RZ 3625 (# 99635) 08/15/2005
Computer Science 12 pages

Research Report

Service-oriented Assurance – Comprehensive Security by Explicit Assurances

Günter Karjoth, Birgit Pfitzmann, Matthias Schunter, Michael Waidner

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland
{gka,bpf,mts,wmi}@zurich.ibm.com

Service-oriented Assurance –
Comprehensive Security by Explicit Assurances

Günter Karjoth, Birgit Pfitzmann, Matthias Schunter, Michael Waidner

IBM Research, Zurich Research Laboratory
Säumerstrasse 4, 8803 Rüschlikon, Switzerland
{gka,bpf,mts,wmi}@zurich.ibm.com

September 5, 2005

Abstract. Flexibility to adapt to changing business needs is a core requirement of today's enterprises. This is addressed by decomposing business processes into services that can be provided by scalable service-oriented architectures. Service-oriented architectures enable requesters to dynamically discover and use sub-services. Today, service selection does not consider security. In this paper, we introduce the concept of Service-Oriented Assurance (SOAS), in which services articulate their offered security assurances as well as assess the security of their sub-services. Products and services with well-specified and verifiable assurances provide guarantees about their security properties. Consequently, SOAS enables discovery of sub-services with the "right" level of security. Applied to business installations, it enables enterprises to perform a well-founded security/price trade-off for the services used in their business processes.

1 Introduction

Enterprises struggle to increase their flexibility to adapt to changing business needs. Service-oriented architectures address this challenge by decomposing enterprises into loosely coupled services, which are hosted on platforms that can adapt to changing load and performance requirements. This trend is reflected by the growth of value networks, in which enterprises specialize on their core competencies and interconnect these critical services to provide a better overall service to their customers.

Whereas current research focuses on how to integrate the business processes of these value networks, security will be a major obstacle to their wide-spread adoption. Cross-enterprise security is still addressed by long-lasting trust relationships, contracts, and manual audits. Emerging service-oriented architectures and flexible usage patterns are slowly invalidating this static closed-world approach. There exists no approach that guarantees overall security while permitting the flexibility required today.

In this paper we propose a new concept called "Service-oriented Assurance (SOAS)" that enables providers to advertise their security, allows customers to monitor the actual security of a service, and provides well-defined recourse for violations of promised security features. SOAS provides a framework to express and validate assurances. An *assurance* is essentially a statement about the properties of a component or service, typically made by the producer of the component or the provider of the service. Besides the specification of the security properties of the component, it adds a definition of how these properties are to be measured and by whom, and a recourse for the case that the promised property does not hold. *Assurance verification* is done by determining the existence or absence of the above properties. Enterprises can then link the required level of security of their IT systems and their business requirements, namely, the level of risk that the enterprise is willing to accept. In conclusion, SOAS empowers enterprises to provide security in dynamic service-oriented architectures while automatically procuring services that offer the right level of security.

This paper first presents the taxonomy concepts of SOAS, the use of SOAS for Web Services, and a basic architecture for monitoring assurances (§2). Next, it describes the actual use of SOAS (§3) and illustrates the concept of assurances by means of some example scenarios, putting particular emphasis on the separation of the assurance from the security mechanisms that achieve the assured property (§4). Finally, it discusses related work (§5) and concludes (§6).

2 Service-oriented Assurance

SOAS is a new paradigm defining security as an integral part of service-oriented architectures. It enables services to formalize and advertise their security assurances. Based on these declarations, services can address the core challenges of secure and flexible service composition:

– What are the security properties of a given service?
– How can the actual security be measured?
– What are the assumptions, failure possibilities, and dependencies of a given service?
– What evidence can be given that a service will or does indeed meet its security promise?
– Which remedies will be taken if a service does not provide the promised security?

In the remainder of this section, we outline the use of SOAS for Web Services, the taxonomy concepts of SOAS, and a basic architecture for monitoring SOAS assurances.

