Enhancing Security on IoT Devices via Machine Learning on Conditional Power Dissipation

Dimitrios Myridakis, Stefanos Papafotikas, Konstantinos Kalovrektis, and Athanasios Kakarountas

1 iSL Laboratory, Department of Computer Science and Biomedical Informatics, University of Thessaly, 35131 Lamia, Greece
2 Department of Informatics, Hellenic Open University, 26335 Patras, Greece; papafotikas@gmail.com
3 Department of Computer Science and Telecommunications, University of Thessaly, 35131 Lamia, Greece; kkalovr@uth.gr
* Correspondence: dmyridakis@dib.uth.gr (D.M.); kakarountas@uth.gr (A.K.)
† This work was conducted when S.P. was studying at Hellenic Open University.

Received: 16 September 2020; Accepted: 23 October 2020; Published: 29 October 2020

Abstract: The rapid development of connected devices and the sensitive data, which they produce, is a major challenge for manufacturers seeking to fully protect their devices from attack. Consumers expect their IoT devices and data to be adequately protected against a wide range of vulnerabilities and exploits. Successful attacks target IoT devices, cause security problems, and pose new challenges. Successful attacks from botnets residing on mastered IoT devices increase significantly in number and the severity of the damage they cause is similar to that of a war. The characteristics of attacks vary widely from attack to attack and from time to time. The warnings about the severity of the attacks indicate that there is a need for solutions to address the attacks from birth. In addition, there is a need to quarantine infected IoT devices, preventing the spread of the virus and thus the formation of the botnet. This work introduces the exploitation of side-channel attack techniques to protect the low-cost smart devices intuitively, and integrates a machine learning-based algorithm for Intrusion Detection, exploiting current supply characteristic dissipation. The results of this work showed successful detection of abnormal behavior of smart IoT devices.

Keywords: IoT; hardware security; smart devices; side-channel analysis and countermeasures; machine learning

1. Introduction

The Internet of Things (IoT) is a new paradigm that is rapidly gaining ground in the scenario of capturing data and making them available for analysis from datacenters distributed worldwide. The basic idea behind this concept is the pervasive presence around people of a variety of computation objects (or things)—such as RFID tags, sensors, motors, cell phones, etc.—which, through unique addressing systems, are capable of interacting with each other, store and exchange data, and collaborate with their ‘neighbors’ to achieve common goals [1]. Initially, IoT appeared in 1999, but since then it has seen rapid growth in recent years, which raises high expectations and predictions of 500 billion devices to be online by 2030 [2]. Thus, considering this exponential growth of the Internet connected devices, which sometimes are weak computing systems (without dedicated to security software and hardware), various issues arise that are problems needing to be addressed, mainly on major security concerns for the designers and manufacturers of IoT devices [3–6].

These concerns are due to the fact that IoT devices do not exploit (and is not needed) the full TCP/IP stack as embedded in Operating Systems (OS) for general purpose computing systems. However,
the common practice for the manufacturers is either the adoption of the full TCP/IP stack (in order to minimize time-to-market, leaving this way backholes and known vulnerabilities unresolved), or the development of a custom implementation of a communication protocol, followed by the resulting compatibility problems. Since the communication framework is not limited to the targeted architecture, the computing and the interconnection capabilities, which are important issues to deal with, such as Information and Communication Security, come into focus. Thus, many low-security IoT devices with low processing power become an easy and attractive target for malicious attacks by hackers, which are used after a success attack to create large botnets. The botnet is a network of bots or otherwise infected machines that are used, under the hacker control, for various malicious activities, such as Distributed Denial of Service (DDoS) attacks.

DDoS is one of the most popular and widespread cyber-attacks. DDoS attacks, such as the one in September 2016 (Mirai), demonstrate the high vulnerability of IoT systems and devices and the magnitude of the risk [7]. In addition, another category of attacks that simply ignores the mathematical properties of a cryptographic system and focuses on its physical application to the hardware is that of side-channel attacks, commonly referred to as SCAs. In particular, cryptographic systems usually leak information about the internal computing process. In practice, this means that attackers can exploit various techniques to extract basic and other secret information from the [8] device. Essentially, side-channel attacks monitor power consumption and electromagnetic emissions while a device performs cryptographic operations. Side-channel attacks against electronic devices and systems are relatively simple and inexpensive to execute. This means that attackers can exploit various side-channel techniques to collect data and extract hidden cryptographic keys. SCAs are used mainly in cryptography and are attacks based on information obtained from the physical analysis of a cryptosystem, without a Brute Force technique or theoretical weakness of comparing (cryptanalysis) algorithms.

This article is an extension of the work [9–12] in conjunction with [13] and presents the exploitation of the SCA approach [14] for security reasons (white hacking). Initially, it applies SCA for monitoring the power dissipation of a targeted IoT device, capturing its electrical behavior and transforming it to data. Then, applying k-Means Clustering Machine Learning (ML) Algorithm [15] for data clustering, the proposed solution provides a range of real-time power analysis capabilities and combines multiple device profiles and scalable endpoints to detect suspicious behavior of Internet-connected devices. The above advantages of the techniques used in this work, combined with the exploitation of the physical characteristics of IoT devices, introduce a new method for intrusion detection to an IoT device. Innovation is found in the fact that an external circuit is capable to detect intrusion attempts without prior knowledge of the under surveillance IoT device or its functionality. Furthermore, it is a low-cost, small-sized and computationally fair solution to enhance security to IoT devices found at households, which are vulnerable to attacks due to the lack of a professional Intrusion Detection System (IDS) in a house or security mechanisms to embedded the IoT device.

