Survey of Security in Embedded System Design

E. Pradeep, M. Sarika

Abstract— **One of the most important challenges that need to be currently faced in securing resource-constrained embedded systems is optimizing the trade-o between resources used and security requirements. Security is constantly evolving as threats change over time. As a device becomes popular exists in the market longer, it becomes more susceptible to attack. Many devices in the past were not designed to be field programmable or accept updates without significant modifications. Those days are gone. Devices today must be field upgradeable to not only change and improve functionality, but to deal with bugs and security issues. Including security planning in the life cycle management of your device is critical. Moreover, it's important that your organization deal with security vulnerabilities as they arise with high priority and rapid response times. This paper first surveys the current situation and then proposes a approach where security is considered from the beginning of the Embedded systems development approach. Obviously, prevention is not the complete solution. A 5-level security strategy assures not only that a system has been properly designed in terms of security, but also that the liabilities of its designers are adequately covered.**

Index Terms— **Embedded systems, attacks, security issues, security planning**

I. INTRODUCTION

Embedded systems are computer systems that are part of larger systems and they perform some of the requirements of these systems. Some examples of such systems are automobile control systems, industrial processes control systems, mobile phones, or small sensor controllers. Embedded systems cover a large range of computer systems from ultra small computer-based devices to large systems monitoring and controlling complex processes. The overwhelming number of computer systems belongs to embedded systems: 99% of all computing units belong to embedded systems today.

Most of such embedded systems are also characterized as real time systems, which mean that the real-time properties such as response time, worse case execution time, etc., are important design concerns. These systems usually must meet stringent specifications for safety, reliability, availability and other attributes of dependability. Due to small size and requirements for mobility, but also extremely low production costs these systems require small and controlled resource consumption, and have limited hardware capacity. The increased complexity of embedded real-time systems leads to increasing demands with respect to requirements engineering, high-level design, early error detection, productivity, integration, verification and maintenance, which increases the importance of an efficient management of life-cycle properties such as maintainability, portability, and adaptability.

A.Computing Systems Security:

Computer security is the protection of computing systems against threats to confidentiality, integrity and availability [2]. In other words, a secure computing system provides three properties: confidentiality, integrity, and availability. All three are essential but depending on the application of system, one or two of them may receive more attention. Confidentiality means that information is disclosed only according to a security policy. Integrity means that information or system structure can be changed according to a security policy and availability means that services of system are available according to a security policy. The security policy addresses constraints on functions, flow among functions and constraints on users' access. Constraints on function and access may be correlated with time, location and/or other parameters. Users are external systems or human users. All of the details about users and constraints is explained precisely in a security policy.

B.Security model and definitions:

Vulnerability, threat, attack and safeguard or countermeasures make up a framework that enables arguing about computer security. Vulnerability is some weakness or fault in the system that could allow security to be violated. A threat is a circumstance or event that could cause harm by violating security. An adversary exploits vulnerability in the system to perform an attack. A safeguard is any technique, procedure, or other measure that reduces vulnerability. Safeguards make threats weaker or less likely. This framework ─vulnerability, threat, attack and safeguard─ is useful for analyzing and evaluating system security, also for deciding what safeguards to use.

II. LITERATURE SURVEY

Embedded device security needs to be integrated into the development lifecycle of the product rather than being an afterthought. The following are high level guidelines that embedded systems designers should consider when addressing security. This is not a prescriptive methodology, but intended to highlight an approach that looks at embedded security as a development lifecycle issue from requirements management, architecture and design and maintenance.

Manuscript received Aug 19, 2016

E. Pradeep, Assistant professor, Kamala Institute of Technology & Science, singapur, Huzurabad,Telangana,India

M. Sarika, Assistant professor, Kamala Institute of Technology & Science, singapur, Huzurabad,Telangana,India

A.Why Embedded Security

Making embedded systems secure is not only to protect resources and assets; it also provides opportunities for new services and new businesses. In [3] it is argued why the security of embedded systems is important. More arguments are listed here that emphasize the significance of embedded systems security.

Pervasive security:

Embedded systems are becoming pervasive as they are becoming cheaper. Their networking degree also is growing to let them have a better synergy by sharing resources and connecting users. They also contain assets of different stakeholders. Networking, sharing resources and holding assets exposes embedded systems to a growing range of threats.

