ISSN 1744-1986

Technical Report N° 2005/04

Validating Security Requirements Using Structured Toulmin-Style Argumentation

Charles B. Haley
Robin C. Laney
Bashar Nuseibeh

21 March 2005

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

Validating Security Requirements Using Structured Toulmin-Style Argumentation

Charles B. Haley Robin C. Laney Bashar Nuseibeh
Department of Computing
The Open University
Walton Hall, Milton Keynes, MK7 6AA, UK
{C.B.Haley, R.C.Laney, B.Nuseibeh} [at] open.ac.uk

Abstract

This paper proposes using structured informal argumentation to assist with determining whether the security requirements for a system satisfy the security goals, and whether an eventual realized system can satisfy the security requirements. We call these arguments 'satisfaction arguments', and propose a systematic approach for their construction. A satisfaction argument is typically probabilistic and unique to the system in its context. We use the argument form proposed by Toulmin for evidence-based argumentation, consisting of claims, grounds, warrants, and rebuttals. Building on our earlier work on trust assumptions and security requirements, we show how using satisfaction arguments assists both in locating inconsistencies between security requirements and their respective goals, and in exposing tacit or inconsistent assumptions about the properties of domains and their possible effects on the eventual security of a system.

1. Introduction

Security is a *wicked problem* [7]. Wicked problems are not amenable to cookbook methods or analysis, because each is unique. There is no perfect solution to security problems – security implementations require a tradeoff between cost and effectiveness. Some assets are not worth protecting, acceptable solutions vary from stakeholder to stakeholder, and the solution space is bounded by what the customer is willing to spend and what technology can provide.

Security requirements therefore pose a particularly challenging problem to the requirements engineer. In previous work [18], we, and others [14], have argued that security requirements may be usefully described as constraints on the functions of a system. They denote operationalized *security goals*, permitting them to be considered along with other functional requirements. One begins by eliciting security goals for assets that are involved in some way with the system. Then for each function of the system, the analyst determines which assets are involved in that function. The analyst then determines the constraints on that function required in order to satisfy the goal(s).

A key validation step for such a process is the ability to show that the proposed security goals adequately express what is needed by the stakeholders, the proposed security requirements adequately satisfy the goals, and the system adequately satisfies the security requirements.

This paper proposes the use of structured informal argumentation to convince a reader that a system can respect the security goals laid upon it. We are partly motivated by evidence that suggests that argumentative approaches can provide useful solutions to wicked problems [3]. The paper proposes a structure for construction of *satisfaction arguments* for security requirements, using Toulmin-style reasoning [26, 27], in the form of claim and rebuttal. Building on our earlier work on trust assumptions [10] and security requirements [18], we show how using satisfaction arguments assists both in locating inconsistencies between security requirements and their respective goals, and in exposing tacit or inconsistent assumptions about the properties of domains and their possible effects on the eventual security of a system.

The paper is structured as follows. In Section 2 we elaborate on the motivation for our work, drawing upon related contributions from the areas of design rationale, safety cases, and domain analysis. Section 3 summarizes our approach to problem analysis and introduces Toulmin arguments upon which we base the work presented in this paper. Section 4 shows how Toulmin arguments are used to construct satisfaction arguments, when reasoning from security goals to security constraints to problem context diagrams. Section 5 presents a discussion and evaluation, including an examination of an *i** security case study published in [17]. Finally, Section 6 concludes the paper and discusses directions for future work.

2. Motivation and Background

Our work is related to, and builds upon, research on design rationale and argument capture, on safety

requirements analysis, and more generally on ideas behind problem domain analysis [21,8].

2.1 Design rationale and argument capture

Design rationale is principally concerned with capturing how one arrived at a decision, alternate decisions, or the parameters that went into making the decision [16]. For example, Buckingham Shum [4] focuses on how rationale (arguments) are visualized, especially in collaborative environments. Potts and Bruns [22], and later Burge and Brown [5], discuss capturing how decisions were made, which decisions were rejected or accepted, and the reasons behind these actions. Mylopoulos et al [19] present a way to represent formally knowledge that was captured in some way, without focusing on the outcome of any decisions. Ramesh and Dhar [23] describe a system for "capturing history in the upstream part of the life cycle." Fischer et al [9] suggest that the explicit process of argumentation can itself feed into and benefit design. Finkelstein and Fuks [8] suggest that the development of specifications by multiple stakeholders, who hold disparate views, may be achieved through an explicit dialogue that captures speech acts, such as assertions, questions, denials, challenges, etc. The representation of the dialogue is then a rationale for the specifications constructed. The common element in all of the above work is the capture over time of the thoughts and reasons behind decisions. Whether the decisions satisfy the needs is not the primary question.

When analyzing security requirements, the ultimate goal is to *convince a reader* that the decisions made satisfy the goals and that nothing is left out that could result in the goals not being satisfied. The process used to arrive at the decisions is relevant only as it relates to completeness. Questions of process and optimality are not brought into the discussion. Of course, we are not saying that there is no use in having the history that lead to the final arguments; this history will certainly be useful if the arguments fail to convince the reader or if the situation changes.

2.2 Safety cases

Kelly [13] argues that "a safety case should communicate a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context." He goes on to show the importance of the distinction between *argument* and *evidence*. An argument calls upon appropriate evidence to *convince* a reader that the argument holds. Attwood and Kelly use the same principles in [2], where they take the position that *argument* forms a bridge between requirements and specification, permitting capture of sufficient information to realize *rich traceability*.

A similar situation exists with regard to security requirements. Combining the two ideas, argument for safety cases and using arguments for traceability, we paraphrase Kelly's quote presented above as: "a *security satisfaction argument* should communicate a clear, comprehensive and defensible argument that a system is *secure enough* to operate in its context."

The techniques proposed by Kelly are not usable for security without modification, primarily because the techniques are focused around objective evidence, component failure, and accident, rather than subjective reasoning and malicious intent.

