Poster: ArgSEC, A Security Administrator's Assistant

Andy Applebaum[1], Karl Levitt[2] and Jeff Rowe[2]
[1]: Student [2]: Faculty
Department of Computer Science
University of California Davis, CA 95616
applebau@ucdavis.edu

I. Introduction

There is nothing more simultaneously vague, ignored, and yet useful than warnings. They serve as abstracted middle-grounds where things are not running quite smoothly nor are they quite broken. They are messages signaling "Be careful! There might be a problem!", necessitating further investigation by the recipient. They are yellow lights, responded to differently by every individual.

But what do we do when we encounter warnings, and how do we handle them from a computer security perspective? What is "user account control"? What is an "untrusted connection" and why does it have risks? How dangerous is a tracking cookie? What does it mean that Java is out of date? How does "Heartbleed" effect me?

Questions like these motivate our goal to better understand this middle-ground "warning" area in the context of secure network administration. In particular, we describe a tool, ArgSEC, to serve as a "security administrator's assistant," tackling questions that include:

- How do I respond to sensor reports about the network?
- Is the network configured properly and securely?
- What are the consequences of modifying my system?

These questions only scratch the surface of the true complexity and difficulty of being a systems administrator. Day-to-day operation can vary widely, and each decision that the administrator makes can have catastrophic results if made in error. Resources (personnel, hardware, etc.) available to the administrator may be furthermore sparse, adding to the stress of operation. Because of this, and because of the high decision consequence and the overall nature of security itself, we believe that a security administrator's assistant would be invaluable to administrators, filling a need while advancing the public good.

II. Background: Argumentation

The concept of argumentation traces its roots back into the early days of philosophy and logic, eventually branching into the field of rhetoric as a tool for modeling debates. Computationally, argumentation has been appropriated within the artificial intelligence community as a logical system for reasoning that performs well under uncertainty and with conflicting information, all the while being accessible to human operators because of its rhetorical roots. Phan Minh Dung's seminal work [1] introduced the notion of "argumentation frameworks," built around a simple logical structure only comprised of atomic arguments and a binary attacks relation over the arguments, proving an equivalence between these frameworks and general non-monotonic logic. Recent developments have extended this original framework, introducing richer arguments, additional relationships, and alternative truth semantics.

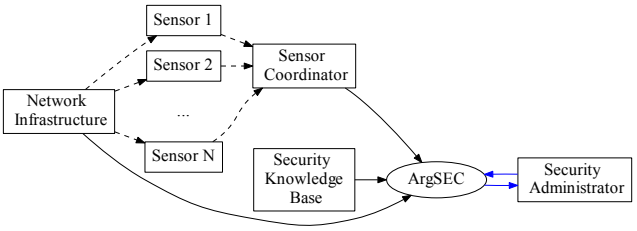

Fig. 1: Design overview of ArgSEC.

Argumentation's primary strength over other logics lies in the fact that it can draw conclusions in the face of contradictions. This quality make it ideal for security, where knowledge, consequence and risk can vary significantly in context. As an example, consider the case of a network under siege by a denial-of-service reflection attack over DNS. The administrator could respond by taking the entire network offline, contacting the DNS server, blocking port 53, blocking the specific DNS server, etc. For each of these responses, argumentation can be exploited to illustrate the arguments both *for* and *against* each specific action: blocking port 53 would stop the attack, but at the cost of DNS and therefore web browsing; blocking the server is still susceptible if the attacker switches servers; taking the network down might cause the attacker to lose interest while still allowing offline activity, but at the cost of online productivity. Argumentation provides a framework that enables the administrator to not only make a more informed decision, but also to better understand the consequences and risk of his or her actions.

III. Overview

We motivated our problem with ubiquitous and abstract "warnings" – put into the security context, we can think of

