FLC-NET: Federated Lightweight Network for Early Discovery of Malware in Resource-constrained IoT

Denis A. Pustokhin[1], Irina V. Pustokhina[*2]

[1] Department of Logistics, State University of Management, 109542, Moscow, Russia
[2] Department of Entrepreneurship and Logistics, Plekhanov Russian University of Economics, 117997, Moscow, Russia
Emails: da_pustohin@guu.ru; pustohina.IV@rea.ru

Abstract

In the past few years, billions of Internet of Things (IoT) devices that lacked adequate security procedures were created and deployed, and more of these devices are on the way as a result of the development of Beyond 5G technologies. Because of their susceptibility to malware, there is a pressing need for reliable methods that can identify infected IoT devices within networks. Precise and early identification of IoT malware is inevitable to achieve IoT security. Nevertheless, prevailing studies of IoT malware detection mostly support certain platforms, need complicated deep learning (DL) models to achieve efficiency, and are centrally trained on the device. The purpose of this study is to introduce a new Federated Learning (FL) Framework, which has been given the name FLC-NET, in order to train numerous distributed edge devices to identify malware cooperatively. After the malware binaries have been encoded into image representations using FLC-NET, a lightweight convolutional network known as LC-NET is introduced to model these malware patterns directly from the image data without any data engineering being required. Because of its lightweight design, LC-NET is suited for use in devices with limited resource availability. After that, sophisticated adversarial training will be offered on FLC-NET in order to collect defensive knowledge against adversarial samples from a variety of clients who will be participating. The FLC-NET is experimentally evaluated on the public malware dataset, and it is demonstrated efficient (Accuracy: 96.1%, f1-score: 95.5), effective, scalable, and resistant to adversarial attacks.

Keywords: Malware Detection; Federated Learning; Deep Learning; Edge/Fog Computing; adversarial attacks.

1. Introduction

The rapid evolution of the Internet of Things (IoT) technologies makes it an essential contributor to sustainable development in different living sectors such as smart homes, transportation, healthcare, education, etc. [1]. It is anticipated that there will be approximately 64 billion Internet of Things devices online by the year 2025 [1]. Unquestionably, the widespread distribution of these devices is bringing about a transformation that is making the world a more hyperconnected place [2]. The IoT paradigm, in conjunction with the newly developed 5G and Beyond 5G (B5G) network technologies, is facilitating new application situations and industries that have never been seen before [3]. Some examples of these new application scenarios and businesses include Smart Cities and Industries 4.0. Nevertheless, in the latest days, the number and variety of cyberattacks have expanded, making conventional security techniques obsolete in a relatively short amount of time. This growth has occurred concurrently with the advancements in technological advances. For this reason, maintaining control over the safety of future network

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

settings made possible by B5G protocols provides open difficulties that need to be resolved using contemporary methods.
Watching the behaviors of the device in order to create behavioral fingerprints or profiles is one method that has become more important in the process of detecting devices that have been compromised by malicious software. Fingerprints may be employed to identify deviations that are the result of intrusions or alterations to malicious software [4]. IoT devices have the ability to track a wide variety of behavior sources, including user engagement, computer networking, resource utilization, technology activities, occurrences, and application activities and occasions. Thus, the application of one or the other may be appropriate given the nature of the goal that is to be accomplished. In a more concrete sense, the dimension of network communications is the one that is utilized the most frequently in the research that is done on detecting malware.

Coping with evolution, the security requirements of IoT devices, systems, and applications come to be serious for academia and industry. It is estimated that there would be around 50 billion IoT devices in global use by 2030. Nevertheless, the autonomous, resource-constrained, and internet-connected nature of IoT leads to the adoption of standard security methods impractical. Consequently, an enormous number of IoT devices are exposed to malware, which is serious malicious software threatening the functionality, confidentiality, and integrity of the IoT system. A typical example of that is the Mirai botnet-based DDoS which is a popular IoT malware. By the compromising IoT device, the attacker becomes able to take the control of this device to initiate malicious attacks. Though the spread of IoT brings many benefits it brings hundreds of millions of unsecured devices that are susceptible to malware. A large number of almost unsecured interconnected devices impose serious security challenges in IoT environments.

Deep learning (DL), which is a subsection of machine learning (ML), has revolutionized the application of Internet of Things (IoT) security, and it has shown a great deal of promise for detecting and fighting against assaults and malware [4]. The construction of a collection of features for training in traditional DL research is dependent on domain knowledge as well as feature engineering [5]. It becomes increasingly difficult to manually update the training features to account for ever-changing forms of malware. This is due to the fact that new forms of malware are always being developed and updated with certain specific variations. In general, malware detection algorithms rely mostly on static, dynamic, or hybrid patterns. However, there are exceptions to this rule. When it comes to malware that affects internet-connected devices, the most prevalent detection approaches rely on either high-level static patterns or dynamic conduct patterns. However, because static techniques are dependent on the analysis of assembly code, bytecode, and file structure, they are not suitable for IoT ecosystems [6]. This is due to the fact that the processing units of IoT devices always belong to different architectures, such as Acorn RISC Machine (ARM), Microprocessor without Interlocked Pipelined Stages (MIPS), Crusoe, and so on. On the other hand, dynamic analysis can only analyze the behavior of malware along a specific execution route. As a result, it is difficult to go across all the routes, which makes it necessary for this category of methods to require a high level of time complexity as well as computational resources. Because it is so heavily dependent on manually built features, it is unable to combat the ever-increasing circulation of different strains of malicious software, which is another one of its limitations. Recently, it has been proved that malware may be detected from the raw bytes of their binary files. This ensures that the detection approach is architecture-agnostic and can work well across a variety of platforms. However, the majority of the methods that are employed in this respect apply extensive DL models with millions of parameters approaching tens of millions, which may be inappropriate to train over resource-constrained IoT devices.

