MIRAGE: A Management Tool for the Analysis and Deployment of Network Security Policies

Joaquin Garcia-Alfaro

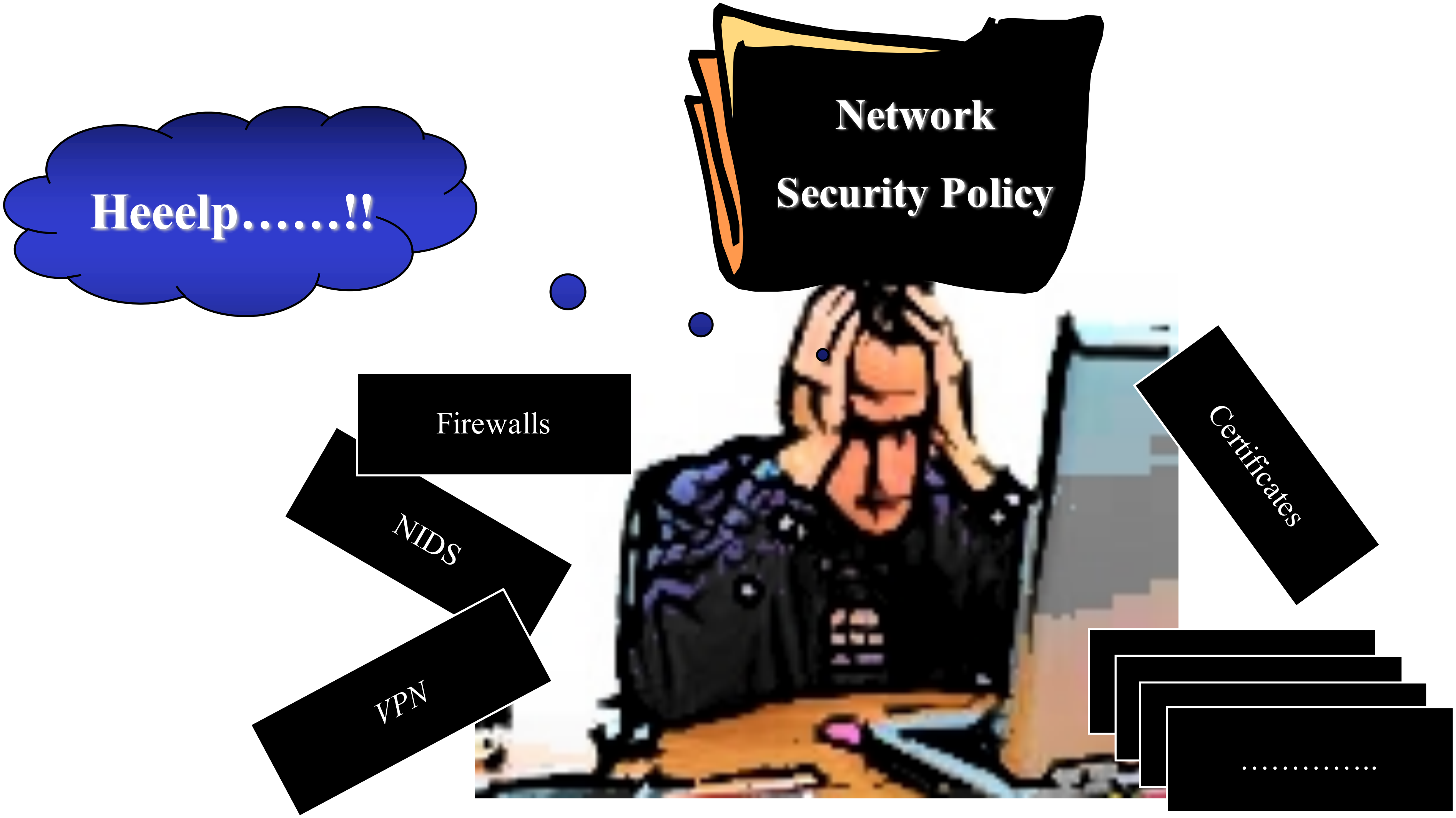
Télécom Bretagne

Joint work with

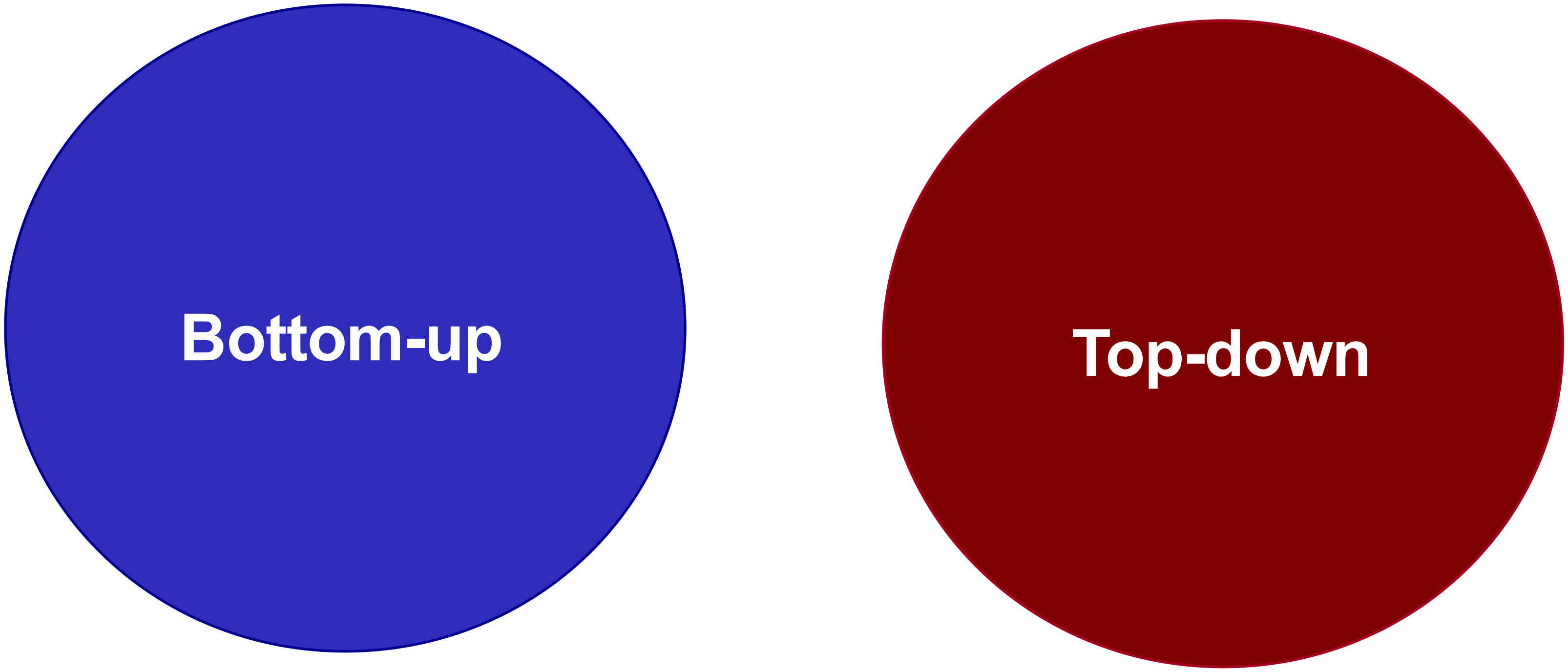
Frédéric Cuppens, Nora Cuppens-Boulahia, Stere Preda, and Thierry Sans

Brief introduction

Management of configuration conflicts (or configuration anomalies in general) is a (very) complex task



Two main strategies



Bottom-up

Top-down

Outline

- **Brief introduction**

- **Bottom-up analysis of filtering configurations**
  - Intra-component analysis
  - Inter-component analysis

- **Top-down refinement of general security policies**

- **Conclusion / Perspectives**

Bottom-up analysis

- **Configuration of Firewalls**
  - When processing packages, conflicts due to rule overlaps can occur within the same policy

  - We can solve this problem by ordering the rules
    - First/Last matching strategy

  ⇒It introduces, however, some other problems
    - Shadowing (i.e., rules that are never applied)
    - Redundancy (i.e., if removed, policy does not change)

Definitions

- **Format of rules**

Condition → accept

or

Condition → deny

Where *condition* is a conjunctive set of attributes in the form:

@source ∧ @destination ∧ port-source ∧ port-destination ∧ protocol

- **Example of Shadowing**

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

$R_2 : s \in 1.0.0.0/24 \wedge d \in 2.0.0.0/16 \wedge \text{sport} \in \text{any} \wedge \text{dport} = 80 \wedge p = \text{tcp} \rightarrow \text{deny}$

- **Example of Redundancy**

$R_1 : s \in 1.0.0.0/24 \wedge d \in 2.0.0.0/16 \wedge \text{sport} \in \text{any} \wedge \text{dport} = 80 \wedge p = \text{tcp} \rightarrow \text{accept}$

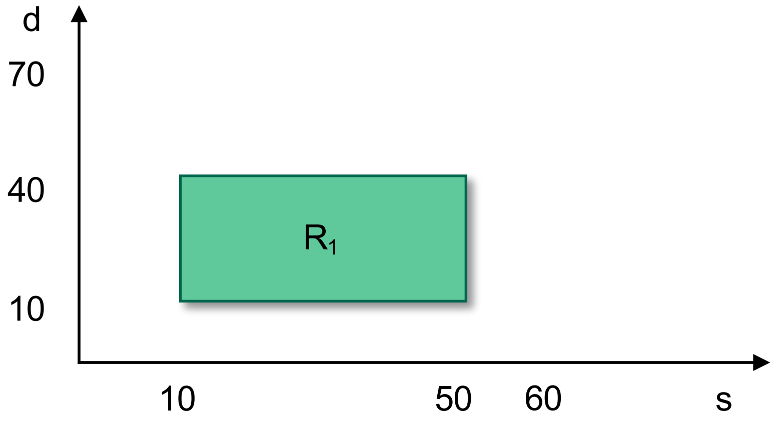
$R_2 : s \in 1.0.0.0/24 \wedge d \in \text{any} \wedge \text{sport} \in \text{any} \wedge \text{dport} = 80 \wedge p = \text{tcp} \rightarrow \text{accept}$

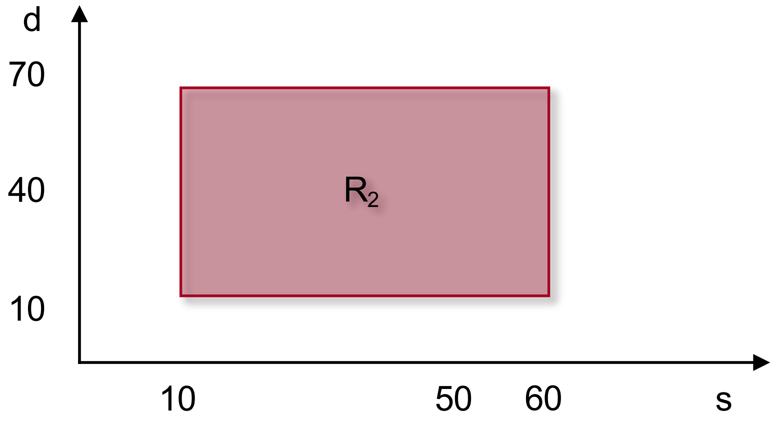
Bottom-up analysis of MIRAGE

- Detection & removal of configuration anomalies
- Based on *rewritting* of rules:
  - Detection: existence of relationships between attributes
  - Removal: transformation from an initial set of rules to an equivalent one which rules free of dependencies
- Example:

R_1 : s ∈ 1.0.0.[10,50] ∧ d ∈ 2.0.0.[10,40] → accept

R_2 : s ∈ 1.0.0.[10,60] ∧ d ∈ 2.0.0.[10,70] → deny





Bottom-up analysis of MIRAGE

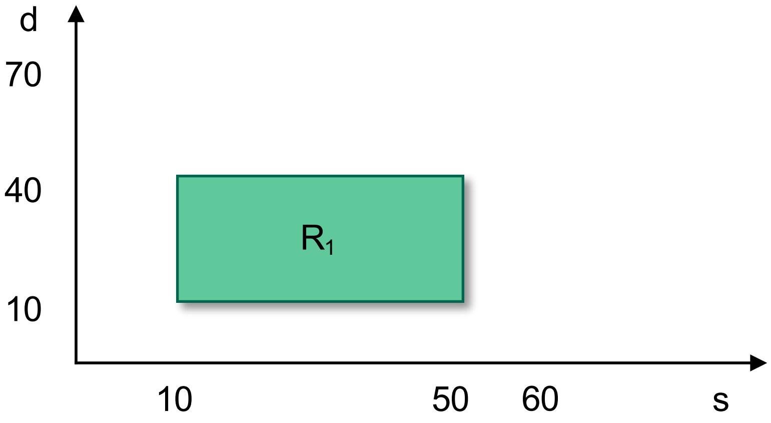
- Detection & removal of configuration anomalies
- Based on *rewritting* of rules:
  - Detection: existence of relationships between attributes
  - Removal: transformation from an initial set of rules to an equivalent one which rules free of dependencies

