The ISC Int'l Journal of Information Security

ISeCure

January 2017, Volume 9, Number 1 (pp. 17–26)

http://www.isecure-journal.org

Side Channel Parameter Characteristics of Code Injection Attacks

Ehsan Aerabi [1,*], Mahdi Kaykha [1], Mahdi Fazeli [1], Ahmad Patooghy [1], and Ahmad Akbari [1]

[1] *Iran University of Science and Technology, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 6 April 2016
Revised: 6 January 2017
Accepted: 24 January 2017
Published Online: 29 January 2017

Keywords:
Embedded Systems, Code Injection, Side Channel.

ABSTRACT

Embedded systems are suggestive targets for code injection attacks in the recent years. Software protection mechanisms, and in general computers, are not usually applicable in embedded systems since they have limited resources like memory and process power. In this paper we investigate side channel characteristics of embedded systems and their applicability in code injection attack detection. The architectural simulation for execution time, power usage and temperature on benchmarks shows that these parameters disclose meaningful and distinguishable behaviours in case of attack.

1 Introduction

Over a decade, embedded systems have outcompeted general purpose computers in their number and applications in our daily life and industries. Therefore, their security are major concern [1]. Variety of hardware and software threats are reported like Energy Exhaustion Attacks, Side Channel Attacks, Malwares, Thermal Viruses and Code Injection Attacks [1]. Code Injection is a big category of software vulnerabilities which allows an attacker to remotely exploit a programming flaw in an application in order to change the program flow and make it to run a malicious code. SQL Injection, Remote File Injection and Shell Injection are three types of Code Injection vulnerabilities. In workstation and server computers, these attacks are countered by Firewalls and Intrusion Prevention Systems. Embedded devices are designed for a specific application and usually come with limited software/hardware resources. On hardware side, these limitations include CPU performance, memory and power usage. On software side, small and fewer running programs, smaller or no operating systems, and lower level programming languages are some of the limitations [2, 3]. Therefore, software monitors and protection mechanisms like firewalls and IPSs impose significant memory and processing overhead and are not appropriate in embedded systems unlike in desktop and server computers. On the other hand, embedded device manufacturers usually do not support third party antiviruses or firewalls. These limitation have made cyber threats a big concern especially for safety-critical devices like those in medical settings. In this paper we conduct a feasibility study on side channel information for code injection attack detection, which have no software overhead to examine. In other words we find distinguishable side channel behaviours like changes in power consumption, execution time and temperature of a device in case of normal operation and when it is attacked by a remote intruder.

The rest of this paper counties as follows: inSection 2 we overview the related literature. Then in Section 3 we discuss about the parameters chosen to be monitored for the attack detection. In Section 4 the implementation of the detection scenario are covered. Section 5 presents the attack detection results and Section 6 shows how to use this technique in a machine

* Corresponding author.
Email addresses: e_aerabi@comp.iust.ac.ir (E. Aerabi), kaykha@comp.iust.ac.ir (M. Kaykha), m_fazeli@iust.ac.ir (M. Fazeli), patooghy@iust.ac.ir (A. Patooghy), akbari@iust.ac.ir (A. Akbari)

ISSN: 2008-2045

learning case study. Finally, the paper is concluded in Section 7.

2 Related Work

2.1 Static Protection

Legacy prevention methods are still usable at compile time like static source code analysis to find programming bugs and then harden it [4–7] and also using managed programming technologies like Java and more secure dialects of C++ [8, 10].

There are some protection mechanisms at architecture-level like in [9]. They mainly focus on allocating and maintaining safe memory areas to protect processes from unauthorized access. This allocation could be done either physically or by means of a bus monitor [11]. [12] presents a model for implementing an execution-only-memory (XOM) and some heuristics like customized instructions and additive information to the cache lines with virtual machine software monitor. These methods are implemented in some commercial processors like Trust-Zone in ARM [13] and Lagrande in Intel [14]. This category of prevention methods usually are not applicable to the legacy application or require source code access, which make them impractical in many cases.

2.2 Dynamic Protection

Dynamic protection methods detect attacks at runtime and could be categorized as software and hardware solutions. Similar to static methods, it is possible to analyse program source codes on compile time but extract vulnerabilities at runtime [15]. This method is precise in case of false positives but it does not guarantee to trace all execution paths at runtime.

In [16] authors extract formal behaviours for some critical programs and detect unauthorized access by analysing their execution traces. In this method, the sequence of events in concurrent execution of programs is used to determine normal behaviours of them.

[17] provides an environment for monitoring program flow control at runtime and forcing the program to follow some security policies like program start points and restricted changes in control transitions which previously have been determined by the system operator. Previously mentioned methods also need either the source code or any access/modification to the internal behaviour of the system under protection.

Anomaly-based detection methods are also available which need a learning phase to understand a program normal behaviour [18–20]. These methods are always prone to false positive and false negative cases because of improper learning or changes in applications.

Using a security co-processor goes back to Tamper-Resistant Crypto Processors for securely storing keys and performing encryption algorithms [21, 22]. In [23], the use of a co-processor for intrusion detection systems (IDS) was introduced. This would lead to an IDS operation which no attack on the target system can hinder. Unlike the other host-based IDS's(HIDS), a co-processor-based IDS(CIDS) monitors the system without complete access to the internal information of the system. Thus, CIDS works generally based on monitoring external and side channel characteristics of a system like time, user ID, memory and file checksums.

