Automated Analysis of Security Requirements through Risk-based Argumentation

Yijun Yu[a,*], Virginia N. L. Franqueira[b], Thein Than Tun[a], Roel J Wieringa[c], Bashar Nuseibeh[a,d]

[a] *The Open University, Milton Keynes, UK*
[b] *University of Derby, Derby, UK*
[c] *University of Twente, Enschede, The Netherlands*
[d] *Lero the Irish Software Engineering Research Centre, University of Limerick, Ireland*

Abstract

Security threats and vulnerabilities of computer-based systems evolve, along with changing security requirements. The associated risks are hard to assess through a formal analysis due to the lack of knowledge about the design, the partial model of potential attacks, and the practical constraints of resources from the organisations. Augmenting the formal analysis with informal reasoning, the RIsk assessment in Security Argumentation (RISA) approach could expose additional security risks.

This work introduces a unified meta-model and an automated tool support for security argumentation for security engineers to represent and reason about these risks. Using a uniform representation of risks and arguments, the automated checking of formal arguments can identify relevant risks and mitigations from publicly available security catalogues. These arguments help security engineers make informed and traceable decisions about practical limitations on their system security.

An example of PIN Entry Device for bank cards illustrates the application of the OpenRISA tool. More example applications can be found at the URL http://computing-research.open.ac.uk/trac/openre.

Keywords: Structured Argumentation, Risk Assessment, Security Analysis, Public Catalogue Search, PIN Entry Device

1. Introduction

Security risks evolve in software-intensive systems. Attackers exploit increasing number of vulnerabilities, ranging from cryptographic protocols to human subjects. Introducing new technologies to such systems often imposes security risks with higher likelihood to do harm to the assets. In practice, security is

*Corresponding authors

not perfect due to limited resources available to security engineers, uncertainties about the attackers' skills and commitment, and incomplete knowledge about evolving threats and vulnerabilities.

Recent years found structured argumentation approaches effective to build safety cases (Kelly, 1998) and to reason about both formal and informal descriptions of software systems.W to demonstrate compliance to laws and regulations (Burgemeestre et al., 2010; Cyra and Górski, 2007), to trace and justify software design decisions (Potts and Bruns, 1988), to establish confidence in software development (Graydon and Knight, 2008), and to build dependability cases to assure compliance in software development (Huhn and Zechner, 2010).

Extending the work on security argumentation (Haley et al., 2008), we have developed a framework for reasoning about security requirements of the system where abstract properties are important. For instance, it is possible to formally prove that an access control model will deny access to the Human Resource (HR) database by those who do not work in the HR department.

However, real-life phenomena could defy generalisation and abstraction, there the framework needs to support the uses of informal arguments. For instance, many HR employees could share a common password, and when one of the employees leaves the department and the common password is not changed, thus access becomes available to someone who is no longer a member of the the HR department.

Through the use of RIsk assessment in Security Argumentation (RISA) method (Franqueira et al., 2011), we have shown how risk assessments iteratively challenge the satisfaction of security requirements. The main limitation of our previous work lies in that the separate models for formal arguments and risk-based arguments, which hinders the automated tool support.

In this work, this limitation is addressed by the means of three contributions of the RISA method. First, we introduce an integrated modelling language to represent risk assessment and arguments uniformly. Second, the OpenRISA tool support extends the OpenArgue (Yu et al., 2011) argumentation tool to perform automated checking of the formal arguments. Third, we incorporate an automated search functionality to match catalogues of security vulnerabilities such as CAPEC (Common Attack Pattern Enumeration and Classification patterns[1]) and CWE (Common Weakness Enumeration[2]) with the keywords derived from the arguments. Compared to the previously *ad hoc* search, the new tool supports a complete coverage of these public catalogues of security expertise.

The remainder of the paper is organised as follows. Section 2 reviews relevant background on the satisfaction of security requirements and security arguments, whilst Section 3 reviews related work. Section 4 provides an overview of the RISA method, Section 5 describes the corresponding OpenRISA tool support. Section 6 demonstrates the tool supported method with a PIN Entry Device (PED) example. Section 7 discusses and points to future research and

[1] `http://capec.mitre.org/`
[2] `http://cwe.mitre.org/`

development work. Finally, Section 8 concludes.

2. Background

The RISA method builds on the notions of *satisfaction of security requirements*, and *outer* and *inner* arguments, introduced by Haley et al. (2008).

2.1. Requirements satisfaction arguments about the problem depicted in a context diagram

Following the requirements engineering method (Jackson, 2001), problem software artefacts are separated into W, S and R, where W represents a description of the world in which the software is to be used (i.e., the system context), S represents the specification of a system, and R represents a description of the requirements. The software within the system context should satisfy the requirements. This semantics of a *requirements problem* can be described by the following equation:

$$W, S \vdash R \tag{1}$$

The world context W consists of *domains* (short for *problem world domains*); elements of the world can be either physical, such as people and hardware, or logical, such as software and data structure. Typically, W also contains assumptions made about these domains.

Using the Problem Frames approach (Jackson, 2001), the analysis of a requirement problem follows the divide-and-conquer principle. First of all, the knowledge of the physical domains in the context directly referred to and/or constrained by the requirement statements R are analysed, in order to bring in indirectly related domains to the analysis until a machine specification S is found. Around S, the collection of the domains in the physical world W are depicted in a diagram as nodes representing the domains and edges representing the phenomena shared between the domains. A shared phenomenon could be an event controlled by one domain and observed by another. As such, the so-called context diagram , illustrated in Figure 1, captures the knowledge of high-level causality about the behaviour between these domains. The OpenRISA tool includes the components of `OpenPF` which support the creation of context diagrams.

2.2. Toulmin-structured arguments and argumentation processes

According to Haley et al. (2008), *outer argument* consists of an argument to show that the instance of the system under examination satisfies the security requirements, "with two important assumptions: that the context is correct and that the implementation will not introduce any conflicting behaviour". Often, both assumptions are too strong to be true in practice. Verification that the system can satisfy the requirements cannot ensure the truth of the assumptions, but it does provide a sound structure for the system that is potentially secure.

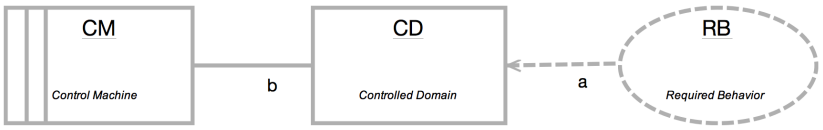

Figure 1: Syntactical elements of a context diagram illustrated by the "Commanded Behaviour" problem frame: a requirement is shown as a dotted oval, a specification is shown as a rectangle with double vertical strips on the left. Problem world domains other than the machine are shown as rectangles. The letter on the solid edge indicates the shared phenomena between the domain nodes. A dotted arrow (or without arrow head) indicates the phenomena of the domain constrained (or referred) by the requirement. The abbreviation above the node name is optional.

Furthermore, the outer argument uses claims about system behaviour to demonstrate that the security requirement is satisfied in a system that conforms to the behaviour specification.

Inner argument (Haley et al., 2008) "consists of structured informal arguments to support the assumptions about system composition and behaviour made in the formal argument". The general purpose of an inner argument is to rebut an outer argument, in one or more rounds, explores how the main claim of the original outer argument fares in the light of new information which gives rise to counterarguments and counter-counter arguments, and so on.

Effective argumentation establishes a *claim* that one wishes to convince the world of. In other words, claim is the object of an argument, a statement that one wishes to convince an audience to accept. In an effective argument, ground truths or facts need to provide the underlying support for an argument, e.g., evidence, facts, common knowledge. In other words, *ground* is a piece of evidence, a fact, a theory, a phenomenon considered to be true. In addition, *warrants* connect and establish relevancy between the grounds and the claim. A warrant explains how the grounds relate to the claim, but not the validity of the grounds. Structurally, warrant is a statement that links a ground and a claim, showing how a claim is justified by ground(s).

Rebuttals express conditions under which the argument does not hold describing what might invalidate any of the grounds and warrants , thus invalidating support to the claim. Grounds and warrants may need to be argued, making them claims, therefore grounds and warrants can also be attacked by the rebuttals. . Here, rebuttal is a counterargument which undermines the support for the claim. Specifically in the case of security-related argumentation, rebuttals represent risks. Rebuttals can be mitigated in order to restore the confidence that the claim of the rebutted argument is true. A mitigation, while negating a rebuttal, introduces additional knowledge to show that the rebuttal, i.e. a risk, can somehow be tolerated. In doing so, a mitigation can also introduce new risks, leading to new iterations of rebuttals and mitigations in argumentation.

Mitigating a rebuttal requires an iterative process, which introduces additional arguments incrementally. The notion of *round* denotes the number of iterations from the beginning. Using the round numbers, cyclic arguments can

be avoided by eliminating the repetitive increments at different rounds. Specific to security satisfaction arguments, there are two types of rebuttals, risks and mitigations. Fig. 2 shows a general iterative argumentation structure , relating inner arguments of logical rebuttals to outer arguments of boundary expansions. Starting from the initial ground about the software system in question, every round of argumentation may introduce additional facts and/or enclose more elements into the system boundary, further the reach of the knowledge.

In summary, each claim of the outer argument can also be represented with Toulmin's argument structure as $(a \xrightarrow{c} b)$, where a is a ground, c is a warrant and b is the claim to be challenged by examining the ground a and the warrant c. Each premise in an outer argument is the beginning for one thread of inner arguments built from several rounds of inner argumentation; it is represented in Fig. 2 as a dotted rectangle related to each premise. Inner arguments are indicated by nested risks and mitigations in different rounds.

