An Axiomatic Approach to Firewall Rule Update

Richard Booth and Wijittra Noisanguan
Mahasarakham University
Faculty of Informatics
Mahasarakham 44150, Thailand
{richard.b,wijittra.n}@msu.ac.th

***Abstract*—*A firewall administrator often needs to* update *a given firewall, to take account of evolving security requirements or to correct a wrongly classified packet. We look at the problem of firewall update, mainly from an* axiomatic *viewpoint. An axiom for firewall update is a property which any reasonable method of updating should fulfill. We propose a number of such axioms. We also give some simple examples of firewall update algorithms and check them against the axioms.*

***Index Terms*—Firewalls, firewall rule update, network security**

I. Introduction

Firewalls are major elements in *network security policies* [1]. A firewall *implements* the security policy via a sequence of rules which controls incoming and outgoing traffic passing though it. Many studies on firewalls have been focused on the question "*What makes a "good" firewall?*". For example a good firewall is usually considered to be one which is *complete*, and which reaches *fast* decisions over whether to accept or deny the packets passing through. These studies usually take a *static* viewpoint, in that the security policy is fixed and does not change.

However, since the packets flowing in the internet have various network characteristics [2], the studies must also investigate the *dynamic* viewpoint. *Dynamics* is very important. A firewall administrator (FA) must sometimes change or update a firewall to allow or deny certain packets. Consider for example the mini firewall in Table I. Suppose the FA finds out source ip address 102.197.137.5 is a hacker. Then the firewall needs to be changed so it denies all packets from this address. We need some method of *automatic firewall update*. Note firewalls may be very large in practice, consisting of thousands of rules.

There are two different approaches we can take with this problem. The first is a *bottom-up* approach, in which we directly specify algorithms for firewall update. The second is a *top-down* approach in which we write down some requirements, or *axioms*, which we expect any *useful* update method to have. The benefit of the top-down approach is that the axioms help to focus direction on what we really expect from a good update method, and provide a *checklist* against which future-designed update algorithms can be tested. This axiomatic approach has been fruitfully applied in other fields, especially in the general field of *belief revision* [3], which is concerned with how a knowledge base should be revised to include new knowledge. In this paper we will use the top-down approach, giving an axiomatic approach to firewall update.

The plan of this paper is as follows. In Section II we review basic firewall notations. Then in Section III we focus on some commonly accepted static properties of firewalls, which we introduce as *static axioms*. This leads to our defined notion of an $\mathcal{X}$*-optimised* firewall, which is just a firewall which satisfies all the axioms in the group of static axioms $\mathcal{X}$. In Section IV we turn to firewall dynamics. We formally define *firewall updater* and state some desirable properties for such updaters via a number of *axioms for firewall update.* We give some example firewall updaters and check them against the axioms. We will see all of these example updaters fail to satisfy at least one axiom. In particular none are guaranteed to output an $\mathcal{X}$-optimised firewall. As a potential remedy to this we look in Section V at the idea of $\mathcal{X}$*-optimised updaters*, which build-in a post-processing optimisation step to the update process. We will see that this approach too will not satisfy the axioms in all cases. Related work is mentioned in Section VI before we conclude in Section VII

II. Firewall Basics

We follow roughly the same notation as Liu et al (see, e.g., [4]). We assume a number of network *fields* $F_1, \dots, F_d$ (e.g., Source ip, Destination port, etc). Each field has a *domain* $D(F_j)$, which we assume is a finite interval of non-negative integers. (This representation is different from that used in "real-life" firewalls, which normally use the "prefix" notation as in Table I. It is more convenient for describing examples. However this interval notation is interconvertible with the prefix notation. For e.g., the domain of Source ip can be written as $[0, 2^{32} - 1]$.) A *packet* $p = (p_1, \dots, p_d)$ is a d-tuple such that $p_j \in D(F_j)$ for each $j = 1, \dots, d$.

A *firewall rule* r_i takes the form $P_i \to a_i$ where P_i and a_i are resp. the *predicate* and *action* of the rule. Each predicate takes the form $(S_1 \wedge \dots \wedge S_d)$, where $S_j \subseteq D(F_j)$ for $j = 1, \dots, d$, and $a_i \in \{\texttt{accept}, \texttt{deny}\}$. (Note other actions are possible in practice, e.g, accept with logging etc, but we will stay with just two possible actions for simplicity.) We say a packet $p = (p_1, \dots, p_d)$ *matches* a predicate $P = (S_1 \wedge \dots \wedge S_d)$ if and only if (iff) $p_j \in S_j$ for $j = 1, \dots, d$, and we say p *matches the rule* $r = P \to a$ whenever p matches the predicate P of r. Two predicates P_1 and P_2 *overlap* iff there is some packet which matches both of them, otherwise they are *disjoint.* Two rules $r_1 = P_1 \to a_1$ and $r_2 = P_2 \to a_2$ *conflict* iff *(i)* their predicates overlap and *(ii)* $a_1 \neq a_2$. In other words they lead to different actions for some packets.

