Enhancing the Reliability of IoT Data Marketplaces through Security Validation of IoT Devices

Yoonjong Na, Yejin Joo, and Heejo Lee
Department of Computer Science and Engineering, Korea University
Seoul, South Korea
{nooryyaa, yjkellyjoo, heejo}@korea.ac.kr

Xiangchen Zhao, Kurian Karyakulam Sajan, Gowri Ramachandran, and Bhaskar Krishnamachari
Viterbi School of Engineering, University of Southern California
Los Angeles, USA
{zhao115, karyakul, gsramach, bkrishna}@usc.edu

***Abstract*—IoT data marketplaces are being developed to help cities and communities create large scale IoT applications. Such data marketplaces let the IoT device owners sell their data to the application developers. Following this application development model, the application developers need not deploy their own IoT devices when developing IoT applications; instead, they can buy data from a data marketplace. In a marketplace-based IoT application, the application developers are making critical business and operation decisions using the data produced by seller's IoT devices. Under these circumstances, it is crucial to verify and validate the security of IoT devices.**

In this paper, we assess the security of IoT data marketplaces. In particular, we discuss what kind of vulnerabilities exist in IoT data marketplaces using the well-known STRIDE model, and present a security assessment and certification framework for IoT data marketplaces to help the device owners to examine the security vulnerabilities of their devices. Most importantly, our solution certifies the IoT devices when they connect to the data marketplace, which helps the application developers to make an informed decision when buying and consuming data from a data marketplace. To demonstrate the effectiveness of the proposed approach, we have developed a proof-of-concept using I3 (Intelligent IoT Integrator), which is an open-source IoT data marketplace developed at the University of Southern California, and IoTcube, which is a vulnerability detection toolkit developed by researchers at Korea University. Through this work, we show that it is possible to increase the reliability of a IoT data marketplace while not damaging the convenience of the users.

***Index Terms*—IoT, security, data marketplace**

I. Introduction

With the advent of IoT data marketplaces, the cities and the communities around the world can build data-driven IoT applications involving tens of data sources [1]–[4]. Multiple IoT device owners can make their data available for the application developers via such IoT data marketplaces. In this model, the device owners are responsible for managing the hardware and software on their IoT edge devices, and let the application developers focus only on paying for and receiving the desired data from the marketplace.

An IoT data marketplace typically comprises of IoT end-devices at the south end, application developers at the north end, and the marketplace middleware in the middle [3]. Following this model, the data generated by the end-devices flows via the marketplace middleware to the IoT applications. As a result, the application developers have to trust the data producers and the data marketplace to make business and policy decisions. Therefore, security analysis of IoT data marketplace should consider both the IoT end devices and the marketplace middleware, and analyze how a security breach in one component may affect the other components in a marketplace-driven IoT ecosystem.

Although the marketplace middleware is acting as a mediator, the data produced by the IoT edge-devices is exceptionally critical for the operation of data marketplaces. When the IoT device owners fail to secure the hardware devices that generate the data, the devices may become susceptible to cyberattacks from malicious actors. Such attacks may compromise the integrity of the end-device, which would, in turn, hamper its data quality while damaging the reputation of the device owner. Note that the application developers may also be making critical business decisions based on the data produced by the end-devices. Compromised devices may lead to wrong or sub-optimal outcomes reducing the effectiveness of IoT data marketplaces. Therefore, the security of the IoT end-devices is essential to the operation of the marketplace, which is the focus of this work.

In this work, we discuss the security vulnerabilities of IoT data marketplaces and present a solution to validate the security vulnerabilities of IoT edge devices that connect to a marketplace. Our security analysis follows a well-known STRIDE security assessment framework to understand the types of threats residing in the IoT data marketplace. Besides, we introduce mechanisms to assess the security of IoT edge devices and issue a certificate based on the vulnerabilities found on the devices. We have developed a proof-of-concept using I3 data marketplace, which is an open-source IoT data marketplace developed at the University of Southern California, and IoTcube, which is a vulnerability detection toolkit developed by researchers at Korea University. Our security assessment framework not only certifies the edge-devices but also informs the extent of the security vulnerabilities to the application developers when buying data from the marketplace. To the best of our knowledge, this is the first security validation framework targeted towards IoT data marketplaces.

Of course there are operational security concerns that need to be dealt on end-devices' level, such as access control loopholes, misconfiguration issues and so on, but such cases

are not in the scope of this study. According to the Software Engineering Institute, about 90% of security issues occur due to software vulnerabilities [5]. Therefore, we can say that this work covers a significant pool of security concerns even if operational security is not in the scope.

The remainder of this paper is structured as follows. Section II provides background on data marketplace and security validation. The security assessment of the data marketplace is discussed in Section III. Section IV explains our security assessment framework. Section V presents our proof-of-concept implementation and evaluation results. Section VI concludes the paper.

