Firewall configuration: An application of multiagent metalevel argumentation

Andy Applebaum[1], Zimi Li[2], Ali Raza Syed[2] and Karl Levitt[1], Simon Parsons[2,3], Jeff Rowe[1], and Elizabeth Sklar[2,3]

[1] Department of Computer Science, University of California, Davis, CA 95616.
{applebau,levitt,rowe}@cs.ucdavis.edu
[2] Department of Computer Science, The Graduate Center
City University of New York, 365 5th Avenue, New York, NY 10016, USA
{zli2,asyed2}@gc.cuny.edu
[3] Department of Computer & Information Science, Brooklyn College,
City University of New York, 2900 Bedford Avenue, Brooklyn, NY 11210, USA
{sklar,parsons}@sci.brooklyn.cuny.edu

Abstract. Firewalls are an important tool in the provision of network security. Packet filtering firewalls are configured by providing a set of rules that identify how to handle individual data packets that arrive at the firewall. In large firewall configurations, conflicts may arise between these rules. Argumentation provides a way of handling such conflicts that illuminates their origin, and hence can help a system administrator understand the effects of a given configuration. We look in particular at the use of a system of metalevel argumentation for firewall configuration, showing how it makes conflicts and their origins especially clear, and showing how different instantiations of a metalevel argumentation system provide alternative ways to resolve conflicts.

1 Introduction

Providing network security is a major problem today, both in academic computer science which aims to come up with new techniques for securing networks, and in the practical world of information technology, where system administrators struggle to prevent unauthorised users from breaking into the networks that they manage. Firewalls, first introduced in 1987 [19], are one of the core components of a network security implementation. A *firewall* is a combination of hardware and software that isolates an organization's internal network from the Internet at large, allowing some packets to pass and blocking others [21]. The decision about which packets to pass and which to block is made according to some *policy*, and the *configuration* of a firewall is the business of implementing this policy.

As we will discuss below, firewall policies are set by specifying a set of rules, and there are a number of well-recognised problems in doing this. These problems relate to rules conflicting, having domains that overlap, and include redundancy (where the effect of a policy differs from that intended because some rules can never have any effect). Such *anomalies* arise from the complexity of setting up firewall policies in complex

environments such as large organisations, especially when different parts of the overall firewall policy are set by different individuals.

In this paper, we discuss how an argumentation-based framework can be used to analyse a firewall policy. In particular, we examine the use of the metalevel argumentation framework of [23], choosing this system as the basis of our investigation because we believe that the metalevel reasoning about the acceptability of arguments helps to make the reasons for conflict between policies especially clear, and makes it easy to understand how different strategies for resolving such conflicts work.

The remainder of this paper is organised as follows. Section 2 provides a brief introduction to firewalls and issues in their confiiguration. Section 3 introduces the specific metalevel argumentation system that we will be using. Section 4 describes several ways in which this metalevel system can be used to represent firewall configurations and potentially resolve conflicts in the firewall rules. Then Section 6 concludes.

2 Problems in firewall configuration

There are different types of firewall which function in different ways — packet-filtering firewalls, application/proxy firewalls, and network address translation. Packet-filtering firewalls operate at the network layer, not allowing packets to pass through the firewall unless they match the established policy rule set. Routers can provide a very common form of packet-filtering firewall. Packet-filtering usually makes decisions based on the following characteristics:

- Source and/or destination IP addresses
- Source and/or destination port numbers
- Protocol types
- Other parameters within the IP header

A network administrator configures the firewall based on the policy, for example blocking and allowing packets based on what protocol they match and which IP address they have as their destination.

Application firewalls, as indicated by the name, work at the application layer. These devices act as proxy machines for requested services. Requests are sent to a proxy machine, which then makes those requests to the Internet on behalf of the local client. A proxy machine acts as a buffer between "bad" remote users and the internal network client machines. Network Address Translation (NAT) also operates at the network layer, providing the capability to change the source and/or destination IP address. This is common when a private address space is used internally. The simplest type of NAT provides a one-to-one relationship between inside and outside IP addresses. In this type of NAT, only the IP header related to the IP address needs to be changed. The rest of the packet can be left unchanged. In the remainder of this paper we will concentrate on packet-filtering firewalls, but believe that the techniques we develop could be applied to the other types we have listed above.

Table 1 lists some possible polices for a packet-filtering firewall and how they would be addressed with a packet-filter firewall. If a firewall were to use this set of rules, when a packet comes in it would be checked against rule 1. If rule 1 applied to that packet,

Table 1. Polices and corresponding filtering rules

Rule	Policy	Firewall Setting
1	Block a malicious sender	Drop all packets from 55.55.55.55
2	Allow Web services	Allow all incoming TCP packets from any IP address, port 80
3	Block DNS services	Drop all incoming UDP packets from any IP address, port 53
4	Block all	Drop all incoming packets from any IP address
5	Allow FTP services	Allow all incoming TCP packets from any IP address, port 21

Table 2. An example of packet filtering firewalls

Rule	Action	Protocol	Source IP	Source port
1	block	$*$	55.55.55.55	$*$
2	allow	TCP	$*$	80
3	block	UDP	$*$	53
4	block	$*$	$*$	$*$
5	allow	TCP	$*$	21

then the action specified by rule 1 would be taken. Otherwise, the packet would be compared to rule 2. This process is repeated until one of the rules correctly specifies the packet; the first rule that does is the one that's applied, ignoring all rules after it. If no rule matches an incoming packet, some default rule (which might, for instance, be to let the packet pass since there is no specific rule to block it), would be applied.

Now consider the slightly more formal example in Table 2 and imagine a packet arriving that uses TCP, coming from 55.55.55.55 on port 80. Rule 1 specifies blocking it while rule 2 says to allow it. Since rule 1 is positioned before rule 2, the ultimate action is to block, but it is clear that there is some kind of a conflict occuring. As another example, the packet using UDP from 55.55.55.55 on port 53 would be blocked; while three of the rules say to block it, only the first one is technically "enforced".

The situations highlighted in both of these examples could be considered problematic, and both are what [1] calls an *anomaly* in a firewall policy. [1] defines four anomalies in terms of relations between rules:

Shadowing Rule a is said to shadow rule b if a has higher-priority than b, a and b specify different actions, and every packet that satisfies b also satisfies a.