In [24] an FPGA implementation of a hardware-assisted anomaly-based HIDS is expressed which detects attacks by comparing valid and malicious behaviour of the underlying system. It can be updated regularly to cope with new malwares attacks and can operate in real-time. Normal behaviour of the system is defined by the sequence of its valid system-calls. Detecting system-calls is performed by monitoring fetched instructions and register's values. Thus, after a sequence of three anomalous system-calls the malicious process is killed. It is claimed that this method is faster than software solutions and has less false positive and false negative alerts. In [25], another hardware-assisted approach is introduced which tracks whether untrusted data can change program control flow when they enter the system or not. [26] describes a software-hardware method to verify code integrity and prevent it from being damaged by the dynamic changes at runtime by storing encrypted checksum along with the executable code blocks at compile time and compare it with updated checksums at runtime.

In [2], a monitoring scheme is introduced for multiprocessor settings in which one of the CPUs is a security processor and monitors the other ones. At compile time, a profile is created by a static analysis of the program's scheduling and is used at runtime for detection. [27] presents another hardware monitor which runs in parallel with the processor. The processor sends information like changing in program flow control or load/store sequence to the monitor and then monitor checks to ensure that information is complying with expected patterns and program is running on its normal track.

In [28], anomalous events are detected by correlating thermal and processing information. A different approach to the problem is introduced in [29] which checks integrity of a program independently at runtime by means of dynamic power consumption. This power consumption fingerprinting method, measures device's current and validates it by applying pattern recognition algorithms. This approach is continued in [30, 31] and then in [36] proposing more sensitive

industrial controllers like PLC and SCADA. A similar anomaly-based method is introduced in [32] and [33]. These methods exploits just power usage to detect attacks.

Time variation can also be used for this purpose. Lu *et al.* present a method that utilizes timing behaviour of the software for attack detection [34]. Another method exploits CPU program counter and cycle per instruction to achieve anomaly detection [35].

In contrast with the previous methods mentioned above, in this paper we aim to present a method which does not require any access to the internal information of the system (e.g., Data Bus, CPU instruction,) any source code review or modification and any extra hardware or architectural change in the system. Therefore, it can be used on legacy software and hardware. The proposed method utilizes different side channel characteristics of the CPU and application including CPU cycles, energy and temperature. Associating three different system parameters would enhance the detection accuracy. This makes our work unique in comparison to the previously mentioned methods.

2.3 Embedded and Industrial Limitations

All of the aforementioned methods require different level of access and different level of modifications to the systems. These levels include operating system, compiler, application source code, application binary, CPU internal attributes, architecture and some external attributes like temperature and power consumption.

Several access and modification constraints might be imposed when it comes to embedded and industrial systems and would limit the number of applicable protection methods. In many, it is impossible or costly to change hardware or software architecture. In some others there is no possibility to access the internal CPU attributes. In third party applications, source codes are not accessible. In these cases, monitoring could be carried out externally. For example, in medical equipment, any modification in hardware or software would void the system guarantee [37].

In this paper we investigate a few behaviours of an external side channel parameters of an embedded systems in presence of code injection attack. These side channel parameters are time, power consumption and temperature. These attributes are important because any protection solution based on them requires no access to source or binary code, no modification and no high level access to the system. We will show how effective they can be under code injection attack detection by implementing some real-world attacks on our benchmarks.

3 Monitoring Parameters

In this section we study how temperature, execution time and power usage behaviour can represent internal device status in cases of code injection attacks.

3.1 Power Consumption

A digital circuit power consumption could be divided into *static* and *dynamic*. Static power consumption is caused by sub-threshold current and leakage. For a long time it was assumed that static power has no connection with the data being processed and hence is not useful for intrusion detection. Despite previous assumptions, researchers found that static power is correlated to the data being processed [38] and have useful information for our case. Nevertheless, in order to exploit static power changes we need ultra precise equipment in the presence of noise and error. Dynamic power usage produced by circuit switching can effectively help us detect such attacks [39]. Running a specific software routine would produce specific dynamic power pattern or signature. Using power signature is an effective method to find or protect some well known hardware attacks like energy exhaustion, hardware Trojans and fault attacks [40, 41]. These methods require pattern recognition algorithms at runtime which impose considerable processing load on monitor. In this work we use total power consumption of a process since it needs less processing overhead at runtime.

3.2 CPU Temperature

Temperature can be another parameter for monitoring. By increasing systems complexity and power density in chips, temperature effects of attacks are more important than before [28]. Different processing workload can produce different thermal status. Therefore, temperature is a candidate for anomaly detection. Thermal changes is also correlated with power usage. Temperature can affect CPU power usage by changing sub-threshold and gate leakage.

3.3 Execution Time

Execution time is another parameter for detecting code injection attacks. Every input data might change a software procedure execution time, which can be used to detect anomalies. Determining execution time is possible as we have complete control of the CPU reset pin.