New look:

In the past decades a lot of research has been carried out in the area of information security and system security. Many mature and well studied solutions exist. Some of the solutions are applicable to embedded systems but some of them cannot be utilized. The common characteristics of embedded systems ─mobile and resource constrained systems─ enforce researchers to take a new look at current solutions. For example, security solutions that consider the life-cycle of software do not consider the disposal phase as we would do in embedded systems because software does not have a disposal phase.

Safety:

Application of embedded systems in areas such as health care, avionics, or car industry where humans are involved raises the issue of safety. For example, the violation of integrity and availability of an artificial hearth, brake of a car and navigation system of an airplane may have disastrous consequences [4]. Attacks are turning from digital-data attacks to human attacks.

Financials:

M-commerce is followed by e-commerce, where mobile devices are the main player in financial transactions. Smart cards with e-wallet function or micropayments are examples of embedded systems in finance. There is enough incentive to break into these systems and there is high benefit for financial institutes to protect their systems.

New business model:

There will be many new applications or business models that strongly depend on the security techniques of embedded systems e.g., pay-TV, video on demand or time-limited services. Investors will invest in these businesses when they are sure their revenue is properly protected.

Privacy:

Some embedded systems are able to sense and capture a huge amount of data about location or status of a user to provide them some services. For example GPS systems process a lot of data about whereabouts of a user. By this information the location and personal information of a user can be observed easily which may affect the user's privacy.

Legal issues:

Some applications have legal concerns, e.g., e-voting or road-toll systems. They should meet applicable governmental standards to be acceptable for usage. They should not be manipulated easily. Producers should implement sound security techniques in their products to receive approval from authorities.

Secure identification of components

Third parties will contribute components and subsystems to a system. The secure identification of them is a major concern for a large number of applications. Counterfeiting products and parts (e.g. printer cartridges and ICs) are areas with urgent need for strong and secure device identification. Also, secure identification is important for access control.

Light-weight crypto:

Since resources are limited in embedded systems, some of the current security solutions are not applicable. New security solutions with less computational requirements, smaller size and lower energy consumption are necessary.

B.Embedded Security Challenges:

Designing secure embedded systems is not straightforward. There are many challenges that should be defied in order to secure them. Some of the challenges are explained below.

Heterogeneity

Most embedded systems are heterogeneous. They include software, hardware, mechanical components, optics, etc., and may consist of different components based on different technologies. Securing a heterogeneous system may be more challenging than a homogeneous system.

Complexity

Embedded systems have constraints which make the application of general security solutions difficult or impossible. Integrating security mechanisms with other functionality requirements is also not straightforward. Some embedded systems have real-time requirements, low power considerations and reliability requirements that should be considered besides security requirements. In fact, security is now a new metric that should be considered besides the other metrics. Meanwhile, security policies may violate other parameters [5]. These issues make security of embedded systems complicated.

Flexibility

Personalization of a system is a desirable feature for users. This implies provision of some flexibility and ability of customization in the system. On the other hand, this flexibility may impact the security of a system. It is challenging to find an equilibrium point of flexibility and security in a system. Also, it is desirable to have a flexible security policy in the system. Since security utilizes resources, in some environments we prefer to reduce the level of security to save resources.

Decentralized control

Not all embedded systems are controlled centrally; some of them are working independently. In some situations maintaining, repairing or restoration of them is done remotely. Some have adaptive behaviors in different

environments. These systems will communicate and interact in ways that were unforeseen during their design. In these scenarios, there should be self-adaptive, self-configuring or self-restoring techniques to preserve security.

Alternative energy sources

Side channel attacks are strong attacks based on information gained from the physical implementation of a cryptosystem, e.g., power consumption, electromagnetic leaks, timing information, or even sound [6, 7]. These can provide an extra source of information which can be exploited to attack the system. These attacks and their countermeasures have been studied for a long time [8]. Introduction of alternative energy sources e.g., light, vibration, walking, etc. might introduce new types of side channel attacks.