2.3 Problem domain analysis

Zave and Jackson in [28], and Jackson in [12], argue that one should construct a *correctness argument* for a system, where the argument is based on known and desired properties of the *domains* involved in the *problem*. To quote Jackson, "Your [correctness] argument must convince yourself and your customer that your proposed machine will ensure that the requirement is satisfied in the problem domain." This position is similar to Kelly's. However, Kelly extends the argument to cover not only the *machine* (in the Jackson sense), but all the behavior of all the components in the system.

A similar situation exists with regard to security requirements. Two significant distinctions must be made, however. The first is that it is very difficult to talk about *correctness* when discussing security. One can *convince* the reader that the proposed system meets the needs, but it is far more difficult to *prove* that the system is correct. The distinction between *convince* and *prove* (or *show*) is important. It is not possible to prove the negative – that violation of security goals do not exist – but one can be convincing that all reasonable outcomes have been addressed. We propose using argumentation to show that in reasonable cases, security goals are satisfied.

3. Argumentation Driven Problem Analysis

We use an approximation of Jackson's problem frames diagrams [12] to represent the system context for a given system function. We do not attempt to identify a particular problem class, but instead enter phenomena and requirements into a system context diagram.

We take this approach because we are not attempting to analyze the wider development problem. We are instead looking at the interaction between domains from a security perspective, which requires us to determine which domains can trigger or see which phenomena.

Figure 1 presents an example diagram, showing the domains involved in a simple room alarm system. As noted above, the diagram is not intended to conform to a known problem class, but it does show the requirement (the function), the constrained domain(s), the inputs, and the phenomena shared between the domains: the domains that are involved in the system within which the machine operates to realize the necessary function.

Our earlier work extended the problem frames approach with *trust assumptions* [10], which are claims about the behavior or the membership of domains included in the system, where the claims are made in

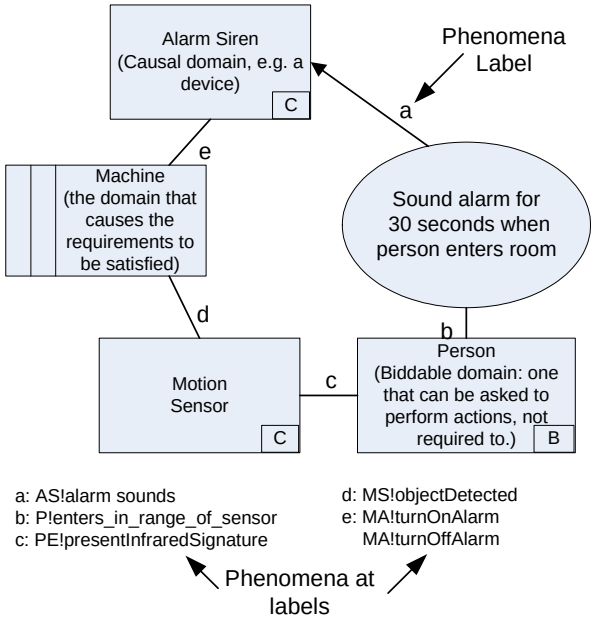


Figure 1 – Example Problem Diagram

order to satisfy a security requirement. These claims represent an analyst's *trust* that the domains will be as described. Trust assumptions should be substantiated by evidence or experience, but in the end they are the analyst's opinion and therefore *assumed* to be true. Said another way, the analyst *trusts* the *assumption* to be true in order to satisfy some security requirement.

Our work revealed the need to adopt a more structured approach to security satisfaction arguments, and this paper explores one such approach. The approaches is based on the work of Toulmin [25], one of the earliest advocates and developers of a formal structure for human reasoning. Toulmin style arguments appeared to be well suited for our purpose, since they allow us to capture where trust assumptions are needed, and where the analyst's reasoning about security is weak.

Toulmin et al [27] describe arguments as consisting of:

1. *Claims*, specifying the end point of the argument, or what one wishes to convince the world of.
2. *Grounds*, providing any underlying support for the argument, such as evidence, facts, common knowledge, etc.. Grounds are expected to be accepted; if they are not then they are themselves *claims* that must be argued separately before being used.
3. *Warrants*, connecting and establishing relevancy between the grounds and the claims. A warrant explains how the grounds are related to the claim, not the validity of the grounds themselves.
4. *Backing*, establishing that the warrants are themselves trustworthy. These are, in effect, grounds for believing the warrants.
5. *Modal qualifiers*, establishing within the context of the argument the reliability or strength of the connections between warrants, grounds, and claims.
6. *Rebuttals*, describing what might invalidate any of the grounds, warrants, or backing, thus invalidating the support for the claim.

Toulmin et al summarize the above as follows [27; pg 27]: "The *claims* involved in real-life arguments are, accordingly, *well founded* only if sufficient *grounds* of an appropriate and relevant kind can be offered in their support. These grounds must be connected to the claims by reliable, applicable, *warrants*, which are capable in turn of being justified by appeal to sufficient *backing* of the relevant kind. And the entire structure of argument put together out of these elements must be capable of being recognized as having this or that kind and degree of certainty or probability as being dependent for its reliability on the absence of certain particular extraordinary, exceptional, or otherwise *rebutting* circumstances."

As noted above, *grounds* can be *claims* in some other argument. The same can be said for *warrants*, although it is more usual to use *backing* to justify a warrant.

Toulmin et al propose a diagram for arguments, shown in Figure 2. Figure 3 presents a worked out example from [27; pg 87, Figure 7-1]: "The weather will be clearing and cooler by tomorrow morning." This sentence is a claim, and is argued as shown in the figure.

The notion of 'claim' is central to the work described in this paper. To ground the idea in Jackson's problem analysis, system requirements are optative statements (statements about what we wish to be true) and therefore claims that should be argued (and in fact Jackson refers to such arguments as *correctness arguments*). For example, the optative statement "the system shall do X" states a claim that given the conditions described in the problem, the system *will* do X. This claim can be disputed and argued.

Indicative statements (statements that are "objectively true" [12]) are both grounds used to justify the claims made by optative statements, and claims in their own right that a reader is asked to accept, with or without justification. This paper proposes that indicative statements must be argued as carefully as optative ones.

