This paper was presented as part of the main technical program at IEEE INFOCOM 2011

Firewall Verification and Redundancy Checking are Equivalent

H. B. Acharya
University of Texas at Austin
acharya@cs.utexas.edu

M. G. Gouda
National Science Foundation
University of Texas at Austin
mgouda@nsf.gov

***Abstract*—A firewall is a packet filter that is placed at the entrance of a private network. It checks the header fields of each incoming packet into the private network and decides, based on the specified rules in the firewall, whether to accept the packet and allow it to proceed or to discard the packet. To validate the correctness and effectiveness of the rules in a firewall, the firewall rules are usually subjected to two types of analysis: verification and redundancy checking. Verification is used to verify that the rules in a firewall accept all packets that should be accepted and discard all packets that should be discarded. Redundancy checking is used to check that no rule in a firewall is redundant (i.e. can be removed from the firewall without changing the sets of packets accepted and discarded by the firewall). In this paper we show that, contrary to the conventional wisdom, these two types of analysis are in fact equivalent. In particular, we show that (1) every verification algorithm can be also used to check whether a rule in a firewall is redundant, and (2) every redundancy checking algorithm can be also used to verify whether the rules in a firewall accept or discard an intended set of packets.**

I. Introduction

A firewall is a packet filter that is placed at a point where a private computer network is connected to the rest of the Internet. The firewall intercepts each packet that is exchanged between the private network and the Internet, examines the fields of the packet headers, and makes a decision to either accept the packet and allow it to proceed on its way, or discard the packet.

The decision that a firewall makes, when it receives a packet, depends on two factors:

1) The values of the fields in the packet headers
2) The sequence of firewall rules that is specified by the firewall designer

A firewall rule consists of a predicate and a decision, which is either accept or discard. When the firewall receives a packet, the firewall searches its sequence of rules for the first rule, whose predicate is satisfied by the values of the fields in the packet headers, and then applies the decision of this rule to the packet.

Note that there are two sets of packets that are associated with each firewall: the set of all packets that are accepted by the firewall, and the set of all packets that are discarded by the firewall.

A firewall property, like a firewall rule, consists of a predicate and a decision, which is either accept or discard. A firewall is said to satisfy a property iff one of the following two conditions holds:

(a) The decision of the property is accept and the firewall accepts every packet that satisfies the predicate of the property
(b) The decision of the property is discard and the firewall discards every packet that satisfies the predicate of the property

Two firewalls are equivalent iff they accept the same set of packets and discard the same set of packets.

A rule in a firewall is said to be redundant iff removing the rule from the firewall yields a firewall that is equivalent to the original firewall.

After the rules of a firewall are designed (by a firewall designer), they are usually validated by subjecting them to two seemingly different algorithms: a verification algorithm and a redundancy checking algorithm. The function of the verification algorithm is to verify that the firewall rules accept all packets that should be accepted and discard all packets that should be discarded. The function of the redundancy checking algorithm is to check whether any rule in the firewall is redundant.

Figure 1 shows an outline of a firewall verification algorithm. It takes as input a firewall F and a property r and produces as output a determination of whether F satisfies r.

Figure 2 shows an outline of a firewall redundancy checking algorithm. It takes as input a firewall F and a rule r in F and produces as output a determination of whether r is redundant in F (and so should be removed from F).

Up until now it has been thought that these two types of algorithms, verification algorithms and redundancy checking algorithms, are quite different becuase they solve quite different problems. But we show in this paper that this is not the case. In particular, we show the following two results:

(i) Any verification algorithm can be also used as a subroutine in checking whether a given rule in a given firewall is redundant in this firewall. This result is illustrated in Figure 3.
(ii) Any redundancy checking algorithm can be also used as a subroutine in verifying whether a given firewall satisfies a given property. This result is illustrated in Figure 4.

These two results indicate that the two important problems of firewall verification and firewall redundancy checking are equivalent and any progress that one can achieve in solving

either problem can be regarded also as a progress in solving the other problem.

