Supporting Requirements Engineers in Recognising Security Issues

Eric Knauss[1], Siv Houmb[2], Kurt Schneider[1],
Shareeful Islam[3], and Jan Jürjens[4]*

[1] Software Engineering Group, Leibniz Universität Hannover, Germany
{eric.knauss,kurt.schneider}@inf.uni-hannover.de
[2] SecureNOK Ltd., Norway sivhoumb@securenok.com
[3] School of Computing, IT and Engineering, University of East London, UK
shareef@daad-alumni.de
[4] Software Engineering, Technische Universität Dortmund and Fraunhofer ISST, Germany,
http://jan.jurjens.de

Abstract. Context & motivation: More and more software projects today are security-related in one way or the other. Many environments are initially not considered security-related and no security experts are assigned. Requirements engineers often fail to recognise indicators for security problems. **Question/problem:** Ignoring security issues early in a project is a major source of recurring security problems in practice. Identifying security-relevant requirements is labour-intensive and error-prone. Security may be neglected in order to finish on time and in budget. **Principal ideas/results:** In this paper, we address this problem by presenting a tool-supported method that provides assistance for requirements engineering, with an emphasis on security requirements. We investigate whether security-relevant requirements can be automatically identified using a Bayesian classifier. Our results indicate that this is feasible, in particular if the classifier is trained with domain specific data and documents from previous projects. **Contribution:** We show how the ability to identify security-relevant requirements can be integrated in a workflow of requirements analysis and reuse of experience. In practice, this can increase security awareness within the software development process. We discuss limitations and potential of this approach.

Key words: secure software engineering, requirements analysis, natural language processing, empirical study

1 Introduction

IT security requirements increasingly pervade all kinds of software systems, sometimes unexpectedly. Security requirements are often not identified during requirements analysis. Thus, security issues are neglected and can cause substantial security problems later. There are standards and best practices available aimed at guiding developers in building secure systems [1]. Nevertheless, identifying requirements with security implications requires security expertise and experience. Unfortunately, security experts

* The work is partly supported by the EU project Secure Change (ICT-FET-231101).

are not always available. Missing security expertise early in a project is one of the main reasons for security problems.

Security requirements may be implicit, hidden, and spread out over different parts of mostly textual requirements specifications. Any bug in the systems might lead to a security weakness. It is tedious and error-prone to search a document manually or evaluate requirements during elicitation. Resources are usually limited for security analysis.

Therefore, fast and efficient identification of security-relevant statements and requirements is a key skill. Ideally, the identification should follow objective rules and be reproducible (e.g. by being automated or supported by a semi-automatic tool). In this paper, we present a tool-supported approach for identifying security-relevant requirements. It uses experience extracted from previously classified requirements documents. We show that Bayesian classifiers can be used to identify security-relevant requirements (with recall > 0.9 and precision > 0.8 in our evaluation setting). Despite the need for domain specific training these classifiers can support security awareness in software evolution scenarios.

Such a classifier can be integrated into a requirements elicitation tool. It then points out security-relevant issues during interviews. In SecReq [2], this task has been supported by the Heuristic Requirements Assistant (HeRA) [3], based on simple keyword-lists. In this paper, we present Bayesian classifiers as an improvement to the SecReq approach: They are easier to train with new experiences and generate more true findings and fewer false positives.

Section 2 provides an overview of the SecReq approach and how it can be used to simulate the presence of a security expert in requirements elicitation. It outlines how Bayesian classifiers can improve security awareness. Section 3 presents the evaluation for our technical solution. Results are discussed in Section 4. Section 5 outlines related work, and Section 6 concludes the paper by summing up the main results and outlining further directions of work from here.

2 Simulating the Presence of a Security Expert

Our SecReq approach assists in security requirements elicitation. It provides mechanisms to trace security requirements from high-level security statements, such as security goals and objectives, to secure design [2]. We aim at making security best practices and experiences available to developers and designers with no or limited experience with security. SecReq integrates three distinctive techniques (see Figure 1): (1) Common Criteria and its underlying security requirements elicitation and refinement process [1], (2) the HeRA tool with its security-related heuristic rules [3], and (3) the UMLsec approach for security analysis and design [4].

Unfortunately, there are not many security experts, and most security guidelines or ”best practices” are written by and for security experts. Also, security best practices such as standards ISO 14508 (Common Criteria), ISO 17799 are static documents that do not account for new and emerging security threats. Security issues can be characterized as known or hidden, generic or domain-specific. Normally, a security expert is absolutely necessary to identify *hidden* security issues, while *known* issues can be identified using security best practices.

