AN ANALYSIS OF SOFTWARE DEPLOYMENT SECURITY FOR INTERNET OF THINGS DEVICES

Ioana Culic[1], Alexandru Vochescu[2], Răzvan Rughiniș[3]

The adoption of Internet of Things in commercial and industrial systems exposes products to a new set of security issues, as what used to be isolated devices are now connected to the Internet. In this context, the need for updates infrastructures dedicated to IoT becomes crucial in maintaining security. The aim of this paper is to analyze the challenges and the characteristics specific to IoT deploy and update solutions with a focus on the security aspect. We extract the main characteristics that make such an infrastructure secure and propose them as pillars for any Internet of Things-specific deploy system.

Keywords: Internet of Things, deployment, updates, security.

1. Introduction

With the advent of the Internet of Things (IoT) era, the number of embedded connected devices has increased exponentially. Until the large adoption of IoT, embedded computers were used mainly in industrial systems. However, as smart washing machines and vacuum cleaners entered people's homes, more and more embedded devices were integrated in a variety of appliances and everyday products.

One of the main changes the IoT revolution brought in the industry is that computing devices that were once designed to work in closed and monitored environments (e.g. embedded computers for assembly lines or industrial robots) are now embedded into end user gadgets and systems that are connected to the Internet and to other third-party services. From a security point of view, this exposes the microcontrollers and embedded computers to a series of security issues that were not relevant before.

In this context, device updates regarding security issues and security improvements have become extremely important for all IoT producers and

[1]Teaching Assistant, Politehnica University of Bucharest, Romania, e-mail: ioana_maria.culic@upb.ro

[2]PhD Student, Politehnica University of Bucharest, Romania, e-mail: alexandru.vochescu@upb.ro

[3]Professor, Politehnica University of Bucharest, Romania, e-mail: razvan.rughinis@upb.ro

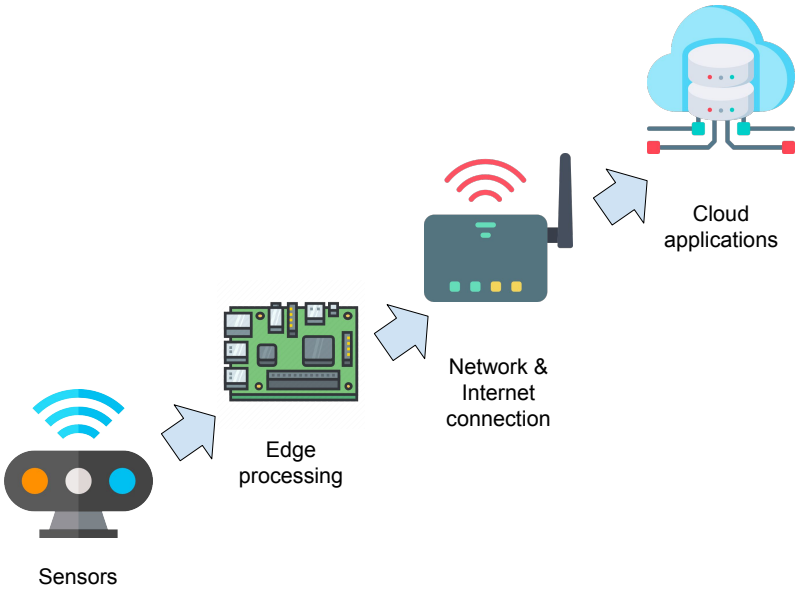
integrators [1]. What is more, other bug fixes or software improvements are also key aspects dependent of a deploy and update infrastructure [2].

As most IoT devices are designed to run autonomously, installed in various locations such as customers' homes, agricultural fields, or vehicles, the mandatory request for any IoT software deploy and update infrastructure is to support wireless data transfer. Therefore, most updates system in this field will refer to an over-the-air (OTA) solution. Tocmai de aceea e problema de bricking si locking.

2. Hardware Taxonomy

Internet of Things systems are complex and rely on multiple communication and implementation layers. This is why industry and research has defined an IoT stack that we illustrate in Figure 1.

FIGURE 1. The Internet of Things stack.



