Future Generation Computer Systems 152 (2024) 30–42

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Adversarial attacks and defenses on ML- and hardware-based IoT device fingerprinting and identification

Pedro Miguel Sánchez Sánchez [a,*], Alberto Huertas Celdrán [b], Gérôme Bovet [c], Gregorio Martínez Pérez [a]

[a] Department of Information and Communications Engineering, University of Murcia, Murcia 30100, Spain
[b] Communication Systems Group (CSG), Department of Informatics (IfI), University of Zurich UZH, 8050 Zürich, Switzerland
[c] Cyber-Defence Campus, armasuisse Science & Technology, 3602 Thun, Switzerland

ARTICLE INFO

Keywords:
Adversarial attacks
Device identification
Internet of things (IoT) security
Context attack
Machine learning

ABSTRACT

In the last years, the number of IoT devices deployed has suffered an undoubted explosion, reaching the scale of billions. However, some new cybersecurity issues have appeared together with this development. Some of these issues are the deployment of unauthorized devices, malicious code modification, malware deployment, or vulnerability exploitation. This fact has motivated the requirement for new device identification mechanisms based on behavior monitoring. Besides, these solutions have recently leveraged Machine and Deep Learning (ML/DL) techniques due to the advances in this field and the increase in processing capabilities. In contrast, attackers do not stay stalled and have developed adversarial attacks focused on context modification and ML/DL evaluation evasion applied to IoT device identification solutions. However, literature has not yet analyzed in detail the impact of these attacks on individual identification solutions and their countermeasures. This work explores the performance of hardware behavior-based individual device identification, how it is affected by possible context- and ML/DL-focused attacks, and how its resilience can be improved using defense techniques. In this sense, it proposes an LSTM-CNN architecture based on hardware performance behavior for individual device identification. Then, the most usual ML/DL classification techniques have been compared with the proposed architecture using a hardware performance dataset collected from 45 Raspberry Pi devices running identical software. The LSTM-CNN improves previous solutions achieving a +0.96 average F1-Score and 0.8 minimum TPR for all devices. Afterward, context- and ML/DL-focused adversarial attacks were applied against the previous model to test its robustness. A temperature-based context attack was not able to disrupt the identification, but some ML/DL state-of-the-art evasion attacks were successful. Finally, adversarial training and model distillation defense techniques are selected to improve the model resilience to evasion attacks, improving its robustness from up to 0.88 attack success ratio to 0.17 in the worst attack case, without degrading its performance in an impactful manner.

1. Introduction

The advances in processing and communication technologies achieved in recent years, enabled by more powerful chips and enhanced connectivity, have motivated an explosion in the deployment of Internet-of-Things (IoT) devices [1]. These devices have generated various use cases and scenarios [2], such as Industry 4.0, Smart Cities and Homes, or Healthcare. Therefore, the typology of IoT devices is also very heterogeneous depending on the required capabilities of each scenario. In this context, Single-Board Computers (SBC), such as Raspberry Pi, have gained prominence due to their flexibility, relatively high processing power and reduced cost.

However, this increase in processing power not only comes with advantages. Cybersecurity issues have a greater impact when more powerful IoT devices are compromised, as they can perform stronger attacks such as more petitions in a Distributed Denial of Service (DDoS) or calculations for cryptojacking [3]. Therefore, the securitization of the IoT scenario leveraging SBCs is a key factor in guaranteeing its correct functioning. One of the most important aspects is the correct identification of each device deployed, avoiding the presence of unauthorized devices. Static identifiers such as credentials or certificates were traditionally assigned to the devices, but attackers can clone or modify these to introduce illegitimate entities [4]. To solve this issue,

* Corresponding author.
E-mail address: pedromiguel.sanchez@um.es (P.M. Sánchez Sánchez).

https://doi.org/10.1016/j.future.2023.10.011
Received 19 July 2023; Received in revised form 5 September 2023; Accepted 20 October 2023
Available online 27 October 2023

the literature has widely explored the usage of device behavior and hardware to identify the devices deployed [5].

Behavior-based IoT device identification tasks can be tackled from different granularity levels depending on the environment requirements [5]. There exist two main approaches: *device model or type identification* (e.g., camera, light bulb, etc.), based on characteristics such as network activities or running processes [6], and *individual device identification*, where devices from the same model are differentiated based on hardware manufacturing variations using low-level component analysis or radio frequency fingerprinting [7]. Individual device identification is the one offering the best security guarantees. However, it requires lower-level behavior monitoring, as chip manufacturing variations must be analyzed to differentiate devices with the same hardware and software [8]. This work focuses on individual device identification and the attacks that can be used to spoil the identification: attacks on the data (through manipulation of the device or environment/context) and attacks on the identification techniques (poisoning). In this sense, hardware performance analysis is one of the most exploited techniques for identification, monitoring how components such as CPU, GPU, or RAM behave or perform when executing a certain task [7]. However, in these solutions, an attacker might exploit the component usage or device context to modify the values generating the fingerprint for device identification and interrupt the identification process.

Following the trend in other fields, the application of Machine Learning (ML) and Deep Learning (DL) techniques has gained prominence in IoT security during the last years, including the device identification task [9]. But with the deployment of these techniques, adversarial attacks against ML/DL models have appeared [10], trying to affect the training process or fool the model predictions [11]. These attacks can affect various steps of the ML/DL pipeline. Usually, they target: (i) the training data to poison the model or include backdoors in it [12]; (ii) the testing data to find vulnerabilities in the trained model and change its predictions [13]; or (iii) privacy to infer data from the model and its gradients [14].

ML/DL-based IoT identification solutions have recently been an objective for adversarial attacks [15], demonstrating that these solutions are also affected by context modifications [16] or when malicious adversarial samples are employed in the identification process [17,18]. However, several questions remain when joining hardware behavior-based individual device identification, ML/DL techniques, and adversarial attacks. Some of these research questions are: (i) which is the best-performing ML/DL technique for device identification based on hardware behavior?; (ii) which is the threat model faced by these solutions?; (iii) how device context affects the identification performance?; (iv) how ML/DL-focused adversarial attacks affect the identification process?; (v) how defense techniques improve the resilience to context- and ML/DL-focused adversarial attacks?

To tackle the previous challenges, the main contributions of the present work are:

- A LSTM-CNN neural network architecture for individual device identification based on hardware performance behavior of device CPU, GPU, memory and storage. Hardware monitoring and data collection are performed from the device itself, leveraging the different components as internal reference points to avoid the usage of external sensors. Then, the proposed model considers performance measurements as data points in a time series for data processing and pattern extraction. The architecture performance is compared with different ML/DL classification approaches using the LwHBench dataset [19]. This dataset contains hardware performance and behavior data from 45 Raspberry Pi devices running identical software images. The proposed LSTM-CNN architecture achieves an average F1-Score of +0.96, correctly identifying all the devices with a +0.80 True Positive Rate.
- The threat model definition of the adversarial situations that might affect a self-contained hardware behavior-based individual device identification solution. It encompasses the complete data lifecycle, from the internal fingerprint generation to its evaluation, usually using ML/DL techniques.
- The analysis of the impact of different context- and ML/DL-focused adversarial evasion attacks on the individual identification framework. In terms of context attacks, this analysis shows that temperature does not have a big impact on the performance of the identification solution, and that other context conditions, such as kernel interruptions, can be mitigated during data collection. Regarding ML/DL evasion attacks, the state-of-the-art approaches are successful when performing a targeted attack during evaluation, achieving up to 0.88 attack success rate, and performing a successful device spoofing.
- The application of different defense techniques aiming to improve the model robustness against the previous adversarial attacks. The defense techniques selected are adversarial training and model distillation. The results show that the combination of both techniques reduces the attack impact to ≈0.18 success rate in the worst case. Additionally, state-of-the-art robustness metrics such as empirical robustness or loss sensitivity also show an effective increase.

The code, data, and DL models associated with the previous results and experiments are publicly available in [20] for reproducibility purposes.

The remainder of this article is structured as follows. Section 2 gives an overview of hardware-based individual device identification, context-focused attacks, and ML/DL-focused attacks. Section 3 describes the ML- and hardware-based device fingerprinting solution for individual device identification. Section 4 gives an overview of the threat model that an IoT device identification solution suffers. Section 5 gives an overview of the implementation and impact of the adversarial attack on device identification. Next, Section 6 describes how the application of defense mechanisms enhances the solution resilience against adversarial attacks. Section 7 contrasts the key takeaways and acknowledges limitations. Finally, Section 8 gives an overview of the conclusions extracted from the present work and future research directions.

2. Related work

This section gives the insights required to understand the concepts used in the following sections and reviews the main works in the literature associated with the present one.

2.1. Hardware-based individual device identification

In [7], the authors compared the deviation between the CPU and GPU cycle counters in Raspberry Pi devices to perform individual identification of 25 devices. The identification was performed using XGBoost, achieving a 91.92% True Positive Rate (TPR). Similarly, [16] performed identical device identification using GPU performance behavior and ML/DL classification algorithms. Accuracy between 95.8% and 32.7% was achieved in nine sets of identical devices, including computers and mobile devices. Only the impact of temperature changes was verified.