Time-to-market

The first product that reaches the market is the winner. Time-to-market is a criterion that forces producers to prevent applying well studied security solutions. In this case, producers emphasize more on legal enforcements. Security solutions which will not cause a delay in time-to-market are essential and valuable for producers.

Security Cost

Security needs more management which leads to higher costs. Having cheap security solutions would make systems more secure, since manufacturers avoid utilizing costly solutions. They prefer to add more functionality than securing current functionalities. Affordable security mechanisms are demanding.

We explained the importance of the security of embedded devices, and existing challenges. Current solutions are mostly as an afterthought and the security is not considered from the beginning. In the next section, it is discussed why security is an afterthought.

III. 5-LEVEL SECURITY IMPLEMENTATION FOR AN EMBEDDED SYSTEM:

The following are high level guidelines that embedded systems designers should consider when addressing security. This is not a prescriptive methodology, but intended to highlight an approach that looks at embedded security as a development lifecycle issue from requirements management, architecture and design and maintenance.

The five steps to improve embedded security are as follows:

End-to-end threat assessment – evaluate the security threats to the device in the various contexts of its lifecycle – development, operation, maintenance. End to end threat assessment of embedded devices needs to consider the following :

Complete product lifecycle analysis – include developer, manufacturer, operator, distributor, retailer and end consumer. Each of these may have impacts on devices security – it's more than just an end user problem. At this point it is important to sort out the priority of information assurance and cyber security. More and more embedded devices are handling confidential personal, business or government data. Protecting this data is an important security consideration.

Attack vector analysis – define and describe the possible entry paths for attacks into the system. First step is to define the physical entry path which may be more than just via network access, is it possible to compromise a device via physical access such as USB or serial ports? Once the physical entry points are defined, then the attack possibilities need to be evaluated. For example, if the device provided web access via TCP/IP port 80, are the vulnerabilities possible? Does the device use a firewall? If not, what TCP/UDP ports are open? Similarly for physical port access – does the device boot from USB if attached? An analysis is required of the permutations possible from physical access plus vulnerabilities possibilities.

Build a risk matrix – given the vast number of permutations possible from step 2) a risk assessment needs to be performed. What is the probability of an attack via this channel? What is the impact of exploitation via this channel? Note that it's important that risk to data and to device operation be handled separately. As discussed in a previous post, information assurance and cyber security may have different priorities based on your device and how it's used.

Create a mitigation strategy – based on the priorities in the risk matrix, decide the highest impact and highest probability threats. Create a strategy for each of these threats; in some cases there are architectural or design choices that can mitigate many threats. For example, partitioning a system into secure and "less-secure" segments might be a sound strategy.

Generate a security design specification – should be generated based on the above assessments. This is part of the overall system design, but should be treated with high priority.

Create a product lifecycle plan - an overall plan on designing, implementing, testing and maintaining security features and mitigations needs to be part of an existing or new development plan.

Security optimized design – make security a number one requirement and design consideration. Leverage modern separation and partitioning techniques, secure communications, and intrusion protection. Some of the most common ways that stolen IP can undermine both the immediate profits and long-term success of legitimate manufacturers include:

Hardware cloning A time-honored tradition in black-market electronics whereby a product's circuit boards, components, and often even its mechanical design are copied and used to produce unlicensed knock-offs. Modern cloning practices usually include use of pirated firmware and FPGA code. When grey-market manufacturers begin selling unauthorized knock-offs of propriety peripherals and accessories (ink cartridges, cables, batteries, and other consumables),the OEM loses a reliable revenue stream.

Overbuilding A relatively recent variant of cloning in which an authorized third-party assembly facility deliberately builds more units than a client has ordered with the intent of selling them through alternate channels. Unless a product was designed with provisions to secure it against this practice, overbuilding is nearly undetectable.

Reverse engineering Even if a competitor does not produce a copy of your product, stolen IP can allow them to inexpensively acquire proprietary technologies and features which give your products market differentiation.

Shortened design cycles Pirated designs allow would-be competitors to bring their products to market quickly,

reducing the time an innovative company gets to enjoy the marketing advantages and premium pricing that a product's unique features make possible.

