530.

[54] Q. Xu, E. W. Xiang, Q. Yang, J. Du, and J. Zhong, "Sms spam detection using noncontent features," IEEE Intelligent Systems, vol. 27, no. 6, pp. 44–51, 2012.

[55] N. N. A. Sjarif, N. F. M. Azmi, S. Chuprat, H. M. Sarkan, Y. Yahya, and S. M. Sam, "Sms spam message detection using term frequency-inverse document frequency and random forest algorithm," Procedia Computer Science, vol. 161, pp. 509–515, 2019.

[56] A. Rahali, A. H. Lashkari, G. Kaur, L. Taheri, F. Gagnon, and F. Massicotte, "Didroid: Android malware classification and characterization using deep image learning," in 2020 The 10th international conference on communication and network security, 2020, pp. 70–82.

[57] L. Wang, H. Wang, R. He, R. Tao, G. Meng, X. Luo, and X. Liu, "Malradar: Demystifying android malware in the new era," Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 6, no. 2, pp. 1–27, 2022.

[58] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "Mamadroid: Detecting android malware by building markov chains of behavioral models," arXiv preprint arXiv:1612.04433, 2016.

[59] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Maldozer: Automatic framework for android malware detection using deep learning," Digital Investigation, vol. 24, pp. S48–S59, 2018.

[60] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," Journal of Ambient Intelligence and Humanized Computing, vol. 10, pp. 3035–3043, 2019.

[61] H. Gao, S. Cheng, and W. Zhang, "Gdroid: Android malware detection and classification with graph convolutional network," Computers & Security, vol. 106, p. 102264, 2021.

[62] Y. Hei, R. Yang, H. Peng, L. Wang, X. Xu, J. Liu, H. Liu, J. Xu, and L. Sun, "Hawk: Rapid android malware detection through heterogeneous graph attention networks," IEEE Transactions on Neural Networks and Learning Systems, 2021.

[63] R. Nix and J. Zhang, "Classification of android apps and malware using deep neural networks," in 2017 International joint conference on neural networks (IJCNN). IEEE, 2017, pp. 1871–1878.

[64] J. Sidhpura, P. Shah, R. Veerkhare, and A. Godbole, FedSpam: Privacy Preserving SMS Spam Prediction. Springer Nature Singapore, 2023, vol. 1793 CCIS. [Online]. Available: http://dx.doi.org/10.1007/978-981-99-1645-0_5

[65] D. Srinivasa Rao and E. Ajith Jubilson, "SMS Spam Detection Using Federated Learning," Lecture Notes on Data Engineering and Communications Technologies, vol. 163, pp. 547–562, 2023.

[66] R. H. Hsu, Y. C. Wang, C. I. Fan, B. Sun, T. Ban, T. Takahashi, T. W. Wu, and S. W. Kao, "A Privacy-Preserving Federated Learning System for Android Malware Detection Based on Edge Computing," Proceedings - 2020 15th Asia Joint Conference on Information Security, AsiaJCIS 2020, pp. 128–136, 2020.

[67] W. Fang, J. He, W. Li, X. Lan, Y. Chen, T. Li, J. Huang, and L. Zhang, "Comprehensive Android Malware Detection Based on Federated Learning Architecture," IEEE Transactions on Information Forensics and Security, vol. 18, pp. 3977–3990, 2023.

[68] A. Chaudhuri, A. Nandi, and B. Pradhan, "A Dynamic Weighted Federated Learning for Android Malware Classification," Lecture Notes in Networks and Systems, vol. 627 LNNS, pp. 147–159, 2023.

[69] H. Fereidooni, A. Dmitrienko, P. Rieger, M. Miettinen, A.-R. Sadeghi, and F. Madlener, "Fedcri: Federated mobile cyber-risk intelligence," in Network and Distributed Systems Security (NDSS) Symposium, 2022.

[70] I. Inuwa-Dutse, M. Liptrott, and I. Korkontzelos, "Detection of spam-posting accounts on twitter," Neurocomputing, vol. 315, pp. 496–511, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218308798

[71] O. Varol, E. Ferrara, C. Davis, F. Menczer, and A. Flammini, "Online Human-Bot Interactions: Detection, Estimation, and Characterization," Proceedings of the International AAAI Conference on Web and Social Media, vol. 11, no. 1, pp. 280–289, may 2017. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14871

[72] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in 31st USENIX Security Symposium (USENIX Security 22). Boston, MA: USENIX

Association, Aug. 2022, pp. 3971–3988. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/arp

[73] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in 2018 International Carnahan conference on security technology (ICCST). IEEE, 2018, pp. 1–7.

