Journal of Theoretical and Applied Information Technology
20th June 2015. Vol.76. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SOFTWARE TAMPERING DETECTION IN EMBEDDED SYSTEMS – A SYSTEMATIC LITERATURE REVIEW

[1]ABDO ALI ABDULLAH AL-WOSABI, [2]ZARINA SHUKUR

[1] PhD Student, Faculty of Information Science and Technology, UKM, Malaysia

[2] Prof. Dr., Faculty of Information Science and Technology, UKM, Malaysia

Email: [1]abdoali8421@gmail.com , [2]zarinashukur@gmail.com

ABSTRACT

Embedded systems (ES) become available anywhere and anytime as an established part of our daily routines. Their usage in sense, store, process, and transfer our personal and private data, such as ATM card, modern cars system, mobile phones, and etc., become irreplaceable. Developers of these systems face significant challenges in code and information security. Whereas, software tampering is one of these challenges, code integrity detection is one of the main approaches used to defeat it. Checking code integrity achieves tamper proofing by method of identification of unauthorized alteration to recognize any tampered code is executed or tampered data are used. For the purpose of this paper, we perform a research methodology based on systematic literature reviews in-order to present different techniques/approaches of code integrity checking in embedded systems. We briefly survey a number of research studies (specifically between 2008 and 2014) related to this issue and present their proposed solutions. Obviously, there is no complete solution, and our aim by conducting this review is to contribute (even a modest effort) on fighting against software tampering.

Keywords: *Embedded Systems, Software Tampering, Tampering Detection, Software Integrity*

1. INTRODUCTION

A person of modern society relying on embedded systems has increased rapidly and the era of digital machines is gaining popularity among users and also devices/machines providers. The advancement in embedded systems applications is providing user-friendly services. Obviously, embedded systems used for implementing people's activities every day, such as the digital scale machines, digital cameras, modern cars, mobiles, ATM cards, and etc., are extensively used most of the time (almost) every day. At the same time, it attracts the attackers to get and exploit potential vulnerabilities in the system software, to gain unauthorized access, for utilizing the system or fetching the data illegally.

Basically, those potential vulnerabilities can be exploited in any ES that not well designed with anti-tamper techniques/algorithms. Embedded system tampering (EST) is such a latent threat which can be implemented by using certain tools for injecting the tampered/malicious codes. Mostly, an attacker's aim is to acquire control over some features of the software with an illegal alteration on the executable code and behavior. A real world example is the detected malpractices in fuel retail outlets in India. Regulation breakers in the fuel retail business deceived enforcers (and also clients) by tampering the software of electronic pumps with the help of certain dubious technicians [1].

Hence, tampering detection is gaining more attention and priority from embedded systems designers and developers [2]. Checking code integrity achieves tamper proofing by method of identification of unauthorized alteration to recognize any tampered code is executed or tampered data are used. As a result, unless appropriate technique/tool detects the system integrity of such devices, it will not lose customers, but also it may initiate undesirable social impacts. Indeed, such technique/tool does not prevent theft but instead discourages software tampers.

2. RELATED DEFINITIONS

System security can be divided into three security concerns; confidentiality, integrity, and availability. Confidentiality is security concern on preventing unauthorized users from illegal access to protected data/software throughout their life-cycles. The second security concern is integrity which

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ensures that the data/software can't be altered or deleted by unauthorized person. To a large extent, confidentiality is about preventing unauthorized reading of data and programs, while integrity is concerned with preventing unauthorized writing. Availability refers to ensuring the system/data will be accessible when an authorized user needs it, and without improper delay or undue halting. For instance, ensuring the system being protected from the denial of service attacks [3].

3. SYSTEMATIC LITERATURE REVIEWS

The aim of this paper is to review the current state of the art related to software tampering in embedded systems, so we have applied the following processes based-on systematic literature reviews mentioned by EBSE Technical Report [4]. The next subsections summarize the outcomes into: the research questions, the search process, inclusion and exclusion criteria, bibliography management and document retrieval, and data extraction and analysis. Also, an appendix shows summary of the related information extracted from the selected primary studies.

