Hardness of Firewall Analysis

Ehab S. Elmallah
Department of Computing Science
University of Alberta
Edmonton, T6G 2E8, Canada
E-Mail: elmallah@ualberta.ca

Mohamed G. Gouda
Department of Computing Science
University of Texas at Austin
Austin, Texas 78712, USA
E-Mail: gouda@cs.utexas.edu

***Abstract*—We identify 13 problems whose solutions can significantly enhance our ability to design and analyze firewalls and other packet classifiers. These problems include the firewall equivalence problem, the firewall redundancy problem, the firewall verification problem, and the firewall completeness problem. The main result of this paper is to prove that every one of these problems is NP-hard. Our proof of this result is interesting in the following way. Only one of the 13 problems, the so called slice probing problem, is shown to be NP-hard by a reduction from the well-known 3-SAT problem. Then, the remaining 12 problems are shown to be NP-hard by reductions from the slice probing problem. The negative results of this paper suggest that firewalls designers may need to rely on SAT solvers to solve instances of these 13 problems or may be content with probabilistic solutions of these problems.**

Keywords: Firewalls, Packet Classifiers, Logical Analysis, Equivalence, Redundancy, Verification, Completeness, 3-SAT, and NP-hard

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 [1]. 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 either to discard the packet or accept it and allow it to proceed on its way.

The decision that a firewall makes to discard or accept a packet depends on two factors:

1) The values of the fields in the packet headers
2) The sequence of rules in the firewall that are 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 three sets of packets that are associated with each firewall: (1) the set of packets that are discarded by the firewall, (2) the set of packets that are accepted by the firewall, and (3) the set of packets that are neither discarded nor accepted by the firewall. This third set is usually empty.

The task of designing, verifying, and analyzing a firewall (especially one with thousands of rules, as usually is the case) is not an easy one [2], [3], and [4]. Performing this task properly usually requires solving thousands of instances of the following problems:

1) **Firewall Verification:**
Show that a given firewall discards or accepts a given set of packets
2) **Firewall Implication:**
Show that a given firewall discards (or accepts, respectively) every packet that is discarded (or accepted, respectively) by another given firewall
3) **Firewall Equivalence:**
Show that two given firewalls discard or accept the same set of packets
4) **Firewall Adequacy:**
Show that a given firewall discards or accepts at least one packet
5) **Firewall Redundancy:**
Show that a given discard (or accept, respectively) rule in a given firewall can be removed from the firewall without changing the set of packets that are discarded (or accepted, respectively) by the firewall
6) **Firewall Completeness:**
Show that any given firewall discards or accepts every packet

Efficient algorithms for solving these six problems can benefit the design, verification, and analysis of firewalls. For example, consider the next three scenarios that occur frequently during the design phase, verification phase, and analysis phase of firewalls.

Scenario 1: A firewall designer designs a firewall that is required to accept some specified sets of packets and to discard other specified sets of packets. After the firewall design is completed, the designer needs to verify that indeed the designed firewall accepts every set of packets that it should accept and discards every set of packets that it should discard. Thus, the designer needs to apply an algorithm, that solves the above firewall verification problem, on the designed firewall. Moreover, if the verification shows that the firewall discards a set of packets that should be accepted or accepts a set of packets that should be discarded, then the designer needs to modify the designed firewall and repeat the verification.

Scenario 2: A firewall can be designed through a series of refinement steps that proceeds as follows. Initially, the firewall is designed to accept all packets. Then at each refinement step,

the designer modifies the firewall slightly to make the firewall discard one more set of packets (that the firewall is required to discard). To check the correctness of each refinement step, the designer needs to apply an algorithm, that solves the above firewall implication problem, to check that indeed the firewall at the end of the refinement step discards every packet that is discarded by the firewall at the beginning of the refinement step.

Scenario 3: After a firewall designer completes the design of a firewall, the designer needs to identify the redundant rules in the designed firewall and remove them from the firewall. (Note that removing the redundant rules from a firewall does not affect the sets of packets that are discarded by or accepted by the firewall.) To identify the redundant rules in the designed firewall, the designer needs to apply an algorithm that solves the above firewall redundancy problem, to check whether each rule in the firewall is redundant.

Recognizing the importance of these problems (to the task of designing, verifying, and analyzing firewalls), many researchers have attempted to develop efficient algorithms that can solve these problems in polynomial time. But the efforts of these researchers (including the authors of the current paper) have failed to develop polynomial algorithms for solving any of these problems. And the time complexity of the best known algorithm to solve any of these problems remains exponential.

In this paper, we show that in fact each one of these problems is NP-hard! This paper is the first to show that any significant problem related to the logical analysis of firewalls is NP-hard. Note that the paper not only proves that one or two of these problems are NP-hard but it also proves that many of these problems are NP-hard.