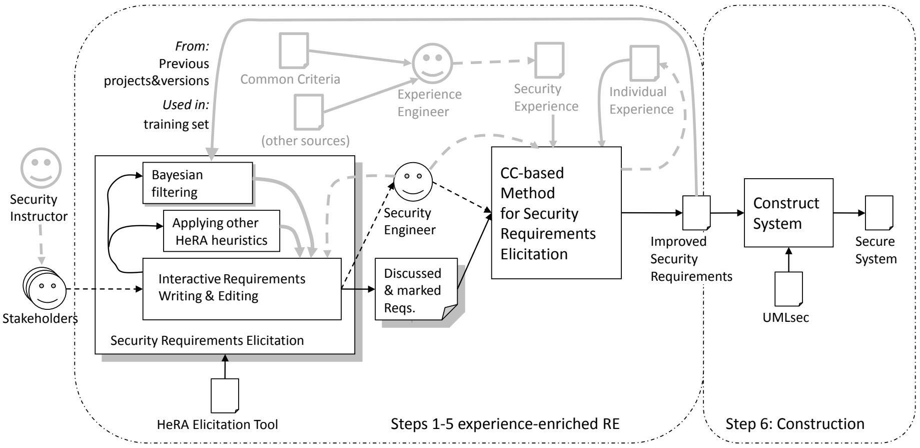

Fig. 1. Overview of SecReq approach. Flow of requirements (black) and experience (grey) is modelled in the FLOW information flow notation [5]. Document symbols and solid arrows indicate documented requirements, experiences, and their flows. Faces symbolise people or groups; direct communication is denoted by dashed lines. Boxes represent activities. The focus of this paper is the activity highlighted by a shadow: Security Requirements Elicitation is refined by applying Bayesian filters. They are trained by reusing documents from earlier projects.

SecReq - and the HeRA tool in particular - guide the translation of these best practices into heuristic rules. They try to make better use of the few security experts around. Rather than having experts do the identification and refinement of all security issues, SecReq reuses their expertise and makes their security knowledge available to non-security experts.

In this paper we describe an improved version of the SecReq approach. We intended to reduce the amount of manual work by making better use of documents and experience from previous projects. Figure 1 shows this improvement as a solid arrow from improved security requirements from previous projects back to the early security requirements elicitation activity (shaded box). We use them to continually train a Bayesian filter. Growing numbers of pre-classified requirements from previous projects will increase the classification ability of that filter. This new experience flow improves the effectiveness and efficiency of the SecReq apporach, because it reduces manual work by leveraging Baysian classifiers. This enables HeRA to address both generic and domain-specific security aspects and to capture experts' tacit knowledge better. Based on this knowledge, heuristic computer-based feedback can simulate the presence of a security expert during security requirements elicitation.

2.1 Classifying Security Requirements.

In order to train the Bayesian classifier, we need pre-classified requirements. During expert classification, we encountered three different types of security requirements. We define:

> *Security requirement: (i)* A (quality) requirement describing that a part of the system shall be secure, or *(ii)* a property which, if violated, may threaten the security of a system.

In our context, security requirements are the result of refining security-relevant aspects. Our goal is to support this process.

> *Security-relevant requirement: (i)* A requirement that should be refined into one or more security requirement(s), or *(ii)* a property that is potentially important for assessing the security of the system.
> *Example:* "The card must ensure that the transaction is performed by the same POS device as was used for the purchase being canceled [. . .]"

During pre-classification, we encountered another type of requirements:

> *Security-related requirement: (i)* A requirement that gives (functional) details of security requirements, or *(ii)* a requirement which arises in the context of security considerations.
> *Example:* "The card and the PSAM must use a public key algorithm for mutual authentication and session key exchange [. . .]"

To support the identification of hidden security aspects, we need to identify *security-relevant requirements*. It took our experts some training to avoid false classification (e.g. classifying a security requirement or security-related requirement as being security-relevant). Furthermore, each and every functional requirement could be regarded to be *somewhat* security-relevant: Safety and Confidentiality of data should always be ensured. Hence, we need a good classification strategy for manual classification. The *classification question* was very instrumental when classifying a requirement:

> *Classification Question:* Are you willing to spend money to ensure that the system is secure with respect to this requirement? Assume there is only a limited budget for refining requirements to security requirements and that there is a need to prioritize and balance cost and risk.

Outputs from security risk analysis approaches (such as CORAS [6], CRAMM [7], and OCTAVE [8]) can be used to support such evaluations, as these provide lists of threats and their related risk level, including potential consequences. Some approaches also directly consider the potential monetary losses. The question then becomes:

> Can you afford to not invest to reduce or remove relevant risks?

