Fault enabled viruses against smart cards

Samiya Hamadouche¹, Jean-Louis Lanet², Mohamed Mezghiche¹

¹ LIMOSE Laboratory, Computer Science Department, Faculty of Sciences, University M’Hamed Bougara of Boumerdès, Independency Avenue, 35000 Algeria
{hamadouche-samiya, mohamed-mezghiche}@umbb.dz
² INRIA LHS-PEC
jean-louis.lanet@inria.fr

Abstract. Smart cards are very secured devices designed to execute applications and store confidential data. Therefore, they become the target of many hardware and software attacks that aim to bypass their embedded security mechanisms in order to gain access to the sensitive stored data. Recently, a new kind of attacks called combined attacks has appeared. They aim to induce perturbations in the application’s execution environment. Thus, correct and legitimate application can be dynamically modified to become a hostile one after being loaded in the card using a fault injection. We call such an application a fault enabled virus. The main purpose of this work in progress is to find a methodology to hide a malicious code inside a well-typed program so that the resulting one is semantically correct (respecting the Java Card Virtual Machine specification) even after the fault injection. Initially, we have demonstrated the possibility to design applications in such a way that they become intentionally hostile after being hit by a laser [1]. This virus construction gets back to a Constraint Satisfaction Problem (CSP) [2]. Indeed, it requires to find a sequence of instructions to add at the beginning of the code to hide in order to join a fragment of inoffensive code in such a way that several constraints are verified. The construction of that sequence must solve two problems: choosing an instruction among the existing ones and computing the memory state preceding it in order to reach the desired state (the end of the inoffensive code). The idea is to represent this problem as a search tree in which the root is the first instruction of the hostile code and at each level the nodes represent the candidate instructions that may precede the parent one. A Branch and Bound method is used to create and explore the tree in order to find paths from the root to the leaves (representing the desired state). Each one of these paths corresponds to a possible wanted sequence.

References