Argumentation Logic to Assist in Security Administration

Jeff Rowe
UC Davis
One Shields Ave
Davis, CA 95616
rowe@cs.ucdavis.edu

Karl Levitt
UC Davis
One Shields Ave
Davis, CA 95616
levitt@cs.ucdavis.edu

Simon Parsons
Brooklyn College, CUNY
2900 Bedford Ave.
Brooklyn, NY 11210
parsons@sci.brooklyn.cuny.edu

Elizabeth Sklar
Brooklyn College, CUNY
2900 Bedford Ave.
Brooklyn, NY 11210
sklar@sci.brooklyn.cuny.edu

Andrew Applebaum
UC Davis
One Shields Ave
Davis, CA 95616
applebau@ucdavis.edu

Sharmin Jalal
UC Davis
One Shields Ave
Davis, CA 95616
sjalal@ucdavis.edu

ABSTRACT

We present our preliminary work in using argumentation logics to reason about security administration tasks. Decisions about network security are increasingly complex, involving tradeoffs between keeping systems secure, maintaining system operation, escalating costs, and compromising functionality. In this paper we suggest the use of argumentation to provide automated support for security decisions. Argumentation is a formal approach to decision making that has proved to be effective in a number of domains. In contrast to traditional first order logic, argumentation logic provides the basis for presenting arguments to a user for or against a position, along with well-founded methods for assessing the outcome of interactions among the arguments. We demonstrate the use of argumentation in a reconfiguration problem, to diagnose the root cause of cyber-attack, and to set policies.

Categories and Subject Descriptors

I.0 [**Computing Methodologies**]: General—*Management, Security, Verification*

Keywords

argumentation, policy, security, diagnosis

1. INTRODUCTION

Security administrators typically find managing their systems daunting. This is true across a broad range of administrators and systems, including (1) home users with personal firewalls, anti-virus software and shared files, (2) administrators of large networks dealing with firewalls of 10,000 rules, intrusion detection systems, and thousands of potentially vulnerable computers, and (3) individual subscribers to online services like cloud applications and social networks, who are trying to understand and apply privacy settings appropriate to their needs. These kinds of users, and most others, have to make decisions that have a significant impact on their systems. However, these decisions must almost always be taken based on information that is not well understood. In a nutshell, users are given too much information, much of it incomplete or inconsistent, and they do not know the consequences of the decisions they are forced to make.

The provisioning of security for a system encompasses many tasks. In our work we consider three, but aim to develop a framework that is broadly encompassing and indicative of how security administrators conduct their business:

- A security policy is established, drawing from a range of information on best practice and taking into account likely attacks and the vulnerability of the system to those attacks.
- Some apparent anomaly in system operation is detected, and the process of diagnosis is undertaken to determine if an attack is underway, and what action, if any, should be taken to ensure system integrity.
- In the aftermath of a successful attack, the system itself, including possibly its security policy must be reconfigured. This reconfiguration needs to ensure protection against a possible attack underway but also future similar attacks without creating new vulnerabilities. Common reconfigurations include installing a patch, strengthening a firewall ruleset to block certain network traffic, and strengthening intrusion detection system rules. The reconfiguration must also respect the service expected of the system.

All of these tasks involve the integration of information from multiple sources, the need to handle information that is noisy and may be inconsistent, and—what we consider particularly important—the requirement to explain the outcome of this integration to humans or to mechanized processes that act on the information.

When establishing a policy, for example, it may be necessary to combine the contradictory advice given by different security experts, or to merge the conflicting rules for how best to secure parts of an enterprise network, the parts including individual workstations, network components, and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

NSPW' 12, September 18–21, 2012, Bertinoro, Italy.
Copyright 2012 ACM 978-1-4503-1794-8/12/09 ...$15.00.

servers. The overall objective of a policy is how best to secure an enterprise. To reach a decision on what policy to adopt, this information must be merged in such a way that a position is established that is consistent, and which satisfies the needs of the organization.

In diagnosis, what appears to be an anomaly may just be an unusual occurrence — our ability to detect anomalies is, after all, imperfect — and even if what we detect is truly anomalous, disparate occurrences may point to different possible attacks. Again information must be merged to form a coherent picture before any decision can be taken.

Reconfiguration clearly has many of the same features as the initial configuration task, but with added complexity. One reconfigures after an attack, and so needs to take into account the new vulnerabilities exposed by the attack, weighing up changes that will provide a defense against another attack of the same kind alongside any restrictions that those changes will make to the current use of the network, and any new potential vulnerabilities that the changes will create.

We are not the first to study these aspects of security, but we believe most work to date, albeit very encouraging and a good starting point for our work, has led to specific, custom solutions and does not provide a structure that can inform humans or mechanized devices that have to act on information given to them.

We use *argumentation* to provide this structure. Argumentation is a type of formal reasoning based on establishing reasons for and against propositions. We believe it is a general mechanism that will substantially augment existing security systems and lead to more powerful and informed security-decision making. In particular we are working to develop a system of argumentation that can support three activities: establishing a security policy, diagnosing attacks, and reconfiguring systems after an attack. We believe that argumentation can also have a role in advising users on privacy settings. In all cases, we hypothesize that argumentation will provide a formal approach to handling the noise and inconsistency in the data that needs to be used in decision making, and will make it possible to extract a consistent set of rules that can be applied to reach a decision. Furthermore, the arguments that are constructed may be used to explain the results of reasoning to human decision makers in a way that clarifies the situation and will improve the quality of future decisions.

