A New Model-based Framework for Testing Security of IoT Systems in Smart Cities using Attack Trees and Price Timed Automata

Moez Krichen[1,2] and Roobaea Alroobaea[3]
[1]*Faculty of CSIT, Al-Baha University, Saudi Arabia*
[2]*ReDCAD Laboratory, University of Sfax, Tunisia*
[3]*College of CIT, Taif University, Saudi Arabia*

Keywords: Model-based, Testing, Security, Internet of Things, IoT, Smart Cities, Attack Tree, Price Timed Automaton, UPPAALL, TTCN-3, Cloud.

Abstract: In this paper we propose a new model-based framework for testing security properties of Internet of Things in Smart Cities. In general a model-based approach consists in extracting test cases from a formal specification either of the system under test or the environment of the considered system in an automatic fashion. Our framework is mainly built on the use of two formalisms namely Attack Trees and Price Timed Automata. An attack tree allows to describe the strategy adopted by the malicious party which intends to violate the security of the considered IOT system. An attack tree is translated into a network of price timed automata. The product of the constructed price timed automata is then computed using the well known UPPAALL platform. The obtained timed automata product serves as input for the adopted test generation algorithm. Moreover our framework takes advantage of the use of the standardized specification and execution testing language TTCN-3. With this respect, the obtained abstract tests are translated into the TTCN-3 format. Finally we propose a cloud-oriented architecture in order to ensure test execution and to collect the generated verdicts.

1 INTRODUCTION

Nowadays Internet of Things (IoT) is playing an important role in our modern society as a technology which allows to connect everyday objects to Internet. These objects are equipped with sophisticated interfaces which give them the capabilities to measure physical aspects from the environment and to interact with other entities by exchanging specific messages.

This new technology has provided a wide generation of valuable and innovative services. In this manner, modern cities are becoming smarter by adopting intelligent systems for water management, traffic control, energy management, street lighting, public transport, etc. However, these services can massively be attacked and compromised by several malicious parties whenever adequate and appropriate security measures are absent.

A few years ago, the smart devices that make up the Internet of Things, such as light bulbs, thermostats, webcams, and many others , were seen as potential targets for attackers to activate or disable remote devices and to harm consumers. Today, the IoT no longer represents a mere target, but a real platform which may be used by attackers to launch dangerous remote aggressions and to cause very serious incidents.

With the emergence of wide-ranging Open Source worms, such as Mirai (Antonakakis et al., 2017), capable of spreading to tens of millions of IoT devices, attackers can exploit these systems to generate a massive influx of traffic and disconnect any company or institution from Internet. Beyond the massive attack of the network, these IoT attack platforms can present other forms of threats, such as information theft and passwords decryption.

The attacks on industrial control systems have taken a disturbing turn. Cyber criminals attack the operational core of vital infrastructures taking advantage of their vulnerability. Recent attacks have not only disrupted the provision of essential services, such as electricity, but have also damaged automation systems to return to normal operation.

For instance the 2015 and 2016 attacks in Ukraine (Boyte, 2017) that caused power outages were perfectly planned and coordinated. The attackers managed to hijack the automation systems to cause power outages and then perform a well-ordered

Krichen, M. and Alroobaea, R.
A New Model-based Framework for Testing Security of IoT Systems in Smart Cities using Attack Trees and Price Timed Automata.
DOI: 10.5220/0007830605700577
In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019)*, pages 570-577
ISBN: 978-989-758-375-9

succession of destructive payloads on workstations, servers, and embedded devices.

In the future, it will be more difficult to recover from breakdowns caused by attacks, and interruptions may be measured in days instead of hours. Events of this type force providers in charge of these infrastructures to think about how to act in the event of an attack.

In this context our goal in this work is to propose a new framework for testing security aspects for IoT systems deployed in smart cities in order to avoid such anomalies or at least to minimize them as much as possible. Our framework is model-based (Krichen et al., 2018a) in the sense that it is based on the use of a formal approach which consists is generating test cases from a given model.

In our case, the considered model corresponds to the behavior of the attacker aiming to violate the system under investigation. First the behavior of the attacker is given as an *attack-tree* (Kordy et al., 2014). The latter is a formalism used to describe the strategy adopted by the attacker to achieve its goal.

