# Detection and Removal of Firewall Misconfiguration

F. Cuppens, N. Cuppens, and **J. Garcia-Alfaro***

*Ecole Nationale Supérieure des Télécommunications de Bretagne,
Multimedia Networks and Services Department,
2, rue de la Châtaigneraie, 35576 Cesson Sévigné - France
E-mail: joaquin.garcia@enst-bretagne.fr

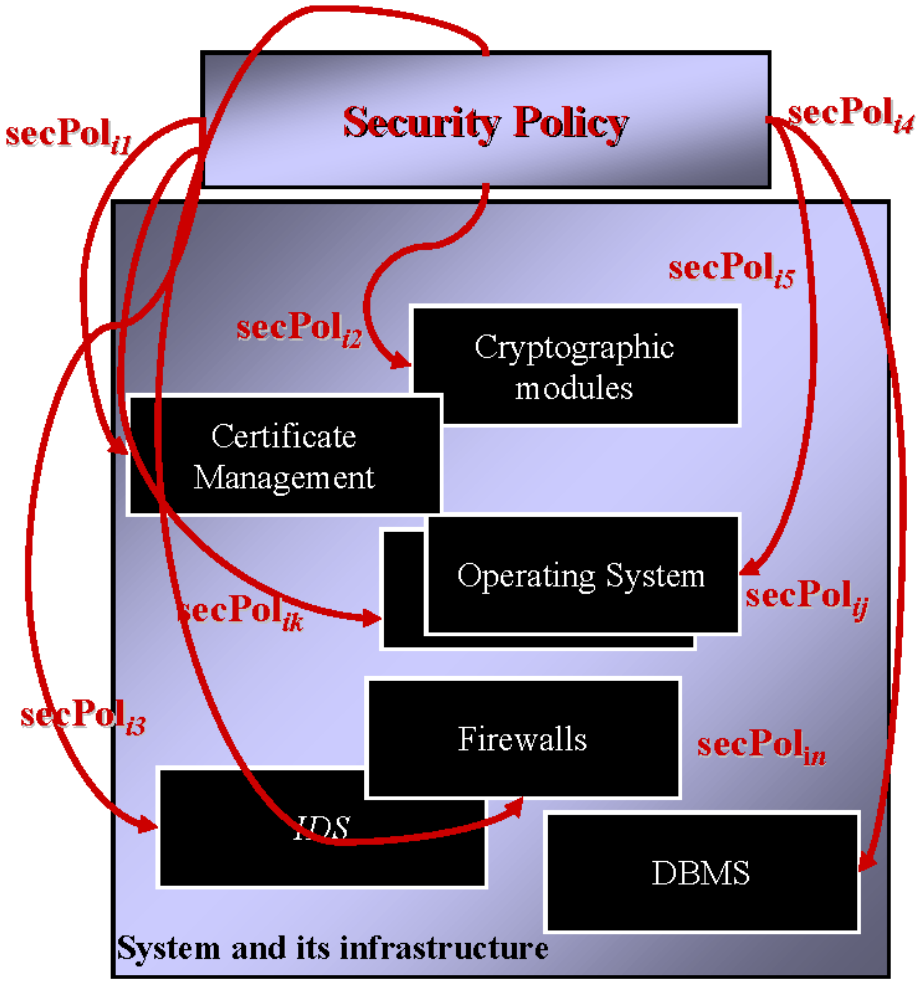
The 2005 IASTED International Conference on Communication, Network and Information Security

# Motivation

- Configuration and managing of network security components
- Manual configuration is a complex task
  - Each component provides its own configuration language
  - The approach used to configure each component is not unique
- We suggest to specify a global policy based on a formal model, and to refine such a policy for each component

