A New Defensive Technique Against Sleep Deprivation Attacks Driven by Battery Usage

Jean Luc Antoine Olivier Fobe[1], **Michele Nogueira**[2], **Daniel Macêdo Batista**[1]

[1]Department of Computer Science, University of São Paulo, Brazil

[2]Department of Computer Science, Federal University of Minas Gerais, Brazil

jean.fobe@usp.br, michele@dcc.ufmg.br, batista@ime.usp.br

Abstract. *A significant amount of IoT devices are essentially powered by batteries and implements mechanisms to save energy, such as the sleep mode. The decision-making process deployed in IoT devices to enter to and exit from sleep mode can be exploited by remote users through sleep deprivation attacks, reducing the battery's lifetime and causing a denial of service. This paper presents a new defensive technique to mitigate and prevent sleep deprivation attacks. It is based on the local battery consumption data, that is an input to control the sleep mode. Performance evaluation carried out in a system based on an ESP32 showed that the technique could increase the battery's lifetime by 51.2% in a scenario under a sleep deprivation attack.*

1. Introduction

The Internet of Things (IoT) comprises heterogeneous devices (e.g., movement sensors, baby monitors, cameras, smartwatches, smartphones) interconnected through different communication technologies. Estimates show that there will be over 10 billion connected IoT devices by the end of 2021 and this number tends to increase in the following years [Statista 2022]. A significant part of these devices is powered essentially by batteries. The IoT battery market points out a growth from USD 9.2 billion in 2020 to USD 15.9 billion in 2025 [Markets and Markets 2020], attracting the attention to design efficient and optimized solutions for using batteries in IoT devices.

IoT devices are prone to attacks [Lima et al. 2009, Rodriguez and Batista 2020]. These devices usually run software developed by third-parties and they are connected to the Internet, making them easy targets due their limitations in design, implementation and resources, such as batteries. A well-known attack against IoT devices lies in sleep deprivation [Pirretti et al. 2006, Nguyen et al. 2019]. An attacker creates means to generate an unwanted increase in battery consumption and prevent the device to switch to the sleep mode. This counters the device to save energy, reducing its active lifespan.

Sleep deprivation attacks become critical when applied against sensitive devices, such as wearable personal fitness trackers (used for personal telemetry) and implantable medical devices (IMD), because they continuously monitor physical signs, and failures can result in serious physical harm to the user. There are defensive techniques against sleep deprivation attacks specific to these sensitive devices. Some of them involve replenishing batteries [Siddiqi et al. 2021], which may not be feasible in all battery constrained systems. Some techniques rely on external software or hardware to reduce the chances of a successful attack [Abdullah et al. 2019] [Hei et al. 2010] and others propose detection

models based on network analysis or network architecture [Monnet et al. 2015] [Alampalayam and Kumar 2003]. There are also proposals of Intrusion Detection Systems (IDS) oriented to specifically detect battery exhaustion attacks [Nash et al. 2005] [Jacoby and Davis 2004], but they do not consider real low-powered devices used in IoT neither consider a precise estimation of power consumption.

This paper presents a technique, based on attack and security models developed by us, that handle the sleep mode and battery usage to manage energy drain and mitigate sleep deprivation attacks. The technique manages to have a sleep mode that cannot be interrupted by radio communication or programmable sensor/device, and real-time energy consumption data from the battery. Data consumption is recorded into a moving average array which calculates the average power consumption. If the power consumption surpasses a predefined threshold, the device calculates and enters to the sleep mode for the time it needs to rectify its consumption. Compared to related works [Hei et al. 2010, Monnet et al. 2015, Alampalayam and Kumar 2003, Abdullah et al. 2019, Nash et al. 2005, Jacoby and Davis 2004], the proposed technique runs directly on low-power sensors, without depending on external agents, regardless a network architecture to prevent a sleep deprivation attack of having effect on the sensor battery lifetime.

Results from experiments with an ESP32, a popular low-power microcontroller, show that the proposed technique keeps the battery of the microcontroller within its expected lifetime and that ESP32 was able to regulate its own sleep. For instance, when ESP32 had the WiFi Power Saving mode enabled and under a sleep deprivation attack, battery has depleted along approximately 4 hours and 29 minutes in a scenario in which the proposed technique is employed, compared to approximately 2 hours and 58 minutes in the original implementation of ESP32. This means that the proposed technique has increased the ESP32 lifetime of 51.2%.

