Recommendations for securing Internet of Things devices using commodity hardware.

Steven J. Johnston, Mark Scott, Simon J. Cox
Faculty of Engineering and the Environment,
University of Southampton,
United Kingdom
Email: sjj698@zepler.org

***Abstract*—The Internet of Things (IoT) describes a world where ubiquitous devices such as sensors are all capable of communicating with the Internet. The concept of Internet enabling devices is not new, however the popularity of IoT promises to increase the number of connected devices considerably.**

Ubiquitous IoT devices have serious security implications as they occur in large numbers, are geographically distributed and can be difcult to physically secure. These devices may contain sensitive or commercially valuable data making them attractive to various forms of attack.

In this paper we provide an overview of the state-of-the-art hardware available and recommend ways for reducing the IoT attack surface. We utilise the most relevant security technologies and make four recommendations for consideration when developing end-to-end IoT systems (full disk encryption, cryptoprocessor, bootloader signing and bootloader encryption) providing a discussion of the benets and limitations with reference to currently available commodity hardware.

***Index Terms*—TPM; Secureboot; TrustZone; IoT; cryptoprocessor; signed bootloader; encrypted bootloader;**

I. Introduction

The Internet of Things (IoT) describes a world where ubiquitous devices such as sensors are all capable of communicating with the Internet. This is not a new concept and has existed across multiple areas such as retail, automotive and industrial applications for many years. Many safety and security systems such as medical devices, CCTV cameras and smoke detectors are already IoT devices, connected to networks which have the capability to process, store and manage the data produced by each of the sensors. The IoT ecosystem is changing and becoming attractive across a wider range of disciplines, hence their emergence in popular technology. The key driver in this uptake is the plummeting cost of small, powerful, energy ef cient processors that are capable of being embedded in almost every single electronic device ever manufactured. Excluding PCs, smartphones and tablets, some sources predict that the IoT will exceed 26 billion devices by 2020 [1]. Even if this unimaginatively large number of devices is an overestimate, there is no escaping that we already exist in an IoT world and that this world is expanding rapidly. Metcalfes law [2] states that '*the value of a telecommunications network is proportional to the square of the number of connected users of the system*', indicating that as the number of IoT devices increases we can expect newer capabilities and greater insights.

Adding devices to such an ecosystem has security implications especially as the devices are physically dif cult to secure, and standards and guidelines from organisations such as the International Telecommunication Union (ITU) and the Trusted Computing Group (TCG) are still topics of discussion [?].

In this paper we look at an IoT architecture for custom devices and provide recommendations to improve security.

The *things* behind the Internet of Things can be a wide variety of data producing applications, devices, sensors or custom objects, ranging from new to already existing devices. Many already produce useful data, for example cars and elevators, which can help with maintenance intervals and already exist in their millions. These mainly just lack Internet connectivity and back end services to process the data. We can assume that many new IoT devices will have Internet connectivity built into them and will be produced in large quantities in an IoT enabled state, but many IoT devices will be custom or community produced, using commodity hardware such as the Raspberry Pi, Hummingboard, Beaglebone.

In this paper we focus on producing custom IoT devices from commodity hardware, ensuring secure data collection storage and transmission to an online server. The security of online cloud providers and social engineering attacks are deemed out of scope for this paper.

Figure 1 shows a generic high-level IoT architecture. At the lowest level we have IoT devices which collect or produce data and can have limited storage. These devices in themselves may be connected directly to the Internet but, often due to power, size, connectivity or RF range restrictions, they might connect via a local wired or wireless network to an IoT Gateway. These gateways act as local data buffers, signal boosters and usually have Internet connectivity. Ultimately the data ends up on a server or data warehouse with enough storage and processing capability to manage the streams and bursts of data from potentially millions of devices. These servers could be a cloud provider or in-house solution which ultimately serves and processes the data for end-user applications.

