

Towards Automated Risk Identification in Service-Oriented Architectures

Lutz Lowis

Institute of Computer Science and Social Studies
Department of Telematics
Albert-Ludwig University of Freiburg
Friedrichstr. 50
79098 Freiburg, Germany
lowis@iig.uni-freiburg.de

Abstract: IT risk management is an important challenge for businesses and software vulnerabilities are a major source of IT risks, as the 2006 CSI/FBI Computer Crime and Security Survey [GLLR06] demonstrates. According to the survey, many companies consider it important to quantify the losses attacks against their IT systems cause but are unable to do so. In service-oriented architectures, we see a promising option of identifying the risk impact a software vulnerability has on the confidentiality, integrity, and availability of business processes. Instead of performing this identification manually, which is a time-consuming task, we present an approach of identifying the risk impact in a highly automated manner, and report on our ongoing work in this area.

1 Introduction

Quantitative information systems risk management, according to [SSp04], is one of the four grand challenges in information security. The 2006 CSI/FBI Computer Crime and Security Survey [GLLR06] also points out the importance of IT risk management, and allows an additional observation: software vulnerabilities (vulnerabilities for short) are a major source of risks to a company's IT system. Seventy percent of the respondents reacted to computer intrusions with patching, meaning that seventy percent of the responding companies had vulnerabilities which were exploited by attackers. It seems to be good news that despite this high percentage of vulnerable companies, the total dollar amount loss dropped in comparison to the last year. However, part of the reason for this drop was the *inability* of some participating companies to provide estimates of their losses. Identifying the risk impact a vulnerability (or, to be more precise, its exploitation – in the following, we omit this for brevity) would have on a business process is the basis for quantification, which allows the estimation or determination of losses. For business processes running on service-oriented architectures (SOA), we are developing a highly automated method of identifying a vulnerability's risk impact.

The risk impact a vulnerability imposes on a company is the possibility of losing the confidentiality, integrity, or availability (CIA) of the business processes running on the company's IT systems. It is a time-consuming task to determine this risk impact

manually. In a SOA, there are several layers involved (cf. figure 1), and there is a possibly large number of components within these layers. Even though both facts complicate the analysis, the business process and service information available (e.g., in form of UML [EP00] diagrams, BPMN [Obj06] diagrams, and BPEL [Org07] scripts) in a SOA can be utilized to identify a vulnerability's risk impact on business processes in a highly automated manner.

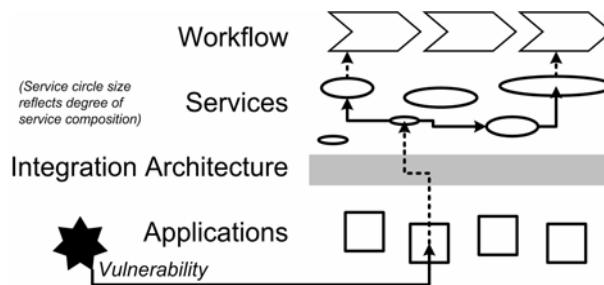


Figure 1: SOA layers and vulnerability coherence

A SOA-based business process is a workflow, which consists of a set of services and the order in which these services are called. Services in turn can either be composed of other services or directly encapsulate an application, e.g., a database, legacy software applications, or backend systems. From the vulnerability viewpoint, the logical flow follows the arrows in figure 1: vulnerabilities are found in applications, the CIA of which their exploitation can harm (for brevity, we will refer to a vulnerability's impact as the impact of its exploitation). The affected applications in turn have an impact on the dependent services, and because of service compositions the impact propagates between services. Finally, the workflow is affected through the services it directly calls. So to identify the risk impact a vulnerability has on a SOA-based business process, the identification must consider all dependencies between and within the different layers.

Based on the risk impact identification all through the application, integration architecture, service, and workflow layer (compare figure 1), it is then possible to continue with the next risk management phase and quantify the identified risks.

In the course of automating the identification phase, the next section first gives some definitions. Then, we present our four-step risk impact identification approach, with one subsection each describing the identification of vulnerabilities, their impact on applications, their impact on services, and finally, the impact on the workflow. Section 3 discusses related work, before we conclude in section 4 and point out our future work.

