RESEARCH-ARTICLE

Firewall policy change-impact analysis

ALEX X LIU, Michigan State University, East Lansing, MI, United States

Published: 23 March 2008
Accepted: 01 November 2011
Revised: 01 September 2010
Received: 01 October 2008

ACM Transactions on Internet Technology (TOIT), Volume 11, Issue 4 (March 2012)
https://doi.org/10.1145/2109211.2109212
EISSN: 1557-6051

Firewall Policy Change-Impact Analysis

ALEX X. LIU, Michigan State University

Firewalls are the cornerstones of the security infrastructure for most enterprises. They have been widely deployed for protecting private networks. The quality of the protection provided by a firewall directly depends on the quality of its policy (i.e., configuration). Due to the lack of tools for analyzing firewall policies, many firewalls used today have policy errors. A firewall policy error either creates security holes that will allow malicious traffic to sneak into a private network or blocks legitimate traffic and disrupts normal business processes, which in turn could lead to irreparable, if not tragic, consequences. A major cause of policy errors are policy changes. Firewall policies often need to be changed as networks evolve and new threats emerge. Users behind a firewall often request the firewall administrator to modify rules to allow or protect the operation of some services.

In this article, we first present the theory and algorithms for firewall policy change-impact analysis. Our algorithms take as input a firewall policy and a proposed change, then output the accurate impact of the change. Thus, a firewall administrator can verify a proposed change before committing it. We implemented our firewall change-impact analysis algorithms, and tested them on both real-life and synthetic firewall policies. The experimental results show that our algorithms are effective in terms of ensuring firewall policy correctness and efficient in terms of computing the impact of policy changes. Thus, our tool can be practically used in the iterative process of firewall policy design and maintenance. Although the focus of this article is on firewalls, the change-impact analysis algorithms proposed in this article are not limited to firewalls. Rather, they can be applied to other rule-based systems, such as router access control lists (ACLs), as well.

Categories and Subject Descriptors: C.2.0 [**Computer-communication Networks**]: General—*Security and protection (e.g., firewalls)*; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; D.2.9 [**Software Engineering**]: Management—*Software configuration management*

General Terms: Algorithms, Management, Security

Additional Key Words and Phrases: Firewall policy, change-impact analysis, access control

ACM Reference Format:

Liu, A. X. 2012. Firewall policy change-impact analysis. ACM Trans. Internet Technol. 11, 4, Article 15 (March 2012), 24 pages.

DOI = 10.1145/2109211.2109212 http://doi.acm.org/10.1145/2109211.2109212

1. INTRODUCTION

1.1. Background

Serving as the first line of defense against malicious attacks and unauthorized traffic, firewalls are cornerstones of network security and have been widely deployed in

---

A preliminary version of this article was published as Liu [2007].

The work of Alex X. Liu is supported in part by the National Science Foundation under Grant No. CNS-0716407, CNS-0916044, and CNS-0845513. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Author's address: A. X. Liu, Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824-1266; email: alexliu@cse.msu.edu.

DOI 10.1145/2109211.2109212 http://doi.acm.org/10.1145/2109211.2109212

Table I. An Example Firewall

Rule	Src IP	Dest IP	Src Port	Dest Port	Protocol	Action
r_1	*	192.168.0.1	*	25	TCP	accept
r_2	1.2.3.4	*	*	*	*	discard
r_3	*	*	*	*	*	accept

businesses and institutions. According to the 2008 CSI/FBI Computer Crime and Security Survey [Richardson 2008], firewalls are the most widely adopted security technology—almost all organizations that participated in the survey use firewalls to protect their private networks. A firewall is placed at the point of entry between a private network and the outside Internet such that all incoming and outgoing packets have to pass through it. The function of a firewall is to examine every incoming or outgoing packet and decide whether to accept or discard it. This function is specified by a sequence (an ordered list) of rules, which is called the policy, i.e., the configuration, of the firewall. Each rule in a firewall policy is of the form $\langle predicate \rangle \rightarrow \langle decision \rangle$. The $\langle predicate \rangle$ of a rule is a Boolean expression over some packet fields such as source IP address, destination IP address, source port number, destination port number, and protocol type. The $\langle decision \rangle$ of a rule can be *accept*, *discard*, or a combination of these decisions with other options such as a logging option. The rules in a firewall policy often conflict. To resolve such conflicts, the decision for each packet is the decision of the first (i.e., highest priority) rule that the packet matches. Table I shows an example firewall.

1.2. Motivation

Although a firewall policy is a mere sequence of rules, correctly maintaining one is by no means easy. First, the rules in a firewall policy are logically entangled because of conflicts among rules and the resulting order sensitivity. Second, a firewall policy may consist of a large number of rules. A firewall on the Internet may consist of hundreds or even thousands of rules in some cases. Last but not least, an enterprise firewall policy often consists of legacy rules that are written by different administrators, at different times, and for different reasons, which makes maintaining firewall policies even more difficult. Analyzing a large and complex sequence of logically related rules is certainly beyond human capability. Effective methods and tools for analyzing firewall policies, therefore, are crucial to the success of firewalls. However, firewall administrators are woefully under-assisted due to the lack of firewall policy analysis tools. Quantitative studies have shown that most firewalls on the Internet are plagued with policy errors [Wool 2004, 2010]. A firewall policy error either creates security holes that will allow malicious traffic to sneak into a private network or blocks legitimate traffic and disrupts normal business processes, which in turn could lead to irreparable, if not tragic, consequences.

Firewall policies are always subject to change due to a variety of reasons. Making policy changes is a major task of firewall administrators. For example, new network threats such as worms and viruses may emerge. To protect a private network from new attacks, firewall policies need to be changed accordingly. Modern organizations also continually transform their network infrastructure to maintain their competitive edge by adding new servers, installing new software and services, expanding connectivity, and so on. In accordance with network changes, firewall policies need to be changed as well, to provide necessary protection.

Unfortunately, making changes is a major source of firewall policy errors. Making correct firewall policy changes is remarkably difficult due to the interleaving nature of firewall rules. For example, when a firewall administrator inserts a new rule into a firewall policy, the meaning of the rules listed under this rule could be incorrectly changed without being noticed. Furthermore, firewall policy changes are made by human administrators, and it is common that human administrators make mistakes. Configuration errors have been observed to be the largest cause of failure for Internet services [Oppenheimer et al. 2003]. A recent Yankee Group report has shown that more than 62% of network downtime is due to human configuration errors [Kerravala 2004]. Administrators often face tremendous pressure to fix problems quickly because firewalls deployed on operational networks often support critical business applications and important communications.

Firewall policy errors can be dangerous and costly. On one hand, if a firewall policy error permits illegitimate communication, outside attackers may use these security holes to launch attacks. On the other hand, if a firewall policy error disallows legitimate communication, it may cause significant loss due to interrupted business. For example, if a firewall policy error prevents the communication between a Web server and its supporting database server, all transactions that need such communication are disrupted. The loss caused by network outages has become increasingly acute. For example, the estimated revenue losses per hour of downtime for the industries of media, banking, and brokerage are 1.2, 2.6, and 4.5 million dollars, respectively [Kerravala 2004]. The reality could be much worse than these published staggering numbers, as the errors of excessive reachability often go undetected, while insufficient reachability is a common source of complaints to operators.

1.3. The Problem

The fundamental problem in changing firewall policies is, how does a firewall administrator know that the change made to the firewall policies is correct? For example, suppose a firewall administrator wants to make a change to the firewall policy to allow a database server to talk to a Web server. How does the administrator know that the change indeed enables this communication? Also, how does the administrator know that the change does not allow some other illegitimate traffic to flow as a side effect, given the subtle behavior of firewall rules? Such questions are difficult to answer given the high complexity of firewall rules.

In this context, if there is a tool that takes a firewall configuration and a proposed change as input, then outputs the precise impact of the change, the errors caused by making policy changes would be greatly reduced. The impact of a change shows all the traffic that was formerly discarded, but is now accepted, and all the traffic that was formerly accepted, but is now discarded. The output impact must be human readable. With this tool on hand, a firewall administrator can examine the change-impact for unintended consequences.

1.4. Key Contributions

In this article, we make the following three key contributions.

(1) We develop a theory foundation for firewall policy change-impact analysis. We identify four types of firewall policy changes: rule deletion, rule insertion, rule modification, and rule swap. For each type of change, we have a theorem that states the decisions of which packets will be changed due to the policy change. These theorems serve as the foundation for developing algorithms for computing firewall policy change-impact.
(2) We present algorithms for firewall policy change-impact analysis. The input of our algorithms includes a firewall policy and a proposed change, and the output is the accurate impact of the change. Using our algorithms, an administrator can verify a proposed change before committing it.

Table II. Impact after Deleting r_1 from the Firewall in Table I

Source IP	1.2.3.4
Destination IP:	192.168.0.1
Source Port:	*
Destination Port:	25
Protocol Type:	TCP
Decision before change:	accept
Decision after change:	discard

(3) We present a way to correlate the impact of a firewall policy change and the high level security requirements that the firewall needs to satisfy. We also present methods for making corrections if the impact of a change is not desirable.

Because the focus of this article is on security policies, we simply use the term "firewall" to mean "firewall policy", "firewall rule set", or "firewall configuration", unless otherwise specified.

1.5. Road Map

The rest of this article proceeds as follows. In Section 2, we show an example application of our firewall change-impact analysis tool. We then give formal definitions in Section 3.1. In Section 3, we present the theory foundation for firewall change-impact analysis. Based on these theorems, we develop algorithms for computing the impact of firewall changes in Section 4. In Section 5, we discuss some further issues for firewall change-impact analysis. In Section 7, we show our experimental results. In Section 8, we examine previous work and compare it with our approach. In Section 9, we give concluding remarks.

2. EXAMPLE

In this section, we show an example application of our firewall policy change-impact analysis tool. Consider the example firewall in Table I. We suppose that the private network behind this firewall has a mail server and a Web server, whose IP addresses are 192.168.0.1 and 192.168.0.2, respectively. We further suppose that this firewall is required by its high level security policies to discard all packets from a malicious host whose IP address is 1.2.3.4.