This paper proceeds as follows. Section 2 overviews the related works. Sections 3 and 4 describe the followed methodology and the proposed solution. Section 5 details the performance evaluation experiments and results. Section 6 concludes the paper and highlights future directions.

2. Related Works

Abdullah *et al.* [Abdullah et al. 2019] wrote about security challenges in the Perception, Network and Application layers of IoT architecture. They also provided security requirements and techniques for each layer. The perception layer, for instance, presents as challenges: Unauthorized Access to the Tags, Tag Cloning, Eavesdropping, Spoofing, and Radio Frequency (RF) Jamming. The network layer is prone to Sybil Attack, Sleep Deprivation, Denial of Service, Malicious Code Injection, and Man-in-the-Middle. The application layer presents vulnerabilities to Malicious Code Injection, Denial of Service, Spear-Phishing, and Sniffing. Abdullah *et al.* described that sleep deprivation attacks can occur in the Network layer, since sensors in a Wireless Sensor Network (WSN) are powered by limited lifetime batteries and nodes are restricted to sleep to extend their lifetimes. A sleep deprivation attack can lead to an early shutdown on a node, limiting the capabilities of a WSN. The work in [Abdullah et al. 2019] presents as a solution against all of these attacks, a blockchain technology. According to the authors, this would keep security, confidentiality, integrity and proper authentication in communications. Similar to our

work, Abdullah *et al.* proposed a solution to mitigate sleep deprivation, but ours will not be based on blockchain. We focused on a solution based on local decisions, without need to trust in external agents to mitigate the attack.

In WSNs, the denial of service attacks can affect the network after the attack, not only during the attack, because some sensors can be low-powered and have this energy completely depleted. Depending on the network's topology, this unplanned shutdown of some sensors can affect the availability of the network. In [Monnet et al. 2015], Monnet *et al.* proposed to use traffic monitoring agents in a clustered WSN to detect potential DoS attackers in the network. These agents need to be trusted, and they need to be elected considering the residual energy of all the nodes inside a cluster to prevent the election itself from being responsible for a DoS. To guarantee a fair election, the authors propose an interactive process that imposes a load balancing between all the nodes and evaluates information obtained from previous communications to avoid suspicious nodes' election. When compared to a (pseudo)-random election, the proposed process was able to increase the lifetime of the network and to avoid that compromised nodes were always selected as a monitoring agent (In the experiments, these compromised nodes were elected only during 12% of the simulation time). Different from our work, Monnet *et al.* did not focus on mitigate the attack. Besides, their solution requires external communication among the nodes during the process of detection. Our proposal does not depend on such communications and take all the decisions locally.

Hei *et al.* [Hei et al. 2010] detailed a resource depletion (RD) attack, its effects on Implantable Medical Devices (IMD), and a solution based on machine learning to detect such attacks. The studied attack is forced authentication, which is triggered by an external reader attempting to connect to the IMD. The authentication process on the device requires communications and computations to be performed, consuming battery power and storage, with every authentication attempt being logged on the device. As a solution, it was proposed to shift the authentication process towards a more capable device, a cellphone, and to create SVM classification models based on the patient's IMD access pattern. The proposed models, both linear and non-linear, achieved a detection rate of over 90% of the RD attacks. Different from our proposal, this solution does not mitigate the attacks. Besides, our proposal does not depend on a second device.

Alampalayam *et al.* [Alampalayam and Kumar 2003] conceptualized an adaptive security scheme against the denial of service threat in mobile agent (MA) systems based on wireless ad hoc network environment. They proposed an Adaptive Security Model (ASM) that addresses protection to and from agents and hosts, provides continuous monitoring, detection and appropriate protection against different active and passive attacks, such as denial of service, packet mistreatment and routing attacks. It is not restricted to a specific type of network domain, and it adapts within a framework to different types of network and application providing flexibility to users for specifying their desired security prevention policies under a given attack scenario. The ASM is composed by three frameworks, Formulation of Vulnerability Metric Framework, System State Characterization and Vulnerability Level Evaluation Framework and Protection Framework. The work simulated the model on an ad hoc network using GloMosim and MATLAB and trained a neural network to fit the security policies and security level specified by users based on network metrics when a denial of service attack is carried out. Different from our work,

its results focus primarily on detection, not on mitigating threats.