In shadowing, the two rules are in conflict on *every* packet.

Correlation Rule a and b are correlated if a and b specify different actions and some packets that satisfy a also satisfy b and vice versa.

In correlation, the rules conflict on *some* packets.

Redundancy Redundancy occurs in two cases. In the first case, redundancy occurs if two rules a and b are such that all packets that satisfy a satisfy b, a and b specify the same actions, and b is higher priority than a.

Table 3. Rules and their implementation in Linux

Policy	Firewall Setting
Block a malicious sender	iptables -A INPUT -p 0 -s 55.55.55.55 -j DROP
Allow Web services	iptables -A INPUT -p tcp -dport 80 -j ACCEPT
Block DNS services	iptables -A INPUT -p udp -dport 53 -j DROP
Block all	iptables -A INPUT -p 0 -j DROP
Allow FTP services	iptables -A INPUT -p tcp -dport 21 -j ACCEPT

In the second case, redundancy occurs if all packets that satisfy *a* also satisfy *b*, *a* and *b* specify the same actions, *a* is higher priority than *b*, and *a* is not involved in any correlation anomalies.

In both cases, the lower priority rule will never be applied.

Generalization Rule *a* is said to generalize rule *b* if *b* has higher priority than *a*, *a* and *b* specify different actions, and every packet that satisfies *b* also satisfies *a*.

In generalization there is shadowing but the conflict is resolved by the priority.

In the example in Table 2, rule 4 shadows rule 5, since every instance of rule 5 is also an instance of rule 4, and they specify different actions. Rule 1 and rule 2 are correlated, since a TCP packet from 55.55.55.55 on port 80 is an instance of both rule 1 and rule 2, and they specify different actions. Rule 4 generalizes rule 2, since every instance of rule 2 is also an instance of rule 4, and they specify different actions. Rule 3 is redundant because every instance of rule 3 is also an instance of rule 4, and they specify the same action. In case our examples might seem a little abstract, consider Table 3. This shows the rules from Table 1 as they would be encoded in the `iptables` configuration file that controls the packet filter built into the Linux kernel.

Now, in a small set of firewall rules such as these, it is easy enough to detect and fix anomalies. However, firewalls, especially in large organizations with many machines on a network, can include many hundreds of rules. In such a case, detection and correction of anomalies is much harder. The problem is even more complex when firewalls are composed of different components, each requiring some part of a policy that is applied to a whole organisation, and as we shall discuss below, one can easily imagine scenarios in which decisions about whether to accept or reject specific packets requires complex reasons that need to combine information from a group of autonomous individuals. See [9] for a similar scenario in the domain of B2B applications. It is the need to deal with these complicated cases that is the reason we are using argumentation. In the remainder of the paper, we will describe how this can be done. In particular, we will use the *metalevel* approach presented in [23] since it provides a general approach to handling anomalies.

3 Metalevel argumentation

The metalevel argumentation framework of [23] is constructed on top of the standard Dung framework [14]. The idea is to make the conditions under which arguments are

classified — for example as justified, rejected and defeated — and the definition of extensions — such as grounded, preferred and stable — expressible in a logical language. The advantage of doing this is that it becomes possible not just to have arguments about objects in some domain, but arguments (meta-arguments) about the status of those arguments. In this section, we introduce enough of this material to apply it to our firewall scenario. The description is an abbreviation of the presentation [23] with some minor modifications and additions (though naturally any faults in the interpretation of the original are ours).

3.1 Argumentation

The formal structure, taken from [23] is as follows. As [23] points out, the formalization is based not on Dung's classic presentation, but on the more recent labelling approach [10, 11, 29, 32] (nicely summarised in [7]). The basic notion is that of a *Dung argumentation framework*, a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$ where $\mathcal{A}$ is a set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is a binary relation on $\mathcal{A}$ that identifies which arguments attack which other arguments.

In Dung's approach, as in [23], arguments are taken to be completely abstract entities with no internal structure, and the attack relation $\mathcal{R}$ is given. As a number of authors have pointed out — for example [4, 20, 26] — it is possible to construct arguments from some logic, and use the relationship between arguments to determine what attacks what. For example, given some logical language $\mathcal{L}$ and an inference relation $\vdash$, we might follow [4] by defining an argument as a pair (H, h) where H is a set of formulae in $\mathcal{L}$, and H is a minimal set such that $H \vdash h$. h is the *conclusion* of such an argument and H is its *support*. In such a formulation, it is typical to say that one argument $H \vdash h$ attacks another $H' \vdash h'$ if $\neg h \in H'$, that is if the conclusion of the attacking argument disagrees with a member of the support of the attacked argument. $\mathcal{A}$ is then the set of all arguments that can be constructed from some set of data Δ, and $\mathcal{R}$ is the set of all attacks between these arguments.

Given a set of arguments and attacks between them, the core of Dung's idea is that not all arguments are equal. It is possible to identify some arguments that we should consider acceptable — their conclusions are valid given what we know — and some that are not. The labelling approach gives us a simple way to determine whether an argument is acceptable or not. The approach can be described in terms of a *labelling function* L which maps from arguments to a set of labels $\{\texttt{IN}, \texttt{OUT}, \texttt{UNDEC}\}$. We then write $in(L)$ to indicate all arguments that are labelled `IN` by L, $out(L)$ to indicate all arguments that are labelled `OUT`, and $undec(L)$ to indicate all arguments that are labelled `UNDEC`.

Defined in this way, there is no relationship between a labelling and the attack relation over a set of arguments. The two are combined through the idea of legality. For a labelling L, an argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$, and an argument $x \in \mathcal{A}$:

1. x is legally `IN` iff x is labelled `IN` and every $y \in \mathcal{A}$ that attacks x is labelled `OUT`.
2. x is legally `OUT` iff x is labelled `OUT` and there is at least one $y \in \mathcal{A}$ that attacks x and is labelled `OUT`.
3. x is legally `UNDEC` iff there is no $y \in \mathcal{A}$ that attacks x such that y is labelled `IN`, and there is at least one $y \in \mathcal{A}$ that attacks x such that y is labelled `UNDEC`.

Note that the UNDEC state occurs when x cannot be labelled IN (because it has at least one attacker that is not OUT), and cannot be labelled OUT (because it has no IN attacker). If an argument is not legally labelled, is is said to be illegally labelled. More precisely, an argument is illegally labelled l, where $l \in \{\texttt{IN}, \texttt{OUT}, \texttt{UNDEC}\}$ if it is not legally labelled l.