Here we briefly explain the meaning of the three rules in Table I. Rule r_1 means that all email packets to the email server are accepted. Note that for a packet, if its destination port number is 25 and its protocol type is TCP, then the packet is an email (SMTP) packet. Rule r_2 means that all packets from 1.2.3.4 are discarded. Rule r_3 means that all packets are accepted. Note that whenever a packet arrives at a firewall, the decision of the first rule that the packet matches is executed.

2.1. Rule Deletion

Suppose that the administrator of this firewall wants to delete rule r_1. Our change-impact analysis tool will output the following impact as shown in Table II. The meaning of the impact is as follows. For the email packets from the malicious host 1.2.3.4 to the email server 192.168.0.1, before deleting rule r_1, the decision for such packets is *accept*; after deleting rule r_1, the decision for such packets is *discard*.

2.2. Rule Insertion

Suppose that the administrator of this firewall wants to insert the following rule above rule r_1:

Src IP	Dest. IP	Src Port	Dest. Port	Protocol	Action
*	192.168.0.2	*	80	TCP	accept

Table III. Impact after Inserting a Rule Above r_1 in the Firewall in Table I

Source IP	1.2.3.4
Destination IP:	192.168.0.2
Source Port:	*
Destination Port:	80
Protocol Type:	TCP
Decision before change:	discard
Decision after change:	accept

Table IV. Impact after Modifying r_1 in the Firewall in Table I

Source IP	1.2.3.4
Destination IP:	192.168.0.1
Source Port:	*
Destination Port:	[1,24]
Protocol Type:	TCP
Decision before change:	discard
Decision after change:	accept
Source IP	1.2.3.4
Destination IP:	192.168.0.1
Source Port:	*
Destination Port:	[25, 65536]
Protocol Type:	TCP
Decision before change:	discard
Decision after change:	accept

The meaning of this new rule is to accept all the HTTP packets to the Web server 192.168.0.2. After the administrator applies this intended change and the original firewall in Table I to our change-impact analysis tool, the tool will output the impact shown in Table III.

2.3. Rule Modification

Suppose that the administrator of this firewall wants to modify rule r_1 to be the following.

Src IP	Dest. IP	Src Port	Dest. Port	Protocol	Action
*	192.168.0.1	*	*	TCP	accept

The meaning of this modified rule is to accept all the TCP packets to the mail server 192.168.0.1. For this intended change, our change-impact analysis tool outputs the impact shown in Table IV.

2.4. Rule Swap

Suppose that the administrator of this firewall wants to swap rules r_1 and r_2. Similarly, for this intended change, our change-impact analysis tool outputs the same impact as shown in Table II.

3. CHANGE-IMPACT ANALYSIS: BACKGROUND AND THEORY

3.1. Background

We now formally define the concepts of fields, packets, and firewalls. A *field* F_i is a variable of finite length (of a finite number of bits). The domain of field F_i of w bits, denoted $D(F_i)$, is $[0, 2^w - 1]$. A *packet* over the d fields $F_1, \ldots, F_d$ is a d-tuple $(p_1, \ldots, p_d)$ where each p_i $(1 \le i \le d)$ is an element of $D(F_i)$. Firewalls usually check the following five fields: source IP address, destination IP address, source port number, destination port number, and protocol type. The lengths of these packet fields are 32, 32, 16, 16, and 8, respectively. We use Σ to denote the set of all packets over fields $F_1, \ldots, F_d$. It follows that Σ is a finite set and $|\Sigma| = |D(F_1)| \times \cdots \times |D(F_d)|$, where $|\Sigma|$ denotes the number of elements in set Σ and $|D(F_i)|$ denotes the number of elements in set $D(F_i)$.

A *rule* has the form $\langle predicate \rangle \rightarrow \langle decision \rangle$. A $\langle predicate \rangle$ defines a set of packets over the fields F_1 through F_d, and is specified as $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d$, where each S_i is a subset of $D(F_i)$ and is specified as either a prefix or a nonnegative integer interval. A *prefix* $\{0,1\}^k\{*\}^{w-k}$ with k leading 0s or 1s for a packet field of length w denotes the integer interval $[\{0,1\}^k\{0\}^{w-k}, \{0,1\}^k\{1\}^{w-k}]$. For example, prefix 01** denotes the interval [0100, 0111]. A rule $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d \rightarrow \langle decision \rangle$ is a *prefix rule* if and only if each S_i is represented as a prefix. A rule $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d \rightarrow \langle decision \rangle$ is called an *atomic rule* if and only if each S_i is specified as either a prefix or a nonnegative integer interval.

A packet matches a rule if and only if the packet matches the predicate of the rule. A packet $(p_1, \ldots, p_d)$ *matches* a predicate $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d$ if and only if the condition $p_1 \in S_1 \wedge \cdots \wedge p_d \in S_d$ holds. We use DS to denote the set of possible values that $\langle decision \rangle$ can be. Typical elements of DS include accept, discard, accept with logging, and discard with logging.

A *firewall* f is a sequence of rules that is complete. A sequence of rules $\langle r_1, \ldots, r_n \rangle$ is *complete* if and only if for any packet p, there is at least one rule in the sequence that p matches. To ensure that a sequence of rules is complete and thus a packet classifier, the predicate of the last rule is usually specified as $F_1 \in D(F_1) \wedge \cdots F_d \in \wedge D(F_d)$.

Two rules in a firewall may *overlap*; that is, a single packet may match both rules. Furthermore, two rules in a firewall may *conflict*; that is, the two rules not only overlap but also have different decisions. Firewalls typically resolve such conflicts by employing a first-match resolution strategy where the decision for a packet p is the decision of the first (highest priority) rule that p matches in f. The decision that firewall f makes for packet p is denoted $f(p)$.

We can think of a firewall f as defining a many-to-one mapping function from Σ to DS. Two firewalls f_1 and f_2 are *equivalent*, denoted $f_1 \equiv f_2$, if and only if they define the same mapping function from Σ to DS; that is, for any packet $p \in \Sigma$, we have $f_1(p) = f_2(p)$. A rule is *redundant* in a firewall if and only if removing the rule does not change the semantics of the firewall.

3.2. Problem Statement

In this article, we consider the following four types of changes that one can make to a firewall $\langle r_1, \ldots, r_n \rangle$. Note that we focus on single rule changes because in practice firewall administrators typically make one rule change a time Mayer private communication. If firewall administrators need to make multiple rule changes, they can use our tool to verify that the impact of each change is correct.

(1) *Deletion.* delete rule r_i, where $1 \leq i \leq n-1$.
(2) *Insertion.* insert a new rule r between r_i and r_{i+1}, where $1 \leq i \leq n-1$.
(3) *Modification.* modify rule r_i to be r_i', where $1 \leq i \leq n-1$.
(4) *Swap.* swap the two rules r_i and r_j, where $1 \leq i < j \leq n-1$.

Recall that the predicate of the last rule in a firewall is always a tautology, which is for the purpose of ensuring the comprehensiveness property of the firewall. Therefore we assume that one does not change the last rule. Actually, given any firewall where the predicate of the last rule is not a tautology, we can modify the predicate of the last rule to be a tautology without changing the semantics of the firewall.

Each rule in a firewall is associated with two sets of packets, a matching set and a resolving set [Liu and Gouda 2005]. More precisely, consider a firewall f that consists of n rules $\langle r_1, r_2, \ldots, r_n \rangle$. The matching set of a rule r_i, denoted $M(r_i)$, is the set of all packets that match r_i. The resolving set of a rule r_i, denoted $R(r_i, f)$, in firewall f is the set of all packets that match r_i, but do not match any r_j $(j < i)$ that is listed before r_i in f. The essence of the resolving set $R(r_i, f)$ of a rule r_i in firewall f is: for any packet p in $R(r_i, f)$, the decision of firewall f for packet p is the decision of rule r_i. Note that the matching set of a rule depends only on the rule itself, while the resolving set of a rule depends on both the rule itself and all the rules listed before it in a firewall.

3.3. Theory Foundation

The following four theorems lay the foundation for computing firewall change-impact. In this article, we use $r.D$ to denote the decision of rule r, and $f(p)$ to denote the decision of the first (i.e., highest priority) rule that p matches in firewall f.

THEOREM 1 (RULE DELETION THEOREM). *Let f be a given firewall $\langle r_1, \ldots, r_n \rangle$. Suppose we delete rule r_i, where $1 \le i \le n-1$. Let f' be the resulting firewall $\langle r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n \rangle$. We use g to denote the firewall $\langle r_{i+1}, \ldots, r_n \rangle$, which consists of the $n-i$ rules $r_{i+1}, \ldots, r_n$ after rule r_i in firewall f. For any packet p in Σ, consider the following two cases.*

(1) *If $p \in R(r_i, f)$, then $f(p) = r_i.D$ and $f'(p) = g(p)$, which means that f and f' may have different decisions for p.*
(2) *If $p \in \Sigma - R(r_i, f)$, then $f(p) = f'(p)$, which means that f and f' have the same decision for p.*

PROOF.

(1) $p \in R(r_i, f)$: By the definition of a resolving set, we have $f(p) = r.D$. Because p does not match any of the rules from r_1 to r_{i-1}, the first rule that p matches in f' and the first rule that p matches in g are the same. Therefore, $f'(p) = g(p)$.
(2) $p \in \Sigma - R(r_i, f)$: Because p does not match r_i, the first rule that p matches in f and the first rule that p matches in f' are the same. Therefore, $f(p) = f'(p)$. □

THEOREM 2 (RULE INSERTION THEOREM). *Let f be the given firewall $\langle r_1, \ldots, r_n \rangle$. Suppose we insert a new rule r between r_i and r_{i+1}, where $1 \le i \le n-1$. Let f' be the resulting firewall $\langle r_1, \ldots, r_i, r, r_{i+1}, \ldots, r_n \rangle$. We use g to denote the firewall $\langle r_{i+1}, \ldots, r_n \rangle$, which consists of the $n-i$ rules $r_{i+1}, \ldots, r_n$ after rule r_i in firewall f. For any packet p in Σ, consider the following two cases.*

(1) *If $p \in R(r, f')$, then $f(p) = g(p)$ and $f'(p) = r.D$, which means that f and f' may have different decisions for p.*
(2) *If $p \in \Sigma - R(r, f')$, then $f(p) = f'(p)$, which means that f and f' have the same decision for p.*

PROOF.