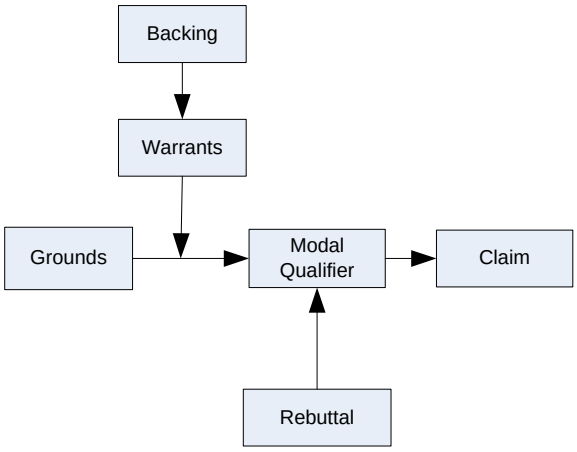


Figure 2 – Generic Toulmin-form argument

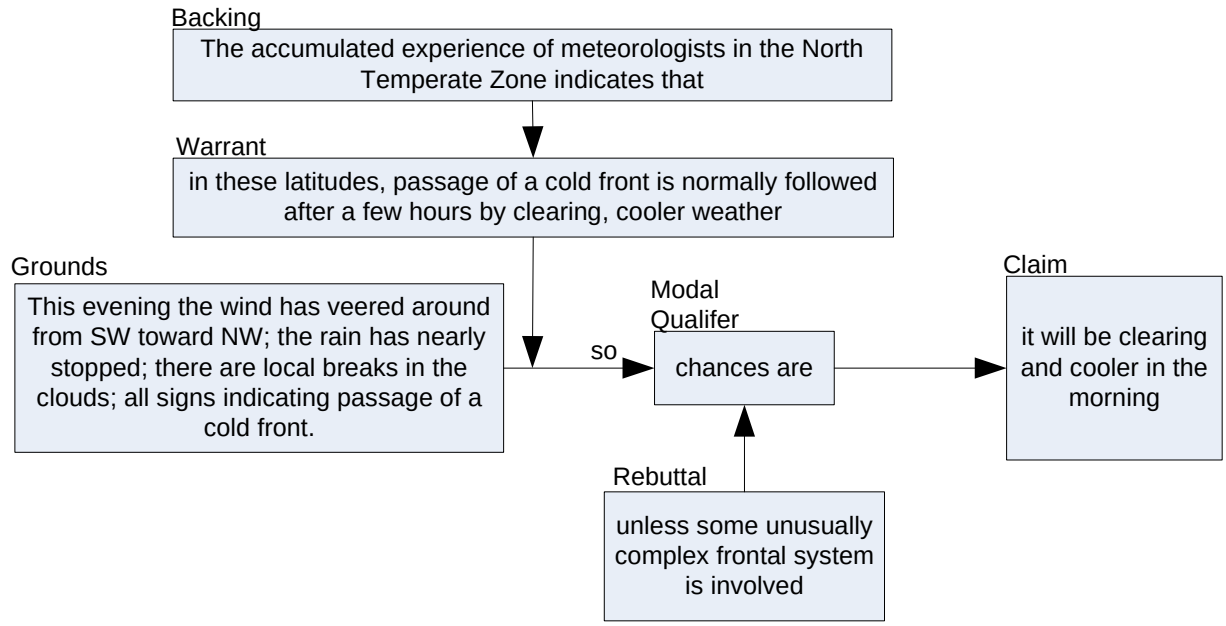


Figure 3 – Example argument for 'weather'

A final note: in this paper, we usually omit the *backing* in the interests of space, when it is obvious.

4. From Goals to Constraints to Domains

Recall the goal of our work: to construct convincing *satisfaction arguments* that a system *can* meet its security goals. The reader may note the use of the word *can*, instead of the word *will*. We use the phrase *can meet …* instead of *will meet …* because we are working in problem space, not solution space. We do not know if the eventual design will respect the specifications and constraints levied upon it, nor if it will introduce unintended vulnerabilities.

To construct security satisfaction arguments, one must have security requirements to be satisfied. We noted in Section 1 that there are two principal steps involved in determining the security requirements for a system: enumerating the security goals (which assets are to be protected, and why), then determining the constraints to apply to the system functions to satisfy the goals. When following these steps, the analyst is implicitly making several *claims*. One set of claims relates to the derivation of the constraints from the security goals: that a function that correctly realizes the constraints will satisfy the goals from which they were derived. The second set of claims relate to the function itself: that the collection of domains (including the machine as it will be) can respect the constraints. The analyst must be able to construct arguments that justify these claims.

This section uses a running example to illustrate construction of security satisfaction arguments.

4.1. Running Example

A simple two-function human resources application will be used in this section to illustrate our uses of argumentation. The first function is retrieval of data, and the second is backup. There are two security goals:

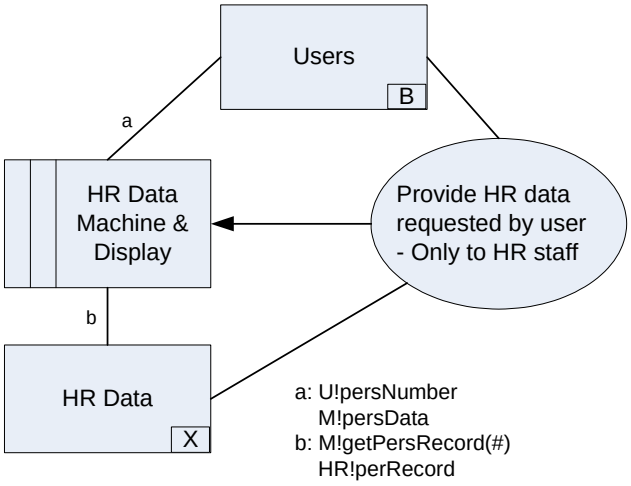


Figure 4 – HR data retrieval problem

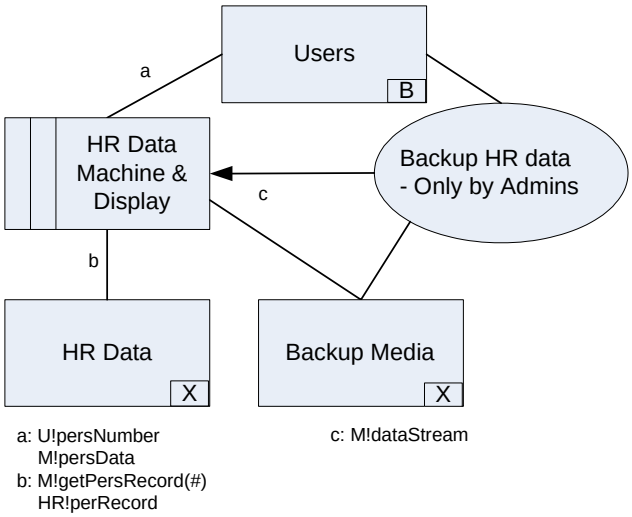


Figure 5 – Example backup problem

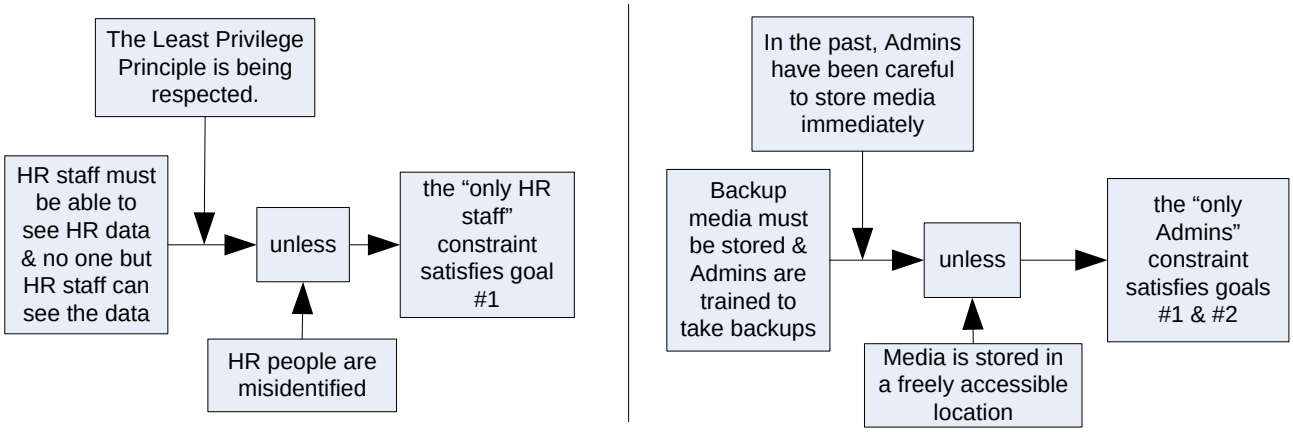


Figure 6 – Arguments that constraints satisfy goals

1) the data is confidential; and 2) data must not be unavailable for more than 24 hours. Figure 4 shows the first problem, and Figure 5 the second. The functional constraints that have been derived from these goals, backed by the arguments in Figure 6, are:

- The data must only be provided to HR staff, (see Figure 4): this satisfies Goal 1.
- The data must only be backed up by administrators (see Figure 5): this satisfies Goal 2

4.2. Goals Satisfied by Constraint(s)

In this first case, the claim being made is that the security goals are satisfied by the constraints. This is a cross-function claim; it is not sufficient to show that the constraints on individual functions each independently satisfy the associated goals; one must also show that there are no inconsistencies across functions.

We see the following claims in our example: 1) In Figure 4, the constraint "Only to HR staff" is sufficient to maintain confidentiality, and in Figure 5, the constraint "Only by Admins" is also sufficient. See Figure 6 for plausible arguments.

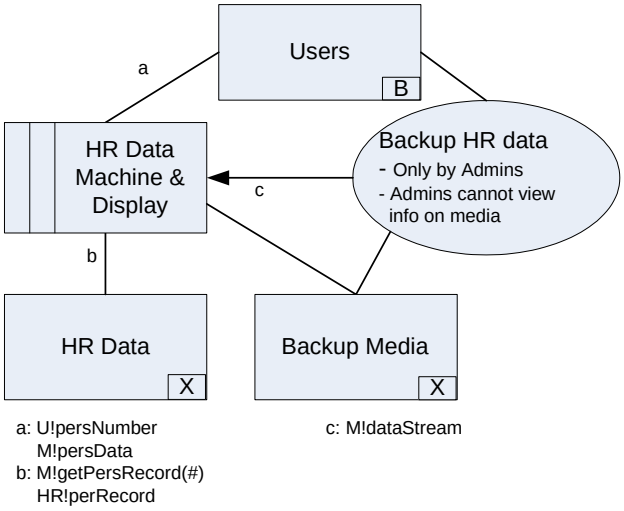


Figure 7 – Corrected 'backup' problem diagram

The rebuttal to the "only HR staff" argument should generate concern. Clearly one does not want this situation to occur. Some assistance will come later, when the constraints themselves are examined. Additional assurance that this state will not occur requires discussion of policies about such topics as delegation, transfer, and termination, all of which are beyond the scope of this paper.

We now must argue whether or not the constraints are *consistent*. A set of constraints is consistent if they have the equivalent effects; one does not permit what another forbids. In our example, it is readily apparent that the constraints are not consistent. The constraint in Figure 4 restricts visibility to HR staff, where the constraint in Figure 5 permits administrators to see the data. Either 'adminstrators' must be added to Figure 4 or administrators must be forbidden from seeing the data on the media in Figure 5. We choose the second, adding another constraint to Figure 5: *Admins cannot view info on media*. This corrected diagram is shown in Figure 7. We must now revisit the argument that claimed that the constraints could be met. To save space, we do not do so here, as the argument is unchanged.