In a second step the proposed tree is translated into an equivalent network of *price timed automaton* (Behrmann et al., 2005) which is a graphical representation which is used to model timed behaviors. The obtained network of price timed automaton serves as an input for the test generation algorithm.

The generated abstract test cases are then translated into the specification and execution standard testing language TTCN-3 (Lahami et al., 2012a). Finally a cloud-oriented test architecture (Tilley and Parveen, 2012) is proposed to run the obtained TTCN-3 concrete tests and to collect the generated verdicts.

The rest of this paper is organized as follows. Section 2 introduces the notion of attack trees. Section 3 defines the model of price timed automata. Section 4 illustrates how attack trees are translated into networks of price timed automata. Section 5 presents an overview about the test generation and execution phases. Section 6 reports on related research works dealing with IoT security testing. Finally Section 7 summarizes the main contributions of the paper and gives some possible directions for future work.

2 ATTACK TREES

Attack trees (AT) (Kordy et al., 2014) correspond to a useful graphical formalism to study the security of critical systems. More precisely an attack tree can be seen as a graphical representation of the attacker strategy represented in the form of a tree.

The root of an AT corresponds to the goal the attacker aims to fulfill. The children of a node in the AT are refinements of the goal of the corresponding parent node into sub-goals.

The refinement of an internal node of an AT can be either conjunctive or disjunctive:

- A conjunctive refinement is used when the fulfilment of all the children's goals is needed to fulfill the parent's goal. In this case we associate an AND-Gate with considered parent node (See Figure 1- (a)).
- A disjunctive refinement is used when the fulfilment of one of the children's goals is enough to fulfill the parent's goal. In this second case an OR-gate is associated with the considered node (See Figure 1-(b)).

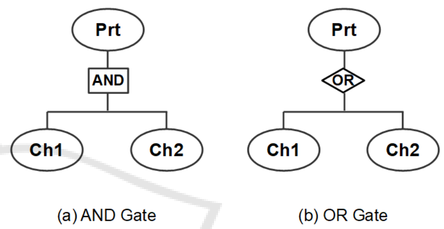

Figure 1: Different possible gates of an attack tree.

We consider a finite set of attribute variables $Attribs = \{Att_1, \cdots, Att_n\}$. These attributes are used to describe the characteristics of the attacker like available resources for instance. We denote by $Vals \subseteq (\mathbb{R}_{\geq 0})^n$ the set of valuations of the attributes.

The leaves of the AT correspond to the elementary actions the attacker has to execute. They are called *basic attack steps* (BAS). Each BAS is equipped with an additional attribute *Time* (which measures the time since the basic attack step started) and two preconditions:

$$Ready2Start : Vals \rightarrow \{0,1\}$$

which indicates that the BAS can be started or not (i.e., has enough resources to start for example); and

$$Able2Succeed : Vals \rightarrow \{0,1\}$$

which indicates whether the BAS can succeed or not. These preconditions are Boolean combinations of linear equations over *Attributes*. Moreover each BAS has an update function:

$$Modify : Vals \times \mathbb{R}_{\geq 0} \rightarrow Vals$$

which updates the attribute values when time elapses. At this level we assume that time dependence is linear between the attributes $\{Att_1, \cdots, Att_n\}$ and the special attribute *Time*.

Let $PrePost(Attribs)$ be the set of all possible triples $(Ready2Start, Able2Succeed, Modify)$ defined with respect to $Attribs$. We also define an Attacker Initializer which gives the initial valuation of the attributes. It is defined as:

$$Init : \{Att_1, \cdots, Att_n\} \to \mathbb{R}_{\geq} 0.$$

The set of attack tree gate types is defined as:

$$Gates = \{AND, OR\}.$$

An attack tree A is formally defined as a tuple $\langle Nds, Chld, Rt, Attribs, Init, Ftrs \rangle$ where:

- Nds is a finite set of attacker nodes;
- $Chld : Nds \to Nds^*$ associates a set of children to each parent node;
- Rt corresponds to the root of the AT A which defines the global goal of the attacker;
- $Attribs$ corresponds to the set of attributes of the AT A;
- $Init$ corresponds to the attacker initializer;
- $Ftrs : Nds \to Gates \cup PrePost(Attribs)$ associates an AND/OR Gate with each internal node and a tuple $(Ready2Start, Able2Succeed, Modify)$ with each leaf of the AT.

