

Innovative Approaches to Enhance Security in Embedded Systems

Amit Marmat^{1*}, Gulshan Waghela²

¹Assistant Professor, Department of Electronics Engineering, School of Engineering & Technology, Vikram University, Ujjain, India

²Bachelor of Engineering Scholar, Department of Electronics Engineering, School of Engineering & Technology, Vikram University, Ujjain, India

Abstract: This review paper shows a continuous advancement and revolution in the field of embedded system and security. In present days various technologies have been proposed towards embedded system, wide range of technologies across an array of industries where embedded system is used, so the security of the system is very necessary to prevent from malicious activity. This paper presents the present trends and classification of embedded system with security network approach. Any more mobile embedded system is also commonly used by the different individuals. But the risk of different types of attack on embedded system network is possible. To enhance the system security from network attack different countermeasures have been done. This countermeasure will improve the system stability and quality towards the users. Different layers of security have been used to protect user's data; called Layered security system.

Keywords: Embedded system, Network, Countermeasures, Architecture, PAM framework, Lightweight Directory Access Protocol (LDAP).

1. Introduction

The first modern, real-time embedded computing system was the Apollo Guidance Computer, developed in the 1960s by Dr. Charles Stark Draper at the Massachusetts Institute of Technology for the Apollo Program.

Basically, embedded systems are the combination of both software and hardware which performs operation of particular tasks. The performance of the system depends on the function of hardware like microprocessors, microcontrollers, memory, input/output communication interfaces and power supply. The system make use of the printed circuit board (PCB) they are programmed with software which controls their operation and commands data to utilize input/output communication interfaces and memory, which terminally produces outputs valuable to the user.

Basically, Embedded system software has fixed and partial flexibility towards users. It performs a specific task constantly as per the set commands. Therefore, embedded system refers as a system that is controlled by a computer that locates inside the system. It is a special computer system which is application centered, based on the computer technology, combination of hardware and software be bespoke to match exact need of application system on function, reliability, cost, volume, power consumption etc. For this a processor is used called embedded

processor control board that controls the program which feed in ROM. Another is the mobile embedded system, which is now a day's become more popular because of its compactness and wide area of serving. It is not considered to be computers. It is usually based on microcontrollers. According to their usage these can be categorized into three categories as small, medium and large. Embedded systems are not standalone but these are used within a complex device.

As compared with the special computer system, consisting few limitations. At the same time, its workflow is also limited. Mobile embedded system is smaller has few resources. At the same time, its functionality also has some limitations as compared with computer system; mobile embedded system has relatively components and less storage space facility. It is also considered as a more vulnerable to system security attacks and the system security performance correspondingly is also low. Therefore, the embedded systems and computer systems are differ greatly in terms of structure and components used in, that tends to some shielding amid embedded units and network devices, so hidden network attacks in a system may be filtered by the network function to considerable extent. The different parameters of attack are investigated.

2. Security Threats

In mobile phones the first malware is cabir in 2004; its main feature is that its ability to use Bluetooth protocol in 10 meters radius to sends all personal data of the users to requested device. Where in industries STUXNET is the first malware found in 2010, capable to hit industrial infrastructure which permit hackers to control critical system. This is due to the most of these devices are Internet enables but the problem arises when it starts interfering our personal information and exposing the information to unauthenticated agents.

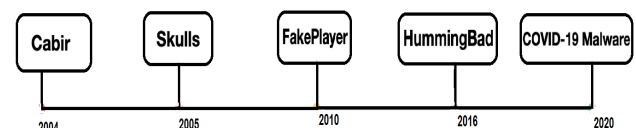


Fig. 1. Mobile malware timeline

The above figure shows the timeline of the security attacks

*Corresponding author: marmatamit@gmail.com

on the cell phones. Cabir in 2004 as already discusses, Skull in 2005 in this the apps icons are replaced by the image of skulls, in august 2010 FakePlayer make fake windows media player icon, HummingBad in 2016 which clicks automatically on intrusive advertisements and COVID-19 Malware is the latest one in the year of 2020, where the attackers used the subject as COVID-19 in variety of malicious campaigns including email spam, BEC, malware, ransomware, and malicious domains.