Each layer is defined by different processing and memory requirements, which reflects on the varied hardware and software platforms employed. With this in mind, we present a taxonomy of the existing hardware included in IoT infrastructures to emphasize the different classes of deploy solutions.

2.1. Microcontrollers

The hardware equipment deployed at the first layer of the IoT stack consists of microcontrollers. These are specialized, small-size devices usually characterized by reduced processing power and memory in order to optimize

energy consumption and ensure long operation life. This makes them suitable to be easily integrated into industrial equipment, sensors, home-use appliances, wearables, etc. with the purpose of gathering data from the environment and executing physical actions (e.g., unlock doorknobs, move robot arms, turn on the lights, etc.) [3].

In terms of remote application deployment, microcontroller devices are associated with a large number of scattered devices to be programmed and managed. What is more, most of these devices do not support any operating system and run only one piece of firmware, albeit some of the more powerful ones can run microcontroller-specific operating systems such as FreeRTOS [4], RIOT OS [5], Contiki [6] and other similar platforms that orchestrate individual tasks.

2.2. Embedded Computers

On top of the sensing layer lie embedded computers that serve two main purposes: assure connectivity among the end nodes and between the end nodes and the cloud, and perform edge processing. These are small-sized computers with higher processing power and memory compared with microcontrollers. Hence they can implement secure communication protocols and run full-fledged operating systems and software applications.

From the deploy infrastructure point of view, embedded computers have the capability to implement complex communication protocols and security policies. What is more, these devices usually have a stable Internet connection to the cloud layer, intermediating the data transmission between end nodes and the cloud. Therefore, deploy systems targeting embedded computers benefit from extra resources and can integrate with more varied technologies.

2.3. High-Performance Computers

The top of the IoT stack consists of cloud services used to store and process large amounts of data. At this layer, the processing is done using clusters of high-performance computers on top of which run various software applications. The current technological advancements in most of the IT fields rely on cloud technologies, making this a heavily explored layer. Software deploy solutions related to the cloud are provided by mature companies such as Microsoft, Amazon, and Google, just to name a few.

In this paper we aim to explore the deploy infrastructures dedicated to the devices situated at the bottom of the IoT stack where environmental data is gathered. Thus we will focus on solutions built for microcontrollers and edge devices.

3. IoT Software Deployment and Updates Challenges

While deploy and update systems are regularly used by all IT fields, these infrastructures dedicated to IoT solutions have special requirements. Despite

the apparent simplicity, IoT systems consist of large networks of heterogeneous sensors and gateway devices on top of which run data-driven decision-making mechanisms. All these components and layers are tightly coupled and one bug or security breach in one of them can propagate and impact the entire network. Considering the complexity and the dynamic nature of the Internet of Things systems, application deploy and update architectures are different from general purpose or mobile applications updates. Therefore, several aspects specific to IoT need to be handled in order to build become suitable tools that can be used to enable commercial success [7].

3.1. The Hardware and Software Heterogeneity Specific to IoT

Internet of Things architectures are heterogeneous infrastructures from the hardware, software, and communication point of view. This impacts the mechanism of deploying new applications or updates, particularly for a remote deployment process. For each layer in the IoT stack, several various technologies and capabilities are employed [8]. Therefore, when implementing an IoT system, decisions have to be made about:

- the hardware components employed by the IoT architectures,
- the communication protocols used to ensure data transmission at different stack layers,
- the operating systems addressing varied requirements,
- the integration of newly developed programming languages and run-times,
- the adoption of development and management platforms that rely on diverse architecture models, methodologies, and APIs.

At the hardware layer, the system needs to run on different hardware architectures such as ARM or x86. What is more, when taking a closer look, at the local processing layer, many different variants for the hardware are available. This impacts the deployed firmware or software, which needs to be compiled separately for each architecture. The hardware capabilities might also need to be considered, depending on whether the deployment targets firmware, operating system features, or application updates [9].

In addition to the existing hardware and software variety, and the new standards and technologies that make the development process volatile and difficult to maintain - another critical factor to consider is the amortisation cost - while newer solutions and technologies emerge often, many commercial deployments must account for the installed hardware solutions for whom the investment has not been recovered yet. This is why firmware and software updates are of great importance and help deliver IoT systems that can continuously improve and increase their technical qualities, over existing hardware, while also presenting novel features to their users, making them more appealing. However, all the variables make the development of a generic deploy solution difficult. In consequence, many IoT producers choose to build custom

solutions, adapted to the specific hardware they use, rather than use a generic platform.

