IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 December 2023, accepted 29 January 2024, date of publication 5 February 2024, date of current version 24 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3362670

RESEARCH ARTICLE

Intrusion Detection for IoT Environments Through Side-Channel and Machine Learning Techniques

ALEJANDRO DOMÍNGUEZ CAMPOS[1], FELIPE LEMUS-PRIETO[1], JOSÉ-LUIS GONZÁLEZ-SÁNCHEZ[2], AND ANDRÉS CARO LINDO[2]

[1]Extremadura Supercomputing, Technological Innovation and Research Center (CénitS), COMPUTAEX, 10071 Cáceres, Spain
[2]Department of Computer and Telematics Systems Engineering, Universidad de Extremadura, 10003 Cáceres, Spain

Corresponding author: Felipe Lemus-Prieto (felipe.lemus@cenits.es)

This work was supported by the Recovery and Resilience Facility, European Union-NextGenerationEU, through "Programa Investigo" of the Consejería de Educación y Empleo of the Junta de Extremadura under Project PI-0177-C22.

ABSTRACT The rise of the Internet of Things (IoT) technology during the past decade has resulted in multiple applications across a large variety of fields. Some of the data processed using this technology can be specially sensitive, and the devices involved can be prone to cyberattacks, which has resulted in a rising interest in the field of information security applied to IoT. This study presents a method for analyzing an IoT network to detect attacks using side-channel techniques that monitor the power usage of the devices. It shows that it is possible to employ a monitoring system powered by Machine Learning to detect intrusions without interfering with the normal behavior of the devices. Tests yield positive results under a range of scenarios, including using a custom dataset, detecting new attacks previously unseen by the models, and detecting attacks in real time. The main advantages of the proposed system are simplicity, reproducibility (both code and data are made available) and portability, since it can be deployed on all kinds of devices and does not have a high demand of resources. Several deployment strategies are proposed, depending on the structure of the target IoT network and the power constraints of the devices.

INDEX TERMS Cybersecurity, intrusion detection system (IDS), Internet of Things (IoT), machine learning, side-channel.

I. INTRODUCTION

The growth experienced by Internet of Things (IoT) technologies since the term was introduced in [1] has been remarkable. The ability to connect millions of different devices has turned IoT into one of the most important technologies of our time.

However, the existence of so many devices connected to the Internet poses a risk from a security perspective. It is imperative to investigate, design and develop cybersecurity techniques that guarantee the preservation of the CIA (Confidentiality, Integrity and Availability) model.

Given the characteristics of IoT environments and the devices involved, many classic cybersecurity measures cannot be directly reused. It is necessary to study those classic measures and determine whether they can be applied to IoT devices and how to do so, as well as exploring new measures that leverage the specifics of these IoT networks to protect the devices.

In order to increase the security of IoT systems, specific Intrusion Detection Systems (IDS) geared to low-resource devices were introduced. Among them are AI-powered IDS, which employ Machine Learning, Deep Learning, or other AI techniques to detect and mitigate attacks against IoT devices.

AI-powered IDS in IoT environments are a somewhat recent research line, since the article that claims to be the first one to apply Machine Learning to develop an IoT-specific IDS was published in 2017 [2]. Since then, the development of IDS in IoT environments using Machine Learning has received scientific interest.

Side-channeling is a common technique in the IoT world. It involves reading external signals from a device, such as power usage, temperature, or CPU time, to find out the task the device is performing on a given instant. Side-channeling

The associate editor coordinating the review of this manuscript and approving it for publication was Pedro R. M. Inácio.

is more often employed to perform attacks on the devices, but it can also be used to protect them.

The objective of this study is developing an IDS for IoT devices that employs side-channel techniques and Machine Learning to detect software attacks. It studies the viability of creating a simple, flexible and scalable system that can recognize attacks and other undesired device behaviors by reading and analyzing their power usage.

In order to better guide the research process, the following research questions are proposed:

- RQ1: Is it possible to develop a system that uses power usage data to detect attacks against IoT devices?
- RQ2: Can the system distinguish between normal behavior and attacks, including the exact attack type?
- RQ3: Can the system detect new attacks that were not included in the data used to train it?
- RQ4: Can the system be deployed on an end device, distinguishing attacks from the power increase caused by the model itself?
- RQ5: Can the system detect attacks happening in real time without an excessive delay?
- RQ6: Can the system learn to detect attacks using data from different kinds of devices simultaneously?

The main contributions of this work are:

1) The introduction of a system capable of detecting attacks against IoT devices by reading their power usage using an external device.
2) Several datasets containing power usage data for multiple attacks under different scenarios. These datasets could be used by the scientific community to perform further research on the topic.
3) The code used for the whole project. Providing the code ensures the study can be easily reproduced and used for research purposes.

The rest of this paper is structured as follows: Section II shows previous work on this research line. Section III describes the methodology used in this study, which includes the hardware components used, the behavior of the system, the developed attacks, how data was processed, which Machine Learning algorithms were used and how models were trained and tested. Section IV details how the concepts introduced in Section III were implemented. Section V shows and analyzes the results obtained. Section VI contains the conclusions of the study and future research lines. The Appendix shows the power usage of the IoT devices under different circumstances.

II. RELATED WORK

As explained before, IDS in IoT environments were introduced just a few years ago. Since then, all kinds of systems have been developed, with varying characteristics and purposes. These systems have been categorized in a review by Thakkar and Lohiya [3], which introduces a taxonomy for IDS in IoT environments. This taxonomy classifies IDS based on four attributes. See Section III for how our system would be classified under this taxonomy.

Most developed IDS work with network-level data, using packet captures to detect abnormal network behavior. Deep Learning techniques are common in this field (such as [4], which stacks nonsymmetric deep autoencoders to simplify the features passed to a regular classifier; [5], which samples network packets over time to create the features that will be passed to a Deep Learning model; or [6], which employs a distributed Convolutional Neural Network to detect botnet attacks).

Machine Learning systems have been developed as well (such as [7], which uses an existing dataset and multiple Machine Learning algorithms to detect different kinds of attacks; or [8], which compares network activity with previously recorded normal behavior to detect the specific attack that is taking place).

Even though network-based system are quite popular, studies that use specific features of the devices (such as power usage) have also appeared in the last years.

The authors previously published a conference paper that employed device power usage reads to detect attacks [9]. The preliminary results were very promising.

Lightbody et al. [10] shows that different kind of software attacks against IoT devices have specific power usage signatures that could be used to recognize which attack is being performed on a device.

This idea has already been applied before. For example, Azmoodeh et al. [11] propose a ransomware detection system based on the power usage of multiple devices. The authors employ simple classification algorithms and time series analysis, achieving an F1 score value of 92.31%.

Shi et al. [12] propose a more versatile system, capable of detecting both software-based and physical attacks. They employ more complex classification algorithms, such as neural networks. The developed system has two detection modes: A fast one, which can detect anomalies in just 5 seconds but has lower accuracy; and a slow one, which needs 3 minutes to perform a detection but has an accuracy of 99%.

Al-tekreeti et al. [13] propose a system that analyzes the frequency spectrum of the power signal of an Android smartphone to detect malicious behavior. They extract 376 features from that analysis and then perform Principal Component Analysis (PCA) to remove redundant features. Finally, they apply three different classification algorithms (Support Vector Machine, Naive Bayes and Decision Trees), with SVM obtaining the best results (an accuracy between 96 and 99 %).

Ding et al. [14] use Deep Learning and device power data obtained through side-channel techniques to detect real-world IoT attacks with accuracy values averaging 90%. It employs signal preprocessing and feature selection to increase the quality of the data provided to the model.

Other works employ different metrics as input to the models, such as [15] and [16], which use electromagnetic signals recorded using an external device, or [17], which records heat produced by the devices to detect the task

performed by the CPU. This information can then be analyzed to detect malware attacks.

Jaafar et al. [18] combine both power reads and network metrics to detect botnet attacks against several IoT devices, achieving an average F1 score of 0.986 with a Random Forest model.

There are also similar works on the field of Smart Homes. Relevant articles here include Dilraj et al. [19] and Nimmy et al. [20], which propose Machine Learning-based models that can detect anomalous patterns on the power usage of smart cameras, although they do not classify those anomalies in separate attack categories. The first paper uses simpler detection algorithms that achieve an accuracy of approximately 94%, whereas on the second paper, which is a continuation of the first, Deep Learning techniques are employed, such as Deep Feedforward Neural Networks (DFNN), which achieve an accuracy of 99.2%.

Reference [21] reviews 32 datasets containing power usage data in Smart Home environments. It concludes that, other than the dataset proposed by the authors themselves, none of the analyzed datasets can be used for anomaly detection because they are unlabeled.

Deep Learning is also employed to process raw data from IoT devices. For example, [22] treats device data as a time series and uses a Convolutional Neural Network (CNN) to predict the next value. This allows comparing the expected value with the real value, which can be used to flag anomalies. Similarly, [23] employs a CNN to detect botnet attacks through power consumption analysis.