Developing a complete security strategy

A complete security strategy must address traditional security concerns about securing the system's wired and wireless network connections. Authentication of network nodes, encryption of network data, message integrity protection, secure key management, and other traditional (but often overlooked) security measures are necessary. After all, your customers won't buy products unless they know their data, services, and infrastructure will be protected against intrusion, theft and sabotage.

Secure runtime selection – build your device from known secure components such as commercial-off-the-shelf (COTS) operating systems, middleware and tools

A number of techniques are available to inject malicious code during routine software upgrades. Once inside your system, the new code can turn into a convenient point of entry to your customer's (or your) network that can be used to gain access to sensitive consumer and corporate data.

The same techniques can be also used by those with more sinister intent to do physical harm. Several Pentagon studies and recent real-world incidents such as the Flame and Stuxnet viruses should serve as clear warnings that cyber-terrorism is a real possibility – especially in applications involving public infrastructure (utilities, communication, transportation) or mission-critical systems.

Part of the design process is to decide which of the issues listed above apply to your product and whether they are a primary or secondary requirement. Once the product's security requirements are defined, they can be used to develop a security strategy which serves as a tool for selecting the technologies and products best-suited to meet the application's unique combination of threats, performance requirements, and cost constraints. The security strategy should also consider whether the security solutions must be capable of being updated to deal with new threats as they emerge.

Depending on the level of security and performance required by your application, you can protect your system using a strategy based on software, hardware, or a mixture of both. Each of these strategies has its own unique advantages and drawbacks.

No security The simplest strategy is to not include any security in a design. In certain cases, the lower bill of material (BOM) and manufacturing costs, faster time-to-market, and lighter microcontroller (MCU) workload in the absence of security-related software outweigh the hidden costs of leaving a product vulnerable to hacking. But since some basic security measures can be implemented at little or no cost, few, if any, applications can afford to ignore security altogether.

Application protection – leverage white listing technology to exclude malware installation on the device.

Software-only solutions If the existing MCU has sufficient memory and processing cycles to support it, a security algorithm can be implemented in software. In most software-only security solutions, critical items such as secret keys are stored in the MCU's existing memory resources (EEPROM, Flash).

Advantages: These solutions are often perceived to be free, although they may have hidden costs due to additional development time and cost.

Disadvantages: Storing keys in unsecured memory resources puts them at risk of exposure. In addition, many firmware or software implementations of cryptographic algorithms are vulnerable to attack due to performance tradoffs, code size reduction efforts, use of general purpose hardware, and/or errors in the code.

Software/hardware hybrid solution (e.g. hardware on client, software on host) A client-side system's MCU can be augmented with a hardware security device that provides secure key storage and implements some, or all, of the security algorithm in hardware.

Advantages: Lower overall solution cost because no security device is required on host.

Disadvantages: In this solution, the host-side system's keys are stored in an unsecured resource, putting them at risk for interception, theft, or alteration. In addition, the software for the host-side algorithm may contain flaws in its implementation which leave it vulnerable to hacking techniques.

Defendable hardware-based solution An all-hardware solution includes tamper-resistant secure key storage devices used at all critical points in the system.

Advantages: With its keys securely stored in a hardened device specifically designed for the purpose and its security algorithm implemented in hardware, the resulting system is much more resistant to hacking without burdening the host processor. In addition, the development time required to bring a fully tested verified product to market is dramatically shortened.

Disadvantages: Many designers avoid all-hardware solutions because they are perceived as adding potentially unnecessary cost to a design.

Development lifecycle and tools – consider security to be part of the entire lifecycle of the device and plan for updates and security fixes well into the product's lifespan

The life-cycle of an embedded system consists of three phases: development, use and disposal.

The development phase includes all activities from the requirement specification of a product to the decision that the system has passed all acceptance tests and is ready to be delivered. The development phase also consists of some sub-phases such as: Requirement specification, Design, Production, Product shipment and Support/maintenance.

The use phase of a system's life-cycle begins when the system is accepted for use and starts the delivery of its services to users. Use phase consists of alternating periods of correct service delivery, service outage, service shutdown and maintenance.