2.2 Using Bayesian Classification to Enhance Security Awareness

Machine learning plays an important role in many fields of computer science, especially in practical applications for software engineering [9, 10]. It allows for using machines to analyze huge amounts of data. A prominent example is Bayesian classification which proved to be valuable in classifying spam mails [11]. Spam filters help to find the few important mails between all that annoying spam mails. We think that the technology in these Spam filters is applicable for identifying among all requirements those that need refinement since they are security-relevant. Our rationale is:

- Bayesian classifiers are superior to simple keyword lists, as they do not only consider keywords indicating security relevance, but also keywords that indicate innocence (i.e., security irrelevance) of a requirement.
- For Spam filtering, good results could be reached with only small training sets. A part of a single specification may be sufficient for training.
- Bayesian classifiers can be trained while being used (e.g. by re-classifying false positives). This immediate feedback can support elicitation of tacit security experience.

Bayesian classifiers use statistical methods for classification. In our case, we want to compute the probability $P(\text{sec}_r)$ that a requirement r is security relevant. A classic technique to do this is the *Naive Bayesian Classification*, a mix of stochastic methods and pragmatic assumptions. Our approach is based on Paul Grahams seminal article, one of the most popular descriptions of such a classifier [11]. Basically, we need to compute weights for distinctive features of our requirements (i.e. the words they are composed of), combine the weights of all features, and finally compare this value with a threshold.

Naive Bayesian Classification has some drawbacks and limitations. Rennie et al. discuss technical limitations as well as strategies to overcome them [12]. Accordingly, Naive Bayesian Classifiers are widespread because they are easy to implement and efficient. This makes them a good choice for our evaluation, despite the known drawbacks. Bayesian classifiers are only as good as their training. It is considerably more difficult for humans to identify security-relevant requirements than identifying spam-mail. In this paper, we evaluate whether machine learning could be used for identification of security-related requirements.

Assessing the Security-Relatedness of a Single Word. For computing the weights of the distinctive features of a requirement r, we need to compute $P(\text{sec}_r|w)$: The probability of r being security-relevant under the condition that it contains the word w. The Bayesian rule allows to compute this as follows [13]:

$$P(\text{sec}_r|w) = \frac{P(w|\text{sec}_r) \cdot P(\text{sec})}{P(w)} \tag{1}$$

$P(\text{sec})$ is the probability of encountering security-relevant requirements in real-world specifications. We do not know this value. Therefore, we assume $P(\text{sec}) = 0.5$ as suggested by Graham [11]. $P(w|\text{sec}_r)$ is the probability that we encounter the word w under the condition that r is security-relevant. We can compute this value based on the training set of classified requirements. Let sec_w be the number of security-relevant requirements that contain w and sec_{total} the number of all security-relevant requirements in the training set. Then the following equation gives us an approximation of $P(w|\text{sec}_r)$:

$$P(w|\text{sec}_r) = \frac{sec_w}{sec_{total}} \tag{2}$$

$P(w)$ is the probability to encounter the word w in a requirement. Based on our training set and the theorem of total probability, this value is computed by the following equation ($P(\text{nonsec})$ and $P(w|\text{nonsec}_r)$ are computed analogues to the sec variants):

$$P(w) = P(\text{sec}) \cdot P(w|\text{sec}_r) + P(\text{nonsec}) \cdot P(w|\text{nonsec}_r) \tag{3}$$

With the assumption of $P(\text{sec}) = P(\text{nonsec}) = 0.5$ the equations (1) and (2) result in the following equation for $P(\text{sec}_r|w)$:

$$P(\text{sec}_r|w) = \frac{\frac{\text{sec}_w}{\text{sec}_{\text{total}}}}{\frac{\text{sec}_w}{\text{sec}_{\text{total}}} + \frac{\text{nonsec}_w}{\text{nonsec}_{\text{total}}}} \tag{4}$$

Weight Combination: All words of a Requirement. $P(\text{sec}_r|w)$ gives the probability of a requirement r being security-relevant under the condition that it contains the word w. Thus, every word in a requirement is a witness for r being security-relevant. Now, we need to combine the evidence given by each witness. Again, we have only limited knowledge and borrow some "naive" assumptions from Bayesian spam filtering:

a) $P(\text{sec}_r|w)$ is pairwise independent for each word w.
b) There is a symmetry between the results $P(\text{sec}_r|w)$ and $P(\text{nonsec}_r|w)$: $P(\text{nonsec}_r|w) = 1 - P(\text{sec}_r|w)$.