These studies generally conclude that this kind of IDS is very promising, sometimes reaching an accuracy value over 99%. However, most of them present one or more deficiencies in their methodology. The most common problems are:

- Only one algorithm is used to analyze the data, which prevents knowing if that algorithm is the most suitable for the task.
- Only one attack is used, so it is not possible to know if the developed system can detect different kinds of threats.
- Multiple attacks are used, but the model does not distinguish between them; it only reports whether an attack is happening or not.
- Only one IoT device is used, therefore there is no guarantee that the deployed system would work on a different environment.
- The developed system performs well, but it is too complex to be deployed on a real IoT device due to excessive resource requirements.
- The datasets or the code used in the study are not public, so the study cannot be reproduced.

Our proposal presents a detection system based on well-known models that can be deployed on the end devices or on intermediate devices, making it more suitable for real IoT networks. The problems listed above are addressed by employing multiple Machine Learning algorithms, multiple attacks and multiple devices. We also provide the dataset and code used for the project, which makes it easier to reproduce the results and create future derived works.

Table 1 provides a comparison between previous works and this proposal. "No. of unique devices" represents how many unique IoT devices were employed in the study, or how many IoT devices were used to create the dataset employed. "No. of algorithms" indicates how many Machine Learning algorithms or systems were used to analyze the data. "No. of attack types" lists how many types of attack were tested, or how many attack categories the employed dataset contains. "Detection type" refers to the type of predictions made by the developed system. If the system can only detect if an attack is taking place, without providing the exact type, the value will be "Boolean". If it outputs the most likely attack, it will be listed as "Best match". If it can detect multiple attacks at once, "Multiple attacks" will be displayed. "Data available?" will be "Yes" if the dataset used or created for the study is public, or "On request" if it can only be obtained by requesting it. Otherwise, the value will be "No". If multiple datasets are used but only some of them are public, the cell will show the number of public and total datasets used. The same logic is applied for the "Code available?" column, which represents whether the source code of the project is publicly available.

III. METHODOLOGY

One of the main limitations found when starting this research line was the lack of existing datasets that could be used as a starting point. There are some datasets that have been used in the past to train models designed to detect different types of attacks, such as standard network intrusions (KDD CUP [24], NSL-KDD [25], CICIDS2017 [26]), wireless network intrusions (AWID2 [27], AWID3 [28]), or even IoT-specific attacks (N-BaIoT [29], BotIoT [30]). However, none of these datasets was deemed adequate for this study, mainly because it requires power usage data as its input.

The analysis of previous works shows a clear trend towards the generation of custom datasets, which is to be expected, since each IoT network tends to have very specific characteristics. This study follows the same approach. In order to create the dataset, a custom IoT setup has been designed. This setup allows emulating an IoT network, monitor its power usage, launch different types of software attacks and analyze power usage data in order to identify the attacks experienced by the devices.

Under the taxonomy introduced by [3], this proposal would be classified as follows:

Placement strategy: The system allows for a centralized, distributed or hybrid approach depending on which models are deployed and where.

Analysis strategy: The system can be considered hybrid, since it combines signature-based analysis (matching power reads with reads from known attacks) with anomaly-based analysis (power reads that deviate from standard behavior will probably be classified as malicious).

TABLE 1. Proposal comparison.

Proposal	No. of unique devices	No. of attack types	No. of algorithms	Detection type	Data available?	Code available?
Shone *et al.* [4]	0	4	1	Best match	Yes	No
De La Torre *et al.* [6]	9	3	1	Best match	1 / 2	No
Hasan *et al.* [7]	21	8	5	Best match	Yes	No
Anthi *et al.* [8]	7	4	9	Best match	On request	Yes
Azmoodeh *et al.* [11]	1	1	4	Boolean	No	No
Shi *et al.* [12]	1	6	2	Best match	No	No
Ding *et al.* [14]	3	1	1	Boolean	No	No
Pham *et al.* [16]	1	3	4	Best match	Yes	Yes
Garg *et al.* [17]	3	4	1	Best match	No	No
Jaafar *et al.* [18]	6	1	3	Boolean	No	Yes
Nimmy *et al.* [20]	1	2	6	Boolean	On request	No
Munir *et al.* [22]	0	1	1	Boolean	Yes	No
Jung *et al.* [23]	3	1	1	Boolean	No	No
Our approach	3	5	5	Multiple attacks	Yes	Yes

Intrusion type: Since multiple attack scripts that target individual devices were developed, the intrusions can be classified as software attacks.

Attack detection technique: Machine Learning techniques are used to simplify the system and lower resource usage on the devices.

A. HARDWARE COMPONENTS

Side-channel techniques can be applied to almost any device, provided that a method to read their power consumption is available. This means that an IDS that employs power reads to detect attacks does not need to be deployed in a particular kind of IoT device, as opposed to many IDS proposed in previous works, which were designed to operate under a specific environment (for example, a Smart Home).

As a result of this, it was decided to employ generic IoT devices for this study. Generic devices can be easily configured to imitate the behavior of many types of devices, which makes them ideal for research experiments.

Another requirement considered when choosing the devices is that they must have networking capabilities, since most attacks against IoT devices come from the internet or from other compromised devices in the same network.

Two different IoT hardware setups were created for this study. A primary setup, used for the majority of the study, and a secondary setup with another set of devices, used to test how the system behaves when deployed on devices with different power usage profiles (research question RQ6).

Both setups have the same basic structure, which is composed of several end devices, one main device, and a power reading system.

The main device sends commands and attacks to the end devices and monitors them. The power reader reads power usage from the end devices and sends that data to the main device. The end devices perform different actions in order to simulate normal device operation.

It is not required that main device be an IoT device, since it simply acts like a control center. A computer or a server can fulfill the same role.

Details about the devices used can be found in Section IV-A.

B. DEVICE BEHAVIOR AND ATTACKS

Each of the three end devices used for the primary setup has a different behavior. The first device acts as a sensor that sends an HTTP POST request to a server every 10 minutes. The second device plays a video every 10 minutes, and the third device is left in an idle state.

The two additional devices used for the secondary setup are always on an idle state.

The objective of creating different behaviors is determining if the resulting attack detection model can be deployed on heterogeneous networks with several kinds of devices without compromising accuracy.

Some additional tests were also performed with more complex behaviors, such as a device that runs one of the attack detection models every few seconds, in order to detect attacks against itself.

Several software attacks were developed for this study. They are launched by the main device against the end devices, and were created to simulate real-world attacks without posing a real risk to the affected devices.

In particular, three attacks were used to train the models, and two additional attacks were used as validation, to check how the models respond to previously unseen attacks (research question RQ3).

Details about each of the five attacks can be found in Section IV-B.

C. CLASSIFICATION ALGORITHMS

When evaluating which classification models to use for the study, two main requirements were considered:

First, it was necessary to find models capable of accurately and timely predicting if a device is being attacked given its recent power usage data.

Second, it was also important to ensure that deployed models do not have high resource requirements, since IoT devices cannot afford to continuously perform resource-heavy tasks like a regular PC would.

Following those ideas, the following algorithms were selected:

- K-Nearest Neighbors (KNN): Each instance is classified depending on its similarity with previously seen instances.

- Random Forest (RF): Builds multiple decision trees and combines their predictions.
- Extreme Boosting Trees (XBT) [31]: Sequentially builds decision trees by applying the Gradient Boosting algorithm. Each tree tries to improve the results of the previous one.
- Time Series Forest (TSF) [32]: Processes data as a time series, building a forest of trees that are created with a specific algorithm. It uses basic features of the data (mean, std and slope) instead of the original input features.
- Feature Summary (FS): Processes data as a time series, summarizing each instance with a list of relevant features (mean, std, P5, P25, P75 and P90). These features (and not the original power ones) are passed to a Random Forest model for classification.

Support Vector Machine and Logistic Regression were also considered, but they were discarded early due to their poor performance.

D. DATA PROCESSING AND FEATURE EXTRACTION

By default, the main device will read power usage data from the end devices every 0.2 seconds. This raw power data has certain transformations applied to it before being passed to the models.

The transformation process depends on two hyperparameters: *group_amount* (*ga*) and *num_groups* (*ng*).

First, data entries are grouped in groups of *group_amount* elements. Each group will contain the average power usage and the mode of the attacks field of all the grouped entries.

Once the groups have been formed, a new dataset is created. In this new dataset, each row represents a temporal window composed of the last *num_groups* groups for each instant of time.

The purpose of this transformation is twofold: Smoothing the data to reduce the effect of voltage peaks that could introduce errors and reducing the size of the final dataset without losing too much information.

As a result of this transformation, the final dataset will have *num_groups* features. For example, if a model is trained with $ga = 5$ and $ng = 60$, it will receive 60 features as input, with each feature representing an average of the original power usage data in groups of five elements. Since the measurement delay is 0.2 seconds, this model will use the last $60 \times 5 \times 0.2 = 60$ seconds of data to make a prediction.

