Chapter

Internet of Things Security and Privacy

Ahmad J. Showail

Abstract

The Internet of Things is becoming more and more popular with time. The extremely low cost of sensors is putting the growth of the Internet of Things on steroids. Many industries such as healthcare, construction, agriculture, and transportation are increasingly leveraging this technology. However, security and privacy are two big concerns when it comes to the future of the Internet of Things. Since most of these "things" that are connected to the Internet are simple devices with limited hardware capabilities, it is nearly impossible to harden them via traditional resource-heavy defenses. In this chapter, we discuss the importance of securing the Internet of Things networks, layout the challenges of the Internet of Things security, and briefly discuss potential solutions in the literature.

Keywords: cyber security, Privacy, Internet of Things

1. Introduction

Since the invention of the first ever network in 1972, computers are being connected using various topologies. Ranging from the traditional pair of wires found in industrial plants that connect sensors and actuators to the process control system, to state-of-the-art IPv6-enabled wearable sensors for medical monitoring, the idea is the same. You have a bunch of sensors that should talk to each other securely, whether on the Ethernet, Bluetooth, Zigbee, or a basic 4–20 milliamp electric circuit. We can safely say that IoT was born after the wedding of Information Technology (IT) and Operation Technology (OT). In fact, Fieldbus technology [1], which is one of the variants that is widely adopted in the industry, is a direct result of the advancements in OT that is trying to make the devices in the field 'smarter'. However, the challenge is how to get these variants to talk to each other, which requires a common ground of communication, such as the Internet Protocol (IP).

We can think of two types of Internet of Things (IoT) implementation, namely greenfield and brownfield [2]. Almost all the implementations in the fields that have no legacy using networked systems, such as health, agriculture, and transportation, are considered as greenfield IoT implementation. On the other hand, a brownfield implementation is the one trying to introduce internet devices in conjunction with traditional networked infrastructure, such as Fieldbus technology in the process automation industry. In both cases, a defined framework for communication is urgently needed to enhance system security and minimize the risk.

On the 21st of Oct 2016, a massive Distributed Denial of Service (DDOS) attack resulted in putting down the web services of famous companies such as Twitter, Amazon, Netflix, Airbnb, and GitHub, among others [3, 4]. The malware name was Mirai and it targets the DNS service provider called Dyn [5]. In the Japanese language, Mirai means 'Future" [6]. In reality, what Mirai did was simply switching Linux-based devices into digital weapons by exploiting the vulnerabilities of IoT devices, like factory default settings. In this specific attack, IP cameras and Digital Video Recorders (DVRs) were used to launch the attack. What makes this possible is the fact that many manufacturers of IoT devices leave the passwords hardcoded in the firmware, allowing hackers to easily connect to them using Telnet or Secure Shell (SSH).

In 2010, a specialized piece of code was developed to target nuclear plants. This malware was called "Stuxnet" [7]. Although the fact that most of the nuclear research centers employ the well-known air gap security mechanism, a poisoned Universal Serial Bus (USB) flash drive was used to infect the Programmable Logic Controller (PLC) responsible for the uranium enrichment centrifuges. As a result, the centrifuges suffered from physical damage due to faster than usual spins for extended periods, and abnormal acceleration and deceleration rates. Iran was not the only country that was affected by the attack. Other countries, such as India and Indonesia, were affected as well.

2. Building blocks of IoT security

Most of the network protocols are designed without having security in mind. Hence, network security is often the aftermath. Also, the heterogeneity of network components might be the gateway for predators to compromise the network. One way to solve this issue is to divide the devices in the network into two groups, trusted and untrusted devices. The former group is equipped with "root-of-trust" that could be as simple as an attestation key supplied by the manufacturer [8]. These trusted devices use secure communication mechanisms, namely secure key storage and cryptographic operations. Attestation protocols are used to assess whether untrusted devices are secure enough to join the trustworthiness group. The borderline between the trusted-devices group and the untrusted ones is usually invisible, as it is very difficult to classify these groups based on the type, manufacturer or even use. Moreover, the attestation process is usually dynamic and might change over time or in response to attacks on the network.

Root-of-trust is nothing but the set of trusted functionalities in the device that are assumed to be trustworthy and could never be compromised [9]. For example, the secure booting functionality of the device is a root-of-trust. Another example is the attestation functionality, which proves the validity of claims using cryptographic mechanisms. In fact, an IoT device might have multiple roots-of trust.