The rest of this paper is organized as follows. In Section II, we formally define the four main concepts of firewalls, namely fields, packets, rules, and firewalls. In Section III, we formally state 13 problems related to the logical analysis of firewalls. In Sections IV through X, we prove that each one of the 13 problems in Section III is NP-hard. Then in Section XI, we outline three research directions that can still enhance our ability to design and analyze firewalls, in light of these negative results. Concluding remarks are in Section XIII.

II. Fields, Packets, Rules, and Firewalls

In this section, we define the four main concepts of firewalls: fields, packets, rules, and firewalls. 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 consecutive integers. This interval is called the domain of the field. A nonempty interval X of consecutive integers can be written as a pair $[y, z]$, where y is the smallest integer in interval X, z is the largest integer in X, and X contains only every integer that is neither smaller than y nor larger than z. Note that if X is $[y, y]$, then X contains only one integer, y.

In this paper, we assume that each packet has d fields, named $f_1, f_2, \cdots$, and f_d. The domain of each field f_j is denoted $D(f_j)$.

(Examples of the d fields in a packet are the source IP address of the packet, the destination IP address of the packet, the transport protocol of the packet, the source port number of the packet, and the destination port number of the packet.)

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

A *rule* in a firewall consists of two parts, a $< predicate >$ and a $< decision >$. A rule is usually written as

$$< predicate > \rightarrow < decision >$$

The $< predicate >$ of a rule is a conjunction of d conjuncts of the form:

$$((f_1 \in X_1) \wedge \cdots \wedge (f_d \in X_d))$$

where each f_j is a field, each X_j is a nonempty interval of consecutive integers taken from the domain $D(f_j)$ of field f_j , and '$\wedge$' is the logical AND operator.

The value of each conjunct $(f_j \in X_j)$ is true iff the value of field f_j is taken from the interval X_j.

The $< decision >$ of a rule is either discard or accept.

A rule whose decision is discard is called a discard rule and a rule whose decision is accept is called an accept rule.

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

$$((f_1 \in X_1) \wedge \cdots \wedge (f_d \in X_d)) \rightarrow < decision >$$

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

A *firewall* is a sequence of rules.

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

1) Packet p matches rule rl
2) Packet p does not match any rule that precedes rule rl in firewall F

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

It follows that for any firewall F and any 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 F' are said to be equivalent iff for every packet p, exactly one of the following three statements holds:

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

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 rule of the form:

$$((f_1 \in X_1) \wedge \cdots \wedge (f_d \in X_d)) \rightarrow < decision >$$

is called an ALL rule iff each interval X_j is the whole domain $D(f_j)$ of field f_j.

Note that every packet matches each ALL rule. Thus, each firewall that has an ALL rule is complete.

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

$$((f_1 \in X_1) \wedge \cdots \wedge (f_d \in X_d)) \rightarrow < decision >$$

where each f_j is a field, each X_j is a nonempty interval of consecutive integers taken from the domain $D(f_j)$ of field f_j, and $< decision >$ is either discard or accept.

A property whose decision is discard is called a discard property, and a property whose decision is accept is called an accept property.

A discard property of the form:

$$((f_1 \in X_1) \wedge \cdots \wedge (f_d \in X_d)) \rightarrow discard$$

is said to discard a packet $(p_1, \cdots, p_d)$ iff the predicate $((p_1 \in X_1) \wedge \cdots \wedge (p_d \in X_d))$ is true. Similarly, an accept property of the form:

$$((f_1 \in X_1) \wedge \cdots \wedge (f_d \in X_d)) \rightarrow accept$$

is said to accept a packet $(p_1, \cdots, p_d)$ iff the predicate $((p_1 \in X_1) \wedge \cdots \wedge (p_d \in X_d))$ is true.

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

a) pr is a discard property and each packet that is discarded by pr is discarded by F
b) pr is an accept property and each packet that is accepted by pr is accepted by F

We end this section by identifying two special classes of firewalls, named discard slices and accept slices. Later in this paper we show that two problems concerning these two special firewall classes are NP-hard. From the fact that these two problems are NP-hard, we show that many problems concerning the design and analysis of general firewalls are also NP-hard.

A firewall that consists of zero or more accept rules followed by an ALL discard rule is called a *discard* slice. Similarly, a firewall that consists of zero or more discard rules followed by an ALL accept rule is called an *accept* slice.

III. Firewall Analysis

In this section we identify 13 problems that need to be solved in order to carry out the logical analysis of firewalls. As discussed below, these problems include firewall verification, firewall implication, and firewall equivalence. Later, we show that each one of these 13 problems is NP-hard. These results indicate that the logical analysis of firewalls is hard, at least theoretically from an asymptotic worst case analysis perspective.

