SecEval: An Evaluation Framework for Engineering Secure Systems*

Marianne Busch, Nora Koch, Martin Wirsing
Ludwig-Maximilians-Universität München
Oettingenstraße 67, 80538 München, Germany
{busch, kochn, wirsing}@pst.ifi.lmu.de

Abstract: Engineering secure software systems is not an easy task. Many methods, notations and tools – we call them knowledge objects – exist to support engineers in the development of such software. A main problem is the selection of appropriate knowledge objects. Therefore, we build the conceptual framework SecEval to support the evaluation and comparison of security features, vulnerabilities, methods, notations and tools. It provides an evaluation process and a model, which comprises concepts related to security context, data collection and data analysis. Our approach is validated by a case study in the area of security testing of web applications.

1 Introduction

Software and security engineers constantly make decisions about which technology should be used in the different phases of the Software Development Life Cycle (SDLC). Therefore, a cost-benefit analysis and a subsequent selection of appropriate methods, tools and notations – so called knowledge objects (KOs) – for a specific task, play an important role in the engineering process. All too frequent, there is no time to investigate on alternatives to well-known KOs or those used so far. Most of the questions which arise are not entirely new, but useful scraps of knowledge are distributed in papers, books or the web, or just exist in the head of colleagues working at another project. Without having a template for their domain, engineers often have to start defining the process of evaluating KOs as well as creating the structure of the results from scratch.

To ease the tasks of recording results and of getting an overview of existing KOs the Common Body of Knowledge (CBK) [CBK13] was implemented as a semantic Wiki within the scope of the EU project NESSoS. As members of the NESSoS project, we gained experience working with this knowledge base and its underlying model, which raised three questions: (a) How could the CBK's model be improved, so that security-related features can also be represented as knowledge objects? (b) How can we use the model not only for recording and comparing features of methods, notations and tools, but also for documenting the search process. (c) How is the process of data collection and data analysis specified, to make sure that emerging research results are comprehensible and valid?

*This work has been supported by the EU-NoE project NESSoS, GA 256980.

Evaluation in the area of cybersecurity does mean for us, e.g., to find out which authentication-related threats can or cannot be mitigated by a method, for which tool-support is implemented. Up to now, these kinds of questions require researchers to document their approaches and results in a self-made way. Consequently, other researchers, who want to build on those results, have to understand many different schemas documenting research processes and their results. This is not only time-consuming, but also error-prone, as misunderstandings easily occur.

We present an evaluation approach, called SecEval, for evaluating security-related KOs. However, we do not claim to provide a one-fits-all model for IT-security (which would horribly overload any model), but introduce an extensible basis. SecEval defines a graphical model, which comprises (a) a security context model describing security properties, vulnerabilities and threats as well as methods, notations and tools; (b) a data collection model, which records how data is gathered when researchers or practitioners do research to answer a question; and (c) a data analysis model specifying, how reasoning on previously collected data, is done.

A simplified example of the process of using SecEval for evaluation is depicted in Fig. 1. Research questions initiate the process of data collection, where sources (as papers, websites, . . .) are gathered. These sources are then analyzed, which means to extract information and record it using SecEval's security context model.

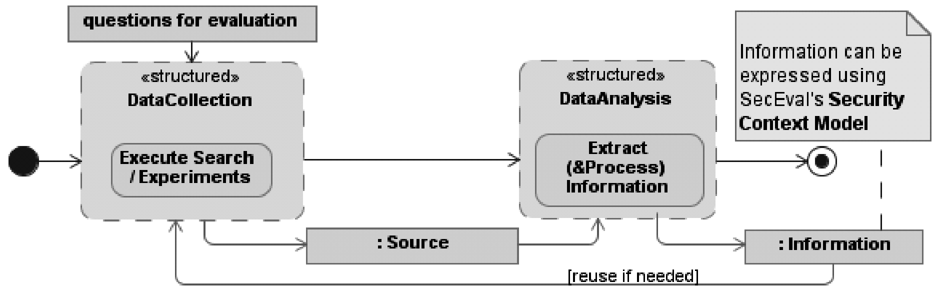

questions for evaluation
«structured»
DataCollection
Execute Search / Experiments
«structured»
DataAnalysis
Extract (&Process) Information
Information can be expressed using SecEval's **Security Context Model**
: Source
: Information
[reuse if needed]

Figure 1: Overview of SecEval's Evaluation Process (full process: [Bus14a])

The remainder of this paper is structured as follows: Sect. 2 presents our evaluation approach called SecEval. In Sect. 3 we validate the approach by a guided review and a case study in the area of security testing of web applications. We discuss related work in Sect. 4 and conclude in Sect. 5.

2 Evaluation Framework SecEval

Our aim is to provide an approach for documenting the evaluation of methods, notations and tools within the scope of secure software systems. The evaluation should also support security properties, vulnerabilities and threats. For the graphical representation of concepts and relationships we selected the UML notation, as we think it fits our needs best.

The full MagicDraw 17.0[1] model of SECEVAL and all diagrams can be downloaded from the web [Bus14a]. Deliverable D2.4 [BK13, Bus14b] of the NESSoS project includes a detailed description of SECEVAL.

We elicited the requirements of such a conceptual framework, i.e. which stakeholders are involved (security engineers, users, attackers), which use cases they perform, which concepts play a role and how they are related. We grouped the identified use cases according to evaluation (e.g., collect data) and SDLC-related (e.g., identify vulnerabilities) concepts.

The use cases from our requirements analysis were a starting point to identify relevant concepts related to security for using and evaluating methods, notations and tools during the software engineering process. We clustered these concepts in three packages: Security Context, Data Collection and Data Analysis. Figure 2 shows the model represented as a UML class diagram.