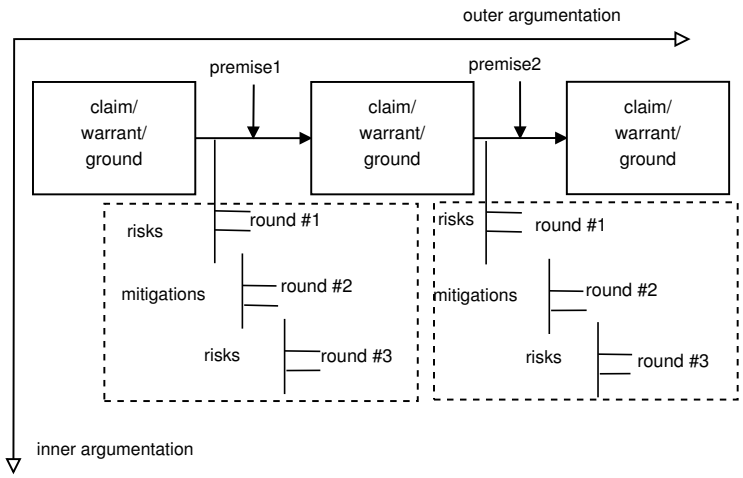

Figure 2: Relationship between outer and inner arguments, as proposed in the framework of Haley et al. (2008)

In the supported iterative and recursive argumentation process, arguments represented as such are incremental in the sense that any claim can be argued against in the next round of argumentation. They are also *non-monotonic* in the sense that the new arguments may introduce new information that overrules the accepted facts in previous rounds.

2.3. Risk arguments for security requirements

According to the BS ISO/IEC 27005 (2011), information security *risk* is associated with the potential that threats will exploit vulnerabilities of one or a group of assets and cause harm to an organization, where *asset* is "anything that has value to the organization and which therefore requires protection". Assets of

relevance are the ones "stored in or accessed by the system-to-be" (Haley et al., 2008). Since vulnerability is the element of risk upon which security engineers can mostly act upon, vulnerabilities are our starting point for the identification of risks affecting the system being designed. Therefore, the RISA method is supported by catalogues that describe security vulnerabilities. *Mitigation*, in risk assessment, refers to treatments that aim at minimising risks. It can act in many ways, e.g., by removing a risk, by changing the negative consequences of a risk, or by avoiding a risk all together; the catalogues also help security engineers in this respect.

Implementing security goals on top of functional requirements, *security requirements* are regarded as constraints on functional requirements that protect the assets from identified harms, controlling security risks in the software system. For example, to maintain the confidentiality of personnel data while handling the requests for HR data, the security requirement shall guarantee that only those requests coming from the members of human resources staff are considered.

In the most general form, the *outer arguments* show whether properties of W and S entail the security requirements (as shown in Equation 1) by using the claims about how the system behaviour conforms to the *domain behaviour premises* of the system; this is captured by Equation 2.

$$\text{(domain, behaviour, premises)} \vdash \text{(security requirements)} \tag{2}$$

Expressed in propositional logic, for example, domain experts can reason about the satisfaction of the argument formally. Of course, the choice of logic is not prescriptive: more expressive logics could also be used as long as the underlying reasoning can be supported.

Outer arguments rely on properties of W and S (*domain behaviour premises*), some of which may turn out to be arguable assumptions. These premises need to be challenged, and be grounded in facts if possible, or taken as true, for instance, on the basis of a lack of contrary evidence.

The general purpose of an *inner argument* is to try to rebut an outer argument. Notice that the outer arguments establish the scope of security assessment, whilst the inner arguments deepen the assessment. The general structure of inner arguments is shown in Fig. 3.

Figure 4 highlights a metamodel of the argument diagram structure.

3. Related Work

Related work is organised around structured argumentation, including its process and representation, and risk assessment.

3.1. Structured Argumentation

Argumentation provides a rationale to convince an audience that a claim should be considered valid. Three qualities are often discussed in the informal argumentation literature: convincingness, soundness, and completeness.

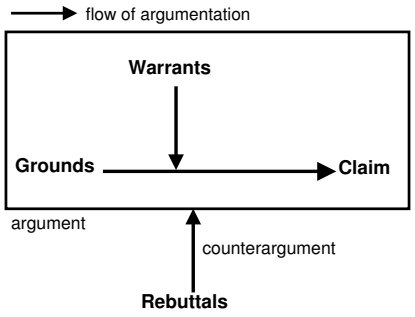

Figure 3: Structure of arguments used in the framework of Haley et al. (2008)

Convincingness relates to whether the argumentation is compelling enough to assure an intended audience that the conclusion reached is reasonable (Haley et al., 2008). Soundness relates to whether the argumentation fulfils the argumentation schema and is based on "true premises" (Graydon and Knight, 2008), i.e. on true grounds and warrants. Completeness relates to whether nothing has been omitted that could lead to a different conclusion about a claim (Graydon and Knight, 2008; Shum and Hammond, 1994).

A known problem in argumentation is the subjectivity involved in identifying arguments and counter-arguments (which relate to soundness), and the difficulty in determining completeness. Proposals to reduce these problems rely on the help of: (i) pre-defined critical questions (Walton, 1996; Atkinson et al., 2004), (ii) what-if scenarios (Baroni et al., 2009), (iii) expert assurance checks (Graydon and Knight, 2008), (iv) guidelines and checklists (Lipson and Weinstock, 2008), or (v) how/why questioning, as proposed in (Haley et al., 2008). However, with a few exceptions e.g., Cyra and Górski (2011), these approaches either rely on the availability of experts or are rather static, i.e. they do not evolve in the speed of the security risk landscape. The OpenRISA support provided to the RISA method allows the practical leverage of evolving public catalogues, updated using input by a pool of security experts from several organisations (`http://cwe.mitre.org/community/index.html`).

3.2. Process of Argumentation

The process of Toulmin-style argumentation (Toulmin et al., 1979), enhanced with recursion from Newman and Marshall (1991), follows a depth-first style and is based on a 2-persons game. It provides a neat and intuitive view about the evolution of an argument in the format of a debate or dialogue between two players. For example, Cohen (1987) regards argumentation as a dialogue, while Walton (1996) regards it as a debate where the opponent asks critical questions.

This process of argumentation, however, is not suitable when reasoning about security with risk assessment. In this case, it often happens that one risk challenges (i.e. rebuts) several premises, one mitigation rebuts several risks, and several mitigations rebut a single risk. As a result, a breadth-first style of

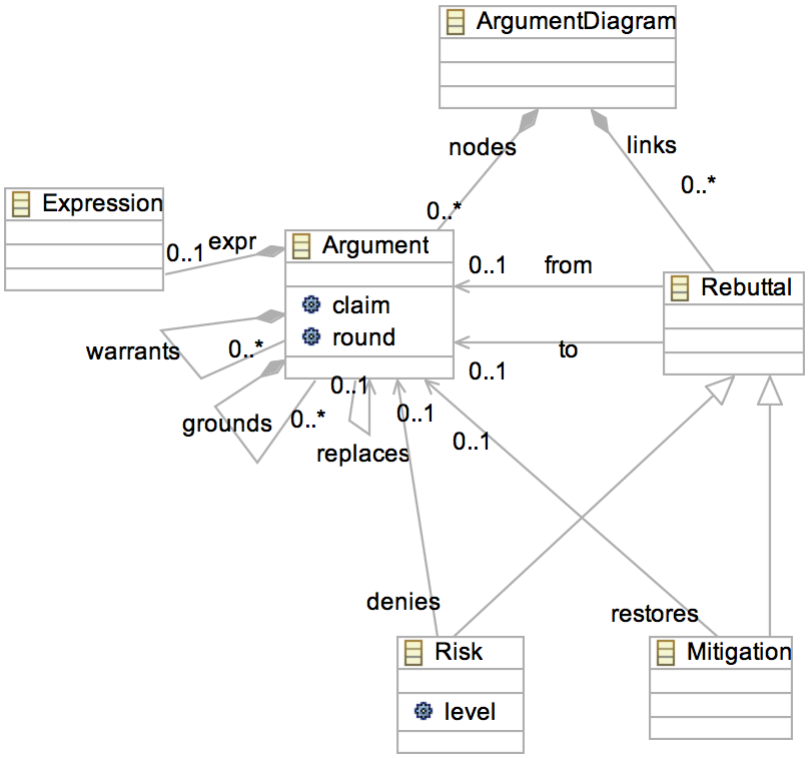

Figure 4: The metamodel of argument diagrams, the detailed structure of a boolean expression is omitted here.

argumentation scales better for practical security reasoning where risks are identified for a bundle or all premises, then mitigations are identified and analysed for all risks, before a next round of argumentation starts. The RISA method is designed for such practical needs, while still providing the ability to reconstruct argument threads via backward traceability. Thus, it is possible to trace mitigations back to risks, then back to premises, which relate to the outer arguments and, ultimately, to the security requirements to be validated in the first place.

The argumentation process implied by Toulmin's model with recursion is well-suited to be represented as a tree structure of arguments, where each (counter)argument rebuts one argument. However, rather than a tree, a more complex graph structure of arguments is needed to represent relationships between arguments in risk-based argumentation. This point is illustrated in Fig. 5 where R1.2 rebuts both premises P1 and P2, mitigation M2.2 rebuts both risks R1.1 and R1.4, and both M2.1 and M2.2 rebut the same risk R1.1. As one can see, the argumentation structure represented in the figure is not a tree since not all nodes (representing arguments) have a unique parent (Bollobas, 2002).

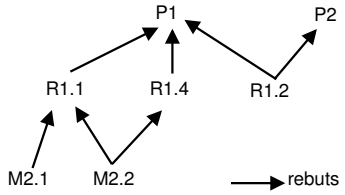

Figure 5: Illustrative example of relationships which can occur between arguments in risk-based argumentation: one risk rebuts two premises, one mitigation rebuts two risks and two mitigations rebut one single risk.

The representation of relations between arguments on a graph structure is no novelty. Dung (1995), unlike Toulmin et al. (1979), abstracted completely from arguments' structures and concentrated on these relations proposing a framework of argumentation useful for a n-persons game.

Dung's framework of argumentation AF is a pair $AF =< AR, attacks >$, where AR is a set of arguments and $attacks$ is a binary relation on AR, i.e. $attacks \subseteq AR \times AR$; $attacks$ relations correspond to Toulmin's *rebuttal* relations. If A attacks B, then $attacks(A, B)$ holds, and this is represented as a directed graph where arguments are nodes and attacks relations are arcs. Therefore, risk-based argumentation is more suitable for the generic graph representation by Dung than for the traditional tree representation of argument-counterargument by Toulmin. In the OpenRISA tool we represent arguments, i.e. the structured relationships between subsequent arguments, as a directed acyclic graph. Cycles between a same set of rebuttals and mitigations are avoided by restricting the target of rebuttals and mitigations to facts and rules in the previous rounds.