Fig. 1 represents the window creation process graphically.

The last transformation applied to the data before passing it to the models is normalizing it to the (0, 1) range.

It is important to highlight that attacks are only active during certain periods while the data is being recorded, so the resulting dataset is unbalanced, containing significantly more regular behavior instances than attack instances. For details on how the different datasets used on the study were created and their contents (including label counts), see Section IV-E.

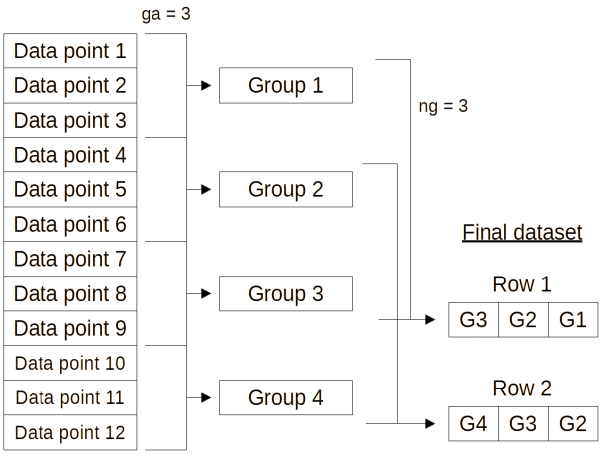

FIGURE 1. Dataset transformation process.

E. MODEL TRAINING AND TESTING

In order to combine all five classifiers with the two hyperparameters that affect how the dataset is built (*ga* and *ng*), all the classifiers were trained and tested with multiple hyperparameter configurations.

The chosen hyperparameter combinations are shown on Table 2. The *Total time* column shows the duration of the time window when using the given hyperparameter values with the set measurement delay of 0.2 seconds. Some other combinations, mainly those with lower total times, yielded worse results and were discarded early.

TABLE 2. Hyperparameter combinations.

ga	*ng*	Total time
5	30	30s
5	60	60s
10	20	40s
10	30	60s
10	40	80s
10	50	100s
20	25	100s

Multiple tests were run under different scenarios. In scenarios that involve both training a model and testing it, 80% of the dataset was used for training, and the remaining 20% was used to test and score the model.

Fig. 2 represents the whole methodology step-by-step.

F. SCENARIOS

As shown by the research questions proposed in Section I, one of the main objectives of this study is to test the system under different kinds of situations. In order to do so, five scenarios were designed. They follow the methodology explained before, but each one has its own peculiarities.

1) SCENARIO 1 - STANDARD BEHAVIOR

This is the main scenario. Its main purpose is answering research questions RQ1 and RQ2. The main device measures power usage on the end devices while they run their set behaviors. A dataset containing 6 hours of power usage

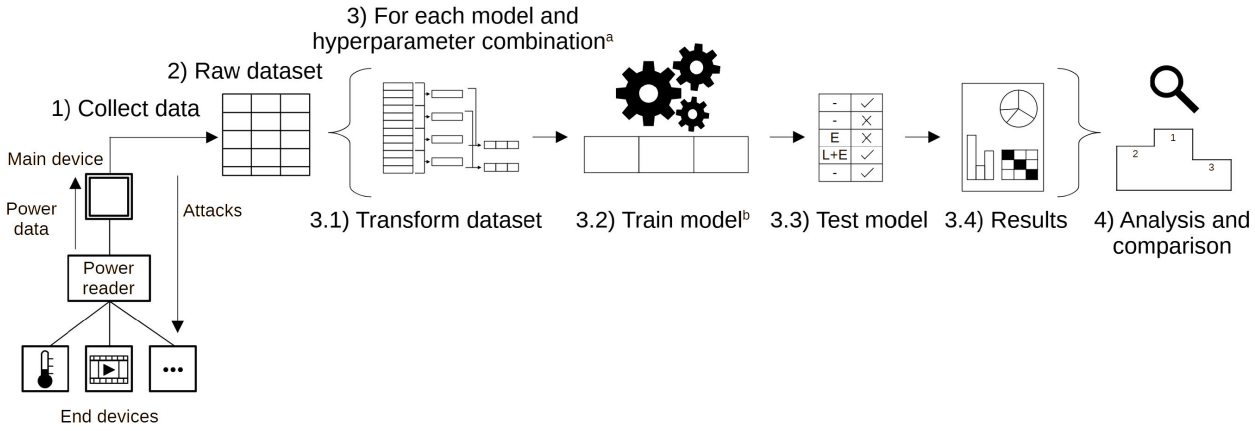

FIGURE 2. Methodology. a) Sometimes only one model is used. b) Training is not always performed.

measurements and attacks on all three end devices was created, and this dataset was then used to train and test each model. The results were used to determine whether the system can properly distinguish attacks or not and which model and hyperparameter combination performed the best.

A second test with a bigger dataset (12 hours) was also performed to find out if increasing the amount of data resulted in higher model accuracy.

2) SCENARIO 2 - VALIDATION ATTACKS

As mentioned in Section III-B, two of the five implemented attacks are used for validation.

In this scenario, two short datasets containing a trace of those validation attacks are passed to each of the models from Scenario 1 in order to see if they can correctly classify them as an attack. This helps answer research question RQ3, since those two attacks were not present in the data used to train the models.

3) SCENARIO 3 - END DEVICE RUNNING MODEL

In a real-world situation, deploying the models on the end devices themselves might be worth considering, since this helps reduce the load on other devices and the IoT network. However, running the model periodically to check for attacks also generates additional power usage, which could be detected as abnormal behavior by a previously trained model.

In order to solve this problem, a new scenario is proposed. In this scenario, one of the end devices runs a model in a continuous loop, checking for attacks every 5 seconds. At the same time, it receives random attacks from the main device, which reads the end device's power usage. The process ends after 6 hours, which results in a 6-hour long dataset that labels the power usage caused by running the model as normal behavior.

Once this new dataset has been created, it is used to train and test a new version of the deployed model, in order to find out how the model performs on this situation. This provides an answer to research question RQ4.

The process is repeated for multiple models, which allows determining which model type is the most accurate when deployed on the end device.

4) SCENARIO 4 - REAL-TIME ATTACK DETECTION

Scenario 4 has some key differences compared to the rest. In this scenario, one of the best performing models from scenarios 1 and 2 is tested live. The model runs on the main device for 30 minutes, trying to detect attacks that are launched against the end devices in real time. No dataset is created.

In order to answer research question RQ5, both accuracy and delay metrics are recorded, which allows determining if the model can detect live attacks and how long it takes to do so.

5) SCENARIO 5 - MULTI-DEVICE DATASET

The last scenario implemented in this study aims to answer RQ6 by creating a dataset that includes measurements from new devices not used during previous scenarios and then training several models with it. Devices from both the primary and secondary hardware setups were used.

G. MODEL POWER USAGE

Since the developed IDS is expected to be deployed on IoT devices, which often have limited power available, we also wanted to briefly compare the power usage of the different models that were used. The power requirements of each model, alongside their accuracy, can be used to determine which one would be a better choice when deploying the system.

IV. IMPLEMENTATION

This section is dedicated to explaining how the proposed methodology has been implemented in more detail.

A. HARDWARE COMPONENTS

As mentioned in Section III-A, two hardware setups were created.

1) PRIMARY SETUP

The primary setup is used throughout the entire study. It is composed of three end devices, one main device, and a power reader.

The three end devices are Raspberry Pi 3 Model B. They can be powered through their general purpose in/out pins (GPIO), which makes it easy to insert a device to read their power usage.

The power usage measurement system used is known as INA3221. It is a low-cost power usage sensor. It has three different channels, so it can measure the power usage of up to three devices at the same time. The INA measures electric current, so this metric is used as an indicator of a device's power usage.

The main device is a Raspberry Pi 4 Model B. This device communicates with the INA3221 using the i2c communication protocol to read power usage data and write it to a file.

A diagram showing the resulting circuit is shown on Fig. 3. The bottom of the image shows the three end devices, the top shows the main device, the center shows the INA3221 device and the left shows the power source that powers the end devices.

2) SECONDARY SETUP

In order to prove that the system can work with multiple kinds of devices (as proposed in research question RQ6), an additional setup using different devices was also created.

The secondary setup is very similar to the primary setup, however it only has two end devices: An Odroid N2 board [33] and an Asus Tinker Board [34]. They are both connected to the INA3221 so the main device (which is still a Raspberry Pi 4) can read their power usage.

These two devices were chosen because their power usage is different than that of the Raspberries.