2 Identifying the risk impact of vulnerabilities

After giving the required definitions, we present our four-step approach of vulnerability risk impact identification.

2.1 Definitions

An **attacker** is a stranger or an employee executing malicious actions against a company's IT systems. An **attack** is the exploitation of one or more vulnerabilities, harming the security policy of the company under consideration. A **(software) vulnerability** is a flaw in an application which allows an attacker to execute an attack. The **security policy** of a company is the set of rules which specify the desired confidentiality, integrity, and availability (CIA) of that company's IT system and its parts. A vulnerability's **risk impact** is the possibility of losing the specified CIA. As long as no attack is performed, a vulnerability causes no damage. The possibility of a vulnerability's exploitation (i.e., an attack) is a risk to all business processes depending on affected applications.

Confidentiality (C). *For applications:* the attacker cannot read data being processed by the application. *For services:* the attacker does not know the exact inner workings of the service. Note that this implies that the service has not been replaced with an attacker's own version of the service. *For service messages:* the attacker cannot read the message.

Integrity (I). *For applications:* the attacker cannot change the application's behaviour. *For services:* the attacker cannot make the service produce arbitrary results. *For service messages:* the attacker cannot change the message.

Availability (A). *For applications:* the attacker cannot stop the application. *For services:* the attacker cannot stop the service from working. *For service messages:* the attacker cannot delete the message.

2.2 Step One: Vulnerability Identification

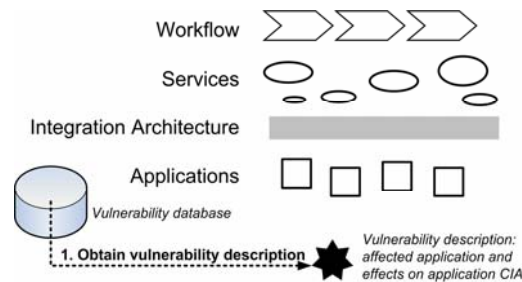


Figure 2: Vulnerability Identification

The first goal is finding vulnerabilities present in the company's IT system. The two main options here are searching for previously unknown vulnerabilities within the company or obtaining information about known vulnerabilities from inside or outside the company. Our focus here lies on determining a vulnerability's risk impact, not on finding new vulnerabilities. Therefore, we assume that vulnerability information is available either from a successful internal search or from external sources such as vulnerability databases (e.g., the NIST National Vulnerability Database (NVDB) [Nat07] with its Common Vulnerabilities and Exposures (CVE) entries, ISS X-Force [XF07], and the

Open Source Vulnerability Database (OSVDB; please note that even though the name might seem to indicate the opposite, the Open Source Vulnerability Database contains vulnerability descriptions of both open source and proprietary software) [Ope07]) (cf. figure 2).

Such vulnerability databases are updated daily and reflect the set of published, widely known vulnerabilities. Figure 3 shows a typical CVE entry in the NIST NVDB; other databases use a very similar format. Besides the self-evident unique vulnerability id, common fields are a textual description, accessibility information (remote and/or local), as well as cross references to the according vulnerability entry in other databases. Without these references, it would be an error prone and time consuming task to find information on a certain vulnerability in different databases, simply because many databases use a custom naming scheme.

Vulnerability Summary CVE-2007-5116
 Original release date: 11/7/2007, Last revised: 11/8/2007, Source: US-CERT/NIST

Overview: Buffer overflow in the polymorphic opcode support in the Regular Expression Engine (regcomp.c) in Perl 5.8 allows context-dependent attackers to execute arbitrary code by switching from byte to Unicode (UTF) characters in a regular expression.

Impact: CVSS Severity (version 2.0): CVSS v2 Base score: 10.0 (High) (AV:N/AC:L/Au:N/C:C/I:C/A:C) (legend)
 Impact Subscore: 10.0, Exploitability Subscore: 10.0

Access Vector: Network exploitable, **Access Complexity:** Low, **Authentication:** Not required to exploit

Impact Type: Provides administrator access, Allows complete confidentiality, integrity, and availability violation, Allows unauthorized disclosure of information, Allows disruption of service

References to Advisories, Solutions, and Tools
 [...] External Source: BID (disclaimer), Name: 26350, Hyperlink: <http://www.securityfocus.com/bid/26350>
 [...]