3.3. Risk Assessment

Risk assessment involves distinct steps of risk identification, risk analysis & evaluation, and risk treatment. There are three basic perspectives to risk identification; it can be asset-driven, threat-driven or vulnerability-driven. The asset-driven approach prescribed, e.g., by the BS ISO/IEC 27005 (2011), Hakemi et al. (2014) and CORAS (Lund et al., 2011), considers assets as the primary focus for the identification of risks. The threat-driven approach adopted, e.g., by the NIST standard (SP 800-30 rev. 1, 2012), identifies threats (sources and events) as the starting point for risk identification. The vulnerability-driven approach, on the other hand, considers vulnerabilities on the target of analysis as attack surface which represent risks; threat agents will, at some point, discover and exploit them. Therefore, it is not imperative to pinpoint the threat agent. Tools based on vulnerability scanning, such as Nesus[3], and the RISA method that uses catalogues of vulnerabilities take this last approach. However, it adopts a context-based elicitation of risks through argumentation allowing logical traceability of risks and mitigations to premises and security requirements.

[3] `http://www.tenable.com/products/nessus`

The identification of risks may be supported by different artefacts and methods. For example, checklists (as in the BS ISO/IEC 27005 (2011) and SP 800-30 rev. 1 (2012)), workshop with stakeholders (as in CORAS (Lund et al., 2011)), threat taxonomies (e.g., OWASP[4]), and in-house catalogues containing information about previous incidents. These approaches have drawbacks; checklists soon become obsolete, workshops are resource-intensive and hard to schedule with stakeholders, taxonomies and in-house catalogues are not comprehensive. The RISA method addresses these drawbacks by considering ever-changing security catalogues maintained by experts around the World. Nevertheless, the OpenRISA tool is flexible enough to be complemented by any of these other approaches. .

Risk identification in early stages of system development is part of requirements elicitation, and is often threat-driven. Such approaches assume an attacker/misuser/abuser as threat agent and elaborate on "what can go wrong with the system in the environment" (Raspotnig and Opdahl, 2013), i.e., actions performed by the threat agent. One example of such approaches is Misuse Case (Sindre and Opdahl, 2005), which assumes an identifiable sequence of actions performed by a misuser. However, both misuser and actions may not be known, or may be irrelevant. Let's consider risk R1.8: *PIN is revealed if sent unencrypted within the PED and the PED can be tampered.* It states that the risk is derived from the combination of a threat (PIN is revealed if sent unencrypted within the PED), and a vulnerability (PED can be tampered). The threat agent could be identified as a generic attacker, however, the actions performed by the attacker to exploit the vulnerability become irrelevant if the defender choose to mitigate the risk, e.g., by encrypting the PIN inside the PED. Now, let's consider risk R1.24: *PIN is revealed by missing PIN field masking.* The threat can be exercised by exploiting the vulnerability, i.e., missing PIN field masking in PED keypad. In this case, the threat agent is not identifiable (it could be, e.g., a customer or a merchant in a shop, or a waiter in a restaurant) and the steps which lead any threat agent to exploit the vulnerability are neither identifiable nor relevant. What is most important is the risk posed, and the mitigations which could solve the vulnerability; this is the rationale of RISA, spelled out by the risk-based argumentation.

4. The Tool-Supported RISA Method

An overview of the data flow for a security analyst to use the OpenRISA tool supported approach is illustrated in Fig. 6. To support the analyst, the OpenRISA tool has four major components: (1) a model-based editor to help the analyst elicit context diagrams from the descriptions of requirements problem frames; (2) the causality (in terms of shared phenomena) in the context diagrams provides the analyst with an initial set of premises to create the outer arguments, using the OpenArgue argumentation tool; (3) the Lucene-based search engine

[4] `https://www.owasp.org`

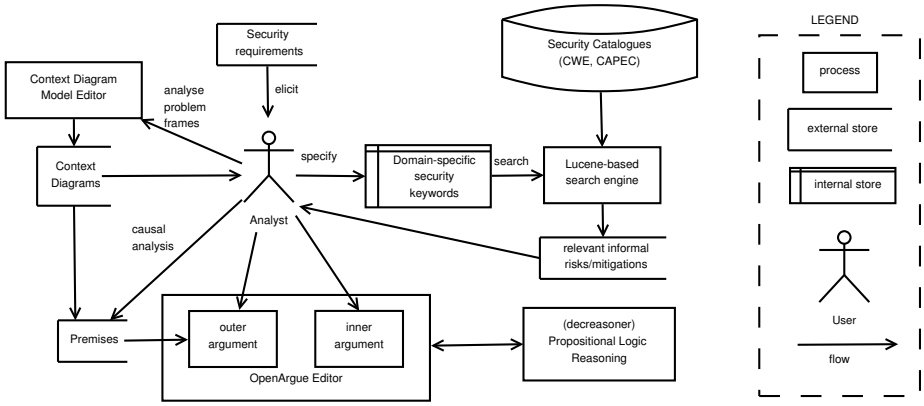

Figure 6: An overview flowchart of the OpenRISA tool support

helps the analyst to find relevant risks and mitigations from the domain specific keywords used in the outer arguments; (4) the analyst can then label the assumptions into formal propositions using inner argument logic expressions in the OpenArgue tool, which will then feed into the proposition logic reasoner by the algorithm to check the reasoning.

These OpenRISA components can be used iteratively therefore there is no predetermined ordering between them. Due to the data flow dependencies, the natural order is recommended by the steps of the RISA method illustrated in Fig. 7. In the following step-by-step descriptions of the RISA method, we show how the components of the tool are used to support the analyst, as depicted in Fig. 6. The steps extend the process of argumentation for security requirements proposed in the Haley et al. framework by incorporating a process of risk assessment.

4.1. Step 1 to Step 3

In Step 1 (Identify Functional Requirements – Fig. 7), functional requirements of the system and the system context (domains and shared phenomena) are identified. These requirements may be derived from the higher-level goals of the system.

In Step 2 (Identify Security Goals), assets that need to be protected, and security goals are identified. Notice that security goals concerns the protection of "valuable assets", which are specific contextual domains not concerned by generic functional goals.

In Step 3 (Identify Security Requirements), security requirements are derived from security goals, and are expressed as constraints on the functional requirements identified in Step 1. System context diagrams (one for each security requirement) are constructed in this step.

4.2. Step 4: Construct Outer Arguments

The outer arguments for security requirements are constructed in Step 4 of

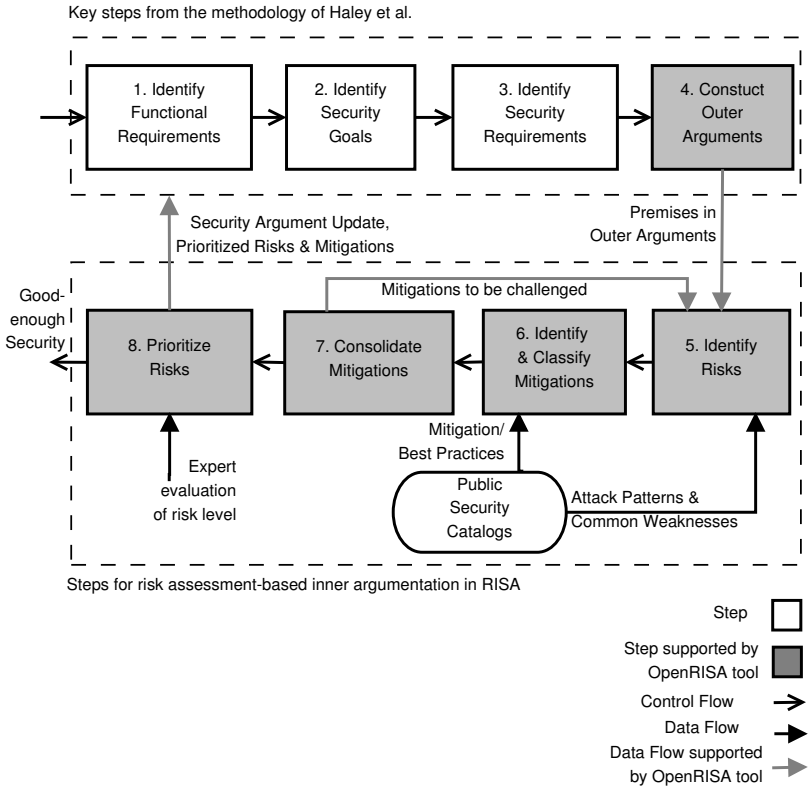

Figure 7: Schematic overview of the RISA method

RISA based on problem-oriented context diagram analysis. These outer arguments make use of domain properties to refer to the domains in the context of analysis which may expand the scope to reason about additional phenomena. Behavioural premises used in the outer arguments may contain risks, which are identified using a systematic risk assessment process in RISA. This is represented in Fig. 7 by the arrow from Step 4 to Step 5.

Steps 5 to 8 correspond to the process of constructing inner arguments in the Haley et al. framework. These four steps show how domain assumptions in outer arguments are challenged by means of risk assessment based on public security catalogues.

Public catalogues provide input for Steps 5 and 6. In the RISA method, we use CAPEC and CWE to feed the identification of risks with descriptions and information about known attack patterns and weaknesses in software, and for information on how these attacks and weaknesses can be mitigated. In-house security catalogues or other methods, such as CORAS' workshops with stakeholders (Lund et al., 2011), may complement public security catalogues.

The recursion of the inner argumentation is represented in Fig. 7 by both

the indirect connection between Step 5 and Step 7 (through Step 6), which involves the process of finding risks and mitigations to these risks, and the direct connection between Step 7 and Step 5, which involves the process of finding new risks in mitigations. Since these risks are attached to arguments and security requirements, prioritising risks indirectly results in the prioritisation of arguments and security requirements.

