Graphical Modeling of Security Arguments: Current State and Future Directions

Dan Ionita[1(✉)], Margaret Ford[2], Alexandr Vasenev[1], and Roel Wieringa[1]

[1] Services, Cybersecurity and Safety group, University of Twente, Drienerlolaan 5, 7522 NB Enschede, The Netherlands
{d.ionita,r.j.wieringa}@utwente.nl
[2] Consult Hyperion, 10-12 The Mount, Guildford GU2 4HN, UK
margaret.ford@chyp.com

Abstract. Identifying threats and risks to complex systems often requires some form of brainstorming. In addition, eliciting security requirements involves making traceable decisions about which risks to mitigate and how. The complexity and dynamics of modern socio-technical systems mean that their security cannot be formally proven. Instead, some researchers have turned to modeling the claims underpinning a risk assessment and the arguments which support security decisions. As a result, several argumentation-based risk analysis and security requirements elicitation frameworks have been proposed. These draw upon existing research in decision making and requirements engineering. Some provide tools to graphically model the underlying argumentation structures, with varying degrees of granularity and formalism. In this paper, we compare these approaches, discuss their applicability and suggest avenues for future research. We find that the core of existing security argumentation frameworks are the links between threats, risks, mitigations and system components. Graphs - a natural representation for these links - are used by many graphical security argumentation tools. But, in order to be human-readable, the graphical models of these graphs need to be both scalable and easy to understand. Therefore, in order to facilitate adoption, both the creation and exploration of these graphs need to be streamlined.

Keywords: Risk assessment · Security requirements
Argumentation · Graphical modeling

1 Introduction

Complete security is impossible and security decisions have to be selective: some risks can be mitigated in several ways while others will have to be accepted. Thus, security decision making involves an opportunity cost - the loss of the value that would have been realized by making an alternative decision.

The ability to trace back previous decisions is important if they have to be defended or revised, or if new security decisions have to taken. Firstly, the decision maker may have to justify mitigation decisions made earlier, for instance

© Springer International Publishing AG 2018
P. Liu et al. (eds.): GraMSec 2017, LNCS 10744, pp. 1–16, 2018.
https://doi.org/10.1007/978-3-319-74860-3_1

in the case of a successful attack [16] or to satisfy the "reasonable security" requirements of regulators [4]. Second, the ever changing security landscape and forces decision makers to frequently revisit security decisions. In fact, the new European GDPR (General Data Protection directive) explicitly requires data controllers and processors to ensure "ongoing" confidentiality, integrity, availability and resilience of processing systems and services [12, art 32(1)(b)]. Third, related systems may face related but not identical risks and therefore, reusing (parts of) the arguments made for similar systems facilitates decision making for given systems [13]. By recording the argumentation behind security decisions, risk assessments can be re-visited when an attack takes place, extended when new risks surface, and re-used in related products or contexts. Altogether, this highlights a need to document security decisions and the rationale behind them.

With respect to previous research, security arguments can be compared to safety cases [1,9,25], in that they summarize the reasons why, and the extent to which, a system is thought to be acceptably secure (or safe). Several techniques for modeling security arguments exist, some inspired from legal argumentation (Toulmin-like argumentation structures [19,22]), others from formal methods (deontic logic [15], defeasible logic [35]). The various approaches differ in their scope and applicability. However, security argumentation schemas have only been applied to toy examples so far and have not yet been adopted (or even evaluated) in practice, raising questions with regard to their practical *usability, utility and scalability.*

A characteristic feature of security risk assessments is that stakeholders with varying backgrounds must contribute to or check the assessment, and that these stakeholders have no time to first learn a specialized language for argumentation. Therefore, we argue difficulties in understanding the notation used to represent security arguments are major threats to the *usability* of any security argumentation methodology and that some form of graphical model is needed in order to enhance understanding. However, a graphical representation with low expressiveness may have reduced *utility* while one which is too granular may face *scalability* issues. This paper looks at argumentation models have evolved over time and cross-examines graphical security argumentation frameworks in order to support researchers in advancing the graphical modeling of security arguments, as well as inform specialists involved in the security requirements elicitation or risk assessment about existing security argumentation frameworks and tools.

We start in Sect. 2 with reviewing argumentation theory and its application in the security domain. In Sect. 3 we review the graphical representation of security arguments provided by graphical argumentation tools available at the time of writing. In Sect. 4 we cross-compare the various representations employed by each of the tools and draw conclusions with regard to their expressiveness and applicability. In Sect. 5 we draw conclusions from the comparison and indicate some topics for future work in the direction of making security argumentation graphs more practically usable.