The contribution of this work is summarized in the following:

1. The first implementation of a low-cost, small sized embedded system, to monitor externally agnostic IoT devices, with the ability to learn in the field of operation.
2. Enhancement of security for IoT devices against DDoS and similar attacks, without the need of a cloud-based IDS.
3. It is mimicking biometrics principles allowing the extension of data collected from external IDS (when connected), similar to that of Condition Monitoring applied to industry.
4. It is not based on network rules and/or patterns of virus, offering a most robust confrontation to attacks of unknown nature.
5. No similar work in the field of embedded systems to the IoT edge to compare with; this is the baseline circuit.

In the rest of this paper, following the introduction, the related work is explored in Section 2. Section 3 presents the proposed IoT device intrusion detection approach. In Section 4, various test
scenarios carried out using the proposed device are described and results from the practical application of experimentation are offered. Finally, in Section 5, conclusions are offered.

2. Related Work

This section explores research findings related to IoT devices security, focusing on the availability of a device. Many studies have shown that IoT smart devices are accessible publicly through the Internet and security is often a reflection of the framework/architecture that was used for the development of many widespread IoT devices. In [16], the basic features of malicious Internet programs that organize and coordinate DDoS attacks are analyzed, while, in [17], other types of attacks that can be applied on IoT devices are presented, specifically: Node Jamming, Physical Damage, Node Tampering, Social Engineering, Malicious Node Injection, Sleep Deprivation Attack, Malicious Code Injection on the Node, Network Attacks: Traffic Analysis Attacks, RFID Spoofing, RFID Cloning, RFID Unauthorized Access, Man In The Middle Attack, Denial of Service, Sinkhole Attack, Routing Information Attacks, Sybil Attack, Software Attacks Virus and Worms, Malicious Scripts, Spyware and Adware, Trojan Horse, Denial of Service, or even Encryption Attacks: Man In The Middle Attack, Side Channel Attacks, and Cryptanalysis Attacks.

Recent research works, such as [18], also revealed cases of Trojans found at a hardware level (Trojan hardware) that are malicious components or even command sequences, which enable hackers to bypass security systems. In [19], a tool to analyze security aspects of distributed IoT programs and thus protect them against buffer overflow attacks is presented. The work in [20] proposes a multi-platform monitoring operation of a system towards anomaly detection that supports groups of heterogeneous devices. Another research work [21] presented the overview of a distributed internal anomaly detection system for IoT, where each node monitors its neighbors; in the case abnormal behavior is detected, the node will block packets from the node, which behaves abnormally in the data link layer and reports to its parent node. Due to the heterogeneity of IoT smart devices, the existing techniques for detecting suspicious behavior or attacks may be considered not effective, or cannot be implemented with security properties, given the enormous variety of such devices [22]. As it is found from the previously mentioned works, most of the works are concerned for the network-wise layers and intrusion or anomaly detection is performed by a system monitoring the network either locally or from a cloud-based platform. However, this has proved to be the main vulnerability of the IoT devices, since a successful attack will result in spread through the network or collapse of the communication, leaving the IoT devices unprotected.

The aforementioned has motivated researchers to examine new approaches for detecting, counter-measuring and shielding smart devices from attacks. In particular, it is a common approach to detect suspicious behavior through various measurable parameters that form the conditions of operation of the device, e.g., power dissipation, ambient temperature, etc. In [23], DDoS attacks are encountered through ML techniques, e.g., for controlling packet transmission rate, packet size, etc. In [24], a method for detecting anomalous operations of house IoT devices is presented, which can learn sequences of user behaviors according to conditions such as time of day, temperature, and humidity. When an operation command arrives, the method compares the current sequence with learned sequences for the current condition. If the sequences do not match, the operation is considered as abnormal. Another approach that uses ML-based feature-group clustering techniques, nodding, and parameter-use for proper education system is found in [25]. In [26], two approaches were proposed that include deep automated encoder models for analyzing time series collected by gravitational wave detectors and provide a classification tag (noise or real signal). The work in [27] focuses on the detection of unexpected sensor data resulting either from the sensor system itself or from the environment under control. A new approach is proposed for automatic detection of anomalies in heterogeneous sensor networks based on cutting-edge data analysis with cloud data analysis. The former exploits an artificial neural network algorithm without supervision, while the cloud data analysis exploits the multi-parameter processing distance algorithm. The research results in [28] represent an attempt
to investigate anomalies in a Multiple IoT scenario (MIoT). First, it suggests a new methodological framework that can make future investigations in this research field easier, coherent, and uniform. Then, in the context of anomaly detection in an MIoT, it defines the so-called “forward problem” and “inverse problem”. The aim of the work in [29] is to investigate the suitability of deep learning approaches for anomaly-based intrusion detection system. In this work, the developed anomaly detection models are based on different deep neural network structures, including convolutional neural networks, autoencoders, and recurrent neural networks.