4.3. Step 5: Identify Risks

In this step, behavioural premises in outer arguments regarding the domains (arrow from Step 4 to Step 5 in Fig. 7) are analysed in terms of potential risks (which rebut the premises). Public security catalogues are then searched to find known security weaknesses and attack patterns.

Automated catalogues Search

The search in both catalogues is automated as follows.

1. Each XML dataset entry (e.g., CAPEC v2.1: `http://capec.mitre.org/data/xml/capec_v2.1.xml`, and CWE v2.5: `http://cwe.mitre.org/data/xml/cwec_v2.5.xml.zip` catalogues) is converted into a text file using the XSL scripts shown in Fig. 8 and 9;
2. Those text files are searched, given keywords, using the `Lucene` Java library (Hatcher and Gospodnetic, 2004) through a command line script (query.sh);
3. The PorterStemAnalyzer from `Lucene`, which implements the Porter Stemming algorithm (Porter, 1980), is used to remove common suffixes from keywords, when applicable. For example, keywords "connection", "connected" or "connecting" will all be reduced to the stem "connect", and the search for this stem in the text will recover catalogue entries containing any such variations in suffixes common in English.

Running a search involves the input of keywords in a query text file ('query.txt'), one keyword per line, and running the command line query script to execute `Lucene`'s search functionalities[5]. Keywords may be composed of more than one word.

Therefore, queries `query.sh CAPEC` and `query.sh CWE` perform the search for given keywords in the CAPEC and CWE catalogues, respectively, and `query.sh CAPEC CWE` performs the search in both catalogues. The output is also a text file (hits.txt) containing the triplex (keyword, catalogue, entry reference), one per line. For example:

```
victim CAPEC attack_pattern_89
password CWE weakness_258
```

[5]The automated search is available in the RISA repository at `http://sead1.open.ac.uk/risa/search.php`

```
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version = "1.0" >
   <output method="text"/>
   <template match="/">
      <for-each select="/Attack_Catalog/Attack_Patterns/Attack_Pattern">
         <document href="split/attack_pattern_{@ID}.txt" method="text">
            <for-each select="./Description/Summary/Text">
               <value-of select="."/>
            </for-each>
            <for-each select="./Example-Instance_Description/Text_Title">
               <value-of select="."/>
            </for-each>
            <for-each select="./Example-Instance_Description/Text">
               <value-of select="."/>
            </for-each>
         </document>
      </for-each>
   </template>
</stylesheet>
```

Figure 8: XSL script used to generate one '.txt' file for each XML-based CAPEC entry (CAPEC XML 1.6), containing natural language text fetched from the fields indicated

```
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version = "1.0" >
   <output method="text"/>
   <template match="/">
      <for-each select="/Weakness_Catalog/Weaknesses/Weakness">
         <document href="split/weakness_{@ID}.txt" method="text">
            <value-of select="./Description/Description_Summary"/>
            <for-each select="./Description/Extended_Description/Text">
               <value-of select="."/>
            </for-each>
         </document>
      </for-each>
   </template>
</stylesheet>
```

Figure 9: XSL script used to generate one '.txt' file for each XML-based CWE entry (CWE XML 2.1), containing natural language text fetched from the fields indicated

4.4. Step 6: Identify & Classify Mitigations

This step involves analysing the catalogue entries related to risks identified in the previous step to (i) find appropriate security mechanisms for mitigating them (arrow from the catalogues to Step 6) and (ii) classify these mitigations according to two categories of risk treatment: *mitigate-by-system* and *mitigate-by-context*.

There are risks for which the obligation to mitigate them is either fully transferred to the system context (when all their mitigations are classified as mitigate-by-context) or full responsibility of the system itself (when all their mitigations are classified as mitigate-by-system). On the other hand, there are risks for which this obligation is shared between the system context and the system. Therefore, in the classification made in this step, we aim to identify which mitigations applicable to a risk belong to which group.

4.5. Step 7: Consolidate Mitigations

Only mitigations assigned to the system, i.e., mitigations classified as *mitigate-by-system* in Step 6, are considered in this step. Mitigate-by-context mitigations are not carried forward since context domains are responsible for implementing them; in terms of the system, they are assumptions about the context considered to be satisfied.

The step involves the consolidation of mitigations reoccurring in several risks (based on the output of the previous step), and consists of (1) numbering mitigations, (2) assigning to each of them a list of risks it rebuts, and (3) updating their description to comply with all these risks, if applicable. Some of these mitigations, themselves, could introduce new risks and therefore should be assessed in a new round of inner argumentation (arrow from Step 7 to Step 5 in Fig. 7).

4.6. Step 8: Prioritise Risks

In the last step, risks are prioritised on the basis of their risk level (e.g., $likelihood \times impact$) from expert estimation, as indicated by an input arrow feeding Step 8 in Fig. 7). The catalogues contain, for some entries, predefined ratings of likelihood and impact but they need to be customized (especially the impact rating) to the system under analysis. These risk levels affect the priority of outer arguments, and therefore, of security requirements to be satisfied (arrow from Step 8 to Steps 1–4, Fig. 7).

When the residual risks from the risk assessment-based inner argumentation are deemed to be acceptable, given limitations found in practice (e.g., limitations of development resources), the system is considered to have reached a level of satisfactory security.

5. The OpenRISA Tool

This section presents a domain-specific modelling language corresponding to the metamodel of outer arguments, inner arguments and risks assessment shown in Fig. 4. The language presented here extends the argumentation language

presented by Yu et al. (2011). This section highlights the syntax and semantics of the integrated argumentation language. It also illustrates the algorithms for checking rebuttals and mitigations in the inner and outer arguments.

5.1. Novelty of OpenRISA

OpenRISA proposes an integrated modelling language for argumentation and risk assessment. For the outer arguments expressed in this new language, the OpenRISA[6] tool supports (1) formal checking if the risks found by searching the catalogues are indeed rebuttals to the satisfaction argument; (2) formal checking if the mitigations to the risks are indeed capable of restoring the rebutted arguments; and (3) prioritising the risks based on a global threshold for selecting the relevant arguments.

In terms of the steps in the RISA method, the OpenRISA tool can be used to (i) check the satisfiability of the outer arguments in Step 4 through formal reasoning, (ii) describe and visualise the informal inner arguments in Steps 5 to Steps 7, and (iii) identify prioritised risks in Step 8.

5.2. Syntax of Outer Arguments

Following Haley et al., we use propositional logic to write the outer arguments. In the syntax of OpenRISA, propositional variables are first defined and described before a formula is written.

```
1 argument: prop_example

3 boolean S1, S2, S3

5 S1 "User enters ID and passcode" with S1
6 S2 "User ID and passcode match a pair
7       of stored  ID and passcode" with S2
8 S3 "User is a valid user" with
9       S1 & S2 -> S3
```

In the above example, three propositional variables `S1`, `S2` and `S3` are defined and described using strings in lines 5–8. The keyword `with` indicates the formal assertion: `S1` and `S2` assert that they are both true, whilst `S3` asserts that $S1 \wedge S2 \rightarrow S3$. Logical connectives are written using the standard encoding: `!` for negation, `&` for conjunction, `|` for disjunction, `->` for implication and `<->` for equivalence. The three statements together therefore say that, $S1$, $S2$, $S1 \wedge S2 \rightarrow S3$. OpenRISA is integrated with the tool `Decreasoner` (Mueller, 2011), and can check the correctness of the propositional formula such as this.

5.3. Visual and Textual Syntax of Inner Arguments

Each inner argument has exactly one claim. A claim is a proposition whose truth value is to be established by the grounds and warrants supporting the claim. When there is no ground or warrant supporting the claim, we take the

[6]The tool is available to be downloaded as an Eclipse rich client applications from `http://sead1.open.ac.uk/risa`.

claim to be self-evident, or a ground. In other words, a ground is an argument with a claim with no supporting ground or warrant. As shown in Fig. 10, visually an argument is represented by a rectangle with three compartments. In the top compartment, the single claim of the argument is written in format of `ID: Description round#` where `ID` is the identifier of the claim, `Description` is a natural language description of the claim, and `round#` is a time stamp indicating the round at which the claim is introduced.

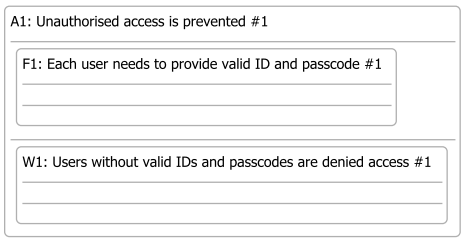

Figure 10: A simple argument

In grounds, the middle compartment, of the argument with the claim `A1` is another argument `F1` with three compartments. Since only the claim of the argument `F1` is given, it is taken as a ground for the argument `A1`. Similarly, `W1` is another ground that warrants that the claim `A1` is true because of the ground `F1`.

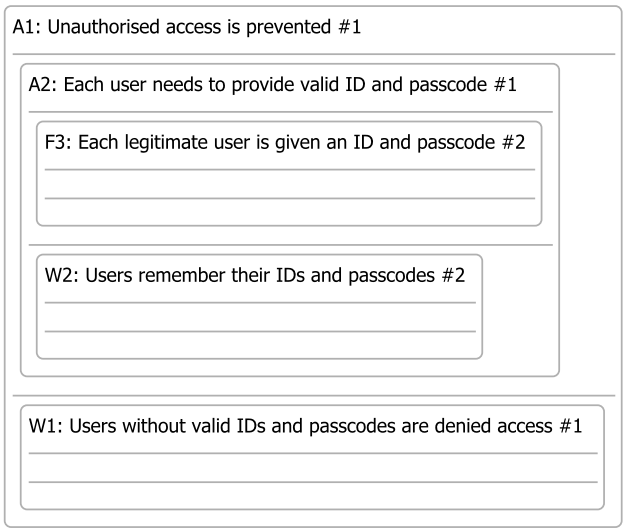

Figure 11: A nested argument

The use of the same notation for an argument, a ground, and a warrant allows the users to easily turn a ground into a claim. For instance, in Fig. 11, the ground `F1` is no longer treated as a ground, rather a claim that needs to be

supported further by a ground and a warrant. As a result `F1` is now changed into the claim of a second argument `A2` supported by `F3` and `W2`. This nested style of syntax is appropriate for representing arguments because during the process of argumentation, grounds are typically challenged, thus leading to additional knowledge to be incorporated into the arguments.