[74] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in Proceedings of the 13th International Conference on Mining Software Repositories, ser. MSR '16. New York, NY, USA: ACM, 2016, pp. 468–471. [Online]. Available: http://doi.acm.org/10.1145/2901739.2903508

[75] G. Sood, virustotal: R Client for the virustotal API, 2021, r package version 0.2.2.

[76] H. Zhang, W. Zhang, Z. Lv, A. K. Sangaiah, T. Huang, and N. Chilamkurti, "MALDC: a depth detection method for malware based on behavior chains," World Wide Web, vol. 23, no. 2, pp. 991–1010, 2020.

[77] C. Jindal, C. Salls, H. Aghakhani, K. Long, C. Kruegel, and G. Vigna, "Neurlux: Dynamic malware analysis without feature engineering," ACM International Conference Proceeding Series, pp. 444–455, 2019.

[78] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated Learning on Non-IID Data Silos: An Experimental Study," Proceedings - International Conference on Data Engineering, vol. 2022-May, pp. 965–978, 2022.

[79] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, H. L. Kwing, T. Parcollet, P. P. d. Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," arXiv preprint arXiv:2007.14390, 2020.

[80] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards Federated Learning at Scale: System Design," feb 2019. [Online]. Available: http://arxiv.org/abs/1902.01046

[81] "Dirichlet distribution," https://en.wikipedia.org/wiki/Dirichlet_distribution.

[82] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized Federated Learning: A Meta-Learning Approach," pp. 1–29, feb 2020. [Online]. Available: http://arxiv.org/abs/2002.07948

APPENDIX A
MORE RESULTS ON THE EFFECTIVENESS OF FL

A. More performance results in the consistent label imbalance Scenarios.

Table XX presents the precision performance for FL-trained models in consistent label imbalance scenarios while Table XXI presents the recall performance.

TABLE XX
THE PRECISION STATS OF SECURITY MODELS UNDER CONSISTENT LABEL IMBALANCE.

Task	Model	PNR		
		0.25	1	4
Spam	Central	0.9889	0.9912	0.9842
	Cross-Device FL	0.9930	0.9884	0.9454
	Cross-Silo FL	0.9952	0.9923	0.9722
Malware	Central	0.9703	0.9779	0.7866
	Cross-Device FL	0.9742	0.9837	0.5000
	Cross-Silo FL	0.9810	0.9846	0.5013

B. Metrics to profile the stability of FL convergence in non-IID scenarios

Given the sequence of models trained during the last 10 rounds of a FL training process, let's define ACC_{STD} as the standard deviation for their respective accuracy values, PRE_{STD} for the precision metric, and REC_{STD} for the recall metric. Table XXII lists the values of these metrics for all

19

TABLE XXI
THE RECALL STATS OF SECURITY MODELS UNDER CONSISTENT LABEL
IMBALANCE.

Task	Model	PNR		
		0.25	1	4
Spam	Central	0.9841	0.9882	0.9905
	Cross-Device FL	0.9474	0.9829	0.9852
	Cross-Silo FL	0.9762	0.9886	0.9926
Malware	Central	0.9608	0.9948	1.0000
	Cross-Device FL	0.9871	0.9967	1.0000
	Cross-Silo FL	0.9699	0.9944	1.0000

the aforementioned non-IID scenarios. In Figure 6, we can see that the whole converging process is unstable specially in the cross-device FL. It further strengthens our observation that non-IID distributions can lead to higher instability of the FL training process in terms of convergence.

TABLE XXII
THE (IN)STABILITY OF THE CONVERGENCE OF FL TRAINING.

Task	Non-IID	Params	ACC_{STD}	PRE_{STD}	REC_{STD}
Spam	Default	$\alpha = 1$	0.0007	0.0042	0.0036
	Quantity	$\alpha = 0.5$	0.0009	0.0037	0.0024
	Label	$\alpha = 0.5$	0.0081	0.0201	0.0116
	CLI	$PNR = \frac{1}{4}$	0.0034	0.0036	0.101
Malware	Default	$\alpha = 1$	0.0005	0.0007	0.0011
	Quantity	$\alpha = 0.5$	0.0012	0.0022	0.0000
	Label	$\alpha = 0.5$	0.0012	0.0021	0.0000
	CLI	$PNR = \frac{1}{4}$	0.0008	0.0016	0.0016

APPENDIX B
MORE RESULTS OF DATA POISONING ATTACKS

A. Data poisoning attacks in label-based non-IID scenarios

Table XXIII presents the attack impact of data poisoning attacks in various label-based non-IID scenarios.