3.1 The Research Questions

The research questions is one of the key factor to conduct any systematic literature review [4]. For the purpose of our research, we have defined the following research questions:

Question 1: What are the potential threats may lead to software tampering in embedded systems?

Question 2: What are the existing techniques/tools could be used to detect software tampering in embedded systems?

Question 3: What are the current issues related to implement tampering detection in embedded systems?

3.2 The Search Process

Essentially, we have started our research by implementing the literature search on related studies, and we have found a number of research studies/papers using UKM Online Library and Internet services. However, the search process would be continued until the end of our study. This process has been conducted by using the search engines on several digital libraries, such as:

- IEEE Xplore,
- ACM Digital Library,
- Scopus,
- Science Direct,
- Springer Link, and
- Google Scholar

Key terms that closely related to this research project are: “Software tampering”, “Software integrity”, “Anti-tamper techniques/tools”, “Tampering detection”, and “Embedded systems”.

3.3 Inclusion and Exclusion Criteria

For the purpose of conducting this review, we have defined criteria to specify those studies to be included and those ignored/excluded studies. We have applied the following inclusion criteria:

- Research studies published between 2008 and 2014 that related to software tampering in embedded systems, and
- Those researches on techniques/tools related to software tampering detection.

On the other hand, we have excluded certain studies that:

- Informal published (no-defined or unknown journal or conference),
- Papers that irrelevant to the above research questions, and
- If there are duplicate versions of the same study (research), then the old version has been excluded.

However, if the research paper has been published in more than one journal or conference, then we have chosen the most complete version.

3.4 Bibliography Management and Document Retrieval

Mendeley Desktop 1.12.1 has been used to manage all citations and bibliography in-order to formulate our thesis report. The key terms defined above have been used for searching on the mentioned search engines. These selected studies appeared on journals/conferences have been scanned using their title and abstract. All relevant papers downloaded for further assessment and for data extraction. Table1 presents these research studies found while conducting searching and scanning on the mentioned digital libraries.

Table 1: *The Number of Found Research Studies*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Digital Library	Total number of found papers		
	Based on the defined key terms	Based on their titles	Based on their abstracts
IEEE Xplore	227	63	41
ACM Digital Library	12	09	08
Scopus	45	22	16
Science Direct	19	12	12
Springer Link	48	18	13
Google Scholar	170	32	18
Total	**521**	**156**	**108**

However, there are a number of researches that gathered from digital libraries are duplicated, so the old version have been neglected according to the exclusion criteria (mentioned above). Also, we have applied scanning and skimming techniques on these 108 papers in-order to capture the closely related studies. Accordingly, the total number of collected studies which have been under consideration for further review is about 50 researches. After reviewing these studies, we have summarized certain information from 15 papers which have been considered as the main selected studies (see the appendix: Table3).

3.5 Data Extraction and Analysis

It is essential to introduce the threat model to figure out the most convenient techniques for securing embedded systems. In general, embedded systems can be exposed to two types of attacks, regarding access to the embedded systems: remote attacks and physical attacks [5], and some other researchers [3], [6] summarize attacks against embedded systems as follows:

- Physical attacks: involves direct tampering with hardware components, such as: spoofing attacks, splicing attacks, and replay attacks.
- Side channel attacks: attempts to indirectly capture a secure data based on side channel information from the system operations, such as: timing attacks, power analysis, and fault analysis attacks
- Software attacks: exploiting potential vulnerabilities (like buffer overflow attacks) on many software, or by injecting malicious code (like Trojan horse programs or viruses) in-order to overwrite data on system memory or cause the processor to execute an unordered or malicious section (/s) of code.
- Network attacks: exploiting potential vulnerabilities on the transmission medium. It could be classified as active attacks (for instance DoS attack), and passive attacks (for instance monitor and eavesdropping, and traffic analysis).

Mainly, the lacking practices of joining efforts to establish security into the development methodology of information technology frameworks are a consequence of various elements that are related to the development process, or the environment in which the framework works. These embedded security challenges may include: heterogeneity, complexity, adaptability, decentralized control, time-to-market pressures, performance, energy efficiency (power consumption), and security cost [6], [7].