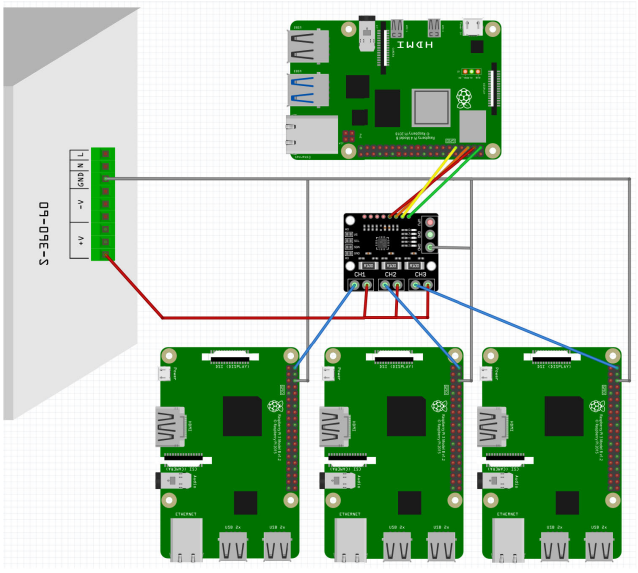

FIGURE 3. Circuit connection diagram for the primary setup.

The secondary setup, unlike the primary setup, is only used during the final part of the study.

B. SOFTWARE ATTACKS

The following attacks were run while creating the datasets that were later used to train the models:

- Mining attack: A cryptocurrency mining program [35] is run on the target device, greatly increasing its CPU usage. The program uses all available cores on the device (four in our setup).
- Login attack: A program run from the main device tries to guess the password of the target device employing brute force to gain unauthorized access. The attack launches multiple concurrent SSH connections, which also forces the target device to handle all the requests at once. The attack has been implemented using the *Hydra* software [36].
- Encryption attack: An encryption program is sent to the target device and executed. The program encrypts some of the files on the target device, simulating the behavior of a ransomware virus.

Following the idea proposed in research question RQ3, two additional attacks were used for validation. They are not present in the datasets used to train the models.

- Password attack: The remote device is forced to run a script that tries to crack a password hash through brute force by computing SHA-256 hashes. The program sleeps at random intervals to avoid being detected.
- Lite Mining attack: Similar to the Mining attack, but it only uses two threads instead of four. This reduces its power usage, making it harder to detect.

See the Appendix for a set of graphs displaying the power usage of each device behavior and attack.

C. CLASSIFICATION ALGORITHMS

All the chosen classification algorithms listed in Section III-C were implemented in Python 3.11.2 using the *scikit-learn* [37] library.

D. DATA RECORDING AND FORMATTING

The main device reads the power usage of all three end devices every 0.2 seconds. The reads for each device are stored in separate CSV files, one for each device. The files contain three columns: A UNIX timestamp, an integer representing the active attacks and the electrical current in mA.

Since the main device keeps track of which attacks are currently active on each device, this ensures that the resulting dataset is properly labeled automatically.

E. SCENARIOS

1) SCENARIO 1 - STANDARD BEHAVIOR

Both datasets used in this scenario were created with the same parameters, only varying in length. They are provided here to facilitate reproduction of the results.

- Event generation seed: 1693795488
- Attacked devices: sensor (1), video (2) and idle (3)

- Duration: 360 minutes / 720 minutes, depending on the dataset
- Time between attacks: 30±10 minutes
- Minimum time between attacks: 60 seconds
- Attacks: Mining (0), Login (1) and Encryption (2)
- Attack duration: 3±1 minutes
- Minimum attack duration: 10 seconds
- Multiple attack chance: 20%
- Short attack chance: 20%
- Short attack average duration: 20 seconds

The ''time between attacks'' and ''attack duration'' fields represent the parameters of the normal distributions used to generate delays between attacks and the duration of the attacks.

Table 3 lists the amount of instances of each type on the 6-hour dataset. Table 4 lists the amount of instances on the 12-hour dataset.

TABLE 3. Instance count - Scenario 1, 6h dataset.

Instance type	Count	Percent
No attack	299353	92.52%
Mining	7791	2.41%
Login	4767	1.47%
Mining + Login	796	0.25%
Encryption	7656	2.37%
Mining + Encryption	2271	0.7%
Login + Encryption	34	0.01%
Mining + Login + Encryption	885	0.27%
Any attack	24200	7.48%
Total	323553	100%

TABLE 4. Instance count - Scenario 1, 12h dataset.

Instance type	Count	Percent
No attack	596017	92.28%
Mining	12224	1.89%
Login	17008	2.63%
Mining + Login	2537	0.39%
Encryption	14331	0.39%
Mining + Encryption	1464	0.23%
Login + Encryption	719	0.11%
Mining + Login + Encryption	1567	0.24%
Any attack	49850	7.72%
Total	645867	100%

2) SCENARIO 2 - VALIDATION ATTACKS

On this scenario, models were run in multi-prediction mode. In this output mode, models will output the probability of each of the possible scenarios. That is, the chance of a power read corresponding to normal behavior and the chance of it corresponding to each of the attack combinations the model has been trained with.

In order to determine the accuracy of the models, their output needs to be converted to a boolean prediction. This is accomplished through the usage of a threshold value. If the total attack chance reported by the model is greater or equal than the threshold, the system considers that an attack is taking place.

We set this threshold to 0.35 after manually tuning it through multiple tests, in an attempt to balance false positives and false negatives.

In a real-world setting, the user would follow a similar approach, changing this threshold in order to control the sensitivity and the false positive rate of the model. A higher threshold reduces the amount of false positives, but risks missing a real attack; whereas a lower threshold increases the odds of detecting an attack, but could result in more false alarms, which would require re-training the model with more data to avoid them.

The parameters used to create the two datasets used in this scenario were the following:

- Event generation seed: 1008264224
- Attacked devices: sensor (1), video (2) and idle (3)
- Duration: 60 minutes
- Time between attacks: 10±5 minutes
- Minimum time between attacks: 60 seconds
- Attacks: Password (3) / Lite Mining (4), depending on the dataset
- Attack duration: 2±1 minutes
- Minimum attack duration: 10 seconds
- Multiple attack chance: 0%
- Short attack chance: 20%
- Short attack average duration: 20 seconds

Each of the two datasets contains 1 hour of data for each of the three devices. Table 5 lists the amount of instances of each type on the Lite Mining dataset, whereas Table 6 lists the amount of instances of each type on the Password dataset.

TABLE 5. Instance count - Scenario 2 - Lite Mining dataset.

Instance type	Count	Percent
No attack	44925	83.31%
Lite Mining	9003	16.69%
Total	53928	100%

TABLE 6. Instance count - Scenario 2 - Password dataset.

Instance type	Count	Percent
No attack	44916	83.31%
Password	9000	16.69%
Total	53916	100%

3) SCENARIO 3 - END DEVICE RUNNING MODEL

The chosen device for this scenario was the idle device.

Three different models were tested, which resulted in three datasets being generated. They share the same parameters, which means they all have the same amount and types of instances. The difference between them is the power usage values, since it depends on the model being run on the device.

- Event generation seed: 1116918666
- Attacked devices: idle (3)
- Duration: 360 minutes
- Time between attacks: 15±5 minutes
- Minimum time between attacks: 60 seconds

- Attacks: Mining (0), Login (1) and Encryption (2)
- Attack duration: 3±1 minutes
- Minimum attack duration: 10 seconds
- Multiple attack chance: 20%
- Short attack chance: 20%
- Short attack average duration: 20 seconds

A higher attack rate was used to compensate for the fact that this dataset only contains data for a single end device, and therefore is three times smaller than a regular dataset. This ensures that there are enough instances of each attack type.

Table 7 lists the amount of instances of each type.

TABLE 7. Instance count - Scenario 3 datasets.

Instance type	Count	Percent
No attack	90582	84%
Mining	5423	5.03%
Login	2361	2.19%
Mining + Login	1227	1.14%
Encryption	7223	6.7%
Mining + Encryption	331	0.31%
Login + Encryption	50	0.05%
Mining + Login + Encryption	637	0.59%
Any attack	17252	16%
Total	107834	100%

4) SCENARIO 4 - REAL-TIME ATTACK DETECTION

Scenario 4 is aimed at knowing how well the model detects all types of live attacks and how long it takes to detect them. Since normal behavior periods are not specially useful for this scenario, a much higher attack rate is used.

Just like in Scenario 2, the model is run in multi-prediction mode with a threshold value of 0.35.

Scenario 4 does not have a dataset file, however the attacks used during the test were also created with a certain set of parameters. Those parameters are listed here.