With the notion of legality tying labellings to attack relations, it is possible to recover Dung's idea that *extensions*, sets of arguments that are somehow coherent, can be identified within an argumentation framework. We do this through the notions of admissibility and completeness. An *admissible* labelling has no arguments that are illegally IN, and no arguments that are illegally OUT. A *complete* labelling is an admissible labelling that, in addition, has no arguments that are illegally UNDEC. Then, given a complete labelling L, we have that:

1. L is a *grounded* labelling iff there is no complete labelling with a smaller set of IN arguments.
2. L is a *preferred* labelling iff there is no complete labelling with a larger set of IN arguments.
3. L is a *stable* labelling if it contains no UNDEC arguments.

If L is a grounded labelling, then every $x \in L$ is in Dung's grounded extension, if L is a preferred labelling then every $x \in L$ is in the preferred extension, and if L is a stable labelling then every $x \in L$ is in the stable extension.

Based on extension membership we can then define the *status* of arguments. If x is in at least one extension, then x is *credulously* justified, if x is in all extensions, then it is *sceptically* justified, and if x is in no extensions, it is *rejected*.

3.2 Metalevel argumentation

In [23], a *metalevel argumentation framework* is defined[1] as a tuple:

$$\langle \mathcal{A}, \mathcal{R}, \mathcal{A}_M, \mathcal{R}_M, \mathcal{C}, \mathcal{L}_\mathcal{C}, \mathcal{D} \rangle$$

where $\mathcal{A}$ is a set of arguments and $\mathcal{R}$ is an attack relation on object level arguments as in the previous section, and $\mathcal{A}_M$ and $\mathcal{R}_M$ are sets of arguments and attacks at the metalevel. $\mathcal{C}$ is a set of claims about the arguments in $\mathcal{A}_M$, that is a mapping from $\mathcal{A}$ to statements, $\mathcal{L}_\mathcal{C}$ is the language in which the claims are made, and $\mathcal{D}$ is a set of constraints on the attack relation $\mathcal{A}_M$ that are determined by the claims. As an example, [23] gives a metalevel argumentation framework that captures Dung's original argumentation system. In this system, $\mathcal{L}_\mathcal{C}$ includes a set of constants and a set of predicates. The set of constants C includes $\ulcorner x \urcorner$ for every $x \in \mathcal{A}$ (it is common practice to quote object level symbols in this way to make them constants at the metalevel), and the set of predicates is:

$$\{\mathit{justified}, \mathit{defeat}, \mathit{rejected}\}$$

and has a set of well formed formulae W defined by the following rules:

[1] This is a less general subset of the system presented in [23], but sufficient for our purposes.

1. If $x \in C$, then $x \in W$
2. If $x, y \in W$, then $(x, y) \in W_{\mathcal{R}}$, $W_{\mathcal{R}} \subset W$
3. If $x \in W$ and $x \notin W_{\mathcal{R}}$, then $\mathit{justified}(x) \in W$
4. If $x \in W$ and $x \notin W_{\mathcal{R}}$, then $\mathit{rejected}(x) \in W$
5. If $x, y \in W$ and $x, y \notin W_{\mathcal{R}}$, then $\mathit{defeat}(x, y) \in W$

In other words, the language $\mathcal{L}_{\mathcal{C}}$ allows us to talk about any of the constants (which will represent arguments in $\mathcal{A}$), attacks between the arguments, whether arguments are justified or rejected, and whether one argument *defeats* another. The notion of defeat is necessary because exactly the kind of thing we want to capture is when there is an attack between two arguments, but there is something at the metalevel which overrides the attack. The labelling of arguments thus depends on defeats not on attacks.

We next need to define $\mathcal{A}_M$, which is the union of $\mathcal{A}_{M1}$, $\mathcal{A}_{M2}$ and $\mathcal{A}_{M3}$ where:

$$\alpha \in \mathcal{A}_{M1}, \mathcal{C}(\alpha) = \mathit{justified}(\ulcorner x \urcorner) \text{ iff } x \in \mathcal{A}$$
$$\beta \in \mathcal{A}_{M2}, \mathcal{C}(\beta) = \mathit{rejected}(\ulcorner x \urcorner) \text{ iff } x \in \mathcal{A}$$
$$\gamma \in \mathcal{A}_{M3}, \mathcal{C}(\gamma) = \mathit{defeat}(\ulcorner x \urcorner, \ulcorner y \urcorner) \text{ iff } (x, y) \in \mathcal{R}$$

so that arguments in $\mathcal{A}_M$ are statements about arguments in $\mathcal{A}$ being justified, rejected and defeating one another. Then the set of constraints on claims, $\mathcal{D}$ contains:

D1 if $\mathcal{C}(\alpha) = \mathit{defeat}(X, Y)$ and $\mathcal{C}(\beta) = \mathit{justified}(Y)$ then $(\alpha, \beta) \in \mathcal{R}_M$.
D2 if $\mathcal{C}(\alpha) = \mathit{defeat}(X, Y)$ and $\mathcal{C}(\beta) = \mathit{rejected}(X)$ then $(\beta, \alpha) \in \mathcal{R}_M$.
D3 if $\mathcal{C}(\alpha) = \mathit{justified}(X)$ and $\mathcal{C}(\beta) = \mathit{rejected}(X)$ then $(\alpha, \beta) \in \mathcal{R}_M$.

which together define the contents of $\mathcal{R}_M$. For example, the first of these says that a claim that X defeats Y is an attack on the claim that Y is justified. As [23] shows, computing the justified arguments in $\mathcal{A}_M$ will identify the justified arguments in $\mathcal{A}$ consistently across the different definitions of extensions.

As presented so far, and as described in [23], this metalevel argumentation system, just like Dung's system [14], has an asbtract notion of an argument. The members of $\mathcal{A}_M$ have no internal structure. However, one can (and we will below) construct the members of $\mathcal{A}_M$ from a set of statements Δ_M in some language $\mathcal{L}_M$ using an inference mechanism $\vdash_M$. When this is done, $\mathcal{L}_M$, like $\mathcal{L}_{\mathcal{C}}$, will contain constants $\ulcorner x \urcorner$ for every $x \in \mathcal{A}$ since $\mathcal{L}_M$ will be statements about these arguments. For example, a sentence in $\mathcal{L}_M$ might describe how one argument is preferred to another, and an argument in $\mathcal{A}_M$ that is constructed from such statements might describe how an attack from $\mathcal{R}$ is not a defeat because of this preference. The attacks between these arguments then populate $\mathcal{R}_M$.