The ecosystem to establish trustworthiness in any IoT framework is composed of several building blocks, as shown in **Figure 1**. The first and foremost is the Trusted Execution Environment (TEE), which is responsible for executing trusted application codes and minimizing security risks. It is also responsible for isolating the process execution from other processes running on the same hardware. The second component is the secure communication channel, which is responsible for preserving the confidentiality and integrity of the data flowing between devices in the network using standard encryption mechanisms. The third component is the authentication process, including both the keys themselves, whether symmetric or asymmetric, and the actual key distribution and authentication protocol. The fourth component is the attestation process, involving the attestation key that is provided by the manufacturer, as well as the verification logic. After that, we must ensure that the devices are capable of securely storing all the keys and data gathered from the sensors. Finally, there is the ability to gather all relevant contextual information, such as location and time.

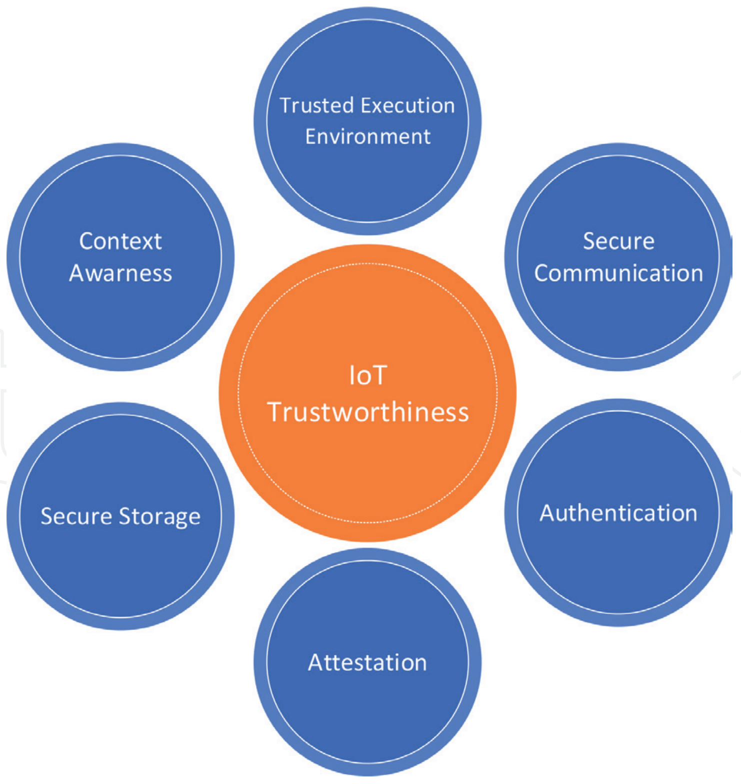


Figure 1.
Building blocks of IoT trustworthiness.

3. IoT device lifecycle

When discussing IoT security, we need first to understand the various stages in which the IoT device is going into overtime. **Figure 2** shows the typical lifecycle of an IoT device. It starts with the development of the software part of the IoT device using the Software Development Kit (SDK) or the Application Programming Interface (API) to hide the complexity. Then, tools are used for building the physical components of the device itself. We cannot stress enough the importance of the right configurations in the lifecycle of the IoT device. In this stage, various parameters are set in multiple components, such as the Central Processing Unit (CPU), the System on Chip (SoC), and the Operating System (OS). After that, it is the time for field deployment and making sure that the connection is established properly. Then, frequent updates are installed to protect the device. Finally, retirement is the reality that outdated devices must face.

4. End-to-end IoT security

The end-to-end concept is important when talking about the security of communication networks. **Figure 3** shows the main components involved in the end-to-end security journey of IoT device communication. Typically, the IoT device will encrypt the data gathered from sensors and send it to the gateway. Sometimes, it might store this data locally after encrypting it. The gateway will decrypt the data and run some analytics, and then encrypt it again to share it with the cloud. The cloud instance will decrypt the data one more time once received and run some analytics before encrypting it again, so it can be stored in the Database (DB).

To increase the portability of nodes, IoT frameworks have been proposed. The IoT framework is a great way to hide the complexity of network topology and type. However, an IoT framework must be designed to support end-to-end secure

