First Step Towards Automatic Correction of Firewall Policy Faults

Fei Chen Alex X. Liu
Computer Science and Engineering
Michigan State University

JeeHyun Hwang Tao Xie
Computer Science
North Carolina State University

What do we do here?

- Most firewall policies are poorly configured and contain faults. [Wool 2004 & 2010]
  - A coworker may mess up your firewall rules
  - Any modification may introduce firewall faults.

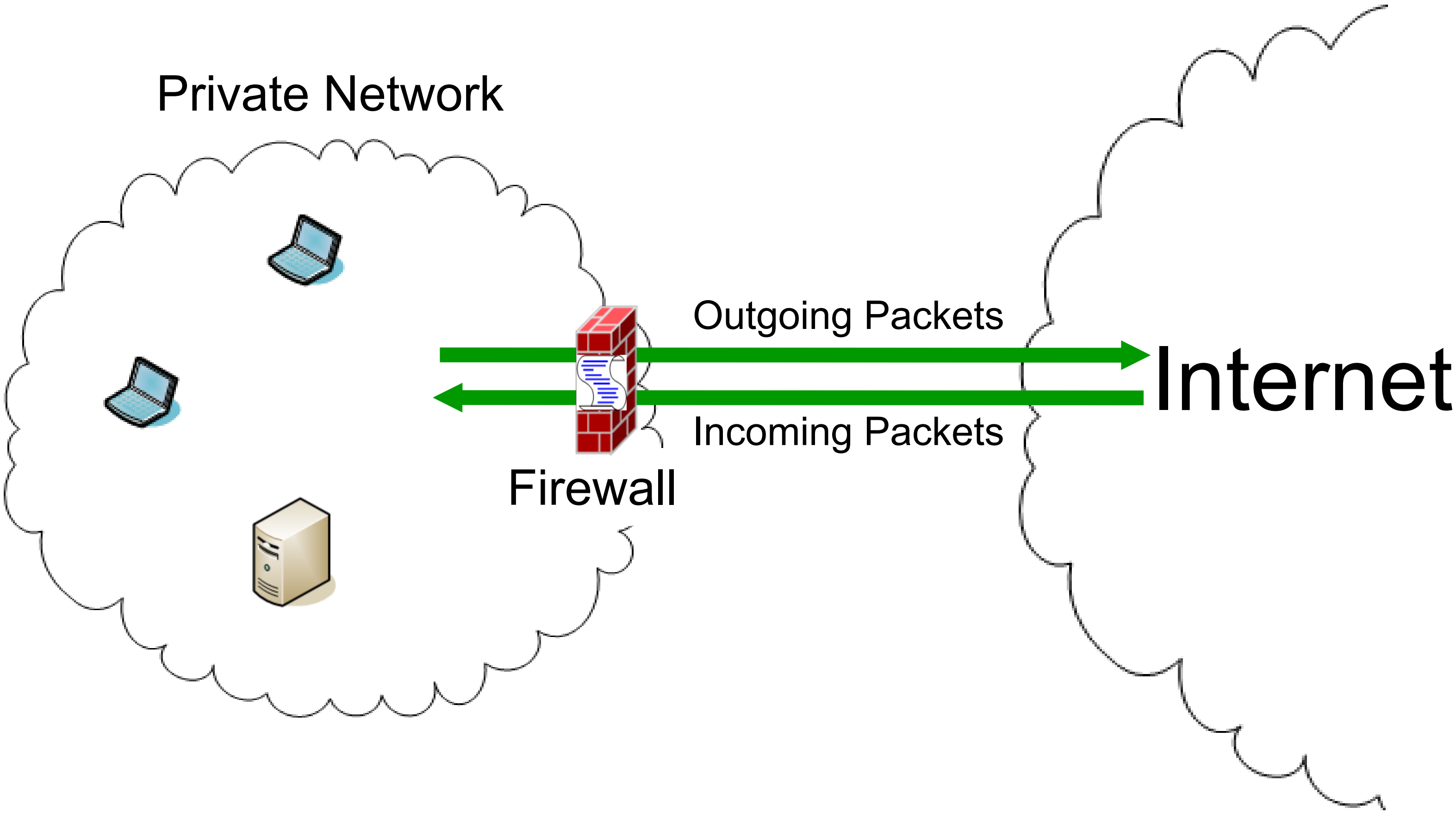
- We invent methods for fixing firewall policies automatically.
  - We first model 5 types of faults.
  - For each type of faults, we develop an algorithm to fix them.
  - Given a faulty firewall policy, we propose a systematic method to fix the faults automatically using the 5 algorithms.

Roadmap

- Background
  - Firewalls
  - Firewall Policies
  - Firewall Policy Faults
- Technical Challenges
- Fault model of firewall policies
  - Five types of faults
- Problem formalization
- Our solution
- Experimental results

Background – Firewalls

- A firewall checks all outgoing and incoming packets
- The firewall policy decides whether to accept or discard a packet



Background – Firewall Policies

- A firewall policy is usually specified as a sequence of rules
- Each rule consists of a predicate and a decision.
  - A predicate typically includes five fields:
    source IP, destination IP, source port, destination port, protocol type
  - Typical decisions are accept and discard.