II. Background on IoT Data Marketplaces, I3, and IoTcube

A. Data Marketplaces

Data marketplaces, including Intelligent IoT Integrator (I3) [3], Ocean Protocol [6], and IOTA Data Marketplace [7], are being developed to enhance the adoption of IoT in smart communities and smart cities. Such data marketplace initiatives focus on building "data rivers" that allow data streams from different entities to be merged, analyzed, processed, and acted upon as needed to support a diverse set of applications [3]. In a commercial aspect, the data marketplace allows edge device owners to sell data streams. At the same time, different applications can buy one or more data streams to develop applications for smart communities. In this work, we consider I3, which is an open-source data marketplace [8] for smart cities developed at the University of Southern California.

B. I3: Intelligent IoT Integrator

I3 is a data marketplace developed at the University of Southern California, in partnership with the City of Los Angeles and a number of academic and industrial partners. It consists of a marketplace middleware, which is responsible for managing users, devices, and data products. Here, the edge device owners and the application developers are considered as the users. When the edge device owners register their data product in the marketplace, they are required to create a hub and manage their devices and data products under it. I3 also allows the application developers to buy data by selecting their desired products from the marketplace registry. Upon buying the data product, the marketplace provides a credential to let the buyer access the data. I3 uses MQTT as a messaging protocol to let the seller exchange the data with the buyer. Note that the marketplace middleware authorizes access to MQTT for sellers and buyers based on their account status. In the case of buyers, I3 authorizes only those who present a valid credential. Fig. 1 shows the building blocks of the I3 data marketplace - version 1, which is the latest version of the marketplace.

The I3 data marketplace is available as open-source software to help the researchers and marketplace enthusiasts[1]. To help the edge device owners and application developers send data to and receive data from the data marketplace, a software development kit is also made available [2].

C. IoTcube

IoTcube[3] is a comprehensive vulnerability testing environment offering black-box, white-box, and network testing. From these features, white-box testing can be used for assessing the security of IoT data marketplaces. This white-box testing is based on a well verified vulnerable code clone detection (VUDDY) [9] technique. Code clone is a fragment of code that is copy-and-pasted from other software codes. Nowadays, a lot of developers produce code clones by copying codes into their software codes when a function they need is well implemented in there. However, this means if the copied code fragment contains a vulnerability, then it propagates to other software, creating the same vulnerabilities. VUDDY solves this problem through a simple, safe, and fast algorithm that detects code clones and their vulnerabilities.

To mitigate such code clone vulnerabilities, analyzing well-known vulnerability in such devices is a fast and effective way. Several databases record what kind of vulnerabilities are in the open-source software. CVE (Common Vulnerabilities and Exposures) is one of the most widely used vulnerability databases. CVE is a dictionary of publicly disclosed cybersecurity vulnerabilities and exposures that is free to search, use, and incorporate into products and services [10]. Since CVE is open to the public and is built through collective intelligence by accepting vulnerability reports from various reliable sources, it contains a lot of data.

IoTcube's vulnerable code clone detection rapidly discovers vulnerabilities in new versions of software code and updates them in IoTcube's vulnerable code database. Also, IoTcube's database has been double-checked and proved by credible sources, and its detection rate has low false positives.

D. Gap

On the one hand, existing data marketplace software, including I3, enables the device owners to register their IoT devices and sell data products. Still, it doesn't provide any solutions to assess the security vulnerabilities of the IoT end-devices. On the other hand, IoTcube delivers a tool for the IoT application developers to check their software before the field deployment, where the onus is on the device owners to assess their software for security vulnerabilities independently.

There is no marketplace architecture with built-in support for vulnerability detection and device certification, which is the focus of this work.

III. Security hazards in IoT Marketplaces

A. An overview of IoT Security

The IoT security has been discussed extensively in the literature [11]–[15]. Soteria [11] discusses the security vulnerabilities of IoT devices and presents an automated security