(1) $p \in R(r, f')$: By the definition of a resolving set, we have $f'(p) = r.D$. Because p does not match any of the rules from r_1 to r_{i-1}, the first rule that p matches in f and the first rule that p matches in g are the same. Therefore, $f(p) = g(p)$.
(2) $p \in \Sigma - R(r, f')$: Because p does not match r_i, the first rule that p matches in f and the first rule that p matches in f' are the same. Therefore, $f(p) = f'(p)$. □

THEOREM 3 (RULE MODIFICATION THEOREM). *Let f be the given firewall $\langle r_1, \ldots, r_n \rangle$. Suppose we modify rule r_i to be r_i' where $1 \le i \le n-1$. Let f' be the resulting firewall $\langle r_1, \ldots, r_{i-1}, r_i', r_{i+1}, \ldots, r_n \rangle$. We use g to denote the firewall $\langle r_{i+1}, \ldots, r_n \rangle$, which consists of the $n-i$ rules $r_{i+1}, \ldots, r_n$ after rule r_i in firewall f. For any packet p in Σ, consider the following four cases.*

(1) *If $p \in R(r_i, f) \cap R(r_i', f')$, then $f(p) = r_i.D$ and $f'(p) = r_i'.D$.*
(2) *If $p \in R(r_i, f) - R(r_i', f')$, then $f(p) = r_i.D$ and $f'(p) = g(p)$.*

(3) *If* $p \in R(r'_i, f') - R(r_i, f)$*, then* $f(p) = g(p)$ *and* $f'(p) = r'_i.D$.
(4) *If* $p \in \Sigma - R(r_i, f) \cup R(r'_i, f')$*, then* $f(p) = f'(p)$*, which means that* f *and* f' *have the same decision for* p.

PROOF.

(1) $p \in R(r_i, f) \cap R(r'_i, f')$: By the definition of a resolving set, we have $f(p) = r_i.D$ and $f'(p) = r'_i.D$.
(2) $p \in R(r_i, f) - R(r'_i, f')$: By the definition of a resolving set, we have $f(p) = r_i.D$. Because p does match any of the rules from r_1 to r_{i-1}, and also does not match r'_i, the first rule that p matches in f' and the first rule that p matches in g are the same. Therefore, $f'(p) = g(p)$.
(3) $p \in R(r'_i, f') - R(r_i, f)$: By the definition of a resolving set, we have $f'(p) = r'_i.D$. Because p does match any of the rules from r_1 to r_i, the first rule that p matches in f and the first rule that p matches in g are the same. Therefore, $f(p) = g(p)$.
(4) $p \in \Sigma - R(r_i, f) \cup R(r'_i, f')$: Because p matches neither r_i nor r'_i, the first rule that p matches in f and the first rule that p matches in f' are the same. Therefore, $f(p) = f'(p)$. □

THEOREM 4 (RULE SWAP THEOREM). *Let* f *be the given firewall* $\langle r_1, \ldots, r_n \rangle$*. Suppose we swap the two rules* r_i *and* r_j *where* $1 \le i < j \le n-1$*. Let* f' *be the resulting firewall* $\langle r_1, \ldots, r_{i-1}, r_j, r_{i+1}, \ldots, r_{j-1}, r_i, r_{j+1}, \ldots, r_n \rangle$*. Let* g *be the firewall* $\langle r_{i+1}, \ldots, r_n \rangle$*, which consists of the* $n-i$ *rules after rule* r_i *in firewall* f*; and* g' *be the firewall* $\langle r_{i+1}, \ldots, r_{j-1}, r_i, r_{j+1}, \ldots, r_n \rangle$*, which consists of the* $n-i$ *rules after rule* r_j *in firewall* f'*. For any packet* p *in* Σ*, consider the following four cases.*

(1) *If* $p \in R(r_i, f) \cap R(r_j, f')$*, then* $f(p) = r_i.D$ *and* $f'(p) = r_j.D$.
(2) *If* $p \in R(r_i, f) - R(r_j, f')$*, then* $f(p) = r_i.D$ *and* $f'(p) = g'(p)$.
(3) *If* $p \in R(r_j, f') - R(r_i, f)$*, then* $f(p) = g(p)$ *and* $f'(p) = r_j.D$.
(4) *If* $p \in \Sigma - R(r_i, f) \cup R(r_j, f')$*, then* $f(p) = f'(p)$.

PROOF.

(1) $p \in R(r_i, f) \cap R(r_j, f')$: By the definition of a resolving set, we have $f(p) = r_i.D$ and $f'(p) = r_j.D$.
(2) $p \in R(r_i, f) - R(r_j, f')$: By the definition of a resolving set, we have $f(p) = r_i.D$. Because p does match any of the rules from r_1 to r_{i-1}, and also does not match r_j, the first rule that p matches in f' and the first rule that p matches in g' are the same. Therefore, $f'(p) = g'(p)$.
(3) $p \in R(r_j, f') - R(r_i, f)$: By the definition of a resolving set, we have $f'(p) = r_j.D$. Because p does match any of the rules from r_1 to r_i, the first rule that p matches in f and the first rule that p matches in g are the same. Therefore, $f(p) = g(p)$.
(4) $p \in \Sigma - R(r_i, f) \cup R(r_j, f')$: Because p matches neither r_i nor r_j, the first rule that p matches in f and the first rule that p matches in f' are the same. Therefore, $f(p) = f'(p)$. □

4. CHANGE-IMPACT ANALYSIS: ALGORITHMS

In this section, we present algorithms for computing the impact of firewall changes based on the theorems in Section 3. Given a firewall and a proposed change, our change-impact analysis algorithms output a set of so-called impacts. An impact is of the form

$$\langle \textit{predicate} \rangle \rightarrow \langle \textit{old decision} \rangle \ \textit{vs.} \ \langle \textit{new decision} \rangle.$$

$$\begin{array}{lllll} r_1: & F_1 \in [20, 50] & \wedge F_2 \in [1, \quad 70] & \rightarrow \textit{accept} \\ r_2: & F_1 \in [1, \ \ 60] & \wedge F_2 \in [40, 100] & \rightarrow \textit{discard} \\ r_3: & F_1 \in [1, \ \ 100] & \wedge F_2 \in [1, \ \ 100] & \rightarrow \textit{accept} \end{array}$$

Fig. 1. A firewall example.

$$\begin{array}{lllll} E_1: \{ & F_1 \in [20, 50] & \wedge F_2 \in [1, \ \ 70] & \rightarrow \textit{accept} \\ \ \ \ \ \} & & & \\ E_2: \{ & F_1 \in [1, \quad 19] & \wedge F_2 \in [40, 100] & \rightarrow \textit{discard} \\ & F_1 \in [51, \ \ 60] & \wedge F_2 \in [40, 100] & \rightarrow \textit{discard} \\ & F_1 \in [20, \ \ 50] & \wedge F_2 \in [71, 100] & \rightarrow \textit{discard} \\ \ \ \ \ \} & & & \\ E_3: \{ & F_1 \in [1, \quad 19] & \wedge F_2 \in [1, \ \ 39] & \rightarrow \textit{discard} \\ & F_1 \in [51, \ \ 60] & \wedge F_2 \in [1, \ \ 39] & \rightarrow \textit{discard} \\ & F_1 \in [61, \ \ 100] & \wedge F_2 \in [1, \ \ 100] & \rightarrow \textit{discard} \\ \ \ \ \ \} & & & \end{array}$$

Fig. 2. Effective rule set of each rule in Figure 1.

The meaning of an impact is: the decision of the packets that satisfy the predicate is changed from $\langle$*old decision*$\rangle$ to $\langle$*new decision*$\rangle$. For ease of understanding, the predicates of all impacts computed for a change are nonoverlapping.

4.1. Rule Deletion

Based on Theorem 1, to compute the impacts of deleting rule r_i, we first need to compute the resolving set $R(r_i, f)$. We represent the resolving set of a rule by a set of nonoverlapping rules, the union of whose matching sets is exactly the resolving set. This set of nonoverlapping rules is called an effective rule set of that rule [Liu and Gouda 2005]. Let r be a rule in a firewall f. Recall that $M(r)$ denotes the set of all the packets that can match rule r. A set of nonoverlapping rules $\{e_1, e_2, \ldots, e_k\}$ is an *effective rule set* of r if and only if the following three conditions hold: (1) $R(r, f) = \bigcup_{i=1}^{k} M(e_i)$, (2) $M(e_i) \cap M(e_j) = \emptyset$ for $1 \leq i < j \leq k$, (3) every e_i has the same decision as r.

How to compute the effective rule-set for a rule in a firewall has been discussed in our previous work on removing redundant rules in firewalls [Liu and Gouda 2005]. Interested readers can refer to Liu and Gouda [2005] for more technical details. Here we show one example. Considering the example firewall in Figure 1. In this firewall, for simplicity, we assume that each packet has only two fields, F_1 and F_2, and the domain of each field is [1,100]. The effective rule-set of each rule is shown in Figure 2. Note that we use E_i to denote the effective rule set of rule r_i.

As shown in Liu and Gouda [2010], the worst-case complexity of computing effective rule-sets using firewall decision diagrams for a firewall policy is $O(n^d)$, where n is the total number of firewall rules in the policy and d is the number of packet fields examined by the policy. The typical values for d, are 4 (source IP, destination IP, destination port, and protocol type) or 5 (source IP, destination IP, source port, destination port, and protocol type). Proving this worst-case complexity is simple. Considering the root of the reduced firewall decision diagram used for computing effective rule sets. It has at most $2n - 1$ outgoing edges, where each edge is labeled with one integer interval. Note that given n intervals in the domain of a field $[0, 2^{32} - 1]$, $[0, 2^{16} - 1]$, or $[0, 2^{8} - 1]$, the resulting nonoverlapping intervals are at most $2n - 1$. Similarly, each nonterminal node has at most $2n - 1$ outgoing edges, where each edge is labeled with one integer interval. Thus, the total number of paths in the resulting firewall decision diagram is at most $(2n - 1)^d = O(n^d)$. This worst-case complexity shows that our algorithms make

$$
\begin{array}{llll}
r'_1: & F_1 \in [1, \quad 60] & \wedge F_2 \in [40, \ 100] & \rightarrow discard \\
r'_2: & F_1 \in [1, \quad 60] & \wedge F_2 \in [1, \quad 39] & \rightarrow accept \\
r'_3: & F_1 \in [61, 100] & \wedge F_2 \in [1, \quad 100] & \rightarrow accept
\end{array}
$$

Fig. 3. A nonoverlapping firewall.

