Available online at www.sciencedirect.com

ScienceDirect

AASRI Procedia 4 (2013) 132 – 137

AASRI Procedia

www.elsevier.com/locate/procedia

2013 AASRI Conference on Intelligent Systems and Control

Neural Network VS. Bayesian Network to Detect Java Card Mutants

Ilhame El Farissi[a,]*, Mostafa Azizi[a], Jean-Louis Lanet[b], Mimoun Moussaoui[a]

[a]*MATSI LAB, ESTO,Mohammed first University,Oujda, Morocco*
[b]*SSD Team Xlim,University of Limoges, Limoges, France*

Abstract

Being a vital element for the different domains such as communication system, authentication, and payment, multiple attackers manipulate the Card fraudulently in order to access to the services offered by this one. Smartcards are often the target of software and hardware attacks. The most recent attacks are based on fault injection which modifies the application behavior. By disrupting the Java Card operation, the fault attack modifies the compiled code intended to be executed in order to meet what the attacker wants instead of the initial program. So, to tackle this problem, we suggest two classification and detection methods based on artificial intelligence, especially the neural and Bayesian networks. Then, we compare between the obtained results of these two methods in terms of the detection rate.

Keywords: SmartCard ; Java Card ; Security; Neural network; Bayesian network; Fault attack; Classification; Detection; Mutant;

1. Introduction

A smartcard is a secure, efficient and cost effective embedded system device comprising of a microcontroller, memory modules (RAM, ROM, EEPROM) serial input/output interfaces and data bus. One

* Corresponding author. Tel.: +212-671-590-065; fax: +212-536-505472.
E-mail address: ilhame.elfarissi@gmail.com.

doi:10.1016/j.aasri.2013.10.021

chip operating system is contained in ROM and the applications are stored in the EEPROM. A smartcard can also be viewed as an intelligent data carrier which can store data in a secure manner and ensure the integrity, the confidentiality and the availability, then the security of data during transactions. Security is one of the important issues in smartcard development and the level of threat imposed by malicious attacks on the integrated software is of high concern. Multiple means are operated to return the secret information, fault injection seems to be the most efficient and effective one. In order to prevent and detect such attacks, smartcard manufacturers try to implement new options in their operating systems. Software fault injection is a technique of producing errors into the program, this technique allows attackers to analyze the system behavior, study its internal state and modify the application execution in order to deduce secret information.

It has been demonstrated in [1] that it is possible to predict the ability of an application to generate a hostile code while the smartcard is hit by a laser. Unfortunately this tool is based on a brute force attack and thus can only be used as a prevention tool. To detect such an attack during the run-time, several counter measures have been described in the literature but often they require additional information. For current platform, there is no solution that can be adapted to different effects of the attack. So, we propose here new schemes based from one side on Neural networks and on Bayesian Network from the other side, to detect during the run-time if an application has been mutated.

2. Java based smartcard

Java Card is a platform of smart cards. It is based on Java, but it has its own specifications in three aspects:

- Restriction of language
- The Run time environment
- The applet life cycle

Furthermore, due to the limited resources in smart cards, the Java Card Virtual Machine (JCVM) is split into two parts:

- Off-card: The byte code verifier (invoking the converter) that converts the java Card program to CAP file and verifies its validity is executed off-card.
- On-card: This part contains the interpreter, the API and the Java Card Run time Environment (JCRE).

Smart cards are nowadays used throughout the world on a daily basis, wherever the notion of digital security appears. Smart cards have strong constraints as for computing power and for memory size. Based on the Java technology, the Java Card architecture is compound of different layers on top of the smart card hardware and the operating system. The most important component is the Java Card Runtime Environment which includes the Java Card Virtual Machine (JCVM) and the API. The JCVM can somehow be seen as an abstract processor with its own instruction set and internal mechanisms. The JCVM provides several mechanisms enforcing the security of the system, based on the security mechanisms inherited from the standard Java specifications and some specific to this platform. As specific component dedicated to the security the firewall enforces applications isolation with the concept of security context. Due to limited memory an important component for the security the Byte Code Verifier (BCV) must be executed on a host platform. Because of the sensitivity of the data they contain and of their inherent robustness, smart cards are the target of particular kinds of attack: hardware attacks.