Rule	Source ip address	Destination ip address	Source port	Destination port	Protocol	Action
r_1	$*.*.*.*$	116.197.137.$*$	$*$	80	TCP	`accept`
r_2	102.197.137.5	$*.*.*.*$	$*$	25	TCP	`accept`
r_3	102.198.226.$*$	$*.*.*.*$	$*$	80	TCP	`deny`

Table I
EXAMPLE OF FIREWALL.

A *firewall* f may then be defined as a finite sequence of rules $f = \langle r_1, \ldots, r_n \rangle$. A firewall f returns a unique action $f(p)$ for each possible packet p. This gives the *semantics* or *output* of f. We simply have

$$f(p) = \begin{cases} a_i, & \text{where } r_i \text{ is the } 1^{\text{st}} \text{ rule in } f \text{ which } p \text{ matches} \\ \texttt{undecided}, & \text{if } p \text{ matches no rule in } f \end{cases}$$

Two firewalls f_1 and f_2 are *equivalent,* written $f_1 \equiv f_2$, iff $f_1(p) = f_2(p)$ for all packets p, i.e., they decide all packets in the same way. In other words, f_1 could be replaced by f_2 (and vice versa) and we would still get the same action for every packet.

Example 1: As an example of a firewall in this notation consider $\langle r_1, r_2, r_3 \rangle$ below, where fields F_1-F_5 correspond to those used in Table I. For simplicity, in our examples only, we assume the fields' domains are [0,12] for source and destination ip address, [0,6] for source and destination port, and [0,1] for {TCP,UDP} protocol. We use these small numbers because they are easier to understand and are enough to illustrate the ideas in this paper.

$$\begin{array}{lcl} r_1: & ([0,12] \wedge [7,9] \wedge [0,6] \wedge [4] \wedge [1]) & \rightarrow \texttt{accept} \\ r_2: & ([6] \wedge [5,12] \wedge [0,6] \wedge [2] \wedge [1]) & \rightarrow \texttt{accept} \\ r_3: & ([5,7] \wedge [0,12] \wedge [0,6] \wedge [4] \wedge [1]) & \rightarrow \texttt{deny} \end{array}$$

If p_1 is the packet $(6, 8, 6, 4, 1)$ then p_1 matches rules r_1 and r_3. These rules have different actions, but since r_1 is the *first* rule in f which p_1 matches we have $f(p_1)$ equals the action of r_1, i.e., $f(p_1) = \texttt{accept}$. However the packet $p_2 = (3, 5, 2, 2, 1)$ does not match any rule in f, so $f(p_2) = \texttt{undecided}$.

III. STATIC AXIOMS: WHAT MAKES A GOOD FIREWALL?

In this section we give a small sample of the main properties which have been proposed as being necessary for a firewall to qualify as being "good". We call them *static axioms*. "Static" because they apply to a general firewall f *at one particular snapshot in time.* Given a *specific* firewall f_0, we will say f_0 *satisfies* a particular axiom if substituting f_0 for f in that axiom results in a true statement.

A first basic requirement is that for any packet p passing through the firewall, there should be a definite decision `accept` or `deny`. A firewall is called *complete* iff $f(p) \neq$ `undecided` for all packets p. Thus our first axiom is:

*Axiom 1 (***C***):* f is complete.

Firewalls usually enforce the **C** axiom by including a *default rule,* r_{n+1} as the last rule in f, i.e., a rule whose predicate P_{n+1} is $(D(F_1) \wedge D(F_2) \wedge \ldots \wedge D(F_d))$. The effect of this is that any packet p which fails to match any rule $r_1, \ldots, r_n$ in f will then match r_{n+1} and will be given the action a_{n+1} of the default rule. Although this is an easy way to enforce **C**, some authors (see [5]) view this solution as unsatisfactory because the default deny rule contributes heavily to the operational cost of the firewall.

Since firewalls must process very large amounts of network traffic in real-time, they need to be optimally configured so as to minimise their required processing-time. They should also not be *wasteful* in their use of space. The next two axioms are motivated by such considerations. Given a firewall $f = \langle r_1, \ldots, r_n \rangle$ and $i \in \{1, \ldots, n\}$, we write f^{-i} to denote the firewall which is the same as f, but with rule r_i removed. Then we say r_i is *redundant* in f iff $f \equiv f^i$. In other words it is possible to remove r_i from f without changing the action on any packet. A redundant rule unnecessarily takes up valuable space in the firewall. Thus our second static axiom is:

*Axiom 2 (***R***):* f contains no redundant rules

R here stands for **"***Non-***R***edundancy"*. Our next axiom goes further than this. It says a firewall should contain the fewest number of rules possible to implement its policy. Here, for any firewall f', $|f'|$ denotes the number of rules in f'.

