Chapter 3
Symbolic-Based Monitoring for Embedded Applications

Pramila Mouttappa
Institut Mines-Telecom, France

Stephane Maag
Institut Mines-Telecom, France

Ana Cavalli
Institut Mines-Telecom, France

ABSTRACT

Testing embedded systems to find errors and to validate that the implemented system as per the specifications and requirements has become an important part of the system design. The research community has proposed several formal approaches these last years, but most of them only consider the control portion of the protocol, neglecting the data portions, or are confronted with an overloaded amount of data values to consider. In this chapter, the authors present a novel approach to model protocol properties of embedded application in terms of Input-Output Symbolic Transition Systems (IOSTS) and show how they can be tested on real execution traces taking into account the data and control portions. These properties can be designed to test the conformance of a protocol as well as security aspects. A parametric trace slicing approach is presented to match trace and property. This chapter is illustrated by an application to a set of real execution traces extracted from a real automotive Bluetooth framework with functional and security properties.

INTRODUCTION: HOW IT ALL BEGAN

Embedded systems are becoming increasingly ubiquitous, controlling a wide variety of popular and safety-critical devices. Testing is the most commonly used method for validating software systems, and effective testing techniques could be helpful for improving the dependability of these systems. However, there are challenges involved in developing such techniques. So, how is this testing performed ? Usually, testing is performed by executing experiments on the implementation, by making observations during the execution of the tests and by subsequently assigning a verdict about the correct functioning of the system. This method is called *active testing*. But, this is not always possible for systems that operate continuously and where direct interfaces are not provided. Further, interfering with such systems can result in

DOI: 10.4018/978-1-4666-6194-3.ch003

misbehavior of the system. To compensate for the limitations of active testing techniques, we have another interesting technique called *passive testing* or *monitoring*. Passive testing can be considered as one of the promising technologies to meet the challenges imposed on software testing.

Passive testing (or Monitoring) consists of recording the trace (i.e., sequence of exchange of messages) produced by the implementation under test and mapped to the property to be tested or specification if it exists. This technique has proved to be a powerful technique for reactive systems (like communication protocols, embedded systems, etc.) testing by observing its input/output behaviors (implementation traces) without interrupting its normal operations. Passive testing helps to observe abnormal behavior in the implementation under test on the basis of observing any deviation from the predefined behavior. This deviation can also sometimes match with certain attack patterns. As the systems evolve, network protocol messages become richer with data values. They are defined as control and data portions. Nevertheless, many works on passive testing are focused only on checking the control portion of the message without taking into account the data part which may result in producing false positive verdicts as explained in (Che, Lalanne, & Maag, 2012).

In this chapter, we introduce a new passive testing methodology based on the integration of symbolic execution of an IOSTS and parametric trace slicing techniques. From the literature we see that most of the work (Bentakouk, Poizat, & Zaidi, 2011; Weiglhofer, Aichernig, & Wotawa, 2010; Gaston, Le Gall, Rapin, & Touil, 2006) is based on active testing using a symbolic execution approach. IOSTS are used here for modeling communicating systems interacting with their environment (behavior or attack). The most important aspect of this IOSTS formalism is that the parameters and variable values are represented by *symbolic values* instead of *concrete data values*, which helps in avoiding the necessity for data enumeration. The IOSTS model is symbolically executed to obtain a tree-like structure with different branches constituting the behavior or attack scenario. The branches or the behaviors of the symbolic tree are monitored against the real system trace using passive testing approach.

The technique of parametric trace slicing (Chen & Rosu, 2009) is used for trace analysis. Trace analysis plays a very important part in passive testing. A parametric trace is defined as a trace containing events with parameters that have been bound to a concrete data value (i.e., valuation) and parametric trace slicing is defined as a technique to slice (or cut) the real protocol execution trace into various slices based on this valuation. We then apply the symbolic execution of properties on the trace slices to provide a test verdict. This approach has been applied to passively test the traces of the Bluetooth protocol implemented in an embedded system dedicated to an automotive framework experimented through the ITEA2 DIAMONDS1 project.

More precisely, the main contributions of this chapter include:

- We briefly define how to formally model the embedded system properties and possible attacks using IOSTS formalism and also discuss the advantages of symbolic passive testing over other existing passive testing or monitoring methodologies.
- The definition of an algorithm for parametric trace slicing by taking into account the data portions contained in the trace events.
- The definition of a novel algorithm to check whether an IOSTS property is satisfied on a real execution trace. We also demonstrate that security attacks can be monitored by this approach.
- Demonstrations on real execution traces extracted from an embedded automotive Bluetooth system have been conducted with a prototype tool.