Sanchez-Rola et al. [21] identified +260 identical computers by measuring the differences in code execution performance. They employed the Real-Time Clock (RTC), which includes its own physical oscillator, to find slight variations in the performance of each CPU. In [8], the author compared the drift between the CPU time counter, the RTC chip, and the sound card Digital Signal Processor (DSP) to identify identical computers. Finally, Deb Paul et al. [22] were able to uniquely identify 20 IoT devices by using an external sensor to leverage the delay

in the signals going through the printed circuit boards (PCBs) of the device. They also verified the identification stability under different temperature conditions, emulating context-attacks.

Other works have explored the usage of Physical Unclonable Functions (PUFs) for IoT device identification [23]. PUFs are digital fingerprinting technologies that leverage inherent manufacturing variations as unique identifiers. It generates hardware-specific cryptographic keys, enhancing security by making the devices virtually impossible to clone or impersonate. However, PUFs are out of the scope of this work, as their usage requires the addition of new hardware components or the low-level modification of the hardware components or firmware, limiting the real-world scalability and usability of the solutions based on this technology [7].

2.2. Context-focused attacks

In hardware-based identification solutions, the context in which the identification code or tasks are executed might influence the collected data and, therefore, the results achieved. In this sense, the temperature can affect the frequency of crystal oscillators or hardware load might introduce delays due to the scheduling between processes. Therefore, a malicious attacker could change these context conditions to affect the identification, making it unusable or generating measurements that mimic another device.

The works described in the previous section briefly discussed context issues that may affect the identification process. [7] showed that device rebooting and other processes running in the device impacted the identification results if proper process isolation mechanisms for data collection were not implemented. Besides, they checked that usual temperature changes based on device load did not affect the results. [16] demonstrated that environment temperatures between 26.4 °C and 37 °C did not affected to the identification results. However, rebooting had an impact on the identification, dropping the results to 50.3% accuracy. The authors also leave voltage variations as a future line to evaluate. In [21], the authors evaluated the identification application under different CPU loads and temperatures, with positive results in both cases. Finally, [8] only mentioned temperature impact analysis as part of future work and no context-based experiments were performed.

As can be seen, none of the previous works on device fingerprinting and identification based on hardware performance behavior has extensively explored the impact that context-focused attacks may have on their results.

2.3. ML/DL-focused adversarial attacks

Adversarial ML/DL [11] is a research field that seeks to develop not only accurate models, but also highly robust models against tampering. It studies the possible attacks against ML/DL models as well as the defense techniques that can secure these. There exist a wide variety of taxonomies for the attacks that an ML/DL model may suffer. In this sense, attacks can be classified in: (i) *Poisoning*, when the model is attacked during training using malicious samples; (ii) *Evasion*, when the model evaluation process is attacked, trying to fool a legitimately trained model; and (iii) *Model Extraction* attacks, where the attacker tries to infer the model based on its predictions.

In this work, the focus is on evasion attacks, as the intention is to full a model trained for device identification, making it misclassify a malicious device for the legitimate one. Here, the main types of attacks are: *non-targeted attacks*, when the objective is just to misclassify a sample to any different class that is not the original one, and *targeted attacks*, when the objective is to evaluate the malicious sample as a concrete objective class. Several evasion attacks can be found as the most common ones in the literature:

- As one of the first evasion attacks for DL, Goodfellow et al. [24] proposed the Fast Gradient Sign Method (FGSM). This attack performs one-step updates in the adversarial sample following the direction of the gradient loss, trying to move the sample into the boundary of a different class. The equation characterizing FGSM can be seen as:

$$X_{adv} = X + \epsilon * sign(\nabla_x J(X, Y)) \tag{1}$$

  where ϵ is a parameter defining the size of the perturbation update and $\nabla_x J(X, Y)$ is the gradient loss function for the sample X.
- Basic Iterative Method (BIM) [25] is a improvement over FGSM by including iterative optimization. This is, applying FGSM several times with small perturbation steps.
- Momentum Iterative Method (MIM) [26] integrates momentum into the iterative FGSM or BIM, avoiding local minimum or overfitting influence in the generated adversarial samples.
- Projected Gradient Descent (PGD) [27] is a generalization of BIM that has no constraints on the iteration steps.
- DeepFool L_2 attack [28] minimizes the Euclidean distance between the original and the adversarial samples by estimating the model decision boundary using a linear classifier.
- Jacobian-based Saliency Map Attack (JSMA) [29] is another common attack in the literature. It uses the Jacobian matrix [30] of the model to find the sensitivity direction of the model and perform feature selection to minimize the number of characteristics modified from the original data sample.
- Boundary Attack [31] generates a random adversarial sample and then performs optimizations in the $L_2 - norm$ of the perturbation to make the sample similar to the original legitimate vector, but maintaining the misclassification result.
- Carlini&Wagner (C&W) Attack [32] proposes a optimization-based adversarial sample generation. It can be applied to three distance metrics: L_0, L_2, L_∞. L_0 measures the number of features to be modified, L_2 measures the Euclidean distance between a benign and adversarial sample, and L_∞ measures the maximum change to any feature.
- Generative Adversarial Network (GAN)-based attack [33] uses GAN models to generate realistic adversarial samples able to fool the classifier.

Numerous defense mechanisms have arisen against the previous attacks and others that may be found in the literature [34]. The objective of these countermeasures is to make the models resistant to adversarial samples. Generally, these can be classified into detection and robustness methods, depending if the aim is to detect crafted malicious samples prior to evaluation or make the model resistant to the evaluation of these, respectively. Besides, defense mechanisms can be attack-specific or attack-agnostic, depending on whether they are focused on improving resilience against a specific attack.

One of the most extended defense techniques to avoid evasion attacks is *Adversarial training* [35], where malicious samples are employed for model training, avoiding the impact of the attacks that generated those samples. *Knowledge distillation* [36] has also been applied for robustness improvement at training. This technique seeks to generate smaller models using the base model outputs as features, hence making the knowledge of the larger model more accessible and efficient to use. It can improve the model robustness by generating smoother decision boundaries and less sensitivity to adversarial samples [37].

2.3.1. Robustness metrics

Robustness metrics provide a quantitative measure to gauge the stability and resilience of neural networks against adversarial perturbations and input variations. The main metrics found in the literature are:

- CLEVER score [38], denoted as Cross Lipschitz Extreme Value for Network Robustness, measures the smallest perturbation required to alter the classification result by utilizing the local Lipschitz constant [39]. The higher the Lipschitz constant, the more sensitive the network is to input perturbations.
- Loss sensitivity [40] calculates the largest variation of the output of a neural network under a small change in its input. Overall, it quantifies the smoothness of a model [41]. A model is considered smoother when there is minimal variation in its output. In mathematical terms, loss sensitivity is depicted by the gradient of the loss function concerning the input data. The gradient magnitude reveals how the loss changes in response to variations in the input. Eq. (2) calculates Loss sensitivity (g), where $\mathcal{L}$ represents the loss function. A smaller value of g (i.e., a smaller variation in the output for perturbed inputs) indicates that the model is smoother and potentially more robust to input variations or adversarial attacks.

$$g = \left\| \frac{\partial \mathcal{L}}{\partial x} \right\|_1 \tag{2}$$

- Empirical robustness, as defined in [28], quantifies the average smallest disturbance necessary to alter a model prediction. This is mathematically represented in Eq. (3), where C stands for a trained classifier, ρ denotes an untargeted attack, and X represents the test data. Initially, adversarial inputs, $\rho(x_i)$, are generated, and the classifier is evaluated against these inputs. Notably, the equation only accounts for the adversarial inputs that successfully deceive the model. Hence, only the indices $I \in 1, 2, n$ where $C(x_i) \neq C(\rho(x_i))$ are considered. The selection of appropriate attacks is intricate, typically relying on the success rate and computational efficiency of FGSM, C&W, and DeepFool attacks.

$$ER(C, \rho, X) = \frac{1}{|I|} \sum_{i \in I} \frac{\|\rho(x_i) - x_i\|}{\|x_i\|} \tag{3}$$

- *Confidence Score* measures the likelihood of accurate sample predictions. It assesses the consistency of predictions; a model with more stable predictions is deemed more robust [41]. Eq. (4) calculates the confidence score as the average of precision scores across all thresholds. Here, T represents the labels, and $Thrs$ denotes the probabilities that a vector is classified correctly.

$$Confidence = \frac{1}{|Thrs|} \Sigma_T \frac{TP}{TP + FP + FN} \tag{4}$$

Each metric offers a unique perspective on robustness, ranging from leveraging local Lipschitz constants to quantifying model smoothness and evaluating the average minimal perturbations required to alter model predictions.

2.3.2. Adversarial ML in IoT identification and security

In this sense, [17] is the closest work to the one at hand. The authors analyzed the impact of different non-targeted and targeted adversarial attacks (FGSM, BIM, PGD and MIM) over a CNN implemented for radiofrequency-based individual device identification. Similarly, Namvar et al. [18] evaluated the resilience of network-based IoT identification ML solutions against adversarial samples generated with FGSM, BIM, and JSMA. They showed how classifier models giving +90% accuracy decrease their performance to 75%–55% when exposed to maliciously crafted samples. From a different perspective, Benegui and Ionescu [42] evaluated the impact of adversarial samples over ML/DL models for user identification based on motion sensors, achieving near to 100% attack success rates with FGSM, JSMA, DeepFool, and Boundary Attacks. Later, [43] demonstrated that a GAN-based attack has more impact than the previous attacks in the user identification context.

From a more generic perspective, [15,44] reviewed the threat of adversarial attacks in ML solutions applied in network security. They proved the high impact of adversarial attacks over ML-based security systems, highlighting the need for more research on attack and defense methods in the area.

