International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 14, No. 2, July 2025, pp. 546~557
ISSN: 2089-4864, DOI: 10.11591/ijres.v14.i2.pp546-557

Classifying IoT firmware security threats using image analysis and deep learning

Abdelkabir Rouagubi, Chaymae El Youssofi, Khalid Chougdali
Engineering Sciences Laboratory, Ibn Tofail University, Kenitra, Morocco

Article Info

Article history:

Received Jan 15, 2025
Revised Jun 2, 2025
Accepted Jun 10, 2025

Keywords:

Deep learning
Firmware-based attacks
Image analysis
Internet of things security
Malware classification

ABSTRACT

As the internet of things (IoT) grows, its embedded devices face increasing vulnerability to firmware-based attacks. The lack of robust security mechanisms in IoT devices makes them susceptible to malicious firmware updates, potentially compromising entire networks. This study addresses the classification of IoT firmware security threats using deep learning and image-based analysis techniques. A publicly available dataset of 32×32 grayscale images, derived from IoT firmware samples and categorized as benignware, hackware, and malware, was utilized. The grayscale images were converted into three-channel RGB format to ensure compatibility with convolutional neural networks (CNNs). We tested multiple pre-trained CNN architectures, including SqueezeNet, ShuffleNet, MobileNet, Xception, and ResNet50, employing transfer learning to adapt the models for this classification task. Both ResNet50 and ShuffleNet achieved exceptional performance, with 100% accuracy, precision, recall, and F1-score. These results validate the effectiveness of our methodology in leveraging transfer learning for IoT firmware classification while maintaining computational efficiency, making it suitable for deployment in resource-constrained IoT environments.

Corresponding Author:

Abdelkabir Rouagubi
Engineering Sciences Laboratory, Ibn Tofail University
Kenitra, Morocco
Email: abdelkabi.rouagubi@uit.ac.ma

1. INTRODUCTION

The internet of things (IoT) is rapidly transforming the way people and industries interact with technology, envisioning a future where interconnected devices autonomously collect and analyze data to perform diverse tasks. IoT devices have been adopted across various sectors, from wearable technologies and smart homes to industrial equipment and smart cities. This technological boom is reflected in the remarkable growth of IoT devices worldwide. According to Statista [1], the number of connected devices is projected to nearly double from 15.9 billion in 2023 to over 32.1 billion by 2030, with the consumer segment, including smartphones and media devices, accounting for around 60% of all IoT devices. By 2033, the highest concentration of IoT devices is expected in China, with an estimated 8 billion connected devices.

However, the rapid expansion of IoT also brings a significant increase in security challenges. As the use of IoT systems grows, so does the potential for vulnerabilities within these interconnected devices, particularly at the firmware level. Recent studies have highlighted the alarming state of IoT security. As reported by Viakoo in 2024 [2], 83% of IT leaders recognize the need to address IoT threats at an application level; however, only half feel prepared to effectively tackle IoT vulnerabilities. Additionally, 22% of organizations reported experiencing serious IoT security incidents that disrupted their operations. Despite the rising number of threats, only 35% of IT leaders feel confident in their organization's ability to secure IoT

environments, particularly due to the difficulty in managing firmware-based vulnerabilities. Furthermore, 71% of IT leaders regret not adopting stronger IoT security measures earlier, especially considering that many IoT devices, particularly those with microcontroller units (MCUs), often run on outdated or insecure firmware.

Firmware vulnerabilities represent a critical weak point in IoT security, exposing devices to risks such as N-day vulnerabilities-flaws that are publicly known yes remain unpatched. These vulnerabilities can be particularly damaging, as highlighted in previous research [3], [4] because they affect the fundamental software that controls device behavior. As the demand for connected devices continues to rise, it is becoming increasingly clear that addressing IoT security issues at the firmware level is essential.

To address this growing challenge, we propose a novel approach for classifying IoT firmware security threats using image analysis and deep learning techniques. Image classification offers a unique and efficient way to detect and mitigate firmware vulnerabilities by converting binary representations of firmware into images. By applying convolutional neural networks (CNNs) models to these images, we can effectively identify patterns and anomalies associated with benignware, hackware, and malware. This method enables automated, scalable analysis of IoT firmware security threats, offering a promising solution for organizations to strengthen their IoT security posture in a world of rapidly increasing connected devices.

Our paper is organized as follows section 1 introduces the topic, including a background on IoT firmware and files, as well as a review of previous work in IoT malware detection. Section 2 describes the research methodology, focusing on the conversion of ELF files to grayscale images and the application of transfer learning using CNN models. Section 3 presents the results and discussion, including a description of the dataset, the experimental setup, the evaluation metrics, and the performance of different CNN models. This section also includes a comparison of the proposed approach with previous studies and a discussion of the findings. Section 4 concludes the paper by summarizing the contributions and discussing potential directions for future research to enhance IoT firmware security.

The executable and linkable format (ELF) [5] is a standard file format used for executables, object code, shared libraries, and core dumps in Unix like operating systems. It provides a flexible and extensible structure for program binaries, making it an ideal choice for firmware development. Understanding its architecture is essential for analyzing IoT firmware, as firmware often adopts ELF for its binaries.

- Header: the ELF header serves as the entry point for the file, describing its structure and providing metadata. Key fields include the file type, machine architecture, and entry point address.
- Program header table: contains segments that describe how to create the process image in memory.
- Section header table: holds sections such as .text (code), .data (initialized data), and .rodata (read only data).
- Sections: logical divisions of the file, often representing executable code, metadata, or other resources.

The architecture of the ELF file is shown in Figure 1. The ELF header is central to this research because it contains critical metadata fields that can be transformed into visual representations for image-based analysis. This transformation leverages the structural patterns in the header to classify firmware into three categories: benignware, malware, and hackware. The ability to distinguish these classes enhances the detection and mitigation of security threats in IoT devices [6]. Among the key components of the ELF header are:

- e_ident: identifies the ELF file and its version,
- e_type: specifies the file type (e.g., executable, shared object),
- e_machine: specifies the target architecture (e.g., ARM, x86),
- e_version, e_entry, e_phoff, e_shoff: additional fields providing metadata about offsets and the entry point.