The 13 problems that we identify in this section can be classified into 7 classes: Problems of Slice Probing, Problems of Firewall Adequacy, Problems of Firewall Verification, Problems of Firewall Implication, Problems of Firewall Equivalence, the Problem of Firewall Redundancy, and the Problem of Firewall Completeness.

Problems of Slice Probing: There are two Slice Probing problems, which we denote SP-D and SP-A. These two problems are defined as follows.

SP-D: Probing of Discard Slices:
Design an algorithm that takes as input a discard slice S and determines whether S discards at least one packet

SP-A: Probing of Accept Slices:
Design an algorithm that takes as input an accept slice S and determines whether S accepts at least one packet

Problems of Firewall Adequacy: There are two Firewall Adequacy problems, which we denote FA-D and FA-A. These two problems are defined as follows.

FA-D: Discard-Adequacy of Firewalls:
Design an algorithm that takes as input a firewall F and determines whether F discards at least one packet

FA-A: Accept-Adequacy of Firewalls:
Design an algorithm that takes as input a firewall F and determines whether F accepts at least one packet

The Problem of Firewall Completeness: The Firewall Completeness problem, which we denote FC, is defined as follows:

FC: Completeness of Firewalls:
Design an algorithm that takes as input a firewall F and determines whether every packet is either discarded or accepted by F

Problems of Firewall Verification: There are two Firewall Verification problems, which we denote FV-D and FV-A. These two problems are defined as follows.

FV-D: Discard-Verification of Firewalls:
Design an algorithm that takes as input a firewall F and a discard property pr and determines whether every packet that is discarded by pr is also discarded by F

FV-A: Accept-Verification of Firewalls:
Design an algorithm that takes as input a firewall F and an accept property pr and determines whether every packet that is accepted by pr is also accepted by F

Problems of Firewall Implication: There are two Firewall Implication problems, which we denote FI-D and FI-A. These two problems are defined as follows.

FI-D: Discard-Implication of Firewalls:
Design an algorithm that takes as input two firewalls F_1 and F_2 and determines whether every packet that is discarded by F_1 is also discarded by F_2

FI-A: Accept-Implication of Firewalls:
Design an algorithm that takes as input two firewalls F_1 and F_2 and determines whether every packet that is accepted by F_1 is also accepted by F_2

Problems of Firewall Redundancy: There are two Firewall Redundancy problems, which we denote FR-D and FR-A. These two problems are defined as follows.

FR-D: Discard-Redundancy of Firewalls:
Design an algorithm that takes as input a firewall F and a discard rule rl in F and determines whether the two firewalls F and $F \setminus rl$ discard the same set of packets, where $F \setminus rl$ is the firewall that is obtained after removing rule rl from firewall F

FR-A: Accept-Redundancy of Firewalls:
Design an algorithm that takes as input a firewall F and an accept rule rl in F and determines whether the two firewalls F and $F \setminus rl$ accept the same set of packets, where $F \setminus rl$ is the firewall that is obtained after removing rule rl from firewall F

Problems of Firewall Equivalence: There are two Firewall Equivalence problems, which we denote FE-D and FE-A. These two problems are defined as follows.

FE-D: Discard-Equivalence of Firewalls:
Design an algorithm that takes as input two firewalls F_1 and F_2 and determines whether F_1 and F_2 discard the same set of packets

FE-A: Accept-Equivalence of Firewalls:
Design an algorithm that takes as input two firewalls F_1 and F_2 and determines whether F_1 and F_2 accept the same set of packets

IV. Hardness of the Slice Probing

In this section, we show that the first Slice Probing problem SP-D is NP-hard by a reduction from the 3-SAT problem. We then show that the Slice Probing problems SP-A is NP-hard by a reduction from SP-D. For convenience, we state the 3-SAT problem next

3-SAT:
Design an algorithm that takes as input a Boolean formula BF of the form $BF = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ where each clause C_k is a disjunction of 3 literals taken from the set of Boolean variables $\{v_1, \cdots, v_d\}$, and determines whether BF is satisfiable (i.e. determines whether there is an assignment of Boolean values to the variables $\{v_1, \cdots, v_d\}$ that makes BF true)

The 3-SAT problem is known to be NP-hard [5]. This means that the time complexity of any algorithm that solves this problem is very likely to require exponential time of $O(n \times 2^d)$, where n is the number of clauses and d is the number of variables in the Boolean formula BF.

(To date, progress in solving the 3-SAT problem has resulted in both deterministic and randomized algorithms with reduced complexity. For example, in [6] the authors present a deterministic algorithm that runs in $O(1.473^n)$ time, and in [7] the authors present a randomized algorithm that runs in $O(1.32113^n)$ time on average.)