The previously presented works are still focusing on network solutions and are based on the availability of high computation power of the system that embeds the ML and other mechanisms to identify anomaly and/or intrusion detection. This means that the proposed solutions are expensive, requiring special purpose hardware for their implementation, which is not by any means low-cost. The urge to find a low-cost and not computationally intensive solution is not satisfied, which is essential for consumer products targeting house use.

Finally, the works in [9–12] by Myridakis et al. study physical features, such as uninterrupted power supply and extract information that, in combination with thresholds or ranges of values, detects irregularities in IoT devices. These are the first attempts to implement an autonomous system capable of providing enhanced security at low-cost. However, they lack the integration of ML algorithms and a robust way to detect intrusion, without knowing the targeted IoT device operation.

3. Materials and Methods

Integrated circuits are made of a plethora of transistors, which act as voltage controlled switches. Current flows along the transistor substrate when load is applied to (or removed from) the gate. This current charges the gates of other transistors and interconnects pins and other circuit loads. Any source of additional electrical load consumes energy and generates electromagnetic noise, both of which are externally detectable. The development of the Side-Channel Monitoring Device system, based on the well-known SCA technique, focuses on monitoring the power supply, extracting useful information for detecting suspicious behavior on Internet-connected devices.

The hypothesis made in this work is that any electrical or electronic device has its own physical characteristics, which under the expected operation is consuming a predetermined amount of power. Thus, when an attack or an abnormal behavior occurs, then a different profile of operation is observed that affects power dissipation (excessive use of communication and processing resources).

This article introduces an innovative digital device in the field of IoT security that is easily adapted to any IoT device as an external mechanism or otherwise as a "smart shell", hereinafter referred as SmartShell. Its operation is based on the SCA attack technique, as mentioned above, and monitors the power supply by analyzing the behavior of the device. Exploiting the k-Means Clustering Algorithm that is embedded in its code, it can be trained to detect suspicious behavior, targeting intrusion attempts. Another advantage of this device is its interoperability, since it can be applied and trained by any IoT device characteristics operation, since it is connected in-between the device’s power supply and the device itself.

Moreover, it operates without the support of a server of high computational power, offering autonomy of operation even when the DDoS is successful.

The workflow of the proposed system is depicted in Figure 1. Initially, the input is sampled and then, using filters for smoothing spikes and sharpen waveform, the signal is available for use by the intrusion detection mechanism. The SmartShell has two operation modes, namely ‘Training’ and ‘Monitoring’. In the case of the first connection of the SmartShell to the targeted IoT device, the Training Mode is performed for a long period. More details for the training Mode are offered in a following subsection. In this mode, the SmartShell monitors normal operation and the IoT device’s modes of operation and creates the clusters of normal operation. Thus, afterwards, during the Monitoring Mode of operation, the SmartShell samples the supply current and checks if the input is fitting one of the identified clusters. If it is not, then an intrusion detection signal is produced,
triggering a fail safe operation (i.e., shut-down). In the case the input is belonging to one of the clusters of normal operation, then the SmartShell repeats the monitoring cycle.

In Figure 2, smart devices are connected to the power grid (power supply) via the proposed SmartShell device. The topology shows the scalability of the proposed solution, allowing a theoretically infinite number of IoT devices to be protected. The IoT devices are connected to the Internet via the ISP router installed at home. The botnets attack to this network of devices, activating sophisticated attacks, which are usually not detected due to the absence of special security hardware installed at home (e.g., hardware firewalls, IDS). The presence of the SmartShells allows autonomous operation preventing the spread of a bot in the home installed devices. A strong characteristic of this solution is that each SmartShell is trained based on the functionality of the attached IoT device, offering a sophisticated monitoring mechanism.

This approach allows autonomous operation of the SmartShell for each IoT device, although an extension of its functionality may be achieved via a house gateway that broadcasts current measurements to a cloud based IDS. This would increase further security, if the home owner has the sources to use such a service. A notice that should be made is that the SmartShell is capable of communicating with the gateway via a different communication channel than that of the IoT devices (e.g., Bluetooth) only in transmitting mode, making it unavailable to attacks to itself. Finally, a collateral benefit from the broadcast of the current measures to a cloud-based IDS would significantly increase awareness of unknown types of attacks in the future.
The SmartShell is based on a low-cost low-power microcontroller (ATM 2560) that continuously monitors current supply to the device. To achieve this, a current supply sensor is developed, as depicted in Figure 3. Specifically the layout of the electric circuit is detailed as follows:

- The monitoring circuit includes a 1 Ohm resistor and a smaller calibrating resistor used for accuracy reasons. The resistor is located between the two inputs of the micro-controller in order to measure the amperage. The power amperage can be calculated through the measurement of the voltage at the two input points according to the following formula:

\[ I = \frac{V_2 - V_1}{R_1} \]  

where \( V_1 \) and \( V_2 \) are the two reference voltages as depicted in Figure 3, and \( R \) is the 1 Ohm resistor.

- Two analog inputs of the micro-controller are connected to the circuit (Figure 3) in order to collect the power measurements. The first input is connected to the point before the resistor while the second is used to measures the voltage at the other end of the resistor.