An example of an AT is given in Figure 2. This AT is inspired from the work of (Kumar et al., 2015). The goal of the attacker here is to crack the password of a protected file. As indicated by the figure the global goal of the attacker can be achieved by:

- Either cracking the password: this sub-goal can in turn be achieved using one of three possible choices (namely: Dictionary, Guessing or Brute Force attacks).
- Or performing a password attack: this sub-goal can be either achieved by a Social Engineering or a Key Logger attacks. The Social Engineering attack is in turn decomposed into two BASs namely: Generic Reconnaissance and Trap Execution. Similarly the Key logger attack is achieved within two BAS: Key Logger Installation and Password Intercept.

More details about this example can be found in the previously mentioned article (Kumar et al., 2015).

3 PRICED TIMED AUTOMATA

The model of Priced timed automata (PTA) (Behrmann et al., 2005) is an extension of timed automata, obtained by assigning costs to actions and locations. Next, we will denote by $\Psi(Y)$ the set of all possible Boolean predicates over a set Y of continuous variables.

A priced timed automaton P is defined as a tuple $\langle Loc, loc_0, Cl, Act, Edg, Inv, Cost \rangle$ where:

- Loc is a finite set of states;
- $loc_0 \in L$ is the initial state;
- Cl is a finite set of clocks;
- Act is finite a set of labels;
- $Edg \subseteq Loc \times \Psi(Cl) \times Act \times 2^{Cl} \times Loc$ gives the set of transitions;
- $Inv : Loc \to \Psi(Cl)$ assigns invariants to locations;
- $Cost : Loc \cup Edg \to \mathbb{N}^n_{\geq 0}$. assigns cost rates to states and costs to edges.

An edge $\langle loc, \psi, act, \lambda, loc' \rangle \in Edg$ defines a transition from location loc to location loc' taking an action act. This edge can only be traversed when the constraint ψ over Cl is true, and the set $\lambda \subseteq Cl$ identifies the subset of clocks which must be reset after the execution of the transition.

A trace of $P = \langle Loc, loc_0, Cl, Act, Edg, Inv, Cost \rangle$ is a sequence of locations and transitions $TR = loc_0 \xrightarrow[\lambda_0]{act_0} {}^{t_0}_{c_0}\, loc_1 \xrightarrow[\lambda_1]{act_1} {}^{t_1}_{c_1}\, loc_2 \cdots$ where:

- For every i, there is a transition $T_i = (loc_i, \psi_i, act_i, \lambda_i, loc_{i+1}) \in E$;
- For every i, $c_i = C(T_i) + t_i \cdot C(l_i)$ is the cost incurred in the transition;
- The initial valuation $V_0 = \overrightarrow{0}$ which assigns 0 to every clock in Cl;
- After each transition, there is a new clock valuation $V_{i+1} = (V_i + t_i)[\lambda_i = 0]$ obtained by increasing every clock in X_i by t_i and re-initializing all clocks in λ_i to 0;
- Each valuation $V_i + t$ for $t < t_i$ must satisfy the invariant $Inv(l_i)$;
- The valuation $V_i + t_i$ must satisfy ψ_i for every i.

Let $\|$ be the parallel product operator over price timed automata. That is given a set of PTAs $\{P_1, P_2, \cdots P_n\}$, $P_1 \| P_2 \| \cdots \| P_n$ will denote the corresponding parallel product obtained by synchronizing the transitions of the component PTAs via joint signals. The formal definition of this operator is given in (Bengtsson and Yi, 2004).