3.2. Resource and Location Constraints

Internet of Things technologies have also been deployed in remote use cases including agriculture, weather forecasting, or gas extraction. In these scenarios, sensors are deployed in remote, difficult to reach places (e.g., on top of windmills, on top of antennas, scattered across large fields) and they are designed to work uninterrupted and unserviced for long periods. In this case, any update solution that requires direct access to the device is unfeasable. Other relevant side effects are that such devices are hard to access for manual reboot or replacement in case there is a software error during or after the update process or that the chances to have an unstable network connection are high.

What is more, all IoT devices are designed to be economic from the power consumption point of view in order to preserve battery. This implies that the deployed hardware has specific constraints such as low-memory or low processing power. These characteristics are also relevant for the updates infrastructure that needs to be efficient and not to bring a significant software overhead [10].

3.3. Security

Since IoT applications are complex systems developed on multiple layers, the security attack surface related to them is extremely wide, targeting the node, gateway, cloud, and also the application deploy infrastructure. At the hardware layer, attacks such as malicious node insertion and hardware exploitation are known to have compromised multiple commercial sensor-based IoT platforms [11]. Further on, we can identify security breaches related to the connectivity characteristic of these devices. As IoT applications rely on information sharing across and between multiple layers, these infrastructures include communication servers (e.g., SSH, MQTT, Apache, NGINX) that enable specific data transmission protocols. These technologies and protocols are known to be heavily exploited security-wise and a plethora of data transit and denial-of-service attacks targeting IoT deployments have been identified [12]. Furthermore, most of the gateways included in the IoT device networks are running Linux-based distributions, where the kernel is admitted as being complex and heavily exploited. Hundreds of new vulnerabilities related to the Linux kernel are identified yearly, and as stated by the Linux community, updates are necessary to ensure the security of these systems [13].

4. Requirements of IoT Update Systems Based on the Application Family

Based on the challenges analyzed above related to the development and maintenance of commercial and industrial IoT systems, we have defined certain characteristics we consider essential for the development of an effective IoT deploy and updates system. Considering the complexity and variety of the Internet of Things systems, we believe a classification based on the deployed application's purpose is required. In this respect, we identified distinct attributes in the modeling of hobby, commercial, or industrial purpose deploy platforms (table 1).

The hobby community refers to professional and non-professional users who use state of the art hardware and software technologies to prototype simple IoT projects. Their purpose is to either tinker with various technologies in order to get familiar with them or to validate the functionality and usage of a certain application.

In contrast, commercial and industrial applications are built to be commercialized and deployed on a large scale. In these cases, the development process is longer, well documented and structured with the purpose of achieving stable and robust end products. The focus, here, falls on testing and maintaining the deployed products. In addition, for industrial applications, they need to be certifiable and built with several specific standards in mind.

Table 1. IoT deployment purpose and main requirements.

Type	Monitoring	Wireless	Rollback	Secure	Certifiable	Planned
Hobby	x					
Commercial	x	x	x	x		
Industrial	x	x	x	x	x	x

4.1. Specific Requirements for Hobby Applications

The development process of IoT applications by the maker community is primarily characterized by utility and rapidity. In this specific case, the developers aim to implement relatively simple applications with the purpose of testing, evaluating, and familiarizing themselves with diverse technologies targeting different layers in the IoT stack (e.g., different hardware platforms, communication infrastructures, cloud services) [14]. Based on this hypothesis, we identified the following characteristics as essential for a hobby-specific deploy platform.

4.1.1. *Accessible Setup Process.* The hardware and software technologies developed for the hobbyist market are designed with a focus on accessibility and

usability. On top of this, a key aspect is the setup process that consists of interconnecting the hardware and software layer and establishing compatibility-related configurations [15].

In the case of deploying new software on the prototyping hardware platform, the setup process needs to be fast and address generic use-cases with a small number of configurations to be made. A universal software solution that is capable of seamlessly integrating a wide variety of libraries, programming languages, communication protocols, and APIs is required. As a result, users should be able to quickly shift between various technologies and hardware platforms.