IDSs oriented to detect anomalies in the battery consumption were previously proposed in [Nash et al. 2005] and [Jacoby and Davis 2004]. In [Nash et al. 2005], the authors used the relation between energy consumption and system load to propose an IDS that detects battery exhaustion attacks. A linear regression model based on parameters such as CPU load and disk accesses was designed. However, the model is not precise and was derived considering a laptop. In our proposal, we consider real low-powered devices used in IoT systems and use specialized sensors to retrieve more precise power consumption values. The battery-based intrusion detection (B-bid) presented in [Jacoby and Davis 2004] is an early warning system based on battery consumption that runs locally in the device. Similar to [Nash et al. 2005], the proposal does not consider low-powered devices and relies on an imprecise estimation of the power consumption.

3. First Step: The Attack Model

The first step towards the mitigation of a sleep deprivation attack against a sensor powered by battery was to model the attack. Modeling the attack allows its understanding, which is employed to guide a solution. We describe the sleep deprivation attack using three different modeling techniques [Al-Mohannadi et al. 2016]: Diamond Model, Kill Chain and Attack Graph. These techniques focus on different aspects of attack modeling and can be combined to complement each other. The Diamond Model focuses on mapping the motivations behind the attacks; the Kill Chain details the steps that make the attack possible; and the Attack Graph gives an overview of the attack in which the damages caused become more evident. The next subsections describe the sleep deprivation attack following each one of those techniques.

3.1. The Diamond Sleep Deprivation Attack Model

In a nutshell, the attack model following the Diamond technique comprises the Adversary, Capability, Infrastructure and Victim components. Their relationship follows: (i) the Adversary uses Infrastructure and develops Capability; (ii) the Capability exploits the Victim; (iii) the Infrastructure connects to the Victim. In this model, the motivations are more important than the actions taken to generate a security incident.

The diamond presented in Fig. 1 illustrates the key components behind a sleep deprivation attack. The motivation, traced by the Attack Path, may connect to other diamonds in a multiple stage attack, but in our case, it simply shows that the motivation may be a vulnerable infrastructure. The role of each component is depicted below:

- **Adversary:** orchestrates attacks in which they use a compromised part of a larger system (e.g., sensors, IMD, wearables) to reinforce the damage or control the Victim. Sleep deprivation may not have any immediate effect. An incident could take effect over weeks or months.
- **Capability:** usually sleep deprivation would not be performed over the Internet, since the target devices should have their traffic secured over a firewall. The attack is most likely to be carried out by another RF-able device in somewhat close proximity to the sensors. In case of WiFi, the proximity would be under 100 meters in outdoor communications. This device would send specially crafted radio messages to the target sensors, preventing them from entering sleep. The attack would be done over the course of weeks or months.

- **Infrastructure:** RF-able devices and their connected networks are necessary for the attack. The device must also be able to enter sleep and must have a battery-restricted energy supply. The Adversary could use some basic knowledge about the network to replay and spoof messages to keep the attack.
- **Victim:** in case of IMDs and Wearables, the victim may be a Politically Exposed Person (PEP) or the company, and services that control the devices. The company responsible for the Infrastructure as well as the population being serviced by the systems which the sensors are part of are the targets.