Client access to data held in a data centre or cloud provider is a well understood problem and is considered out of scope for this paper. For ubiquitous IoT computing to become a reality we need to establish the validity and authenticity of the data from the producer/sensors to the back end data centre/cloud servers. This can be divided into two areas of concern 1)

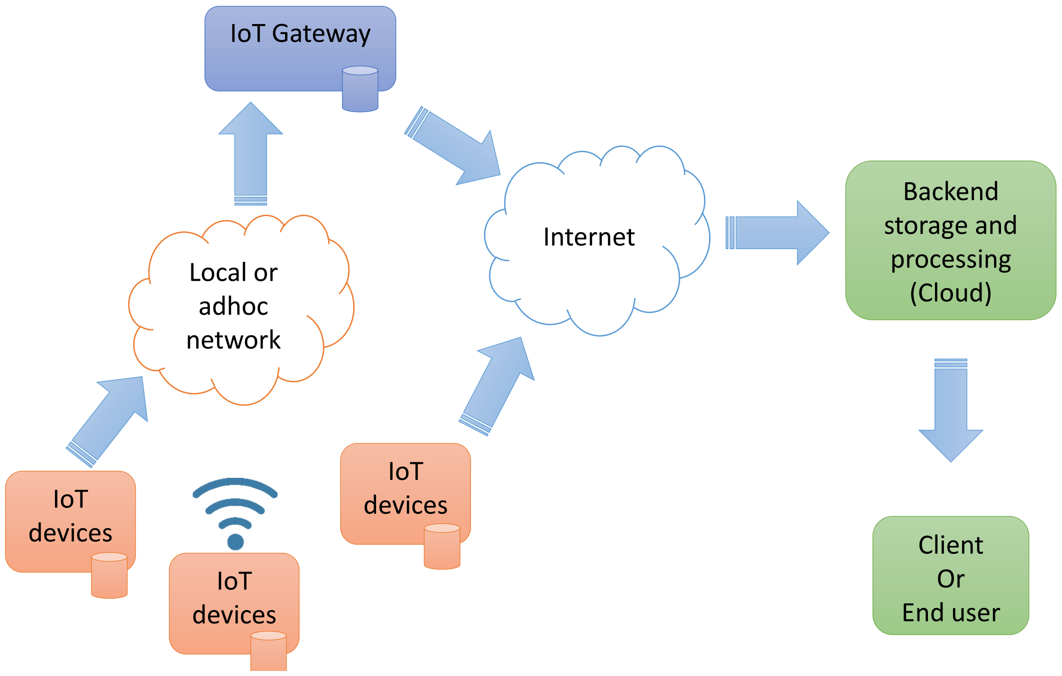

Fig. 1. IoT architecture overview

transmitting the data to the data store and 2) building an IoT device that is secure.

II. Building a secure IoT device

In this section we discuss strategies that would help to secure an IoT device, summarised in Figure 2. Building a secure IoT device requires careful consideration as there are many angles of attack; regardless of the application, reducing this attack surface is paramount, despite of the perceived value of the data. IoT devices differ from many traditional technology deployments in that protecting the device from physical intrusion may not be possible, especially on remote installs.

Although it can also be assumed that physical access to a device will result in a compromise [3], reducing the attack surface increases the cost and effort. In this paper we consider an IoT device as an embedded device running an OS (probably Linux) and custom applications.

The structure of this section is cumulative, the nal recommendation assumes the implementation of the previous recommendations. Multiple recommendations are provided to offer a degree of exibility as some require hardware speci c features.

Recommendation 1 – Full disk encrypted storage

In order to secure data, we recommend that the IoT device le systems held are encrypted. Encryption usually requires entry of a password or provision of a key at the necessary point in the boot process. The implications of this for IoT devices – which are often in remote locations – must be considered, as password entry on every reboot may prove dif cult or impractical. Storing an encryption key on a locally accessible SD card or unencrypted partition on the device would invalidate the protection provided by the encrypted storage as an attacker would also have access to the key. A secure cryptoprocessor provides a less vulnerable way of storing the key and is discussed below.