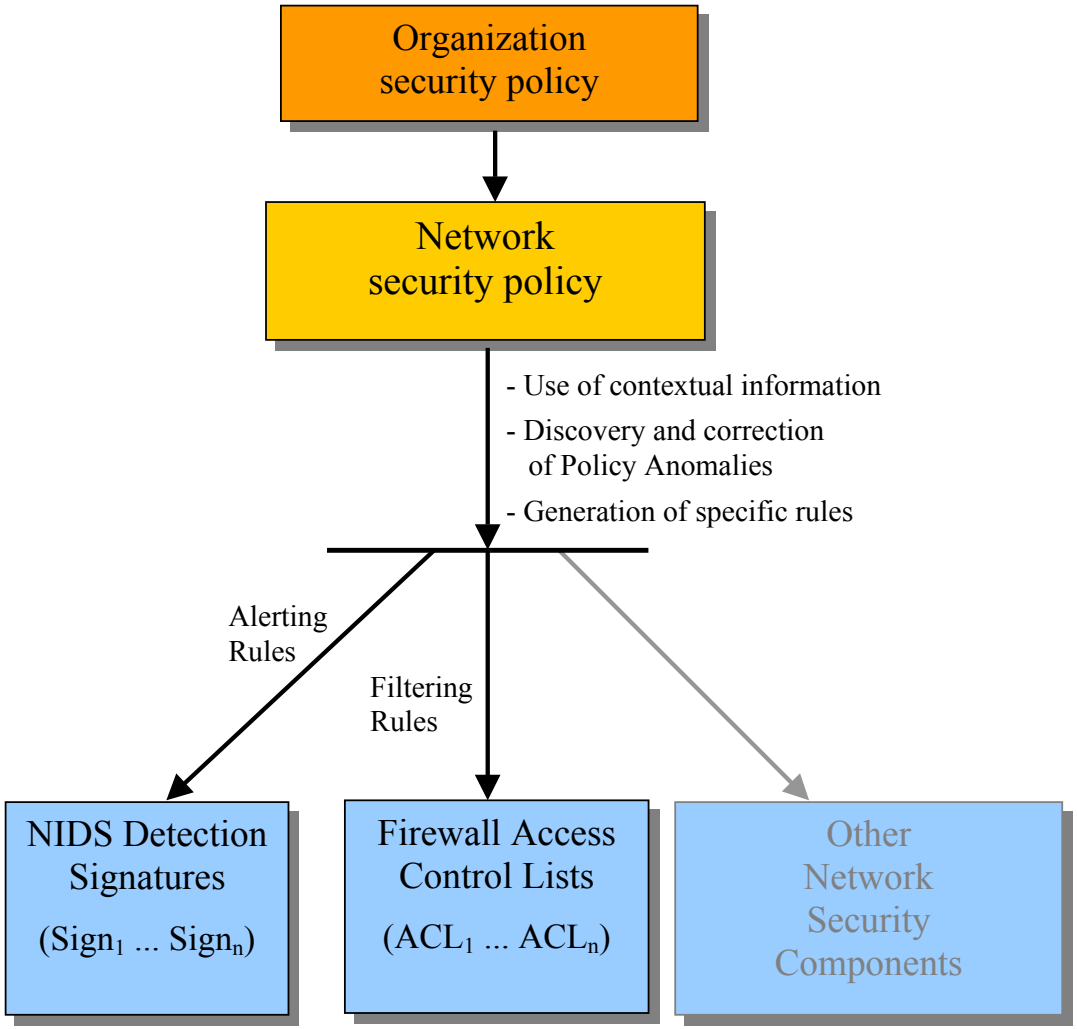
# Introduction

- Definition of a global security policy for the whole system
- Refinement process:
  - → Configuration of specific security policies according to each component within such a global security policy