- The device to be monitored is connected serially to the resistor. The circuit is completed by connecting a Direct Current (DC) power supply 5 V for the power jack.

- The captured data related to electrical current supply are stored in a buffer (memory of the micro-controller).

- Finally, based on the captured data, the training algorithm creates profiles of different operation modes. After training, any deviation from them enables the SmartShell to detect abnormal operation.
Although the hardware implementation is easy and straightforward, further actions were needed. Improving the signal-to-noise ratio (SNR or S/N) with a software technique was a necessary action, exploiting the programming capabilities of the SmartShell’s micro-controller. Thus, a Moving Window algorithm was used to normalize signal deviations. In this way, spikes can be eliminated when they occur infrequently, while a more frequent appearance of spikes is maintained for abnormal detection. This signal smoothing technique is called the moving average. From the raw data sequence \([y_1, y_2, \ldots, y_N]\), we created a corresponding smoothed data sequence. The smoothed point \((y_k)_s\) is the mean of an odd number \(2n + 1\) \((n = 1, 2, 3, \ldots)\) of the raw data sequences \(y_{k-n}, y_{k-n+1}, \ldots, y_{k-1}, y_k, y_{k+1}, \ldots, y_{k+n-1}, y_{k+n}\), i.e.:

\[
(y_k)_s = \sum_{i=-n}^{i=n} y_{k+i} / (2n + 1) \tag{2}
\]

The odd number \(2n + 1\) is the window width. The larger the window width, the more intense the smoothing. The SNR can be further enhanced by increasing the window width or by multiple window passes (smoothing at already smoothed points). During average moving window processing, a spike calculation is also conducted, comparing value \(y_k\) to the thresholds \(y_{\text{thres}, \text{max}}\) and \(y_{\text{thres}, \text{min}}\). Let us assume that the spike is positive, that is, the signal is ascending; then, \(sp_k\) is set to 1 only in case \(y_k\) is greater than \(y_{\text{thres}, \text{max}}\). The \(sp_k\) is also set to 1 in case the spike is negative, that is, the signal is descending and the \(y_k\) is less than \(y_{\text{thres}, \text{min}}\). Then, the final population of spikes is calculated in a time window including an odd number of samples, e.g., \(2m + 1\), where \(m > n\).

\[
sp_k = \begin{cases} 
1 & \text{if } y_k > y_{\text{thres}, \text{max}} \text{ and } y_k - y_{k-1} > 0, \\
1 & \text{if } y_k < y_{\text{thres}, \text{min}} \text{ and } y_k - y_{k-1} < 0, \\
0 & \text{if otherwise}
\end{cases} \tag{3}
\]

where the \(k\)-th sample has a value of \(y_k\), which is compared to the appropriate threshold \(y_{\text{thres}, \text{max}}\) or \(y_{\text{thres}, \text{min}}\), respectively, to its previous value. The calculation is performed for \(2m + 1\) samples \(y_{k-m}, y_{k-m+1}, \ldots, y_{k-1}, y_k, y_{k+1}, \ldots, y_{k+m-1}, y_{k+m}\), where \(m > n\). Then, a rough estimation of the identified spikes in the \(2m + 1\) consecutive samples is performed with the following equation:

\[
\text{spikes} = \sum_{i=-m}^{i=m} sp_{k+i} \tag{4}
\]

The selection of \(m, n, y_{\text{thres}, \text{max}},\) and \(y_{\text{thres}, \text{min}}\) in this work was considered as information given by the manufacturer of the IoT device. The typical value for \(m\) is 5000 and for \(n\), it is 20. The spikes that set an alarm were selected to be 50 in order to avoid false positives due to random spikes originating from the power grid.

### 3.2. Implementation of the Training Algorithm

The K-Means Clustering algorithm was selected due to its simplicity and the ability to be fast and efficient, even when running on small processors with low capabilities. Furthermore, the small memory footprint allows its embedding to low-cost and low-power micro-controllers, as the one of SmartShell. This resulted in stable training of the system through the creation of different clusters and low user interaction.

The training algorithm (K-Means Clustering Algorithm) has two inputs:

- The training set (cluster initialization) contains the training data of the IoT device current consumption, recorded as long as the user sets SmartShell on training mode.
- The \(k\) value, where \(k\) is the number of clusters that the algorithm is going to create. This value is user defined and represents the different modes of the IoT device, for which the device has differential current consumption.
For example, if an IoT device has 2 modes (stand by and broadcasting), the k should be 2.

There are six steps for the k-Means Clustering algorithm, which are as follows:

1. Sorts the training set to ascending order.
2. Sets randomly k centroids in the training set.
3. Creates clusters of the data which are closest to centroids.
4. Calculates the mean value of each cluster and moves centroids there.
5. Repeats steps 3 and 4
6. Finishes when the previous mean values are the same as the last ones.

Since centroids are setting randomly in the beginning of the algorithm, it is possible for the algorithm to not create the best selected clusters. To eliminate this possibility, the algorithm has to be executed repeatedly.

(steps 1–6) several hundred times. Every time after step 6, the system compares the total variation of the clusters that has been produced with the previous minimum total variation that has been found and stores the clusters with the smallest one.