Fig. 1. Firewall verification algorithm.

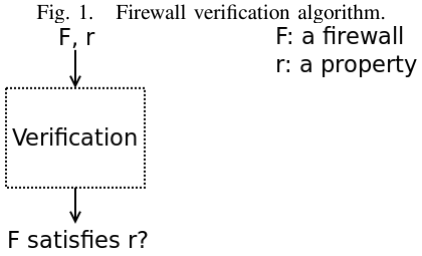


Fig. 2. Firewall redundancy checking algorithm.

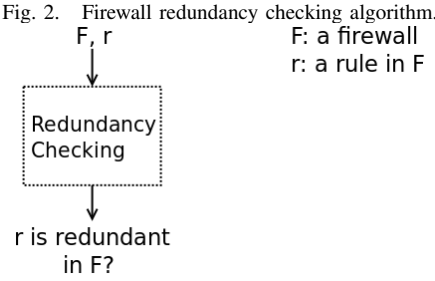


Fig. 3. Using verification in redundancy checking.

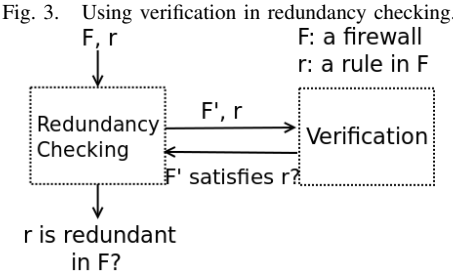
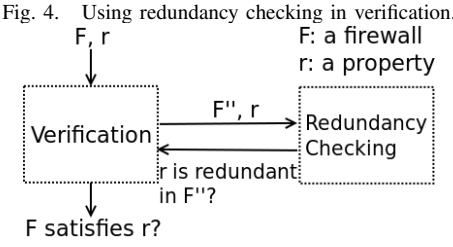


Fig. 4. Using redundancy checking in verification.



II. Packets, Rules, Firewalls, and Properties

In this section, we define the main four terms in this paper: packets, rules, firewalls, and properties. We start our presentation by introducing the concept of a field.

A *field* is a variable whose value is taken from a nonempty interval of non-negative integers called the *domain of the field*.

In this paper, we assume that there are d fields, named f_1, .., and f_d, in the headers of each packet. (Examples of these fields are the source IP address, the destination IP address, the transport protocol, the source port number, and the destination port number.)

The domain of each field f_j is denoted $D(f_j)$.

A *rule* r is of the form:

$$r : f_1 \in X_1 \wedge .. \wedge f_d \in X_d \rightarrow < decision >$$

Note that r is the name of the rule, each f_j is a field, each X_j is a nonempty interval of nonnegative integers taken from the domain $D(f_j)$ of field f_j, and $< decision >$ is either accept or discard.

A rule whose decision is accept (or discard, respectively) is called an *accept rule* (or *discard rule*, respectively).

A *packet* p is a tuple $(p_1, .., p_d)$ of d nonnegative integers, where each integer p_j is taken from the domain $D(f_j)$ of field f_j.

A packet $(p_1, .., p_d)$ is said to *match* a rule r of the form:

$$r : f_1 \in X_1 \wedge .. \wedge f_d \in X_d \rightarrow < decision >$$

iff the predicate $(p_1 \in X_1 \wedge .. \wedge p_d \in X_d)$ is true.

A *firewall* is a nonempty sequence of rules.

A packet is said to *match* a firewall F iff the packet matches at least one rule in F.

A firewall F is called *complete* iff every packet matches F. A firewall F is called *partial* iff some packet does not match F.

A rule r is called a *full rule* iff it is of the form:

$$r : f_1 \in X_1 \wedge .. \wedge f_d \in X_d \rightarrow < decision >$$

where each interval X_j is the domain $D(F_j)$ of field F_j. Note that every packet matches any full rule. Thus, each firewall that has a full rule is complete.