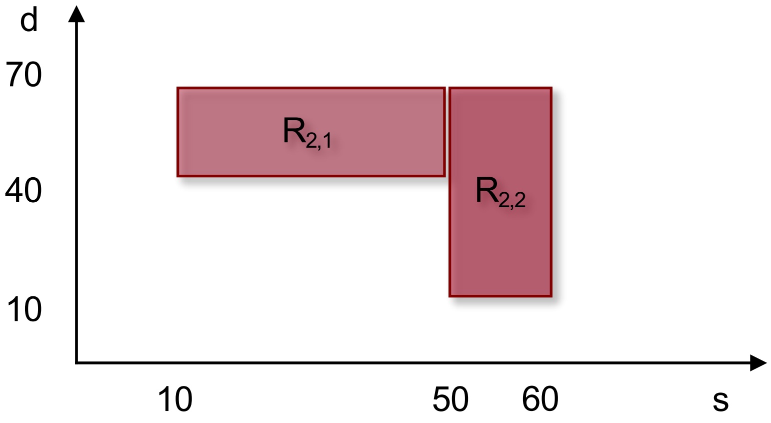
- Example:

R_1 : $s \in 1.0.0.[10,50] \wedge d \in 2.0.0.[10,40]$ → accept

$R_{2,1}$: $s \in 1.0.0.[51,60] \wedge d \in 2.0.0.[10,70]$ → deny

$R_{2,2}$: $s \in 1.0.0.[10,50] \wedge d \in 2.0.0.[41,70]$ → deny





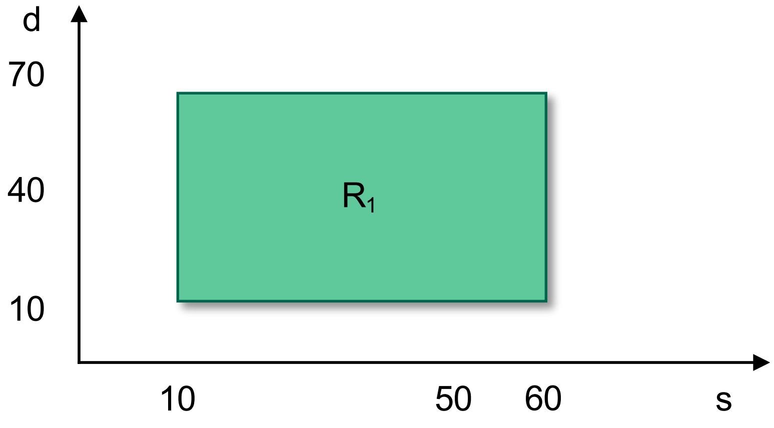
Bottom-up analysis of MIRAGE

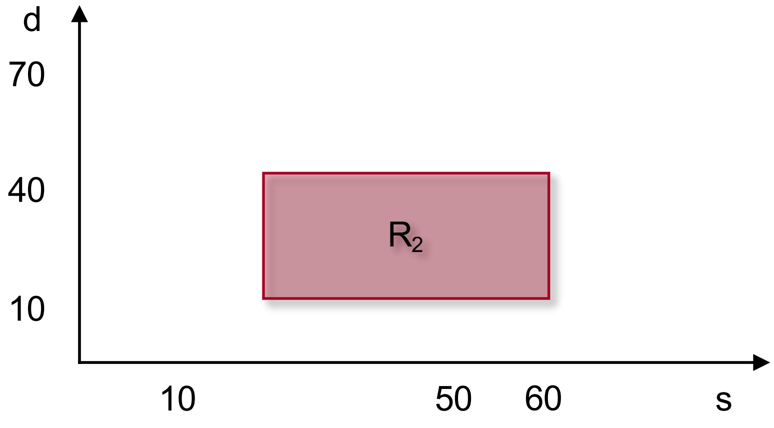
- Detection & removal of configuration anomalies
- Based on *rewritting* of rules:
  - Detection: existence of relationships between attributes
  - Removal: transformation from an initial set of rules to an equivalent one which rules free of dependencies

- Example:

$R_1 : s \in 1.0.0.[10,60] \wedge d \in 2.0.0.[10,70] \rightarrow \text{accept}$

$R_2 : s \in 1.0.0.[10,50] \wedge d \in 2.0.0.[10,40] \rightarrow \text{deny}$



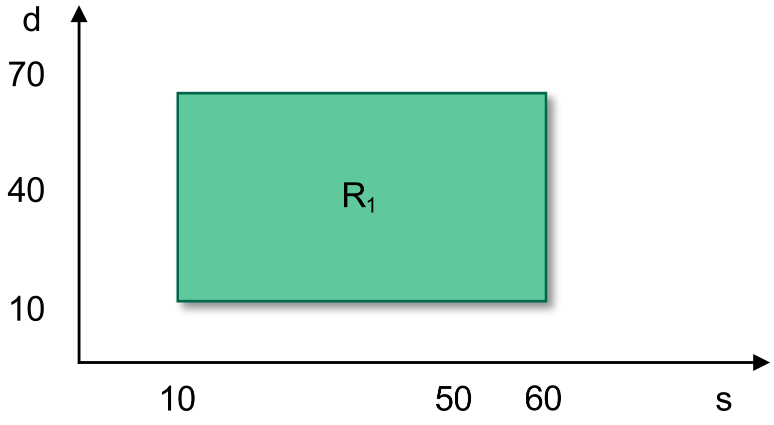


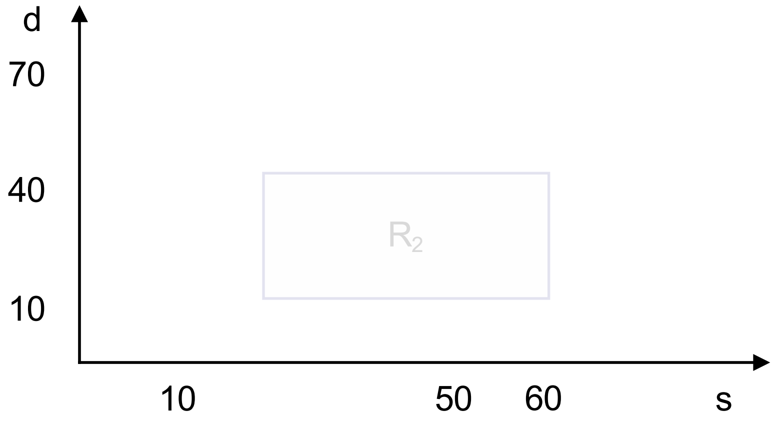
Bottom-up analysis of MIRAGE

- Detection & removal of configuration anomalies
- Based on *rewritting* of rules:
  - Detection: existence of relationships between attributes
  - Removal: transformation from an initial set of rules to an equivalent one which rules free of dependencies
- Example:

R_1 : s ∈ 1.0.0.[10,60] ∧ d ∈ 2.0.0.[10,70] → accept

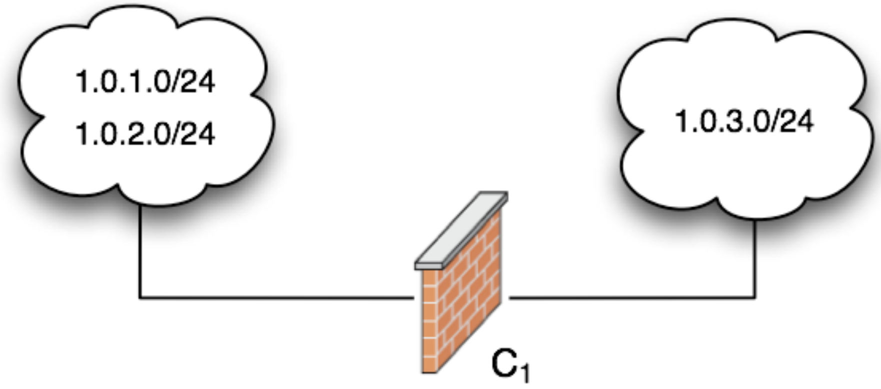
R_2 : ∅ → deny





Intra-component Analysis (1/2)

- Deterministic analysis of standalone configurations
- Taxonomy on anomalies:
  - Intra-component Shadowing
  - Intra-component Redundancy
- Example:



$C_1\{R_1\}$: {tcp,1.0.2.[1,30]:any, 1.0.3.[20,45]:any} → true

$C_1\{R_2\}$: {tcp,1.0.2.[20,60]:any, 1.0.3.[25,35]:any} → false

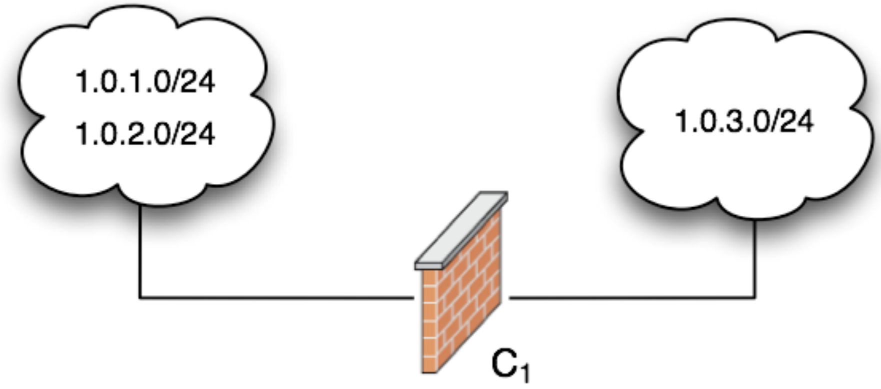
$C_1\{R_3\}$: {tcp,1.0.2.[30,70]:any, 1.0.3.[20,45]:any} → false

$C_1\{R_4\}$: {tcp,1.0.2.[15,45]:any, 1.0.3.[25,30]:any} → true

... ...

Intra-component Analysis (2/2)

- Deterministic analysis of standalone configurations
- Taxonomy on anomalies:
  - Intra-component Shadowing
  - Intra-component Redundancy
- Example:



R_2 is redundant to R_1, R_3
R_4 is shadowed by R_2, R_1

$C_1\{R_1\}$: {tcp,1.0.2.[1,30]:any, 1.0.3.[20,45]:any} → true
$C_1\{R_2\}$: {tcp,1.0.2.[20,60]:any, 1.0.3.[25,35]:any} → false
$C_1\{R_3\}$: {tcp,1.0.2.[30,70]:any, 1.0.3.[20,45]:any} → false
$C_1\{R_4\}$: {tcp,1.0.2.[15,45]:any, 1.0.3.[25,30]:any} → true
... ...

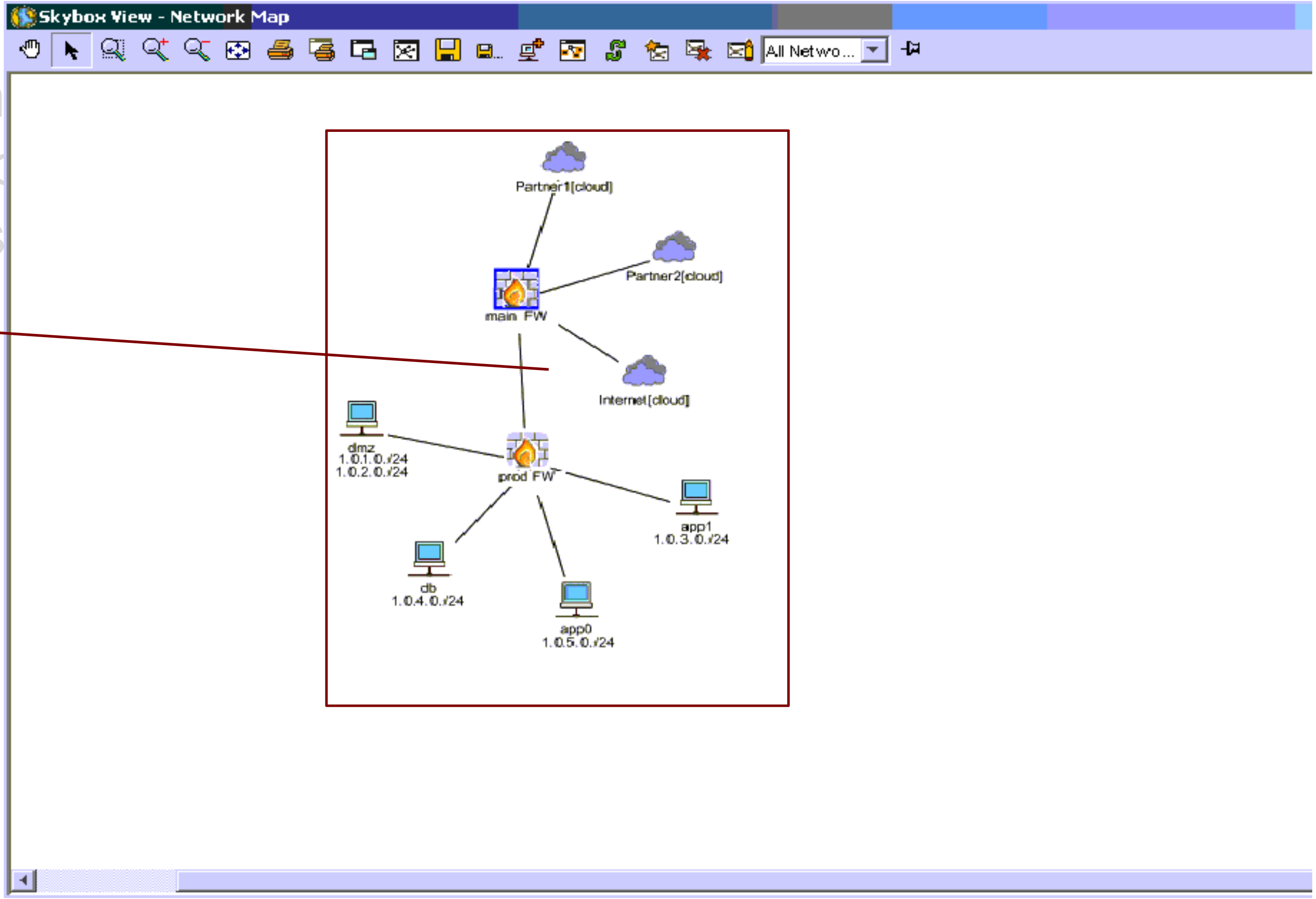
Topology of the System

- MIRAGE also manages the description of the security architecture topology, to guarantee the proper execution of the audit processes