Since arguments can be nested, it is often useful to know at what stage during the argumentation process claims, grounds, and warrants are introduced: this is indicated by `round#`. In this example, it is clear that `F3` and `W2` (round 2) were introduced after `A1`, `A2` and `W1` (round 1).

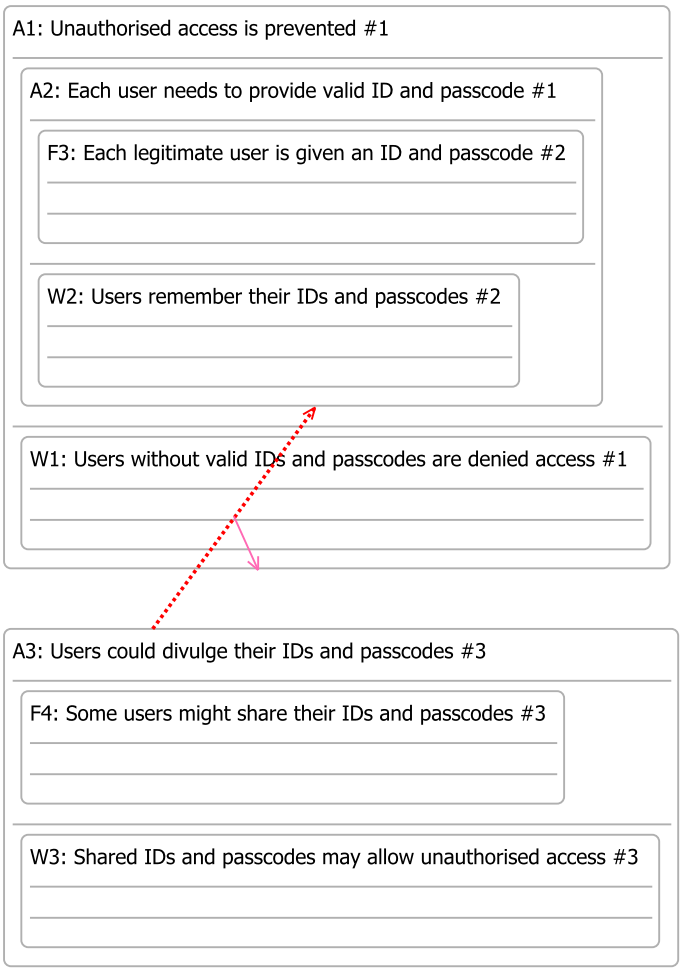

Figure 12: An argument and its rebuttal

As well as nesting sub-arguments within arguments, arguments may be related to other arguments through rebuttal and mitigation relationships. Fig. 12 shows how the nested argument (`A1` containing the sub-argument `A2`), is rebut-

ted by another argument A3. The rebuttal relationship is represented by the dotted red line, indicating that the claim of the argument A2 cannot hold because some unauthorised people can also obtain valid IDs and passcodes. The effect of this rebuttal is not only that the argument A2 is false, but the argument A1 is also false as well: this is indicated by the solid pink arrow pointing at the boundary of A1. The solid pink link shows the scope of the rebuttal, and is particularly useful when there are several levels of nesting, and when there are several rounds of arguments, as it shows clearly the highest level argument that has been rebutted. Note that the choice of cue for different types of edges is a combination of dashed/solid arrows and colors as experience showed that color is not always the best cue for visualisation Ernst et al. (2006).

In the RISA method, a rebuttal to a security argument represents a risk, and may be addressed by a mitigation. Generally, a mitigation restores the claim that has been negated by a rebuttal. In the example in Fig. 12, the mitigation to A3 could be an argument that claims that legitimate users are instructed not to divulge their IDs and passcodes. Notice that this mitigation argument does not necessarily say that A3 is false: it simply says that the rebuttal can be tolerated by giving legitimate users certain responsibility. Since a system cannot prevent a user from divulging their ID and passcode, the rebuttal A3 remains valid. This is a kind of residual risk to the system. Diagrammatically, a mitigation relationship is represented by a solid green arrow.

5.4. Integrated Syntax for Arguments and Risks

The graphical syntax of arguments is represented as simple user-editable textual syntax, and the two syntaxes are supported by bi-directional synchronisation between the editors created using the Eclipse Modelling Framework (EMF) and the Graphical Modelling Framework (GMF). Furthermore, the textual syntax may also contain formal arguments in propositional logic, which are ignored by the synchronisation. The informal arguments shown in Fig. 12, and their formalisation are described by the following textual syntax.

In the RISA method, the outer arguments are elicited through a systematic procedure from the premises of the causal phenomena in the context diagrams , whilst the inner arguments, in particular, the identification of rebuttals and mitigations are based on risk assessment. OpenRISA syntax now allows entries from catalogues, such as CAPEC and CWE, to be included in the informal argument.

5.5. Automated Reasoning about Arguments in Integrated Syntax

The OpenRISA tool supports the checking of risks and mitigations from inner arguments based on propositional logic, and identification of risks. It checks the overall satisfaction of the claims of security requirements for the validity of a risk or a mitigation. When it is iteratively applied to all the rounds, it is possible to identify whether or not the logic rebuttal holds or not. The reification of the logical literals and the natural language expressions of the facts are, however, not checked by the logic reasoning.

```
argument: arg_rebuttal

boolean A1, A2, A3, F3, F4, W1, W2, W3

A1 rebutted by A3 on A2

A1 "Unauthorised access is prevented" round 1 {
   supported by
      A2 "Each user needs to provide
            valid ID and passcode" round 1 {
      supported by
         F3 "Each legitimate user is given
               an ID and passcode" round 2
            warranted by
            W2 "Users remember their IDs and
                  passcodes" round 2
      }
      warranted by
         W1 "Users without valid IDs and
              passcodes are denied access" round 1
}

A3 "Users could divulge their IDs and
    passcodes" round 3 priority 8 {
   supported by
      F4 "Some users might share their IDs and
          passcodes"  round 3
   warranted by
      W3 "Shared IDs and passcodes may allow
           unauthorised access" round 3
}
```

This reasoning is implemented by the two algorithms discussed below.

When formalising the argumentation as propositions, the basic structure of an argument is transformed into the following formula:

$$P_G \wedge P_W \rightarrow P_C \tag{3}$$

where P_G is the conjunction of the propositions of the grounds, P_W is the conjunction of the propositions of the warrants, and P_C is the conjunction of propositions of the claim.

Note the difference between $\rightarrow$ in (3) and $\vdash$ in (2): the propositional logic is used here to realise one formal mechanism. For domain experts, propositional logic is conceptually simpler than high-order logic and practically supported by reasoning tools. Instead of high-order logic rules, we capture risks and mitigations as the iterative argumentation structure and use the associated algorithmic process to achieve the non-monotonic reasoning.

The tool automatically extracts the identifiers of the claims as propositional literals and constructs a propositional formula accordingly in the conjunctive normal form. Syntactically, every informal claim may be accompanied by a propositional formula, such as the formula for `S3`. The tool first parses the syntax of the propositional formula, and adapting the `xtext` unparsing API, it can weave both the implicit logic rule (3) with the user-defined rules into a syntactically correct propositional statement according to the BNF production rules of Event Calculus.

Our algorithms traverse the round-based incremental argumentation structure to check all possible rebuttals and mitigations between adjacent rounds for effective ones: any rebuttal negates the argued claim, and any mitigation removes the negation of the previous rebuttals.

When adding further rounds of arguments, the logic knowledge base is non-monotonic. Therefore, support for the initial claims has to be re-examined after each round of increment. The output of the algorithms presents the effective argumentation process as a directed acyclic graph where nodes are the incremental arguments and edges are the rebuttal or mitigation relationships between these arguments of adjacent rounds.

Since arguments are typically constructed incrementally, in several rounds, there are usually many sequences of arguments, rebuttals and mitigations in an argument graph. Algorithm 1 enumerates all possible sequences in the argumentation process (Lines 3-15). For each sequence, Algorithm 2 is used to identify the incremental arguments that rebut and mitigate the previous arguments (Lines 8-11). Note that an incremental argument is checked only if its priority is between the lower and upper bounds of a user-defined range. In other words, a rebuttal with the risk below a certain level or a mitigation with the cost above a certain threshold[7] will not be used for reasoning.

Input: $A^* = \{a \mid a.r \in \mathcal{N}\}$: a set of incremental arguments a, annotated by a natural number $a.r \in \mathcal{N}$ to indicate the round of a ;
Output: Rebuttals $R^* = \{(a_{i'}, a_{i-1}, a_r)\}$ and
Mitigations $M^* = \{(a_{i'}, a_{i-1}, a_i)\}$ where $1 \leq i' < i \leq \max_{a \in A^*}(a.r)$.

```
1  begin
2      S := {()} // initially a set of an empty sequence
3      for i = 1, max_{a∈A*}(a.r) do
4          S' := {}
5          for s ∈ S do
6              // each round of argumentation
7              for each a ∈ A* | a.r = i do
8                  s' := concat(s, a)
9                  R, M := CheckingArgumentRelationships(s') // see Algorithm 2
10                 R*, M* := R* ∪ R, M* ∪ M
11                 S' := S' ∪ {s'}
12             end
13         end
14         S := S'
15     end
16 end
```

Algorithm 1: Note that set notations are used extensively, where the round number of argument structure is element of $\mathcal{N}$ which is the set of natural numbers.

On each sequence of incremental arguments s', rebuttals and mitigations may only appear alternately. Algorithm 2 takes the input from the sequence

[7] Although the OpenRISA tool already supports the assignment of cost to mitigations, the RISA method at its current state does not take advantage of this feature.

s' and generates the output as rebuttals R and mitigations M. Specifically, Line 6 updates the knowledge bases at round r by taking into account the newly introduced, removed and modified facts; Line 10 invokes an external reasoning tool (`Decreasoner`) Mueller (2011) to turn the encoded propositional logic formula into a satisfiability formula for a solver. Lines 14-29 convert the satisfiability evaluation results into risks and mitigations.