1) Full disk encryption
2) Sealed key storage and system integrity veri cation
   - A secure cryptoprocessor
   - A trusted measured boot
   - A disk encryption key, sealed by the cryptoprocessor
   - A trusted measured operating system
3) Signed boot loader
4) Encrypted boot loader

Fig. 2. Recommendations for securing an IoT device

Recommendation 2 – Use a cryptoprocessor to store encryption key and modify boot loader to verify system integrity

A cryptoprocessor can help with the provision of a key in a safer manner by encrypting the key in such a way that it cannot be used unless the IoT device is in a known state – a process known as *sealing*. With this approach, manual entry of a password is no longer required.

We tested this with a Trusted Platform Module (TPM), an international standard for a secure cryptoprocessor [4]; in our case, we used an Atmel AT97SC3204T. TPMs have a number of Platform Conguration Registers (PCRs) – at least sixteen to meet the Trusted Computing Groups specication for a TPM [5] – which are shielded locations that are 160 bits (20 bytes) each and store a SHA-1 cryptographic hash [6], [7], with all bits set to zero at boot. PCRs cannot be reset or given a specic value, they can only be extended with another SHA-1 hash which is done by concatenating the previous and new hashes and then creating a new hash of the resultant 40 byte value.

Data sealed by the TPM can only be decrypted if the chosen PCRs are in a specic state. This means that decryption keys can be stored in a sealed state and only be released if the required extensions of the PCRs have been made. The measurements of the system would normally be done by the boot loader and would include the rmware, boot loader image and the kernel being booted. The encryption key would be released by customising the boot sequence. We tested with Arch Linux and an encrypted LUKS root partition by creating a custom initial RAM disk (initrd) to retrieve and unseal the encryption key during the boot process, using the IBM software TPM library (ibmswtpm).

The security of this approach relies on the correct measurements being made in the correct order as well as the security of the operating systems disk encryption method and the security of the TPM hardware.

The verication of the system involves establishing a chain of trust from the start of the boot process to ensure that unauthorised software cannot be executed before the key is released. Where this chain of trust begins depends on the hardware available. One potential attack would be to reverse engineer the boot loader and then bypass the secure boot loader with their own, but perform the same measurements, giving access to the encrypted keys. We will therefore now present two other con gurations, each improving the trustworthiness of the device, but each limiting the choice of hardware for the IoT device.

Recommendation 3 – Use hardware where the boot ROM veries boot loader signature

The rst part of the chain of trust is provided by the boot ROM. If it can arbitrarily execute any boot loader, an attacker can circumvent our rst approach to trusted boot. Using a boot ROM that can verify the boot loader by checking its signature before executing it provides protection against an attackers custom boot loader. Even by reverse engineering the boot loader to discover what measuring needs to be done, these measurements cannot be performed to trick the TPM into revealing its encryption keys because a secure boot ROM will not execute the boot loader if it has not been authorised, veried by a signature. These keys are permanently embedded in the hardware and cannot be modied once written.

This approach is still not foolproof though because physical disassembly of the hardware and attachment of the TPM to another system may permit the necessary measurements (again, determined by reverse engineering) to be performed and encryption keys to be discovered.

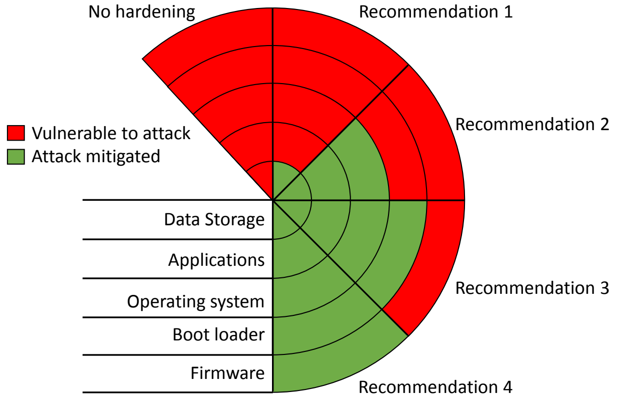

Fig. 3. IoT recommendation vulnerabilities overview

Recommendation 4 – Use hardware that supports an encrypted boot loader with a key known to boot ROM

To provide protection of boot loader code from reverse engineering, some hardware supports encryption of the boot loader code. This feature improves the security of the trusted boot greatly. Combined with only executing signed boot loaders, this makes compromising the system non-trivial.

