Intelligent Environments 2021
E. Bashir and M. Luštrek (Eds.)

doi:10.3233/AISE210074

Perception of Security Issues in the Development of Cloud-IoT Systems by a Novice Programmer

Fulvio CORNO [a], Luigi DE RUSSIS [a] and Luca MANNELLA [a,1]
[a] *Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy*

Abstract. It is very hard (or ineffective) to take an old system and add to it security features like plug-ins. Therefore, a computer system is much more reliable designed with the approach of security-by-design. Nowadays, there are several tools, middlewares, and platforms designed with this concept in mind, but they must be appropriately used to guarantee a suitable level of reliability and safety. A security-by-design approach is fundamental when creating a distributed application in the IoT field, composed of sensors, actuators, and cloud services. The IoT usually requires handling different programming languages and technologies in which a developer might not be very expert.

Through a use case, we analyzed the security of some IoT components of Amazon Web Services (AWS) from a novice programmer's point of view. Even if such a platform could be secure by itself, a novice programmer could do something wrong and leave some possible attack points to a malicious user.

To this end, we also surveyed a small pool of novice IoT programmers from a consulting engineering company. Even if we discovered that AWS seems quite robust, we noticed that some common security concepts are often not clear or applied, leaving the door open to possible issues.

Keywords. AWS, cloud, cybersecurity, IoT, novice programmers, security

1. Introduction

Designing a system is one of the more critical parts of the development life cycle since this phase's decisions could be part of the product's final release. For this reason, it is essential to design new systems having (at least) the main security concepts in mind. In this way, it will be less probable to expose possible attack points when the final product is ready for the users.

Unfortunately, this is not always so easy, especially for a programmer that is new to a particular technology. It is challenging to avoid "by design" security issues if the programmer does not feel comfortable with a specific technology. The Information and Communications Technology world is changing continuously: several technologies, products, frameworks, and software libraries appear and die every year. So, in a diversi-

[1]Corresponding Author: Luca Mannella, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy; E-mail: luca.mannella@polito.it.

fied programming field like the Internet of Things (IoT), even an experienced programmer could be considered a novice in a new application domain from a certain perspective.

In this work, we will not talk about someone that is currently learning how to program, but we are going to focus our analysis on *novice IoT programmers*, those software developers that had never developed a full-working and ready-for-production IoT system. To understand better novice IoT programmers' security perception, we analyzed a use case related to a teaching activity recently conducted by our research group. This use case is focused on *Amazon Web Services* (AWS), which currently is one of the most complete, widespread, and reliable cloud computing platforms available on the market [1]. AWS respects *confidentiality, integrity, and availability* (CIA) of the customer's data [2], and it provides services on demand: the customers pay only for resources that effectively consume. According to all these characteristics, it is quite probable that a novice programmer starts from AWS for developing a cloud IoT application with a robust and easy back-end. Even if such a platform could be secure by itself [2], a lack of experience could bring the developers to leave some possible attack points to a malicious user. Therefore, in case something goes wrong, the programmer can not wholly blame Amazon.

For all these reasons, we decided to conduct a survey on a small group of novice IoT programmers — taken from a consulting engineering company in Italy — to understand how much a lack of knowledge could be potentially harmful to the final product of this kind of developer.

In this paper, we are going to have an analysis of the related literature in Section 2. In Section 3, we will present the use case taken into account in our research activity, focusing on the possible reliability issues in Section 3.1. Afterward, in Section 4, we will conduct an AWS analysis focused on the previously highlighted attack points. Meanwhile, in Section 5, we will examine the perception of these issues by our group of novice IoT programmers. To conclude, we will discuss in Section 6 what is the actual severity of the previously cited attack points considering both the AWS design and the developers' perceptions. In the end, in Section 7, we will provide some considerations related to this topic, and we will propose some insights for future works.

2. Related Work

The study of the way of thinking of novice programmers is not a new field of research; in 1989, Soloway and Spohrer published an entire book [3] to understand better the main issues of this class of programmers. More recently, in 2005, Lahtien et al. [4] conducted a quite extensive survey — counting 559 students and 34 teachers from 5 different countries — focused on the most common issues in learning how to programming by a university student. In this study, scholars pointed out that novice programmers' biggest problem should not be to understand a computer science course's concepts but, instead, learning how to apply them. This issue could be one of the reasons why, even if the attendees of our survey are aware of some security concepts, they did not apply them while developing their prototypes.

