Defining and Matching Test-Based Certificates in Open SOA

Marco Anisetti, Claudio A. Ardagna, Ernesto Damiani
Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
Crema (CR), 26013, Italy
Email: firstname.lastname@unimi.it

***Abstract*—Following the Service-Oriented Architecture (SOA) and the Cloud paradigms, an increasing number of organizations implement their business processes and applications via runtime composition of services made available on the cloud by single suppliers. This scenario however introduces new security risks and threats, as the service providers may not provide the level of assurance required by their customers. There is therefore the need of a new certification scheme for services that provides trusted evidence that a remote service has some security properties, and a matching infrastructure to compare service certificates with users' certification preferences. In this paper, we provide a first solution to the definition of a test-based certification process for SOA.**

I. Introduction

Service-Oriented Architecture (SOA) and Cloud computation paradigms provide the basis to integrate applications across a global ICT infrastructure, allowing remote users to access information and services supplied by service providers. Organizations increasingly implement their business processes via run-time (as opposed to design-time) selection and composition of such services [5], that communicate over the ICT infrastructure by using standard Web protocols and technology [3].

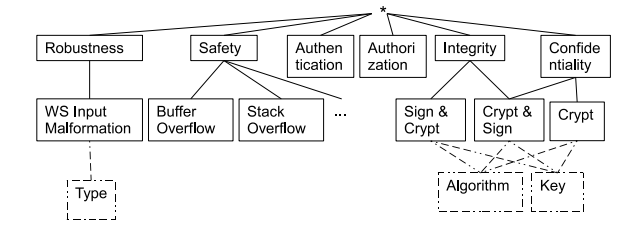
Such a flexible implementation of a business process in an open service ecosystem exposes applications to a number of new security risks and threats. This scenario increases users' concerns about the security of remote services, e.g. against penetration or Denial-of-Service attacks, as well as about the protection of the data disclosed to them, e.g. as input parameters. Pushing the SOA vision on an open ICT infrastructure requires careful re-thinking of current development, testing, and verification methodologies, and introduces the need of new assurance techniques that will increase the users' trust that services will satisfy their functional and non-functional requirements.

Certification can play a role to establish a trust model suitable for service ecosystems; but existing certification techniques and protocols (e.g., Common Criteria [4]) have been defined for traditional software components rather than services. Indeed existing certification schemes try and provide engineers in charge of software procurement with trusted evidence, based on testing and formal verification signed by a trusted third party, that a software product has some features, conforms to specified requirements, and behaves as expected [1]. In this position paper, we put forward the idea that the definition of a certification scheme that can be used at run-time to make trusted assurance information available in a service ecosystem [2]. Certification of services can in fact represent an important plus for open SOAs, allowing service users to evaluate different services and select the appropriate ones for service composition. In the remainder of the paper, we present the challenges and a first solution to the problem of test-based certification of services, including how test-based certificates can be automatically matched with users' preferences on the certification process.

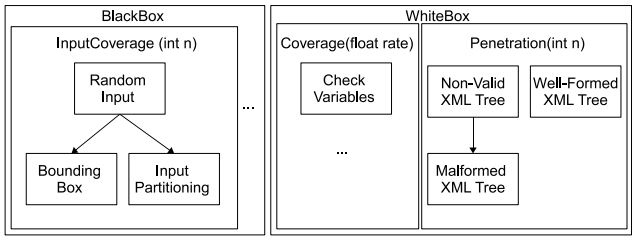
II. Basic concepts of test-based certification for SOA

According to Damiani et al. [1], "*test-based certificates are evidence-based proofs that a test carried out on the software has given a certain result, which in turn shows (perhaps with a certain level of uncertainty) that a given property holds for that software. In particular, test-based certification of security-related properties is a complex process, identifying a set of high-level security properties and linking them to a suitable set of white- and black-box software tests*". Starting from the above definition, a certification process for services should first define a machine-readable certificate format linking a set of properties with the evidence supporting them. To this aim, a service-oriented certification scheme defines a hierarchy $\mathcal{H}_P$ of security properties and the classes of tests that can be used to prove that some test has been carried out on a service and a given property holds.

Hierarchy $\mathcal{H}_P$ of security properties. Formally, a hierarchy $\mathcal{H}_P$ of security properties is a pair ($\mathcal{P}$,$\succeq_P$), where $\mathcal{P}$ is the set of all security properties, and $\succeq_P$ is a partial order relationship over $\mathcal{P}$. Given two properties p_i and p_j in $\mathcal{P}$, we say that $p_i \succeq_P p_j$ if p_i is an abstraction of p_j. For instance, (return parameter) *integrity* is an abstraction of properties *crypt&sign*, *sign&crypt* (i.e., *integrity*$\succeq_P$*crypt&sign*, *integrity*$\succeq_P$*sign&crypt*). This means that each request for a service that guarantees a property of *integrity* of its return parameter will also be satisfied by a service showing a certificate for a property that is dominated by *integrity*. Figure 1(a) shows an example of hierarchy of security properties. As shown in the figure, each security property (black squares) is also