Hence, these elements may make designers hesitant to concentrate on information security accurately from early phases of system development, as it is considered as exercise in futility. In [7], the study defines a four level security strategy in-order to overcome these challenges. The proposed strategy consists of:

1. Preventing the event or presentation of vulnerabilities by enhancement of design and development processes,
2. Applying different tolerance methods, for example, vulnerability recognition, attack recovery, and self-adaptive procedures,
3. Vulnerability elimination during the development stage and during the utilization stage, and
4. Vulnerability predicting that accomplished by conducting a system assessment with respect to attack occurrence.

Furthermore, Babar *et al.* [6] identified the main features of the security framework and architecture consist of: lightweight cryptography, physical security, standardized security protocols, secure operating systems, future application areas, and secure storage.

Indeed, wide varieties of research carry out on three main types of solutions: hardware level solution, software level solution, and a combination of these two. For example, solution can be fulfilled by incorporating a hardware system as an external checker/tester, or on the product level where a trusted part of code exists to verify the targeted system security. Table 2 (in the appendix) summarizes the proposed solutions (software only approach, hardware only approach, and hybrid

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

approach) versus some of mentioned issues and challenges [6].

As shown above, we have presented the views of a number of researchers on: threat model, integrating security into the development methodology of ES, and the main types of the proposed solutions. Furthermore, more reviewed studies related to data and software security on embedded systems are listed below.

Invention by Schwartz *et al.* [8] suggests creating numerous hashes for executable software, the combination of which represent a signature of a whole executable software. Every individual hash corresponds to a particular part of the executable software (we call it code segment); such that every fractional digest is a signature of less than all of the code bytes. As a request to load a code segment (e.g., a page or something else) of the code into memory from a storage device, a verification hash of the code segment is calculated. Then the verification digest is contrasted with a fractional digest of the numerous hashes to verify integrity of the code segment.

Solution based on analyzing the real-time execution of code section has also been proposed. Zimmer *et al.* [9] utilize worst-case execution time (WCET) bounds data to recognize code tampering in real-time cyber-physical systems (CPS) by instrumenting the tasks and schedulers to confirm timing analysis results in-order to ensure that the execution time has not exceeds the expected time bounds.

Another research's objective is to develop a secure mechanism for ensuring the software integrity of the ES that does not need a peripheral hardware and infrastructure for generating the security key, storage and management, and gives an adequate security level. Also, saving the code in an encrypted format; cryptographic keys are created in real time, on interest, before the execution of the encoded code module [10].

On the other hand, Roger's outline applies a Parallel Message Authentication Code (PMAC) algorithms that takes into account utilizing a single hardware encryption module for both encryption and validation, henceforth it is system-resource wise and cost-effective [3]. They utilize the block cipher encryption as a part of their signature generation process (a CBC-MAC scheme is utilized). They introduce a mechanism for saving memory costs by securing various instructions and/or data blocks with a single signature.

Also, hardware monitor solution has been applied by a number of researchers. In [11], authors have presented a checking system to verify the proper software execution. Their protection is focused around monitoring the embedded system processor utilizing system resources that are discrete from the code binary, initially introduced for embedded systems in general. Since both encryption and validation are regularly computationally intensive, so, some authenticated-encryption algorithms have been suggested.

However, study has been fulfilled for remote outlines to permit verification without physical existence over a network. Remote verification needs a secure network protocol. For instance, Basile *et al.* [12] add hardware level components to externally verify the system integrity. They concentrate on recognizing if executed code has been altered by utilizing a field programmable gate array (FPGA) to construct a secure architecture. The researchers' objective is to design a system that makes it hard to conduct a successful real world attack. However, the researchers do note that this technique for security is not expected for high security systems such as military and government.

Whenever pure hardware solution or pure software solution fails, a combination of hardware and software solutions can be exploited. For example, Gelbart *et al.* [5] proposed a system that joint compiler-hardware approach to protect the software in the embedded system by encrypting data and code in the memory. They used FPGA to decrypt executables, and validate the code integrity before to be executed on the processor.

Additionally, Nimgaonkar *et al.* [13] introduce Memory Integrity Verification (MIV) to ensure data and code integrity. Data and code would be encrypted before inserting it into the memory and it would be decrypted after reading it. This prevents an attacker from observing or modifying the protected data/code. They consider the energy efficiency and have used the Merkle hash tree with time-stamps and time-stamp cache to reduce the energy consumption of the verification procedure.