Vulnerable software and versions
 Configuration 1
 [...] Running on Debian, Debian Linux, 4.0, Powerpc
 [...]

Technical Details
 Vulnerability Type (View All) Buffer Errors (CWE-119), CVE Standard Vulnerability Entry: <http://cve.mitre.org/cgi-bin/cvename.cgi>

Figure 3: An actual CVE entry

Many vulnerability databases are online accessible and offer email notifications or RSS feeds (e.g., NIST NVDB and OSVDB). This provides the opportunity to frequently receive the latest information on newly published vulnerabilities in a machine readable format. The vulnerability descriptions obtained in this way (see bottom of figures 2 and 4) are the required input for the second step.

2.3 Step Two: Application Identification

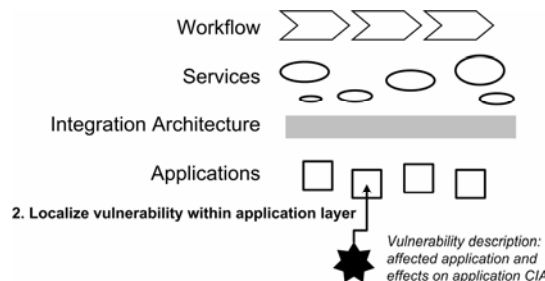


Figure 4: Application Identification

For each vulnerability identified in the first step, the question is now if this vulnerability affects any application of the company and if yes, where and how. This means the affected applications must be localized and the vulnerability's impact regarding the application's confidentiality, integrity, and availability must be determined (cf. figure 4).

The affected application's name and version can automatically be extracted from a vulnerability description. Then, there are several ways of localizing the affected applications. If the company under consideration maintains a database of all its applications, a simple database query will show if and where the vulnerability can be found within the company. If there is no database, either a network scan or a search in the log files can reveal whether the company runs the affected application or not.

How can the vulnerability's impact on application CIA be identified? The Common Vulnerability Scoring Standard (CVSS) [RF07] defines a set of metrics to score a vulnerability. These metrics describe not only the accessibility mentioned above, but also the impact of a vulnerability on the corresponding application's CIA. It is important to note that this does not equal the impact on the business process, which depends on the service composition. Some databases, for example, the NIST NVDB and ISS X-Force, include CVSS values in their entries (compare figure 3). Should the vulnerability description at hand not contain CVSS values, sometimes the cross references can be used to obtain these values from other databases.

2.4 Step Three: Service Identification

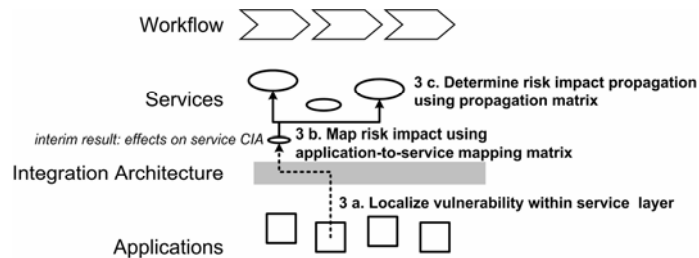


Figure 5: Service Identification

This step includes the localization of directly affected services and the examination how the vulnerability's impact propagates through the service layer. Because messages link services to each other, the impact on message CIA must be identified as well.

When using web services, the business process is often modeled in UML, as Business Process Modeling Notation (BPMN) [Obj06] diagram, or described in a Business Process Execution Language (BPEL) [Org07] script. All contain *services*, not applications. Since vulnerability descriptions typically refer to *applications* (cf. section 2.3), there is a technical challenge of mapping vulnerabilities in applications to vulnerabilities in the services directly depending on these applications. Referring to step 3a in figure 5, a logical link between the applications, the integration architecture, and the services must be created, which allows to unambiguously identify the location of the

vulnerability in both the application layer and the service layer. Implementing a solution to this challenge is one of our current tasks.

