

Security Monitoring Approach in Embedded System

K. Rahimunnisa
Asst. Professor, Dept. of ECE
Karunya University
Coimbatore, India

Dr. S.Sureshkumar
HOD & Prof., Dept. of EEE
Karunya University
Coimbatore, India

Rincy Merrin Varkey
PG Scholar, Dept. of ECE
Karunya University
Coimbatore, India

ABSTRACT

Embedded systems have been developed greatly in terms of scaling and integration of devices. With each new generation the complexity of the application also increases with its enhancement. But little concern is been given over the secure program execution. Thus the software program becomes susceptible to attacks. There has been a rising need for an efficient security system for embedded system applications. For the past years various monitoring techniques have been proposed to verify the execution. This paper proposes a model for verifying the execution of the program during run-time using encryption algorithm. Run-time verification combines monitoring with formal methods by checking specified behavior at run-time.

General Terms

Embedded systems, Reliability, Security monitoring system.

Keywords: Encryption, Run-time monitoring, Instruction integrity.

1. INTRODUCTION

The integrity and security of embedded systems have been a major concern as they play a significant role in areas such as education, health care, gaming consoles, network security devices, ambient intelligence, consumer electronics, avionics, car industry, controllers in industrial plants, etc.. Embedded system is not only a hardware module but also comprises of software module. They are designed and developed for specific applications and not as general purpose computing device. Embedded systems have common characteristics such as they should be efficient in terms of power consumption, size, run-time requirements, weight and cost. For embedded systems used in applications like RFID, Smart Cards, Mobile e-commerce etc... security is an important concern as increasingly sensitive data being exchanged on the computer networks. This concern necessitates the use of security protocols to protect the sensitive data from unauthorized snooping or manipulation by adversaries. The methods usually in computers networks are firewalls or intrusion detection software which uses complicated rules and technique. On designing a security protocol during the system design process is not only less complicated but also cost efficient.

Usually, security is explained using two legitimate entities interacting with each other are referred to as A and B and the

adversary is called E. Data exchanged on the computer networks are susceptible to the follow adversarial threats:

1. E observes data being exchanged between A and B with the intention of finding out the content.
2. E modifies communication between A and B.
3. E impersonates as A to B, or as B to A.
4. E disrupts the network so that A and B cannot communicate with each other.

Therefore it's not only enough to verify logical correctness and timing constraints of a system but should be able to guarantee the level of security. Hence it is necessary to define a security mechanism with the following considerations like using independent resources, low complexity and faster detection. It is necessary for a security protocol to be confidential, to be able to detect unauthorized access and to prevent unauthorized entity to pose as a legitimate entity [10].

Embedded systems have certain constraints which makes it pron to attacks. It has limited processing capability due to lesser hardware and memory size. This makes it difficult to use anti-virus software and firewalls. It has limited power supply as it is battery operated device. This makes it more susceptible to attacks when connected to a network which implies the need for a security monitoring system.

The remaining part of the paper is as follows: section II briefly discusses on related works, section III gives an overview of the monitoring system and section IV explains the experimental results and section V gives the conclusion and future work.

2. RELATED WORKS

Over the years various methods have been found to ensure the security of the system in various aspects like physical, thermal or verifying the integrity of the program. Ravi et al. in [1] describes various techniques that can be implemented for physical security. For security of the system during run-time various approach using static analysis dynamic analysis. The paper [2] describes the use of control flow graph to verify the execution of the program at run time. This method is practical and compatible with existing software. Control flow graph can also be used with machine codes. It provides a useful foundation for security protocol. But the use of control flow graph method is vulnerable to attack when the same block structure as the original code will go undetected for a while during run-time. In paper [3] the permissible behavior of the

application is captured at different level of granularity namely inter procedural, intra procedural and instruction stream integrity. This approach however detects deviation after a few instruction cycle unlike in paper [9]. Another approach is the SAFES method in [4] is a reconfigurable architecture. A SAFES reconfigurable architecture is proposed based on the following i) reconfigurable security primitive, ii) reconfigurable hardware monitor and iii) hierarchy of security controllers. This approach enables in dynamically configuring the monitoring of the system thereby making it flexible for detecting various attacks. Another approach for embedded security is SAFE-OPS [5]. It uses a combination of compiler and architecture technique. The authors R.G. Ragel and S. Parameswaran in [6] discuss about IMPRES technique which is related to fault detection in bit flips. The method described by the authors is verifying the application memory by method of checksum however it can handle code injection as well as bit flips but only deals with a basic block in a program and does not verify the integrity of the each instruction.