The disposal phase is the disposal of an embedded system including media, software, components and data stored on the device. It starts when the system is no longer used or it no longer delivers services. It can also be transferring of the system from one person to another. The disposal phase is essential to prevent inadvertent release of data, information, or software. Security consideration in this phase is beneficial to protect sensitive information from disclosure and adhere to copyright, statutory, and regulatory requirements. In this

phase, if user is not going to use the system anymore and he/she discards it, the availability of the system is not of his/her concern, although if the system is transferred to another user, its availability is the new user's concern. In disposal phase, the confidentiality and integrity of system and the data stored on it, is not less important to the user than in use phase

CONCLUSION

Security of embedded systems is very important. Strong security mechanisms prevent damages and economical losses while also offering new business opportunities. However, sound security solutions are not attained easily. There are many challenges that should be defied. Although security consideration as an afterthought seems to have short-term incomes and less development difficulties, one simple security breach in a product could result in deletion from the market. A sound solution considers the security from the beginning and analyzes the life-cycle of the system to detect the vulnerabilities from the birth to the death of system. After discovering the sources and the reasons of vulnerabilities, safeguards should be embedded in the design methodology. Although designers and developers try hard to prevent all conceivable attacks, since the use environments and behavior of users cannot be predicted, it is not fully guaranteed that the system is secure. In addition to prevention techniques, tolerance techniques applied in the system help to provide service in presence of failure or attack. Removal and forecasting techniques help to have assurance in the security of the system.

REFERENCES

[1] Peter Marwedel, "Embedded System Design", 1st edition, Kluwer Academic Publishers:Hardbound, pp.1-8, 2003.
[2] Rita C. Summers, "Secure Computing: Threats and Safeguards", pp. 3-11, McGraw-Hill, 1997.
[3] Paar, C., Weimerskirch, A. "Embedded security in a pervasive world", Inf. Secur. Tech. Rep. 12,3, pp.155-161. Jan. 2007. [4] Carey Goldberg, "Heart devices vulnerable to hack attack", The Boston Globe, March 12, 2008. available online: http://www.boston.com/news/local/articles/2008/03/12/heart_devices_vulnerable_to_hack_attack/
[4] Srivaths Ravi , Paul Kocher , Ruby Lee , Gary McGraw , Anand Raghunathan, "Security as a new dimension in embedded system design", In Proc. ACM/IEEE Design Automation Conf., pp. 753-760, June 2004.
[5] Weingart S., "Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses", Workshop on Cryptographic Hardware and Embedded Systems, 2000.
[6] J. J. Quisquater, D. Samide, "Side channel cryptanalysis", In proceedings of the SECI 2002, pp. 179-184.
[7] Ravi, S., Raghunathan, A., and Chakradhar, S. ,Tamper Resistance Mechanisms for Secure Embedded Systems", In Proceedings of the International Conference of VLSI Design. pp 605-611, 2004.
[8] Srivaths Ravi, Anand Raghunathan, Paul Kocher, Sunil Hattangady, "Security in embedded systems: Design challenges", ACM Transactions on Embedded Computing Systems (TECS) ,Volume 3 , Issue 3, Pages: 461 - 491, 2004.
[9] David Hwang, Patrick Schaumont, Ingrid Verbauwhede, Shenglin Yang, "Multilevel Design Validation in a Secure Embedded System", IEEE Transactions on Computers archive, Pages: 1380 - 1390, 2006.
[10] Joe Grand, "Practical Secure Hardware Design for Embedded Systems", Proceedings of the 2004 Embedded Systems Conference, San Francisco, California.
[11] Eric Uner, "A Framework for Considering Security in Embedded Systems", Embedded.com, Sept. 2005.
[12] Wayne Jansen , Serban Gavrila, Vlad Korolev, Thomas Heute, Clément Séveillac, "A Unified Framework for Mobile Device Security", Proceedings of the International Conference on Security and Management (SAM'04), pp. 9-14, June 2004. [14] Ingrid Verbauwhede1 and Patrick Schaumont, "Design methods for Security and Trust", Design, Automation & Test in Europe Conference & Exhibition, pp. 1-6, DATE '07, 2007.
[13] Divya Arora, Srivaths Ravi, Anand Raghunathan and Niraj K. Jha, "Architectural Enhancements for Secure Embedded Processing", NATO Workshop on Security and Embedded Systems, VOL 2, pp. 18-25, August 2005.