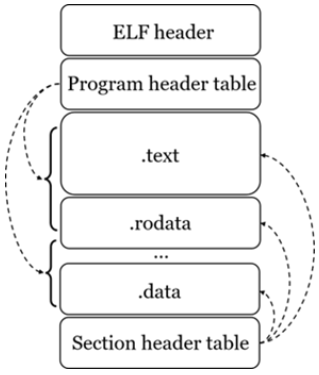


Figure 1. ELF architecture

CNNs [7] have proven to be highly effective for image classification tasks, and in this research, they are employed to classify IoT firmware into three categories: benignware, malware, and hackware. The following CNN models are used due to their efficiency, adaptability, and effectiveness for image-based analysis, particularly in resource-constrained environments like IoT devices.

MobileNet [8]–[10] is a family of lightweight CNN architectures designed by Google for mobile and embedded vision applications. The main innovation of MobileNet is the use of depthwise separable convolutions, which reduce computational cost and the number of parameters. This efficiency makes it suitable for real-time applications on devices with limited computing power. MobileNetV1 introduced depthwise separable convolutions, while MobileNetV2 improved the architecture with inverted residual blocks and linear bottleneck layers. MobileNetV3 further refines the design using neural architecture search (NAS) to optimize accuracy and efficiency.

ResNet [11], [12], or residual networks, are deep CNNs that address the vanishing gradient problem using residual learning. By employing skip connections, ResNet allows the network to learn residual functions based on input activations, enabling the training of deeper networks. Popular versions of ResNet include ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. In this study, ResNet50 was used due to its balance between depth and performance. ResNet is highly effective for tasks such as image recognition, object detection, and transfer learning.

ShuffleNet [13] is a lightweight CNN architecture designed to achieve high accuracy with reduced computation and memory usage, making it ideal for mobile and embedded devices. Its key innovations include pointwise group convolutions and channel shuffling, which help reduce computational cost while maintaining strong accuracy. ShuffleNetV2 builds on these principles with optimizations like balanced feature dimensions and efficient element-wise operations.

SqueezeNet [14] is a lightweight CNN architecture designed to achieve AlexNet-level accuracy with significantly fewer parameters. It uses “fire modules”, which consist of a squeeze layer with 1×1 filters followed by an expand layer with 1×1 and 3×3 filters. This design reduces the number of parameters while maintaining model accuracy. SqueezeNet is particularly well-suited for resource-constrained environments where storage and processing power are limited.

Xception (extreme inception) [11], [15], [16] is a CNN architecture that builds upon the inception model by introducing depthwise separable convolutions. This innovation reduces computational costs while maintaining high accuracy. Xception replaces traditional inception modules with depthwise separable convolutions, which split the convolution operation into depthwise and pointwise convolutions, improving efficiency and performance.

These models were adapted using transfer learning techniques, leveraging pre-trained weights to fine-tune the models on the transformed grayscale images derived from ELF headers. This approach allows the models to effectively classify IoT firmware into benignware, malware, and hackware, providing high accuracy while reducing computational demands for real-time analysis.

The detection and classification of IoT malware have garnered significant attention due to the rapid proliferation of IoT devices and the increasing complexity of security threats. Over the years, researchers have explored various approaches ranging from traditional machine learning techniques to advanced deep learning models. These studies have utilized diverse datasets, including grayscale images of malware binaries and firmware samples, to develop classification frameworks tailored to IoT-specific challenges. Recent works have highlighted the potential of CNNs and other deep learning architectures in achieving high accuracy by leveraging the unique structural and textural patterns present in malware representations. In this section, we summarize notable contributions in this field, focusing on their methodologies, datasets, and outcomes.

Su *et al.* [17] introduced a lightweight approach for detecting distributed denial-of-service (DDoS) malware in IoT environments using image-based analysis. Their method involved converting binary files into grayscale images, allowing a CNN to classify these images. The study demonstrated the effectiveness of this approach, achieving an accuracy of 94.0% in distinguishing between benign software and DDoS malware, and 81.8% accuracy when differentiating between benign software and two prominent malware families.

Azmoodeh *et al.* [18] proposed a robust IoT malware detection framework leveraging deep eigenspace learning. Their approach focused on analyzing operational code (OpCode) sequences extracted from IoT applications to classify them as benign or malicious. By employing graph-based representations of OpCode sequences and applying a deep learning classifier in the eigenspace, the model achieved an impressive accuracy of 99.68%, along with a precision of 98.59%, and a recall of 98.37%.

Karanja *et al.* [19] proposed an IoT malware classification method using Haralick texture features extracted from grayscale images of malware binaries. The approach utilized a dataset consisting of Mirai, Bashlite, and benign files, with random forest (RF) achieving the highest accuracy of 95%, followed by naïve Bayes at 89%, and K-nearest neighbor (KNN) at 80%.

Arul and Punidha [20] introduced an innovative algorithm for detecting firmware-based attacks on IoT devices. Their approach involved transforming firmware files into binary pixel image patterns and employing deep pattern mining techniques to classify malicious firmware. The method analyzed inter and intra-cluster patterns, leveraging continuous probability distributions such as skewness and kurtosis for classification. Tested on 960 compact executable files, the approach achieved a 96.12% true positive rate with a 0.09% false positive rate.

Noever and Noever [21] proposed a deep learning approach to classify malicious IoT firmware using CNNs. They converted binary headers of 40,000 firmware samples into 32×32 grayscale thumbnail images. Using transfer learning with the MobileNetV2 architecture, their method achieved accuracies of 100% for malware, hackware, and unknown classes on larger images (224×224) and 94%-98% on smaller images (96×96).

