Towards a Model-Based Testing Framework for the Security of Internet of Things for Smart City Applications

Moez Krichen[1,2(✉)], Omar Cheikhrouhou[3,4], Mariam Lahami[2], Roobaea Alroobaea[3], and Afef Jmal Maâlej[2]

[1] Faculty of CSIT, Al-Baha University, Al Baha, Saudi Arabia
moez.krichen@redcad.org
[2] ReDCAD Laboratory, University of Sfax, Sfax, Tunisia
{mariam.lahami,afef.jmal}@redcad.org
[3] College of CIT, Taif University, Taif, Saudi Arabia
{o.cheikhrouhou,r.robai}@tu.edu.sa
[4] ISIMA, University of Monastir, Monastir, Tunisia

Abstract. This is a work in progress in which we are interested in testing security aspects of Internet of Things for Smart Cities. For this purpose we follow a Model-Based approach which consists in: modeling the system under investigation with an appropriate formalism; deriving test suites from the obtained model; applying some coverage criteria to select suitable tests; executing the obtained tests; and finally collecting verdicts and analyzing them in order to detect errors and repair them.

Keywords: Internet of Things · Smart cities · Security models Generation · Coverage · Test · Security · Verdicts

1 Introduction

Internet of Things (IoT) is a promising technology that permits to connect everyday things or objects to the Internet by giving them the capabilities to sense the environment and interact with other objects and/or human beings through the Internet.

This evolving technology has promoted a new generation of innovative and valuable services. Today's cities are getting smarter by deploying intelligent systems for traffic control, water management, energy management, public transport, street lighting, etc. thanks to these services. Nevertheless, these services can easily be compromised and attacked by malicious parties in the absence of proper mechanism for providing adequate security.

Recent studies have shown that the attackers are using smart home appliances to launch serious attacks such as infiltrating to the network or sending malicious email or launching malicious actions such as Distributed Denial of Service (DDoS) attack. Therefore, security solutions need to be proposed, set up and tested to mitigate these identified attacks.

R. Mehmood et al. (Eds.): SCITA 2017, LNICST 224, pp. 360–365, 2018.
https://doi.org/10.1007/978-3-319-94180-6_34

In this work, we aim to adopt a Model-Based Security Testing (MBST) approach to check the security of IoT applications in the context of smart cities. The MBST approach consists in specifying the desired IoT application in an abstract manner using an adequate formal specification language and then deriving test-suites from this specification to find security vulnerabilities in the application under test in a systematic manner.

The work introduced here is a piece of a broader approach dealing with the security of IoT applications for smart cities and consisting of the following steps:

- Identify and assess the threats and the attacks in smart cities IoT applications.
- Design and develop security mechanisms for standard protocols at the application and the network layer.
- Evaluate the performance and the correctness of the proposed security protocols using simulation and implementation on real devices.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries about IoT and smart cities. Section 3 discusses main threats and challenges related to these two fields. Section 4 presents our approach. Section 5 reports on related research efforts dealing with IoT security testing. Finally Sect. 6 concludes the paper.

2 Preliminaries

2.1 Internet of Objects

Recent advances in communication and sensing devices make our everyday objects smarter. This smartness is resulted from the capability of objects to sense the environment, to process the captured (sensed) data and to communicate it to users either directly or through Internet. The integration of these smart objects to the Internet infrastructure is promoting a new generation of innovative and valuable services for people. These services include home automation, traffic control, public transportation, smart water metering, waste and energy management, etc. When integrated in a city context, they make citizens' live better and so form the modern smart city.

2.2 Smart Cities

In October 2015, ITU-T's Focus Group on Smart Sustainable Cities (FG-SSC) agreed on the following definition of a smart sustainable city: "A Smart Sustainable City (SSC) is an innovative city that uses information and communication technologies (ICTs) and other means to improve quality of life, efficiency of urban operation and services, and competitiveness, while ensuring that it meets the needs of present and future generations with respect to economic, social and environmental aspects".

3 Threats and Challenges

3.1 Threats

Indeed, connecting our everyday "things" to the public Internet opens these objects to several kinds of attacks. Taking the example of a traffic control system. If the hackers could insert fake messages to these traffic control system devices, they can make traffic perturbations and bottlenecks. Another example related to home automation, if attackers gain access to smart devices such as lamps, doors, etc., it could manipulate doors and steal the house properties. The main security threats in the IoT are summarized and they can be summarized as follows: 1. Cloning of smart things by untrusted manufacturers; 2. Malicious substitution of smart things during installation; 3. Firmware replacement attack; 4. Extraction of security parameters since smart things may be physically unprotected; 5. Eavesdropping attack if the communication channel is not adequately protected; 6. Man-in-the-middle attack during key exchange; 7. Routing attacks; 8. Denial-of-service attacks; and 9. Privacy threats.