4 Implementation

To evaluate proposed monitoring parameters, normal operation of the system is determined when it is supplied by a comprehensive valid inputs data sets. Then we compare it with the operation under the attack

data sets. We chose "Mibench which is an opensource test bench based on EDN Embedded Microprocessor Benchmark Consortium [43]. This test bench is comprised of six areas: automatize and industrial control, consumer device, office automation, networking, security and telecommunication. We study Qsort program from automation and industrial control category and SHA and Rijndael from security category. These three are among the most general and frequently used programs available in the test bench. SHA and Rijndael(AES) are two cryptographic algorithms used in everyday life for data encryption. Qsort is among the best sorting algorithms and it is used in many programs. All of these are tested on MARS simulator of MIPS processor [44].

We chose the following attack shell codes and inserted them into the test benches

- *MIPS execve shellcode*: this is a 60-byte shell code used to gain remote access from a target system [45].
- *MIPS stdin-read shellcode*: this is a 40-byte shell code used to gain remote command execution from a target system [46].
- *MIPS -reboot() shellcode*: this is a 32-byte shell code used to remotely reset a target system [47].

Table 1 contains all three shellcodes assembly.

Table 1. MIPS Code Injection shellcodes.

MIPS reboot() shellcode	MIPS stdin-read shellcode	MIPS execve shellcode
`lui $6,0x4321`	`li $4,-0x7350`	`li $6,0x7350`
`ori $6,$6,0xfedc`	`$dpatch:`	`LB: bltz $6,LB`
`lui $5,0x2812`	`bltz $4,$dpatch`	`li $15,0x7350`
`ori $5,$5,0x1969`	`slti $4,$0,-1`	`slti $6,$0,-1`
`lui $4,0xfee1`	`li $15,-29`	`addiu $29,$29,-32`
`ori $4,$4,0xdead`	`nor $15,$15,$0`	`li $15,-41`
`li $2,0x4088`	`addu $5,$31,$15`	`nor $15,$15,$0`
`syscall`	`li $6,0x0201`	`addu $4,$31,$15`
	`li $2,0x0205`	`sw $4,-24($29)`
	`syscall`	`sw $0,-20($29)`
	`li $24,0x7350`	`addi $5,$29,-24`
		`li $2,4011`
		`syscall`

We integrated simulation tools including Wattch, HotSpot and PTscalar [48] which are used to emulate power usage and temperature of systems. They are designed to work with the popular architecture-level simulator SimpleScalar. PTscalar is a modified cycle precision version of SimplaScalar, which accepts system configuration and parameters and simulates power and temperature for each clock cycle. Table 2 describes initial configuration for the simulation scenario.

Table 2. PTscalar initial configuration.

Title	Input Simulation Parameters	Quantity
Number of CPU Cores	-	1
CPU Modules	TOTAL_MODULES	18
Initial Temperature	TOTAL_ MODULES	35(C)
CPU Frequency	STD_ FREQ	3 (GHz)
CPU Supply Voltage	VDD	1 (v)
Memory Supply Voltage	SVdd	1.3 (v)
Data Bus Width	DATA_ WIDTH	64 bits
Address Bus Width	ADDR_ WIDTH	32 bits
L1 Cache Size	cache:dl1	128KB/32-bit data width/4-way
L1 Cache Size	cache:dl2	1MB/64-bit data width/4-way

5 Results

In this section, experimental results of the *MIPS execve shellcode* attack on Qsort, SHA and Rijndael are presented. Other attacks are excluded from this paper to comply with page number constraint.

We choose four combinational signatures from the monitoring parameters of temperature, execution time and power usage as follows:

- *CE*: Execution Cycles and Total Energy Consumption.
- *CT*: Execution Cycles and Temperature.
- *ET*: Total Energy Consumption and Temperature.
- *CET*: Execution Cycles, Total Energy Consumption and Temperature.

Measurement unit is centigrade for temperature and joule for total energy consumption. For Qsort and Rijndael test-benches, an input set comprised of 200 elements and for SHA, an input set including 500 elements are generated and applied. Figure 1 illustrates the results in which blue dots and red circles represent measured quantities for the safe and attacked systems, respectively. Each dot or circle illustrates the measured parameters for one of the elements in the input set. Obtained result includes all CE, CT, ET and CET signatures for all three Qsort, SHA and Rijndael benchmarks.

In these figures, we can observe that monitoring signatures are affected by the attack inputs that means they might be useful for our purpose. To ensure that