It is not always so obvious that constraints are not consistent. For example, confidentiality constraints could easily clash with legally mandated access to information. Availability constraints could clash with integrity constraints; e.g., the need to take internally consistent backups of a running system. Determining inconsistencies requires careful thought about *who* is doing *what function* to *what information*, and *why*.

A full treatment of techniques to handle inconsistency is outside the scope of this paper [21, 25], however, in some cases constraints must be weakened. In our example, we chose to maintain the constraint in Figure 4, while adding a new constraint to Figure 5. We could also have chosen to weaken the constraint in Figure 4 to include 'HR staff and administrators', in which case we would need to consider whether we must also weaken the constraint in Figure 5. In any event, we would be required to re-argue the case that the constraints satisfy the security goals.

4.3. Problem Respects Constraints

We now move on to the second phase, verifying that the constraints can be respected by the problem as described. To accomplish this, we construct a *satisfaction argument*. The claim being made is as stated above: the constraints can be respected.

The satisfaction argument is constructed by using a two-layer argument. The first layer examines each domain in the problem, arguing that the domain respects the constraints. The second layer is a composition of the resulting arguments; arguing that because all domains respect the constraints, the problem as a whole does so as well.

We begin with Figure 4: the HR problem. There are three domains in the problem: the biddable domain 'users', the lexical domain 'HR data', and the machine. Beginning with the 'users', we attempt to construct the argument. The results are shown in Figure 8.

The attempt to construct the argument exposes a major problem. We have no way, given what we see in the problem diagram, to know if a 'user' is a member of HR staff, so we cannot construct a plausible argument. At this point, we have (at least) three choices:

1. Introduce some physical restriction, such as a guard, to ensure that the membership of the domain 'users' is restricted to the set of appropriate people.
2. Introduce phenomena into the system permitting authentication and authorization of a 'user'.
3. Introduce a *trust assumption* (TA) stating that we assert that the membership of 'users' is limited to HR staff, even though no information is available to support the assertion.

We choose option 2. The resulting problem diagram is shown in Figure 10. We now require the user to supply some sort of credentials along with the request for information. These credentials are passed to some external authentication and authorization engine, which answers *yes* or *no*. If the answer is *yes*, then the machine responds to the user with the data; otherwise, the data is refused.

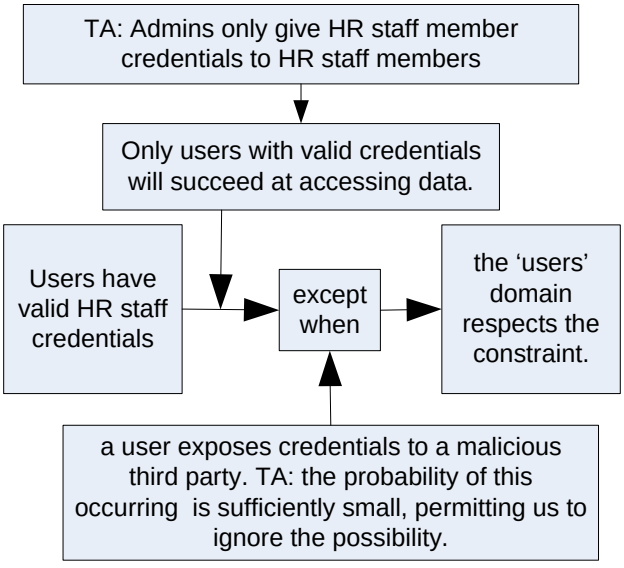


Figure 9 – Revised 'users' domain argument

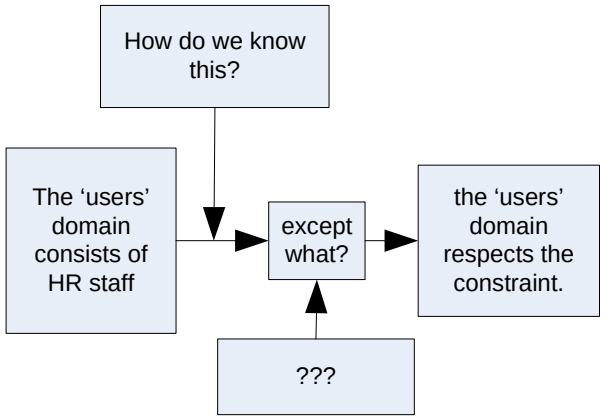
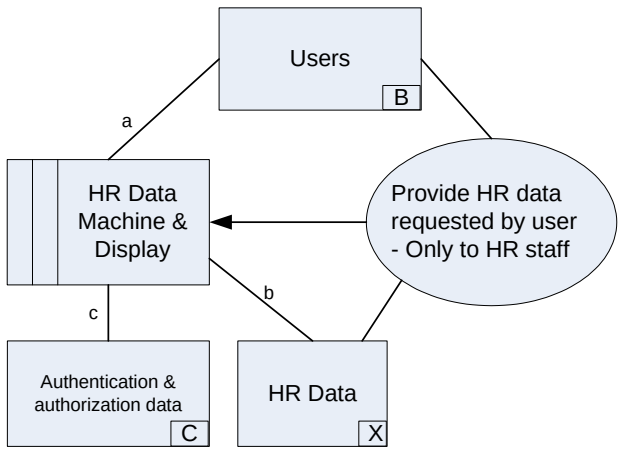


Figure 8 – Initial argument for 'users' domain

We must now construct the satisfaction argument for the new 'users' domain. To jump ahead, the argument will fail, because we are unable to assure ourselves with the information we have in hand that someone who has HR staff credentials is actually a member of HR staff. We solve this problem by introducing two trust assumptions into the argument. Figure 9 shows the resulting argument. If the trust assumptions shown in Figure 9 are deemed too risky, then additional analysis must be performed to verify how credentials are administered, and why social engineering attacks will not work in this case.

We must now construct similar arguments for each of the three remaining domains. Rather than discuss each domain individually, Figure 11 presents the resulting three arguments.