The security of this recommendation relies on the hardware implementation to ensure that the boot loader encryption keys remain protected.

III. Discussion

A. *IoT device*

An approach following these recommendations would improve the security of the stored data and increase the trustworthiness of the IoT device to the rest of the system, although TPM attacks have been demonstrated [8]–[10].

The trusted boot recommendations rely on creating what is known as a Static Root of Trust Measurement (SRTM) with the later recommendations having a more dependable root of trust, as shown in Figure 3. For these to work reliably, the chain of trust should extend from the boot ROM into the operating system. Any weaknesses in the chain of trust would reduce its effectiveness.

The component that measures itself and the rest of the BIOS in an SRTM is the Core Root of Trust for Measurement (CRTM). A well implemented CRTM is critical [11] and must meet the requirements of the Trusted Computing Groups BIOS specication [12]; recommendation 2 does not, as the boot loader can be replaced. A more appropriate approach might be to use a Dynamic Root of Trust Measurement (DRTM) which gives the ability to create a trusted environment dynamically even without a reliable CRTM. This is done with support from a CPU and a TPM with special PCRs that can only be reset by the CPU running trusted code. Intel call this technology Trusted Execution Technology (TXT) [13] and

AMD have a corresponding technology with Secure Virtual Machine (SVM) [14].

Some ARM chips, used by many IoT devices, use a different approach known as TrustZone which allows the system to run two operating systems alongside each other, one secure and one non-secure. Applications in the non-secure environment can communicate with the secure environment with a Secure Monitor Call (SMC) instruction [15].

AMD and Intel use the TPM as the root of trust. With ARMs TrustZone, a key or a software TPM embedded in the secure world [16] can be used as the root of trust for the device, for example secure boot on the Nokia Lumia [17], [18].

The objective of IoT devices is to communicate with online servers and there needs to be reassurances about the data collected. The data needs to be stored and transmitted con- dentially, ensuring data is not leaked and the data provider needs to be authenticated ensuring that the data has originated from the IoT device. An often overlooked requirement is around data integrity, the receiver needs to know that it has not been tampered with – injecting fake or false data can be hard to detect. For an IoT device to be useful non-repudiation, authentication, integrity and condentiality must be conrmed.

The operating system and application layer are considered out of scope for this paper, but are vulnerable to an attack which could result in compromising the entire system. General good practice is to select an OS with continued support and a well know hardening model, disable unutilised services and ensure the OS is kept updated as new vulnerabilities are discovered. The application layer attack surface can be more dif cult to assess as it often contains custom code, libraries and open communication ports. It is important that the security implications at these two layers is well understood.

Securing the communication channel may be harder than it appears since technologies such as SSL, SSH, Enveloped Public Key Encryption (EPKE) rely on a private key, hash or certicate that resides on the IoT device, making the device vulnerable to various attacks. Improvements can be made by storing any private keys in a TPM and further enhanced by using one of the recommendations in Section II to measure the system and use the TPM's Attestation Identity Key (AIK) to conrm the state of the system. TPM version 1.2 also supports Direct Anonymous Attestation (DAA) which offers a degree of privacy.

Encrypting the IoT device storage ensures that both the operating system, applications and data remain secure. In this work we used the Linux Unied Key Setup (LUKS) for full disk encryption as it encrypts the OS and applications, without the users having to customise their IoT device application layer.