*Axiom 3 (***M***):* For any firewall f', if $f' \equiv f$ then $|f| \leq |f'|$

M stands for "***M****inimisation*". It is a *stronger* axiom than **R**, in that any firewall which satisfies **M** also satisfies **R**. Other axioms to minimise a firewall's processing time can be considered. For example [6], [7] suggest to use probabilistic considerations to minimise the average length of the path a packet must make through the firewall before it finds its first matching rule.

Our final static axiom is a *syntactic constraint.* A predicate $S_1 \wedge \ldots \wedge S_d$ is called *simple* iff each S_j is a single unbroken interval. A rule r_i is simple iff its predicate P_i is simple. E.g., all the rules in Example 1 are simple, but if we replaced, say, "$[5, 7]$" with "$[0, 1] \cup [5, 7]$" in r_3 then r_3 would *not* be simple because there is a "gap" in this interval $(2, 3, 4$ are missing). Most firewall implementations (e.g. [8]) work only with simple rules, so we might require:

*Axiom 4 (***S***):* f contains only simple rules.

Given all this, we make the following definition:

Definition 1: Let $\mathcal{X}$ be a (non-empty) subset of {**C**,**R**,**M**,**S**}. A firewall is *$\mathcal{X}$-optimised* iff it satisfies the axioms in $\mathcal{X}$.

So for example, a firewall is **CS**-optimised iff it satisfies the axioms **C** and **S**, i.e., is complete and contains only simple rules.

A. Firewall Optimisation

A typical firewall may fail to satisfy one or several of the static axioms. Much research has been devoted to developing efficient algorithms which can *correct* any given firewall into an equivalent firewall which does satisfy the axioms. We make the following general definition.

Definition 2: Let $\mathcal{X}$ be a (non-empty) subset of {**C**,**R**,**M**,**S**}. A *firewall $\mathcal{X}$-optimiser* is any function Opt which, given any firewall f, returns an $\mathcal{X}$-optimised firewall $Opt(f)$ such that $[Opt(f)](p) = f(p)$ for all packets p such that $f(p) \neq \texttt{undecided}$.

Many algorithms for optimising firewalls have been proposed in the literature. **R**-optimisers are presented in [9], [10], (see also [11]), while [4] gives an **S**-optimiser. The authors of [12] provide an **M**-optimiser for one-dimensional firewalls only (i.e., involving only one field F_1). For general multi-dimensional firewalls their algorithm generates near-optimal results. It was noted already in [13] that the **M**-optimisation problem is NP-hard for the 2-dimensional case. Note that **C**-optimisation has been less well-addressed, since most authors tend to assume completeness is already given.

IV. Firewall Update

We now move on to firewall update. In this paper we are interested in the following problem:

> **Update Problem** Given a firewall f, predicate P and action a, compute a new firewall f' such that $f'(p) = a$ for all packets p matching P.

Note we do not make the assumption P is a simple predicate. The Update Problem above mentions only one given f, P and a. But actually what we need is a single *method* (hopefully efficiently implementable) which can be applied to *any* given input f, P and a and will return a new firewall. Formally we make the following definition.

Definition 3: A *firewall updater* is any function $\star$ which, given any firewall f, predicate P and action a, returns a new firewall $f \star (P, a)$.

Note there may be a number of different ways to modify f so that every packet matching P gets action a, and we do not necessarily assume that there will be a new rule added.

A. Axioms for Firewall Update

We now go through our axioms for firewall update. Each one is a requirement on a general firewall updater $\star$. We will sometimes call these *dynamic axioms*. In contrast to the static firewall axioms, they are concerned with how a firewall is *changing over time*. To say that a specific firewall updater $\star'$ *satisfies* a particular axiom means that substituting $\star'$ for $\star$ in that axiom results in a true statement.

In the Update Problem it was assumed that the FA wants a new firewall in which all packets matching P receive action a. Our first, fundamental, property says that a firewall updater $\star$ should be "successful", in that after applying it all packets matching P do indeed get action a. We emphasise that this and the rest of our axioms are meant to hold for *any* f and *any* (P, a) etc.

---

*Axiom 5 (***Success***):* For all packets p, if p matches P then $[f \star (P, a)](p) = a$.