4 Arguing about firewall policies

Having introduced some of the issues in firewall configuration, and the metalevel argumentation approach of [23], in this section we discuss how the latter can be used to model some aspects of firewall configuration in order to illuminate anomalies and potentially provide a means to support system administrators in solving them.

4.1 Scenario

We consider a simple scenario in which an organization operates a hierarchical network of routers. The root node, R, is the master router which ultimately implements the firewall policy for the organization. The child nodes, R_1 and R_2, are gateways to different departments within the organization which have different requirements. R_1 and R_2 are stakeholders in the implemented policy and send their preferred policies to R. R then combines these policies to create the overall policy for the organization. If the policies put forward by R_1 and R_2 conflict, R must resolve the conflict in order to create this overall policy.

Currently this merging of policies would be done by hand. In the simplest case, this is done by just concatenating the firewall rules. It is not hard, though, to imagine the process being automated with the routers being under the control of software agents. A_R is the agent controlling R and A_{R_1} and A_{R_2} are the agents controlling R_1 and R_2 respectively. Indeed, a process that implements a software-based packet filtering firewall would meet the description of a basic reactive agent [33], receiving a sequence of percepts in the form of data about incoming packets, and making a sequence of "accept"/"block" decisions. The policies for R_1 and R_2 are set by system administrators in the relevant departments, A_{R_1} and A_{R_2} advocate for their policies with A_R, this agent merges the policies and the combined policy is set by the system administrator with overall responsibility for the whole organization, based on information provided by the agent controlling R.

4.2 A first metalevel argumentation model

Now suppose that R_1 has a policy to deny all DNS traffic in order to enhance system security, while R_2 has a policy to allow HTTP traffic in order to support web services. We can model R_1's policy as:

$$\begin{aligned} &\mathit{secure\_system} \\ &\mathit{secure\_system} \Rightarrow \neg \mathit{allow\_DNS} \\ &\neg \mathit{allow\_DNS} \Rightarrow \neg \mathit{allow\_UDP} \end{aligned}$$

using a simple logical language[2]. In addition to the justification of R_1's policy this captures the fact that disallowing DNS traffic is achieved by blocking UDP traffic. From this set of policy information, it is possible for A_{R_1} to construct the argument:

$$\begin{aligned} &(\{\mathit{secure\_system}, \mathit{secure\_system} \Rightarrow \neg \mathit{allow\_DNS}, \\ &\quad \neg \mathit{allow\_DNS} \Rightarrow \neg \mathit{allow\_UDP}\}, \neg \mathit{allow\_UDP}) \end{aligned} \tag{1}$$

[2] See [28] for a full description, but in short the language allows for default rules with conjunctions of formulae as antecedents and single formulae as consequents. The default implication is denoted here by $\Rightarrow$, and inference is by generalised modus ponens which combines $\alpha_1 \wedge \ldots \alpha_n \Rightarrow \beta$ with $\alpha_1, \ldots, \alpha_n$ to give β. Generlized modus ponens is the only inference rule.

which has the conclusion to block UDP traffic. We will call this argument n since it concerns name resolution. Similarly, R_2's policy can be modelled as:

$$allow\_WS$$
$$allow\_WS \Rightarrow allow\_TCP$$

giving A_{R_2} the argument:

$$(\{allow\_WS, allow\_WS \Rightarrow allow\_TCP\}, allow\_TCP)$$

with the conclusion that TCP traffic should be allowed. We will call this argument t.

We can imagine that A_R engages both A_{R_1} and A_{R_2} in an inquiry dialogue to discover their requirements (for example following the protocol in [25]), a process that results in both n and t (including all the information on which they are based) being passed to A_R. In addition, A_R knows that:

$$allow\_WS \Rightarrow allow\_DNS$$
$$allow\_DNS \Rightarrow allow\_UDP$$

since web services require name resolution and hence require UDP. In addition to n and t, A_R can thus construct an argument w (an argument about the requirements of web services):

$$(\{allow\_WS, allow\_WS \Rightarrow allow\_DNS, allow\_DNS \Rightarrow allow\_UDP\}, allow\_UDP)$$

Clearly w and n attack one another[3]. In the formulation given above, we then have:

$$\mathcal{A} = \{w, n, t\},$$
$$\mathcal{R} = \{(w, n), (n, w)\}$$

which has a single grounded extension $\{t\}$ which does not specify what to do about UDP traffic. Before we consider how we can use argumentation to represent different solutions to this scenario, let's work through the full metalevel formulation. The previous section gave a metalevel formulation of a standard Dung framework. In this, the set of metalevel arguments includes statements about the justification and defeat of every argument in $\mathcal{A}$, and statements about defeat for every attack in $\mathcal{R}$. Thus we have:

$$\begin{aligned}\mathcal{A}_M = \{&defeat(\ulcorner w \urcorner, \ulcorner n \urcorner), defeat(\ulcorner n \urcorner, \ulcorner w \urcorner),\\ &justified(\ulcorner w \urcorner), justified(\ulcorner n \urcorner), justified(\ulcorner t \urcorner),\\ &rejected(\ulcorner w \urcorner), rejected(\ulcorner n \urcorner), rejected(\ulcorner t \urcorner)\}\end{aligned}$$

The constraints on claims then result in the following set of attacks:

$$\begin{aligned}\mathcal{R}_M = \{&(defeat(\ulcorner n \urcorner, \ulcorner w \urcorner), justified(\ulcorner w \urcorner)), (justified(\ulcorner w \urcorner), rejected(\ulcorner w \urcorner)),\\ &(rejected(\ulcorner w \urcorner), defeat(\ulcorner w \urcorner, \ulcorner n \urcorner)), (defeat(\ulcorner w \urcorner, \ulcorner n \urcorner), justified(\ulcorner n \urcorner)),\\ &(justified(\ulcorner n \urcorner), rejected(\ulcorner n \urcorner)), (rejected(\ulcorner n \urcorner), (defeat(\ulcorner n \urcorner, \ulcorner w \urcorner)),\\ &(justified(\ulcorner t \urcorner), rejected(\ulcorner t \urcorner))\}\end{aligned}$$

[3] The form of attack here is a *rebut*, an attack between the conclusions of arguments. While rebuts can be problematic in some argumentation systems [12], they do not cause problems when arguments are, as here, chains of defeasible rules.