Table 1 shows a comparison between the different solutions reviewed in this section. It can be seen how none of the previous works combines the application of context- and ML/DL-focused attacks. Besides, some solutions require external sensors or components to perform the hardware-based identification. Finally, the ML/DL-focused attack papers in the context of device or user identification have not explored the reward from the defense mechanisms available in the literature. Therefore, the present work solves a gap in the literature, providing useful insights in the impact of attack and defense techniques on the context of hardware- and ML/DL-based individual device identification.

3. Individual device identification

The present section describes the ML/DL framework implemented for hardware-based individual device identification. It sets the baseline results for later attack and defense technique impact analysis. This approach follows the higher privilege principle [45]. This is, using the highest privileges for the data collection and processing software deployed in the sensor, isolating it from other processes that might potentially tamper the process. Even though this countermeasure is taken, in the next sections, it is assumed that an attacker could tamper with the solution.

3.1. Dataset collection and preprocessing

This subsection describes how the hardware behavior of the device is monitored from the device itself in order to generate the fingerprint representing its internal characteristics.

3.1.1. Dataset collection

For individual device identification based on hardware behavior, the imperfections in the chips contained in the device should be monitored to be later evaluated. As seen in Section 2, this task has usually been tackled in the literature by comparing components using different crystal oscillators or base frequencies, as deviations in the performance of these components can be noticed from the device itself.

To implement the individual device identification framework, a dataset leveraging metrics related to the hardware components contained in some devices was required. The dataset was named LwHbench and more details are available in [19]. In this sense, the dataset collected performance metrics from CPU, GPU, Memory, and Storage from 45 Raspberry Pi devices from different models for 100 days, enough time to accurately model the performance of the hardware contained in each device. Different functions were executed in these components, using other hardware elements (running at different frequencies) as references for performance measurement. Table 2 summarizes the set of functions monitored. These functions represent a list of common operations executed in each component, trying to measure its performance. Note that other similar operations could be leveraged in the data collection process.

The dataset contains per device model: 505 584 samples from 10 RPi 1B+, 784 095 samples from 15 RPi4, 547 800 samples from 10 RPi3 and 548 647 samples from 10 RPiZero. During the data collection process, several countermeasures were taken to avoid the effect of noise introduced by other processes running in the devices: fixed component frequency, so the hardware performance is stable; kernel level priority, so other processes cannot interrupt the execution of the code; code executed in an isolated CPU core (in multi-core devices), so other processes are not present in the CPU trying to get resources; and the disabling of memory address randomization, so the memory performance is not degraded by accessing at random points of the stack. Besides, the dataset was collected under several temperature conditions, allowing the impact analysis of this context characteristic in the component performance.

Table 1
Comparison of previous works on Context and ML/DL-focused adversarial attacks in identification solutions.

Work	Platform and Objective	Attack type	Attack technique	Defense	Results
si te [8] (2007)	Computer identification	✗	✗	✗	Computer identification based on the comparison of three physical oscillators using t-test statistic
[21] (2018)	Computer identification	Context-focused	CPU load, temperature	Process isolation	265 computers uniquely identified. No effect from CPU load and temperature
[16] (2022)	Computer and mobile identification	Context-focused	Temperature changes and device reboot	✗	95.8% and 32.7% accuracy in nine sets of identical devices. Accuracy drop with device rebooting
[22] (2022)	IoT device identification	Context-focused	Temperature and voltage changes	✗- ML attacks theoretically considered	20 IoT devices identified using the delay in the PCB signals. Temperature and voltage changes impact analyzed.
[7] (2023)	IoT device identification	Context-focused	Temperature changes and device rebooting	Process isolation	91.92% average TPR in 25 devices. No effects from temperature changes and device rebooting
[42] (2020)	User identification	DL-focused	Non-targeted and targeted attacks (FGSM, JSMA, DeepFool, Boundary)	✗	Near to 100% attack success over CNNs models with different depths (from 4 to 12 layers)
[43] (2021)	User identification	DL-focused	GAN-based attack	✗	GAN-generated samples were more effective than FGSM, Deepfool and Boundary when performing adversarial attacks
[18] (2021)	IoT device identification	ML/DL-focused	Non-targeted attacks (FGSM, BIM, JSMA)	✗	Accuracy decreased from +90% to 75%–55%
[17] (2021)	IoT device identification	ML/DL-focused	Non-targeted and targeted attacks (FGSM, BIM, PGD and MIM)	✗	Proven vulnerability to targeted attacks with +80% attack success rate.
This work (2023)	IoT device identification	Context and ML/DL-focused	Context: Temperature changes, CPU load, device rebooting ML/DL: FGSM, BIM, MIM, PGD, JSMA, Boundary Attack, C&W	Process isolation, Adversarial training, Model distillation	+0.96 average F1-Score. Resilience to temperature and process-based context attacks. ML/DL evasion attack resilience improved using model distillation and adversarial training.

Table 2
Features available in LwHBench dataset [19].

Component	Function	Monitored feature
–	timestamp	Unix timestamp
	temperature	Device core temperature
CPU	1 s sleep	GPU cycles elapsed during 1 s CPU sleep
	2 s sleep	GPU cycles elapsed during 2 s CPU sleep
	5 s sleep	GPU cycles elapsed during 5 s CPU sleep
	10 s sleep	GPU cycles elapsed during 10 s CPU sleep
	120 s sleep	GPU cycles elapsed during 120 s CPU sleep
	string hash	GPU cycles elapsed during a fixed string hash calculation
	pseudo random	GPU cycles elapsed while generating a software pseudo-random number
	urandom	GPU cycles elapsed while generating 100 MB using */dev/urandom* interface
	fib	GPU cycles elapsed while calculating Fibonacci number for 20 using the CPU
GPU	matrix mul	CPU time taken to execute a GPU-based matrix multiplication
	matrix sum	CPU time taken to execute a GPU-based matrix summation
	scopy	CPU time taken to execute a GPU-based graph shadow processing
Memory	list creation	CPU time taken to generate a list with 1000 elements
	mem reserve	CPU time taken to fill 100 MB in memory
	csv read	CPU time taken to read a 500 kB *csv* file
Storage	read × 100	100 CPU time measurements for 100 kB storage read operations
	write × 100	100 CPU time measurements for 100 kB storage write operations

Table 3
Feature set extracted for validation.

Operation collected	Python code function	Sliding windows	Statistics extracted	No. features
10 s sleep	*time.sleep(10)*			40
120 s sleep	*time.sleep(120)*			40
string hashing	*hashlib.sha256(str)*	10 Sliding windows.		40
urandom	*os.urandom()*	Group sizes:	Minimum,	40
matrix mul	*vc.cond_mul()*	10, 20, 30, 40,	maximum,	40
matrix sum	*vc.cond_add()*	50, 60, 70,	mean,	40
list creation	*list.append()*	80, 90, 100	median	40
memory reserve	*cgroup.set_memory()*			40
CSV read	*pandas.read_csv()*			40
1st storage read	*os.read()*			40
1st storage write	*os.read()*			40
Total				440

3.1.2. Data preprocessing

As the first preprocessing technique and following the approach of [7], sliding-window-based feature extraction was performed per device, extracting statistical features such as median, average, maximum, minimum, and summation. The reasoning behind this preprocessing is that the distribution of raw feature values from each device may overlap due to the limited variability in the component performance. Therefore, statistical values such as median or average help to differentiate between partially overlapping distributions. Only some of the available raw features were selected for this step, as keeping a low feature number helps to lighten the ML/DL model training. Table 3 describes the set of features extracted from the dataset of each device.

In addition to sliding windows, it was decided to directly evaluate the raw data vectors without the sliding window processing described above. The reasoning behind this approach is that having a large dataset of raw values can work well in the case of DL models, which can automatically extract internal insights from the data. In this approach, only timestamp and temperature features were filtered, using the rest of the values (215 values in total) as features for the models.

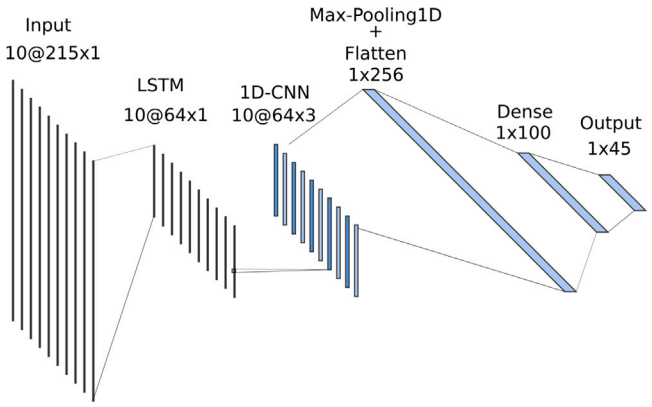

Fig. 1. LSTM-1DCNN architecture proposed.

Finally, it is also decided to perform a time series-based evaluation, concatenating together the available samples in groups of 10 vectors. This grouping technique allows the application of time series DL methods such as LSTM and 1D-CNN models [46,47]. These models can extract complex trends in the data that may achieve better results than the isolated processing and evaluation of individual samples.

3.2. LSTM-1DCNN architecture

This work proposes a client–server framework that leverages an LSTM-1DCNN neural network architecture for the classification of the performance samples obtained from the device. These models have shown good performance in very varied time series scenarios, such as human activity recognition [48], gold price forecast [49], or DNA protein binding [50]. The data generated and preprocessed in the device are sent to a server for model training and later device evaluation and identification.