Edge computing (EC) and fog computing (FC) are both examples of distributed computing concepts that move computational and network resources closer to end users. EC and FC are also abbreviated as "edge computing" and "fog computing," respectively. In recent years, the idea of edge intelligence has gained a lot of traction thanks to the implementation of DL solutions at devices that are located on the edges of networks. Federated learning, often known as FL, is a paradigm for distributed learning that was developed to enable numerous distributed devices to collaborate in the training of a single model by making use of the data that is stored locally on each device. This makes it possible to save a significant amount of network time and resources that would otherwise be squandered on centralized training on the cloud. FL has brought about a sea change in the way that intelligent solutions for the internet of things are conceived of and implemented. The study of FL for malware detection is in its early stages, despite the fact that it has been quite successful in other areas, such as the detection of intrusions and attacks.

Despite the attempts made to detect IoT malware, there are three primary concerns that are still not addressed in the literature. 1) **Security,** IoT malware causes a broad range of catastrophic impacts ranging from interrupting

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

system/device functionality to committing cybercrimes [7]. **2) Limited resources,** IoT devices are resource-constrained by nature making the complex DL solution impractical to apply in the real world. the complexity of malware detectors is a major barrier to applicability and sustainability. 3) **Efficiency,** refers to the precise detection of malware irrespective of the underlying platform, hardware, or any other settings. Thus, an efficient DL model is required for the IoT devices deployed in the real world to effectively process the data without increasing the error rate [8], [9].

To address the above challenges, this work presents a new federated malware detection framework. The contributions of this work can be pointed out as follow:

- To the best of the author's knowledge, this is the first study to introduce an agnostic FL solution, named FLC-NET, for distributed malware detection in IoT networks.
- In FLC-NET, the bytecode of malware files is transformed into an image representation. Then, a novel lightweight convolutional model, based on depthwise separation, is presented for detecting malware from the generated images in resource-constrained IoT devices.
- Experimental evaluations on real-world IoT malware from public datasets demonstrated the efficiency and effectiveness of the LC-NET and FLC-NET in both centralized and federated scenarios, respectively.

The residual part of this article is organized as follows: Section 2 discusses the pertinent literature. Section 3 primarily discusses the proposed solution. In section 4, we deliberate the experimental details, the results, the analysis, and the discussions. Following, section 5 introduces the primary conclusions of this study and points out future directions.

2. Related Work

With the ubiquitous spread of IoT devices in recent years, scholars and researchers have been devoting great efforts toward malware detection.

A. Deep Learning for Malware Detection

The proliferation of achievements of intelligence-based security poses a demand for supplying some intelligence to IoT devices to identify malware, making DL a promising candidate and favorable solution to detecting and defending malware in IoT [1], [2]. For example, Yuan et al [10] presented a byte-centered malware classification framework in which the malware binaries are firstly converted into Markov images based on bytes transfer probability matrices. Then, two-dimensional CNN was applied to model and classify malware from Markov images. Yuan et al [11] proposed a lightweight convolutional model to learn malware analysis from multidimensional Markov images generated from raw bytes of malware binary. Li et al [12] experimentally validated that adversarial training of ensemble malware detectors can considerably their resistibility against a broad range of adversarial attacks. In addition, Feng et al [13] presented tailored DL models based on LSTM and GRU networks for modeling binary features to detect Android malware before installation on mobile devices instead of detecting them after installation on servers. Besides, Sudhakar et al [14] applied a convolutional network (Res-Net 50) to classify malware with previous knowledge of binary code. The literature contains many studies whose contribution revolves around fine-tuning pre-trained models for classifying malware [15], [16], [17].