Indeed, information security and privacy has direct influences on the current smart metering infrastructure, and intelligent vehicles. So, a number of studies carried out to discuss related issues on these two fields. For instance, seals can increase the level of system integrity since they detect tampering when it is occurred, and then protective action could be applied when seal is tampered with. Ransom *et al.* [14] present invention

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

that introduce the favored embodiments relate to a system and method of securing information formed, saved and transferred by an energy management (EM) device that is secured by a tamper-detection seal unit functioning to considerably detect illegal access to the EM device and specify any such action. In one embodiment, the system incorporates: producing the information; the information being categorized by an integrity; identifying when the tamper-detection seal unit specifies that unapproved access has happened; and securing the integrity of the information in response to that alteration (i.e., applying the assigned protective action (/s)).

A trusted computing base with secure storage and public key cryptography can also be mentioned here. In [15], the researchers outline multiparty processing units (local substations) to compute the sum of their energy consumption without revealing user's information. They proposed that the current smart metering structure ought to be reconsidered with a specific end goal to supplant a one-sided trust idea with a more versatile architecture in which meter devices have a trusted segment and have a certain level of independence.

Furthermore, Kumari *et al.* [16] propose usage control mechanisms for information that have to be shared over the network by smart meters connected to online social websites. They suggest sending information that is to be controlled just to information users that induce the information provider of having usage control mechanisms present and activated.

In terms of developing an applicable framework to protect code-integrity against an intended tampering, Nilsson *et al.* [17] have introduced secure firmware updates over the air (SFOTA) protocol on intelligent vehicle in-order to secure the transmission of the firmware code between the portal and a vehicle. The proposed framework facilitates code verification for firmware updates based on simple hash chain calculation on memory contents, challenge-response mechanism, and include random numbers to prevent pre-image attacks. However, the key management for using and storing the encryption key is not considered well as they assumed to use a single cryptographic key for all the car's control units.

On the other hand, mobile phones influenced by the same issues too. In [18], the study proposed the Specification Based Intrusion Detection Framework (SBIDF) that exploits whether there are hardware interrupts to classify a purely programming initiated activity and human initiated activity. It characterize specifications to identify the typical conduct pattern, and impose this specification to all third party applications on the cellphone at run-time by observing the inter-component interface pattern among critical modules. At whatever point these critical modules start up for implementation, the Authentication Module calculates an md5 hash over the TEXT portion of the module. So as to locate the integrity of the critical module, the Authentication Module compares the hash with a precomputed value of the hash of that segment. This precomputed hash value is available in the Specification Database. It could be computed by the phone stack supplier before delivery to the client, and after that statically saved in the Specification Database for future utilization.

4. REAL WORLD EXAMPLES

There exist a number of real world projects on data and code security in embedded systems. For instance, EVITA and INSIKA projects have been introduced and managed on European countries. The EVITA project (http://www.evita-project.org/) introduced three different security modules for protected vehicles on-board communications giving the principle for the prevention of external car connection. They introduced Hardware Security Modules (HSM) that facilitate means to secure the platform safety, to guarantee the integrity and secrecy of significant item, and to improve cryptographic processes; accordingly securing crucial resources of the system. The components of the HSM are: Symmetric Cryptographic Engine, Asymmetric Cryptographic Engine, Hash Engine, Random Number Generator, and Secure CPU.

Whereas, forgery on taxation information has become an important issue in all member states of the European Union, specialists have to propose a technology solution in-order to fight against manipulation of cash taking. On 2008, the German working group on cash registers started the INSIKA project (http://www.insika.de/en/) funded by the German Federal Ministry of Economics and Technology. Aim of this project is to introduce an applicable innovation for prohibiting information deception in Electronic Cash Registers (ECR). The main idea is based on digital signatures to detect any illegal modifications to the protected information. The basic idea of this project is based on asymmetric cryptography (public and private key algorithm), and SHA-1 algorithm.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CONCLUSION

Software can be altered by attackers or malicious users while running or at the rest. This kind of alteration is known as "Software Tampering". The primary objective of software tampering is to gain control over some aspects of the software with an unauthorized modification that alter the software program code and behavior. In this paper, we reviewed a number of researches related to software tampering in embedded systems. We focused on three main solutions: software only approach, hardware only approach, and hybrid approach. We then summarize different solutions, based on mentioned approaches, proposed by the selected studies. We emphasize that there is no complete solution, and the major aim of this paper is to contribute positively (even a modest effort) on combating such misuse acts.