Next, we describe a polynomial translation of any instance of the 3-SAT problem to an instance of the SP-D problem such that any solution of the 3-SAT instance yields a solution of the SP-D instance and vice versa. The existence of this polynomial translation indicates that the SP-D problem is NP-hard and that the time complexity of any algorithm that solves this problem is very likely to be exponential.

Translating an instance of 3-SAT to an instance of SP-D proceeds as follows:

1. The 3-SAT instance is defined by a Boolean formula BF and the SP-D instance is defined by a discard slice S
2. Each Boolean variable v_j that occurs in formula BF is translated to a field f_j in slice S.
3. The domain of values for each variable v_j is the set $\{false, true\}$ and the domain of values for each field f_j is the set $\{0, 1\}$. Value false of each variable v_j is translated to value 0 of the corresponding field f_j. Similarly, value true of each variable v_j is translated to value 1 of the corresponding field f_j.
4. Each clause C_k in formula BF is translated to an accept rule R_k in slice S as follows. First, if the literal v_j occurs in clause C_k, then the conjunct $(f_j \in [0, 0])$ occurs in the predicate of rule R_k. Second, if the literal $\overline{v_j}$ occurs in clause C_k, then the conjunct $(f_j \in [1, 1])$ occurs in the predicate of rule R_k. Third, if no literal of v_j occurs in clause C_k, then the conjunct $(f_j \in [0, 1])$ occurs in the predicate of rule R_k
5. Add an ALL discard rule at the bottom of slice S

From this translation, an assignment of values $(\mathrm{val}(v_1), \cdots, \mathrm{val}(v_d))$ makes a clause C_k true iff the corresponding packet $(\mathrm{val}(f_1), \cdots, \mathrm{val}(f_d))$ does not match the corresponding accept rule R_k. Thus, we draw the following two conclusions:

1) If there is an assignment of values $(val(v_1), \cdots, val(v_d))$ that makes the Boolean formula BF true, then the corresponding packet $(val(f_1), \cdots, val(f_d))$ does not match any of the accept rules in the discard slice S and matches only the last ALL discard rule. In other words, if the Boolean formula BF is satisfiable, then the discard slice S discards at least one packet
2) If the discard slice S discards at least one packet, then the Boolean formula BF is satisfiable

Therefore, any solution of the 3-SAT instance yields a solution of the SP-D instance and vice versa. This completes our proof of the following theorem.

Theorem 1. Problem SP-D is NP-hard.

Having established that problem SP-D is NP-hard, we can now use this problem to establish that problem SP-A is also NP-hard.

Theorem 2. Problem SP-A is NP-hard.
Proof: We describe a polynomial translation of any instance of the SP-D problem to an instance of the SP-A problem such that any solution of the SP-D instance yields a solution of the SP-A instance and vice versa.

Translating an instance of SP-D to an instance of SP-A proceeds as follows:

1. An instance of SP-D is defined by a discard slice S
2. Replacing every discard (or accept, respectively) decision in S by an accept (or discard, respectively) decision yields an accept slice denoted S'
3. The accept slice S' defines an instance of SP-A

From this translation, packet p is discarded by slice S iff packet p is accepted by slice S'. Thus, we draw the following two conclusions:

1) If slice S discards at least one packet, then slice S' accepts at least one packet
2) If slice S' accepts at least one packet, then slice S discards at least one packet

Therefore, any solution of the SP-D instance yields a solution of the SP-A instance and vice versa.

V. Hardness of Firewall Adequacy

In this section, we employ problem SP-D, which we have shown to be NP-hard in the previous section, to show that the two Firewall Adequacy problems FA-D and FA-A are also NP-hard. First, we show that the FA-D problem is NP-hard by a reduction from the SP-D problem. Then we show that the FA-A problem is NP-hard by a reduction from the FA-D problem.

Theorem 3. Problem FA-D is NP-hard.
Proof: We describe a polynomial translation of any instance of the SP-D problem to an instance of the FA-D problem such that any solution of the SP-D instance yields a solution of the FA-D instance and vice versa.

Translating an instance of SP-D to an instance of FA-D proceeds as follows:

1. An instance of SP-D is defined by a discard slice S
2. Because each discard slice is a special case of a firewall, the discard slice S can be viewed as a firewall denoted F
3. Firewall F defines an instance of FA-D

From this translation, packet p is discarded by slice S iff packet p is discarded by firewall F. Thus, we draw the following two conclusions:

1) If slice S discards at least one packet, then firewall F discards at least one packet
2) If firewall F discards at least one packet, then slice S discards at least one packet

Therefore, any solution of the SP-D instance yields a solution of the FA-D instance and vice versa.

Theorem 4. Problem FA-A is NP-hard.
Proof: We describe a polynomial translation of any instance of the FA-D problem to an instance of the FA-A problem such that any solution of the FA-D instance yields a solution of the FA-A instance and vice versa.