A firewall F is said to *accept* (or *discard*, respectively) a packet p iff F has an accept (or discard, respectively) rule r such that the following two conditions hold:

1) p matches r
2) p does not match any rule that precedes r in F

A firewall F is said to *ignore* a packet p iff p matches no rule in F.

Note that a complete firewall ignores no packet whereas a partial firewall ignores at least one packet.

Note also that for any given firewall F and any given packet p, exactly one of the following three statements holds:

(a) F accepts p
(b) F discards p
(c) F ignores p

Two firewalls F and G are said to be *equivalent* iff for every packet p, exactly one of the following three statements holds:

1) Both F and G accept p
2) Both F and G discard p
3) Both F and G ignore p

A *property* s has the same form as a rule in a firewall:

$$s : f_1 \in Y_1 \wedge .. \wedge f_d \in Y_d \rightarrow < decision >$$

Note that s is the name of the property, each f_j is a field, each Y_j is a nonempty interval of nonnegative integers taken from the domain $D(f_j)$ of field f_j, and $< decision >$ is either accept or discard.

A property whose decision is accept (or discard, respectively) is called an *accept property* (or *discard property*, respectively).

A packet $(p_1, .., p_d)$ is said to *match* a property s of the form:

$$s : f_1 \in Y_1 \wedge .. \wedge f_d \in Y_d \rightarrow < decision >$$

iff the predicate $(p_1 \in Y_1 \wedge .. \wedge p_d \in Y_d)$ is true.

Note that rules and properties have the same syntax and semantics. Thus, we sometimes treat a rule in a firewall as if it is a property and vice versa.

III. Verification of Firewalls

A firewall F is said to *satisfy* a property s iff one of the following two conditions holds.

1) s is an accept property and each packet that matches s is accepted by F.
2) s is a discard property and each packet that matches s is discarded by F.

A *firewall verification algorithm* is an algorithm that takes as input any given firewall F and any given property s and determines whether F satisfies s. Below, we show that any firewall verification algorithm can also be used to determine whether any given rule in any given firewall is redundant and can (and should) be removed from the firewall.

Several firewall verification algorithms are given [1], [2], [3], [4], and [5]. The time and space complexity of the algorithms in [1], [2], and [3], when applied to a firewall F and a property s, are both $O(n^d)$, where n is the number of rules and d is the number fields in the given firewall F.

The time and space complexity of the algorithm in [4], when applied to a firewall F and a property s, are both $O(nd)$ where n is the number of rules and d is the number of fields in F. However, this algorithm is probabilistic, which means that sometimes when the algorithm concludes that F satisfies s, the conclusion is wrong. (On the other hand, every time the algorithm concludes that F does not satisfy s, the conclusion is correct.)

The most efficient firewall verification algorithm is the one presented in [5]. The time complexity of this algorithm is $O(n^d)$ and the space complexity is $O(nd)$. This algorithm is based on the concept of projection firewall, which is a firewall that is constructed by combining any given firewall with any given property.

But before we can present the construction algorithm for projection firewalls (Algorithm 1 below), we need first to introduce three new concepts:

- A rule overlapping a property
- The projection of a rule over a property
- A rule covering a property

Let r be a rule and s be a property of the following form:

$$r : f_1 \in X_1 \wedge .. \wedge f_d \in X_d \rightarrow < r.decision >$$
$$s : f_1 \in Y_1 \wedge .. \wedge f_d \in Y_d \rightarrow < s.decision >$$

Rule r is said to *overlap* property s iff every intersection of an interval X_j in r with the corresponding interval Y_j in s is nonempty.

If rule r overlaps property s, then define the *projection* of r *over* s, denoted r/s , as the following rule:

$$r/s : f_1 \in (X_1 \cap Y_1) \wedge .. \wedge f_d \in (X_d \cap Y_d) \rightarrow < r.decision >$$

Rule r is said to *cover* property s iff every interval X_j in r contains the corresponding interval Y_j in s.

Algorithm 1 is the algorithm for constructing projection firewalls.

