André Miede, Nedislav Nedyalkov, Dieter Schuller, Nicolas Repp, Ralf Steinmetz:
Cross-organizational Security – The Service-oriented Difference. In: Asit Dan, Frédéric Gittler, Farouk Toumani: Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops, p. 72--81, Springer, September 2010. ISBN 978-3-642-16131-5. (Extended Version.)

Cross-organizational Security – The Service-oriented Difference

André Miede, Nedislav Nedyalkov, Dieter Schuller,
Nicolas Repp, and Ralf Steinmetz

Multimedia Communications Lab (KOM) – Technische Universität Darmstadt
Department of Electrical Engineering & Information Technology
Merckstraße 25, D-64283 Darmstadt, Germany
http://www.KOM.tu-darmstadt.de
Corresponding Author: Andre.Miede@KOM.tu-darmstadt.de

Abstract Service-oriented Architectures (SOA) are a powerful paradigm to address integration challenges for information technology systems in enterprises. The service-based integration of legacy systems and business partner systems makes it necessary to introduce and adapt suitable SOA security measures in order to secure the enterprise both within and for cross-organizational collaboration. While there is an active research community for SOA security, standard literature on the topic has not yet identified the influence of the SOA paradigm on security aspects in a structured manner, especially in an enterprise context. In our paper, we work towards this goal by identifying the main elements of cross-organizational SOA in form of a conceptual model and by discussing these elements regarding their impact on security issues. Based on this, we define and structure important research challenges for SOA security.

1 Introduction

Globalization and technological advancements are major drivers for enterprises in the modern world. Enterprises face constant challenges to react to rapidly changing market requirements in order to stay competitive. An important factor to achieve this goal is the underlying enterprise information technology (IT). It has to provide solutions to model, operate, and adapt business processes efficiently and effectively. In the last years, the integration of both company-wide and inter-company IT systems has emerged as a big challenge in this context [1,2]. Often, enterprise IT systems have not been designed and developed as a holistic concept but they have rather grown in a heterogeneous and organic fashion over time. This causes serious problems when changing things at the process-level due to the high amount of redundant data and code.

Paradigms such as *Service-oriented Architectures* (SOAs) [3, 4] offer technological and organizational possibilities to evolve towards a more horizontally organized IT landscape and, thus, to improve the alignment between the business and the IT side – which is difficult in silo-like, vertically organized landscapes.

SOAs are based on the "service" concept, where services can be seen as black boxes representing business functionalities. In a typical SOA setup, a service consumer learns about a suitable service provider via a service registry and then requests functionality from this service provider via an arbitrary communication channel. These services are used to assemble business processes as service compositions and may even cross enterprise boundaries, thus, enabling *service-based, cross-organizational workflows* [5, 6]. In the last years, the SOA concept has become a successful way of addressing the challenges mentioned above, e. g., using *Web service technology* as an implementation.

In order to enable service-based, cross-organizational collaboration, the security of the participating systems, exchanged messages, and used communication channels has to be ensured. Achieving and guaranteeing basic IT security goals such as confidentiality, authentication, authorization, non-repudiation, integrity, and availability [7–9] is an absolute must in this context and still an active topic both in research and industry. Although security introduces additional costs and has an impact on the Quality of Service, unsecured business transactions are not an option in most scenarios.

The concrete application scenario for our research is the domain of service-based collaboration between organizations, e. g., enterprises. In order to execute its business processes, an enterprise often needs to integrate third-party services offered by different external service providers. An example for this is the retrieval of credit ratings for customers from an external rating agency in a credit application workflow or the request for different types of market data from a stock exchange for a financial derivatives workflow. Fig. 1 depicts an example for a generic trading process and shows how business process steps can be mapped on services. The called services can be composed of other services which can also be loosely coupled, i. e., dynamically selected from among different providers based on performance and cost criteria.