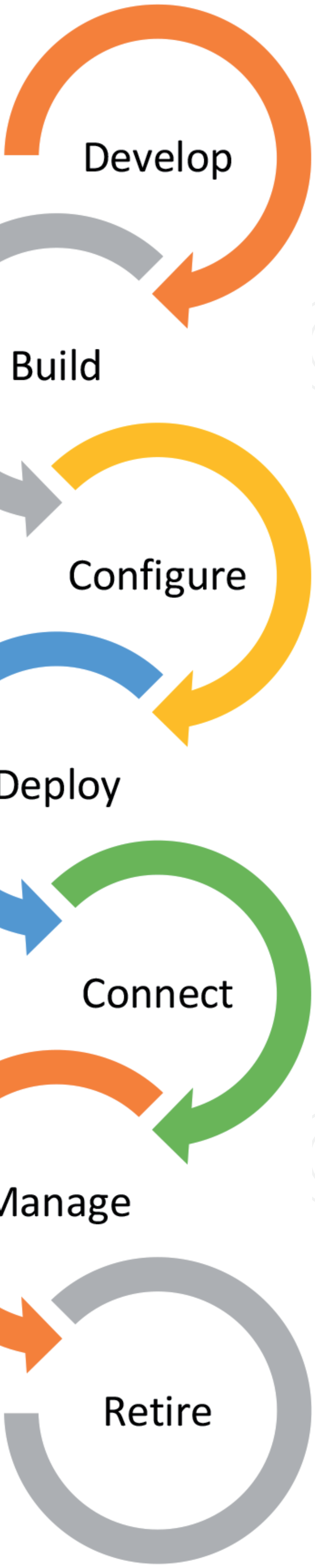


Figure 2.
IoT device typical lifecycle.

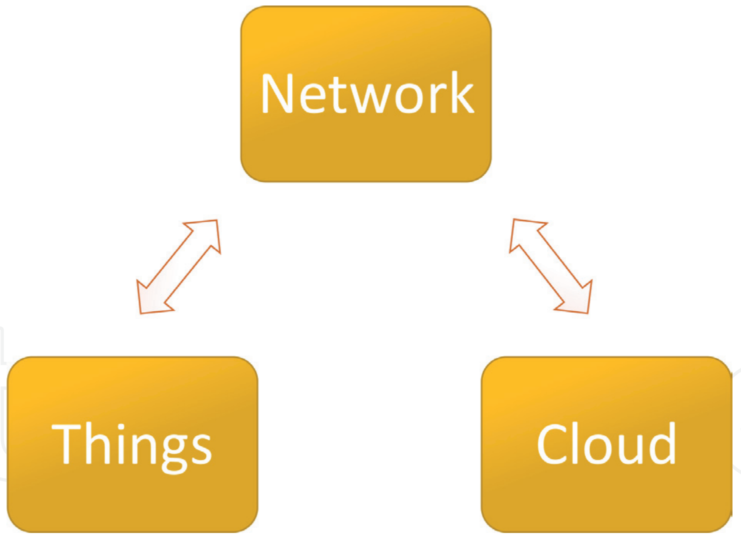


Figure 3.
Components of end-to-end security in IoT.

IoT framework	Security approach
Open Connectivity Foundation (OCF)	Tackles the security using three strategies: 1. Access control 2. Message encryption. 3. Device lifecycle management. Issue: (no security interoperability with other frameworks such as AllJoyn or UPnP)
AllSeen Alliance/AllJoyn	End-to-end security in the application layer using leaf nodes.
Universal Plug and Play (UPnP)	Security was not in the initial design and it was added later as an optional service through the IoT management and control architecture.
Lightweight Machine 2 Machine (LWM2M)	It achieves security using a secure message exchange with the Datagram Transport Layer Security (DTLS) and an access control list using the bootstrap server.
One Machine to Machine (OneM2M)	By design, it has the capability of performing authorization, access control, data protection as well as privacy preservation.
Open Platform Communications-Unified Architecture (OPC-UA)	OPC-UA is designed with security in mind. Distribute security functions over two layers, namely: the session layer and the secure channel layer. The former is the one responsible for authentication and access control, whereas the latter is taking care of message encryption, using Transport Layer Security (TLS) and HyperText Transfer Protocol Secure (HTTPS).
Data Distribution Service (DDS)	Security is achieved through three techniques: 1. Message security enveloping 2. Security tokens 3. Security plugin modules to add on services, such as authentication, access control and encryption.

Table 1.
Approaches of various IoT platforms.

communication between nodes, whether from an authentication, privacy, or confidentiality point-of-view. In fact, frameworks vary significantly when it comes to the implementation of this notion of end-to-end security [10, 11]. Some of these discrepancies are highlighted in **Table 1**.