2.1 From Service Level Agreements to SOAS

Web Services are the preferred way of describing services in a service-oriented architecture. If a component needs a certain service, it discovers potential providers via directories and brokers, e.g., using UDDI, WSDL, and WS-Resource descriptions, and then engages with a specific service provider. In particular in cross-domain scenarios, this engagement is governed by a Service Level Agreement (SLA), e.g., expressed in WS-Agreement, which summarizes the requester's and provider's agreement on what the service is supposed to do. An SLA defines the quality of service, how and by whom that quality is measured, and what has to happen if the service quality is insufficient. Today SLAs are often implicit (in particular for services within one organization) and in most cases fairly static and pre-negotiated. But this is expected to change – in the future service providers will be selected more dynamically and hence SLAs will be more pervasive and negotiated in real time. This negotiation will become part of the overall process and of the overall Web Services stack.

Service-oriented Assurance adds security to this picture: Before two components engage in a service, they provide each other with assurances, i.e., security guarantees, as part of the SLA negotiation process. Examples are promises to provide certain process or data isolation, to comply with a regulation, or to accept a certain risk, or also statements of identity, etc., together with arguments why these properties hold, such as certificates for Common Criteria security evaluations, hardware-based integrity statements, or identity certificates and digital signatures.

Depending on how the SLA defines the manner in which security quality is measured, components may gather evidence during operation, i.e., information that documents and maybe even proves the state of transactions or the security posture of the component. This information can be security alarms, entries in log files, authenticated messages received from other components, hardware-based integrity measurements, etc. If something goes wrong, this information becomes the basis for fault diagnosis and forensics. Once a problem has been identified, the assurances will point to the components responsible for solving the problem and for covering damages. This is particularly important in a cross-domain scenario involving different organizations, where the result may be an actual financial recourse.

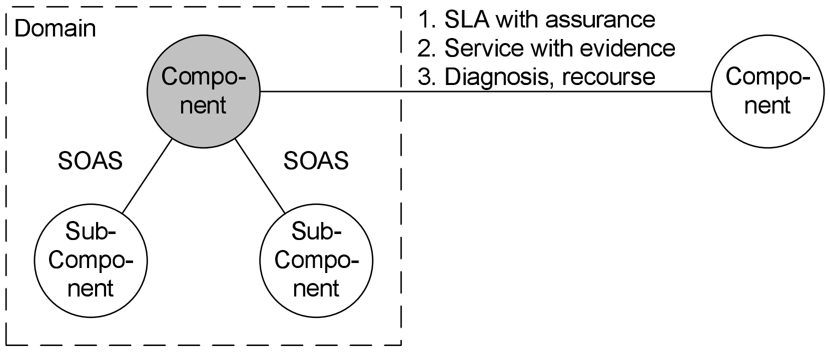

Fig. 1. Service-oriented Assurance

Figure 1 summarizes the use of assurances. We mainly consider the gray component on the left, e.g., a business process. It uses the service of another component (process), which may be in another domain, and of local sub-components. In a service-oriented architecture, there is no great difference between these two uses, except that there is by necessity a stronger dependency on some local components, e.g., the underlying operating system. In a first step, the processes negotiate an SLA with assurances. This is done in the context of the processes' own service assurances to their users. Secondly, during normal service, both processes may gather evidence of their own correct operation and of the operation of their partner; some of this evidence may be exchanged explicitly.

In case of problems, diagnosis and forensics should be possible based on the evidence gathered, and the SLA will provide procedures for a potential cross-domain recourse.

2.2 High-level Model

To enable the formalization of statements that express the security promised for a given service, SOAS defines a model as shown in Figure 2. In theoretical terms, this is in essence a meta-model of service descriptions.[1] Note that SOAS only formalizes the structure of security statements and that properties must not necessarily be expressed in a formal way; they may simply denote a certification such as a security label like EAL-4 or a privacy seal like TRUSTe having a precise meaning given from outside the SOAS model. The figure is drawn in UML, a widely used graphical design language.