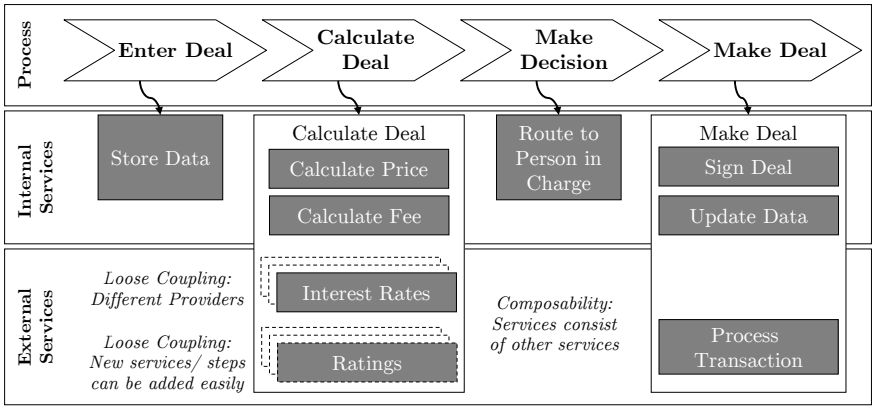

Figure 1. Example for a generic trading process and its mapping on services

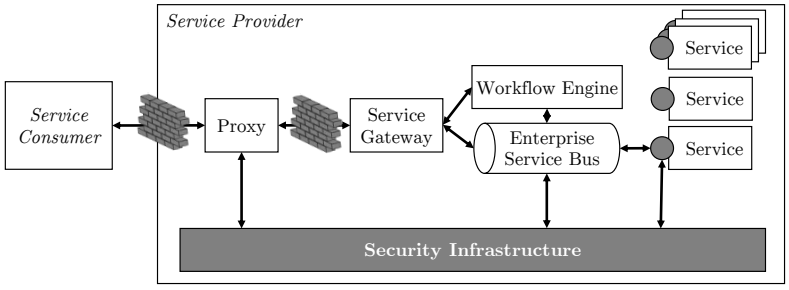

Figure 2. Generic cross-organizational SOA scenario (based on [6, 10–12])

While for simple applications the basic SOA scenario as described above still captures all relevant elements for the message exchange, cross-organizational or enterprise SOA has to use more infrastructure, as depicted in Fig. 2, which has a greater impact on securing this kind of collaboration.

It is important to note that in an enterprise setting, the service consumer and service provider do not connect their services directly due to their internal architecture. Instead, a proxy within a demilitarized zone is the end-point for incoming communication and performs basic operations, e. g., the general authorization of the request. A service gateway then establishes the connection with a workflow engine (if workflow control is necessary and possible) or the enterprise service bus (ESB) for further routing towards the target service and other operations. Before actually reaching the target service itself, further proxies – often also called adaptors or connectors – are used to decouple security operations from the core functionality of the service [2, 13]. The aspect of point-to-point-communication vs. end-to-end-communication and transport-level-security vs. message-level-security has already been discussed in the literature (e. g., [13]), but the impact of this setup goes further as described in the following.

Based on this scenario, the goal of this paper is to analyze all major elements of cross-organizational Service-oriented Architectures regarding their impact on security. This is a step towards a better understanding of the security implications SOA has in general and in particular. It aims at identifying important research areas in this field in order to make SOA more secure.

The rest of the paper is structured as follows: Section 2 gives an overview of the current understanding of security in SOA, based on the standard literature in this area. Section 3 introduces a conceptual model for cross-organizational SOA which identifies and orders its core elements. Based on this model, the security impact of the identified elements is discussed. Section 4 sums the findings up by including them into an integrated approach towards SOA security. The paper closes with a brief outlook on future work.

2 Related Work

This section gives a state-of-the-art overview how SOA security is seen by major researchers in the field [5,10,13,14]. Discussions of SOA security are mainly based

on traditional goals of IT security which are then complemented with regard to SOA particularities [13, 14]:

- *Authentication* and *Authorization*: an interoperable authentication scheme of all the service-calls inside a supply chain is needed, as different authentication schemes of (maybe even anonymous) partners must be aligned. A desirable feature is the introduction of a Single-Sign-On possibility across the whole service supply chain.
  In addition, a service can be called in different ways by different services inside its own domain or outside organizational boundaries. These multi-client capabilities require run-time security checks [5]. A commonly suggested approach is to introduce the concept of *security-as-a-service*, in this context a service which manages identities and their roles. Such a service is a Policy-Decision-Point (PDP), here a centralized identity provider. Whenever another service needs to make a security decision (Policy-Enforcement-Point) it sends a request to the PDP. The term security-as-a-service is also found and discussed in [13] along with policy-driven security. Security issues are therefore addressed by policies for non-functional needs. This is a way of specifying security requirements outside of applications as security models and it is difficult to be hard-coded into an application itself [10].
- *Integrity* and *Confidentiality*: each service along a supply chain has access to the initial message from a service consumer. Services along the way change some parts of the message where necessary to fulfill their part of the business process and forward it afterwards. On this account it is useful to enable signing and encryption not only of the whole message but also of parts of it. In this manner *message-level security* or *end-to-end security* is achieved for sensitive information inside a message. [13] also emphasize this newly-emerged term in the SOA context. End-to-end security for a consumer and provider is needed through the whole service supply chain (as opposed to point-to-point security between two directly connected applications).
- *Non-repudiation*: an SOA service call can be part of a critical business process. Therefore, it is also very important that legal regulations are abided by and that a party is incapable of denying actions already taken. The exchange of signed and time-stamped messages is required. Mechanisms already discussed for integrity and confidentiality can be used to achieve this security goal.
- *Availability*: due to the fact that the SOA paradigm can be used to build distributed systems, the possibility of an outage of some services has to be considered. Nevertheless the flow of the business process must not be hindered. To achieve availability redundancies may be introduced, any single-point-of-failure or bottlenecks must be avoided. As the number of consumers is not necessarily known prior to service usage, the services themselves and the overall system must be implemented bearing scalability in mind.