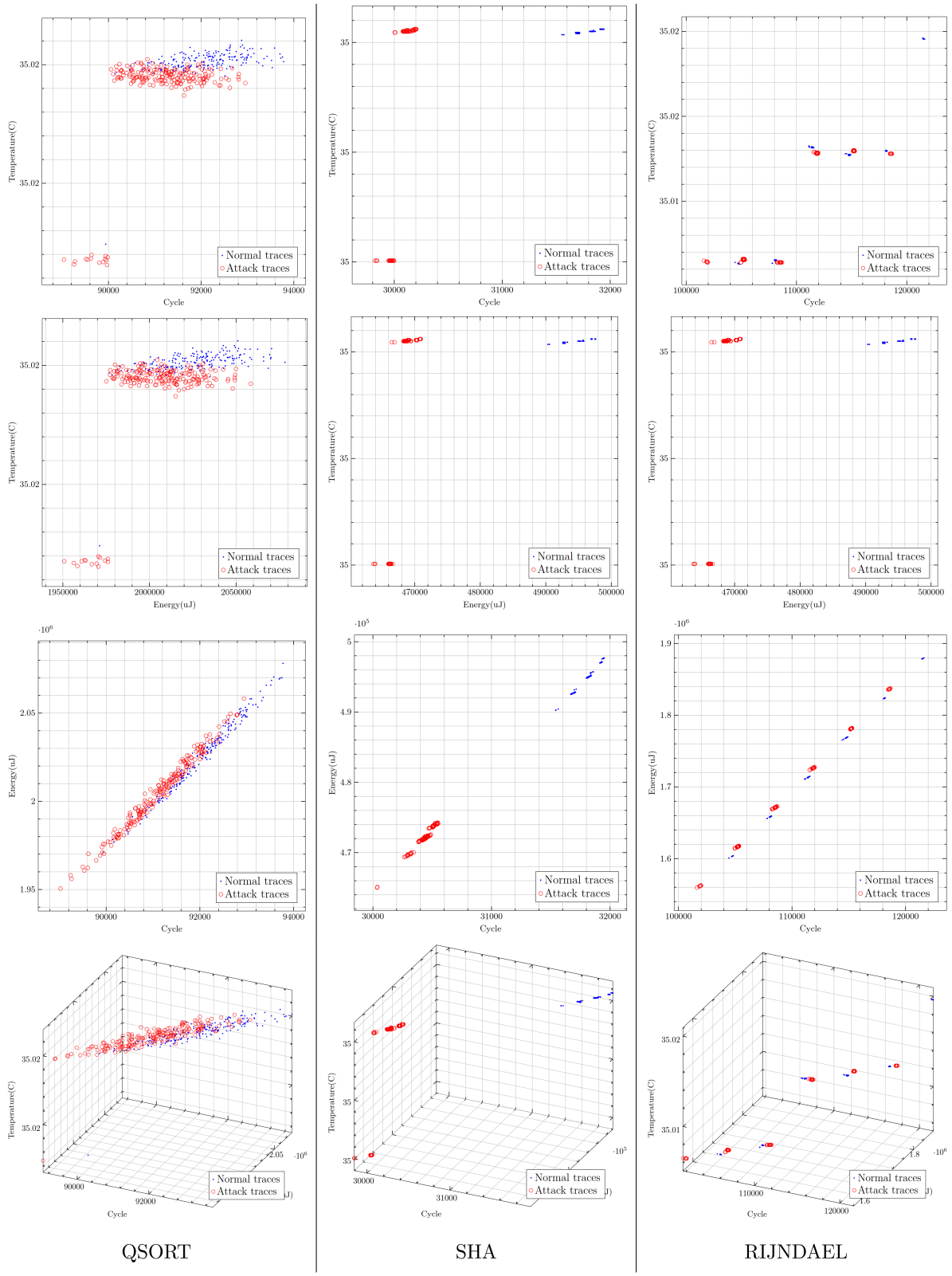

Figure 1. Effect of MIPS execve shellcode attack to Qsort,SHA and Rijndael program on energy, temperature and time parameters.