APPENDIX

Table 3 summarizes certain information from a number of reviewed studies. This table represents the main ideas extracted from selected researches, which ordered chronologically by the published year.

REFERENCES

[1] G. Anand, "Electronic fuel pumps not tamper-proof," *TheHindu.com*, 27-Oct-2013.

[2] G. Myles and H. Jin, "A Metric-Based Scheme for Evaluating Tamper Resistant Software Systems," pp. 187–202, 2010.

[3] A. Rogers and A. Milenković, "Security extensions for integrity and confidentiality in embedded processors," Microprocess. Microsyst., vol. 33, no. 5–6, pp. 398–414, Aug. 2009.

[4] S. E. Group, "Guidelines for performing Systematic Literature Reviews in Software Engineering," 2007.

[5] O. Gelbart, E. Leontie, B. Narahari, and R. Simha, "A compiler-hardware approach to software protection for embedded systems," Comput. Electr. Eng., vol. 35, no. 2, pp. 315–328, Mar. 2009.

[6] S. Babar, A. Stango, and N. Prasad, "Proposed embedded security framework for internet of things (iot)," … Theory Aerosp. …, pp. 1–5, 2011.

[7] S. Mirjalili and A. Lenstra, "Security observance throughout the life-cycle of embedded systems," *Proc. 2008 Int. ...*, 2008.

[8] J. D. Schwarts, Y. L. Sie, and P. J. Hallin, "(12) United States Patent," vol. 2, no. 12, 2009.

[9] C. Zimmer, B. Bhat, F. Mueller, and N. Carolina, "Time-Based Intrusion Detection in Cyber-Physical Systems," pp. 109–118, 2010.

[10] A. Venþkauskas and I. Mikuckieno, "Generation of the Secret Encryption Key Using the Signature of the Embedded System," vol. 41, no. 4, pp. 368–375, 2012.

[11] S. Mao and T. Wolf, "Hardware Support for Secure Processing in Embedded Systems," vol. 59, no. 6, pp. 847–854, 2010.

[12] C. Basile, S. Di Carlo, and a. Scionti, "FPGA-Based Remote-Code Integrity Verification of Programs in Distributed Embedded Systems," IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev., vol. 42, no. 2, pp. 187–200, Mar. 2012.

[13] S. Nimgaonkar, M. Gomathisankaran, and S. P. Mohanty, "TSV: A novel energy efficient Memory Integrity Verification scheme for embedded systems," J. Syst. Archit., vol. 59, no. 7, pp. 400–411, Aug. 2013.

[14] I. D. Ransom, V. Ca, E.- Etheridge, B. B. Ca, S. J. Harding, V. Ca, M. F.- Hirschbolds, and T. M. Kiister, "(12) United States Patent," vol. 2, no. 12, 2010.

[15] F. D. Garcia and B. Jacobs, "Homomorphic Encryption," pp. 226–238, 2011.

[16] P. Kumari, F. Kelbert, and A. Pretschner, "Data Protection in Heterogeneous Distributed Systems : A Smart Meter Example," 2011.

[17] D. K. Nilsson, L. Sun, and T. Nakajima, "A Framework for Self-Verification of Firmware Updates over the Air in Vehicle ECUs," 2008 IEEE Globecom Work., pp. 1–5, Nov. 2008.

[18] A. Chaugule, Z. Xu, and S. Zhu, "A Specification Based Intrusion Detection Framework for Mobile Phones," pp. 19–37, 2011.