3. System Security Requirements

Embedded system performs very important task, which can be hacked by mischievous entities. Embedded is a very complex system in which various bodies has been involved to perform a principle and used in vast applications. So that the security demand also vary based on the purpose of the system. In modern cell phone there are various features like wireless voice, multimedia, and data communications. Example there are many cells phones which runs on the different operating system like Apple iOS, Android and Microsoft iOS. In this the cellular services provider also included because they are also use different handling devices from different manufacturers and same as the content provider on the cell phones, while the primary concern is the user's data security.

The main concern of security rise when the user send or receive sensitive information by using public networks, so the hackers may easily perform malicious activity and takes personal data. From an end-user perspective, common security requirements of embedded systems are crucial to ensure reliability, safety, and privacy. Here are some key aspects:

1. *Data Protection*: Users expect embedded systems to securely handle their data. This includes encryption of sensitive information both at rest and in transit, ensuring that unauthorized access or tampering is prevented.
2. *Authentication and Access Control*: Secure authentication mechanisms (such as passwords, biometrics, or multi-factor authentication) should be implemented to ensure that only authorized users or devices can access the system's functionalities and data.
3. *Secure Communication*: Embedded systems often communicate over networks (wired or wireless). Secure protocols (like TLS/SSL) should be used to protect data exchanged between the embedded device and external systems, preventing interception or manipulation by malicious actors.
4. *Update and Patch Management*: Regular updates and patches should be provided by manufacturers to fix vulnerabilities and improve security. End-users rely on these updates to keep their embedded systems protected against emerging threats.
5. *Physical Security*: Depending on the application, physical security measures may be necessary to prevent unauthorized access to the device itself. This could include tamper-resistant packaging or hardware security modules.
6. *Secure Boot and Firmware Integrity*: Ensuring that the

embedded system boots securely and that its firmware hasn't been altered is essential. Secure boot mechanisms and digital signatures can help verify the integrity of the boot process and firmware updates.

7. *Privacy Considerations*: Embedded systems should respect user privacy by limiting data collection to what is necessary for operation, providing clear privacy policies, and ensuring that data is anonymized or pseudonymized where possible.
8. *Resilience to Attacks*: Embedded systems should be designed to withstand and recover from attacks, such as denial-of-service (DoS) attacks or attempts to exploit vulnerabilities. This resilience ensures continuity of service and user trust.
9. *Auditing and Logging*: Keeping audit logs of system activities helps in detecting and investigating security incidents. End-users benefit from transparent logging practices that allow them to understand system behavior and identify anomalies.
10. *User Education and Awareness*: Lastly, educating end-users about safe practices, such as setting strong passwords or recognizing phishing attempts, enhances overall system security. Awareness programs can empower users to take an active role in protecting their embedded systems.

Embedded systems must meet robust security requirements to safeguard user data, maintain operational integrity, and ensure a trustworthy user experience. Manufacturers and developers play a critical role in implementing these measures to meet the expectations and needs of end-users in diverse application environments.

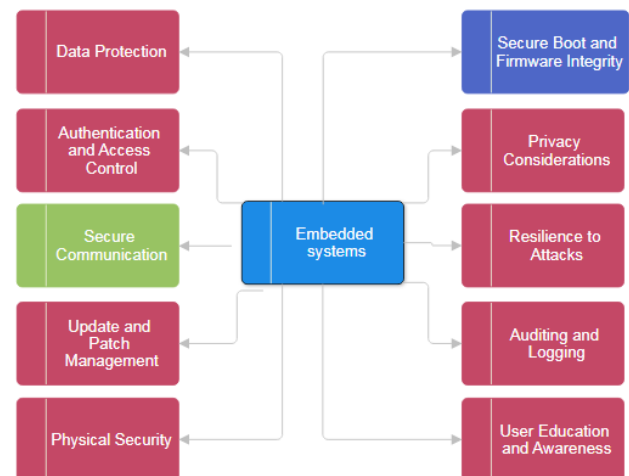


Fig. 2. Embedded system framework

4. Countermeasures

Cryptographic algorithms form the backbone of secure communication and data integrity in the digital world. Here's an overview of three fundamental classes of cryptographic algorithms:

- 1) *Symmetric Ciphers* (for efficient bulk data encryption): Symmetric ciphers use the same key for both encryption and decryption of data. They are

characterized by their efficiency and speed, making them suitable for encrypting large amounts of data.