Translating an instance of FA-D to an instance of FA-A proceeds as follows:

1. An instance of FA-D is defined by a firewall F
2. Replacing every discard (or accept, respectively) decision in F by an accept (or discard, respectively) decision yields a new firewall F'
3. Firewall F' defines an instance of FA-A

From this translation, we conclude that firewall F discards at least one packet iff firewall F' accepts at least one packet. Therefore, any solution of the FA-D instance yields a solution of the FA-A instance and vice versa.

VI. Hardness of Firewall Completeness

In this section, we employ problem SP-D, which we have shown to be NP-hard in Section IV, to show that the Firewall Completeness problem FC is NP-hard.

Theorem 5. Problem FC is NP-hard.
Proof: We describe a polynomial translation of any instance of the SP-D problem to an instance of the FC problem such that any solution of the SP-D instance yields a solution of the FC instance and vice versa.

Translating an instance of SP-D to an instance of FC proceeds as follows:

1. An instance of SP-D is defined by a discard slice S
2. Let F denote the firewall that results from removing the last (ALL discard) rule from slice S
3. Firewall F, which consists entirely of accept rules, defines an instance of FC

From this translation, we conclude that the discard slice S discards at least one packet iff firewall F is not complete (i.e., F ignores at least one packet.) Therefore, any solution of the SP-D instance yields a solution of the FC instance and vice versa.

VII. Hardness of Firewall Verification

In this section, we employ problem SP-A, which we have shown to be NP-hard in Section IV, to show that the two Firewall Verification problems, namely FV-D and FV-A, are also NP-hard. First, we show that the FV-D problem is NP-hard by a reduction from the SP-A problem. Then we show that the FV-A problem is NP-hard by a reduction from the FV-D problem.

Theorem 6. Problem FV-D is NP-hard.
Proof: We describe a polynomial translation of any instance of the SP-A problem to an instance of the FV-D problem such that any solution of the SP-A instance yields a solution of the FV-D instance and vice versa.

Translating an instance of SP-A to an instance of FV-D proceeds as follows:

1. An instance of SP-A is defined by an accept slice S
2. Because each accept slice is a special case of a firewall, the accept slice S can be viewed as a firewall denoted F
3. Firewall F and the ALL discard property pr together define an instance of FV-D

From this translation, we conclude that the accept slice S accepts at least one packet iff this packet is discarded by property pr and not discarded by Firewall F. Therefore, any solution of the SP-A instance yields a solution of the FV-D instance and vice versa.

Theorem 7. Problem FV-A is NP-hard.
Proof: We describe a polynomial translation of any instance of the FV-D problem to an instance of the FV-A problem such that any solution of the FV-D instance yields a solution of the FV-A instance and vice versa.

Translating an instance of FV-D to an instance of FV-A proceeds as follows:

1. An instance of FV-D is defined by a firewall F and a discard property pr
2. Replacing every discard (or accept, respectively) decision in F and in pr by an accept (or discard, respectively) decision yields another firewall F' and an accept property pr'
3. Firewall F' and the accept property pr' together define an instance of FV-A

From this translation, we conclude that every packet that is discarded by property pr is discarded by firewall F iff every packet that is accepted by property pr' is accepted by firewall F'. Therefore, any solution of the FV-D instance yields a solution of the FV-A instance and vice versa.

VIII. Hardness of Firewall Implication

In this section, we employ problem FV-D, which we have shown to be NP-hard in the previous section, to show that the two Firewall Implication problems, namely FI-D and FI-A, are also NP-hard. First, we show that the FI-D problem is NP-hard by a reduction from the FV-D problem. Then we show that the FI-A problem is NP-hard by a reduction from the FI-D problem.

Theorem 8. Problem FI-D is NP-hard.
Proof: We describe a polynomial translation of any instance of the FV-D problem to an instance of the FI-D problem such that any solution of the FV-D instance yields a solution of the FI-D instance and vice versa.

Translating an instance of FV-D to an instance of FI-D proceeds as follows:

1. An instance of FV-D is defined by a firewall F and a discard property pr
2. Because each property is a special case of a firewall, the discard property pr can be viewed as a firewall denoted F'
3. The two firewalls F and F' together define an instance of FI-D

From this translation, we conclude that every packet that is discarded by property pr is discarded by firewall F iff every packet that is discarded by firewall F' is discarded by firewall F. Therefore, any solution of the FV-D instance yields a solution of the FI-D instance and vice versa.

Theorem 9. Problem FI-A is NP-hard.
Proof: We describe a polynomial translation of any instance of the FI-D problem to an instance of the FI-A problem such that any solution of the FI-D instance yields a solution of the FI-A instance and vice versa.

