Predictive Simulation for Building Trust Within Service-Based Ecosystems

Emilia Cioroaica
Fraunhofer IESE
Kaiserslautern, Germany
emilia.cioroaica@iese.fraunhofer.de

Said Daoudagh
ISTI-CNR
Pisa, Italy
said.daoudagh@isti.cnr.it

Eda Marchetti
ISTI-CNR
Pisa, Italy
eda.marchetti@isti.cnr.it

***Abstract*—Modern vehicles extend their system components outside the typical physical body, relying on functionalities provided by off-board resources within complex digital ecosystems. Focusing on the service-based connection within automotive smart ecosystems, in this paper we present the method of predictive simulation, based on the synergistic combination of Digital Twin execution and interface-based testing approaches, used for building trust in the interactions between a safety critical system and third parties.**

***Index Terms*—Automotive, Building Trust, Digital Twin, Interface Testing, Malicious Behavior, Virtual Evaluation, Smart Ecosystems**

I. Introduction

In the process of building innovative products and services outside traditional organizational boundaries, organizations around the world are transitioning towards joining complex software ecosystem so as to support value added interactions between actors around a common platform [1], possibly accounting for multiple organizations, external developers and users [2]. When joining an ecosystem, various actors such as developers, organizations, and users can create together innovative businesses. In an ecosystem, actors have collaborative and competitive goals and they target each other according to their goals. In this way, competitive and collaborative relationships between actors in an ecosystem are formed. In context of autonomous vehicles, actors' interaction is realized through provision of software components.

In particular, in the automotive domain emerging new technologies enable traditional vehicle manufacturers to keep up with the speed of users' needs through provision of off-board services that connect to on-board systems within a vehicle. An example is the intelligent key [3], which is an off-board device that communicates with the vehicle for opening its doors and activating the internal network.

For ruling and harmonizing the developing of these new services, that many times include vehicle's access via web services, a series of standards such as ISO 20077 [4] and ISO 20078 [5] are providing specific guidelines, design methodologies and interfaces. For instance, according to ISO/DIS 20077-1:2016, an extended vehicle includes all technical components that enable a vehicle's function, i.e., a task, an action or an activity that must be achieved in order to satisfy a functional requirement. This include also the on-board and off-board data and system required to perform this function.

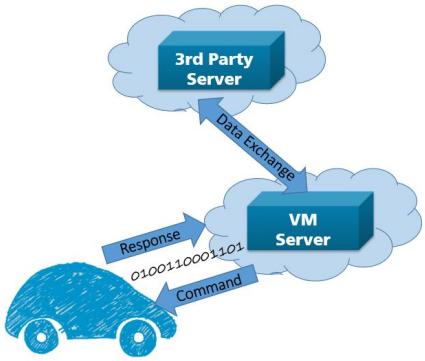

Fig. 1. Could-based connection to vehicle

A typical extended cloud-based vehicle is schematized in Fig. 1, that includes: the physical interfaces of the road vehicle, such as the OBD (On-board Diagnostic) Connector; the server at the manufacturer side (VM Server); the web services interplayed between the manufacturer's server and the other servers managed by service providers (3rd Party Server). Thus, within this ecosystem, software solutions used by a vehicle and hosted by third party providers in the cloud are made available to a variety of users that can connect to the corresponding cloud service via a multitude of devices. While opening the path towards development of multiple business applications, this scenario also raises multiple security constraints. An attacker is willing to take advantage of every technology that interacts with the outside world as a potential entry point to a system. Moreover, these cloud services are executed as black boxes and can be accessed by third-party applications which are themselves executed as black boxes.

Despite the guidelines and specifications provided by the reference standards, the complexity of these extended vehicles still requires the implementation of additional security measures and assessment procedures. This becomes even more stringent in case of communication-based extension and remote connections, because they open the path to specific cybersecurity threats and network, architectural vulnerabilities and risks [6]. In this paper we propose an approach for building

trust in the execution of black box service oriented approaches that connect to safety critical systems.

In what follows, Section II presents an overview of the state of the art and section III presents our methodology. Conclusions and future research agenda are then discussed in Section IV.

II. State of the Art

In this section we provide a general overview of the methods used for assuring safety of systems that partly operate in the cloud, emphasizing the standard requirements for automotive systems together with an overview of current tools and practices used on testing REST APIs by emphasizing the distinction from our approach. We focus our tool analysis on REST APIs, due to the fact that technically, the provision of cloud services capable of storing software applications (Software-as-a Service), data processing (Platform-as-a-Service) or the computing infrastructure (Infrastructure-as-a-Service) interact with other services via these interfaces.

