Verification of Distributed Firewalls

Mohamed G. Gouda
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-0233, U.S.A.
gouda@cs.utexas.edu

Alex X. Liu
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824-1266, U.S.A.
alexliu@cse.msu.edu

Mansoor Jafry
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-0233, U.S.A.
mansoor.jafry@gmail.com

***Abstract*—The private computer network of any large enterprise has tens, or even hundreds, of firewalls. These firewalls are placed at the entry points of the network (where the network is connected with the rest of the Internet), and at many chosen points within the network. The result is a complex firewall network that seems hard to understand or analyze. In this paper, we propose a method for verifying the correctness of firewall networks with tree topologies. Our method is based on identifying two types of properties of firewall trees: accept and discard properties. An accept (or discard) property of a firewall tree specifies a class of packets that should be accepted (or discarded, respectively) by the firewall tree. We present two algorithms that can be used to decide whether a given firewall tree satisfies a given, accept or discard, property of that tree.**

I. Introduction

A firewall is placed at an entry point where a private computer network is connected to the outside Internet. It intercepts all the packets that are exchanged between the private computer network and the rest of the Internet, and examines the IP, TCP and UDP headers of each intercepted packet, and decides whether to accept the packet (and allow it to proceed to its destination) or to discard the packet.

Because the private computer network of a large enterprise consists of many inter-connected constituent networks, the private network of a large enterprise has tens or even hundreds of firewalls. These firewalls are placed at the entry points of the private network and at the entry point of each constituent network within the private network. As an example, consider the private computer network of some university. Such a network would consist of many inter-connected constituent networks for various academic, administrative and service departments within the university. A firewall is often placed at the entry point of each of these constituent networks.

One can view the private computer network of an enterprise as a network of tens or even hundreds of firewalls, which is hard to understand or verify its correctness. In this paper we address this problem by presenting a method for deciding whether a given firewall network accepts or discards a given class of packets. Note that the focus of this paper is on stateless firewalls. Our contributions in this paper are as follows: (1) We present a formal model of firewall networks with tree topologies. (2) We identify two classes of properties of firewall trees; we call these properties accept and discard properties. An accept (or discard) property of a firewall tree specifies a class of packets that should be accepted (or discarded, respectively) by the firewall tree. (3) We develop two algorithms that can be used to decide whether a given firewall tree satisfies a given, accept or discard, property. The time and space complexity of each of these algorithms is practically polynomial.

II. Related Work

Prior papers in the area have focused on the analysis of a single firewall. This paper is the first one that addresses the correctness verification of distributed firewalls. The difference between analysis and correctness verification of firewall networks is two-fold. First, the function of analysis is to identify redundant rules, conflicts, and anomalies in the firewalls. The identified conflicts and anomalies are not necessarily design errors in the firewalls. Rather they can be intended to reduce the total number of rules in each firewall. Second, the function of correctness verification is to check whether the firewall network satisfies its intended specification which is given as a set of accept or discard properties. Note that if the analysis of a firewall network shows that the network has no redundant rules and no conflicts or anomalies, then this does not imply the firewall network operates in accordance with its intended specification. Even in this case, verification of the firewall network is necessary to show that the network operates in accordance with its intended specification.

In [1], the authors presented a systematic method for analyzing single firewalls. In [2] and [3], the authors discussed the concept of conflicts between different rules in a single firewall, and presented algorithms for detecting firewall conflicts. In [4] and [5], the authors introduced a classification of firewall conflicts into different anomaly classes, and they presented several algorithms for detecting these anomalies in a single firewall. This analysis can also be used in verifying the security policies in IPsec and VPN, as demonstrated in [6].

A framework for understanding the vulnerabilities in a single firewall was outlined in [7], and an analysis of these vulnerabilities was presented in [8]. A quantitative study of the configuration of errors of a single firewall was reported in [9]. Methods for issuing queries and getting answers concerning the function of a single firewall were discussed in [1] and [10]. Also, methods for designing a single firewall from its given specification were presented in [11] and [13]. Methods for analyzing firewall networks were presented in [17] and [18].