[1] https://github.com/ANRGUSC/I3-Core

[2] https://github.com/ANRGUSC/I3-SDK

[3] http://iotcube.net/

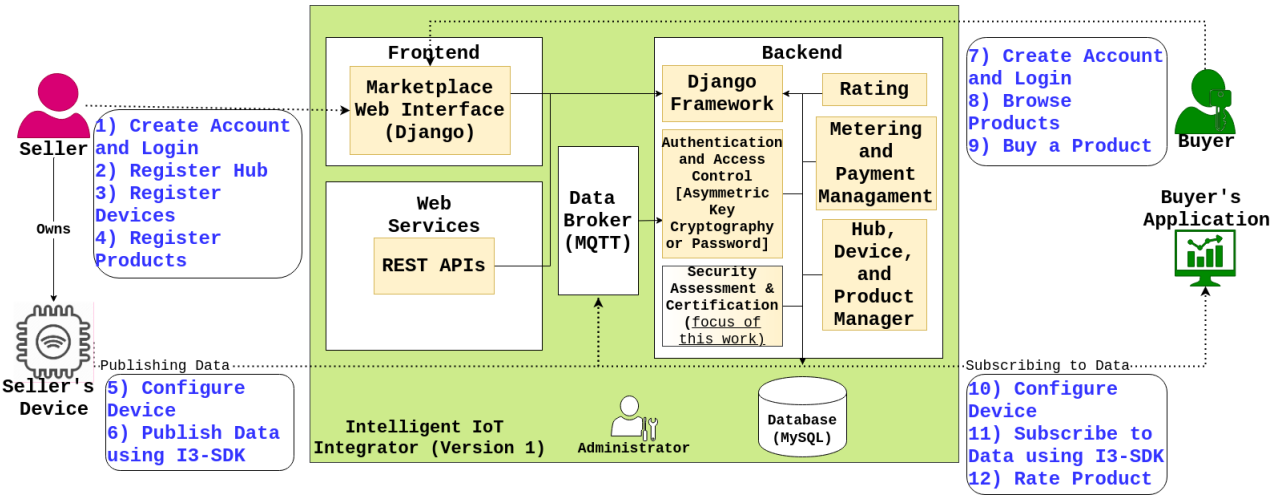

Fig. 1: Architecture of I^3-v1 [8]. Contemporary data marketplaces lack support for security validation and certification. In this work, we will extend the I3-v1 data marketplace by integrating security assessment and certification features.

analysis platform. Mavropoulos *et al.* [12] presents ASTo, a tool to analyze the threats and vulnerabilities of IoT devices and software. A security analysis testbed and an architecture for analyzing the IoT system has been presented in [13]. All these efforts highlight the security issues in the IoT devices, while Soteria [11] and ASTo [12] present methods and tools to identify security issues. However, such solutions only encourage the researchers and the practitioners in the IoT domain to take actions to mitigate the threats imposed by malicious attackers. Our work focuses on connecting such security assessments to the IoT data marketplace to allow automatic identification of security vulnerabilities and certify devices based on the level of threat.

B. Security threats in IoT data marketplaces

To build a trusted marketplace, we have to investigate the kind of threats that exist in it and how such threats impact its stakeholders, including the device owners, system administrators, and application developers. To that end, we classify the marketplace data flows into two categories, as shown in Fig. 2:

- **Data path between the seller (that is, IoT end-devices) and the marketplace:** This data flow considers the security issues that arise between the edge devices' and the marketplace middleware.
- **Data path between the marketplace and the buyer:** This data flow focuses on the interaction between the data marketplace middleware and the application developers' devices.

Both of these data flows and the software involved in the data exchange process could be susceptible to cyberattacks. To study the security vulnerabilities of IoT data marketplace, we choose the STRIDE threat model [16] for our security analysis. We choose STRIDE threat model because the IoT data marketplace possesses various security requirements, and the STRIDE threat model is tailor-made for such environment. STRIDE defines six threat categories; spoofing identity, tampering with data, repudiation, information disclosure, denial of service, the elevation of privilege [16].

C. Threat modeling

Fig. 2 shows two types of data flows and associated threats for the IoT data marketplace. The following list explains the security threats that may happen on IoT data marketplaces:

1. **Spoofing**. The device owners have to register their devices at the data marketplace and get credentials to publish data. Malicious users may steal the credential from a device and send data to a marketplace by impersonating a device. For the flow between the seller and the marketplace, this attack could severely damage the reputation of the device owner while disrupting the data quality and the effectiveness of the entire data marketplace. Regarding the data flow between the marketplace and the buyer, a buyer could buy data sent by attacker to marketplace with stolen credentials.
2. **Tampering**. Malicious users may manipulate software services, device drivers, and other critical system elements. For the data flow between the seller and the marketplace, an attacker could destroy the integrity of the seller and the device by tampering with the data and the software. Regarding the data flow between the marketplace the buyer, the application developer may not be able to reliably process the data if an attacker tampers with the data, hardware, or the software.
3. **Repudiation** When an attack happens on a device, logs provide activity traces through which malicious actions

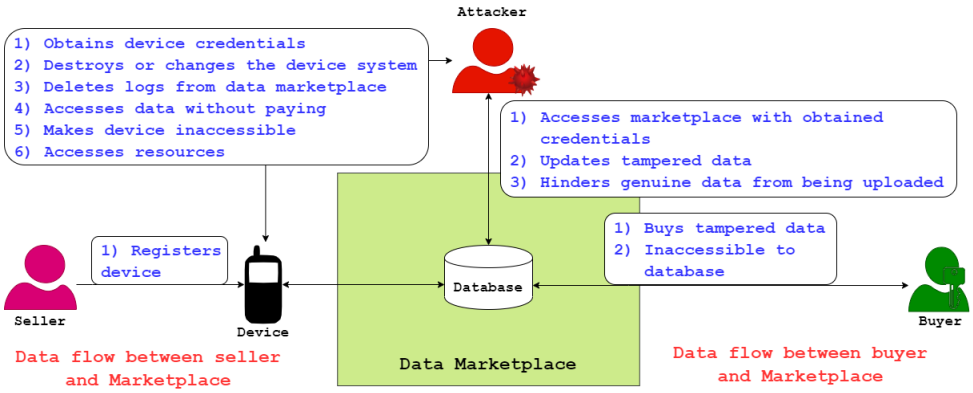

Fig. 2: IoT Marketplace Threat Model

can be captured. But, malicious users may delete important logs, including any information received from the data marketplace.

