Software Security Analysis on Internet-of-Things Applications

Steven Ngo
Dept. of Computer Science and Software Engineering
California Polytechnic - San Luis Obispo
San Luis Obispo, CA
sngo12@calpoly.edu

Dongfeng Fang
Dept. of Computer Science and Software Engineering
California Polytechnic - San Luis Obispo
San Luis Obispo, CA
dofang@calpoly.edu

***Abstract*—In this paper, we present a software security analysis on Internet-of-Things (IoT) applications. We first present a system model of an IoT system from a software perspective. Based on the proposed system model, we discuss the software attack models and security objectives. Software security analysis is presented. Based on the software security analysis, we propose a software complexity and information score framework. A smart-home case study is presented to show our proposed framework can be used to analyze the security of the system.**

I. Introduction

Internet-of-things (IoT) devices have been useful in enhancing people's quality-of-lives, ranging from medical equipment such as pacemakers and infusion pumps to smart-home devices such as security cameras and voice-operated assistants [1], [2]. Due to the resource limitations of IoT devices, software implementations of IoT tend to be limited in resource consumption and development time as much as possible. A lot of these devices also are designed with new and innovative features at the forefront in an attempt to grab new consumers as tech companies compete against one another to get their devices first to market. As a result, security implementations are either postponed for future updates or never revisited, and easily exploitable vulnerabilities such as guessable default passwords and exploitable default configurations do not get patched up or changed [3]. This introduces additional entry vectors for malicious users who may want to compromise the IoT devices in order to gain access to sensitive information or even hijack the devices' computational power for attacks [4].

Due to the heterogeneous nature of IoT devices, it is difficult to create security solutions that can be applied across a wide range of devices [5], [6]. Some devices may be lacking in their amount of memory and computational power, such as when comparing a home automation hub with a smart display and operating system against door locks that have only just enough to connect to a wireless network. Considering estimations from Huawei say there will be close to 100 billion IoT devices by 2025, it is essential that all devices have security implementations that provide moderate level of countermeasures or error mitigation [7].

This study will focus more on the software security aspect of IoT devices, as well as looking into other perspectives including implementation issues that arise during the software development life-cycle and technical issues that are the result of the software's programming language composition. In addition, we also will be focusing on smart-home IoT devices due to their exponentially increasing prevalence, their convenience of purchasing and set-up, and their ease of usage. Smart-home devices include smart plugs, locks, hubs, thermostats, and home security cameras. The compromised smart-home IoT devices can lead to a violation of a home and its residents' privacy as well as weaken the home's physical security [8].

There are a relatively few amount of academic literature that focus on software security within the larger pool of general IoT security. General threats and current issues within IoT security, which include a wide variety of attacks that deal with communication protocols and cryptography are studied in [7] [9] [10]. The different perspectives regarding the practices and methodologies within software engineering and programming as it relates with IoT are discussed in [11] [12] [13] [14] [15]. Common malware to IoT devices that prey on vulnerabilities within a device's software and settings, as well as major malware incidents and networks such as the Mirai botnet, are studied in [3] [4].

Therefore, in this paper, we present a software security analysis on IoT applications. Our contributions are summarized as the following:

- A general IoT network architecture is proposed based on software perspectives.
- Software attack models and security objectives are presented based on our proposed system model.
- A software complexity and information scoring system for the software and firmware of IoT devices is proposed.
- A case-study using our proposed scoring system is carried out based on our smart-home lab set-up.

The rest of this paper is organized as followed. In section II we propose a custom system model of an IoT general network architecture based on a software perspective. In section III, a software security analysis of IoT devices and its security implementations of varying perspectives is presented. In section IV, we propose a software complexity and information score framework. In section V, we apply the score framework to a custom case study using a simulated IoT smart-home system. In section VI, the conclusion is presented.