Regardless of the technology employed, the encrypted data will require unlocking using a private key which is a potential attack point, especially since the IoT device needs to reboot without user intervention.

IV. Conclusion

Creating IoT devices using commodity off-the-shelf hardware should be done cautiously especially where removable storage is used. Regardless of the technology adopted, physical access to a device means that the security can be compromised, hardening a device only raises the effort required. However there is a big difference between reading data from an unencrypted removable SD card and extracting keys from a cryptoprocessor, subverting the boot process or emulating a CPU using a FPGA.

In this paper we recommend four technologies to harden an IoT device, i) disk encryption ii) cryptoprocessor iii) signed boot and iv) encrypted boot. As the security of custom embedded devices becomes more of a focus [19] we can expect newer hardware to include a richer set of features, such as TrustZone on ARM.

This work explores the currently available technologies applicable to hardening IoT devices, in future work we will show further applications using example commodity hardware.

Publication dataset: DOI:10.5258/SOTON/402140

References

[1] P. Middleton, P. Kjeldsen, and J. Tully, "Forecast: The internet of things, worldwide," 2013.

[2] B. Metcalfe, "Metcalfe's Law: A network becomes more valuable as it reaches more users," 1995.

[3] J. Brossard, "Hardware backdooring is practical," in *Blackhat Briengs and Defcon conferences*, Las Vegas, 2012.

[4] ISO, Geneva, "Information technology – trusted platform module," 2009.

[5] TCG, *TPM Main Specication Level 2 Version 1.2, Revision 116: Part 1 - Design Principles*, 2011.

[6] R. Rivest, "The MD5 message-digest algorithm," RFC 1321 (Informational), Internet Engineering Task Force, 1992.

[7] NIST, "Secure hash standard (SHS)," 2012.

[8] E. R. Sparks, "A security assessment of trusted platform modules," Department of Computer Science, Dartmouth College, Tech. Rep. TR2007-597, 2007.

[9] B. Kauer, "OSLO: Improving the security of trusted computing," in *USENIX Security*, vol. 7, 2007.

[10] J. Winter and K. Dietrich, "A hijackers guide to the LPC bus," in *Public Key Infrastructures, Services and Applications*. Springer, 2012, pp. 176–193.

[11] J. Butterworth, C. Kallenberg, X. Kovah, and A. Herzog, "BIOS chronomancy: Fixing the core root of trust for measurement," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. New York: ACM, 2013.

[12] TCG, *TCG PC Client Specic Implementation Specication for Conventional BIOS, Version 1.21 Errata*, 2012.

[13] Intel, "Intel Trusted Execution Technology: Software development guide," 2014.

[14] AMD, "AMD64 architecture programmers manual volume 2: System programming," 2011.

[15] ARM, "ARM security technology: Building a secure system using TrustZone technology," 2009.

[16] S. Zhao, Q. Zhang, G. Hu, Y. Qin, and D. Feng, "Providing root of trust for ARM TrustZone using on-chip SRAM," in *Proceedings of the 4th International Workshop on Trustworthy Embedded Devices*. New York: ACM, 2014, pp. 25–36.

[17] Nokia, "Trusted computing in Nokia Lumia," in *RSA 2013 – Trusted Computing: Billions of Secure Endpoints in 10 Years*, 2013.

[18] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Trusted execution environments on mobile devices," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. New York: ACM, 2013.

[19] J. Watson, "Internet of Things: potential risk of crime and how to prevent it," *Home of ce* , 2014.

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWX

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

abcde mn rst xA I T

abcdefghi klmnopqrstuvw y I M P STU W Z (

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ’() ,-./:; @[ ]

01234 6 abcdefghi klmnopqrstuvw y ABCDEFGHI KLMNOP RSTUVWXY &

abcdefghi klmnopqrstuvw y I M P STU W Z

01234 6 abcdefghi klmnopqrstuvw y ABCDEFGHI KLMNOP RSTUVWXY & ,-. :