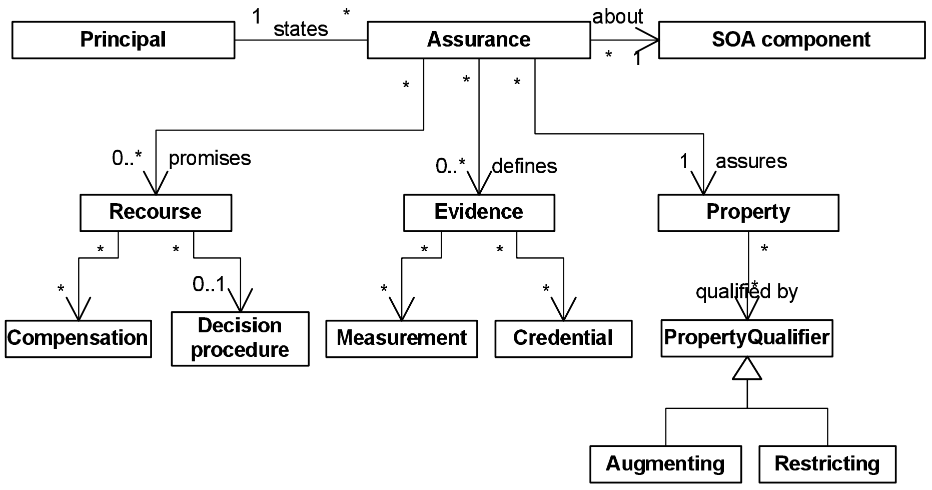

Fig. 2. Assurances on SOA components

The security of a SOA component is established by the properties it is expected to implement. Thus, declaring (security) *properties* is the foundation of the SOAS model. Properties of SOA components can be written texts (like contracts) but will often be machine-readable to enable automation and comparison beyond equality tests. Simple properties may define the I/O syntax of a service by, e.g., promising to adhere to a WSDL schema. Properties may also make statements about the actual behavior of a service stating, for example, the privacy of personal data. More examples are given in §4. This can be done using formal specifications or textual descriptions of the expected services.

[1] A meta model of SOA exists in [1], but it does not classify service descriptions or service properties. Given a meta model, corresponding models can be serialized either automatically, i.e., a "language" is defined implicitly by the UML model, or manually, e.g., into extensions of existing Web Services specification languages.

Properties either hold unconditionally or are qualified. A *qualifier* augments or restricts a given property. In an augmenting qualifier such as availability or performance, the resulting qualified property always implies the original property. Restricting qualifiers are key to modeling security. Typically they express environmental assumptions needed, e.g., a trust model specifying entities that are assumed to be correct, failure probabilities, or validation methods.

An *assurance* is a statement about a property of a *SOA component* or service, made by a *principal* such as the producer of the component or the provider of the service. Assurances define the evidence the component has to deliver to show it indeed provides the desired properties, and a specification of recourse if the component fails to provide these properties.

Evidence describes the information provided by the service to support the assurance, typically by enhancing the credibility of other elements. Mostly provided in the form of credentials, evidence may corroborate that the principal builds good components by customer references or a formal certification. Or it may corroborate the component properties, e.g., by supplying a certificate of a claimed Common Criteria evaluation or by describing the procedure used for determining the mean time between failures. Or it may corroborate a recourse, e.g., by showing that the principal has reserved funds. Evidence also defines how the property can be measured and by whom. For instance, it can be the retrieval of a log file by the service provider, a third-party audit, or a measurement signed by secure hardware included in the platform [10].

A *recourse* consists of a decision procedure and possible compensations. For dispute resolution, agreement on the interpretation of the measurement is essential to enable the parties to agree whether a property is fulfilled. A *decision procedure* provides instructions on how to deal with cases where, for example, some properties are not immediately measurable because, for instance, probabilities depend on how long one measures, or secrecy violations may not be noticed at once. If neither party fully trusts the other's measurements, the decision procedure may require that both sides measure in parallel or may even state that in case of conflicts additional proofs are provided by third parties [4].

Whenever the stated security property does not hold or can no longer be guaranteed, a *compensation* states a penalty, e.g., a sum of money, or defines a remediation process that re-instates security and is considered to be sufficient to satisfy the property. Whereas a penalty does not necessarily require the assurance to be re-established, remediation on the other hand is a well-defined process how a violation can be removed and how the system is guaranteed to reach a state that provides the given asurance. An example of the latter is the property of absence of viruses. The provider can guarantee that once a virus is discovered during a regular check, the virus is removed within 1 h and integrity of the installation is re-verified.