In English, the probability of finding an adjective is affected by the probability of finding a noun. Assumption (a) does not hold for natural languages. Nevertheless, Spam-filters work under these assumptions, which allow us to derive the following equation from Bayes' theorem: Let $P(\text{sec}_r|w_i)$ be the probability of r being security-relevant under the condition that it contains the word w_i. Let $w_i, 1 \leq i \leq n$ be the n words contained in r.

$$P(\text{sec}_r) = \frac{\prod P(\text{sec}_r|w_i)}{\prod P(\text{sec}_r|w_i) + \prod(1 - P(\text{sec}_r|w_i))} \tag{5}$$

For classification, we compare $P(\text{sec}_r)$ with a threshold. Based on Graham's article, we classify the requirement r as being security-relevant, if $P(\text{sec}_r) > 0.9$ [11]. The approach we described is called *Naive Bayesian Classification*, due to its simplifying assumptions. This technique works in spam filters. Although improved solutions exist, we chose the classic variant to investigate whether even this simple variant would work in security requirements identification.

3 Evaluation of Bayesian Classifiers

This section discusses the quality of classifiers and how they can be used to assist in security requirements elicitation. First, we define our evaluation goals in Section 3.1. Then we describe our strategy to reach these goals and the general process of evaluation in Section 3.2). Finally, we show and discuss the results for each evaluation goal in Sections 3.3, 3.4, and 3.5.

3.1 Evaluation Goals

In order to evaluate our Bayesian classifiers, we define three evaluation goals:

(G1) Evaluate accuracy of classifiers for security-relevant requirements.
(G2) Evaluate if trained filters can be transferred to other domains.
(G3) Evaluate how useful practitioners consider automatically identifying security requirements.

For the goals (G1) and (G2) we used expert evaluation and data mining to create meaningful test data, as described in Section 2. Subsets of this test data were used to train and evaluate the Bayesian Classifiers. Our evaluation strategy had to ensure that training and evaluation sets were kept disjoint. In the context of this paper, goal (G3) is informally evaluated by asking experts for their opinions about classification results. See Section 4.3 for the implications of our results on industrial practice. A more formal evaluation remains future work.

3.2 Evaluation Strategy

Assessing the quality of machine learning algorithms is not trivial:

- *Use disjoint training and evaluation data.* We must not use the same requirements for training and evaluation.
- *Select training data systematically.* For reproducible and representative results, we need to systematically choose the requirements we use for training.
- *Avoid overfitting.* We need to show that our approach is not limited to the specific test data used. Overfitting happens when the Bayesian classifier adjusts to the specific training data.

Typically, *k-fold cross validation* is used to deal with these concerns [9, 14]. This validation method ensures that statistics are not biased for a small set of data [15]. The dataset is randomly sorted and then split into k parts of equal size. $k-1$ of the parts are concatenated and used for training. The trained classifier is then run on the remaining part for evaluation. This procedure is carried out iteratively with a different part being held back for classification each time. The classification performances averaged over all k parts characterizes the classifier. According to [9], we used $k = 10$: With larger k, the parts would be too small and might not even contain a single security-relevant requirement.

We used standard metrics from information retrieval to measure the performance of Bayesian classifiers: precision, recall, and f-measure [16]. Based on the data reported in [14], we consider f-measures over 0.7 to be good. For our purpose, high recall is considered more important than high precision. A classifier is regarded useful in our SecReq approach if precision is at least 0.6, and recall is at least 0.7.

We used three industrial requirements documents for evaluation; the Common Electronic Purse Specification (ePurse) [17], the Customer Premises Network specification (CPN) [18], and the Global Platform Specification (GP) [19]. As described in detail below, we experimented with various different training sets applied to each of the three real-world specifications.

Table 1 provides an overview of the three specifications we used for evaluation of our classifiers: For each specification (left column), we list the total number of requirements they contain (2nd. column) and the number of requirements considered security-relevant (3rd. column). We used either experts (see Sect. 2) or existing databases for identifying security-relevant requirements (last column).

Table 1. Industrial requirements specifications used for evaluation.

Document	*total reqs.*	*security-relevant reqs.*	*security-relevance determined by*
Common Electronic Purse (ePurse)	124	83	expert
Customer Premises Network (CPN)	210	41	database
Global Platform Spec. (GP)	176	63	expert

3.3 Accuracy of Security Classifiers: G1

To test the accuracy of the Bayesian classifier, we use 10-fold cross validation on each of our classified specifications. In Figure 2 we also show the results for smaller training sets. *Training size* gives the number of parts in the 10-fold cross validation considered for training. The trend shown in Figure 2 helps to evaluate whether the training set is sufficient. Results exceed the above-mentioned thresholds for recall and precision. Hence, we consider the classifier useful.

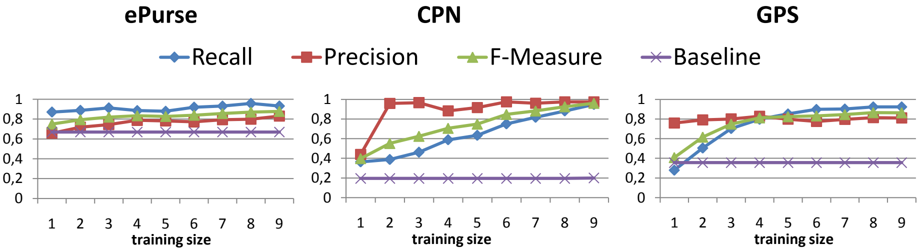

Fig. 2. Results of 10-fold cross validation using only one specification. Baseline is the precision we get when classifying all req. to be security-relevant.