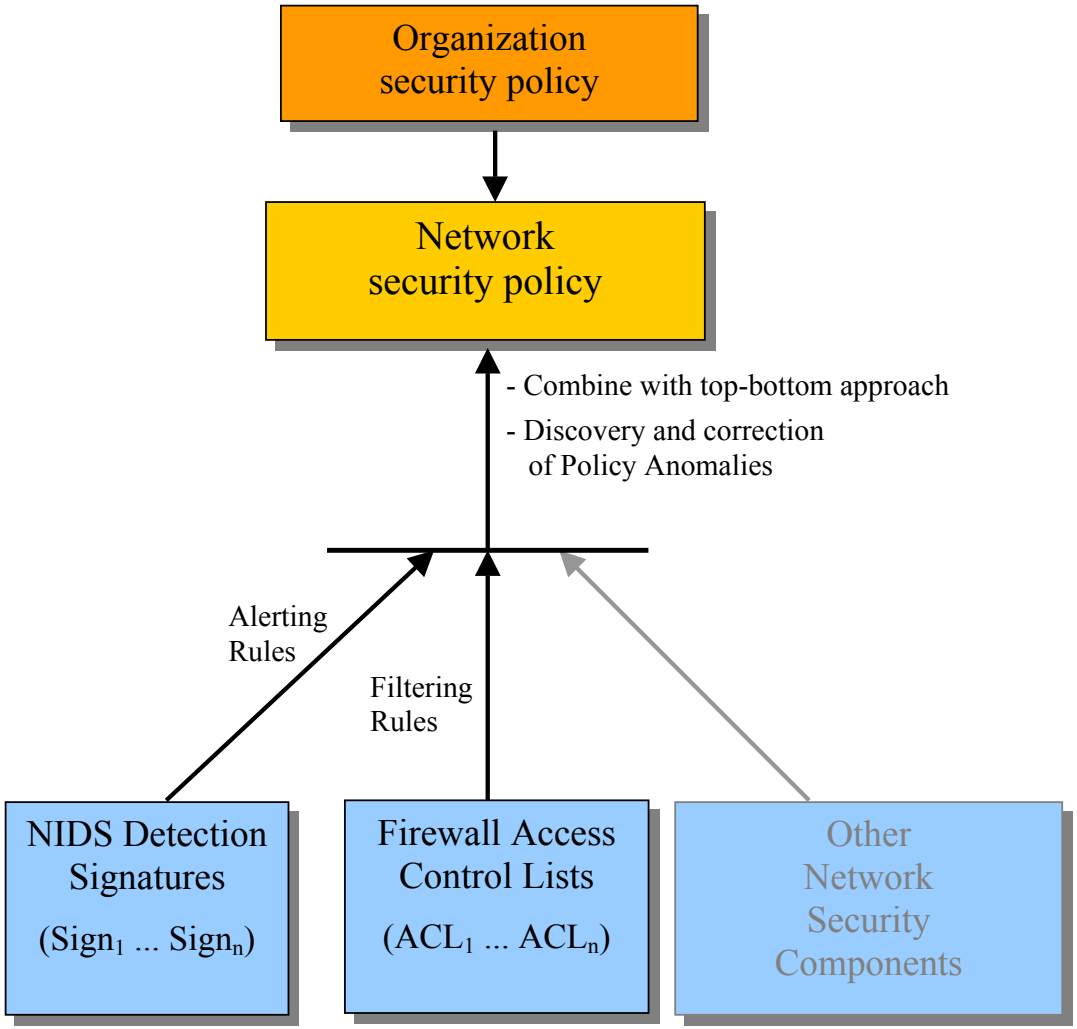


# Introduction

# Introduction



# Problem addressed here



# Bottom-top approach

- We just point out to firewall's filtering rules:

$$Condition \rightarrow accept$$

$$or$$

$$Condition \rightarrow deny$$

  - Condition over a set of attributes

$$@source \wedge @destination \wedge sport \wedge dport \wedge protocol$$

  - Example:

$$s \in 1.0.0.0/24 \wedge d \in any \wedge p = tcp \wedge dport = 80 \rightarrow accept$$

# Policy anomalies

- When processing packages, conflicts due to rule overlaps can occur within the filtering policy
- This conflict can be solved by ordering the rules
  - *First matching strategy*
- It introduces, however, other problems
  - Redundancy
  - Shadowing

# Definitions

- Redundancy
  - Let $R$ be a set of filtering rules, and let $r \in R$
  - Then, rule $r$ is redundant in $R$ iff we can remove $r$ from $R$ and the filtering policy does not change
- Example

$$R1 : s \in 1.0.0.0/24 \wedge d \in 2.0.0.0/16 \wedge p = tcp \wedge dport = 80 \rightarrow accept$$

$$R2 : s \in 1.0.0.0/24 \wedge d \in any \wedge p = tcp \wedge dport = 80 \rightarrow accept$$