Topology of the System

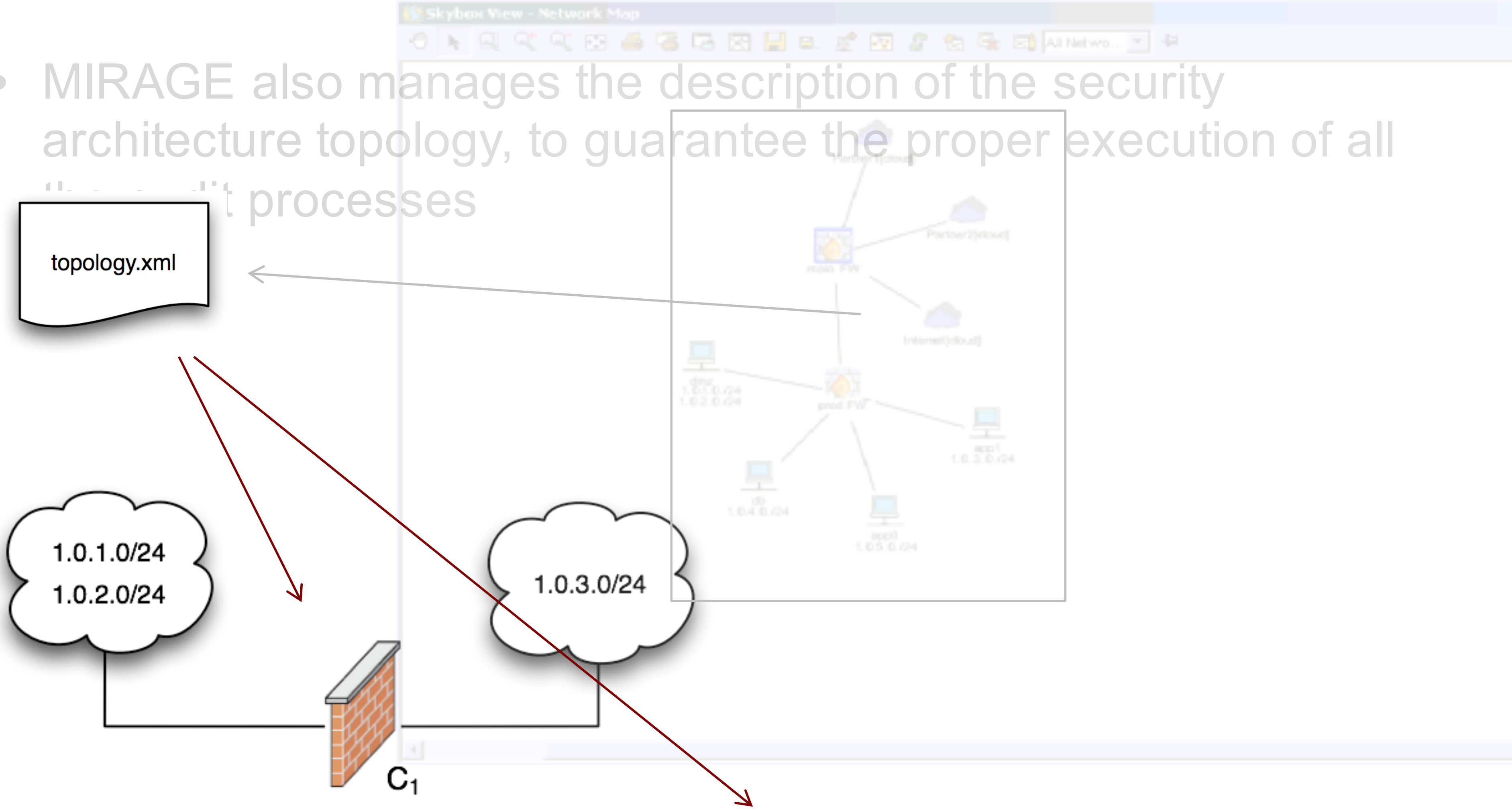
- MIRAGE also m architecture topo th audit proces

topology.xml



Topology of the System

- MIRAGE also manages the description of the security architecture topology, to guarantee the proper execution of all the audit processes



$C_1\{R_1\}$: {tcp,1.0.2.[1,30]:any, 1.0.3.[20,45]:any} → true

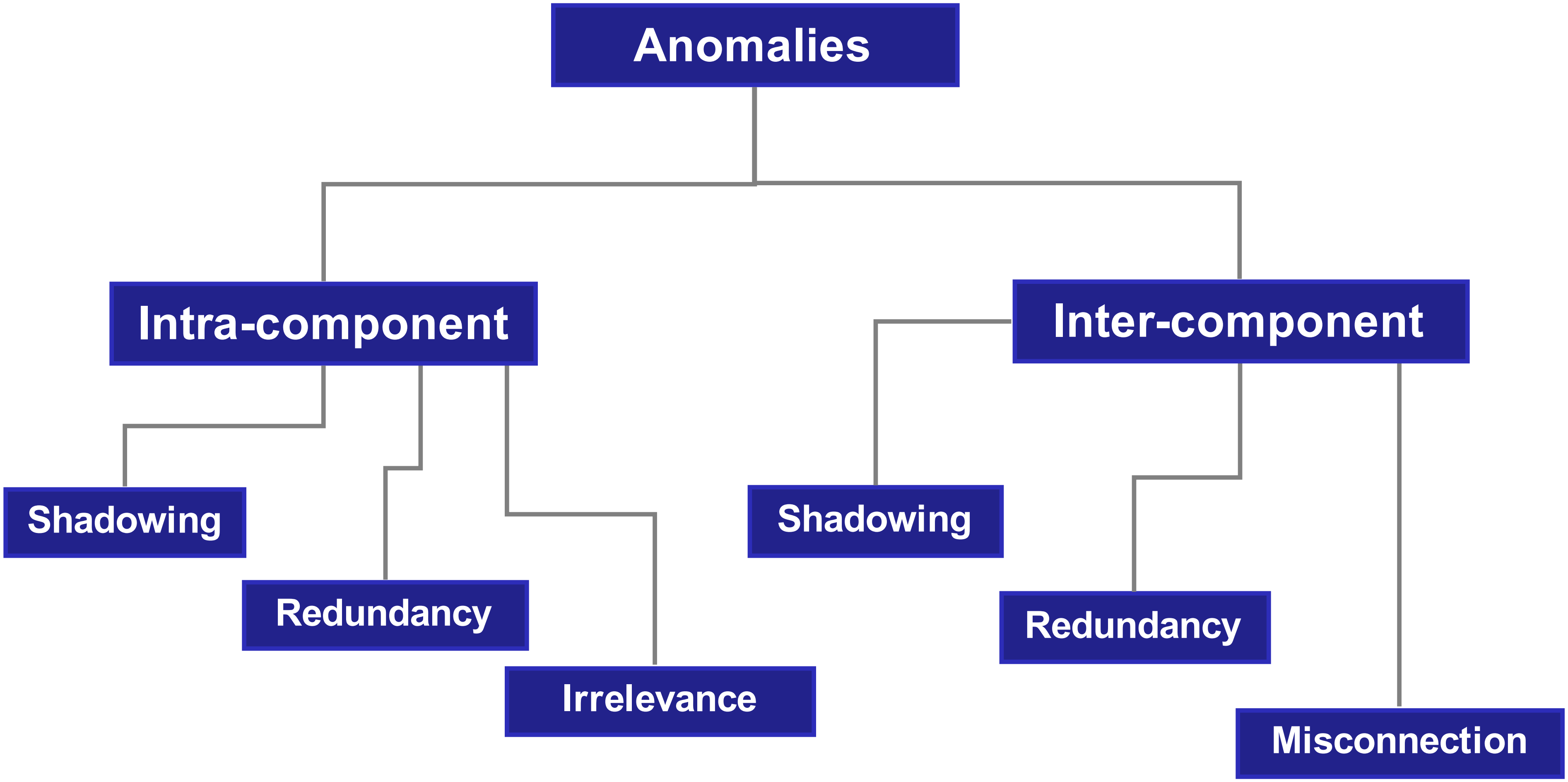
$C_1\{R_2\}$: {tcp,1.0.2.[20,60]:any, 1.0.3.[25,35]:any} → false

$C_1\{R_3\}$: {tcp,1.0.2.[30,70]:any, 1.0.3.[20,45]:any} → false

$C_1\{R_4\}$: {tcp,1.0.2.[15,45]:any, 1.0.3.[25,30]:any} → true

... ...