- Event generation seed: 2143702874
- Attacked devices: sensor (1), video (2) and idle (3)
- Duration: 30 minutes
- Time between attacks: 1.25±0.5 minutes
- Minimum time between attacks: 40 seconds
- Attacks: Mining (0), Login (1), Encryption (2) and Lite Mining (4)
- Attack duration: 1±0.25 minutes
- Minimum attack duration: 30 seconds
- Multiple attack chance: 10%
- Short attack chance: 0%

5) SCENARIO 5 - MULTI-DEVICE DATASET

This scenario combines data from devices from both the primary and secondary hardware setups. In particular, the dataset contains data from the following devices, with 6 hours of measurements per device:

- Raspberry Pi sensor
- Raspberry Pi video player
- Raspberry Pi idle device
- Odroid N2 (idle)
- Asus Tinker Board (idle)

The Odroid N2 and Asus Tinker Board devices have different power usage wile idling than the Raspberry Pi devices, which makes them suitable to create this mixed dataset.

Although a single dataset file is passed to the models, said file was actually created through the composition of two sub-datasets.

The first one is the same dataset used in Scenario 1. Refer to Section IV-E1 for details.

The second one was generated separately for the Odroid N2 and Asus Tinker Board devices, using the following parameters:

- Event generation seed: Not recorded
- Attacked devices: Odroid N2 (4) and Asus Tinker Board (5)
- Duration: 360 minutes
- Time between attacks: 10±5 minutes
- Minimum time between attacks: 40 seconds
- Attacks: Mining (0), Login (1), Encryption (2) and Password (3)
- Attack duration: 5±1 minutes
- Minimum attack duration: 40 seconds
- Multiple attack chance: 20%
- Short attack chance: 25%
- Short attack average duration: 20 seconds

Just like in Scenario 3, this dataset was created with a higher attack rate to compensate for the fact that it only contains data for two devices, and therefore contains less instances.

Table 8 lists the amount of instances of each type on the final dataset.

TABLE 8. Instance count - Scenario 5 datasets.

Instance type	Count	Percent
No attack	455788	84.73%
Mining	18490	3.44%
Login	13411	2.49%
Mining + Login	996	0.19%
Encryption	21066	3.92%
Mining + Encryption	4513	0.84%
Login + Encryption	859	0.16%
Mining + Login + Encryption	885	0.16%
Password	13939	2.59%
Mining + Password	4079	0.76%
Login + Password	821	0.15%
Mining + Login + Password	7	< 0.01%
Encryption + Password	6	< 0.01%
Mining + Encryption + Password	414	0.08%
Login + Encryption + Password	1341	0.25%
Mining + Login + Encryption + Password	1320	0.25%
Any attack	82147	15.27%
Total	537935	100%

6) SCENARIO SUMMARY

Table 9 summarizes the main features of each scenario.

The "Device behavior" column shows which devices were used and what behavior they were running (Blank: Not used, S: Sensor, V: Video, I: Idle, R: Running a model).

TABLE 9. Scenario summary.

Scenario	Device behavior					Attacks launched	Model train behaviors	Test prediction type	Detection type	Research Questions
	1	2	3	4	5					
1	S	V	I			M,L,E	S,V,I,M,L,E	Exact attack	Offline	RQ1,RQ2
2	S	V	I			P,LM	S,V,I,M,L,E	Boolean	Offline	RQ3
3			R			M,L,E	R,M,L,E	Exact attack	Offline	RQ4
4	S	V	I			M,L,E,LM	S,V,I,M,L,E	Boolean	Real-time	RQ5
5	S	V	I	I	I	M,L,E,P*,LM*	S,V,I,M,L,E,P	Exact attack	Offline	RQ6

The "Attacks launched" column specifies which attacks were used (M: Mining, L: Login, E: Encryption, P: Password, LM: Lite Mining). Attacks marked with an asterisk (*) were only launched against devices 4 and 5.

The "Model train behaviors" column shows which behaviors and attacks the models were trained with.

The "Test prediction type" column refers to the format of the output when the model is tested. "Exact attack" means the model is trained and tested with a single dataset. The model is then asked to assign the most likely label to each test instance. "Boolean" means the model is tested with a dataset that contains labels it was not trained with, so the model must simply predict if an attack is happening or not.

Counting the number of individual trials (step 3 depicted on Fig. 2) run on each scenario gives the following totals:

- During Scenario 1, all five algorithms are trained with the seven hyperparameter combinations shown on Table 2, resulting in 35 total tests.
- Scenario 2 works similarly, with all algorithms being trained for both the Lite Mining and Password datasets, therefore it contains 70 total tests.
- For Scenario 3, four different models are run on the end device, and for each model, all 35 possible test combinations are run, resulting in 140 tests.
- A single model is run during Scenario 4.
- Scenario 5 is similar to Scenario 1, with five algorithms and seven hyperparameter combinations, resulting in another 35 tests.

In total, 281 tests were run across all five scenarios. These counts have been summarized on Table 10.

TABLE 10. Individual tests performed.

Scenario	Number of datasets	Number of algorithms	Hyperparameter combinations	Total tests
1	1	5	7	35
2	2	5	7	70
3	4	5	7	140
4	-	1	1	1
5	1	5	7	35
Total				281

V. RESULTS AND DISCUSSION

This section contains the results of training and testing different model variants on each scenario, as well as some commentary on them.

Only the most relevant rows of the result tables are included in this section. A document containing the full version of all the result tables can be found on this article's supplementary material, available online at https://ieeexplore.ieee.org.

In order to measure the quality of the models, the following metrics were used:

- F1 score: The F1 score is the harmonic mean between recall and precision. It is a value between 0 (worst) and 1 (best). Since input data usually has more than two classes, the final F1 score was computed using the *micro* method, which accounts for label imbalance (normal behavior instances are significantly more common in the dataset).
- TP_c (True positives, correct): Instances that represent an attack, were classified as such and the predicted attack type is correct.
- TN (True negatives): Instances that represent normal behavior and were classified as such.
- TP_i (True positives, incorrect): Instances that represent an attack and were classified as such, but the predicted attack type is incorrect.
- FP (False positives): Instances that represent normal behavior but were classified as an attack.
- FN (False negatives): Instances that represent an attack but were classified as normal behavior.

A. TEST RESULTS - SCENARIO 1

The summarized results of testing all the models and hyperparameter combinations are shown on Table 11. Models are listed using the name of the algorithm followed by the value of the *ga* and *ng* hyperparameters. This table only includes the top three variants, as well as the best variant of each model.

TABLE 11. Scenario 1 results - 6h dataset.

Rank	Model	F1 score	TP_c	TN	TP_i	FP	FN
1	TSF 10 50	0.9983	471	5960	1	2	8
2	TSF 5 60	0.9978	953	11926	2	14	12
3	FS 5 60	0.9978	951	11927	2	13	14
7	RF 10 50	0.9966	462	5958	10	4	8
13	XBT 5 60	0.9957	921	11931	30	9	16
28	KNN 5 60	0.9907	872	11915	60	25	35

These results prove that models can properly distinguish between attacks and regular device behavior. The low amount of incorrect true positives (TP_i) shows that they are also capable of determining the exact type of attack with high accuracy. These two facts give a positive answer to research questions RQ1 and RQ2.

The table also shows that TSF is the best performing model, closely followed by FS. RF and XBT also achieve good results, whereas KNN performs the worst.

All the models have a high F1 score, however it should be noted that this happens due to the label imbalance present in the dataset (most instances represent normal behavior, which is usually labeled correctly by all models).

Since the amount of instances seen by each model depends on the *group_amount* parameter, metrics other than the F1 score cannot be directly compared across models with different *group_amount* values.

It is worth noting how TSF and FS are the best models despite the fact that they do not directly use the power reads to make their predictions, relying only on summarizing features. This shows that processing the data as a time series is a more effective approach than treating each power read as an independent variable.

A second test was performed with a bigger dataset containing 12 hours of data. The results can be seen on Table 12.

TABLE 12. Scenario 1 results - 12h dataset.

Rank	Model	F1 score	TP_c	TN	TP_i	FP	FN
1	TSF 5 60	0.998	1972	23776	7	28	16
2	TSF 10 50	0.9977	972	11887	5	10	14
3	FS 5 60	0.9976	1949	23789	18	15	28
8	RF 5 60	0.9949	1914	23753	54	51	27
9	XBT 5 60	0.9946	1901	23759	71	45	23
22	KNN 5 60	0.9906	1800	23756	146	48	49

The results of the 12-hour version of the scenario remain similar to the results of the 6-hour version. This proves that bigger datasets are not required to properly train the models and achieve good results.

Finally, the time required to train and test the models was also measured. The entire set of models trained with the 6-hour dataset took 16 minutes to train and test, whereas the set trained with the 12-hour dataset took 34 minutes. These times can be considered reasonable.

B. TEST RESULTS - SCENARIO 2

In this scenario, model output is based on whether the total attack chance reported by the models is lower or higher than the set threshold of 0.35. This effectively results in a boolean prediction, so the TP_i metric is not used here.

Table 13 lists the most relevant results for each of the tested models when making predictions on the dataset containing traces of the Lite Mining attack. Three additional columns have been added. Th_{max} and Th_{min} display the range of the threshold parameter for which the F1 score of the given model is maximized. *Best F1* shows what that maximum F1 score would be.