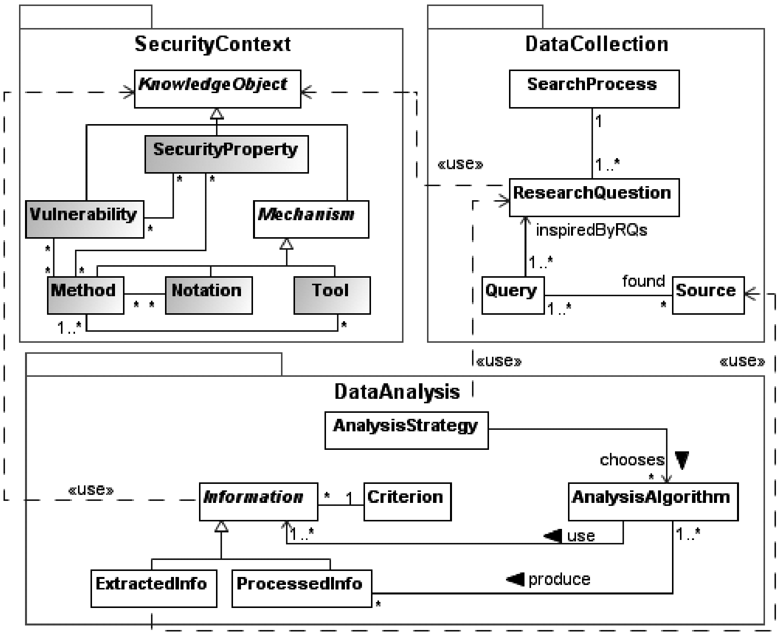


Figure 2: SECEVAL: Model Overview

2.1 Security Context

The aim of Security Context package (shown in Figure 3) is to provide a structure for the classification of (security-related) methods, notations and tools together with security properties, vulnerabilities and threats. We introduce an abstract class `Mechanism` from which the classes `Method`, `Notation` and `Tool` inherit common attributes such as

[1]MagicDraw. `http://magicdraw.com`

`goals`, `costs`, `basedOnStandards`, etc. In this paper we use the upper-case term "Mechanism" when referring to a method, a notation or a tool. We focus on security aspects, but the model can also record non-security Mechanisms.

Once Mechanisms are described by the model, it is easy to get an overview of existing security-related methods, tools and notations for a certain area. Furthermore, the package should serve as a flexible basis for a knowledge base and as a starting point for an evaluation. This means that it can be adopted to fit the needs of the researcher to examine a concrete research question (which does not have to be scientific).

In Fig. 3, for convenience enumerations' texts are grey and the background of classes which can directly be instantiated is colored. All attributes and roles are typed; however the types are not shown in the figures due to brevity. The main characteristics of Mechanisms are specified as boolean types (can.., has.., is..). In an implementation of our model, it should be possible to add further items to enumerations.

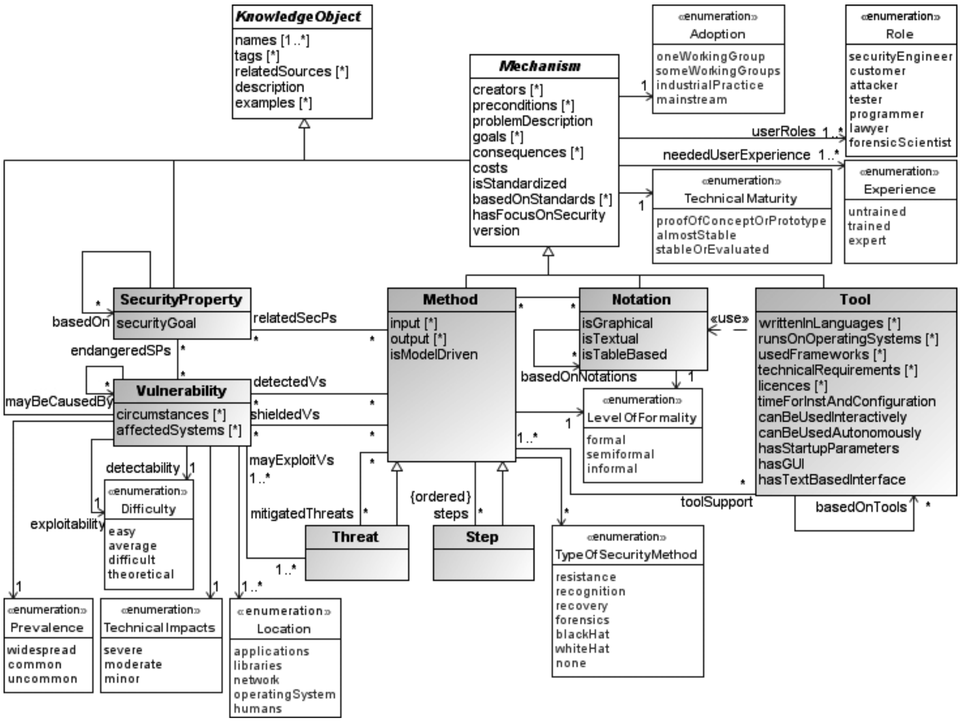


Figure 3: SECEVAL: Security Context

As mentioned above, a MECHANISM is an abstract notion for a method, notation or tool. It can be described by a problem statement, by the goals it strives for, by its costs and by the consequences it implies. Mechanisms can be based on standards or be standardized themselves. They can have arbitrary many creators, as companies, inventors or developers. Before applying a Mechanism, the preconditions that are necessary for using it have to be fulfilled. Furthermore, an estimation regarding technical maturity and adoption in practice should be given. Several levels of usability can be stated according to the experience a user needs to employ a Mechanism, e.g., a certain Mechanism should best be applied by experts.

A METHOD has some general attributes, such as as input, output and if it is model-driven, which are used to describe the method at a high level of abstraction. For extensive methods, each step of the method can also be described in detail, if necessary. A method or step can be supported by notations or tools.

For a NOTATION, we consider characteristics such as whether the notation is graphical, textual or based on a tabular representation. We also added a level of formality, which ranges from informal to formal. Notations can be based on other notations, for example many context-specific extensions for UML exist.

The description of a TOOL covers the information of languages it is written in, of operating systems it supports, of frameworks it uses and of technical requirements, which have to be fulfilled in order to use it. The tool (or its parts) are released under certain licenses. Additionally, the needed time for installation and configuration can be provided. Booleans describe if the tool can be used interactively or autonomously, if it has start parameters, a GUI or a text-based user interface. A tool can be based on other tools, which is the case when libraries are used or when plugins are written.

During our experience with the CBK, we noticed that tools as well as methods would be better described according to the phases of the SDLC, because attributes which are used to describe a method or tool are related to the SDLC phases they cover. As far as we know, no phase-related attributes are needed to describe features of notations. Figure 4 depicts our `Method` class and the abstract class `MAreasOfDev`, which is a wildcard for detailed information about the method. A method can support several development phases. The phases of the SDLC are the same we have chosen to classify tools and methods in the NESSoS project [BK11]: requirements, design, implementation, testing, assurance, risk & cost, service composition and deployment. We added an additional category to distinguish methods and tools that operate at the runtime of a system.