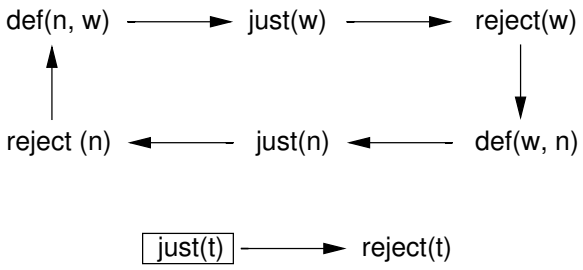

Fig. 1. The metalevel argument graph for the basic firewall example. Each argument corresponds to an argument or an attack at the object level. A box is drawn around arguments that can legally be labelled IN

The first six of these attack relations form a cycle in the argument graph shown in Figure 1, (this is the weather example from [23, page 19] without the preference argument), which has no consistent labelling. The last pair of arguments listed in $\mathcal{R}_M$ can be labelled consistently so that t is justified. The result, then, is the single stable extension:

$$E = \{justified(\ulcorner t \urcorner)\}$$

and the corresponding single stable extension of the object level (as we already identified just considering the object level system) is $\{t\}$.

We see the advantage of A_r using a metalevel framework, rather than just a standard Dung framework which would enable it to reach the same conclusion, that A_R can use the metalevel framework to explain the the resulting policy to the adminstrator with overall responsibility for the organization. The argument graph in Figure 1 makes it clear that the fact there is no justified argument for w or n is the symmetry between them. Each attacks the other, and there is no reason to privilege one attack over the other.

4.3 Metalevel argumentation using preferences

While the resolution of the conflicting policies achieved above is correct from the perspective of argumentation theory, it is not very satisfying from an application point of view — the resulting policy is unhelpful since it provides no decision on UDP traffic. A natural way to improve the situation is to express some kind of preference between web services and security (in our case) to resolve the conflict between w and n one way or the other. This is not a new idea, having been introduced at the object level in argumentation systems such as [3, 27].

As discussed in [23], preferences can easily be introduced into a metalevel argumentation framework. If we follow the approach described in [23], we consider that stating a preference w $\gg_P$ n — that the argument in favor of allowing UDP to support web services is strictly preferred to the argument in favor of blocking UDP to enhance security — is equivalent to stating a metalevel argument that n does not defeat w.

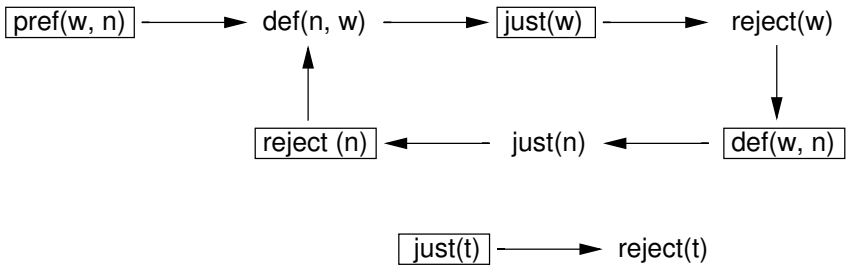

Fig. 2. The metalevel argument graph for the firewall example with preferences. Each argument corresponds to an argument or an attack at the object level. A box is drawn around arguments that can legally be labelled IN

The formal description of the metalevel system is that of the previous section, with the same set of arguments and attacks at the object level:

$$\mathcal{A} = \{w, n, t\},$$
$$\mathcal{R} = \{(w, n), (n, w)\}$$

but with an additional argument — the argument that w is preferred to n at the metalevel (the claim language $\mathcal{L}_C$ contains an additional predicate to express this preference:

$$\begin{aligned}\mathcal{A}_M = \{&\mathit{defeat}(\ulcorner w \urcorner, \ulcorner n \urcorner), \mathit{defeat}(\ulcorner n \urcorner, \ulcorner w \urcorner),\\ &\mathit{justified}(\ulcorner w \urcorner), \mathit{justified}(\ulcorner n \urcorner), \mathit{justified}(\ulcorner t \urcorner),\\ &\mathit{rejected}(\ulcorner w \urcorner), \mathit{rejected}(\ulcorner n \urcorner), \mathit{rejected}(\ulcorner t \urcorner),\\ &\mathit{preferred}(\ulcorner w \urcorner, \ulcorner n \urcorner)\},\end{aligned}$$

The set of metalevel attacks also has an additional member, the attack of $\mathit{preferred}(\ulcorner w \urcorner, \ulcorner n \urcorner)$ on $\mathit{defeat}(\ulcorner n \urcorner, \ulcorner w \urcorner)$

$$\begin{aligned}\mathcal{R}_M = \{&(\mathit{defeat}(\ulcorner n \urcorner, \ulcorner w \urcorner), \mathit{justified}(\ulcorner w \urcorner)), (\mathit{justified}(\ulcorner w \urcorner), \mathit{rejected}(\ulcorner w \urcorner)),\\ &(\mathit{rejected}(\ulcorner w \urcorner), \mathit{defeat}(\ulcorner w \urcorner, \ulcorner n \urcorner)), (\mathit{defeat}(\ulcorner w \urcorner, \ulcorner n \urcorner), \mathit{justified}(\ulcorner n \urcorner)),\\ &(\mathit{justified}(\ulcorner n \urcorner), \mathit{rejected}(\ulcorner n \urcorner)), (\mathit{rejected}(\ulcorner n \urcorner), (\mathit{defeat}(\ulcorner n \urcorner, \ulcorner w \urcorner)),\\ &(\mathit{preferred}(\ulcorner w \urcorner, \ulcorner n \urcorner), \mathit{defeat}(\ulcorner n \urcorner, \ulcorner w \urcorner)),\\ &(\mathit{justified}(\ulcorner t \urcorner), \mathit{rejected}(\ulcorner t \urcorner))\}\end{aligned}$$

As Figure 2 shows, this additional attack now breaks the cycle (this section of the argument graph is now exactly the weather example from [23, page 19]), and we have a single stable extension at the metalevel:

$$\begin{aligned}E = \{&\mathit{justified}(\ulcorner w \urcorner), \mathit{defeat}(\ulcorner w \urcorner, \ulcorner n \urcorner),\\ &\mathit{rejected}(\ulcorner n \urcorner), (\mathit{preferred}(\ulcorner w \urcorner, \ulcorner n \urcorner), \mathit{justified}(\ulcorner t \urcorner)\}\end{aligned}$$

with the corresponding object level extension $\{w, t\}$, with a policy that allows TCP and UDP.