Additional aspects of SOA security are addressed by further literature in this area. Josuttis [5] also presents the concepts of the ESB, which manages the

interaction between services in a domain. When connected to an ESB, services are open to the outside world, the "natural" firewall of being only known to a certain client or having a non-open protocol of work is non-existent. Further discussion addresses the statelessness of services which might be compromised to ensure the security of the overall system.

Hafner and Breu [10] touch yet other aspects and features of SOA such as the loose coupling and composability between services which have a considerable impact on security. Another fact that should also not be underestimated is that each partner wants to stay in control of their own part of the service chain, which leads naturally to a decentralized style architecture. Therefore security-critical applications involving many partners and crossing domain borders evolve.

There is also a number of official standards which address the issues with the introduction of the SOA paradigm [5, 10, 13] on a technological level. OASIS[1] has introduced WS-* standards addressing security challenges for Web services, a widely known and used implementation of SOA. WS-Security, for example, specifies a standard way to embed security tokens in the header of a message. The tokens are used to digitally sign or encrypt the message or parts of it and how to embed these parts within the message. The data is encrypted with a symmetric encryption scheme and then inserted into the message according to the standards XML-Signature and XML-Encryption by the W3C[2].

The main tenor of current SOA security research is that conventional security measures are not sufficient in the SOA context. In summary, the major aspects of SOA security in standard literature are the following:

1. A switch from point-to-point-security towards end-to-end-security is necessary. For example, secure communication channels by SSL or TLS[3] or authentication by digital signatures are seen as a prerequisite but they need to be extended to fulfill the goal of building a secure architecture based on the SOA paradigm.
2. A dedicated security infrastructure is required to integrate different security mechanisms in a flexible and automated manner. The concept of security-as-a-service is seen as an important component of this infrastructure.
3. If any structure is chosen, the preferred structure for presenting the topic is according to the classic IT security goals (as described above).
4. Another trend is to equalize SOA security with Web service security, reducing SOA security requirements to Web service security standards and their configuration.

Having such an incomplete and rather unstructured understanding of security – a mission-critical element of any IT architecture – is a major obstacle towards a dedicated security analysis. Without a clear understanding of the impact the SOA paradigm and its elements have on security, an assessment of risks or the prevention of attacks is difficult.

[1] OASIS: Organization for the Advancement of Structured Information Standards
[2] W3C: The World Wide Web Consortium
[3] SSL: Secure Sockets Layer; TLS: Transport Layer Security

3 Cross-organizational Security – The Service-oriented Difference

In this section, we propose two steps towards a more detailed and structured approach for SOA security. First, we introduce a means to capture the elements of cross-organizational SOA. Second, building on these elements, their impact on security is discussed. Both aim at getting a better understanding of the particular security requirements the SOA paradigm poses. Furthermore, it lays the foundation for an integrated SOA security approach and the proposal of a research agenda.

3.1 A Conceptual Model for Service-oriented Architectures

In order to understand the differences which SOA security requires for cross-organizational collaboration, we have assembled an inventory of the major elements used for such a scenario. These elements are represented and structured in the form of a conceptual model, which is based on SOA definitions and descriptions found in standard literature on the topic, i. e., books from practitioners and researchers [1, 2, 5, 6, 15]. Fig. 3 shows the conceptual model. In the following, it is described in more detail to lay the foundation for the security analysis of its elements in the next section.

