Mitigating Type Confusion on Java Card

Jean Dubreuil, Smart Secure Devices (SSD) Team, University of Limoges, Limoges Cedex, France

Guillaume Bouffard, Smart Secure Devices (SSD) Team, University of Limoges, Limoges Cedex, France

Bhagyalekshmy N. Thampi, Smart Secure Devices (SSD) Team, University of Limoges, Limoges Cedex, France

Jean-Louis Lanet, Smart Secure Devices (SSD) Team, University of Limoges, Limoges Cedex, France

ABSTRACT

One of the challenges for smart card deployment is the security interoperability. A smart card resistant to an attack on a given platform should be able to guarantee the same behavior on another platform. But the current implementations do not comply with this requirement. In order to improve such standardization the authors propose a framework based on annotations with an external pre-processing to switch the Java Card Virtual Machine (JCVM) into a secure mode by activating a set of countermeasures. An example has been proposed in this paper for implementing a countermeasure against type confusion with a fault attack. Smart cards are often the target of software, hardware or combined attacks. In recent days most of the attacks are based on fault injection which can modify the behavior of applications loaded onto the card, changing them into mutant applications. This countermeasure requires a transformation of the original program byte codes which remain semantically equivalent. It needs a modification of the JCVM which stays backward compatible and a dedicated framework to deploy these applications. Thus, the proposed platform can resist to a fault enabled mutant.

Keywords: Countermeasures, Faults, Java Card, Smart Card, Typed Stack, Viruses

1. INTRODUCTION

Owing to NFC technology, the usage of mobile phone for e-transaction will increase drastically in upcoming years. This transition in technology requires more applications to be developed in mobile phones and some of them are highly sensitive and need high level security. For example, nowadays using mobile phones for payment, ticketing, mobile TV, etc. are common. Thus, it raises the question of confidence of the terminal that delivers these services. If a platform is multi-applicative, then loading a new application can reduce the security of already installed application. Through virtualization, the platforms have reached interoperability.
at functional level and it will be difficult to obtain it at security level. In case of this security interoperability property has not been achieved, then the consequence will imply a costly certification process for each application, for all configurations of the application. This could be a main point for not adopting NFC applications and thereby reducing the development’s effort for mobile transactions. Critical applications in a mobile phone should be hosted by a Secure Element (SE) often implemented with a smart card.

Mobile Network Operators (MNO) are looking to open their native SE: the Universal Subscriber Identity Module (USIM) Card (ESTI, 2005) to the third party service providers, in order to allow them to develop value added services, like m-payment or m-transport etc., on NFC based technology. To warrant security for third party applications hosted in USIM Card, MNO have chosen to certify their SE under the Common Criteria (CC) (The Common Criteria for Information Technology Security Evaluation) scheme (Common Criteria Organization, 2012). This will facilitate post issuance for third party applications downloading without existing CC certification loss which means an industrial process will have to be in place, which has to warrant the innocuousness of every candidate’s application for a download. In particular, an application certified on a given platform must have the same behavior in other platforms also.

Nowadays most of the USIM cards are based on a Java Card Virtual Machine (JCVM). Java Card is a type of smart card that implements the standard Java Card 3.0 (Sun, 2010) in one of the two editions “Classic Edition” or “Connected Edition”. Such a smart card embeds a Virtual Machine (VM), which interprets application byte codes already romized with the operating system or downloaded after issuance. Due to security reasons, the ability to download code into the card is controlled by a protocol defined by Global Platform (Global Platform, 2011). This protocol ensures that, the code owner has the necessary credentials to perform the particular action.

Java Cards have shown an improved robustness compared to native applications concerning many attacks. They are designed to resist numerous attacks using both physical and logical techniques. Currently, the most powerful attacks are hardware based attacks and especially fault attacks. A fault attack modifies a part of the memory content or signal on internal bus and leads to deviate from the exploitable behavior, not by an attacker. A comprehensive consequence of such attacks is mentioned in Iguchi-Cartigny and Lanet (2010). Although fault attacks have been generally used in the literature from a cryptanalysis point of view (Aumüller, Bier, Fischer, Hofreiter, & Seifert, 2003; Hemme, 2004; Piret & Quisquater, 2003), they can also be applied to every code layer embedded in a device. For example, by choosing the exact byte of a program, an attacker can bypass logical tests. To avoid such attacks, several countermeasures have been designed to protect the execution flow, the integrity of Java fields, the confidentiality of the byte code, etc.

Designing efficient countermeasures against fault attacks are important not only for smart card manufacturers but also for application developers. Manufacturers need countermeasures with the lowest cost in terms of memory and processor usage. The coverage (reduction of the number of succeeding attacks) and the detection latency (number of instructions executed between an attack and its detection) are the most important metrics in the development process of smart card. In order to minimize the impact of fault attacks, developers need to implement countermeasures in their applicative code. Examples of such applicative countermeasures are redundant branch instructions, counters, redefining the value of true and false constants, etc. But in this case the developer should have the knowledge of underlying platform architecture, which can differ from each smart card model. This low interoperability of the security aspects between different platforms is a huge problem for smart card application developments and certifications.
Software Quality and the Open Source Process
www.igi-global.com/chapter/software-quality-open-source-process/29527?camid=4v1a

Combining Indexing Units for Arabic Information Retrieval
www.igi-global.com/article/combining-indexing-units-for-arabic-information-retrieval/166540?camid=4v1a