ISSN No:-2456-2165

A Review Paper: Embedded Security

R. Binusha Rachel
PG Scholar, ME Applied Electronics
PSG College of Technology

Abstract:- Embedded devices has its hands over all fields, however they are easily vulnerable for external attacks, any third party can easily cause its system to fail by modifying any one of the component that they depend. Further the codes that these devices execute are more vulnerable to the attacks as a single change in its execution can change its behaviour. Therefore it is necessary that these devices are highly secured. Security can be increased at the design level to be more hard to external attacks, further continuous monitoring of the codes provide means of checking if any harmful codes are executed. This paper provides a review of the existing methods to detect if the code being executed is malicious or not thereby deciding whether to allow or stop the execution also few design strategy which could increase the security of the embedded devices.

I. INTRODUCTION

Embedded security is a major part to be careful, as they span over a vast area. Mistakes could lead to many hazardous like property damage, security breaches, personal injury and even death. Increasing the security of the system will cause an increase in the cost. To have a advantage over the market by cutting the security corners will be dangerous, as many of them are with real time deadlines even a small extra delay will make it vulnerable. Being energy operated systems draining the battery of the system can cause the system failure. Embedded systems are available in many forms which can be easily programmed to waste the resources and even cause destruction. Privacy can also be easily exploited by attackers, this serves as a major problem when it comes with military details. Thus security of embedded systems stays as a major threat in spite of the developments.

The need for the secure embedded devices stands as a major factor with its development. The codes executed could be easily altered; therefore it is necessary to provide some means of checking to check if the code being executed is completely normal. If there is any presence of malicious code running then the system should be stopped and prevented from further execution.

II. METHODOLOGIES

All embedded systems consist of inbuilt code running within it, for the proper functioning of the embedded system it is necessary that these codes are executed completely without any change in its flow of execution. The attackers can easily make the system to fail by changing its flow in its execution; thereby it is necessary to maintain some methods to check if there is any malicious code running behind i.e checking if there is any eviation from its normal flow.

A. IC Metric Detection

IC metrices are unique identifiers which are generated with the specific program. The program counter value and the number of cycles per instruction are selected as the IC metric features from which the unique characteristics of the system could be determined. The PC value can be captured at any time during the execution of the program; however a large number of data has to be processed for the feature extraction which degrades the system performance. In CPI more instructions could be analysed in less interval of time.

p, q are the program running on hardware *f* and *f (p)* is the set of characteristics extracted from *p*, Then f (p) is the IC metrics of *p*, only if the following two conditions are satisfied:

1) *f (p)* is obtained from *p* running on *f* .
2) Program *q* is a copy of *p* => *f (p)* = *f (q).*

The Average CPI is calculated based from (1):

$$CPI = C/I \quad (1)$$

Where *i* is the total number of executed instructions, *c* is the number of cycles for executing *i* instructions.

In [1] *CPI* value is taken for 10000 instructions per interval, choosing a interval of lesser instructions per cycle yields more features about the program. The information derived from the CPI at specific window interval is treated as the base and other malicious code running with not predicting the same level of CPI there by further execution of the code is prevented.

In [2] a self-organising maps (SOM)-based approach is used to increase the security level of embedded system by detecting abnormal program behaviour. Cycle per Instruction (*CPI*) is used to extract corresponding Program Counter (*PC*) values, and use it as IC Metrics features for correct program identification, allowable to execute on the embedded architecture, and an unsupervised Self-Organising Map (SOM) is used to classify the behaviour of

the embedded system by giving the PC and the CPI values as the input during the training session. The analyser based designed is capable of classifying and recognising between known and unknown programs while the programs are running. Here each phase and peak is measured to ascertain the originality of the program in execution. Any significant difference shows that the numbers of function calls, characteristic of function call and PC signature are different to the original program, and an abnormal behaviour notification is signified.