The **Success** axiom specifies what should be the action in $f \star (P, a)$ for all packets matching P. But what about those packets which don't match P? What should be the action in $f \star (P, a)$ for them? Here, we adopt a principle which is well-known from the area of belief revision known as the *minimal-change principle* [3]. Roughly, it says that when updating any knowledge base to accept new information, the changes in the knowledge base should be kept as small as possible. In our setting this may be expressed by the following axiom, which says that for packets which don't match P, the action in the new, updated firewall should be the same as in the original f.

---

*Axiom 6 (***Preservation***):* For all packets p, if p does not match P and $f(p) \neq \texttt{undecided}$ then $[f \star (P, a)](p) = f(p)$.

The reader may notice that **Preservation** does *not* say that the actions for *all* packets not matching P should be preserved, but only for those such that $f(p) \neq \texttt{undecided}$ (we do not assume the initial firewall f is complete). If we had $f(p) = \texttt{undecided}$ and $[f \star (P, a)](p) = f(p)$ then that would mean $[f \star (P, a)](p) = \texttt{undecided}$ and so $f \star (P, a)$ would fail to satisfy axiom **C** from Section III. But as we have seen, **C** is a desirable property for firewalls. This brings us to our next group of axioms. Let $\mathcal{X}$ be a (non-empty) subset of {**C**,**R**,**M**,**S**}.

Axiom 7 ($\mathcal{X}$**-Optimisation***):* $f \star (P, a)$ is an $\mathcal{X}$-optimised firewall (see Definition 1).

Note $\mathcal{X}$**-Optimisation** is really not just one axiom but rather a *group* of axioms - one for each possible $\mathcal{X}$. Thus there is **R-Optimisation** ($f \star (P, a)$ contains no redundant rules), **CS-Optimisation** ($f \star (P, a)$ is always a complete, simple firewall), etc. $\mathcal{X}$**-Optimisation** gives the link between the static axioms and the dynamic ones. It just says that a firewall updater should always return a "good" firewall (i.e., satisfying all the static axioms in $\mathcal{X}$). However, as we mentioned after Definition 2, in practice **M-Optimisation** will be difficult to satisfy due to computational limits. It is expecting *too much* to be able to find a updater which satisfies it. Instead we may try to replace it with less demanding axioms.

---

*Axiom 8 (***M-Containment***):* $|f \star (P, a)| \leq |f| + 1$

This axiom places a limit on the amount by which a firewall is allowed to *grow* after updating. It says the number of rules in the firewall *after* updating should not increase by more than

Algorithm 1 InsertLast

```
Input: firewall $f = \langle r_1, \dots, r_n \rangle$, rule $P \to a$
Output: firewall InsertLast$(f, P \to a)$
InsertLast$(f, P \to a) \Leftarrow \langle r_1, \dots, r_n, (P \to a) \rangle$
```

one. In other words we should allow space in the firewall for *at most one* extra rule to bring in the new information (P, a). This axiom seems to be more realistic than **M-Optimisation**.

Now suppose the FA desires $\mathcal{X}$**-Optimisation** for some *particular* subset $\mathcal{X}$ of $\{\mathbf{C},\mathbf{R},\mathbf{M},\mathbf{S}\}$. Suppose the initial firewall f is *already* $\mathcal{X}$-optimised **and** it *already* gives action a to every packet matching P. In this case it seems there is nothing that needs to be done – f already behaves the way the FA wants, so updating should leave f unchanged:

Axiom 9 ($\mathcal{X}$**-Vacuity***):* If f is $\mathcal{X}$-optimised and $f(p) = a$ for all packets p matching P, then $f \star (P, a) = f$.

Again note that this is really a set of axioms - one for each $\mathcal{X}$. $\mathcal{X}$**-Vacuity** can be seen as another application of the minimal change principle to firewall update.

B. Some Example Firewall Updaters

We now describe some example firewall updaters. In what follows we assume $f = \langle r_1, \dots, r_n \rangle$.

1) First updater $\star_1$*:* In our first operator we simply add $P \to a$ into the last position in f using the InsertLast algorithm given in Algorithm 1. That is, we define for each f, P, a,

$$f \star_1 (P, a) := \texttt{InsertLast}(f, P \to a).$$

The updater $\star_1$ is not really suitable as a useful firewall updater, because it can be shown to satisfy the following unwanted axiom:

*Axiom 10 (***Strong Preservation***):* If $f(p) \neq$ undecided then $[f \star (P, a)](p) = f(p)$.