Following a conservative estimation, a vulnerability's impact on application CIA can be mapped one-to-one to the impact on service and message CIA. This is certainly appropriate for availability, because if an application is not available, the dependent service will not be available, and the dependent service will usually not generate any messages while it is not available. However, regarding confidentiality and integrity, the mapping can become more complex. For example, harming the confidentiality of an application by reading a database's content does not necessarily imply insight into the dependent service (harming the confidentiality of the service), or being able to read all messages the service generates (harming the confidentiality of the messages). We are currently developing a mapping matrix for this challenge (corresponds to step 3b in figure 5). For the rest of this paper, we assume the conservative one-to-one mapping. Please note that this mapping is different from the vulnerability effect propagation discussed later, because the effect mapping takes place between two SOA layers (one application and the directly affected services), whereas the effect propagation happens within one SOA layer (between a service and all dependent services).

Having identified the services which directly depend on the vulnerable application, and also how their CIA is influenced, the impact propagation must be determined (cf. step 3c in figure 5). In order to automate this determination, it must be formalized how breaches of confidentiality, integrity, and availability propagate between services. One possible formalization is the propagation matrix we suggest in figure 6.

Propagation effect Called service has ...	Within service layer	
	Outgoing messages lose	Calling service loses
no Confidentiality	C	-
no Integrity	I	I / A / -
no Availability	A	A / -

Figure 6: Vulnerability effect propagation matrix

The first column of figure 6 contains the effects a vulnerable service has on its outgoing messages: if a service loses its confidentiality, integrity, or availability, the messages that service creates are affected in the same way.

In the second column, the effects on the calling service are shown, which are none regarding confidentiality, because reading the input to a service does not tell the attacker how the service works. A breach of message integrity does only propagate if the calling service cannot detect that breach. If the breach is detected but cannot be fixed, the calling service's availability is harmed. If the calling service is able to detect the integrity breach and fix it, the effect is none. Regarding availability, the effect on the calling service is none only if the breach is detected and a substitute service can be called.

Based on the vulnerability localization, the effect propagation can now automatically be determined by creating a bottom-up propagation path. Starting with a service which is vulnerable according to the vulnerability localization, this service is assumed as “called service” and dependent services are regarded as “calling services” (cf. figure 6). Now, for each of the three possible effects (loss of confidentiality, integrity, and availability), the propagation matrix is used to find out what the effect on the calling service is. Then, the procedure is repeated for every calling service in its new role as called service. In the end, all services which directly or indirectly depend on the vulnerable service have been identified, showing what the vulnerability's impact on service CIA is.

2.5 Step Four: Workflow Identification

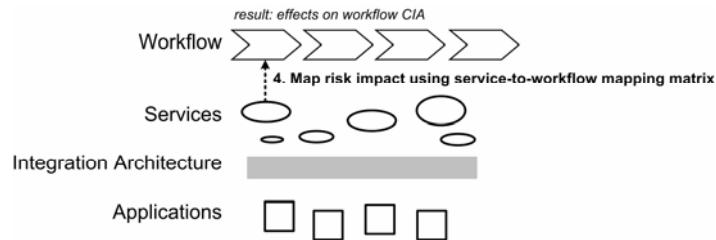


Figure 7: Workflow Identification

Finally, the risk impact arising from the affected services needs to be calculated. This can be achieved by mapping the results of the third step, which represent the vulnerability's impact on the service layer, to the workflow layer (cf. figure 7). Again, as in section 2.4, a mapping matrix is one possible solution. In figure 8, we suggest such a matrix.

Mapping \ Service has ...	Service to workflow layer	
	Local workflow loses	Global workflow loses
no Confidentiality	C	C
no Integrity	I / I and A	I / I and A / -
no Availability	A / -	A / -

Figure 8: Service-to-workflow-layer mapping matrix

Based on the mentioned vulnerability information on the one hand, and using business process models with their web service composition description on the other, the complete approach then works as follows. First, the affected application and the security effects on that application are extracted from a vulnerability description (compare section 2.2). Then, that precise application is localized within the application layer by using, e.g., application name and version number to differentiate between applications (as described in section 2.2). In the next step, the vulnerability can be mapped through the integration layer to the service layer by means of a deployment diagram or similar information (grey area in figure 7). Following the service identification (see section 2.4),

the risk impact of the vulnerability under consideration is determined regarding the workflow by using a mapping matrix such as the one in figure 8.