2 Background

Structured argumentation has its roots in legal reasoning, with examples of diagrams being used to capture the justifications of judges or juries dating back to as early as 1913 [17]. With the advent of computers, attempts to capture reasoning first behind design decisions and later behind decisions in general also proliferated [8,14,26]. Significant effort was invested into developing complex tools and even more complex approaches for automated decision support [27,33]. However, it quickly became apparent that capturing arguments is most useful when no formal proof is possible but defensibility of the decision is required [30]. This is of course the case for legal reasoning, but a similar situation exits in the fields of safety and security. Indeed, safety arguments (in the form of safety cases) were quickly adopted by the industry as a standard way of claiming their systems are safe [7]. With laws such as the European DPD (Data Protection Directive) and, in the US, Sect. 5 of the Federal Trade Comission (FTC) act requiring companies to show that they took reasonable steps in protecting their customer's data, argumentation-based approaches started finding their way into the field of information security. The remainder of this section explores the evolution of argumentation structures from the court of law to their more recent incarnations in security requirements engineering and in risk assessment. We leave the discussion of the graphical representations of these arguments for the next sections.

2.1 Argumentation Modeling

Stephen Toulmin laid the foundations for modeling arguments in his 1958 book The Uses of Argument [41]. He proposed subdividing each argument into six components (as shown in Fig. 1): a central *claim*, some *grounds* to support that claim, a *warrant* connecting the claim to the evidence, a factual *backing* for the warrant, a *qualifier* which restricts the scope of the claim and finally a *rebuttal* to the claim. He later identified applications of his framework in legal reasoning [40].

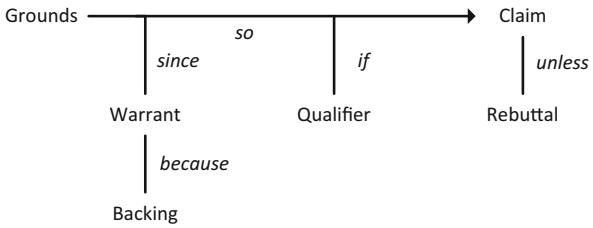

Fig. 1. The Toulmin argument structure

In the late 1980's and early 90's, argumentation models started being used to support design decisions. Specifically, the emerging field of design rationale began investigating ways to capture how one arrives at a specific decision, which alternate decisions were or should have been considered, and the facts and

assumptions that went into the decision making [26]. In 1989 MacLean et al. [28] introduced an approach to represent design rationale which uses a graphical argumentation scheme called QOC (for Questions, Options and Criteria) - depicted in Fig. 2. Buckingham Shum et al. [38] later showed how the QOC notation can be used as a representative formalism for computer-supported visualization of arguments, with applications in collaborative environments. Mylopoulos et al. [31] introduced Telo, a language for representing knowledge about an information system intended to assist in its development. Similarly, Fischer et al. [14] claim that making argumentation explicit can benefit the design process itself.

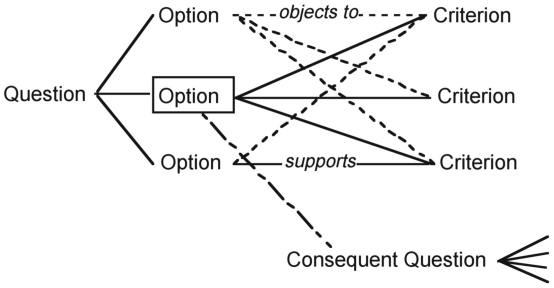

Fig. 2. The Questions, Options and Criteria (QOC) graphical argumentation scheme

Soon, modeling of arguments found even wider applications in decision making - especially when related to critical systems - where they started being used to make expert judgment explicit, usually by means of so-called 'cases' [8]. Safety cases, for instance, are structured arguments, supported by evidence, intended to justify that a system is acceptably safe for a specific application in a specific operating environment [9]. These arguments should be clear, comprehensive and defensible [25]. Two established approaches to safety cases are the CAE (Claims Arguments Evidence) notation [10] and the GSN (Goal Structuring Notation) [24].

Both approaches prescribe a graphical representation of the argumentation structure but differ in terms of what this structure contains. The CAE was developed by Adelard, a consultancy, and views safety cases as a set of *claims* supported by *arguments*, which in turn rely on *evidence*. Although these concepts are expressed using natural language, the cases themselves are represented as graphs and most implementations suggest their own graphical symbols. Figure 4 shows the CAE representation used by the Adelard's own ASCE tool [1]. The GSN (Fig. 3) was developed by the University of York and provides a more granular decomposition of safety arguments into *goals, context, assumptions, strategy, justifications* and *solutions* [24]. The arguments are also represented as a graph, with one of two types of links possible between each pair of nodes: (1) a decompositonal *is solved by* between a goal and one or more strategies or between a strategy and one or more goals, as well as (2) a contextual *in context of*

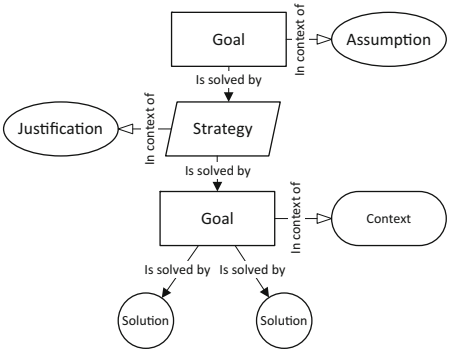

Fig. 3. The Goal Structuring Notation (GSN)