Complete taxonomy of Anomalies


Anomalies

Intra-component

Inter-component

Shadowing

Redundancy

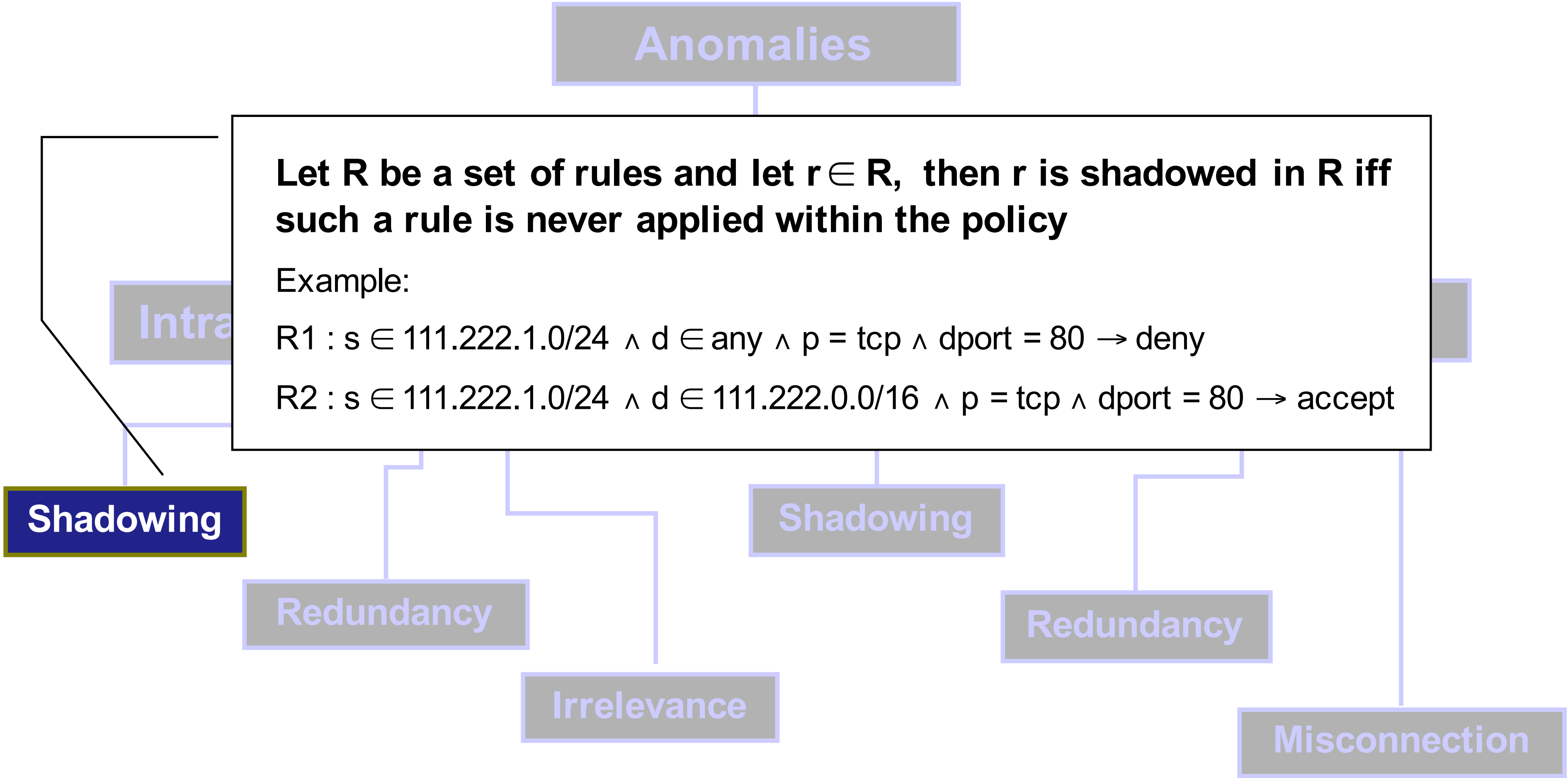
Irrelevance

Shadowing

Redundancy

Misconnection

Complete taxonomy of Anomalies



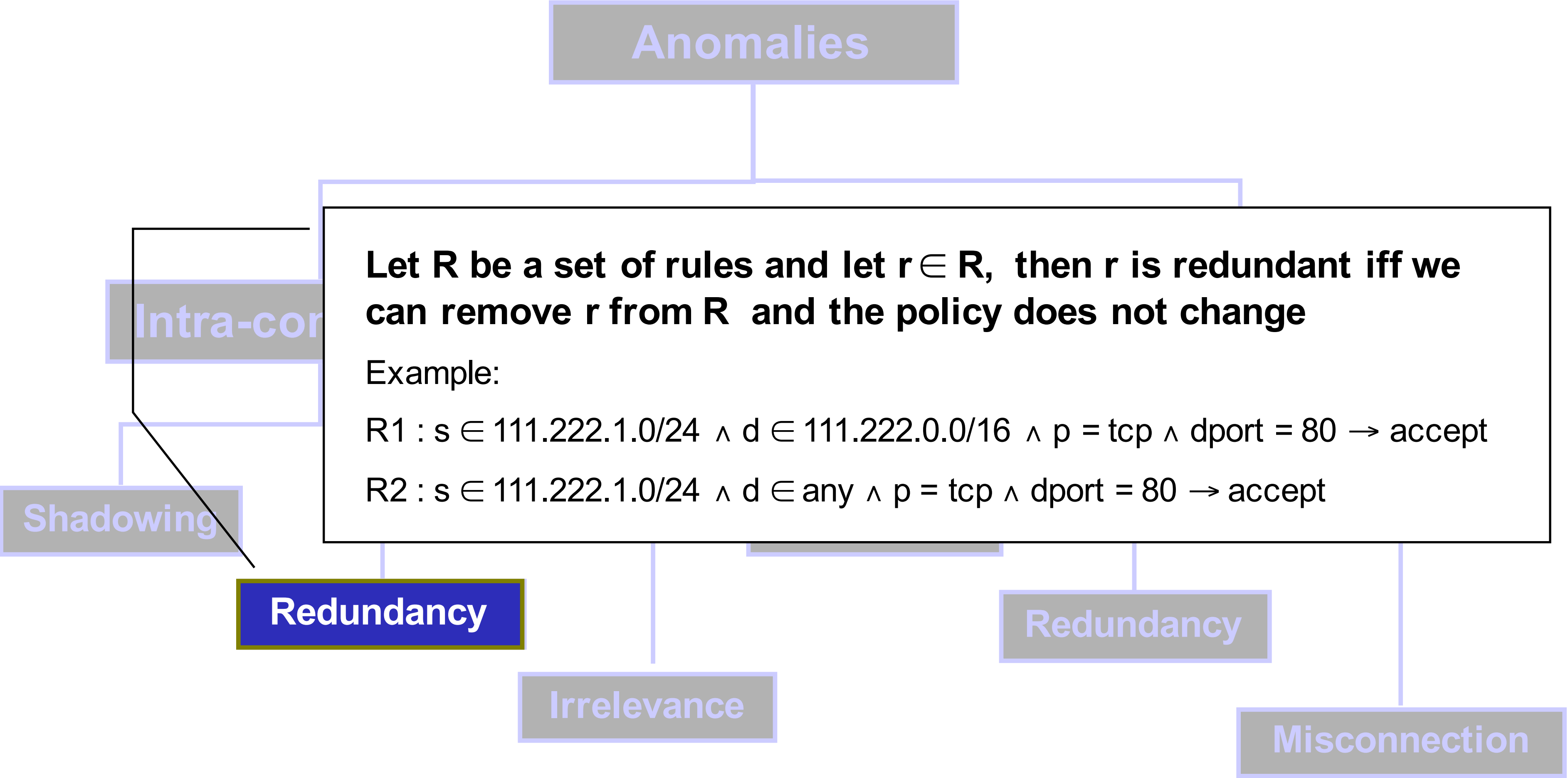
Let R be a set of rules and let r ∈ R, then r is shadowed in R iff such a rule is never applied within the policy

Example:

R1 : $s \in 111.222.1.0/24 \wedge d \in any \wedge p = tcp \wedge dport = 80 \rightarrow deny$

R2 : $s \in 111.222.1.0/24 \wedge d \in 111.222.0.0/16 \wedge p = tcp \wedge dport = 80 \rightarrow accept$

Complete taxonomy of Anomalies



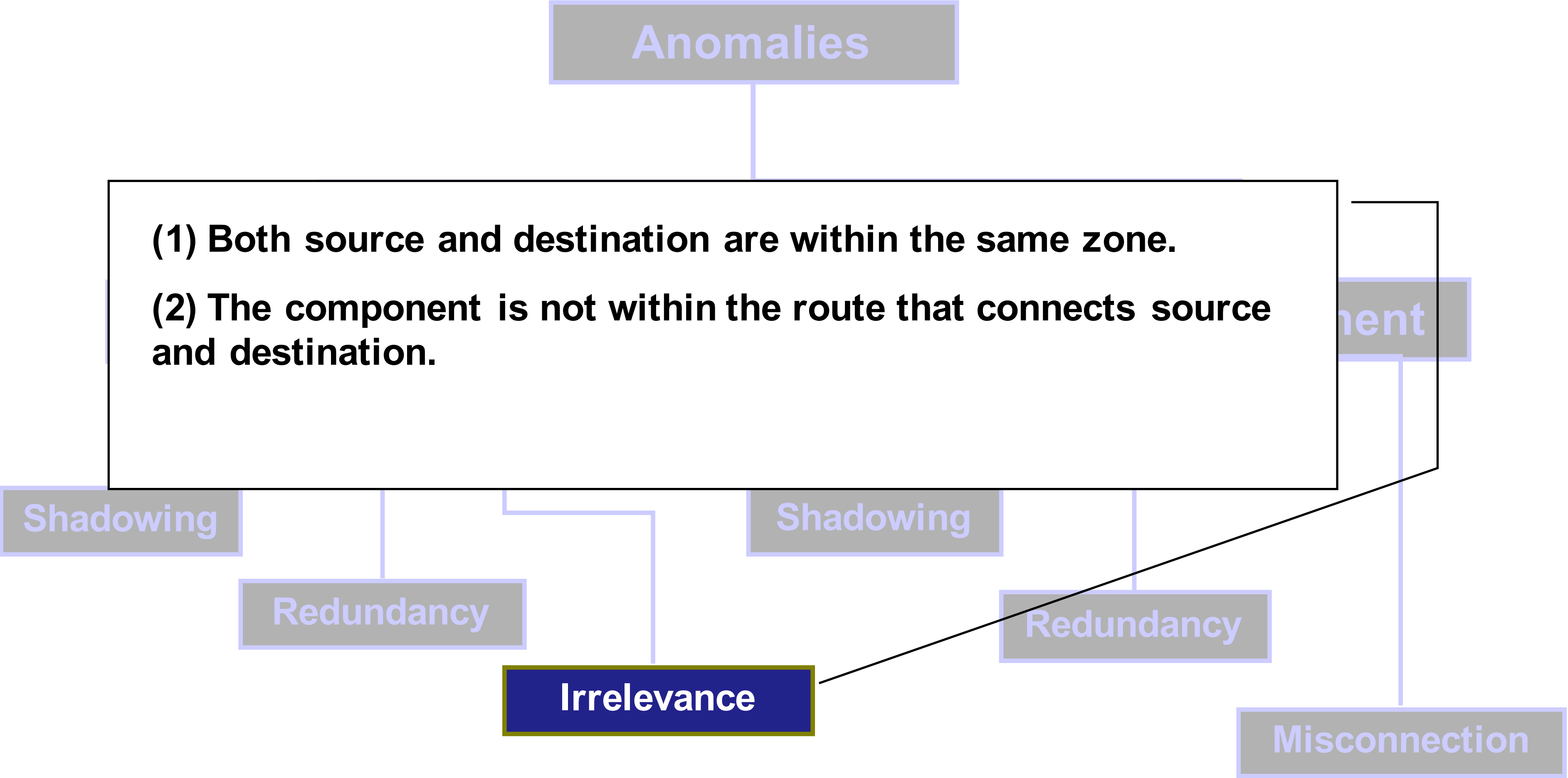
Let R be a set of rules and let $r \in R$, then r is redundant iff we can remove r from R and the policy does not change

Example:

R1 : $s \in 111.222.1.0/24 \wedge d \in 111.222.0.0/16 \wedge p = tcp \wedge dport = 80 \rightarrow accept$

R2 : $s \in 111.222.1.0/24 \wedge d \in any \wedge p = tcp \wedge dport = 80 \rightarrow accept$