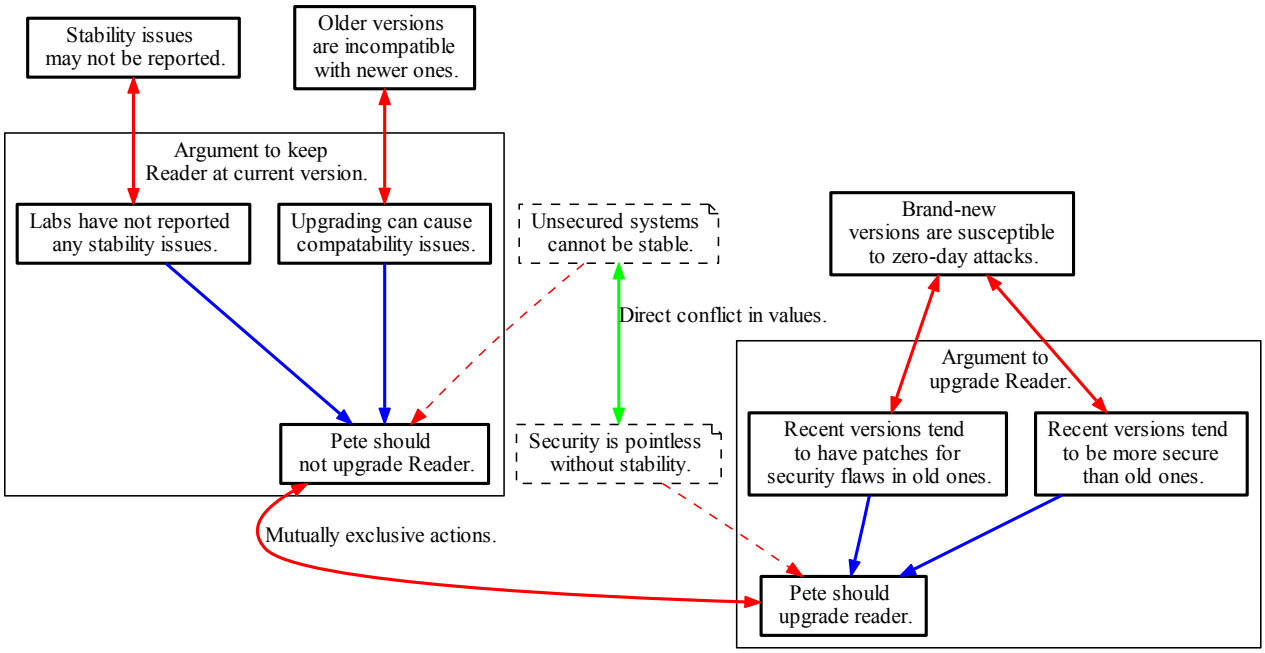

Fig. 2: Example visualization of a non-formal argument web to upgrade Adobe Reader or keep it at its current version. Blue edges imply support while red imply opposition/attacking. The dotted boxes and edges denote a difference in moral values. While not decisive, this kind of argument graph frames the debate of whether Pete (the administrator) should upgrade the program or not, illustrating the moral/tautological issues as well as the debate itself. Further analysis could, for example, seek more information (Pete asking the labs if there are any stability issues) or checking to see if any new vulnerabilities have been found in the upgraded version.

these warnings as reports generated by independent sensors each capable of detecting some sort of problem. Specific details of the sensors can vary widely, with example sensors including an intrusion detection system, an ingress/egress firewall, a vulnerability scanner, or even the constituents of the network themselves. These reports are consolidated and passed to ArgSEC to represent the current state of the network.

Individually, these reports may be indecipherable or unclear when shown directly to the administrator. Thus, when given these reports, ArgSEC attempts to draw inferences and extrapolate towards the future. This requires the use of an expansive knowledge base, composed in two main parts: *objects*, or things in the world, and *rules*, which relate and build upon objects. Argumentation is coded both syntactically inside the knowledge base as well as in the way reasoning is performed: starting with the initial reports, arguments are constructed by chaining rules together towards a conclusion and ultimately assembled into a framework for reasoning.

The consolidation of sensor reports into inferences drawn from the knowledge base creates a baseline of operation for ArgSEC. But, because ArgSEC is an *assistant*, one of the key mechanisms that we need is the ability for the administrator to ask *questions*. The lowest level example of this is when given a specific sensor report, the administrator may want to make queries for more information. In response, we construct arguments about the accuracy of the generating sensor, whether it is consistent with other reports, the actions that could be taken in response to it, the consequences of ignoring it, etc. The administrator may also ask what knowledge would be needed to reach a specific conclusion – i.e., what presumptions are necessary to conclude my system is secure? Firewall rules and general access control policy can be analyzed and condensed into a "big picture" configuration through the use of the knowledge base, allowing the administrator to query whether the current configuration meets desired policy.

In all of these cases, argumentation is essential as it exposes the *underlying reasoning* behind conclusions. This ultimately allows the administrator to make *better informed* decisions with the same amount of information and the end result is not only better performance from the administrator, but also the administrator's institution's reliability, and, in wide terms, for security as a whole.

References

[1] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial Intelligence*, vol. 77, pp. 321 – 357, 1995.