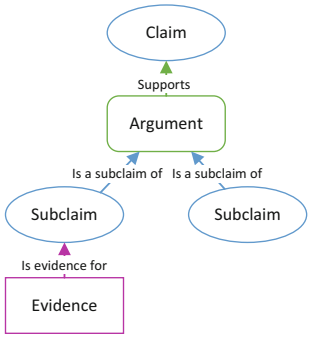

Fig. 4. The Claims Arguments Evidence (CAE) notation

between a goal, strategy or solution and an assumption, justification or context. The notation comes with a well defined graphical language which - according to its creator - attempts to strike a balance between power of expressiveness and usability [25].

Other, more general representations such as concept maps [29], mindmaps [2] or generic diagrams can of course also be used to represent and share knowledge, including arguments [11]. These representations have no (formal or informal) argumentation semantics and we ignore them in the rest of the paper.

2.2 Argumentation in Security

The success of safety cases has inspired other similar approaches, such as trust cases [18], conformity cases [8] and, in the field of security, assurance cases [3,32] used to show satisfaction of requirements and misuse cases [39] used to elicit security requirements. Similarly, argumentation schemes for design rationale have been adapted to provide support for security decisions. Recently, argumentation modes have been used to encode the entire risk assessment process, from risk identification to countermeasure selection. This subsection provides an overview of these applications.

Arguing Satisfaction of Security Requirements. Assurance cases are an argumentation-based approach similar to the safety cases described in Sect. 2.1. They use structured argumentation (for instance using the GSN or CAE notations) to model the arguments of experts that a system will work as expected. However, while safety cases only make claims pertaining to the safe operation of a system, assurance cases are also concerned with other important system functions, in particular security and dependability [37].

Haley et al. [21] laid the groundwork for an argumentation framework aimed specifically at validating security requirements. It distinguishes between *inner* and *outer* arguments. Inner arguments are formal and consist mostly of claims

about system behavior, while outer arguments are structured but informal and serve to justify those claims in terms of trust assumptions. Together, the two form a so-called "satisfaction argument".

Supporting the Elicitation of Security Requirements. Misuse cases - a combination of safety cases and use cases - describe malicious actions that could be taken against a system. They are used to identify security requirements and provide arguments as to why these requirements are important [39].

Rowe et al. [36] suggest using argumentation logic to go beyond formalizing domain-specific reasoning and automatically reason about security administration tasks. They propose decomposing each individual argument into a Toulmin-like structure and then representing defeasability links between the arguments as a graph. This would allow both encoding unstructured knowledge and applying automated reasoning, for example by using theorem provers. They suggest two applications: attack diagnosis, where experts argue about the root-cause of an attack, and policy recommendation, where security requirements are elicited.

Haley et al. [20] built their conceptual framework for modeling and validating security requirements described in [21] into a security requirements elicitation process, which can help distill security requirements from business goals. The same authors later integrated their work on modeling and elicitation of security requirements into a unified framework for security requirements engineering [19]. The framework considers the context, functional requirements and security goals before identifying security requirements and constructing satisfaction arguments for them. However, it does not consider the risks the system may or may not be facing when not all security requirements are satisfied, or when not all security goals are achieved.

Argumentation-Based Risk Assessment. Franqueira et al. [15] were among the first to propose using argumentation structures to reason about both risks and countermeasures in a holistic fashion. OpenArgue (discussed in Sect. 3.1) supports the construction of argumentation models. Their proposed method, RISA (RIsk assessment in Security Argumentation) links to public catalogs such as CAPEC (Common Attack Pattern Enumeration and Classification) and the CWE (Common Weakness Enumeration) to provide support for security arguments using simple propositional logic. The method does not consider the possibility that a security threat may not be totally eliminated. Later, Yu et al. [43] integrated the RISA method and Franqueira's argumentation schema into a unified argumentation meta-model and implemented it as part of tool - OpenRISA - which partly automates the validation process. This tool is discussed in Sect. 3.1.

Prakken et al. [35] proposed a logic-based method that could support the modeling and analysis of security arguments. The approach viewed the risk assessment as an argumentation game, where experts elicit arguments and counter-arguments about possible attacks and countermeasures. Arguments derive conclusions from a knowledge base using strict or defeasible inference

rules. The method is based on the ASPIC+ framework [34] and uses defeasible logic. This restricts its usability in practice.

Prakken's solution inspired a simplified approach, which used spreadsheets to encode and analyze the arguments [22]. Each argument was decomposed into only a *claim* and one or more supporting *assumptions* or *facts*. Similar to Prakken's approach, any argument could counter any other argument(s) and formulas (this time built-into the spreadsheets) were used to automatically compute which arguments were defeated and which were not. Argumentation spreadsheets are discussed in detail in Sect. 3.2.