Asam *et al.* [22] proposed a novel IoT malware detection framework based on a channel-boosted and squeezed CNN. The architecture incorporated advanced techniques such as edge exploration, multipath dilated convolutions, and channel squeezing to improve feature extraction from malware images. When tested on a benchmark IoT dataset, the model achieved an impressive 97.93% accuracy, an F1-score of 0.9394, and an AUC-ROC of 0.9938, showcasing its effectiveness.

Yapıcıoğlu *et al.* [23] proposed a method to detect malicious firmware installations on IoT devices using a multilayer CNN. The approach involved converting firmware binary data into images and training a CNN model for classification. This model was deployed on an embedded board for real-time detection, offering portability and cost-effectiveness. The study compared the CNN model with an autoencoder (AE) and achieved superior results, with the CNN model reaching a maximum accuracy of 91.19% under optimal conditions.

Abu-Mahfouz *et al.* [24] developed a CNN based approach to detect vulnerabilities in home router firmware. A manually curated dataset of 1,450 firmware samples was converted into images and processed using filters like histogram of oriented gradients (HOG), local binary patterns (LBP), and Gabor. Among these, the HOG filter delivered the best performance, achieving an average accuracy of 85.81%.

Panda *et al.* [25] proposed a transfer learning-based approach for detecting IoT malware using image-based analysis. The method involved converting malware binaries into grayscale images and employing pre-trained CNNs for feature extraction. Models like ResNet50 and MobileNetV2 were fine-tuned on the extracted features to classify malware. Tested on an IoT-specific dataset comprising grayscale malware images, the framework achieved a remarkable accuracy of 98.65% in distinguishing between benign and malicious software.

Mehrban and Ahadian [26] proposed a CNN-LSTM hybrid model for detecting IoT malware, addressing the growing complexity of security challenges in IoT environments. The study employed K-fold cross-validation for training and evaluation, achieving an accuracy of 95.5%, which surpassed existing methods. The CNN component was used for superior feature extraction, while the long short-term memory (LSTM) classifier provided enhanced accuracy in classification. The research highlighted the potential of combining convolutional and recurrent neural networks for malware detection, offering a robust framework for improving IoT security. The study also suggested future exploration of support vector machines (SVMs) and distributed detection strategies for further enhancing system resilience against IoT threats.

Dong and Kotenko [27] introduced an advanced image-based malware analysis framework tailored for IoT security in smart cities. The study employed variational autoencoders (VAEs) and conditional variational autoencoders (CVAEs) to process greyscale (32×32) and RGB (128×128) malware images derived from ELF binaries and the Big2015 dataset. Coupled with a hybrid CNN-GRU classifier, the framework achieved 100% accuracy on greyscale data using CVAE and CNN_VAE, and 99.17% accuracy on RGB data with CVAE.

Al-Musawi and Khammas [28] proposed an image-based IoT malware detection method utilizing several pre-trained deep CNN models, including AlexNet, VGG-16, VGG-19, InceptionV3, and MobileNet. Their study introduced a novel strategy that applied the chi-square goodness-of-fit method to an image dataset prior to model training, which enhanced performance and reduced dataset size. The approach was evaluated using the IOT_Malware dataset based on standard performance metrics. Among the models, VGG-19 achieved the highest accuracy of 99.09%, followed closely by VGG-16 with 98.55%. MobileNet, InceptionV3, and AlexNet also performed well, with accuracies of 98.10%, 98.01%, and 97.65%, respectively.

Ghahramani *et al.* [29] introduced "deep image," a novel method for online malware detection in IoT environments. Their approach involves monitoring malware behavior, extracting dynamic features, and converting them into sparse binary images for analysis. The study evaluated three distinct methodologies clustering, probabilistic, and deep learning for analyzing these image datasets. Among the tested approaches, the deep learning-based method demonstrated the highest performance, achieving top results in seven out of eight evaluated metrics, achieving an accuracy of 98.28%.

Sehgal *et al.* [30] proposed the intelligent malware threat detection (IMTD) framework for identifying and classifying IoT malware. Their approach utilized transfer learning methods combined with image visualization and data augmentation techniques to enhance detection performance. By converting malware samples into images, the framework leveraged pre-trained models to analyze visual patterns indicative of malicious activity. Tested on the MalImg dataset, the IMTD framework achieved an accuracy of 98.38%

Taşcı [31] proposed an optimized 1D CNN model tailored for classifying IoT-related attacks and malware. The model incorporates convolutional, self-attention, and GELU activation layers, enhanced by dropout and normalization techniques to mitigate overfitting. Evaluated across three benchmark datasets CIC IoT 2023, CIC-MalMem-2022, and CIC-IDS2017 the model achieved outstanding results, including accuracies of 98.36%, 99.90%, and 96.64%, respectively.

2. RESEARCH METHOD

2.1. Conversion of ELF files to grayscale images

To enable image-based classification of IoT firmware, our study systematically converts ELF binaries into grayscale images. The process is outlined as follows and illustrated in Figure 2:

- Input ELF files: the dataset consists of IoT firmware binaries in ELF format, categorized into three classes: (malicious, benign, and hacked*)*. ELF files are a widely used executable file format in IoT and embedded systems.
- Byte extraction: each ELF file undergoes a process where the first 1024 bytes of its binary content are extracted. These bytes are significant as they typically include metadata and structural information critical for classification.
- Hexadecimal to decimal conversion: the extracted bytes are initially represented in hexadecimal format (e.g., 00, FF, A3). Each byte is converted into its decimal equivalent (base-10), resulting in values ranging from 0 to 255. This step translates raw binary data into numerical values interpretable as grayscale intensities.
- Mapping to grayscale intensities: the decimal values are directly mapped to grayscale levels: i) 0 is mapped to black, representing the lowest intensity, ii) 255 is mapped to white, representing the highest intensity, and iii) intermediate values (e.g., 128) represent varying shades of gray.
- Reshaping to form images: the sequence of 1024 grayscale values is reshaped into a 32×32 matrix, forming a two-dimensional image. Each pixel in the image corresponds to one byte from the ELF file, visually encoding the binary data.