# Definitions

- Shadowing
  - Let $R$ be a set of filtering rules, and let $r \in R$
  - Then, rule $r$ is shadowed in $R$ iff such a rule is never applied within filtering policy
- Example

$$R1 : s \in 1.0.0.0/24 \wedge d \in any \wedge p = tcp \wedge dport = 80 \rightarrow accept$$

$$R2 : s \in 1.0.0.0/24 \wedge d \in 2.0.0.0/16 \wedge p = tcp \wedge dport = 80 \rightarrow accept$$

# Related Work

- Some algorithms has been proposed in order to detect such anomalies within a set of filtering rules
  - E. Al-Shaer and H. Hamed,
    Firewall Policy Advisor for Anomaly Detection and Rule Filtering
    *Best paper award at IEEE/IFIP Integrated Management (IM'2003)*
- Proposal:
  - Analyze all the pair of rules
- It does not detect, however, all the possible cases

# Example of anomalies not detected

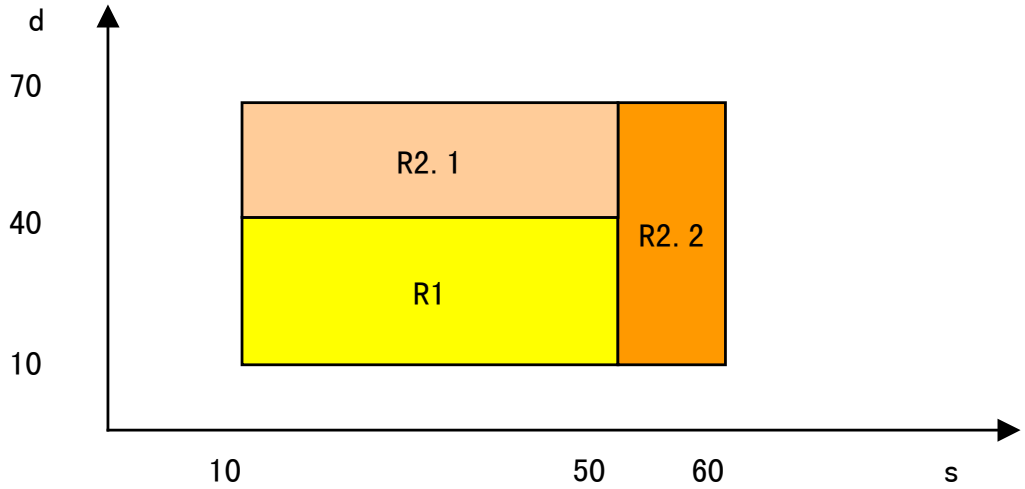
- Shadowing
  - $R1 : s \in 1.0.0.[10, 50] \rightarrow accept$
  - $R2 : s \in 1.0.0.[40, 90] \rightarrow accept$
  - $R3 : s \in 1.0.0.[30, 80] \rightarrow deny$
  - Rule $R3$ is never applied
- Redundancy
  - $R1 : s \in 1.0.0.[10, 50] \rightarrow deny$
  - $R2 : s \in 1.0.0.[40, 70] \rightarrow accept$
  - $R3 : s \in 1.0.0.[50, 80] \rightarrow accept$
  - Rule $R2$ is redundant

# Our proposal

- Complete analysis based on rewriting of rules
  - F. Cuppens, N. Cuppens, and J. Garcï¿½a,
    Detection and Removal of Firewall Misconfiguration
    *The 2005 IASTED International Conference on Communication, Network and Information Security*
- Audit process of firewall setups:
  - $\rightarrow$ Detection: existence of relationships between attributes
  - $\rightarrow$ Removal: transformation from an initial set of rules to an equivalent one which rules free of dependencies