3.4 Transferability of Classifiers Trained in a Single Domain: G2.a

Classifying industrial specifications manually was very time-consuming. It was needed for training the classifiers. Reuse of trained classifiers could reduce that effort. Therefore, we evaluated the quality of classification when we applied a trained classifier to specifications from different projects - without additional training. In order to produce comparative results, we used 10-fold cross validation in all cases, but varied the specifications used for training and for applying the classifiers.

Table 2 shows our results. The first column indicates which specification was used for training. We list the quality criteria (recall, precision, and f-m.) when applying the respective classifier to each of the three industrial specifications in the last three columns. Values on the main diagonal are set in italics: they represent the special case of (G1) reported above, where the *same* specification was used for training and for testing. Even in those cases, the 10-fold cross validation ensured that we never used the same requirements for training and evaluation.

Table 2. Training classifier with one specification, applying it to another.

Training	Applying to:	ePurse	CPN	GP
ePurse	recall	**0.93**	0.54	0.85
	precis	**0.83**	0.23	0.43
	f-measure	**0.88**	0.33	0.57
CPN	recall	0.33	**0.95**	0.19
	precis	0.99	**0.98**	0.29
	f-measure	0.47	**0.96**	0.23
GP	recall	0.48	0.65	**0.92**
	precis	0.72	0.29	**0.81**
	f-measure	0.58	0.4	**0.86**

The results in Table 2 are surprisingly clear: f-measures on the diagonal are 0.86 and higher (same specification for training and test). All other f-measures are far below 0.7: whenever we used different specifications for training and evaluation, transferability is very limited. A filter cannot easily be used in a different context.

3.5 Transferability of Classifiers Trained in Multiple Domains: G2.b

If we apply a Bayesian classifier trained with a specification from one domain to a different domain, we get poor results (see G2.a). This could either point to the fact that we cannot transfer classifiers to other domains or that we used a bad training set. To investigate this, we carried out a third evaluation run where the classifier was trained with values from a mix of specifications. For this, we join the requirements from two or three specifications as input for the 10-fold cross validation. The results in Table 3 show: When we used more than one specification for training, the filter became more generally applicable. If we used two specifications in training, the evaluation for the third specification delivered better results than after a single-specification training (G2.a).

Table 3. Training with more than one specification.

Training	Applied to:	cross-eval	ePurse	CPN	GP
ePurse + CPN	recall	0.93	0.95	0.85	0.56
	precis	0.81	0.80	1	0.51
	f-m.	0.87	0.87	0.92	0.53
ePurse + GP	recall	0.96	0.98	0.85	0.85
	precis	0.80	0.78	0.26	0.8
	f-m.	0.87	0.87	0.40	0.82
CPN + GP	recall	0.87	0.31	0.75	0.88
	precis	0.82	0.84	0.88	0.81
	f-m.	0.85	0.46	0.81	0.84
ePurse + CPN	recall	0.91	0.95	0.85	0.88
+ GP	precis	0.79	0.80	0.94	0.78
	f-m.	0.84	0.87	0.89	0.83

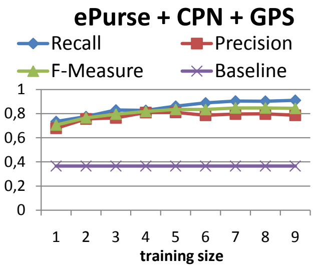

Fig. 3. 10-fold cross validation, multiple training

Combination of different specifications in training made the classifier more generally applicable. Obviously, classification quality is not only based on domain-specific terms - which would not occur in the second training specification. Thus, a good domain-independent classifier can be created with a sufficiently large training set.

The bottom entry in Table 3 shows the results when we combined all three specifications for training. Now we got good results for all three specifications included in the evaluation. Figure 3 shows the learning curve, by giving the results when using less than 9 parts for training. The learning curve grows not as fast as in the Figure 2, probably because the classifier cannot leverage the domain specific concepts. Nevertheless, we get a recall of 91 %, a precision of 79 %, and a f-measure of 84 % - results that clearly show that the trained classifier is suitable to support security requirements elicitation in all of the three domains used for training.