Firewall Policy

	Src IP	Dst IP	Src Port	Dst Port	Protocol	Decision
r_1	1.2.3.*	192.168.1.1	*	25	TCP	Accept
r_2	1.2.3.9	192.168.1.1	*	25	*	Discard
r_3	*	*	*	*	*	Discard

Packet

Src IP	Dst IP	Src Port	Dst Port	Protocol	Payload
1.2.3.5	192.168.1.1	78	25	TCP	

- Conflict Resolution: first-match

Background – Firewall Policy Faults

- Most firewall policies are poorly configured and contain faults. [Wool 2004 & 2010]
- It is dangerous to have faults in a firewall policy. A policy fault
  - either allows malicious traffic to sneak into the private network
  - or blocks legitimate traffic and disrupts normal business processes
- A faulty policy evaluates some packets to unexpected decisions.
  - Such packets are called misclassified packets of a faulty firewall policy
- Manually locating and correcting firewall faults are impractical.
  - A firewall may consist of thousands of rules
- Automatically correcting firewall faults is an important problem.

Roadmap

- Background
  - Firewalls
  - Firewall Policies
  - Firewall Policy Faults
- Technical Challenges
- Fault model of firewall policies
  - Five types of faults
- Problem formalization
- Our solution
- Experimental results

Three Key Technical Challenges

- It is difficult to determine the number of policy faults and the type of each fault.
  - A set of misclassified packets can be caused by different types of faults and different number of faults.

- It is difficult to correct a firewall fault.
  - A firewall policy may consists of a large number of rules.
  - Each rule has a predicate over multi-dimensional fields.

- It is difficult to correct a fault without introducing other faults
  - Due to the first match, correcting faults in a firewall rule affects the functionally of all the subsequent rules.

Roadmap

- Background
  - Firewalls
  - Firewall Policies
  - Firewall Policy Faults
- Technical Challenges
- Fault model of firewall policies
  - Five types of faults
- Problem formalization
- Our solution
- Experimental results

Fault Model of Firewall Policies (1/2)

- We propose a fault model that includes five types of faults

(1) Wrong order: the order of firewall rules is wrong.

	Src IP	Dst IP	Src Port	Dst Port	Protocol	Decision
r_1	1.2.3.*	192.168.1.1	*	25	TCP	Accept
r_2	1.2.3.9	192.168.1.1	*	25	*	Discard

Correction technique: Order Fixing

(2) Missing rules: some rules are missed in the firewall policy.

r^*

	Src IP	Dst IP	Src Port	Dst Port	Protocol	Decision
r_1	1.2.3.*	192.168.1.1	*	25	TCP	Accept
r_2	1.2.3.9	192.168.1.1	*	25	*	Discard

Correction technique: Rule Addition

(3) Wrong predicates: the predicates of some rules are wrong.

	Src IP	Dst IP	Src Port	Dst Port	Protocol	Decision
r_1	1.2.3.*	192.168.1.1	*	25	TCP	Accept

Correction technique: Predicate Fixing

Fault Model of Firewall Policies (2/2)

(4) Wrong decisions: the decisions of some rules are wrong.

	Src IP	Dst IP	Src Port	Dst Port	Protocol	Decision
r_1	1.2.3.*	192.168.1.1	*	25	TCP	Accept
r_2	1.2.3.9	192.168.1.1	*	25	*	Discard

Correction technique: Decision Fixing

(5) Wrong extra rules: some rules are not needed in the policy.

	Src IP	Dst IP	Src Port	Dst Port	Protocol	Decision
r_1	1.2.3.*	192.168.1.1	*	25	TCP	Accept
~~r_2~~	~~1.2.3.9~~	~~192.168.1.1~~	~~*~~	~~25~~	~~*~~	~~Discard~~
r_3	*	*	*	*	*	Discard

Correction technique: Rule Deletion

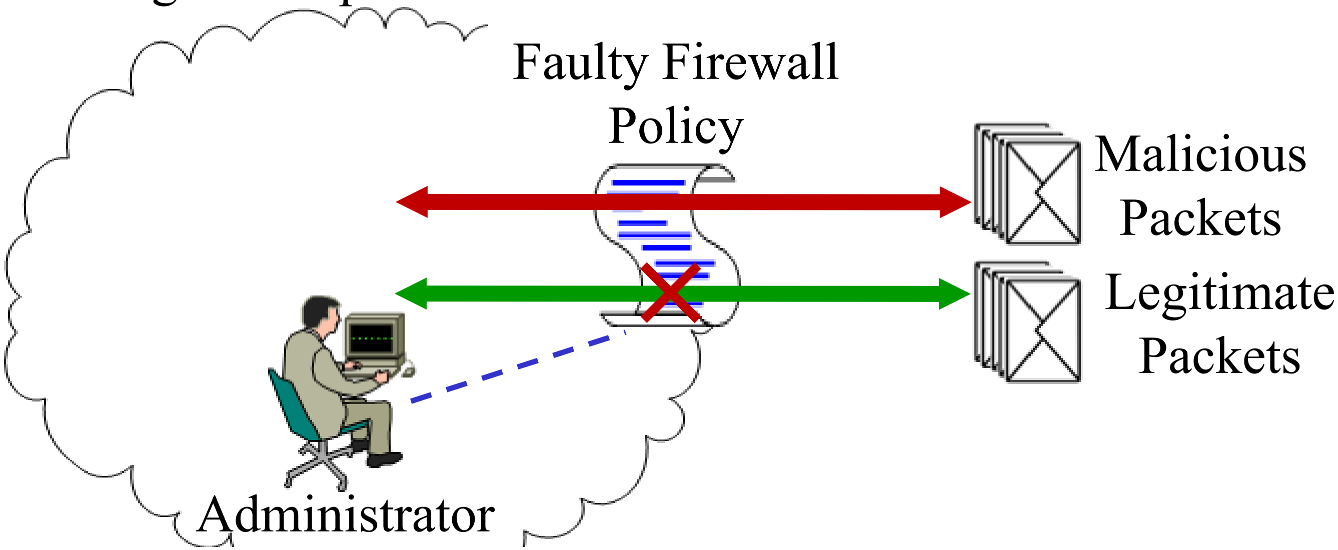
Each operation of these five techniques is called a modification.