B. Dynamic Tracking Tags

In [3] the dynamic tracking system tags the untrusted information as trained ones and tracks its propagation in the security system. It focuses on tracking the flow of the external data into the processor which can help prevent the illegal operations caused by these external data or program, such as stealing user's private information stored in the system. It decides which data taint needs to propagate, and decides which data needs to be checked. Although the mechanism of information flow monitoring can detect some common attacks, it may result in the high false positive rate for other safety program in the system as a single program behave differently in different environment.

C. Memory Monitoring

Paper [4] achieves the purpose of detecting malicious attacks by protecting data space of the memory management, when the program runs and prevents malicious code from unauthorized modification of data space of a program. Memory monitoring needs to make a detailed analysis about the security of a program itself, including the type of instructions executed and the boundary information of the program's data space, which determine whether the instructions executed have threats on the program's data space, here a single monitoring strategy cannot prevent all the malicious attacks as the machine instruction compiled by different compiler there by producing different false negative results.

D. Integrating Information Flow and Memory Management

Instead of considering as separate dimension both the information flow tracking and the memory management are integrated together at the operation level with a new multi strategy in [5]. Every word is appended with 4 tag bits to track the information flow, Because of the added tag bits, several modifications are applied in the hardware architecture, such as 4 bits extension in all registers, caches, memories and data bus for the taint propagation.

A typical program data space includes stack section, heap section, global data section and text section. In stack segment protection, the range protected is reduced and only return address and stack pointers which have relatively fixed position in stack are protected. As the buffer overflow can cause loss of needed information thereby dynamic functions like malloc are used prior which defines the needed size thereby preventing the overflow. Global data section contains the storage of the pointer of the address of the control data which has to be protected. Similarly in each of the segment only the needed information and the one which are frequently exposed to attacks are protected. Finally, the design is mapped to an FPGA development board and prototype system is developed. Experimental results show that compared with a single strategy of information flow tracking and single strategy of memory monitoring, the multi-strategy combining the both can effectively detect more kinds of attacks at run time.

E. Hardware Trojan Threats Detectability Metric (HDM)

A novel metric for hardware Trojan detection coined as Hardware Trojan Threats (HTT) detectability metric (HDM) is introduced in [6], that uses a weighted combination of normalized physical parameters. HTTs are identified by comparing the HDM with an optimal detection threshold, if the monitored HDM exceeds the estimated optimal detection threshold, then it is tagged as malicious.

Using existing Trojan implementations, seven HTTs were designed and implemented on a FPGA testbed, 86% of the implemented Trojans were detected with HDM. As a single parameter lacks the ability to capture HTTs with distinct features, a combination of parameters capturing different characteristics of the Trojan is designed. Therefore the metric is constructed that uses a combination of different physical parameters such as power consumption, timing variation, utilization and leakage current to detect various sorts of HTTs. HTT detectability metric (HDM) uses a weighted combination of normalized physical parameters which are more important in the detection.

F. Detection of Malicious Software

The short sequences of the system calls which are made are consistent during normal execution. These sequences are not the same for abnormal system calls that is they may differ from the normal flow. A closer monitoring of these could help differentiating the abnormal software [14=6].

Arora et al. [7] formulated a hierarchical runtime monitoring framework to check program behavior at basic block level and verify instruction stream integrity using cryptographic hash tables for secure embedded processing. A similar work to this was done by [IMPRES] where instead of hash tables check sum were used, however any mismatch of them from the original value predicted the presence of abnormal software.

The information captured from various instruction are used to determine if the operation are valid or not []. However they don't act immediately once the error is

detected instead they find the error only at the end of the sequence.

A secure execution of program can be implemented by identifying the non-instruction memory pages by the NX (No eXecute) or XD(eXecute Disable) bit, this avoids the change of the data flow to a particular piece of code that's belong to data memory []. However the buffer flow overlow attack cannot be rectified. By using bounded checks and protecting the function pointers with XORing those with a secret key the buffer overflow attack could be prevented [].