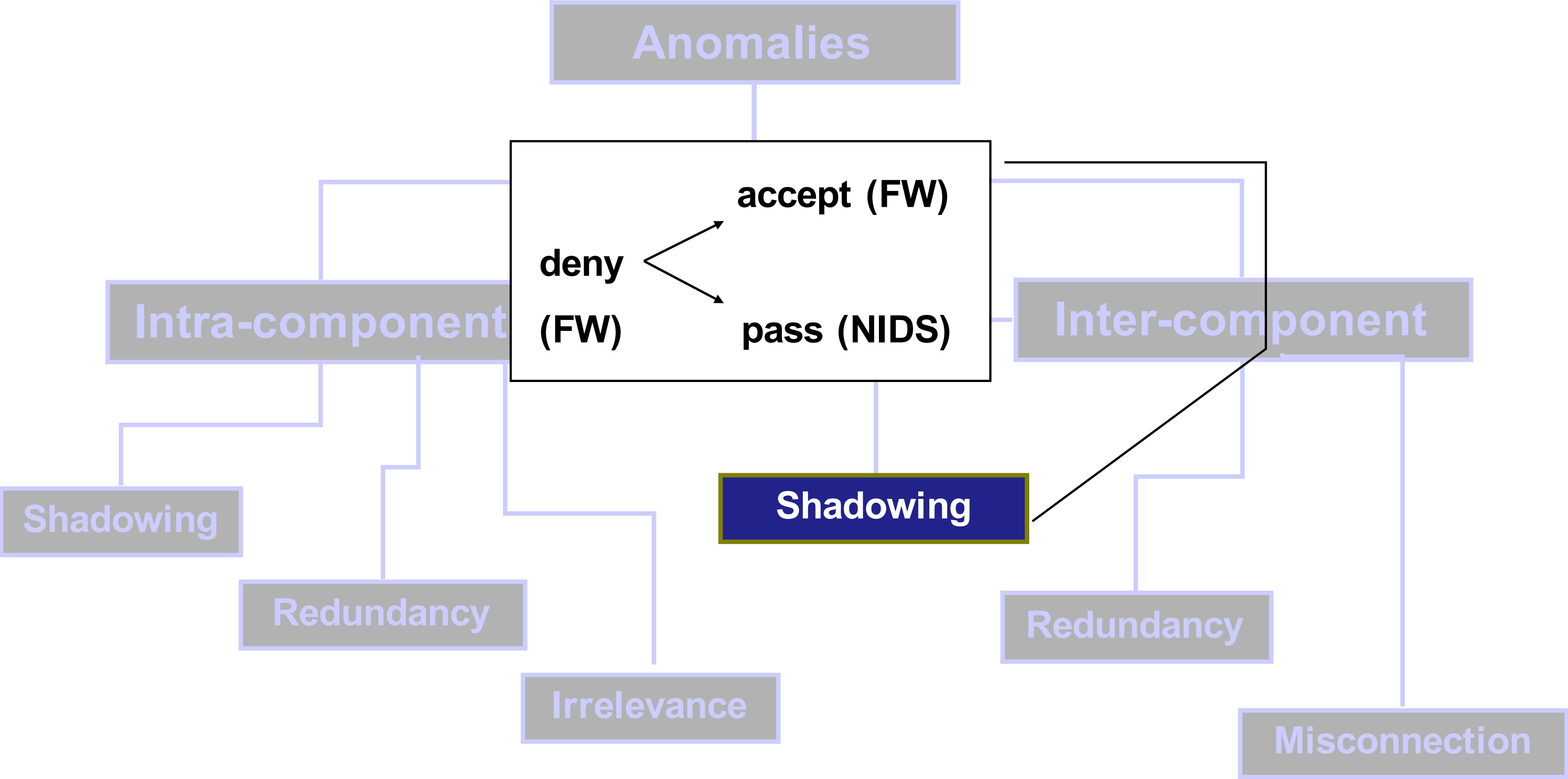
Complete taxonomy of Anomalies



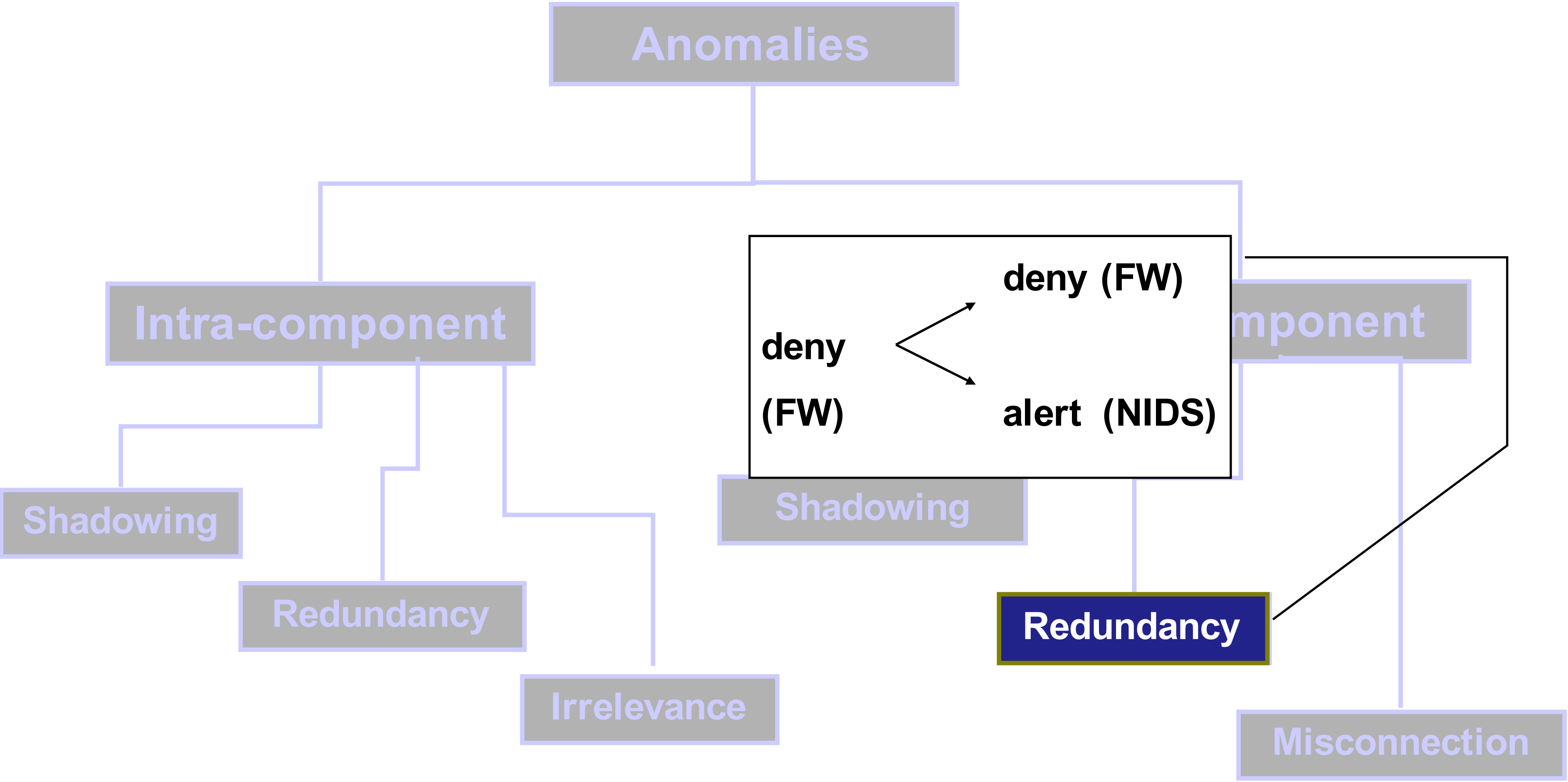
(1) Both source and destination are within the same zone.

(2) The component is not within the route that connects source and destination.

Complete taxonomy of Anomalies



Complete taxonomy of Anomalies



Anomalies

Intra-component

Shadowing

Redundancy

Irrelevance

mponent

Shadowing

Redundancy

Misconnection

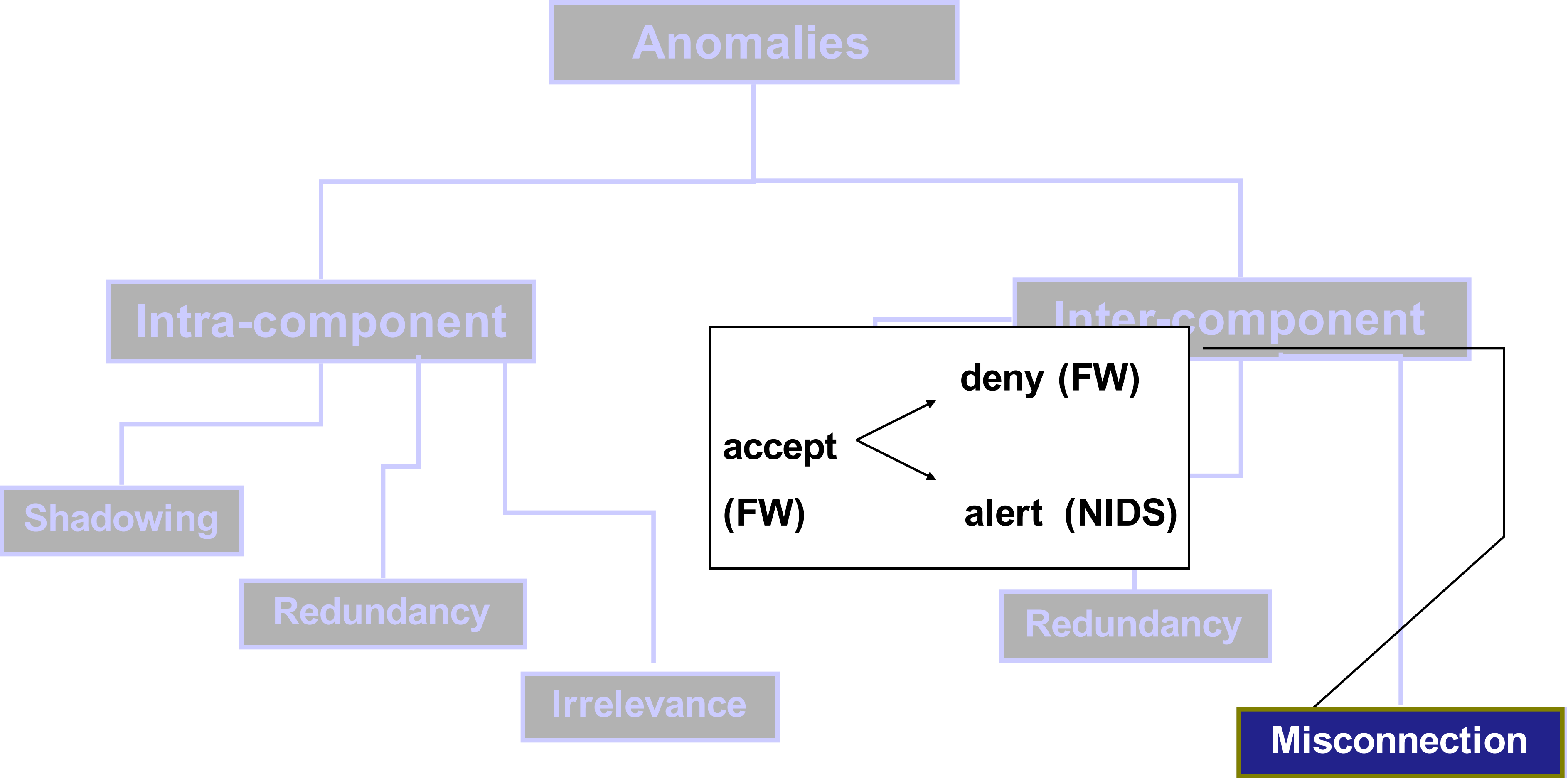
deny

(FW)

deny (FW)

alert (NIDS)

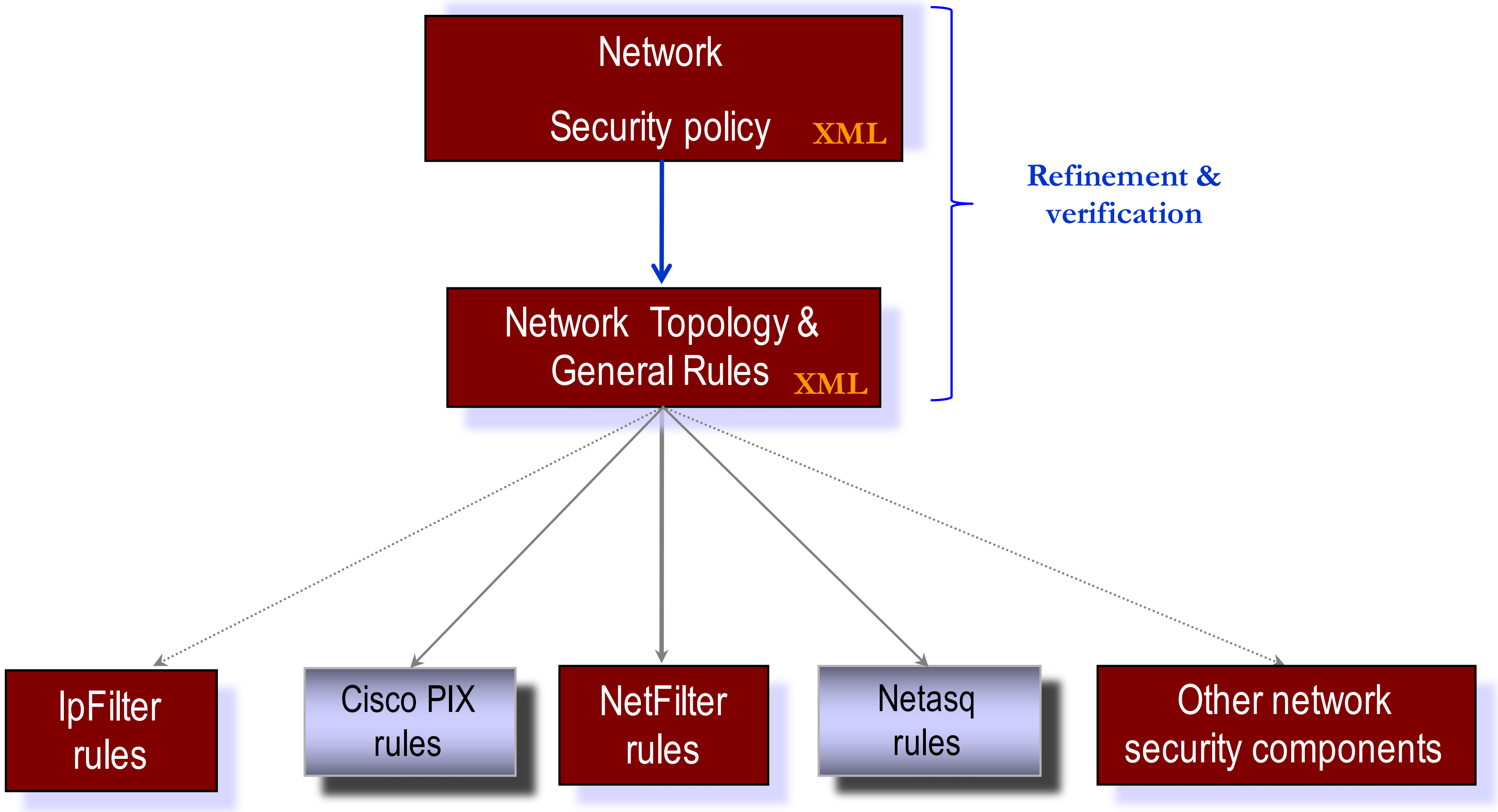
Complete taxonomy of Anomalies



Outline

- Brief Introduction
- Bottom-up analysis of filtering configurations
  - Intra-component analysis
  - Inter-component analysis
- **Top-down refinement of general security policies**
- **Conclusion / Perspectives**