B. The FL training processes under data poisoning attacks

Figure 12 presents the training process for security detection models under data poisoning attacks.

When $M = 40\%$, we further explored the effect of different sample poisoning rates p, and Figure 13 presents the respective training processes for security detection models under such data poisoning attacks.

APPENDIX C
MODEL POISONING UNDER IMPRACTICAL SETTINGS

Table XXIV presents the results of model poisoning attacks under high but impractical attacking settings, i.e., $M \in \{10\%, 20\%, 30\%, 40\%\}$. As we can see, the higher the fraction of compromised FL clients (M), the more the attack impact is, which applies to both security prediction tasks and all three attack algorithms, and thus is aligned with previous works [21], [48]. Besides, among the three attack algorithms, LIE has negligible attack impact in almost all experiments except for attacking the Android malware detection task with
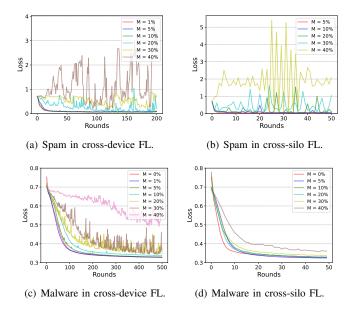

(a) Spam in cross-device FL.   (b) Spam in cross-silo FL.

(c) Malware in cross-device FL.   (d) Malware in cross-silo FL.

Fig. 12. The training process of data poisoning via label flipping for the default FL settings of M.

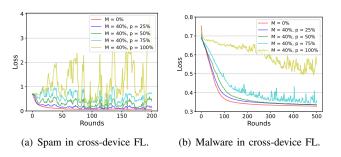(a) Spam in cross-device FL.   (b) Malware in cross-device FL.

Fig. 13. The training process of data poisoning via label flipping for the default FL settings of p.

$M = 40\%$, while MIN-MAX and MIN-SUM performed much better.

Then, when it comes to the byzantine resilience of security prediction tasks, the two security prediction tasks vary significantly in their byzantine resilience (adversarial robustness). Particularly, when $M \leq 20\%$, the task of Android malware detection tends to be more resistant against model poisoning attacks than SMS spam detection. However, as M continues to increase, the malware detection model drops significantly in its performance or even fails to converge, while the SMS spam detection model becomes more resilient and suffers from less than 8% decrease in accuracy across all attacking experiments. Such a robustness difference may be attributed to their difference in model architecture as well as the number of model parameters. Specifically, the SMS spam detection model, a transformer model, has 167 million parameters, which is much larger than the CNN model for malware detection.

The effect of robust AGR on mitigating model poisoning attacks with impractically high fractions of compromised FL clients. Although above model poisoning attacks are unlikely to occur in real-world FL scenarios due to their

TABLE XXIII
The attacking impact in accuracy decrease for **DATA POISONING ATTACKS** under label-based non-IID distribution.

Poisoning Setting	Non-IID	Spam		Malware	
		Device	Silo	Device	Silo
$M = 5\%, p = 50\%$	Balanced	0.0013 ± 0.0008	-0.0002 ± 0.0014	0.0093 ± 0.0008	0.0044 ± 0.0017
	$\alpha = 0.5$	-0.0024 ± 0.0072	-0.0001 ± 0.0009	0.0020 ± 0.0024	0.0031 ± 0.0020
	$\alpha = 1$	-0.0009 ± 0.0047	-0.0003 ± 0.0007	0.0006 ± 0.0033	-0.0012 ± 0.0024
	$\alpha = 5$	-0.0002 ± 0.0012	-0.0001 ± 0.0005	0.0073 ± 0.0009	0.0002 ± 0.0028
	$\alpha = 10$	-0.0014 ± 0.0012	0.0002 ± 0.0006	0.0056 ± 0.0009	-0.0109 ± 0.0013
$M = 5\%, p = 100\%$	Balanced	0.0023 ± 0.0022	-0.0007 ± 0.0010	0.0020 ± 0.0012	0.0007 ± 0.0011
	$\alpha = 0.5$	-0.0018 ± 0.0092	0.0006 ± 0.0007	0.0013 ± 0.0037	0.0134 ± 0.0014
	$\alpha = 1$	-0.0001 ± 0.0042	0.0013 ± 0.0009	-0.0014 ± 0.0040	-0.0052 ± 0.0028
	$\alpha = 5$	0.0019 ± 0.0018	-0.0004 ± 0.0010	0.0096 ± 0.0010	-0.0011 ± 0.0019
	$\alpha = 10$	-0.0012 ± 0.0014	-0.0007 ± 0.0008	0.0008 ± 0.0016	0.0000 ± 0.0016