Even in our research group, this is not the first research activity related to novice programmers. We started studying *novice IoT programmers* of 2 different editions of the "Ambient Intelligence" course (2014 and 2015), held in Politecnico di Torino. In a previous study [5], we highlighted some of the most painful points for this class of developers.

In particular, we discovered that novice programmers perceived as extremely difficult the tasks regarding: integrating various subsystems, interaction with proprietary third-party services, and the configuration of mobile, web, or hybrid applications. Moreover, we are still studying and working on possible solutions to help programmers develop better IoT solutions like a computational notebook focused on IoT technologies [6].

However, it is not so common to find studies related to the *security perception* of a novice programmer. Usually, researchers analyzed this kind of perception from a non-technical point of view. For example, Varga et al. [7] published a work related to the cyber-threat perception of actors belonging to the Swedish financial sector.

Nowadays, the benefits of cloud computing are clear both for technicians and for the general public. However, every technology can expose security threats. Scholars have studied possible security issues of cloud computing since the very beginning of its worldwide diffusion [8][9]. On the one hand, according to the security section of their website[2], Amazon puts much effort into keeping AWS a reliable service. In their white papers, Amazon ensures that the IT infrastructure that AWS provides to its customers is designed and managed in alignment with security best practices and several IT security standards [10]. This commitment also seems to be confirmed by the work of other scholars [11]. On the other hand, one of the key points stressed inside the AWS policy is the shared responsibility model [12]. Since AWS and its customers share control over the IT environment, security is not entirely a duty of Amazon, but it is a responsibility shared with its customers. When it comes to managing security and compliance in the AWS Cloud, each party has distinct responsibilities. For this reason, the programmers must not underestimate their role in keeping their application developed through AWS secure.

Meanwhile, regarding IoT systems, the recent literature shows that IoT systems seem to be in a dangerous situation. According to the research activities of Kumar et al. [13], IoT systems are becoming more and more important in people's houses; approximately 40% of houses all around the world have at least an IoT device, and about 70% in North America. In their work, Kumar et al. analyzed an extensive data set of IoT devices (83M) in a considerable amount of real houses (15.5M). They discovered that a surprising number of devices still support File Transfer Protocol (FTP) and Telnet — both considered not secure nowadays. Furthermore, on a specific day, they perform an in-depth analysis of the data coming from the users who are actively using *Avast Wi-Fi inspector*[3]). They find out that 62% of the scanning houses contained at least one known vulnerability. In another recent study [14], Kafle et al. demonstrate a lateral privilege escalation attack on a cloud IoT environment (Google Nest[4]). In their study, they explain as, quite often, the insecurity of the platform is related to errors of third-party programmers. Even if Nest is trying to keep its platform secure through a review process, they do not execute this review on "new" applications (i.e., applications downloaded by less than 50 users) that are more likely applications developed by novice programmers.

Even if other scholars are studying countermeasures to the weaknesses of IoT systems (e.g., through sharp Intrusion Detection Systems [15]), we decided to conduct this work to understand better what is the real *security perception* of novice IoT programmers. In particular, we are interested in getting relevant insights about the possibility of helping them design more reliable IoT systems from the very beginning.

[2] https://aws.amazon.com/security/, last visited on February 26th, 2021.
[3] https://www.avast.com/internet-security, last visited on February 28th, 2021.
[4] https://store.google.com/category/google_nest, last visited on February 28th, 2021.

3. Use Case

During the end of 2020, our research group was teaching a professional training course for a consulting engineering company in Turin, Italy. The course's main goal was to teach a small group of programmers how to develop a Cloud-IoT-based application from scratch. This course started by introducing the IoT world, explaining possible advantages, disadvantages, and challenges. It also explained one of the most used protocols in this field: Message Queuing Telemetry Transport (MQTT). After this part, the programmers were introduced to cloud computing technologies starting from a general perspective. Subsequently, they were introduced to Amazon Web Services (AWS), focusing on cloud computing for IoT. In the end, the software developers learned some additional concepts related to how to develop a web server using the Representational State Transfer (REST) approach.

The course was organized in 4 non-consecutive days (one day per week), with theoretical sections in the mornings and practical laboratories in the afternoons. To conclude the course, each attendee has to develop a full-working prototype with all the components shown in Figure 1. This prototype had to be presented in a final lecture.