Another feature that has been added in the k-Means Clustering Algorithm is to set centroids in the training set manually for the first time, one at the beginning, one at the end, and the rest of them at positions with step (s) after the first centroid s = (tsLast − tsFirst) / k − 1 since the data set is one-dimensional (where tsLast and tsFirst are the last and the first values of the training set).

For example, if we have a training set in mA:

\[
\text{ts} = [3, 4, 5, 7, 9, 11, 42, 43, 44, 49, 52, 55, 58, 94, 95, 96, 99, 100]
\]

and k = 3,

the centroids for the first time are going to be set to the following positions:

\[
\text{ts} = [13, 4, 5, 7, 9, 11, 42, 43, 44, 49, 52, 55, 58, 94, 95, 96, 99, 100].
\]

which, most of the time, creates clusters with the smallest total variation, but, even if it fails, it is going to be executed by randomly setting centroids several hundred times. The output of a k-Means Clustering algorithm are the created k Clusters and the system stores every min and max of each one, which are the thresholds that are used by the Intrusion Detection algorithm.

3.3. Implementation of the Detection Algorithm

The Intrusion Detection algorithm was selected to be simple, since this work is aiming to serve as the baseline for autonomous systems embedding ML capabilities and are low-cost and small sized. This requirement imposes significant limitations in both memory capacity and computational power.

Since the input signals have been pre-processed in order to remove unnecessary spikes and smoothing of the waveform, it is expected to observe consecutive spikes and areas of high activity when an intrusion is performed. Thus, a moving window on the time series is counting how many times such activity was measured. This is performed by comparing each input to the minimum and maximum values of each cluster. In a given short time period, the appearance of 5 out of range values is considered suspicious and a warning is triggered. Each warning has a Time-to-Live (TTL) after which it is removed. The presence of three live consecutive warnings in a given time period triggers an intrusion detection, and the SmartShell activates a security procedure.

Although there is a variety of actions to be performed after an intrusion detection, since the aim of this work is to prevent the formation of botnets, the default security procedure includes disabling the targeted IoT device. Since there was an intrusion detection, and the SmartShell operates autonomously, without knowledge of the network of IoT devices, the default procedure was selected to be the one followed in a pandemic. As already mentioned, this work is bio-inspired mimicking the detection of an unexpected operation, as well as the activation of the basic protocol for such situations, which is lockdown. In this way, the infected IoT device will not be able to contaminate any other, since it is removed from the network. Although this procedure raises questions about the control that is gained by the SmartShell, in general, it is the same approach followed by sophisticated IDS in an industrial environment.
4. Results

In this section, the proposed solution is tested. The following installation will be considered for performing the intrusion detection tests.

1. A custom surveillance digital IP camera serves as a ‘Target IoT device’ properly programmed to broadcast the captured video over the Internet (connected to WiFi).
2. The SmartShell is inserted online that is in-between the power supply and the under protection IoT device.

The custom IP surveillance camera consists of a Raspberry Pi 3 B + micro-controller and an RPI 8MP camera board version 2, which is the IoT device on which the attacks were targeted. The attacks were carried out external devices, in this case a mobile phone, which is available to anyone and may be used easily for DoS attacks when someone is close to the home network. The attacks, carried out on the target device, were performed from a mobile phone, through the Termux application. The technique of the attack was DoS Attack and was carried out with the Hummer tool, on the IP of the target device and more specifically on the port 8554. A number of packets flood the communication channel to successfully perform the DoS attack.

For experimentation reasons, the user can modify and change the boot time of the device as well as its training time and the numbers of clusters that are going to create. In the future, this is expected to be dynamically set by the training algorithm. In the present experiment, the times of the above situations were 5 min for the “Boot Device” mode, 1920 min (32 h) for the “Training” mode and $k = 3$. The rate of the power supply sample of the SmartShell detection device was 100 ms and this was for real-time information about the attack.

Figure 4 shows an example of the targeted IoT device (web camera) current consumption on different modes (Boot, Stand By, Broadcasting, Movement). While an IoT device is booting, the current consumption is not stable, and it should not have been included in the training or an intrusion detection algorithm. The training or the intrusion detection mode can start when the boot sequence has finished. While the system is in training mode, it creates different data clusters for the modes of operation of the IoT device, as illustrated in Figure 4 Stand By (24–38 mA), Broadcasting (58–72 mA), and Movement (102–116 mA).

Figure 4. Web camera current consumption and creation of clusters example.

Those clusters are used by the intrusion detection algorithm while checking in real time the current consumption of the IoT device. If the current consumption value of the IoT device is not in between the lower and upper thresholds of the created clusters, the system detects the malfunction or an attack to the IoT device.
It should be stressed out that it is expected to get different results for identical devices installed in locations with different conditions (e.g., indoor or outdoor), or even to the same IoT device when installed in different locations of varied power quality. This explains the necessity of qualitative analysis of the input, in contrast to prior quantitative analysis performed by [9–11], which is prone to the characteristics of the power grid in the area.

In Figure 5, the measurements of the supply current during the experiment (32 h) are shown, without the use of any filter (hardware or software). As a result, there is a lot of noise in the signal (spikes) that affects the detection of the attack using thresholds, as it may be observed within the yellow frame.