(a)



(b)

Fig. 1. An example of a hierarchy of security properties (a) and classes of tests (b)

characterized by a set A of class attributes (dashed squares) that refer to a set of threats the service proves to counteract or to specific characteristics of the security function that is certified. Each attribute $a \in A$ is characterized by a total order relationship $\geq$. For instance, a service may expose a certificate proving that it supports a (return parameter) *confidentiality* property with a 3DES algorithm and a key of 112bits. Here, we use class attributes to distinguish between properties that are operationally different although share the same name. Class attributes also simplify the matching process between the certificate held by a service and the users' preferences. A user may in fact require a service proving *confidentiality* with 3DES algorithm and a key of at least 168bits.

Classes of tests. Each security property in $\mathcal{P}$ can be associated with zero or more test units used to certify it. Test units are organized in hierarchies. Formally, a hierarchy $\mathcal{H}_T$ of test units is a pair ($\mathcal{T}$,$\succeq_T$), where $\mathcal{T}$ is the set of all test units, and $\succeq_T$ is a partial order relationship over $\mathcal{T}$. Given two test units t_i and t_j in $\mathcal{T}$, $t_i \succeq_T t_j$ if t_i is an abstraction of t_j. Test units are then organized in classes of tests having a set *TA* of test attributes. Each test attribute $ta \in TA$ is characterized by a total order relationship $\geq$. Figure 1(b) shows an example of classes of tests together with hierarchies of test units. For instance, in Figure 1(b), the class *Penetration(int n)*, where n is a test attribute representing the cardinality of the test set, contains the test units *Non-Valid XML Tree*, *Well-Formed XML Tree*, and *Malformed XML Tree*. For instance, *Non-Valid XML Tree*$\succeq_T$*Malformed XML Tree*, while *Malformed XML Tree*$\not\succeq_T$*Well-Formed XML Tree*.

In our vision, each test-based certificate is composed by two main sections: *i)* a (set of) property and related class attributes (that we will call *security property* in the following); *ii)* a (set of) evidence signed by a third party proving that the service supports that property. An evidence is composed by a set of test units providing detailed specification of testing practices (environment, run-time context) and outcome, together with the set of test attributes associated with the classes the test units belong to.

III. Double-matching strategy

The traditional SOA paradigm consists of an infrastructure where services are searched and composed at run-time based on the users' preferences.Our service certification scheme should then be integrated within the existing SOA infrastructure and complement it by providing a mechanism where users define their preferences in terms of certified properties, evidence, and tests, and automatically match them against the certificates awarded to the services. Run-time certificate matching will then permit the users to evaluate if the assurance level provided by the service certificate is compatible with her own preferences.

In our scenario, we need a double-matching strategy which involves a check both on security properties (property-match) and on evidences (evidence-match) in the certificate. More in detail, let S be a service with a certificate $C\big((P_1,E_1)\ldots(P_n,E_n)\big)$, where P_i is a security property (p_i,A_i) with p_i a property in $\mathcal{P}$ and $A_i = \{a_{i,1},\ldots,a_{i,l}\}$ a set of class attributes, and E_i is an evidence (t_i,*TA*$_i$) with t_i a test unit in $\mathcal{T}$ and *TA*$_i = \{ta_{i,1},\ldots,ta_{i,k}\}$ a set of test attributes. Also, let $R\big((P'_1,E'_1)\ldots(P'_m,E'_m)\big)$ be the preferences of the user in the form of a service request, where P'_j is a security property (p'_j,A'_j) with p'_j the requested property in $\mathcal{P}$ and $A'_j = \{a'_{j,1},\ldots,a'_{j,s}\}$ a set of class attributes, and E'_j is an evidence ($t_j{}'$,*TA*$'_j$) with $t_j{}'$ a test unit in $\mathcal{T}$ and *TA*$'_j = \{ta'_{j,1},\ldots,ta'_{j,z}\}$ a set of test attributes. In the following, for the sake of simplicity, we will consider certificates and requests of the form C(P,E) and R(P',E'), respectively, where P=(p,A), E=(t,*TA*), P'=(p',A'), and E'=(t',*TA*$'$). The matching process between a certificate C and a request R, denoted as $C \times R$, is defined as follows.

Definition 3.1 (Matching process): Let C(P,E) be a certificate awarded to a service and R(P',E') be a request by a user. The matching process $C \times R$ is a process that first compares P and P' (property-match). If and only if this comparison succeed, the matching process compares E and E' (evidence-match). The matching process is successful if and only if both property-match and evidence-match succeed.

The matching process $C \times R$ can return two results as follows:

1) *match*, if and only if:
   - $p' \succeq_P p$, and $\forall a' \in A', \exists a \in A$ s.t $a \geq a'$ (property-match), and
   - $t' \succeq_T t$, and $\forall ta' \in TA', \exists ta \in TA$ s.t $ta \geq ta'$ (evidence-match).
2) *no match*, otherwise.