Later, a dedicated tool was developed which employed the same simplified argument structure but without differentiating between assumptions and facts. However, most arguments were found to refer to attacks, while most counter-arguments proposed countermeasures to attacks. To simplify this, and further improve usability, an online version was developed which also flattened the inter-argument structure by only allowing counter-arguments to refer to countermeasures. These tools are part of the ArgueSecure family, discussed in Sect. 3.3.

3 Graphical Security Argumentation Tools and Techniques

Both (formal) first order logic and (informal) structured argumentation provide methods for analyzing the interaction between arguments. However, structured argumentation also provides a foundation for presenting arguments for or against a position to a user [36]. Of the argumentation notations reviewed above, we now zoom in on those having a graphical representation, and discuss this representation in more detail by applying them to the same sample scenario. Graphical security argumentation modeling tools mainly differ in the amount of detail they use to describe the structure of, and links between, arguments. Therefore, we evaluate each tool on its expressiveness in terms of intra-argument granularity E1 (i.e. the number of components an argument has to be decomposed into), inter-argument granularity E2 (i.e. how many types of rebuttals are possible). Tools also provide secondary functionality, usually aimed at improving usability and scalability. We therefore also identify relevant features provided by each tool (labelled F1–F6) and summarize everything in a cross-comparison table (Table 1), to be discussed in Sect. 4.

3.1 OpenArgue/OpenRISA

OpenArgue is an argumentation modeling tool featuring both a syntax editor and a graphical editor, which comes with the ability to derive an argumentation diagram from a textual specification [42]. OpenArgue assumes security requirements are known at the time of analysis and focuses on identifying ways by which these requirements could be invalidated. This means all arguments are linked to a specific security requirement (F1). It benefits from syntax highlighting as well as a built-in model checker which can identify formal inconsistencies in the

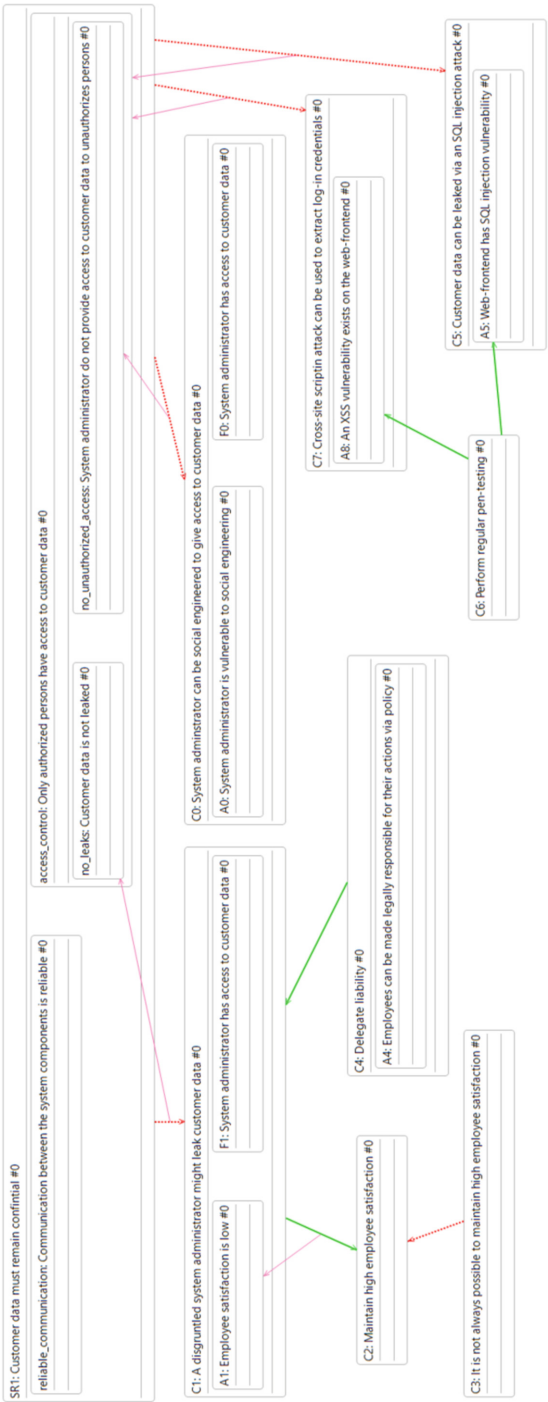

Fig. 5. OpenArgue - sample assessment

argumentation diagram. OpenArgue has a simplified Toulmin intra-argument structure consisting of a central *claim*, supported by *grounds*, the relevance of which is supported by *warrants* (E1 = 3). However, OpenArgue allows specifying rather complex inter-argument relationships: arguments can rebut or mitigate one or more other arguments (F3, F4) by challenging either their grounds or their warrants (E1= 4). This can lead to inter-twined graphical representations of the argumentation model that are hard to understand. This effect is amplified by the fact that the tool does not come with a custom editor but rather uses a generic Eclipse UML editor and thereby poses significant usability and scalability issues. Figure 5 shows a sample assessment built using OpenArgue.