The IoT framework is composed of three different layers: the data object layer, the node interaction layer, and the platform abstraction layer, also known as the connectivity and hardware abstraction layer. These three layers are shown in **Figure 4**. The data object layer is responsible for physical and logical node-to-device mapping. Also, it is the one responsible for managing the node Access Control List (ACL). The second layer is responsible for inter-node communication. The end-point security context must be achieved in this layer. Finally, the platform layer could be further

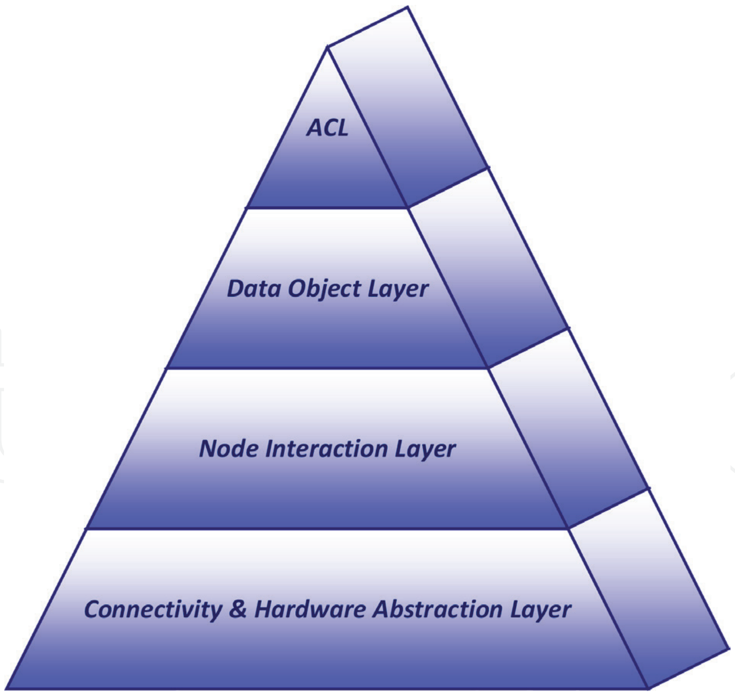


Figure 4.
Layers of IoT framework.

partitioned into three sub-layers, namely: network, sensor, and actuator, as well as security layer. Basically, an IoT network uses the same Internet layering model. This allows the IP to run on top of legacy industrial IoT protocols, like Fieldbus. The only difference is replacing the application layer with the IoT framework layer.

5. Securing IoT nodes

Nodes in any network must be able to do three basic tasks:

1. Neighborhood discovery

2. Authentication

3. Secure communication

To achieve IoT security effectively, we should focus on protecting the device, user identity, and data. We should manage the security at runtime as well. These points are illustrated in **Figure 5**.

The IoT Ecosystem is composed of the device, network, framework, and system management. By system management, we mean the procedure to maintain, replace and retire services. Also, it includes the procedures to update the firmware and apply security updates. In fact, requirements for system management vary a lot with different implementations. Obviously, brownfield and greenfield implementations have different system management requirements.

Let us talk about securing the IoT device itself. IoT devices are often resource constrained when it comes to memory space (storage), computation power, or even battery life. Therefore, the selection of which cryptographic algorithm to use is crucial. Although it is associated with high security impersonation risk, symmetric key cryptography is considered the most suitable type for IoT use. This is because it requires a small memory size and literally no hardware acceleration. Moreover, it is considered post-quantum safe given that the key size is increased from 128 to 256 bits [12].

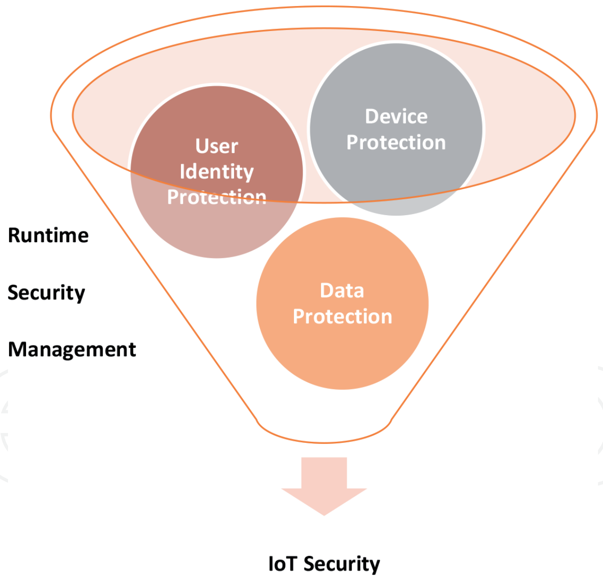


Figure 5.
Technologies involved in securing IoT devices.

Device Identity Composition Engine (DICE) [13, 14] is a secure IoT initiative proposed by Trusted Computing Group (TCG). It tackles the issues of secure booting and attestation without the need of a dedicated co-processor. DICE uses a hardware-based unique number generated among the device's boot called Unique Device Secret (UDS). The device identifier is nothing but the result of hashing the UDS with the device firmware. Thus, any modification to the firmware will result in a different device identifier, which will mark the device as 'untrusted' in the network.