The network architecture combines LSTM and 1D-CNN layers to extract patterns in the series fed as input. The main benefit of this approach is that combines the recursion patterns extracted by the LSTM layer, due to its memory capabilities, with the space patterns extracted by the 1D-CNN layer, as kernels are applied to close features to derive more complex ones.

Fig. 1 describes the neural network architecture explained above, depicting the size of each layer. The LSTM layer is configured to return sequences, so the 1D-CNN layer can be applied afterward in those sequences. After the 1D-CNN layer, Max-Pooling is applied. Finally, a fully connected layer of 100 neurons is added before the last layer with 45 outputs, one per device. In the implementation, the LSTM layer has 64 neurons, the 1D-CNN layer uses *ReLU* is used as activation function in the hidden layers and *ADAM* is used as optimizer during training (Table 4 show the complete list of hyperparameters tested).

3.3. Classification-based device identification performance

Once the two data preprocessing approaches were applied to generate two datasets, one with raw values and another with sliding-window-based features, the next step was to compare the proposed LSTM-1DCNN model with the most common ML/DL classification approaches. In [7], the authors directly applied ML classifiers using CPU and GPU-related statistical features similar to the ones described in the previous section. Moreover, LSTM and 1D-CNN networks were also tested for the time series approaches. Finally, a more complex multi-input network that combined one LSTM and one 1D-CNN input layer was also implemented for comparison, this model is denoted as Multi_1DCNN_LSTM. The experiments were performed in a server equipped with an *AMD EPYC 7742* CPU and an *NVIDIA A100* GPU.

Table 4
Classification algorithms and hyperparameters tested against the proposed architecture.

Model	Hyperparameters tested
Naive Bayes	No hyperparameter tunning required
k-NN	$k \in [3, 20]$
SVM	$C \in [0.01, 100]$, $gamma \in [0.001, 10]$ $kernel \in \{'rbf', 'linear', 'sigmoid', 'poly'\}$
AdaBoost	$n\_estimators \in [10, 100]$
XGBoost	$lr \in [0.01, 0.3]$, $max\_depth \in [3, 15]$ $min\_child\_weight \in [1, 7]$, $gamma \in [0, 0.5]$, $colsample\_bytree \in [0.3, 0.7]$
Decision Tree	$max\_depth \in [None, 5, 10, 15, 20]$ $min\_samples\_split \in [2, 3, 4, 5]$
Random Forest	$number\_of\_trees \in [50, 1000]$ $max\_depth \in [None, 5, 10, 15, 20]$ $min\_samples\_split \in [2, 3, 4, 5]$
MLP	$n\_layers \in [1, 3]$, $neurons\_layer \in [100, 500]$, $batch\_size \in [32, 64, 128, 256, 512]$ $activation = relu$, $optimizer = [SGD, adam, adamax]$
1D-CNN	$filters = [16, 32, 64, 128]$, $kernel\_size = [3, 5, 7]$, $n\_layers = [1, 2, 3]$, $optimizer = [SGD, adam, adamax]$
LSTM	$neurons = [10, 100]$, $n\_layers = [1, 2, 3]$, $optimizer = [SGD, adam, adamax]$
Multi_1DCNN_LSTM	$input\_layers = [2, 3]$, $cnn\_filters = [16, 32, 64, 128]$, $cnn\_kernel\_size = [3, 5, 7]$, $lstm\_neurons = [10, 100]$ $n\_layers = [1, 2, 3]$, $optimizer = [SGD, adam, adamax]$

Table 4 describes the algorithms and hyperparameters tested against the proposed architecture. Besides, for the algorithms requiring data normalization, *QuantileTransformer* [51] was applied, as the data from the different device models had different distributions based on their hardware capabilities. 80% of the data was used for training and cross-validation, while 20% was used for testing. The train/test splitting was done without vector shuffling to avoid that possible order correlation in the vectors might influence the results.

Table 5 depicts the classification results for each algorithm (with its best hyperparameter setup) in both of the generated datasets. The performance metrics are Accuracy, average Precision, average Recall, and average F1-Score: (TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives)

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall\ or\ TPR = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - Score = \frac{2 \times precision \times recall}{precision + recall} \tag{8}$$

It can be seen that the LSTM-1DCNN model is the classification model with the best classification performance, achieving around 0.96 in all the reported metrics in the case of the usage of raw data features. It also shows how the time series approaches using DL-based models are the ones with the best performance, achieving +0.93 in all the reported metrics and improving XGBoost, which was the model with the best performance in similar literature solutions. Besides, Fig. 2 shows the confusion matrix for each device. It can be appreciated that all devices show +0.80 TPR (True Positive Rate), therefore having positive identification of all of them.

The comprehensive experimentation conducted in this study underscores the superior performance of the LSTM-1DCNN model in device identification tasks, particularly when utilizing raw data features. This model not only outperformed traditional machine learning classifiers but also demonstrated a significant improvement over the best-performing models cited in related literature. From this experiment, it is interesting to observe that the use of raw data features

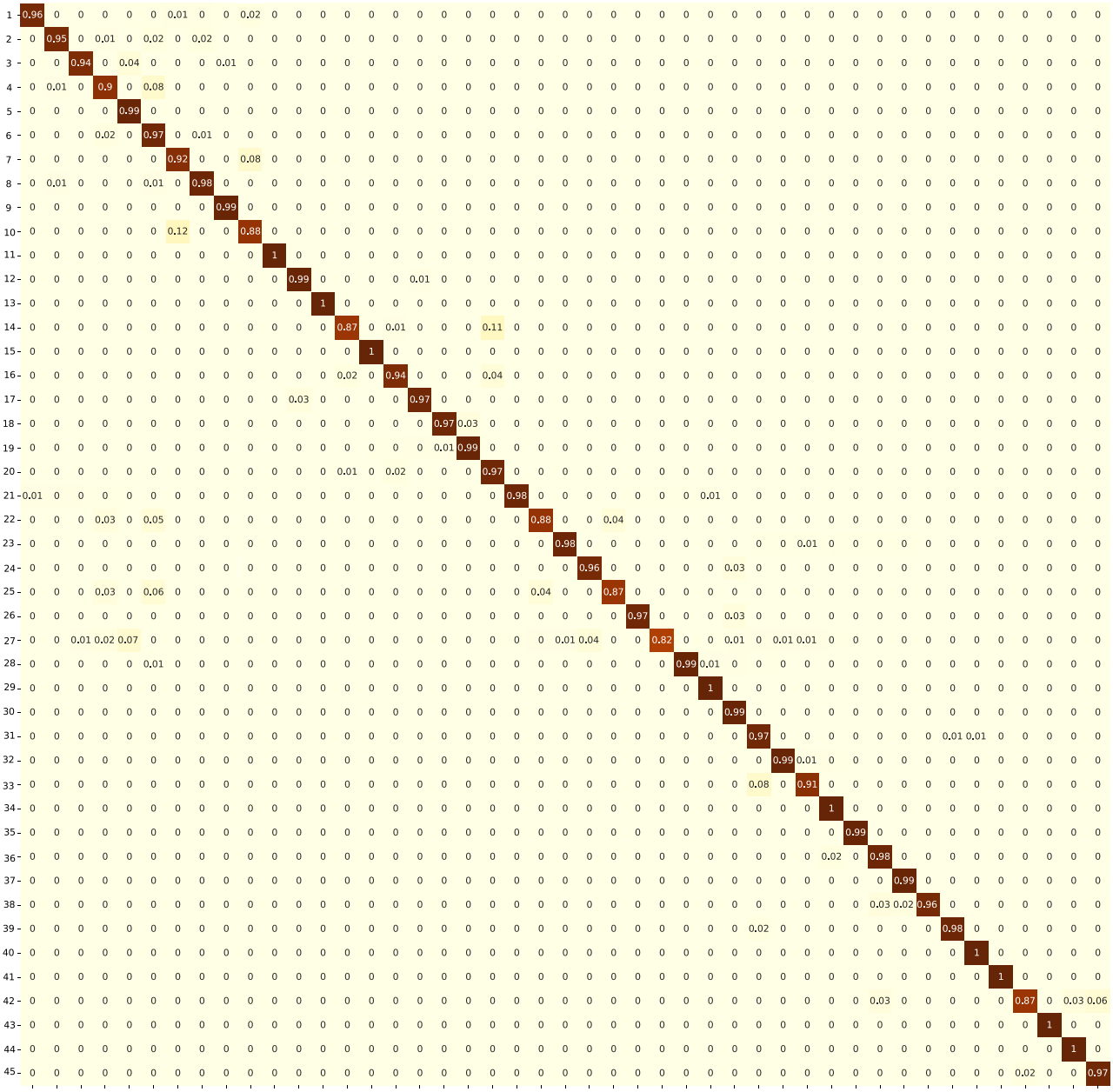

Fig. 2. Individual device identification confusion matrix.

Table 5
Baseline classification model performance.