OpenRISA is an extension of OpenArgue which can, in addition, check the argumentation model against online knowledge bases and verify that the risks identified are valid rebuttals.

3.2 Argumentation Spreadsheets

Tables have long served as a convenient means of storing and communicating structured data [6]. The argumentation spreadsheets attempt to decompose arguments into three elements: a claim, one or more assumptions and one or more facts [22] (E1 = 3). Each row encodes one argument divided across several columns. A screenshot of a sample assessment using argumentation spreadsheets is shown in Fig. 6.

ARGUMENTS										TAGS	
Claim		Assumptions		Facts		Re-buts	Asset(s)	Status	Notes	Assets	
#	txt	#	txt	#	txt		ID(s)	IN / OUT	Transf./ Red.	ID	NAME
C0	System adminstrator can be social engineered to give access to customer data	A0	System administrator is vulnerable to social engineering	F0	System administrator has access to customer data			IN		T1	policy
C1	A disgruntled system administratot might leak customer data	A1	Employee satisfaction is low	F1	System administrator has access to customer data			OUT		T2	web-frontend
C2	Maintain high employee satisfaction	A2	-	F2	-	A1		IN			
C3	It is not always possible to maintain high employee satisfaction	A3	-	F3	-	C2		OUT			
C4	Delegate liability	A4	Employees can be made legally responsible for their actions via policy	F4	-	C3	T1,	IN	Transf.		
C5	Customer data can be leaked via an SQL injection attack	A5	web-frontend has SQL injection vulnerability	F5	-		T2,	OUT			
C6	Perform regular pen-testing of the web-frontend	A6	-	F6	-	A5	T2,	IN	Red.		
C7	Cross-site scripting attack can be used to extract log-in credentials	A8	An XSS vulnerablity exists on the web-frontend	F8	-		T2,	OUT			
C8	Perform regular pen-testing of the web-frontend	A9	-	F9	-	A8	T2,	IN	Red.		

Fig. 6. Argumentation spreadsheets - sample assessment

An argument describes either a risk or a risk mitigation and can rebut one argument of an opposite type. This leads to a linear, attacker versus defender game-like process of filling in the table: first, a risk is described; then, either

the risk is accepted or a counterargument describing a mitigation is added; this back-and-forth rhetoric can continue until the risk is completely eliminated or the residual risk is accepted. The tool keeps track of each argument's state (IN for arguments without rebuttals or arguments whose rebuttal was defeated and OUT for arguments with an IN rebuttal), as well as automatically tagging arguments which mention one of the assets in the asset column (F2). Finally, the user can tag risk mitigation arguments with either a "Red." or a "Transf." tag signifying that the suggested countermeasure only partially mitigates the risk, or that it transfers the risk to a third-party (F5). Since this means a rebuttal can be full or partial, the spreadsheets score a 2 on the inter-argument granularity.

In total therefore, four types of risk response exist: (1) ignore (undefeated attacker argument), (2) eliminate (defeated attacker argument), (3) mitigate (partially defeated attacker argument) and (4) transfer.

3.3 ArgueSecure

ArgueSecure is an umbrella terms for a pair of tools - one online and one offline - derived from the spreadsheets described in the previous section.

The *offline* version is designed to streamline the manual process of filling in the argumentation spreadsheets. Therefore, ArgueSecure-online maintains the risk assessment game philosophy, where attacker arguments make claims with regard to risks and defender arguments rebut them by describing mitigations. However, it drops the concept of "facts" and does not support linking arguments to assets [23] (E1 = 2), and it also does not differentiate between partial or full mitigation (E2 = 1). In addition to the other tools reviewed so far, it provides keyboard shortcuts, various report generation functionality (F9) and differentiates between implemented and planned countermeasures (F7). It represents the risk assessment as an indented, collapsible list, with color-coded argument statuses (see screenshot in Fig. 7).

The *online* version makes the risk assessment process a collaborative one: participants no longer have to be in the same room; they can contribute remotely and asynchronously thereby transforming the argumentation model into a living document which keeps track of risks as they are discovered and countermeasures as they are proposed and implemented (F6). ArgueSecure-online also comes with a simplified interface (see Fig. 8 for a screenshot) aimed at non-experts and similar report generation functionality as its offline counterpart (F9), as well as differentiating risk transfers from other types of mitigations (F5). However, it also introduces new functionality, namely the ability to define many-to-many relationships between risks and attacks (F3) and between attacks and defenses (F4) and re-introduces the ability to relate arguments to assets (F2). This turns the argumentation model into a graph, but in order to avoid inter-twined links, the tool represents it as a tree by duplicating nodes with multiple incoming links.