Roadmap

- Background
  - Firewalls
  - Firewall Policies
  - Firewall Policy Faults
- Technical Challenges
- Fault model of firewall policies
  - Five types of faults
- Problem formalization
- Our solution
- Experimental results

Detection of Faulty Firewall Policies

- A faulty firewall policy is detected when
  - administrators find that the policy allows some malicious packets or blocks some legitimate packets.



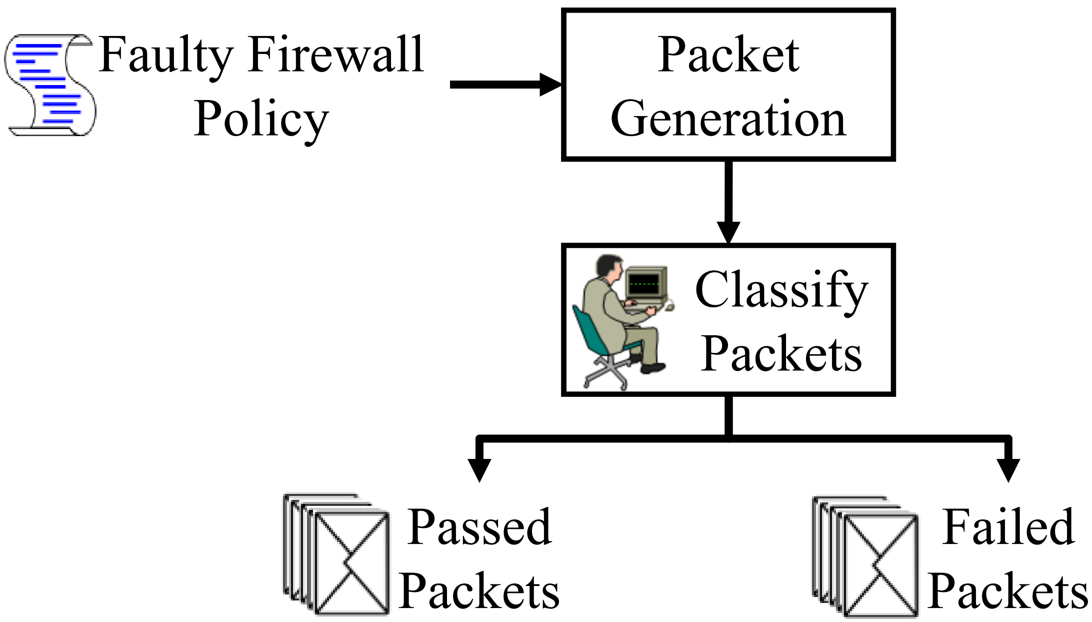
- These packets cannot provide enough information about the faults
  - The number of these observed packets is typically small
- Bruteforce testing every possible packets needs 2^{104}
- How to generate test packets for faulty firewall policies?

Generating Test Packets for Faulty Policies

- We employ the automated packet generation techniques in [Hwang et al. 2008] to generate test packets
- Administrators identify passed/failed tests automatically or manually

According to security requirements for the firewall policy,

- If the decision of a packet is correct, administrators classify it as a passed test.
- Otherwise, administrators classify it as a failed test.



Problem Statement

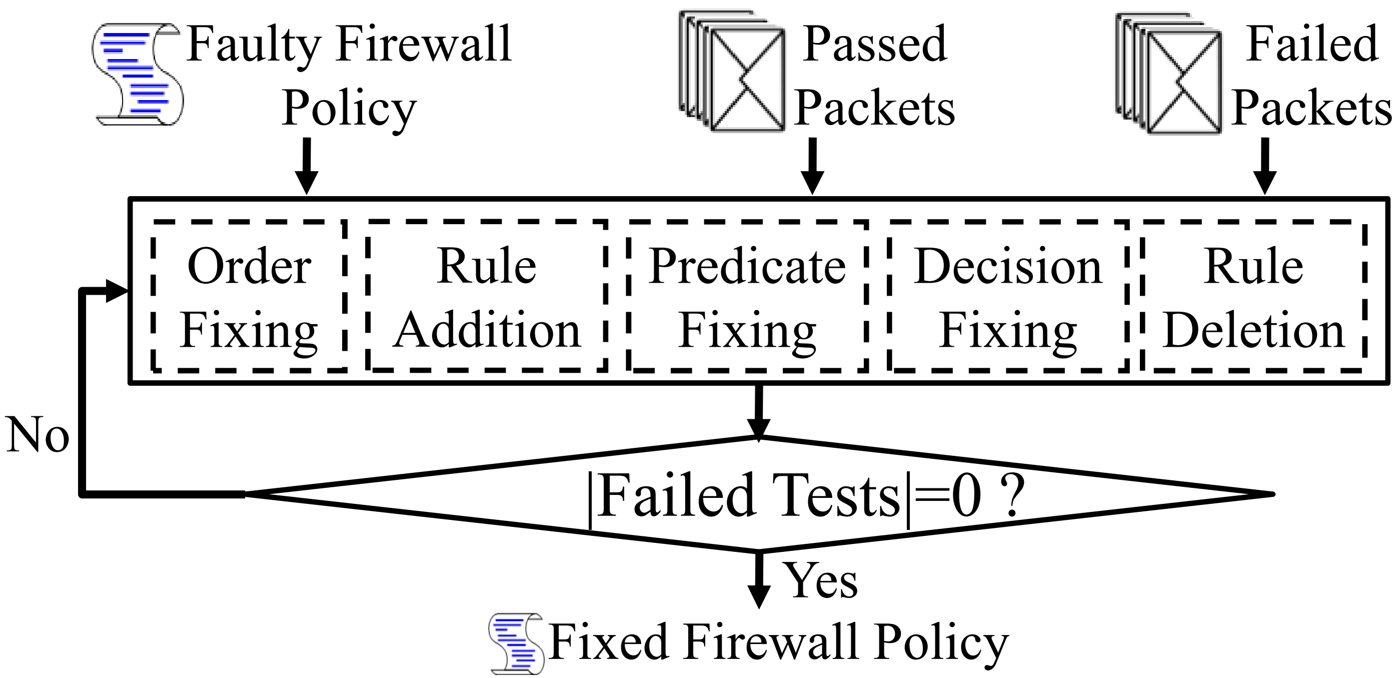
- Input:
  (1) A faulty firewall policy FW
  (2) A set of passed tests PT, $|PT| \geq 0$
  (3) A set of failed tests FT, $|FT| > 0$
- Output:
  A sequence of modifications $<M_1, \ldots, M_m>$, where M_j ($1 \leq j \leq m$) denotes one modifition, satisfies the following two conditions:
  (1) After applying $<M_1, \ldots, M_m>$ to FW, all tests in PT and FT become passed tests.
  (2) No other sequence that satisfies the first condition has the smaller number of modifications than m.
- This is a global optimization problem and hard to solve because
  - a policy may consist of a large number of rules, and
  - different combinations of modifications can be made.