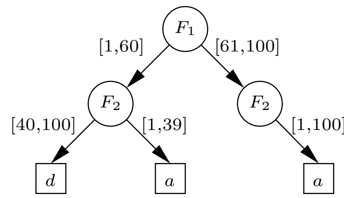

Fig. 4. A firewall decision diagram.

sense only when the rules in a firewall are extremely overlapped, which is not true in practice [Baboescu et al. 2003; Gupta 2000]. In our experiments, computing effective rule-sets is very efficient.

Next, we discuss how to compute the impact of rule deletion through an example. Consider the firewall in Figure 1. Suppose the change is to delete rule r_1, whose effective rule set E_1 is $\{F_1 \in [20, 50] \wedge F_2 \in [1, 70] \rightarrow accept\}$. According to Theorem 1, the question that we need to answer is: which packets that satisfy $F_1 \in [20, 50] \wedge F_2 \in [1, 70]$ are discarded by firewall $\langle r_2, r_3 \rangle$?

To answer this question, we first convert firewall $\langle r_2, r_3 \rangle$ to an equivalent nonoverlapping firewall, as shown in Figure 3. A *nonoverlapping* firewall is a firewall whose rules are nonoverlapping.

Now the question is: which packets that satisfy $F_1 \in [20, 50] \wedge F_2 \in [1, 70]$ are discarded by the firewall in Figure 3? Given that this firewall is nonoverlapping, we can answer this question by checking which discard rule in this firewall overlaps with the predicate $F_1 \in [20, 50] \wedge F_2 \in [1, 70]$. Obviously, the answer is rule r'_1. For the packets that satisfy both predicates $F_1 \in [20, 50] \wedge F_2 \in [1, 70]$ and $F_1 \in [1, 60] \wedge F_2 \in [40, 100]$, the decision by the original firewall is *accept*, but the decision by the modified firewall is *discard*. Therefore, the impact of deleting rule r_1 from the firewall in Figure 2 is as follows.

$$F_1 \in [20, 50] \wedge F_2 \in [40, 70] \rightarrow accept\ vs.\ discard.$$

For efficiency purposes, we represent a nonoverlapping firewall using a firewall decision diagram [Gouda and Liu 2007]. For example, the nonoverlapping firewall in Figure 3 can be represented using the firewall decision diagram in Figure 4.

The pseudocode of the algorithm for computing the impacts of rule deletion is shown in Figure 5. In this paper, we use *t.root* to denote the root of a firewall decision diagram t, $I(e)$ to denote the label of an edge e, $F(v)$ to denote the label of a node v.

4.2. Rule Insertion

According to Theorem 2, computing the impacts of rule insertion is similar to that for rule deletion. Let f be the given firewall $\langle r_1, \ldots, r_n \rangle$. Suppose we insert a new rule r between r_i and r_{i+1}, where $1 \leq i \leq n-1$. Let f' be the resulting firewall $\langle r_1, \ldots, r_i, r, r_{i+1}, \ldots, r_n \rangle$. To compute the impacts of inserting rule r, we first compute the effective rule-set of rule r in firewall f'. Second, we construct a firewall decision diagram for the firewall $\langle r_{i+1}, \ldots, r_n \rangle$. (Note that $\langle r_{i+1}, \ldots, r_n \rangle$ is complete because the last rule r_n can match any given packet.) Third, we traverse the firewall decision diagram to check which decision path conflicts with a rule in the effective rule set of

Computing Impacts of Rule Deletion
Input : A firewall $\langle r_1, \cdots, r_n \rangle$.
Output Change-impacts of deleting rule r_i.
Steps:
1. Compute the effective rule set E_i of rule r_i;
   Let E_i be $\{e_1, \cdots, e_m\}$.
   $Impacts := \emptyset$;
2. Construct a firewall decision diagram t from $\langle r_{i+1}, \cdots, r_n \rangle$;
3. **for** $i := 1$ **to** m **do** **Compare**($t.root,\ e_i$);
   return E;

Compare($v,\ (F_1 \in S_1) \wedge \cdots \wedge (F_d \in S_d) \rightarrow \langle dec \rangle)$)
/*Let $(F_1 \in S_1') \wedge \cdots \wedge (F_d \in S_d') \rightarrow F(v)$ be the rule defined by the decision path containing v;*/
1. **if** (v is a terminal node) and ($\langle dec \rangle \neq F(v)$)
   then $Impacts := Impacts \cup \{(F_1 \in S_1 \cap S_1') \wedge \cdots \wedge (F_d \in S_d \cap S_d') \rightarrow \langle dec \rangle\ vs.\ F(v)\}$;
2. **if** (v is a nonterminal node) **then**
   /*Let F_j be the label of v*/
   for each edge e in $E(v)$ **do**
     if $I(e) \cap S_j \neq \emptyset$ **then**
       Compare($e.t,\ (F_1 \in S_1) \wedge \cdots \wedge (F_d \in S_d) \rightarrow \langle dec \rangle)$)

Fig. 5. Computing impacts of rule deletion.

r. For each conflict discovered, we output an impact. Due to space limitations, we omit the details of the algorithm for computing the impacts of rule insertion.

4.3. Rule Modification

Based on Theorem 3, to compute the impact of modifying rule r_i in firewall f to be r_i', we only need to consider the packets in the three sets: $R(r_i, f) \cap R(r_i', f')$, $R(r_i, f) - R(r_i', f')$, and $R(r_i', f') - R(r_i, f)$, where f' is the resulting firewall $\langle r_1, \ldots, r_{i-1}, r_i', r_{i+1}, \ldots, r_n \rangle$, because the decisions for the rest of the packets (those in $\Sigma - R(r_i, f) \cup R(r_i', f')$ remain unchanged. Thus we need to know how to compute $R(r_i, f) \cap R(r_i', f')$ and $R(r_i, f) - R(r_i', f')$, where f' denotes the firewall after modifying r_i. Because of symmetry, the algorithms for computing $R(r_i, f) - R(r_i', f')$ and $R(r_i', f') - R(r_i, f)$ are the same. Next, we discuss how to compute them.

Given two resolving sets R_a and R_b, which are represented by the effective rule-sets $\{e_1, \ldots, e_m\}$ and $\{\varepsilon_1, \ldots, \varepsilon_l\}$ respectively. Then we have $R_a \cap R_b = \cup_{i=1}^{m} \cup_{j=1}^{l} (M(e_i) \cap M(\varepsilon_j))$. Note that $M(e_i) \cap M(\varepsilon_j)$ can be computed as follows. Let rule e_i be $(F_1 \in S_1) \wedge \cdots \wedge (F_d \in S_d) \rightarrow \langle decision \rangle$ and rule ε_j be $(F_1 \in S_1') \wedge \cdots \wedge (F_d \in S_d') \rightarrow \langle decision \rangle$. Let rule r be $(F_1 \in S_1 \cap S_1') \wedge \cdots \wedge (F_d \in S_d \cap S_d') \rightarrow \langle decision \rangle$. Then we have $M(e_i) \cap M(\varepsilon_j) = M(r)$.

Given two resolving sets R_a and R_b, which are represented by the effective rule-sets $\{e_1, \ldots, e_m\}$ and $\{\varepsilon_1, \ldots, \varepsilon_l\}$ respectively. Let r be the rule that any packet can match and f be the firewall $\langle \varepsilon_1, \ldots, \varepsilon_l, e_1, \ldots, e_m, r \rangle$. Then we have $R_a - R_b = \cup_{i=1}^{m} R(e_i, f)$. In other words, $R_a - R_b$ is the union of the effective rule set of every e_i in firewall $\langle \varepsilon_1, \ldots, \varepsilon_l, e_1, \ldots, e_m, r \rangle$.

Let f be the given firewall $\langle r_1, \ldots, r_n \rangle$. Suppose we modify rule r_i to be r_i', where $1 \leq i \leq n-1$. Let f' be the resulting firewall $\langle r_1, \ldots, r_{i-1}, r_i', r_{i+1}, \ldots, r_n \rangle$. The change-impacts of modifying rule r_i can be computed in the following steps.

(1) Compute the effective rule-set of rule r_i in firewall f, and that of rule r_i' in firewall f'.
(2) If r_i and r_i' have the same decision, then skip this step. Otherwise, compute $R(r_i, f) \cap R(r_i', f')$. If $R(r_i, f) \cap R(r_i', f') \neq \emptyset$, then generate impacts accordingly. Note that the decision for any packet in $R(r_i, f) \cap R(r_i', f')$ is changed from the decision of r_i to that of r_i'.
(3) Compute $R(r_i, f) - R(r_i', f')$ as follows. Let $\{e_1, \ldots, e_m\}$ and $\{\varepsilon_1, \ldots, \varepsilon_l\}$ be the effective rule-sets of rule r_i in firewall f and rule r_i' in firewall f' respectively.

Compute the effective rule-sets of $e_1, \ldots, e_m$ in firewall $\langle \varepsilon_1, \ldots, \varepsilon_l, e_1, \ldots, e_m, r \rangle$ where r is a rule that any packet can match. Let U be the union of these effective rule-sets. Then U represents $R(r_i, f) - R(r_i', f')$.

(4) Construct a firewall decision diagram from firewall $\langle r_{i+1}, \ldots, r_n \rangle$.

(5) Traverse the firewall decision diagram to check which decision path conflicts with a rule in U. Whenever a conflict is found, our tool outputs an impact.

(6) Compute $R(r_i', f') - R(r_i, f)$ by computing the effective rule-sets of $\varepsilon_1, \ldots, \varepsilon_l$ in firewall $\langle e_1, \ldots, e_m, \varepsilon_1, \ldots, \varepsilon_l, r \rangle$, where r is a rule that any packet can match. Let U' be the union of these effective rule sets. Then U' represents $R(r_i', f') - R(r_i, f)$.

(7) Traverse the firewall decision diagram built from firewall $\langle r_{i+1}, \ldots, r_n \rangle$ to check which decision path conflicts with a rule in U'. Whenever a conflict is found, our tool outputs an impact.

4.4. Rule Swap

Based on Theorem 4, computing the impacts of swapping two rules is similar to that of rule modification. Due to space limitations, we omit the details of the algorithm for computing the impacts of rule swap.