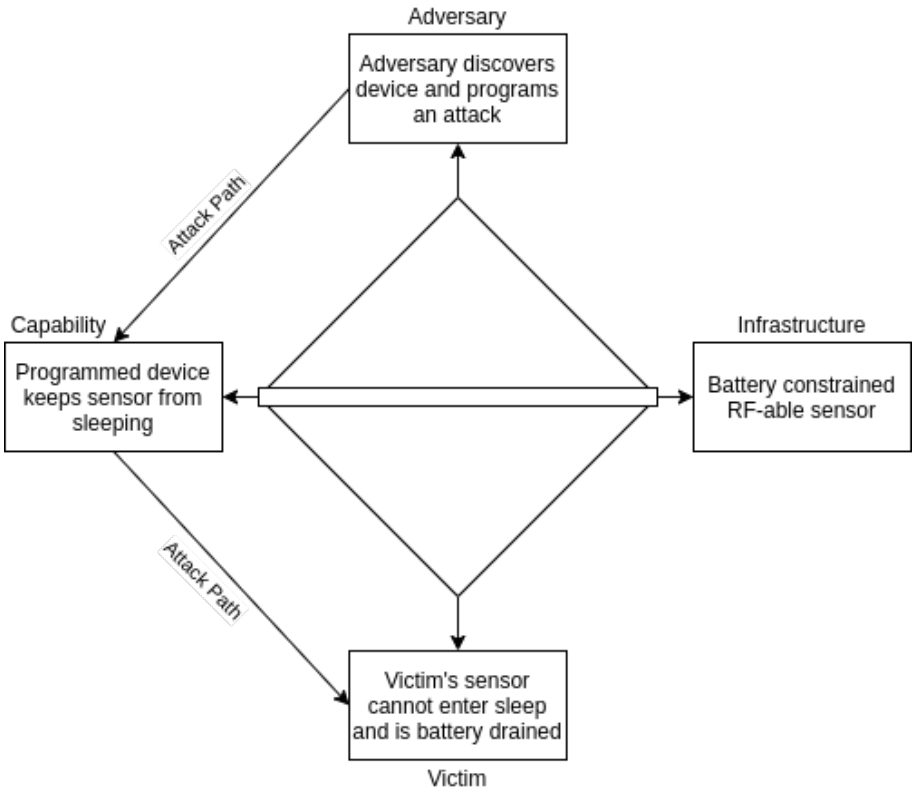

Figure 1. Illustration of the diamond sleep deprivation attack model

From the diamond, one infers that the attack path may jump from the Victim's sensor to another target, most likely an IoT system infrastructure and that a battery constrained RF-able sensor may be all that is needed to carry out an attack.

3.2. The Kill Chain Sleep Deprivation Attack Model

The Kill Chain technique has seven steps, i.e., the attackers need to go through those steps to achieve their goals. The seven steps are: reconnaissance, weaponization, delivery, exploitation, installation, command and control, and action on objectives. It focuses on establishing the actions taken to generate an incident. This technique represents common steps in an attack and their relationship, different from the Diamond Model, which focuses on motivations between attacker and victim. This representation is useful in stopping an attack flow as the disruption of any of the steps will cause the chain to be unsuccessful and the attacker to not reach his/her objectives. Each step of the Kill Chain technique considering the sleep deprivation attack is described as follows:

1. **Reconnaissance:** the attacker searches about the company that owns the sensors and is knowledgeable about where the sensors are located and their accessibility. The attacker would also know the routine of the team members in charge of the sensors maintenance and their reaction towards battery depletion. If the sensor is wearable or an IMD, the attacker also has information about the person carrying the sensor.
2. **Weaponization:** the attacker programs devices to continuously send specially crafted radio signals to keep the sensor from sleeping. If battery constrained, these devices should have a longer lifetime than their target.
3. **Delivery:** devices are placed by the attacker in proximity to their targets and do not need any interaction from the victim to start the attack. The victim would not perceive the attack at its beginning.
4. **Exploitation:** devices start to keep the sensors out of sleeping and battery life begins to degrade. Personnel monitoring the sensor may notice that its battery is draining faster.
5. **Installation:** battery is significantly drained. The next step, realized by the Victim, would be to change the sensor battery, if that is possible, or adapt the system which the sensor is part of to work without it.
6. **Command and Control:** the attacker would not be able to control any resources in this attack, but s/he may intentionally disrupt the system which the sensor integrates to leverage an automatic response.
7. **Action on objectives:** the victim is forced to take an action which the attacker was expecting, opening new unwanted attack vectors.

The Victim can not see how the attacker attempts an attack. It could only be anticipated from the kill chain technique that the attacker has taken those steps. Going reverse order to the chain, from the attack, could become clear that the attacker took time to search about the devices before acting. Once the attack is recognized, one can understand all the steps and find patterns to identify or prevent it.