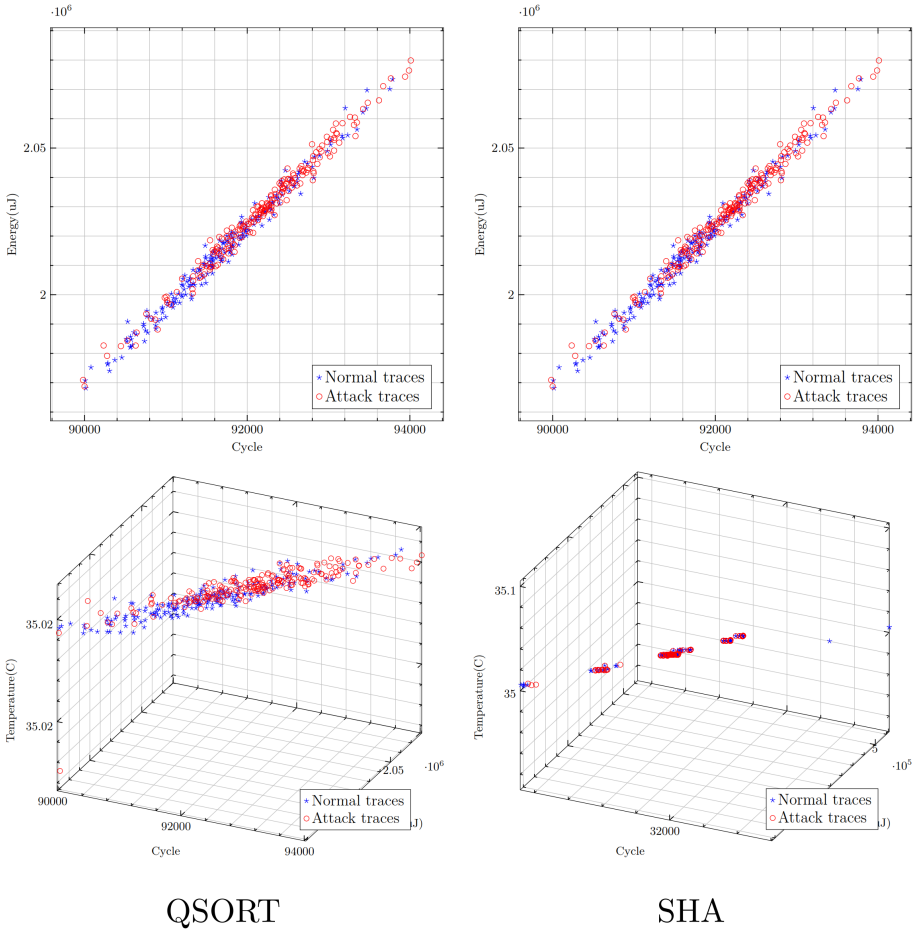

Figure 2. Effect of input variations on side channel parameters.

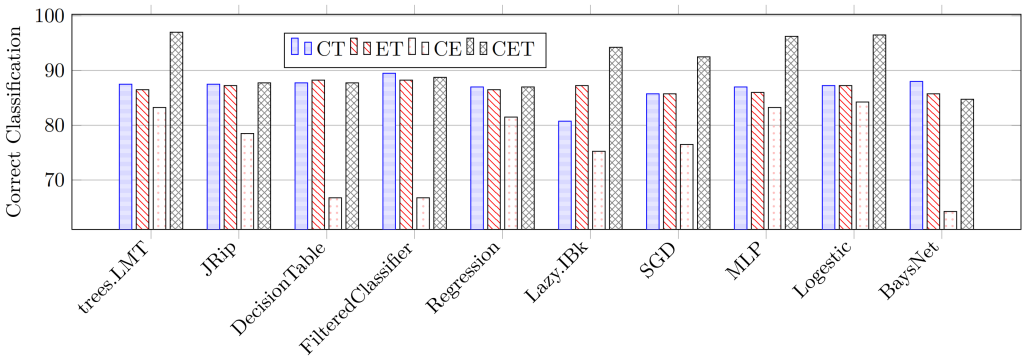

Figure 3. Success rate of machine learning algorithms in finding "MIPS execve shellcode" attack on Qsort testbench- (C:cycle - T:temperature - E:energy).

attack signatures are practically observable and detectable, they should not be overshadowed by the signatures resulted from input data variations. Therefore, we investigated how much the monitoring parameters are affected also by variations of valid inputs. We now show how different input data sets can affect the monitoring signatures. In this case we just show Qsort and SHA results for CE and CET parameters and will omit other programs from the paper. Figure 2 illustrates the result for two set of inputs each of which with 200 entries processed by Qsort and SHA test benches. It is obvious that total energy variations in each data set are not significant enough to overshadow the attack signatures.

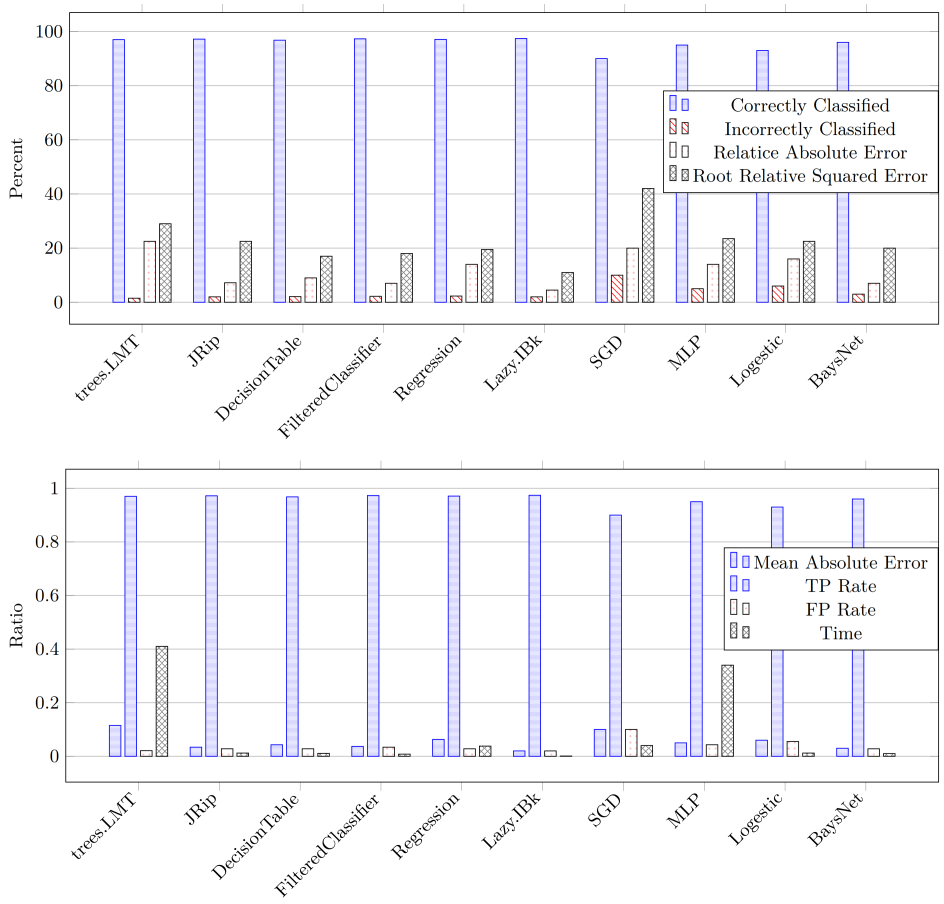

Figure 4. Comparison of machine learning algorithms.