4 Discussion and Implications on Industrial Practice

In Section 3 we described the process of evaluating our concepts. It is important to note that for evaluation purposes we did not use the Bayesian classifier in the way it was designed for (compare Section 2.2):

- *For evaluation* we used a complete specification. Parts of the specifications were used for training, other parts were used for evaluation of recall and precision.
- *In practice* we suggest to use the Bayesian Classifier in an Elicitation tool. Each requirement is classified immediately after it has been written down.

This feedback can be used during an elicitation meeting for immediate clarification on how to proceed with security-relevant requirements. Later, it could be used to generate a list of security-relevant requirements to discuss with security experts. In our SecReq approach, we trigger a refinement wizard that allows laymen to start with the refinement themselves.

In this section we discuss whether the observed results are sufficient for employing the filter in practice at its current status. Then we take a look at the validity of our evaluation of the Bayesian classifier filter. Finally, we summarise the discussion with practitioners and describe how they perceive the implications of the filter in practice, meaning their development projects.

4.1 Interpretation of Evaluation Results

As shown in Section 3 we achieved very good results in cases where the classifier is applied to the requirements from the same source as it was trained with. We also observed poor results in cases where the classifier was applied to a different requirements specification than the one it was trained with. We also observed that the combination of training sets from different sources produces a classifier that works well with requirements from all sources. This shows that a general classifier for security relevance can be created with larger training sets and specifications from more domains.

To summarise, in its current status the classifier is indeed a very valuable addition for example in the context of software evolution or product lines. I.e., the classifier could be trained using the last version of the requirements specification and than offer precious help in developing the new software version. Typically, subsequent specifications resemble their predecessor in large parts and add only small new parts. Evaluation of this situation is covered by k-fold cross validation, as large $(k - 1)$ parts of a specification are used for training and applied to a small held-out part. Therefore, the results in Figure 2 apply to this situation. In other situations, the learning curve in Figure 3 and tests with systematic training with falsely classified requirements show that the classifier quickly adopts to new domains.

4.2 Discussion on Validity

Wohlin et al. define types of threats to validity for empirical studies [20]. We consider threats to construct, internal, external, and conclusion validity to be relevant to our evaluation.

Construct Validity. In our case, the assumptions made on the classification question and our interpretation of what comprises a good result is critical to determining the goodness of the evaluation. When it comes to the classification question there are many alternative ways to define security-relevance. However, our classification was an effective choice in practice as it helped us to adjust our classification in a way that our security experts could agree on the majority of requirements. Next it is important to consider whether it was sound to apply the classifier on final versions of requirements during the evaluation. This depends on the level of abstraction on which the functional information is presented. In practice the requirements are regularly refined from high level functional requirements to low-level descriptions of security-related aspects.

Internal Validity. We used k-fold cross validation and avoided using identical requirements in training and evaluation, as well as overfitting. Randomly choosing requirements for training is not the best way to produce a good filter. Ideally, we would train the filter systematically with false positives and false negatives, until it produces good results. Preliminary tests show that this even increases the performance of the classifier with very small training sets.

External Validity. External validity addresses the level of generalisability of the results observed. We used three real-world requirement specifications from different domains and authors. We have no reason to doubt the applicability of our approach on different specifications.

Conclusion Validity. We used specifications from two different domains in our evaluation. Therefore, we cannot guarantee that our results would hold for a third domain. To leverage this threat, we share our evaluation tool, classified data sets, and databases of learned words at `http://www.se.uni-hannover.de/en/re/secreq`. We invite others to replicate our experiment, or use our results.

4.3 Implications on industrial practice (G3)

In practice, there will rarely be budget to tackle all relevant security aspects. Some of them may even conflict. Hence, developers need to get the right security. For this reason, the classification question (see Section 2) focuses on money (which includes costs, schedule, effort, resources, etc.), where money covers both development costs but also the cost associated with the lack of a critical security feature in the end-product. When it comes to techniques and tools for security elicitation support, such a tool needs to help a developer getting security right, including being able to separate out the important and prioritised security aspects and hidden security requirements that are somehow concerned with potential business and money consequences (loss and gain). Furthermore, such support must be integrated in a natural way such that the tool supports the way the developer work in the security requirements elicitation process and not the other way around. In practice, spending money on something that is not going to end up in the final system is considered a waste of time and effort. This is a sad reality in industrial development.

The Bayesian classification as an addition to SecReq not only contributes to a more effective and focused security elicitation process, but also in separating important from not so important security-relevant aspects. In particular, the Bayesian classification and security expert simulation in HeRA, with its ability to train the classification to be system and project specific, directly enables effective reuse of earlier experience, as well as prioritising and company specific security-related focus areas or policies. The ability to first train the classification engine to understand how to separate important security-relevant aspects from not so important, and then use this newly gained knowledge to traverse functional descriptions and already specified security requirements have a promising potential to contribute in a better control of security spending in development projects.