III. Firewall Decision Diagrams

We now formally define the concepts of fields, packets, firewalls, and firewall decision diagrams. A *field* F_i is a variable of finite length (*i.e.*, 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 $F_1, \cdots, F_d$ is a d-tuple $(p_1, \cdots, p_d)$ where each p_i $(1 \leq i \leq d)$ 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. Without loss of generality, we assume that d is at least 3. We also assume that the first three fields F_1, F_2, and F_3 of a received packet are as follows: (1)F_1 is the incoming interface via which the firewall receives the packet, (2) F_2 is the IP address of the original source of the received packet, and (3) F_3 is the IP address of the ultimate destination of the received packet. Note that F_1 is not really a field in any header of the received packet. Nevertheless, we will refer to it as a field throughout this paper.

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 packet matches a rule if and only if the packet matches the predicate of the rule. A packet $(p_1, \cdots, p_d)$ *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. Typically, the value for $\langle decision \rangle$ is either *accept* or *discard*.

A sequence of rules $\langle r_1, \cdots, r_n \rangle$ 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 firewall, the predicate of the last rule is usually specified as $F_1 \in D(F_1) \wedge \cdots F_d \in \wedge D(F_d)$. A *firewall* $\mathbb{C}$ is a sequence of rules that is complete. 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 (i.e., highest priority) rule that p matches in f. The decision that firewall f makes for packet p is denoted $f(p)$.

The crucial tool used in our firewall network analysis framework is the Firewall Decision Diagram (FDD) [11], [12]. A *Firewall Decision Diagram* (FDD) over fields $F_1, \cdots, F_d$ is an acyclic and directed graph that has the following five properties: (1) There is exactly one node that has no incoming edges. This node is called the *root*. The nodes that have no outgoing edges are called *terminal* nodes. (2) Each node v has a label, denoted $F(v)$, such that $F(v) \in \{F_1, \cdots, F_d\}$ if v is a nonterminal node and $F(v) \in accept,\ discard$ if v is a terminal node. (3) Each edge $e{:}u \rightarrow v$ is labeled with a nonempty set of integers, denoted $I(e)$, where $I(e)$ is a subset of the domain of u's label (*i.e.*, $I(e) \subseteq D(F(u))$). (4) A directed path from the root to a terminal node is called a *decision path*. No two nodes on a decision path have the same label. (5) The set of all outgoing edges of a node v, denoted $E(v)$, satisfies the following two conditions: (i) *Consistency*: $I(e) \cap I(e') = \emptyset$ for any two distinct edges e and e' in $E(v)$. (ii) *Completeness*: $\bigcup_{e \in E(v)} I(e) = D(F(v))$.

We define a *full-length ordered FDD* as an FDD where in each decision path, all fields appear exactly once and in the same order. An FDD construction algorithm, which converts a firewall to an equivalent full-length ordered FDD, is in [13]. For ease of presentation, in the rest of this proposal, we use the term "FDD" to mean "full-length ordered FDD" if not otherwise specified.

IV. Firewall Networks

A *firewall network* is an undirected graph with two types of nodes: *firewall nodes* and *domain nodes*. In a firewall network each (undirected) edge, called $interface$, connects a firewall node with a domain node. Each interface in a firewall network has an integer identifier such that the identifiers of all interfaces that are incident on the same firewall node are distinct. Each leaf node in a firewall network is a domain node. Fig. 1 shows a firewall network example with two firewall nodes, a and b, and five domain nodes, u through y. This network has six interfaces with identifiers in the range 0..2. All four leaf nodes in this network, namely u, v, x and y, are domain nodes.

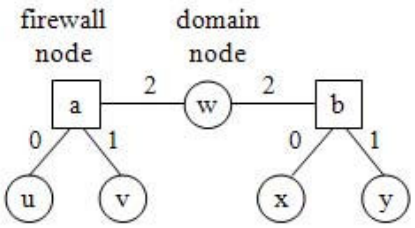


Fig. 1. A firewall network example

Associated with each firewall node u in a firewall network is a firewall, as defined in section III, denoted FW.u. The first conjunct, namely $F_1 \in S_1$, in the predicate of each rule in FW.u is such that S1 is a subset of the set of identifiers of all interfaces that are incident on node u in the firewall network. As an example, consider firewall node b in the firewall network in Fig. 1. Associated with node b is a firewall FW.b. The first conjunct, $F_1 \in S_1$ in the predicate of each rule in FW.b is such that S_1 is a subset of $\{0,1,2\}$.