For example a method, as e.g., Microsoft's Security Development Lifecycle[2], can be used as a basis for designing secure applications, but also covers other phases. In this case, the attributes of the classes `DesignM` and `ImplementationM` and others would be used to describe the method. The meaning of attributes should be self-explaining, for further details and for the according SDLC refinement for tools, the reader is referred to [Bus14a].

We adopted the abstract KNOWLEDGEOBJECT (KO) which is used in the CBK to record most information of elements which are described. For SECEVAL, we applied separation of concerns so that only very general descriptions remain as attributes in a KO, which can be applied to all elements (cf. Fig. 3). Therefore, the class `KnowledgeObject` has names, tags and related sources, which could be any kinds of sources, as publications or URLs. A description and examples enable easy learning of KOs, i.e. security properties, vulnerabilities and mechanisms.

We represent security issues, such as confidentiality, integrity and privacy by the class

[2]Microsoft SDL. `https://www.microsoft.com/security/sdl`

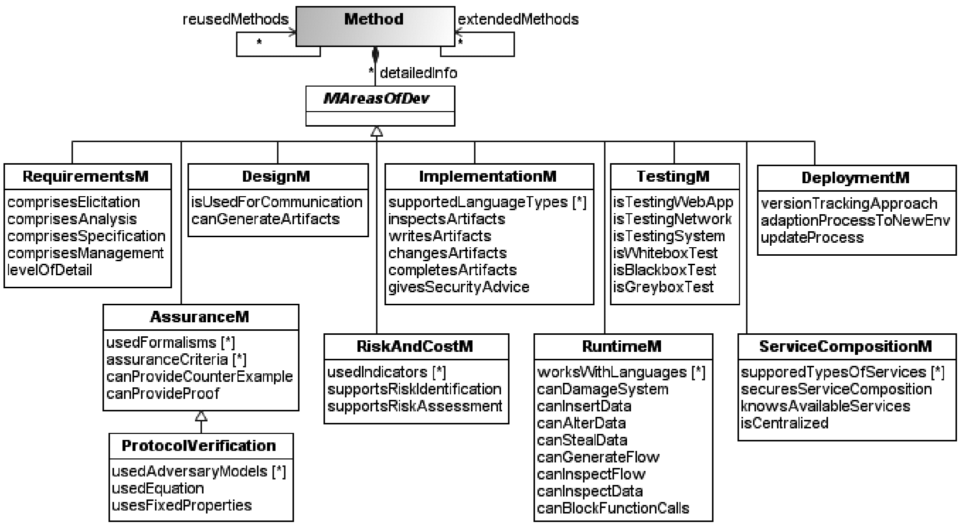


Figure 4: SECEVAL's Security Context: Details of Methods

SECURITY PROPERTY. The attribute `SecurityGoal`, which is denoted by a string, describes the goal of the property. For instance "integrity refers to the the trustworthiness of data or resources" [Bis02, p.5].

A VULNERABILITY is "a weakness that makes it possible for a threat to occur" [Bis02, p.498]. Thus, it endangers security properties. Examples are XSS, SQL Injection, Buffer Overflows, etc. Methods can detect such vulnerabilities or shield them from being exploited by a threat. Every vulnerability is located at least in one location (which is modeled as a UML enumeration). Furthermore, we include the categorization scheme from OWASP TOP 10 [Fou13b] (which is adapted from the OWASP Risk Rating Methodology [Fou13a]) using prevalence, impact level, detectability and exploitability. Regarding the latter two roles, the `Difficulty` "theoretical" means that it is practically impossible to detect or exploit a vulnerability (cf. Figure 3).

A THREAT is "a potential occurrence that can have an undesirable effect on the system assets or resources" [Bis02, p.498]. We treat a threat as a kind of method which is vicious. At least one vulnerability has to be affected, otherwise a threat is not malicious (and the other way around), which is denoted by the multiplicity [1..*]. Additionally, threats can be mitigated by other methods.

2.2 Data Collection

High-quality data is the basis for an evaluation, as the best analysis strategy cannot make up for low-quality data. Our aim is to create a schema which describes properties that have to be defined before starting collecting data. Such an approach is particularly needed, if the data collection has to be systematic. Therefore, we base our approach on Kitchenham's systematic literature review [KC07].

In order to collect data, it is common to define a search process (c.f. Fig. 5) which specifies several steps called process phases. Each phase may follow another approach, e.g., the search can be automated or not, or it can be a depth-first or a breadth-first search. Depth-first means, that the aim of a search is to extract a lot of detail information about a relatively small topic, whereas a breadth-first search is good to get an overview of a broader topic.

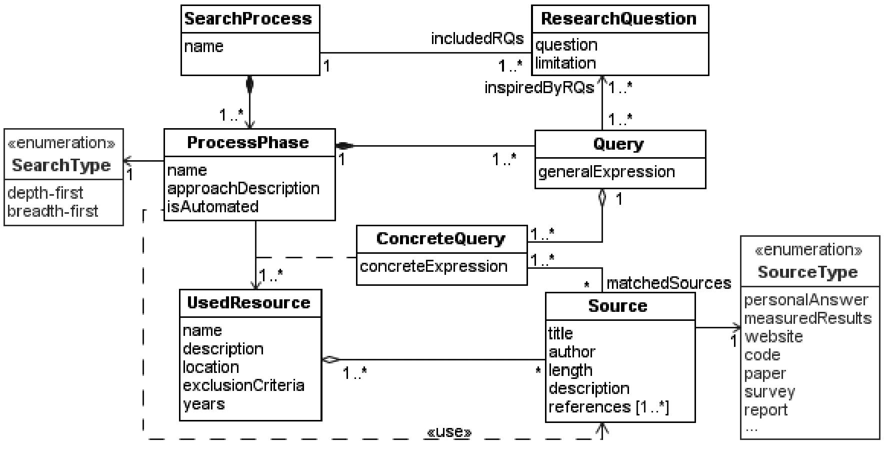


Figure 5: SECEVAL: Data Collection

Similar to Kitchenham's literature review, research questions are used to define the corner stones and the goals of the search. Please note that for us the term "research" does not necessarily refer to *scientific* research. Queries can be derived from the research questions. They are then used and refined in the phases of the search process. As different search engines support different types of queries, concrete queries are specific for each resource, as e.g., Google Scholar. Queries can also refer to questions which are used as a basis for experiments (cf. Sect. 3).

It is important to choose resources that will serve as data sources for the evaluation. The use of an association class for `ConcreteQuery` (depicted by a dashed line) denotes that for each pair of `ProcessPhase` and `UsedResource`, the class `ConcreteQuery` is instantiated. The concrete search expression is derived from a general search expression.