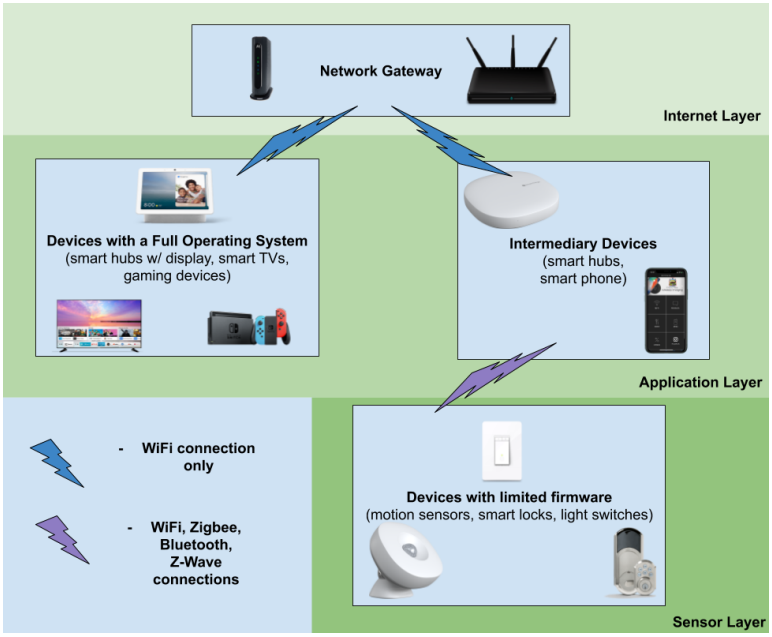

Fig. 1. A General IoT Network Architecture (software-defined)

II. System Model

In this section, we introduce a general IoT network architecture and system environment based on a software perspective, as well as software-based attack models and security objectives.

A. General Network Architecture - Software Perspective

In Figure 1, a general IoT network architecture is shown. The system model has three layers and four classifications that have been created based on the networking connections between types of devices as well as the complexity level and functionality of IoT devices' software and firmware.

- **Network Gateway**: These are often routers and gateways that assist in the connectivity between all internet-functioning devices and act as the bridge between a system of IoT devices, any cloud services required by the devices, and the rest of the internet. They contain software to assist with establishing connections through WiFi over wide and local areas, and network settings are often configured and managed through a locally-available web portal.
- **Devices with a Full Operating System**: These devices have more complex software systems, which may include being run on an operating system (OS) and be interacted with for various user functionality through an interface screen. These may include smart hubs with displays and gaming consoles, and these devices most likely have openly-available information regarding what OS they run, or are based off of, and what programming languages compose such OS's. The software in these devices are complex in scope and functionality and often allow users to interact with applications and settings hosted on and provided through the device itself. All of the different applications (apps) available on these devices, including first-party and third-party ones, also serve as software that are open to malicious attacks.
- **Intermediary Devices**: We create a separate classification for devices that don't necessarily have as complex software as devices with a full operating system, but act as an in-between for devices with limited firmware and network gateways. Both intermediary devices and devices with a full operating system connect to network gateways through WiFi, but within IoT, additional communication protocols such as ZigBee, Zwave, and Bluetooth are used to connect devices with hardware or networking constraints to smart hubs, which then relay information to network gateways or other devices in the network. These smart hubs still have their own software implementations to deal with network connections, and with that come additional potential vulnerabilities. As a result of this specification, some devices may have an overlap in terms of where they can be classified, primarily smart hubs.
- **Devices with Limited Firmware**: These devices rely on very little to no running software and instead run primarily on firmware for establishing network connections and executing its intended functions only as needed by users. In a smart-home environment, this include devices such as smart locks, smart door sensors, smart light switches, and smart plugs. Direct user interaction with such devices are limited to either their intended mechanical functions such as manually (un)locking a smart lock or there may even be a lack of any user interaction apart from its initial set-up. A smart door sensor would simply be set up and then automatically send information about any movement

to other devices classified as intermediary ones.

B. Software Attack Models

Throughout the three layers, there are different methods to compromise the IoT devices that fall under their respective classifications from our software-defined IoT network architecture. We will focus primarily on attacks that relate most to software and applications. More traditional methods of hacking still apply to IoT, but there are types of attacks that have made a resurgence in frequency or play off of the differences between older systems like computers and the newer IoT devices. We will discuss further in-depth the different aspects of IoT software security in the next section.