The final step in the process is constructing the argument for the problem as a whole. The claim being argued is 'The described problem respects the constraints'. The grounds are the claims of the individual arguments. The warrant must show that these claims are internally consistent, and any rebuttal must be acceptable. In this example, the argument is



a: U!persNumber(#,credentials)
M!accessDenied
M!persData
b: M!getPersRecord(#)
HR!perRecord
c: M!isAuthorized?(credentials)
AA!isOK(Boolean)

Figure 10 – New HR staff problem diagram

straightforward, and, in the interest of brevity, is not shown in this paper.

Once we have successfully argued that all of the domains in the problem respect the constraints, it remains to verify that any *rebuttal* in the original 'constraints satisfy goals' argument (Figure 6, left side) has been accounted for. There was one such rebuttal, "HR people are misidentified". This rebuttal is clearly covered by the arguments for each domain. If it were not so clear, then a last argument would need to be constructed around the claim 'HR people are *not* misidentified'. This claim would then be argued like the others, including the possible addition of trust assumptions.

Again, for brevity, we do not present diagrams of the arguments for the backup problem. The arguments show that in order to respect the confidentiality constraint, the information on the media should be encrypted to prevent Admins from viewing it while stored on the media.

5. Discussion and Evaluation

We now examine some of the issues arising from our work, which we discuss in order to evaluate our approach.

5.1. Finding Rebuttals, Grounds, and Warrants

One question that arises is "how does the analyst find rebuttals, grounds, and warrants?" Unfortunately, and because we are dealing with a wicked problem, we cannot propose a recipe. It is worthwhile to recall, however, that grounds are indicative statements in the context of the argument, while claims are optative statements in the same context.

The method we propose is for the analyst first to choose which claim is being argued, and then gather the grounds that are pertinent to the claim. The analyst then asks him- or herself the question "what can prevent this claim from being true?", making a list of the results. This raw list is the initial set of rebuttals. Experience suggests that many, if not most, of the rebuttals will disappear through the application of warrants to the grounds, but some will not. In some cases the warrants will not be strong enough, and trust assumptions will be added to fill in the gaps. In other cases, a trust assumption will be inserted into the rebuttal stating that the event(s) described in the rebuttal are not to be considered. A third alternative is to increase the scope of the analysis to better determine the system's behavior.

5.2. Problem vs Solution Space

A reasonable objection to argumentation as described in this paper is that we are designing the system in order to determine its requirements. To some extent, this is

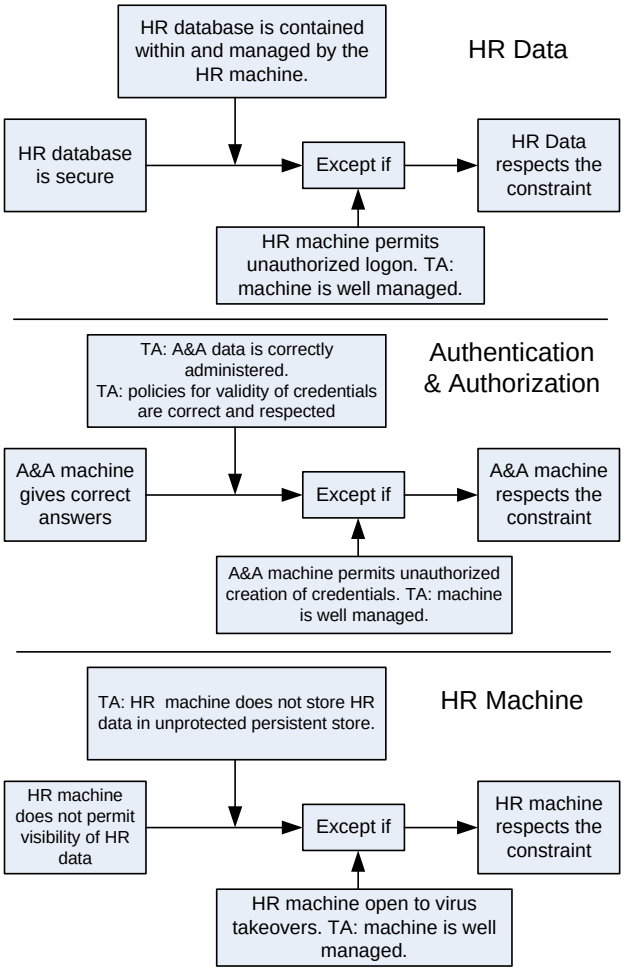


Figure 11 - Remaining Arguments

true; the *domains* included in the system are iteratively being more finely described.

However, we argue that the part of the system being constructed is the *machine*, and we are not designing that. By applying an iterative process that interleaves requirements and design [20], we are specifying the *environment* (or *context*) that the machine lives within. These specifications include additional domains that need to exist (perhaps inside the machine), and additional phenomena required to make use of the additional domains.

