Genericity of a model-based intrusion testing method

Aymerick Savary[1,2], Mathieu Lassale[1,2], Jean-Louis Lanet[1], and Marc Frappier[2]

[1] Université de Limoges
[2] Université de Sherbrooke

Penetration testing is a way to detect vulnerability into a system. It is often a tedious task based on the know-how of the tester. We have proposed a methodology to improve this process through its automatisation called Vulnerability Tests Generation (VTG). This can be obtained thanks to a model of the System Under Test (SUT). It is based on the specification mutation and the MBT (Model-Based Testings) methods. To evaluate the genericity of this method, we apply it on two different cases studies. The SV (Structure Verification) mechanism of the Java Byte Code Verifier (BCV) and the EMV (Europay Mastercard Visa) protocol.

In this paper we present the adaptation of the method and the first results obtained. The SV mechanism covers the static constraint model (without transition), while EMV which is a Java Card application, corresponds to a more abstract model.

We used the Event-B language through the Rodin platform to model our SUT. We make the assumption that the accepted behavior of our SUT are completely represented by the initial model, so the unmodeled behavior represented unaccepted behavior of our SUT. By applying some mutation, that relax constraints on the initial model, we try to explore the unmodeled behavior. Because the unmodeled behavior is a huge space, we need to control the mutation with mutation criteria. In our tool, we have implemented mutation rules for first order predictates, corresponding to axioms, invariants and guards in Event-B. The result of the mutation of predicate is then used to generate the mutant contexts and mutant machines. For context, the mutant contexts are obtained by making the Cartesian product between sets composed by the original axiom and its mutations. Because we are looking for faults in the SUT to fix it, we want to obtain the minimum fault set as possible. For machines, the mutation of invariants is possible but does not correspond to our test purpose. The mutant event is obtained by replacing it's guard by it's mutation. If several mutations exist for the guard of an event, each mutation will generate a new event. The new machine is generated by cloning the initial machine, then a mutant event is added. The original event could be kept or not in the faulty machine depending on the user mutation criteria choice. To implement these mutations, we propose a Rodin plug-in based on Rodin and TOM.

The abstract tests are extracted from the mutants model by MBT algorithms. For Event-B contexts we look for values that satisfy the axioms. Because a mutant axiom is in the set of axioms of a mutant context, the values will also satisfy the mutant axiom. In the case of machines, we look for traces that reach our mutant event. This traces will be the preamble of our tests. The body will be the fault event. ProB is used as a MBT tool to extract tests corresponding to attacks.

To test the EMV protocol, the first version of VTG have been used without any modification. The model of the SV mechanism is only composed by Event-B contexts. VTG had not been designed to mutate contexts. We proposed a new algorithm, based on the same predicate mutation rules. We also improved our implementation. With the use of TOM, the user can propose his own mutation rules for predicate.