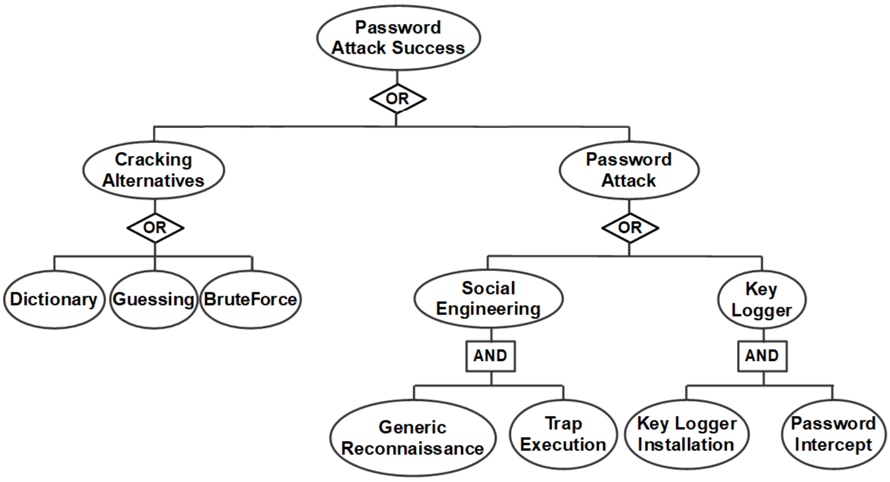

Figure 2: An example of an attacker tree inspired by the work of (Kumar et al., 2015).

4 FROM ATTACK TREES TO PRICE TIMED AUTOMATA

In this section we explain how an attack tree is transformed into a network of price timed automata. The proposed transformation is borrowed from the work of (Kumar et al., 2015).

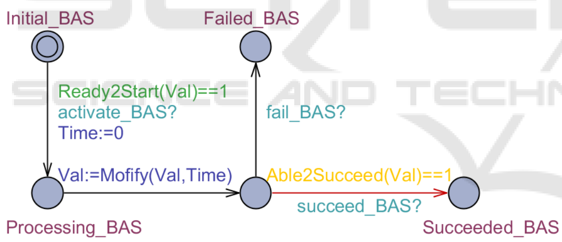

Figure 3: A priced timed automaton for a basic attack step.

First in Figure 3 we draw the price timed automaton corresponding to a basic attack step. The proposed PTA has five nodes. The considered BAS is activated when the input-signal *activate_BAS*? is received from the corresponding parent-node PTA. In order to execute this input-signal the condition $Ready2Start(Val) == 1$ must hold. The clock variable *Time* is reset to zero as soon as the BAS is activated. The attributes of the attacker are updated through the transition labeled with $Val := Mofify(Val,Time)$. At the end of the execution of the BAS the PTA reaches either state *Succeeded_BAS* or *Failed_BAS*. In order to reach the state *Succeeded_BAS* the condition $Able2Succeed(Val) == 1$ must hold.

In Figure 4, we propose a PTA which corresponds to a parent node connected to two children via an

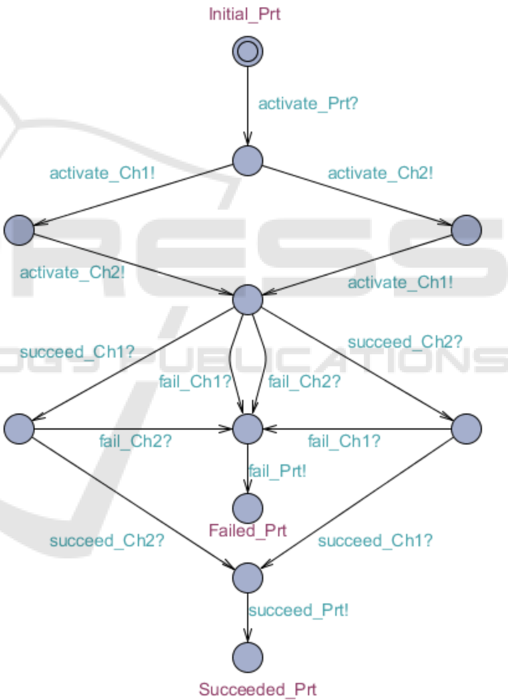

Figure 4: A priced timed automaton for an AND gate and a parent node having two children.

AND gate. This PTA is activated after receiving the input-signal *activate_Prt*?. After that an activation output-signal is sent to each child PTA. If a success signal is received from both children then the parent PTA moves to its success state.