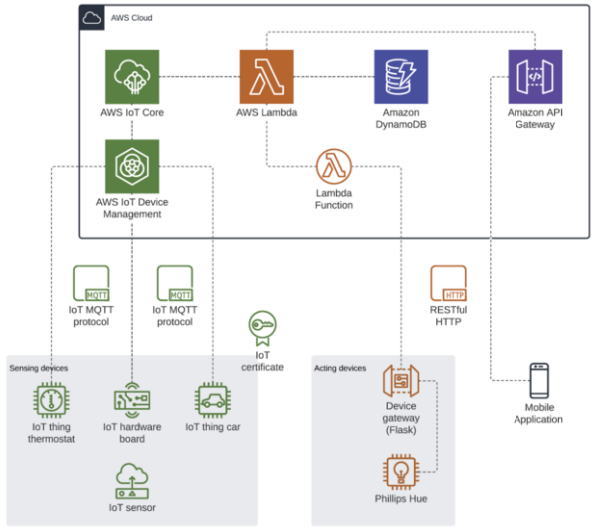

Figure 1. The architectural schema of the use case architecture.

The main components of the project could be divided into four different categories:

- **Sensing devices**: devices used to retrieve some data from the physical world;
- **Acting devices**: devices used to act on the physical world (e.g., a smart lamp);
- **AWS Cloud**: a cloud back-end server able to manipulate and store all the data necessary for the application;
- **Front-end devices**: used to interact with the back-end (e.g., a *mobile application*).

3.1. Main Architecture Attack Points

Considering the proposed the architecture (Figure 1), we defined the main attack points according to our knowledge and to the state of the art of security issues in the IoT field (e.g., [13] [14] [15] [16]). At the end of our analysis, we identified these five main attack points in the use case architecture:

1. the data flow between a *sensor* and the *AWS back-end*;
2. the data flow between *AWS back-end* and a *physical actuator* (e.g., a smart lamp);
3. the data flow between the *AWS API Gateway* and the *user's device*;
4. the developed code stored inside *AWS back-end* (e.g., AWS Lambda component);
5. the *back-end database* offered by Amazon (i.e., Dynamo DB).

For instance, a possible attack that could occur on all the *data flows* could be a Man-In-The-Middle (MITM) attack[5] or a Replay attack[6]. Furthermore, a possible issue, if the database is not well protected and the data stored are not encrypted, could be a steal of information — that could be directly used by the malicious user or sold. Finally, our analysis's last attack point is the code on the AWS back-end, for example, on the Lambda component. This component is probably the most critical point. If the code on the back-end of the application is compromised, the attacker will become able to do whatever she wants with the developed system.

4. AWS Analysis

In this section, starting from the architecture shown in Figure 3, and from the points presented in Section 3.1, we have identified how a malicious user can conduct these attacks, and we verified what countermeasures are used by AWS components.

4.1. Data Flow Protection

Three of the attack points highlighted in Section 3.1 are related to data flows. During transmission, data could be potentially eavesdropped, tampered with, and forged. The most common way to counter these dangers is by using cryptography on the transmitted data. According to AWS documentation [10], users must establish all the connections with the AWS back-end using *HTTPS*. This protocol is an extension of traditional HTTP using the Transport Layer Security (TLS) protocol to encrypt the data flow. For this reason, HTTPS is also called *HTTP over TLS*. TLS is a widespread protocol based on asymmetric cryptography. At the current time, TLS is considered a reliable and secure protocol, so the Amazon approach is coherent with the state-of-the-art. Furthermore, if a user requires additional security, AWS offers the possibility to create an Amazon Virtual Private Cloud (VPC) and use IPsec to contact the VPC.

[5] An attack in which a malicious user intercepts and modifies the data meanwhile they are being transmitted.
[6] An attack in which a valid data pack is stored and sent by a malicious user in a different moment.

4.2. AWS Back-End Protection

To have access to AWS Back-end services (like AWS Lambda) users must be authenticated. AWS provides two different types of user account: *root* user and *Identity and Access Management* (IAM) user. The root user is created by a customer when she registers for the first time to AWS. It is the account with the highest privileges. The root account can create several IAM accounts and assign them the necessary privileges (e.g., the right of writing on a particular database or the right to use a specific service). Even in this case, registration and login data flows are protected using HTTPS. Indeed, assuming that AWS implemented the registration and login processes correctly, the main possible failure points are two: an *inadequate privileges policy* and *bad passwords*.

An example of a good privileges policy is the *Principle of Least Privilege* (POLP). This principle's idea is that any user, program, or process should have only the *bare minimum* privileges necessary to perform its function. Furthermore, it is a good habit not to share an account between multiple users. Each user should have their accounts. Even if Amazon strongly recommends using the root account only for creating IAM accounts (and for other very few tasks), this best practice is not enforced at all. From our perspective, AWS should force its customers to create at least one IAM account. This approach will help new users to understand this security concept better.