Model	Raw data features				Sliding-window features			
	Accuracy	Avg. precision	Avg. recall	Avg. F1	Accuracy	Avg. precision	Avg. recall	Avg. F1
	Single vector approaches							
Naive Bayes	0.4569	0.4735	0.4569	0.4473	0.6829	0.6935	0.6829	0.6719
k-NN	0.4526	0.4679	0.4526	0.4472	0.5274	0.5410	0.5274	0.5285
SVM	0.7838	0.7955	0.7829	0.7849	0.7375	0.7434	0.7318	0.7297
AdaBoost	0.0705	0.0060	0.0705	0.0110	0.0706	0.0060	0.0706	0.0110
XGBoost	0.9059	0.9173	0.9056	0.9087	0.7498	0.7655	0.7498	0.7461
Decision Tree	0.7816	0.7896	0.7825	0.7837	0.6932	0.7045	0.6932	0.6910
Random Forest	0.8549	0.8664	0.8542	0.8570	0.7487	0.7615	0.7487	0.7430
MLP	0.8895	0.8960	0.8880	0.8899	0.6840	0.6988	0.6758	0.6749
	Time series approaches (10 values)							
1D-CNN	0.9428	0.9453	0.9428	0.9428	0.6941	0.7170	0.6941	0.6862
LSTM	0.9346	0.9430	0.9346	0.9346	0.7225	0.7345	0.7225	0.7147
LSTM_1D-CNN	**0.9602**	**0.9626**	**0.9602**	**0.9602**	0.7149	0.7287	0.7149	0.7080
Multi_1DCNN_LSTM	0.9535	0.9553	0.9535	0.9535	0.6784	0.6947	0.6784	0.6700

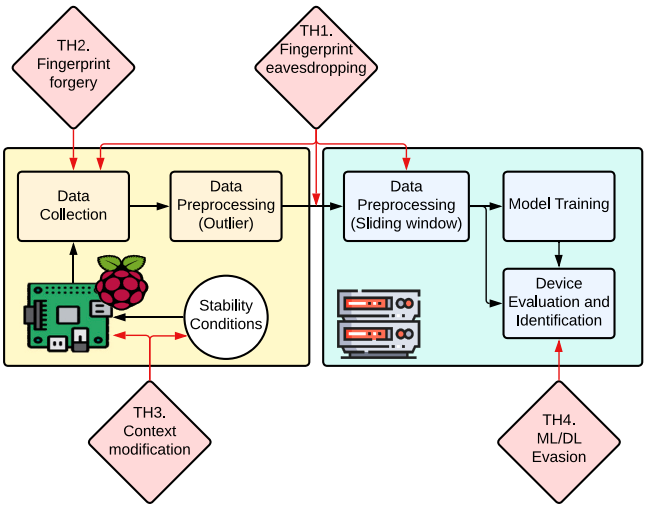

Fig. 3. Threat impact on the different steps of the identification process.

instead of sliding-window ones achieved better results in most of the classification models, something that contrasts with previous works in the literature [7]. This can be a consequence of using a larger dataset than the ones used previously, which also includes information about memory and storage ([7] only included features regarding CPU and GPU).

4. Threat model

This section details the threat model faced by an ML/DL-based device identification solution based on internal hardware performance monitoring. In this sense, an attacker may try to affect the two different sides of the identification: (i) the hardware generating the data or (ii) the ML/DL models in charge of the data evaluation. Fig. 3 reflects the hardware-based identification process and where the different threats can disturb the solution.

- *TH1. Fingerprint eavesdropping and hijacking*. An adversary could read the data composing a fingerprint, either at the level of in-device data collection, communication, or during processing (in a server or the device itself), and then use it in another device to impersonate the identity of the first. This threat implies a reduced knowledge of the fingerprint generation process and the functions and components used during the process. This threat primarily exploits the vulnerabilities in data transmission and storage. An attacker might employ techniques like packet sniffing to capture data during transmission. They could also exploit weak encryption methods or even unencrypted data storage to access the fingerprint data. Once the data is accessed, it can be replayed or used in another device to impersonate the original device. The attacker might also exploit weak authentication protocols or lack of multi-factor authentication to gain unauthorized access.
- *TH2. Fingerprint forgery*. Since the components and frequencies of the devices are public, an attacker with knowledge about the functions that are executed to generate the fingerprint could try to generate a new one that resembles that of a legitimate device. This threat would be triggered possibly on a trial/error or brute force basis. This threat requires thorough knowledge of the implementation of the fingerprint generation process and the values composing the fingerprint. This threat involves a deep understanding of the device hardware and software components. An attacker might use tools to monitor the device performance metrics, such as CPU usage, memory allocation, and power consumption, to reverse engineer the fingerprint generation process. They might also exploit public documentation or even insider information to gain knowledge about the specific algorithms and processes used. Once they have this knowledge, they can craft or modify fingerprints to impersonate legitimate devices.
- *TH3. Context modification*. As the fingerprint is based on data collected from the performance of the execution of certain tasks in the software, an attacker may try to modify the conditions under which the fingerprint is generated. This can neglect to successfully recognize a legitimate device or generate fingerprints that pretend to mimic another device. The context can be modified from several perspectives, for example raising the device temperature (using external tools or exhaustively using the hardware) or introducing software that may add kernel interruptions in the fingerprint collection program, which should be isolated from these interactions as much as possible. This threat exploits the environmental and operational conditions under which the fingerprint is generated. For instance, an attacker might use external heaters or coolers to manipulate the device temperature. They could also run resource-intensive tasks to change the device performance metrics. On the software side, they might introduce malware or other software that interrupts or alters the fingerprint collection process. For instance, a malware that causes frequent CPU spikes could distort the fingerprint. Additionally, rebooting the device frequently or altering its clock speed can also impact the fingerprint generation.
- *TH4. ML/DL evaluation evasion*. In ML/DL-based solutions, an attacker with enough knowledge or access to the evaluation model can be able to craft malicious data samples to fool the ML/DL solution. These samples can target and impersonate a specific device following a trial and error approach or using a targeted attack. This threat capitalizes on the vulnerabilities in ML/DL models. An attacker, with knowledge of the model architecture and parameters, can craft adversarial samples that the model misclassifies. Techniques like gradient ascent on the input data or perturbing the input data in a way that the model output changes can be employed. The attacker might also exploit transferability, where adversarial samples crafted for one model can fool another model. They could use tools and libraries specifically designed for crafting adversarial attacks, or even exploit weak spots in the model architecture, like layers with fewer neurons or weak activation functions. In this sense, several adversarial attacks have been proposed in the literature as shown in Section 2.

Therefore, a proper individual device identification solution has to consider and evaluate the previous threats in order to ensure correct functioning and attack resilience. In essence, while ML/DL-based device identification solutions offer advanced capabilities, they are not immune to threats. Ensuring robustness against these threats requires a multi-faceted approach, encompassing secure data transmission and storage, robust fingerprint generation processes, resilient ML/DL models, and continuous monitoring and updating of the system to counter emerging threats.

5. Adversarial attacks

This section shows the results of the different adversarial attacks tested on the previous ML/DL-based device identification model. These adversarial attacks reflect the implementation of the threats described in the previous section. The objective is to measure how vulnerable the model is to these attacks if an adversary wants to impersonate a given device or disrupt the identification process. The threats reflecting the tested attacks are: *TH2, TH3, and TH4*. *TH1. Fingerprint eavesdropping and hijacking* is assumed to be solved by using encryption in all communications and the higher privilege principle in all the processes of the device and server.

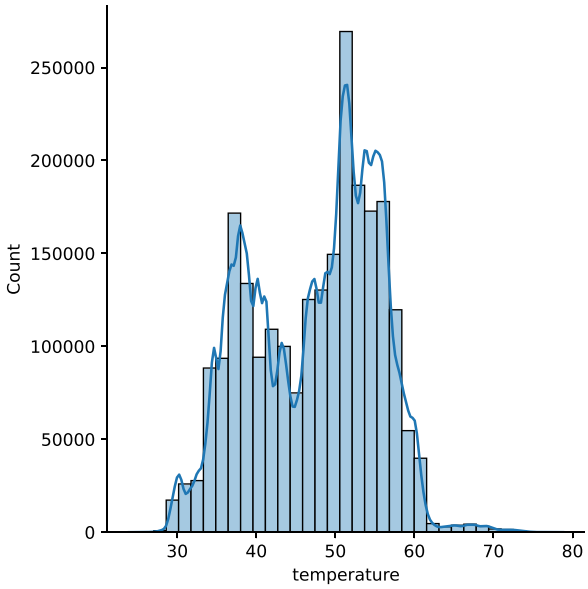

Fig. 4. Temperature distribution in the collected dataset.

5.1. Identification disruption attack (TH3)

The objective of an identification disruption attack is to deny the identification of a legitimate device by performing a context-based attack. Therefore, it is implementing *TH3. Context modification*. Here, the adversary seeks to modify the device hardware performance metrics by changing the device environmental and contextual conditions. Then, the device can no longer be identified properly, and the service is affected. In this attack, the objective is not the generation of adversarial samples that mimic a different device, but the generation of noisy samples that make the identification process of the legitimate device not possible. This experiment evaluates how resilient the identification solution is to context and environmental changes.

Based on the results in previous research [7,16], the dataset was collected considering context stability conditions as mentioned in Section 3. These conditions ensured that the collected data was not affected by other processes in the device. Therefore context-based attacks leveraging factors such as device rebooting or kernel interruption from other processes are not successful because these were already considered during data collection. Moreover, [7] proved that if the stability and isolation measurements are not included during data collection, the hardware-based identification becomes unstable and does not work properly when the context changes.