Algorithm 1 Constructing Projection Firewalls

Input: a firewall F and a property s
Output: a firewall denoted F/s, called the projection of F over s

```
F/s := an empty sequence of rules;
for each rule r in F do
  if r overlaps s then
    add the rule r/s at the tail of the sequence F/s
  end if
  if r covers s then
    exit the for loop
  end if
end for
```

Utilizing the concept of projection firewalls, the following two theorems present necessary and sufficient conditions for a firewall F to satisfy a property s. Theorem 1 applies if s is an accept property whereas Theorem 2 applies if s is a discard property. These two theorems are the basis upon which the efficient firewall verification algorithm in [5] is established.

Theorem 1. *A firewall F satisfies an accept property s iff the projection firewall G/s discards no packet, where G is firewall F after adding a full discard rule at the end.*

□

Theorem 2. *A firewall F satisfies a discard property s iff the projection firewall G/s accepts no packet, where G is firewall F after adding a full accept rule at the end.*

□

In Section V below, we show that the verification algorithm in [5], or any other firewall verification algorithm for that matter, can also be used to detect redundant rules in any given firewall.

IV. Redundancy Checking in Firewalls

Let F be a firewall and let r be a rule in F. Rule r is said to be *redundant in* F iff the two firewalls F and $F - r$ are equivalent, where $F - r$ is firewall F after removing rule r from it. (Recall that, as mentioned above in Section II, two firewalls are equivalent iff they accept, discard, and ignore the same packets.)

A *firewall redundancy checking algorithm* is an algorithm that takes as input any firewall F and any rule r in F and determines whether or not r is redundant in F.

Firewall redundancy checking algorithms are useful because any firewall that has a large number of redundant rules is inefficient to check, whenever a packet arrives at the firewall, whether to accept or discard the packet.

The following theorem presents a necessary and sufficient condition for determining whether a rule in a firewall is redundant. This condition can then be employed in designing a firewall redundancy checking algorithm.

Theorem 3. *Let F be a firewall and r be a rule in F. Rule r is redundant in F iff for every packet p that matches rule r, at least one of the following two conditions holds:*

1) *Packet p matches one or more rules that precede r in F.*
2) *Packet p matches one or more rules that follow rule r in F, and the first such rule has the same decision, accept or discard, as rule r.*

□

Unfortunately, if the necessary and sufficient condition in Theorem 3 is used in designing a firewall redundancy checking algorithm, then the time complexity of this algorithm will be proportional to the number of packets that match the checked rule, which can be quite large. This makes the algorithm too expensive to be practical.

Instead of this expensive algorithm, we seek in this paper firewall redundancy checking algorithms whose time complexity is a function of n and d only, where n is the number of rules and d is the number of fields in the checked firewall. Towards this end, we show, in Section V, that any firewall verification algorithm can be also used to check redundant rules in firewalls. Moreover, the time and space complexity of the verification algorithm when used in detecting whether a given rule is redundant in a give firewall are the same as the time and space complexity of the same algorithm when used to verify whether a given firewall satisfies a given property. Thus, our efficient firewall verification algorithm in [5], whose time complexity is $O(n^d)$ and whose space complexity is $O(nd)$, can be used, with the same complexity, to check redundant rules in firewalls.

Similarly we show, in Section VI, that any firewall redundancy checking algorithm can be also used, with the same complexity, in verifying firewalls.

These results of Sections V and VI indicate that the two problems of firewall verification and of firewall redundancy checking are equivalent and any progress that one can achieve in solving either problem can be also regarded as a progress in solving the other problem.

V. Using Verification in Redundancy Checking

In this section, we show that any firewall verification algorithm can be also used to detect redundant rules in firewalls. Specifically, we present an algorithm, named Algorithm $V-to-R$, that takes as input any firewall F, any rule r in F, and any firewall verification algorithm V, and uses Algorithm V to determine whether r is redundant in F. We also show that that the time and space complexity of Algorithm $V-to-R$ are the same as the time and space complexity, respectively, of Algorithm V.

Algorithm $V-to-R$ is shown below as Algorithm 2.