Again, we believe that this metalevel structure provides a means to explain the outcome to the adminstrator. Comparing Figure 2 and Figure 1, it is clear that the preference for w over n "fixes" the cycle of arguments so that $defeat(\ulcorner n \urcorner, \ulcorner w \urcorner)$ does not hold, resulting in w being justified.

4.4 Structured metalevel argumentation

As noted above, the idea of using preferences to resolve conflicting arguments is not new. What the metalevel approach brings that is new is the ability to clearly see what the preferences are doing, that is *how* they resolve the conflict. Examining the metalevel arguments and attacks, it is clear that the preferences resolve the conflict by defeating $defeat(n, w)$, and in turn preventing that argument from making w unjustified. In applications such as ours, where the justification for using argumentation is to be able to explain to users the structure of the problem and how to reason about it, this ability to use the metalevel system to explain how arguments are resolved at the object level is a powerful feature.

Furthermore, metalevel argumentation can capture more than just the application of preferences. Appendix A, for example, shows how we can use the same metalevel argumentation framework to apply ideas from value-based argumentation [8] to capture a situation in which different parties have different views about which policy to adopt.

However, even value-based argumentation doesn't fully exploit the power of the metalevel framework. The astute reader will have spotted that though we have described how structured arguments can be used at the object level to connect firewall rules to arguments — as in the argument labelled (1) for example — we have yet to explore the construction of arguments at the metalevel. Allowing reasoning at this level allows us to construct arbitrary arguments at the metalevel that can resolve conflicts at the object level.

Consider, as an example, this variation on the use of preferences, where A_R has the following information in its Δ_M:

$$prefer(promote\_WS, be\_secure)$$
$$achieves(\ulcorner w \urcorner, promote\_WS)$$
$$achieves(\ulcorner n \urcorner, be\_secure)$$
$$prefer(X,Y) \wedge achieves(Z,X) \wedge achieves(W,Y) \Rightarrow preferred(Z,W)$$
$$preferred(Z,W) \wedge (W,Z) \Rightarrow (preferred(Z,W), defeat(W,Z))$$

where X, Y, Z and W are variables, $\wedge$ is conjunction, and *achieves* is a predicate that captures the relationship between an object level argument and a metalevel proposition. This is metalevel information about a preference for arguments about firewall policies that promote web services over security, about the specific policies supported by the object level arguments w and n, about how to combine preferences and information about arguments in general (if you prefer one policy to another, then you prefer the argument that supports it), and about how preference relates to defeat.

From this information, it is clear that A_R can construct an argument for

$$(preferred(\ulcorner w \urcorner, \ulcorner n \urcorner), defeat(\ulcorner n \urcorner, \ulcorner w \urcorner))$$

which of course is the crucial piece in the use of preferences to resolve the circle of attacks in the metalevel representation of the conflict between w and n (see Figure 2 again). Abstracting from this, we have a general mechanism by which we can program A_R to figure out how to resolve conflicts in object level arguments — we provide it with knowledge in Δ_M from which it can construct metalevel arguments about which attacks are themselves defeated.

One particularly interesting kind of reasoning that one might perform at this level is reasoning about trust. A_R might wish to resolve the conflict between w and n based on what it knows about the trustworthiness of A_{R_1} and A_{R_2}. In [24], we described an argumentation system that could be used to infer the degree of trust between agents, and how this derived information could be combined with beliefs from those agents. Such reasoning could be employed in the metalevel argumentation framework to identify attacks on $defeat(\ulcorner n \urcorner, \ulcorner w \urcorner)$ or $defeat(\ulcorner w \urcorner, \ulcorner n \urcorner)$. For example:

$$
\begin{aligned}
& trust(self, A_{R_1}) \\
& \neg trust(self, A_{R_2}) \\
trust(self, X) \wedge \neg trust(self, Y) & \Rightarrow more\_trustworthy(X, Y) \\
& source(\ulcorner n \urcorner, A_{R_1}) \\
& source(\ulcorner w \urcorner, A_{R_2}) \\
source(X, Y) \wedge source(W, Z) \wedge more\_trustworthy(Y, Z) & \Rightarrow preferred(X, W)
\end{aligned}
$$

from which A_R could, using elements from the previous example, construct an argument for:

$$(preferred(\ulcorner n \urcorner, \ulcorner w \urcorner), defeat(\ulcorner w \urcorner, \ulcorner n \urcorner))$$

which would provide an alternative way to resolve the object level conflict between w and n — the fact that n is provided by a more trustworthy source than w is an argument against w defeating n.

4.5 Discussion

The examples in the previous section have demonstrated how metalevel argumentation can be used to represent firewall policies in such a way that a conflict in the rules is exposed at the metalevel, and how different approaches to metalevel reasoning can be used to resolve this conflict. The key advantage that we see to using the metalevel approach is the fact that it makes clear where conflicts arise, and how they are resolved (when they are resolved). In firewall configuration, the final decision about how to set the firewall up is going to be taken by a human system adminstrator — someone who is not an expert in argumentation — and so any additional clarity will make their job much easier. Of course it remains to be seen how much, if at all, the metalevel approach helps in making decisions about resolving firewall anomalies, but we aim to address this in future work through human subject experiments.

We also note that we started by discussing four kinds of anomaly in firewall rules — shadowing, correlation, redundancy and generalization. Our examples are all instances of shadowing. Showing that metalevel argumentation can help in general with detecting

and resolving firewall anomalies will require that we also demonstrate that the approach can deal with the other forms of anomaly. This, again, is work we aim to carry out in the future.