4.**1**.2. *Speed.* The process of flashing and running the updated software on the prototyping embedded device needs to happen fast, with little overhead [16]. At this point, when the hardware platform is in proximity to the user, the deploy solution's focus is to efficiently transfer and run the new application on the device, even to the detriment of implementing certain checkups and advanced security policies.

4.**1**.3. *Application Monitoring.* Most of the deployments done in the process of building an IoT prototype are for testing and debugging purposes, thus debug and other similar log messages are used to monitor the system's behavior. Having access to an interface dedicated to classifying and displaying various application reports and messages is important. While there are many tools specially designed for application monitoring and debugging, in this use-case, we consider it is important to have this feature integrated into the main deploy solution and exclude the overhead related to switching between different development platforms.

4.**2**. Specific Requirements for Commercial Applications

Deploying and updating applications for commercial devices requires a notably different approach from the one presented above. In this case, the update is delivered on a platform that is already in use and which is deployed on the customers' premises or as part of a larger system of sensors and gateway devices. The purpose of the deployment is also different from the one in the above section. Updates for commercial product applications are made for delivering new features, fixing bugs, or security breaches [17]. Therefore, we propose the following characteristics for an update system designed to deploy applications onto varied commercial hardware devices.

4.**2**.1. *Wireless Communication.* The main characteristic of commercial devices is that they are deployed on the customers' premises, be them people's homes or remote agricultural locations. Therefore, servicing these products is expensive if updates need to be made through a physical connection to the device. In this context, the principal characteristic of an updates infrastructure dedicated to commercial IoT applications is to support wireless data transfer

and communication. Another consequence is the need for a complex platform that can also support the remote monitoring and diagnosis of the deployed devices. Thus, the need for a physical intervention should be necessary only in case of major hardware or software failures.

4.**2**.2. *Application Versioning & Rollback.* The versioning of the software deployed on the devices is a mandatory requirement for all updates platforms, no matter their purpose. However, in the case of commercial IoT products, this is defined in close relation with the possibility of performing full rollback to a previous version [18].

This necessity is another consequence of the remote placement of the devices and the increased possibility of an unstable network connection. This can result in interrupted update processes and in this case, the system should automatically rollback to the previous application version and keep on functioning. What is more, manual rollbacks should also be performed in case bugs or other malfunctions are related to the latest software version.

4.**2**.3. *Security.* Ever since its launch, the major concern of the IoT field is security. As we are now referring to an explosion in the number of connected devices, the risks they are exposed to has increased exponentially. Any device that was initially designed to run in an isolated environment and that is now connected to the network has a major increase in security risks and exposure to potential security attacks.

In this context, software deploy infrastructures for such devices are also a potential target for malicious activities. Any software that interacts remotely with the deployed device opens a new possibility for an attack. Therefore, OTA updates systems need to be heavily secured, from the point of storing the data to be deployed, to the network communication and how the data reaches the end device. For this, security policies and regulations need to be implemented, making this a mandatory requirement for any IoT deploy and update infrastructure.

4.**3**. Specific Requirements for Industrial Applications

When analyzing the specifications and requirements of software and hardware equipment designed for industrial use, the criteria are more rigid and clearly standardized. This also reflects on the characteristics we define for the software update solution. While the commercial-specific requirements detailed above are equally relevant for an industry-grade platform, we also need to consider other extra demands specific for an industrial model.

4.**3**.1. *Certifiable.* In the case of industrial technologies, certifications are mandatory. This also applies to any software updates solution that is intended to be used for deploying applications on industrial equipment [19]. Therefore, depending on the exact field of usage, specific evaluations and procedures have to be passed in the development and testing process. The solution needs to

comply with precise standards and respond to external parameters with specific values, all with the purpose of certifying it as proper for the intended usage.

4.**3**.2. *Stability.* Any industrial technology needs to be robust and stable, capable to run uninterrupted for prolonged periods, and in harsh conditions. Although a software platform is not affected by environment parameters such as humidity and temperature, it is required however to withstand demanding input-output operations coupled with highly intensive processing. This requires the platform to be stable and carefully tested so that the probability of failure is within industry-specific limits.