The Extreme Boosting Trees algorithm is not included since it does not support probability-based predictions.

Results show that the Lite Mining attack can be detected with an acceptable level of accuracy. The F1 score is slightly lower than the one obtained by the best model in Scenario 1, but this is to be expected.

The best performing model in this scenario is still TSF, but on its *5 60* variant. The amount of false positives is significant. The reason for this is that the model takes too long to flag the end of the attack, which is not too concerning. Regular device behavior does not get incorrectly flagged, and it still would not even if the threshold value was lowered.

Another interesting point is that the highest *Best F1* value corresponds to *RF 10 40*. This means that if the threshold value was raised, *RF 10 40* would achieve the best results, surpassing TSF.

Table 14 lists the main test results for each of the tested models when making predictions on the dataset containing traces of the Password attack.

It can be seen that the F1 scores in this case are significantly lower. The False Negative Rate of the best model is $465/(1353 + 465) = 25.58\%$, so a good portion of the attack instances were not detected. This is to be expected, since the Password attack is the hardest attack to detect, due to its low power usage and its random sleep intervals designed to avoid detection.

The score of the model could be improved by lowering the threshold, as indicated by the Th_{min} and the Th_{max} columns. The optimal threshold for the best model would be between 0.165 and 0.16, however, lowering the threshold this much would raise the amount of false positives and worsen the results of the model when trying to classify Lite Mining instances.

The best performing model in this case is RF. TSF and FS struggle to detect the Password attack, requiring very low thresholds to achieve decent F1 scores.

The results of this scenario provide an answer to research question RQ3: It is possible to detect new attacks that the model was not trained with, although the accuracy might be lower depending on the specific behavior of the attack.

C. TEST RESULTS - SCENARIO 3

Different runs were performed for Scenario 3. On each run, a different model type was running on the end device while the dataset was created. The best performing models from scenarios 1 and 2 were chosen for this scenario (4 in total). Table 15 lists the test results for each of the deployed models.

To add some context to the data, a table listing the test results for all variants of the deployed models has been included in this article's supplementary material.

These results show that the models still achieve great results, despite the fact that a model was running on the tested device, increasing its power consumption. Therefore, it is proven that the system still works under these conditions, answering research question RQ4.

The Feature Summary model performs best when deployed in one of the end devices. It can properly distinguish between attacks and the power spikes caused by running the model itself.

TABLE 13. Scenario 2 results - Lite mining attack.

Rank	Model	F1 score	TP	TN	FP	FN	Th_{max}	Th_{min}	Best F1
1	TSF 5 60	0.9904	1775	8731	76	26	0.435	0.43	0.993
2	RF 10 40	0.9894	918	4300	54	2	0.585	0.575	0.9951
3	TSF 10 30	0.9887	897	4347	37	23	0.4	0.395	0.9898
7	KNN 10 50	0.9828	872	4282	59	31	0.613	0.613	0.9844
21	FS 5 30	0.9606	1804	8473	407	14	0.77	0.765	0.9647

TABLE 14. Scenario 2 results - Password attack.

Rank	Model	F1 score	TP	TN	FP	FN	Th_{max}	Th_{min}	Best F1
1	RF 5 30	0.952	1353	8829	48	465	0.165	0.16	0.9879
2	RF 5 60	0.9376	1203	8740	64	598	0.175	0.17	0.9855
3	RF 10 40	0.9367	621	4319	33	301	0.195	0.19	0.9863
4	KNN 10 40	0.9247	564	4313	39	358	0.29	0.267	0.9302
13	FS 5 60	0.9099	1167	8483	321	634	0.05	0.045	0.9281
18	TSF 20 25	0.8856	169	2153	20	280	0.03	0.025	0.9554

TABLE 15. Scenario 3 results.

Model	F1 score	TP_c	TN	TP_i	FP	FN
FS 5 60	0.997	679	3610	3	2	8
TSF 5 60	0.9947	675	3604	2	6	15
TSF 10 50	0.9902	324	1802	8	0	13
RF 10 50	0.9828	309	1802	31	3	3

The TSF model also achieves a decent result, specially on its 5 60 variant, which once again outperforms the 10 50 variant, just like in Scenario 2.

The Random Forest model had another variant that performed better, so it might be possible to increase its score by deploying and testing the variant *RF 5 60* instead. However, it is unlikely that this will result in a better score than the one obtained by FS and TSF, given the large gap in F1 scores between those models and the best RF variant.

The RF model has a low amount of false positives and false negatives, but a higher amount of incorrect true positives. Most of those come from the misclassification of instances where multiple attacks are active at the same time. Since the model is being run on the device, the difference between a combination of two attacks, such as Mining + Login, and a single attack, such as Mining only, is barely noticeable, so it is not too much of an issue if the model cannot make this distinction.

D. TEST RESULTS - SCENARIO 4

In Scenario 4, the model was running while attacks were being run against the end devices.

The model chosen to be deployed was *TSF 5 60*, since it was the second best performing model from Scenario 1 and the best performing model on the Lite Mining attack test in Scenario 2. All types of attacks were used, except for the Password attack, since Scenario 2 already proved that this model is not good at detecting it.

Besides the previously introduced metrics, two new ones were added for this scenario: *Attack detection delay* and *Attack over detection delay*. These two values measure how long it took the model to detect the beginning and the end of an attack, respectively. They are measured in model runs: If the model detects an attack the first time the model runs after the attack begins, the value will be 0. If it detects it on its second run, it will be 1, and so on.

The result metrics for this scenario are shown on Table 16.

TABLE 16. Scenario 4 results.

F1 score	TP	TN	FP	FN	Average attack detection delay	Average attack over detection delay
0.9181	465	495	61	22	0.55 model runs	1.5526 model runs

The detection delay can be converted to seconds to make it easier to visualize. Since the model is run every 5 seconds, in average, it will be run 2.5 seconds after an attack starts. It then takes $5 \times 0.55 = 2.75$ seconds to detect the attack, so the model takes an average of 5.25 seconds to detect an attack after it starts. Similarly, it needs an average of $2.5+5\times 1.5526 = 10.263$ seconds to detect the end of an attack. These times can be reduced by running the model more frequently.

These results show how the model takes longer to detect the end of an attack than the beginning, which results in an increased amount of false positives. However, this behavior is not specially concerning, since the main purpose of the model is detecting the start of an attack as soon as possible.

Deeper analysis of the output generated by the script shows that the model tends to take longer to detect the Encryption attack, likely since its power usage is similar to the one observed when device 2 plays its video. The model also takes longer to detect the end of a Login attack, since it contains drops in power usage while the attack is active, which makes it difficult to know if a power drop represents the end of the attack or not.

In general, the model can detect live attacks with a reasonable delay, which positively answers research question RQ5.

E. TEST RESULTS - SCENARIO 5

During Scenario 5, the best models from previous scenarios were trained with the multi-device dataset, with 20% of instances being used to test the trained models.

Table 17 shows the most relevant results for this scenario.

TABLE 17. Scenario 5 results.

Rank	Model	F1 score	TP_c	TN	TP_i	FP	FN
1	TSF 5 60	0.998	3256	18159	15	18	10
2	FS 5 60	0.9971	3237	18159	25	18	19
3	TSF 10 50	0.9961	1605	9063	14	9	19
8	RF 5 60	0.9933	3166	18149	97	28	18
12	XBT 5 60	0.99511	3115	18152	142	25	24
28	KNN 5 60	0.9828	2959	18129	280	48	42

The results are very positive. It can be seen how TSF is once again the best performing model. Comparing them to the results TSF obtained in Scenario 1, the F1 score is only slightly lower than the one obtained by the *10 50* variant (0.9983), and higher than the one obtained by the *5 60* variant (0.9978). This proves that it is possible to combine data from different devices and still achieve good prediction results, which answers research question RQ6.

F. MODEL POWER USAGE

As stated before, the power consumption of the different models can be an important factor when determining which one to deploy on a real IoT network.

Power usage was measured by deploying the models on the idle device. The models run every 5 seconds for a total of 2 minutes. The final power usage was obtained by averaging the power usage during that time period.

The average power usage for each model was:

- TSF: 435.519 mA
- FS: 330.7 mA
- KNN: 329.323 mA
- RF: 324.757 mA
- XBT: 324.451 mA

See the Appendix for a set of graphs displaying the power usage of each model.

Results show that the TSF model, while being the most accurate, is also the one with the highest energy consumption by far. The rest of the models all have a similar power usage.

This means that in IoT networks where energy consumption is an important factor (such as in a network of battery-powered devices), using one of the models that has a lower power usage (such as FS) would likely be the best option, as long as the small decrease in model accuracy is acceptable. If power usage is not a concern, using the most precise model (in this case, TSF) would be a better choice.

G. DEPLOYMENT STRATEGIES

The results of the study can be used to proposed different ways in which the system can be deployed. In particular, the following three deployment strategies are proposed.