For example, the general search expression could be "recent approaches in Security Engineering" and we want to ask Google Scholar and a popular researcher. For Google Scholar we could use ""Security Engineering" 2012..2013" as a concrete search expression and the concrete expression for asking a researcher could read: "I'm interested in Security Engineering. Which recent approaches in Security Engineering do you know?".

If a concrete query matches sources, as papers, websites or personal answers, we classify the source at least by author and description (as an abstract) and provide information about the type of source and at least one reference where to find it. The process of data collection and data analysis is depicted in Fig. 1.

2.3 Data Analysis

Data is collected with the purpose to obtain an answer to research questions based on the analysis of the data. According to Kitchenham, the procedure how to collect as well as analyze data belongs to the "review protocol" and has to be specified in the first place.

Figure 6 depicts relevant concepts for analyzing data. First, we have to specify which type of strategy we want to use. Are we limited to quantitative analysis or do we focus on qualitative analysis? Accordingly, one can later refer to Kitchenham's checklists for quantitative and qualitative studies [KC07, tables 5 and 6] to ensure the quality of the own answers to the research questions.

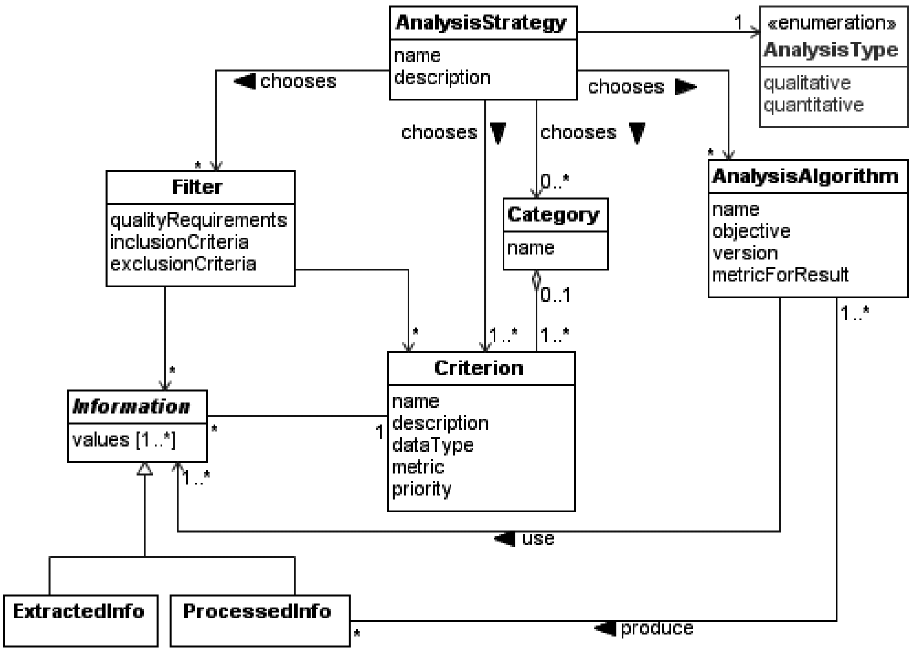


Figure 6: SECEVAL: Data Analysis

The analysis strategy requires to select the used categories & criteria, algorithms for analysis, and filters according to the research question. Criteria can be grouped by categories.

A criterion gives more information about data values as it defines the data type (string, list of booleans, ..) and the metric (milliseconds, ..). In addition, a priority can be defined which is useful when Mechanisms should be compared.

Information can be extracted from the sources which were found in the data collection phase (see ≪use≫ dependency starting from the class `ExtractedInfo` in Fig. 2), or they can be processed using an analysis algorithm. This algorithm does not have to be executable on a computer. The analysis strategy defines which algorithm is employed and makes sure that the result of the algorithm fits to a criterion regarding meaning and metric.

Besides, a filter can be specified to disqualify results according to certain criteria as costs or quality. This filter is finer grained than the filter that is defined by `UsedResource`'s attribute `exclusionCriteria` used in the data collection, which only can be based on obvious criteria, as e.g., the language the source is written in. In addition to this, the filter for data analysis accesses information as well as criteria and thus can exclude, e.g., Mechanisms from the evaluation that do not meet a high-priority requirement.

A valid question is how information, criteria and the security context model fit together. This is shown in Fig. 2: information can be stored in an instance of our security context model, which provides a sound basis when collecting data about KOs. Consequently, the attributes `name` and `dataType` of a `Criterion` can be left blank when information is stored in an instance of our model, as attributes have a name and are typed. However, these attributes are needed when describing information which is not directly related to an instance of a knowledge object or not meaningful without their connection to a concrete analysis process.

In summary, it can be said that, contrary to the context model, neither the collection of data nor the data analysis are security specific and thus can be applied in the same way to other domains.

3 Validation of SecEval

Coming up with a broad evaluation model for security KOs is challenging, because many different areas of expertise are needed. For validating and improving SecEval we conducted a guided interview with project partners, who encompass the broad area of secure software development. Besides, we performed a case study on security testing of web applications using SecEval.

3.1 Guided Interview

A Guided Interview is "a one-on-one directed conversation with an individual that uses a pre-determined, consistent set of questions but allows for follow-up questions and variation in question wording and order."[3] We hold this kind of interview in a slightly modified way: first we explained our basic model (especially the basic Security Context Model). Second, we handed out a description of the draft version of SecEval and a questionnaire, which can be found in [Bus14a]. Finally, 14 international senior researchers, who are experts in different areas of security engineering, gave us feedback.