Algorithm 2 $V-to-R$

Input: a firewall F,
a rule r in F, and
a firewall verification algorithm V
Output: determination of whether r is redundant in F
let G denote firewall F after making the decision of each rule, that precedes rule r in F, be the same as the decision of rule r.
let $G-r$ denote firewall G after removing rule r from it.
let $(G-r)/r$ denote the projection of firewall $(G-r)$ over rule r. (Note that, in this case, rule r is viewed as a property.)
call the input firewall verification algorithm V to verify whether firewall $(G-r)/r$ satisfies rule r. (Note that, in this case, rule r is viewed as a property.)
if $(G-r)/r$ satisfies r **then**
  Report that r is redundant in F.
else
  Report that r is not redundant in F.
end if

The next two theorems give the time and space complexity of Algorithm $V-to-R$ as functions of the time and space complexity, respectively, of Algorithm V.

Theorem 4. *Let V denote the firewall verification algorithm employed in Algorithm $V-to-R$. Let $T_V(n,d)$ denote the time complexity of Algorithm V when this algorithm is applied to a firewall with n rules and d fields. Then the time complexity of Algorithm $V-to-R$, denoted $T_{V-to-R}(n,d)$, can be computed as follows:*

$$T_{V-to-R}(n,d) = T_V(n-1,d) + O(nd)$$

□

Theorem 5. *Let V denote the firewall verification algorithm employed in Algorithm $V-to-R$. Let $S_V(n,d)$ denote the space complexity of Algorithm V when this algorithm is applied to a firewall with n rules and d fields. Then the space complexity of Algorithm $V-to-R$, denoted $S_{V-to-R}(n,d)$, can be computed as follows:*

$$S_{V-to-R}(n,d) = S_V(n-1,d) + O(nd)$$

□

Because $T_V(n,d)$ is at least $O(nd)$, we conclude from Theorem 4 that the time complexity of Algorithm $V-to-R$ is the same as the time complexity of Algorithm V. Similarly,

because $S_V(n,d)$ is at least $O(nd)$, we conclude from Theorem 5 that the space complexity of Algorithm $V-to-R$ is the same as the space complexity of Algorithm V.

VI. Using Redundancy Checking in Verification

In this section, we show that any firewall redundancy checking algorithm can be also used to verify firewalls. Specifically, we present an algorithm, named Algorithm $R-to-V$, that takes as input any firewall F, any property r, and any firewall redundancy checking algorithm R, and uses Algorithm R to verify whether F satisfies r. We also show that that the time and space complexity of Algorithm $R-to-V$ are the same as the time and space complexity, respectively, of Algorithm R.

Algorithm $R-to-V$ is shown below as Algorithm 3.

Algorithm 3 $R-to-V$

Input: a firewall F,
a property r, and
a redundancy checking algorithm R
Output: determination of whether F satisfies r
let G denote the firewall that consists of property r, as the first rule, followed by the sequence of rules in firewall F. (Note that, in this case, property r is viewed as a rule.)
call the input redundancy checking algorithm R to check whether the first rule in firewall G, namely rule r, is redundant in G.
if r is redundant in G **then**
  Report that F satisfies r.
else
  Report that F does not satisfy r.
end if

The next two theorems give the time and space complexity of Algorithm $R-to-V$ as functions of the time and space complexity, respectively, of Algorithm R.

Theorem 6. *Let R denote the redundancy checking algorithm employed in Algorithm $R-to-V$. Let $T_R(n,d)$ denote the time complexity of Algorithm R when this algorithm is applied to a firewall with n rules and d fields. Then the time complexity of Algorithm $R-to-V$, denoted $T_{R-to-V}(n,d)$, can be computed as follows:*

$$T_{R-to-V}(n,d) = T_R(n+1,d) + O(nd)$$

□

Theorem 7. *Let R denote the redundancy checking algorithm employed in Algorithm $R-to-V$. Let $S_R(n,d)$ denote the space complexity of Algorithm R when this algorithm is applied to a firewall with n rules and d fields. Then the space complexity of Algorithm $R-to-V$, denoted $S_{R-to-V}(n,d)$, can be computed as follows:*

$$S_{R-to-V}(n,d) = S_R(n+1,d) + O(nd)$$

□

Because $T_V(n,d)$ is at least $O(nd)$, we conclude from Theorem 6 that the time complexity of Algorithm $R-to-V$ is the same as the time complexity of Algorithm R. Similarly, because $S_V(n,d)$ is at least $O(nd)$, we conclude from Theorem 7 that the space complexity of Algorithm $R-to-V$ is the same as the space complexity of Algorithm R.