2. BACKGROUND AND RELATED WORK

2.1 Argumentation

The term "argumentation", as used in computer science, refers to a broad range of work with its roots in philosophy and the study of human reasoning. In this tradition, a seminal work is Toulmin's "The Uses of Argument" [54] which began to formalize the idea that in reasoning it is not just the conclusion that is important, but also the reason that the conclusion was reached. In Toulmin's view, all conclusions are *defeasible* (i.e., possible to invalidate). Conclusions are constructed from data about the world, they are supported by a *warrant*, a reason for thinking that the conclusion holds, and this itself is derived from some *backing* (experience or experimental data). This whole structure is an *argument* All conclusions may be subject to a *rebuttal*, and the final truth will only be determined by taking account of all such rebuttals (each of which is itself an argument), examining the warrants, and reaching a verdict about which argument is most plausible. This view is summarized in the schema of Figure 1.

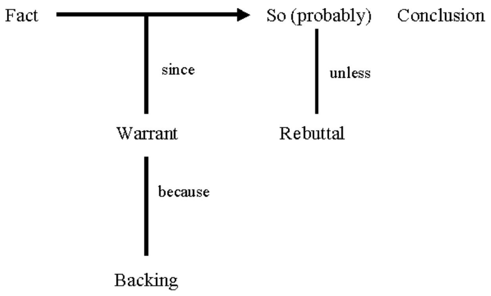


Figure 1: Toulmin's schema

There are three aspects of Toulmin's work that have strongly influenced research in computer science:

- The idea that all conclusions are defeasible suggests that argumentation fits closely with work on nonmonotonic logic and attempts to deal with inconsistent information.
- The idea that all conclusions can be disputed fits naturally with legal reasoning and communication between agents (autonomous systems whether human or intelligent software systems).
- The schema itself, which can be viewed as an explanation of why the conclusion was entertained, has been used in areas where complex reasoning needs to be examined.

On the defeasible reasoning side, the earliest work within computer science is due to Loui [28] and Lin [27, 26]; the former laid out, in broad terms, the capabilities of argumentation, while the latter focused on capturing existing systems of nonmontonic reasoning (such as default logic [46] and autoepistemic logic [32]). At the same time, Pollock [39, 40] explored the possibilities of argumentation in great detail, examined the structure of arguments and considered different ways in which one argument might defeat another.

The seminal work in this area, however, is due to Dung [10], who introduced the idea that one could consider the properties of a set of arguments at a purely abstract level. Given a set of arguments, and a relation that identifies pairs of arguments where one attacks the other, it is possible to establish several well-founded principles under which sets of *acceptable* arguments may be determined. For example, any argument that is not attacked is acceptable, and any argument that is attacked by an acceptable argument is not acceptable (this leads to a simple fixpoint definition of acceptable). This basic idea has been extended with preferences [3, 2, 41] and weights [11, 33].

From preferences and weights, it is a short step to combining argumentation with approaches for handling uncertainty. Krause [24] was among the first to consider how argumentation could capture different representations of uncertainty, while Kohlas [22] and his collaborators have written widely on combinations of argumentation and probability. The use of argumentation to handle uncertainty leads naturally to its use for decision making [18], and to reason about risks [30].

The notion of acceptability proposed by Dung [10] has a natural reading in terms of a dispute—one agent puts forward an argument, a second agent puts forward an argument that attacks the first argument, and then the first agent attempts to defeat the second argument. This correspondence can be exploited to describe interactions between agents. This perspective was first proposed by Sycara [52, 53] and soon became a standard approach to modeling negotiation [23, 35, 36, 37, 43]. More recent work has used argumentation to underpin a range of kinds of interaction [17, 29, 31, 38, 63], and the adversarial nature of argumentation-based interactions provides a particularly close fit with legal reasoning [5, 15, 42].

In the determination of acceptability, the relationship between arguments is crucial—the structure of the *argument graph* determines which arguments are acceptable. When we take a less abstract view, and examine the premises and conclusions of arguments, the resulting structure, like that in Figure 1, is also important. Using pictures of the structure of arguments to understand them is not new—the approach goes back at least as far as the work of Wigmore [60]. Reed *et al* [45] discuss the history of such diagrams and the use of software tools, such as Arucaria [44], Carneades [16], and Rationale [56], which were developed to allow users to draw argument graphs to help them reach better decisions.

2.2 Automated Security Management

There has been considerable work on automated security management, much of it in the security research literature and in products. Much of the work has a basis in formal logic and, consequently, uses various kinds of theorem provers and other automated reasoning tools, especially model checkers. Also, much of the work is aimed at a particular security component, such as firewalls (see below); this is not a criticism, as it is essential to understand individual security components and services before more general security issues can be successfully pursued. None of the work seems to lead to automated explanations for security decision making and none uses argumentation as the framework.

Below we highlight just a few of the notable work on security management to indicate that reasoning methods are feasible to improve the security and performance of security components. Then we present more detail on a knowledge-based approach to automated reconfiguration as background for casting this and other approaches to security management in terms of argumentation.

Reasoning about firewall rules [1] has been a fruitful application of logic for security. This work (and numerous others) has shown that firewall rules can be viewed as a (large) collection of simple programs which can be optimized and analyzed for inconsistencies. The analysis is not complicated, but the ability to handle 10,000 or more rules is impressive.

It has been recognized [61] that automated reconfiguration of a system undergoing attack can be cast as a planning problem; again there is other work on this topic, some coupling it with automated diagnosis, as described below. This cited paper also shows the relevance of game theory, provided it is possible to enumerate the moves of the defender, which is easier than accurately enumerating those of the attacker. There is been considerable work on the formulation of security policies in terms of logic, even for practical systems such as SE Linux. What is missing is formally linking policies to an actual implementation, although the verification community has done this for simple systems. As an illustration, we have shown [48] that it is possible to reason about the rules for a specification-based intrusion detection system (IDS) with respect to a policy for which the IDS is intended to detect policy violations. The work closest to our vision of automating reason in support of security administration is Cycorp's [25], which is now a product to assist security managers in assessing risk, such as to various kinds of attacks.