The authors in [7] Kurthartha Patel and Shri Paremshwaran describes the method of using program map and trace file technique to verify the system during run-time to address code injection attacks. The method considers a system of N processors and FIFO is used to communicate between two processors. Each processor uses a FIFO queue structure to monitor processor. The program map is attained by static analysis which maps the flow of the program and a trace file is also generated. The trace file is generated by adding the execution time of the instructions in a block. This technique however is unable to detect data corruption. The work explained below verify the program execution where the hash value of the instruction at run-time is compared in the monitoring logic. This method however unable to detect data corruption. An approach to defending against buffer overflow attacks is described by Shao et al. in [8]

The work in this paper is related to [9] where hashing of each instruction is stored in the memory which is compared with real time values obtained. In [9] the author does mention intrusion detection in single or few instructions. This paper however focuses on improving the detection within a single instruction cycle but instead of using compressed hash value we use a 128-bit hashing function.

3. SECURITY SYSTEM

Our approach provides an efficient method to ensure the security of the system and to enable faster detection of attacks. It consists mainly of application file, memory, processor, MD5 module and monitoring logic as shown in Fig. 1. The application is the executable binary file of the application program used in the system. This is stored in the

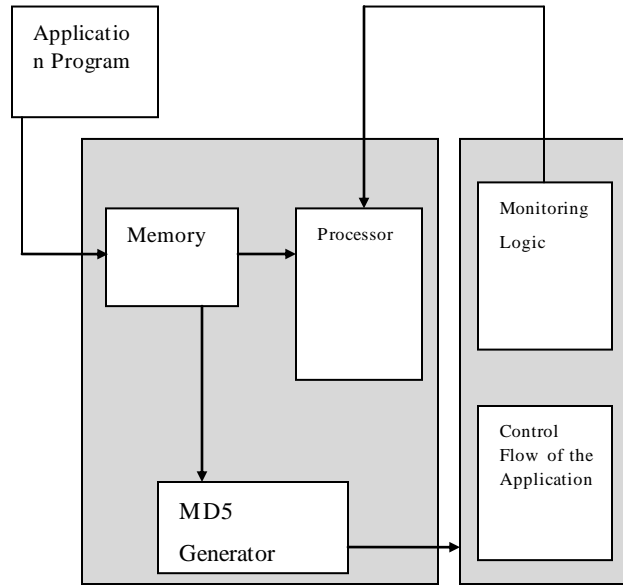


Figure 1. Block Diagram of the Architecture

memory. When each instruction is been simultaneously accessed from the memory, its corresponding hash value is generated which is then given to the monitoring logic. Here it compares the value of instruction with a predefined value for the corresponding instruction. The application is loaded into memory where it is stored as binary. When the processor fetches the instruction and executes it, the encrypted value of the instruction is generated using MD5 algorithm. This is done for verifying the instruction integrity.

Though there are other hashing algorithms like SHA1, SHA2, SHA512, etc.. MD5 is used as it has lesser computational rounds (i.e. 4) when compared to SHA (i.e. 80). This reduces the memory requirement for computation, enabling the available memory for utilization for the processor. On using the compressed hash values [9] it will reduce the memory space required for the storage but the reliability decreases as there is a slight probability for two instructions to have the same hash value, for which we use MD5. The control flow graph of the application is generated offline by the programmer and its corresponding encrypted value is stored in the monitoring logic. This is compared with encrypted value of each instruction at run time. The encrypted value is given to the monitoring logic. This is compared with the encrypted value from the control flow graph.

If there is any deviation in the program execution then an attack is detected. On the detection of an attack an interrupt is sent through the feedback to initiate shutdown. After which the system can recover by suitable recovery systems.