Instead, the password policy enforced by AWS for root and IAM accounts is quite robust. Currently, it forces the users to create passwords with a minimum length of 8 symbols and at least two of the following three characteristics:

- including both lowercase and uppercase characters:
- including a number;
- including a non-alphanumeric symbol.

Unfortunately, we noticed that AWS seems not to have taken any particular countermeasures for dictionary attacks (for example, a password like "Amaz0nWS" is correctly accepted when registering a new account). On the other hand, for the IAM accounts, there is the possibility of let AWS auto-generates the password. The generated passwords have a length of 16 characters and include all the characteristics previously cited.

4.3. AWS Back-End Database Protection

Amazon provides different database solutions (both relational and NoSQL), and it offers database-related services as data-warehouse. In particular, in our use case, the developers have to use DynamoDB, a NoSQL database service that provides fast and predictable performance with seamless scalability. Amazon DynamoDB is accessible via TSL-encrypted endpoints.

By default, the DynamoDB service encrypts all data at rest to enhance data security. A customer can use the default encryption, the AWS-owned Customer Master Key (CMK), or the AWS-managed CMK to encrypt all data. DynamoDB also offers support to switch encryption keys between the AWS-owned CMK and AWS-managed CMK.

To avoid the risk of losing data, a user can set up automatic backups using a particular template in AWS Data Pipeline that was created just for copying DynamoDB tables. AWS offers the possibility to decide between full or incremental backups in the same or in a different region.

To control who can use the DynamoDB resources and API, the user has to set up permissions in the IAM service. Through IAM policy, a user can also specify a fine-grained access control policy (e.g., to allow or deny access to specific rows or columns). Additionally, each request to the database must contain a valid HMAC-SHA256 signature. HMAC (Hash-based Message Authentication Code) is an authentication code to be sent together with the request, generated using SHA256 (Secure Hash Algorithm 256), a cryptographic hash function belonging to the SHA-2 family. Even in this case, this could be considered a state-of-the-art approach. The AWS Software Developer Kits automatically sign user's requests; however, each user can write their HTTP requests proving the signature in the header of the requests.

5. Developer Perspective on AWS Security

The purpose of this section is to understand if the lack of knowledge of a novice programmer could be compensated or not by the AWS countermeasures cited in Section 4. To do this, we prepared a survey starting from the use case presented in Section 3. In this section, we will present the survey structure and the answers provided by our participants.

5.1. Structure of the Survey

At the very end of the course, after the final review of the projects — when the students are already aware if they successfully passed the exam or not — we asked the attendees to fulfill a survey. Thanks to these survey results, we have an initial idea of the perception of the security issues during the development of a Cloud-IoT system from a novice programmer's point of view, focusing on AWS.

We divided the survey into three distinct sections: "background and individual studying", "possible attack points", and "countermeasures and best practices".

In the first section, we asked the participants how much they think to be cybersecurity experts, how much they think that security is important in IoT systems, and who is in charge of the security of something developed on AWS infrastructure. Then, we showed Figure 1 as a reminder of their systems' architecture. We demanded the attendees to think about the possible security issues of the architecture and how they managed these issues during the development (if they did not manage the issues, we also asked why).

In the section "possible attack points", we introduced what we already explained in Section 3.1. We required the students to sort the attack points from the easiest to the hardest to attack and according to the possible severity if an attack succeeded. Furthermore, the programmers had to specify the worst possible consequence if a malicious user attacks the most critical point and, to conclude, how many of those points they considered while developing their projects.

In the last section, we wanted to understand if they tried to take some countermeasures (e.g., robust passwords, different accounts with different privileges, encryption algorithms, etc.) and if they are aware that AWS provides some components to help them manage IoT security (e.g., AWS IoT Device Defender[7]).

[7] https://aws.amazon.com/iot-device-defender/, last visited on February 27th, 2021.

5.2. Survey Results

In total, the course had 9 *novice IoT programmers*; we were able to collect answers from 6 of them (so our survey had an answer rate of 67%). All the attendees were male.

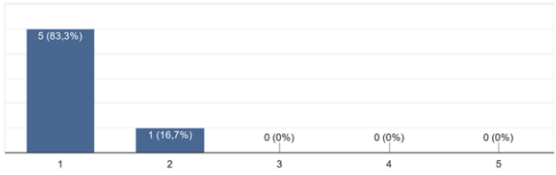

Figure 2. Novice programmers' competence perception about cybersecurity.