B. Federated Learning and Privacy Preservation

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

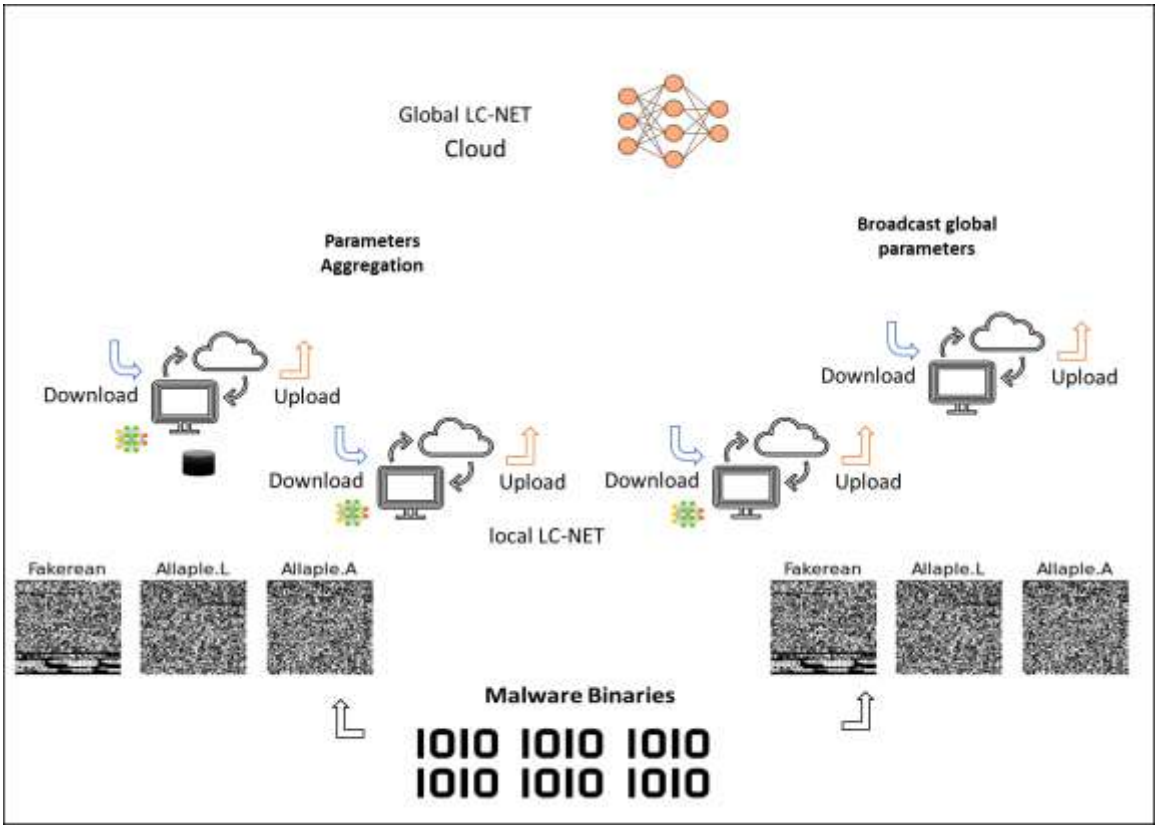

Figure 1: System model for the proposed FED-MAL Framework.

The distributed nature of IoT systems has revolutionalized the way the intelligence system is designed. In this spectrum, FL came to be dominating distributed learning strategy for the majority of IoT applications. Pie et al [18] presented a semi-supervised FL framework, called FedMalDE, for detecting malware using knowledge transfer methods that investigates the inherent relationship between annotated and unannotated data to deduce labels for unlabeled examples. In FedMalDE, a specially designed subgraph aggregated capsule network (SACN) was applied to effectively detect different malicious activities. Besides, Rey et al [19] investigated the opportunities supported by the FL paradigm regarding the detection of IoT malware and explored the security challenges it encountered. The authors also evaluated supervised (MLP) and unsupervised (AE) federated models for detecting malware-compromising IoT devices. Moreover, Makkar [20] developed an FL framework for secure data sharing among different IIoT devices, where the consensus method was presented to allow consensus computing among edge participants. This way, the proposed framework reaches superior proficiency and improved security for training DL models. However, none of the above studies considered the potential of adversarial attacks on the malware detector. Though great attention has been paid to federated intrusion detection [21], however, federated malware detection is still in the infancy stages. Motivated by that, this work contributes to the body of knowledge by proposing a novel federated malware detector, which is described in the next section.

C. Fog Computing

Unlike conventional cloud computing, FC enables serving latency-sensitive calls from IoT devices with little networking overhead, low power, and small delay. Essentially, fog networks can be regarded as a way of offloading basic IoT resources [22]. In this context, FC was integrated to achieve Hierarchical Caching in federated training [23]. Additionally, fog nodes were geospatially integrated into the FL to serve as local aggregators, which were accountable for identifying demographics for sharing place-based information for applications with similar environments [24]. More, in [25], the fog nodes were integrated into FL to act as an intermediary processing unit to deliver local services while helping the cloud aggregate the local updates throughout the training. Motivated by the above promise of FC, the FLC-NET takes to integrate the FC to defend against adversaries during the training.

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

II. PROPOSED FLC-NETEF FRAMEWORK

Away from the centralized DL on either local devices or cloud servers, the proposed system takes the advantage of geographically distributed edge devices and fog nodes to empower the collaborative training of FLC-NET for malware detection. Cloud system, also referred to as the coordination node. It comprises the cloud servers accountable for hyperparameter initialization, initial model training, and broadcasting the initial model to the fog nodes. it aggregates the results from local classifiers during training rounds and accordingly upgrades the parameters of global classifiers. On the other hand, the fog system comprises the fog nodes that act as an intermediate computation between the edge tier and the cloud. It has more powerful computational resources compared with edge devices. These nodes store the private and sensitive data aggregated from the corresponding smart city sector. Each fog device gets the initial classifier from the cloud tier and followingly starts local model training using its private data. Then, they push up the updated local parameters to the cloud server to aggregate the results of distributed training. This process is repeated every communication round till the global model gets converged. Edge system consists of resource-constrained end users' devices that generate the data used for federated training. These devices are very heterogeneous and generate distinct volumes of data with different levels of privacy. These data might encompass private information (i.e., health records, business information, etc.) thereby locally stored on the allocated fog node without transmission over the IoT network.