Below we provide more detailed background on the problem of automated reconfiguration of a system undergoing an attack, as this kind of reasoning is what we want our argumentation system to carry out.

Detecting known internet attacks is a solved problem. Many sophisticated counter measures such as automatic signature generation and distribution [20, 21, 47, 59] and automatic patch generation to fix vulnerabilities have also been developed. An open question, however, in secure systems research is, how might automatic system reconfiguration be performed to combat malicious attacks? Making this decision is hard in the presence of uncertain information. Suspending a service component is oftentimes desirable if it protects the larger system but it would obviously be harmful in response to a false alarm. Deliberate triggering by a malicious adversary might also cause self-inflicted denial-of-service. Intuitively, it seems desirable to shut-down the service until it is clear whether there is an attack or not. Balancing the consequences of maintaining a suspected service and risking its malicious faults against that of denying the service for protection is the key aspect of the reasoning addressed by argumentation logic.

Previously, we have used cost-sensitive, control theoretic principles to automate responses to global attacks in collaborative alert sharing intrusion detection systems (IDS). In that work, we found that for certain scenarios, to leave oneself open to attack might be the least expensive option as demonstrated by our algorithms. We also show that these algorithms do not need a great deal of information to make decisions.

In the control-theoretic model, the system consists of two main features: (1) a discrete-time dynamic system and (2) a cost function that is additive over time. The cost function is additive in the sense that the cost incurred accumulates over time. However, because of the presence of uncertainty in the actual state, the cost is generally a random variable and cannot be meaningfully optimized. We therefore formulate the problem as an optimization of the *expected cost* where the expectation is with respect to the joint distribution of the random variable involved. The optimization is over the controls, where each control, is chosen based on the current observation of the system. This is called **closed loop** optimization as opposed to **open loop** optimization when all controls have to be decided at once at time 0 without any knowledge of the state of the system at any time later.

Mathematically, in closed-loop optimization, we want to find a sequence of functions, mapping the system state into a control which when applied to the system minimizes the total expected cost. This sequence is referred to as a *policy* or *control law.* For each policy. Details of how we applied this to automatically block global attacks in a collaborative system can be found in [8].

The main drawback to this approach comes from assigning the appropriate cost values that accompany control actions. These values are subjective judgements to be made by expert administrators. Yet even an expert has no systematic way to configure such a system based upon single integer cost values in a way that is correct and consistent. It isn't clear how local costs combine if global optimization across an entire enterprise is desired. Costs of stopping services to all hosts, for example, isn't always the sum over the cost of stopping services to a single host. Adding additional cost combination functions only makes the configuration problem more difficult.

3. AN ARGUMENTATION FORMALISM FOR INTELLIGENT RECONFIGURATION

In our previous work, we have examined algorithms for automated reaction as a countermeasure to spreading Internet worm attacks [34, 4, 7, 8]. In this work, we used the probabilistic properties of imperfect early warning sensors to calculate an optimal response. This response involves stopping useful services and incurs some cost in our model. Compromise by the worm attack also incurs cost. Since the decision is based upon uncertain information, the question is: how is the automated response policy adjusted to minimize expected cost over time in the face of false reports? The major drawback to this work comes from the dependence upon numerical cost values. There is no principled way to set the values correctly for initial costs. Furthermore, cost values do not easily combine. For example, the cost of shutting down all 100 computers in an enterprise is not always 100 times more costly than shutting down a single computer. Argumentation is a promising alternative to our previous cost models, since it supports a much richer variety of decisions that might only require a partial ordering of consequences rather than relative numerical values.

For this problem, observable facts obtained by an administrator monitoring the system might include:

1. The number of IDS reports from sensors seen in the last time period.

2. The volume of traffic to a specific port.

3. Reports from cooperating partners that they have seen an Internet worm.

Conclusions would be:

1. Do nothing.

2. Turn off the potentially vulnerable service.

3. Filter traffic to the vulnerable at the gateway firewall.

Argumentation provides a mechanism to weigh up the evidence provided by the observable facts and relate them to conclusions about low-level system behavior. For example, consider the problem of reacting to reports of an Internet worm targeting a vulnerability in web servers. The *grounds* for action are the fact that there is evidence of a worm attack underway against web servers in our enterprise network. We wish to arrive at a reasonable course of action; in this case, is it a reasonable conclusion that traffic to port 80 should be blocked? Following Toulmin's formalization, the *warrant* in the argument structure is that blocking traffic to port 80 will thwart a worm that propagates to vulnerable web servers. The *backing* is that randomly scanning Internet worms find victims by attempting to connect to port 80 on random IP addresses. A serious *rebuttal* in this case, is that web services are essential to enterprise operations and will cause serious harm in the event of a false alarm. The argument structure for this problem is shown in figure 2.

