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
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
Problem addressed here

- Combine with top-bottom approach
- Discovery and correction of Policy Anomalies

NIDS Detection
Signatures
(Sign₁ ... Signₙ)

Firewall Access
Control Lists
(ACL₁ ... ACLₙ)

Other Network Security Components
Bottom-top approach

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

\[ \text{Condition} \rightarrow \text{accept} \]

\[ \text{or} \]

\[ \text{Condition} \rightarrow \text{deny} \]

- Condition over a set of attributes

\[ @\text{source} \land @\text{destination} \land \text{sport} \land \text{dport} \land \text{protocol} \]

- Example:

\[ s \in 1.0.0.0/24 \land d \in \text{any} \land p = \text{tcp} \land dport = 80 \rightarrow \text{accept} \]
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**

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

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

  $$R_1 : s \in 1.0.0.0/24 \land d \in \text{any} \land p = tcp \land dport = 80 \rightarrow \text{accept}$$

  $$R_2 : s \in 1.0.0.0/24 \land d \in 2.0.0.0/16 \land p = tcp \land dport = 80 \rightarrow \text{accept}$$
Related Work

Some algorithms have been proposed in order to detect such anomalies within a set of filtering rules


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. García,
    Detection and Removal of Firewall Misconfiguration
    *The 2005 IASTED International Conference on Communication, Network and Information Security*

- Audit process of firewall setups:
  - Detection: existence of relationships between attributes
  - Removal: transformation from an initial set of rules to an equivalent one which rules free of dependencies
Removal of dependencies

**Example:**

- $R_1 : s \in 1.0.0.[10, 50] \land d \in 2.0.0.[10, 40] \rightarrow deny$
- $R_2 : s \in 1.0.0.[10, 60] \land d \in 2.0.0.[10, 70] \rightarrow accept$

**Once applied our algorithm:**

- $R_1 : s \in 1.0.0.[10, 50] \land d \in 2.0.0.[10, 40] \rightarrow deny$
- $R_{2.1} : s \in 1.0.0.[51, 60] \land d \in 2.0.0.[10, 70] \rightarrow accept$
- $R_{2.2} : s \in 1.0.0.[10, 50] \land d \in 2.0.0.[41, 70] \rightarrow accept$
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

Example:

R1 : s ∈ [10,50] → deny
R2 : s ∈ [40,90] → 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 R2/R1**

- **R1**: $s \in [10,50] \rightarrow$ deny
- **R2**: $s \in [51,90] \rightarrow$ 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 R5/R1

- \( R_1 : s \in [10,50] \rightarrow \text{deny} \)
- \( R_2 : s \in [51,90] \rightarrow \text{accept} \)
- \( R_3 : s \in [60,100] \rightarrow \text{accept} \)
- \( R_4 : s \in [30,80] \rightarrow \text{deny} \)
- \( R_{5.1} : s \in [1,9] \rightarrow \text{accept} \)
- \( R_{5.2} : s \in [51,70] \rightarrow \text{accept} \)
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

- $R_1: s \in [10,50] \rightarrow \text{deny}$
- $R_2: s \in [51,90] \rightarrow \text{accept}$
- $R_3: s \in [60,100] \rightarrow \text{accept}$
- $R_4: s \in [30,50] \rightarrow \text{deny}$
- $R_{5.1}: s \in [1,9] \rightarrow \text{accept}$
- $R_{5.2}: s \in [51,70] \rightarrow \text{accept}$
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

- $R_1: s \in [10, 50] \rightarrow \text{deny}$
- $R_2: s \in [51, 90] \rightarrow \text{accept}$
- $R_3: s \in [60, 100] \rightarrow \text{accept}$
- $R_4: \emptyset \rightarrow \text{deny}$  \[\text{R4 is shadowed}\]
- $R_{5.1}: s \in [1, 9] \rightarrow \text{accept}$
- $R_{5.2}: s \in [51, 70] \rightarrow \text{accept}$
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 \( \Rightarrow \) R2 is redundant
R3 : s ∈ [60,100] → accept
R4 : ∅ → deny \( \Rightarrow \) 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

- $R_1 : s \in [10,50] \rightarrow$ deny
- $R_2 : \emptyset \rightarrow$ accept  
  \[\text{R2 is redundant}\]
- $R_3 : s \in [60,100] \rightarrow$ accept
- $R_4 : \emptyset \rightarrow$ deny  
  \[\text{R4 is shadowed}\]
- $R_{5.1} : s \in [1,9] \rightarrow$ accept
- $R_{5.2} : s \in [51,59] \rightarrow$ accept
Implementation of a first prototype

MIRAGE version 0.1.0 - misconfiguration manager

File: Browse... Send

Current files:
- nocFW.xml
- devFW.xml

IntraFW-Detection-and-Removal on selected file
Clear and Reload
Remove files, Clear, and Reload

Output Window

Memory Limit: 700
CPU Time Limit: 100

/*Unserializing de

/*Motivation Exa

R1: {
  [192.170.16.0]
  [192.170.16.1]
} --> accept
R2: [192.170.16.0]
R3: [192.170.16.0, 192.170.19.255, 0.0.0.255.255.255.255, 1.65535, 1.65535, 1.2] --> accept
R4: {
  [268435456.285212671, 3232370688.323237171, 1.65535, 21.21, 1.1]
  [268435456.285212671, 3232370688.323237171, 1.65535, 37.37, 1.1]
} --> accept (2 subconditions)

Number of rules == 3

/*Transformation from long-integer-format to IPv4-dotted-format*/

R3: [192.170.16.0, 192.170.19.255, 0.0.0.255.255.255.255, 1.65535, 1.65535, 1.2] --> accept
R4: {
  [16.0.0.0.16, 16.255.255.255, 192.170.19.255, 1.65535, 21.21, 1.1]
  [16.0.0.0.16, 16.255.255.255, 192.170.19.255, 1.65535, 37.37, 1.1]
} --> accept (2 subconditions)

Number of rules == 3

/* warnings */
R1[redundancy]=true
R2[redundancy]=true

/* Whole process done i 0.025240 seconds. */
/* Memory allocated: 658064 (bytes) ~ 642 (kbytes) */

F. Cuppens, N. Cuppens, and J. Garcia
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
Extend our proposal to avoid policy anomalies due to the existence of multi-component setups – **inter-component discovery and removal**