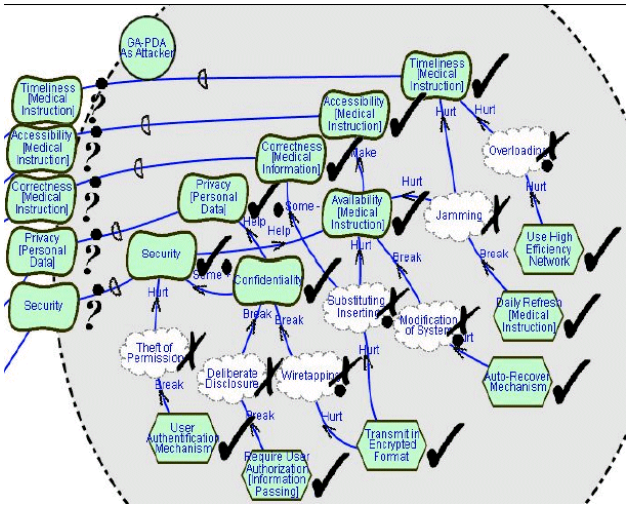


Figure 12 – Diagram from [17], unmodified

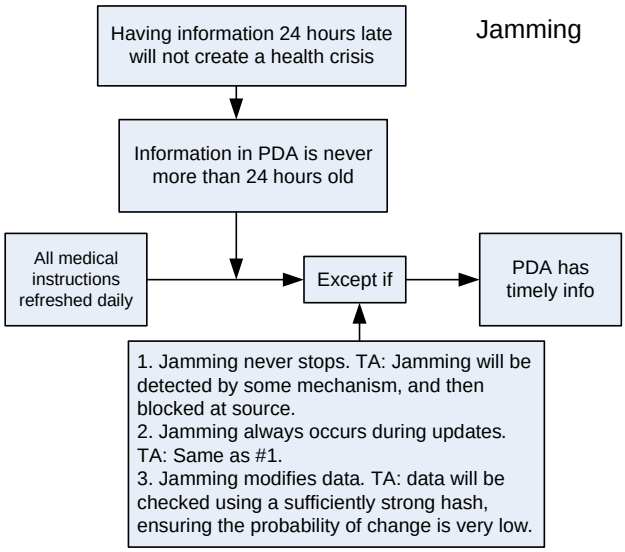


Figure 13 – 'Jamming' with trust assumptions

5.3. Applicability to other RE Frameworks

Another issue is that our security arguments are closely tied to Jackson problem analysis. Our position is that the principle of arguing security applies to any framework or method that makes identifiable claims. To demonstrate, we look at security claims made in a recent paper using *i** [17]. In it Liu et al identify threats such as "theft of permission", "deliberate disclosure", "wiretapping", and "jamming" (amongst others), along with the countermeasures (respectively) "user authentication mechanism", "require user authorization", "use encryption", and "use high efficiency network". Figure 12 (copied without modification from [17]) shows these relationships.

In effect, Liu et al are making the claim that the countermeasures negate the threats. Each countermeasure is a separate claim. Applying our argument structure, we discover that there are no grounds supplied that support the claims, other than the intuitive ones implied by the name. It is easy to generate 'rebuttals', such as "the administrator could sell authentication information", "authorization can be forged", etc.

Figure 14 presents arguments for two countermeasures, "Jamming" and "Auto-Recovery Mechanism," where the arguments make plausible assumptions about what was behind what Liu et al wrote in [17]. We see from these arguments that there are several significant and important rebuttals. In addition, the warrants are not clear, and should therefore have backing (which was not supplied in [17], probably for space reasons).

In the end, one would almost certainly add trust assumptions to negate some of the rebuttals; one possibility would be 'jamming will be detected by as-of-yet undersigned mechanisms, so continuous and periodic jamming are not a major a concern'. Figure 13 presents just such an argument.

5.4. Goal Hierarchies and Arguments

It is reasonable to ask, "What is the difference between a goal hierarchy and the argumentation proposed in this paper?" After all, if a parent goal can be considered a to be claim, then child goals (sub-goals) support that claim.

We postulate that one should 'argue' that the child goals do indeed support the parent. Taking the position that the child goals are 'grounds', then the warrants that connect and justify the grounds to the claim should be provided. Doing so may help show correctness. Adding the rebuttals may assist in showing completeness.

One must also consider terminal goals. KAOS ([15] and many others) separates such goals into two categories: *requirements*, goals to be discharged by the software to be; and *assumptions*, goals discharged by

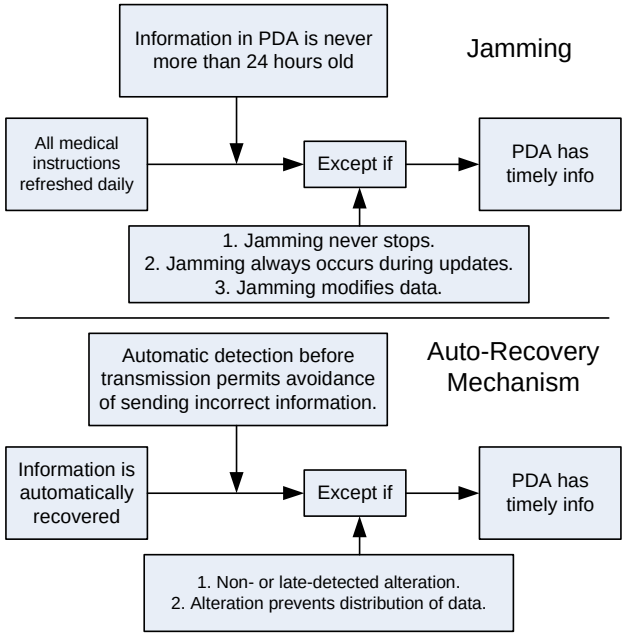


Figure 14 – Arguments for example in [17]

other actors in the environment. Requirements goals should be argued to show possibility; that circumstances can be envisaged where the software can discharge the goals in a secure fashion. Assumption goals should be argued for plausibility; that it is reasonable to make the assumption, and that any circumstances where the assumption is invalid need not be taken into consideration.

Therefore, structured argumentation can be used as another tool to verify that the goal hierarchy accurately reflects what is desired, and that the behavior implied by the goals is consistent with the security goals of the system.

6. Conclusions & Future Work

We have shown how *satisfaction arguments*, based on the structured informal argumentation proposed by Toulmin et al [27], facilitate ensuring that a system can meet its security goals. The structure behind the arguments, and in particular the insistence on the *rebuttal*, assists in finding system-level vulnerabilities. These vulnerabilities are removed either through modification of the problem or through addition of *trust assumptions* that explain why the vulnerability can be discounted. Finally, we have shown how the satisfaction arguments presented here can be used in other RE frameworks to test implicit or explicit claims.

One area that we are actively looking at is tool support for diagramming the arguments. One promising candidate is Araucaria [24]. Other options include a tool constructed around problem context diagrams. We have already begun experimenting with adapting the argument capture tool Compendium [1], for describing and navigating through IBIS-style arguments [6]

Another area of focus is to combine the ideas from this paper together with our earlier work on threat descriptions [11]. The goal is to present a coherent method to elicit security goals through the enumeration of assets and the threats that they are subject to, and then to link the resulting problem context diagrams together to aid consistency checking.

Finally, we continue to examine industrial case studies in order to test these ideas more thoroughly.