3.2 Challenges

Due to its specific characteristic, new issues are raised in the area of IoT:

- Data collection trust: If the huge collected data is not trusted (e.g., due to the damage or malicious input of some sensors), the IoT service quality will be greatly influenced and hard to be accepted by users.
- User privacy: In order to have intelligent context-aware services, users have to share their personal data or privacy such as location, contacts, etc. Providing intelligent context-aware services and at the same time preserving user privacy are two conflicting objectives that induce a big challenge in the IoT.
- Resource Limitation: Most of IoT devices are limited in terms of CPU, memory capacity and battery supply. This renders the application of the conventional Internet security solutions not appropriate.
- Inherent complexity of IoT: the fact that multiple heterogeneous entities located in different contexts can exchange information with each other, further complicates the design and the deployment of efficient, inter-operable and scalable security mechanisms.

4 Proposed Approach

In this section, we define a workflow that covers the different steps of a classical model based testing process, namely: Model Specification, test generation, test selection, test execution and evaluation activities as depicted in Fig. 1.

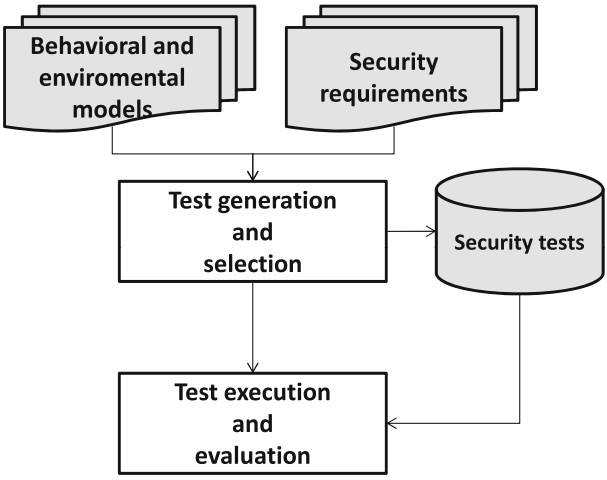

Fig. 1. Model based security testing process.

4.1 Modelling Issues

We use *timed automata* [2] with deadlines [4] to model the applications under test and the security aspects of interest. A timed automaton over the set of actions Act is a tuple $A = (Q, q_0, X, \mathsf{Act}, e)$, where: Q is a finite set of *locations*; $q_0 \in Q$ is the initial location; X is a finite set of *clocks*; e is a finite set of *edges*. Each edge is a tuple (q, q', ψ, r, d, a), where: $q, q' \in Q$ are the source and destination locations; ψ is the *guard*, a conjunction of constraints of the form $x \# c$, where $x \in X$, c is an integer constant and $\# \in \{<, \leq, =, \geq, >\}$; $r \subseteq X$ is a set of clocks to *reset* to zero; $d \in \{\mathsf{lazy}, \mathsf{delayable}, \mathsf{eager}\}$ is the *deadline*; $a \in \mathsf{Act}$ is the action.

4.2 Test Generation and Selection

The used test generation technique is based on model checking. The main idea is to formulate the test generation problem as a reachability problem that can be solved with the model checker tool UPPAAL [3]. However, instead of using model annotations and reachability properties to express coverage criteria, the observer language is used.

In this direction, we reuse the finding of Hessel et al. [7] by exploiting its extension of UPPAAL namely UPPAAL CO$\surd$ER[1]. This tool takes as inputs a model, an observer and a configuration file. The model is specified as a network of timed automata (.xml) that comprises a SUT part and an environment part. The observer (.obs) expresses the coverage criterion that guides the model exploration during test case generation. The configuration file (.cfg) describes mainly the interactions between the system part and the environment part in terms of

[1] http://user.it.uu.se/~hessel/CoVer/index.php

input/output signals. It may also specify the variables that should be passed as parameters in these signals. As output, it produces a test suite containing a set of timed traces (.xml).

Our test generation module is built upon these well-elaborated tools. The key idea here is to use UPPAAL CO$\sqrt{}$ER and its generic and formal specification language for coverage criteria to generate tests for security purposes.