5. DISCUSSION

5.1. Prefix and Intervals

Real-life firewalls usually check five packet fields: source IP address, destination IP address, source port number, destination port number, and protocol type. Of these five fields, the first two are usually represented using prefix formats, and the last three are usually represented using integer intervals. Note that prefix formats and interval formats are interchangeable. For example, IP prefix 192.168.0.0/16 can be converted to the interval from 192.168.0.0 to 192.168.255.255, where an IP address can be regarded as a 32-bit integer. As another example, the interval [2, 8] can be converted to 3 prefixes: $001*$, $01*$, 1000.

To compute firewall change-impacts, we first convert the source and destination IP addresses from prefix formats to integer intervals. Note that every prefix can be converted to only one integer interval. Second, we run the algorithms described in Section 4 for computing firewall change-impacts. (Note that the impacts produced by our algorithms are in interval formats.) Third, for each impact computed, we convert the source and destination IP addresses from intervals to prefixes. Thus, the format of outputs is similar to that of original firewall rules, which are easy to understand for firewall administrators. (A $w-$bit integer interval can be converted to at most $2w - 2$ prefixes [Gupta and McKeown 2001].)

5.2. Making Corrections

After the impacts of a change are computed, the firewall administrator needs to verify that the impacts are indeed intended. If not all impacts are desirable, one approach is to revise the proposed change and compute impacts again; another approach for the firewall administrator is to commit desired impacts by correcting undesired impacts. Next, we show an example to illustrate the latter approach.

Consider the two impacts in Table IV. If the first impact is exactly what the administrator intends to do, and the second impact is not desired, we can keep the proposed change and add the following rule derived from the second (undesired) impact to the beginning of the modified firewall.

Src IP	Dest. IP	Src Port	Dest. Port	Protocol	Action
1.2.3.4	192.168.0.1	*	[25,65536]	TCP	discard

5.3. Time and Space Complexity Analysis

Let n be the number of rules in a firewall, and d be the total number of distinct packet fields that are examined by a firewall. The time and space complexity of our change-impact analysis algorithms is $O(n^d)$. Despite the high worst-case complexities, our algorithms are practical for two reasons. First, d is bounded and is typically small. Real-life firewalls typically examine five packet fields: source IP address, destination IP address, source port number, destination port number, and protocol type. Second, the worst cases of our algorithms are extremely unlikely to happen in practice. To trigger the worst case, the rules in a firewall need to be exceedingly overlapping, which does not happen in real-life firewalls according to the statistics on real-life rule-sets in Gupta [2000].

5.4. Change-Impact Analysis of Multiple Firewalls

Given the policy of a single firewall, the algorithms presented here can be used to compute its change-impact. This is the most common scenario of firewall change-impact analysis because the access control policy in a firewall is typically available only to its administrator. Where the policies of multiple interconnected firewalls are available to a central authority, with the help of the algorithms for computing network reachability proposed by Khakpour and Liu [2010], our algorithms can be used to compute the change-impact of multiple firewalls as well.

Given a network with multiple firewalls, for any two subnets A and B, the algorithms in Khakpour and Liu [2010] compute all the packets that traverse from A to B as follows. Let $p_1, p_2, \ldots, p_n$ be all the paths from A to B. For each path p_i $(1 \leq i \leq n)$, suppose p_i contains k firewalls $f_1, f_2, \ldots, f_k$, they compute all the packets that can be accepted by all the firewalls on this path, denoted $A(p_i)$, by intersecting the set of packets that can be accepted by each firewall f_i, denoted $A(f_i)$; that is, $A(p_i) = A(f_1) \cap A(f_2) \cap \cdots \cap A(f_k)$. Finally, all the packets from A to B can be computed as $A(p_1) \cup A(p_2) \cup \cdots \cup A(p_n)$.

To compute the change-impact of multiple firewalls, we first need to identity all the paths from one subnet to another where the firewall policies on the path are changed. Only the reachability of these subnet pairs is potentially changed. Consider path p with k firewalls $f_1, f_2, \ldots, f_k$, where f_i $(1 \leq i \leq k)$ is changed to f_i'. Using the algorithms presented in this article, we can compute two sets, S_{ad}, denoting all the packets that are accepted by f_i but discarded by f_i', and S_{da} denoting all the packets that are discarded by f_i but accepted by f_i'. Then, the set $A(f_1) \cap \cdots A(f_{i-1}) \cap S_{ad} \cap A(f_{i+1}) \cap \cdots \cap A(f_k)$ is the set of all packets that path p accepts before changing f_i to f_i' but discards after the change, and the set $A(f_1) \cap \cdots A(f_{i-1}) \cap S_{da} \cap A(f_{i+1}) \cap \cdots \cap A(f_k)$ is the set of all packets that path p discards before changing f_i to f_i' but accepts after the change.

5.5. Infeasibility of BDD Based Change-Impact Analysis

Some prior work (such as Yuan et al. [2006]) has used a Binary Decision Diagram (BDD) [Bryant 1986] for analyzing firewall policies. A BDD is a rooted, directed, acyclic graph that represents a Boolean function. In a BDD, each nonterminal node is labeled by a Boolean variable and it has only two outgoing edges, labeled 0 and 1 respectively. Each edge represents an assignment of 0 or 1. A BDD has only two terminal nodes, labeled 0 and 1 respectively. BDDs are efficient for some firewall policy analysis functions such as verifying whether a firewall policy satisfies an access control requirement (such as server A must be able to communicate with server B via port 8000), which can be easily implemented by first converting the firewall policy to BDD b_1 and the access control requirement to another BDD b_2 and then testing whether logically b_1 implies b_2. However, BDDs are not suitable for computing the change-impact of firewall policies because the computing result is not human-readable. First, the computing result, which is a BDD, is not human-readable because every node in a BDD represents only a bit of a packet, not a field of a packet Using BDDs to compute the change-impact of a firewall policy, which is represented as a BDD, is simple. Suppose we change a firewall f into f'. We can represent the set of all packets that f accepts using a BDD denoted B_a and the set of all packets that f discards using a BDD denoted B_d. Similarly, we can represent f' using two BDDs B'_a and B'_d. Therefore, $B_a - B'_a$ represents all the packets that f accepts but f' discards, and $B_d - B'_d$ represents all the packets that f discards but f' accepts. Although computing $B_a - B'_a$ and $B_d - B'_d$ is simple, interpreting the meaning is nearly impossible for a firewall administrator. Second, if we generate human-readable rules from $B_a - B'_a$ and $B_d - B'_d$, it will typically result in an excessive number of rules, in terms of even millions. We have implemented a BDD-based solution using CUDD package [Somenzi 2009]. Unfortunately, even computing the change-impact of a small firewall often results in millions of rules. While compressing millions of rules may not be impossible, it is by no means simple and there are no known effective methods for this purpose. In contrast, using FDDs, we can easily compute human-readable firewall change-impact in a rule-like format.

5.6. Complexity of Computing Change-Impact Analysis Using Logical Minuses among Firewall Rules

Note that the complexity of directly computing effective rule-sets using logical minus operations, without firewall decision diagrams, is $O(d^n)$ (exponential over the number of original rules) and therefore is computationally infeasible. This complexity is calculated as follows. Let the first rule in the original firewall be $(F_1 \in S_{11}) \wedge (F_2 \in S_{12}) \wedge \cdots \wedge (F_d \in S_{1d}) \rightarrow dec_1$ and the second rule be $(F_1 \in S_{21}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d}) \rightarrow dec_2$. The effective rule-set of the second rule can be calculated by $((F_1 \in S_{21}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d})) - ((F_1 \in S_{11}) \wedge (F_2 \in S_{12}) \wedge \cdots \wedge (F_d \in S_{1d})) \rightarrow dec_2$. Using De Morgan's law, we have $((F_1 \in S_{21}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d})) - ((F_1 \in S_{11}) \wedge (F_2 \in S_{12}) \wedge \cdots \wedge (F_d \in S_{1d})) = ((F_1 \in S_{21}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d})) \wedge \overline{(F_1 \in S_{11}) \wedge (F_2 \in S_{12}) \wedge \cdots \wedge (F_d \in S_{1d})} = ((F_1 \in S_{21}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d})) \wedge ((F_1 \in \overline{S_{11}}) \vee (F_2 \in \overline{S_{12}}) \vee \ldots \vee (F_d \in \overline{S_{1d}})) = ((F_1 \in S_{21} \cap \overline{S_{11}}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d})) \vee ((F_1 \in S_{21}) \wedge (F_2 \in S_{22} \cap \overline{S_{12}}) \wedge \cdots \wedge (F_d \in S_{2d})) \vee \ldots \vee ((F_1 \in S_{21}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d} \cap \overline{S_{1d}}))$. Note that for each $1 \leq i \leq d$, $S_{2i} \cap \overline{S_{1i}}$ is one interval or the union of two nonadjacent intervals. Thus, the nonatomic rule $((F_1 \in S_{21} \cap \overline{S_{11}}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d})) \vee ((F_1 \in S_{21}) \wedge (F_2 \in S_{22} \cap \overline{S_{12}}) \wedge \cdots \wedge (F_d \in S_{2d})) \vee \ldots \vee ((F_1 \in S_{21}) \wedge (F_2 \in S_{22}) \wedge \cdots \wedge (F_d \in S_{2d} \cap \overline{S_{1d}})) \rightarrow dec_2$ can be decomposed into at most $2d$ atomic rules. Similarly, rewriting the j-th rule will end up with at most $(2d)^{j-1}$ atomic rules. Therefore, converting an overlapping firewall of n rules to an equivalent nonoverlapping firewall directly using logical operations will result in $\frac{(2d)^n-1}{2d-1} = O(d^n)$ nonoverlapping rules. Although d is a constant (of typically 4 or 5), n can be very large (on the order of thousands for enterprise firewalls).