The Kill Chain shows us the steps that an attacker took to reach his/her goals, meaning that if we can prevent any of these steps, we prevent the attack from reaching a successful outcome. If we stop the battery from draining at an increasing rate, the attack should have no effect.

3.3. The Sleep Deprivation Attack Graph

This technique provides an overview of the attack represented by paths taken to its success. Much like the Kill Chain, it describes the steps taken to execute an attack with the advantage of providing relations between the items modeling the attack. Fig. 2 illustrates the attack graph model for the Sleep Deprivation attack. The oval shapes denote targets and the arrows show the relationships between actors and components, both indicated by rectangles. The attacker seeks to damage or leverage the target company infrastructure or devices. S/He does so by researching the infrastructure targeting sensors. The attack on the sensors compromises batteries and affect the IoT system. This damages the company infrastructure, but it also prompts an action from the maintenance team, which the attacker may use to trigger another exploitation. The attacker may intercept credentials that the maintenance team uses for their activities, for instance.

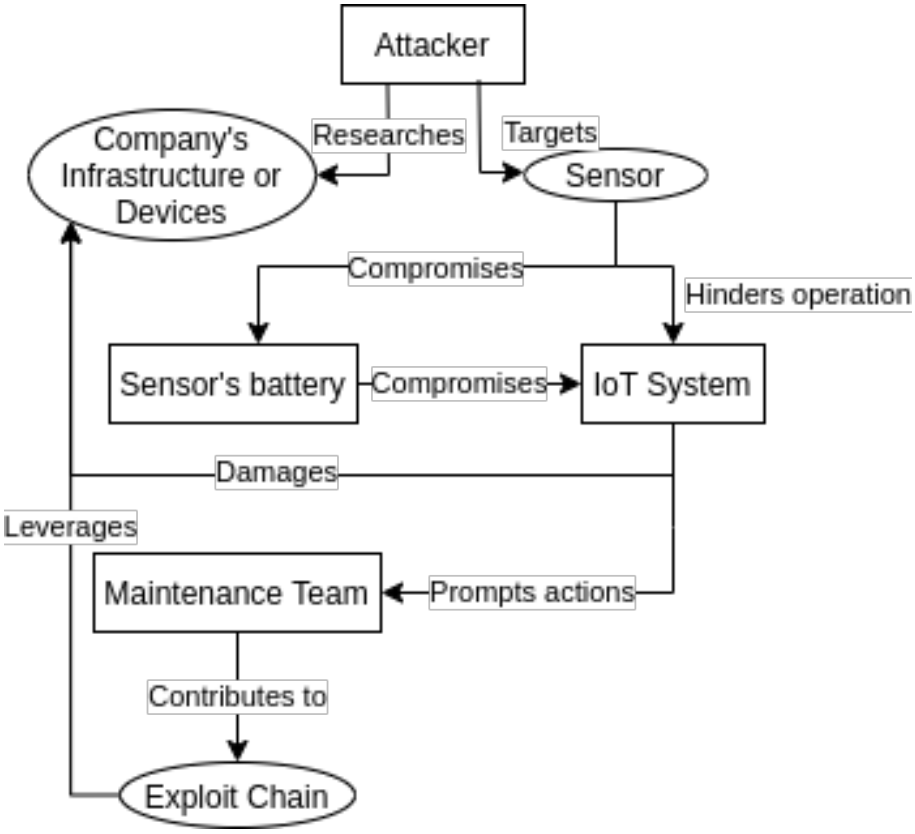

Figure 2. Illustration of the Sleep Deprivation Attack Graph

The attack graph shows us that if we stop the battery from draining at an increasing rate, the company's infrastructure or devices will not be damaged or exploited.

4. Second Step: The Security Model

Attack modeling helps us to provide mitigation solutions to stop attacks from being successfully carried out. There can be many mitigation or security solutions to one kind of attack, we based our solution directly on the sensor, because it would be architecture agnostic, it would work for any sensor network that has battery powered sensors, and it would not rely on a centralized management system, such as an operations center for a facility. The solution adds some computation to be done on the side of the sensor, however it consists of collecting sensor data and calculating a moving average which do not make its implementation resource usage intensive, nor it adds computational lag to the sensor. The proposed solution adds a requirement for the sensor to have its own power consumption data, not adding much complexity to sensor or battery circuit.