- 2) *Asymmetric Ciphers (for secure key exchange, digital signatures, and communication over untrusted networks)*: Asymmetric ciphers use a pair of keys: a public key and a private key. The public key is freely available and used for encryption, while the private key is kept secret and used for decryption. This approach addresses the key distribution problem in symmetric cryptography. Key characteristics of asymmetric ciphers include. Asymmetric ciphers are computationally more intensive than symmetric ciphers but provide significant advantages in key management and secure communication over entrusted networks.
- 3) *Secure Hash Algorithms (for data integrity verification and password storage)*: Secure hash algorithms, or hash functions, transform input data into a fixed-size output (hash value or digest) in such a way that any change to the input produces a completely different output. Hash functions are designed to be one-way, meaning it should be computationally infeasible to reverse the process and obtain the original input data from its hash value.

These classes help in designing secure systems and protocols that rely on cryptography to protect sensitive information in the digital age.

5. Embedded Processing Architectures for Security of User Data

Embedded processing architectures play a crucial role in ensuring the security of user data across a wide range of devices, from IoT devices to mobile phones and embedded systems in critical infrastructure. Here's an exploration of how these architectures contribute to data security.

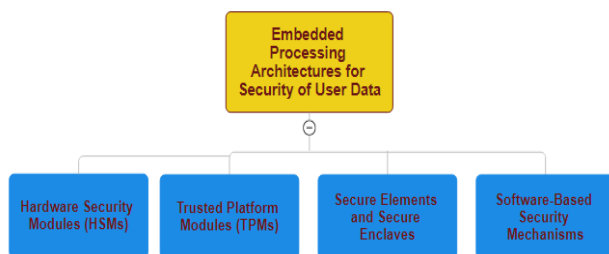


Fig. 3. Embedded system processing architecture framework

1. *Hardware Security Modules (HSMs)*: Hardware Security Modules are dedicated hardware devices specifically designed to manage and safeguard digital keys and perform cryptographic operations securely. They provide:

- *Physical Security*: HSMs are tamper-resistant and often include features like secure enclosures, environmental sensors, and mechanisms to detect and respond to physical attacks.
- *Key Management*: HSMs handle key generation, storage, and encryption/decryption operations,

ensuring that sensitive cryptographic keys never leave the secure hardware boundary.

- *Use Cases*: They are widely used in industries requiring stringent security measures, such as financial institutions for secure transactions and compliance with cryptographic standards like FIPS (Federal Information Processing Standards).

2. *Trusted Platform Modules (TPMs)*: Trusted Platform Modules are embedded microcontrollers that provide a hardware-based root of trust for securing devices and data. They offer:

- *Secure Storage*: TPMs store sensitive data such as cryptographic keys, passwords, and certificates in a secure manner, protecting them from unauthorized access.
- *Remote Attestation*: TPMs enable devices to prove their identity and integrity remotely, which is crucial for establishing trust in IoT networks and cloud environments.
- *Security Functions*: They support secure boot, measured boot, and sealing/unsealing operations, ensuring that devices only run trusted software and protect against firmware attacks.
- *Use Cases*: TPMs are integral to endpoint security in PCs, servers, and IoT devices, ensuring data confidentiality, integrity, and authenticity.

3. *Secure Elements and Secure Enclaves*: Secure Elements and Secure Enclaves are dedicated hardware or isolated areas within a processor that provide secure execution environments for sensitive applications and data:

- *Isolation*: They create isolated execution spaces protected from the main processor and other applications, ensuring that even if the main system is compromised, sensitive data remains secure.
- *Secure Processing*: Secure elements/enclaves perform cryptographic operations and handle sensitive data without exposing it to the rest of the system.
- *Use Cases*: They are used in mobile phones for secure payment transactions (e.g., using Apple's Secure Enclave), in IoT devices for securely storing device credentials, and in cloud environments for protecting sensitive workloads.

4. *Software-Based Security Mechanisms*: Beyond dedicated hardware, embedded processing architectures also employ various software-based security mechanisms:

- *Secure Boot*: Ensures that only trusted firmware and software components are loaded during the boot process, preventing unauthorized software from executing.
- *Encryption Libraries*: Software libraries provide implementations of cryptographic algorithms for data encryption, ensuring confidentiality and integrity.
- *Access Control and Authentication*: Embedded systems implement access control mechanisms and authentication protocols (like OAuth, JWT) to verify the identity of users and devices.