Regarding temperature, an attacker could try to rise the device temperature by externally interacting with the device with a heat source or by extensively executing resource-demanding tasks in the device. To represent this issue during data collection, the environmental conditions were modified by adding heatsinks to the components and turning on/off fans attached to the devices during data collection. In this sense, Fig. 4 shows the temperature distribution in the samples contained in the dataset. It can be seen that the temperature during data collection varied from 30 °C to +60 °C.

The temperature conditions were randomly varied during data collection. Therefore, the train and test datasets employed in Section 3 do not have a temperature-based bias. However, an attacker might induce new temperature conditions not seen during fingerprint generation to disrupt the identification service. To test this attack, the base dataset of each device was ordered based on the temperature and then divided into train and test samples following an 80/20 ordered split. Then, a new model was generated to compare its performance with the one selected in the previous section. This experiment was repeated both in ascending and descending order. Note that temperature was only used for data ordering and not as a feature.

Table 6
Temperature-based context attack.

Model	Accuracy	Avg. precision	Avg. recall	Avg. F1-Score	Min. TPR
Baseline order	0.9602	0.9626	0.9602	0.9602	0.8045
Temperature ascending order	0.9621	0.9652	0.9621	0.9619	0.6387
Temperature descending order	0.9394	0.9480	0.9392	0.9402	0.6682

Table 6 compares the baseline model with the ones trained by ordering the samples according to the temperature they were collected at. It can be seen how evaluating samples generated at new temperatures does not significantly affect the model performance, with only a 0.03 decrease in the average of the metrics in the case of descending order and even a slight improvement for ascending order. However, the minimum TPR of the evaluated devices (the metric employed for threshold-based identification) is reduced to 0.6682 and 0.6387, respectively. This drop to around 0.65 is observed in two devices in both configurations. Although all the devices can still be identified by setting a threshold in the 0.50 TPR value, these results show that some devices can be more affected by temperature variations.

This experiment demonstrated that temperature conditions do not excessively impact the device identification performance, as the average performance does not degrade when evaluating data generated under temperatures different from the ones during training. The results of the temperature-based context attack experiment revealed that while the identification model performance was not significantly affected by new temperature conditions, there was a notable decrease in the minimum TPR for some devices. This indicates that while the majority of devices remained identifiable, certain devices were more susceptible to temperature variations. This issue, for some devices, can lead to wrong identification if the TPR-based threshold is defined at a high value. Therefore, it can be concluded that the identification approach is resilient to temperature conditions but an eye should be kept to ensure that all the devices meet the performance requirements.

5.2. Device spoofing attacks (TH2, TH4)

In the device spoofing attacks, the adversary performs an evasion attack over the already trained ML/DL model, modifying the evaluated data to change the model outputs, so a malicious device is identified as a legitimate one. In this setup, complete knowledge of the model by the attacker was assumed, therefore having a *white-box* evasion attack. This attack fulfills both *TH2. Fingerprint forgery* and *TH4. ML/DL evaluation evasion*, as malicious fingerprint samples are generated in order to fool the model during evaluation. This attack could also occur as a consequence of a side-channel attack over the data collection process where the adversary is able to modify the samples according to his objective.

As the objective was to fool the device identification model, only targeted attacks make sense to evaluate how easy it is to impersonate other devices. In this sense, one device from each RPi model present in the dataset is selected as the "target class" (constant for attack hyperparameter optimization). Then, the samples from the rest of the devices from each model are used to impersonate that device. Concretely, the selected targets are:

- RPiZero with MAC *80:1f:02:f1:e3:e0*
- RPi1 with MAC *b8:27:eb:87:a7:ce*
- RPi3 with MAC *b8:27:eb:dc:61:2f*
- RPi4 with MAC *dc:a6:32:e4:48:9e*

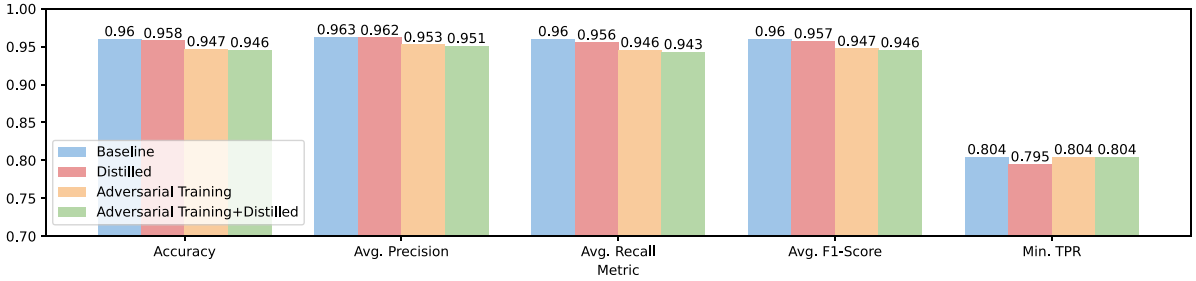

Fig. 5. Performance metrics of the robust models on the legitimate test set.

Table 7
Adversarial attack results.

Attack	Attack success rate	Time
FGSM, $\epsilon = 0.05$	0.3056	8.79 s
BIM, $\epsilon = 0.5$	0.8823	752.64 s
MIM, $\epsilon = 0.05$	0.8537	793.97 s
PGD, $\epsilon = 0.6$	0.8823	748.06 s
NewtonFool, $eta = 0.1$	0.0994	1168.94 s
C&W, L_2	0.1766	63 734.18 s
C&W, L_{inf}	0.0834	142 087.72 s
JSMA, $\theta = 0.1$	Fails	–
Boundary attack	0.2507	379 232.28 s

Table 8
Attack ASR on the robust models.

Attack	Baseline model	Distilled model	Adversarial training	Adversarial training + Distilled
FGSM, $\epsilon = 0.05$	0.3056	0.2725	0.2704	0.1561
BIM, $\epsilon = 0.5$	0.8823	0.3024	0.1482	0.1631
MIM, $\epsilon = 0.05$	0.8537	0.7950	0.1918	0.1784
PGD, $\epsilon = 0.6$	0.8823	0.2741	0.1155	0.1235
NewtonFool	0.0994	0.0600	0.0846	0.0839
C&W, L_2	0.1766	0.1190	0.0953	0.0952
C&W, L_{inf}	0.0834	0.0835	0.0848	0.0841
JSMA, $\theta = 0.1$	Fails	Fails	Fails	Fails
Boundary attack	0.2507	0.0989	0.0886	0.0861

For the implementation of the attacks, the Adversarial Robustness Toolbox (ART) [52] is employed, as it provides straightforward implementations for the attacks detailed in Section 2. Attack Success Rate (ASR) was considered as the metric for the experiments. In targeted attacks, this metric can be defined as the accuracy of the adversarial samples on the malicious labels. Besides, as the use case was related to device identification using performance-based metrics, the distance between benign and adversarial samples was irrelevant. Note that this metric would be important in other use cases, such as image recognition, where the adversarial and benign samples should not be distinguishable by a human.

The attacks selected to be tested are the ones explained in Section 2. However, the DeepFool attack is discarded as it is only untargeted. For this reason, a variant called NewtonFool [53] is used in this work. For each attack, an iteration in its main hyperparameters has been performed to find the most successful configuration (the one with a higher ASR). Table 7 shows the ASR results for each attack together with the execution time of the adversarial sample generation.

Different target devices were also tested with similar results to the reported in Table 7. Note that the exact results may vary if other devices were selected as the target, but the objective was to measure the model vulnerability to adversarial attacks. In this sense, FGSM, BIM, and MIM attacks show an ASR over 0.85. All these attacks achieve a +0.50 success in all the devices employed as adversaries. Therefore, these attacks would fully compromise an identification solution setting a threshold in the 0.50 TPR. In contrast, the NewtonFool attack cannot generate adversarial samples complex enough to target the selected class and only generates the misclassification of the data used as a base for crafting adversarial samples. This experiment demonstrated how the model was vulnerable to targeted adversarial evasion attacks, with over 0.85 ASR in some cases. These attacks could perform device spoofing if he/she has enough knowledge about the model or enough trial and error evaluations.

This experiment underscored the model susceptibility to targeted adversarial evasion attacks. With certain attacks achieving an ASR of over 0.85, it is evident that an adversary, equipped with sufficient knowledge about the model or through iterative evaluations, can successfully execute device spoofing. This revelation underscores the importance of fortifying identification models against such adversarial threats to ensure the security and integrity of device identification processes.

6. Defense techniques

This section analyzes how defense techniques can improve the model robustness against ML/DL evasion attacks. ART [52] was also used to implement the defense techniques for the model-focused attacks, as it includes several model-focused defense techniques. Note that this defense section focuses on device spoofing attacks because the defenses for context-based attacks were already applied during data collection as explained in the previous sections.

The first defense approach applied was to perform adversarial training, using crafted samples as part of the dataset used for model generation. In this sense, untargeted adversarial samples were generated using FGSM, PGD and BIM attacks and concatenated to the original training dataset. Then, a new model was trained from scratch using the new training dataset. Besides, defensive model distillation [37] was also applied over the baseline model to compare the robustness of each resulting model. Finally, the model trained using adversarial samples was also distilled. This combination had unstable behavior during model generation, requiring several attempts to avoid gradient explosion issues. Fig. 5 shows the results of each defense model when evaluating the legitimate test dataset (when no attack is present).