3. Fault Attack

Developed by Bonech, DeMillo and Lipton [3], the fault attack aims to disrupt the physical environment of the processor in order to produce errors. Initially, the fault attack targeted public-key cryptographic algorithms such as RSA and DES. Faults are injected into the chip by perturbing its environment execution.

Consequently, the writable memories (code and variable modifications) or the chip registers(the stack pointer, the program counter…) are disrupted. These perturbations allow an attacker to gain illegal access to system data or services. In the literature [4,5,6,7], there are several means to produce fault attacks into smart card but presently laser beam attack is the most efficient and difficult to tackle with.

4. Mutant application

The fault attacks targeting the Java Card, aim to generate mutants by modifying one or more instructions in order to execute what the attacker wants instead of the original program. This class of attack can be used, for example, to avoid the PIN verification by replacing the instruction corresponding by the 'nop' one equivalent to 'nothing'.

There are many types of mutants and multiple mechanisms allow their detection partially. In this paper, we propose to ameliorate the protection techniques against the mutants by adopting artificial intelligence methods such as the neural network and Bayesian network which are able to detect any changes made to the system and then we compare between the obtained results in the two cases.

In order to shape the fault attack effect into a Java Card, we use the function debit() of the applet wallet illustrated as follows: [8]

```
Private void debit(APDU apdu) {
if ( pin.isValidated() ) {  // access authentication
        …
        byte amountDebit = buffer[ISO7816.OFFSET_CDATA];  // get debit amount
        if (( amountDebit < 0 ) || ( amountDebit > AMOUNT_MAX_TRANSACTION) ) // check debit amount
                ISOException.throwIt(AMOUNT_SW_INVALID_TRANSACTION);
        if ( (short)( balance - amountDebit ) < (short)0 )          // check the new balance
                ISOException.throwIt(SW_BALANCE_NEGATIVE);
        balance = (short) (balance - amountDebit);
 } else {
ISOException.throwIt( SW_REQUIRED_PIN_VERIFICATION);
}
```

The mutant generation was initially defined simultaneously by [9,10] using the combined attacks. In our case, after compilation of the program above, we obtain the .class file, which must be loaded in the card.

In the normal case, the byte code executed must correspond to the code loaded in the card. But, the fault attacks stated above modify it. This change is called mutant.

Some of existing mutants allow the attackers to execute a piece of code without the required rights. For example, the debit() function allows the user to charge the card balance if the code PIN entered is validated. This verification is performed by the instruction ‘if(pin.isValidated()) which corresponds in the byte code to the instruction ‘ifeq 91’. This one can be neglected by replacing it by the instruction ‘nop’.

5. Realization

5.1. Construction of the control flow graph

In order to ensure the Java Card operating accurately, and that the program executes all the conditional instructions like ‘ifeq’, it was necessary to create the graph flow control which represents all the possible paths taken by the program during its interpretation and whose vertices are the elementary blocks.

The elementary blocks whose entry point does not contain code that is the target of a jump instruction and the exit point is an instruction:

- Which starts a method

- An unconditional branch target(ret , goto,goto_w, jsr_w , jsr)
- A conditional branch target(iflt ,ifeq, ifne , ifle, ifge , ifgt, ifnonnull , ifnull, if_icmpne , if_icmpeq, if_icmpgt , if_icmplt, if_icmpge , if_icmple, if_acmpne , if_acmpeq, fcmpl , lcmp, dcmpl, fcmpg , dcmpg)
- A conditional composed branch target (lookupswitch, tableswitch)
- Following the return instruction type(return, ireturn, dreturn, lreturn, freturn, areturn).

Also reinforce the graph control flow generated [11] by adding a verifying system that verifies whether the program execution was done normally or not. In practical terms, we talk about a smart system integrated into the card which must be able to distinguish between the normal interpretation of the Java Card applet and the attack attempt.