# Removal of dependencies

- Example:
  - $R1 : s \in 1.0.0.[10, 50] \wedge d \in 2.0.0.[10, 40] \rightarrow deny$
  - $R2 : s \in 1.0.0.[10, 60] \wedge d \in 2.0.0.[10, 70] \rightarrow accept$
- Once applied our algorithm:
  - $R1 : s \in 1.0.0.[10, 50] \wedge d \in 2.0.0.[10, 40] \rightarrow deny$
  - $R2.1 : s \in 1.0.0.[51, 60] \wedge d \in 2.0.0.[10, 70] \rightarrow accept$
  - $R2.2 : s \in 1.0.0.[10, 50] \wedge d \in 2.0.0.[41, 70] \rightarrow accept$



# Detection of redundancy and Shadowing

- Two phases
  - $\rightarrow$ Phase 1: rewriting when decision is different
  - $\rightarrow$ Phase 2: rewriting when decision, after test of redundancy, is the same

Example:

$R1 : s \in [10,50] \rightarrow$ deny

$R2 : s \in [40,90] \rightarrow$ accept

$R3 : s \in [60,100] \rightarrow$ accept

$R4 : s \in [30,80] \rightarrow$ deny

$R5 : s \in [1,70] \rightarrow$ accept

# Detection of redundancy and Shadowing

- Two phases
  - → Phase 1: rewriting when decision is different
  - → Phase 2: rewriting when decision, after test of redundancy, is the same

Phase 1 : rewriting R2/R1

R1 : s ∈ [10,50] → deny

R2 : s ∈ [51,90] → accept

R3 : s ∈ [60,100] → accept

R4 : s ∈ [30,80] → deny

R5 : s ∈ [1,70] → accept

# Detection of redundancy and Shadowing

- Two phases
  - → Phase 1: rewriting when decision is different
  - → Phase 2: rewriting when decision, after test of redundancy, is the same

Phase 1 : rewriting R5/R1

R1 : s ∈ [10,50] → deny

R2 : s ∈ [51,90] → accept

R3 : s ∈ [60,100] → accept

R4 : s ∈ [30,80] → deny

R5.1 : s ∈ [1,9] → accept

R5.2 : s ∈ [51,70] → accept

# Detection of redundancy and Shadowing

- Two phases
  - → Phase 1: rewriting when decision is different
  - → Phase 2: rewriting when decision, after test of redundancy, is the same

## Phase 1 : rewriting R4/R2

R1 : s ∈ [10,50] → deny

R2 : s ∈ [51,90] → accept

R3 : s ∈ [60,100] → accept

R4 : s ∈ [30,50] → deny

R5.1 : s ∈ [1,9] → accept

R5.2 : s ∈ [51,70] → accept

# Detection of redundancy and Shadowing

- Two phases
  - → Phase 1: rewriting when decision is different
  - → Phase 2: rewriting when decision, after test of redundancy, is the same

## Phase 2 : rewriting R4/R1

R1 : s ∈ [10,50] → deny

R2 : s ∈ [51,90] → accept

R3 : s ∈ [60,100] → accept

R4 : ∅ → deny  R4 is shadowed

R5.1 : s ∈ [1,9] → accept

R5.2 : s ∈ [51,70] → accept

# Detection of redundancy and Shadowing

- Two phases
  - → Phase 1: rewriting when decision is different
  - → Phase 2: rewriting when decision, after test of redundancy, is the same

## Phase 2 : redundancy test over R2

R1 : s ∈ [10,50] → deny

R2 : ∅ → accept  R2 is redundant

R3 : s ∈ [60,100] → accept

R4 : ∅ → deny  R4 is shadowed

R5.1 : s ∈ [1,9] → accept

R5.2 : s ∈ [51,70] → accept

# Detection of redundancy and Shadowing

- Two phases
  - → Phase 1: rewriting when decision is different
  - → Phase 2: rewriting when decision, after test of redundancy, is the same