According to our hitherto observations, confidentiality is the most delicate security property in SOA-based business processes, because once it is lost in a service, the whole workflow loses it. Availability is a more agreeable property in this sense, because its loss can rather easily be detected and it can be maintained by involving substitute services. Integrity seems to be the most complex property, because it can be maintainable if the calling service can detect and fix the loss of integrity (e.g., by sending a second request), but it can also lead to a loss of availability if its breach is detected without having the possibility to fix it.

In a SOA, the workflow is the business process, which means that after identifying the risk impact on workflows (or workflow parts), the risk impact identification is finished.

3 Related Work

The effects a vulnerability's exploitation has can be determined by penetration testing [Bis03] the production system. Because this can easily interrupt the production and incur high costs for restoring the production system, penetration tests on a duplicate system might seem to be a better solution. However, this still incurs costs for creating a duplicate system. Completely duplicating all web services might simply be infeasible, and separately testing single services does not allow an automated determination of the vulnerability effect propagation. Following the SOA approach, a model of the business process and its services is essential. This model (e.g., in BPMN or BPEL) or a derivation thereof can be used to identify the business process impact of a vulnerability's exploitation, allowing for an inexpensive, realistic, and complete approach: Inexpensive, because the model is already available, realistic, because the model itself is executed in terms of BPEL scripts, and complete, because the whole composition is examined rather than separate services. For these reasons, we suggest a model-based approach.

Vulnerability taxonomies such as [Krs98] and [BRP05], depending on their focus, offer a comprehensive explanation of how vulnerabilities come into existence, which different vulnerability types exist, and which attacks can be performed through them. While a suitable taxonomy of web service vulnerabilities could be used to develop a vulnerability effect mapping matrix, to our knowledge at this time no such taxonomy of web service vulnerabilities and their effects on web services exist.

Vulnerability description standards such as the Application Vulnerability Description Language (AVDL) [Org04] and the Open Vulnerability and Assessment Language (OVAL) [Cor07], focus on describing *how* a vulnerability can be exploited. This is useful when testing for or trying to fix a vulnerability. However, regarding the *effect* or risk impact of a vulnerability, the Common Vulnerability Scoring Standard values (CVSS) mentioned in section 2.2 are more relevant.

The approach of estimating potential IT security losses presented in [LS06] uses port scan data as its basis. While this is useful to estimate the general network risk situation of a company, it does not allow for a precise determination of which business processes are facing what risk regarding their CIA. However, such a coarse grained approach could be used to trigger the risk impact identification approach we have presented, so that the identification is performed more often when the network risk seems to be high.

Powerful analysis approaches and tools exist to check source code or running applications for vulnerabilities (e.g., [MLL05], [CM04], [WKP05]). The vulnerabilities found through such analysis tools serve as input to our approach, e.g., after being published in a vulnerability database.

UMLsec [Jan04] is an extension of the UML for secure systems development. It can be used to identify vulnerabilities in, e.g., networks and cryptographic protocols, which many applications rely on. The attack graph generation approach in [OBM06] can be used to find new attacks in enterprise networks. As a side effect, new vulnerabilities might be identified. Again, the vulnerabilities found through such approaches provide the vulnerability information input to the risk impact determination for SOA-based business processes we have presented.

4 Conclusion and Future Work

IT risk management requires the identification of all risks a company's IT system faces. Software vulnerabilities put SOA-based business processes at risk, because they affect applications, which affect services, which affect the workflow. The workflow in turn represents the business process, so that identifying the risk impact a vulnerability's exploitation would have on the workflow means identifying the risk impact on the business process. With the approach we have presented, it is possible to identify this impact in a highly automated manner. This allows for frequently updated risk identification data, which is necessary to maintain an up-to-date overview of the risks to a company's IT system, or, from another viewpoint, a company's business processes. By performing an IT risk quantification on these identification results, a company can produce estimates on the losses an attack on its IT systems might have or has had.