With the knowledge obtained by the attack model, the next step is to provide a security model. Babar *et al.* [Babar et al. 2010] presented a generic security model for IoT considering privacy, security and trust. This is a conceptual approach, however, focusing more on the data being processed by IoT devices and systems than the software running to keep those devices operational. We define a security model that enforces a policy and dictates how battery resources should be used. The policy specifies a threshold in power consumption which the device must not surpass. If the consumption surpasses this limit, the device recalculates and enters in a period of sleep to preserve power.

The model balances availability, power consumption and sleep. The device must remain available for communications for a given period of time. It will not be always available due to sleep, but these availability periods should be informed by the device to the rest of the system it is part of. There is a limit specified by a policy about how much energy the device can consume in a given amount of time. This limit is employed to activate the sleep mode whenever the consumption exceeds the limit. Sleep mode will be used to control the device power consumption also making it unavailable for communications. Before entering in sleep mode the device must inform for how long its sleep cycle will be in effect so that the system it is part of may deal with its unavailability.

The security model defined for the sleep deprivation attack can be translated to an algorithm (Algorithm 1) run by the device to control its sleep cycle based on energy consumption. Algorithm 1 presents the sleep cycle to control the device sleep, ensuring that power consumption stays bellow what is defined in the security policy. A set of variables assists in the policy management. Their values must be beforehand specified by the applied policy and device specifications. Respectively, these variables are defined as follows. MAS, the moving average size, sets the number of samples of power consumption the moving average array holds. It must be larger than 2. $SAMPLING\_PERIOD$ is the period of time when power consumption measures are collected. $POLICY\_PERIOD$ is the amount of time the device runs until a policy check is made. $POLICY\_POWER$ is the threshold that the device must not pass and is verified from time-to-time. It must not be lower than the value of $SLEEP\_POWER$. $SLEEP\_POWER$ is the device power drain when in sleep mode. It should remain constant as processing power is only used to keep states and program variables.

The algorithm works by collecting power measurements from the sensor and storing them to a moving average array, each time the power is measured, the average power variable $avgpower$ is updated, as well as the avg array. When a $POLICY\_PERIOD$ amount of time passes, the average power is compared to $POLICY\_POWER$ and if the average power is found greater, both the moving average at avg and $avgpower$ are updated until the average power value goes below the one defined by $POLICY\_POWER$, thus the $sleep\_time$ is calculated as how many iterations it takes to decrease the consumption by entering sleep mode. Lastly the device updates the $time$ variable and enters sleep mode for a period defined by $sleep\_time$. With this algorithm, the device is able to keep its battery life as specified by the policy.

5. Experiments and Results

In order to apply the security model and test its effectiveness, we have designed a circuit using an ESP32 [Espressif Systems 2022] microcontroller in which its total power drain is measured and used to calculate the sleep time of the sensor, i.e., the ESP32 itself. It follows the Algorithm 1 proposed by us to enforce the policy defined by the security model. The experiment comprises ESP32, model ESP-WROOM-32 with a 802.11n WiFi antenna, an INA219 power monitor, a 3.2V LiFePO4 battery and a laptop to program the ESP32 and carry out attacks. The sensor firmware was built using Arduino IDE with binaries provided by Espressif and runs a PubSubClient MQTT client, version 2.8.0, which publishes the power measurements to a mosquitto MQTT broker, version 2.0.10, on the laptop. The laptop OS is Arch Linux with 4.19.20-1-vfio-lts kernel and the laptop's WiFi antenna is Qualcomm Atheros QCA9377 802.11ac.

```
Result: Moving average stored in avg array and average power
        consumption in avgpower
for i ∈ 1..MAS do
    avg[i] ← 0;
end
time ← get_time();
sleep_time ← 0;
while true do
    time_diff ← get_time();
    if time_diff − time % SAMPLING_PERIOD then
        measured_power ← get_power();
        avgpower ← −avg[MAS];
        for i ∈ MAS..2 do
            avg[i] ← avg[i − 1];
            avgpower ← avgpower + avg[i];
        end
        avg[1] ← measured_power/MAS;
        avgpower ← avgpower + avg[1];
        if time_diff − time % POLICY_PERIOD and
         avgpower > POLICY_POWER then
            while avgpower > POLICY_POWER do
                avgpower ← −avg[MAS];
                for i ∈ MAS..2 do
                    avg[i] ← avg[i − 1];
                    avgpower ← avgpower + avg[i];
                end
                avg[1] ← SLEEP_POWER/
                MAS;
                avgpower ← avgpower + avg[1];
                sleep_time ← sleep_time + SAMPLING_PERIOD;
            end
            time ← get_time();
        end
    end
    enter_sleep_mode(sleep_time);
    sleep_time ← 0;
end
```