Roadmap

- Background
  - Firewalls
  - Firewall Policies
  - Firewall Policy Faults
- Technical Challenges
- Fault model of firewall policies
  - Five types of faults
- Problem formalization
- Our solution
- Experimental results

Automatic Correction of Firewall Policy Faults

- We propose a greedy algorithm to address this problem.
  - For each step, we correct one fault in the policy such that |PT| increases.
  - To determine which technique should be used, we try the five correction techniques and then find the one that maximizes |PT|.



Running Example

$r_1: F_1 \in [1, 5] \wedge F_2 \in [1, 10] \rightarrow a$
$r_2: F_1 \in [1, 6] \wedge F_2 \in [3, 10] \rightarrow a$
$r_3: F_1 \in [6,10] \wedge F_2 \in [1, 3] \rightarrow d$
$r_4: F_1 \in [7,10] \wedge F_2 \in [4, 8] \rightarrow a$
$r_5: F_1 \in [1,10] \wedge F_2 \in [1, 10] \rightarrow d$

A faulty firewall policy

$p_1: (3, 2) \rightarrow a$
$p_2: (5, 7) \rightarrow a$
$p_3: (6, 7) \rightarrow a$
$p_4: (7, 2) \rightarrow d$
$p_5: (8,10) \rightarrow d$

A set of passed tests

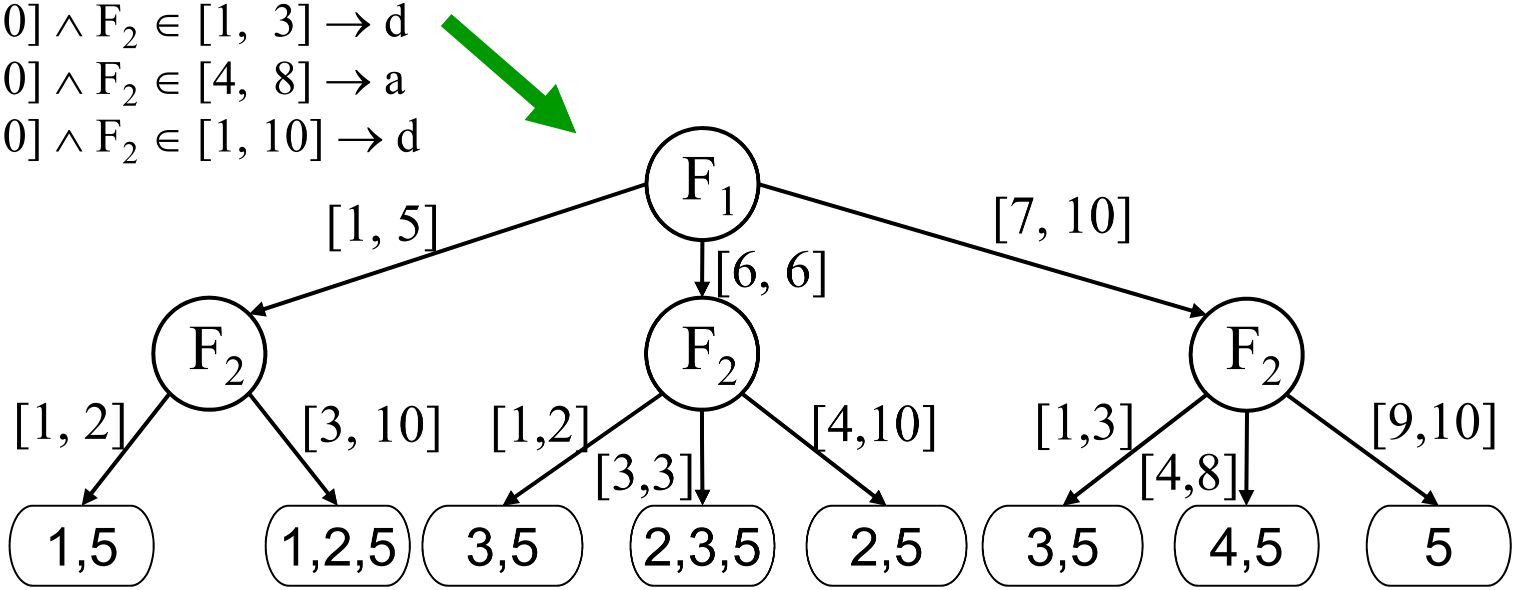
$p_6: (6, 3) \rightarrow d$
$p_7: (7, 9) \rightarrow a$
$p_8: (8, 5) \rightarrow d$

A set of failed tests

Order Fixing (1/2)

- Swapping every two rules is computationally expensive.
  - There are (n-1)(n-2)/2 pairs of rules that can be swapped
- We use all-match firewall decision diagrams (all-match FDDs) [Liu et al. 2008] as the core data structure.
  - Any firewall policy can be converted to an equivalent all-match FDD.