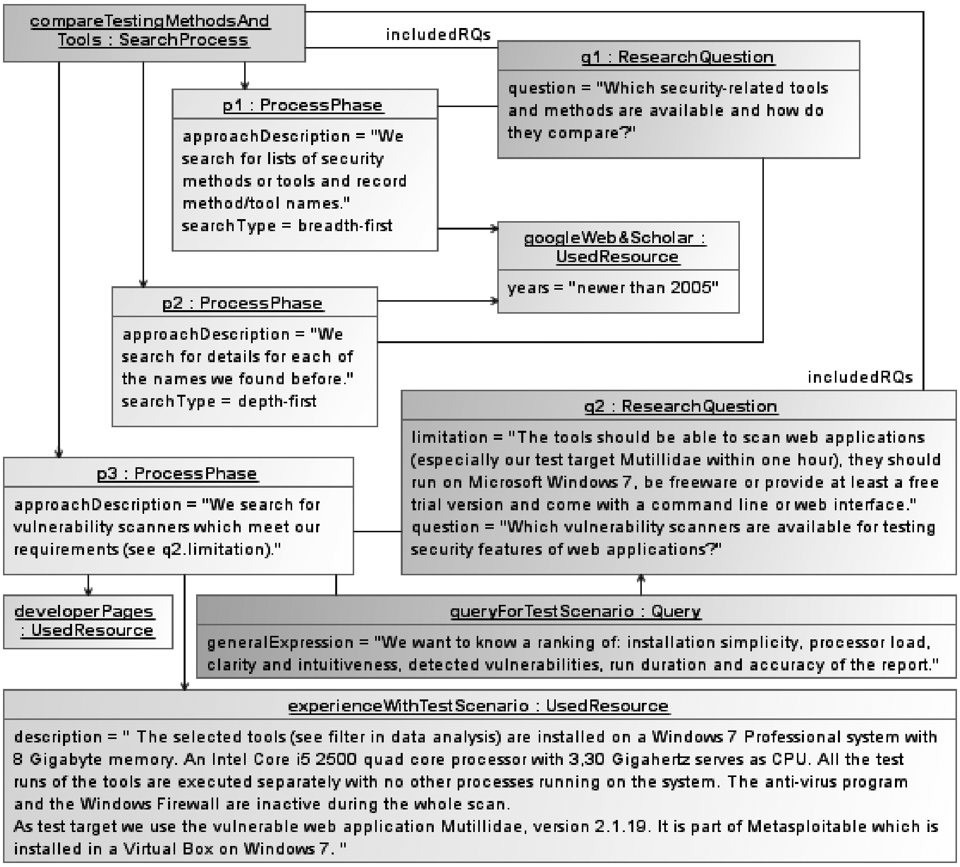
The answers and discussions helped us to improve SecEval. Among other changes, further attributes were added and some classes and enumerations were splitted to emphasize the idea of separation of concerns. In addition, we extended SecEval for risk rating and experimental approaches [Bus14a].

[3] Education dictionary. http://www.mondofacto.com/facts/dictionary?guided+interview

3.2 Case Study

With 27% of breaches within hacking, web applications of larger companies are a worthwhile target for hackers [Ver13, p.35]. An approach to harden web applications is to identify security flaws through "penetration testing" or "vulnerability scanning". These methods are supported by many commercial and open-source tools. In this section, we use our SECEVAL approach to evaluate vulnerability scanners for web applications.

Data Collection The first step consists in defining the plan to collect data. This is done by an instance model as shown in Fig. 7, which depicts instances of the classes we have already defined in Fig. 5. For example, instances of the class `ResearchQuestion` define the two research questions, a high-level and a concrete one. We used identical background colors for instances of the same classes and omitted all `name` attributes in case a name (e.g., `p3`) is given in the header of an instance.



compareTestingMethodsAnd Tools : SearchProcess
includedRQs
q1 : ResearchQuestion
p1 : ProcessPhase
googleWeb&Scholar : UsedResource
p2 : ProcessPhase
includedRQs
q2 : ResearchQuestion
p3 : ProcessPhase
developerPages : UsedResource
queryForTestScenario : Query
experienceWithTestScenario : UsedResource

q1 : ResearchQuestion
question = "Which security-related tools and methods are available and how do they compare?"

p1 : ProcessPhase
approachDescription = "We search for lists of security methods or tools and record method/tool names."
searchType = breadth-first

googleWeb&Scholar : UsedResource
years = "newer than 2005"

p2 : ProcessPhase
approachDescription = "We search for details for each of the names we found before."
searchType = depth-first

q2 : ResearchQuestion
limitation = "The tools should be able to scan web applications (especially our test target Mutillidae within one hour), they should run on Microsoft Windows 7, be freeware or provide at least a free trial version and come with a command line or web interface."
question = "Which vulnerability scanners are available for testing security features of web applications?"

p3 : ProcessPhase
approachDescription = "We search for vulnerability scanners which meet our requirements (see q2.limitation)."

queryForTestScenario : Query
generalExpression = "We want to know a ranking of: installation simplicity, processor load, clarity and intuitiveness, detected vulnerabilities, run duration and accuracy of the report."

experienceWithTestScenario : UsedResource
description = " The selected tools (see filter in data analysis) are installed on a Windows 7 Professional system with 8 Gigabyte memory. An Intel Core i5 2500 quad core processor with 3,30 Gigahertz serves as CPU. All the test runs of the tools are executed separately with no other processes running on the system. The anti-virus program and the Windows Firewall are inactive during the whole scan.
As test target we use the vulnerable web application Mutillidae, version 2.1.19. It is part of Metasploitable which is installed in a Virtual Box on Windows 7. "

Figure 7: Case Study: Data Collection

Research question `q1` ("Which security-related tools and methods are available and how do they compare?", cf. Fig. 7) is very general. In the first process phase `p1`, 13 methods and 18 tools were selected [Sch13]. More detailed information was gathered in the second process phase `p2` about: vulnerability scanning, penetration testing, fuzzing and the classification into black- grey- and white-box testing. Examples for tools are WSFuzzer, X-Create and WS-Taxi, just to mention a few. As we already added most of the found methods and tools to the CBK [CBK13], we focus on `q2` in this section.

Research question `q2` ("Which vulnerability scanners are available for testing security features of web applications?") is a typical question which could be asked by security engineers working in a company. The "sources" (i.e., tools) we selected for analysis were [Lac13]: a) Acunetix Web Vulnerability Scanner[4], b) Mavituna Security - Netsparker[5], c) Burp Scanner[6], d) Wapiti[7], e) Arachni[8], f) Nessus[9], g) Nexpose[10] and h) Nikto[11].

The instance `experienceWithTestScenario` describes how the data is gathered by testing the vulnerability scanners.

Data Analysis For analyzing collected data we define an analysis strategy and select a filter which enforces the requirements (`limitations`) defined for question `q2`. Figure 8 depicts instances of the data analysis model we defined in Fig. 6.