4. **Information Disclosure** Malicious users may steal confidential information from the device including the seller's and buyer's credential to either impersonate the device or get access to the data from the marketplace without paying for it.
5. **Denial of Service** Malicious users may make the device unavailable preventing the seller from sending data to the marketplace by exploiting network and software vulnerabilities.
6. **Elevation of Privilege** Malicious users may take over the device and do actions such as read, write, or access to systems or resources that they should not be accessing to.

As shown in this threat model, the security vulnerabilities of the devices may affect buyers, sellers and also the data marketplace. Most importantly, these threats heavily affect the edge devices owned by the sellers. Therefore, we focus on assessing the security of the edge devices in this work.

D. Examples of Security Vulnerabilities

In this section, we review some examples of security vulnerabilities that could affect the IoT devices, which could, in turn, impact the IoT data marketplaces. BlueBorne is a type of security vulnerability in Bluetooth protocol. If an IoT device uses an old version of the operating system or Bluetooth protocol that is vulnerable to BlueBorne, a malicious user may access, leak information, eavesdrop data, and do many other unauthorized actions from the device. If we apply this vulnerability to the IoT data marketplace, a malicious user may access data that is published to the marketplace. DirtyCow is another type of security vulnerability from old versions of the Linux kernel that allows a malicious user to gain a higher level of system authority. If an IoT device has both BlueBorne and DirtyCow vulnerabilities, a malicious user may access the device and execute DirtyCow exploit code using BlueBorne vulnerability and gain higher level permissions on the device. Eventually, the malicious user can destroy the device, modify the application running on the device, make the device inaccessible, and many other things.

It is crucial to identify what kinds of vulnerabilities reside in devices to mitigate such threats. By knowing this, the device owner or the application developer can prevent malicious users from exploiting their devices. If there is a device using an old version of Linux kernel, for example, there exists both BlueBorne and DirtyCow vulnerabilities. The application developer or device owner may not know about the vulnerability. However, by analyzing device vulnerabilities when registering it to the data marketplace, they may find the vulnerabilities that reside in the device and then patch the device with the stable version of the software. However, finding unknown vulnerabilities from the device takes not only time but also lots of efforts that may discourage developers. Thus using vulnerable code clone detection tools such as VUDDY is useful for finding vulnerabilities from devices. By using such tools, the device owners can easily find known vulnerabilities.

IV. Enhancing Marketplace Security through Vulnerability Detection

From section III, it is clear that the IoT data marketplace must provide support for security validation of the devices. In particular, securing the IoT end-devices becomes critical to protect the integrity of the devices while encouraging the application developers to trust the data provided by the marketplace. In this section, we will discuss our security assessment and certification mechanisms.

A. Selling and Buying Process in a Data Marketplace

For the integration of security assessment mechanisms, it is essential to understand how the sellers and buyers interact with the data marketplace. A seller will have to register and sell their data using the data marketplace, while a buyer will have to register and pay for the wanted product. The interaction process described below is based on the I3-v1 data marketplace [8], but closely resembles other commercial data marketplaces, including Streamr and Terbine.io.

From sellers' point of view, they have to

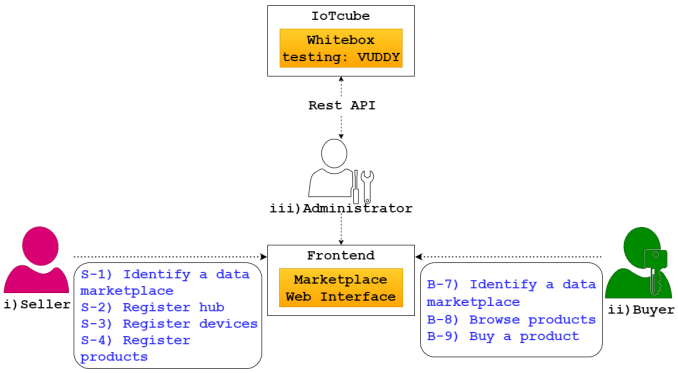

Fig. 3: Data Flow in the IoT Marketplace and its Vulnerability Validation

S-1) identify a data marketplace platform that helps them sell data. Here, the assumption is that a local community or city administration deploys and manages a marketplace middleware to help the device owners connect with the application developers.