$r_1: F_1 \in [1, 5] \wedge F_2 \in [1, 10] \rightarrow a$
$r_2: F_1 \in [1, 6] \wedge F_2 \in [3, 10] \rightarrow a$
$r_3: F_1 \in [6,10] \wedge F_2 \in [1, 3] \rightarrow d$
$r_4: F_1 \in [7,10] \wedge F_2 \in [4, 8] \rightarrow a$
$r_5: F_1 \in [1,10] \wedge F_2 \in [1, 10] \rightarrow d$

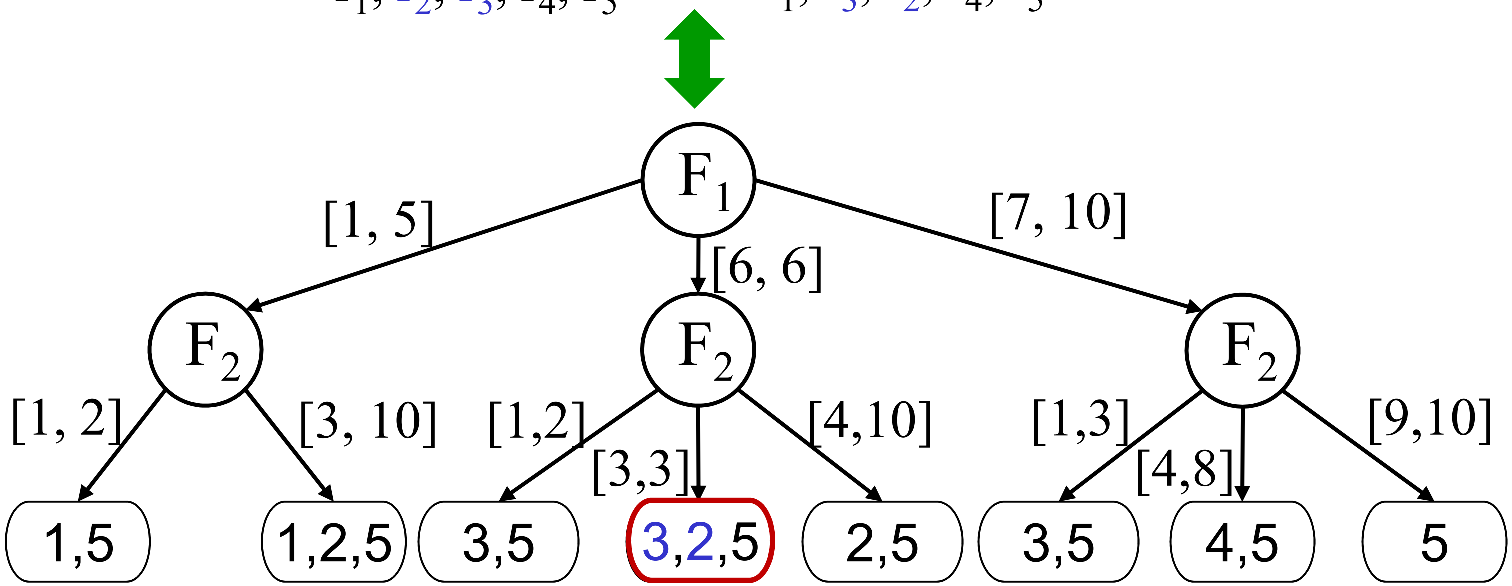


Order Fixing (2/2)

- All-match FDD has the following nice property.

  Swapping two rules is equivalent to swapping the sequence numbers of the two rules in the terminal nodes of all-match FDD

$$\langle r_1, r_2, r_3, r_4, r_5\rangle \Rightarrow \langle r_1, r_3, r_2, r_4, r_5\rangle$$



- For the running example, this technique can find that swapping r_2 and r_3 can increase |PT| by 1
  - change the failed test (6, 3) → d to a passed test

Rule Addition

- Bruteforce addition for each position is computationally expensive
  - The number of possible rules that can be added for each position is $O(2^{204})$.
- The basic idea of rule addition is that for each position
  - Find all possible failed tests that can be corrected by adding a rule