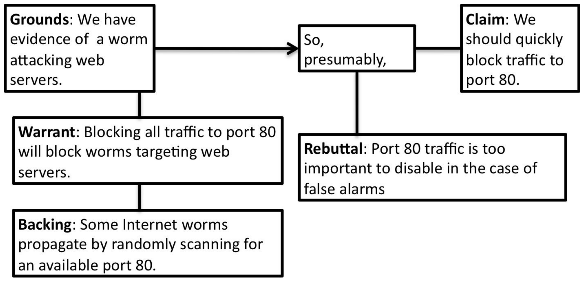
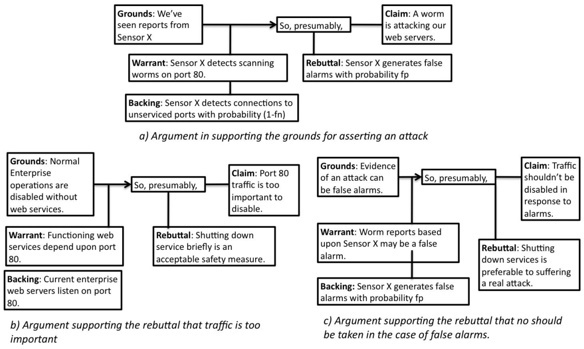

Grounds: We have evidence of a worm attacking web servers.
So, presumably,
Claim: We should quickly block traffic to port 80.
Warrant: Blocking all traffic to port 80 will block worms targeting web servers.
Rebuttal: Port 80 traffic is too important to disable in the case of false alarms
Backing: Some Internet worms propagate by randomly scanning for an available port 80.

Figure 2: Response Argument

Although this formulation may seem over simplistic, it is important to note that each of the elements are themselves arguments in their own right. The grounds, warrant, backing and rebuttal are all claims from other arguments in the framework. Suppose that the grounds asserting a worm attack at the outset was formed from the grounds that Sensor X generated an alert. This would be warranted because Sensor X detects scanning worms, with the backing that Sensor X detects connections to unserviced ports with probability $(1 - fn)$, where fn is the sensor's false negative rate. A rebuttal in this case, would be that Sensor X generates false alarms with probability fp. The corresponding argument structure is shown in figure 3(a). The claim in the rebuttal is supported on two separate grounds: that the suspect services are too important to disable, and that the attack reports aren't completely reliable. The structures of these sub-arguments are shown in figure 3(b) and 3(c). The final claims in both of these contribute to the grounds of the rebuttal in the original argument stating the importance of the traffic. Notice also that the backing for argument 3(c) is the same as the rebuttal in argument 3(a).


Grounds: We've seen reports from Sensor X
So, presumably,
Claim: A worm is attacking our web servers.
Warrant: Sensor X detects scanning worms on port 80.
Rebuttal: Sensor X generates false alarms with probability fp
Backing: Sensor X detects connections to unserviced ports with probability (1-fn)
Grounds: Normal Enterprise operations are disabled without web services.
So, presumably,
Claim: Port 80 traffic is too important to disable.
Warrant: Functioning web services depend upon port 80.
Rebuttal: Shutting down service briefly is an acceptable safety measure.
Backing: Current enterprise web servers listen on port 80.
Grounds: Evidence of an attack can be false alarms.
So, presumably,
Claim: Traffic shouldn't be disabled in response to alarms.
Warrant: Worm reports based upon Sensor X may be a false alarm.
Rebuttal: Shutting down services is preferable to suffering a real attack.
Backing: Sensor X generates false alarms with probability fp

a) Argument in supporting the grounds for asserting an attack

b) Argument supporting the rebuttal that traffic is too important

c) Argument supporting the rebuttal that no should be taken in the case of false alarms.

Figure 3: Sub Arguments

By refining the overall reasoning into connected subarguments, the high-level reasoning structure is populated with the relevant technical details at lower levels. Relationships between the detailed facts contributing to the decision-making are made explicit in the formal argumentation structure, and changing the strength of individual assertions allows different conclusions to be reached. Eventually, the overall problem of intelligent reconfiguration is represented by a graph of supporting claims. The graph for the example above is shown in figure 4.

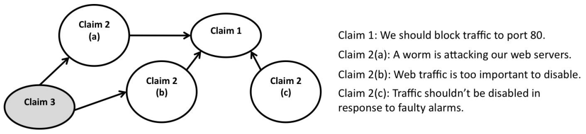

Claim 2 (a)
Claim 1
Claim 2 (b)
Claim 2 (c)
Claim 3
Claim 1: We should block traffic to port 80.
Claim 2(a): A worm is attacking our web servers.
Claim 2(b): Web traffic is too important to disable.
Claim 2(c): Traffic shouldn't be disabled in response to faulty alarms.

Figure 4: Argument Graph

Claim 3 is the argument supporting the claim that Sensor X generates false alarms with probability fp. If this claim, backed by observations of Sensor X's performance, were called into question, then two lines of reasoning would be undermined, with potentially significant changes to the final conclusion that traffic to port 80 would be blocked. We can use techniques like those from [3, 10] to establish which claims hold in situations like this.

The example above represents a specific *case* to be reasoned about. We believe that argument models like these can be useful in a wide variety of cyber security situations. In the example, the specific attack being considered is common to all argumentation elements. The specific attack serves as a token that defines a portion of the reasoning in this case. Presumably a very similar line of reasoning would apply to a variety of similar cases of distributed attacks. We are identifying common sub-arguments, and argument subgraphs that can be generalized and used in a wide variety of situations. Rather than building new argument graphs for each new case encountered, the argument is composed from components already vetted in similar situations. Developing specialized argumentation schemes [57] allows one to capture common patterns of reasoning like this, and arguments for and against the conclusion of a worm attack can be combined in well-founded ways [6]. This form of reasoning has been shown to perform well in medical diagnosis [12, 13, 14] and in applications where the focus is less on identifying a state (as here where we are identifying if is there a worm) and more on choosing a course of action [9, 58] (in our example, how to respond to the worm if we decide this is necessary).