This axiom expresses that information in the original firewall f always has *priority* over the new information (P, a). It says the actions for all packets not undecided in f should stay the same after updating. This is like the opposite of **Success**, which says new information should have priority over the old firewall. This axiom implies **Preservation** (hence its name), so we see $\star_1$ satisfies **Preservation**. However, as might be expected, $\star_1$ does not satisfy **Success.** This means there is *some* f, P and a, and some packet p matching P such that $[f \star_1 (P, a)](p) \neq a$. (For example, given any predicate-action pair (P, a) let f be *any* firewall whose first rule is $r_1 = P \to b$, where $b \neq a$. Then $[f \star_1 (P, a)](p) = b \neq a$ for *all* p matching P.)

2) Second updater $\star_2$*:* With $\star_2$ we add a *rule-deletion* step prior to $\star_1$. First we delete all rules $r_i = P_i \to a_i$ which conflict with $P \to a$, and then add $P \to a$ to the end. This procedure is described by the AddRule2 algorithm (Algorithm 2). Then

$$f \star_2 (P, a) := \texttt{AddRule2}(f, P \to a).$$

Algorithm 2 AddRule2

```
Input: firewall $f = \langle r_1, \dots, r_n \rangle$, rule $(P \to a)$
Output: firewall AddRule2$(f, P \to a)$
for each rule $r_i$ in $f$ do
    if $r_i$ and the rule $P \to a$ conflict then
        delete $r_i$
    end
end
$f' \Leftarrow \langle r_{k_1}, r_{k_2}, \dots, r_{k_s} \rangle$
    /* the undeleted rules of $f$ */
AddRule2$(f, P \to a) \Leftarrow$ InsertLast$(f', P \to a)$
```

Algorithm 3 AddRule3

```
Input: firewall $f = \langle r_1, \dots, r_n \rangle$, rule $P \to a$
Output: firewall AddRule3$(f, P \to a)$
for each rule $r_i$ in $f$ do
    if $r_i$ and $(P \to a)$ conflict and $P_i \neq P$ then
        $[r_i] \Leftarrow ((P_i \cap P) \to a_i, (P_i \cap P^c) \to a_i)$
    else
        $[r_i] \Leftarrow (r_i)$
    end
end
AddRule3$(f, P \to a) \Leftarrow$ AddRule2$(\langle [r_1], \dots, [r_n] \rangle, P \to a)$
```

The operator $\star_2$ does satisfy **Success,** but suffers from a different problem: It fails to satisfy **Preservation** as the next example shows**.**

Example 2: Let f be the firewall from Example 1 in Section II, let $P = ([7, 12] \wedge [7, 9] \wedge [0, 6] \wedge [4] \wedge [1])$ and let $a =$ deny. Then $f \star_2 (P, a) = \langle r_2, r_3, (P \to a) \rangle$. Consider the packet $p_1 = (6, 8, 6, 4, 1)$. This packet does not match P, but we have $f(p_1) =$ accept (because $p_1$matches r_1) and $[f \star_2 (P, a)](p_1) =$ deny.

3) Third updater $\star_3$*:* To fix the problem with $\star_2$, a further *rule-splitting* step may be introduced prior to $\star_2$. If a rule r_i in f conflicts with $P \to a$ and is such that $P_i \neq P$ then we split r_i into the two disjoint rules $(P_i \cap P) \to a_i$ and $(P_i \cap P^c) \to a_i$. (Here "$\cap$" and "$\cdot^c$" resp. denote component-wise set intersection and complement.) This "separates out" that part of r_i which is conflicting with $P \to a$, to ensure the part which doesn't conflict is kept. We then perform Addrule2 on the resulting firewall. The process is described by the algorithm AddRule3 (Algorithm 3). We define:

$$f \star_3 (P, a) := \texttt{AddRule3}(f, P \to a).$$

Example 3: Let f, P, a be the same as in Example 2. Then $f \star_3 (P, a) = \langle r_1', r_2, r_3, (P \to a) \rangle$, where $r_1' = ([0, 6] \wedge [7, 9] \wedge [0, 6] \wedge [4] \wedge [1]) \to$ accept.

4) Fourth updater $\star_4$*:* Finally a fourth operator, defined by simply putting $P \to a$ to the front of f using the simple InsertFirst algorithm (Algorithm 4).

$$f \star_4 (P, a) := \texttt{InsertFirst}(f, P \to a).$$

$\star_4$ also satisfies **Success** and **Preservation**. In fact one can check that for all f, P, a we have $f \star_3 (P, a) \equiv f \star_4 (P, a)$,

Algorithm 4 `InsertFirst`

```
Input: firewall $f = \langle r_1, \ldots, r_n \rangle$, rule $(P \to a)$
Output: firewall InsertFirst$(f, P \to a)$
InsertFirst$(f, P \to a) \Leftarrow \langle (P \to a), r_1, \ldots, r_n \rangle$
```

	$\star_1$	$\star_2$	$\star_3$	$\star_4$
Success	×	✓	✓	✓
Preservation	✓	×	✓	✓
$\mathcal{X}$**-Optimisation**	×	×	×	×
M-Containment	✓	✓	✓	✓
$\mathcal{X}$**-Vacuity**	×	×	×	×

Table II
SHOWING WHICH UPDATERS SATISFY WHICH AXIOMS

i.e., updating using $\star_4$ results in a new firewall which gives to every packet the same action we would get if we updated using $\star_3$ instead.