S-2) register the device by creating a hub for their organization.

S-3) register their device under their hub. In this phase, the seller can select between password-based and public-key based authentication mechanism to establish a secure connection with the marketplace middleware.

S-4) create the data product associated with the device. During the product registration phase, the seller is required to enter information about the data product, including what type of data it produces, where the device is physically deployed by entering its GPS coordinates, and the price of the data.

At the end of the product registration phase, the device owner will receive credentials to start publishing data from his/her device to the marketplace middleware.

From the buyers' point of view, they have to:

B-1) identify a data marketplace platform just like a seller does.

B-2) browse through the products uploaded by various sellers.

B-3) buy the desired data product by paying the price set by the user.

After making a payment for the data, the buyer will receive a credential to establish a data connection with the marketplace middleware to start receiving the data from the seller's device.

This whole process is illustrated in Fig. 3. It shows that the device owners are not providing any information to assess the security of the device and the data. At the buyer end, the application developers are buying the data without verifying the security of the device that produces the data.

B. Design Choices

The marketplace architecture presented in Fig. 1 indicates the absence of a security assessment and certification mechanism. In this section, we will go through the details of how security validation can be done by the seller, buyer, and the marketplace administrator.

a) seller: If vulnerability validation is done on the seller's side, the seller would have to validate their devices on their own before registration. This job will be very troublesome for them, thereby elevating the entrance barrier to the marketplace. On top of that, the validation will be done only at the moment of registration, leaving them exposed to newly occurring vulnerabilities, or the sellers are required to validate the security of their devices frequently.

b) buyer: On the other hand, if vulnerability validation is done on the buyer's side, the buyer would have to assess for security on their own before buying a product. And, the buyer needs information such as the operating systems, software services, device drivers, and other software that the seller is running on his/her device to assess the security vulnerability reliably. The seller may not be willing to share this information with the buyers due to privacy concerns, or it may help a malicious buyer to exploit the seller's device by exploiting the known vulnerabilities.

c) administrator: If the system administrator does vulnerability validation, all of the above troubles would be solved, given that the assessment process is not too big nor heavy for the administrator's system. And, the seller is already trusting the data marketplace. Besides, the marketplace alerts the seller whenever new vulnerabilities are reported to the public databases such as CVE.

When considering these pros and cons, the security validation process should be handled by the administrator. We explain our architecture in the next section.

C. Validating the Security of IoT Devices in IoT Data Marketplaces

Our security validation feature extends the marketplace interaction process presented in section IV-A. In particular, we extend the device registration process with additional fields, wherein the device owner is required to enter their device model and kernel versions. Recall from the device registration

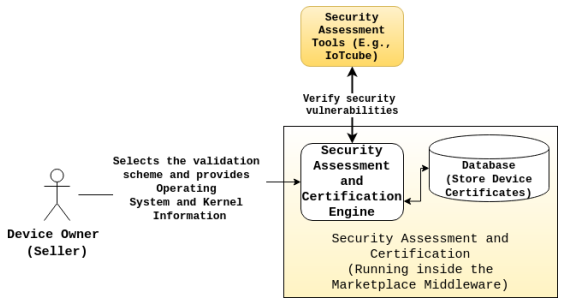

Fig. 4: Integration of Security Assessment and Certification Engine to a Data Marketplace

process described in section IV-A that the device owner only selects the authentication mechanism during the registration process.

Fig. 4 shows the architecture of the proposed security assessment and certification feature for the IoT data marketplaces. During the device registration phase, device owners are required to select the desired validation scheme and provide information about the operating system and kernel versions. The security assessment engine then connects with a third-party tool such as IoTcube to verify the security vulnerabilities. Based on the outcome of the security assessment, the certification scheme issues a certificate following the rules presented in TABLE I. Since forcing the security analysis to users may make them dither over joining the marketplace, user-friendly UI and easy-to-follow guidelines are essential to help the sellers adopt this security assessment feature (Fig. 5 shows how this can be implemented in a data marketplace). The security analysis is divided into two parts, as follows:

1. **Weak Validation**. Validate security only with some essential information, including the device model and the kernel version.
2. **Strong Validation**. Validate security by providing the source code.

For the first method, weak validation, users who want to enroll their device to the marketplace provides the information on two things: OS kernel version and device model. Then, the marketplace middleware will provide a preprocessed security analysis report with the information provided. One drawback of using this method is that there is a higher chance of false-positiveness. But this method is valid for users who cannot furnish the source code of the device or who think code level analysis is a nuisance. Also, if the user wants a deeper level of vulnerability scan after the weak validation, they may go through the strong validation, which is explained below.