VII. Related Work

In this section, we provide a brief survey of the current state of the art in firewall research. As firewalls are a critical component of enterprise and government cyber security, they have been extensively studied. The literature on firewalls falls broadly into four major groups: firewall testing, firewall analysis, firewall verification, and firewall design.

1) *Firewall Testing:*
To test a given firewall F, one generates many packets for which the "expected" decisions of F, accept or discard, are known a priori. The generated packets are then sent to F, and the actual decisions of F for these packets are observed. If the expected decision for each generated packet is the same as the actual decision for the packet, one concludes that the given firewall F is correct. Otherwise, the given firewall F has errors. Different methods of firewall testing differ in how the testing packets are generated. For instance, the test packets can be hand-generated by domain experts to target specific vulnerabilities in the given firewall F, or generated from the formal specifications of the security policy of the given firewall F, as in [6]. A scheme for targeting test packets for better fault coverage is given in [7]. Al-Shaer et al. provide a complete framework to generate targeted packets and obtain good coverage in testing in [8].
2) *Firewall Analysis:*
To analyze a given firewall F, one applies an algorithm to identify (some or all of the) vulnerabilities, conflicts, anomalies, and redundancies in the given firewall F. A systematic method for analyzing firewalls is presented in [9]. The concept of conflicts between rules in a firewall is due to [10] and [11]. A framework for understanding the vulnerabilities in a single firewall is outlined in [12], and an analysis of these vulnerabilities presented in [13]. [14] is a quantitative study of configuration errors for a firewall. An example of an efficient firewall analysis algorithm is given in FIREMAN [15].
3) *Firewall Verification:*
To verify a given firewall F against a given property R, one applies an algorithm (similar to the one discussed in this paper) to verify whether or not F satisfies R. The question of how to query a given firewall and obtain the answer (whether or not it satisfies a given property) is discussed in [9] and [1]. These algorithms are proved to be $O(n^d)$ in [16]. We present a new, $O(nd)$-space algorithm in [5].
4) *Firewall Design:*
To ensure a firewall does not have vulnerabilities or

other problems, it can be designed from the outset using structured algorithms. Such algorithms, that can generate a firewall from its specification, are provided in [2] and [17]. We develop a new approach to the problem of firewall design and demonstrate how to design a firewall as a set of modules, in [18].

Our current paper demonstrates that one of the most studied problems of firewall analysis, namely redundancy checking, is equivalent to firewall verification. In fact, one can use a redundancy checker to verify a firewall, and a verifier to detect all redundant rules in the firewall. As we have developed a fast engine for firewall verification [5], we will in future work use the results presented in this paper to develop a fast redundancy checker for firewalls.

VIII. Concluding Remarks

We have shown in this paper that the two problems of firewall verification and firewall redundancy checking are equivalent in the following sense. Any algorithm that can be used to solve either problem can be also used to solve the other problem in the same time and space complexity.

This result is interesting because the two problems of firewall verification and firewall redundancy checking are not equally important. Specifically, the problem of firewall verification addresses the critical issue of firewall correctness. And so this problem is more important to solve than the problem of firewall redundancy checking which addresses the less critical issue of firewall execution time.

The two problems of firewall verification and firewall redundancy checking, defined in this paper, can be both generalized. And we believe that the two resulting general problems are also equivalent. First, the firewall verification problem can be generalized to verifying whether a given firewall satisfies a given property that consists of one or more rules (instead of a single rule as defined in this paper) that have the same decision. Second, the firewall redundancy checking problem can be generalized to checking whether any given sequence of consecutive rules, that have the same decision, in a given firewall is redundant. We conjecture that these two problems are equivalent.