Similarly Figure 5 is an illustration of the PTA corresponding to a parent node connected to two children via an OR gate. In this case receiving a success signal from one of the two children is enough to guar-

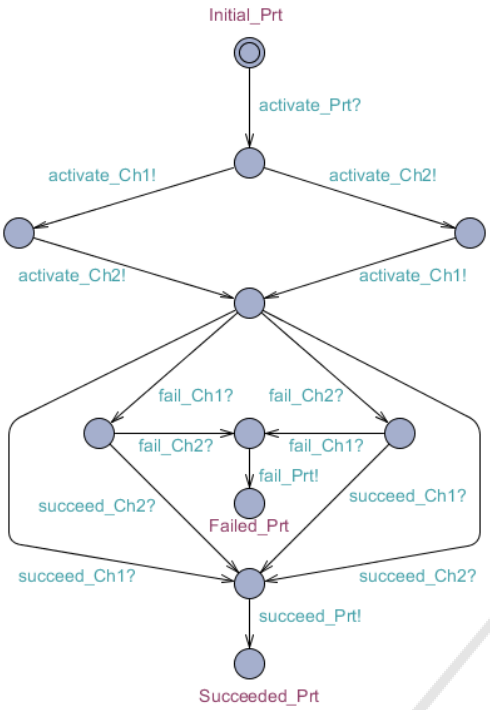

Figure 5: A priced timed automaton for an OR gate and a parent node having two children.

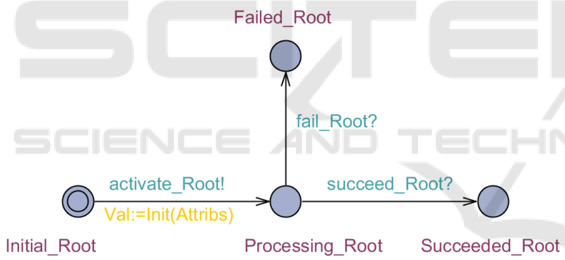

Figure 6: Priced timed automaton corresponding to the global goal of the attacker.

antee the success of the parent PTA.[1]

Finally Figure 6 gives a PTA which corresponds to the execution of the global goal of the attacker. The output signal *activate_Root*! will be the first action to be executed by the network of obtained PTAs.

5 TEST GENERATION AND EXECUTION

Test generation consists in extracting abstract test cases from the obtained network of PTAs. For this purpose we may use UPPAAL CORA (Behrmann et al., 2005; Rasmussen et al., 2004) which is an extension of the platform UPPAAL. This extension is enriched with additional variables used for optimal reachability analysis.

As already mentioned the proposed framework in this work is based on the TTCN-3 standard (ETSI, 2015). For this purpose, we will take advantage from the work of (Lahami et al., 2016; Lahami et al., 2012a). Next we give a brief recall about the main constituents of the TTCN-3 reference architecture as illustrated in Figure 7:

- **Test Management (TM):** manages the whole test process by starting and stopping tests;
- **Test Logging (TL):** manages all log events;
- **TTCN-3 Executable (TE):** runs the compiled TTCN-3 code;
- **Component Handling (CH):** places parallel test components and guarantees communication between them;
- **Coding and Decoding (CD):** encodes and decodes received from and sent to the TE;
- **System Adapter (SA):** adjusts the communication with the application or system under test;
- **Platform Adapter (PA):** implements the set of external functions.

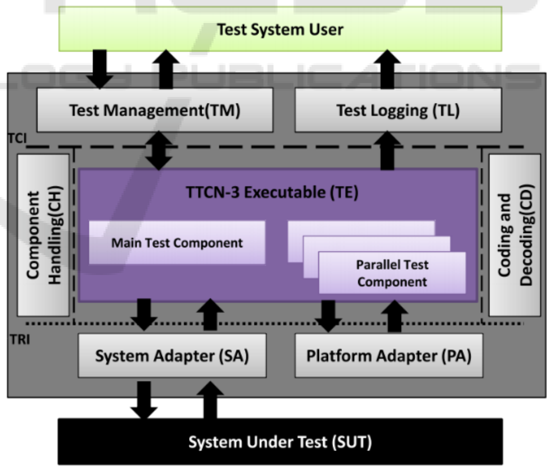

Figure 7: TTCN-3 Architecture (Lahami et al., 2016).