Finally, while the argumentation structure described this far identifies the relationship between the reasoning components, there is still the question of how the weights are defined and taken into account in the final analysis. Ultimately it is up to the administrator to judge which resources are important. Argumentation models can, however, make the codification of the actual values more transparent and less prone to misconfiguration. Rather than basing the component weights upon numerical value, a simple ranking of the level of support into a small number of fixed categories can be used. The key idea here is that support is based upon belief rather than truth as in classical logic. Generally, the more independent reasons to sustain a claim, the greater the confidence in the decision. One approach is to use an aggregation function that simply assigns the greatest support of the components to the claim. For example, suppose there are two ranks: normal support and conclusive support. A rebuttal with conclusive support would outweigh a warrant with normal support and the argument claim would be set conclusive. If there were two contributing claims to a argument grounds then concluding claim would be assigned the weight of the highest ground. Multiple rebuttals and multiple backings would have their support combined as,

$$S_c = (p - c)/(p + c) \tag{1}$$

where S_c is support for the claim and p and c are the pros and cons; the number of backing supports and the number of rebuttals. The attractiveness of this simple model is the ease with which it can be configured. We will also investigate more sophisticated alternatives including Dempster-Shafer based methods for assigning prior weights. The question here is how sensitive the overall case claim is to specific aggregation procedures.

3.1 Argumentation for Attack Diagnosis

Another security application for argumentation-based reasoning is in security diagnosis. Oftentimes specific symptoms of malicious attacks are no different than natural faults or transient unusual but valid use of the system. Aside from isolating the true source of the problem, oftentimes one want to know quickly whether the behavior is the result of an attack or not. For example, in critical infrastructures, such as a SCADA controlled power generation plant, devices operating outside the bounds of tolerance in the control system may indicate failure of some component. The proper course of action would be to switch to a redundant component while a repair is performed. This typically is done by the integrated safety control system which has been designed to handled random natural faults. If instead, however, an attacker has gained access to the SCADA system and is manipulating the device to cause damage, operators must be alerted before the attacker also gains access to the backup device which may cause catastrophic failure.

What is needed here is a method for taking all available *symptoms* and finding the associated *problem* the provides the best match. The most common approach to solving this problem is to use Bayesian reasoning to get a probabilistic likelihood estimate for the candidate problems. This works well for natural faults since the conditional probabilities involved are known with some accuracy. The difficulty in applying this to diagnosing security events is similar to the difficulty in automating response to attacks. It is very difficult to configure the system with accurate probabilities associated with attacker actions; attackers wo do a variety of actions depending upon specific goals or methods.

To address this issue we have previously employed a codebook based approach borrowed from the network management community [62]. The codebook approach is based upon a causality graph of problems and symptoms. Problems and symptoms are both nodes in the directed graph and directed edges represent causal relationships between them. Problems cause a variety of symptoms, and problems also cause other problems. Symptoms are terminal nodes since they aren't the cause of problems themselves (symptoms may cause other symptoms but this results in no additional information and can be pruned). The causality graph is then used to produce a problem/symptom codebook where each problem is associated with a bit-vector of associated symptoms. Simple symptom lookup would be sufficient except that there may be false alarms or missing symptom information. To address this the codebook approach uses minimal distance decoding. The problem whose code has minimal Hamming distance to the symptom vector is decoded. In general the codebook approach can correct observation errors in $k-1$ symptoms and detect k errors as long as k is less than or equal to the radius of the codebook. The drawback to this approach is that all symptoms are considered with equal likelihood. When reasoning about the root cause of a set of symptoms, some problems/symptoms carry greater weight. Alerts of a gas exiting a leaking pump carry more importance than vibrations detected at the pump. The codebook approach would treat these two with equal importance, perhaps even eliminating the critical alert due to redundancy.

We employ an argumentation based reasoning approach to this security diagnosis problem. The main advantages to using argumentation are that it doesn't depend upon accurate known prior probabilities yet it can still handle weighted claims. The conclusion reached by a argumentation-based diagnosis uses the reasoning chain used to reach the conclusion instead of relying upon the detailed values of the numbers that went into the assigned weights. A partial ordering of weights is sufficient to give a useful result.

As a simple example, consider the situation where streaming video-conference viewed on a laptop computer has its quality suddenly reduced. Two competing lines of reasoning must now be followed: one that ends with a justified claim that a denial-of-service attack is being conducted against the wide area network, and the other that a natural fault in the local wireless access point is creating the problem. Suppose that a variety of symptoms are available to an administrator: the performance of the video-conference service in other portions of the wide area network, the volume of traffic on critical network links, IDS reports of obvious DDoS attacks, etc. Good performance of the service in other parts of the network would be a strong rebuttal to the claim of DDoS attack against the shared WAN. Heavy volumes of traffic on a WAN link would only be weak backing for DDoS attack because a sub-argument claiming heavy, yet valid, use of the network for other services could still cause the same symptom and undermine the DDoS claim. High quality IDS reports of DDoS attacks would be exceptionally strong backing for the DDoS claim but, in the case of a rare false alarm might still be over-ridden by the combined weight of the other arguments when everything else appears normal.

The main challenge in performing this type of diagnosis in IP network environments is encoding sufficient knowledge into the argumentation framework to make useful decisions. Our approach is to start with the limited number of symptoms available. Possible problems associated with these are assigned as claims or warrants with the symptom as backing, similar to the codebook approach. The existence of certain symptoms actually undermine the claim of some problems, a negative correlation case the codebook approach fails to take into account, and are added as rebuttals to the appropriate argument claims. Finally, symptoms are place in a partial order with an increasing weight of evidence.