These embedded processing architectures collectively form a layered approach to securing user data across diverse devices and applications. They enable robust security measures such as encryption, secure key management, tamper resistance, and trusted execution environments, essential for protecting sensitive information in today's interconnected and data-driven world. Integrating these technologies ensures that user data remains safe from both physical and digital threats, maintaining trust and compliance with privacy regulations.

6. The Design Methodology and Tool requirements for Embedded security Systems

The Designing secure embedded system requires a comprehensive methodology and specific tools to address the unique challenges and threats posed by the embedded environment. Here's a short note on the design methodology and tool requirements for embedded system security:

A. Design Methodology

1. *Threat Modeling*: Identify potential threats and vulnerabilities specific to the embedded system, considering both hardware and software aspects. This involves understanding potential attack vectors and the assets (data, functionality) that need protection.
2. *Security Requirements Specification*: Define security requirements early in the design phase based on the threat model. These requirements should cover aspects like authentication, encryption, access control, secure boot, and secure communication protocols.
3. *Secure Design Principles*: Employ secure design principles such as least privilege, defense-in-depth, and separation of privilege to minimize the attack surface and mitigate potential vulnerabilities.
4. *Security Architecture Design*: Design a layered security architecture that includes hardware security (e.g., secure elements, TPMs), software security mechanisms (e.g., secure boot, encryption libraries), and application-level security controls.
5. *Secure Development Lifecycle (SDLC)*: Integrate security into the entire development lifecycle, including requirements, design, implementation, testing, deployment, and maintenance phases. Use secure coding practices and conduct regular security reviews.

B. Tool Requirements

1. *Static Code Analysis Tools*: Tools that analyze source code or binaries for security vulnerabilities such as buffer overflows, insecure cryptographic implementations, and other common coding mistakes.
2. *Dynamic Analysis Tools*: Tools that analyze software at runtime to detect vulnerabilities like memory corruption, injection attacks, and unauthorized access.
3. *Firmware Analysis Tools*: Tools specialized in analyzing firmware images for vulnerabilities, verifying secure boot mechanisms, and ensuring integrity.

4. *Encryption and Cryptographic Tools*: Libraries and tools for implementing strong encryption algorithms, key management, and secure communication protocols (like, OpenSSL, WolfSSL).
5. *Penetration Testing Tools*: Tools for conducting penetration tests to identify potential vulnerabilities and weaknesses in the embedded system's security defenses.
6. *Device Management and Monitoring Tools*: Tools for managing and monitoring deployed devices, including remote updates, logging, and intrusion detection.
7. *Security Compliance Tools*: Tools to verify compliance with security standards and regulations (e.g., FIPS, Common Criteria, GDPR).

C. Integration and Deployment

- *Tool Integration*: Integrate security tools into the development environment and CI/CD pipelines to automate security testing and ensure continuous security validation.
- *Training and Awareness*: Train developers and stakeholders on secure coding practices, security risks, and the proper use of security tools.
- *Update and Patch Management*: Implement processes for timely deployment of security updates and patches to address newly discovered vulnerabilities.

By adopting a systematic design methodology and leveraging appropriate security tools, embedded system developers can enhance the resilience of their systems against evolving threats and ensure the confidentiality, integrity, and availability of user data and critical functionalities.

7. Conclusion

While cryptography, network security and computer security have traditionally investigated various security issues, secure embedded systems design remains in its infancy in terms of research and widespread deployment. Taking a fresh look at the problem of providing security in cyberspace is not as challenging as it may seem. The good news is that unlike the problem of providing security in cyberspace, it is not as difficult to secure emerging environments or embedded systems. Securing the application-limited realm of embedded systems is increasingly viable in the short term. However, the limited resources inherent to embedded devices present considerable new obstacles to achieving the desired levels of security.

References

- [1] R. Nagar and S. K. Bhatt, "A Comparative analysis of a Conventional and cross phase UPQC for enhancing the Power Quality," *Int. J. Sci. Tech. Higher Educ. (IJSHRE)*, vol. 7, no. 7, 2018.
- [2] A. Marmat and R. Nagar, "Appropriateness of Parametric Bootstrap System for Appraise Mean Time to Failure of a Problematical System," *Int. J. Res. Eng. Sci. Manage. (IJRESM)*, vol. 3, no. 12, pp. 161–163, Dec. 2020.
- [3] N. Moodley, "The Heston model: a practical approach," Master's thesis, Faculty of Science, Univ. of the Witwatersrand, Johannesburg, South Africa, 2005.
- [4] H. He, C. Maple, T. Watson, A. Tiwari, J. Mehnen, Y. Jin, and B. Gabrys, "The security challenges in the IoT enabled cyber-physical systems and