5 Related Work

Given the importance of security, the central role that firewalls play in ensuring security, and the complexity of configuring firewalls, there has been considerable work on approaches to support this configuration. [1] implemented a set of algorithm in "Firewall Policy Advisor", a user-friendly tool. These use a firewall policy tree to deal with centralized and distributed firewalls [2]. [34] introduced FIREMAN, a static analysis toolkit to check anomalies in individual firewalls as well as among distributed firewalls.

Similar work based on a logic background has been done. [16] proposed a formal logic in understanding the actual meaning of the firewall rules. Logic may be used to prove the properties and detect a number of anomalies within the rule sets. [18, 17] used ordered binary decision diagrams (BDD's) to represent a rule set as a boolean expression to analyze the rule sets. However, the system does not allow the definition of rules using a logic programming syntax. [15] presented a tool based on constraint logic programming (CLP) for analyzing firewall rules. They implemented it using Eclipse CLP, which makes it easy to express and extend the knowledge base of the system. [5] described a technique based on Argumentation for Logic Programming with Priorities (LPP). This allows administrators using high-level abstractions specifying their network security requirements. In [6], they extend their previous work to automatically generate firewall policies from higher-level requirements.

More work associate with firewall configuration has been published. [13] developed harnessing models for policy conflict analysis, taking into account the semantic information. [35] proposed a policy algebra framework for security policy enforcement in hybrid firewalls, allowing the basic algebra used into the rule sets, such as addition, conjunction, subtraction, summation. [22] presented a firewall analysis engine named Fang, based on a combination of a graph algorithm and a rule-base simulator.

Our use of metalevel argumentation sets our work apart from all of the work cited. All approaches to reasoning about firewall rules yet published have concentrated on object level reasoning. Our work is the first we are aware of to look at reasoning about firewall rules at the metalevel. However, as we start to look at bringing in reasoning about trust at the metalevel, we are clearly beginning to overlap with the work of Villata *et al.* [30, 31] who have written quite extensively about how to represent and resolve arguments that attack arguments about the trustworthiness of agents. If this kind of reasoning were to be incorporated into our framework, it would require a second level of metareasoning — the first metalevel would be used to make statements about the trustworthiness of arguments and the effect of such statements on object-level defeats, and the second metalevel would make statements attacking these statements of trustworthiness. Resolving arguments at the second level would then inform which statements hold at first level, and hence what arguments were preferred at the object level,

6 Conclusions

This paper has discussed the application of metalevel argumentation to the problem of modelling firewall configurations. In particular, we have shown how to use the metalevel argumentation system of [23] such that object level arguments are concerned with which packets to accept and block in a firewall, and the metalevel arguments identify — and potentially resolve — conflicts between these object level arguments due to shadowing anomalies in the firewall rules. We have argued that since a human system administrator will ultimately have to set the firewall policy, the use of a metalevel formalism — which makes explicit the (metalevel) arguments that explain why conflicts in rules arise and how they may be resolved — is appropriate. However, our planned future work to model anomalies other than shadowing will be required to tell if metalevel argumentation is sufficient to capture all forms of firewall anomaly, and human subject experiments will be required to test our hypothesis that the metalevel arguments aid human understanding and decision making.

Acknowledgements

Research was partially funded by the National Science Foundation, under grant CNS 1117761 and Army Research Laboratory and Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, the National Science Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Thanks to Sanjay Modgil for helpful comments on the paper.

Appendix: Metalevel argumentation with values

In a *Value based Argumentation Framework*, we are concerned with values promoted by each attack and their relative strengths. This is necessarily subjective and audience dependent [8, 23] and provide a way for R to reason about implementing policies specific to R_1 and R_2. To see how this might work, consider a variation on the example discussed above, where the network is that of a research university[4]. R_1 belongs to a research computing facility (IT) which uses BitTorrent (BT) to provide updates to the host machines. This is deemed to be mission-critical by IT. R_2 belongs to the Chancellor's office (CO), which has deemed BitTorrent to be a legal liability and thus seeks to deny any BT traffic.

Let c denote the policy "Deny BitTorrent" and a denote the policy "Allow BitTorrent". As in the examples in Section 4, we have a situation in which there are mutual attacks between the arguments. In a value-based framework, we can annotate the policies to introduce values associated with the arguments by different parties. For example,

[4] This is, of course, a fictional university and bears no ressemblance to any institution at which the authors may have worked.

"Deny BitTorrent" may be associated with the "Preventing Piracy" (p) value, while "Allow BitTorrent" may be associated with the "Allowing Mission Critical Services" (m) value. R may use reasoning specific to audiences pertinent to R_1 and R_2. If R_1 serves research related computing facilities, the relevant audience is the advocates for the corresponding policy, IT (r_1). And if R_2 is controlling access to student dormitories, then the relevant audience is the advocates for the corresponding policy, CO (r_2). Further, each audience indicates which values are more important: r_1 prefers p to m, while r_2 prefers m to p.

We can formulate this as a value-based framework (VAF) in the language of metalevel argumentation as (following [23]):

$$(\mathcal{A} = \{c, a\}, \mathcal{R} = \{(c, a), (a, c)\},$$
$$\mathcal{V} = \{p, m\}, \{val(c) = p, val(a) = m\}, \mathcal{P} = \{r_1 = \{(m, p)\}, r_2 = \{(p, m)\}\})$$

which leads to two audience-specific VAFs (aVAFs) for r_1 and r_2. Our set of claims then includes the values, $val(c) = p$ and $val(a) = m$, and the preferences over values, $preferred_{r_1}(m, p)$ and $preferred_{r_2}(p, m)$.