We believe that the argumentation approach will be particularly useful in critical infrastructure protection systems such as SCADA controlled power grid installations, automated manufacturing and civil services. Unlike general purpose IT systems, these cyber-physical systems are constrained by the physical processes they control and the limited functionality of the devices involved. Encoding knowledge of the normal system operations in this case can be based upon the design specifications of the process itself and the capabilities and programming of the control system.

3.2 Argumentation as Policy Recommendation

Setting correct access control and privacy policies has never been an easy task for system administrators. Traditional policies rely upon restricting access based upon authenticated identities. In this environment, users with limited technical expertise might be expected to implement a reasonable access control policy on their personal computing resources because: 1) these resources were only shared by the limited number individuals with physical access 2) these individuals were highly trusted family members or friends, and 3) network access control to the single internet connection can be implemented using a variety of inexpensive commercial devices. With the recent upswing in the participation in online social networks, however, the problem of access control has changed. The resource to be protected is the personal information of the users themselves. Access to that information is potentially available to every user of the online social networking service; in the case of Facebook the number of active users is 500 million. In addition to users, commercial applications are deployed on the Facebook platform. In one well publicized case, the popular Zynga facebook game developer was found to be selling private information to third party advertising firms. Managing private information, so that it is shared with trusted parties in the social network and denied otherwise is exceedingly difficult for typical users.

We apply the argumentation framework as a method for configuring access control policies in this complex environment. In our previous work, we have developed dynamic models for trusted communications within the social network environment. In the KarmaNet[50] work, we show how desirable messages can be transmitted to unknown members in a social network from well-behaved members, while undesirable messages result int the suppression of the messaging ability of spammers. This is accomplished by nodes maintaining trust values of all connected peers and setting a probability of message forwarding as a function of trust. We prove that, using this system, the number of undesirable messages is bounded over the lifetime (rather than a fixed time window) of a spammer. We implemented this scheme in our Davis Social Links suite of applications on Facebook [49] as a method for controlling built-in messaging. Also, under active development is a Facebook application privacy proxy service called the Facebook Application Identity Transformation and Hypervisor (FAITH)[55]. Users of Facebook applications can gain fine-grained control over access to their private information by requiring that third-party application interactions be mediated by FAITH. A remaining obstacle to managing privacy with these systems is the difficulty in configuring a reasonable access control policy to unknown and untrusted applications. Another application of the argument-based security management is in the configuration of these complex privacy policies in online social network environments. For example, a user may be uncertain whether to grant permissions to an application. Using the social network, users could solicit arguments from trusted users within a circle of friends. Trust values, supplied by the Davis Social Links system could serve as weights on backing as well as on rebuttal. Arguments from different friends could be combined into an overall case to make complex policy decisions. Additionally, social network members who have supplied useful policy arguments in the past would have their trust values upgraded accordingly. Users supplying poor arguments would be downgraded and have less of an influence.

4. TOWARDS AN ARGUMENTATION-BASED SECURITY MANAGER'S ASSISTANT

Argumentation-based reasoning is seldom used in live systems. Most implementations assist knowledgable users in formalizing their domain-specific reasoning procedures for analysis. We wish to go beyond this and produce a proof-of-concept argumentation prototype engine to evaluate the use of argumentation for security reasoning. There are numerous theorem provers[51, 19], that could be the core for our prototype, but we propose to use the Cambridge HOL system. It is the theorem prover for which we have the most familiarity and it is the easiest to modify and extend, but the downside to this flexibility is that it is likely the slowest of the popular theorem provers; this lack of performance can be somewhat ameliorated with special purpose decision engines, such as with argumentation based security reasoning. The flexibility of HOL permits the mechanization of inference rules for standard forward reasoning (e.g., driven by sensor alerts) or *tactics*, which are inference rules mechanized for backwards reasoning. These are employed when details of an argument are explained or challenged. HOL provides many build in inferences, collected as *theories* for standard reasoning. We extend the standard reasoning to handle uncertainty, for example as in the Dempster-Shafer theory.

5. DISCUSSION

Based upon the discussions in the workshop, we take this opportunity to clarify some of the points in our approach. The argumentation framework that we present is primarily intended to support novel new reasoning strategies rather than to provide an alternate method for arriving at a proven conclusion. Rule-based reasoning systems in cyber-security, such as static analysis of programs, verification of policy, and planning of reactions to adversary actions, typically cannot handle conflicting assertions. These are considered to be errors in configuration to be resolved by the user or administrator. In fact, the goal of many rule based systems is explicitly intended to discover these conflicts, as in static analysis. With argumentation, however, we wish to extend rule-based reasoning to support reasoning about security in the presence of inherent conflicting assertions. Argumentation logic structures, then, have the unfamiliar feature of not providing for definite resolution of a proposition into a final proven claim, as in standard logics. Rather than proving the validity of a conclusion, argumentation encodes the structure of reasoning for *and* against claims. The question then becomes, what new reasoning capabilities can argumentation logic provide? One example is questions like, "What new information do I need to best resolve the conflict?" In a firewall configuration, say, deciding whether to block a port or not would depend upon a variety of factors. It isn't obvious how the pros and cons of the decision are related. The question of whether an attack has occurred or not might be irrelevant in a specific argument structure if the critical question is how much you value the service in the first place. If that question were answered, all other conflicting assertions would be defeated allowing for resolution. In a distributed computing environment, expressing a decision procedure in argumentation logic would allow a user to realize that the final decision actually depended upon whether you trusted one actor over another, say. Given that fact, all other arguments might be defeated. The argumentation logic itself doesn't guarantee any resolution of conflicts. In fact it allows for reasoning even in the presence of such conflict.