6. Hardware and firmware security in IoT

Figure 6 shows the number of hardware and firmware security vulnerabilities in the past 20 years [15]. It is very clear that the number is on the rise and it should get the community's attention.

To achieve hardware-based security in IoT networks, we need to make sure that four aspects are taken care of, which are:

1. Device Identity
2. Boot Protection
3. Storage Protection
4. Runtime Protection

Intel has utilized available hardware-based technologies to achieve secure IoT networks, as illustrated in **Figure 7**. In the following paragraphs, we explain these technologies briefly:

6.1 Intel trusted execution technology (TXT)

TXT [16] is nothing but a set of hardware extensions to allow advanced security features, such as the measured launch environment and protected execution. TXT allows the user to run a specific program in an isolated space, protecting it from

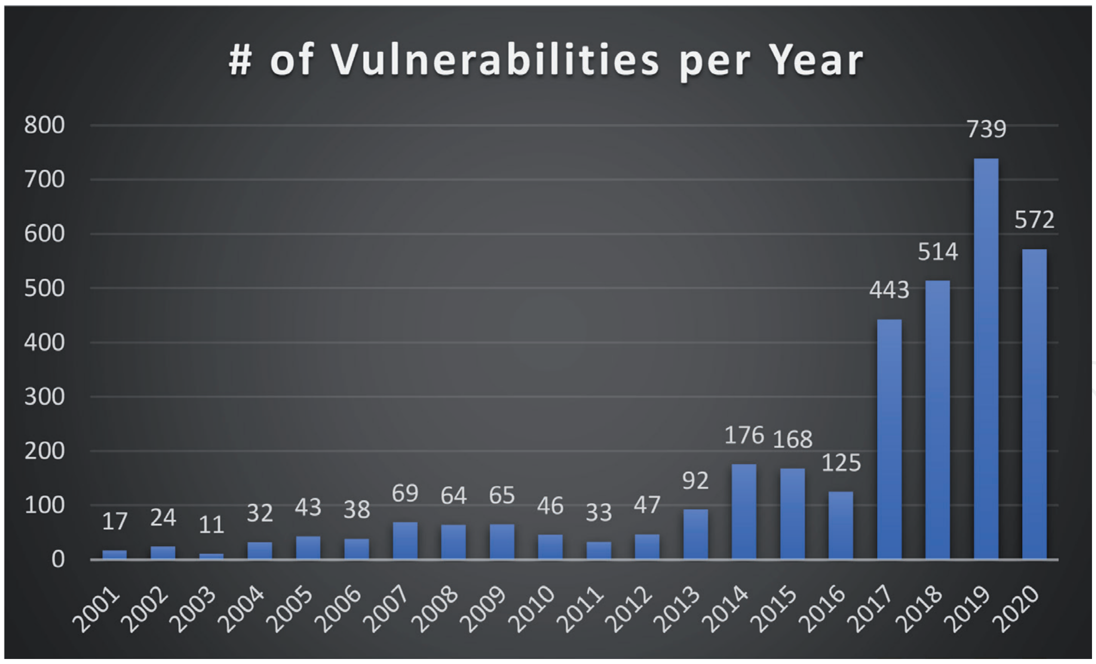
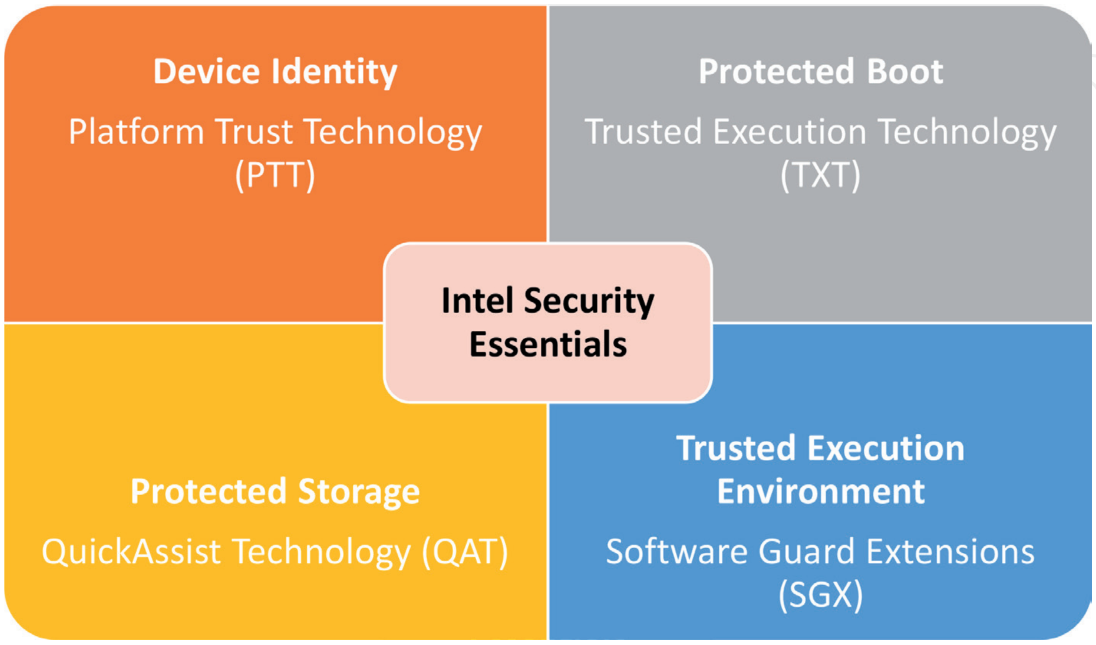