TABLE XXIV
The attacking impact of **MODEL POISONING ATTACKS** with high but impractical M.

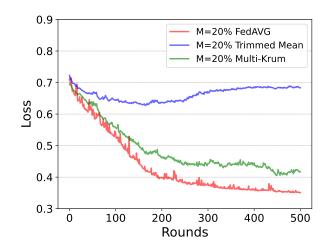| Task	M	Non-IID	LIE	MIN-MAX	MIN-SUM
Spam	10%	0.5	-0.0020	0.0231	0.0226
		1	0.0003	0.0314	0.0397
		5	0.0011	0.0745	0.0632
		10	0.0003	0.0672	0.0669
	20%	0.5	-0.0033	0.0281	0.0196
		1	-0.0000	0.0456	0.0387
		5	-0.0002	0.0400	0.0585
		10	-0.0009	0.0757	0.0662
	30%	0.5	-0.0009	0.0378	0.0365
		1	-0.0020	0.0320	0.0344
		5	0.0018	0.0812	0.0617
		10	-0.0001	0.0457	0.0649
	40%	0.5	-0.0005	0.0396	0.0311
		1	0.0022	0.0452	0.0450
		5	0.0035	0.0478	0.0555
		10	-0.0007	0.0725	0.0756
Malware	10%	0.5	0.0056	0.0027	-0.0054
		1	0.0039	0.0089	0.0062
		5	-0.0015	-0.0055	-0.0074
		10	0.0001	-0.0002	-0.0024
	20%	0.5	0.0072	0.0205	0.0145
		1	0.0067	0.0134	0.0115
		5	0.0001	0.0024	0.0068
		10	-0.0028	0.0045	-0.0038
	30%	0.5	0.0046	0.1025	0.3148
		1	0.0079	0.0262	0.1964
		5	-0.0016	0.0101	0.0081
		10	-0.0012	0.0080	0.0059
	40%	0.5	0.0106	0.4312	0.4760
		1	0.0041	0.4639	0.4551
		5	0.0009	0.0242	0.4689
		10	0.0019	0.0088	0.0236

Fig. 14. The FL training processes with or without the enforcement of robust AGRs, for Android malware detection, under model poisoning attacks.

TABLE XXV
The attacking impact of **MODEL POISONING** attacks, for Android malware detection, under **DIFFERENT AGR ALGORITHMS** while the client-side data follows the label-based Dirichlet distribution with $\alpha = 0.5$.

M	AGR	LIE	MIN-MAX	MIN-SUM
10%	FedAVG	0.0056	0.0027	-0.0054
	Trimmed Mean	0.0038	0.0093	0.0165
	Multi-Krum	-0.0027	0.0089	0.0021
20%	FedAVG	0.0072	0.0205	0.0145
	Trimmed Mean	0.0069	0.0313	0.1562
	Multi-Krum	0.0195	0.0164	0.0154
30%	FedAVG	0.0046	0.1025	0.3148
	Trimmed Mean	0.0204	0.4721	0.4798
	Multi-Krum	0.0188	0.4575	0.0156
40%	FedAVG	0.0106	0.4312	0.4760
	Trimmed Mean	0.0352	0.4798	0.4800
	Multi-Krum	0.0347	0.4760	0.0105

[1] Each cell denotes the decrease in model accuracy that is achieved by a model poisoning attack under a specific AGR and label-based non-IID data distribution.

assumption on high fractions of compromised FL clients, we still evaluated the effectiveness of multiple robust AGRs that have been claimed in previous works [51], [18] for defending against such model poisoning attacks. However, as listed in Table XXV, for most attacking settings on Android malware detection, neither Trimmed Mean nor Multi-Krum has notable defensive impact. Instead, as M gets larger, they even intensify the impact of the respective attack, such as making the training process fail to converge, as further illustrated in Figure 14.