A. Safety for Cloud-Based Application

In the process of evaluating safety of applications in cloud-based scenario [7], the criticality of the situation is typically given by the application. In a systematic top down approach, the demands of an application running in the cloud are carefully evaluated and critical features are identified. Then, based on analyzed requirements and vertical safety interfaces [8], it is evaluated how failures propagate vertically and what can be guaranteed in terms of safety, and implicit trust.

In this process, traditional methods start with a vertical interface and for each service make a claim is made followed by an assurance case for the service.

B. Testing Web Interfaces

Since their first introduction, web services have become an interesting challenge for testing activity due to their complex nature and the absence of source code. Different approaches have been developed that focus on testing a composition of services are investigated in [9], coverage of the data flow or control flow information [10], enhancing black-box testing [11]–[13], fuzzy testing approaches as described in [14], and penetration testing [15]. Different than other approaches our approach: i) is executed during runtime in a virtual predictive environment; ii) it aims at detecting malicious behaviors through the execution of Digital Twin (DT) in a simulated environment while the real system is running.

C. The Use of Digital Twins

Among the solutions available for increasing the level of trust in the extended vehicles part of digital ecosystems, those relying on Digital Twin (DT) [16] are currently catching on. Here, the predictive simulation approach deployed on a vehicle enables its safe reconfiguration by triggering a safe fail-over behavior [17]. In this paper we uplift the concept of predictive simulation in context of service runtime dynamic evaluation, where an invalid sequence discovered by the predictive simulation can be followed by the real world execution of services.

III. Methodology and Concept

Building the ecosystem trust within an extended vehicle is typically the outcome of testing activities performed in a testing environment during design time. In this paper we move forward the trust evaluation process towards runtime, by exploiting the simultaneous DT execution in a virtual environment. In this case the DT is stimulated by a crafted sequences of requests, generated by a testing approach. In this manner the DT can preliminary and timely provide expected answers that can be exploited for evaluating possible services' vulnerabilities and security risks, enabling a runtime virtual environment to predict through DT execution the hidden faults of service and calls enabling a prompt, timely reaction based on activation of specific countermeasures.

As depicted in Fig. 2, a DT of a service in an ecosystem is the machine-readable representation of the software/component and therefore can be exploited for predicting the service behavior under specific conditions such as attacks. Multiple DTs communicate with each other to enable the creation of a holistic scenarios and situations in which the outcome of a decision is evaluated.

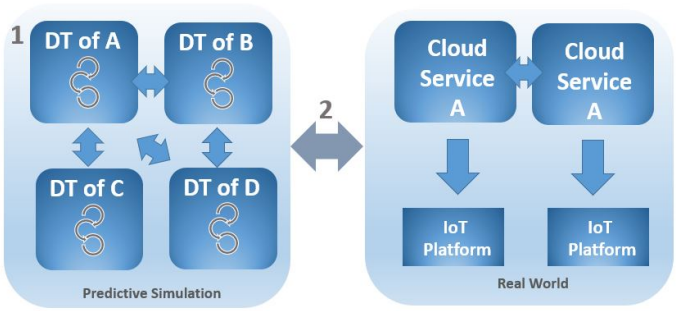

Fig. 2. Virtual evaluation through digital twin execution

Further on a synergistic combination of DT and interface-based testing approaches is exploiting the power of web based testing generation approaches for stimulating the DT execution in several conditions and with different data sets. Through the support of a simulated environment, the DT can therefore predict malicious interaction and possible vulnerability threats before they occur in the real-time execution. This will enable a safeguarding of a trusted system execution and implicit trusted evolution of safety-critical ecosystems that comprise of vehicles extended with 3rd party services that communicate via web services executed as black boxes.

In order to build up trust in the runtime service interaction, we propose a methodology of virtual evaluation based on digital twin execution of the interacting services in three logical steps: 1) Predictive simulation set-up; 2) Test case generation and DT evaluation; 3) Run time execution.

A. Predictive Simulation Set-up

1) *Digital Twin Provision.* A cloud service is deployed within a digital ecosystem together with its corresponding DT. The DT of a service is a subset of its Swagger

specification and specifies the list of requests that can be handled by the service and the responses the service can provide. The DT is an executable description of the service interface that can be executed in a simulated environment.

2) *Provision of Design Time Evidence of Trust.* The Digital Twin is being deployed together with an *Interface of Trust*. The interface of trust contain design time evidence of tested behavior and provides evidence of trust for skipping runtime prediction.

B. Test Case Generation and DT Evaluation

The trustworthiness of an interacting service is evaluated by exercising the corresponding DT behavior within the predictive simulation environment. For a sequence of requests describing an evaluation scenario, the execution of the DT provides a projection of the service behavior in interaction with other services within an ecosystem. In this way, the process of building trust in the dynamic interaction between services does not require execution of the services that can potentially be subject of an attack, but merely evaluation of the behavior of its DT in a secured virtual environment.