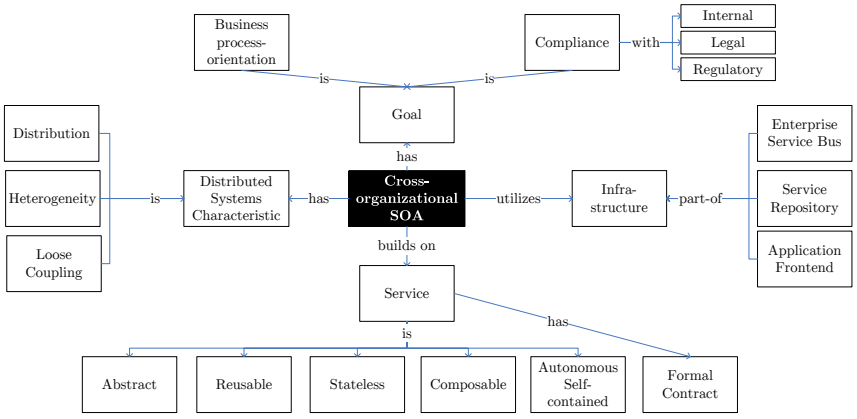

Figure 3. Conceptual model for cross-organizational SOA (based on SOA concepts, characteristics, and ingredients described in [1, 2, 5, 6, 15])

1. *Goal*: just as any IT system, an SOA has to support achieving certain objectives an organization has.
   (a) *Business Processes-orientation* [1, 2, 5, 6]: in order to realize improved business-IT-alignment, the underlying IT has to be as close as possible to the business processes. This includes technical support such as a

workflow engine or the close cooperation between IT and functional departments, i. e., facilitated by special modeling and transformation tools. Due to its organizational impact, it is not only an implementation issue but should rather be seen as a strategic goal of the overall organization.

(b) *Compliance* [2, 5]: as an SOA operates towards the fulfillment of organizational goals and processes, it must comply with internal, legal, and regulatory requirements. Examples for such requirements are the Sarbanes Oxley Act (SoX) or the accords of the Basel Committee on Banking Supervision (Basel II). Compliance aspects can vary in different countries and, therefore, have a serious impact on cross-organizational workflows if the participating enterprises operate in different legal systems (cf. the German Privacy Law). A means to achieve compliance is SOA governance, a special branch of corporate and IT governance.

2. *Characteristic of Distributed Systems*: some of the basic SOA elements are nothing new, but have been adopted from general distributed systems.
   (a) *Distribution* [2, 5]: in a cross-organizational SOA, the services, the infrastructure, the underlying machinery, etc. is logically and often physically distributed. This results in different owners of the separate assets, shared infrastructure, open boundaries, the establishment of service market places, and semi-anonymous service consumers and providers.
   (b) *Heterogeneity* [1, 2, 5, 6, 15]: bringing together different legacy systems and/or those of different business partners makes heterogeneity a challenging fact in cross-organizational SOA. Thus, high interoperability has to be achieved using machine-readable, platform and language independent, and ideally open standards for communication and coordination.
   (c) *Loose Coupling* [1,2,5,6,15]: this denotes the dimension to which software elements build on each other. In general, it increases flexibility, scalability, and fault tolerance due to the possibility of replacing elements at run-time and, thus, reducing dependencies. However, it leads to more complexity because of possible side-effects and increased requirements regarding service comparability and discovery.
3. *Infrastructure*: in order to operate an SOA, a number of dedicated architectural elements are used.
   (a) *Service Repository* [1, 2, 5, 15]: we use the terms "repository" and "registry" synonymously. It does not contain the services themselves but facilitates service discovery by providing service descriptions to requestors. The repository can be within an organization (for internal services), it can be part of an external service provider's system, or it can be part of a service market place.
   (b) *(Enterprise) Service Bus* (ESB) [1, 2, 5]: the ESB connects the service consumers and providers in an SOA, both within an organization and between organizations. In addition to connectivity, an ESB can provide support for intelligent message routing, message transformation for protocol and interface independence, communication patterns (i. e., asynchronous), logging, auditing, etc.

(c) *Application Frontend* [1]: the application frontend is the concept which triggers and coordinates the actions of the organization's systems. It can be instantiated, e. g., by a graphical user interface, long-living processes, or batch programs.

4. *Service*: a service is the basic concept and core of an SOA. It is the technical representation and encapsulation of high-level business functionality.
   (a) *Formal Contract* [1, 5, 15]: this describes and specifies a service, its interface, purpose, functionality, the terms of information exchange, and applicable constraints. It includes specifications about the Quality of Service, i. e., in the form of Service Level Agreements (SLA).
   (b) *Abstraction* [5, 15]: services are a logical and technical layer to hide underlying logic and implementations. This includes a coarse granularity of the services to separate the interface from the internal structures.
   (c) *Reusable* [2,5,15]: ideally, functionality should be implemented only once and then be reused. Services support their potential reuse, but due to organizational and technological challenges, this might not happen.
   (d) *Stateless* [5, 6, 15]: this means that services do not maintain a client-specific state between service invocations. A common reason for this is the negative impact of state on loose coupling. However, while statelessness applies to services directly, state maintenance can be deferred to other parts of the system, e. g., the backend, which can be a legacy or general data-storing system being wrapped by services. Statelessness is a controversially discussed characteristic of services, an overview of its different aspects can be found in [5].
   (e) *Composable* [5,15]: in order to realize business and IT alignment, combinations of different services represent business processes of an organization. This means that services make use of other services by calling their interfaces, processing their output, etc.
   (f) *Autonomous/ Self-contained* [5, 15]: a service should be as independent as possible from other services. In addition, at execution-time of a service call, the service should be in control over the represented functionality, i. e., the underlying system.