Theorem 1: The satisfaction of the dynamic axioms by the firewall updaters $\star_1$-$\star_4$ is summarised in Table II.

In the rows for $\mathcal{X}$**-Optimisation** and $\mathcal{X}$**-Vacuity**, $\mathcal{X}$ stands for *any* (non-empty) subset of {**C**,**R**,**M**,**S**}, because in fact each of $\star_1$-$\star_4$ fails to satisfy those axioms for *all* possible choices of $\mathcal{X}$. In the case of $\mathcal{X}$**-Vacuity** this is because, when updating with (P, a), *all* of these updaters *always* insert the rule $P \to a$, even if f is optimised and $f(p) = a$ already for all packets p matching P.

In the case of $\mathcal{X}$**-Optimisation** some of the optimisation axioms are satisfied by some of $\star_1$-$\star_4$ in special cases, but certainly not in all. For example all except $\star_2$ will satisfy **C-Optimisation** in the special case when the original firewall f is already complete, and all except $\star_3$ satisfy **S-Optimisation** if f is already **S**-optimised and P is a simple predicate. ($\star_3$ fails this because the rule-splitting step may fragment a simple rule into non-simple parts.) However none of the updaters satisfy **R-Optimisation** even in the special case when f already contains no redundant rules, because the insertion of new rule $P \to a$ might itself be redundant or might cause some other rules in f to become redundant. We give examples here for $\star_3$ and $\star_4$:

Example 4: Assume for simplicity there is just one field F_1 with domain $[0, 10]$, and let the initial firewall be

$$f = \langle ([0,5] \to \texttt{accept}),\ ([5,10] \to \texttt{deny}) \rangle.$$

Clearly neither of these 2 rules is redundant in f, so f is **R**-Optimised. Then $f \star_3 ([5], \texttt{accept}) = \langle ([0,5] \to \texttt{accept}), ([6,10] \to \texttt{deny}), ([5] \to \texttt{accept}) \rangle$. The last rule here is clearly redundant, because the only packet matching it, i.e., (5), matches an earlier rule, so it will never be used. This shows $\star_3$ fails to satisfy **R-Optimisation**. For $\star_4$ we have$f \star_4 ([6,10], \texttt{deny}) = \langle ([6,10] \to \texttt{deny}), ([0,5] \to \texttt{accept}), ([5,10] \to \texttt{deny}) \rangle$. Here, the insertion of the new rule at the start of f cause the last rule to be redundant, thus showing $\star_4$ fails **R-Optimisation** too.

How to define an updater which satisfies $\mathcal{X}$**-Optimisation** in all cases? The next section explores one idea.

V. OPTIMISED UPDATERS

At first glance, it seems there is a simple way we can modify a given firewall updater to ensure $\mathcal{X}$**-Optimisation** is satisfied. We just add an $\mathcal{X}$-optimisation step, i.e., we use the following two steps: (1) Update f using an updater like the ones defined on the previous subsection to get new firewall $f \star (P, a)$, then (2) pass $f \star (P, a)$ on to an $\mathcal{X}$-optimiser (see Definition 1). Formally we make the following definition:

Definition 4: Let $\mathcal{X}$ be a (non-empty) subset of {**C**,**R**,**M**,**S**} and let Opt be a firewall $\mathcal{X}$-optimiser. For any firewall updater $\star$ the *$\mathcal{X}$-optimised version of $\star$ (using Opt)* is the firewall updater $\star^{Opt}$ defined by setting, for any firewall f and predicate-action pair (P, a), $f \star^{Opt} (P, a) := Opt(f \star (P, a))$.

Note that given any updater $\star$ and optimiser Opt, $\star^{Opt}$ is still a firewall updater according to Definition 3, so we can ask if it satisfies the dynamic axioms of Section IV-A. By Definition 1, if Opt is an $\mathcal{X}$-optimiser then $Opt(f \star (P, a))$ is an $\mathcal{X}$-optimised firewall for any $\star$, so $\star^{Opt}$ definitely satisfies $\mathcal{X}$**-Optimisation**. Also by Definition 1 we know $Opt(f \star (P, a))$ gives the same action as $f \star (P, a)$ for all packets p such that $f \star (P, a) \neq \texttt{undecided}$. Hence we also have:

Proposition 1: Let $\star$ be any firewall updater and $\star^{Opt}$ an $\mathcal{X}$-optimised version. If $\star$ satisfies **Success** and **Preservation**, then $\star^{Opt}$ satisfies them too.