Let $n = \max_{a \in A^*} a.r$ be the number of rounds in the argumentation process, the complexity of Algorithm 1 is $\mathcal{O}(m)$ after factoring out all possible arguments that need checking where the input size $m = |S| = \Pi_{i=1..n}|\{a|a.r = i\}|$ is the total number of possible sequences of argumentations. For example, if there is one original claim to be argued, suppose there are 2 incremental arguments at the first round, 3 incremental arguments at the second round, and 2 incremental arguments at the third round, then the number of possible argumentation sequences is $1 \times 2 \times 3 \times 2 = 12$.

Algorithm 2 has a complexity of $\mathcal{O}(n^2)$ set operations for incremental updates of the knowledge bases, plus $\mathcal{O}(n^2)$ inquiries of the knowledge bases to verify all the claims. According to our implementations using `Decreasoner`, the automated reasoning is quick especially when the arguments are introduced incrementally. Although the SAT solver is called by $\mathcal{O}(n^2)$ times, for a more complex argument diagram of 64 nodes and 5 rounds, it produces the reasoning results in less than 10 seconds. Even though satisfiability problem computation is NP-complete in terms of computational complexity for the worse cases, with the practical purposes in our case study it is shown quite effective.

Of course, if one only would check the argument for one particular claim rather than all possible claims, the complexity can reduce to $\mathcal{O}(n)$ by converting the loop in Lines 8-11 into a single iteration.

6. The PIN Entry Device (PED) example

PIN Entry Device (PED) is a type of device widely-deployed and used by consumers to pay for goods with debit or credit smartcards at the Points-Of-Sale (POS).

When using the device, cardholders typically insert their cards, issued by a financial institution, into a card-reader interface of the PED, enter the PIN using the PED's keypad, and confirm the transaction value via a display on the PED itself. Then smartcard-based systems are expected to authenticate cardholders via the PIN and verify the card details against a public-key certificate before transactions can be completed successfully. These certificates are usually stored on the chip but they can also be stored on the magnetic strip for compatibility with card-readers that have not adopted this technology.

Most PEDs used in Europe implement the EMV (EuroPay, MasterCard and Visa) protocol in the process of authentication and authorization of payment transactions. This protocol drives the communication at the PED-card interface and the PED-bank interface. The protocol in principle allows only encrypted transmission of the PIN across these interfaces when the PED, card and bank

Input: $s = (a_1, \ldots, a_n)$ for $n \in \mathcal{N}$ is a sequence of incremental arguments in the form of $a_n.G, a_n.W \vdash a_n.C$;

Output: For the argument at the i'-th round with a claim $\mathcal{KB}_{i'} \vdash a_{i'}.C$ where $1 \leq i' < i \leq n$:
a) rebuttals: $R = \{(a_{i'}, a_{i-1}, a_i) \mid \vdash a_{i'}.C) \wedge (\mathcal{KB}_i \vdash \neg a_{i'}.C\}$;
b) mitigations: $M = \{(a_{i'}, a_{i-1}, a_i) \mid (\mathcal{KB}_{i-1} \vdash \neg a_{i'}.C) \wedge (\mathcal{KB}_i \vdash a_{i'}.C)\}$.

```
1  begin
2      KB_0 := {}
3      for i = 1, n do
4          // update the knowledge base at the i-th round:
5          a = s[i]
6          KB_i := KB_{i-1} ∪ a.G_r ∪ a.W_r \ {A^-_{i'→i}}
7          // verify the satisfaction on the original claims
8          for i' = 1, i, +1 do
9              a' = s[i']
10             V_{i,i'} := eval (KB_i ⊢ a'.C) // decreasoner
11         end
12     end
13     // output the verification results
14     for i' = 1, n − 1 do
15         if V_{i',i'} then
16             for i = i' + 1, n do
17                 if V_{i−1,i'} ∧ ¬V_{i,i'} then
18                     R := R ∪ {(a_{i'}, a_{i−1}, a_i)}
19                 else
20                     break
21                 end
22                 if ¬V_{i−1,i'} ∧ V_{i,i'} then
23                     M := M ∪ {(a_{i'}, a_{i−1}, a_i)}
24                 else
25                     break
26                 end
27             end
28         end
29     end
30 end
```

Algorithm 3: Note that set notation is used extensively, the grounds (G), warrants (W) and claim (C) components of the argument structure are accessed by "." operator. The symbol $\mathcal{KB}$ stands for the knowledge base which is a collection of propositions in the conjunctive normal form.

support asymmetric cryptography. However, many card issuers in Europe make the conscious decision to adopt a low-cost EMV option in their smartcards which researchers (Drimer et al., 2008) have found to the be vulnerable, since it can be triggered to transmit unencrypted PIN on the interface PED-card.

6.1. The PED example using the `OpenRISA` *tool*

We have developed the PED example following the RISA method using the OpenRISA tool. Note that the example is mainly used for the illustration purpose of the overall tool-supported approach rather than for the validation of the approach itself.

6.2. Step 1 to Step 3

The PED documentation (Card Payment Group, 2003; Drimer et al., 2008; Mastercard, 2004) allowed us to identify the PED overall functional requirement (FR) – step 1, security goal (SG) – step 2, and two security requirements (SR1) and (SR2) – step 3:

(FR1) Allow consumers to pay at Points-Of-Sale with PIN
(SG) Protect the PIN
(SR1) PIN entered by consumers shall remain confidential during payment transactions at Points-Of-Sale
(SR2) PIN entered by consumers shall remain accurate during payment transactions at Points-Of-Sale (i.e. integrity of the PIN shall be preserved)

Also part of step 3, the system context W in the entailment (1) is elaborated. The functional requirement of the PED helps us to delimit the context; from (FR1), we identify five domains: consumer, card, merchant, terminal and bank. Fig. 13 shows the context of the PED system and its security requirements. The notation is unusual in two ways: (i) it treats the PED system as a machine with its own components, and (ii) it represents shared phenomena by directed arrows. In adopting (ii) we show graphically that a shared phenomenon is controlled by one domain and observed by another (Jackson, 2001). Therefore, the domain B with the arrow head observes the phenomenon p controlled by the domain A without the arrow head. Textually, this is represented by $A!p$. Notice that the diagram shows the shared phenomena related not only to PIN, but also to the card details and the transaction value, which are relevant to the PED payment transactions.

Note that the assignment of the formal propositional formula to risks and mitigations still need to be done manually because it is easy for automation to introduce errors. Therefore, these steps cannot be fully automated. On the other hand, an explicit documentation of the formula and automated reasoning tool support make it possible to investigate the intended logic meanings behind the informal arguments to diagnose surprising outcomes.

6.3. Step 4: Construct Outer Arguments

According to the entailment (2) and derived from the PED overall behaviour from the moment the consumer enters a PIN ($consumer!PIN$) until a payment

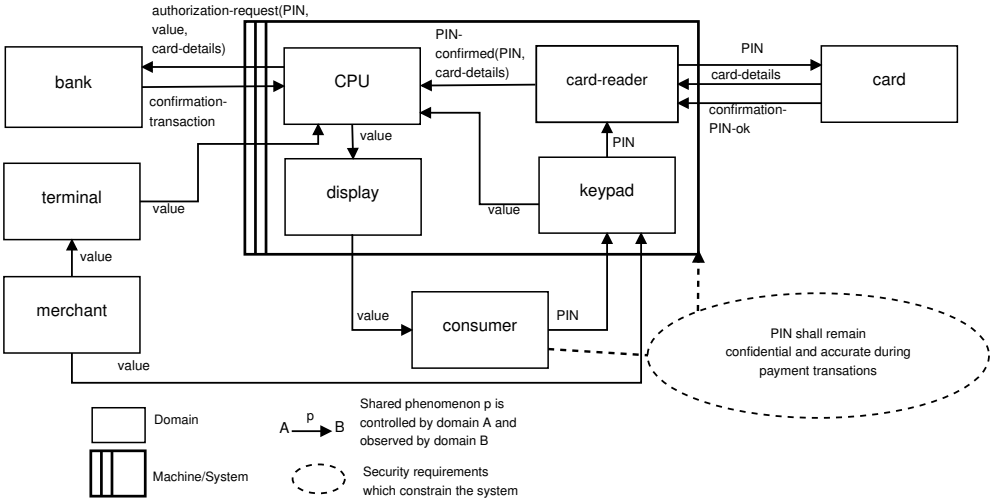

Figure 13: System context of the PED system and its security requirements

transaction is confirmed by the bank ($bank!confirmation-transaction$), we have the following outer argument:

$$(\text{consumer!PIN} \implies \text{bank!confirmation-transaction}) \vdash (\text{SR1} \wedge \text{SR2}) \qquad (4)$$

Premises P1–P6 are warrants that argument *consumer!PIN* $\vdash$ *bank!confirmation-transaction* is true and therefore that SR1 and SR2 are satisfiable.

Warrants P1–P6, derived from Fig. 13, are expressed in the OpenRISA tool notation as follows. Note that for the illustration purposes, we show the premises by proposition labels with an index starting from 1 instead of 0, while in the implementation it is easier to start these as an array indexed from 0.