During the workshop, much of the discussion focused upon resolution systems. A resolution system is a procedure for converting the inherently ambiguous argument claims into a structure with a single, well-defined conclusion. Although argumentation logics don't inherently provide for resolution of propositions, one can devise a variety mechanisms to achieve resolution for a given argumentation logic structure. Typical approaches are to assign weights to specific claims. Weights are combined for competing reasoning chains and the highest weighted argument wins. We have used the Dempster-Shafer theory of evidence to weight arguments based upon the likelihood of the claim's truth. The trustworthiness of the actor supplying the argument is also a possible weighting mechanism in a distributed computing environment. Another approach is to use the costliness of accepting a specific claim as true. One could imagine a wide variety of clever mechanisms for argument resolution in cyber-security applications and we believe this could provide a fruitful avenue for future work in the research community. However, we wish to make it clear that these argument resolution mechanisms are somewhat arbitrary and their suitability is domain dependent. Argumentation logic's novelty comes from the *lack* of any requirements for proposition resolution. In fact, its purpose is to support reasoning when resolution isn't possible due to inherent, unresolved logical conflicts.

Finally, we wish to thank the NSPW organizers and participants for the fruitful discussions that are only possible in the refreshing interactive format of this workshop.

6. REFERENCES

[1] Ehab Al-Shaer, Charles R. Kalmanek, and Felix Wu. Automated security configuration management. *J. Network Syst. Manage.*, 16(3):231–233, 2008.

[2] L. Amgoud and C. Cayrol. On the acceptability of arguments in preference-based argumentation framework. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 1–7, 1998.

[3] L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artifical Intelligence*, 34(3):197–215, 2002.

[4] Ivan Balepin, Sergei Maltsev, Jeff Rowe, and Karl Levitt. Using Specification-Based Intrusion Detection for Automated Response. In *Recent Advances in Intrusion Detection: 6th international symposium, RAID 2003, Pittsburgh, PA, USA, September 8-10, 2003: proceedings*, page 136. Springer-Verlag New York Inc, 2003.

[5] T. J. M. Bench-Capon, T. Geldard, and P. H. Leng. A method for the computational modelling of dialectical argument with dialogue games. *Artificial Intelligence and Law*, 8:233–254, 2000.

[6] P. Besnard and A. Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128:203–235, 2001.

[7] Senthil G. Cheetancheri, John M. Agosta, Denver H. Dash, Karl N. Levitt, Jeff Rowe, and Eve M. Schooler. A distributed host-based worm detection system. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, page 113. ACM, 2006.

[8] Senthil G. Cheetancheri, John M. Agosta, Karl N. Levitt, S. Felix Wu, and Jeff Rowe. Optimal Cost, Collaborative, and Distributed Response to Zero-Day Worms-A Control Theoretic Approach. In *Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, page 250. Springer, 2008.

[9] A. S. Coulson, D. W. Glasspool, J. Fox, and J. Emery. Rags: A novel approach to computerized genetic risk assessment and decision support from pedigrees. *Methods of Information in Medicine*, 2001.

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

[11] P. E. Dunne, A. Hunter, P. McBurney, S. Parsons, and M. Wooldridge. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artificial Intelligence*, (in press).

[12] J. Emery, R. Walton, A. Coulson, D. Glasspool, S. Ziebland, and J. Fox. Computer support for recording and interpreting family histories of breast and ovarian cancer in primary care (rags): Qualitative evaluation with simulated patients. *British Medical Journal*, 319(7201):32–36, 1999.

[13] J. Emery, R. Walton, M. Murphy, J. Austoker, P. Yudkin, C. Chapman, A. Coulson, D. Glasspool, and J. Fox. Computer support for recording and interpreting family histories of breast and ovarian cancer in primary care: Comparative study with simulated cases. *British Medical Journal*, 321(7252):28–32, 2000.

[14] J. Fox, V. Patkar, and R. Thomson. Decision support for health care: the PROforma evidence base. *Informatics in Primary Care*, 14(1):49–54, 2006.

[15] T. F. Gordon. The Pleadings Game: An exercise in computational dialectics. *Artificial Intelligence and Law*, 2:239–292, 1994.

[16] T. F. Gordon, H. Prakken, and D. Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–11):875–896, 2007.

[17] K. Greenwood, T. Bench-Capon, and P. McBurney. Structuring dialogue between the People and their representatives. In R. Traunmüller, editor, *Electronic Government: Proceedings of the Second International Conference (EGOV03), Prague, Czech Republic*, Lecture Notes in Computer Science 2739, pages 55–62, Berlin, Germany, 2003. Springer.

[18] A. Kakas and P. Moraitis. Argumentation based decision making for autonomous agents. In J. S. Rosenschein, M. Wooldridge, T. Sandholm, and M. Yokoo, editors, *2nd International Conference on Autonomous Agents and Multi-Agent Systems*, New York, NY, 2003. ACM Press.

[19] M. Kaufmann and J S. Moore. A precise description of the ACL2 logic. In *http://www.cs.utexas.edu/users/moore/publications/km97a.ps.gz*. Dept. of Computer Sciences, University of Texas at Austin, 1997.

[20] H. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of the USENIX Security Symposium*, 2004.

[21] H. Kim, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of the IEEE Symposium on Security and Privacy 2005*, Oakland, CA, 2005.

[22] J. Kohlas. Probabilistic argumentation systems: A new way to combine logic with probability. *Journal of Applied Logic*, 1(3–4):225–253, June 2003.

[23] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104(1–2):1–69, 1998.

[24] P. Krause, S. Ambler, M. Elvang-Gørannson, and J. Fox. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, 11 (1):113–131, 1995.

[25] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):32–38, 1995.

[26] F. Lin. An argument-based approach to non-monotonic reasoning. *Computational Intelligence*, 9:254–267, 1993.