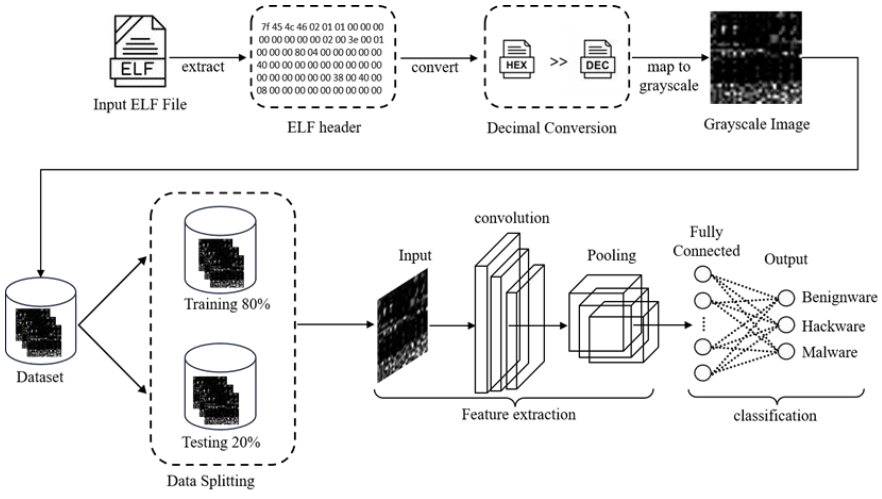


Figure 2. Firmware image conversion and classification architecture

2.2. Convolutional neural network models and transfer learning

To effectively classify IoT firmware using pre-trained CNN architectures, transfer learning was employed. This approach allows the reuse of knowledge from models trained on large datasets, such as ImageNet, by fine-tuning them for the specific task of IoT firmware classification.

The input to the CNNs was prepared by converting grayscale images into three-channel red, green, and blue (RGB) format to ensure compatibility with standard CNN architectures. Each grayscale image, initially of size 32×32×1, was duplicated across three channels, forming a final input of size 32×32×3. In addition to resizing the input, the output layers of the CNN architectures were modified to match the number of target classes in the dataset: malicious, benign, and hacked. Fully connected layers were replaced with custom layers to classify the three categories effectively.

The training process involved optimizing the models using the Adam optimizer, which adjusts the learning rate dynamically based on momentum and gradient information. The hyperparameters were set as follows: $\beta1=0.9$ (decay rate for momentum), $\beta2=0.999$ (decay rate for squared gradients), and $\epsilon=10^{-8}$ (stabilization term). The initial learning rate was fixed at 0.001 to allow for gradual convergence, and a mini-batch size of 128 was used, with the data shuffled after each epoch. The models were trained over 10 epochs to balance computational efficiency with effective learning.

This study applied transfer learning to two well-known CNN architectures: ResNet50 and ShuffleNet. ResNet50, a deep residual network with 50 layers, was chosen for its ability to extract complex features and mitigate the vanishing gradient problem through skip connections. ShuffleNet, a lightweight network designed for efficiency on resource-constrained devices, was selected for its potential applicability to IoT environments. Both models were fine-tuned on the prepared dataset, leveraging pre-trained weights while adapting to the specific task of IoT firmware classification.

3. RESULTS AND DISCUSSION

3.1. Dataset

To evaluate the proposed methodology, we produced a public dataset titled IoT firmware image classification, which has been published on Kaggle [32]. The dataset consists of grayscale images derived from IoT firmware binaries, classified into three categories: benignware, hackware, and malware, as presented in Figure 3.

Figure 3. Dataset categories

The dataset is divided into training 80% and testing 20% subsets, ensuring a proportional representation of each class in both splits. Table 1 summarizes the composition of the dataset. The dataset contains a total of 38,887 samples, with 31,109 samples allocated for training and 7,778 samples for testing. Notably, the majority of the dataset comprises benignware, while hackware and malware are underrepresented, reflecting the real-world imbalance in IoT firmware threats.

Table 1. Dataset statistics

Subset	Total	Benignware	Hackware	Malware
Training	31,109	30,458	82	569
Testing	7,778	7,615	21	142
Total	38,887	38,073	103	711

3.2. Experimental setup

The experiments were conducted on a workstation with the following hardware configuration: i) processor: Intel Core i7-11900; ii) RAM: 16 GB; and ii) GPU: NVIDIA GeForce RTX 3060. The software environment included: i) operating system: windows 11 64-bit; ii) software: MATLAB R2023a (update 7); and iii) deep learning framework: MATLAB deep learning toolbox

This setup ensured optimal performance for training and evaluating deep learning models, leveraging GPU acceleration for computationally intensive tasks.

3.3. Evaluation metrics

The performance of the proposed methodology was evaluated using the following standard metrics: Accuracy: the ratio of correctly classified samples to the total number of samples.

$$Accuray = \frac{True\ Positives\ (TP) + True\ Negatives\ (TN)}{True\ Positives\ (TP) + True\ Negatives\ (TN) + False\ Positives\ (FP) + False\ Negatives\ (FN)} \quad (1)$$

Precision: the ratio of true positive predictions to the total positive predictions, indicating the relevance of positive classifications.

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)} \quad (2)$$

Recall: the ratio of true positive predictions to the total actual positives, measuring the sensitivity of the model.

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)} \quad (3)$$

F1-score: the harmonic mean of precision and recall, providing a balanced evaluation metric.

$$F1 - Score = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)} \quad (4)$$

3.4. Model performance

To evaluate the effectiveness of the proposed methodology, we trained and tested several CNN architectures on the prepared dataset. The selected models included SqueezeNet, MobileNet, Xception, ShuffleNet, and ResNet50, chosen for their varying characteristics such as efficiency, scalability, and feature extraction capabilities. In Figure 4 we present the training and validation accuracy and loss curves of the ResNet50. The accuracy and loss curves indicate stable training and validation, with no signs of overfitting or underfitting.