6. EFFECTIVENESS EVALUATION

The purpose of our firewall policy change-impact analysis tool is to help administrators to ensure the correctness of their firewall policies in making changes. Our tool to a firewall administrator is like a debugger to a programmer. To evaluate the effectiveness of our tool, we conducted a case study on a real-world firewall policy with its administrator. This firewall policy was deployed on a campus network. A key challenge in this case study is to collect firewall policy changes. To address this challenge, we simulate past firewall policy changes by incrementally adding rules. In practice, firewall policies often grow in a bottom up fashion. With IP addresses and port numbers anonymized due to privacy and security concerns, this firewall policy with 87 rules is shown in

Table V. The Real Firewall Policy with Rules 52 to 87

#	Src IP	Dest IP	Src Port	Dest Port	Protocol	Decision
52	*	*	*	6667	TCP	discard
⋮	⋮	⋮	⋮	⋮	⋮	⋮
87	*	*	*	*	IP	accept

Table VI. The Real Firewall Policy with Rules 51 to 87

#	Src IP	Dest IP	Src Port	Dest Port	Protocol	Decision
51	62.78.103.*	*	*	*	IP	discard
52	*	*	*	6667	TCP	discard
⋮	⋮	⋮	⋮	⋮	⋮	⋮
87	*	*	*	*	IP	accept

Table VII. Difference Between the Two Real Firewall Policies in Tables V and VI

Src IP	Dest IP	Src Port	Dest Port	Protocol	Decision in V	Decision in VI
62.78.103.*	157.96.128.*	*	*	IP	accept	discard

the Appendix. Starting from the last default rule, we add one rule at a time. Each time after adding the new rule, we use our change-impact analysis tool to compute the impact and ask the administrator to verify whether the firewall policy after the change is correct at that stage. More specifically, let $\langle r_1, r_2, \ldots, r_{87}\rangle$ be the firewall in this case study. First, we compute the semantic difference between policies $\langle r_{87}\rangle$ and $\langle r_{86}, r_{87}\rangle$, then ask the administrator whether adding rule r_{86} was correct when the firewall policy has only rule r_{87}. Second, we compute the semantic difference between policies $\langle r_{86}, r_{87}\rangle$ and $\langle r_{85}, r_{86}, r_{87}\rangle$ and ask the administrator whether adding rule r_{85} was correct when the firewall policy has rules r_{86} and r_{87}. This process continues until we finish adding all the rules. While this experimental setup in our case study may not include all the changes that happened to this firewall policy, our firewall policy change-impact analysis tool still was able to help the administrator to identify some errors latent in this policy. These errors fall into two categories: errors discarding legitimate traffic and errors accepting illegitimate traffic. Beyond these two categories of errors, our tool also helped the administrator to identify some redundant rules. A rule is redundant in a firewall policy if and only if removing the rule from the policy does not change the semantics of the policy. Next, we present the errors and redundancy identified in this case study, using our tool.

6.1. Errors Discarding Legitimate Traffic

Our change impact analysis tool helped the administrator to identify that rule 51 blocks some legitimate packets by comparing the two real firewall policies in Tables VI and V. The change impact before and after adding rule 51 is shown in Table VII, where Columns 6 and 7 show the decisions made by these two firewall policies, respectively. Note that 157.96.128.* in Table X is a set of IP addresses that are used for providing public services to customers, whose source IP addresses can be any possible addresses. However, after adding rule 51, none of the packets that match the predicates in Table VII can pass through the firewall, which could disrupt the public services provided to the customers.

6.2. Errors Accepting Illegitimate Traffic

Our change impact analysis tool helped the administrator to identify that rule 6 allows some illegitimate packets by comparing the two real firewall policies in Tables VIII and IX. The change impact before and after adding rule 6 is shown in Table X, where Columns 6 and 7 show the decisions made by these two firewall policies, respectively.

Table VIII. The Real Firewall Policy with Rules 7 to 87

#	Src IP	Dest IP	Src Port	Dest Port	Protocol	Action
7	32.45.186.83	*	*	*	IP	discard
⋮	⋮	⋮	⋮	⋮	⋮	⋮
87	*	*	*	*	IP	accept

Table IX. The Real Firewall Policy with Rules 6 to 87

#	Src IP	Dest IP	Src Port	Dest Port	Protocol	Action
6	*	157.96.252.66	*	*	IP	accept
7	32.45.186.83	*	*	*	IP	discard
⋮	⋮	⋮	⋮	⋮	⋮	⋮
87	*	*	*	*	IP	accept

Table X. Difference Between the Two Real Firewall Policies in Tables IX and VIII

Src IP	Dest IP	Src Port	Dest Port	Protocol	Decision in VIII	Decision in IX
32.45.186.83	157.96.252.66	*	*	IP	discard	accept
231.49.182.251	157.96.252.66	*	*	IP	discard	accept
0.0.0.0	157.96.252.66	*	*	IP	discard	accept
157.96.119.*	157.96.252.66	*	*	IP	discard	accept
157.96.120.*	157.96.252.66	*	*	IP	discard	accept
157.96.121.*	157.96.252.66	*	*	IP	discard	accept
157.96.122.*	157.96.252.66	*	*	IP	discard	accept
157.96.130.*	157.96.252.66	*	*	IP	discard	accept
157.96.138.*	157.96.252.66	*	*	IP	discard	accept
157.96.139.*	157.96.252.66	*	*	IP	discard	accept
157.96.143.*	157.96.252.66	*	*	IP	discard	accept
157.96.144.*	157.96.252.66	*	*	IP	discard	accept
157.96.158.*	157.96.252.66	*	*	IP	discard	accept
157.96.252.*	157.96.252.66	*	*	IP	discard	accept
178.95.49.*	157.96.252.66	*	*	IP	discard	accept
32.45.186.83	157.96.252.66	*	*	IP	discard	accept
62.78.103.*	157.96.252.66	*	*	IP	discard	accept
255.255.255.255	157.96.252.66	*	*	IP	discard	accept

Table XI. The Real Firewall Policy with Rules 79 to 87

#	Src IP	Dest IP	Src Port	Dest Port	Protocol	Action
79	*	157.96.138.101	*	*	IP	deny
78	*	157.96.138.101	*	5166	TCP	accept
⋮	⋮	⋮	⋮	⋮	⋮	⋮
87	*	*	*	*	IP	accept

Note that 32.45.186.83 and 231.49.182.251 in the first two rows of Table X are two malicious IP addresses, and the administrator should block the packets from these two IP addresses. However, after adding rule 6, all the packets that match the predicates in the first two rows of Table X can pass through the firewall, which opens a security hole in the network protected by the firewall.

6.3. Redundant Rules

Our change impact analysis tool helped the administrator to identify that rules 74, 75, 77, and 78 are redundant in the real firewall policy. As an example, we show that rule 78 is redundant. When comparing the policies in Tables XI and XII, our change impact analysis tool shows that there is no difference between these two policies. In other words, adding rule 78 does not affect the functionality of the policy in Table XI.

Table XII. The Real Firewall Policy with Rules 78 to 87

#	Src IP	Dest IP	Src Port	Dest Port	Protocol	Action
78	*	157.96.138.101	*	5166	TCP	accept
⋮	⋮	⋮	⋮	⋮	⋮	⋮
87	*	*	*	*	IP	accept

Table XIII. Performance of Change-Impact Analysis Algorithms on Two Real-Life Firewalls

	# Rules	Deletion/Insertion	Modification	Swap
Firewall I	1298	1844 ms	3101 ms	6218 ms
Firewall II	3542	750 ms	1250 ms	2672 ms

Note that because of the first-match semantics, removing rule 78 does not affect the functionality of rules 1 to 77.

7. EFFICIENCY EVALUATION

We conducted extensive performance evaluation of the algorithms presented in this article, namely the four firewall change-impact analysis algorithms of rule deletion, insertion, modification, and swap. We implemented these algorithms in Java JDK 1.6. First, we ran our algorithms on two real-life firewalls. Second, we stress-tested our algorithms on a large number of synthetic firewalls. These experiments were carried out on a SUN workstation running Debian Linux with a 2.2Ghz CPU and 1 GB memory. In both cases, the experimental results show that our algorithms perform and scale well.

We obtained two real-life firewalls of relatively large sizes for this study. One firewall is from a university, and it has 1298 rules. The other firewall is from a private company, and it has 3542 rules. For each firewall and each type of change, we randomly generate the changes and use our algorithms to compute the impact of each change. Table XIII shows the performance of our algorithms on these two firewalls.

Firewall configurations are considered confidential due to security concerns. To further evaluate the performance of our algorithms on large firewalls, we run our algorithms on synthetic firewalls of large sizes. The predicate of each rule in our synthetic firewalls has five fields: source IP, destination IP, source port, destination port, and protocol. We based our generation method upon the model of synthetic rules of Rovniagin and Wool [2004].

In our experiments, we generate 100 firewalls for each fixed size. For each firewall, we run each of our algorithms 100 times. For the change-impact analysis algorithm for rule deletion, we randomly pick one rule to delete, and then compute the deletion impact. The evaluation of the change-impact analysis algorithms for rule insertion, modification, and swap is done in a similar random fashion.

Figure 6 shows the average running time of our change-impact analysis algorithm for rule deletion/insertion with standard deviation. Note that our algorithms for computing the impact of rule deletion or insertion take a similar amount of time. Figures 7 and 8 show the average running time of our change-impact analysis algorithm for rule modification and swapping, respectively, with standard deviation. These figures show that our change-impact analysis algorithms scale well for firewalls of large size. For a firewall that has 3000 rules, any of the four change-impact analysis algorithms takes less than 2 seconds. From these experimental results, we also observe that the change-impact analysis algorithms for rule deletion and insertion are faster than the algorithm for rule modification, and the algorithm for rule modification is faster than that for rule swap. These observations conform to the Theorems 1, 2, 3, and 4.

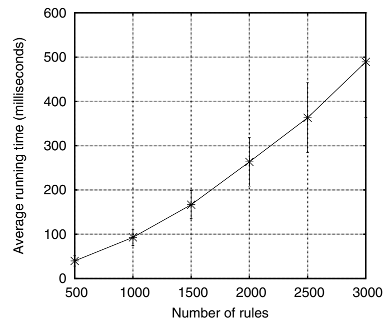

Fig. 6. Performance of change-impact analysis algorithm for rule deletion/insertion.

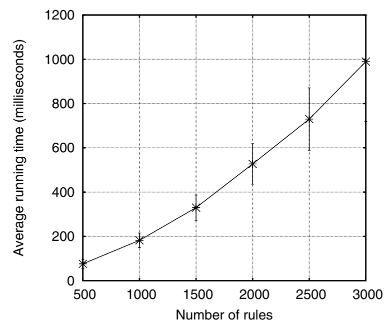

Fig. 7. Performance of change-impact analysis algorithm for rule modification.