![Figure 5. Measurements without using a software filter.](image)

However, after the application of the first software filter, as observed in Figure 6, the values obtained are noise-free and the signal’s form is now suitable for the ML algorithm to be trained. The filter removed random spikes which were of no value to the analysis of the supply current.

![Figure 6. Measurements by applying the first software filter.](image)

Finally, to eliminate any possibility of an appearance of a false positive indication of an attack, a second software filter was applied for further smoothing of the signal. The results of the measurements are presented in Figure 7, where the detection of the attack can be seen within the yellow frame.

The operation of the SmartShell device had four operating modes which are analyzed below, as derived by the ML algorithm. The red line in the diagrams shows the current intensity, while the blue and yellow line the values from the first and second software filter, respectively, which were mentioned above. In addition, in the upper left part of the images, the current mode of the SmartShell device is indicated. Each IoT device starts using multiple resources, resulting in greater current intensity in its operation. Thus, an initial idle time was selected to start the IP camera (5 min), without starting its training. The above mode is called “Boot Device” and can be seen in Figures 8 and 9. It was noticed that the camera from one point onwards, begins to reach a normal behavior (see Figure 9, after the green frame), as the values converge.
Then, the device, after the initial idle time, starts the training, and this is reflected in Figure 10 within the green frame. During the training, the functionality and the conditions of the targeted device (e.g., IP camera) are studied. For example, taking a still image, taking a moving image, taking a picture during the day or at night, or waiting and more. Finally, it creates the clusters corresponding to the operational modes of the targeted IoT device. For reasons of presentation, the clusters are set
to two by the user for this experiment. This condition is called “Training Started” and is shown in Figures 10 and 11.

After the device has completed its training cycle, it enters a permanent intrusion detection mode, during which the device detects any attacks made on the camera. This condition is called “ID Mode Started” and is shown in Figures 12 and 13.
Finally, in Figure 14 and within the green frame, the detection of the attack is detected as depicted at the characteristics in a very short time (real-time detection) by the device, since the rate of sampling as mentioned above is 100 ms. During the detection of the attack, the system updates by displaying a message “Intrusion Detected Started” as seen in Figures 14 and 15 top left. The current mode of operation of the device is called “Intrusion Detected Started”.

In Figures 16 and 17, the immediate response of the device’s system to its reset in “ID Mode Started”, after the attack, is shown. This is easily discernible, as in the green frame, the detection of the normal behavior of the device is depicted. At the same time, the message with the current situation is presented in the upper left part of the figures.

Summarizing the operation of the SmartShell detection device, four operating modes were observed, as follows:

1. The “Boot Device” mode, in which the Smart Shell waits the IP camera to boot, without starting training or intrusion detection.
2. The “Training Started” mode, in which the Smart Shell was trained in the functions of the camera (e.g., continuous flow of movement, stand by, broadcasting, etc.) and creates the different clusters of those functions which are going to used by the Intrusion Detection algorithm.
3. The “ID Mode Started” mode, in which the device, even after completing the training, detects any attacks that would be made on the camera, based on the previous training.
4. In addition, finally, the “Intrusion Detected Started” mode, in which the attacks were now detected, and the system informed by presenting a message.

The proposed solution is autonomous and activates security protocols after the intrusion detection. Since this is the first device of its kind (embedding an ML algorithm), it is essential to perform comparisons to similar solutions, although they do not embed an ML algorithm for the detection of the intrusion. Furthermore, although there are no datasets for this purpose, in Table 1, the proposed work is compared to the only similar implementations during three different types of attacks; DoS attack as described before, a mirai attack and a zero-day attack. In parentheses, the percentage of false positive detection is provided. The measurements for the outdoor installations were conducted during summer, since there are no data at the moment available for the rest of the seasons.

In order to highlight the extra advantages of the proposed solution, installations of the SmartShell to various environments was further examined. For the competitive implementations, we considered pre-configuration, since there is no training feature.
Figure 16. Intrusion Detection Mode after Detected 1.

Figure 17. Intrusion Detection Mode after Detected 2.

Table 1. Comparison of works on intrusion detection by autonomous embedded systems. Numbers correspond to the percentage of true positive intrusion detection and, in parentheses, the corresponding percentage of false positive detection.

| Work       | DoS (in) | Mirai (in) | Zero-Day (in) | DoS (out) | Mirai (out) | Zero-Day (out) |
|------------|----------|------------|---------------|-----------|-------------|----------------|
| [9]        | 95 (2)   | 100 (2)    | 100 (2)       | 81 (22)   | 100 (15)    | 69 (40)        |
| [10]       | 96 (2)   | 100 (2)    | 100 (1)       | 83 (17)   | 100 (15)    | 70 (22)        |
| [11]       | 96 (2)   | 100 (1)    | 100 (1)       | 84 (15)   | 100 (13)    | 72 (20)        |
| [12]       | 98 (2)   | 100 (1)    | 100 (1)       | 86 (12)   | 100 (11)    | 75 (20)        |
| This work  | 100 (0)  | 100 (0)    | 0 (100)       | 100 (1)   | 100 (2)     | 0 (100)        |
| Mod. work  | 100 (0)  | 100 (0)    | 100 (1)       | 100 (1)   | 100 (2)     | 72 (20)        |