A. LC-NET.

The design of the proposed LC-NET is briefly comprised of a patch embedding layer and convolutional blocks. Unlike linear embedding in vision transformers, LC-NET applies two-dimensional convolution (Conv2D) to generate patch embeddings to offer better image superiority without incurring lower calculations. Unlike the work [26], t utilizes three standard convolutions and a pixel shuffling layer, the proposed LC-NET generates patch embeddings using two depth-wise separable convolutions (DwConv) to accomplish super-resolution patching. Particularly, a 5×5 convolutional kernel is adopted to produce 32 feature maps from the received malware image I_{mal}, as follows:

$$F_{\mathcal{M}} = Conv_{5\times5}(I_{mal}) \quad (1)$$

Whereas the $Conv_{5\times5}$ denotes the conventional convolution, and $F_{\mathcal{M}}$ denotes the acquired maps of features from the malware image. Next, two DwConv layers are applied to play non-linear projection of obtained features:

$$F_{\mathcal{M}'} = PwConv_{1\times1}(DwConv_{3\times3}(PwConv_{1\times1}(DwConv_{3\times3}(F_{\mathcal{M}})))) \quad (2)$$

whereas $DwConv_{3\times3}$ denotes the DwConv with kernel size 3×3, $PwConv_{1\times1}$ denotes pointwise convolution (PwConv), and $F_{\mathcal{M}'}$ denote the non-linearly projected representations from the stacked convolutions. Depth-wise departure is a method of factorization of Convolution by dividing the conventional convolution into DwConv and PwConv, considerably decreasing the convolution calculation. The number of computations in basic convolution can be formulated as follow:

$$X_1 = D \times D \times M \times N \times D_W \times D_H \quad (3)$$

whereas the number of computations in DwConv can be formulated as follow
:

$$X_2 = D \times D \times M \times D_W \times D_H + M \times N \times D_W \times D_H, \quad (4)$$

This in turn implies a significant reduction with:

$$X_3 = \frac{D \times D \times M \times D_W \times D_H + M \times N \times D_W \times D_H}{D \times D \times M \times N \times D_W \times D_H} \quad (5)$$

The batch normalization (BN) layer is applied after the stack of the convolutions prior to the activation function, aiming to normalize the distribution of eigenvalues. This normalization helps accelerates the training convergence

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

and also relieves the "gradient dispersion" trouble of the DL model by making the deep network simpler and steadier. The computation of the BN layer is formulated as follows:

$$\begin{aligned} \hat{y}_i(j) &= \frac{y_i(j) - \mu}{\sigma} \\ \hat{z}_i(j) &= \gamma \cdot \hat{y}_i(j) + \beta \end{aligned} \quad (6)$$

By the end of this stack of blocks, Global Average Pooling (GAP) is applied to generate vectorized representation by taking the average of each feature map. Then, the output of the GAP is passed to malware detection decision at the output SoftMax layer, with the number of units equal to the number of classes. Cross-entropy loss is applied as a loss function to update the parameters of the model:

$$L(p,q) = -\sum_x p(x) \cdot log\, q(x) \quad (7)$$

The $p(x)$ denotes the actual label of malware images in the training set, and $q(x)$ denotes the prediction made by the network. To rapidly train the network according to the above loss function, the Adam optimizer is applied as follows:

$$\begin{aligned} m_{t+1} &= \beta_1 m_t + (1-\beta_1)\frac{\partial L}{\partial w}, \\ v_{t+1} &= \beta_2 v_t + (1-\beta_2)(\frac{\partial L}{\partial w})^2, \\ \hat{m}_{t+1} &= \frac{m_{t+1}}{1-\beta_1^t}, \hat{v}_{t+1} = \frac{v_{t+1}}{1-\beta_2^t}, \\ w_{t+1} &= w_t - \eta \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \epsilon}, \end{aligned} \quad (8)$$

whereas $\frac{\partial L}{\partial w}$ denotes the gradient of weight parameters *w*. the symbols m_t and v_t denote the first and second-moment estimates, respectively. the symbol t denotes the training step. The symbol η denotes the learning rate. $\beta 1$ and $\beta 2$ represent the exponential decay rates for the moment estimates.

B. FLC-NET

The proposed FLC-NET is simply an extension to LC-NET to be collaboratively trained using malware data distributed over edge/ fog devices. In this setting, the participating IoT devices are referred to as clients. The training of FLC-NET passes through multiple stages as described below:

Step 1: The cloud server is set to select the clients to participate in training according to some predefined condition (e.g., resource availability, distance, etc.). Then, the hyper-parameters of the LC-NET are initialized. The format of malware images is determined as either gray or RGB. The number of communications rounds is also specified. The number of adversarial samples to be generated during the training. By end of the stage, all the above settings are broadcasted to the selected over a secure communication channel.

Step 2: On the fog side, the malware binaries are converted to image representation as specified by the cloud server. A simple but useful technique is applied at this stage to visualize raw bytes of malware as an image without entailing any domain knowledge or feature engineering (see Fig. 1). In particular, malware files are provided a given in form of $8-bit$ unsigned integers, which are followingly rearranged in a two-dimensional matrix. After that, a color map is applied to the generated matrix to visualize malware as an image. The height dimensions of malware images are permitted to change according to the malware size, while the width dimension is static. Typically, malware belonging to the same class or family seemed analogous in outline and design in the image. By the end of this stage, the training image data are allocated to the clients in the proximity of fog nodes.