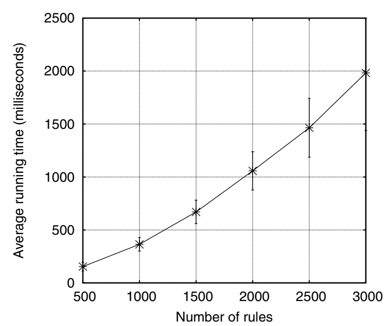

Fig. 8. Performance of change-impact analysis algorithm for rule swapping.

8. RELATED WORK

Extensive studies have been done on analyzing the change-impact of general programs in software engineering and programming language communities (e.g., Horwitz [1990], Bohner and Arnold [1996], Lee et al. [2000], Kung et al. [1994], Tonella [2003], Rajlich and Gosavi [2004], and Ren et al. [2006]). However, little work has been done on analyzing the change-impact of firewall policies. Firewall policies and general programs are fundamentally different. While accurately and completely computing the impact of software changes is nearly impossible in general, the algorithms presented in this article can compute the accurate and complete impact of firewall policy changes.

Fisler et al. [2005] studied change-impact analysis of access control policies in their seminal paper. They proposed a solution called Margrave, which uses multiterminal binary decision diagrams to compute the impact of access control policy changes and verify whether an access control policy satisfies a given property. Their work shares similar goals with our work; however, applying their solution to firewall policy change-impact analysis will result in an MTBDD with $2^{33} = 12$giga variables. In Fisler et al. [2005], every attribute-value pair is encoded as one variable in the MTBDD. For example, given an access control policy with two rules, `professor can write grades` and `student can read grades`, Margrave will encode this policy using five Boolean variables corresponding to five by attribute-value pairs: (1) subject is a professor, (2) subject is a student, (3) object is the grade, (4) action is write, and (5) action is read. Margrave makes sense for application access control policies; however, it is impractical for firewall policies because it always requires to building an MTBDD with 12G variables for even small firewall policies.

Some firewall analysis methods have been proposed in Liu et al. [2004], Liu and Gouda [2009], Liu [2008, 2009], Gouda et al. [2008], Liu and Gouda [2005, 2010], Hazelhurst et al. [2000], Eronen and Zitting [2001], Baboescu and Varghese [2002], Eppstein and Muthukrishnan [2001], Hari et al. [2000], and Moffett and Sloman [1994]. However, none of them deals with the change-impact analysis of firewall policies. In Liu and Gouda [2009] proposed a firewall query language and firewall query processing algorithms. Firewall queries are questions concerning the function of a firewall. An example of a firewall query is "Which computers in the private network can receive packets from a known malicious host in the outside Internet?" Liu and Gouda [2005, 2010] studied the redundancy issues in firewall policies and gave an algorithm for removing all the redundant rules in a firewall policy. In Hazelhurst et al. [2000], some ad hoc "what if" questions that are similar to firewall queries were discussed. In Eronen and Zitting [2001], expert systems were proposed to analyze firewall rules. Detecting potential firewall policy errors by conflict detection was discussed in Baboescu and Varghese [2002], Eppstein and Muthukrishnan [2001], Hari et al. [2000], and Moffett and Sloman [1994].

Similar to conflict detection, some anomalies were defined and techniques for detecting anomalies were presented in Al-Shaer and Hamed [2004] and Yuan et al. [2006]. Note that such anomaly detection algorithms fundamentally differ from our change-impact analysis. First, the purposes are different. In firewall policy anomaly detection, the purpose is to detect whether a firewall policy contains some predefined anomalies. For example, in a firewall policy $f = \langle r_1, \ldots, r_n\rangle$, for any two rules r_i and r_j, where $i < j$, if the matching set of r_i is a superset of the matching set of r_j ($M(r_i) \supseteq M(r_j)$), then this is called a shadowing anomaly according to the definition in Al-Shaer and Hamed [2004]. In firewall policy change-impact analysis, the purpose is to compute the exact impact of a policy change. In essence, firewall policy anomaly detection concerns the correctness of a firewall policy itself whereas firewall policy change-impact analysis concerns the correctness of a firewall policy change. Second, the inputs are different. The input to a firewall policy anomaly detection algorithm is a firewall policy. The inputs to a firewall policy change-impact analysis algorithm are a firewall policy and a change. Third, the algorithms are different. Fourth, the outputs are different. The outputs of a firewall policy anomaly detection algorithm are all the cases that satisfy some predefined anomaly conditions. The outputs of a firewall policy change-impact analysis algorithm are all the packets whose decisions are changed due to the policy change and these packets are represented in human-readable rule format.

Liu [2008, 2009] presented an algorithm for verifying firewall policies. The verification of distributed firewalls is studied in Gouda et al. [2008]. Note that firewall verification methods only give an answer of yes when a firewall policy satisfies an access control requirement (such as server A must be able to communicate with server B via ports ranging from 8000 to 9000) or not otherwise. Such a yes/no answer is not what we want in firewall policy change-impact analysis. Given a firewall f and an access control requirement, we can use firewall verification methods to verify whether firewall f satisfies the access control requirement. After we change firewall f to f', indeed, we can use the same verification method to verify whether f' satisfies the access control requirement. However, such verification methods cannot tell what the exact change-impact is. For example, if firewall f allows server A to communicate with server B via ports ranging from 8000 to 9000, but firewall f' after the change only allows server A to communicate with server B via ports ranging from 8000 to 8500, firewall verification methods can only tell us that the access control requirement of server A being able to communicate with server B via ports ranging from 8000 to 9000 is not satisfied.

Some firewall design methods have been proposed in Bartal et al. [1999], Guttman [1997], Liu and Gouda [2004, 2008], Gouda and Liu [2005, 2007]. These works aim at creating firewall rules, while we aim at analyzing firewall rules. Gouda and Liu [2007] and Liu and Gouda [2004] proposed using decision diagrams for designing firewalls. Gouda and Liu [2005] also proposed a model for specifying stateful firewall policies. Guttman [1997] proposed a Lisp-like language for specifying high-level packet filtering policies. Bartal et al. [1999] proposed a UML-like language for specifying global filtering policies. Liu and Gouda [2004, 2008] applied the technique of design diversity to firewall design.

There is previous work on firewall testing [Hoffman et al. 2003; Jürjens and Wimmel 2001; Hoffman and Yoo 2005; Senn et al. 2005; Lyu and Lau 2000]. These testing techniques involve injecting packets into a firewall and checking whether the decisions of the firewall concerning the injected packets are correct. There are some tools currently available for network vulnerability testing, such as Satan and Nessus. These vulnerability testing tools scan a private network based on the current publicly known attacks, rather than the requirement specification of a firewall. Although these tools can possibly catch some of the errors that allow illegitimate access to the private networks, they cannot find the errors that disable legitimate communication between the private network and the outside Internet. Methods for testing firewall policies are studied in Hwang et al. [2008].

9. CONCLUSIONS

Making changes to firewall policies is a major task that firewall administrators perform; yet, it is also a major source of firewall policies errors. To address this issue, in this article, we propose a framework for conducting firewall policy change-impact analysis. Our contributions are three-fold. First, we lay the theory foundation for firewall change-impact analysis. Second, we present algorithms for computing firewall policy change-impacts. Third, we present methods for correlating the impact of a firewall policy change and the high level security requirements that the firewall needs to satisfy as well as methods for making corrections if the impact of a change is not desirable. Our algorithms can be practically used in the iterative process of firewall policy design and maintenance.

APPENDIX

Below is the anonymized version of the real-world firewall policy with 87 rules used in our case study.

#	Src IP	Dest IP	Src Port	Dest Port	Protocol	Action
1	67.54.138.163	157.96.119.153	*	9100	TCP	accept
2	67.54.138.163	157.96.119.153	*	161	UDP	accept
3	*	*	*	*	53	deny
4	*	*	*	*	55	deny
5	*	*	*	*	77	deny

6	*	157.96.252.66	*	*	IP	accept
7	32.45.186.83	*	*	*	IP	deny
8	*	157.96.139.14	*	443	TCP	deny
9	231.49.182.251	*	*	*	IP	deny
10	*	*	*	3127	TCP	deny
11	*	*	*	2745	TCP	deny
12	*	*	4000	*	UDP	deny
13	*	*	*	111	UDP	deny
14	*	*	*	111	TCP	deny
15	*	*	*	2049	UDP	deny
16	*	*	*	2049	TCP	deny
17	*	*	*	7	UDP	deny
18	*	*	*	7	TCP	deny
19	*	*	*	6346	TCP	deny
20	*	*	*	7000	TCP	deny
21	*	*	*	161	UDP	deny
22	*	*	*	162	UDP	deny
23	*	*	*	1993	UDP	deny
24	*	*	*	67	UDP	deny
25	*	*	*	68	UDP	deny
26	*	*	*	49	UDP	deny
27	178.95.49.*	*	*	*	IP	deny
28	157.96.119.*	*	*	*	IP	deny
29	157.96.120.*	*	*	*	IP	deny
30	157.96.121.*	*	*	*	IP	deny
31	157.96.122.*	*	*	*	IP	deny
32	157.96.130.*	*	*	*	IP	deny
33	157.96.138.*	*	*	*	IP	deny
34	157.96.139.*	*	*	*	IP	deny
35	157.96.143.*	*	*	*	IP	deny
36	157.96.144.*	*	*	*	IP	deny
37	157.96.158.*	*	*	*	IP	deny
38	157.96.252.*	*	*	*	IP	deny
39	*	157.96.139.9	*	1949	UDP	accept
40	*	157.96.139.10	*	1949	UDP	accept
41	*	157.96.120.2	*	1949	UDP	accept
42	*	157.96.139.9	*	1949	TCP	accept
43	*	157.96.139.10	*	1949	TCP	accept
44	*	157.96.120.2	*	1949	TCP	accept
45	255.255.255.255	*	*	*	IP	deny
46	0.0.0.0	*	*	*	IP	deny
47	*	157.96.119.*	*	*	IP	deny
48	*	157.96.139.10	*	109	TCP	accept
49	*	157.96.139.10	*	110	TCP	accept
50	*	157.96.139.10	*	143	TCP	accept
51	62.78.103.*	*	*	*	IP	deny
52	*	*	*	6667	TCP	deny
53	*	*	*	6112	TCP	deny
54	*	*	*	109	TCP	deny
55	*	*	*	110	TCP	deny
56	*	*	*	1433	UDP	deny
57	*	*	*	1434	UDP	deny
58	*	*	*	135	TCP	deny
59	*	*	*	137	TCP	deny
60	*	*	*	138	TCP	deny
61	*	*	*	139	TCP	deny
62	*	*	*	445	TCP	deny