Journal of Theoretical and Applied Information Technology
20th June 2015. Vol.76. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPENDIX

Table 2: Comparison for Existing Solutions

Solution Approach	Solved issues (challenges)				Comments
	Cost	Flexibility	Performance	Power Consumption	
Software only	Yes	Yes	Partially No	No	Sometimes leads to overwhelm the processing capacity of the embedded system GPP
Hardware only	No	No	Yes	Yes	
Hybrid approach (Software and Hardware)	Partially Yes	Yes	Yes	Yes	Requires a clear vision of the complete system and a good interaction between the hardware designers, the software designers, and the security experts

Table 3: Summary of Selected Primary Studies

Sr. No.	Details	
1	**Year**	2013
	Author(/s)	Satyajeet Nimgaonkar, Mahadevan Gomathisankaran, and Saraju P. Mohanty
	Title	TSV: A novel energy efficient Memory Integrity Verification scheme for embedded systems
	Methodology	Timestamps Verification (TSV) instrument by utilizing Merkle hash tree. Throughout the write operation, the information scrambled through the memory encryption block before storing it to the off-chip untrusted memory. Throughout a read operation, the information is initially decoded and the hash location of the information and the hash of the information is examined against the hash that is saved in the hash cache. In the event that the hash equals then it is reasoned that the condition of the information is legal, if no then, it is presumed that the information is tampered.
	Finding(/s)	They proposed a novel energy effective methodology (called TSV) to afford Memory Integrity Verification (MIV) in ES. The energy savings with TSV approach can run from 36% to 81%, contrasted with base case results, based on the amount of timestamps that can be saved in the TS cache.
	Answer question(/s) (Refer to 3.1)	Q1, Q2, & Q3
2	**Year**	2012
	Author(/s)	Cataldo Basile, Stefano Di Carlo, and Alberto Scionti
	Title	FPGA-Based Remote-Code Integrity Verification of Programs in Distributed Embedded Systems
	Methodology	They propose the use of Field-programmable gate arrays (FPGAs) as a center of trust to securely calculate code integrity validations taking into account memory checksums, and introduces a protected protocol to convey them to an assigned confirmation entity. They utilized FPGAs, and code integrity verification using SHA-1 and AES.
	Finding(/s)	The utilization of reconfigurable computing to securely perform remote code trustworthiness confirmation of code in distributed ES. Likewise, the utilization of remote dynamic update of reconfigurable gadgets to raise the intricacy of mounting assaults.
	Answer question(/s) (Refer to 3.1)	Q1, Q2, & Q3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3	**Year**	2012
	Author(/s)	Algimantas Venckauskas, Nerijus Jusas, Irena Mikuckiene, and Stasys Maciulevicius
	Title	Generation of the Secret Encryption Key Using the Signature of the Embedded System
	Methodology	The secure encoding key of the program module is created from the headers of the code to be secured and from the hashes of the ES hardware and code parts (CPU, RAM, ROM, BIOS, OS, and so on), utilizing the fastest and most straightforward logical processes (XOR, OR, SHIFT).
	Finding(/s)	The introduced technique creates high entropy keys (entropy rate near to 1) with no extra hardware and structure cost. Likewise, The entropy of the secure encoding key produced by utilizing the hash generation functions based on OR and XOR processes and SHA-2 function is most elevated.
	Answer question(/s) (Refer to 3.1)	Q2, & Q3
4	**Year**	2011
	Author(/s)	Flavio D. Garcia, and Bart Jacobs
	Title	Privacy-Friendly Energy-Metering via Homomorphic Encryption
	Methodology	They utilize an integration of Pallier additive homomorphic encoding and additive secure sharing with public-key algorithm.
	Finding(/s)	They propose the No-Leakage protocol for protected connection with energy meters and for piracy identification (leak) in a security method. The existing intelligent metering framework must be re-examined so as to supplant a one-sided trust supposition by a more versatile structural planning where energy meters have a trusted segment and impose a certain level of independence.
	Answer question(/s) (Refer to 3.1)	Q2
5	**Year**	2011
	Author(/s)	Ashwin Chaugule, Zhi Xu, and Sencun Zhu
	Title	A Specification Based Intrusion Detection Framework for Mobile Phones
	Methodology	The study utilizes the md5 hash and description database to portray the typical conduct style, and observe the inter-component connection style between distinct modules.
	Finding(/s)	They propose a Specification Based Intrusion Detection Framework (SBIDF) to identify the unapproved malicious behavior (strange intrudes). The assessment of simulated conduct of actual malware demonstrates that they would be able to distinguish all forms of malware that endeavors to access critical services without having user's license.
	Answer question(/s) (Refer to 3.1)	Q2
6	**Year**	2011
	Author(/s)	Sachin Babar, Antonietta Stango, Neeli Prasad, Jaydip Sen, and Ramjee Prasad
	Title	Proposed Embedded Security Framework for Internet of Things (IoT)
	Methodology	Study and investigation of embedded protection, particularly in the field of Internet of Things (IoT). They outlined the protection needs taking into account computational time, energy utilization and memory needs of the systems.
	Finding(/s)	They propose an embedded protection framework and architecture that can assist designers and developers to convey more protected systems. Their result consider the security from the earliest starting point i.e. from the designing stage to the operation phase. Additionally, the researchers deduce that the suitable system-level study will empower the determination of the candidate solutions for the hardware and programming modules.
	Answer question(/s) (Refer to 3.1)	Q1, Q2, & Q3
7	**Year**	2011
	Author(/s)	Prachi Kumari, Florian Kelbert, and Alexander Pretschner
	Title	Data Protection in Heterogeneous Distributed Systems: A Smart Meter Example
	Methodology	Apply use control techniques at each area outside the energy filed. These