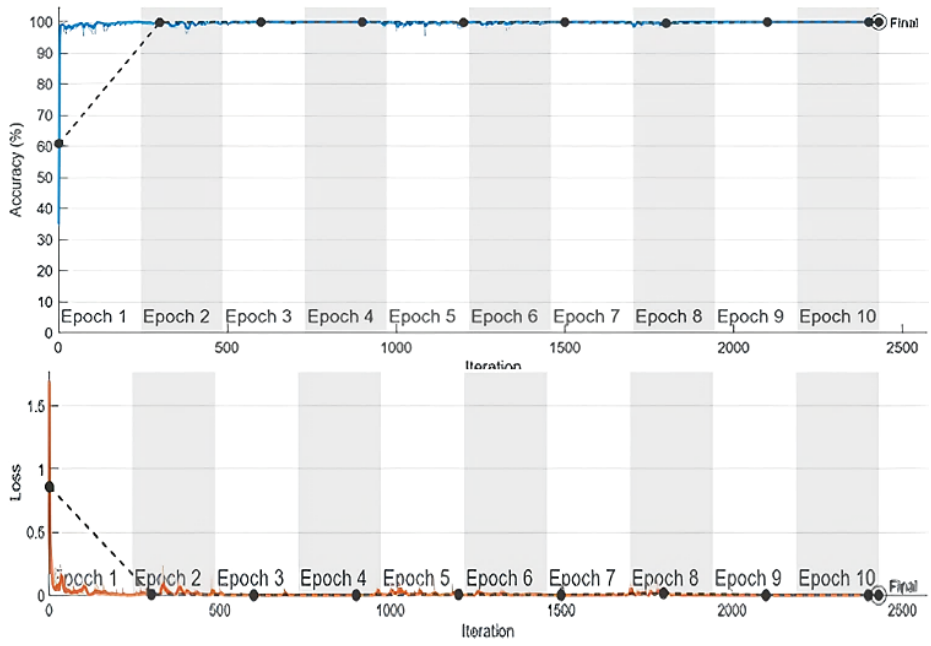


Figure 4. Training and validation accuracy and loss curves for ResNet50

To evaluate the performance of a classification model, we provide the confusion matrices and the ROC curves of the CNN models in Table 2 (in Appendix). The confusion matrices juxtapose the actual class labels with those predicted by the model, offering a comprehensive analysis of the model's efficacy. The graph illustrates the trade-off between the genuine positive rate and the false positive rate for various classification criteria.

The performance of each model in classifying the three types of firmware, namely benignware, hackware, and malware, is shown in full in Table 3. This table includes evaluation metrics such as accuracy, precision, recall, and F1-score for each class, which are used to compare the performance of the models. Since the dataset is imbalanced, the accuracy can be misleading. Which is shown in the SqueezeNet, 99.7% accuracy and 66% F1-score, because the benignware class represents 98% of the dataset, predicting every IoT firmware as benignware would yield 98% accuracy, but the model would fail to identify the other IoT firmware. Also, the precision and recall are more informative than accuracy and provide insight into the types of errors; in our case, we focus more on the hackware and malware that represent the security threats.

Table 3. Summarizes the performance metrics for each model

Model	Accuracy	Precision (%)			Recall (%)			F1-score (%)		
	(%)	benignware	hackware	malware	benignware	hackware	malware	benignware	hackware	malware
SqueezeNet	99.73	99.7	0	100	100	0	100	99.85	0	100
MobileNet	99.94	100	95	97.9	100	90.5	100	100	92.70	98.94
Xception	99.43	99.4	100	100	100	100	69	99.70	100	81.66
ShuffleNet	100	100	100	100	100	100	100	100	100	100
ResNet50	100	100	100	100	100	100	100	100	100	100

The confusion matrix of the Resnet50 and ShuffleNet shows a perfect classification meaning that the models can correctly predicted all classes. Therefore, our proposed method is capable of accurately identifying the legitimate firmwares intended for IoT devices, which are designed with no malicious intent. Additionally, unauthorized activities use the modified firmwares to exploit vulnerabilities, prevent data theft, or disrupt device functionality.

Among the tested models, ShuffleNet and ResNet50 achieved perfect scores across all metrics, demonstrating their capability to handle the dataset effectively. Designed as a lightweight architecture, ShuffleNet achieved 100% accuracy while maintaining computational efficiency, making it suitable for resource-constrained IoT environments. ResNet50, with its deeper architecture and residual connections, also achieved a 100% F1-score, showcasing its strength in extracting complex patterns from grayscale images.

Other models, such as MobileNet and Xception, also performed well, with F1-score of 97.2% and 94.5%, respectively. MobileNet demonstrated a strong balance between F1-score and computational cost, while Xception excelled in precision, achieving 99.8%. SqueezeNet, while achieving an accuracy of 99.7%, exhibited lower recall, precision, and F1-score, indicating challenges in generalizing across all classes.

3.5. Comparison with previous studies

To evaluate the significance of our proposed methodology, we compare it with other studies in the field of IoT firmware classification. Table 4 provides a detailed comparison of datasets, models, and performance metrics across the studies. The results of our study demonstrate the effectiveness of using pre-trained CNN architectures for IoT firmware classification. By transforming firmware binaries into grayscale images and subsequently converting them into three-channel RGB format, our methodology leverages the power of transfer learning while preserving the structural integrity of the data.