As it may be observed in Table 1, the proposed solution presented the best score in comparison to the competitive ones (best performing shown in bold). The main disadvantage of the proposed system was detected in a zero-day attack, in which the boot sequence is part of the attack and, during this time period, the system remains idle, since it has no pre-defined minimum and maximum values for the clusters. After applying pre-defined values similar to those described in [11], the modified version of the proposed work presents sufficient intrusion detection. All works presented a high score
in a mirai attack, since it causes high activity; in the DoS attack, the scores were fair, since activity may be controlled; in a zero-day attack, the modified proposed work has fair results. As expected, the experimental results that were based on measurements on different installations showed that the modeling of power dissipation in a controlled environment (e.g., indoor) is near the theoretic model (e.g., laboratory conditions), in contrast to the results in outdoor conditions, which have significant deviations due to the conditions (e.g., temperature) [30] and the supply power quality during a season [31].

5. Conclusions

In this article, an autonomous current monitoring system that exploits the technique of SCA was presented that may be used to detect suspicious behavior, to protect smart devices intuitively, and also incorporate an ML algorithm for Intrusion Detection. The system exploits current supply characteristic dissipation for the optimal result. The proposed system is novel and uses the k-Means Clustering algorithm with unsupervised training. The results of this work showed successful detection, imaging, and reporting of attacks on smart IoT devices in real time.

The results depicted the robust intrusion detection performed by the proposed solution in contrast to competitive autonomous low-cost and small-sized implementations. A disadvantage of the proposed solution is the initial idle status during boot sequence, which may be used for attacking the under protection IoT device.

Since this is the first device of its kind and, due to the lack of benchmarks for such purposes, it is in our intention to use this work as a baseline circuit to create datasets for future works that want to be compared with our solution. Thus, from a technical point of view, there are no datasets to demonstrate its effectiveness at the moment. Furthermore, there is significant potential for future work in areas, such as security and privacy, as well as the addition and combination of additional physical features of the devices, or even improving the device itself with the automated detection of k value.

Author Contributions: Conceptualization, A.K.; methodology, D.M. and S.P.; software, D.M. and S.P.; validation, D.M. and S.P. and K.K.; formal analysis, A.K.; investigation, D.M. and S.P.; resources, A.K. and D.M.; data curation, D.M. and S.P.; writing—original draft preparation, A.K., D.M. and S.P.; writing—review and editing, A.K., D.M., S.P. and K.K.; visualization, A.K. and D.M.; supervision, A.K.; project administration, A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We would like to thank Dimitrios Schinianakis and Georgios Spathoulas for the fruitful discussions on how the system may become efficient.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

- **DC**: Direct Current
- **DDoS**: Distributed Denial of Service
- **IoT**: Internet of Things
- **IP**: Internet Protocol
- **ML**: Machine Learning
- **OS**: Operating System
- **RFID**: Radio-Frequency Identification
- **SCA**: Side-Channel Attack
- **SNR**: Signal-to-Noise Ratio
- **TTL**: Time-to-Live
References

1. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* 2010, 54, 2787–2805. [CrossRef]
2. Griffiths, F.; Ooi, M. The fourth industrial revolution-Industry 4.0 and IoT [Trends in Future I&M]. *IEEE Instrum. Meas. Mag.* 2018, 21, 29–43.
3. Luong, N.C.; Hoang, D.T.; Wang, P.; Niyato, D.; Kim, D.I.; Han, Z. Data collection and wireless communication in internet of things (iot) using economic analysis and pricing models: A survey. *IEEE Commun. Surv. Tutor.* 2016, 18, 2546–2590. [CrossRef]
4. Li, S.; Tryfonas, T.; Li, H. The Internet of Things: A security point of view. *Internet Res.* 2016, 26, 337–359. [CrossRef]
5. DiMase, D.; Collier, Z.A.; Carlson, J.; Gray, R.B.; Linkov, I. Traceability and risk analysis strategies for addressing counterfeit electronics in supply chains for complex systems. *Risk Anal.* 2016, 36, 1834–1843. [CrossRef]
6. Petit, J.; Shladover, S.E. Potential cyberattacks on automated vehicles. *IEEE Trans. Intell. Transp. Syst.* 2015, 16, 546–556. [CrossRef]
7. Zhu, Z.; Lu, G.; Chen, Y.; Fu, Z.J.; Roberts, P.; Han, K. Botnet research survey. In Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference, Turku, Finland, 28 July–1 August 2008; pp. 967–972.
8. Zhou, Y.; Feng, D. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. *IACR Cryptol. EPrint Arch.* 2005, 2005, 388.
9. Myridakis, D.; Spathoulas, G.; Kakarountas, A. Supply Current Monitoring for Anomaly Detection on IoT Devices. In Proceedings of the 21st Pan-Hellenic Conference on Informatics, Larissa, Greece, 28–30 September 2017; pp. 1–2.
10. Myridakis, D.; Spathoulas, G.; Kakarountas, A.; Schoinianakis, D.; Lueken, J. Anomaly detection in IoT devices via monitoring of supply current. In Proceedings of the 2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin), Berlin, Germany, 2–5 September 2018; pp. 1–4.
11. Myridakis, D.; Spathoulas, G.; Kakarountas, A.; Schinianakis, D.; Lueken, J. Monitoring Supply Current Thresholds for Smart Device’s Security Enhancement. In Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), Santorini Island, Greece, 29–31 May 2019; pp. 224–227.
12. Myridakis, D.; Spathoulas, G.; Kakarountas, A.; Schinianakis, D. Smart Devices Security Enhancement via Power Supply Monitoring. *Future Internet* 2020, 12, 48. [CrossRef]
13. Papafotikas, S.; Kakarountas, A. A Machine-Learning Clustering Approach for Intrusion Detection to IoT Devices. In Proceedings of the 2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSTM), Piraeus, Greece, 20–22 September 2019; pp. 1–6.
14. Liu, Y.; Wei, L.; Zhou, Z.; Zhang, K.; Xu, W.; Xu, Q. On code execution tracking via power side-channel. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1019–1031.
15. Jain, A.K. Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* 2010, 31, 651–666. [CrossRef]
16. Angrishi, K. Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets. *arXiv* 2017, arXiv:1702.03681.
17. Sopori, D.; Pawar, T.; Patil, M.; Ravindran, R. Internet of things: Security threats. *Int. J. Adv. Res. Comput. Eng. Technol.* 2017, 6, 263–267.
18. Yang, K.; Hicks, M.; Dong, Q.; Austin, T.; Sylvester, D. A2: Analog malicious hardware. In Proceedings of the 2016 IEEE symposium on security and privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 18–37.
19. Teixeira, F.A.; Pereira, F.M.; Wong, H.C.; Nogueira, J.M.; Oliveira, L.B. SIoT: Securing Internet of Things through distributed systems analysis. *Future Gener. Comput. Syst.* 2019, 92, 1172–1186. [CrossRef]
20. Stiawan, D.; Idris, M.Y.; Malik, R.F.; Nurmaini, S.; Budiarto, R. Anomaly detection and monitoring in Internet of Things communication. In Proceedings of the 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, Indonesia, 5–6 October 2016; pp. 1–4.
21. Thanigaivelan, N.K.; Nigussie, E.; Kanth, R.K.; Virtanen, S.; Isoaho, J. Distributed internal anomaly detection system for Internet-of-Things. In Proceedings of the 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2016; pp. 319–320.

22. Nguyen, T.D.; Marchal, S.; Miettinen, M.; Fereidooni, H.; Asokan, N.; Sadeghi, A.R. DIoT: A federated self-learning anomaly detection system for IoT. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 756–767.

23. Doshi, R.; Apthorpe, N.; Feamster, N. Machine learning ddos detection for consumer internet of things devices. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24 May 2018; pp. 29–35.

24. Yamauchi, M.; Ohsita, Y.; Murata, M.; Ueda, K.; Kato, Y. Anomaly Detection in Smart Home Operation From User Behaviors and Home Conditions. *IEEE Trans. Consum. Electron.* 2020, 66, 183–192. [CrossRef]

25. Muller, S.; Lancrenon, J.; Harpes, C.; Le Traon, Y.; Gombault, S.; Bonnin, J.M. A training-resistant anomaly detection system. *Comput. Secur.* 2018, 76, 1–11. [CrossRef]

26. Corizzo, R.; Ceci, M.; Zdravevski, E.; Japkowicz, N. Scalable auto-encoders for gravitational waves detection from time series data. *Expert Syst. Appl.* 2020, 151, 113378. [CrossRef]

27. Cauteruccio, F.; Fortino, G.; Guerrieri, A.; Liotta, A.; Mocanu, D.C.; Perra, C.; Terracina, G.; Vega, M.T. Short-long term anomaly detection in wireless sensor networks based on machine learning and multi-parameterized edit distance. *Inf. Fusion* 2019, 52, 13–30. [CrossRef]

28. Cauteruccio, F.; Cinelli, L.; Corradini, E.; Terracina, G.; Ursino, D.; Virgili, L.; Savaglio, C.; Liotta, A.; Fortino, G. A framework for anomaly detection and classification in Multiple IoT scenarios. *Future Gener. Comput. Syst.* 2020, 114, 322–335. [CrossRef]

29. Naseer, S.; Saleem, Y.; Khalid, S.; Bashir, M.K.; Han, J.; Iqbal, M.M.; Han, K. Enhanced network anomaly detection based on deep neural networks. *IEEE Access* 2018, 6, 48231–48246. [CrossRef]

30. Malewski, M.; Cowell, D.M.; Freear, S. Review of battery powered embedded systems design for mission-critical low-power applications. *Int. J. Electron.* 2018, 105, 893–909. [CrossRef]

31. Lezhnyuk, P.; Komar, V.; Kravchuk, S.; Sobchuk, D. Mathematical modeling of operation quality of electric grid with renewable sources of electric energy. In Proceedings of the 2017 International Conference on Modern Electrical and Energy Systems (MEES), Kremenchuk, Ukraine, 15–17 November 2017; pp. 324–327.

**Publisher’s Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).