It can be seen how the main performance metrics were not degraded in an impactful manner. Only ≈0.02 performance decrease was noticed in accuracy and average precision, recall, and F1-Score. Besides, the minimum TPR was maintained at 0.8 for the adversarial training model and its distilled version. Once it was verified that the robustness techniques did not decrease the identification performance, the next step was to verify if the new models were robust against the evasion attacks. Table 8 shows the ASR for the different attacks when applied to each model (the baseline one and the ones including robustness techniques).

The combined approach of adversarial training and distillation consistently outperformed other techniques, registering the lowest ASR in seven out of the eight successful attacks. Most notably, the ASR for the most potent attacks on the baseline model, namely BIM, MIM, and PGD, witnessed a significant drop from around 0.85/0.88 to a range of 0.12/0.18. This reduction places the ASR below the 0.5 TPR threshold earmarked for device identification, highlighting the efficacy of the combined defense approach.

Table 9
Robustness evaluation results.

Metric	Baseline model	Distilled model	Adversarial training	Adversarial training + Distilled
CLEVER untargeted, *radius = 8, norm = 2*	*Avg.*:0.0056 *Dev.*:0.0052	*Avg.*:0.0058 *Dev.*:0.0059	*Avg.*:0.0045 *Dev.*:0.0033	*Avg.*:0.0052 *Dev.*:0.0043
CLEVER targeted, *radius = 8, norm = 2*	*Avg.*:0.0218 *Dev.*:0.0363	*Avg.*:0.0239 *Dev.*:0.0305	*Avg.*:0.0210 *Dev.*:0.0243	*Avg.*:0.0240 *Dev.*:0.0394
Loss sensitivity	5.1391	4.7920	6.8399	6.7756
Empirical robustness, *FGSM* $\epsilon = 0.05$	0.0616	0.0613	0.0648	0.0639

Avg.: Average, *Dev.*: Standard Deviation.

The combination of adversarial training and distillation has proven to be a potent defense mechanism against ML/DL evasion attacks. This research underscores the importance of continuously refining and enhancing defense techniques to stay ahead of evolving adversarial threats, ensuring the security and reliability of ML/DL models in real-world applications.

6.1. Robustness metrics

There also exist some additional metrics that evaluate how robust a model is by analyzing its parameters and outputs. Therefore, it is relevant to analyze the state-of-the-art metrics in this sense to quantify how the application of robustness techniques improved the model.

ART [52] includes the following metrics regarding model robustness: Cross Lipschitz Extreme Value for nEtwork Robustness (CLEVER) score [38], Loss sensitivity [40], and Empirical robustness [28]. Table 9 shows the values for these metrics using the test dataset as samples for evaluation. Note that CLEVER score is a metric calculated per sample. Therefore, the average and standard deviation are given in the table.

Although there is not a very great change on these metrics, and even the results for CLEVER untargeted are worse in the adversarial trained models than in the base model, it can be seen how in the case of CLEVER targeted, the score rises 10% from 0.0218 to ≈0.024 in both distilled models. Loss sensitivity score is increased from 5.1391 to 6.8399 and 6.7756 in the adversarial trained and adversarial trained +distilled models, respectively. Finally, Empirical robustness is slightly increased from 0.0616 to 0.0648 and 0.0639, a ≈5%.

While the improvements in robustness metrics might appear subtle, they are indicative of the potential benefits that robustness techniques can bring to the table. Especially in the realm of adversarial attacks, even marginal enhancements in robustness can be crucial in thwarting potential threats. This analysis underscores the importance of continuously refining and employing robustness techniques, ensuring that ML/DL models remain resilient in the face of evolving adversarial challenges.

7. Discussion

This section articulates the constraints inherent to the proposed solution and delivers key insights gleaned from the performed study. Based on the set of experiments conducted in this work and after the comparison with the literature, some important insights and conclusions can be extracted as lessons learned but also as limitations. The list of lessons learned is as follows:

- The use of raw data features provided better results for most of the classification models, as compared to sliding-window features, which contrasted with the previous works. This may occur because the LSTM-1DCNN model can use underlying correlations and information in the raw data to learn more accurate patterns. Another possible reason could be the use of a larger dataset that also includes information about memory and storage.
- Temperature conditions did not excessively impact the device identification performance. The average performance did not degrade when evaluating data generated under temperatures different from the ones during training. However, for some devices, it could lead to wrong identification if the TPR-based threshold is defined at a high value.
- The initial device identification model was found vulnerable to targeted adversarial evasion attacks, with over 0.85 ASR in some cases. These attacks could potentially compromise the identification solution by setting a threshold in the 0.50 TPR.
- The application of defense techniques, specifically adversarial training and model distillation, improved the robustness of the device identification model against ML/DL evasion attacks. The model combining adversarial training and distillation offered the best robustness against such attacks.

In contrast, the following limitations are observed and should be addressed in future research in the area:

- While the device identification model performed well under the context-based attack, some devices could still be more affected by temperature variations, leading to potential misidentification if a more impactful attack is performed. One way to address this limitation is to collect data from a wider range of temperatures during the training process.
- The model that combined adversarial training and distillation had unstable behavior during model generation, requiring several attempts to avoid gradient explosion issues. Such instability necessitated multiple attempts to generate a functioning model, significantly escalating the resources and time required in the model generation process. Furthermore, this instability could potentially result in the generation of less accurate or less generalized models.
- The degradation in performance on benign samples is a trade-off that must be considered when using robustness techniques. While adversarial training and model distillation improved the robustness of the model, the performance metrics were slightly degraded, with about a 0.02 performance decrease in accuracy, and average precision, recall, and F1-Score.
- The robustness techniques may not be effective against all types of evasion attacks. Untested attacks, such as GAN-based methods, could have a high ASR if full access to the identification model is available. Therefore, active iteration of the defense techniques is necessary as attack methods evolve.

8. Conclusions and future work

The explosion in IoT device deployment has motivated the development of new device identification solutions based on hardware behavior and ML/DL processing. However, these solutions face adversarial attacks that try to evade their functionality. This work explored the performance of hardware behavior-based device identification. For

that, the LwHBench dataset containing samples from 45 Raspberry Pi devices running identical software images was used to train ML/DL classifiers in charge of performing individual identification of each device. A DL model combining LSTM and 1D-CNN layers offered the best performance with an average F1-Score of 0.96, identifying all the devices by setting a threshold in +0.80 TPR. This model improved the performance of previous approaches in the literature. Afterward, a temperature-based attack and nine ML/DL evasion attacks were executed to measure the model performance degradation. In this case, the baseline model was robust against temperature context changes. However, some ML/DL evasion attacks successfully fooled the identification system, reaching up to 0.88 attack success rates and demonstrating its vulnerability to these attacks. Finally, model distillation and adversarial training defense techniques were applied during the model training, improving the model resilience to the ML/DL evasion attacks. These techniques improved the model robustness, being the combination of adversarial training and model distillation the best defense approach. Only a ≈0.02 decrease was noticed in accuracy, precision, recall, and F1-Score metrics, without a decrease in the minimum TPR, which is the metric used for setting the threshold for device identification.

In future work, more adversarial attack and defense techniques, such as the ones based on generative models, will be applied to fully improve the solution robustness. Besides, it is planned to add trust metrics in the individual device identification framework, in-depth evaluating the fairness and robustness of the predictions. Another research perspective to be tested is the fully distributed model generation, leveraging federated learning to avoid data sharing and centralization.

CRediT authorship contribution statement

Pedro Miguel Sánchez Sánchez: Methodology, Data curation, Software, Investigation, Formal analysis, Software, Writing – original draft. **Alberto Huertas Celdrán:** Conceptualization, Writing – original draft, Writing – review & editing, Resources. **Gérôme Bovet:** Writing – review & editing, Supervision, Funding acquisition. **Gregorio Martínez Pérez:** Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is published in a previous article and freely available to other researchers.

Acknowledgments

This work has been partially supported by *(a)* the Swiss Federal Office for Defense Procurement (armasuisse) with the CyberForce and DEFENDIS (CYD-C-2020003) projects and *(b)* the University of Zürich UZH, Switzerland.

References

[1] J. Wang, M.K. Lim, C. Wang, M.-L. Tseng, The evolution of the Internet of Things (IoT) over the past 20 years, Comput. Ind. Eng. 155 (2021) 107174.

[2] K. Shafique, B.A. Khawaja, F. Sabir, S. Qazi, M. Mustaqim, Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios, IEEE Access 8 (2020) 23022–23040.

[3] A.H. Celdrán, J. von der Assen, K. Moser, P.M.S. Sánchez, G. Bovet, G.M. Pérez, B. Stiller, Early detection of cryptojacker malicious behaviors on IoT crowdsensing devices, in: NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, 2023, pp. 1–8.

[4] A. Srivastava, S. Gupta, M. Quamara, P. Chaudhary, V.J. Aski, Future IoT-enabled threats and vulnerabilities: State of the art, challenges, and future prospects, Int. J. Commun. Syst. 33 (12) (2020) e4443.

[5] P.M. Sánchez Sánchez, J.M. Jorquera Valero, A. Huertas Celdrán, G. Bovet, M. Gil Pérez, G. Martínez Pérez, A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1048–1077.

[6] Y. Meidan, M. Bohadana, A. Shabtai, J.D. Guarnizo, M. Ochoa, N.O. Tippenhauer, Y. Elovici, ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis, in: Proceedings of the Symposium on Applied Computing, 2017, pp. 506–509.

[7] P.M.S. Sánchez, J.M.J. Valero, A.H. Celdrán, G. Bovet, M.G. Pérez, G.M. Pérez, A methodology to identify identical single-board computers based on hardware behavior fingerprinting, J. Netw. Comput. Appl. 212 (2023) 103579.