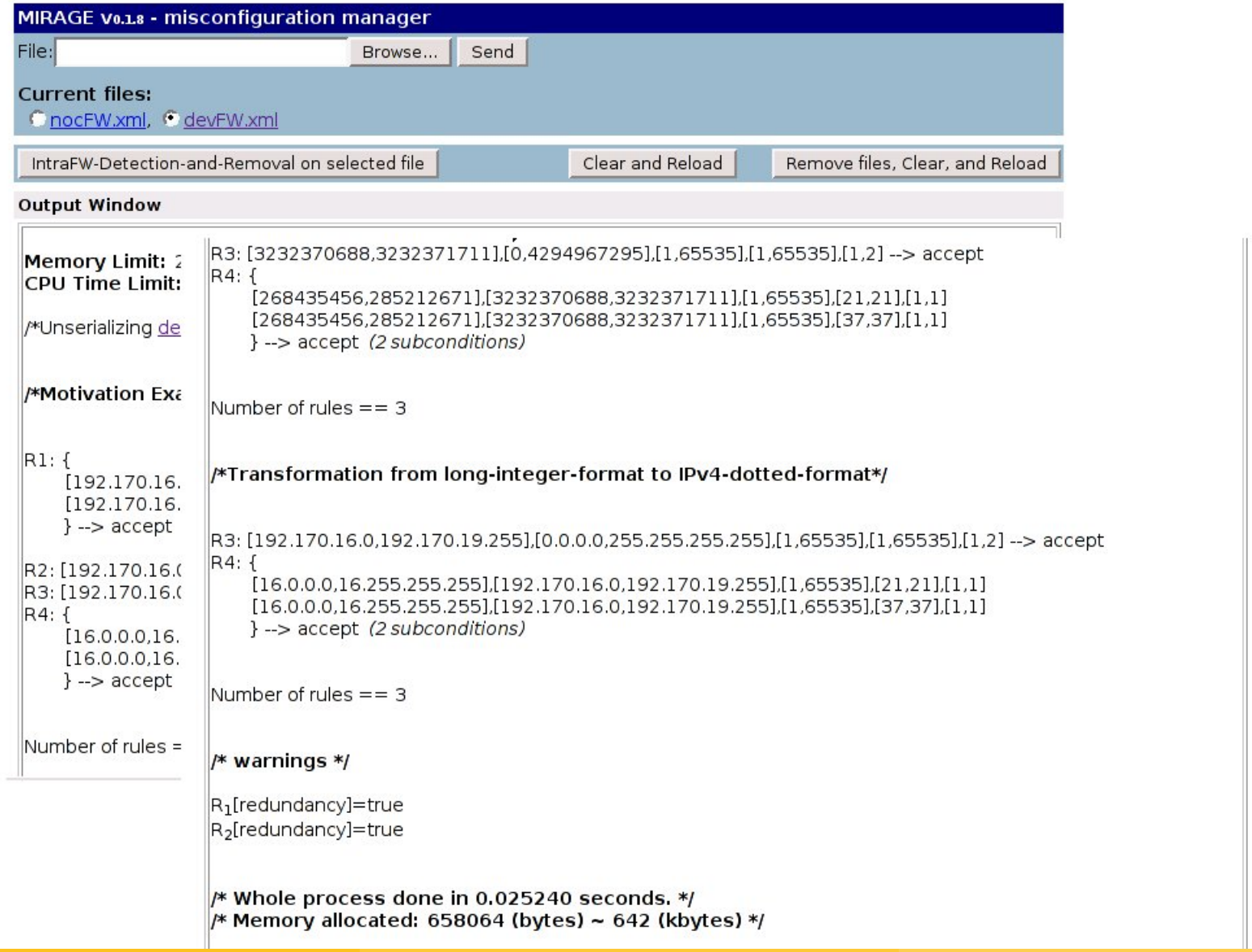
## Phase 2 : rewriting R5/R3

R1 : s ∈ [10,50] → deny
R2 : ∅ → accept
R3 : s ∈ [60,100] → accept
R4 : ∅ → deny
R5.1 : s ∈ [1,9] → accept
R5.2 : s ∈ [51,59] → accept