[27] F. Lin and Y. Shoham. Argument systems: a uniform basis for nonmonotonic reasoning. In *Proceedings of the 1st International Conference on Knowledge Representation and Reasoning*, pages 245–255, San Mateo, CA, 1989. Morgan Kaufmann.

[28] R. P. Loui. Defeat among arguments: a system of defeasible inference. *Computational Intelligence*, 3(3):100–106, 1987.

[29] N. Maudet and F. Evrard. A generic framework for dialogue game implementation. In *Proceedings of the 2nd Workshop on Formal Semantics and Pragmatics of Dialogue*, University of Twente, The Netherlands, May 1998.

[30] P. McBurney and S. Parsons. Risk agoras: Dialectical argumentation for scientific reasoning. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Stanford, CA, USA, 2000. UAI.

[31] P. McBurney and S. Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language, and Information*, 11(3):315–334, 2002.

[32] R. C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[33] S. Nielsen and S. Parsons. An application of formal argumentation: Fusing Bayesian networks in multi-agent systems. *Artificial Intelligence*, 171(10–15):754–775, 2007.

[34] Dai Nojiri, Jeff Rowe, and Karl Levitt. Cooperative Response Strategies for Large Sacle Attack Mitigation. In *Proceedings of the DARPA Information Survivability Conference and Exposition. DISCEX*, 2003.

[35] S. Parsons and N. R. Jennings. Negotiation through argumentation — a preliminary report. In *Proceedings of Second International Conference on Multi-Agent Systems*, pages 267–274, 1996.

[36] S. Parsons and P. McBurney. Argumentation-based dialogues for agent coordination. *Group Decision and Negotiation*, 12(5):415–439, 2003.

[37] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261—292, 1998.

[38] S. Parsons, M. Wooldridge, and L. Amgoud. Properties and complexity of formal inter-agent dialogues. *Journal of Logic and Computation*, 13(3):347–376, 2003.

[39] J. L. Pollock. Defeasible reasoning. *Cognitive Science*, 11:481–518, 1987.

[40] J. L. Pollock. How to reason defeasibly. *Artificial Intelligence*, 57:1–42, 1992.

[41] H. Prakken and G. Sartor. Argument-based logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 1997.

[42] H. Prakken and G. Sartor. Modelling reasoning with precedents in a formal dialogue game. *Artificial Intelligence and Law*, 6:231–287, 1998.

[43] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *Knowledge Engineering Review*, 18(4):343–375, 2003.

[44] C. Reed and G. Rowe. Araucaria: Software for argument analysis, diagramming and representation. *International Journal of AI Tools*, 14(3–4):961–980, 2004.

[45] C. Reed, D. Walton, and F. Macagno. Argument diagramming in logic, law and artificial intelligence. *Knowledge Engineering Review*, 22(1):87–109, 2007.

[46] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[47] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated worm fingerprinting. In *In OSDI*, pages 45–60, 2004.

[48] Tao Song, Calvin Ko, Jim Alves-Foss, Cui Zhang, and Karl N. Levitt. Formal reasoning about intrusion detection systems. In *RAID*, pages 278–295, 2004.

[49] M. Spear, X. Lu, N. Matloff, and S. F. Wu. Davis Social Links or: How I Learned To Stop Worrying And Love The Net. In *SCA '09: Proceedings of the International Symposium on Social Computing Applications*, 2009.

[50] M. Spear, X. Lu, N. Matloff, and S. F. Wu. KarmaNET: Leveraging Trusted Social Paths to Create Judicious Forwarders. In *IFCIN '09: Proceedings of the First International Conference on Future Information Networks*, 2009.

[51] SRI International. Pvs specification and verification system. `http://pvs.csl.sri.com/`.

[52] K. Sycara. Argumentation: Planning other agents' plans. In *Proceedings of the Eleventh Joint Conference on Artificial Intelligence*, pages 517–523, 1989.

[53] K. Sycara. Persuasive argumentation in negotiation. *Theory and Decision*, 28:203–242, 1990.

[54] S. Toulmin. *The Uses of Argument*. Cambridge University Press, Cambridge, England, 1958.

[55] UCDavis DSL Group. Dsl faith. `http://http://apps.facebook.com/dsl_faith/`.

[56] T. van Gelder. The rationale for Rationale$^{\text{TM}}$. *Law, Probability and Risk*, 6:23–42, 2007.

[57] D Walton, C. Reed, and F. Macagno. *Argumentation Schemes*. Cambridge University Press, Cambridge, UK, 2008.

[58] R. Walton, C. Gierl, H. Mistry, M. P. Vessey, and J. Fox. Evaluation of computer support for prescribing (CAPSULE) using simulated cases. *British Medical Journal*, 315:791–795, 1997.

[59] Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In Erland Jonsson, Alfonso Valdes, and Magnus Almgren, editors, *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings*, volume 3224 of *Lecture Notes in Computer Science*, pages 203–222. Springer, 2004.

[60] J. H. Wigmore. The problem of proof. *Illinois Law Review*, 8(2):77–103, 1913.

[61] Yu-Sung Wu, Bingrui Foo, Yu-Chun Mao, Saurabh Bagchi, and Eugene H. Spafford. Automated adaptive intrusion containment in systems of interacting services. *Computer Networks*, 51(5):1334–1360, 2007.

[62] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. *Communications Magazine, IEEE*, 34(5):82–90, 1996.

[63] T. Yuan, D. Moore, and A. Grierson. Educational human-computer debate: A computational dialectics approach. In G. Carenini, F. Grasso, and C. Reed, editors, *Proceedings of the Workshop on Computational Models of Natural Argument*, 2002.