Acknowledgements:

The authors wish to thank Michael Jackson for his continuous involvement and support, and both Jonathan Moffett and Luncheng Lin for their pertinent (and sometimes pointed) criticisms and suggestions. Thanks also to Simon Buckingham Shum for a number of helpful conversations about argumentation, and to Clara Mancini for extending the Compendium tool to support Jackson's problem diagrams. The financial support of the Leverhulme Trust is gratefully acknowledged, as is the EU for supporting the E-LeGI project, number IST-002205.

References:

[1] "Compendium Institute." http://www.compendiuminstitute.org/.

[2] K. Attwood, T. Kelly, and J. McDermid, "The Use of Satisfaction Arguments for Traceability in Requirements Reuse for System Families: Position Paper," *Proceedings of the International Workshop on Requirements Reuse in System Family Engineering, Eighth International Conference on Software Reuse*. Carlos III University of Madrid, Madrid Spain, 5 Jul 2004, pp. 18-21.

[3] S. Buckingham Shum and N. Hammond, "Argumentation-Based Design Rationale: What Use at What Cost?," *International Journal of Human-Computer Studies*, vol. 40 no. 4, April 1994, pp. 603-652.

[4] S.J. Buckingham Shum, "The Roots of Computer Supported Argument Visualization," *Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making*, P. A. Kirschner, et al., Eds. London UK, Springer-Verlag, 2003, pp. 3-24.

[5] J.E. Burge and D.C. Brown, "An Integrated Approach for Software Design Checking Using Design Rationale," *Proceedings of the First International Conference on Design Computing and Cognition*, J. S. Gero, Ed. Cambridge MA USA, Kluwer Academic Press, 19-21 July 2004, pp. 557-576.

[6] J. Conklin, "Hypertext: An Introduction and Survey," *IEEE Computer*, vol. 20 no. 9, 1987, pp. 17-41.

[7] P. DeGrace and L.H. Stahl, *Wicked Problems, Righteous Solutions; A Catalogue of Modern Software Engineering Paradigms*, Prentice Hall, 1990.

[8] A. Finkelstein and H. Fuks, "Multiparty Specification," *Proceedings of the 5th International Workshop on Software Specification and Design*. Pittsburgh PA USA, 1989, pp. 185-195.

[9] G. Fischer, A.C. Lemke, R. McCall, and A. Morch, "Making Argumentation Serve Design," *Design Rationale Concepts, Techniques, and Use*, T. Moran and J. Carrol, Eds., Lawrence Erlbaum and Associates, 1996, pp. 267-293.

[10] C.B. Haley, R.C. Laney, J.D. Moffett, and B. Nuseibeh, "The Effect of Trust Assumptions on the Elaboration of Security Requirements," *Proceedings of the 12th International Requirements Engineering Conference (RE'04)*. Kyoto Japan, IEEE Computer Society Press, 6-10 Sep 2004, pp. 102-111.

[11] C.B. Haley, R.C. Laney, and B. Nuseibeh, "Deriving Security Requirements from Crosscutting Threat Descriptions," *Proceedings of the Third International Conference on Aspect-Oriented Software Development (AOSD'04)*. Lancaster UK, ACM Press, 22-26 Mar 2004, pp. 112-121.

[12] M. Jackson, *Problem Frames*, Addison Wesley, 2001.

[13] T.P. Kelly, *Arguing Safety - A Systematic Approach to Safety Case Management*, D.Phil Dissertation, University of York, York, 1999.

[14] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*, United Kingdom: John Wiley and Sons, 1998.

[15] A. van Lamsweerde, "Goal-oriented Requirements Engineering: A Guided Tour," *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering (RE'01)*. Toronto, Canada, IEEE Computer Society Press, 27-31 Aug 2001, pp. 249-263.

[16] J. Lee and K.Y. Lai, "What's in Design Rationale?," *Human-Computer Interaction - Special Issue on Design Rationale*, vol. 6 no. 3-4, 1991, pp. 251-280.

[17] L. Liu, E. Yu, and J. Mylopoulos, "Security and Privacy Requirements Analysis Within a Social Setting," *Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03)*. Monteray Bay, CA USA, 8-12 Sept 2003.

[18] J.D. Moffett, C.B. Haley, and B. Nuseibeh, *Core Security Requirements Artefacts*, Technical Report 2004/23, Department of Computing, The Open University, Milton Keynes UK, June 2004.

[19] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos: Representing Knowledge About Information Systems," *ACM Transactions on Information Systems (TOIS)*, vol. 8 no. 4, October 1990, pp. 325 - 362.

[20] B. Nuseibeh, "Weaving Together Requirements and Architectures," *Computer (IEEE)*, vol. 34 no. 3, Mar 2001, pp. 115-117.

[21] B. Nuseibeh, S. Easterbrook, and A. Russo, "Leveraging Inconsistency in Software Development," *IEEE Computer*, vol. 33 no. 4, April 2000, pp. 24-29.

[22] C. Potts and G. Bruns, "Recording the Reasons for Design Decisions," *Proceedings of the 10th International Conference on Software Engineering (ICSE'88)*. Singapore, IEEE Computer Society, 1988, pp. 418-427.

[23] B. Ramesh and V. Dhar, "Supporting Systems Development by Capturing Deliberations During Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 18 no. 6, June 1992, pp. 498-510.

[24] C. Reed, "Araucaria." 2005, http://www.computing.dundee.ac.uk/staff/creed/research/araucaria.html.

[25] W.N. Robinson, S.D. Pawlowski, and V. Volkov, "Requirements Interaction Management," *Computing Surveys (ACM)*, vol. 35 no. 2, June 2003, pp. 132-190.

[26] S.E. Toulmin, *The Uses of Argument*, Cambridge: Cambridge University Press, 1958.

[27] S.E. Toulmin, R.D. Rieke, and A. Janik, *An Introduction to Reasoning*, New York: Macmillan, 1979.

[28] P. Zave and M. Jackson, "Four Dark Corners of Requirements Engineering," *Transactions on Software Engineering and Methodology (ACM)*, vol. 6 no. 1, Jan 1997, pp. 1-30.