OpenArgue: Supporting Argumentation to Evolve Secure Software Systems

Yijun Yu*, Thein Than Tun*, Alessandra Tedeschi†, Virginia N. L. Franqueira‡ and Bashar Nuseibeh*§
* The Open University, Milton Keynes, UK
† DeepBlue, Rome, Italy
‡ University of Twente, Enschede, The Netherlands
§ Lero, Irish Software Engineering Research Centre Limerick, Ireland

***Abstract*—When software systems are verified against security requirements, formal and informal arguments provide a structure for organizing the software artifacts. Our recent work on the evolution of security-critical software systems demonstrates that our argumentation technique is useful in limiting the scope of change and in identifying changes to security properties. In support of this work, we have developed OpenArgue, a tool for syntax checking, visualizing, formalizing, and reasoning about incremental arguments. OpenArgue has been integrated with requirements engineering tools for Problem Frames and i*, and applied to an Air Traffic Management (ATM) case study.**

***Index Terms*—Security Requirements; Argumentation; Problem Frames; i*; ATM**

I. Introduction

As long-lived software systems evolve, checking whether their properties satisfy evolving security requirements needs to be done continuously. Haley et al. [1] introduced the use of argumentation for the validation of security requirements, which we have extended in two ways: first, we have proposed an approach for deriving the changes in security properties of an evolving software system using argumentation [2], and second, we have incorporated risk assessment into the argumentation process in order to focus on practical security [3]. In support of these, we have developed an Eclipse-based automated tool, OpenArgue to support argumentation, and then applied it to a significant Air Traffic Management (ATM) case study. The OpenArgue tool supports informal argumentation by checking and visualizing its structures. The tool also supports formalization and reasoning of incremental arguments described in propositional logic. Integrated with existing plugins we developed for requirements engineering approaches, including Problem Frames, and i*, the tool can show traceability between them through model transforming using a security ontology [2]. In collaboration with DeepBlue, the tool is applied to a significant ATM case study.

II. OpenArgue: An Argumentation Tool

The structure of informal arguments in our meta-model, shown in Fig. 1, is similar to the structure of Toulmin-style arguments. Informal arguments are formalized typically in propositional logic. The main concepts supported by this conceptual model include the following:

- An argument has one claim, zero or more ground(s), and zero or more warrant(s);

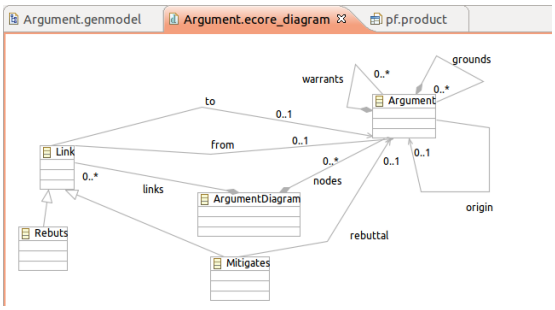

Fig. 1. A simplified meta-model of arguments

- A *claim* is a predicate whose truth is established by an argument;
- A *ground* is a piece of evidence, a fact, a theory, a phenomenon considered to be true;
- A *warrant* is either a fact or a sub-argument that shows how facts justify the claim;
- A *rebuttal* uses one argument to rebut another, falsifying the claim of the rebutted argument;
- A *mitigation* uses one argument to mitigate a rebutted argument, restoring the claim of the rebutted argument;
- An *argument diagram* has zero to many nested incremental arguments, rebuttals and mitigations;
- A *round of argumentation* indicates a temporal ordering among the increments in the argumentation.

A. Providing syntax highlighting editors

The argument syntax is defined using the extended BNF. Illustrated with a fragment of the ATM example from [2], Fig. 2 shows the argument editors providing syntax highlighting and input validation, in textual and graphical forms.

B. Visualizing and synchronous editing of models

Arguments described in plain text are used to generate argument diagrams, and the edited argument diagrams are synchronized back into the textual input format, through model synchronization between the EMF and GMF editors.

C. Formalizing arguments using propositional logic

When formalizing the arguments, the basic structure of an argument is transformed into the following formula:

$$Ground \wedge Warrant \rightarrow Claim \quad (1)$$

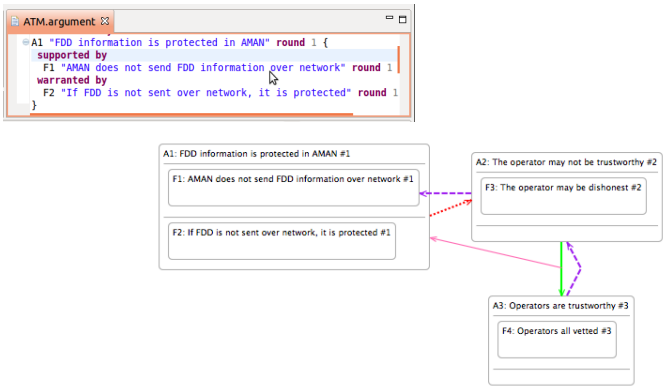

Fig. 2. Syntax highlighting and graphical editors in OpenArgue

OpenArgue automatically extracts the identifiers of the claims as propositional literals and constructs a propositional formula in the conjunctive normal form. Claims, grounds and warrants may contain complex propositional formulae annotated on the informal arguments as user-defined parts of the syntax (the tool checks the validity of the Boolean expression as well). OpenArgue then generates a syntactically correct propositional statement for an entire diagram, including both the implicit rules (1) and the user-defined ones.

D. Reasoning about rebuttals and mitigations

OpenArgue is integrated with the decreasoner, an off-the-shelf reasoning tool that translates propositional formulae into problems for SAT-solvers. The integrated tool supports logical deduction to check whether an argument is valid, and model finding to obtain counterexamples to the argument. On the basis of these results, rebuttals and mitigations are generated and visualized. Our algorithm traverses the entire structure of arguments such that all possible rebuttals and mitigations between adjacent rounds are checked, ensuring that the rebuttals and mitigations are effective: a rebuttal does negate the original claim, and a mitigation does restore the negated claim [2].

E. Integrating with other RE tools

OpenArgue is tightly integrated with open-source RE tools for the Problem Frames (OpenPF) and *i** (OpenOME), and the model transformation engine Viatra2. It allows elements of the requirements model to be hyperlinked with arguments, and performs model evolution through change patterns [2].

F. Deploying and lowering the adoption barriers

The tool is available to be downloaded as an Eclipse rich client platform from http://sead1.open.ac.uk/pf. In addition, we offer a Web service as a modeling wiki (Miki) [4] for users to benefit from its full modeling features without large network downloads, http://computing-research.open.ac.uk/trac/openre.

III. Case Study

We have applied OpenArgue to analyse the impact on the security of the Arrival Management (AMAN) system [2], a software component supporting the Air Traffic Controllers in the approach phase, when a new IP-based communication network is introduced. The AMAN exchanges with other ATM actors and processes, and presents to Air Traffic Controllers sensitive data about the flight that have to be protected.

First, Problem Frames diagrams are created showing the context of change, and the security properties that need to be maintained after the change. After describing the behaviors and properties of the system, several rounds of argumentation are carried out assessing the satisfaction of the security requirements after the change. Initial arguments are created from the perspective of the "defender" of the system, whilst rebuttals are created from the perspective of "attackers" and mitigations are created from the perspective of defenders responding to the attackers' actions. The argumentation at every increment is formally checked using propositional logic. When there is a rebuttal to which no mitigation can be found, the tool exposes a vulnerability. When the properties of the mitigations cannot be mapped to the properties of the existing domains, this indicates that some changes in security properties need to be implemented in order to make the system secure.

IV. Some Related Work

Argumentation has been extensively applied to build safety cases [5], to demonstrate compliance to laws and regulations [6], and to define trusted bases of dependable software systems [7]. Unlike these, our tool support is primarily aimed at reasoning about the satisfaction of security requirements of evolving systems.

V. Conclusions

The initial feedback we received from DeepBlue confirms that: (i) argumentation is intuitive to ATM experts; (ii) arguments are useful when designing and structuring software artifacts; (iii) the use of informal and formal arguments is helpful to domain experts; and (iv) the integrated tool support for argumentation facilitates the adoption of the approach.

Acknowledgment

Financial support of the SecureChange project, SFI grant 03/CE2/I303_1, and the Sentinels program are gratefully acknowledged.

References

[1] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Trans. Softw. Eng.*, vol. 34, pp. 133–153, January 2008.
[2] S. Consortium, "D.3.2 methodology for evolutionary requirements," SecureChange Project, Tech. Rep., 2011.
[3] V. N. L. Franqueira, T. T. Tun, Y. Yu, R. Wieringa, and B. Nuseibeh, "Risk and argument: A risk-based argumentation method for practical security," in *RE*, 2011.
[4] Y. Yu, M. Petre, and T. T. Tun, "Miki: a synchronous modeling wiki for software requirements," in *FlexiTools*, 2011.
[5] T. P. Kelly, "Arguing safety – A systematic approach to safety case management," Ph.D. dissertation, University of York, 1998.
[6] B. Burgemeestre, J. Hulstijn, and Y.-H. Tan, "Value-based argumentation for justifying compliance," in *DEON'10*, 2010, pp. 214–228.
[7] E. Kang and D. Jackson, "Dependability arguments with trusted bases," in *RE*, 2010, pp. 262–271.