One of the possible deployment strategies would be a centralized system, where a main device gathers information from the end devices and runs the attack detection model using that information.

Such a deployment would avoid having to run the models on the end devices directly, which reduces the overall power usage of the network and reduces the chances of an attacker tampering with the system.

The best model to deploy on this situation would be the Time Series Forest model, since it has the best results. Both the *10 50* and the *5 60* variants seem like good choices. The former has a higher F1 score in Scenario 1, but the latter seems to perform better in other scenarios.

A distributed deployment approach is also an option. In this case, the models would run directly on the end devices. The main device can be used for monitoring or omitted entirely.

In this situation, the best model to deploy on the final IoT devices could be either Feature Summary or Random Forest. The former is better at distinguishing the exact type of attack that is taking place (as seen in Table 15), whereas the latter is better at detecting new types of attacks (as seen in Tables 13 and 14). Both models have a low power usage, which makes them useful for this situation.

Finally, it would also be possible to deploy the system following a hybrid approach. A simpler model with less energy consumption could be run on the end devices, perhaps in boolean prediction mode, while a stronger model in multi-prediction mode is run on the main device. This approach offers a trade-off between energy consumption and accuracy.

Fig. 4 shows the three proposed deployment strategies.

Regardless of which model is deployed, the user will likely have to adjust the attack detection threshold to balance true and false positives. The optimal threshold will depend on the regular behavior of the device, as well as on whether the model is deployed on a main device or on an end device.

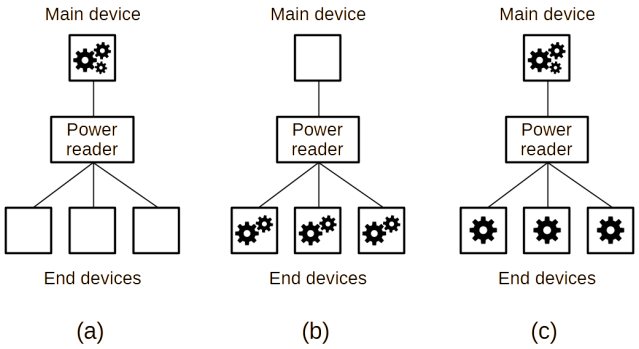

FIGURE 4. Model deployment strategies. (a) Centralized deployment. (b) Distributed deployment. (c) Hybrid deployment.

VI. CONCLUSION AND FUTURE WORK

The results obtained in Section V allow us to draw some general conclusions, as well as to answer the research questions proposed in Section I.

It was proven that an attack detection system based on the power usage of IoT devices is a viable strategy that can detect attacks against the devices (RQ1). The F1 score levels achieved are very positive, although it is partially due to the dataset being unbalanced.

It is also possible to distinguish between different types of attacks, as long as their power usage differs (RQ2). However, the system cannot detect attacks that do not cause a noticeable increase in power usage, since it is the only predictor variable used. The system is also capable of detecting attacks that were not present during its training phase, although the exact accuracy depends on the specifics of the attacks (RQ3).

Models can be run on an end device and successfully detect attacks without being affected by their own power spikes (RQ4). The system can also detect attacks in real time with an acceptable level of delay (RQ5). It is also capable of detecting attacks even when the provided data has been collected from multiple devices with different power reads (RQ6).

A study on the power usage of the different models was performed, concluding that some models might be preferred when operating on a power-constrained network. Three different strategies on how the system could be deployed were also proposed.

It has been proven that it is not necessary to use a large dataset to train the model. Results from Scenario 1 show that the 6-hour dataset is enough to get satisfactory results. Furthermore, increasing the size of the dataset does not improve the results.

After arriving at these conclusions, we also propose the following future research lines:

1) Performing tests with more complex device behaviors, to ensure that the model does not incorrectly flag them as attacks.
2) Testing the models against real malware in order to determine if the system is capable of successfully detecting it as malicious behavior.
3) Employing oversampling and/or undersampling techniques to balance the datasets provided to the models.

APPENDIX
POWER USAGE CHARTS

This appendix contains the power usage charts for all the device behaviors, attacks, and models.

A. DEVICE BEHAVIORS AND ATTACKS

Fig. 5 contains several graphs that show power usage reads on the devices when different behaviors and attacks are run on them.

1) Fig. 5(a) shows power usage on the sensor device. Two small peaks of power usage can be seen near the middle and the end of the graph. Said peaks correspond to sensor reads, which happen every 10 minutes.
2) Fig. 5(b) shows power usage on the video player device. The peaks of power usage that correspond to the moments when the video is played (every 10 minutes) can be seen clearly.
3) Fig. 5(c) shows power usage on the idle device. The power usage remains constant for the most part.
4) Fig. 5(d) displays power usage when the Mining attack is active. This is a very resource-demanding attack, since it exhausts the CPU in all available cores of the device.
5) Fig. 5(e) displays power usage when the Login attack is active. The device must both handle SSH connections and password hash calculations. Since the connections do not come at regular intervals, power usage is constantly rising and falling.

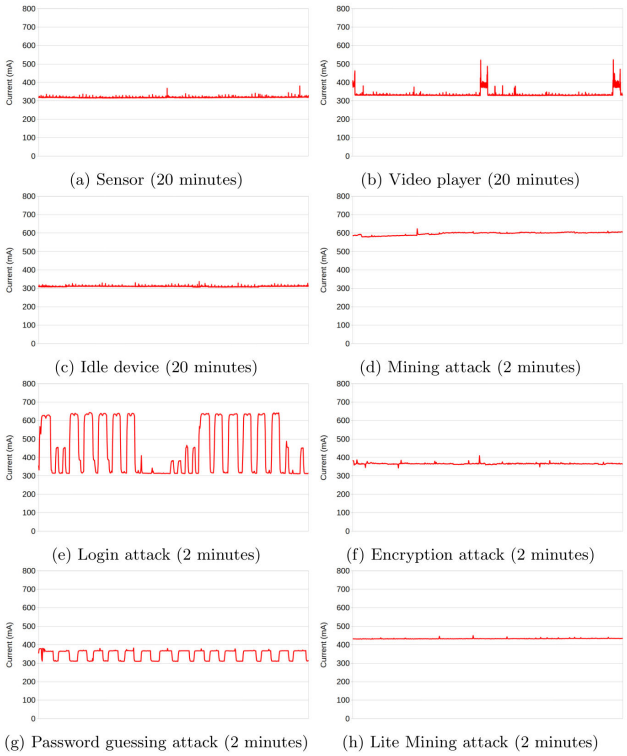

FIGURE 5. Power usage for all the device behaviors and attacks.

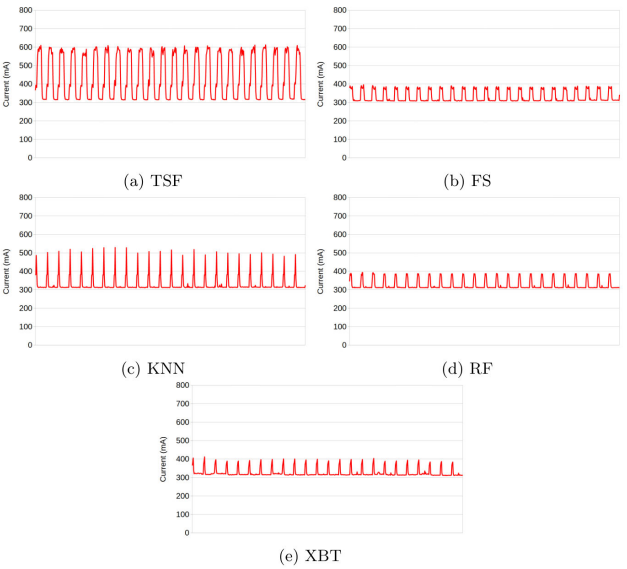

FIGURE 6. Model power usage.

6) Fig. 5(f) displays power usage when the Encryption attack is active. This attack only uses one core, so the total power usage is lower than that of other attacks.
7) Fig. 5(g) displays power usage when the Password attack is active. This attack only uses one core and sleeps at random intervals, so power usage is not especially high nor constant.
8) Fig. 5(h) displays power usage when the Lite Mining attack is active. This attack only uses 2 cores, so the power usage is lower than the one observed during the standard Mining attack.

B. MODELS

Fig. 6 represents the power usage reads of each model over 2 minutes.

ACKNOWLEDGMENT

The datasets and code used for this project, as well as instructions on how to reproduce the results, can be found on the following repository: https://github.com/CenitS-COMPUTAEX/IOT/tree/v1. This paper has supplementary material available at https://ieeexplore.ieee.org provided by the authors.

The authors would like to thank the anonymous reviewers for their comments and suggestions.