6 Case Study

In this section, we further study effectiveness of side channel parameters of code injection attacks which were introduced in the previous section, by a machine learning case study. In this experiment we used WEKA [49] which is an appropriate machine learning platform for our work. We also fed the software with normal and attacked inputs and used *cross-fold variation* method with value of *10-fold* for learning phase. Top 10 most successful learning algorithms were chosen and their results are depicted in Figure 3. In this figure, all parameter signatures along with the detection rate of each method for Qsort test bench and MIPS execve shellcode have been illustrated. It asserts effectiveness of the side channel parameters in detection scenarios. A detailed comparison among these methods are presented in Figure 4. Rate of correct and incorrect classification, relative absolute error, rooted relative squared error, mean absolute error, true positive rate, false positive rate and time are our comparison parameters. Among these, success classification rate and mean absolute error are the most important ones. In this study, the best classification results belong to trees.LMT, Lazy.IBK and Regression. On the other hand the lowest mean absolute errors result belong to Lazy.IBK, JRip and FilteredClassifier. Finally, we compared side channel parameters and their practicality for detecting code injection attack in Figure 5. Cycle-Temp-Energy has the best distinguishable characteristics followed by Cycle-Temp, Energy-Temp and Cycle-Temp.

7 Conclusion

In this paper we examined some combinations of side channel parameters including temperature, energy and CPU cycle in embedded systems and their characteristics in case of code injection attack incident. These experiments show that the combination of these parameters can successfully distinguish normal and attacked behaviour of the embedded software. We then applied machine learning algorithms to identify attack from safe inputs. These results show that further studies on code injection vulnerability detection based on side channel characteristics of embedded systems are feasible.

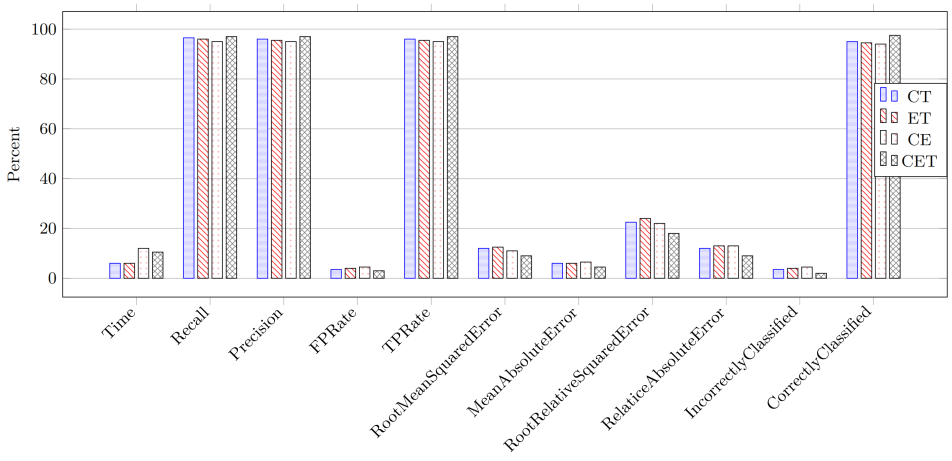

Figure 5. Comparison of side channel parameters in detecting Code Injection attack

References

[1] Sri Parameswaran and Tilman Wolf, "*Embedded systems security - an overview*", Design Automation for Embedded Systems 12, no. 3, pp. 173-183, 2008.

[2] Krutartha Patel, Sridevan Parameswaran, and Seng Lin Shee, "*Ensuring secure program execution in multiprocessor embedded systems: a case study*", In Hardware/Software Codesign and System Synthesis (CODES+ ISSS) 2007 5th IEEE/ACM/IFIP International Conference on, pp. 57-62, IEEE, 2007.

[3] Tammy Noergaard, "*Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers*". Access Online via Elsevier, 2005.

[4] M. Howard and D. LeBlanc, "*Writing Secure Code*". Microsoft Press, 2002.

[5] G. Hoglund and G. McGraw, "*Exploiting Software: How to Break Code*" Addison-Wesley, 2004.

[6] Ken, and Dawson Engler Ashcraft, "*Using programmer-written compiler extensions to catch security holes*", In Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on, pp. 143-159, IEEE, 2002.

[7] William R., Jonathan D. Pincus, and David J. Sielaff Bush, "*A static analyzer for finding dynamic programming errors*", Software-Practice and Experience 30, no. 7, pp. 775-802, 2000.

[8] Jeremy, Matthew Harren, Scott McPeak, George C. Necula, and Westley Weimer Condit, "*CCured in the real world*", ACM SIGPLAN Notices 38, no. 5, pp. 232-244, 2003.

[9] Srivaths, Anand Raghunathan, and Srimat Chakradhar Ravi, "*Tamper resistance mechanisms for secure embedded systems*", In VLSI Design, 2004.