Algorithm 1: Sleep cycle algorithm

Fig. 3 shows how we connected the ESP32 board with the INA219 and the battery. Note that we did not use AA batteries, we used a single LiFePO4 battery, but merely left it on the diagram to illustrate the connections. ESP32 presents both a hybrid sleep mode and a deep sleep mode. In the hybrid sleep mode, the ESP32 antenna remains active during sleep, whereas in the deep sleep mode, only the co-processor chips are kept powered on. The hybrid sleep mode is vulnerable to sleep deprivation attacks. In order to test the security model, a malicious traffic is sent to the device WiFi antenna to increase its power consumption. The malicious traffic lies in sending a controlled number of TCP SYN packets. The experiments have employed the *hping* [Salvatore Sanfilippo 2006] program to generate packets every millisecond. As expected, when our mitigation technique is employed, the device is able to detect the increase in power usage and enter in the deep sleep mode for a period of time that would rectify its power consumption. The device does not power its antenna during sleep, ensuring no sleep deprivation could be carried out.

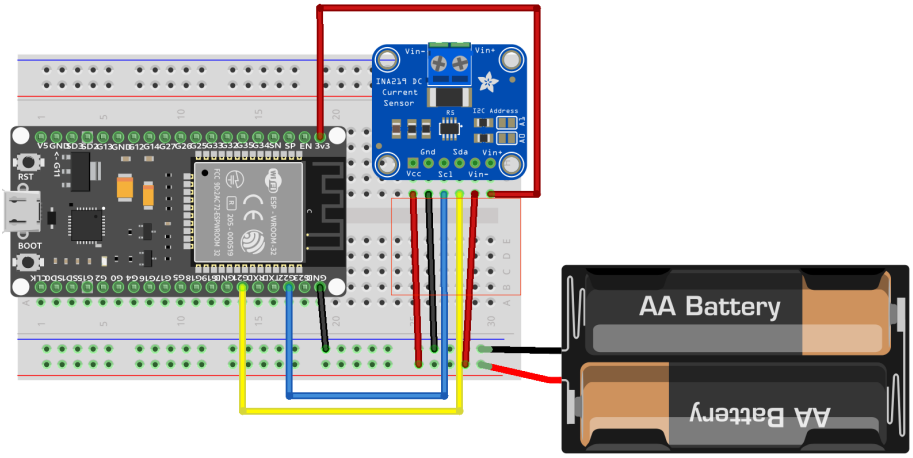

Figure 3. ESP32 Dev Kit board connected with the INA219

The experiments employ an array with size 32 for the moving average samples, a sampling period of 1 second, a policy period of 60 seconds, a power (policy) threshold of 180mW and calculated the device sleep power consumption at 0.5mW. Table 1 shows how long the sensor battery took to deplete over WiFi power saving mode (vulnerable to sleep deprivation) and sleep mode (protected by Algorithm 1). By the results, it is possible to see that the Security model was effective in the experimental scenario. Comparing the environment under attack without the mitigation technique and with the mitigation technique, the mitigation technique was capable to increase the battery's lifetime by 51.2%.