- Brute-force Dictionary Attacks: The most direct attack is brute-forcing login information on IoT devices' web-access portals or anywhere else that requires device-specific credentials. According to the Open Web Application Security Project (OWASP) IoT Top 10 list of security risks and vulnerabilities, the number one vulnerability throughout all IoT devices is easily guessable, default usernames and passwords that make it easy for hackers to either simply look them up online or at most use brute-forcing scripts to execute dictionary attacks [16]. Unauthorized access into a device compromises it entirely, allowing the hacker to do whatever they wish as if they have administrative access.
- Malware/Custom Attacks Based on CVEs: Malware is another highly-effective method of compromising IoT devices. Similar to malware found in traditional computing environments, there are still a wide range of malware used to attack IoT devices including worms, trojan horses, spyware, and rootkits [3]. Infection and self-propagating methods used in more traditional computing environments still apply within IoT. These include scanning for open ports and testing a dictionary list of default login credentials and relying on device-specific common vulnerabilities and exploits (CVEs) for specially-crafted attacks, which may include buffer overflows, SQL injections, privilege escalation, and cross-site scripting where interfaces are available [5]. IoT malware can infect any of our four device classifications.
- Software Update Attacks: Without proper validation of software and firmware updates for IoT devices, they are susceptible to a variety of attacks that could involve intercepted updates that are then modified with an attacker's custom malware. They may also involve purposefully replayed updates that are known to be flawed but are still from the original manufacturer to bypass weak validation and authentication techniques [17]. These flaws can then be taken advantaged of by attackers to compromise the device in any way they can configure their attacks, including to steal confidential information or use the device's computational power in their botnets.
- Attacks due to Lack of Software Update: If an IoT device can't receive any software or firmware updates at a very little to no frequency, attackers can capitalize off of any vulnerabilities that were a result of poor implementation and coding practices during the device's development phase. Even if these vulnerabilities are to be eventually discovered by security researchers, without a proper system in place for any patch fixes, it is much more cumbersome to address such issues due to the nature of certain IoT devices (especially ones that may not be always on WiFi and rely on other devices).

C. Security Objectives

When considering software security design and implementations for smart-home IoT devices and systems, these security objectives should be at the fore-front. Based on our attack model, we propose the following security objectives:

- Availability: Especially for IoT devices that provide home physical security, including security cameras and smart locks, it is crucial for an IoT device to be active and functioning as intended for as long as a user needs it, which could be around-the-clock for certain physical security-based devices. One scenario to consider is that a potential home invader could hack into and disable a smart lock installed on a house's front door with a simple buffer overflow attack if the device's software does not have any input validation on the passcode, completely taking away any aspect of the device's availability.
- Robustness: There should be countermeasures put in place in the event a device is disabled or compromised from a malicious attack to mitigate any further damage. This could include alerts to the user combined with automatic reboots and safe restarts with detection of anomalous behavior. If possible, the IoT device's software should also be developed to withstand and still function properly against excessive tampering and errors, with one method of accomplishing this by following secure coding practices such as implementing input validation on all user-input interfaces or adhering to the principle of least privileges for access control.
- Privacy: If an IoT device can easily be compromised and taken over by a hacker due to improper security implementations and poor software development practices, it could quickly turn into that hacker's spying device depending on what the device's original functionality is. Home cameras may turn into malicious monitoring devices without the original users even knowing, while compromised movement sensors may send time information back to command-and-control servers so that home invaders can determine when the best times to rob a house are. Any devices with the ability to pick up on noise may also be turned against the original users to tap into their conversations with potentially sensitive information.
- Patchability: This refers to an IoT device's ability to be sent software updates as necessary, as this would be essential if a IoT development team discovers bugs and vulnerabilities within their software at a post-development phase (after shipping products worldwide, months into mainline usage) and they need to send out patch fixes

to prevent any further damage. This is a lot harder for IoT devices to achieve, especially with devices with limited firmware and may not always maintain an active connection to the internet due to their reliance on another device. Some devices may not even support the reception of further updates due to it being more cumbersome to implement such an ability or more expensive to maintain IoT device software.

- Integrity: From the perspective of software integrity, a device's software needs to be able to handle changes and updates without its base functionality being affected (unless the original user is explicitly making custom changes). It should also not be susceptible to rogue software updating attacks through properly authenticating origin of updates and validating any changes in software against the original manufacturers.