- opportunities for evolutionary computing & other computational intelligence," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 1015–1021.
- [5] E. Levy, "Crossover: Online pests plaguing the off line world," *IEEE Secur. Privacy*, vol. 1, no. 6, pp. 71–73, Nov. 2003.
 - [6] M. Patrick, "How MCUs Actually Fight Security Attacks on Embedded Systems," *New Electronics*, Nov. 2019. [Online]. Available: <http://www.newelectronics.co.uk/electronics-technology/how-mcusactually-fight-security-attacks-on-embedded-systems/176924/>
 - [7] S. Ali, T. Al Balushi, Z. Nadir, and O. K. Hussain, "ICS/SCADA system security for CPS," in *Cyber Security for Cyber Physical Systems*, vol. 768, Muscat, Oman: Springer-Verlag, 2018, pp. 89–113.
 - [8] R. M. Lee, M. J. Assante, and T. Conway, "German steel mill cyber attack," ICS Defense Use Case (DUC), SANS Ind. Control Syst. (ICS), Swansea, U.K., Tech. Rep., Dec. 2014. [Online]. Available: https://ics.sans.org/media/ICS-CPPE-case-Study-2-GermanSteelworks_Facility.pdf
 - [9] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*. Hoboken, NJ, USA: Wiley, 2002.
 - [10] D. Papp, Z. Ma, and L. Buttyan, "Embedded systems security: Threats, vulnerabilities, and attack taxonomy," in *Proc. 13th Annu. Conf. Privacy, Secur. Trust (PST)*, Jul. 2015, pp. 145–152.
 - [11] P. V. Pham Van and N. N. Binh, "Embedded system architecture design and optimization at the model level," *Int. J. Comput. Commun. Eng.*, vol. 1, no. 5, pp. 345–349, Nov. 2012.
 - [12] A. A. Suzen, B. Duman, and B. Sen, "Benchmark analysis of jetson TX2, jetson nano and raspberry PI using deep-CNN," in *Proc. Int. Congr. Hum.-Comput. Interact., Optim. Robotic Appl. (HORA)*, Jun. 2020, pp. 1–5.
 - [13] National Science Foundation, "Cyber-Physical Systems (CPS) Nsf08611," 2008. [Online]. Available: <https://www.nsf.gov/pubs/2008/nsf08611/nsf08611.htm>
 - [14] P. Marwedel, "Embedded system hardware," in *Embedded System Design*, Dordrecht, Netherlands: Springer, 2011, pp. 119–175.
 - [15] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. 11th IEEE Int. Symp. Object Component-Oriented Real-Time Distrib. Comput. (ISORC)*, May 2008, pp. 363–369.
 - [16] S. Hameed, F. I. Khan, and B. Hameed, "Understanding security requirements and challenges in Internet of Things (IoT): A review," *J. Comput. Netw. Commun.*, vol. 2019, pp. 1–14, Jan. 2019.
 - [17] European Commission, "Cyber-Physical Systems | Digital Single Market," Aug. 2019. [Online]. Available: <https://ec.europa.eu/digital-single-market/en/cyberphysical-systems>
 - [18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial Internet of Things (IIoT): An analysis framework," *Comput. Ind.*, vol. 101, pp. 1–12, Oct. 2018.
 - [19] S. I. Tay, T. C. Lee, N. Z. A. Hamid, and A. N. A. Ahmad, "An overview of industry 4.0: Definition, components, and government initiatives," *J. Adv. Res. Dyn. Control Syst.*, vol. 10, no. 14, pp. 1379–1387, 2018.
 - [20] S. Ntalampiras, "Automatic identification of integrity attacks in cyberphysical systems," *Expert Syst. Appl.*, vol. 58, pp. 164–173, Oct. 2016.
 - [21] A. Pasquini, *Computer Safety, Reliability and Security*, vol. 1698. Berlin, Germany: Springer, 1999. [Online]. Available: <https://link.springer.com/book/10.1007%2F3-540-48249-0>
 - [22] S. Ravi, P. Kocher, R. Lee, G. McGraw, and A. Raghunathan, "Security as a new dimension in embedded system design," in *Proc. 41st Annu. Conf. Design Autom.*, 2004, pp. 753–760.