6. Conclusions and Future Works

Depending on the IoT devices configuration, sleep deprivation attacks can significantly compromise critical systems, such as eHealth systems that depend on implantable medical devices. This paper presented a security model and a sleep cycle algorithm to mitigate sleep deprivation attacks permanently. Performance evaluation carried out in a system based on an ESP32, a popular low-power microcontroller, showed that the technique could increase the battery's lifetime by 51.2% in a scenario under a sleep deprivation

Table 1. Total time until battery depletion under WiFi power saving and sleep mode

With Mitigation	Scenario	Battery Time(s)
No	WiFi Power Saving	15533
	WiFi Power Saving under attack	10672
Yes	Sleep Mode	16211
	Sleep Mode under attack	16137

attack. Future works may address some attack scenarios where the sensor is in hybrid sleep, with its radio antenna still in operation, or where the attack detection is made outside the sensor's hardware. There should also be considerations made about the sensors availability and denial of service attacks that aim to put the sensor into sleep mode and if traditional denial of service mitigation still applies.

Acknowledgments

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9. It is also part of the FAPESP proc. 18/23098-0.

References

Abdullah, A., Hamad, R., Abdulrahman, M., Moala, H., and Elkhediri, S. (2019). CyberSecurity: A Review of Internet of Things (IoT) Security Issues, Challenges and Techniques. In *Proceedings of the 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–6.

Al-Mohannadi, H., Mirza, Q., Namanya, A., Awan, I., Cullen, A., and Disso, J. (2016). Cyber-Attack Modeling Analysis Techniques: An Overview. In *Proceedings of the IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 69–76.

Alampalayam, S. and Kumar, A. (2003). An Adaptive Security Model for Mobile Agents in Wireless Networks. In *Proceedings of the 2003 IEEE Global Telecommunications Conference (GLOBECOM)*, volume 3, pages 1516–1521 vol.3.

Babar, S., Mahalle, P., Stango, A., Prasad, N., and Prasad, R. (2010). Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT). In *Proceedings of the International Conference on Network Security and Applications*, pages 420–429.

Espressif Systems (2022). ESP32 Series Datasheet. `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`. Accessed 06/15/2022.

Hei, X., Du, X., Wu, J., and Hu, F. (2010). Defending Resource Depletion Attacks on Implantable Medical Devices. In *Proceedings of the 2010 IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5.

Jacoby, G. and Davis, N. (2004). Battery-Based Intrusion Detection. In *Proceedings of the 2004 IEEE Global Telecommunications Conference (GLOBECOM)*, volume 4, pages 2250–2255 Vol.4.

Lima, M. N., dos Santos, A. L., and Pujolle, G. (2009). A Survey of Survivability in Mobile Ad Hoc Networks. *IEEE Communications Surveys & Tutorials*, 11(1):66–77.

Markets and Markets (2020). Battery Market for IoT by Type, Rechargeability, End-use Application, and Geography - Global Forecast to 2025. `https://www.marketsandmarkets.com/Market-Reports/battery-iot-market-153084557.html`. Accessed 06/15/2022.

Monnet, Q., Hammal, Y., Mokdad, L., and Ben-Othman, J. (2015). Fair Election of Monitoring Nodes in WSNs. In *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

Nash, D., Martin, T., Ha, D., and Hsiao, M. (2005). Towards an Intrusion Detection System for Battery Exhaustion Attacks on Mobile Computing Devices. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 141–145.

Nguyen, V.-L., Lin, P.-C., and Hwang, R.-H. (2019). Energy Depletion Attacks in Low Power Wireless Networks. *IEEE Access*, 7:51915–51932.

Pirretti, M., Zhu, S., Vijaykrishnan, N., McDaniel, P., Kandemir, M., and Brooks, R. (2006). The Sleep Deprivation Attack in Sensor Networks: Analysis and Methods of Defense. *International Journal of Distributed Sensor Networks*, 2(3):267–287.

Rodriguez, L. G. A. and Batista, D. M. (2020). Program-Aware Fuzzing for MQTT Applications. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, page 582–586.

Salvatore Sanfilippo (2006). Hping – Active Network Security Tool. `http://www.hping.org/`. Accessed 06/15/2022.

Siddiqi, M. A., Serdijn, W. A., and Strydis, C. (2021). Zero-Power Defense Done Right: Shielding IMDs from Battery-Depletion Attacks. *Journal of Signal Processing Systems*, 93:421–437.

Statista (2022). Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2030. `https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/`. Accessed 06/15/2022.