III. Software Security Analysis

There is a greater need for both general and more-specific studies into the software security of IoT devices and their implementations, as it is no longer acceptable to rely on development practices and security implementations/techniques from more traditional systems to support the exponentially growing variety of IoT devices. In this section, we aim to provide additional insight into various aspects of IoT software security that utilizes existing information as a foundation and reference point but branch into more difficult challenges that still need to be addressed. Four various aspects are discussed, including programming language vulnerabilities, resource constraints, IoT malware in-depth, and software engineering/product development.

A. Programming Language Vulnerabilities

Embedded systems are primarily developed in programming languages that allow for explicit memory management, such as C and C++. While this is beneficial for IoT devices, it makes the device vulnerable to memory corruption attacks including buffer overflows and out-of-bound reads that would either allow the hacker to carry out remote code executions or simply crash the device if the error is not handled gracefully [10]. Remote code executions may result in the overwriting of important data, privilege escalation, or injection of malware.

B. Resource Constraints

Traditional computer systems often are able to utilize security implementations that can deter and prevent generic memory corruption attacks through randomizing and reordering address and stack spaces or diversify compiled code structure. Implementations like these include address space layout randomization (ASLR) and code obfuscation through no-operation instructions [5]. But, hardware and resource constraints make these much more difficult to implement on IoT devices, as such implementations may require certain hardware components that are fairly rare in IoT or be considered unnecessary usage of process and memory resources that could be better used elsewhere for battery consumption and performance time considerations [11]. For example, ASLR normally requires a memory management unit (MMU) or relocatable/loadable software libraries to implement, but IoT devices normally lack a MMU as well as such software libraries.

C. IoT Malware In-Depth

With the high and ever-growing number of active IoT devices being permeated all throughout society, including homes, hospitals, civil infrastructure, and industry, it is a target-rich environment for IoT malware, as already seen with the Mirai botnet and its variants [4]. The diversity, location spread, and abundance of IoT devices makes them suitable members of botnets that rely on their collective computational power to carry out distributed denial-of-service attacks (DDoS), but botnets are not the only result from malware. They've also been known to engage in crypto-currency mining, monitor and spy on sensitive information and communications, and destroy a device's internals. A large majority of IoT devices are infected with brute-force dictionary attacks on default credentials, balanced dictionaries, and more recently the exploitation of common vulnerabilities/CVEs. These attacks are all preventable with proper security implementations and usage of security best practices, yet the fact remains that it is still a growing and evolving issue for device developers and manufacturers, and attack methods are only getting more sophisticated by the day.

D. Software Engineering/Product Development

Some of the security issues listed above, including hard-coded, easily-guessable login credentials, programming language-based/code-based vulnerabilities, and CVEs, can be remedied and mitigated through usage of secure and clean coding practices throughout software and product development. Unfortunately, developers are too busy coming up with novel and exciting features within the short time-frame they already have to push out their own IoT products against other tech companies racing to do the same in an effort to increase profits. As a result, security implementations are passed over for features that are more marketable to consumers and secure-coding best practices are often overlooked in fast-paced software development methodologies such as with Agile [12]. While such fast-paced methodologies allow for increased productivity and frequent release cycles for a company's product development timeline, a device's level of security suffers as a result simply because of how security implementations, vulnerability patches, and attack countermeasures within IoT devices are currently viewed as afterthoughts or considered to have little to no practical consideration. [11] involves a case study of two startup companies expanding into the area of IoT that analyzed what size role did security have throughout their development processes Both faced security-related challenges within their use of agile processes and sprint delivery, which included their approaches to integrating security requirements into their development processes, third-party component security, meeting market-specific regulations

and security standards, and run-time security (live patches and updates).

