Security solutions for embedded systems

Nayani Srinivas Vijayanand

Oulu University of Applied sciences
School of engineering,
Oulu, Finland.
t3nasr00@students.oamk.fi
vijay.nayani@gmail.com

Abstract

This document discusses on various security aspects and solutions that should be considered for designing a secured embedded systems. This paper further discusses on common kinds of security attacks prevalent in embedded systems and best approaches that can be employed for mitigating.

Keywords: Embedded systems, Wireless base stations, Secure boot, Chain of trust, IPSEC , Hardware security, Firewall security, Kernel security, Cryptographic algorithms, Trusted Execution Environment (TEE) , Intel Identification Protection Technology (IPT) .

1 Introduction

Security in electronic systems has been gaining importance due to ever increasing amount of attacks in today's world. Operating a secure environment has been of paramount importance right from high end systems like space shuttles, defense & control systems, Internet routers, telecom infrastructure equipment to right until low end consumer appliances and personal gadgets. Primary focus of the paper is to draw attention on aspects of system wide approach to security in embedded systems, types of security attacks and best approaches in dealing them. Providing a secure systems or service in practice means to address wide variety of challenging security concerns which includes data confidentiality and integrity, authentication, privacy, denial of service and digital content protection.

1.1 Driving forces for designing a secure systems

Some of the primary drivers for developing fool-proof security solutions in embedded domain are

- Advent of interconnectedness of devices: Internet of Things (IoT) in today's world is posing a bigger threat in embedded world. A compromised device in the network can have a cascading effect on whole network.
- Increased use of open source software by companies in an effort to bring down the software costs is turning out as breading grounds for hackers.
- Ever increasing use of microprocessors in day today life which has carried the risk of compromising on personal information, For ex: washing machines, cameras etc.

1.2 Security design constraints in embedded systems

Embedded systems are self-contained systems designed to work on SoC (System on Chip) often with real time computing restrictions unlike general-purpose operating

systems.
Greater computational demands, limited battery life, hardware capabilities & low flexibility pose challenges to implement fool proof security architecture.

Developing an embedded system with tamper resistance [1] measures to sustain physical and side-channel attacks poses challenges due to systems minimal foot print and increased circuit complexity.
Cost and complexity can also be major contributors that can influence the security architecture of an embedded system. For example, task of incorporating physical security mechanisms in a single-chip cryptographic module.

2 Security requirements

Security requirements vary according to end usage of the system. For example, security considerations for point-of-sale (POS) terminals typically consumer gadgets may be quite different from those applicable to a wireless base station. Embedded systems, because of their physical accessibility, also raise some unique security concerns.

An Internet server which is well protected by firewalls has little to worry about from attacks, unlike for a wireless cellular base station deployed on user premises. Often also the value of the intellectual property (IP) in these devices makes them an enticing target for sophisticated attacks.

Never the less , this section jots down some of typical security requirements that needs to be addressed by large variety of embedded devices ranging from point of sale to devices customized for dedicated functionalities.

- Device Availability: Systems must ensure they don't fall pray for Denial of Service (DoS) attacks and shall be able to perform its intended duties without disruption.
- User Authentication: All systems shall need to validate the users by effective mechanisms before they are allowed to gain access to system resources.
- Content security: Systems need to enforce restrictions on usage of digital content in the system typically DRM etc..
- Secure storage: Devices shall need to ensure the integrity and confidentiality of the stored content. Typical examples are service/device certificates, authentication keys, and passwords. Etc.
- Secure network/ communication: Devices working on network shall provide services and access to only authorized devices on the network. This includes authenticating communicating peers, ensuring confidentiality and integrity of communicated data, preventing repudiation of a communication transaction, and protecting the identity of communicating entities. Typical examples are secure bank transactions, PIN codes etc..
- Restricting unauthorized software: Devices shall be able to restrict running unauthorized software which can seriously dent the functionality of the device and depriving the legitimate revenue for the service providers. Also running unauthorized software on embedded devices can be far more serious in other areas such as military and aerospace applications, where dangerous equipment and even state secrets can fall into unfriendly hands.
- Software Reverse Engineering: Devices shall take steps to ensure that reverse engineering shall not succeed. Several tools and techniques are can be used to reverse engineer embedded software to extract useful and valuable information. Typical examples are disassembler tools for extracting file systems.