Top-down Approach



Network

Security policy XML

Network Topology & General Rules XML

Refinement & verification

IpFilter rules

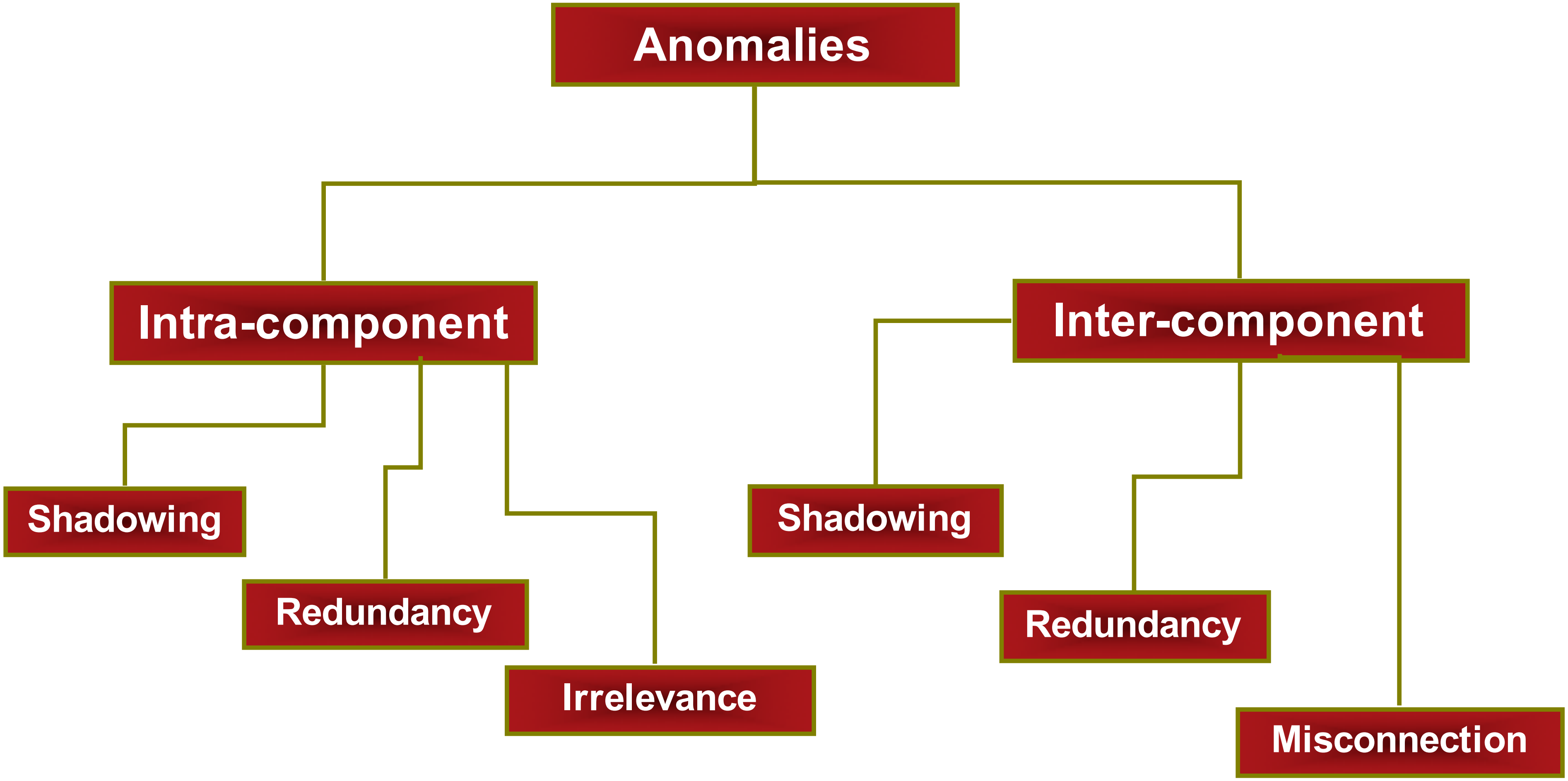
Cisco PIX rules

NetFilter rules

Netasq rules

Other network security components

Address same taxonomy of Anomalies



Anomalies

Intra-component

Shadowing

Redundancy

Irrelevance

Inter-component

Shadowing

Redundancy

Misconnection

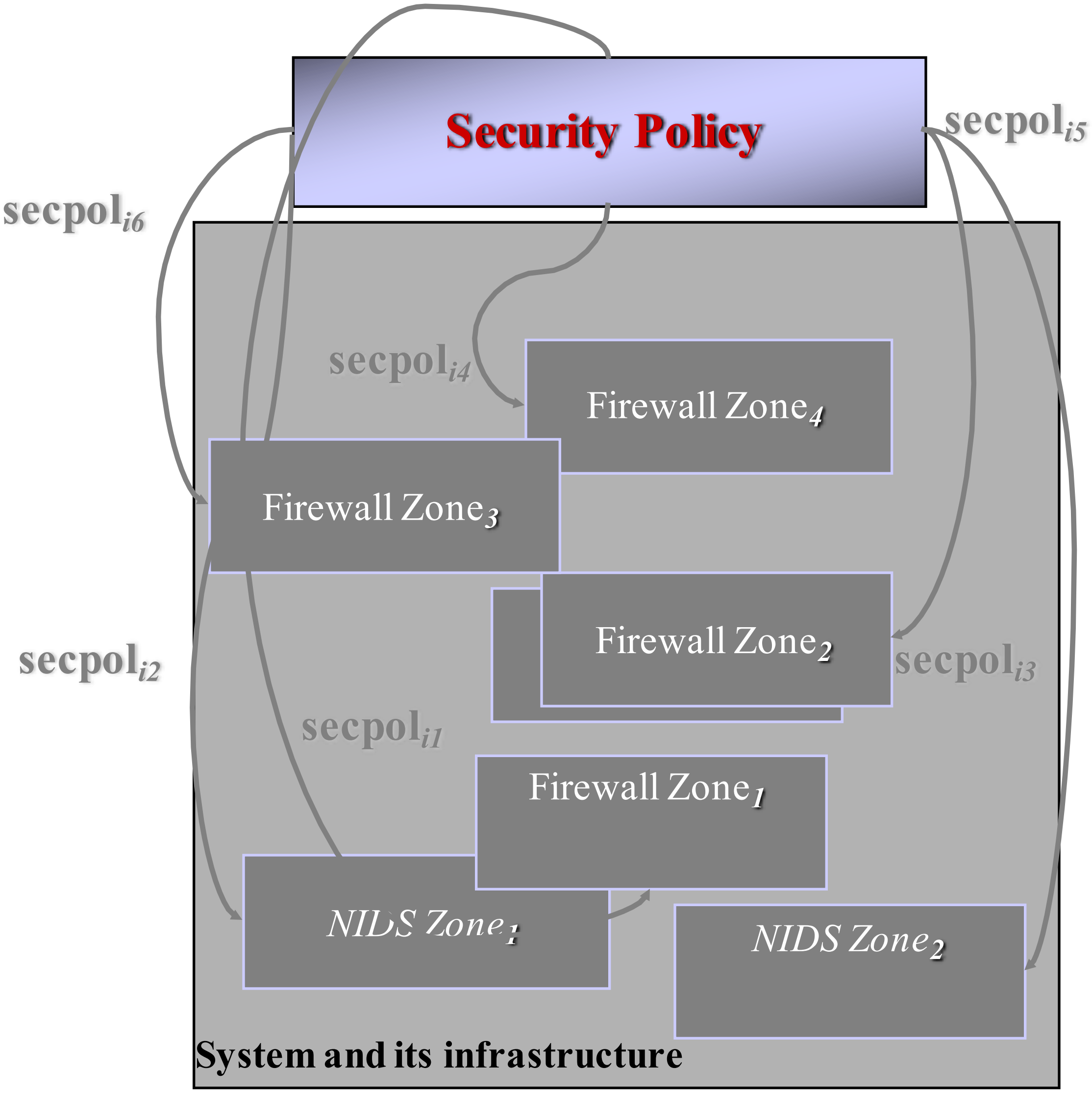
Refinement of Global Policies

Security Policy

- Definition of a global security policy for the whole information system
- Then, perform a transformation process in order to configure a specific instance of the security policy for every component within the information system

Refinement of Global Policies

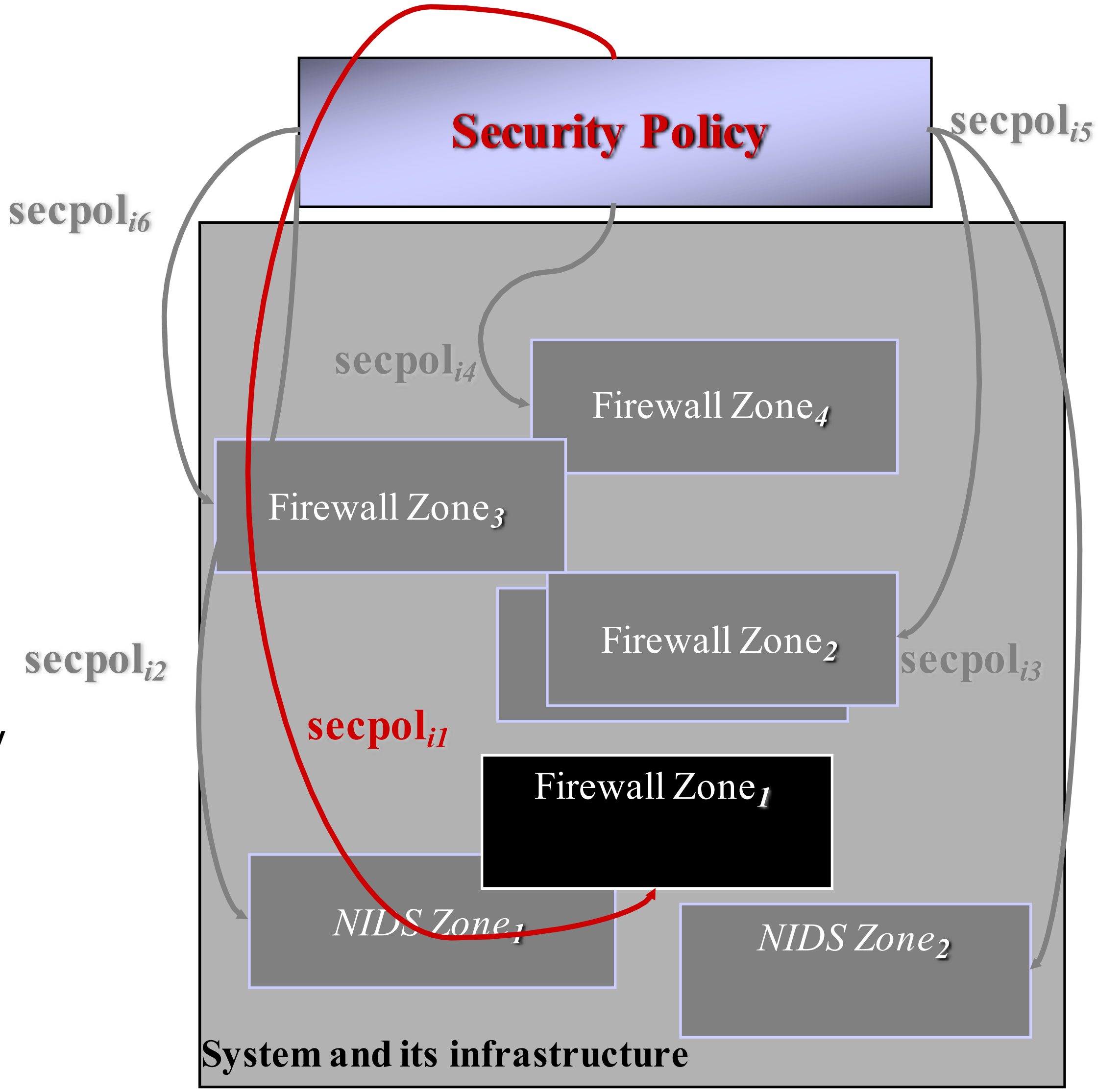
- Definition of a global security policy for the whole information system
- Then, perform a transformation process in order to configure a specific instance of the security policy for every component within the information system



Refinement of Global Policies

- Definition of a global security policy for the whole information system

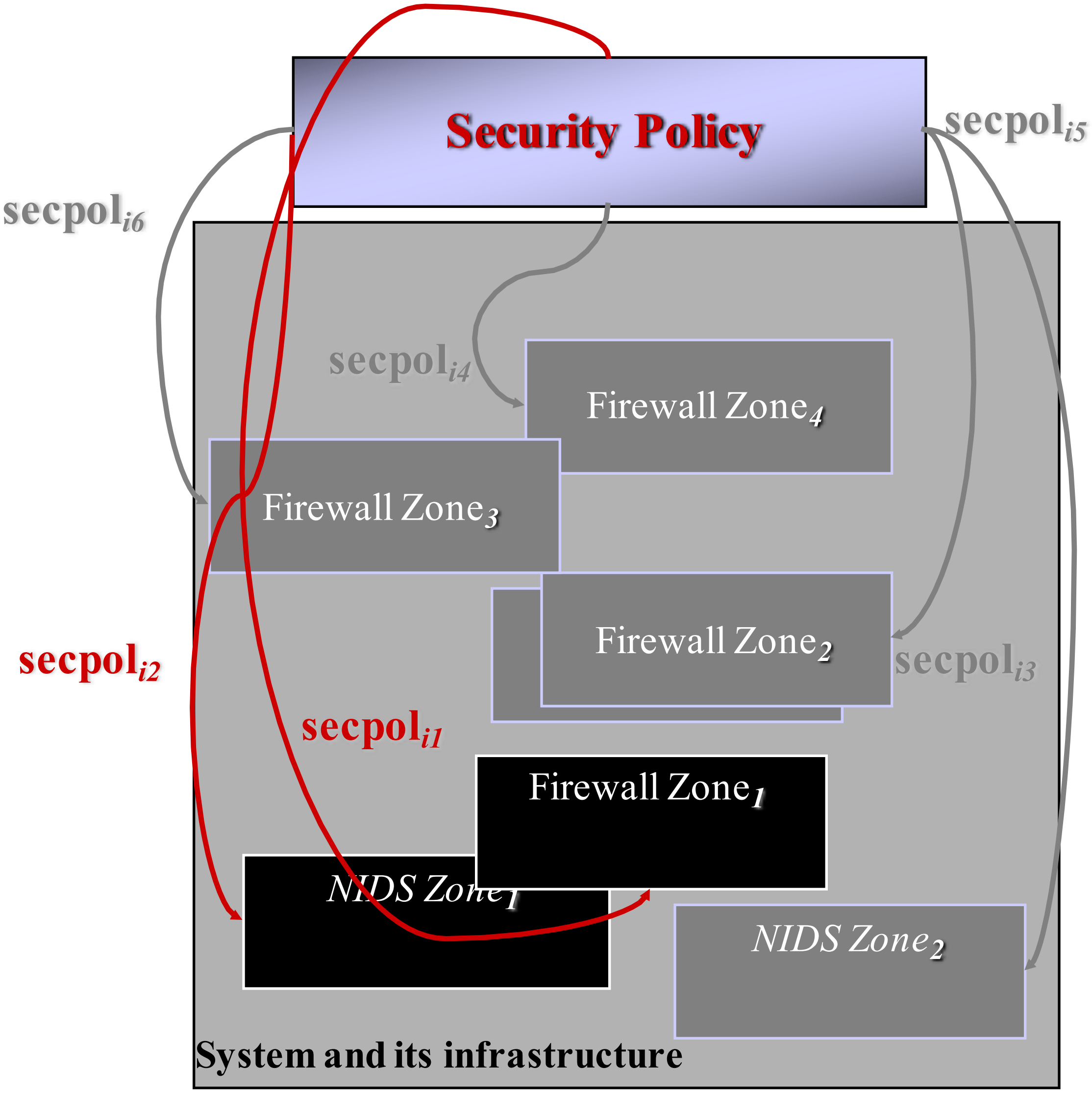
- Then, perform a transformation process in order to configure a specific instance of the security policy for every component within the information system