$r^*: F_1 \in [\ ,\ ] \wedge F_2 \in [\ ,\ ] \rightarrow dec$	$p_7: (7, 9) \rightarrow a$	$p_6: (6, 3) \rightarrow d$	$p_8: (8, 5) \rightarrow d$
$r_1: F_1 \in [1, 5] \wedge F_2 \in [1, 10] \rightarrow a$			
$r^*: F_1 \in [\ ,\ ] \wedge F_2 \in [\ ,\ ] \rightarrow dec$	$p_7: (7, 9) \rightarrow a$	$p_6: (6, 3) \rightarrow d$	$p_8: (8, 5) \rightarrow d$
$r_2: F_1 \in [1, 6] \wedge F_2 \in [3, 10] \rightarrow a$		$p_6: (6, 3) \rightarrow d$	
$r^*: F_1 \in [\ ,\ ] \wedge F_2 \in [\ ,\ ] \rightarrow dec$	$p_7: (7, 9) \rightarrow a$		$p_8: (8, 5) \rightarrow d$
$r_3: F_1 \in [6, 10] \wedge F_2 \in [1, 3] \rightarrow d$			
$r^*: F_1 \in [\ ,\ ] \wedge F_2 \in [\ ,\ ] \rightarrow dec$	$p_7: (7, 9) \rightarrow a$		$p_8: (8, 5) \rightarrow d$
$r_4: F_1 \in [7, 10] \wedge F_2 \in [4, 8] \rightarrow a$			$p_8: (8, 5) \rightarrow d$
$r^*: F_1 \in [\ ,\ ] \wedge F_2 \in [\ ,\ ] \rightarrow dec$	$p_7: (7, 9) \rightarrow a$		
$r_5: F_1 \in [1, 10] \wedge F_2 \in [1, 10] \rightarrow d$			

  - Compute a rule that matches the maximum number of failed tests
    - For adding a rule between r_1, r_2, we can compute $F_1 \in [6, 8] \wedge F_2 \in [3, 5] \rightarrow d$ to correct two failed tests $p_6: (6, 3) \rightarrow d$ and $p_8: (8, 5) \rightarrow d$.

Evaluation Setup

- We generate faulty firewall policies from 40 real-life policies.
  - Each faulty policy contains one type of fault, and the number of faults ranges from 1 to 5.
  - For each faulty policy, we employed the packet generating technique [Hwang et al. 2008] and then classified them into passed and failed tests
  - We applied our greedy algorithm to produce the fixed policy.
- Methodology
  - Difference ratio over FW_{real}, FW_{faulty}, and FW_{fixed}

$$\frac{\Delta(FW_{real}, FW_{fixed})}{\Delta(FW_{real}, FW_{faulty})}$$

  - The average number of modifications

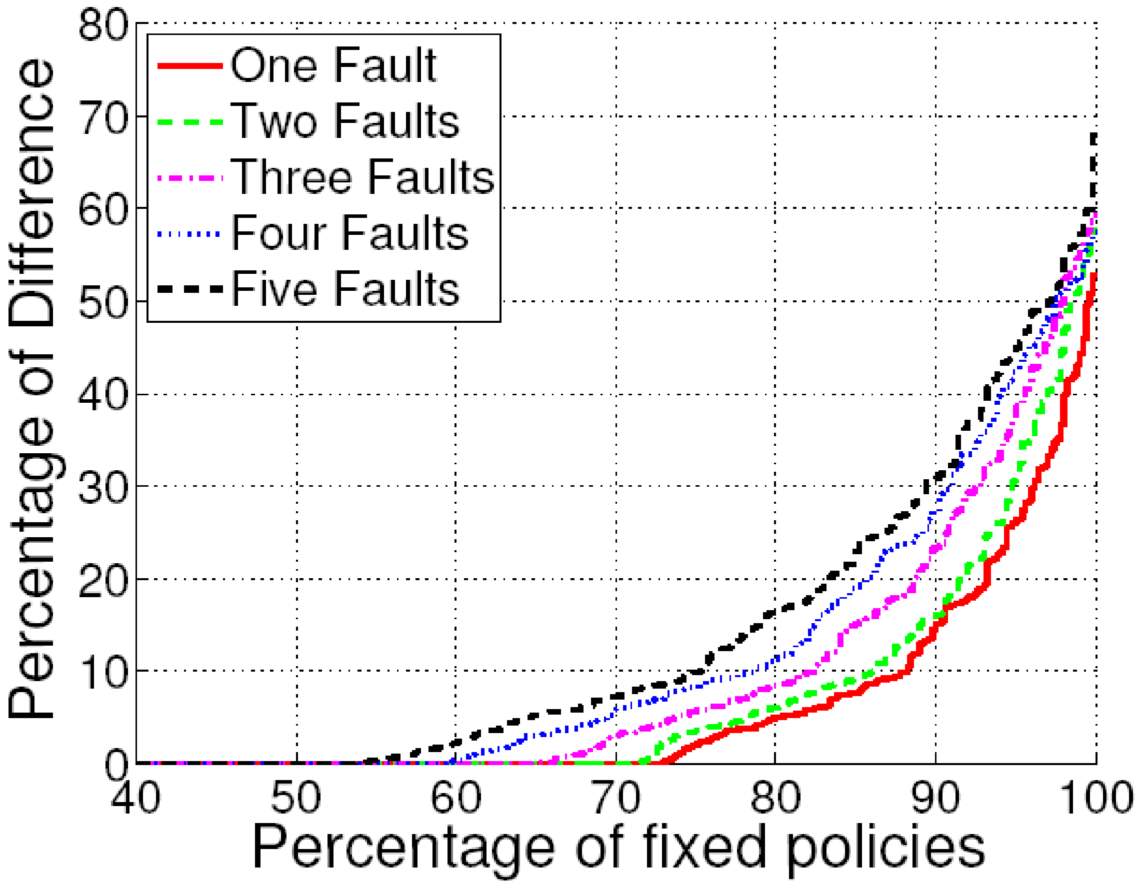
Roadmap

- Background
  - Firewalls
  - Firewall Policies
  - Firewall Policy Faults
- Technical Challenges
- Fault model of firewall policies
  - Five types of faults
- Problem formalization
- Our solution
- Experimental results

Effectiveness (1/4)