Associated with each domain node u in a firewall network is a set of IP addresses (from the Internet) denoted DM.u. The sets of IP addresses associated with the domain nodes of a firewall network are required to satisfy the following condition. For each IP address (from the Internet), there is exactly one domain node u in the firewall network such that the IP address is in DM.u. It follows that each firewall network has a domain node u, that can be regarded as "the rest of the internet", where DM.u contains all the IP addresses (from the Internet)

that are not associated with any of the other domain nodes in the firewall network A firewall network that has no cycles is referred to as a $firewall\ tree$. For example, the firewall network in Fig. 1 is a firewall tree.

V. Accept properties of firewall trees

Let N be a firewall tree, and let u and v be two distinct domain nodes in N. An $accept\ property\ from$ u to v in N is of the form: $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d \rightarrow accept$ such that (1) T_2 is a subset of DM.u, and (2) T_3 is a subset of DM.v. Recall that DM.x is the set of IP addresses associated with domain node x in the firewall tree N. Let N be a firewall tree, and u and v be two distinct domain nodes in N. Also, let $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d \rightarrow accept$ be an accept property from u to v in N. Tree N is said to $satisfy$ the accept property if and only if the following condition holds: if any packet, that satisfies the predicate $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ of the accept property is generated in domain u, and if this packet is routed in N towards its ultimate destination in domain v. then this packet is accepted by every firewall from u to v.

Next, we describe an algorithm for deciding whether a firewall tree N satisfies a given accept property of N.

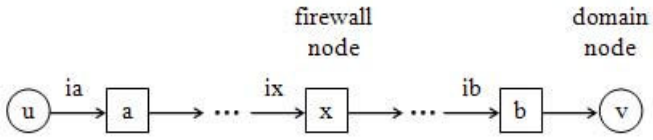


Fig. 2. Path from domain u to domain v in firewall tree N.

Step (a) in this algorithm can be regarded as projecting firewall FW.x on its incoming interface ix. The function of this step is to remove from firewall FW.x any rule that has nothing to do with packets that FW.x receives through the incoming interface ix.

Step (c) in Algorithm 2 can be regarded as projecting the firewall decision diagram FD.x on the predicate $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ of the accept property. The function of this step is to remove from the firewall decision diagram FD.x all the edges that cannot be traversed when trying to use FD.x to decide whether to accept or discard any packet that satisfies the predicate $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$.

VI. Discard Properties of Firewall Trees

Let N be a firewall tree, and let u and v be two distinct domain nodes in N. A $discard\ property$ from u to v in N is of the form: $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d \rightarrow discard$ such that (1) each of u and v is connected by an interface with the same firewall node in N, and (2) T3 is a subset of DM.v. Recall that DM.v is the set of IP addresses associated with domain node v. Let N be a firewall tree, and u and v be two distinct domain nodes in N. Also let $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d \rightarrow discard$ be a discard property form u to v in N. Tree N is said to $satisfy$ the discard property if and only if the following condition holds. If any packet that satisfies the predicate $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ of the discard property ever reaches domain u, and if this packet is routed to the firewall that is adjacent to both u and v, then the packet is discarded by this firewall.

There is a significant difference between an accept property and a discard property of a firewall tree N. First, in order that N satisfies an accept property, each packet that satisfies the predicate of the property, needs to be accepted by every firewall that the packet encounters as it travels from its original source to its ultimate destination in N. Second, in order that N satisfies a discard property, each packet that satisfies the predicate of the property needs to be discarded by the last firewall that the packet encounters before it reaches its ultimate destination in N. This is because the last firewall that a packet

Algorithm 1 Verifying Accept Property

Input: A firewall tree N with two distinct domain nodes u and v, and an accept property from u to v in N: $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d \rightarrow accept$.