Step 3: by taking the advantage of more resources always available in the fog tier than in the edge tier, the fog nodes take the responsibility of adversarial examples for the nearby clients according to the designated attack schemes in

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

the initialization stage. This in turn means offloading the task of the generation from the resource-constrained devices. the sample generation can be accomplished only at the beginning of training or periodically repeated during training. The latter case implies multiple communication between the clients and the fog servers.

Step 4: By the completion of local adversarial training, the clients uploaded gradient information to the coordination node through a secure network channel. This is referred to as the communication round between the clients and the server. The cloud receives the local updates from all clients to aggregate them using the Fed-Avg algorithm. Once the aggregations get completed the parameters of the global model are now updated, then, the global updates are broadcasted to all participating clients to update their local model accordingly. The local LC-NET starts local training using updated parameters.

III. Experiments and Analysis

This section dive into the details of the implementation settings of the experimentations performed in this work. Following, the experimental results and the related discussions and analysis are systematically discussed.

A. Experimental Design

To implement the experiments of this study, Python 3.8.0 is used as a programming language, while Tensorflow is used for developing DL models. In physical terms, all experimentations are running on 32-bit Windows 10 installed on a PC. This device is armed with Intel (R) Xeon (R) CPU E5-2670 0@ 2.60GHz CPU, accelerated with NVIDIA GTX GPU, and has 32GB RAM.

B. Evaluation measures

In this work, four performance metrics are used to evaluate the performance of the proposed FLC-NET. Given the false negative (FN), true negative (TN), false positive (FP), and true positive (TP), the metrics can be defined as follow:

$$Precision = \frac{TP}{TP + FP} \times 100 \quad (9)$$

$$Recall = \frac{TP}{TP + FN} \times 100 \quad (10)$$

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (11)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (12)$$

C. Dataset Description

A public malware dataset is used for training and evaluating the proposed FLC-NET in our experiments. First, Malimg is a public real-world malware dataset comprising a total of 9435 samples belonging to 25 classes of malware. The distribution of samples across different classes and families is reported in Table 4. The data is augmented by applying a set of image transformations such as flipping, rotations, translation, and cropping. 20% of the data was hold-out for testing while the remaining are used for training. During the training, 20% of the data is used for validation. The hold-out test enables analyzing the generalization ability of the model.

Table 1: The class distribution of Malimg data.

#	Family	Class	No. instances
1	Backdoor	Agent.FYI	116
2		Rbot!gen	158
3	Dialer	Instantaccess	431
4		Adialer.C	125
5		Dialplatform.B	177
6	PWS	Lolyda.AA 1	213
7		Lolyda.AA 2	184

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

8		Lolyda.AA 3	123
9		Lolyda.AT	159
10	Rogue	Fakerean	381
11	Trojan	C2Lop.P	146
12		C2Lop.gen!G	200
13		Alueron.gen!J	198
14		Malex.gen!J	136
15		Skintrim.N	80
16	Trojan Downloader	Swizzor.gen!I	132
17		Swizzor.gen!E	128
18		Wintrim.BX	97
19		Dontovo.A	162
20		Obfuscator.AD	142
21	Worm	Allaple.L	1591
22		Allaple.A	2949
23		VB.AT	408
24		Yuner.A	800
25	Worm:AutoIT	Autorun.K	106

D. Comparative Analysis

Table 2: comparative analysis under different performance metrics

Methods	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score**
DCNN [10]	**96.24**	**95.85**	96.31	96.08
IMCEC(ResNet) [27]	97.66	97.57	97.34	97.45
IMCFN [15]	97.85	97.44	97.72	97.58
LCNN [11]	**96.24**	**95.85**	96.31	96.08
LC-NET	98.82%	98.72%	98.81	98.32

To analyze the FLC-NET performance, we start with comparative experiments. In this regard, the model comparisons are firstly performed in a centralized setting by comparing the LC-NET with cutting-edge centralized methods for malware detection. The results of these comparisons are reported in Table 2. As shown, the proposed LC-NET is achieving a significant performance improvement over complex competing methods, which validates the competitive advantage of LC-NET.

Table 3: comparative analysis in terms of the number of parameters

Methods	**#Total Parameters**	**#Trainable Parameters**	**#Non-Trainable Parameters**
DCNN [10]	15,269,721	552,985	14,716,736
IMCEC(ResNet) [27]	23,638,937	23,585,817	53,120
IMCFN [15]	134,362,969	126,727,705	7635,264
LCNN [11]	74,429	74,429	0
LC-NET	46,585	46,585	0

Going further, the comparative analysis is extended to compare the computational efficiency of the LC-NET by comparing its number of parameters with the number of parameters of the competing methods, as shown in Table 3. Notably, the number of parameters of LC-NET is less than the most competing methods. this finding reflects the

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

computing and time efficiency of the LC-NET making it a proper choice for detecting malware in resource-constrained IoT devices.

E. Statistical analysis

Beyond the competitive advantages of the LC-NET, additional experiments are performed to assess the statistical significance of the results. To do so, the Friedman statistical test is used to contrast the obtained predictions against those achieved by competing studies. The results of these experiments are reported, in terms of p-values, in Table 4. In these experiments, a statistical threshold $\sigma = 0.05$ is set for the Friedman test. Notably, almost all p-values not exceeding the threshold σ suggest the denial of the null hypothesis. In other words, the outputs from the LC-NET statistically diverge from the ones obtained from the competing studies on both accuracy and f1-score. The implementation of statistical analysis is performed using a Scientific Python research library named SciPy [28].