Figure 6.
Statistics of hardware and firmware vulnerabilities for the past 20 years.

other software in the system. For this reason, it improves the trust in the application's execution environment. As a result, important data can be protected from adversaries running malicious code on the same platform.

6.2 Intel QuickAssist technology (QAT)

This technology supports crypto-acceleration and compression-acceleration in the hardware level. By offloading the security-related computation to a special adapter, the system can utilize its CPU computation power in something else [17]. For example, wireless security and routing algorithms can surely benefit from this technology.



Device Identity
Platform Trust Technology (PTT)
Protected Boot
Trusted Execution Technology (TXT)
Intel Security Essentials
Protected Storage
QuickAssist Technology (QAT)
Trusted Execution Environment
Software Guard Extensions (SGX)

Figure 7.
Intel approach to achieve IoT security.

6.3 Intel platform trust technology (PTT)

This technology is very much related to an older technology called TPM (Trusted Platform Module). The main idea is to store keys in protected chips to authenticate hardware and software components. You can think of it as a digital fingerprint to the machine that is set by the manufacturer. PTT [18, 19] is implemented in the firmware to allow even low cost, low power devices like tablets to benefit from this technology.

6.4 Intel software guard extension (SGX)

Data can still be vulnerable while it is being stored or executed. SGX is a technology that fosters the isolation of process execution and memory allocation [20]. It empowers user-level code to have its own regions in the memory, called enclaves. These enclaves do not grant access to other processes with higher privileges. SGX basically strengthens the defenses by reducing the system's attack surface.

7. OS security in IoT

An operating system is considered the vehicle to control hardware through software. It is the lowest level software in the system hierarchy. OS security takes care of several tasks, such as separate execution and memory allocation, secret storage, and avoidance of programming errors.

The selection of the best operating system to use in IoT environments depends on several factors, like the system type, computing power, and threat level. IoT devices usually have limited power and computation capabilities resulting in limited choices of CPUs. Hence, OS security will be working on a best effort approach. Zephyr OS [21] is an open-source real time operating system that is specifically designed for resource constrained systems. It is unique in a sense that it was designed with security in mind. Consequently, it supports separate thread execution as well as separate memory storage. Moreover, it defines two levels of authority, which are the user level and the supervisor level. However, it lacks a proper authorization mechanism, which is a serious weakness [11].

As shown in **Table 2**, 80% of the top ten products with the highest number of distinct vulnerabilities reported in the past 20 years are found to be operating

#	Product name	Vendor name	Product type	# of vulnerabilities
1	Debian Linux	Debian	OS	3067
2	Android	Google	OS	2563
3	Linux Kernel	Linux	OS	2357
4	Mac OS X	Apple	OS	2212
5	Ubuntu Linux	Canonical	OS	2007
6	Firefox	Mozilla	Application	1873
7	Chrome	Google	Application	1858
8	IOS	Apple	OS	1655
9	Windows Server 2008	Microsoft	OS	1421
10	Windows 7	Microsoft	OS	1283

Table 2.
Top 10 products by total number of distinct vulnerabilities over 20 years [22].

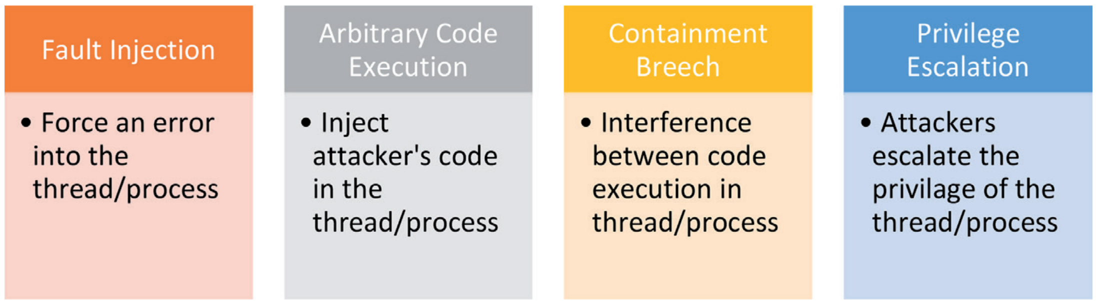

Fault Injection
• Force an error into the thread/process
Arbitrary Code Execution
• Inject attacker's code in the thread/process
Containment Breech
• Interference between code execution in thread/process
Privilege Escalation
• Attackers escalate the privilege of the thread/process