- For wrong decision faults

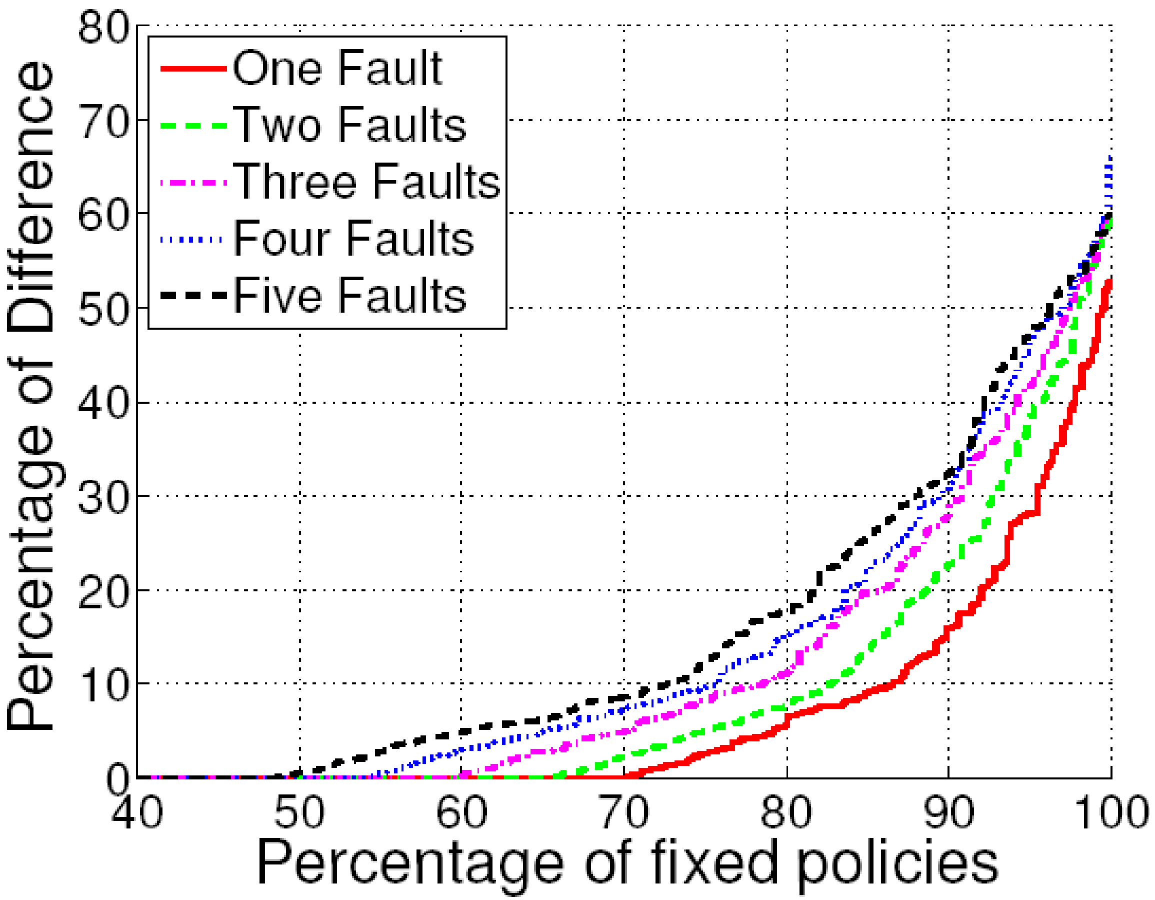
  The percentages of fixed policies that are equivalent to their corresponding real-life policies are 73.5%, 68.8%, 63.7%, 59.3%, and 53.8%, respectively.



Effectiveness (2/4)

- For wrong order faults

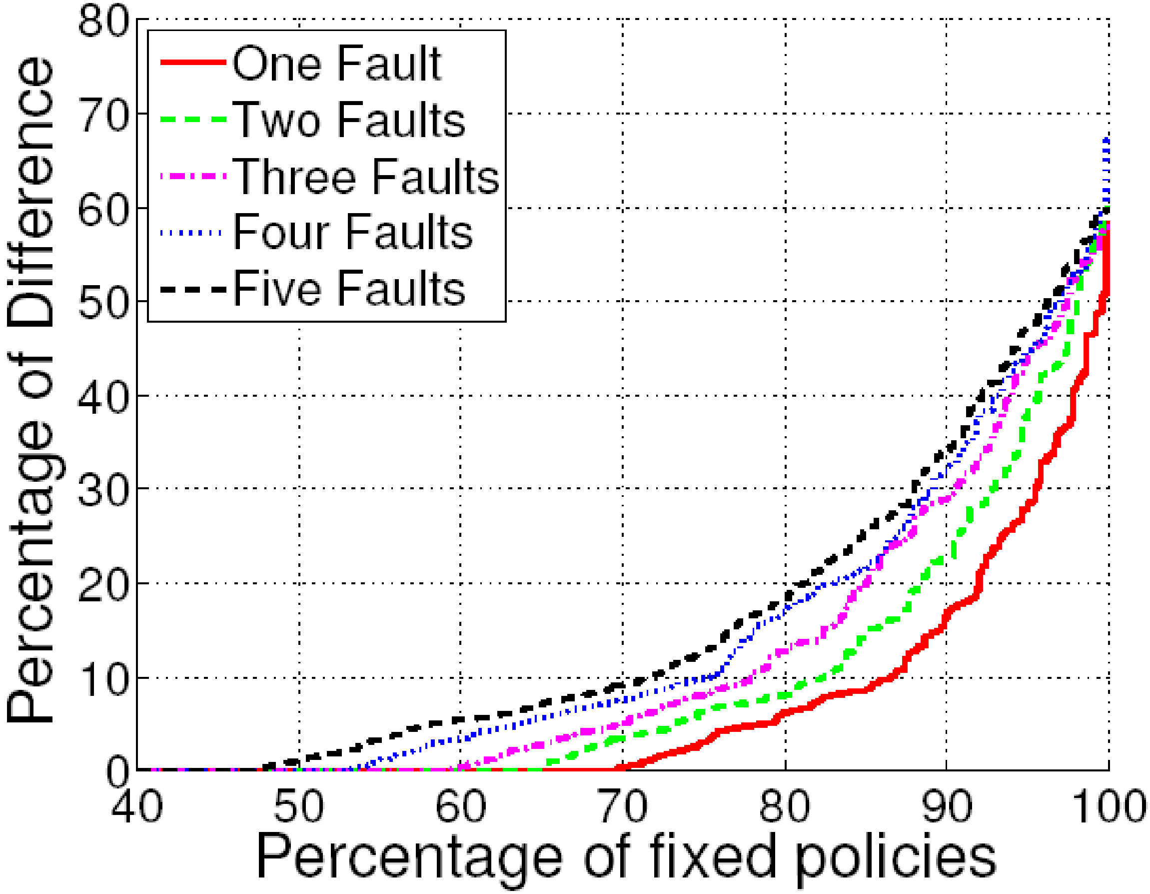
  The percentages of fixed policies that are equivalent to their corresponding real-life policies are 69.7%, 64.2%, 59.7%, 54.3%, and 48.9%, respectively.