An overview of how some of the different service concepts relate to each other can be found in Erl [15].

3.2 Security Impact of SOA Concepts

The security analysis of the previously discussed SOA elements is based on typical security concepts such as *asset*, *threat*, and *vulnerability*. We briefly describe these concepts in the following, a simplified overview of how they relate to each other and to other important security concepts is shown in Fig. 4.

- *Assets* are the objects and concepts to be protected from attacks [17]. The assets relevant for SOA security are described above in Section 3.1.

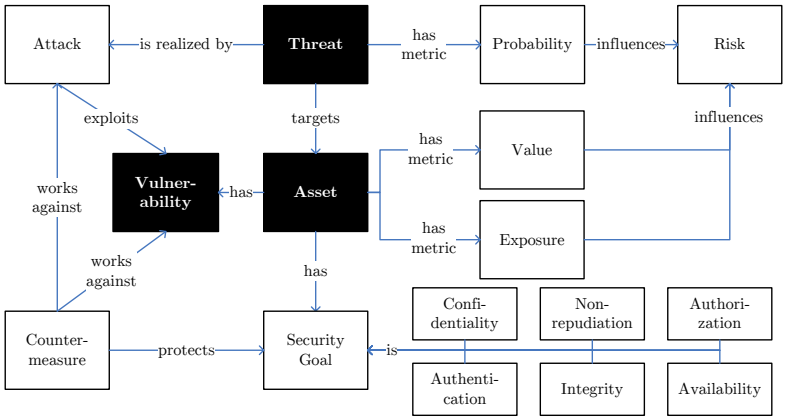

Figure 4. Important IT security aspects and their relationship (this is the core of a larger, more detailed metamodel for IT security [16])

- *Threat* is a possible way a system can be attacked [17]. Threats can be categorized in four classes according to their consequences [18]. While the general classes are the same for SOA, possible scenarios for them are briefly outlined:
  - *Disclosure* is the unauthorized access to data. In an SOA context this could be for example the interception of messages or the forced access to a restricted service.
  - *Deception* is the provision of false data (which is believed to be true). From a service consumer point of view this could be an attacker masquerading as a (known) service provider.
  - *Disruption* aims at preventing an asset from correct operation, i. e., Denial of Service (DoS). From a service provider point of view this could be a "consumer" flooding services with requests.
  - *Usurpation* leads to losing control of the asset to an unauthorized entity. A possible SOA scenario could be a hijacking of the service registry, resulting in requests being redirected to malicious services (see *Deception*).
- *Vulnerability* is a point or characteristic which enables an attacker to bypass the security mechanisms of a system [7, 19].

This is the focus of the following analysis. Each of the identified cross-organizational SOA assets is analyzed regarding its vulnerabilities and their exposure towards the threat classes.

1. *Goals* of an SOA offer new objectives for attackers and introduce new integration challenges.
   (a) *Business Processes-orientation* is another driver for security requirements, e. g., the "four-eyes principle", where two persons have to be successfully authenticated in order to authorize an action. It is an important aspect of many business processes and has to be supported by the underlying IT. Thus, if these requirements cannot be communicated

and modeled in a way which both business and technical experts can understand and implement, an SOA can lose many of its advantages. Especially, introducing security as an afterthought can put flexibility and the integration capabilities at stake.

(b) *Compliance* attacks can seriously compromise an organization, e. g., by making an enterprise fail to comply with privacy laws or by manipulating or deleting audit records. This can result in fines, the revocation of licenses, or the loss of reputation and trust. Compliance vulnerabilities are not necessarily of a technical nature, but can also target governance processes within the organization, i. e., reference processes or key performance indicators.