From one side, our programmers feel very inexperienced about cybersecurity; on a scale from 1 to 5, 5/6 participants answer 1, while the other select 2. On the other side, they think that cybersecurity is quite important, and it depends on the severity of the implemented software solution. All the novice programmers think that the implemented architecture could include a security issue, but no one said to have done something to mitigate the highlighted problems. At the question related to whom is in charge of the security of what is developed on AWS, 4/6 participants answer "both developer and AWS". Instead, 2/6 consider the responsibility *entirely of the developer*.

The novice programmers consider the AWS database the most secure component among the presented attack points. In contrast, the data flows between the AWS back-end and the sensors/actuators are considered the less secure points. The most critical point in case of a successful attack is the data flow from the back-end to the actuators, followed by the developed Lambda functions on the back-end and the data flow from sensors to the back-end. 5 participants consider as the worst possible consequence a cyber-physical attack: a security breach in cyberspace that impacts the physical environment (e.g., activating or deactivating a machine). They are mainly afraid that an attacker can take control of the system to damage a machine or a person. The second biggest concern seems to be data loss. The novice programmers consider potentially attackable at most 2 of the 5 attack points (3/6 answers). At the same time, 2 participants did not think to anyone of the presented attack points during the development phase.

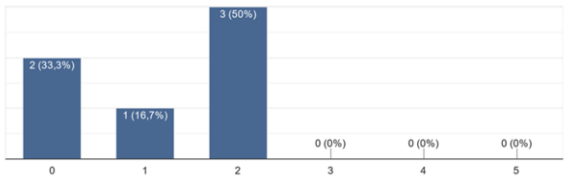

Figure 3. The number of attack points considered attackable during the development phase.

All the participants created a strong password (probably because AWS enforces it), but only 1 developer have used the service for creating an IAM account, while all the others use only the root account. Furthermore, only 2 participants specify that they did not create IAM accounts but are aware that they should had. Talking about data flows

cryptography, 4 novice programmers did not check if the platform uses TLS for encryption or not. Only 1 developer verifies that TLS is used on the data flow from the sensors to the AWS back-end. Finally, regarding the data stored inside DynamoDB, 5/6 participants did not think if they were encrypted or not. To conclude, nobody thinks about using an additional tool provided by Amazon to improve his solution's security. Only one participant said to have heard about *IoT Device Defender*, but he decided not to use it for the time constraint.

6. Discussion

It is interesting to notice that even if our group of programmers thinks that cybersecurity is quite critical, they do not feel to be experts in this field. Perhaps, it is a consequence of this perception if they did not pay too much attention to security requirements during their solution development. Furthermore, nobody thought to use an additional tool to compensate for their lack of knowledge and enhance their solution's robustness (e.g., using AWS IoT Device Defender). All the participants thought that the architecture could include security issues, but no one acted to mitigate the problem. This is a little bit strange considering that for two respondents, the solution's security is entirely the responsibility of the developer (meanwhile, for all the others is equally divided between the developer and AWS). We could try to explain this lack of action by considering that this project is just the outcome of a training course, and cybersecurity aspects were not the focus of our course.

The AWS database is considered the most secure point in the architecture. This perception could be quite dangerous considering that most programmers do not take care of database encryption. Even if AWS takes care of data at rest, programmers should remember to properly manage critical information (e.g., users' passwords). Our use case does not include users' registration, but properly hashing users' passwords is crucial in a more complex application. According to our participants, the less secure points are the data flows from and to the AWS back-end. They are mainly apprehensive about cyber-physical attacks. Nevertheless, 4 programmers did not think about the security of the communication channels. Even in this case, Amazon does not allow connection not protected with TLS, so, fortunately, a lack of competence does not create a potential attack point.

We noticed that thanks to AWS policy, all the participants had created a strong password. On the contrary, almost no one had used AWS IAM service to create separate accounts with different privileges. Using a unique account for all the possible operations could create many problems if a malicious user owns the password. From our perspective, this issue could be mitigated if AWS enforce users to create at least an IAM account. A tutorial phase dedicated to this particular issue could help novice programmers understand why it is important to use a good privilege policy like Least Privilege's principle.