For the second method, strong validation, users need to provide their devices' source code for a more in-depth scan. Here, vulnerability detection tools can provide more in-depth scan results of vulnerabilities. This method is valid for users who want a deeper level of vulnerability scan and who want to know more specifically about the vulnerabilities currently residing in their devices.

TABLE I: Certificate level criteria for weak validation and strong validation

Certificate Level	Weak Validation	Strong Validation
Level 5	-	No severe vulnerabilities and named vulnerabilities
Level 4	No severe vulnerabilities and named vulnerabilities	Either severe vulnerabilities or named vulnerabilities
Level 3	Either severe vulnerabilities or named vulnerabilities	Both severe vulnerabilities and named vulnerabilities
Level 2	Both severe vulnerabilities and named vulnerabilities	-
Level 1	Device not analyzed	

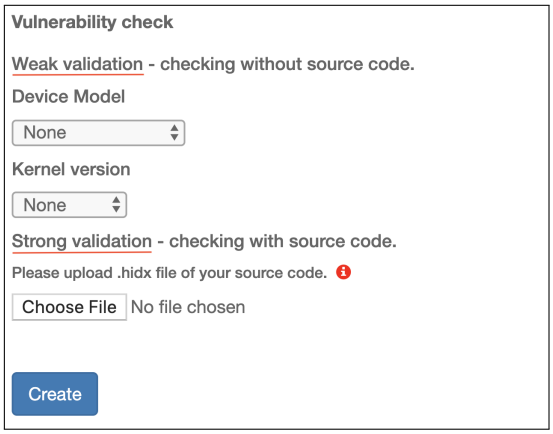

Fig. 5: Implementation of Security Validation During Device Enrollment Phase

After the security validation process, the marketplace middleware issues a certificate attesting the security level of the device, based on the information provided by the device owner. TABLE I shows the certificate levels. As shown on the table, we gave more incentives to strong validation than weak validation. The vulnerability has been considered severe if the Common Vulnerability Scoring System (CVSS) score for the vulnerability was higher than 7.0. CVSS is measured and given to most of CVE entrances, thereby giving a good reference point for the level of severity. Named vulnerabilities are vulnerabilities that affected a wide range of computer systems with high impact that a name has been given. The vulnerabilities mentioned in section III-D are such vulnerabilities.

V. Proof of Concept Implementation and Evaluation

We have developed a proof-of-concept implementation using I3 (see section II-B), and IoTcube (see section II-C).

As shown in Fig. 5, we extended the I3 data marketplace's frontend to let the device owner enter his operating system and kernel version during the device registration phase. Upon receiving the necessary information, if the device owner opted for a weak validation, the marketplace middleware provides vulnerability information based on preprocessed assessment data through IoTcube. In the case of strong validation, the seller will be required to download and use hmark, which is a vulnerability detection tool provided by IoTcube. This tool will process the seller's source code and produce a file hashes of the source code in hidx format. The marketplace middleware will receive this hidx file from the seller, which will be sent to the IoTcube's security validation engine through its REST-API. The engine will then run a check against its database and return the known vulnerabilities.

The marketplace middleware generates a security certificate based on the number of vulnerabilities identified in the device. When a buyer browses through the marketplace, he/she can make a decision based on the security level.

TABLE II: Performance Evaluation on 6 Popular IoT Operating Systems

OS Name	Version	LoC[a]	Time 1	Hidx size	Analysis time
Raspbian	rpi-4.1.y	14,448,394	110m39s	32M	0.9322s
	rpi-4.19.y	18,127,058	139m6s	41M	1.0688s
RIOT	2017.01	177,663	3m28s	488K	0.0909s
	2020.01	372,754	6m43s	1.1M	0.1023s
TizenRT	1.1_P_R[b]	1,291,458	13m18s	3.3M	0.1462s
	3.0_GBM	2,533,436	26m34s	6.5M	0.1966s
Huawei LiteOS	1.1.1	13,916	13s	28K	0.0807s
	[c]50B039	599,624	8m1s	1.2M	0.1009s
ARM Mbed OS	5.2.0	1,671,606	7m13s	3.1M	0.1295s
	5.15.1	3,686,895	31m16s	7.7M	0.2085s
Contiki	release-3.0	283,285	2m12s	529K	0.0921s
	release-v4.4	257,168	1m48s	478K	0.0908s

We use IoTcube-based vulnerability detection method because it is easy to use, lightweight, and fast. To further explain, the detection tool can be used in any common environment settings, available for Windows, Linux, and OSX. It is also very light: about 100MB for Windows and about 10MB for Linux and OSX. Since it is easy to use, it solves the problem of high entrance barrier. On top of that, as it is light, it allows the marketplace platform to handle a lot more jobs. Details on the evidence of fastness will be explained in the section V-B.

