Misconfiguration Management of Network Security Components

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
Introduction

- Security policy
- Policy Decision Point (PDP)
- Diagnosis and counter-measures
- Topology monitor
- Event collection
- Policy Enforcement Point (PEP)
- Topology Sensors
- Alerts
  - Fusion
  - Correlation
  - Reaction
- Policy Instantiation Engine (PIE)

{secPol_i}
Introduction: Policy Enforcement Point

Policy Enforcement Point (PEP)

Policy Decision Point (PDP)
- Fusion
- Correlation
- Reaction

Topology Sensors

Topology monitor

{secPol_i}

Event collection

OSs
FWs
IDSs
VPNs

Alerts
- Fusion
- Correlation
- Reaction

Policy Instantiation Engine (PIE)

F. Cuppens, N. Cuppens, and J. Garcia
MIRAGE
Introduction: Topology monitor

- Topology Sensors
- Diagnose and counter-measures
- Event collection
- Security policy
- Policy Enforcement Point (PEP)
- Policy Instantiation Engine (PIE)
  - Fusion
  - Correlation
  - Reaction
- Alerts

F. Cuppens, N. Cuppens, and J. Garcia

MIRAGE
Introduction: Policy Instantiation Engine

Policy Instantiation Engine (PIE)

- Fusion
- Correlation
- Reaction

Policy Enforcement Point (PEP)

OSs  FWs  IDSs  VPNs

Event collection

Topology Sensors

Diagnosis and counter-measures

Security policy

Policy Decision Point (PDP)
Introduction: Problem addressed here
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
Top-bottom approach

- Organization security policy
- Network security policy
- Firewall Access Control Lists (ACL₁ ... ACLₙ)
- NIDS Detection Signatures (Sign₁ ... Signₙ)
- Other Network Security Components

Alerting Rules

Filtering Rules

- Use of contextual information
- Discovery and correction of Policy Anomalies
- Generation of specific rules
Bottom-top approach

Organization security policy

Network security policy

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

Alerting Rules

Filtering Rules

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:

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

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:

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

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

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

  $R1: s \in 1.0.0.0/24 \land d \in \text{any} \land p = tcp \land dport = 80 \rightarrow \text{accept}$

  $R2: s \in 1.0.0.0/24 \land d \in 2.0.0.0/16 \land p = tcp \land dport = 80 \rightarrow \text{accept}$
Some algorithms have been proposed in order to detect such anomalies within a set of filtering rules:

  *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. García
    Misconfiguration Management of Network Security Components
    7th Int. Symposium on System and Information Security (SSI05)

- 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:

- $R_1 : s \in [10,50] \rightarrow \text{deny}$
- $R_2 : s \in [40,90] \rightarrow \text{accept}$
- $R_3 : s \in [60,100] \rightarrow \text{accept}$
- $R_4 : s \in [30,80] \rightarrow \text{deny}$
- $R_5 : s \in [1,70] \rightarrow \text{accept}$
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 \( R_2/R_1 \)

- \( R_1: s \in [10,50] \rightarrow \text{deny} \)
- \( R_2: s \in [51,90] \rightarrow \text{accept} \)
- \( R_3: s \in [60,100] \rightarrow \text{accept} \)
- \( R_4: s \in [30,80] \rightarrow \text{deny} \)
- \( R_5: s \in [1,70] \rightarrow \text{accept} \)
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 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

\[
\begin{align*}
R_1 & : s \in [10,50] \rightarrow \text{deny} \\
R_2 & : s \in [51,90] \rightarrow \text{accept} \\
R_3 & : s \in [60,100] \rightarrow \text{accept} \\
R_4 & : \emptyset \rightarrow \text{deny} \quad \text{R4 is shadowed} \\
R_{5.1} & : s \in [1,9] \rightarrow \text{accept} \\
R_{5.2} & : s \in [51,70] \rightarrow \text{accept}
\end{align*}
\]
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

\[ R_1 : s \in [10,50] \rightarrow \text{deny} \]
\[ R_2 : \emptyset \rightarrow \text{accept} \]  \(\text{R2 is redundant}\)
\[ R_3 : s \in [60,100] \rightarrow \text{accept} \]
\[ R_4 : \emptyset \rightarrow \text{deny} \]  \(\text{R4 is shadowed}\)
\[ R_{5.1} : s \in [1,9] \rightarrow \text{accept} \]
\[ R_{5.2} : s \in [51,70] \rightarrow \text{accept} \]
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: rewriting R5/R3

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

MIRAGE v0.1.0 - misconfiguration manager

Current files:
- nocFW.xml
- devFW.xml

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

Output Window

Memory Limit: 7
CPU Time Limit: /*Unserializing de */

/* Motivation Example

R1: {
[192.170.16.0, 192.170.16.0] -> accept
}

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

R3: [192.170.16.0, 192.170.19.255, 0.0.0.0.255.255.255.255, [1,65535],[1,65535],[1,2] -- > accept
R4: {
[268435456,285212671],[3232370686,3232371711],[1,65535],[21,21],[1,1]
[268435456,285212671],[3232370686,3232371711],[1,65535],[37,37],[1,1]
} -- > accept (2 subconditions)

Number of rules == 3

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

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

<table>
<thead>
<tr>
<th>Intermediate</th>
<th>Beginner</th>
<th>Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>50</td>
<td>75</td>
</tr>
<tr>
<td>100</td>
<td>125</td>
<td>150</td>
</tr>
</tbody>
</table>

### Processing time

<table>
<thead>
<tr>
<th>Intermediate</th>
<th>Beginner</th>
<th>Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>5</td>
<td>7.5</td>
</tr>
<tr>
<td>10</td>
<td>12.5</td>
<td>15</td>
</tr>
</tbody>
</table>

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