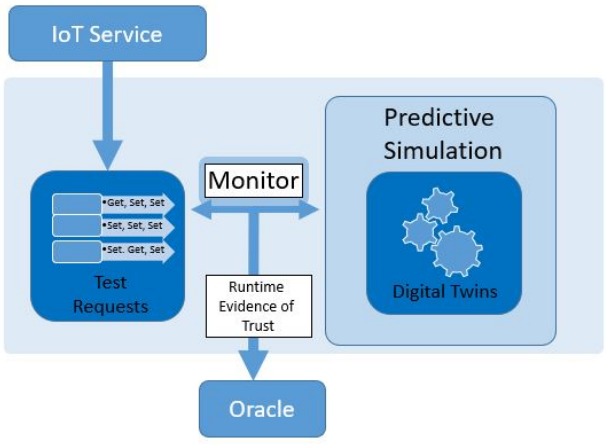

Fig. 3. Rest based testing scenario.

As described in Fig. 3, the process includes the following activities.

1) *Generation of Test Cases.* For this different approaches like combinatorial testing (such as the pivotal proposal of [18]) or fuzzy testing methodologies (such as [14]) can be considered. The intent is to exercise, by the generation of service requests and responses, situations in which violations or malicious behavior can be experienced. For this a lightweight static analysis of the entire Swagger specification can be exploited for the generation of tests that exercise the corresponding cloud services through its REST API.
2) *Test Cases Execution.* The generated tests that exercise the corresponding cloud services are executed through its REST API provided by the corresponding DT.
3) *Oracle Derivation.* The collected DT responses are compared against the interface of trust and in case of unforeseen hazardous situations it derives the runtime rules of trust that form an oracle. For the oracle definition, standards and specific properties are exploited for deriving its verdict.

C. Run Time Execution

During the run time execution a conformity monitoring between the execution of the invoking service and the resulting oracle is performed. This phase requires execution of the invoking service on a platform. Conformity is checked by validating the behavior of the DT against the behavior of the real service and it requires a trustworthy monitoring platform.

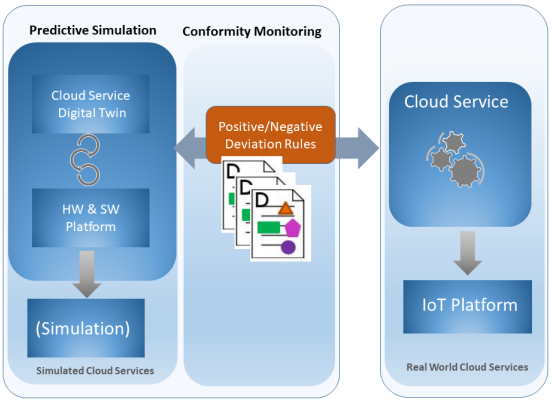

Fig. 4. Execution and Conformity Monitoring.

As depicted in Fig. 4, the service under evaluation is executed in relation to digital twins of interacting services. This process consists of multiple phases:

1) *Creation of Runtime Trusted Behavior Signature.* The execution of digital twins results into a sequence of requests and responses which are fed into an internal validation engine. The sequence of requests and responses generated by the digital twins execution in a predictive simulation are recorded as events. The order, the type and number of events then form the *service interaction valid signature* against which the real world execution of services is being monitored. The validation engine uses as reference for trusted behavior known sequences of requests and responses which have been previously validated during design time.
2) *Conformity Monitoring.* After passing the predictive simulation, the monitoring engine then compares predictive behavior with actual behavior.
Malicious behavior triggered through specific sequences are detected when, for example, the predictive simulation, supported by design time evidence invalidates a sequence of requests and responses, while the real world execution of services is executed fully. We refer to this type of deviation as a *positive deviation*. Intervention of the business owner is required in order to investigate the cause and nature of deviation. A *true positive deviation*

can be caused by the advancements of the services at the side of the organization, part of the ecosystem, whereas a *false positive deviation* can be caused by the malicious introduction or/and activation of an intended fault, referred in the literature as *logic bomb* [19].

The reverse case, when predictive simulation validates a sequence of responses and requests while the real world execution of services fails, can be caused by any other type of failures, not necessarily a malicious attack. A possible cause of failure, can, for example be: denial of service.

IV. Conclusions and Future Work

In this paper, we have presented an approach for building trust in the execution of 3rd party black box services regarded as an extension of safety-critical systems by elevating the concept of predictive simulation recently proposed for supporting the process of trust evaluation in the domain of embedded systems.