A. Evaluation

Here, we analyzed and evaluated the performance and effectiveness of the implementation described in section IV-C. This evaluation was done with six popular IoT devices' operating systems: Raspbian, RIOT, TizenRT, Huawei LiteOS, ARM Mbed OS, and Contiki. The most recent stable version and one of the older versions from more than two years back have been selected. As for the environment, a machine with Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz and 16GB RAM running Ubuntu 18.04.4 LTS was used.

B. Performance

As explained in section V, the security analysis is done when the seller enrolls his/her device, and it can be done in two ways: weak and strong. In the case of weak validation, it does not take any time to show the validation results. It is because the OS versions and device models' vulnerability information is already evaluated by IoTcube and saved in the marketplace middelware, ready to show as soon as the device is enrolled. To evaluate the time it takes for strong validation, we need to consider the following two sections:

1. **Time 1:** time to convert source code into hidx,
2. **Time 2:** time to send the hidx file through IoTcube API and get the result back.

Time 1 can be evaluated by calculating the time hmark conversion tool takes. Time 2 depends on the size of the hidx file and the time it takes for IoTcube to analyze the received hidx. The results are shown in the following TABLE II.

[a]Short for Lines of Code
[b]Short for Public_Release
[c]Abridged "V200R001"

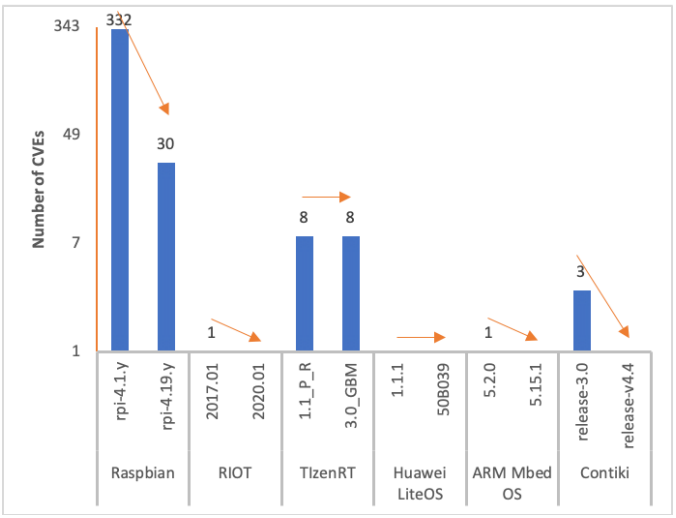

Fig. 6: Vulnerabilities in 6 Popular IoT Operating Systems This bar graph shows that the number of CVEs tend to reduce when the version is more recent. However, there are some exceptions.

From the results, we can conclude that the validation process is swift overall. This fact has also been proven through the comparison with other existing tools in VUDDY's Evaluation section [9]. On top of that, as explained in section V, our proof-of-concept security validation method is easy to use and lightweight. Such factors have a decisive effect on user experience.

C. Effectiveness

Here, we will check the effectiveness of weak and strong validations. The Fig. 6 shows the number of CVEs detected from analyzing the test subjects with IoTcube's white-box testing. The results show that, in general, the more recent versions have fewer vulnerabilities. The newer versions come out with security fixes on previously reported vulnerabilities, thereby reducing flaws in the latest release versions.

However, there are some singularities to this tendency. In the case of Raspbian, the number of CVEs decrease dramatically. As the version name implies, Raspbian uses Linux Kernel. Code similarity analysis provided by IoTcube proves this, showing that rpi-4.1.y version of Raspbian (A) and 4.1 version of Linux Kernel (B) are almost the same software, showing similarity level of $(A \cap B)/A = 97.93\%$ and $(B \cap A)/B = 99.23\%$. Since Linux Kernel is a widely used open source operating system kernel, thousands of vulnerabilities have been reported and recorded in the CVE database. As one of the oldest open source project, tens of thousands of developers have been contributing to this project for more than a decade, discovering and fixing issues. On top of that, fatal vulnerabilities directly affecting the kernel such as DirtyCow and BlueBorne broke out in 2016 and 2017 consecutively, raising the alarm of big security loophole issues for the kernel

developers. For such reasons, Linux kernel continues to get more secure as years go by, thus explaining the dramatic drop of number of CVEs in Raspbian.

In the case of TizenRT, it keeps its number of CVEs. The first thing to note is that all of their vulnerabilities come from third-party libraries, meaning they are all code clone vulnerabilities. When looking into them in detail, we found out that one of the vulnerabilities from version 1.1_Public_Release had been fixed in version 3.0_GBM. Still, a new vulnerability appeared from another third party library. More specifically, the fixed vulnerability from version 1.1_Public_Release was from Jouni Malinen's hostapd library's wps_common.c code. In version 3.0_GBM this vulnerability is fixed, but a new vulnerability appears in nghttp2 external library, in nghttp2_frame.c code.