Table 4. Summary of IoT malware detection papers

Reference	Dataset	Model	Accuracy (%)
[19]	Mirai, Bashlite, benign files	RF, NB, KNN	95, 89, 80
[20]	960 compact executables Files	FA-PCM	96.12 TPR, 0.09 FPR
[21]	40,000 firmware samples	MobileNetV2	90-100
[22]	Benchmark IoT malware Dataset	squeezed CNN	97.93
[23]	Manually curated dataset (firmware binary data)	CNN	91.19
[24]	1,450 firmware samples	CNN	85.81
[27]	Greyscale and RGB datasets	AE, VAE, and CVAE	99.17
Our model	Publicly available dataset	ShuffleNet and ResNet50	100

As shown in Table 4, our approach achieves a notable advancement in IoT firmware classification. Both ResNet50 and ShuffleNet achieved 100% F1-score across all classes, outperforming the results of existing studies. Unlike prior works, which often rely on handcrafted features or custom models, our methodology employs a fully automated deep learning pipeline. The transformation of grayscale images into three-channel RGB format enables compatibility with advanced CNN architectures, ensuring optimal feature extraction.

In comparison with previous works, we achieved high performance in detecting the types of IoT firmware with low computational requirements using transfer learning. This approach outperforms the handcrafted features and custom algorithms, which, while effective, achieved lower accuracies due to limited generalizability:

- Studies such as [4], [5] relied on traditional methods like handcrafted features and custom algorithms, which, while effective, achieved lower accuracies due to limited generalizability.
- Noever and Noever [21] achieved similar accuracy using MobileNetV2 on larger input resolutions (224×224), but at the cost of higher computational requirements.
- Asam *et al.* [22] demonstrated innovative feature extraction techniques with their CNN channel boosted CNN, but failed to achieve 100% accuracy.

The proposed models demonstrate perfect classification performance, indicating that they have successfully predicted all classes without error. This exceptional accuracy highlights the robustness in detecting various IoT firmware types. Specifically, the models effectively identify:

- Legitimate firmware (benignware): firmware designed for IoT devices that functions as intended, with no malicious intent or harmful modifications.
- Modified firmware (hackware): firmware altered for unauthorized purposes, such as exploiting vulnerabilities or enabling unauthorized access to IoT devices.
- Malicious firmware (malware): firmware intentionally crafted to cause harm, steal sensitive data, or disrupt device functionality.

Utilizing advanced deep learning architectures such as ResNet50 and ShuffleNet, our methodology guarantees accurate identification of firmware types. This functionality is essential for preserving IoT security, since it facilitates the rapid detection and remediation of vulnerabilities from hacked or malevolent firmware. Our approach provides a balance between computational economy and classification accuracy, rendering it especially appropriate for resource-limited IoT settings. Nevertheless, our study possesses certain limitations. The collection, although varied, may not include all potential variants in IoT firmware files, especially for uncommon or emerging malware kinds.

4. CONCLUSION

This study proposed an innovative methodology for IoT firmware classification, utilizing pre-trained CNN architectures. By transforming firmware binaries into grayscale images and converting them into three-channel RGB format, the framework preserved structural patterns while ensuring compatibility with deep learning models. The experimental results demonstrated the approach's robustness and efficiency by achieving 100% F1-score across the categories: benignware, hackware, and malware. The ability to distinguish between benign and harmful firmware types not only safeguards IoT ecosystems from potential vulnerabilities but also minimizes the risks associated with data breaches and operational disruptions. Consequently, our method provides a reliable and efficient solution for enhancing the security posture of IoT devices. These results validate the methodology's effectiveness in resource-constrained IoT environments and its ability to handle imbalanced datasets.

Building upon this foundation, our future work will explore Swarm-Net, a framework for firmware attestation in IoT swarms using graph neural networks (GNNs). By integrating these techniques, Swarm-Net aims to address the scalability and real-time security challenges in large-scale IoT deployments, advancing IoT firmware security against emerging threats.

REFERENCES

[1] Statista, "IoT connections worldwide 2022-2033 | Statista," Statista, 2024, [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.
[2] Viakoo, "2024 IoT Security Crisis: By The Numbers," Inc. [Online]. Available: https://www.viakoo.com/2024-iot-security-crisis/.
[3] N. Nino, R. Lu, W. Zhou, K. H. Lee, Z. Zhao, and L. Guan, "Unveiling IoT Security in Reality: A Firmware- Centric Journey," in *33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA*, 2024, pp. 5609–5626.
[4] K. M. Harahsheh and C.-H. Chen, "A Survey of Using Machine Learning in IoT Security and the Challenges Faced by Researchers," *Informatica*, vol. 47, no. 6, pp. 1–54, May 2023, doi: 10.31449/inf.v47i6.4635.
[5] N. Partush, "Labeled-Elfs." 2021. [Online]. Available: https://github.com/nimrodpar/Labeled-Elfs/.