5.2. Neural network Definition

The artificial neural network is based on the biological neurons functioning. It is composed of several elements interconnected in order to solve analysis problem, diagnosis, classification one…

As the biological neuron system, the artificial neural networks have the ability to learn and respond to problems which must be part of what he has learned. An artificial neural network is composed of three types of layers; input, hidden and output ones. Each layer contains one or multiple neurons. The neuron is considered as an automaton with threshold. To be activated, it must receive a signal exceeding the threshold. Its output, which is calculated by taking into account the parameter 'synaptic weight', aliments the neurons belonging to the next layer. [12]

The strengths of the neural network reside in learning, objects identification and a better interpretation. Because of that, there has been a reason for using it to detect intrusions [13]. Since, a strong and intelligent neural network is not necessary large, we thought to implement it into a smartcard in order to detect the fault attacks.

This is what we have done previously when we have used the physical parameters of the card to aliment the network. Theoretically, the principle of implementation is possible and the system designed in [14] is able to detect any changes made in the electrical properties. But practically, these parameters are under the control of the attacker. Therefore, it is necessary to change the entry points of the network.

Consequently, instead of electrical properties we have taken the first instructions numbers of vertices of the control flow graph.

And with respect to the output, we have a single one that takes:

- 0 if it is a normal case program execution
- 1 if it is an abnormal case meaning an attack

5.3. Naive Bayesian network

The naive Bayesian network is the simplest form of a Bayesian Network (BN). It contains one parent and multiple leaf nodes, with the assumption that there is a high degree of independence between the leaves in the context of their parent. [15]

The principle of naive Bayesian network is different from that of the neural network. Indeed, the naive Bayesian network is based on the calculation of the conditional probabilities of each input in the context of his parent.

Also used to detect intrusions [16,17], we have provided to the Bayesian network the same data used in the neural network. And after the calculation of the conditional probabilities associated with each input variable, the network generates the class:

- C0 in the normal operation of the program

- C1 in the case of an attack.

5.4. Obtained results

The construction of a neural network involves two phases, learning and validation. During learning, the network calculates the output, compares it with the desired one in order to adjust the "weight" in such a way as to obtain the minimum possible difference between the values of the two outputs.

To test the network, it is necessary to pass on new patterns, in our case, new attacks. If the network returns the expected values in most cases, we say that the network can detect attacks. In the case of the debit() function, the network was able to detect 126 invalid paths from 131 and no false alarms (from 7 valid paths) was mounted.

According to the Bayesian network, first of all, it is necessary to calculate and fill the conditional probability tables. Then, we developed a C program to test the operation of the Bayesian network. Taking the same patterns used in the neural network, we found that the Bayesian network is able to detect only 25 attacks among 131. In addition of the low detection rate, the network considers four of valid paths as attacks.

The table below summarizes the results get, with:

Network	True positive	True negative	False positive	False negative
Neural network	126	5	0	7
Bayesian network	25	106	4	3

5.5. The neural network simulation

From the results mentioned above, we opted for the neural network implementation. Then, once the network is validated, we have developed it in C language in order to integrate it in a java virtual machine and simulate its operation.

On one hand, the c program is divided into two files. The first one contains all necessary data to calculate the output, namely the synaptic weights. And the second file treats the input network which is returned from the virtual machine, by using the appropriate parameters and calculates the output via the backpropagation equation.

On the other hand, according to the rules of control flow graph construction, we have modified the open source Java virtual machine that we have exploited(Avian[18]) in such a way that the integrated neural network is alimented by the number of the first instruction of the blocks executed during the class interpretation.

At the end of the class interpretation, the network implemented in the Avian virtual machine has all the necessary elements to calculate the output which corresponds to 0 in the normal functioning and take the value 1 in 96% of the abnormal cases.