Output: A decision whether or not N satisfies the given accept property.

1) Since the firewall tree N has no cycles, N has exactly one directed path from u to v. Without loss of generality, let this path be as shown in Fig. 2.
2) Check whether each packet, that satisfies the predicate $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ of the accept property, is accepted by every firewall in the directed path from u to v in Fig. 2. If each such packet is indeed accepted by every firewall in the directed path, then the firewall tree N satisfies the accept property. Otherwise N does not satisfy the accept property.
3) To check whether each packet that satisfies $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ is accepted by a firewall FW.x in the directed path in Fig. 2, do the following steps:
   a) Let ix be the identifier of the incoming interface of FW.x in the directed path in Fig. 2. For each rule in FW.x, if the first conjunct $F_1 \in S_1$ of this rule is such that ix is not an element of S_1, then remove the rule from FW.x. Otherwise, remove the first conjunct from the rule.
   b) Construct a decision diagram FD.x for the firewall FW.x using Algorithm 1 above. The non-leaf nodes in the constructed FD.x are labeled with the fields F_j where j is in the range 2..d.
   c) For each non-leaf node labeled Fj in FD.x, and for each edge e outgoing from this node, if edge e is labeled with a set whose intersection with Tj is empty, then remove edge e along with each node and each edge that follow e in FD.x.
   d) If the remaining FD.x has only accept leaf nodes, then each packet, that satisfies $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$, is accepted by FW.x. Otherwise, some such packets are discarded by FW.x.

encounters before it reaches its ultimate destination is regarded as the *true protector* of the domain that hosts the ultimate destination.

Next, we describe an algorithm for deciding whether a firewall tree N satisfies a given discard property of N.

Algorithm 2 Verifying Discard Property

Input: A firewall tree N with two distinct domain nodes u and v, and a discard property from u to v in N: $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d \rightarrow discard$.

Output: A decision whether or not N satisfies the given discard property.

1) Since the firewall tree N has no cycles, N has exactly one firewall nod ebetween the two domain nodes u and v. Without loss of generality, let the directed path from u to v be as shown in Fig. 3.
2) Check whether each packet, that satisfies the predicate $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ of the discard property, is discarded by the firewall FW.y in the firewall node y located in the path from u to v in Fig. 3. If each such packet is indeed discarded by FW.y, then the firewall tree N satisfies the discard property. Otherwise, N does not satisfy the discard property.
3) To check whether each packet that satisfies $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ is discarded by FW.y, do the following steps:
   a) Let iy be the identifier of the incoming interface of FW.y in the directed path in Fig. 3. For each rule in Fw.y, if the first $F_1 \in S_1$ of this rule is such that iy is not an element of S_1, then remove the rule from FW.y. Otherwise, remove the first conjunct from the rule.
   b) Construct a decision diagram FD.y for the firewall FW.y using Algorithm 1. The non-leaf nodes in the constructed Fd.y are labeled with the fields F_j where j is in the range 2..d.
   c) For each non-leaf node labeled F_j in FD.y, and for each edge e outgoing from this node, if edge e is labeled with a set whose intersection with T_j is empty, then remove edge e along with each node and each edge that follows e in FD.y.
   d) If the remaining FD.y has only discard leaf nodes, then each packet that satisfies $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ is discarded by FW.y. Otherwise, some such packets are accepted by FW.y.

Step (a) in this algorithm can be regarded as projecting firewall FW.y on its incoming interface y. Also, Step (c) in this algorithm can be regarded as projecting the firewall decision diagram FD.y on the predicate $F_2 \in T_2 \wedge F_3 \in T_3 \wedge \ldots \wedge F_d \in T_d$ of the discard property.

VII. Simulation Results

In this section, we discuss several simulation experiments that we conducted to measure the performance of our algorithms as they are applied to synthetic firewalls and properties, generated at random. (The reason for conducting our experiments on synthetic firewalls and properties is that we were unsuccessful in convincing some administrators of firewall networks to allow us to study the firewalls and properties of their networks.)