Attacks to be detected in real-time, while they are occurring by means of a hardware supported HIDS (Host based intrusion detection system) is implemented in [7]. The System Call Detector block receives the information from the running programs which it uses to detect the execution of system calls. The System Call Detector has two outputs: SyscallP, which indicates the execution of a system call, and Syscall-Num, which is the number of the system call being executed. Each executing process is associated with a Syscall Sequence Recognizer, a finite state machine (*FSM*) that recognizes the sequences of system calls that can be generated by the correct program. If the system call sequence deviates from a known correct sequence, the state machine enters its detection state and asserts as the Attack Detected output. Since the system has multiple programs running, multiple FSMs has to be implemented, one FSM per program thereby a technique called Concurrent FSMs is implemented, using a demultiplexer to switch between the calls.

G. Choosing Secure Components

With the functional and the non-functional characteristics of the security system, the limitations of the embedded devices, their various interconnections the devices are designed with a increased security level in [8]. With the known intruder model, the analysis is done which defines the functional system requirements and the non-functional constraints needed for the security. For each of the security component define an alternative implementing the same function. On the basis of the relationship between them, the rules are defined for selecting the security components. The non-functional metrices of the components are calculated. The alternative components are ordered based on declining the values of the established non-functional components, thereby ensuring optimal selection of security component. This method when verified proved that best components could be selected following the procedure.

H. Boot System Modification

The embedded system architecture described in [9] currently focuses on confidentiality and integrity by protecting the boot process, code and data, and communications from unauthorized access and alternation. The boot time security is defined with A major commercial security technology is the Trusted Platform Module (TPM), which is the standard to enable trust in computing platforms. It also includes capabilities such as remote attestation, encryption, decryption, and sealed storage, hence software application can use a TPM chip to authenticate hardware devices. Further each TPM chip is provisioned with a unique and secret key making it further secure. The manufactured parts had to be inexpensive and cause as little disruption as possible to the current processing architecture, additionally many other reasons bought down the TPM.

I. Encryption with Boot System

This paper [10] combines the encryption process to a boot verification technique. Encryption is the foundation of information security but simply using standard cryptographic primitives do not guarantee the adequate implementation of security functions. The manner that the cryptographic primitives are assembled and coordinated into the desired application-specific security functions is critical to their effectiveness.

Based on LOCKMA, a security coprocessor (*S-COP*) that implements cryptographic primitives in hardware is developed. The added benefits of hardware implementation include hardware separation of sensitive keys from non- sensitive data and code, much faster computation time, and lower power use, the *S-COP* is responsible to secure the boot process, load only trusted software, and ensure that the unique identity is intact, before, during, and after boot process. The S-COP also uses key management to support secure communications between subsystems to protect information-in-transit. An optical PUF is developed When the system is provisioned for deployment, the PUF is applied which finalize the software code encrypted with the loaded PUF derived key. The system will not start if its PUF value is incorrect, causing a failed software decryption. In an unsecured architecture, the CPU directly reads in the BIOS. Without authentication, the CPU is vulnerable to maliciously modified BIOS. The S-COP secured architecture addresses this vulnerability by authenticating the BIOS and applications. When the system powers up, the S- COP halts the CPU while it performs authentication. It first reads the PUF and derives a key. The key is used to decrypt the BIOS. If successful, the CPU is released to execute the BIOS.

J. Hardware Assisted Method

A hardware based mechanism to process sensitive information in complete isolation without requiring any software process is proposed in [11], by placing critical processing units in a secure region, to process the sensitive information securely a secure region is designed. The modules placed inside this secure region can only be accessed by the outside modules through a secure interface, the access control signals are verified before passing data to

the modules present in the secure region. The ”*secure mode*” module will extract the secure access rights whereas the ”*PLB mode*” module will extract the application access rights from the input signal. The ”*access rule check*” will perform access rights verification. In case of any mismatch, as it would be in the event of security breach, the alert signal is generated by the secure interface, stopping further flow of data to the secure region modules.