2.3 Monitoring Security Properties

Besides giving *explicit security assurances* as outlined above, SOAS must also support the *verifiability* of these assurances. Whereas the first challenge requires a language for specifying the assurances to be included into SLAs, verifiability of these assurances may be achieved by providing measurements supporting the evidence in the stated properties. Figure 3 outlines possible interactions between measurement components of a SOAS-enabled service. The interactions are structured into two phases. Before actually providing a service, the provider and requester agree on the Security SLA that describes the desired security. Once an agreement is reached, the service will be provided and its security can be monitored. Depending on the trust model, an optional observer can act as a referee to decide whether the properties are indeed met.

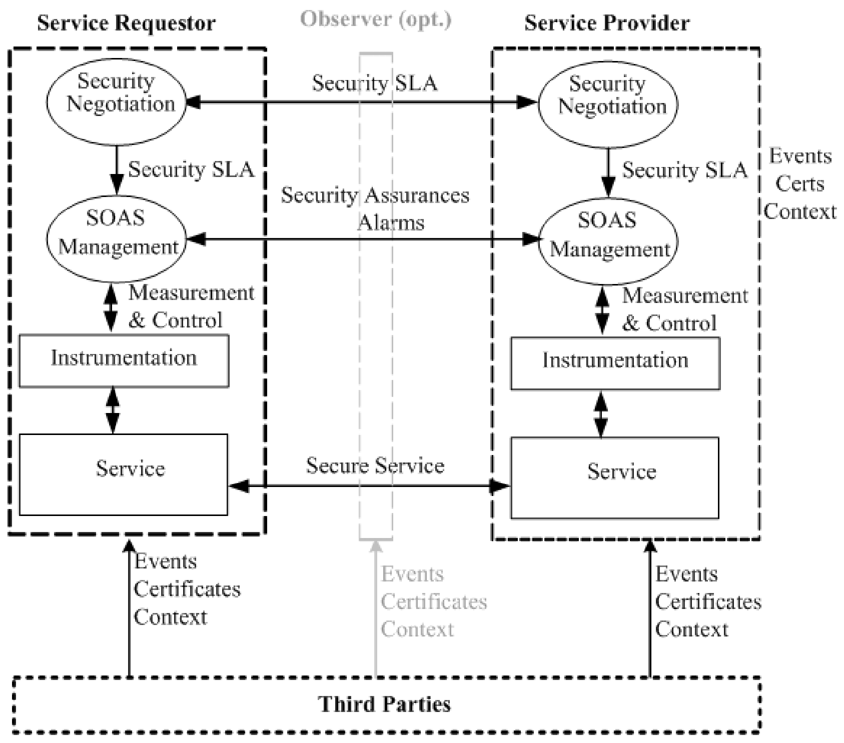

Fig. 3. Run-time monitoring of Service-oriented Assurance

To enable security monitoring, both the service and the client are instrumented. This instrumentation measures the security parameters of the service, input to a SOAS Management component verifying the given security assurance.

2.4 Types of Security Evidence

For assessing the security properties, data from various sources are needed. Both parties may evaluate system states – including security policies in place – and events that are provided through instrumentation as well as certificates, and other context from third parties. Examples are certificates of acceptable software as well as events that represent newly discovered vulnerabilities of the installed code base.

In principle, we distinguish three main types of security measurements depending on the time period addressed:

– An *audit log* makes it possible to verify whether a system has privided the assurance in the past. Examples include log files of past virus scans, an execution history listing the executables loaded in memory [10], and access records for confidential data.
– The current *state* enables one to prove statements about a given moment in time. This can include ownership proofs about the provider implemented by certificates or the absence of a virus at this point by a virus scan.
– Convincing a verifier that certain *policies* are enforced enables a system to indicate that a security property is likely to hold in the future. Examples are training programs, access-control policies, and policies for remediating specific failures.

Depending on the trust model, additional evidence may be needed to convince the verifier that the given data is accurate. In particular for policies, it is essential to convince the verifier that they will not be replaced with inappropriate ones. A similar challenge exists for state and audit logs. If the verifier does not trust the systems providing the service, additional evidence such as audits at random times may be needed. Furthermore, evidence of insecurity may turn up in various ways, e.g., by finding confidential data on the Internet or via whistle blowers.

Properties and thus their measurements may be qualitative or quantitative. Whereas the former simply determine whether a property holds, the latter measurements determine how strong the property is. Quantative security is not yet common-place but there are example properties such as quantative information flow or measures like the number of known vulnerabilities and k-anonymity.