5 Related Work

Security Requirements. A significant amount of work has been carried out on security requirements engineering in particular relating to tools support in security requirement engineering process, cf. e.g. [21–25]. However, this work does not usually employ techniques from natural language processing to detect security requirements, as we do here.

Processing of Natural Language in Requirements. Requirements are often specified using natural language, if only as an intermediate solution before formal modelling. As natural language is inherently ambiguous [26], several approaches have been proposed to automatically analyse natural language requirements in order to support requirements engineers in creating good requirements specifications [27, 28, 9, 29]. Kof, Lee et al. work on extracting semantics from natural language texts [27, 28] by focusing on the semi automatic extraction of an ontology from a requirements document. Their focus is on identifying ambiguities in requirements specifications. It would be interesting to compare our results to the performance of security specific adoptions of these approaches. A comparable methodical approach is proposed by Kiyavitskaya et al. [29] for extracting requirements from regulations. These methodical and semi-automatic approaches resemble our SecReq approach, the results in this paper describe a supportive technique needed in such a method. Chantree et al. describe how to detect nocuous ambiguities in natural language requirements [9] by using word distribution in requirements to train heuristic classifiers (i.e. how to interpret the conjunctions *and/or* in natural language). The process of creating the dataset is very similar to our work: collection and classification of realistic samples based on the judge of multiple experts to enhance the quality of the dataset. The reported results (recall = 0.587, precision = 0.71) are useful in the described context, but are too low for the SecReq approach.

6 Conclusion

In this paper, we addressed the problem that security becomes increasingly important in environments where there may not be any security experts available to assist in requirements activities. This situation leads to the risk that requirements engineers may fail to identify, or otherwise neglect, early indicators for security problems. We presented a tool-supported method that provides assistance for the labour-intensive and error-prone first round of security requirements identification and analysis. The tool support makes use of a trained Bayesian classifier in order to heuristically categorise requirements statements as *security-relevant* resp. *less security-relevant*. We also showed how HeRA, our heuristic requirements assistant tool can be used to integrate that filter mechanism into a secure software development process. Note, that the approach is not restricted to HeRA, and can be used in other elicitation tools.

We evaluated this approach using several industrial requirements documents; ePurse, CPN, and GP. Our experiences with this ”real-life” validation was overall positive: According to the numerical results, the approach succeeds in assisting requirements engineers in their task of identifying security-relevant requirements, in that it reliably identifies the majority of the security-relevant requirements (recall > 0.9) with only few false positives (precision > 0.8) in software evolution scenarios. Our evaluation of different training strategies shows that the classifier can quickly be adopted to a new domain when no previous versions of requirements specifications are available for training. This could be done by a security expert during a first interview.

Our approach does not aim at completeness in a strict logical sense. There is no 100% guarantee that all security-relevant requirements are found, nor that no non-security-relevant requirements are falsely reported. This is, however, a limitation that is directly imposed by the current limitations from computational linguistics (essentially, the fact that a true automated text understanding is currently not available). In general, security experts cannot give such a guarantee, either. Therefore, we believe that the approach provides useful assistance in that it supports requirements engineers to identify security-relevant requirements, when no security expert is present. Even if security experts are present, our approach helps them to focus on already identified requirements and thereby efficiently use their limited time. Moreover, since this selection process is supported by automated tools, its execution is easy to document and it is repeatable and thus well auditable. This adds another level of trustworthiness to the process, compared to an entirely manual assessment.

Based on our work we see two main topics for future research. On the one hand, it would be interesting, if the approach could be applied to other types of cross-cutting quality requirements (e.g. safety or usability). On the other hand, the application in industrial practice will show the efficiency of our approach.