Figure 8.
Mechanisms used to compromise operating systems of IoT devices.

systems [22]. These statistics are not surprising because OS allows the attackers to access almost any part of the system with high privileges. Attackers use different ways to compromise the operating system. Some of these methods are shown in **Figure 8**. A rootkit is a good example of malware that uses these techniques to penetrate the OS and take over some of its tasks.

8. IoT network security

Most of the things in IoT will be connected wirelessly. Actually, there are many technologies available nowadays for connecting devices wirelessly, some of them belongs to Personal Area Networks (PANs), and some to Wireless Local Area Networks (WLANs) and Wide Area Networks (WANs). When talking about security in wireless networks, there are two aspects that must be considered. First, it is important to secure the data in transit using encryption mechanisms. Otherwise, anyone will have access to the data since air is a shared medium. The second one is the security of the wireless devices themselves, such as routers and access points. Unauthorized access to these devices might allow the attackers to reconfigure the network or forward the traffic to unwanted destinations.

Ethernet Time-Sensitive Networking (TSN) is state-of-the-art technology in industrial IoT that promises to bridge the gap between IT and OT [23]. Being vendor agnostic is a great feature of TSN and allows a large degree of interoperability. Furthermore, building it on top of Ethernet allowed a seamless interaction with non-TSN network devices in a plug and play fashion. Moreover, critical and non-critical traffic can co-exist with no worry about the potential increase in latency, thanks to the use of tight-time synchronization methods. Another important feature that allows the coexistence of the high and low priority traffic in the same network is Traffic Scheduling. In fact, TSN uses the notion of multiple queues to store packets with different priorities. TSN implements redundancy on the packet level by transmitting two duplicate packets through two different routes in the network. The one that arrives earlier will be processed whereas the other is simply discarded. This is a great way of assuring reliability in industrial-based networks. Finally, it is important to note that it is possible to use TCN as a link-layer protocol in any framework. OPC-UA is an example of such a case [24].

9. Conclusion

Security and privacy are important aspects of IoT networks. Given the widespread use of IoT devices in many fields, keeping the network secure is becoming increasingly important. Similarly, preserving data integrity is essential, especially when IoT sensors are used in the medical field. In this chapter, we have encouraged and supported the need for IoT security and privacy by giving examples of past attacks on IoT networks. Then, we described in detail the building blocks of IoT trustworthiness to illustrate the challenges facing IoT system engineers. After that, we explained the typical lifecycle of an IoT device, starting from the development phase until the device is retired. Moreover, we introduced the concept of end-to-end security in IoT networks. Also, we compared seven different approaches for securing IoT platforms and highlighted the strengths and weaknesses of each approach. Finally, we briefly presented challenges and potential solutions for securing IoT from the OS, hardware, network, and from a device point-of-view. The Intel approach to achieve IoT Security is presented as an example.

Nomenclature

Section 1	
IT	Information Technology
OT	Operation Technology
IP	Internet Protocol
IoT	Internet of Things
DDOS	Distributed Denial of Service
DVR	Digital Video Recorder
SSH	Secure Shell
USB	Universal Serial Bus
PLC	Programmable Logic Controller
Section 2	
TEE	Trusted Execution Environment
Section 3	
SDK	Software Development Kit
API	Application Programming Interface
CPU	Central Processing Unit
SoC	System on Chip
OS	Operating System
Section 4	
DB	Database
OCF	Open Connectivity Foundation
UPnP	Universal Plug and Play
LWM2M	Lightweight Machine 2 Machine
DTLS	Datagram Transport Layer Security
OneM2M	One Machine to Machine
OPC-UA	Open Platform Communications-Unified Architecture
TLS	Transport Layer Security
HTTPS	HyperText Transfer Protocol Secure
DDS	Data Distribution Service
ACL	Access Control List
Section 5	
DICE	Device Identity Composition Engine
TCG	Trusted Computing Group
UDS	Unique Device Secret
Section 6	
TXT	Trusted Execution Technology
QAT	QuickAssist Technology
PTT	Platform Trust Technology