Table 4: The results of the Friedman statistical test in terms of p-value

Methods	Accuracy (%)	F1-score
LC-NET vs DCNN [10]	2.09E-11	8.54E-06
LC-NET vs IMCEC [27]	5.64E-05	9.20E-03
LC-NET vs IMCFN[15]	9.98E-09	6.62E-08
LC-NET vs LCNN [11]	8.37E-07	7.61E-09

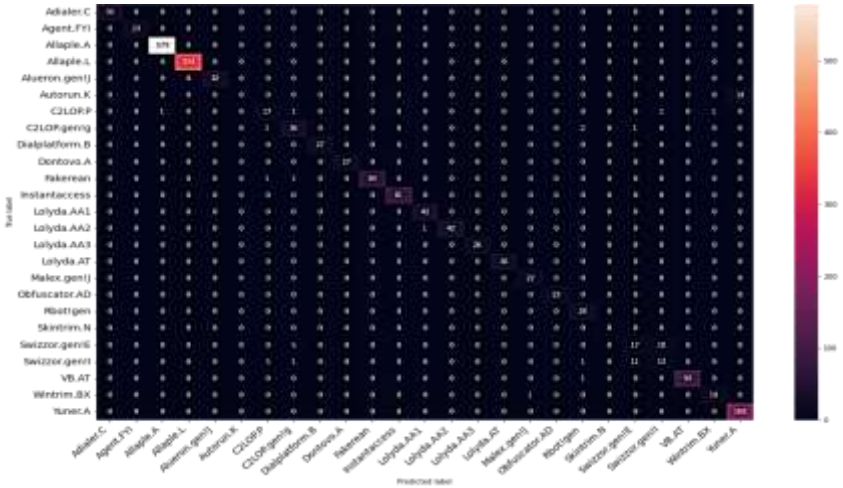

Figure 2: The Confusion matrix of the proposed FED-MAL

By shifting toward the federated scenario, it could be noted that FLC-NET shows robust classification performance (Accuracy: 96.1%, precision: 95.08%, recall: 96.1%, f1-score: 95.5). compared to the prediction results obtained by centralized LC-NET, there is a slight drop in performance but with no statistical difference (p-value:0.80213). To further understand the performance of FLC-NET, the corresponding confusion matrix is displayed in Fig.2, where the class-level performance can be observed. Furthermore, to analyze the discriminative power of the model, the Receiver operating characteristic (ROC) curve analysis is performed, and the corresponding results are shown in Fig. 3. As noticed, the average AUC of FLC-NET is 100% demonstrates the efficiency of FLC-NET in discriminating different classes of IoT malware.

F. Number of Fog nodes

Federated training with multiple clients is a challenging task and hence is usually used to assess the scalability of the model. In this regard, the proposed FLC-NET is evaluated multiple times by increasing the number of participating fog nodes each time. The malware detection performance (Accuracy, F1-score) of these experiments is shown in Fig. 4. It could be noted that the FLC-NET encountered a marginal decline in the classification performance when increasing the number of nodes from 6 to 20. This decline can be attributed to the decrease in the amount of training

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2023

data at local nodes. This observation suggests a great scalability performance for the proposed FLC-NET making them an appropriate choice for IoT environments.