REFERENCES

[1] K. Ashton, "That 'Internet of Things' thing," *RFID J.*, vol. 22, no. 7, pp. 97–114, 2009.
[2] P. Shukla, "ML-IDS: A machine learning approach to detect wormhole attacks in Internet of Things," in *Proc. Intell. Syst. Conf. (IntelliSys)*, Sep. 2017, pp. 234–240, doi: 10.1109/INTELLISYS.2017.8324298.
[3] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges," *Arch. Comput. Methods Eng.*, vol. 28, no. 4, pp. 3211–3243, Jun. 2021, doi: 10.1007/s11831-020-09496-0.
[4] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.
[5] W.-C. Shi and H.-M. Sun, "DeepBot: A time-based botnet detection with deep learning," *Soft Comput.*, vol. 24, no. 21, pp. 16605–16616, Nov. 2020, doi: 10.1007/s00500-020-04963-z.
[6] G. De La Torre Parra, P. Rad, K.-K.-R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102662, doi: 10.1016/j.jnca.2020.102662.
[7] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. A. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet Things*, vol. 7, Sep. 2019, Art. no. 100059, doi: 10.1016/j.iot.2019.100059.
[8] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9042–9053, Oct. 2019, doi: 10.1109/JIOT.2019.2926365.
[9] F. Lemus, J. S. Rivero, C. Castañares, A. Caro, and J. L. González, "Detección de ataques en entornos IoT mediante técnicas de canal lateral y de inteligencia artificial," in *Actas de las VIII Jornadas Nacionales de Investigación en Ciberseguridad: Vigo, 21 a 23 de junio de 2023*. Vigo, Spain: Universidade de Vigo, 2023, pp. 299–3043. [Online]. Available: https://hdl.handle.net/11093/4952
[10] D. Lightbody, D.-M. Ngo, A. Temko, C. C. Murphy, and E. Popovici, "Attacks on IoT: Side-channel power acquisition framework for intrusion detection," *Future Internet*, vol. 15, no. 5, p. 187, May 2023, doi: 10.3390/fi15050187.
[11] A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K.-R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *J. Ambient Intell. Humanized Comput.*, vol. 9, no. 4, pp. 1141–1152, Aug. 2018, doi: 10.1007/s12652-017-0558-5.
[12] Y. Shi, F. Li, W. Song, X.-Y. Li, and J. Ye, "Energy audition based cyber-physical attack detection system in IoT," in *Proc. ACM Turing Celebration Conf. China*, May 2019, pp. 1–5, doi: 10.1145/3321408.3321588.
[13] M. Al-Tekreeti, T. Kapoor, R. Manzano, A. Albasir, K. Naik, N. Goel, and A. J. Kozlowski, "Machine learning based malware detection in wireless devices using power footprints," in *Proc. Int. Symp. Syst. Eng. (ISSE)*, Oct. 2019, pp. 1–8, doi: 10.1109/ISSE46696.2019.8984518.
[14] F. Ding, H. Li, F. Luo, H. Hu, L. Cheng, H. Xiao, and R. Ge, "DeepPower: Non-intrusive and deep learning-based detection of IoT malware using power side channels," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 33–46, doi: 10.1145/3320269.3384727.
[15] N. Chawla, H. Kumar, and S. Mukhopadhyay, "Machine learning in wavelet domain for electromagnetic emission based malware analysis," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3426–3441, 2021, doi: 10.1109/TIFS.2021.3080510.
[16] D.-P. Pham, D. Marion, M. Mastio, and A. Heuser, "Obfuscation revealed: Leveraging electromagnetic signals for obfuscated malware classification," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2021, pp. 706–719, doi: 10.1145/3485832.3485894.
[17] N. Garg, I. Shahid, E. Avllazagaj, J. Hill, J. Han, and N. Roy, "ThermWare: Toward side-channel defense for tiny IoT devices," in *Proc. 24th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2023, pp. 81–88, doi: 10.1145/3572864.3580339.
[18] F. Jaafar, D. Ameyed, A. Barrak, and M. Cheriet, "Identification of compromised IoT devices: Combined approach based on energy consumption and network traffic analysis," in *Proc. IEEE 21st Int. Conf. Softw. Quality, Rel. Secur. (QRS)*, Dec. 2021, pp. 514–523, doi: 10.1109/QRS54544.2021.00062.
[19] M. Dilraj, K. Nimmy, and S. Sankaran, "Towards behavioral profiling based anomaly detection for smart homes," in *Proc. TENCON IEEE Region Conf. (TENCON)*, Oct. 2019, pp. 1258–1263, doi: 10.1109/TENCON.2019.8929235.
[20] K. Nimmy, M. Dilraj, S. Sankaran, and K. Achuthan, "Leveraging power consumption for anomaly detection on IoT devices in smart homes," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 10, pp. 14045–14056, Oct. 2023, doi: 10.1007/s12652-022-04110-6.
[21] Y. Himeur, A. Alsalemi, F. Bensaali, and A. Amira, "Building power consumption datasets: Survey, taxonomy and future directions," *Energy Buildings*, vol. 227, Nov. 2020, Art. no. 110404, doi: 10.1016/j.enbuild.2020.110404.
[22] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019, doi: 10.1109/ACCESS.2018.2886457.
[23] W. Jung, H. Zhao, M. Sun, and G. Zhou, "IoT botnet detection via power consumption modeling," *Smart Health*, vol. 15, Mar. 2020, Art. no. 100103, doi: 10.1016/j.smhl.2019.100103.
[24] S. Hettich, "KDD cup 1999 data," in *UCI KDD Archive*. Univ. of California, Department of Information and Computer Science, 1999.
[25] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.
[26] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.
[27] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st Quart., 2016, doi: 10.1109/COMST.2015.2402161.
[28] E. Chatzoglou, G. Kambourakis, and C. Kolias, "Empirical evaluation of attacks against IEEE 802.11 enterprise networks: The AWID3 dataset," *IEEE Access*, vol. 9, pp. 34188–34205, 2021, doi: 10.1109/ACCESS.2021.3061609.

[29] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul. 2018, doi: 10.1109/MPRV.2018.03367731.

[30] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019, doi: 10.1016/j.future.2019.05.041.

[31] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[32] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Inf. Sci.*, vol. 239, pp. 142–153, Aug. 2013, doi: 10.1016/j.ins.2013.02.030.

[33] *Odroid N2 Board*. Accessed: Dec. 14, 2023. [Online]. Available: https://www.hardkernel.com/shop/odroid-n2-with-4gbyte-ram/

[34] *Asus Tinker Board*. Accessed: Dec. 14, 2023. [Online]. Available: https://www.asus.com/us/networking-iot-servers/aiot-industrial-solutions/all-series/tinker-board/

[35] T. Pruvot and L. Jones. (2017). *Cpuminer*. [Online]. Available: https://github.com/tpruvot/cpuminer-multi

[36] M. van Hauser Heuse. (2022). *Hydra*. [Online]. Available: https://github.com/vanhauser-thc/thc-hydra

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

ALEJANDRO DOMÍNGUEZ CAMPOS received the B.Sc. degree in software engineering from the University of Extremadura, Spain, in 2021, and the M.Sc. degree in artificial intelligence from the Polytechnic University of Madrid, in 2022. He is currently a Researcher with the Extremadura Supercomputing, Technological Innovation and Research Center (CínitS).

FELIPE LEMUS-PRIETO was born in Badajoz, Spain, in 1982. He received the B.S. degree in telecommunication engineering from the University of Seville, in 2010, and the M.S. degree in IT management from the University of Extremadura, in 2016, where he is currently pursuing the Ph.D. degree.

From 2010 to 2011, he was a Drive Test Teams Coordinator at Telecommunications Consulting Firm. From 2011 to 2020, he was the Communication and Security Administrator with the Extremadura Supercomputing, Technological Innovation, and Research Center, (Cénits). Since 2020, he has been In Charge of the Networks and Communications Functional Unit at Cénits, where his main activities are the management of the supercomputing infrastructure, the procurement of IT equipment, and participation in research and development projects. His research interests include network communications, information security, and the Internet of Things.

JOSÉ-LUIS GONZÁLEZ-SÁNCHEZ received the Engineering and Ph.D. degrees in computer science from the Polytechnic University of Cataluña, Barcelona, Spain, in 2001.

He was the System and Network Manager for several years at several private and public organizations. He was also the Scientific Director of the Extremadura Supercomputing, Technological Innovation, and Research Center (Cénits). He is currently a full-time Associate Professor with the Department of Computer Systems and Telematics Engineering, University of Extremadura, Spain, and the General Manager of the FundeSalud Foundation.

ANDRÉS CARO LINDO received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Extremadura, Spain, in 1993, 1998, and 2006, respectively.

Since 1999, he has been an Associate Professor with the Department of Computer and Telematics Systems Engineering, University of Extremadura, where he is currently the Laboratory Head of the Media Engineering Group. He has participated in the management of both national and international research and development projects and private financing. He is the co-author of numerous research SCI journal articles and book chapters. His research interests include cybersecurity, big data, machine learning, and pattern recognition.