3 Applying SOAS

To gain security addressed consistently and naturally at the right places, SOAS should be integrated in the overall software architectures and tooling. However, SOAS can to some extent also be retrofitted to legacy systems by documenting and exposing their known security properties and publishing corresponding security assurances. This explicit expression of security enables service selectors to consider security as a criterion for service selection, which in turn creates a reward for security. This reward will enable appropriate economic mechanisms that lead to the highest levels of security where this is most beneficial, and allows security to increase in economically cost-effective steps.

To implement fine-grained assurance statements, security assurances must be propagated and implemented along the software stack. As a consequence, software components would either implement their own security mechanisms (e.g., a banking application using one-time passwords), use security mechanisms from lower layers (e.g., SSL encryption), or use a security infrastructure for transparent protection (e.g., anti-virus or isolation services).

Propagating explicit assurances of components requires explicit exposure of assumptions as well as a more active security management in each component. Accordingly, components need to identify their assurances based on the assurances offered by sub-components. Another important aspect is that in order to enable its use in an open environment where not all components trust each other, SOAS creates an incentive that components enable verifiability of their security properties. This means that components can produce evidence that the claimed properties are actually true. A (simple) example is ownership: A service can use attribute certificates to prove that it is owned by a certain entity. This enables higher-layer services to measure and validate the assurances provided by lower-layer services. More complex scenarios include components that guarantee to report their security status honestly based on a well-defined measurement method.

Another important application of SOAS is to use security assurances to select appropriate services and to compose services. Once sub-components declare their service guarantees, services can factor security into the decision which sub-service to select. Loosely speaking, SOAS enables a service to discover the sub-service with the right level of security and the best cost/risk trade-off. Based on the assured security properties including potential qualifiers, a service may decide which tasks to entrust to each particular service. If a sub-service does not provide the full guarantees that are needed, a service can decide to augment the guarantees, e.g., by running replicas or obtaining additional recourses that remedy the losses in case of failure.

4 Example: Security of an Outsourced Business Process

In this section we illustrate the concept of assurances based on a larger example where a bank outsources a business process to an external provider. We first identify the overall assurances and then elaborate on two specific properties.

4.1 Overall Security Agreement

The overall goal is to manage the security of a business process by identifying and verifying the security properties of its sub-processes. In practice, this initial usage of SOAS is possible without automatic verifiability of the security properties. Instead, assurance can be achieved by signed statements of the service provider containing sufficient compensation in case well-defined security measurements indicate that the promised security cannot be or has not been achieved.

Let us consider a bank that outsources a sub-process such as payment processing to an outsourcing company. As is currently done, the bank and the outsourcing provider establish a service-level agreement for the outsourced process. This SLA defines the actual service as well as key performance indicators such as availability and throughput. This SLA can be negotiated and fixed using WS-Agreement. It will also define the WSDL interfaces to the service.

Because the sub-process is critical to the business of the bank, security requirements have to be added. Using SOAS, the bank and the outsourcing provider agree upon the actual security to be provided by the sub-process, how it is measured, and what compensation will be offered in case of failure. These security guarantees may cover different aspects of the outsourcing infrastructure and can be structured into distinct properties:

Basic integrity properties. The provider assures integrity properties on the input and the output data. In addition, it may state that input data of any kind will not lead to buffer overflows.

User management. The provider defines how users are authenticated and how access to the sub-process is restricted to the appropriate applications in the bank.

Basic infrastructure. The provider defines which availability is guaranteed and how it will be achieved, e.g., by replication, backups, and disaster recovery measures.

Isolation. The provider guarantees that this business process is completely isolated from other business. In particular, isolation holds even if processes are executed on behalf of other banks.

Application quality control. The provider defines how applications are tested and how quality is achieved. In particular, the provider guarantees that only applications that have passed a well-defined test-suite will be used to provide the service.

Security policies. The provider guarantees that the process is managed according to well-defined security policies. These policies include staff education, proper security zoning and boundary control, as well as emergency response for the corresponding services. The security policies also include virus protection and intrusion detection and response measurements.

All these properties are declared using signed statements. Because they are difficult to be verified automatically, validation can still be achieved by external auditors or audit teams from the bank or from the provider. Once the provider fails to comply, appropriate recourses from the initial assurance are used to remedy or compensate a failure.