At this level we are interested in defining a set of rules for transforming abstract test cases into concrete TTCN-3 tests.

The adopted transformation algorithm may be inspired by the following works (Axel Rennoch and Schieferdecker, 2016; Hochberger and Liskowsky, 2006; Ebner, 2004). Table 1 gives some examples of the rules to use to derive TTCN-3 tests from abstract test cases. These rules are concisely explained below:

[1]It is worth noting that the two previous cases can be easily extended to the situation where a parent node has more than two children.

- **R1:** This rule generates a new TTCN-3 module for each abstract test suite;
- **R2:** This rule transforms each test sequence into a TTCN-3 test case;
- **R3:** This rule associates a TTCN-3 timer with each abstract timed behavior;
- **R4:** This rule transforms each test sequence into a TTCN-3 function;
- **R5:** This rule transforms the abstract channels into TTCN-3 templates.

Table 1: TTCN-3 Transformation Rules.

R#	Abstract Concepts	TTCN-3 Concepts
R1	Test Suite	TTCN-3 Module
R2	Single Trace	TTCN-3 Test Case
R3	Timed Behavior	TTCN-3 Timer
R4	Test Sequence	TTCN-3 Function
R5	Channel	TTCN-3 Template

Cloud computing can be used in the field of software testing to deal with the problem of lack of resources and the considerable cost of building a distributed test solution during the testing activity. Consequently, the notion of *Cloud testing* is increasingly emerging in order to offer cost-effective and efficient testing facilities. As defined by (Gao et al., 2011), it corresponds to testing activities (namely test case generation, test case execution and test result evaluation) on a cloud-oriented environment.

The proposed cloud testing architecture is built based on TaaS (Testing as a Service) concepts. Figure 8 outlines an overview of its different components of this architecture.

- **Test Management GUI:** offers a GUI (Graphical User Interface) charged with manages the whole testing process.
- **Resource Management:** enables flexibility and elasticity during the testing process.
- **Test Component Management:** offers services which create/delete test components and start/stop their execution.
- **Runtime Monitoring:** gives the status of the resources of each VM (e.g., memory, CPU, etc.).

6 RELATED WORK

Authors of (Felderer et al., 2016) proposed an interesting survey on dozens of articles related to model-based security testing chosen from the most relevant digital sources and classified with respect to specific criteria. However this review did not cover any work dealing security issues for IoT and smart cities. In the opposite way, the authors of (Ahmad et al., 2016) presented a model-based approach to test IoT systems but they did not consider security aspects in anyway. Moreover the authors of (Wang et al., 2017) proposed a formal framework based on timed automata for analyzing security properties of cyber-physical systems. In (Krichen et al., 2018a), the authors proposed a preliminary work which introduced a model based approach for testing security aspects of IoT systems in smart cities. Regarding the use of attack trees we mention the following works (Aslanyan et al., 2016) (Kammüller et al., 2016) (Kumar et al., 2015) (Kordy et al., 2014) which adopted this formalism to model and analyse security attacks. However none of these works has attempted to use testing techniques to check the ability of the considered systems to defend themselves against security attacks.

7 CONCLUSION

In this work we proposed a new approach for testing security aspects for IoT systems in Smart Cities. The proposed approach is based on the use of attack trees which correspond to a graphical representation of the strategy adopted by an attacker in order to violate the IoT system. We proposed a transformation method to translate a given attack tree into a network of price timed automata. The latter is then used as input for the test generation algorithm for producing abstract test cases. The obtained test cases are translated into concrete TTCN-3 test scenarios. Finally a cloud oriented testing architecture is proposed in order to execute tests and collect testing results.

Many extensions are possible for this work. First we may attack advantage from the work of (Lahami et al., 2016; Krichen, 2012; Lahami et al., 2012b; Krichen and Tripakis, 2006) to build a decentralized testing architecture. Moreover we may adopt the methodology proposed in (Krichen et al., 2018b; Maâlej and Krichen, 2016; Maâlej et al., 2013; Maâlej et al., 2012b; Maâlej et al., 2012a) to combine security and load tests for IoT applications. Finally we may exploit the same techniques presented in (Bensalem et al., 2007) in order to refine abstract test cases before translating them into TTCN-3.