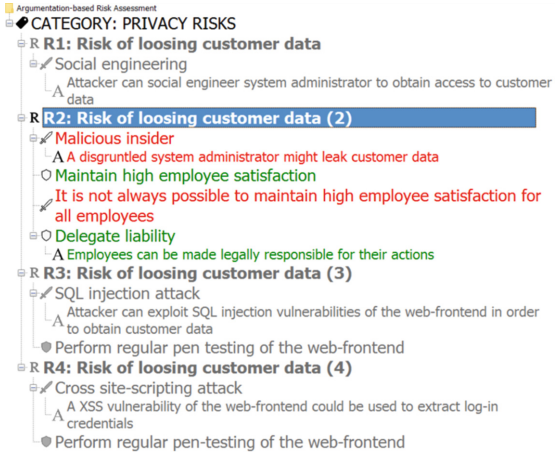

Fig. 7. Arguesecure-offline - sample risk assessment

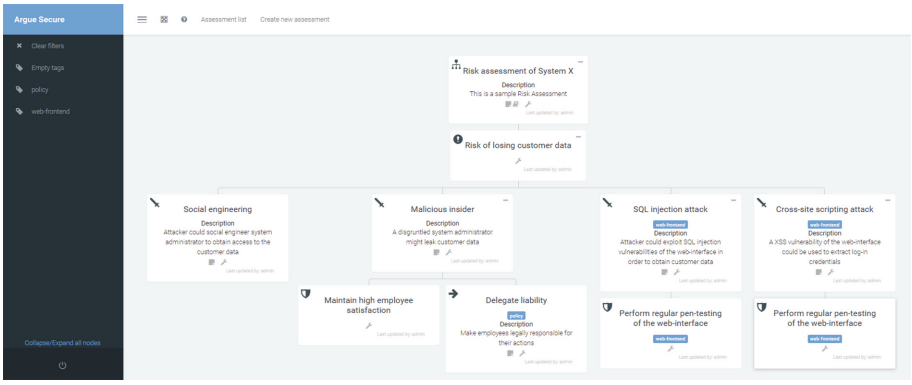

Fig. 8. Arguesecure-online - sample risk assessment

4 Comparison and Discussion

Table 1 compares the features provided by the graphical security argumentation modeling tools described in the previous section.

In OpenRISA (and its predecessor, OpenArgue) a risk, by definition violates a known security requirement. ArgueSecure does not have such a restriction, allowing more flexibility and creativity but for this reason it also cannot support linking countermeasures back to security requirements. Therefore the two have slightly different application scenarios: OpenRISA assumes Security Requirements are known and they need to be implemented, while ArgueSecure can help identify them.

OpenRISA and both of the ArgueSecure tools use a graph to encode the argumentation model. The argumentation spreadsheet could also be represented

Table 1. Feature comparison of graphical security argumentation modeling tools

	OpenArgue	Arg. sheets	AS-offline	AS-online
E1: intra-argument granularity	3	3	2	2
E2: inter-argument granularity	4	2	1	1
F1: ability to relate to security req.	Y	N	N	N
F2: ability to relate to assets	N	Y	N	Y
F3: multiple attack vectors per risk	Y	N	N	Y
F4: multiple mitigations per attack	Y	N	N	Y
F5: supports risk transfer	N	Y	Y	Y
F6: collaborative	N	N	N	Y
F7: differentiates between implemented and planned mitigations	N	N	Y	N
F8: search and filtering	N	N	N	Y
F9: export and reports	N	N	Y	Y

as a graph, with each row describing one node and its links. In addition, most generic argumentation tools (such as ASCAD) also use graphs. We therefore conclude that graphs are a suitable representation for security arguments (O1).

The ArgueSecure tools only decompose an argument into two parts: a summarized claim, together with some support for that claim (AS-offline calls them claim plus assumptions, and AS-online, title plus description). OpenArgue adds the concept of a warrant. An argumentation spreadsheets call the warrant an inference rule and adds facts, which differ from assumptions in the sense that they cannot be rebutted. However, these fields are often left empty in practice, as the inference rule or backing are mostly considered obvious [23] (e.g. a vulnerability creating a risk) and the difference between a fact and an assumption is many times only philosophical (e.g. facts can change with the specification). This leads us to our second observation: in order to describe a security argument, one needs to be able to specify at least the vulnerability or vulnerabilities involved, the risk they create and which mitigations are relevant (O2).

ArgueSecure only shows which argument attacks which other argument, but in the argumentation spreadsheets it is possible to specify which component of the argument is being attacked and in OpenRISA even how. Because OpenRISA allows to model rebuttal relationships in more detail, the resulting diagrams are more complex and inter-twined (see Fig. 5 vs. Fig. 8). The argumentation spreadsheets also model rebuttals in similar detail, but due to the tabular representation, the result is more compact and readable (see Fig. 6). ArgueSecure, which only supports binary rebuttal relationships between attacks and countermeasures and therefore manages to achieve similar scalability as the argumentation spreadsheets (see Figs. 8 and 7). Our tentative observation is that while rebuttals are necessary for relating security arguments, anything other than binary rebuttals can pose a significant scalability challenge (O3).