We can then formulate the audience specific metalevel argumentation framework for r_1 as follows.

$$\begin{aligned}
\mathcal{A} &= \{ c, a\}, \\
\mathcal{R} &= \{ (c, a), (c, a)\}, \\
\mathcal{A}_M &= \{ preferred_{r_1}(\ulcorner m \urcorner, \ulcorner p \urcorner), defeat(\ulcorner m \urcorner, \ulcorner p \urcorner), defeat(\ulcorner p \urcorner, \ulcorner m \urcorner), \\
&\quad\ justified(\ulcorner m \urcorner), justified(\ulcorner p \urcorner), rejected(\ulcorner m \urcorner), rejected(\ulcorner p \urcorner)\}, \\
\mathcal{R}_M &= \{ (preferred_{r_1}(\ulcorner m \urcorner, \ulcorner p \urcorner), defeat(\ulcorner p \urcorner, \ulcorner m \urcorner)), \\
&\quad\ (defeat(\ulcorner p \urcorner, \ulcorner m \urcorner), justified(\ulcorner m \urcorner)), \\
&\quad\ (justified(\ulcorner m \urcorner), rejected(\ulcorner m \urcorner)), \\
&\quad\ (rejected(\ulcorner m \urcorner), defeat(\ulcorner m \urcorner, \ulcorner p \urcorner)), \\
&\quad\ (defeat(\ulcorner m \urcorner, \ulcorner p \urcorner), justified(\ulcorner m \urcorner)), \\
&\quad\ (justified(\ulcorner m \urcorner), rejected(\ulcorner m \urcorner)), \\
&\quad\ (rejected(\ulcorner m \urcorner), (defeat(\ulcorner p \urcorner, \ulcorner m \urcorner))\}.
\end{aligned}$$

and a corresponding metalevel argumentation framework can be formulated for r_2. This leads to two audience specific frameworks for r_1 and r_2. The preferred extensions, for each audience, are:

$$E_{r_1} = \{preferred_{r_1}(a_m, c_p), justified(a), defeat(a, c), rejected(c)\}$$

and

$$E_{r_2} = \{preferred_{r_2}(c_p, a_m), justified(c), defeat(c, a), rejected(a)\}.$$

With the audience-specific extensions, the system administrator may reason that the CO prefers values promoted by preventing piracy over allowing mission critical services, while the IT prefers values promoted by allowing mission critical services over preventing piracy.

References

1. E.S. Al-Shaer and H.H. Hamed. Firewall policy advisor for anomaly discovery and rule editing. In *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, pages 17–30. IEEE, 2003.
2. E.S. Al-Shaer and H.H. Hamed. Discovery of policy anomalies in distributed firewalls. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2605–2616. IEEE, 2004.
3. 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.
4. 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.
5. A. Bandara, A. Kakas, E. Lupu, and A. Russo. Using argumentation logic for firewall policy specification and analysis. *Large Scale Management of Distributed Systems*, pages 185–196, 2006.
6. A.K. Bandara, A.C. Kakas, E.C. Lupu, and A. Russo. Using argumentation logic for firewall configuration management. In *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on*, pages 180–187. IEEE, 2009.
7. P. Baroni, M. Caminada, and M. Giacomin. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 2011.
8. T. J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–428, 2003.
9. J. Bentahar, R. Alam, Z. Maamar, and N. C. Narendra. Using argumentation to model and deploy agent-based B2B applications. *Knowledge-Based Systems*, 23(7):677–692, 2010.
10. M. W. A. Caminada. On the issue of reinstatement in argumentation. In *Proceedings of the 10th European Conference on Logic in Artificial Intelligence*, pages 111–123, Liverpool, UK, 2006.
11. M. W. A. Caminada. An algorithm for computing semi-stable semantics. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 222–234, Verona, Italy, 2007.
12. M. W. A. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171(5–6):286–310, 2007.
13. S. Davy and B. Jennings. Harnessing models for policy conflict analysis. *Inter-Domain Management*, pages 176–179, 2007.
14. 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.
15. P. Eronen and J. Zitting. An expert system for analyzing firewall rules. In *Proceedings of the 6th Nordic Workshop on Secure IT Systems (NordSec 2001)*, pages 100–107, 2001.
16. J. Govaerts, A. Bandara, and K. Curran. A formal logic approach to firewall packet filtering analysis and generation. *Artificial Intelligence Review*, 29(3):223–248, 2008.
17. S. Hazelhurst. Algorithms for analysing firewall and router access lists. Technical report, Department of Computer Science, University of the Witwatersrand, 2000.
18. S. Hazelhurst, A. Fatti, and A. Henwood. Binary decision diagram representations of firewall and router access lists. Technical report, Department of Computer Science, University of the Witwatersrand, 1998.
19. K. Ingham and S. Forrest. Network firewalls. *Enhancing computer security with smart technology*, pages 9–40, 2006.
20. 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.

21. J.F. Kurose and K.W. Ross. *Computer networking: a top-down approach*. Addison-Wesley, 2010.
22. A. Mayer, A. Wool, and E. Ziskind. Fang: A firewall analysis engine. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 177–187. IEEE, 2000.
23. S. Modgil and T. J. M. Bench-Capon. Metalevel argumentation. *Journal of Logic and Computation*, 6(21):959–1003, 2011.
24. S. Parsons, E. Sklar, and P. McBurney. Using argumentation to reason with and about trust. In *Proceedings of the 8th International Workshop on Argumentation in Multiagent Systems*, Taipei, Taiwan, 2011.
25. 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.
26. H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1:93–124, 2010.
27. H. Prakken and G. Sartor. Argument-based logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 1997.
28. Y. Tang, K. Cai, P. McBurney, E. Sklar, and S. Parsons. Using argumentation to reason about trust and belief. *Journal of Logic and Computation*, (to appear), 2011.
29. B. Verheij. A labeling approach to the computation of credulous acceptance in argumentation. In *Proceedings of the 20th International Joint Conference on Aritificial Intelligence*, pages 623–628, Hyderabad, India, 2007.
30. S. Villata, G. Boella, D. M. Gabbay, and L. van der Torre. Arguing about trust in multiagent systems. In *Proceedings of the 11th Symposium on Artificial Intelligence of the Italian Association for Artificial Intelligence*, Brescia, Italy, 2010.
31. S. Villata, G. Boella, D. M. Gabbay, and L. van der Torre. Arguing about the trustworthiness of the information sources. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, Belfast, UK, 2011.
32. G. Vreeswijk. An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *Proceedings of the 1st International Conference on Computational Models of Argument*, pages 109–120, Liverpool, UK, 2006.
33. M. Wooldridge. *An Introduction to Multiagent Systems*. Wiley, 2nd edition, 2009.
34. L. Yuan, H. Chen, J. Mai, C.N. Chuah, Z. Su, and P. Mohapatra. Fireman: A toolkit for firewall modeling and analysis. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.
35. H. Zhao and S.M. Bellovin. Policy algebras for hybrid firewalls. In *Annual Conference of ITA (ACITA)*, volume 2007, 2007.