3.1 Message Digest (MD5) Algorithm

MD5 is a message digest algorithm developed by Ron Rivest at MIT. This has been the most widely used secure hash algorithm particularly in Internet-standard message authentication. The processing involves the following steps [10].

Step 1. Append Padding Bits

The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bit of less being a multiple of 512 bits long. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

Step 2. Append Length

A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^64, then only the low-order 64 bits of b are used. (These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.) At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words. Let M [0 ... N-1] denote the words of the resulting message, where N is a multiple of 16.

Step 3. Initialize MD Buffer

A four-word buffer (A, B, C, D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal

- Word A: 01 23 45 67
- Word B: 89 AB CD EF
- Word C: FE DC BA 98
- Word D: 76 54 32 10

Step 4. Process Message in 16-Word Blocks

It defines four auxiliary functions, each take three 32-bit words as input and produces one 32-bit word as output. The functions used are as follows:

- $F(X, Y, Z) = (X \wedge Y) + (\text{not}(X) \wedge Z)$
- $G(X, Y, Z) = (X \wedge Z) + (Y \wedge \text{not}(Z))$
- $H(X, Y, Z) = X \text{ xor } Y \text{ xor } Z$
- $I(X, Y, Z) = Y \text{ xor } (X + \text{not}(Z))$

4. EXPERIMENTAL RESULTS

A simple password verification program is used here as the application program. The program verifies the password entered by the user. If the password is incorrect a flag is generated which indicates an error has occurred. Date of birth is entered next after the password is entered correctly. In case if date of birth entered is incorrect a flag is again generated to indicate the error. On providing the correct entries the control is transferred to the main application of the program.

The system is modeled using TurboC. The flowchart of the above program is as shown in Fig. 2.

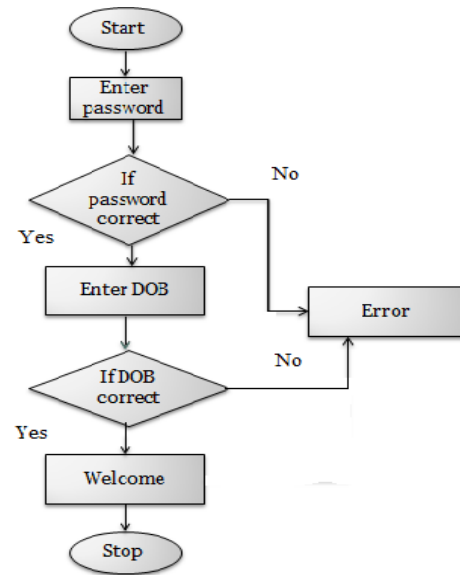
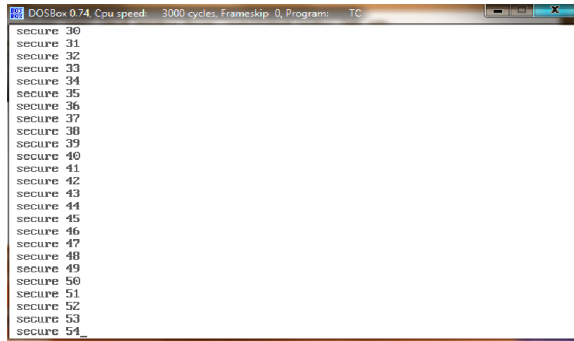


Figure 2. Flow chart of the password program

The encryption program uses MD5 algorithm, it reads each instruction from the program file. The encrypted value of each line is generated and is verified with the corresponding pre-defined hash value as shown in Fig. 3. The predefined hash value is used to forms the comparison logic which verifies the integrity of the instruction during run-time. Any deviation from the value stored in comparison logic will be detected that indicates the instruction is been comprised or the system is under attack. As each instruction is been executed, its corresponding hash value is