Refinement of Global Policies

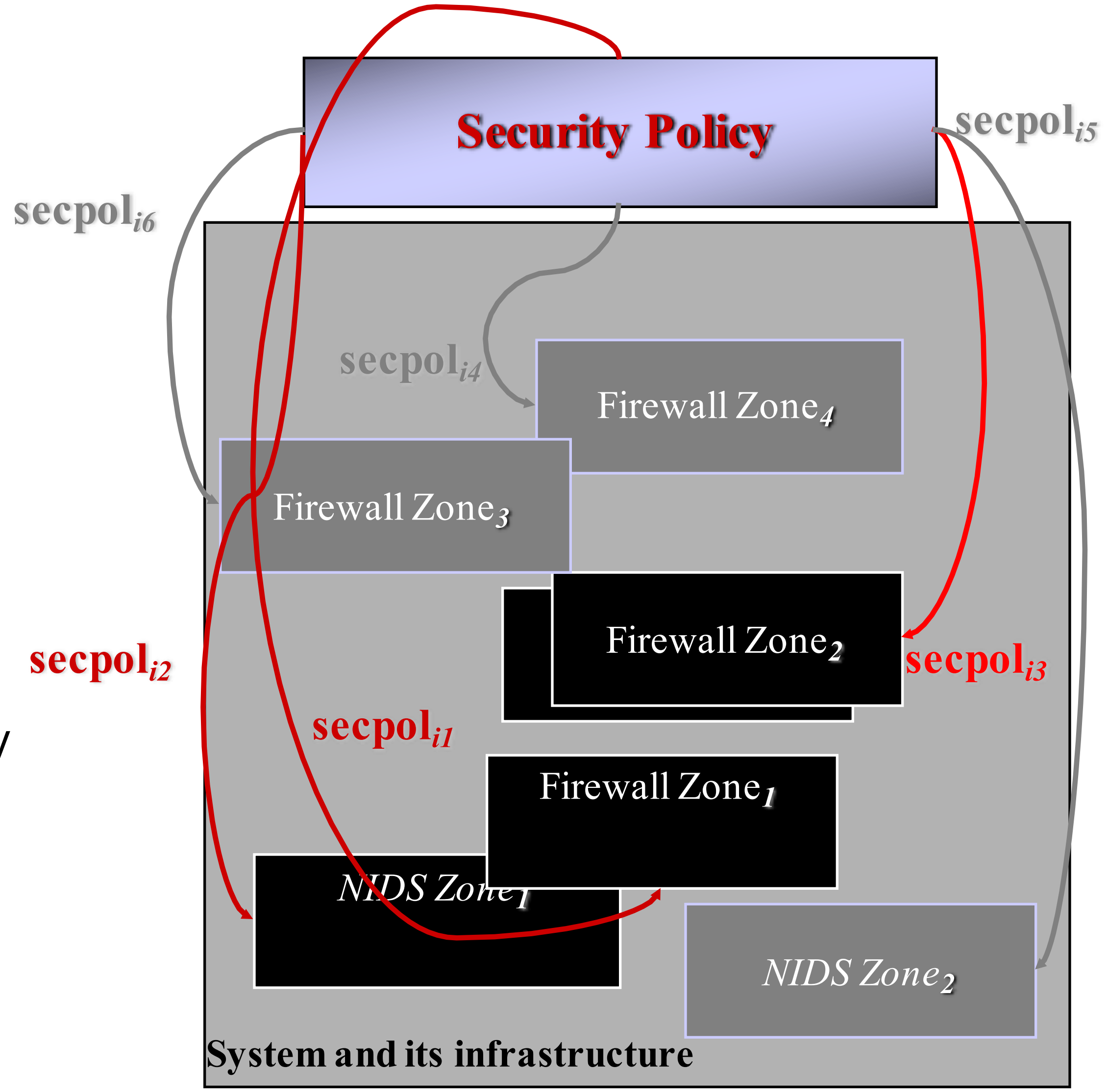
- Definition of a global security policy for the whole information system

- Then, perform a transformation process in order to configure a specific instance of the security policy for every component within the information system



Refinement of Global Policies

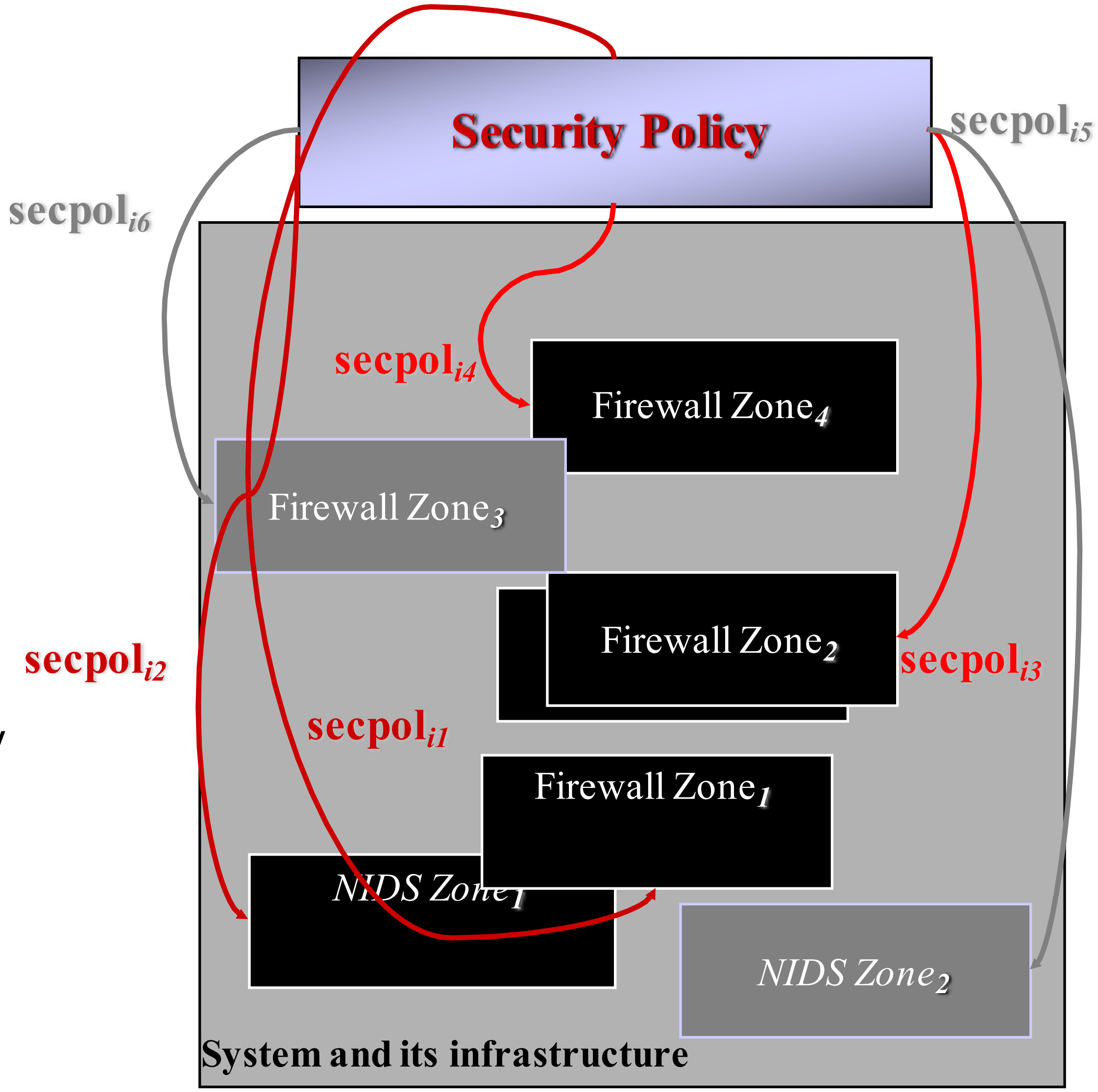
- Definition of a global security policy for the whole information system
- Then, perform a transformation process in order to configure a specific instance of the security policy for every component within the information system



Refinement of Global Policies

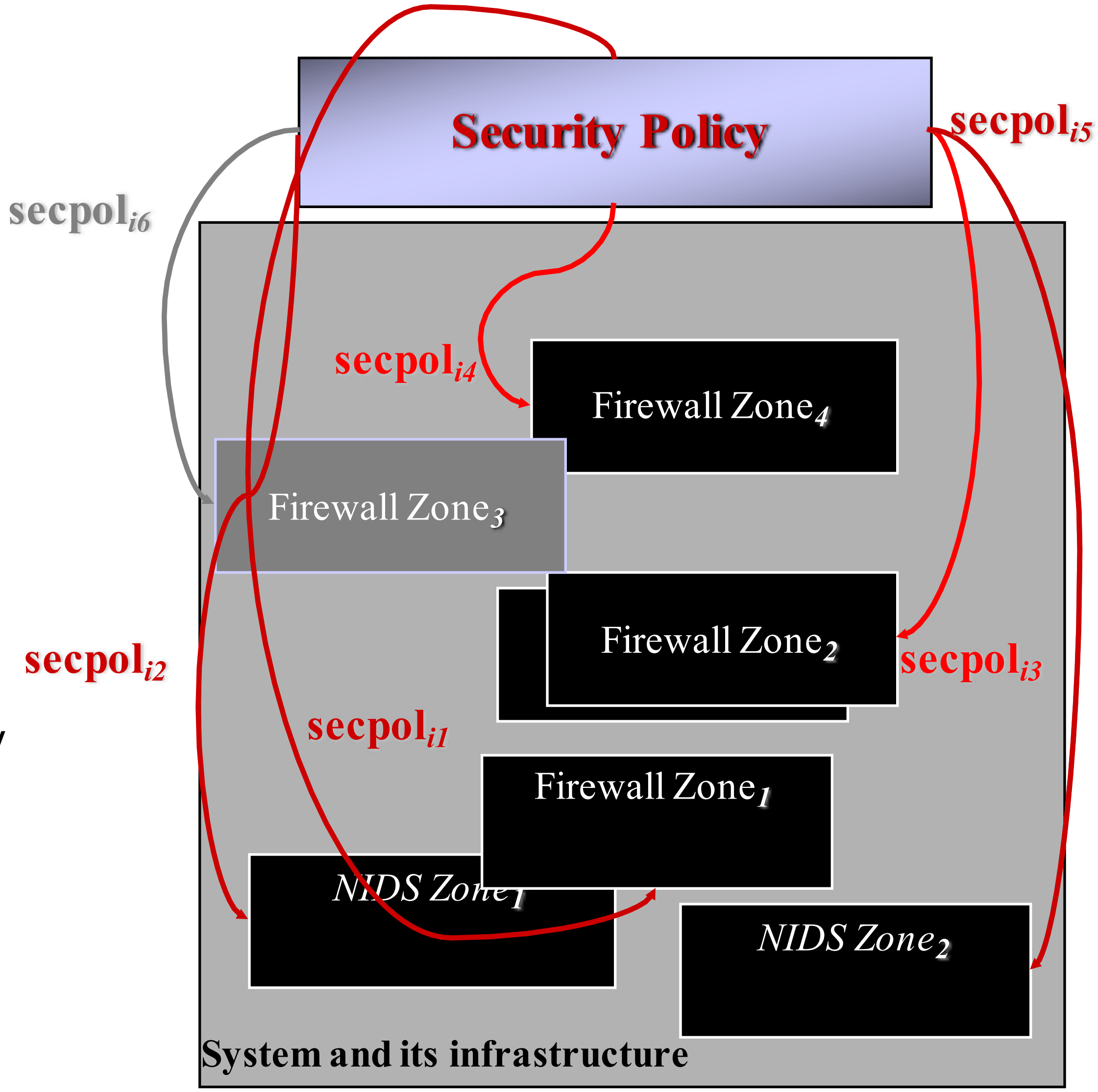
- Definition of a global security policy for the whole information system

- Then, perform a transformation process in order to configure a specific instance of the security policy for every component within the information system



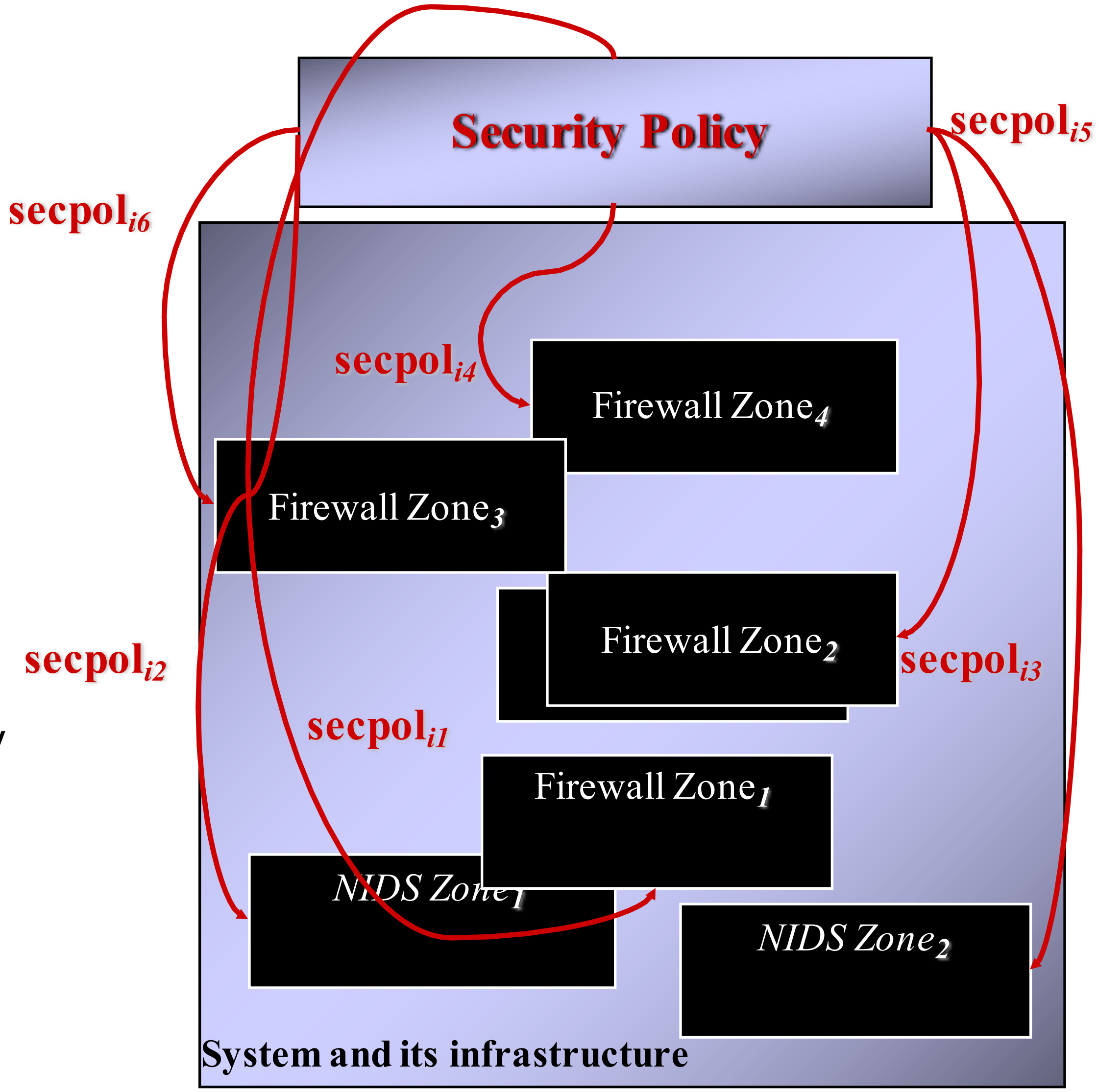
Refinement of Global Policies

- Definition of a global security policy for the whole information system
- Then, perform a transformation process in order to configure a specific instance of the security policy for every component within the information system