Although there can be more than one sequence of events when explaining the behaviour of the context diagram, a rebuttal only requires one instance of such to demonstrate that the claim does not hold.

```
1  argument: PED-example

3  boolean A1, A2, P1, P2, P3, P4, P5, P6

5  A1 "Confidentiality and integrity of PIN are preserved" round 1 {
6     supported by
7        A2 "consumer!PIN -> bank!confirmation-transaction" round 1 {
8        warranted by
9           P1 "consumer!PIN -> keypad!PIN" round 1
10          P2 "keypad!PIN -> card-reader!PIN" round 1
11          P3 "card-reader!PIN -> card!confirmation-PIN-ok" round 1
12          P4 "card!confirmation-PIN-ok -> card-reader!PIN-confirmed"
13             round 1
14          P5 "card-reader!PIN-confirmed -> CPU!authorization-request"
15             round 1
16          P6 "CPU!authorization-request -> bank!confirmation-transaction"
17             round 1
18       }
19 }
```

Table 1: Keywords related to premises P1–P3 used to search both catalogues in the PED example

Domains/ phenomena	**Keyword**
consumer	social victim
PIN	PIN password encrypt unencrypt cryptograph
PED	tamper physic log
card	card

6.4. Step 5: Identify Risks

Using domains and shared phenomena from the system context diagram (e.g., Fig. 13), we specify a list of keywords which is not exhaustive to demonstrate the search of catalogues. (Table 1) is an extract of the keywords used to search the catalogues.

The choice of keywords related to domains and shared phenomena depends on human judgement. However, the process can be made more systematic if supported by guide-words inspired by the HAZOP technique. HAZOP (HAZard and OPerability) has been used for decades for the analysis of hazards and operational threats in the safety field (Ericson, 2005). More recently, HAZOP has been adapted with guide-words tailored for the security field (Winther et al., 2001). Pre-defined guide-words associated with relevant terms (e.g., password, human subject, payment) could be created based on terms which often appear in security catalogues.

From the analysis of CWE and CAPEC entries returned by the search functionality of the OpenRISA, a list of 35 risks and 19 mitigations were identified for the PED example, after eliminating the false positives returned by the keywords search according to the domain expertise . These risks invalidate the satisfaction of this security argument A1 by rebutting its warrants P1–P6 in practice. For instance, risk R1 rebuts argument A1 (premise P1) and is warranted by CAPEC-455 and CAPEC-89, While risk R8 rebuts argument A1 (premises P2 and P5) and is warranted by CWE-311, CWE-325 and CAPEC-439.

Risks are expressed in the OpenRISA tool notation as follows; note that this listing complements the previous one defining A1.

6.5. Steps 6 and 7: Identify, Classify & Consolidate Mitigations

The CAPEC and CWE entries describe possible mitigations. For instance, from the catalogues we derived the following mitigations to rebut risks R1 and R8, restoring argument A1, and expressed in the OpenRISA tool notation; note that this listing complements the previous one defining risks (round 2).

```
argument: PED-example

boolean A1, R1, ... R35

A1 rebutted by R1, ..., R35

R1 "PIN is collected by fake PED set to allow pharming attack" round 2 {
  warranted by
     CAPEC-455 "CAPEC-455"
     CAPEC-89 "CAPEC-89"
}
...
R8 "PIN is revealed if sent unencrypted within the PED and the PED can
    be tampered" round 2 {
  warranted by
     CWE-311 "CWE-311"
     CWE-325 "CWE-325"
     CAPEC-439 "CAPEC-439"
}
...
```

```
argument: PED-example

boolean R1, ..., R35, M1, ..., R19

R1 mitigated by M1, M2, M3
R8 mitigated by M5, M6, M7
...
M1 "PED should use secure connection (encryption) to handle PIN between
    the moment it is entered by the consumer until it reaches the PED
    keypad" round 3 {
}
M2 "PED should have auditing mechanisms to log device replacements,
    keypad input and any event affecting its integrity" round 3 {
}
M3 "PED should have authentication mechanisms to restrict replacement &
    deployment procedures to be performed only by authorized technicians"
    round 3 {
}
M5 "Transmission of PIN within PED, between PED & bank and PED & card
    should use well-vetted encryption algorithms, and well-tested
    implementations of these algorithms" round 3 {
}
M6 "PED design should allow upgrade of cryptographic algorithm for
    communication of PIN within PED, between PED & bank and PED & card"
    round 3 {
}
M7 "Requirements for the selection of an encryption algorithm and its
    implementation should be clearly stated for the PED" round 3 {
}
...
```

6.6. Step 8: Prioritise Risks

The catalogues provide off-the-shelf support for the estimation of risk level based on expert ratings for likelihood and impact in terms of confidentiality, integrity and availability. However, these ratings must be customized and depends on expert judgement. The RISA method can be used in conjunction with any technique that estimates risk level – quantitative, semi-quantitative or even qualitative (where high, medium, low is assigned to risk level).

7. Discussions

We organise our discussions around two areas: the catalogues search from experience gained with the PED example and short-term future work.

7.1. Systematic Search of Catalogues

The RISA method takes advantage of an automated keyword-based search for the CAPEC and CWE security catalogues, using pre-defined fields converted from XML to text. The systematic and repeatable search process incorporated to the RISA method allowed us to identify 35 risks from the analysis of 207 attack patterns and weaknesses. While the *ad hoc* search applied to the same example, as reported in (Franqueira et al., 2011), allowed us to identify only 9 risks, based on the analysis of a small number of catalogue entries. Despite this significant improvement in catalogues coverage provided by the new search functionality, several challenges remain.

There are infinite possibilities of keywords which can be used to search the catalogues related to a same premise. We partially addressed this issue by applying the Porter Stemming algorithm (Porter, 1980) to remove common English suffixes. Therefore, keywords provided are automatically reduced to stems (when applicable), increasing the coverage of the catalogues while decreasing the number of keywords. Nevertheless, translating an information need into a searchable query is not necessarily straight forward (Borgman, 1996). We reflect on our experiences using RISA's catalogue search in terms of precision and recall illustrated by examples.

Keyword *encrypt* uncovers weakness CWE-807 (Reliance on Untrusted Inputs in a Security Decision) used as reference for risk R1.18. However, weakness CWE-20 (Improper Input Validation) would be even more appropriate as a reference for this risk. CWE-20 was not found because its description summary and extended description contained none of the keywords searched. This issue may affect the identification of mitigations and the prioritisation of risks.

RISA's search using keyword *log* returns catalogue entries related to *log file(s)*, *audit log(s)* and *logging*, which represent true-positives and should be analysed. Yet, it also returns false-positives related to, e.g., *log in* or *logged*. This raises a question about whether to increase the precision of keywords (e.g., using compound keywords), or copping with an increased number of search results. We used the off-the-shelf Lucene search engine to compute the relevance

of queries (keywords selected from outer argument) and documents (public catalogues), therefore it has room for missing matches, in addition to the risk of generalisation/specialisation mismatches.

There are other factors which influence the choice of keywords, such as awareness of security jargon, familiarity with the catalogues themselves, and grasp of terms related to the system domain, and technologies involved. Apart from Porter' stems, we proposed the use of guided-words, inspired by HAZOP (Winther et al., 2001; Ericson, 2005), as another way to make the choice of keywords more systematic. However, all in all, the choice of keywords affecting search results is an intrinsic problem in several other domains as well, such as in Web search. But, since end-users practice and training may minimise this issue in online search (Kim, 2001), we expect the same to happen with requirements engineers.

7.2. Future Work

After the discussions, we identify two areas of further research, as well as a list of short-term improvements to RISA.

7.2.1. Feedback from the Risk Assessment Steps to the Requirements Satisfaction Steps

Although OpenRISA provides mechanisms for maintaining traceability between arguments and requirement models, they can enhance in a number of ways. First, risks assessment can identify new problem world domains, changes in behaviour and properties of the problem world domains, and even new security requirements. Such new knowledge has to integrate with the requirements models. This is currently done manually, but tool support for this task will be helpful.

Second, since there can be a number of rebuttals and mitigations to an argument, the size of argument models often increase quickly after a few rounds of argumentation. It is difficult to visualise large argument models, and heuristics for partitioning arguments are necessary in OpenRISA.

Third , the keywords general or specific to security domain need to be collected to guide the search of the risk for the argumentation-based elicitation. We hope to extend the current ad hoc approach with HAZOP-like guided-words (as mentioned in Section 6.4) based on domain ontology that has been introduced by the effort of the entire research community, and based on terms that often appear in the security catalogues.

Finally, we aim to integrate other requirements languages in order to support their integration with risk assessment frameworks. For instance, we have developed a tool supporting the Problem Frames approach, called OpenPF (Tun et al., 2009). One way to integrate the two tools will be by means of graphical traceability. The integrated tool can allow each risk and mitigation to be explicitly linked with a problem world domain of a problem diagram drawn in OpenPF. Similarly, each problem world domain in the problem diagram can link to an argument or any part of it, maintaining the traceability between problem diagrams and argument diagrams.

7.2.2. Risk-based Prioritisation of Arguments

Currently, RISA prioritizes arguments based on risk level depending on experts' estimation of risks. Security catalogues supporting RISA provide some pre-defined, generic, ratings of likelihood and impact on confidentiality, integrity and availability but they still need customisation. Prioritization via risk level fits into a risk-averse perspective which favours risk over, for instance, the cost of implementing mitigations. However, prioritisation could follow a risk-taking perspective where the cost of mitigations is more important than risks, or a perspective in-between which gives importance to both risks and mitigations taking cost/benefit into consideration (Franqueira et al., 2010, e.g.,). Although the current version of the OpenRISA tool already supports priority value for mitigations, more development is necessary to fully operationalise these prioritisation possibilities.

Another approach to overcome this risk-only prioritisation would be to adopt a value-based argumentation approach, where a value is associated with an argument and affects the conclusion about a claim. This accounts for the fact that stakeholders have different priorities, which might conflict. Adding value to arguments is a way to make these priorities explicit and, therefore, expose them to scrutiny and criticism (Graydon and Knight, 2008). For instance, Burgemeestre et al. (2010) consider, in their example, arguments flagged with the values *safety* and *auditability*, Baroni et al. (2009) consider *safety* and *cost*, while Bench-Capon (2003) discusses a moral debate where an ordering on values allows distinct audiences to set different preferences, e.g., life has priority over property or the other way round.

Value-based argumentation applied to RISA would be helpful, for example, to allow different prioritisations of arguments based on preferences among different values associated with mitigations, such as cost of implementing them and their operational cost, or even different values associated with risks, such as likelihood, and impact. Such direction requires changes in the OpenRISA tool to support more than one priority value for arguments.