With regard to usability, the two ArgueSecure tools are the only ones which use icons. In addition, they also attempt to manage scalability by collapsing and expanding any part of the argumentation graph. The online version even supports filtering nodes by tags. In our toy examples scalability was not much of an issue. But realistic assessments can have hundreds of nodes, and therefore features to help navigate the argumentation graph are critical to making it human-writable and human-readable (O4).

5 Conclusions and Outlook

Perfect security is invisible, and also impossible. Security arguments can show that a system is secure to some extent by providing structured, but human-readable explanations as to which risks were considered and how they were mitigated. This is important for a variety of reasons, ranging from certification to compliance, and from awareness to assurance.

Unsurprisingly, most argumentation modeling tools employ a simplified version of Toulmin's argument structure for conceptualizing security arguments but vary in terms of either the granularity by which they decompose the argument or in the way they represent inter-argument structures. However, very few tools exist which address the specifics of security argumentation, and their audience is mostly academic.

Indeed, confronting the tools of Sect. 3 with practical security arguments shows that in order to be usable, security argumentation techniques need to be simple and reduce themselves to the essential information that needs to be present in order to argue about (in-)security of a system or software: the links between mitigations, risks and system components or modules. As these links can be of type "many-to-many", graphs are a natural fit for representing these links.

In the words of Buckingham-Shum [5], diagramming tools differ not only in the type of information they are able to represent, but especially in regard to the trade-off they make between expressiveness and usability. This is true also for argumentation graphs, which can explode in size when all known risks and relevant mitigations pertaining to a real system are added. Therefore, ensuring scalability is critical to maintaining reasonable usability. Only some of the tools available provide ways of navigating the graph, for example by searching, filtering or collapsing parts of the argumentation structure. We believe this topic has to be better investigated before security argumentation modeling becomes usable in practice. To further enhance scalability, automation and re-usability are also relevant topics not only in security argumentation, but security in general. Future work could look therefore into ways by which the argumentation graph can be filled in semi-automatically, for instance by recognizing patterns, linking to knowledge bases or parsing the output of vulnerability scanners. This might require (re-)introducting some level of formalism into the argumentation structure.

References

1. Adelard Safety Case Development (ASCAD) Manual, London, UK (2010)
2. Beel, J., Langer, S.: An exploratory analysis of mind maps. In: Proceedings of the 11th ACM Symposium on Document Engineering, pp. 81–84. ACM (2011)
3. Bloomfield, R.E., Guerra, S., Miller, A., Masera, M., Weinstock, C.B.: International working group on assurance cases (for security). IEEE Secur. Priv. **4**(3), 66–68 (2006)
4. Breaux, T.D., Baumer, D.L.: Legally "reasonable" security requirements: a 10-year FTC retrospective. Comput. Secur. **30**(4), 178–193 (2011)
5. Buckingham Shum, S.: The Roots of Computer Supported Argument Visualization, pp. 3–24. Springer, London (2003)
6. Campbell-Kelly, M.: The History of Mathematical Tables: From Sumer to Spreadsheets. Oxford University Press, Oxford (2003)
7. Cleland, G.M., Habli, I., Medhurst, J.: Evidence: Using Safety Cases in Industry and Healthcare. The Health Foundation, London (2012)
8. Cyra, L., Górski, J.: Support for argument structures review and assessment. Reliab. Eng. Syst. Saf. **96**(1), 26–37 (2011). Special Issue on Safecomp 2008
9. Defence standard 00-56 issue 4 (part 1): Safety management requirements for defence systems, July 2007
10. Emmet, L.: Using claims, arguments and evidence: a pragmatic view-and tool support in ASCE. www.adelard.com
11. Eppler, M.J.: A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. Inf. Vis. **5**(3), 202–210 (2006)
12. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Off. J. Eur. Union L119/59, 1–88, May 2016. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC
13. Firesmith, D.G.: Analyzing and specifying reusable security requirements. Technical report DTIC Document (2003)
14. Fischer, G., Lemke, A.C., McCall, R., Morch, A.I.: Making argumentation serve design. Hum.-Comput. Interact. **6**(3), 393–419 (1991)
15. Franqueira, V.N.L., Tun, T.T., Yu, Y., Wieringa, R., Nuseibeh, B.: Risk and argument: a risk-based argumentation method for practical security. In: RE, pp. 239–248. IEEE (2011)
16. Gold, J.: Data breaches and computer hacking: liability & insurance issues. American Bar Association's Government Law Committee Newsletter Fall (2011)
17. Goodwin, J., Fisher, A.: Wigmore's chart method. Inf. Logic **20**(3), 223–243 (2000)
18. Górski, J., Jarzębowicz, A., Leszczyna, R., Miler, J., Olszewski, M.: Trust case justifying trust in an it solution. Reliab. Eng. Syst. Saf. **89**(1), 33–47 (2005)
19. Haley, C., Laney, R., Moffett, J., Nuseibeh, B.: Security requirements engineering: a framework for representation and analysis. IEEE Trans. Soft. Eng. **34**(1), 133–153 (2008)
20. Haley, C.B., Laney, R., Moffett, J.D., Nuseibeh, B.: Arguing satisfaction of security requirements. In: Integrating Security and Software Engineering: Advances and Future Visions, pp. 16–43 (2006)