In the following we discuss the match/no match scenarios by means of two examples on the single case study shown

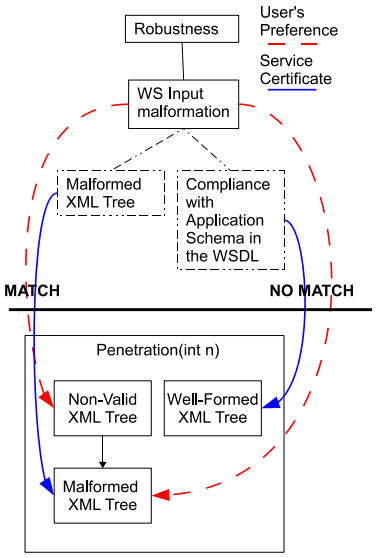


Fig. 2. An example of test-based certificate matching

in Figure 2. Figure 2 also shows two instance values for the class attribute `type`: *Malformed XML Tree* and *Compliance with the Application Schema in the WSDL*.

Example 1 (Match). The left part of Figure 2 presents an example of a successful matching between a service certificate (blue line) and the user's preferences in the form of a service request (red dashed line). In particular, let us consider an (enhanced) UDDI registry that contains a set of services together with their certificates.[1] Also, let us consider a service S in the registry that has a certificate C proving a security property (p,A)=(*Robustness.WS Input Malformation*,{`Type`=*Malformed XML Tree*}) with evidence (t,*TA*)=(*Penetration test using Malformed XML Tree*,{`n`=k}). Suppose now that a user submits a request R to the registry searching for a service that has a certificate proving a generic security property (p',A')=(*Robustness.WS Input Malformation*,{}) with evidence (t',*TA'*)=(*Penetration test using Non-Valid XML Tree*,{`n`=m}), with $k > m$. The registry searches among its set of services and selects all services that expose a certificate C with a security property (p_i,A_i) that matches with the security property (p',A') in R (property-match). Service S is considered for matching since $p' \succeq_P p$ based on the hierarchy in Figure 1(a)[2] and the set of class attributes is empty in R meaning that any combination of attributes in the certificate is acceptable. Among the selected services, the registry compares the evidence in the certificate with the evidence in the request (evidence-match). Considering service S, it is clear that the evidences match because $t' \succeq_T t$ and $k > m$. As a result, $C \times R$=*match*.

[1]Extensions to UDDI registry metadata including test outcomes have been proposed by several research groups [6].

[2]In the example, p and p' are the same security property. In general, p' may be an abstraction of p.

Example 2 (No-Match). The right part of Figure 2 presents an example of failed matching between service certificate (blue line) and user's preferences (red dashed line). Again, let us consider a UDDI registry that contains a set of services together with their certificates. Also, let us consider a service S that has a certificate C proving a security property (p,A)=(*Robustness.WS Input Malformation*,{`Type`=*Compliance with Application Schema in the WSDL*}) with evidence (t,*TA*)=(*Penetration test using Well-Formed XML Tree*,{`n`=k}). Suppose now, that a user is submitting a request R to the registry searching for a service that has a certificate proving a generic security property (p',A')=(*Robustness.WS Input Malformation*,{}) with evidence (t',*TA'*)=(*Penetration test using Malformed XML Tree*,{`n`=m}), with $k > m$. As in Example 1, service S is considered for matching since $p' \succeq_P p$ based on the hierarchy in Figure 1(a) and the set of class attributes is empty in R (property-match). However, in this example, there is no evidence-match because, although $k > m$, $t' \not\succeq_T t$. As a result, $C \times R$=*no-match*.

IV. Conclusions and Future Work

We presented some preliminary ideas on some basic mechanisms for integrating test-based certification in a SOA scenario. Our future work will focus on the definition of a machine-readable certificate format, the definition of complete hierarchy of security properties and classes of tests, the matching and comparison between different classes of tests and between non-quantitative or unordered test units (e.g., penetration tests involving different inputs), and the problem of certifying dynamically composed services, starting from the composition of the certificates of their basic components.

Acknowledgements

This work was partly founded by the European Commission under the project ASSERT4SOA (contract n. FP7-257351).

References

[1] E. Damiani, C.A. Ardagna, and N. El Ioini. *Open source systems security certification*. Springer, New York, NY, USA, 2009.

[2] E. Damiani and A. Maña. Toward ws-certificate. In *Proc. of the ACM Workshop on Secure Web Services (SWS 2009)*, Chicago, IL, USA, November 2009.

[3] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[4] D.S. Herrmann. *Using the Common Criteria for IT security evaluation*. Auerbach Publications, 2002.

[5] M.P. Papazoglou. Web services and business transactions. *World Wide Web*, 6(1):49–91, 2003.

[6] W.T. Tsai, R. Paul, Z. Cao, L. Yu, A. Saimi, and B. Xiao. Verification of Web services using an enhanced UDDI server. In *Proc. of the 8th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2003)*, Guadalajara, Mexico, January 2003.