Translating an instance of FI-D to an instance of FI-A proceeds as follows:

1. An instance of FI-D is defined by two firewalls F and F'
2. Replacing each discard (or accept, respectively) decision by accept (or discard, respectively) decision in firewalls F and F' yield the two firewalls G and G'
3. The two firewalls G and G' together define an instance of FI-A

From this translation, we conclude that every packet that is discarded by firewall F is discarded by firewall F' iff every packet that is accepted by firewall G is accepted by firewall G'. Therefore, any solution of the FI-D instance yields a solution of the FI-A instance and vice versa.

IX. Hardness of Firewall Redundancy

In this section, we employ problem FV-D, which we have shown to be NP-hard in Section VII, to show that the two Firewall Redundancy problems FR-D and FR-A are also NP-hard. First, we show that the FR-D problem is NP-hard by a reduction from the FV-D problem. Then, we show that the FR-A problem is NP-hard by a reduction from the FR-D problem.

Theorem 10. Problem FR-D is NP-hard.
Proof: We describe a polynomial translation of any instance of the FV-D problem to an instance of the FR-D problem such that any solution of the FV-D instance yields a solution of the FR-D instance and vice versa.

Translating an instance of FV-D to an instance of FR-D proceeds as follows:

1. An instance of FV-D is defined by a firewall F and a discard property pr
2. Because each property can be viewed as a rule, the discard property pr can be viewed as a discard rule denoted rl. Let F' denote the firewall that results from placing rule rl at the top of firewall F. (Note that firewall F is the same as firewall $F' \setminus rl$.)
3. Firewall F' and its top (discard) rule rl together define an instance of FR-D

From this translation, we conclude that every packet that is discarded by the discard property pr is discarded by firewall F iff the two firewalls F and F' discard the same set of packets. Therefore, any solution of the FV-D instance yields a solution of the FR-D instance and vice versa.

Theorem 11. Problem FR-A is NP-hard.
Proof: We describe a polynomial translation of any instance of the FR-D problem to an instance of the FR-A problem such that any solution of the FR-D instance yields a solution of the FR-A instance and vice versa.

Translating an instance of FR-D to an instance of FR-A proceeds as follows:

1. An instance of FR-D is defined by a firewall F and a discard rule rl in F
2. Replacing each discard (or accept, respectively) decision in firewall F by an accept (or discard, respectively) decision yields a new firewall F'. Let rl' denote the accept rule in F' which corresponds to the discard rule rl in F.
3. Firewall F' and the accept rule rl' in F' together define an instance of FR-A

From this translation, we conclude that the two firewalls F and $F \setminus rl$ discard the same set of packets iff the two firewalls F' and $F' \setminus rl'$ accept the same set of packets. Therefore, any solution of the FR-D instance yields a solution of the FR-A instance and vice versa.

X. Hardness of Firewall Equivalence

In this section, we rely on the NP-hardness of the FR-D problem, which is established in the previous section, to show that the two Firewall Equivalence problems FE-D and FE-A are also NP-hard. First, we show that the FE-D problem is NP-hard by a reduction from the FR-D problem. Then, we show that the FE-A problem is NP-hard by a reduction from the FE-D problem.

Theorem 12. Problem FE-D is NP-hard.
Proof: We describe a polynomial translation of any instance of the FR-D problem to an instance of the FE-D problem such that any solution of the FR-D instance yields a solution of the FE-D instance and vice versa.

Translating an instance of FR-D to an instance of FE-D proceeds as follows:

1. An instance of FR-D is defined by a firewall F and a discard rule rl in F
2. Let G and G' denote the two firewalls F and $F \setminus rl$ respectively
3. The two firewalls G and G' together define an instance of FE-D

From this translation, we conclude that the two firewalls F and $F \setminus rl$ discard the same set of packets iff the two firewalls G and G' discard the same set of packets. Therefore, any solution of the FR-D instance yields a solution of the FE-D instance and vice versa.

Theorem 13. Problem FE-A is NP-hard.
Proof: We describe a polynomial translation of any instance of the FE-D problem to an instance of the FE-A problem such that any solution of the FE-D instance yields a solution of the FE-A instance and vice versa.

Translating an instance of FE-D to an instance of FE-A proceeds as follows:

1. An instance of FE-D is defined by two firewalls F and F'
2. Replacing each discard (or accept, respectively) decision by an accept (or discard, respectively) decision in firewalls F and F' yield two new firewalls G and G'
3. Firewalls G and G' together define an instance of FE-A

From this translation, we conclude that the two firewalls F and F' discard the same set of packets iff the two firewalls G and G' accept the same set of packets. Therefore, any solution of the FE-D instance yields a solution of the FE-A instance and vice versa.