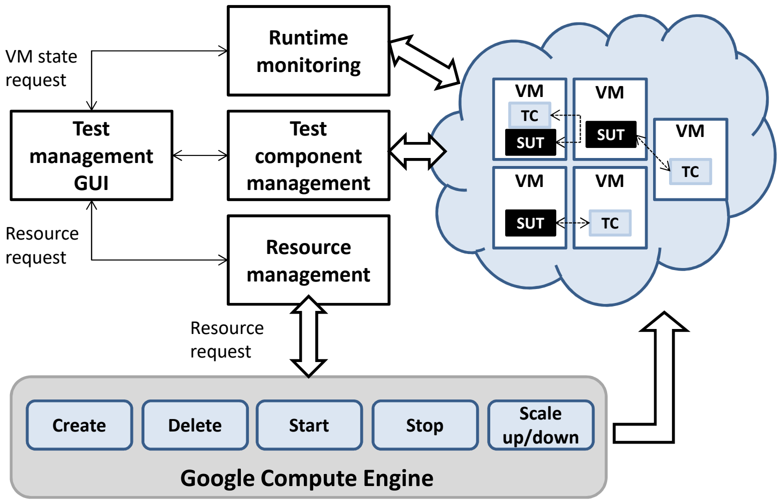

Figure 8: Cloud Testing Architecture Overview.

REFERENCES

Ahmad, A., Bouquet, F., Fourneret, E., Le Gall, F., and Legeard, B. (2016). Model-based testing as a service for iot platforms. In Margaria, T. and Steffen, B., editors, *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*, pages 727–742, Cham. Springer International Publishing.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., and Zhou, Y. (2017). Understanding the mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1093–1110, Vancouver, BC. USENIX Association.

Aslanyan, Z., Nielson, F., and Parker, D. (2016). Quantitative verification and synthesis of attack-defence scenarios. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 105–119.

Axel Rennoch, Claude Desroches, T. V. and Schieferdecker, I. (2016). TTCN-3 Quick Reference Card.

Behrmann, G., Larsen, K. G., and Rasmussen, J. I. (2005). Priced timed automata: Algorithms and applications. In de Boer, F. S., Bonsangue, M. M., Graf, S., and de Roever, W.-P., editors, *Formal Methods for Components and Objects*, pages 162–182, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bengtsson, J. and Yi, W. (2004). *Timed Automata: Semantics, Algorithms and Tools*, pages 87–124. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bensalem, S., Krichen, M., Majdoub, L., Robbana, R., and Tripakis, S. (2007). A simplified approach for testing real-time systems based on action refinement. In *ISoLA 2007, Workshop On Leveraging Applications of Formal Methods, Verification and Validation, Poitiers-Futuroscope, France, December 12-14, 2007*, pages 191–202.

Boyte, K. J. (2017). A comparative analysis of the cyberattacks against estonia, the united states, and ukraine: Exemplifying the evolution of internet-supported warfare. *Int. J. Cyber Warf. Terror.*, 7(2):54–69.

Ebner, M. (2004). TTCN-3 Test Case Generation from Message Sequence Charts. In *Proceeding of the Workshop on Integrated-reliability with Telecommunications and UML Languages (WITUL'04)*.

ETSI (2015). Methods for Testing and Specification (MTS), The Testing and Test Control Notation version 3, TTCN-3 Language Extensions: TTCN-3 Performance and Real Time Testing.

Felderer, M., Zech, P., Breu, R., Büchler, M., and Pretschner, A. (2016). Model-based security testing: A taxonomy and systematic classification. *Softw. Test. Verif. Reliab.*, 26(2):119–148.

Gao, J., Bai, X., and Tsai, W.-T. (September 2011). Cloud testing- issues, challenges, needs and practice. *Software Engineering : An International Journal (SEIJ)*.

Hochberger, C. and Liskowsky, R., editors (2006). *Informatik 2006 - Informatik für Menschen, Band 2, Beiträge der 36. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 2.-6. Oktober 2006 in Dresden*, volume 94 of *LNI*. GI.

Kammüller, F., Nurse, J. R. C., and Probst, C. W. (2016). Attack tree analysis for insider threats on the iot using isabelle. In Tryfonas, T., editor, *Human Aspects of Information Security, Privacy, and Trust*, pages 234–246, Cham. Springer International Publishing.