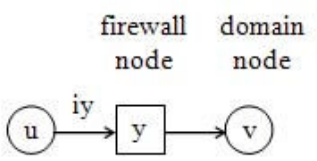


Fig. 3. The only path from domain u to domain v in tree N.

In our first set of experiments, we generated firewalls at random and computed an equivalent FDD for each generated firewall. Each rule in each generated firewall examines five fields, F_1 through F_5, where

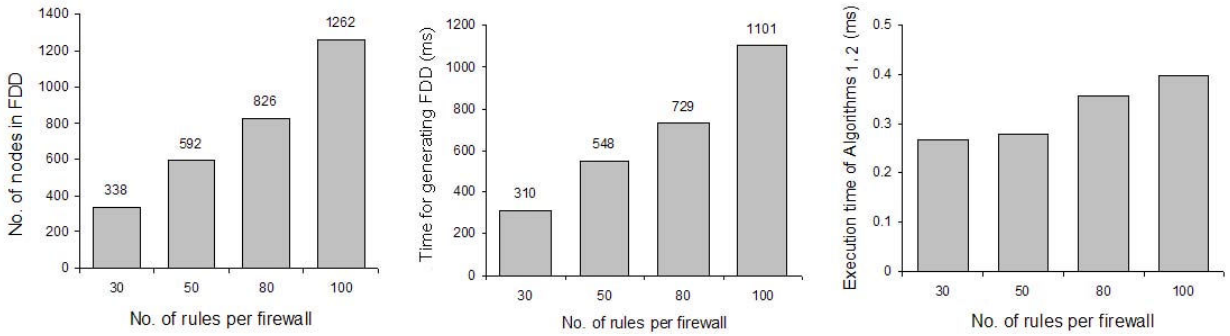
- F_1 stands for the incoming interface
- F_2 stands for the IP address of the original source
- F_3 stands for the IP address of the ultimate destination
- F_4 stands for the destination port number
- F_5 stands for the transport protocol

For simplicity, we assumed that the value domain D_i for each field F_i is as follows: D_1 = {0..10}, D_2 = {0..1000}, D_3 = {0..1000}, D_4 = {0..1000}, D_5 = {0..1}.

For each value of n, n=30, 50, 80, and 100, we performed the following four steps: (1) We generated 10 random firewalls of n rules each. (2) For each generated firewall in Step (1), we computed an equivalent FDD. (3) We computed the average number of nodes in each FDD computed in Step (2). (4) We also computed the average execution time of computing each FDD.

The results of these experiments are shown in Figures 4(a) and 4(b). The results in Fig. 4(a) are particularly interesting. According to the worst case complexity analysis of Algorithm 1, the number of nodes in an FDD, which is equivalent to a firewall, is $O(n^d)$ where n is the number of rules in the firewall, and d is the number of fields that are examined by each rule in the firewall. Thus, for a firewall where n=100 and d=5, one expects that an equivalent FDD has millions or even billions of nodes. But this turned out not to be the case. According to Fig. 4(a), an FDD, which is equivalent to a firewall where n=100 and d=5, has on average 1200 to 1400 nodes. This suggests that the practical complexity of Algorithm 1 is $O(n.d)$, which is much smaller than its worst-case complexity of $O(n^d)$.

In our second set of experiments, we generated 5 random accept or discard properties that are satisfied, for each of the firewalls generated earlier and computed the average execution time (of Algorithm 1 and Algorithm 2, respectively) that is needed to establish that each generated property satisfies its firewall. The results of these experiments are plotted in



(a) Average number of nodes in FDDs generated from firewalls of various sizes

(b) Time for generating FDD

(c) Execution time for checking accept and discard properties

Fig. 4. Simulation Results

Fig. 4(c).

Comparing Fig. 4(b) and 4(c), we conclude that the execution time needed to check whether a given FDD satisfies a given (accept or discard) property is dominated by the execution time needed to generate the FDD from a given firewall.

The experiments described in this section were carried out using a C++ program that has 1130 lines of code. The host machine, son which the experiments were carried out, has the following specifications: (1) processor: Intel Core Duo 1.73 Ghz, (2) memory: 1 GB.