2. *Characteristics of Distributed Systems* bring along with their advantages also their security challenges to the area of SOA.
   (a) *Distribution* of services extends the security perimeter. Where systems were rather closed with limited access, e. g., only from within a specific department, services make these systems now available (and thus attackable) all over an organization and across its boundaries. Along with distribution come also different owners of resources and services, which makes the alignment of security interests and their enforcement challenging – especially if a central, common control instance is missing. These uncertain conditions are a fruitful environment for attackers to exploit potential disputes of authority between organizations.
   (b) *Heterogeneity* of communication and underlying systems is embraced by the SOA paradigm. However, different security concepts and implementations of the underlying technology have to be integrated seamlessly on the service layer, for example, to align countermeasures. Otherwise attackers can exploit the transitions between systems and security components, i. e., different user identities and roles. An important element towards a general security approach are open standards (cf. Section 2).
   (c) *Loose Coupling* is an important prerequisite for improved flexibility, but it can make it harder to establish trust between service parties due to spontaneous and even semi-anonymous communication among machines. Thus, reputation systems have to be established, for example. Side-effects must also be minimized, such as switching from a well-protected service to one which offers no protection at all. This makes it necessary to make services comparable not only regarding their functionality (e. g., semantics) and Quality of Service (e. g., SLAs) but also regarding the level of security they offer.
3. *Infrastructure* is necessary to operate an SOA but has both many new attack points and possible means to protect the organization.
   (a) *Service Repository* is an important, but very exposed part of the infrastructure and offers new attack scenarios. Availability is crucial here, as service consumers need the registry to find suitable service providers. Disruption threats, i. e., in the form of an DoS attack by massive fake requests, seriously affect the operation of a service-based organization. In addition, attacks on the communication between registry and service

consumers and providers, i. e., to hijack requests, are likely and feasible, too. Last but not least, poisoning the repository with entries about non-existent or even malicious services, e. g., in order to undermine the trust between all participants, is yet another possible attack scenario.

(b) *(Enterprise) Service Bus* serves as a mediator for the communication between service consumers and service providers within and between organizations, thus, requires end-to-end-security for the whole way.[4] Furthermore, an ESB creates an open system based on standardized protocols and can be used by attackers as well, e. g., to make service calls or to access service results of other consumers. Thus, the ESB has to integrate and automatize security mechanisms. On the other hand, the ESB must not only protect access to services but itself as well. Both disruption and usurpation of ESB operations are very attractive attack scenarios in order to take over control or to damage the organization, e. g., via Distributed Denial of Service attacks.

(c) *Application Frontends* can be – as described above – both an interface for a human user or another machine. A serious threat in this context is deception, e. g., in the form of phishing, where a human user is lured into providing his credentials to a malicious interface, which then misuses this information for its own purposes. Another threat arises in the form of usurpation, e. g., by manipulating or exchanging a batch program in order to execute a completely different process than intended. In general it is unclear, how information provided to an frontend, e. g., credentials, is used in the following services and what reaches the backend systems.

4. *Services* are the main components of an SOA and, therefore, a likely target for attacks.

(a) *Formal Contract* increases the exposure of a service as its interface is publicly described and available. This can be exploited manually and automatically, i. e., by automatic scans for weaknesses of the interface or by guessing additional, non-described interface functionality. This threat must not be confused with a call for "security by obscurity", it rather requires a critical revision of what information is made publicly available, what other information can be derived from this, and how both can be misused for attacks.

(b) *Abstraction* has a multiplicator effect on security. Where before a single application was secured, in an SOA this application can be abstracted by multiple services (or even service compositions). Now each of these abstractions has to be secured and, therefore, an attacker has many potential targets to choose from and often needs to succeed only with one. On the other hand, a security flaw in one application can affect the security of multiple services. Additionally, many abstraction layers increase the overall complexity of the system, at least in terms of security, i. e., propagating security information through all these layers safely. Attackers can specifically target different layers or their seams for weaknesses.

[4] Point-to-point connections are possible with special ESB setups but are no longer feasible with the scenario depicted in Fig. 2.

(c) *Reusable* services can introduce very serious security side-effects, because not all contexts in which they might be used can be foreseen. Thus, different contexts have different vulnerabilities, which cannot be secured against in advance. A service might also be reused in way it was not intended for, creating a vulnerability. For example, a service to access employee information was originally developed to show a *subset* of this information in a web frontend. The service is then reused – or better misused – by an insider to access and to show the full information. In this context a dedicated SOA governance is required to coordinate the reusability of services, especially from a security standpoint.

(d) *Stateless* services are a challenge for security considerations, because a security "context" – storing and communicating information about the current session – is very important, e. g., when calling the same service multiple times for a business process or when a service consists of multiple other services. Furthermore, maintaining no state allows attackers to record messages and to perform replay attacks if no suitable countermeasures are taken.