R2 is redundant


R4 is shadowed

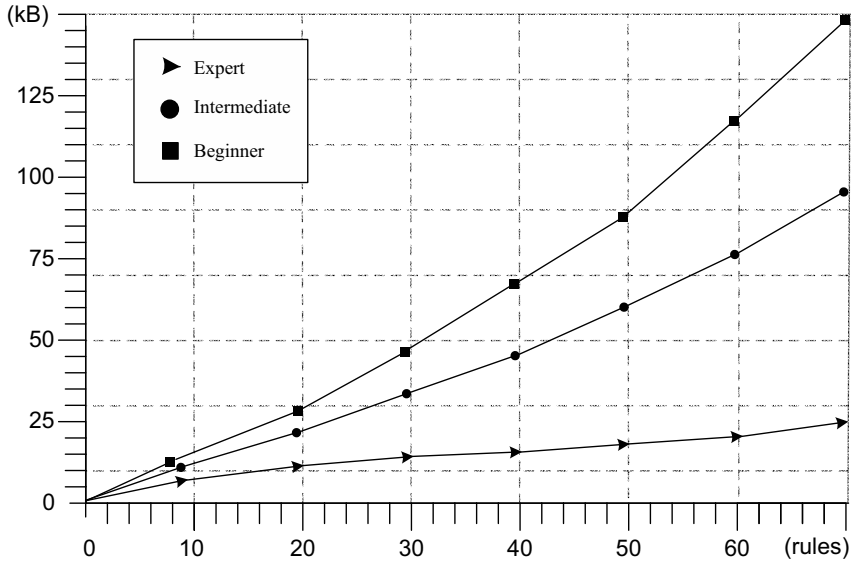
# Implementation of a first prototype



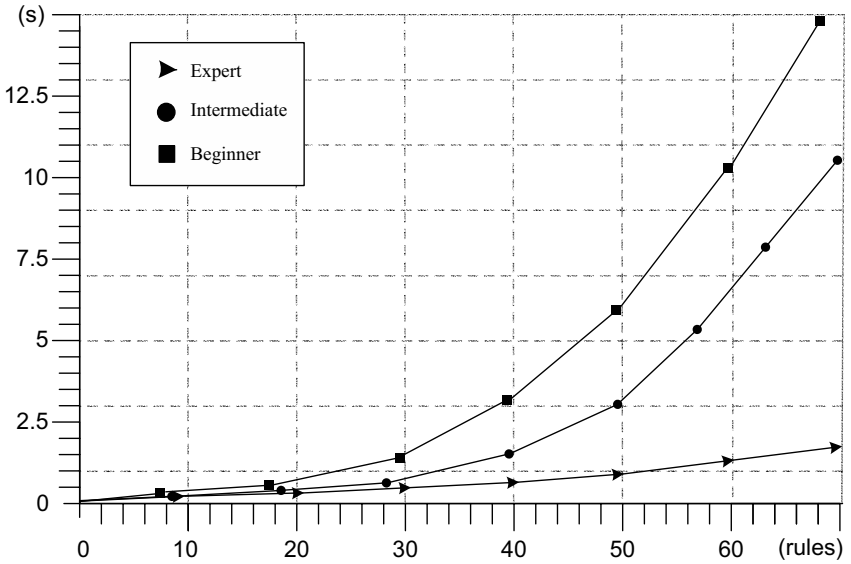
# Deployment and evaluation

- Carried out on an **Intel-Pentium M 1.4 GHz** processor with **512 MB RAM**, running **Debian GNU/Linux 2.6.8**, and using **Apache/1.3** with **PHP/4.3** interpreter configured

*Memory space*




*Processing time*



# Conclusions

- Audit process of firewall setups to both detect and eliminate configuration anomalies
  - → Detection: existence of relationships between attributes
  - → Removal: transformation from an initial set of rules to an equivalent one which rules free of dependencies
- Implementation in a software prototype
  - → It demonstrates the practicability of our work
  - → Although the evaluation points to strong requirements, it is reasonable for off-line analysis

# Work in progress

- Extend our proposal to avoid policy anomalies due to the existence of multi-component setups – **inter-component discovery and removal**