IV. Proposed Software Complexity and Information Score Framework

In an effort to expand how we can rate IoT devices and their security level from a stronger software-based perspective, we propose a software complexity and information scoring framework/system. This is inspired from scoring systems such as the Common Vulnerability Scoring System (CVSS) from the Forum of Incident Response and Security Teams (FIRST) and the risk assessment framework from the Open Web Application Security Project. Our framework focuses more on the device itself as well as the information that is more readily available from a consumer's perspective. There are multiple sections and scoring options that are assigned based on whichever criteria is met. Three different perspectives are considered. The proposed framework is as follows:

A. Updates:

- Has the device's software/firmware been updated?
  - Yes and updated within last 6 months
  - Yes and not updated within last 6 months
  - No
- How are updates activated/installed?
  - Multiple options
  - Singular option
  - It doesn't update

B. Settings Interaction:

- What is the device's main method for interaction with?
  - Directly through UI
  - Through another device (app, web portal, smart hub)
  - Can't be interacted with

C. Operating System/Programming Language:

- Is there an operating system or known firmware running on it?
  - Yes and programming language info. is known
  - Only OS/firmware info. is known
  - Not found

In practical case, a numerical range can be assigned to each question. To find the score of an individual device, add up the scores from each criteria/question and divide by four (finding the average). For example, consider a device that's had its software updated within the past 6 months, has multiple update options, can directly be interacted with through the UI, and there's an operating system running on it with known programming language information. This device would receive the maximum score, numerical value depending on the range.

V. Smart-Home Case Study

One popular IoT system environment and practical usage is with the concept of smart homes, where everyday household activities are enhanced with IoT devices. This may include automatically adjusting house temperatures through smart thermostats, physical security upgrades through smart locks and door sensors, or energy consumption monitoring and automatic device power-offs through smart plugs and power strips [8]. With IoT devices simulating an entire smart-home system, we perform a software security case study that involved gathering information about the device's updatability and software specifications, analyzing the devices' software complexity and information, and referencing our data against existing vulnerabilities and previous attacks. We use our software-based IoT network model from section II and our proposed software complexity and information scoring framework from section IV to assist in our case study.

A. Software Complexity and Information Score

We first had to find as much openly-accessible information as we could about all of the devices that composed our smart-home IoT environment, which included official device manufacturer websites, online store pages, and published device instruction sets, usage manuals, and media outlets that covered such devices. The IoT devices that are a part of our smart-home environment include the following, broken down into the classifications from our software-based IoT network model shown in Table IV-C. After finding as much official information as possible on each device, we proceeded to use such information to score each individual device based on our software complexity and information scoring framework.

In Figure 2, we have a histogram representing the scores from all of our listed devices. There were a total of 30 devices, with 1 network gateway, 7 devices with a full operating system, 1 intermediary device, and 21 devices with limited firmware.

B. Ethics Statement

When performing the case study and survey with all of the smart-home devices, including finding information on all of the listed devices, we relied on what can be found from a device's manufacturer's site and their support pages or technology/industry news sites. We avoided using any information from user-driven forums or media to keep our analysis and conclusions as official as possible. We also did not have physical access to any of the devices we mentioned and studied throughout our research period.

VI. Conclusion

In this paper, we presented a software security analysis on IoT applications. We first modeled a general IoT system from a software perspective. The software attack models and security objectives are discussed. The software security analysis is also presented. A software complexity and information score framework is proposed. Based on a smart home system, we applied our proposed software complexity and information score