6. Conclusion

In this paper we present a new approach for detecting a fault attack on a Java based smartcard. We propose to evaluate the integration into the interpreter of two different techniques Neural Network and Bayesian Network. We compared the two approaches and the Neural Network seems suitable for this highly constrained device. We used an open source Java virtual machine to implement our counter measures. The size of both of them is affordable with the smartcard domain and they are compatible with the platforms. The platform that implements this countermeasure has a competitive advantage. In fact, an application executed on a platform without detection system will be more prone to fault attack than if the platform contains this countermeasure.

According to this study, we found that neural networks are more suitable in the case of mutant detection in a smartcard, but the problem is that the smartcard does not support floating-point numbers and the network needs this type of numbers. Therefore, it is necessary to emulate the floating-point calculation, and to pay attention to the program size and the execution time.

References

[1] J.-B. Machemie, C. Mazin, J.-L. Lanet, J. Cartigny, SmartCM A Smartcard Fault Injection Simulator, IEEE International Workshop on Information Forensics and Security (WIFS 2011) pp. 1-6, Foz do Iguaçu, Brazil, November 29th-December 2nd, 2011

[2] D.Bonech, R.DeMillo, R.Lipton, New Threat Model Breaks Crypto Codes. Bellcore Press Release, 25 September, 1996.

[3] Sun Microsystems. Java Card TM 2.2.2 Virtual Machine (JCVM) Specification, Mars 2006.

[4] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks, Springer-Verlag, 2002, pp.2-12.

[5] C. Aumuller, P. Bier, W. Fischer, P. Hofreiter, and J.P. Seifert. Fault attacks on RSA with CRT: Concrete results and practical countermeasures. Lecture Notes in Computer Science, pages 260–275, 2003.

[6] O.Kmmerling, M.-G. Kuhn, Design principles for tamper-resistant smartCard processors, WOST'99 Proceedings of the USENIX Workshop on SmartCard Technology on USENIX Workshop on SmartCard Technology, pp. 9-20,1999.

[7] J.J. Quisquater and D. Samyde. Eddy current for magnetic analysis with active sensor. In Proceedings of Esmart, volume 2002, 2002.

[8] A.A.Sere, J.Iguchi-Cartigny, J-L.Lanet, Automatic detection of fault attack and countermeasures, In: proceedings of the 4th workshop on Embedded Systems Security. ACM. 2009, pp. 1-7.

[9] G.Barbu, H.Thiebeald, and V.Guerin, Attacks on Java Card 3.0 Combining Fault and Logical Attacks, In: Smartcard Research and Advanced Application, Card is 2010 LNCS 6035 (2010),pp. 148-163.

[10] E.Vetillard and A.Ferrari, Combined Attacks and Countermeasures, In: Smartcard Research and Advance Application, Card is 2010 LNCS 6035 (2010), pp. 133-147.

[11] G. Bouffard, J.-L. Lanet, J.-B. Machemie, J.-Y. Poichotte and J.-P. Wary, Evaluation of the Ability to Transform SIM Applications into Hostile Applications, CARDIS 2011, LNCS 7079, September 14-16, 2011, pp. 1-17.

[12] J.P.Rennard, Neural networks, Vuibert, ISBN 2711748308, First Edition, 2006.

[13] R.Beghdad, Critical study of neural networks in detecting intrusions, Computers & security 27, p 168-175, 2008.

[14] I.El Farissi, M.Azizi, M.Moussaoui, Detection of smartcard attacks using neural networks, International Conference on Multimedia Computing and Systems (ICMCS) 2012, pp. 949-954.

[15] S.Benferhat, T.Kenaza, to an overall assessment of Bayesian classifiers for intrusion detection, 5^{th} francophone days on Bayesian network(JFRB2010),2010

[16] N.Ben Amor, S.Benferhat, Z.Elouedi, Naïve Bayes vs Decision Trees in Intrusion Detection Systems, ACM Symposium on Applied Computing, 2004

[17] S.Benferhat, T.Kenaza, A.Mokhtari, naive Bayesian networks for detecting coordinated attacks, francophone days on Bayesian network(JFRB2008),2008.

[18] http://oss.readytalk.com/avian/index.html