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

		techniques intercept each demand to get an information and permit the flow of information just if a confirmation of presence of an identical mechanism at the other side is given. Additionally, utilize digital certificates (approved by a certificate authority) for the protected verification of information requesters to perform the detective enforcement techniques. Likewise, encoding the usage controlled information and guaranteeing that just authenticated usage control techniques can unscramble the information.
	Finding(/s)	They presented a framework for enforcing information security in a diverse distributed system including of an intelligent meter linked to a web based social networks (WBSN).
	Answer question(/s) (Refer to 3.1)	Q2
8	**Year**	2010
	Author(/s)	Christopher Zimmer, Balasubramanya Bhat, Frank Mueller, and Sibin Mohan
	Title	Time-Based Intrusion Detection in Cyber-Physical Systems
	Methodology	Analyzing the execution time to identify an application's best case execution time (BCET) and worst case execution time (WCET) boundaries that permits check of a function's deadline. Their methodology supplements network-centric security with application-level intrusion recognition.
	Finding(/s)	They identify the execution of prohibited instructions in real time ES. Such intrusion disclosure uses data acquired by static timing check. For real time ES, timing boundaries on code areas are readily accessible as they are previously identified before the schedulability check. The study shows how to give micro-timings for numerous granularity scales of system code. Through boundaries monitoring of these micro-timings, they create methods to distinguish intrusions in a self-checking mode by the application, and by the OS scheduler.
	Answer question(/s) (Refer to 3.1)	Q1, Q2, & Q3
9	**Year**	2010
	Author(/s)	Douglas S. Ransom, E. Etheridge, Stewart J. Harding, Markus F. Hirschbolds, Theresa M. Koster, and Simon H. Lightbody
	Title	System and Method for Seal Tamper Detection for Intelligent Electronic Devices
	Methodology	They identify a wide varieties of existing security protocols, algorithms, and methods for revealing and reacting to system tampering in an Energy Management (EM) device, for example, seal, SSL, TLS, IPSec, RSA, DES, digital signature, audit log, hash function, and etc.
	Finding(/s)	Introducing a system and security methods to discover an unauthorized altering in an Energy Management device.
	Answer question(/s) (Refer to 3.1)	Q2
10	**Year**	2010
	Author(/s)	Shufu Mao, and Tilman Wolf
	Title	Hardware Support for Secure Processing in Embedded Systems
	Methodology	Analyze the binary code of an ES application and infer a control flow diagram. They utilize a checking subsystem that works along with the embedded processor. The monitoring module confirms that just processing procedures are performed that coincide with the initially installed code. Any unauthorized execution would disturb the manner of execution procedures, and consequently, alarm the monitoring module.
	Finding(/s)	Their outcomes demonstrate that exclusively depending on control flow data, as it has been carried out in the past, is not a proficient method for recognizing attacks. Rather, they have proposed a hash based style that utilizes low memory and can distinguish deviations from intentional processing within an individual instruction cycle.
	Answer question(/s) (Refer to 3.1)	Q1, Q2, & Q3
11	**Year**	2009