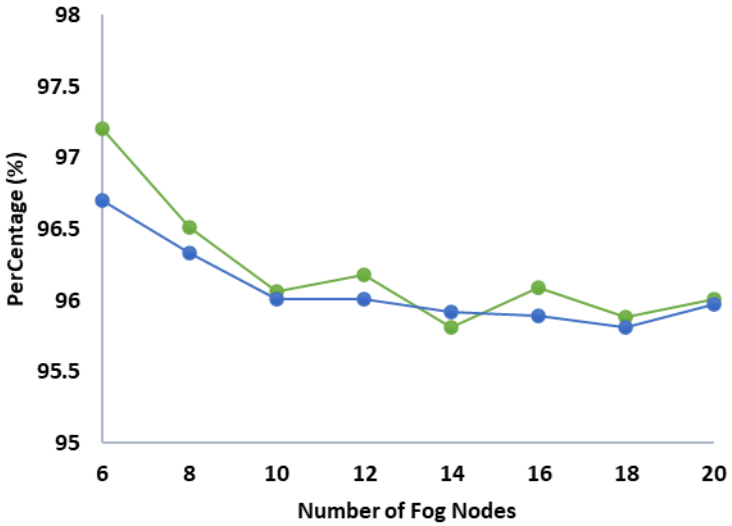

Figure 4: Scalability analysis for the FED-MAL

G. Number of communication rounds

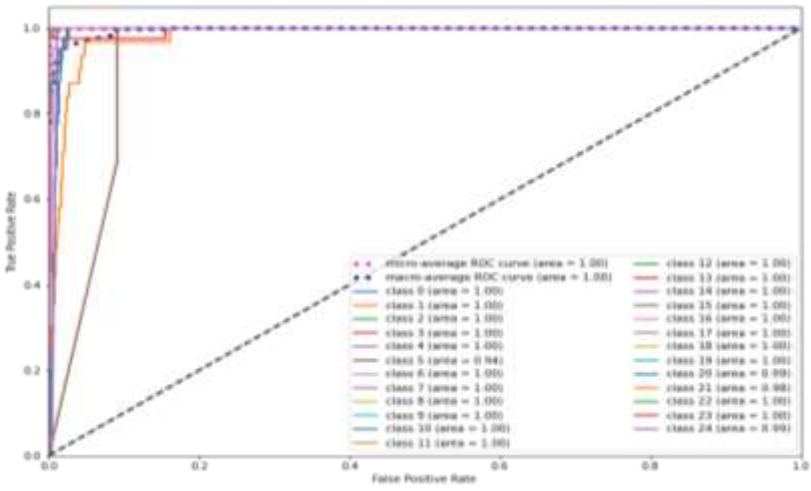

Figure 3: The Receiving operating curve for the proposed FED-MAL

DOI: https://doi.org/1
Received: September 18, 2022 Accepted: November 26, 2022

The number of communications between the clients and the cloud server is an essential aspect to be considered when it comes to analyzing an FL solution. In this way, FLC-NET is evaluated many times, whereby the number of communication rounds is increased by 3 each time. The detection performance per time is reported in Fig. 5. Notably, the FLC-NET rapidly converges after 25 rounds of communications, which indicates a small number of interactions between the fog nodes and the server. This in turn implies that the FLC-NET is communication efficient in terms of network latency and communication overhead.

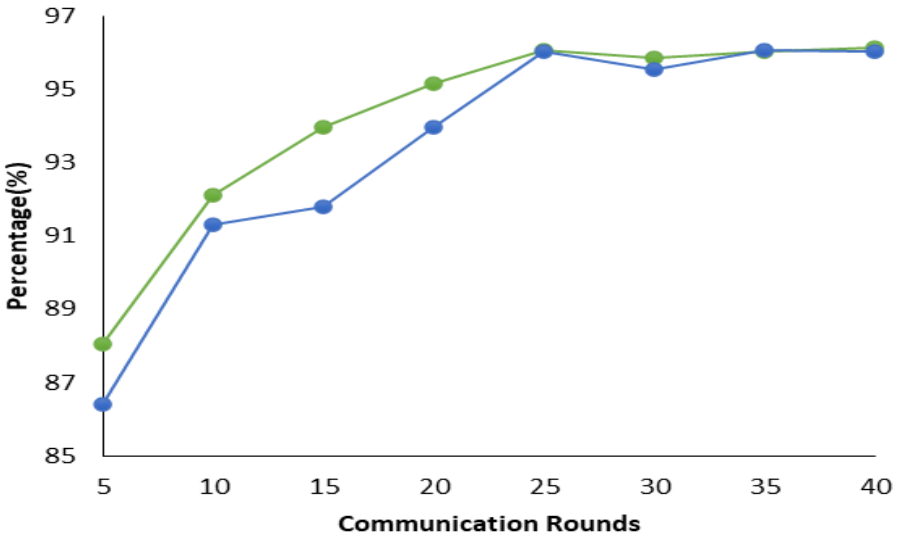

Figure 5: Convergence Analysis of the proposed FED-MAL in terms of a number of communication rounds.

3. Conclusion

In this work, an innovative FL framework, called FLC-NET, is presented to detect malware from image representations of malware binaries. In FLC-NET, a lightweight network, named LC-NET, is trained locally to detect malware on fog nodes or resource-constrained devices. More, adversarial federated training is presented to empower the framework to defend against adversarial attacks during the training. Experimental findings demonstrated that FLC-NET is precise, computationally efficient, scalable, and secure against adversarial attacks. Thus, the FLC-NET can be deployed as a tool for detecting IoT malware in real-world IoT applications such as smart homes, smart transportation, smart cities, etc. Given the findings in the previous section, the FLC-NET can be extended in the future in different ways. First, blockchain technology will be integrated to extend FLC-NET to alleviate the issues related to centralized aggregation. Second, the LC-NET will be extended to generate model-agnostic explanations for its decisions.

References

[1] N. Magaia, R. Fonseca, K. Muhammad, A. H. F. N. Segundo, A. V. Lira Neto, and V. H. C. De Albuquerque, "Industrial Internet-of-Things Security Enhanced with Deep Learning Approaches for Smart Cities," *IEEE Internet Things J.*, 2021, doi: 10.1109/JIOT.2020.3042174.