Kordy, B., Piètre-Cambacédès, L., and Schweitzer, P. (2014). Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13-14:1 – 38.

Krichen, M. (2012). A formal framework for black-box conformance testing of distributed real-time systems. *IJCCBS*, 3(1/2):26–43.

Krichen, M., Cheikhrouhou, O., Lahami, M., Alroobaea, R., and Jmal Maâlej, A. (2018a). Towards a model-based testing framework for the security of internet of things for smart city applications. In Mehmood, R., Bhaduri, B., Katib, I., and Chlamtac, I., editors, *Smart Societies, Infrastructure, Technologies and Applications*, pages 360–365, Cham. Springer International Publishing.

Krichen, M., Maâlej, A. J., and Lahami, M. (2018b). A model-based approach to combine conformance and load tests: an ehealth case study. *International Journal of Critical Computer-Based Systems*, 8(3-4):282–310.

Krichen, M. and Tripakis, S. (2006). Interesting properties of the real-time conformance relation. In *Theoretical Aspects of Computing - ICTAC 2006, Third International Colloquium, Tunis, Tunisia, November 20-24, 2006, Proceedings*, pages 317–331.

Kumar, R., Ruijters, E., and Stoelinga, M. (2015). Quantitative attack tree analysis via priced timed automata. In Sankaranarayanan, S. and Vicario, E., editors, *Formal Modeling and Analysis of Timed Systems*, pages 156–171, Cham. Springer International Publishing.

Lahami, M., Fakhfakh, F., Krichen, M., and Jmaïel, M. (2012a). Towards a TTCN-3 Test System for Runtime Testing of Adaptable and Distributed Systems. In *Proceedings of the 24th IFIP WG 6.1 International Conference Testing Software and Systems (ICTSS'12)*, pages 71–86.

Lahami, M., Krichen, M., Bouchakwa, M., and Jmaiel, M. (2012b). Using knapsack problem model to design a resource aware test architecture for adaptable and distributed systems. In *Testing Software and Systems - 24th IFIP WG 6.1 International Conference, ICTSS 2012, Aalborg, Denmark, November 19-21, 2012. Proceedings*, pages 103–118.

Lahami, M., Krichen, M., and Jmaïel, M. (2016). Safe and Efficient Runtime Testing Framework Applied in Dynamic and Distributed Systems. *Science of Computer Programming (SCP)*, 122(C):1–28.

Maâlej, A. J., Hamza, M., Krichen, M., and Jmaiel, M. (2013). Automated significant load testing for WS-BPEL compositions. In *Sixth IEEE International Conference on Software Testing, Verification and Validation, ICST 2013 Workshops Proceedings, Luxembourg, Luxembourg, March 18-22, 2013*, pages 144–153.

Maâlej, A. J. and Krichen, M. (2016). 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, October 6-7, 2016.*, pages 123–140.

Maâlej, A. J., Krichen, M., and Jmaiel, M. (2012a). Conformance testing of WS-BPEL compositions under various load conditions. In *36th Annual IEEE Computer Software and Applications Conference, COMPSAC 2012, Izmir, Turkey, July 16-20, 2012*, page 371.

Maâlej, A. J., Krichen, M., and Jmaiel, M. (2012b). Model-based conformance testing of WS-BPEL compositions. In *36th Annual IEEE Computer Software and Applications Conference Workshops, COMPSAC 2012, Izmir, Turkey, July 16-20, 2012*, pages 452–457.

Rasmussen, J. I., Larsen, K. G., and Subramani, K. (2004). Resource-optimal scheduling using priced timed automata. In Jensen, K. and Podelski, A., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 220–235, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tilley, S. and Parveen, T. (2012). *Software Testing in the Cloud: Perspectives on an Emerging Discipline*. IGI Global, Hershey, PA, USA, 1st edition.

Wang, T., Su, Q., and Chen, T. (2017). Formal analysis of security properties of cyber-physical system based on timed automata. In *Second IEEE International Conference on Data Science in Cyberspace, DSC 2017, Shenzhen, China, June 26-29, 2017*, pages 534–540.