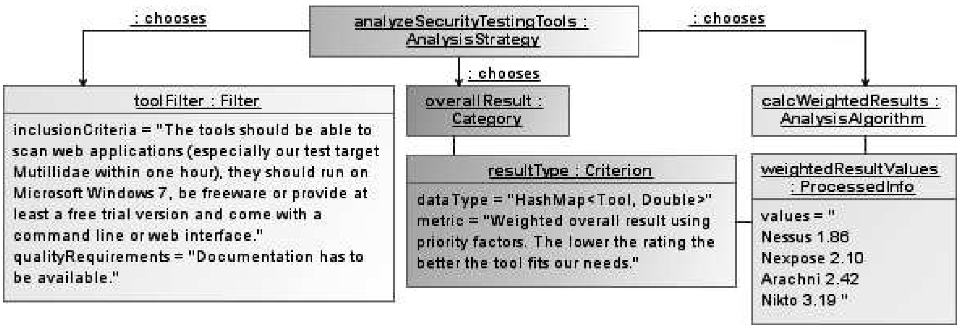

analyzeSecurityTestingTools : AnalysisStrategy
: chooses
: chooses
: chooses
toolFilter : Filter
inclusionCriteria = "The tools should be able to scan web applications (especially our test target Mutillidae within one hour), they should run on Microsoft Windows 7, be freeware or provide at least a free trial version and come with a command line or web interface."
qualityRequirements = "Documentation has to be available."
overallResult : Category
resultType : Criterion
dataType = "HashMap<Tool, Double>"
metric = "Weighted overall result using priority factors. The lower the rating the better the tool fits our needs."
calcWeightedResults : AnalysisAlgorithm
weightedResultValues : ProcessedInfo
values = "
Nessus 1.86
Nexpose 2.10
Arachni 2.42
Nikto 3.19 "

Figure 8: Case Study: Data Analysis – Results

Before going into detail about particular results of our experiments, we first take a look at the overall result regarding our research question `q2`. Figure 8 thus depicts an instance of the class `ProcessedInfo`, which is called `weightedResultValues`. Only four tools passed our filter: Arachni and Nikto, which provide command-line interfaces and Nessus and Nexpose, which also provide web interfaces. From our list of tools from

[4]Acunetix. `http://www.acunetix.com`
[5]Netsparker. `https://www.mavitunasecurity.com/netsparker`
[6]Burp Scanner. `http://portswigger.net/burp/scanner.html`
[7]Wapiti. `http://www.ict-romulus.eu/web/wapiti`
[8]Arachni. `http://www.arachni-scanner.com`
[9]Nessus. `http://www.tenable.com/de/products/nessus`
[10]Nexpose. `https://www.rapid7.com/products/nexpose`
[11]Nikto. `http://www.cirt.net/Nikto2`

above, the trial of $a)$ only allows to scan predefined sites. Tools $b)$ and $c)$ do not support a command line or web interface in the versions that are free. A run of tool $d)$ on our test target Multidae[12] took six hours.

Apart from information available online, we experimented with the tools that passed the filter, in order to obtain data for our tool evaluation (`q2`). We evaluate the following (weighted it as indicated in the brackets, cf. `queryForTestScenario`): installation simplicity [0.5], costs[1], processor load while scanning[1], clarity and intuitiveness (i.e. user-friendliness) [1], run duration of a scan [1], quality of the report [2] and the number of detected vulnerabilities [4]. Lower factors of a criterions' priority denote that we consider the criterion less important. Table 1 contains the measured results as well as the average[13] and weighted[14] results. The results can also be represented by UML diagrams, as can be seen in [Bus14a].

Tool	**Inst.**	**Costs**	**CPU**	**Clarity**	**Time**	**Vuln.**	**Report**	**AVG[13]**	**WAVG[14]**
Nessus	1	2	2	1	4	1	2	1,86	1,86
Arachni	1	1	4	4	2	1	3	2,29	2,42
Nexpose	4	4	1	2	3	3	1	2,57	2,10
Nikto	1	1	3	4	1	4	4	2,57	3,19

Table 1: Case Study: Final Tool Ranking (adapted from [Lac13])

Security Context Model To allow security engineers to easily access the data we collected, we added entries for Nessus, Arachni, Nexpose and Nikto to the CBK [CBK13]. However, the CBK does not provide fine-grained categories for entering security-specific information. As SecEval's context model is more detailed, we modeled the context of vulnerability scanning of web applications and two of the tested tools: Nessus and Nikto. Figure 9 shows an instance diagram of the context model, which we have already depicted in Fig. 3.

The three vulnerabilities that are modeled are the top 3 from OWASP's top 10 project 2013 [Fou13b]. Vulnerabilities may be caused by other vulnerabilities, as e.g., unvalidated input can lead to injection vulnerabilities. The association between vulnerabilities, as well as further supported methods are not depicted in Fig. 3, but the interested reader is referred to the model example that can be downloaded [Bus14a]. The main advantage of a web-based implementation of SecEval would be that connections to existing elements (like other methods or vulnerabilities), could be added without building the knowledge base from scratch.

We recommend using additional classes for extensions, e.g., a class to detail a test run, using attributes as run duration or processor load. Although building the instance model was straight forward, a future implementation as a kind of semantic wiki would be more user-friendly.

[12]NOWASP (Mutillidae). `http://sourceforge.net/projects/mutillidae`
[13]AVG: average
[14]WAVG: weighted average according to ratings