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

[2] X. X. X. Wang *et al.*, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," *IEEE Commun. Surv. Tutorials*, 2020.
[3] M. L. Loureiro and M. Alló, "Sensing climate change and energy issues: Sentiment and emotion analysis with social media in the U.K. and Spain," *Energy Policy*, 2020, doi: 10.1016/j.enpol.2020.111490.
[4] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Syst.*, 2020, doi: 10.1016/j.knosys.2019.105124.
[5] M. A. Ferrag, L. Shu, H. Djallel, and K. K. R. Choo, "Deep learning-based intrusion detection for distributed denial of service attack in agriculture 4.0," *Electron.*, 2021, doi: 10.3390/electronics10111257.
[6] Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: a deep learning–based intrusion detection framework for securing IoT," *Trans. Emerg. Telecommun. Technol.*, 2022, doi: 10.1002/ett.3803.
[7] A. Fu, X. Zhang, N. Xiong, Y. Gao, H. Wang, and J. Zhang, "VFL: A Verifiable Federated Learning with Privacy-Preserving for Big Data in Industrial IoT," *IEEE Trans. Ind. Informatics*, vol. 18, no. 5, pp. 3316–3326, 2022, doi: 10.1109/TII.2020.3036166.
[8] S. Henna and A. Davy, "Distributed and Collaborative High Speed Inference Deep Learning for Mobile Edge with Topological Dependencies," *IEEE Trans. Cloud Comput.*, 2020, doi: 10.1109/TCC.2020.2978846.
[9] Y. Huang *et al.*, "A Lightweight Collaborative Deep Neural Network for the Mobile Web in Edge Cloud," *IEEE Trans. Mob. Comput.*, 2020, doi: 10.1109/TMC.2020.3043051.
[10] B. Yuan, J. Wang, D. Liu, W. Guo, P. Wu, and X. Bao, "Byte-level malware classification based on markov images and deep learning," *Comput. Secur.*, 2020, doi: 10.1016/j.cose.2020.101740.
[11] B. Yuan, J. Wang, P. Wu, and X. Qing, "IoT Malware Classification Based on Lightweight Convolutional Neural Networks," *IEEE Internet Things J.*, 2022, doi: 10.1109/JIOT.2021.3100063.
[12] D. Li and Q. Li, "Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection," *IEEE Trans. Inf. Forensics Secur.*, 2020, doi: 10.1109/TIFS.2020.3003571.
[13] R. Feng, S. Chen, X. Xie, G. Meng, S. W. Lin, and Y. Liu, "A Performance-Sensitive Malware Detection System Using Deep Learning on Mobile Devices," *IEEE Trans. Inf. Forensics Secur.*, 2021, doi: 10.1109/TIFS.2020.3025436.
[14] Sudhakar and S. Kumar, "MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things," *Futur. Gener. Comput. Syst.*, 2021, doi: 10.1016/j.future.2021.06.029.
[15] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Comput. Networks*, 2020, doi: 10.1016/j.comnet.2020.107138.
[16] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, "A Survey of Android Malware Detection with Deep Neural Models," *ACM Computing Surveys*. 2021, doi: 10.1145/3417978.
[17] D. Li, Q. Li, Y. (Fanny) Ye, and S. Xu, "Arms Race in Adversarial Malware Detection: A Survey," *ACM Comput. Surv.*, 2023, doi: 10.1145/3484491.
[18] X. Pei, X. Deng, S. Tian, L. Zhang, and K. Xue, "A Knowledge Transfer-based Semi-Supervised Federated Learning for IoT Malware Detection," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–1, 2022, doi: 10.1109/TDSC.2022.3173664.
[19] V. Rey, P. M. Sánchez Sánchez, A. Huertas Celdrán, and G. Bovet, "Federated learning for malware detection in IoT devices," *Comput. Networks*, 2022, doi: 10.1016/j.comnet.2021.108693.
[20] A. Makkar, T. W. Kim, A. K. Singh, J. Kang, and J. H. Park, "SecureIIoT Environment: Federated Learning empowered approach for Securing IIoT from Data Breach," *IEEE Trans. Ind. Informatics*, 2022, doi: 10.1109/TII.2022.3149902.
[21] E. M. Campos *et al.*, "Evaluating Federated Learning for intrusion detection in Internet of Things: Review and challenges," *Comput. Networks*, 2022, doi: 10.1016/j.comnet.2021.108661.
[22] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surv. Tutorials*, 2018, doi: 10.1109/COMST.2018.2814571.
[23] Z. Yu, J. Hu, G. Min, Z. Wang, W. Miao, and S. Li, "Privacy-Preserving Federated Deep Learning for Cooperative Hierarchical Caching in Fog Computing," *IEEE Internet Things J.*, 2021, doi: 10.1109/JIOT.2021.3081480.
[24] R. Saha, S. Misra, and P. K. Deb, "FogFL: Fog-Assisted Federated Learning for Resource-Constrained IoT Devices," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8456–8463, 2021, doi: 10.1109/JIOT.2020.3046509.

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022

[25] Y. Liu, Y. Dong, H. Wang, H. Jiang, and Q. Xu, "Distributed Fog Computing and Federated Learning enabled Secure Aggregation for IoT Devices," *IEEE Internet Things J.*, pp. 1–1, 2022, doi: 10.1109/JIOT.2022.3176305.

[26] K. W. Hung, Z. Zhang, and J. Jiang, "Real-time image super-resolution using recursive depthwise separable convolution network," *IEEE Access*, 2019, doi: 10.1109/ACCESS.2019.2929223.

[27] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-Based malware classification using ensemble of CNN architectures (IMCEC)," *Comput. Secur.*, 2020, doi: 10.1016/j.cose.2020.101748.

[28] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nat. Methods*, 2020, doi: 10.1038/s41592-019-0686-2.

[29] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," 2017, doi: 10.1007/978-3-319-66399-9_4.

[30] A. Al-Dujaili, A. Huang, E. Hemberg, and U. M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," 2018, doi: 10.1109/SPW.2018.00020.

[31] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," 2016, doi: 10.1109/EuroSP.2016.36.

[32] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[33] D. Gibert, M. Fredrikson, C. Mateu, J. Planes, and Q. Le, "Enhancing the insertion of NOP instructions to obfuscate malware via deep reinforcement learning," *Comput. Secur.*, 2022, doi: 10.1016/j.cose.2021.102543.

DOI: https://doi.org/10.54216/IJWAC.060204
Received: September 18, 2022 Accepted: November 26, 2022