4.3 Test Execution and Verdict Analysis

For the execution of the obtained security tests, we aim to use a standard-based test execution platform, called TTCN-3 test system for Runtime Testing (TT4RT), developed in a previous work [9]. To do so, security tests should be mapped to the TTCN-3 notation since our platform supports only this test language. Then, test components are dynamically created and assigned to execution nodes in a distributed manner.

Each test component is responsible for (1) stimulating the SUT with input values, (2) comparing the obtained output data with the expected results (also called oracle) and (3) generating the final verdict. The latter can be pass, fail or inconclusive. A pass verdict is obtained when the observed results are valid with respect to the expected ones. A fail verdict is obtained when at least one of the observed results is invalid with respect to the expected one. Finally, an inconclusive verdict is obtained when neither a pass or a fail verdict can be given. After computing for each executed test case its single verdict, the proposed platform deduces the global verdict.

5 Related Work

In this section we give a very brief overview on contributions form the literature and from our previous work related to Model-Based Security Testing (MBST) for IoT Applications in Smart Cities.

Authors of [6] propose a good survey on more than one hundred publications on model-based security testing extracted from the most relevant digital libraries and classified according to specific criteria. Even though this survey reports on a large number of articles about MBST it does not contain any reference to IoT applications or Smart Cities. Contrary to that the authors of [1] propose a model-based approach to test IoT platforms (with tests provided as services) but they do not deal with security aspects at all.

In this work we aim to combine these two directions namely: Model-Based testing and Security Testing for IoT applications in Smart Cities. For that purpose we will take advantage of our previous findings [5,8–10] related to these fields. In [5] a survey about Secure Group Communication in Wireless Sensor Networks is proposed. We will extend the notions proposed in this survey to the case of IoT applications. We will also exploit our previous results about test techniques of dynamic distributed systems [8,9]. Finally we will adopt the same methodology as in [10] to combine security and load tests for IoT applications.

6 Conclusion

Our work is at its beginning and a lot of efforts are needed at all levels on both theoretical and experimental aspects. First we need to deal with modelling issues. In this respect we need to extend our modelling formalism and to identify the particular elements of IoT applications to model (using extended timed automata). Models must not be big in order to avoid test number explosion. For that purpose we need to keep an acceptable level of abstraction. As a second step we have to adapt our test generation and selection algorithms to take into account security requirements of the applications under test. The new algorithms must be validated theoretically and proved to be correct. In the same manner we need to upgrade our tools to implement the new obtained algorithms. Finally we need to validate our approach with concrete examples with realistic size.

References

1. Ahmad, A., Bouquet, F., Fourneret, E., Le Gall, F., Legeard, B.: Model-based testing as a service for IoT platforms. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016, Part II. LNCS, vol. 9953, pp. 727–742. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47169-3_55
2. Alur, R., Dill, D.: A theory of timed automata. Theor. Comput. Sci. **126**, 183–235 (1994)
3. Behrmann, G., David, A., Larsen, K.G.: A tutorial on Uppaal. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30080-9_7
4. Bornot, S., Sifakis, J., Tripakis, S.: Modeling urgency in timed systems. In: de Roever, W.-P., Langmaack, H., Pnueli, A. (eds.) COMPOS 1997. LNCS, vol. 1536, pp. 103–129. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49213-5_5
5. Cheikhrouhou, O.: Secure group communication in wireless sensor networks: a survey. J. Netw. Comput. Appl. **61**, 115–132 (2016)
6. Felderer, M., Zech, P., Breu, R., Büchler, M., Pretschnerr, A.: Model-based security testing: a taxonomy and systematic classification. Softw. Test. Verif. Reliab. **26**(2), 119–148 (2016)
7. Hessel, A., Larsen, K.G., Mikucionis, M., Nielsen, B., Pettersson, P., Skou, A.: Testing real-time systems using UPPAAL. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) Formal Methods and Testing. LNCS, vol. 4949, pp. 77–117. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78917-8_3
8. Krichen, M.: A formal framework for black-box conformance testing of distributed real-time systems. IJCCBS **3**(1/2), 26–43 (2012)
9. Lahami, M., Krichen, M., Jmaïel, M.: Safe and efficient runtime testing framework applied in dynamic and distributed systems. Sci. Comput. Program. (SCP) **122**(C), 1–28 (2016)
10. Jmal Maâlej, A., Krichen, M.: A model based approach to combine load and functional tests for service oriented architectures. In: Proceedings of the 10th Workshop on Verification and Evaluation of Computer and Communication System, VECoS 2016, Tunis, Tunisia, 6–7 October 2016, pp. 123–140 (2016)