3 Security approaches

Embedded devices need to adopt comprehensive security solution that addresses many of the above mentioned security concerns at the system level.

Providing comprehensive secure solution is a system-wide approach in which security emanates from software execution environment and permeates throughout the hardware interfaces, buses and IP blocks.

In practice building a best of class secure device boils down to below considerations

- Device when doing secure operations shall take the help of designated software designed to take care of secure operations. Typically operations that fall under this category are software authentication , user identify verification, decryption & encryption functionalities, accessing secure information , establishing secure communication channels etc..
- Specially designated software meant to execute secure operations shall take the help and exploit the underlying hardware that is specially built for supporting secure operations effectively.

Thus a combination of hardware based software solution provides a robust foundation for software to run the device in perfect secure mode.

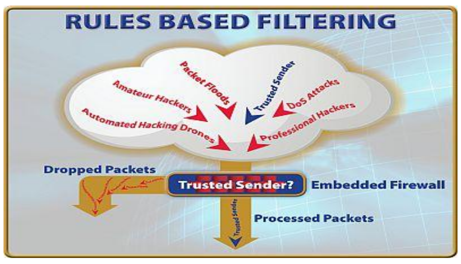
4 Attacks and mitigation methods

Approach for providing security solutions varies depending upon attack to address and hence forth difficult to generalize. Certain solutions can be a combination of software and hardware based while others are purely either software or hardware. Choice of security solutions are based on target of attack to mitigate robustness needed for handling such an attack. This section focuses on typical attacks and ways to mitigate them by employing software / hardware or both.

4.1 Software based solutions

Most general attacks on systems include

- Network security: Common network securities include dictionary attacks, DoS (Denial of services), TCP hijackings, and sniffing attacks. Most of these attacks can be mitigated by effectively configuring the firewall with an chains based filtering and frequent monitoring the network traffic. In Linux world, range of tools like nmap, netstat, wire shark can be deployed to track and monitor the packets. Attacks like eavesdropping, MITM (Man In the Middle Attack) can be mitigated by establishing secure communication by employing IPSEC protocols.



- Operating system /kernel security: Linux is predominantly used in embedded systems is open source based and hence forth vulnerable to attacks. Linux can special security patches to harden it and plug the potential loopholes and enhancing the operating system kernel security. Typical example include LIDS (Linux Intrusion Detection System) , grsecurity kernel patches, SE-Linux , LSM (Linux Security Modules) etc..
- Local and file system security: Employing more conservativeness approach at giving minimum amount of

privileges a to user groups. Creation of group accounts needs to be discouraged as they accountability is difficult to enforce. Files system access and permissions for executable partitions in particular need to be disabled to curtail unauthorized installations and thereafter executions.

- System Services: Security employed by system services is paramount and is more often the start point for initiating and executing secure operations. Applications and service providers (daemons) shall need to carry user authentications, integrity checking and establish secure communication channels before providing the actual services. It is often suggested to use operating system level provided authentication modules/schemes instead of employing application specific ones. Typical examples include secure shell services, PGP, DHCP, Web servers, file servers, DNS bindings, LDAP accesses, VPNs, authentication via PAM modules in Linux based systems, IPSEC protocols for secure communication. Etc.