21. Haley, C.B., Moffett, J.D., Laney, R., Nuseibeh, B.: Arguing security: validating security requirements using structured argumentation. In: Proceedings of Third Symposium on Requirements Engineering for Information Security (SREIS 2005) held in conjunction with the 13th International Requirements Engineering Conference (RE 2005) (2005)
22. Ionita, D., Bullee, J.W., Wieringa, R.J.: Argumentation-based security requirements elicitation: the next round. In: 2014 IEEE 1st Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE), pp. 7–12. Springer, Heidelberg, August 2014
23. Ionita, D., Kegel, R., Baltuta, A., Wieringa, R.: Arguesecure: out-of-the-box security risk assessment. In: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), pp. 74–79, September 2016
24. Kelly, T., Weaver, R.: The goal structuring notation - a safety argument notation. In: Proceedings of Dependable Systems and Networks 2004 Workshop on Assurance Cases (2004)
25. Kelly, T.P.: Arguing Safety: A Systematic Approach to Managing Safety Cases. University of York, York (1999)
26. Lee, J., Lai, K.Y.: What's in design rationale? Hum.-Comput. Interact. **6**(3–4), 251–280 (1991)
27. Liao, S.H.: Expert system methodologies and applications - a decade review from 1995 to 2004. Exp. Syst, Appl. **28**(1), 93–103 (2005)
28. Maclean, A., Young, R.M., Moran, T.P.: Design rationale: the argument behind the artefact. In: Proceedings of the Computer Human Interaction conference (CHI) (1989)
29. Markham, K.M., Mintzes, J.J., Jones, M.G.: The concept map as a research and evaluation tool: further evidence of validity. J. Res. Sci. Teach. **31**(1), 91–101 (1994)
30. Mosier, K.L.: Myths of expert decision making and automated decision aids. In: Naturalistic Decision Making, pp. 319–330 (1997)
31. Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M.: Telos: representing knowledge about information systems. ACM Trans. Inf. Syst. (TOIS) **8**(4), 325–362 (1990)
32. Park, J.S., Montrose, B., Froscher, J.N.: Tools for information security assurance arguments. In: Proceedings of the DARPA Information Survivability Conference, DISCEX 2001, vol. 1, pp. 287–296 (2001)
33. Polikar, R.: Ensemble based systems in decision making. IEEE Circ. Syst. Mag. **6**(3), 21–45 (2006)
34. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument Comput. **1**, 93–124 (2010)
35. Prakken, H., Ionita, D., Wieringa, R.: Risk assessment as an argumentation game. In: Leite, J., Son, T.C., Torroni, P., van der Torre, L., Woltran, S. (eds.) CLIMA 2013. LNCS (LNAI), vol. 8143, pp. 357–373. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40624-9_22
36. Rowe, J., Levitt, K., Parsons, S., Sklar, E., Applebaum, A., Jalal, S.: Argumentation logic to assist in security administration. In: Proceedings of the 2012 New Security Paradigms Workshop, NSPW 2012, pp. 43–52. ACM, New York (2012)
37. Rushby, J.: The interpretation and evaluation of assurance cases. SRI International, Menlo Park, CA, USA (2015)
38. Shum, S.J.B., MacLean, A., Bellotti, V.M.E., Hammond, N.V.: Graphical argumentation and design cognition. Hum.-Comput. Interact. **12**(3), 267–300 (1997)
39. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requirements Eng. **10**(1), 34–44 (2005)

40. Toulmin, S., Rieke, R., Janik, A.: An Introduction to Reasoning. Macmillan, Basingstoke (1979)
41. Toulmin, S.E.: The Uses of Argument. Cambridge University Press, Cambridge (1958)
42. Yu, Y., Tun, T.T., Tedeschi, A., Franqueira, V.N.L., Nuseibeh, B.: Openargue: supporting argumentation to evolve secure software systems. In: 2011 IEEE 19th International Requirements Engineering Conference, pp. 351–352, August 2011
43. Yu, Y., Franqueira, V.N.L., Tun, T.T., Wieringa, R., Nuseibeh, B.: Automated analysis of security requirements through risk-based argumentation. J. Syst. Soft. **106**, 102–116 (2015)