[6] A. Kosikowski, D. Cho, M. Ninan, A. Ralescu, and B. Wang, "EvilELF: Evasion Attacks on Deep-Learning Malware Detection over ELF Files," in *2023 International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2023, pp. 1702–1709, doi: 10.1109/ICMLA58977.2023.00258.
[7] P. K. Sarangi, B. Sharma, L. Rani, and M. Dutta, "Satellite Image Classification Using Convolutional Neural Network," in *Advances in Aerial Sensing and Imaging*, Wiley, 2024, pp. 333–353, doi: 10.1002/9781394175512.ch15.
[8] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv*, 2017, doi: 10.48550/arXiv.1704.04861.
[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.
[10] H. Andrew *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1314–1324.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9909, pp. 630–645, doi: 10.1007/978-3-319-46493-0_38.
[13] H. Chen, A. Zhang, C. Gong, W. Liang, and Z. Wang, "Fault Diagnosis Method for Photovoltaic Panels Based on Improved ShuffleNet V2 and Infrared Images," in *2022 IEEE 7th International Conference on Power and Renewable Energy, ICPRE 2022*, 2022, pp. 447–451, doi: 10.1109/ICPRE55555.2022.9960396.
[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv*, 2016, doi: 10.48550/arXiv.1602.07360.
[15] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.
[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826, doi: 10.1109/CVPR.2016.308.
[17] J. Su, V. D. Vasconcellos, S. Prasad, S. Daniele, Y. Feng, and K. Sakurai, "Lightweight Classification of IoT Malware Based on Image Recognition," in *Proceedings-International Computer Software and Applications Conference*, 2018, vol. 2, pp. 664–669, doi: 10.1109/COMPSAC.2018.10315.
[18] A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust Malware Detection for Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 88–95, Jan. 2019, doi: 10.1109/TSUSC.2018.2809665.
[19] E. M. Karanja, S. Masupe, and M. G. Jeffrey, "Analysis of internet of things malware using image texture features and machine learning techniques," *Internet of Things*, vol. 9, p. 100153, Mar. 2020, doi: 10.1016/j.iot.2019.100153.
[20] E. Arul and A. Punidha, "Firmware Attack Detection on IoT Devices Using Deep Binary Pattern Classification Mining (FA-PCM)," *ICT Analysis and Applications: Proceedings of ICT4SD 2020* 2021, pp. 379–387, doi: 10.1007/978-981-15-8354-4_38.
[21] D. Noever and S. E. M. Noever, "Deep Learning for Identifying Malicious Firmware," *CS & IT Conference Proceedings*, 2021, pp. 63–70, doi: 10.5121/csit.2021.111506.
[22] M. Asam *et al.*, "IoT malware detection architecture using a novel channel boosted and squeezed CNN," *Scientific Reports*, vol. 12, no. 1, p. 15498, Sep. 2022, doi: 10.1038/s41598-022-18936-9.
[23] C. Yapıcıoğlu, F. Y. Okay, and M. Demirci, "Malicious Firmware Detection on Embedded Systems Using Deep Learning," in *2023 Innovations in Intelligent Systems and Applications Conference (ASYU)*, Oct. 2023, pp. 1–6, doi: 10.1109/ASYU58738.2023.10296778.
[24] A. Abu-Mahfouz, S. Alrabaee, M. Khasawneh, M. Gergely, and K.-K. R. Choo, "A Deep Learning Approach to Discover Router Firmware Vulnerabilities," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 1, pp. 691–702, Jan. 2024, doi: 10.1109/TII.2023.3269774.
[25] P. Panda, Om K. C U, S. Marappan, S. Ma, M. S, and D. V. Nandi, "Transfer Learning for Image-Based Malware Detection for IoT," *Sensors*, vol. 23, no. 6, p. 3253, Mar. 2023, doi: 10.3390/s23063253.
[26] A. Mehrban and P. Ahadian, "Malware Detection in IoT Systems using Machine Learning Techniques," *International Journal of Wireless & Mobile Networks*, vol. 15, no. 6, pp. 13–23, 2023, doi: 10.5121/ijwmn.2023.15602.
[27] H. Dong and I. Kotenko, "Image-based malware analysis for enhanced IoT security in smart cities," *Internet of Things*, vol. 27, p. 101258, Oct. 2024, doi: 10.1016/j.iot.2024.101258.
[28] M. H. Al-Musawi and B. M. Khammas, "Image-Based IoT Malware Detection Method Using Deep CNNs," in *2024 1st International Conference on Emerging Technologies for Dependable Internet of Things (ICETI)*, Nov. 2024, pp. 1–8, doi: 10.1109/ICETI63946.2024.10777169.
[29] M. Ghahramani, R. Taheri, M. Shojafar, R. Javidan, and S. Wan, "Deep Image: A precious image based deep learning method for online malware detection in IoT environment," *Internet of Things*, vol. 27, p. 101300, Oct. 2024, doi: 10.1016/j.iot.2024.101300.
[30] R. Sehgal, R. Matam, and E. Kalaimannan, "IMTD: Intelligent Malware Threat Detection Using Transfer Learning Methods in IoT Environment," *Social Science Research Network*. NY: 4962382, 2024, doi: 10.2139/ssrn.4962382.
[31] B. Taşcı, "Deep-Learning-Based Approach for IoT Attack and Malware Detection," *Applied Sciences*, vol. 14, no. 18, p. 8505, Sep. 2024, doi: 10.3390/app14188505.
[32] TECPERSON, "IoT Firmware Image Classification," [Online]. Available: https://www.kaggle.com/datasets/datamunge/iot-firmware-image-classification.