Refinement of Global Policies

- Definition of a global security policy for the whole information system
- Then, perform a transformation process in order to configure a specific instance of the security policy for every component within the information system



Specifying network security policies with OrBAC

- Objective of a network security policy
  - *Specify rules to control interaction between hosts that use network services to send messages.*

- Define concrete entities in network domain
  - **SUBJECT:** a host, a group of hosts, a (sub)network, etc. (all identified by their IP addresses)
  - **ACTION:** a network service (e.g., tcp, udp, HTTP, ...)
  - **OBJECT:** a message sent to destination hosts (i.e., subjects)

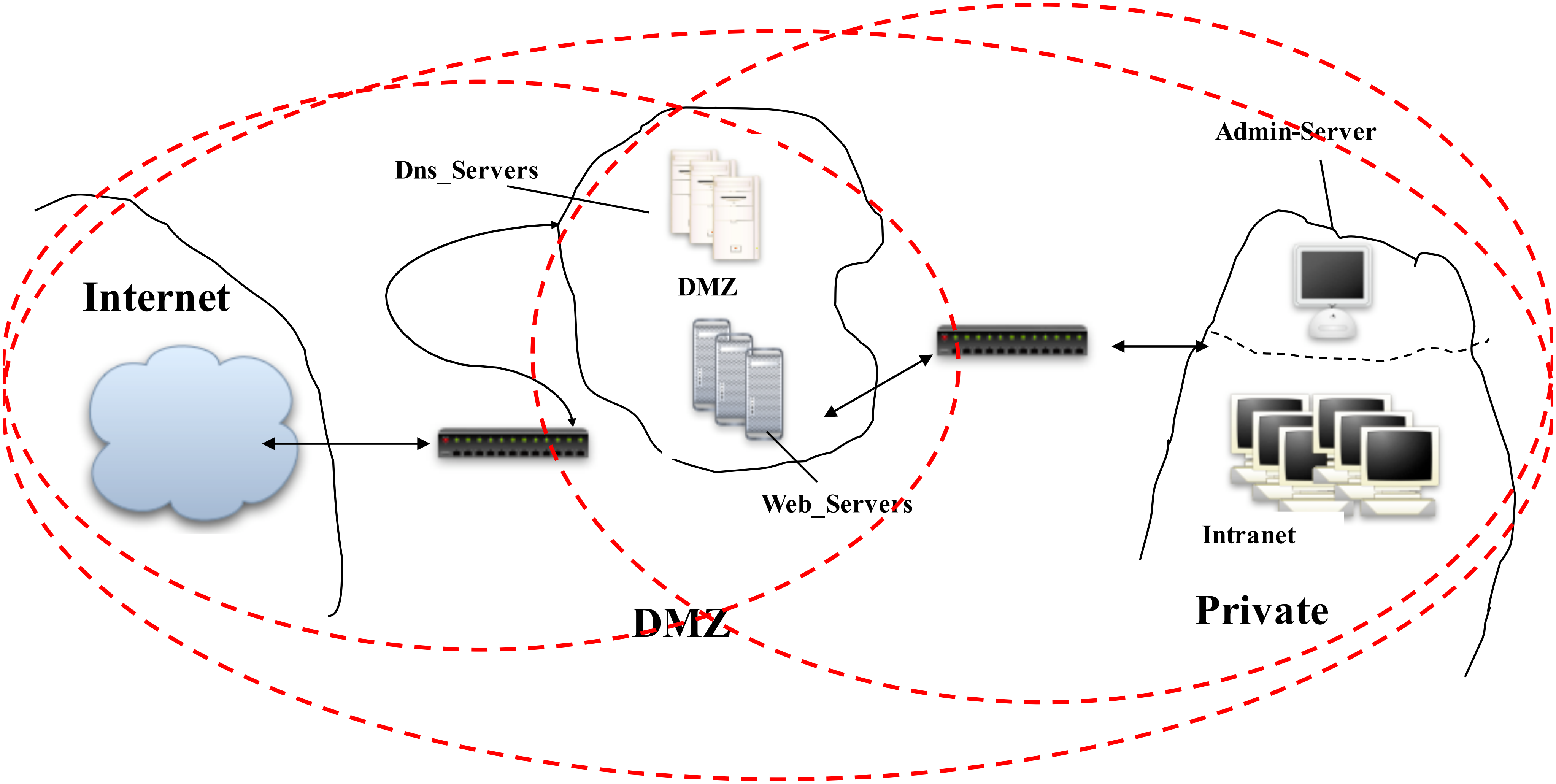
Examples based on the previous network

- **Roles:** abstraction of subjects (i.e., hosts):
  - *Web_servers, DNS_Servers, Admin_server, Internet, Intranet.*

- **Activities:** abstraction of actions (i.e., network services):
  - *Web_http, DNS_resolution, Administration, Mail_SMTP.*

- **Views:** abstraction of objects (i.e., network messages):
  - *to_Web_servers, to_DNS_Servers, to_Admin_server, to_Internet, to_Intranet .*

Sample network



How to specify permissions

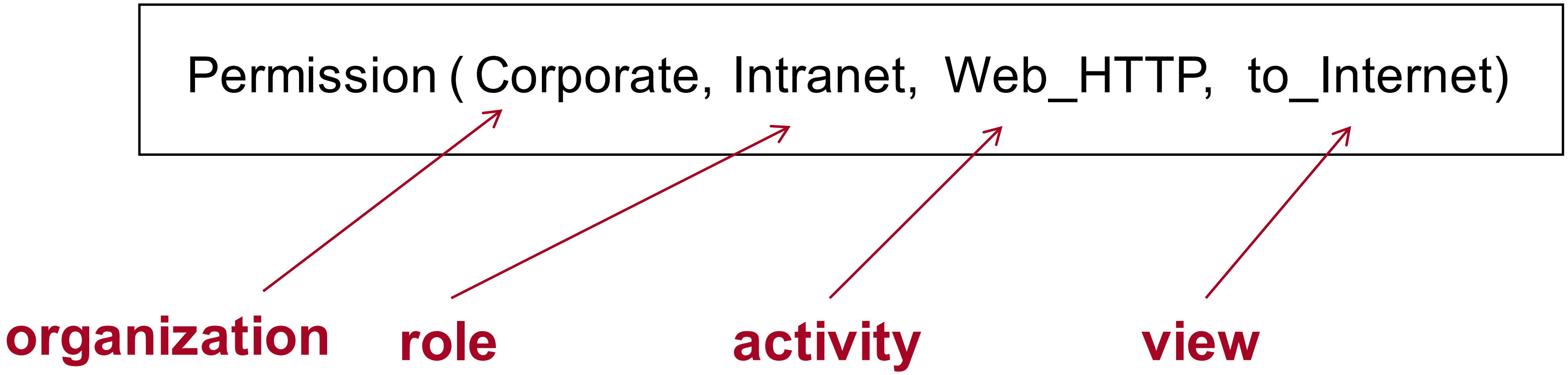
- **Example**:

In the *Corporate network*, *Intranet hosts* can send *web requests* to *Internet hosts*

How to specify permissions

- **Example**:

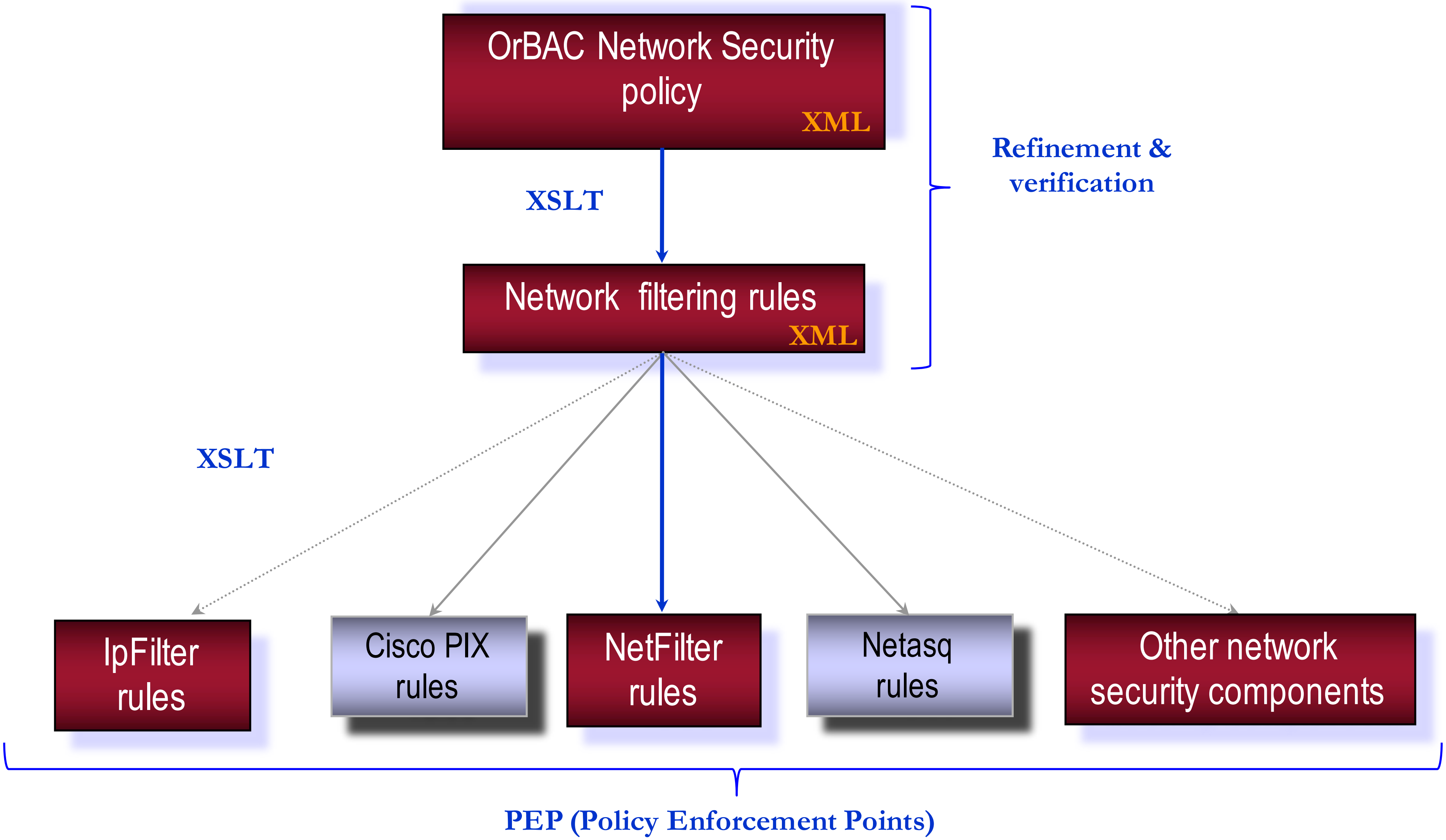
In the *Corporate network*, *Intranet hosts* can send *web requests* to *Internet hosts*



Permission (Corporate, Intranet, Web_HTTP, to_Internet)

organization **role** **activity** **view**

Refinement (MIRAGE example)



OrBAC Network Security policy XML

XSLT

Network filtering rules XML

Refinement & verification

XSLT

IpFilter rules

Cisco PIX rules

NetFilter rules

Netasq rules

Other network security components

PEP (Policy Enforcement Points)

Conclusion

- **Bottom-up approach**
  - Ad hoc analysis of network configurations
  - Analysis of other security components (e.g., VPN routers)

- **Top-down approach**
  - Global approach
  - Dynamic reconfiguration

- **Combining & improving both approaches**