4.**3**.3. *Planned Updates.* In factories, the assembly lines maintenance and upgrades are attentively planned so the downtime is reduced to a minimum. In the case of software updates, the process needs to be predictable and aligned with the global upgrade procedure. The system used for the software updates needs the capability to predict the exact amount of time required for a successful deployment up until the moment the new version is running on the devices. The launch-time (date and hour) of the update operation has to be planned and introduced in the system.

5. **Secure IoT Software Updates Solutions**

As software deploy and updates is an important part of any IoT infrastructure, many solutions have been proposed by both research and commercial communities, all with a focus on security and data integrity. However, most solutions proposed in the research literature are generic approaches that have never been implemented into a concrete use-case. In this context, we analyze the most popular approaches of implementing secure and reliable updates infrastructures for IoT devices with a focus on commercial solutions that have been widely used by the industry.

Table 2 outlines the main characteristics of the solutions we detail below.

Table 2. IoT deploy solutions comparison.

	Ubuntu Core	Mender	Balena
Open-Source	x	x	partially
Kernel updates	x	x	x
Isolation for security	x	-	x
No proprietary store	-	x	-
Automatic rollback	x	x	x

5.1. Ubuntu Core

Ubuntu Core is built on top of the Ubuntu Linux distribution and the snap package manager to enable application deployment and updates on IoT devices.

In this framework, the main application isolation mechanism is based on permissions that are implemented using snap connectors [20]. As each application is assigned a specific Linux user and data folder, it can write data in its own folder and only in some clearly defined parts of the file system outside, where the user has write permissions. Some of these writes are not traceable and cannot be rolled back (e.g.: write on the SD card).

Devices having an Internet connection can be remotely updated via the snapcraft store. This requires that application developers have an Ubuntu One account. However, the protocol implemented by the snapcraft store is open source and can be extended so users can opt out of linking the devices to the Ubuntu store.

Software deployment is made in a transactional manner and only partial updates are supported. Both applications and kernel parts can be updated, but this requires several independent updates.

5.2. Mender

Mender is an open source IoT application development and deploy system based on Yocto Linux [21].

It enables robust updates based on a dual partition mechanism which ensures the device keeps functioning even if the update fails. This is possible as each device has an active and a passive partition. The new deployment is made on the passive partition, while the active one is running. Once the new version is available on the passive partition, this becomes the active one. In case this is not possible (failed update), the system rolls back to the other partition. However, this requires the device to reboot for the new version to be in place. This enables the device to keep running even after a failed kernel update. The update process on the devices is handled by an Update Module, that is downloaded together with the new software version. This module handles all the update policies such as software rollback. As the Update Module is an independent executable, it can be customized to implement both kernel and application-level updates.

In the case of application updates, these can be made in an incremental manner, without the need for two different partitions. For this, the system introduces the notion of Mender artifacts which are used for partial updates. An artifact is an update packet that targets a specific application. Therefore, the updates can be packaged under various formats (e.g, docker image for containers, .deb for an application). An important advantage of Mender is that it allows for other package managers to be included in the architecture.

From the security point of view, each application needs to implement its isolation policies. This can be a major downside when several concurrent applications run on a device.

A key difference between Mender and the previously presented solutions is that it can be installed on the users' premises so the application developers are not locked in using a third-party cloud service.

5.3. Balena

Balena [22] is implemented in a similar manner to Mender using Yocto Linux distributions. In addition, Balena uses containers for application deployment. This also represents an advantage from the security point of view, as containers ensure process isolation. The container engine (balena) used in the implementation is a modified docker engine and allows differential docker layer updates. The advantage of this is that the bandwidth and time necessary to perform updates is reduced. However, this process slows down the building of the updates.

The platform is partially open source, but in order to use all its capabilities, developers need to register to the Balena cloud.

6. Conclusion

This paper focuses on the challenges and characteristics specific to the implementation of secure and robust deploy and update systems for Internet of Things devices. By analyzing the IoT implementation stack, we identify the heterogeneity of the IoT ecosystem as one of the main challenges in the development of such an infrastructure. In addition, there are multiple other characteristics specific to IoT development such as the remote device location or resource constraints, making the development of a secure updates infrastructure difficult.