APPENDIX

Table 2. Confusion matrix and the ROC curve for each model

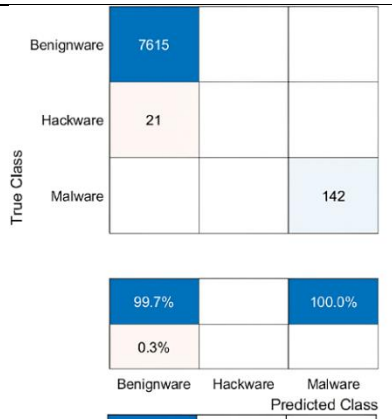
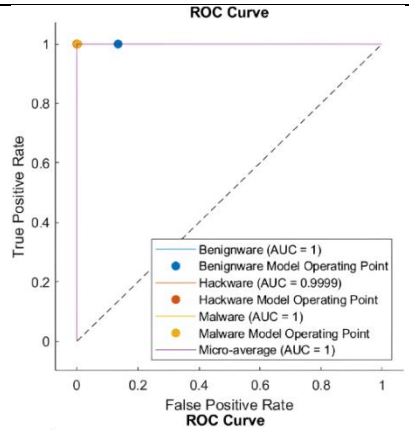
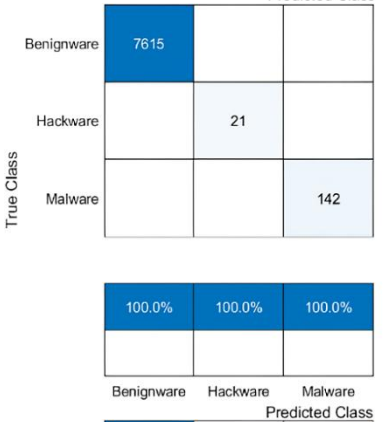
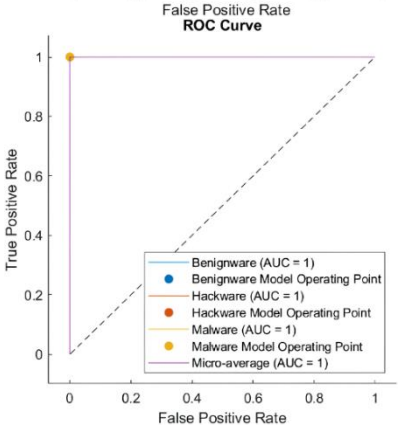
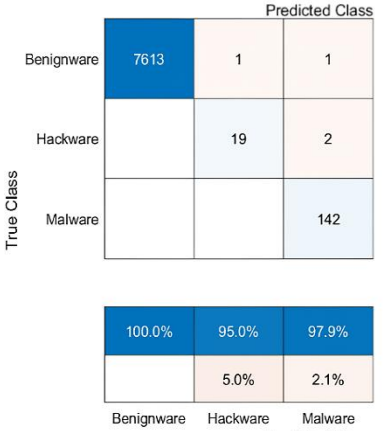
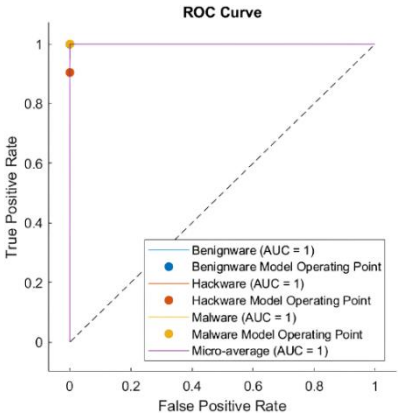
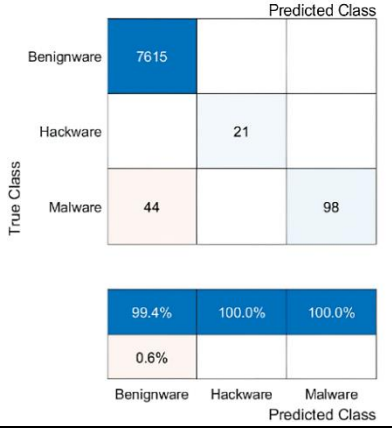
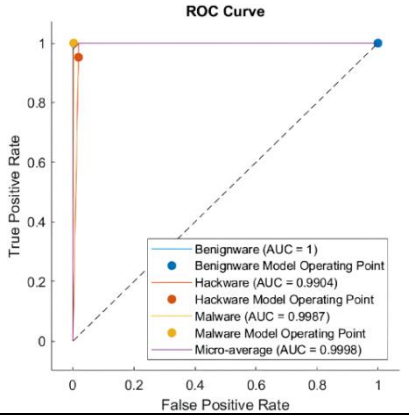
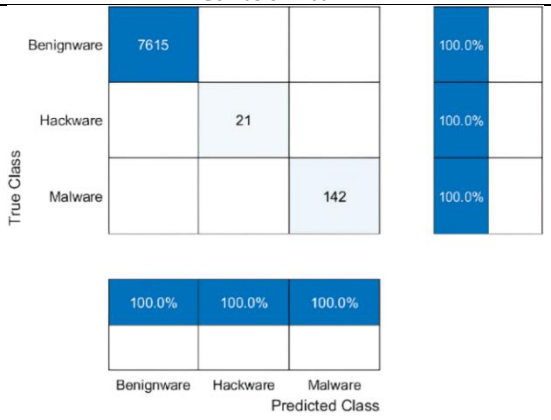
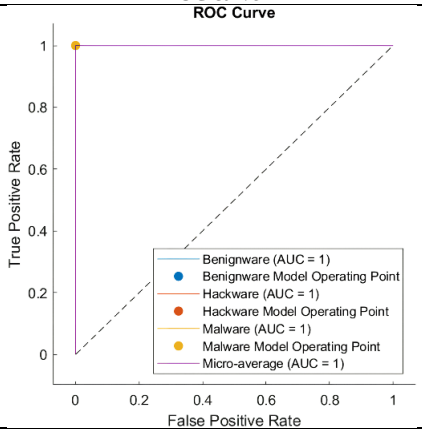
Algorithm	Confusion matrix	ROC curve
SqueezeNet		
ShuffleNet		
MobileNet		
Xception		

Table 2. Confusion matrix and the ROC curve for each model (*continued*)

Algorithm	Confusion matrix	ROC curve
Resnet50		

BIOGRAPHIES OF AUTHORS

Abdelkabir Rouagubi received a Professional Master's degree in Internet of Things and Mobile Services from the National School of Computer Science and Systems Analysis (ENSIAS), Rabat, Morocco. He is currently pursuing a Ph.D. at the Laboratory of Engineering Sciences, Doctoral Studies Center in Science and Technology (CED-ST), National School of Applied Computer Science (ENSA), Ibn Tofail University, Kenitra, Morocco. His main research areas include IoT, security, infrastructure, and cloud. He can be contacted at email: abdelkabir.rouagubi@uit.ac.ma.

Chaymae El Youssofi received her Master's degree in Information System Security from the National School of Applied Computer Science (ENSA) in Kenitra, Morocco. She is currently pursuing a Ph.D. in Computer Science at ENSA, affiliated with the Engineering Sciences Laboratory at Ibn Tofail University, Kenitra, Morocco. Her research focuses on Android security, artificial intelligence, and intelligent systems. She can be contacted at email: chaymae.elyoussofi@uit.ac.ma.

Khalid Chougdali received the Ph.D. degree in Computer Science from Mohamed V-Agdal University, in 2010. He is currently an Associate Professor of computer science with the National School of Applied Sciences, Ibn Tofail University, Kenitra. His main research interests include information security, pattern recognition, and biometrics. He can be contacted at email: khalid.chougdali@uit.ac.ma.