In the case of Huawei LiteOS, no vulnerabilities have been detected in either versions. In the earlier version 1.1.1, as it is shown by the Lines of Code in TABLE II, the OS is still at an early stage of building, and does not use any third-party libraries. In the V200R00150B039 version, the volume has increased, but we found out that this version still used only two third-party libraries. This means unlike other open source projects, Huawei LiteOS does not use much of code clones. Furthermore, no vulnerabilities from Huawei LiteOS itself had been reported into CVE database. Such are the reasons why no vulnerabilities have been detected in either versions.

The overall evaluation shows that checking through weak validation is effective and the results are much more accurate when the current source code the seller is using is assessed for vulnerabilities. From here we can conclude that code clone detection fits well with IoT data marketplaces for real world implementation both in performance and effectiveness aspect.

VI. Conclusion

IoT data marketplaces are gaining traction in the context of smart cities. The community-driven operational model of the IoT data marketplace allows the device owners to sell their data to the application developers. In this work, we have analyzed the security of IoT data marketplaces and shown that the security of the IoT devices is one of the critical determinants of trust in a marketplace-based application. Besides, a vulnerability detection solution has been presented to inform the security vulnerabilities of IoT end-devices to the sellers. We have also presented a proof-of-concept implementation by integrating I3, which is an open-source IoT data marketplace developed by the University of Southern California, with IoTcube, which is a well-known vulnerability detection tool developed by the Korea University.

Furthermore, we have also shown how our proof-of-concept implementation verifies the software used by the IoT devices and provide a security certificate based on the number of vulnerabilities and their severity levels. To understand the overhead added by our software validation process, we have carried out evaluations using well-known IoT operating systems, including Contiki and RIOT. We have shown that our proof-of-concept analyzes the security vulnerabilities in hundreds of milliseconds while helping the device owners and application developers to make an informed decision and that our security model and proof-of-concept is applicable in real world IoT device marketplace.

Acknowledgment

This work is supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2019-0-01697, Development of Automated Vulnerability Discovery Technologies for Blockchain Platform Security and No.2019-0-01343, Regional strategic industry convergence security core talent training business) and the USC Viterbi Center for Cyber-Physical Systems and the Internet of Things (CCI).

References

[1] G. S. Ramachandran, R. Radhakrishnan, and B. Krishnamachari, "Towards a decentralized data marketplace for smart cities," in *2018 IEEE International Smart Cities Conference (ISC2)*, pp. 1–8.

[2] K. R. Özyilmaz, M. Doğan, and A. Yurdakul, "Idmob: Iot data marketplace on blockchain," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 11–19.

[3] B. Krishnamachari, J. Power, S. H. Kim, and C. Shahabi, "I3: An iot marketplace for smart communities," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys, 2018, pp. 498–499.

[4] C. Perera, "Sensing as a service (s2aas): Buying and selling iot data," *arXiv preprint arXiv:1702.02380*, 2017.

[5] U.S. Department of Homeland Security, "Software Assurance," [Online]. Available: https://www.us-cert.gov/sites/default/files/publications/infosheet_SoftwareAssurance.pdf, accessed on June 2020.

[6] Ocean Protocol Foundation, "Ocean Protocol: A Decentralized Substrate for AI Data and Services," [Online]. Available: https://oceanprotocol.com/, bigchainDB GmbH and DEX Pte. Ltd, Tech. Rep., March 2018.

[7] IOTA Foundation, "IOTA Data Marketplace," [Online]. Available: https://data.iota.org, accessed on September 2019.

[8] X. Zhao, K. Karyakulam Sajan, G. Ramachandran, and B. Krishnamachari, "Demo abstract: The intelligent iot integrator data marketplace — version 1," in *Proceedings of the 5th ACM/IEEE Conference on Internet of Things Design and Implementation*, ser. IoTDI, 2020.

[9] S. Kim, S. Woo, H. Lee, and H. Oh, "Vuddy: A scalable approach for vulnerable code clone discovery," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 595–614.

[10] "Common Vulnerabilities and Exposures (CVE)." [Online]. Available: https://cve.mitre.org/

[11] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: Automated iot safety and security analysis," in *2018 USENIX Annual Technical Conference (USENIX ATC)*, Jul., pp. 147–158.

[12] O. Mavropoulos, H. Mouratidis, A. Fish, and E. Panaousis, "Asto: A tool for security analysis of iot systems," in *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, pp. 395–400.

[13] V. Sachidananda, S. Siboni, A. Shabtai, J. Toh, S. Bhairav, and Y. Elovici, "Let the cat out of the bag: A holistic approach towards security analysis of the internet of things," in *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, ser. IoTPTS, 2017, p. 3–10.

[14] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the iot world: Present and future challenges," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483–2495, 2018.

[15] F. M. Tabrizi and K. Pattabiraman, "Design-level and code-level security analysis of iot devices," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 3, May 2019.

[16] M. Howard and S. Lipner, *The security development lifecycle*. Microsoft Press Redmond, 2006, vol. 8.