[8] T.J. Salo, Multi-factor fingerprints for personal computer hardware, in: MILCOM 2007-IEEE Military Communications Conference, 2007, pp. 1–7.

[9] Y. Liu, J. Wang, J. Li, S. Niu, H. Song, Machine learning for the detection and identification of internet of things devices: A survey, IEEE Internet Things J. 9 (1) (2021) 298–320.

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

[11] K. Sadeghi, A. Banerjee, S.K. Gupta, A system-driven taxonomy of attacks and defenses in adversarial machine learning, IEEE Trans. Emerg. Top. Comput. Intell. 4 (4) (2020) 450–467.

[12] F. Suya, S. Mahloujifar, A. Suri, D. Evans, Y. Tian, Model-targeted poisoning attacks with provable convergence, in: International Conference on Machine Learning, PMLR, 2021, pp. 10000–10010.

[13] H. Kwon, Y. Kim, K.-W. Park, H. Yoon, D. Choi, Multi-targeted adversarial example in evasion attack on deep neural network, IEEE Access 6 (2018) 46084–46096.

[14] M. Al-Rubaie, J.M. Chang, Privacy-preserving machine learning: Threats and solutions, IEEE Secur. Priv. 17 (2) (2019) 49–58.

[15] O. Ibitoye, R. Abou-Khamis, A. Matrawy, M.O. Shafiq, The threat of adversarial attacks on machine learning in network security–a survey, 2019, arXiv preprint arXiv:1911.02621.

[16] T. Laor, N. Mehanna, A. Durey, V. Dyadyuk, P. Laperdrix, C. Maurice, Y. Oren, R. Rouvoy, W. Rudametkin, Y. Yarom, Drawnapart: A device identification technique based on remote GPU fingerprinting, 2022, arXiv preprint arXiv:2201.09956.

[17] Z. Bao, Y. Lin, S. Zhang, Z. Li, S. Mao, Threat of adversarial attacks on DL-based IoT device identification, IEEE Internet Things J. 9 (11) (2021) 9012–9024.

[18] A. Namvar, C. Thapa, S.S. Kanhere, S. Camtepe, Evaluating the security of machine learning based IoT device identification systems against adversarial examples, in: International Conference on Service-Oriented Computing, Springer, 2021, pp. 800–810.

[19] P.M.S. Sánchez, J.M.J. Valero, A.H. Celdrán, G. Bovet, M.G. Pérez, G.M. Pérez, LwHBench: A low-level hardware component benchmark and dataset for Single Board Computers, Internet Things 22 (2023) 100764.

[20] P.M. Sánchez Sána.namvar@student.unsw.edu.auchez, Adversarial_identification, 2023, https://github.com/sxz0/Adversarial_Identification. [Online; accessed 13-July-2023].

[21] I. Sanchez-Rola, I. Santos, D. Balzarotti, Clock around the clock: Time-based device fingerprinting, in: 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1502–1514.

[22] S.D. Paul, F. Zhang, P. SLPSK, A.R. Trivedi, S. Bhunia, RIHANN: Remote IoT hardware authentication with intrinsic identifiers, IEEE Internet Things J. 9 (24) (2022) 24615–24627.

[23] A. Shamsoshoara, A. Korenda, F. Afghah, S. Zeadally, A survey on physical unclonable function (PUF)-based security solutions for Internet of Things, Comput. Netw. 183 (2020) 107593.

[24] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.

[25] A. Kurakin, I.J. Goodfellow, S. Bengio, Adversarial examples in the physical world, in: Artificial Intelligence Safety and Security, Chapman and Hall/CRC, 2018, pp. 99–112.

[26] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, Boosting adversarial attacks with momentum, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9185–9193.

[27] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, 2017, arXiv preprint arXiv:1706.06083.

[28] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.

[29] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2016, pp. 372–387.

[30] K.J. Waldron, S.-L. Wang, S.J. Bolin, A study of the Jacobian matrix of serial manipulators, J. Mech. Transm. Autom. Des. 107 (2) (1985) 230–237.

[31] W. Brendel, J. Rauber, M. Bethge, Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, 2017, arXiv preprint arXiv:1712.04248.
[32] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 Ieee Symposium on Security and Privacy (Sp), Ieee, 2017, pp. 39–57.
[33] W. Hu, Y. Tan, Generating adversarial malware examples for black-box attacks based on GAN, 2017, arXiv preprint arXiv:1702.05983.
[34] I. Rosenberg, A. Shabtai, Y. Elovici, L. Rokach, Adversarial machine learning attacks and defense methods in the cyber security domain, ACM Comput. Surv. 54 (5) (2021) 1–36.
[35] E. Wong, L. Rice, J.Z. Kolter, Fast is better than free: Revisiting adversarial training, 2020, arXiv preprint arXiv:2001.03994.
[36] G. Hinton, O. Vinyals, J. Dean, et al., Distilling the knowledge in a neural network, 2015, arXiv preprint arXiv:1503.02531, 2(7).
[37] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: 2016 IEEE Symposium on Security and Privacy (SP), IEEE, 2016, pp. 582–597.
[38] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, L. Daniel, Evaluating the robustness of neural networks: An extreme value theory approach, 2018, arXiv preprint arXiv:1801.10578.
[39] G. Wood, B. Zhang, Estimation of the Lipschitz constant of a function, J. Global Optim. 8 (1996) 91–103.
[40] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M.S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al., A closer look at memorization in deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 233–242.
[41] F. Yu, et al., Interpreting and evaluating neural network robustness, in: 2019 International Joint Conferences on Artificial Intelligence, 2019, pp. 4199–4205.
[42] C. Benegui, R.T. Ionescu, Adversarial attacks on deep learning systems for user identification based on motion sensors, in: International Conference on Neural Information Processing, Springer, 2020, pp. 752–761.
[43] N. Pourshahrokhi, M. Smith-Creasey, M. Ghassemian, S. Kouchaki, Generative adversarial attacks on motion-based continuous authentication schemes, in: 2021 14th International Conference on Security of Information and Networks (SIN), Vol. 1, IEEE, 2021, pp. 1–6.
[44] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, M. Colajanni, Modeling realistic adversarial attacks against network intrusion detection systems, Digit. Threats: Res. Pract. (DTRAP) 3 (3) (2022) 1–19.
[45] C. Rossow, C.J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, M. Van Steen, Prudent practices for designing malware experiments: Status quo and outlook, in: 2012 IEEE Symposium on Security and Privacy, IEEE, 2012, pp. 65–79.
[46] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures, Neural Comput. 31 (7) (2019) 1235–1270.
[47] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D.J. Inman, 1D convolutional neural networks and applications: A survey, Mech. Syst. Signal Process. 151 (2021) 107398.
[48] K. Xia, J. Huang, H. Wang, LSTM-CNN architecture for human activity recognition, IEEE Access 8 (2020) 56855–56866.
[49] Z. He, J. Zhou, H.-N. Dai, H. Wang, Gold price forecast based on LSTM-CNN model, in: 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE, 2019, pp. 1046–1053.
[50] Y. Zhang, S. Qiao, S. Ji, Y. Li, DeepSite: bidirectional LSTM and CNN models for predicting DNA–protein binding, Int. J. Mach. Learn. Cybern. 11 (4) (2020) 841–851.
[51] M.M. Ahsan, M.P. Mahmud, P.K. Saha, K.D. Gupta, Z. Siddique, Effect of data scaling methods on machine learning algorithms and model performance, Technologies 9 (3) (2021) 52.
[52] M.-I. Nicolae, M. Sinn, M.N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, B. Edwards, Adversarial robustness toolbox v1.2.0, 2018, CoRR, 1807.01069.
[53] U. Jang, X. Wu, S. Jha, Objective metrics and gradient descent algorithms for adversarial examples in machine learning, in: Proceedings of the 33rd Annual Computer Security Applications Conference, 2017, pp. 262–277.

Pedro Miguel Sánchez Sánchez received the M.Sc. degree in computer science from the University of Murcia. He is currently pursuing his Ph.D. in computer science at University of Murcia. His research interests are focused on continuous authentication, networks, 5G, cybersecurity and the application of machine learning and deep learning to the previous fields.

Alberto Huertas Celdrán received the M.Sc. and Ph.D. degrees in computer science from the University of Murcia, Spain. He is currently a postdoctoral fellow associated with the Communication Systems Group (CSG) at the University of Zurich UZH. His scientific interests include medical cyber–physical systems (MCPS), brain–computer interfaces (BCI), cybersecurity, data privacy, continuous authentication, semantic technology, context-aware systems, and computer networks.

Gérôme Bovet is the head of data science for the Swiss DoD, where he leads a research team and a portfolio of about 30 projects. His work focuses on machine/deep learning approaches applied to cyber-defence use cases, with emphasis on anomaly detection, adversarial and collaborative learning. He received his Ph.D. in networks and systems from Telecom ParisTech, France, in 2015.

Gregorio Martínez Pérez is Full Professor in the Department of Information and Communications Engineering of the University of Murcia, Spain. His scientific activity is mainly devoted to cybersecurity and networking, also working on the design and autonomic monitoring of real-time and critical applications and systems. He is working on different national (14 in the last decade) and European IST research projects (11 in the last decade) related to these topics, being Principal Investigator in most of them. He has published 160+papers in national and international conference proceedings, magazines and journals.