Effectiveness (3/4)

- For wrong extra rule faults

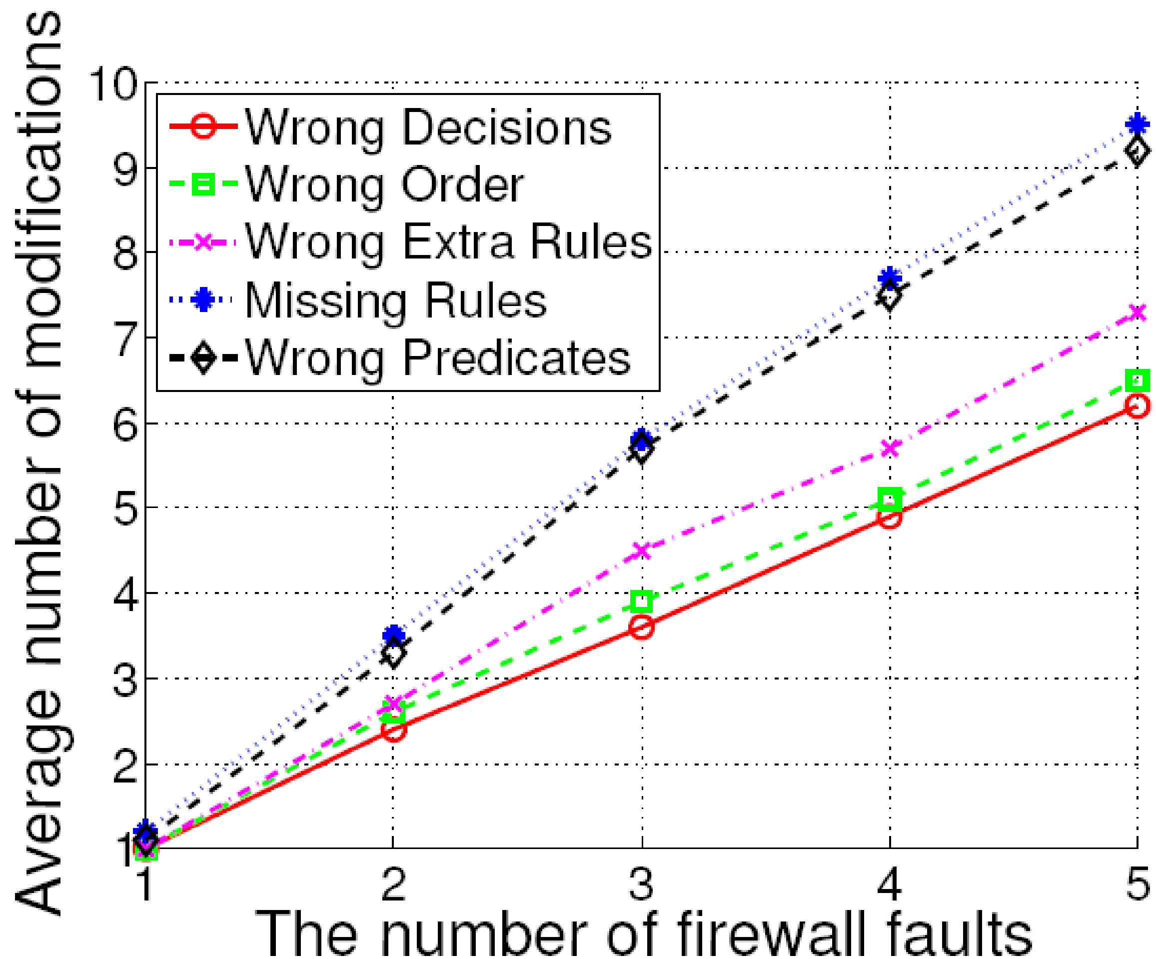
  The percentages of fixed policies that are equivalent to their corresponding real-life policies are 68.3%, 63.5%, 59.3%, 53.2%, and 47.3%, respectively.



Effectiveness (4/4)

- In terms the number of modifications

  The number of modifications of our approach is close to the minimum number.



Contributions

- Propose the first comprehensive fault model for firewall policies
- Propose the first systematic approach that can automatically correct all or part of the misclassified packets of a faulty policy.
- Conduct extensive experiments on real-life firewall policies to evaluate the effectiveness of our approach.

Questions

Thank you!