Our future work consists of implementing the application, service, and workflow identification steps described in sections 2.3 to 2.5. We are currently implementing the application identification, which has as input a vulnerability description and outputs the location of the affected applications and the vulnerability's impact on their CIA. Also, we are examining which type of model (e.g., BPMN or custom UML diagram) is best suited for the service and workflow identification; our current favorites are activity and deployment diagrams. If our analysis shows that considering risk impact propagation within the application layer and/or within the workflow layer would increase the precision of our approach, we will also develop propagation matrices for these layers, similar to the one presented in figure 6.

References

- [Bis03] Bishop, M: Computer Security. Addison-Wesley, Pearson Education, Boston, 2003.
- [BRP05] Vanden Berghe, C.; Riordan, J.; Piessens, F.: A Vulnerability Taxonomy Methodology applied to Web Services. In: Proceedings of the 10th Nordic Workshop on Secure IT Systems, 2005.
- [CM04] Chess, B.; McGraw, G.: Static Analysis for Security. IEEE Security and Privacy, vol. 2, no. 6, 2004. P. 76-79
- [Cor07] MITRE Corporation: Open Vulnerability and Assessment Language (OVAL). 2007. <http://oval.mitre.org/oval/index.html> (last accessed Oct. 2, 2007)
- [EP00] Eriksson, H.E.; Penker, M.: Business Modeling With UML. John Wiley and Sons, Inc., New York, 2000.
- [GLLR06] Gordon, L.A.; Loeb, M.P.; Lucyshyn, W.; Richardson, R.: CSI/FBI Computer Crime And Security Survey. Computer Security Institute, 2006.
- [Jan04] Jürjens, J.: Secure Systems Development with UML. Springer, 2004.
- [Krs98] Krsul, I.V.: Software Vulnerability Analysis. Ph.D. Thesis, Purdue University, 1998.
- [LS06] Lee, V.C.S.; Shao, L.: Estimating Potential IT Security Losses. IEEE Security and Privacy, vol. 4, no. 6, 2006. P. 44-52
- [MLL05] Martin, M.; Livshits, B.; Lam, M.S.: Finding Application Errors and Security Flaws Using PQL: a Program Query Language. In: 20th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, San Diego, California, USA, 2005.
- [Nat07] National Institute of Standards and Technology (NIST): National Vulnerability Database. 2007. <http://nvd.nist.gov/nvd.cfm> (last accessed Oct. 2, 2007)
- [Obj06] Object Management Group (OMG): Business Process Modeling Notation (BPMN). 2006. <http://www.bpmn.org/> (last accessed Oct. 2, 2007)
- [OBM06] Ou, X.; Boyer, W.F.; McQueen, M.A.: A Scalable Approach to Attack Graph Generation. In: Proceedings of the Conference on Computer and Communications Security, Alexandria, Virginia, USA, 2006.
- [Ope07] Open Source Vulnerability Database (OSVDB): Open Source Vulnerability Database. 2007. <http://osvdb.org> (last accessed Oct. 2, 2007)
- [Org04] Organization For The Advancement Of Structured Information Standards (OASIS): Application Vulnerability Description Language (AVDL). 2004. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=avdl (last accessed Oct. 2, 2007)
- [Org07] Organization For The Advancement Of Structured Information Standards (OASIS): Business Process Execution Language (BPEL). 2007. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel (last accessed Oct. 2, 2007)
- [RF07] Forum Of Incident Response And Security Teams (FIRST): Common Vulnerability Scoring System (CVSS). 2007. <http://www.first.org/cvss> (last accessed Oct. 2, 2007)
- [SSp04] Smith, S.W.; Spafford, E.: Grand Challenges in Information Security: Process and Output. IEEE Security and Privacy, vol. 2, no. 1, 2004. P. 69-71
- [WKP05] Weber, S.; Karger, P.A.; Paradkar, A.: A Software Flax Taxonomy: Aiming Tools At Security. In: Software Engineering for Secure Systems, St. Louis, Missouri, USA, 2005.
- [XF07] IBM Internet Security Systems X-Force: Alerts and Advisories. 2007. <http://xforce.iss.net/xforce/alerts> (last accessed Oct. 2, 2007)