The proposed solution was evaluated by integrating it within an image processing based authentication system and implementing the final design on a Spartan-3A DSP FPGA Video Starter Kit(*VSK*). A delay of one clock cycle only to perform the access verification was introduced.

On analysing the output of the system using different synthetic, when a attack is launched using buffer overflow technique, the access control signal is overwritten but the secure controller output is not changed. When control flow reached at the ”access rule check” module, the verification step failed and attack alert signal is generated by the secure interface module.

K. Hardware Assisted Run Time Monitoring

External monitor are used to detect wheather the program runs in the correct flow, else prevent the execution of the program. During run time by the static program analysis the static characteristics are obtained. The program counter, the instruction register, pipeline status are generally treated as the parameters. An invalid signal is transferred when there is a violation from the normal execution by the monitor, which in turn enable the user to take the necessary actions. With FSM they are implemented in [11]

L. Security System Base on TCM and FPGA and AEGIS

System security can be integrated using Aegis, which integrates security kernel, encryption and integrity to a single processor. The integrity is verifies first, the PUF creates a secret password for the encryption module, After two checks the data is accepted. In spite of this rule being secure it is considered as waste of time and resources by [12]. By checking the intergrity of instructions and data in flash chip the security in the embedded system is provided based on TCM and FPGA before the execution. It has an FPGA controller separating the processor and BOOTROM which is the major region to provide the trust. When powered on the controller fully reads the bootRom which is send to the TPM module after performing encryption, which then decides if to give service or not for the application [13].

M. Trustzone Based

In Morden ARM devices trustzone is increasingly used as a security component. It provide completely two different environment for execution namely the secure mode and the non secure mode. A addition bit indicates which mode is being executed. They become increasingly useful when operated as a Dual-OS approach where each one operates in different world, Further various other features like memory segementations make them very useful towards virtualisation [16].

III. CONCLUSION

Embedded system security does not only deal with these detection of malicious software and design security but it also includes various other features like its compatibility with the other devices, its reliability in its changing environment etc. However in design perceptive increasing the security by incorporating the encryption with the booting system is a simple and effective means. The mentioned methods of detecting the abnormal code can be easily implemented and they are effective means of stopping the system from executing the malicious code thereby saving the device.

REFERENCES

[1]. X.Zhai, K. Appiah, S.Ehsan, G. Howells “Application of IC metrices for embedded system security” f0urth international conference on emerging security technologies, 2013.
[2]. X. Zhai, K. Appiah, G. Howells, “ A method of detecting abnormal program behaviour on embedded devices” IEEE Transaction on information forensics and security, 2015.
[3]. Z.Liu, X.S.Zhang and X.D.Li “Proactive vulnerability finding via information flow tracking”, International conference on multimedia information networking and security, 2010.
[4]. M.Dalton, H. Kannan and C. Kozyrakis “Real world buffer overflow protection for user and kernel space “ International conference on dependable systems and networks, 2008.
[5]. L.Dongfang, Z.Xin and T. Qiaoling “The design and implementation of embedded security CPU based on Multi-stratergy” Chinese journal of electronics, 2016.
[6]. D.M.Shila and V.Venugopal “Design, Implementation and security analysis of Hardware Torjan Threats in FPGA” Communication and information systems security symposium, 2014.
[7]. S.Lukacs, A.V.Lutas “Hardware Virtualization Based Security for Embedded Systems” IEEE 2014.
[8]. V.Desnitsky, A.Chechulin, I. Kotengo and D. Levshun “ Application of a technique for secure embedded device design based combining security components for the creation of a perimeter protection system” 24th Euromicre international conference on parallel, distributed and network based processing” 2016.
[9]. M.Vai, B.Nahill, J.Kramer “Secure architecture for embedded systems “ IEEE 2015.
[10]. A.Saeed, A.Ahmadinia and M.Just “Hardware assisted secure communication fpr FPGA based embedded system” IEEE 2015.