4.2 Security Management – Customer Isolation

In an outsourcing environment, data owned by different customers must be isolated; i.e., no information may flow between customers except through well-defined business processes. This causes no problem in today's outsourcing environment, where most resources and applications are dedicated to one customer. In the case of shared applications or resources, however, they must be certified to provide appropriate isolation. As a consequence, a property promised may be that "there is no information flow between all services of this customer and any service provided to another customer".

To securely implement above guarantees, the provider either has to provide dedicated resources for each customer or to guarantee that no shared resource leaks any information between customers. This guarantee for shared applications can be done by means of an evaluation and certification. An alternative is the provision of virtualized resources (such as logical partitions) that are dedicated to each customer, enabling different customers to share one machine but still providing guaranteed separation. However, as it is hard to analyze whether a shared application allows information flow or not, both parties may have to accept some level of risk.

Depending on the trust the bank puts into the provider, the actual mechanisms that are used as well as their verifiability will differ. One way to provide assurance is to provide a signed statement of the provider or an auditor. If the trust in the provider or the auditor was unjustified, the customer may notice a violation only if the undesired information flow has visible effects, e.g., secret data clearly being used by competitors. For these cases, there must be compensation. The decision procedure may be aided by watermarking techniques. However, mechanisms where the service provider is trusted to notify someone of security violations are known and can be effective, as the experience with the California Senate Bill No. 1386 shows.[2]

Property qualifiers can be used to define limitations of virtualization including the requirement that certain services be not virtualized, virtualized on a dedicated resource, or hosted on machines satisfying certain criteria such as physical security protection or location [2]. An example of the latter are the concerns that Canadian personal data will fall under the US Patriot Act once they are hosted on machines that are physically in the USA [8].

4.3 Security Management – Virus Protection

Service users require that machines hosting critical services follow basic security guidelines. The property that is promised by a service is that the machine providing a service is managed according to well-known security guidelines. Such guidelines usually require sound patch management, firewalls, and appropriate virus protection.

Assurance of appropriate virus protection, for example, can be implemented in diffent ways. Using certification and recourse, the service provider promises to manage the machines according to the guidelines and certifies this including recourse. As virus attacks are usually quite visible by loss of availability, the bank may not require specific measurements in the assurance if the recourse is sufficient to cover potential losses. Alternatively, sound virus protection may be indicated by means of an audit trail of recent virus scans to convince a verifier that no virus activity was detected while a given service was being provided. Moreover, assurance for this property can be provided by means of integrity-based computing (IBC) mechanisms. For virus protection, IBC can prove at regular intervals that a virus scanner has been resident in memory and not been invalidated.

5 Related Work

Several models and languages formalize agreements (contracts) on electronic services [4, 11, 12], covering agreement specification as well as system architecture. However, they mainly focus on specific aspects of services. For example, WSLA is a language for the specification of quality-of-service agreements for Web services. Besides providing a type system for SLA artifacts, WSLA identifies the contractual parties, specifies the

[2] Summaries of incidents cataloged on PIPEDA and Canadian Privacy Law can be found at `http://www.privacylawyer.ca/blog/2005/02/summaries-of-incidents-cataloged-on.html`.

characteristics of the service and its observable parameters, and defines the guarantees and constraints that may be imposed on the SLA parameters [4].

WS-Agreement is a standardization effort defining an agreement format, an agreement establishment protocol, and a runtime agreement monitoring interface. Agreement terms represent contractual obligations, including specific guarantees given [5]. Guarantee terms specify service level objectives, a qualifying condition under which objectives are to be met, and a business value giving the importance of meeting these objectives.

The Composite Assurance Mapping Language (CAML) provides a notation for claim trees for the assurance arguments related to enterprise security objectives, providing causalities, relationships, vulnerabilities, threats, and other system- and environment-related issues [6]. A CAML specification hierarchically refines security claims about the system into sub-claims that, eventually, are linked with the evidence that a claim is satisfied. Refinement is supported by the general strategy, assumptions, and dependencies, justifying reasons, and contextual models.

Security properties of components can be measured and verified by using products such as Symantec Enterprise Security Manager or IBM Tivoli Security Compliance Manager (SCM). SCM gathers information from multiple computer systems, such as registry and application information, analyzes the data, and produces reports to reveal adherence to security policies. Collectors retrieve specific data by reading files or running an executable program. Data collected on client systems is stored in a database on the server. Conditions are expressed as SQL statements that analyze data from the database tables to provide a list of client machines violating the conditions.