References

1. International Standardization Organization: ISO 15408:2007 Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 2, CCMB-2007-09-001, CCMB-2007-09-002 and CCMB-2007-09-003 (September 2007)
2. Houmb, S.H., Islam, S., Knauss, E., Jürjens, J., Schneider, K.: Eliciting Security Requirements and Tracing them to Design: An Integration of Common Criteria, Heuristics, and UMLsec. Requirements Engineering Journal **15**(1) (March 2010) 63–93
3. Knauss, E., Lübke, D., Meyer, S.: Feedback-Driven Requirements Engineering: The Heuristic Requirements Assistant. In: International Conference on Software Engineering (ICSE'09), Formal Research Demonstrations Track, Vancouver, Canada (2009) 587 – 590
4. Jürjens, J.: Secure Systems Development with UML. Springer (2005)
5. Schneider, K., Stapel, K., Knauss, E.: Beyond Documents: Visualizing Informal Communication. In: Proceedings of Third International Workshop on Requirements Engineering Visualization (REV 08), Barcelona, Spain (2008)
6. den Braber, F., Hogganvik, I., Lund, M., Stølen, K., Vraalsen, F.: Model-based security analysis in seven steps - a guided tour to the CORAS method. BT Technology Journal **25**(1) (2007) 101–117
7. Barber, B., Davey, J.: The use of the CCTA risk-analysis and management methodology [CRAMM] in health information systems. In Degoulet, P., Lun, K., Piemme, T., Rienhoff, O., eds.: MEDINFO '92, North-Holland, Elsevier (1992) 1589–1593
8. Alberts, C., Dorofee, A.: Managing Information Security Risks: The OCTAVE (TM) Approach. Addison-Wesley, New York, USA (2002)
9. Chantree, F., Nuseibeh, B., de Roeck, A., Willis, A.: Identifying Nocuous Ambiguities in Natural Language Requirements. In: Proceedings of the 14th IEEE International Requirements Engineering Conference, Minneapolis, USA, IEEE Computer Society (2006) 56–65
10. Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements Engineering Journal **13**(3) (September 2008) 207–239
11. Graham, P.: A Plan for Spam (2002) Web, January 2011, http://www.paulgraham.com/spam.html.
12. Rennie, J.D.M., Shih, L., Teevan, J., Karger, D.R.: Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC (2003)

13. Russell, S., Norvig, P.: Artificial Intelligence: a Modern Approach. Prentice Hall, New Jersey (1995)
14. Ireson, N., Ciravegna, F., Califf, M.E., Freitag, D., Kushmerick, N., Lavelli, A.: Evaluating machine learning for information extraction. In: ICML '05: Proceedings of the 22nd international conference on Machine learning, Bonn, Germany, ACM (2005) 345–352
15. Weiss, S.M., Kulikowski, C.A.: Computer systems that learn : classification and prediction methods from statistics, neural nets, machine learning, and expert systems. M. Kaufmann Publishers, San Mateo, Calif. (1991)
16. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press, Addison Wesley (1999)
17. CEPSCO: Common Electronic Purse Specification (ePurse) http://web.archive.org/web/*/http://www.cepsco.com, accessed Apr 2007.
18. TISPAN, ETSI: Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Services requirements and capabilities for customer networks connected to TISPAN NGN. Technical report, European Telecommunications Standards Institute
19. GlobalPlatform: Global Platform Specification (GPS) http://www.globalplatform.org, accessed Aug 2010.
20. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation In Software Engineering: An Introduction. Kluwer Academic Publishers, Boston / Dordrecht / London (2000)
21. Chung, L.: Dealing with Security Requirements During the Development of Information Systems. In Rolland, C., Bodart, F., Cauvet, C., eds.: CAiSE. Volume 685 of Lecture Notes in Computer Science., Springer (1993) 234–251
22. Dubois, E., Wu, S.: A framework for dealing with and specifying security requirements in information systems. In Katsikas, S.K., Gritzalis, D., eds.: SEC. Volume 54 of IFIP Conference Proceedings., Chapman & Hall (1996) 88–99
23. Lin, L., Nuseibeh, B., Ince, D.C., Jackson, M., Moffett, J.D.: Introducing Abuse Frames for Analysing Security Requirements. In: RE, IEEE Computer Society (2003) 371–372
24. Giorgini, P., Massacci, F., Mylopoulos, J.: Requirement Engineering Meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard. In Song, I.Y., Liddle, S.W., Ling, T.W., Scheuermann, P., eds.: ER. Volume 2813 of Lecture Notes in Computer Science., Springer (2003) 263–276
25. Heitmeyer, C.L., Archer, M., Leonard, E.I., McLean, J.: Applying Formal Methods to a Certifiably Secure Software System. IEEE Trans. Software Eng. **34**(1) (2008) 82–98
26. Berry, D., Kamsties, E.: 2. Ambiguity in Requirements Specification. In: Perspectives on Requirements Engineering. Kluwer (2004) 7–44
27. Kof, L.: Text Analysis for Requirements Engineering. PhD thesis, Technische Universität München, München (2005)
28. Lee, S.W., Muthurajan, D., Gandhi, R.A., Yavagal, D.S., Ahn, G.J.: Building Decision Support Problem Domain Ontology from Natural Language Requirements for Software Assurance. International Journal of Software Engineering and Knowledge Engineering **16**(6) (2006) 851–884
29. Kiyavitskaya, N., Zeni, N., Breaux, T.D., Antón, A.I., Cordy, J.R., Mich, L., Mylopoulos, J.: Automating the Extraction of Rights and Obligations for Regulatory Compliance. In Li, Q., Spaccapietra, S., Yu, E., Olivé, A., eds.: Proceedings of 27th International Conference on Conceptual Modeling. Lecture Notes in Computer Science, Barcelona, Spain, Springer (2008) 154–168