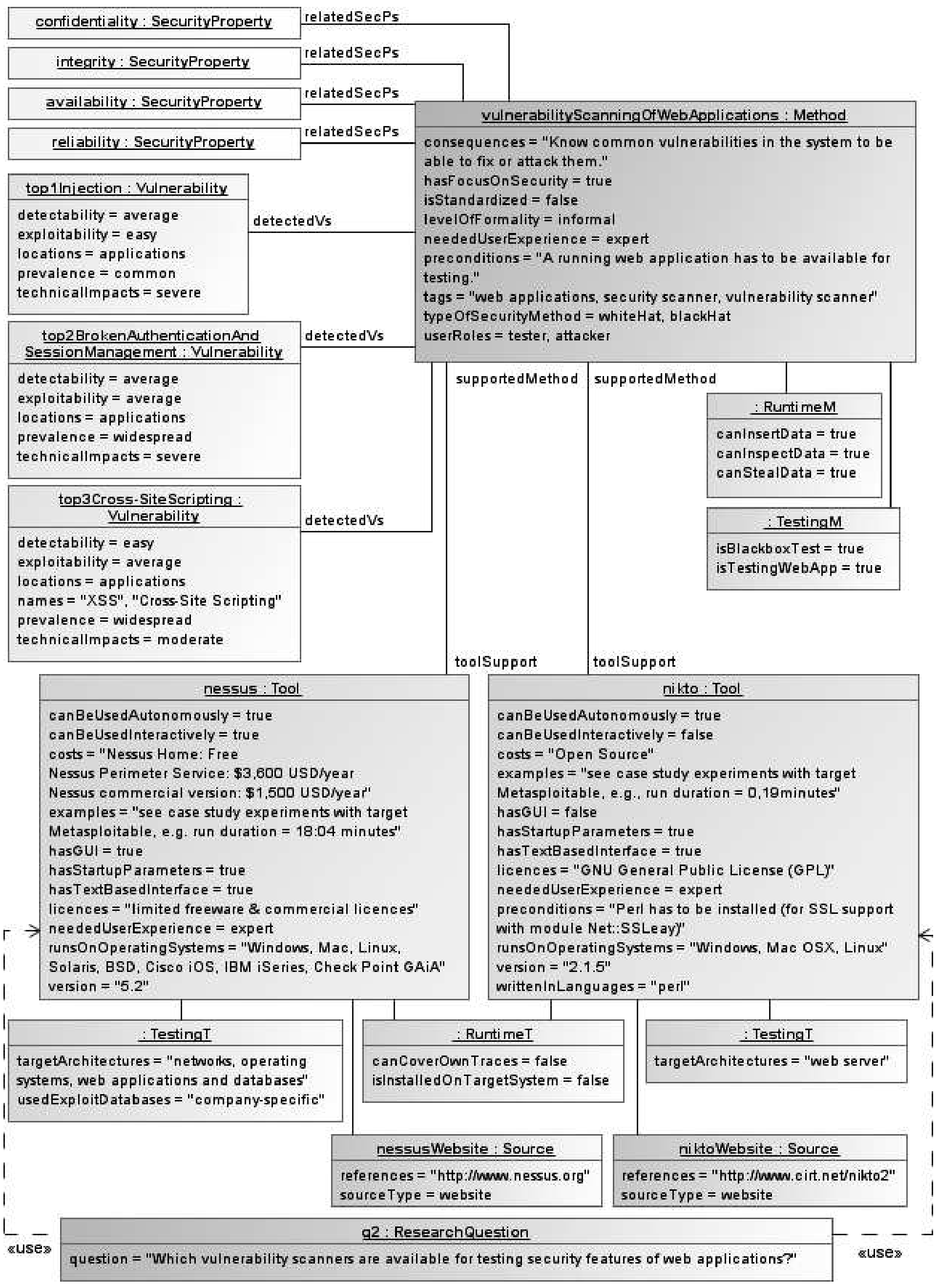

confidentiality : SecurityProperty
integrity : SecurityProperty
availability : SecurityProperty
reliability : SecurityProperty
relatedSecPs
relatedSecPs
relatedSecPs
relatedSecPs
vulnerabilityScanningOfWebApplications : Method
consequences = "Know common vulnerabilities in the system to be able to fix or attack them."
hasFocusOnSecurity = true
isStandardized = false
levelOfFormality = informal
neededUserExperience = expert
preconditions = "A running web application has to be available for testing."
tags = "web applications, security scanner, vulnerability scanner"
typeOfSecurityMethod = whiteHat, blackHat
userRoles = tester, attacker
top1Injection : Vulnerability
detectability = average
exploitability = easy
locations = applications
prevalence = common
technicalImpacts = severe
detectedVs
top2BrokenAuthenticationAndSessionManagement : Vulnerability
detectability = average
exploitability = average
locations = applications
prevalence = widespread
technicalImpacts = severe
detectedVs
top3Cross-SiteScripting : Vulnerability
detectability = easy
exploitability = average
locations = applications
names = "XSS", "Cross-Site Scripting"
prevalence = widespread
technicalImpacts = moderate
detectedVs
supportedMethod
supportedMethod
: RuntimeM
canInsertData = true
canInspectData = true
canStealData = true
: TestingM
isBlackboxTest = true
isTestingWebApp = true
toolSupport
toolSupport
nessus : Tool
canBeUsedAutonomously = true
canBeUsedInteractively = true
costs = "Nessus Home: Free
Nessus Perimeter Service: $3,600 USD/year
Nessus commercial version: $1,500 USD/year"
examples = "see case study experiments with target Metasploitable, e.g. run duration = 18:04 minutes"
hasGUI = true
hasStartupParameters = true
hasTextBasedInterface = true
licences = "limited freeware & commercial licences"
neededUserExperience = expert
runsOnOperatingSystems = "Windows, Mac, Linux, Solaris, BSD, Cisco iOS, IBM iSeries, Check Point GAiA"
version = "5.2"
nikto : Tool
canBeUsedAutonomously = true
canBeUsedInteractively = false
costs = "Open Source"
examples = "see case study experiments with target Metasploitable, e.g., run duration = 0,19minutes"
hasGUI = false
hasStartupParameters = true
hasTextBasedInterface = true
licences = "GNU General Public License (GPL)"
neededUserExperience = expert
preconditions = "Perl has to be installed (for SSL support with module Net::SSLeay)"
runsOnOperatingSystems = "Windows, Mac OSX, Linux"
version = "2.1.5"
writtenInLanguages = "perl"
: TestingT
targetArchitectures = "networks, operating systems, web applications and databases"
usedExploitDatabases = "company-specific"
: RuntimeT
canCoverOwnTraces = false
isInstalledOnTargetSystem = false
: TestingT
targetArchitectures = "web server"
nessusWebsite : Source
references = "http://www.nessus.org"
sourceType = website
niktoWebsite : Source
references = "http://www.cirt.net/nikto2"
sourceType = website
g2 : ResearchQuestion
question = "Which vulnerability scanners are available for testing security features of web applications?"
«use»
«use»

Figure 9: Case Study: Instances of Context Model (excerpt)

4 Related Work

Evaluation approaches are often tailored to the needs of a specific area. We start by introducing general approaches and continue with those which are security-specific.

General Evaluation Approaches. KITCHENHAM et al. [KC07] specify so called "Systematic Literature Reviews" in software engineering. The aim is to answer research questions by systematically searching and extracting knowledge of existing literature. Our approach, SECEVAL, is based on their work. We focus instead on the use of arbitrary resources, as source code or experiments which are carried out to answer a research question. In contrast to Kitchenham's approach, our data collection process is iterative, and more specific for a chosen context as we define a detailed structure for recording results.

SIQINU (Strategy for understanding and Improving Quality in Use) [BPO13] is a framework for evaluating the quality of a product version. It uses the conceptual framework C-INCAMI, which specifies concepts and relationships for measurement and evaluation. SIQinU defines a strategy using UML activity diagrams whereas C-INCAMI is specified by a UML class diagram.