(e) *Composable* services must integrate security mechanisms throughout the whole service chain. Seams for exploitation can arise by unclear organizational responsibilities and incomplete integration of different security technology. It is also possible for attackers to introduce malicious services into the composition, i. e., to take over control, to intercept sensitive data, or to falsify information.

(f) *Autonomous/ Self-contained* services face the challenging question of "who is in charge of security?". If the service is fully self-contained, it has to include security functionality as well, otherwise it would depend on dedicated security services or the infrastructure, i. e., for access decisions. Autonomous services can introduce additional vulnerabilities as the tight coupling of security and service functionality introduces greater complexity and is more error-prone, e. g., if security updates are performed and one service is forgotten. Decoupling service functionality and security, i. e., via a security proxy, can leave the core service unprotected if the proxy can be bypassed and if the service can also be contacted directly.

In this section we have shown how common IT security concepts such as assets, threats, and vulnerabilities relate to the elements of a cross-organizational SOA. The general IT security requirements and threats basically do not differ from those an SOA faces. Single aspects of the above analysis may already be known from research and experience in the domain of distributed systems, but the SOA security challenge is to master the mix, interactions, and dependencies of all of them.

4 Conclusions and Future Work

In this paper, we presented a model of cross-organizational SOA concepts in order to analyze these elements regarding their impact on security. *Cross-organizational*

SOA security deals with the application of core IT security concepts such as threats, vulnerabilities etc. (cf. Fig. 4) on the elements of cross-organizational SOA such as loose coupling, composability etc. (cf. Fig. 3). We evaluated the security impact of single cross-organizational SOA elements and their relationships. While single security aspects of these elements are already well-known, i. e., for distributed systems characteristics, the combination of and relationships between the cross-organizational SOA elements as well as their impact makes cross-organizational SOA a special security challenge. Attackers thrive on uncontrolled complexity such as the manifold security challenges a cross-organizational SOA features. Thus, a structured approach to investigating and solving these challenges helps to decrease attack risks.

Furthermore, a common and clear understanding of research and industry topics in this area is necessary to work towards an integrated SOA security approach as depicted in Fig. 5. While the steps are well-known from general IT security, their contents have to be adapted to specific SOA-requirements, i. e., as we did in this paper for the analysis layer.

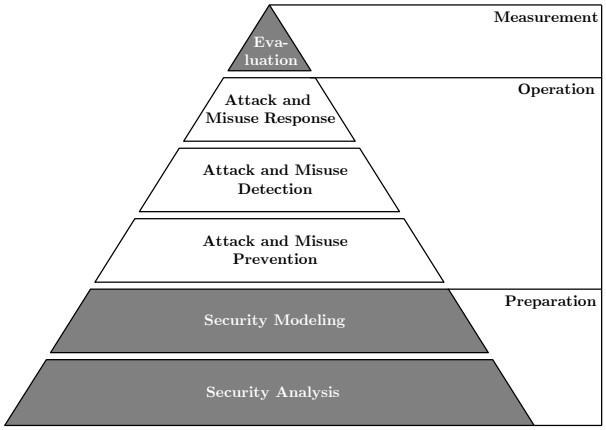

Figure 5. Overview of integrated (SOA) security approach

1. *Preparation*: security is a diverse topic spanning across departments, organizations and affecting both functional and non-functional requirements. This makes a thorough preparation process necessary, i. e., to identify and capture the SOA-specific impact on cross-organizational security.
   (a) *Security Analysis* includes a fundamental investigation of all involved assets, vulnerabilities they might have, threats they face, potential attacks, a calculation of associated risks, possible countermeasures, and so on. In this context, an analysis of the applicability of "Jericho-style security"[5] on the SOA paradigm might prove beneficial.

[5] `http://www.opengroup.org/jericho/`

(b) *Security Modeling* is the basis for operating an organization and its IT systems. It consists of security-aware (business) process modeling, i. e., by annotations to or extensions of already existing languages such as BPMN[6] to achieve compliance with laws and regulatory requirements. Furthermore, it includes the configuration of policies, e. g., to automatize standard security choreographies such as certain sequences of encryptions and signatures. Finally, on an organizational level this should include following risk management best practices, i. e., for security certification and accreditation of the used systems [20].