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

	Author(/s)	Austin Rogers, and Aleksandar Milenkovic
	Title	Security extensions for integrity and confidentiality in embedded processors
	Methodology	Their methodology is nearly totally hardware approach, need no compiler assistance and just low OS support. Integrity is guaranteed utilizing runtime check of cryptographically robust signatures embedded in the code and data. Information blocks are further secured from replay attacks by utilizing sequence values. The sequence values are secured utilizing a tree like structure. Secrecy is guaranteed by encoding instructions and information sections and signatures utilizing a dissimilar one-time pad (OTP) encoding technique.
	Finding(/s)	They proposed a number of cost effective architectural additions adequate for mid-range to high-end ES processors. These additions guarantee the integrity and secrecy of both code and information, presenting low execution overhead (1.86% for code and 14.9% for information).
	Answer question(/s) (Refer to 3.1)	Q1, Q2, & Q3
12	**Year**	2009
	Author(/s)	Olga Gelbart, Eugen Leontie, Bhagirath Narahari, and Rahul Simha
	Title	A compiler-hardware approach to software protection for embedded systems
	Methodology	Their approach has three components. The first is design: the utilization of assisting FPGA that they call it as the FPGA guard. The second is a back-end compiler module that controls the instructions such that every instruction segment has a mark. The third is a detection algorithm, executed on the FPGA, which inspects the marks of instruction segments to confirm legitimate execution (control flow and instruction-integrity verification). They utilize AES for encoding of information and code and CRC or SHA-1 calculation to facilitate integrity testing of data and instructions.
	Finding(/s)	They proposed CODESSEAL framework which depends on a compiler to append security to the system and on FPGA to dynamically check the code and its data at execution with less efficiency penalties.
	Answer question(/s) (Refer to 3.1)	Q1, Q2, & Q3
13	**Year**	2009
	Author(/s)	Jonathan D. Schwartz, Yu Lin Sie, and Philip Joseph Hallin
	Title	Systems and Methods for Validating Executable File Integrity Using Partial Image Hashes
	Methodology	Creating numerous fractional hash values of the code for verification prior loading the software. Every fractional hash of the hash values stand-for a hash of a particular segment of code. Then combine the code with its related fractional hash values into a system catalog or into a self-signed file.
	Finding(/s)	Introducing a framework and methods to reinforce code-integrity based on examining of numerous fractional hash values that corresponding to the code.
	Answer question(/s) (Refer to 3.1)	Q2
14	**Year**	2008
	Author(/s)	S. Hasan Mirjalili, and Arjen K. Lenstra
	Title	Security Observance throughout the Life-Cycle of Embedded Systems
	Methodology	Analyze the whole life cycle of ES and convenient countermeasures are integrated in the system design.
	Finding(/s)	They proposed a methodology that considers the security factor from the starting point of the system design of ES during their whole life cycle. A 4-level protection method guarantees not just that a system has been appropriately outlined in terms of security, additionally that the responsibilities of its designers are properly managed. While the usage situations and users' actions can't be anticipated, it is not completely ensured that the system is always secure.
	Answer question(/s) (Refer to 3.1)	Q3
15	**Year**	2008
	Author(/s)	Dennis K. Nilsson, and Lei Sun, Tatsuo Nakajima

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

	Title	A Framework for Self-Verification of Firmware Updates over the Air in Vehicle ECUs
	Methodology	They used virtualization mechanisms to aid a functional and a control system. The control system conducts a memory check of the downloaded firmware involves a challenge-response technique between the gateway and the control system to validate the downloaded binary. Likewise, computing a hash sequence corresponding to the memory data and a challenge. A random value is incorporated to generate randomness in the digest computation to prohibit an attacker from executing a replay attack. So, the last digest value is utilized as the check code.
	Finding(/s)	They have developed an architecture for self-validation of remote firmware updates in electronic control units (ECUs). The architecture comprises of a trusted gateway and a car. It is fundamental to not just confirm that the firmware has been downloaded properly but additionally to validate that the firmware has been precisely flashed to memory.
	Answer question(/s) (Refer to 3.1)	Q2