Several commercial deploy and update systems dedicated for IoT infrastructures have taken into account these challenges and focused on proposing a secure and robust architecture. The main common characteristics of these platforms are the implementation of isolation policies, containerisation, application versioning, and dual partition mechanisms.

This analysis enabled us in identifying the main characteristics of IoT updates systems and will enable us in the further research for secure deploy infrastructures for the Internet of Things.

REFERENCES

[1] *N. Mtetwa et al.*, Secure Firmware Updates in the Internet of Things: A survey, 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC), 2019, 1-7.

[2] *J. Bauwens et al.*, Over-the-Air Software Updates in the Internet of Things: An Overview of Key Principles, IEEE Communications Magazine, **58**(2020), 35-41.

[3] *A. Mamta and R. Tiwari*, Smart Home Automation System Based on IoT using Chip Microcontroller, 9th International Conference on Computing for Sustainable Global Development, 2022, 564-568.
[4] *Amazon Web Services, Inc*, https://www.freertos.org, accessed: 05.07.2022.
[5] *Imprint*, https://www.riot-os.org, accessed: 05.07.2022.
[6] *Contiki*, https://www.contiki-ng.org, accessed: 05.07.2022.
[7] *A. Taivalsaari and T. Mikkonen*, A Roadmap to the Programmable World: Software Challenges in the IoT Era, IEEE Software, **34**(2017), 72-80.
[8] *H. Atlam and R. Walters and G. Wills*, Internet of Things: state-of-the-art, challenges, applications, and open issues, International Journal of Intelligent Computing Research (IJICR), **9**(2018), 928-938.
[9] *S. Jaouhari and E. Bouvet*, Secure firmware Over-The-Air updates for IoT: Survey, challenges, and discussions, Internet of Things Journal, **18**(2022), 1-12.
[10] *N. Safa et al.*, An opportunistic resource management model to overcome resource-constraint in the Internet of Things, Concurrency and Computation: Practice and Experience, **31**(2019).
[11] *T. Alladi et al.*, Consumer IoT: Security Vulnerability Case Studies and Solutions, IEEE Consumer Electronics Magazine, **9**(2020), 17-25.
[12] *A. Radovici and C. Rusu and R. Şerban*, A Survey of IoT Security Threats and Solutions, 17th RoEduNet Conference: Networking in Education and Research, 2018, 1-5.
[13] *H. Chen et al.*, Linux Kernel Vulnerabilities: State-of-the-Art Defenses and Open Problems, APSys '11: Proceedings of the Second Asia-Pacific Workshop on Systems, **5**(2011).
[14] *S. Mora and F. Gianni and M. Divitini*, RapIoT Toolkit: Rapid Prototyping of Collaborative Internet of Things Applications, 2016 International Conference on Collaboration Technologies and Systems (CTS), 2016, 438-445.
[15] *R. Matei and A. Radovici*, Remote management system for embedded devices: Wyliodrin, 15th RoEduNet Conference: Networking in Education and Research, 2016, 1-5.
[16] *H. Kondaveeti et al.*, A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations, Computer Science Review, **40**(2021).
[17] *H. Derhamy et al.*, A survey of commercial frameworks for the Internet of Things, 2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA), 2015, 1-8.
[18] *C. Gündoğan et al.*, RReliable firmware updates for the information-centric internet of things, Proceedings of the 8th ACM Conference on Information-Centric Networking, 2021, 59-70.
[19] *H. Boyes et al.*, The industrial internet of things (IIoT): An analysis framework, Computers in Industry, **101**(2018), 1-12.
[20] *J. Wyngaard*, Ubuntu Core Snaps for Science, AGU Fall Meeting Abstracts, **2017**(2017), 1-12.
[21] *R. Maheshwari and K. Rajalakshmi*, A Comparative Study of Deployment Tools from IoT Perspective, Advances in Computational Intelligence and Communication Technology, 2021, 293-306.
[22] *R. Botez et al.*, Containerized application for IoT devices: comparison between balenaCloud and Amazon web services approaches, 2020 International Symposium on Electronics and Telecommunications (ISETC), 2020, 1-4.