4.2 Hardware (SoC) based security solutions

Traditional approaches to security in embedded environments require a separate security processor with its own carefully controlled access and execution environment. Although highly secure, this approach suffers from a number of disadvantages, most notably high system cost and a lack of programmability. Specially designed SoCs provide security extensions implemented in the hardware logic that can help to eliminate these disadvantages by offering following secure services

- Separate secure environment

:A separate trusted execution environment (TEE) is provided by Soc for facilitating secure transactions in an isolated box from the “normal” processing environment where the device operating system and applications run. Examples are ARM's Trust zone technology and Intel's Integrated protection Technology (IPT).

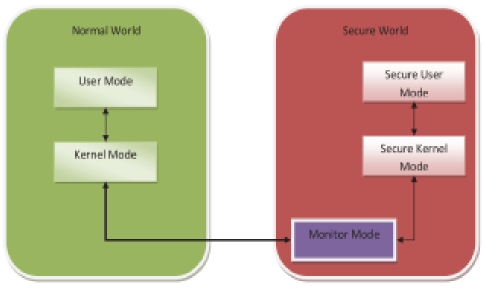
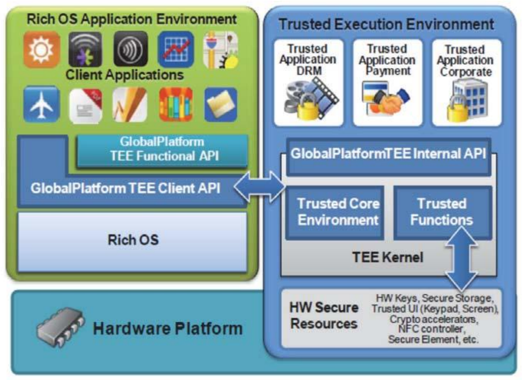


Figure 1: Normal and secure world environment

- Secure platform API: Semiconductor vendors provide programming interfaces for the applications aimed at performing secure operations in a TEE environment. This API makes higher layer applications transparent to the underlying TEE hardware implementation.


Rich OS Application Environment
Client Applications
GlobalPlatform TEE Functional API
GlobalPlatform TEE Client API
Rich OS
Trusted Execution Environment
Trusted Application DRM
Trusted Application Payment
Trusted Application Corporate
GlobalPlatform TEE Internal API
Trusted Core Environment
Trusted Functions
TEE Kernel
HW Secure Resources
HW Keys, Secure Storage, Trusted UI (Keypad, Screen), Crypto accelerators, NFC controller, Secure Element, etc.
Hardware Platform

- Security aware hardware peripherals: The internal buses have security awareness built into the hardware by which both secure and non-secure access can be easily distinguished. Conditional access to different buses in the system can be granted depending upon the execution environments from the secure operations are carried upon.

- Secure boot: Device boot up is the genesis for any secure solution .Reason being, any device that starts up in a compromised environment can potentially circumvent all other runtime security measures. Since security should start as early in the boot process as possible, on-chip ROM code plays a critical role in the secure boot process. Its job is to run only the trusted software post successful authentication and verification. ROM code checks the boot loader software has been signed by the OEM with the corresponding private key whose public counterpart is present in device in secure location. The OEM's public key is programmed into the device while manufacturing cannot be erased afterwards. The boot loader and subsequent software are responsible for doing this similar job when they start invoking other software down the line. This scheme allows for setting up a complete chain of trust.

5 Conclusions

Regardless of the overall software architecture, three key pieces of software are needed to implement a secure solution: secure boot for starting the device, platform software to manage the secure and normal worlds, and lastly, secure applications to provide service to the user.

6 Abbreviations

SoC: System on Chip.
IoT: Internet of things.
IPT: Identification Protecting Technology.
TEE: Trusted Execution Environment.
IPSEC: Internet Protocol Security.
OEM: Original Equipment Manufacturer

7 References

- http://www.ti.com.cn/cn/lit/wp/spry22/spry228.pdf
- [1]ANDERSON, R. AND KUHN, M. 1996. Tamper Resistance Cautionary http://www.cl.cam.ac.uk/users/rja14/tamper.html
- ACM Transactions on Embedded Computing Systems, Vol. 3, No. 3, August 2004