[10] Dinakar, Sumant Kowshik, Vikram Adve, and Chris Lattner Dhurjati, "*Memory safety without runtime checks or garbage collection*", In ACM SIGPLAN Notices, vol. 38, no. 7, pp. 69-80, ACM, 2003.

[11] CryptocellTM. Discretix Technologies Ltd. [Online]. http://www.discretix.com

[12] D. Lie et al., "*Architectural support for copy and tamper resistant software*", in Proc. ACMArchitectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 168177, 2000.

[13] R. York. A New Foundation for CPU Systems Security. ARM Limited. [Online]. http://www.arm.com/products/processors/technologies/trustzone/index.php

[14] LaGrande Technology for Safer Computing. Intel Inc. [Online]. http://www.intel.com/technology/security

[15] D. Clarke, B. Gassend, M. van Dijk, and S. Devadas G. E. Suh, "*AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing*", in Proc. Intl Conf. Supercomputing (ICS 03), pp. 160171, June 2003.

[16] Calvin Ko, Manfred Ruschitzka, and Karl Levitt, "*Execution monitoring of security-critical programs in distributed systems: A specification-based approach*", In Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on, pp. 175-187, IEEE, 1997.

[17] Vladimir Kiriansky, Derek Bruening, and Saman Amarasinghe, "*Secure execution via program shepherding*", Proceedings of the 11th USENIX security symposium, vol. 6, no. 2, pp. 191206, 2002.

[18] Steven A., Stephanie Forrest, and Anil Somayaji Hofmeyr, "*Intrusion detection using sequences of system calls*", Journal of computer security 6, no. 3, pp. 151-180, 1998.

[19] Stephanie, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff Forrest, "*A sense of self for unix processes*", In Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on, pp. 120-128, IEEE, 1996.

[20] Sachin P., and Stephen R. Tate Joglekar, "*Protomon: Embedded monitors for cryptographic protocol intrusion detection and prevention*", In Information Technology: Coding and Computing. Proceedings. ITCC 2004. International Conference on, vol. 1, pp. 81-88, IEEE, 2004.

[21] Divya Arora, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha, "*Secure embedded processing through hardware-assisted run-time monitoring*", Proceedings of the conference on Design, Automation and Test in Europe-Volume 1. IEEE Computer Society, pp. 178-183, 2005.

[22] M. Kuhn, "*The TrustNo 1 Cryptoprocessor Concept. CS555 Report*", Purdue University (http://www.cl.cam.ac.uk/?mgk25/), 1997

[23] X. Zhang, L. Doorn, T. Jaeger, R. Perez, and R. Sailer, "*Secure coprocessor-based intrusion detection*", in Proc. ACM SIGOPS European Wkshp, 2002.

[24] Mehryar Rahmatian, I. Harris, H. Kooti, and Elaheh Bozorgzadeh, "*Hardware-Assisted Detection of Malicious Software in Embedded Systems*", Embedded Systems Letters, IEEE, 2012.

[25] G. E. Suh, J. Lee, and S. Devadas, "*Secure program execution via dynamic information flow tracking*", Dept. of EECS, MIT, Tech. Rep., 2003.

[26] Roshan G. Ragel and Sri Parameswaran, "*IMPRES: integrated monitoring for processor reliability and security.*", In Proceedings of the 43rd annual Design Automation Conference, pp. 502-505, ACM, 2006.

[27] Shufu Mao and Tilman Wolf, "*Hardware support for secure processing in embedded systems*", Computers, IEEE Transactions on 59, no. 6, pp. 847-854, 2010.

[28] Tilman, Shufu Mao, Dhruv Kumar, Basab Datta, Wayne Burleson, and Guy Gogniat Wolf, "*Collaborative Monitors for embedded System security*", Proc. Wkshp. of Embedded System Security, 2006.

[29] Carlos R. Aguayo, and Jeffrey H. Reed Gonzalez, "*Power fingerprinting in SDR and CR integrity assessment*", In Military Communications Conference, 2009. MILCOM 2009, pp. 1-7, IEEE, 2009.

[30] Carlos R. Aguayo, and Jeffrey H. Reed Gonzalez, "*Power fingerprinting in SDR integrity assessment for security and regulatory compliance*", Analog Integrated Circuits and Signal Processing 69, no. 2-3, pp. 307-327, 2011.

[31] Carlos R. Aguayo, and Jeffrey H. Reed Gonzalez, "*Detecting unauthorized software execution in SDR using power fingerprinting*", In MILITARY COMMUNICATIONS CONFERENCE, 2010-MILCOM 2010, pp. 2211-2216, IEEE, 2010.

[32] Tang, Adrian, Simha Sethumadhavan, and Salvatore J. Stolfo. "Unsupervised anomaly-based malware detection using hardware features." International Workshop on Recent Advances in Intrusion Detection, 2014.

[33] Liu, Hong, Hongmin Li, and Eugene Y. Vasserman. "Practicality of Using Side-Channel Analysis for Software Integrity Checking of Embedded Systems." International Conference on Security and Privacy in Communication Systems, 2015.

[34] Lu, Sixing, Minjun Seo, and Roman Lysecky. "Timing-based anomaly detection in embedded systems." The 20th Asia and South Pacific Design Automation Conference, 2015.