TPM	Trusted Platform Module
SGX	Software Guard Extensions
Section 8	
PAN	Personal Area Network
WLAN	Wireless Local Area Network
WAN	Wide Area Network
TSN	Time-Sensitive Networking

Author details

Ahmad J. Showail[1,2]

1 Taibah University, Madinah, Saudi Arabia

2 University of Prince Mugrin, Madinah, Saudi Arabia

*Address all correspondence to: ashowail@taibahu.edu.sa

References

[1] Foundation Fieldbus [Internet]. Available from: http://www.foundationfieldbus.com/ [Accessed: 2021-01-11]

[2] Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A. R., & Tarkoma, S. (2017, June). Iot sentinel: Automated device-type identification for security enforcement in iot. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS) (pp. 2177-2184). IEEE.

[3] Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. Computer, 50(7), 80-84.

[4] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J.,... & Zhou, Y. (2017). Understanding the mirai botnet. In 26th {USENIX} security symposium ({USENIX} Security 17) (pp. 1093-1110).

[5] Dynamic DNS [Internet]. Available from: https://account.dyn.com/ [Accessed: 2021-01-11]

[6] Mirai Botnet (malware) [Internet]. Available from: https://en.wikipedia.org/wiki/Mirai_(malware) [Accessed: 2021-01-11]

[7] Langner, R. (2011). Stuxnet: Dissecting a cyberwarfare weapon. IEEE Security & Privacy, 9(3), 49-51.

[8] Schiffman, J., Moyer, T., Jaeger, T., & McDaniel, P. (2011). Network-based root of trust for installation. IEEE Secur. Priv., 9(1), 40-48.

[9] Abera, T., Asokan, N., Davi, L., Koushanfar, F., Paverd, A., Sadeghi, A. R., & Tsudik, G. (2016, June). Things, trouble, trust: on building trust in IoT systems. In Proceedings of the 53rd Annual Design Automation Conference (pp. 1-6).

[10] Ammar, M., Russello, G., & Crispo, B. (2018). Internet of Things: A survey on the security of IoT frameworks. Journal of Information Security and Applications, 38, 8-27.

[11] Cheruvu, S., Kumar, A., Smith, N., & Wheeler, D. M. (2020). Demystifying Internet of Things Security: Successful IoT Device/Edge and Platform Security Deployment (p. 488). Springer Nature.

[12] Bernstein, D. J. (2010, May). Grover vs. McEliece. In International Workshop on Post-Quantum Cryptography (pp. 73-80). Springer, Berlin, Heidelberg.

[13] Jäger, L., & Petri, R. (2020, August). DICE harder: a hardware implementation of the device identifier composition engine. In Proceedings of the 15th International Conference on Availability, Reliability and Security (pp. 1-8).

[14] Device Identity Composition Engine (DICE), Trusted Computing Group [Internet]. Available from: https://trustedcomputinggroup.org/work-groups/dice-architectures/ [Accessed: 2021-01-12].

[15] The National Institute of Standards and Technology (NIST) Vulnerability Database [Internet]. Available from: https://nvd.nist.gov/vuln/search [Accessed: 2021-01-12].

[16] Intel Trusted Execution Technology (TXT) [Internet]. Available from: https://www.intel.com/content/www/us/en/support/articles/000025873/technologies.html [Accessed: 2021-01-12].

[17] Intel QuickAssist Technology (QAT) [Internet]. Available from: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/quickassist-adapter-8950-brief.pdf. [Accessed: 2021-01-12].

[18] Intel Platform Trust Technology (PTT) [Internet]. Available from: https://static.onlogic.com/resources/downloads/OnLogic-PTT-One-Pager.pdf [Accessed: 2021-01-12].

[19] Strengthening Security with Intel Platform Trust Technology. [White Paper]. Available from: https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/enterprise-security-platform-trust-technology-white-paper.pdf. [Accessed: 2021-01-12].

[20] Intel Software Guard Extensions (SGX) [Internet]. Available from: https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html [Accessed: 2021-01-12].

[21] The Zephyr Project [Internet]. Available from: https://zephyrproject.org/ [Accessed: 2021-02-05].

[22] Top 50 Products By Total Number Of "Distinct" Vulnerabilities [Internet]. Available from: https://www.cvedetails.com/top-50-products.php [Accessed: 2021-01-12].

[23] Finn, N. (2018). Introduction to time-sensitive networking. IEEE Communications Standards Magazine, 2(2), 22-28.

[24] Schwarz, M. H., & Börcsök, J. (2013, October). A survey on OPC and OPC-UA: About the standard, developments and investigations. In 2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT) (pp. 1-6). IEEE.