Our equivalence result in this paper, concerning firewall verification and firewall redundancy checking, is established under the assumption that firewalls are stateless. Whether the same result can be also established for stateful firewalls, for example as specified in [19], remains an open problem that merits further research.

References

[1] A. X. Liu and M. G. Gouda, "Firewall policy queries," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2009.

[2] M. G. Gouda and A. X. Liu, "Structured firewall design," *Computer Networks*, vol. 51, pp. 1106–1120, 2007.

[3] E. Al-shaer, W. Marrero, A. El-atawy, and K. Elbadawi, "Network configuration in a box: Towards end-to-end verification of network reachability and security," in *In Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2009.

[4] H. B. Acharya and M. G. Gouda, "Linear-time verification of firewalls," in *In Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2009, pp. 133 –140.

[5] ——, "Projection and division: Linear-space verification of firewalls," in *In Proceedings of the 30th IEEE International Conference on Distributed Computing Systems*, 2010, pp. 736 – 743.

[6] J. Jürjens and G. Wimmel, "Specification-based testing of firewalls," in *PSI '02: Revised Papers from the 4th International Andrei Ershov Memorial Conference on Perspectives of System Informatics*. London, UK: Springer-Verlag, 2001, pp. 308–316.

[7] A. El-Atawy, K. Ibrahim, H. Hamed, and E. S. Al-Shaer, "Policy segmentation for intelligent firewall testing," *Secure Network Protocols, 2005. (NPSec). 1st IEEE ICNP Workshop on*, pp. 67–72, Nov. 2005.

[8] E. S. Al-Shaer, A. El-Atawy, and T. Samak, "Automated pseudo-live testing of firewall configuration enforcement." *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 3, pp. 302–314, 2009.

[9] A. J. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," in *IEEE Symposium on Security and Privacy*, 2000, pp. 177–187.

[10] D. Eppstein and S. Muthukrishnan, "Internet packet filter management and rectangle geometry," in *SODA*, 2001, pp. 827–835.

[11] H. Adiseshu, S. Suri, and G. M. Parulkar, "Detecting and resolving packet filter conflicts," in *INFOCOM*, 2000, pp. 1203–1212.

[12] M. Frantzen, F. Kerschbaum, E. E. Schultz, and S. Fahmy, "A framework for understanding vulnerabilities in firewalls using a dataflow model of firewall internals," *Computers & Security*, vol. 20, no. 3, pp. 263–270, 2001.

[13] S. Kamara, S. Fahmy, E. E. Schultz, F. Kerschbaum, and M. Frantzen, "Analysis of vulnerabilities in internet firewalls," *Computers & Security*, vol. 22, no. 3, pp. 214–232, 2003.

[14] A. Wool, "A quantitative study of firewall configuration errors," *IEEE Computer*, vol. 37, no. 6, pp. 62–67, 2004.

[15] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra, "Fireman: A toolkit for firewall modeling and analysis," *Security and Privacy, IEEE Symposium on*, vol. 0, pp. 199–213, 2006.

[16] M. G. Gouda, A. X. Liu, and M. Jafry, "Verification of distributed firewalls," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2008.

[17] A. X. Liu and M. G. Gouda, "Diverse firewall design," *IEEE Transaction on Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1237–1251, 2008.

[18] H. B. Acharya, A. Joshi, and M. G. Gouda, "Firewall modules and modular firewalls," in *In Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2010.

[19] M. G. Gouda and A. X. Liu, "A model of stateful firewalls and its properties," in *In Proceedings of the IEEE International Conference on Dependable Systems and Networks*, 2005, pp. 320–327.

[20] D. Hoffman and K. Yoo, "Blowtorch: a framework for firewall test automation," in *ASE '05: Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. New York, NY, USA: ACM, 2005, pp. 96–103.