[35] Zhai, Xiaojun, et al. "A method for detecting abnormal program behavior on embedded devices." IEEE Transactions on Information Forensics and Security 10.8 (2015): 1692-1704.

[36] Jeffrey H., and Carlos R. Aguayo Gonzalez Reed, "*Enhancing Smart Grid cyber security using power fingerprinting: Integrity assessment and intrusion detection*", Future of Instrumentation International Workshop (FIIW), IEEE, 2012.

[37] Shane S. Clark et al., "*WattsUpDoc: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices*", In Presented as part of the 2013 USENIX Workshop on Health Information Technologies, 2013.

[38] A. Moradi, "Side-Channel Leakage through Static Power" , Cryptographic Hardware and Embedded Systems CHES 2014 Volume 8731 of the series Lecture Notes in Computer Science pp 562-579

[39] John L. Hennessy and David A. Patterson, "*Computer architecture: a quantitative approach*" , 4th ed.: Elsevier, 2007.

[40] Dakshi, Selcuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar Agrawal, "*Trojan detection using IC fingerprinting*", In Security and Privacy, 2007. SP'07. IEEE Symposium on, pp. 296-310, IEEE, 2007.

[41] Li-Wei, and Hong-Wei Luo Wang, "*A power analysis based approach to detect Trojan circuits*", In Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011 International Conference on, pp. 380-384, IEEE, 2011.

[42] Dongwoo Lee, Wesley Kwong, David Blaauw, and Dennis Sylvester, "*Analysis and minimization techniques for total leakage considering gate oxide leakage*", In Proceedings of the 40th annual Design Automation Conference, pp. 175-180, ACM, 2003.

[43] Matthew R. Guthaus et al., "*MiBench: A free,*

commercially representative embedded benchmark suite", In Workload Characterization. WWC-4. 2001 IEEE International Workshop on, pp. 3-14, IEEE, 2001.

[44] MARS (MIPS Assembler and Runtime Simulator) An IDE for MIPS Assembly Language Programming. [Online]. http://courses.missouristate.edu/kenvollmar/mars/

[45] linux execve 60 bytes shellcode. [Online]. http://www.shell-storm.org/shellcode/files/shellcode-80.php

[46] 40 byte MIPS/Irix PIC stdin-read shellcode. [Online]. http://www.win.tue.nl/~aeb/linux/hh/phrack/P56-16

[47] rigan. Linux/MIPS - reboot() - 32 bytes. [Online]. http://packetstormsecurity.com/files/107735/Linux-MIPS-reboot-Shellcode.html

[48] Weiping Liao. PTscalar - University of California at Los Angeles. [Online]. http://eda.ee.ucla.edu/PTscalar/

[49] Weka 3: Data Mining Software in Java. [Online]. http://www.cs.waikato.ac.nz/ml/weka/

Ehsan Aerabi is a research assistant at dependable systems & architectures (DSA) laboratory in Iran University of Science and Technology. He received his M.S. in computer architecture from Isfahan University of Technology and his B.S. in computer engineering from Amirkabir University of Technology. His research interests spans over computer architecture, computer networks and cyberphysiscal systems.

Mahdi Kaykha is a graduate student from computer engineering department of Iran University of Science and Technology. He received his BS from University of Sistan and Baluchestan in computer engineering.

Mahdi Fazeli received M.S. and Ph.D. degrees in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2005 and 2011, respectively. He has been with the department of computer engineering, Iran University of Science and Technology (IUST), Tehran, since 2011, where he is currently an assistant professor. He has established and chaired two research laboratories at IUST since 2012, namely, the dependable systems and architectures laboratory and the networked and embedded system laboratory. He is the author or co-author of over 50 papers in reputable journals and conference proceedings. His current research interests include reliable issues in VLSI circuits and emerging technologies, dependable embedded systems, low power circuits and systems, fault-tolerant computer architectures, fault injection, and reliability modeling and evaluation.

Ahmad Patooghy received his M.S. and Ph.D. in computer engineering from Sharif University of Technology, Tehran, Iran, in 2005 and 2011, respectively. He is currently an assistance processor at department of computer engineering, Iran University of Science and Technology, Tehran, Iran. His research interests include hardware design and test, architectural design of multi- and many-core chips, dependability and security evaluation of VLSI circuits, fault injection, and analytical modeling. Dr. Patooghy initiated the "dependable systems and architectures laboratory" at Iran University of Science and Technology in 2012 and has chaired the Laboratory since then.

Ahmad Akbari received his B.S. degree in electrical engineering from Isfahan University of Technology, and M.S. degree in communications engineering from Isfahan University of Technology, Iran, in 1987 and 1989, respectively. He received his Ph.D. in electrical engineering from University of Rennes 1, Rennes, France, in 1995. He joined the computer department of Iran University of Science and Technology in 1995 where he is the head of RCIT (research center for information technology) and head of the department. Dr. Akbari has established ASPL (Audio and Speech Processing Lab) and NRG (Network Research Group) including graduate students and researchers from different areas of computer and electrical engineering. The research programs in the NRG lab focus on the five main streams of cloud computing, big data analytics, software-defined network (SDN) and network functions virtualization (NFV), multimedia streaming and signal processing, and network security and intrusion detection systems (IDS).