63	*	*	*	135	UDP	deny
64	*	*	*	137	UDP	deny
65	*	*	*	138	UDP	deny
66	*	*	*	139	UDP	deny
67	*	*	*	445	UDP	deny
68	*	*	*	143	TCP	deny
69	*	*	*	515	TCP	deny
70	*	*	*	512	TCP	deny
71	*	*	*	514	UDP	deny
72	*	*	*	69	UDP	deny
73	*	*	*	514	TCP	deny
74	*	157.96.138.138	*	5900	TCP	accept
75	*	157.96.138.138	*	5166	TCP	accept
76	*	157.96.138.138	*	*	IP	deny
77	*	157.96.138.101	*	5900	TCP	accept
78	*	157.96.138.101	*	5166	TCP	accept
79	*	157.96.138.101	*	*	IP	deny
80	*	157.96.138.80	*	*	IP	deny
81	*	157.96.138.82	*	*	IP	deny
82	*	157.96.138.234	*	*	IP	deny
83	*	157.96.138.235	*	*	IP	deny
84	*	157.96.138.236	*	*	IP	deny
85	*	157.96.128.*	*	*	IP	accept
86	*	157.96.140.*	*	*	IP	deny
87	*	*	*	*	IP	accept

ACKNOWLEDGMENTS

The author would like to thank Fei Chen for his help in collecting experimental results. The author also would like to thank the editor Trent Jaeger and the anonymous referees for their constructive comments and valuable suggestions on improving the presentation of this article. In addition, the authors thank Alain Mayer, the Chief Technology Officer of Redseal, Inc., for giving us this intellectually interesting and practically useful problem of computing the change-impact of firewall policies.

REFERENCES

AL-SHAER, E. AND HAMED, H. 2004. Discovery of policy anomalies in distributed firewalls. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM).* 2605–2616.

BABOESCU, F., SINGH, S., AND VARGHESE, G. 2003. Packet classification for core routers: Is there an alternative to CAMs? In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communication Societies (InfoCom).*

BABOESCU, F. AND VARGHESE, G. 2002. Fast and scalable conflict detection for packet classifiers. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP).*

BARTAL, Y., MAYER, A. J., NISSIM, K., AND WOOL, A. 1999. Firmato: A novel firewall management toolkit. In *Proceedings of the IEEE Symposium on Security and Privacy.* 17–31.

BOHNER, S. AND ARNOLD, R. 1996. An introduction to software change impact analysis. In *Software Change Impact Analysis,* S. Bohner and R. Arnold, Eds., IEEE Computer Society Press, 1–26.

BRYANT, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Comput. 35*, 8, 677–691.

EPPSTEIN, D. AND MUTHUKRISHNAN, S. 2001. Internet packet filter management and rectangle geometry. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*. 827–835.

ERONEN, P. AND ZITTING, J. 2001. An expert system for analyzing firewall rules. In *Proceedings of the 6^{th} Nordic Workshop on Secure IT Systems (NordSec).* 100–107.

FISLER, K., KRISHNAMURTHI, S., MEYEROVICH, L., AND TSCHANTZ, M. 2005. Verification and change impact analysis of access-control policies. In *Proceedings of the International Conference on Software Engineering (ICSE)*. 196–205.

GOUDA, M., LIU, A. X., AND JAFRY, M. 2008. Verification of distributed firewalls. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM).*

GOUDA, M. G. AND LIU, A. X. 2005. A model of stateful firewalls and its properties. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks (DSN).* 320–327.

GOUDA, M. G. AND LIU, A. X. 2007. Structured firewall design. *Comput. Netw. J. 51*, 4, 1106–1120.

GUPTA, P. 2000. Algorithms for routing lookups and packet classification. Ph.D. thesis, Stanford University.

GUPTA, P. AND MCKEOWN, N. 2001. Algorithms for packet classification. *IEEE Network 15,* 2, 24–32.

GUTTMAN, J. D. 1997. Filtering postures: Local enforcement for global policies. In *Proceedings of the IEEE Symposium on Security and Privacy*. 120–129.

HARI, A., SURI, S., AND PARULKAR, G. M. 2000. Detecting and resolving packet filter conflicts. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM).* 1203–1212.

HAZELHURST, S., ATTAR, A., AND SINNAPPAN, R. 2000. Algorithms for improving the dependability of firewall and filter rule lists. In *Proceedings of the Workshop on Dependability of IP Applications, Platforms, and Networks.*

HOFFMAN, D., PRABHAKAR, D., AND STROOPER, P. 2003. Testing iptables. In *Proceedings of the Conference of the IBM Centre for Advanced Studies (CASCON).* 80–91.

HOFFMAN, D. AND YOO, K. 2005. Blowtorch: a framework for firewall test automation. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE).* 96–103.

HORWITZ, S. 1990. Identifying the semantic and textual differences between two versions of a program. In *Proceedings of the ACM International Conference on Programming Language Design and Implementation (SIGPLAN)*. 234–245.

HWANG, J., XIE, T., CHEN, F., AND LIU, A. X. 2008. Systematic structural testing of firewall policies. In *Proceedings of the 27th IEEE International Symposium on Reliable Distributed Systems (SRDS)*.

JÜRJENS, J. AND WIMMEL, G. 2001. Specification-based testing of firewalls. In *Proceedings of the 4th International Conference on Perspectives of System Informatics (PSI).*

KERRAVALA, Z. 2004. As the value of enterprise networks escalates, so does the need for configuration management. In *Enterprise Computing & Networking*, The Yankee Group Report.

KHAKPOUR, A. R. AND LIU, A. X. 2010. Quantifying and querying network reachability. In *Proceedings of the 29th International Conference on Distributed Computing Systems (ICDCS).*

KUNG, D. C., GAO, J., HSIA, P., WEN, F., TOYOSHIMA, Y., AND CHEN, C. 1994. Change impact identification in object oriented software maintenance. In *Proceedings of the International Conference on Software Maintenance (ICSM)*. 202–211.

LEE, M., OFFUTT, A. J., AND ALEXANDER, R. T. 2000. Algorithmic analysis of the impacts of changes to object-oriented software. In *Proceedings of the 34th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS).* 61–70.

LIU, A. X. 2007. Change-impact analysis of firewall policies. In *Proceedings of the 12th European Symposium Research Computer Security (ESORICS).* 82–96

LIU, A. X. 2008. Firewall policy verification and troubleshooting. In *Proceedings of the IEEE International Conference on Communications (ICC)*.

LIU, A. X. 2009. Firewall policy verification and troubleshooting. *J. Comput. Netw. 53,* 16, 2800–2809.

LIU, A. X. AND GOUDA, M. G. 2009. Firewall policy queries. *IEEE Trans. Parallel Distrib. Syst. 20,* 6, 766–777.

LIU, A. X. AND GOUDA, M. G. 2004. Diverse firewall design. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN).* 595–604.

LIU, A. X. AND GOUDA, M. G. 2005. Complete redundancy detection in firewalls. In *Proceedings of the 19th Annual IFIP Conference on Data and Applications Security*. 196–209.

LIU, A. X. AND GOUDA, M. G. 2008. Diverse firewall design. *IEEE Trans. Parallel Distrib. Syst. 19*, 8.

LIU, A. X. AND GOUDA, M. G. 2010. Complete redundancy removal for packet classifiers in TCAMs. *IEEE Trans. Parallel Distrib. Syst* 21, 4, 424–437.

LIU, A. X., GOUDA, M. G., MA, H. H., AND NGU, A. H. 2004. Firewall queries. In *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS).* 124–139.

LYU, M. R. AND LAU, L. K. Y. 2000. Firewall security: Policies, testing and performance evaluation. In *Proceedings of the 24th International Conference on Computer Systems and Applications (COMPSAC)*. 116–121.

MOFFETT, J. D. AND SLOMAN, M. S. 1994. Policy conflict analysis in distributed system management. *J. Organizational Comput. 4,* 1, 1–22.

OPPENHEIMER, D., GANAPATHI, A., AND PATTERSON, D. A. 2003. Why do Internet services fail, and what can be done about it? In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS)*.

RAJLICH, V. AND GOSAVI, P. 2004. Incremental change in object-oriented programming. *IEEE Softw.*, 2–9.

REN, X., CHESLEY, O. C., AND RYDER, B. G. 2006. Using a concept lattice of decomposition slices for program understanding and impact analysis. IEEE Trans. Softw. Eng. 32, 9, 718–732.

RICHARDSON, R. 2008. CSI/FBI computer crime and security survey. www.cse.msstate.edu/∼cse6243/readings/CSIsurvey2008.pdf.

ROVNIAGIN, D. AND WOOL, A. 2004. The geometric efficient matching algorithm for firewalls. In *Proceedings of the 23rd IEEE Convention of Electrical & Electronics Engineers in Israel (IEEEI).* 153–156.

SENN, D., BASIN, D., AND CARONNI, G. 2005. Firewall conformance testing. In *Proceedings of the International Conference on Testing of Communicating Systems (TESTCOM).*

SOMENZI, F. 2009. Cudd: Cu decision diagram package (release 2.4.1). http://vlsi.colorado.edu/ fabio/CUDD/.

TONELLA, P. 2003. Using a concept lattice of decomposition slices for program understanding and impact analysis. *IEEE Trans. Softw. Eng. 29,* 6, 495–509.

WOOL, A. 2004. A quantitative study of firewall configuration errors. *IEEE Comput. 37*, 6, 62–67.

WOOL, A. 2010. Trends in firewall configuration errors: Measuring the holes in swiss cheese. *IEEE Internet Comput. 14,* 4, 58–65.

YUAN, L., CHEN, H., MAI, J., CHUAH, C.-N., SU, Z., AND MOHAPATRA, P. 2006. Fireman: A toolkit for firewall modeling and analysis. In *Proceedings of the IEEE Symposium on Security and Privacy*.

Received October 2008; revised September 2010, November 2011; accepted November 2011