2. *Operation*: one of the main goals of security is to deal with intentional misuse and attacks. Thus, dedicated security controls have to be implemented in order to realize service security solutions for different purposes.
   (a) *Prevention* works against the violation of general security goals. The Web service stack already includes various standards to address this issue.
   (b) *Detection* is important to notice when prevention is failing, i. e., when attacks occur. Classic approaches include intrusion detection systems or firewalls, a cross-organizational SOA requires further message- and service-based approaches due to its abstraction from network operations.
   (c) *Response* has to be triggered when detection was successful. It can include, e. g., the activation of additional countermeasures, attack mitigation by cutting off any access to an asset, forensics, or counter-attacks.
3. *Measurement*: security solutions have different qualitative and quantitative impacts, which can and have to be measured.
   (a) *Security Evaluation* uses various metrics to determine how effective prevention, detection, and response are. Important in this context is the concept "Quality of Protection" [21], which aims at bundling different measures for security and which is used in analogy to "Quality of Service". In addition, the side-effects of security, i. e., on performance and usability have to be taken into account and evaluated.

Our next steps will be to extend the presented analysis and to build a model for SOA-specific attack scenarios and suitable countermeasures, i. e., in the *Internet of Services* scenario. Gaining a better and structured understanding of how attacks on SOAs work, what elements they consist of, and how to respond to them allows for developing effective defense mechanisms, thus, brings us closer to safer service-based cross-organizational collaboration. Attack knowledge must not be left exclusively to attackers.

Acknowledgments

This work is supported in part by E-Finance Lab e. V., Frankfurt am Main, Germany and BearingPoint Management and Technology Consultants. Furthermore, the authors would like to thank the SOA security working group of TeleTrusT Deutschland e. V. for fruitful discussions and the anonymous reviewers for their constructive feedback.

[6] BPMN: Business Process Modeling Notation

References

1. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series). Prentice Hall PTR (11 2004)
2. Melzer, I., et al.: Service-orientierte Architekturen mit Web Services. Konzepte – Standards – Praxis. 2nd edn. Spektrum Akademischer Verlag (4 2007)
3. Papazoglou, M.P.: Service-oriented Computing: Concepts, Characteristics and Directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003. (2003) 3–12
4. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., Krämer, B.J.: Service-Oriented Computing Research Roadmap. In: Dagstuhl Seminar Proceedings on Service Oriented Computing (SOC). (2006)
5. Josuttis, N.M.: SOA in Practice: The Art of Distributed System Design (Theory in Practice). O'Reilly Media, Inc. (8 2007)
6. Newcomer, E., Lomow, G.: Understanding SOA with Web Services (Independent Technology Guides). Addison-Wesley Professional (12 2004)
7. Eckert, C.: IT-Sicherheit: Konzepte – Verfahren – Protokolle. 5th edn. Oldenbourg (11 2007)
8. Schneier, B.: Secrets and Lies: Digital Security in a Networked World. 1 edn. Wiley (1 2004)
9. Bishop, M.: Computer Security: Art and Science. Addison-Wesley Professional (12 2002)
10. Hafner, M., Breu, R.: Security Engineering for Service-Oriented Architectures. Springer, Berlin (2008)
11. Buecker, A., Ashley, P., Borrett, M., Lu, M., Muppidi, S., Readshaw, N.: Understanding SOA Security Design and Implementation. IBM Corp. (IBM Redbooks), Riverton, NJ, USA (2007)
12. Chanliau, M.: Web Services Security: What's Required to Secure a Service-oriented Architecture? White Paper (10 2006) Oracle Corp.
13. Kanneganti, R., Chodavarapu, P.: SOA Security. Manning Publications (1 2008)
14. Bundesamt für Sicherheit in der Informationstechnik: SOA-Security-Kompendium: Sicherheit in Service-orientierten Architekturen (2008)
15. Erl, T.: Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall PTR (8 2005)
16. Miede, A., Gottron, C., König, A., Nedyalkov, N., Repp, N., Steinmetz, R.: Cross-organizational Security in Distributed Systems. Technical Report KOM-TR-2009-01, Technische Universität Darmstadt (6 2009)
17. Schneier, B.: Beyond Fear: Thinking Sensibly About Security in an Uncertain World. Springer (5 2003)
18. Shirey, R.W.: Security Architecture for Internet Protocols: A Guide for Protocol Designs and Standards. Internet Draft (11 1994)
19. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems. 2 edn. Wiley (4 2008)
20. Ross, R., Swanson, M., Stoneburner, G., Katzke, S., Johnson, A.: Guide for the Security Certification and Accreditation of Federal Information Systems. National Institute of Standards and Technology (NIST) Special Publication 800-37 (5 2004)
21. Gollmann, D., Massacci, F., Yautsiukhin, A., eds.: Quality Of Protection: Security Measurements and Metrics. 1 edn. Springer (8 2006)