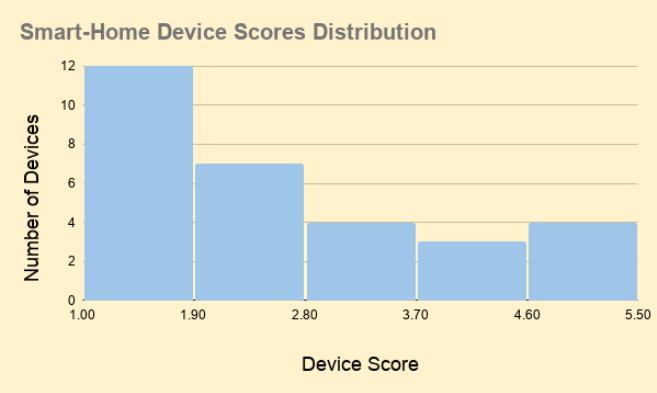

Fig. 2. Framework score applied to IoT devices in the case study

TABLE I
LIST OF IOT DEVICES IN SIMULATED SMART-HOME ENVIRONMENT

Classifications from System Model	Device Names
Network Gateways	D-Link WiFi Router - AC1750 Wireless Internet
Devices with a Full Operating System	Google Nest Hub Max, Amazon Echo Show 5, Facebook Portal, Nintendo Switch, Samsung 43" Smart TV (2018 and 2019 models), Samsung 3rd Generation SmartThings Hub
Intermediary Devices	Philips Hue Smart Hub
Devices with Limited Firmware	Philips Hue Smart Bulb, GE Enbrighten Smart Light Dimmer, Ecobee3 Lite Smart Thermostat (2nd gen.), Wyze Cam Pan, Samsung SmartThings Motion Sensor, Linkind PIR Motion Sensor, Ecolink Flood & Freeze Sensor, Ring Video Doorbell (1st gen.), Ring Video Doorbell Pro, Kasa Smart Plug Power Strip, AmazonBasics Microwave, Google Nest Protect Smoke + Carbon Monoxide Alarm, Google E Nest Automated Climate Thermostat, WGCC Bluetooth Fingerprint Padlock, Gosund Mini Smart Plug, Kwikset SmartCode 888 Smart Lock, August Home Smart Lock Pro + Bridge, Schlage Connect Smart Deadbolt, Centralite Micro Door Sensor, Dome Home Automation Water Shut-off Valve, Centralite Smart Plug Mini

framework. The results show that currently, most IoT devices in the system do not have sufficient security implementations from a software perspective.

VII. ACKNOWLEDGEMENT

We would like to thank the Cal Poly - San Luis Obispo to supporting this project as a part of the college's Summer Undergraduate Research Program 2020.

REFERENCES

[1] D. Zaldivar, L. A. Tawalbeh, and F. Muheidat, "Investigating the security threats on networked medical devices," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020, pp. 0488–0493.

[2] D. Fang, Y. Qian, and R. Q. Hu, "Security for 5g mobile wireless networks," *IEEE Access*, vol. 6, pp. 4850–4874, 2018.

[3] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 161–175.

[4] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, 2017, pp. 267–272.

[5] A. Koivu, L. Koivunen, S. Hosseinzadeh, S. Laurén, S. Hyrynsalmi, S. Rauti, and V. Leppänen, "Software security considerations for iot," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016, pp. 392–397.

[6] D. Fang, Y. Qian, and R. Q. Hu, "A flexible and efficient authentication and secure data transmission scheme for iot applications," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3474–3484, 2020.

[7] J. Zhang, H. Chen, L. Gong, J. Cao, and Z. Gu, "The current research of iot security," in *2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC)*, 2019, pp. 346–353.

[8] Z. B. Celik, E. Fernandes, E. Pauley, G. Tan, and P. McDaniel, "Program analysis of commodity iot applications for security and privacy: Challenges and opportunities," *ACM Comput. Surv.*, vol. 52, no. 4, Aug. 2019. [Online]. Available: https://doi.org/10.1145/3333501

[9] K. S. Niraja, Murugan, and P. Csr, "Security risks in internet of things: A survey," 12 2017, pp. 1–7.

[10] J. Deogirikar and A. Vidhate, "Security attacks in iot: A survey," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, pp. 32–37.

[11] A. N. Duc, R. Jabangwe, P. Paul, and P. Abrahamsson, "Security challenges in iot development: A software engineering perspective," ser. XP '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3120459.3120471

[12] A. Taivalsaari and T. Mikkonen, "A roadmap to the programmable world: Software challenges in the iot era," *IEEE Software*, vol. 34, no. 1, pp. 72–80, 2017.

[13] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An in-depth analysis of iot security requirements, challenges, and their countermeasures via software-defined security," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 250–10 276, 2020.

[14] C. Bellman and P. C. van Oorschot, "Best practices for iot security: What does that even mean?" 2020.

[15] I. Jacobson, I. Spence, and P.-W. Ng, "Is there a single method for the internet of things?" *Communications of the ACM*, vol. 60, pp. 46–53, 10 2017.

[16] "Owasp internet of things project." [Online]. Available: https://wiki.owasp.org/index.php/

[17] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, and E. Baccelli, "Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check," *IEEE Access*, vol. 7, pp. 71 907–71 920, 2019. [Online]. Available: https://hal.inria.fr/hal-02351794