VIII. Concluding Remarks

The technical contributions of this paper are three-fold. First, we formally specify what we mean by a firewall tree. Second, we identify two classes of properties, namely accept and discard properties, of firewall trees. Third, we give two algorithms that can be used to verify whether any given firewall tree satisfies a given accept or discard property of that tree.

Acknowledgment

We are thankful to Anand Iyer for providing programming help in performing the simulations in Section VII. The work of Mohamad G. Gouda is supported in part by grant 0520250 from the National Science Foundation. The work of Alex Liu is supported by the National Science Foundation under Grant No. CNS-0716407.

References

[1] A. Mayer, A.Wool, and E. Ziskind. Fang: A firewall analysis engine. In Proceedings of IEEE Symp. on Security and Privacy, pages 177-187, 2000.

[2] D. Eppstein and S. Muthukrishnan. Internet packet filter management and rectangle geometry. In Symp. on Discrete Algorithms, pages 827-835, 2001.

[3] A. Hari, S. Suri, and G. M. Parulkar. Detecting and resolving packet filter conflicts. In Proceedings of IEEE INFOCOM, pages 1203-1212, 2000.

[4] E. Al-Shaer and H. Hamed. Discovery of policy anomalies in distributed firewalls. In IEEE INFOCOM'04, pages 2605-2616, March 2004.

[5] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan. Conflict classification and analysis of distributed firewall policies. *IEEE Journal on Selected Areas in Communications (JSAC)* , 23(10):2069-2084, 2005.

[6] H. Hamed, E. Al-Shaer, and W. Marrero. Modeling and Verification of IPsec and VPN Security Policies. In Proceedings IEEE International Conference on Network Protocols, Nov 2005.

[7] M. Frantzen, F. Kerschbaum, E. Schultz, and S. Fahmy. A framework for understanding vulnerabilities in firewalls using a dataflow model of firewall internals. *Computers and Security*, 20(3):263-270, 2001.

[8] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen. Analysis of vulnerabilities in internet firewalls. *Computers and Security*, 22(3):214-232, 2003.

[9] A. Wool. A quantitative study of firewall configuration errors. *IEEE Computer* , 37(6):62-67, 2004.

[10] A. X. Liu, M. G. Gouda, H. H. Ma, and A. H. Ngu. Firewall queries. In Proceedings of the 8th International Conference on Principles of Distributed Systems, LNCS 3544, T. Higashino Ed., Springer-Verlag, pages 124-139, December 2004.

[11] M. G. Gouda and A. X. Liu. Structured firewall design. *Computer Networks Journal (Elsevier)*, 51(4):1106–1120, March 2007.

[12] M. G. Gouda and A. X. Liu. Firewall design: consistency, completeness and compactness. In *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems (ICDCS-04)*, pages 320–327, March 2004.

[13] A. X. Liu and M. G. Gouda. Diverse Firewall Design, In Proceedings of the IEEE International Conference on Dependable Systems and Networks (DSN-04), Florence, Italy, June 2004.

[14] R. E. Bryant. Graph-based algorithms for boolean function manipulation. IEEE Trans. on Computers, 35(8):677-691, 1986.

[15] K. Strehl and L. Thiele. Interval diagrams for efficient symbolic verification of process networks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(8):939-956, 2000.

[16] M. G. Gouda and A. X. Liu. A model of stateful firewalls and its properties. In Proceedings of the IEEE International Conference on Dependable Systems and Networks (DSN-05), pages 320-327, June 2005.

[17] Lihua Yuan and Hao Chen and Jianning Mai and Chen-Nee Chuah and Zhendong Su and Prasant Mohapatra FIREMAN: a toolkit for firewall modeling and analysis. In Proceedings of the IEEE Symposium on Security and Privacy, May 2006.

[18] J. Garcła-Alfaro and F. Cuppens and N. Cuppens Analysis of Policy Anomalies on Distributed Network Security Setups In Proceedings of the 11th European Symposium On Research In Computer Security (Esorics), Hamburg, Germany, September, 2006.