XI. Where We Go from Here

Figure 1 shows an outline of our proof, presented in the five sections IV through X, that each one of the 13 problems in Section III is NP-hard. This proof outline is a directed graph where each node represents one problem and where each directed edge from node P to node P' indicates that problem P' is shown to be NP-hard by a reduction from problem P. Note that in this graph, each node P has exactly one incoming edge labeled by a number k to indicate that the NP-hardness of problem P is proven in Theorem k.

From this proof outline, each one of the 13 problems is shown to be NP-hard by an ultimate reduction from the 3-SAT problem. As mentioned in Section IV, the time complexity of any algorithm that solves the 3-SAT problem is very likely to be of $O(n \times 2^d)$, where n is the number of clauses in the Boolean formula and d is the number of Boolean variables in the Boolean formula.

Thus, assuming that the firewall fields are all binary, the time complexity of any algorithm that solves any of the 13 problems in Section III is very likely to be of $O(n \times 2^d)$, where n is the number of rules in a firewall, and d is the number of bits that are checked by the firewall rules in the headers of every packet. For most firewalls, n is at most 2000 rules, and d is at most 120 bits. Therefore, assuming that the firewall fields are all binary, the time complexity of any algorithm that solves any of the 13 problems in Section III is very likely to be of $O(2000 \times 2^{120})$.

At first, this large time complexity may discourage many researchers from trying to solve any of the 13 problems in Section III. But it turns out that researchers can take advantage of the following three techniques in order to avoid this large complexity in many practical situations.

1. Using SAT-Solvers:
As discussed in [8], each instance of the 13 problems in Section III can be translated into an instance of the SAT problem and then can be easily solved (in many practical situations) using any of the available SAT-solvers, such as Minisat [9].

Indeed the experimental results reported in [8] are impressive. For example, it is shown that many instances of the Firewall Equivalence problem, the Firewall Implication problem, and the Firewall Redundancy problem can all be solved using the Minisat solver [9] and the firewall generator Classbench [10]. More importantly, solving each of these problem instances, which involves one or two firewalls of about 2,000 rules each, takes less than 5 Seconds.

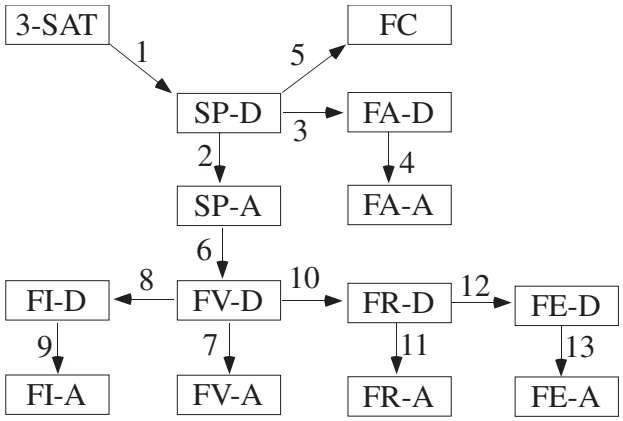

Fig. 1: Hardness reductions between 13 firewall analysis problems

2. Adopting Integer Fields:
The large time complexity of $O(2000 \times 2^{120})$ for solving any of the 13 problems in Section III is based on our assumption that the firewalls in these problems have a large number (around 120) of Boolean fields. Technically, this assumption can be replaced by assuming that the firewalls in these problems have a small number (around 5) of integer fields. Adopting this new assumption, it is shown in [11], [12], [13], [14], and [15] that there are algorithms, whose time complexity is of $O(n^{e+1})$, that solve the Slice Probing problem, the Firewall Verification problem, and the Firewall Redundancy problem. In this case, n is the number of rules in a firewall and e is the number of integer fields that are checked by the firewall rules in the headers of every packet. For most firewalls, n is at most 2000 rules, and e is at most 5 integer fields. Therefore, the time complexity of any algorithm that solves any of the 13 problems in Section III is very likely to be of $O(2000^6)$ which is much smaller than $O(2000 \times 2^{120})$.

3. Accepting Probabilistic Solutions:
The large time complexity of $O(n^{e+1})$ for any algorithm to solve any of the 13 problems in Section III is based on the implicit requirement that the algorithm be deterministic. It is possible to drastically reduce this time complexity if one is willing to accept probabilistic algorithms that solve these problems.

For example, a probabilistic algorithm for solving the Firewall Verification problem is proposed in [16]. This algorithm determines whether any given firewall satisfies any given property. The time complexity of this algorithm is optimally linear of $O(n \times e)$, where n is the number of rules in the given firewall and e is the number of integer fields that are checked by the firewall rules in the headers of each packet. The only problem of this algorithm is that sometimes when the algorithm returns a determination that the given firewall satisfies the given property, the returned determination is incorrect. A large simulation study showed that the probability of an incorrect determination is negligible.