A. Limitations

Possible communication latencies when performing the predictive simulation form a considerable challenge in deploying our approach in real world systems. However, technological advancements towards fast transfer of data deployed in vehicles could support the implementation of our method. Deployment of the platform in this scope needs to be done on the vehicle. When no concerns of data privacy exists, cloud computing can at most be used in predictive simulation for scenarios where the response time is not crucial. This approach requires regular information transfer to the system when the smart agents run.

B. Future Research Directions

As part of future work is the definition of digital twins for web services as precisely formulated sequences of events that through dynamic execution enables a simulator the possibility to create predictable artefacts based on model execution. Further on, it is assumed that, if the execution of the web service is in accordance to its specifications, the system can be trusted to behave as expected in real-world situations as well. This is not necessarily true, as the control algorithm under evaluation can exhibit different behavior when it is executed in a virtual environment than when it is executed in the real environment, to mask potential malicious behavior. Development of a solution for this challenge requires further research on the DT definition of a service can be a subset of the Swagger specification that can provide as specification the list of requests that can be handled and the responses the service can provide.

Virtual evaluation is realized using virtual entities communicating with each others through virtual signals. Additionally, the behavioral model of each entity represents an abstraction of the system behavior towards the scope of the evaluation. These two aspects together with the fact that software components under evaluation can hide malicious behavior if they are able to detect when they are under evaluation drive the need of building trust in virtually evaluated software based systems. Consequently, further research can go in the direction of providing means to detect such kind of deception in order to ensure trust in the virtual evaluation.

Acknowledgment

This work has been funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 952702 (BIECO).

References

[1] K. Manikas and K. M. Hansen, "Software ecosystems - A systematic literature review," *J. Syst. Softw.*, vol. 86, no. 5, pp. 1294–1306, 2013.

[2] R. Capilla, E. Cioroaica, B. Buhnova, and J. Bosch, "On autonomous dynamic software ecosystems," *IEEE Transactions on Engineering Management*, pp. 1–15, 2021.

[3] K.-s. Kim, I.-s. Song, Y.-s. Lee, and S.-b. Choi, "The implementation of start stop system with the obd-ii interface in the automotive smart key system," *Multimedia Tools and Applications*, vol. 74, no. 20, pp. 8993–9005, 2015.

[4] "ISO 20077-1," https://www.iso.org/standard/66975.html, [Online; accessed 06-December-2021].

[5] "ISO 20078-1," https://www.iso.org/standard/66978.html, [Online; accessed 06-December-2021].

[6] D. Lyu, L. Xue, and X. L. Le Yu, "Remote attacks on vehicles by exploiting vulnerable telematics," 2016.

[7] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: Automated iot safety and security analysis," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 147–158.

[8] B. Zimmer, S. Bürklen, M. Knoop, J. Höfflinger, and M. Trapp, "Vertical safety interfaces–improving the efficiency of modular certification," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2011, pp. 29–42.

[9] D. Petrova-antonova, D. Manova, and S. Ilieva, "Testing Web Service Compositions : Approaches , Methodology and Automation," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 1, pp. 159–168, 2020.

[10] D. Corradini, A. Zampieri, M. Pasqua, and M. Ceccato, "Restats: A test coverage tool for restful apis," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021, pp. 594–598.

[11] E. Viglianisi, M. Dallago, and M. Ceccato, "Resttestgen: automated black-box testing of restful apis," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*. IEEE, 2020, pp. 142–152.

[12] C. Bartolini, A. Bertolino, S. Elbaum, and E. Marchetti, "Whitening soa testing," in *Proc. of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. New York, NY, USA: ACM, 2009, p. 161–170.

[13] A. Gómez, M. Iglesias-Urkia, A. Urbieta, and J. Cabot, "A model-based approach for developing event-driven architectures with asyncapi," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2020, pp. 121–131.

[14] V. Atlidakis, P. Godefroid, and M. Polishchuk, "Rest-ler: automatic intelligent rest api fuzzing," *arXiv preprint arXiv:1806.09739*, 2018.

[15] D. Garg and N. Bansal, "A systematic review on penetration testing," in *2021 2nd Global Conference for Advancement in Technology (GCAT)*, 2021, pp. 1–4.

[16] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Modeling, simulation, information technology & processing roadmap," *National Aeronautics and Space Administration*, 2012.

[17] E. Cioroaica, T. Kuhn, and B. Buhnova, "(do not) trust in ecosystems," in *IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2019, pp. 9–12.

[18] C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini, "Ws-taxi: A wsdl-based testing tool for web services," in *2009 International Conference on Software Testing Verification and Validation*, 2009, pp. 326–335.

[19] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.