7.2.3. Further Enhancement for Adoption

Presenting all increments of argumentations in one diagram may not be the most scalable solution when the diagram is large, even though the textual input can support very large structures. The diagram part of the OpenRISA tool may improve with learning special-purpose visualisations such as tree-based view of arguments in (Cyra and Górski, 2011).

Apart from visualisations, there are other ways to elicit arguments in the graphic representation of ASPIC+, e.g., (Prakken et al., 2013) proposed to obtain them from a participatory game. The argumentation format proposed by (Ionita et al., 2014) stripped the representation much down from ASPIC+, which have also received positive adoption in practice. The OpenRISA representation has a great potential to present a simplified view to the practitioners.

Furthermore, the propositional logic reasoning behind the formal argumentation needs to be understandable for the users. In (Yu et al., 2014), we have

shown that a deductive theorem prover can provide provenance on how to deduce the conclusion of risks and mitigations from the grounds and warrants. A future work is to provide more traceability in the tool for such explanations on the fly.

8. Conclusion

Argumentation approaches organise the evidence for or against the claims of software security. They strike a balance between perfect security and practical limitations. This paper has proposed a tool, OpenRISA, in support for using argumentation and risk assessment together to reason about the satisfaction of security requirements. OpenRISA has three main features. First, it supports representing both argumentation and security risk assessment in an integrated modelling language. Second, it provides an automated search for publicly available catalogues of common attacks and weaknesses, namely CWE and CAPEC, as evolving sources for security risk assessment. Third, it checks the soundness of the formalised arguments challenged by the security risk assessment. The tool has been demonstrated through an example of PIN Entry Device system, part of which has been illustrated in the paper.

Acknowledgement

The work is supported in part by the ERC Advanced Grant 291652 (Adaptive Security And Privacy, http://asap-project.eu), the SFI grant 03/CE2/I303_1, and Sentinels (http://www.sentinels.nl). We would like to thank our colleague Paul Piwek for feedback on earlier draft of the paper.

Atkinson, K., Bench-Capon, T., McBurney, P., 2004. Justifying Practical Reasoning. In: CMNA'04. pp. 87–90.

Baroni, P., Cerutti, F., Giacomin, M., Guida, G., 2009. An Argumentation-Based Approach to Modeling Decision Support Contexts with What-If Capabilities. In: AAAI Fall Symposium. Technical Report SS-09-06. AAAI Press, pp. 2–7.

Bench-Capon, T. J. M., 2003. Persuasion in Practical Argument Using Value-based Argumentation Frameworks. Journal of Logic and Computation 13 (3), 429–448.

Bollobas, B., October 2002. Modern Graph Theory, 2nd Edition. Springer Press.

Borgman, C. L., July 1996. Why Are Online Catalogs Still Hard to Use? Journal of the American Society for Information Science 47, 493–503.

BS ISO/IEC 27005, 2011. Information technology – Security techniques– Information security risk management.

Burgemeestre, B., Hulstijn, J., Tan, Y.-H., 2010. Value-Based Argumentation for Justifying Compliance. In: DEON'2010. Springer, pp. 214–228.

Card Payment Group, Jul 2003. PIN Entry Device Protection Profile. Common Criteria Portal, www.commoncriteriaportal.org/files/ppfiles/PED_PPv1_37.pdf, last visited Jul 2010.

Cohen, R., January 1987. Analyzing the Structure of Argumentative Discourse. Computational Linguistics 13, 11–24.

Cyra, L., Górski, J., 2007. Supporting Compliance with Safety Standards by Trust Case Templates. In: Proc. of the ESREL'07 (European Safety and Reliability) Conference: Risk, Reliability and Societal Safety. Vol. 2. Taylor & Francis Ltd, pp. 1367–1374.

Cyra, L., Górski, J., 2011. Support for argument structures review and assessment. Rel. Eng. & Sys. Safety 96 (1), 26–37.
URL http://dx.doi.org/10.1016/j.ress.2010.06.027

Drimer, S., Murdoch, S. J., Anderson, R., May 2008. Thinking Inside the Box: System-Level Failures of Tamper Proofing . In: SP'2008. IEEE Press, pp. 281–295.

Dung, P. M., September 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and N-person Games. Artificial Intelligence 77, 321–357.

Ericson, C. A., 2005. Hazard Analysis Techniques for System Safety. Wiley-Interscience.

Ernst, N., Yu, Y., Mylopoulos, J., Sept 2006. Visualizing non-functional requirements. In: REV '06: First International Workshop on Requirements Engineering Visualization. pp. 2–2.

Franqueira, V. N. L., Houmb, S., Daneva, M., 2010. Using Real Option Thinking to Improve Decision Making in Security Investment. In: IS'2010 (OTM Conferences). LNCS. Springer Press, pp. 619–638.

Franqueira, V. N. L., Tun, T. T., Yu, Y., Wieringa, R., Nuseibeh, B., 2011. Risk and Argument: A Risk-based Argumentation Method for Practical Security. In: RE'11 Proceedings. IEEE Press, pp. 239–248.

Graydon, P., Knight, J., July 2008. Success Arguments: Establishing Confidence in Software Development. Tech. Rep. CS-2008-10, University of Virginia.

Hakemi, A., Jeong, S. R., Ghani, I., Sanaei, M. G., 2014. Adapting OCTAVE for risk analysis in legacy system migration. TIIS 8 (6), 2118–2138.
URL http://dx.doi.org/10.3837/tiis.2014.06.018

Haley, C., Laney, R., Moffett, J., Nuseibeh, B., 2008. Security Requirements Engineering: A Framework for Representation and Analysis. IEEE Trans. Softw. Eng. 34 (1), 133–153.

Hatcher, E., Gospodnetic, O., 2004. PLucene in Action (In Action series). Manning Publications Co., Greenwich, CT, USA.

Huhn, M., Zechner, A., 2010. Arguing for Software Quality in an IEC 62304 Compliant Development Process. In: ISoLA'10: Proc. of the 4th International Conference on Leveraging Applications of Formal Methods, Verification, and Validation - Volume Part II. Springer-Verlag Press, pp. 296–311.

Ionita, D., Bullee, J.-W., Wieringa, R., Aug 2014. Argumentation-based security requirements elicitation: The next round. In: Evolving Security and Privacy Requirements Engineering (ESPRE), 2014 IEEE 1st Workshop on. pp. 7–12.

Jackson, M., 2001. Problem Frames: Analysing and Structuring Software Development Problems. Addison-Wesley/ACM Press.

Kelly, T. P., 1998. Arguing Safety - A Systematic Approach to Safety Case Management. Ph.D. thesis, University of York.

Kim, K.-S., 2001. Information-seeking on the Web: Effects of user and task variables. Library and Information Science Research 23 (3), 233–255.

Lipson, H., Weinstock, C., May 2008. Evidence of Assurance: Laying the Foundation for a Credible Security Case, department of Homeland Security; online: `https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/assurance/973-BSI.html`, last visited Feb 2011.

Lund, M. S., Solhaug, B., Stlen, K., 2011. Model-driven risk analysis - the coras approach. pp. I–XVI, 1–460.

Mastercard, October 2004. Payment Card Industry POS PIN Entry Device Security Requirements. m2m Group website, `http://www.m2mgroup.ma/livresetdocs/security%20risk.htm`, last visited Feb 2011, version 7 1.0, Revised March 2005.

Mueller, E. T., 2011. The Discrete Event Calculus Reasoner. `http://decreasoner.sourceforge.net/`.

Newman, S. E., Marshall, C. C., 1991. Pushing Toulmin Too Far: Learning from an Argument Representation Scheme. Tech. Rep. SSL-92-45, Xerox PARC.

Porter, M. F., July 1980. An algorithm for suffix stripping. Program 14 (3), 130–137.

Potts, C., Bruns, G., 1988. Recording the reasons for design decisions. In: ICSE'88. IEEE Press, Los Alamitos, CA, USA, pp. 418–427.

Prakken, H., Ionita, D., Wieringa, R., 2013. Risk assessment as an argumentation game. In: Leite, J., Son, T., Torroni, P., van der Torre, L., Woltran, S. (Eds.), Computational Logic in Multi-Agent Systems. Vol. 8143 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 357–373. URL http://dx.doi.org/10.1007/978-3-642-40624-9_22

Raspotnig, C., Opdahl, A., 2013. Comparing Risk Identification Techniques for Safety and Security Requirements. Systems and Software 86, 1124–1151.

Shum, S. B., Hammond, N., 1994. Argumentation-based Design Rationale: What Use at What Cost? Int. Journal of Human-Computer Studies 40 (4), 603–652.

Sindre, G., Opdahl, A. L., 2005. Eliciting security requirements with misuse cases. Requirements Engineering Journal 10 (1), 34–44.

SP 800-30 rev. 1, 2012. Guide for Conducting Risk Assessments. Published by NIST: http://csrc.nist.gov/publications/nistpubs/800-30-rev1/sp800_30_r1.pdf.

Toulmin, S., Rieke, R., Janik, A., 1979. An Introduction to Reasoning. Macmillan.

Tun, T. T., Yu, Y., Laney, R. C., Nuseibeh, B., 2009. Early identification of problem interactions: A tool-supported approach. In: Glinz, M., Heymans, P. (Eds.), REFSQ. Vol. 5512 of Lecture Notes in Computer Science. Springer, pp. 74–88.

Walton, D. N., 1996. Argumentation Schemes for Presumptive Reasoning. Lawrence Erlbaum Associates, Mahwah NJ, USA.

Winther, R., Johnsen, O., Gran, B. A., 2001. Security Assessments of Safety Critical Systems Using HAZOPs. In: SAFECOMP'01. Springer-Verlag Press, pp. 14–24.

Yu, Y., Piwek, P., Tun, T. T., Nuseibeh, B., 2014. Towards explaining rebuttals in security arguments. In: 14th Workshop on Computational Models of Natural Argument, 10 December 2014, Krakow, Poland.

Yu, Y., Tun, T. T., Tedeschi, A., Franqueira, V. N. L., Nuseibeh, B., 2011. OpenArgue: Supporting Argumentation to Evolve Secure Software Systems. In: RE'11. IEEE press, pp. 351–352.