From this result, since we know $\star_3$ and $\star_4$ satisfy **Success** and **Preservation** from the previous subsection, we can be sure their $\mathcal{X}$-optimised versions $\star_3^{Opt}$ and $\star_4^{Opt}$ will too. However, any $\mathcal{X}$-optimised version $\star_1^{Opt}$ of $\star_1$ still does not satisfy **Success**, and any $\mathcal{X}$-optimised version $\star_2^{Opt}$ of $\star_2$ still does not satisfy **Preservation**.

A. Checking $\mathcal{X}$-Vacuity: a Special Case

An interesting question is whether a particular $\mathcal{X}$-optimised version of $\star$ satisfies $\mathcal{X}$**-Vacuity**. In general this will depend on the particular optimiser Opt and updater $\star$ being used. To give just an example we will focus in this subsection on the case $\mathcal{X}$ =**R**. Furthermore we will use the algorithm for **R**-optimisation of Liu et al, described in [9] (and its more efficient version in [10]), which we call `RemoveRedundant`. We refer the reader to those papers for details. The basic idea is that the algorithm works its way backwards through the input firewall f, checking each rule for redundancy with the rules which have not yet been deleted. If it is redundant it is removed, and then the next rule is checked. Let us define $Opt^{\mathbf{R}}(f) := \texttt{RemoveRedundant}(f)$.

For each i we may now ask the question "Does $\star_i^{Opt}$ satisfy **R-Vacuity** when $Opt = Opt^{\mathbf{R}}$?" Well, in this case $\star_1^{Opt}$ satisfies **R-Vacuity** because the only change $\star_1$ makes to f is to add $P \to a$ to the end of f, and if $f(p) = a$ already for all p matching P then $P \to a$ is redundant and will be removed in the first step of $\texttt{RemoveRedundant}(f \star_1 (P, a))$. To see $\star_3^{Opt}$ and $\star_4^{Opt}$ **fail** to satisfy **R-Vacuity** in this case consider the following continuation of Example 4.

Example 5: Let f be the **R**-optimised firewall from Example 4. Note that $f(5) = \texttt{accept}$, but passing the result of $f \star_3 ([5], \texttt{accept})$ to $Opt^{\mathbf{R}}$ gives $f \star_3^{Opt} (P_1, a_1) = \langle ([0,5] \to$

	$\star_1^{Opt}$	$\star_2^{Opt}$	$\star_3^{Opt}$	$\star_4^{Opt}$
Success	×	✓	✓	✓
Preservation	✓	×	✓	✓
R-Optimisation	✓	✓	✓	✓
M-Containment	✓	✓	✓	✓
R-Vacuity	✓	×	×	×

Table III
SHOWING WHICH **R**-OPTIMISED UPDATERS SATISFY WHICH AXIOMS

$\texttt{accept}), ([6, 10] \to \texttt{deny})\rangle \neq f$. Although the first rule in f is unchanged, the second rule is changed even though what happens after the first rule in f is not relevant to the new information $([5], \texttt{accept})$. Also note $f(p) = \texttt{deny}$ for all packets matching $[6, 10]$, but passing $f\star_4([6, 10], \texttt{deny})$ to $Opt^{\mathbf{R}}$ gives $f\star_4^{Opt}(P_2, a_2) = \langle([6, 10] \to \texttt{deny}), ([0, 5] \to \texttt{accept})\rangle \neq f$.

Theorem 2: The satisfaction of the dynamic axioms by $\star_1^{Opt}$-$\star_4^{Opt}$ (when $\mathcal{X} = \mathbf{R}$ and $Opt = Opt^{\mathbf{R}}$) is shown in Table III.

Thus we see that even though adding an optimisation step takes care of the $\mathcal{X}$**-Optimisation** axioms it is unable to handle $\mathcal{X}$**-Vacuity** in general. From this negative result we suggest that, if we want some method which fulfills *all* the axioms, it will be necessary to perform update and optimisation in a *single integrated step* rather than split these into two distinct steps. This is the subject of our ongoing work.

VI. Related Work

As we said earlier, most previous work on firewalls has focused on static properties of a firewall f and on algorithms for *repairing* firewalls which fail to meet some static requirements (see [4],[9],[11],[14],[15],[16],[17]). In these works, the authors define error discovery, rule optimisation and rule analysis, and packet filtering which is different from the problem of correct firewall rule update. However, there has been *some* research on dynamic f, where which comes from any new information (P, a) outside f (see [18]). They add a new rule into f, then they fix problems inside f.