Also trusted computing allows one to verify the integrity of a platform (attestation), whereby secure boot and strong isolation guarantee integrity. Remote attestation authenticates software to remote parties. However, attestation based only on the configuration of software and hardware components entails the problem of managing the multitude of possible configurations, system updates, and backups [3, 7, 9]. A trusted virtual machine, as for example proposed by Haldar et al [3], can execute platform-independent code to attest programs, thus certifying various properties of code running under it by explicitly deriving or enforcing them. SOAS assurances may provide the language to express these properties and the way they should be verified.

6 Conclusion

Service-oriented Assurance enables products and services to provide well-specified security guarantees, which can be monitored and validated. These assurances enable enterprises to select services that offer the right level of security. Our example illustrates that it is feasible to specify important security properties in a vendor-agnostic and platform-independent way. As a consequence, we believe that SOAS is the logical future of security in service-oriented architectures.

Our proposal is only a first step in this direction. Further work is required in the formalization of a broad range of specific security properties and on assurance verification as well as on service composition. There is still a long way to go before security risks are comprehensively managed and become normal economic factors on the business layer. Nevertheless we have demonstrated a framework that shows how the objectives stated above can be achieved and that first meaningful ways exist to instantiate this framework based on current software and hardware capabilities.

Acknowledgments

We want to thank Bob Blakley, Tom Corbi, Bruce Harreld, Linda Henry, Anthony Nadalin, Nataraj Nagaratnam, Chris O'Connor, Charles Palmer, Ronald Perez, and Andreas Wespi for interesting feedback.

References

1. L. Baresi, R. Heckel, S. Thöne, and D. Varró. An architectural style for service-oriented architectures. `www.upb.de/cs/ag-engels/ag_engl/People/Thoene/MRDSA/SOA-Metamodel.pdf`, Sept. 2003.
2. J. L. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn, and R. Cáceres. Trusted Virtual Domains: Toward secure distributed services. In *Workshop on Hot Topics in System Dependability*, 2005.
3. V. Haldar, D. Chandra, and M. Franz. Semantic remote attestation: A virtual machine directed approach to trusted computing. In *USENIX Virtual Machine Research and Technology Symposium*, pages 29–41, 2004.
4. A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management, Special Issue on E-Business Management*, 11(1), Mar. 2003. Plenum Publishing Corporation.
5. H. Ludwig, A. Dan, and R. Kearney. Cremona: an architecture and library for creation and monitoring of WS-agreements. In *2nd International Conference on Service Oriented Computing (ICSOC '04)*, pages 65–74, New York, NY, USA, 2004. ACM Press.
6. J. S. Park, B. Montrose, and J. N. Froscher. Tools for information assurance arguments. In *DARPA Information Survivability Conference and Exposition II (DISCEX'01)*, volume 1, pages 287–296, 2001.
7. J. Poritz, M. Schunter, E.V. Herreweghen, and M. Waidner. Property attestation — scalable and privacy-friendly security assessment of peer computers. IBM Research Report RZ 3548, 2004.
8. Public Sector Outsourcing, Information & Privacy Commissioner for British Columbia. Privacy and the USA Patriot Act - Implications for British Columbia. `www.oipcbc.org/sector_public/usa_patriot_act/pdfs/report/privacy-final.pdf`, Oct. 2004.
9. A.-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: Caring about policies, not mechanisms. In *New Security Paradigm Workshop 2004*, pages 67–77. ACM Press, 2005.
10. R. Sailer, T. Jaeger, X. Zhang, and L. van Doorn. Attestation-based policy enforcement for remote access. In *11th ACM Conference on Computer and Communications Security*, pages 308–317. ACM Press, 2004.
11. J. Skene, D. Lamanna, and W. Emmerich. Precise service level agreements. In *26th Int. Conference on Software Engineering*, pages 179–188. IEEE Computer Society Press, 2004.
12. V. Tosci, B. Pagurek, and K. Patel. WSOL – a language for the formal specification of classes of service for web services. In *International Conference on Web Services (ICWS'03)*, pages 375–381. CSRA Press, 2003.