XII. Related Work

The importance of the logical analysis and verification of firewalls has been recognized since the year 2000 [1]. This recognition has led early on to some attempts to identify configuration errors and vulnerabilities in firewalls that were in operation at the time [2], [3], and [4]. These early attempts, though useful in practice, did not develop into a mature theory for the logical analysis and verification of firewalls.

Later on, a robust and full theory for the logical analysis and verification of firewalls was developed [11]–[15]. The objective of this theory was to design efficient algorithms that can solve: firewall equivalence problems [14], firewall redundancy problems [12], and [13], and firewall verification problems [11] and [15].

It turns out that the time complexity of each algorithm that was designed in this theory is exponential! Yet until the current paper, no one was able to prove that any problem in this theory is NP-hard. The current paper not only proves that one or two problems in this theory are NP-hard but it also proves that many problems in the theory are NP-hard.

The fact, that the time complexity of all algorithms in the theory of logical analysis of firewalls is exponential, was alarming. This alarm led researchers to propose two new research directions. First, some researchers proposed to design probabilistic algorithms for solving the problems in the theory [11]. Second, other researchers proposed to rely on SAT solvers to solve the problems in the theory [8] and [9]. The results in the current paper will undoubtedly bolster and add credence and significance to these new research directions, as discussed in Section XI.

XIII. Concluding Remarks

In this paper, we identified 13 important problems related to the analysis of firewalls and showed that each one of these problems is NP-hard. This means that the time complexity of any algorithm that can solve any of these problems is very likely to be exponential.

Our proofs of these NP-hardness results were based on reductions from the relatively new problem of Slice Probing. This fact confirms the central role that the Slice Probing problem plays in the analysis of firewalls. Future research in the analysis of firewalls should be mindful of this problem.

Some of the 13 problems discussed in this paper can be shown to be NP [5]. Examples of these problems are the Slice

Probing problems. The remaining problems can be shown to be co-NP [5]. Example of these problems are the Firewall Implications problems.

It is possible to think of other problems related to the analysis of firewalls and show that these problems are also NP-hard by reductions from the 13 problems in Section III. For example consider the following problem

Firewall Exclusion:
Show that if any given firewall discards (or accepts, respectively) a packet, then another given firewall does not discard (or does not accept, respectively) the same packet

We believe that this problem can be shown to be NP-hard by a reduction from the Firewall Implication problem.

In Section XI, we pointed out three research directions that can be pursued in order to enhance our ability to design and analyze firewalls, in light of the NP-hardness results in this paper.

REFERENCES

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

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

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

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

[5] M. R. Garey and D. S. Johnson, *Computers and intractability : A guide to the theory of NP-completeness*. San Francisco: W. H. Freeman, 1979.

[6] T. Brueggemann and W. Kern, "An improved deterministic local search algorithm for 3-SAT," *Theoretical Computer Science*, vol. 329, no. 13, pp. 303 – 313, 2004.

[7] K. Iwama, K. Seto, T. Takai, and S. Tamaki, "Improved randomized algorithms for 3-sat," in *Algorithms and Computation*, ser. Lecture Notes in Computer Science, O. Cheong, K.-Y. Chwa, and K. Park, Eds. Springer Berlin Heidelberg, 2010, vol. 6506, pp. 73–84.

[8] S. Zhang, A. Mahmoud, S. Malik, and S. Narain, "Verification and synthesis of firewalls using SAT and QBF," in *20th IEEE International Conference on Network Protocols (ICNP)*, 2012, pp. 1–6.

[9] N. Eén and N. Sörensson, "An extensible sat-solver," in *Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, E. Giunchiglia and A. Tacchella, Eds. Springer Berlin Heidelberg, 2004, vol. 2919, pp. 502–518.

[10] D. Taylor and J. Turner, "Classbench: A packet classification benchmark," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 499–511, 2007.

[11] H. B. Acharya at al., "Projection and division: Linear space verification of firewalls," in *Proceedings of the 30th International Conference on Distributed Computing Systems (ICDCS)*, 2010, pp. 736–743.

[12] H. B. Acharya and M. G. Gouda, "Firewall verification and redundancy checking are equivalent," in *Proceedings of the 30th IEEE International Conference on Computer Communication (INFOCOM)*, 2011, pp. 2123–2128.

[13] A. X. Liu et al., "Complete redundancy removal for packet classifiers in TCAMs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 424–437, 2010.

[14] A. X. Liu and M. G. Gouda, "Diverse firewall design," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 1237–1251, 2008.

[15] 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 *17th IEEE International Conference on Network Protocols (ICNP)*, 2009, pp. 123–132.

[16] H. B. Acharya et al., "Linear-time verification of firewalls," in *Proceedings of the 17th IEEE International Conference on Network Protocols (ICNP)*, 2009, pp. 133–140.