7. Conclusions

From this preliminary analysis, we can conclude that AWS seems to be a good choice for implementing a secure Cloud-IoT solution even for a novice IoT programmer. The

platform implements concepts at the state-of-the-art, and it has many useful tools to start developing. AWS is quite able to compensate for the lack of knowledge of the novice developers involved in the study. Our main suggestion is related to the accounts' policies. We believe that forcing users to create separate accounts through the AWS IAM service could be a good starting point to improve the novice programmers' awareness of good privileges policies.

We strongly believe that security-by-design is the best way to improve the reliability of every system. To achieve this goal, the primary security principles should be inside programmers' minds since the beginning of the development cycle. Thanks to this study, we understood that novice IoT developers tend to consider their system's security as an important feature but to manage it as an additional component.

In our future works, we will have a more focused survey on a larger sample of novice IoT programmers to understand better this phenomenon's dimension. Later on, we would like to provide some best practices and tools to help such novice programmers develop much more reliable IoT systems.

Acknowledgment

We want to acknowledge the employees who voluntarily decided to participate in the survey to conduct this research activity.

References

[1] Bala R, Gill B, Smith D, Wright D, Ji K. Magic Quadrant for Cloud Infrastructure and Platform Services. Gartner Inc.; 2020. Available from: https://www.gartner.com/doc/reprints?id=1-1ZDZDMTF&ct=200703&st=sb.

[2] Inc AWS. Introduction to AWS Security. 410 Terry Avenue North Seattle, WA 98109 United States: Amazon Web Services Inc.; 2020. Available from: https://d1.awsstatic.com/whitepapers/Security/Intro_to_AWS_Security.pdf.

[3] Soloway E, Spohrer JC. Studying the novice programmer. 365 Broadway, Hillsdale, New Jersey 07642 United States: Lawrence Erlbaum Associates, Inc.; 1989.

[4] Lahtinen E, Ala-Mutka K, Järvinen HM. A study of the difficulties of novice programmers. Acm sigcse bulletin. 2005;37(3):14–18.

[5] Corno F, De Russis L, Sáenz JP. Pain Points for Novice Programmers of Ambient Intelligence Systems: An Exploratory Study. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC). vol. 1. IEEE; 2017. p. 250–255.

[6] Corno F, De Russis L, Sáenz JP. Towards Computational Notebooks for IoT Development. In: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems; 2019. p. 1–6.

[7] Varga S, Brynielsson J, Franke U. Cyber-threat perception and risk management in the Swedish financial sector. Computers & Security. 2021;p. 102239. Available from: https://www.sciencedirect.com/science/article/pii/S0167404821000638.

[8] Carroll M, Van Der Merwe A, Kotze P. Secure cloud computing: Benefits, risks and controls. In: 2011 Information Security for South Africa. IEEE; 2011. p. 1–9.

[9] Behl A, Behl K. An analysis of cloud computing security issues. In: 2012 World Congress on Information and Communication Technologies. IEEE; 2012. p. 109–114.

[10] Inc AWS. Amazon Web Services: Overview of Security Processes. 410 Terry Avenue North Seattle, WA 98109 United States: Amazon Web Services Inc.; 2020. Available from: https://d0.awsstatic.com/whitepapers/aws-security-whitepaper.pdf.

[11] Narula S, Jain A, et al. Cloud Computing Security: Amazon Web Service. In: 2015 Fifth International Conference on Advanced Computing & Communication Technologies. IEEE; 2015. p. 501–505.

[12] Taggart M, Roach B, Woods P. Amazon Web Services: Risk and Compliance. 410 Terry Avenue North Seattle, WA 98109 USA: Amazon Web Services Inc.; 2020. Available from: https://d1.awsstatic.com/whitepapers/compliance/AWS_Risk_and_Compliance_Whitepaper.pdf.

[13] Kumar D, Shen K, Case B, Garg D, Alperovich G, Kuznetsov D, et al. All things considered: an analysis of IoT devices on home networks. In: 28th USENIX Security Symposium (USENIX Security 19); 2019. p. 1169–1185.

[14] Kafle K, Moran K, Manandhar S, Nadkarni A, Poshyvanyk D. Security in Centralized Data Store-Based Home Automation Platforms: A Systematic Analysis of Nest and Hue. ACM Transactions on Cyber-Physical Systems. 2020 Dec;5(1):1–27.

[15] Anthi E, Williams L, Słowińska M, Theodorakopoulos G, Burnap P. A supervised intrusion detection system for smart home IoT devices. IEEE Internet of Things Journal. 2019;6(5):9042–9053.

[16] Huang DY, Apthorpe N, Li F, Acar G, Feamster N. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies. 2020;4(2):1–21.