MOODY [Moo03] proposes an evaluation approach which is based on experiments. Practitioners use methods and afterwards answer questions about perceived ease of use, perceived usefulness and intention to use. A figure how Moody's approach can be integrated can be found online [Bus14a].

The CBK (Common Body of Knowledge) [BEHU12] defines a model for software engineers to describe knowledge objects (KOs), which are methods, techniques, notations, tools or standards. Techniques are methods which do not specify activities (in our terminology: "steps") for applying the method. The CBK is implemented as a semantic Wiki [CBK13] and serves as a knowledge base containing all relevant information about existing KOs. Unlike the CBK, SECEVAL is not implemented yet. In contrast to the CBK, SECEVAL focuses on security-related features and provides a fine-grained model. Additionally, it defines a process for the evaluation of KOs.

Security-specific Evaluation Approaches. Security-related frameworks often consider concrete software systems for their evaluation. An example is the OWASP RISK RATING METHODOLOGY [Fou13a], where the risk for a concrete application or system is estimated. We added vulnerability-dependent features of the OWASP model to SECEVAL, as e.g., the difficulty of detecting or exploiting a vulnerability. Features that are related to a concrete system and the rating of a possible attack are introduced as an extension of SECEVAL, which can be found online [Bus14a].

Humberg et al. [HWP+13] propose a two-step approach to support compliant and secure outsourcing of business processes using the concept of ontologies to formalize compliance and regulatory aspects of IT-security. They show how they can apply it to analyze the content of documents in a unified way in order to detect dependencies. Our means are similar, as we want to represent methods, notations and tools in a structured and methodological way. However, we focus on the selection of KOs and not on compliance issues.

The i* [Uni] metamodel is the basis of a vulnerability-centric requirements engineering framework introduced in [EYZ10]. The extended, VULNERABILITY-CENTRIC I* METAMODEL aims at analyzing security attacks, countermeasures, and requirements based on vulnerabilities. The metamodel is represented using UML class models.

Another approach that focuses on vulnerabilities is described by Wang et al. [WG09] Their concept model is less detailed than the i* metamodel. They create a knowledge base that can be queried using a language for the semantic web, called SWRL. Unlike our approach, they do not use graphical models.

5 Conclusion

We present a conceptual framework – called SECEVAL– for the structured evaluation of methods, tools and notations in the area of secure software. SECEVAL specifies (a) an improved, flexible security context model (b) a model that records the way how data is collected (c) an analysis model which defines the analysis strategy, and the filters and algorithms it uses on the collected sources. A UML model is used to represent concepts and relationships of these three concerns (depicted as UML packages): context, data collection and data analysis. Furthermore, SECEVAL was improved using a guided interview and we additionally provided a case study about methods and tools from the area of security testing. The research question of our case study focuses on the selection of vulnerability scanners for web applications.

Summarizing, SECEVAL provides a structure for evaluating research questions related to secure software engineering. We think that this eases the process of doing research in the area of security no matter if the research question aims at scientific or engineering issues.

When implementing the security context model in the future, it will be helpful to add axioms to our model. In our case, we could think about rules to describe dependencies between attributes, like a method should not extend the same version of itself. Additionally, we plan to conduct a case study using SECEVAL for a comprehensive evaluation of knowledge objects of the domain of secure web modeling.

References

[BEHU12] Kristian Beckers, Stefan Eicker, Maritta Heisel, and Widura Schwittek (UDE). NESSoS Deliverable D5.2 – Identification of Research Gaps in the Common Body of Knowledge. http://www.nessos-project.eu/media/deliverables/y2/NESSoS-D5.2.pdf, 2012.

[Bis02] Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley Professional, 1st edition, 2002.

[BK11] Marianne Busch and Nora Koch. NESSoS Deliverable D2.1 – First release of

Method and Tool Evaluation. `http://www.nessos-project.eu/media/deliverables/y1/NESSoS-D2.1.pdf`, 2011.

[BK13] Marianne Busch and Nora Koch. NESSoS Deliverable D2.4 – Second Release of the Method and Tool Evaluation. `http://www.nessos-project.eu/media/deliverables/y3/NESSoS-D2.4.pdf`, 2013.

[BPO13] Pablo Becker, Fernanda Papa, and Luis Olsina. Enhancing the Conceptual Framework Capability for a Measurement and Evaluation Strategy. *4th International Workshop on Quality in Web Engineering* , (6360):1–12, 2013.

[Bus14a] Marianne Busch. SecEval – Further Information and Figures. `http://www.pst.ifi.lmu.de/~busch/SecEval`, 2014.

[Bus14b] Marianne Busch. Secure Web Engineering supported by an Evaluation Framework. In *Modelsward 2014*. Scitepress, 2014.

[CBK13] CBK. Common Body of Knowledge. `http://nessos-project.eu/cbk`, 2013.

[EYZ10] Golnaz Elahi, Eric Yu, and Nicola Zannone. A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. *Requirements Engineering*, 15(1):41–62, 2010.

[Fou13a] OWASP Foundation. OWASP Risk Rating Methodology, 2013. `https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology`.

[Fou13b] OWASP Foundation. OWASP Top 10 – 2013, 2013. `http://owasptop10.googlecode.com/files/OWASPTop10-2013.pdf`.

[HWP$^+$13] Thorsten Humberg, Christian Wessel, Daniel Poggenpohl, Sven Wenzel, Thomas Ruhroth, and Jan Jürjens. Ontology-Based Analysis of Compliance and Regulatory Requirements of Business Processes. In *3nd International Conference on Cloud Computing and Services Science*, 2013.

[KC07] Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[Lac13] Christian Lacek. In-depth comparison and integration of tools for testing security features of web applications, 2013. Bachelor Thesis.

[Moo03] Daniel L. Moody. The method evaluation model: a theoretical model for validating information systems design methods. In C. U. Ciborra, R. Mercurio, M. de Marco, M. Martinez, and A. Carignani, editors, *ECIS*, pages 1327–1336, 2003.

[Sch13] Stefanie Schreiner. Comparison of security-related tools and methods for testing software, 2013. Bachelor Thesis.

[Uni] RWTH Aachen University. i* notation. `http://istar.rwth-aachen.de`.

[Ver13] Verizon. Vector for hacking actions. *Data Breach Investigations Report*, 2013. `http://www.verizonenterprise.com/resources/reports/es_data-breach-investigations-report-2013_en_xg.pdf`.

[WG09] Ju An Wang and Minzhe Guo. Security Data Mining in an Ontology for Vulnerability Management. In *Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS '09. International Joint Conference on*, pages 597–603, 2009.