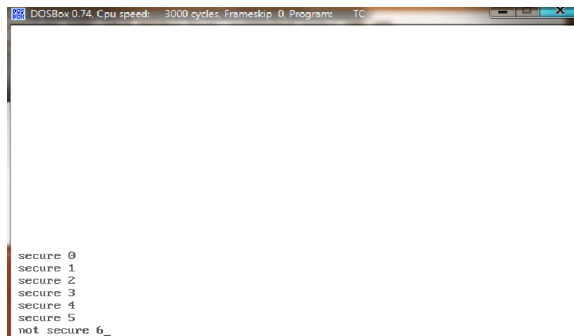


Figure 3. The hash value of each instruction



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip: 0, Program: TC
secure 30
secure 31
secure 32
secure 33
secure 34
secure 35
secure 36
secure 37
secure 38
secure 39
secure 40
secure 41
secure 42
secure 43
secure 44
secure 45
secure 46
secure 47
secure 48
secure 49
secure 50
secure 51
secure 52
secure 53
secure 54
```

Figure 4. The output when the program behaves normally



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip: 0, Program: TC
secure 0
secure 1
secure 2
secure 3
secure 4
secure 5
not secure 6_
```

Figure 5. The output when the program is under attack

calculated simultaneously which is compared with the hash value stored in the monitoring logic. The output in Fig. 4 shows a scenario when the program behaves according to its permissible behavior. If an error occurs it stops the execution indicating an attack as shown in Fig. 5. Here on varying the binary value in a particular memory location, due to the mismatch that arises in the monitoring logic the execution stops indicating the attack.

5. CONCLUSION

We have presented an efficient security protocol which enables faster detection. Here the security is ensured in the highest level. Any deviations in execution of the application program will initiate a feedback which halts the execution of the system. We are presently working on combining both hash and address pattern monitoring technique. This method provides a better detection of buffer overflow as well as verifies instruction integrity.

6. ACKNOWLEDGMENTS

The authors would like to thank Karunya University for the facilities provided for research.

7. REFERENCES

- [1] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper Resistance Mechanisms for Secure, Embedded Systems," Proc. 17th Int'l Conf Very Large Scale Integration Design (VLSI Design '04), pp. 605-611, Jan 2004.
- [2] M. Abadi, M. Budi, U. Erlingsson, and J. Ligatti, "Control-Flow Integrity Principles, Implementations, and Applications," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 340-353, Nov. 2005.
- [3] Divya Arora, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha, "Hardware Assisted run-Time Monitoring for Secure program execution on embedded processors" IEEE Transaction Very Large Scale Integration Systems (VLSI), vol.14.no.12,pp.1295-1307,Dec.2006
- [4] G. Gogniat, T. Wolf, W. Burleson, J.-P. Diguët, L. Bossuet, and R. Vaslin, "Reconfigurable Hardware for High-Security/High-Performance Embedded Systems: The SAFES Perspective," IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol. 16, no. 2, pp. 144-155, Feb. 2008
- [5] J. Zambreno, A. Choudhary, R. Simha, B. Narahari, and N. Memon, "SAFEOPS: An Approach to Embedded Software Security," ACM Trans. Embedded Computing Systems, vol. 4, no. 1, pp. 189-210, Feb. 2005.
- [6] R.G. Ragel and S. Parameswaran, "IMPRES: Integrated Monitoring for Processor Reliability and Security," Proc. 43rd Ann. Conf. Design Automation(DAC), pp. 502-505, July 2006.
- [7] Kurthatha Patel, Sri Paremshwaran and Seng Lin Shee "Ensuring Secure program execution in multiprocessor embedded systems ", in Proceedings of the 5th IEEE/ACM international conference on Hardware/software co-design and system synthesis, 2007, pp. 57-62
- [8] Z. Shao, Q. Zhuge, Y. He, and E.H.-M. Sha, "Defending Embedded Systems Against Buffer Overflow via Hardware/Software," Proc. 19th Ann.Computer Security Applications Conf. (ACSAC), pp. 352-363, Dec. 2003
- [9] Shufu Mao and Tilman Wolf, Senior Member, IEEE, "Hardware Support for Secure Processing in Embedded Systems" IEEE transactions on computers, vol. 59, no. 6, June 2010
- [10] William Stallings, "Cryptography and Network Security", PHI Publishers, Second Edition 2007,