One work which is quite related to ours is [19], which gives an algorithm to compute the *impact* on f of any proposed change to f. The change can be the insertion or deletion or modification of a rule in a *pre-specified* position, and the impact is measured as the number of packets which get mapped to a different action, before and after the change. This work is complementary to ours, in that we are trying to automatically find the *correct* position to add/delete/modify rules in order to minimise the impact of a firewall policy change.

Finally there is a vast body of work in philosophy and knowledge representation on knowledge update, aka *belief revision*. A leading textbook in the area is [3].

VII. Conclusion and Future Work

We looked at the problem of firewall update, mainly from axiomatic viewpoint. We proposed several axioms for firewall update, which are meant to express some requirements which any reasonable method of update should fulfill, and which can be thought of as a checklist against which any specifically proposed update algorithm should be checked. We gave some simple examples of firewall updaters and showed that none of them was able to satisfy all the axioms simultaneously.

For future work the most important next step will be to find and implement an efficient algorithm for firewall update which satisfies all of the axioms. Also, in this paper we took the **Success** axiom as a fundamental axiom which is never to be broken. However this axiom is not entirely unquestionable - maybe we don't always want new information to have priority over the old. We will look at firewall update in the context where this axiom is left out. Finally we will look at also at the related problem of *automatically merging* different firewalls coming from different FAs into a single firewall.

Acknowledgements

Thanks are due to Dr Somnuk Puangpronpitag and the ISAN group, Faculty of Informatics, MSU for their support and helpful comments.

References

[1] E. D. Zwicky, S. Cooper, and D. B. Chapman, *Building Internet Firewalls*. O'Reilly, 2000.
[2] C. M. Kozierok, *The TCP/IP Guide : a comprehensive, illustrated internet protocols reference*. William Pollock, 2005.
[3] S. O. Hansson, *A Textbook of Belief Dynamics*. Kluwer Academic Publishers, 1999.
[4] M. G. Gouda and A. X. Liu, "Structured firewall design," *Computer Networks*, vol. 51, no. 4, pp. 1106–1120, 2007.
[5] S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greenberg, "Traffic-aware firewall optimization strategies," *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 5, pp. 2225–2230, June 2006.
[6] E. W. Fulp, "Optimization of firewall policies using directed acyclical graphs," in *Proceedings of the IEEE Internet Management Conference*, 2005.
[7] H. Hamed and E. Al-Shaer, "Dynamic rule-ordering optimization for high-speed firewall filtering," in *ASIACCS*, pp. 332–342, 2006.
[8] ipchains, "http://tldp.org/howto/ipchains-howto.html,"
[9] A. X. Liu and M. G. Gouda, "Complete redundancy detection in firewalls," in *DBSec*, pp. 193–206, 2005.
[10] A. X. Liu, C. R. Meiners, and Y. Zhou, "All-match based complete redundancy removal for packet classifiers in TCAMs," in *Proceedings of the 27th Annual IEEE Conference on Computer Communications (Infocom)*, (Phoenix, Arizona), April 2008.
[11] T. Chomsiri and C. Pornavalai, "Firewall rules analysis," in *Security and Management*, pp. 213–219, 2006.
[12] A. X. Liu, E. Torng, and C. Meiners, "Firewall compressor: An algorithm for minimizing firewall policies," in *Proceedings of the 27th Annual IEEE Conference on Computer Communications (Infocom)*, (Phoenix, Arizona), April 2008.
[13] D. Applegate, G. Calinescu, D. S. Johnson, H. J. Karloff, K. Ligett, and J. Wang, "Compressing rectilinear pictures and minimizing access control lists," in *SODA*, pp. 1066–1075, 2007.
[14] A. X. Liu and M. G. Gouda, "Diverse firewall design," in *DSN*, pp. 595–604, 2004.
[15] S. P. Hidalgo, R. Ceballos, and R. M. Gasca, "Fast algorithms for consistency-based diagnosis of firewall rule sets," in *ARES*, pp. 229–236, 2008.
[16] H. Adiseshu, S. Suri, and G. M. Parulkar, "Detecting and resolving packet filter conflicts," in *INFOCOM*, pp. 1203–1212, 2000.
[17] S. Khummanee, J. Tungmondee, and P. Daraboot, "Applied variable-length subnet mask(vlsm) with matching firewall," in *International Joint Conference Computer Science and Software Engineering*, 2008.
[18] S. Khummanee, J. Tungmondee, and P. Daraboot, "Auto-optimized firewall policy management," in *The National Conference on Computing and Information Technology*, 2008.
[19] A. X. Liu, "Change-impact analysis of firewall policies," in *ESORICS*, pp. 155–170, 2007.