# Contents

1 Introduction  
2 Data models  
   Overall workflow  
3 Harmonization process  
4 Refinement process  
5 Use of the tool  
   Creating a new IT Policy  
   Access control  
   Data protection  
   Harmonization  
   Consistency conflict  
   Minimization  
   Separation of Duty  
   Enrichment  
   Refinement  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Data models</td>
<td>3</td>
</tr>
<tr>
<td>Overall workflow</td>
<td>3</td>
</tr>
<tr>
<td>Harmonization process</td>
<td>5</td>
</tr>
<tr>
<td>Refinement process</td>
<td>5</td>
</tr>
<tr>
<td>Enrichment</td>
<td>6</td>
</tr>
<tr>
<td>Use of the tool</td>
<td>6</td>
</tr>
<tr>
<td>Creating a new IT Policy</td>
<td>6</td>
</tr>
<tr>
<td>Access control</td>
<td>6</td>
</tr>
<tr>
<td>Data protection</td>
<td>10</td>
</tr>
<tr>
<td>Harmonization</td>
<td>12</td>
</tr>
<tr>
<td>Consistency conflict</td>
<td>12</td>
</tr>
<tr>
<td>Minimization</td>
<td>13</td>
</tr>
<tr>
<td>Separation of Duty</td>
<td>16</td>
</tr>
<tr>
<td>Enrichment</td>
<td>18</td>
</tr>
<tr>
<td>Refinement</td>
<td>21</td>
</tr>
</tbody>
</table>
1 Introduction

This document provides an overview of the ITPolicy Tool, which is the tool used to manage IT policies, perform conflict analysis and refine IT level policies (only topological independent) into Abstract Configuration in the PoSecCo workflow.

This service consists of a set of different modules with their own user interface. All of this modules and their options are documented in the following sections.

In Sec. 2 a short description of IT policies and the overall workflow is provided.

For an introduction to the functionalities provided by the ITPolicy Tool see Sec. 3, 4.

Finally, Sec. 5 is devoted to explaining how the tool works, its functionalities and the graphical user interface.
2 Data models

The IT Policy Tool is built on the IT layer Security meta-model. The IT layer Security meta-model supports the representation of authentication and access control properties. The Unified Modeling Language is used for the definition of the model. Specifically, UML class diagrams are used to define the structure of the meta-model, as UML class diagrams are today the most common solution for the representation of conceptual and logical structure of information containers.

Figure 1: IT level Security Meta-model.

Figure 1 shows a graphical representation of the complete IT Policy meta-model. The IT meta-model consists of six interconnected sub-models, each focused on a specific aspect of the security policy.

- the Principal meta-model is used to describe the organization of identities, how users are structured into groups and how roles are assigned to users;
- the Security rule meta-model describes the structure of IT policies and targets and introduces a taxonomy of authorization and authentication rules;
- the Privilege meta-model is used to describe the structure of privileges that are specified for system authorizations;
- the Authentication property meta-model is used to describe the structure of properties that are associated with authentication rules;
- the Resource meta-model describes the structure of the static, dynamic and communication resources that are associated with the privileges in the authorizations and with the authentication properties.
- the Security domain meta-model contains the entity that denotes the concept of security domain, supporting the realization of policies.

Overall workflow

Figure 2 shows the general overview of the workflow which implements the generation from IT policies of abstract authentication and authorization configurations. The workflow is composed by the following steps:

- Retrieve the IT Policy and the landscape
Process the IT Policy and the landscape. The first step is to retrieve the IT Policy and the landscape from the PoSecCo repository. Internal representations of the IT Policy and Landscape descriptions are created. The formats of these representations support the evaluation of queries against these descriptions.

Determine Enrichment Type. The IT Policy refers to entities that may correspond to one or more entities in the landscape as represented at the IT level in the PoSecCo Repository. Each entity may be specialized (i.e., enriched) according to its profile, as described in the landscape or as provided by the security designer in the interaction with the tool supporting this phase. For instance, ITInterfaceModels may correspond to one or more ITInterfaces in the landscape. Each ITInterface in the landscape has a particular type (HTTPMethod for web applications, WebServiceOperation for web services and EJBMethod for Enterprise Java Bean applications). In the same way, an ITResourceModel instance may correspond to several ITResources, and one of them can be an instance of a family of systems (e.g., DBMSs), associated with a specific version (e.g., MySQL 5.4).
Enrichment of Entities. This step covers the actual enrichment of the IT Policy. For the management of the access control policy, we assume that each ITSecurityRule within the IT Policy is decomposed and further enriched.

The authentication and authorization subjects, which are instances of ITPrincipal, are enriched to entities that are in use by authentication information providers. These providers include LDAP Directory Servers, Windows Authentication and Linux Pluggable Authentication Modules. In the enrichment process, human intervention is required to map IT Policy subjects to entities stored in authentication sources and to provide the location of the authentication providers. For the enrichment of interface objects, we consider those ITInterfaceModels that refer to one or more ITInterfaces in the landscape. For instance, in the generation of access control policies for J2EE application servers the relative path of the ITInterface corresponding to the ITResource deployed on the J2EE server needs to be computed by traversing the landscape model.

Security rules describe the authentication and authorization configuration of the system. An important component is the definition of ITActions, which the IT Policy refers to. ITActions will be enriched with concrete actions that can be performed on real systems. For example, an ITAction such as ‘access’ may be mapped to HTTP methods such as ‘GET’ and ‘POST’.

Consider Candidate Solutions and Choose. This phase represents the application of the technical refinement strategies, as discussed above. Depending on the type of element, a variety of approaches will be used. We foresee approaches that are automated, or manual or a combination of both. We identified that the translation of IT Policies into abstract access control configurations (for operating system, DBMSs, Web servers and Web application servers) does not yield a large number of alternative configurations. Hence, we expect a manual approach for selecting an authorization configuration out of a set of alternatives. This is in contrast to network policies. The processing of the network policy will instead rely on the evaluation of a potentially large number of alternative configurations, with the need to apply a selection criterium to choose the “optimal” configuration among all those that would be able to satisfy the design requirements.

Generate Abstract Configurations. The outcome of the earlier step is used to compose the configuration rules of an abstract authorization configuration and an abstract authentication configuration, respectively. All the configuration rules in an abstract configuration correspond to one application/ITResource on which the abstract configuration can be deployed. Thus, when composing abstract configurations, configuration rules are grouped by the application/ITResource on which the rules are to be deployed.

3 Harmonization process

This section presents the list of reasoning services provided by the ITPolicy Tool in order to perform conflict analysis.

Policy Incompatibility: given a set of authorizations $A$, check whether exist pairs of authorizations $(a_1, a_2)$ such that $a_1$ and $a_2$ apply to the same request and have opposite sign;

Policy Minimization: given a set of authorizations $A$, check whether a subset of those authorizations $R$ exists that is dominated by other authorizations;

Separation Of Duty Satisfiability: given a set of authorizations $A$, check whether they satisfy a set of Separation of Duty (SoD) constraints.

4 Refinement process

The policy refinement process consists of translating an enriched, integrated model of the IT policy and the system’s landscape (FSM), into abstract configurations. Essentially, refinement performs all the necessary lookups on the landscape and IT policy to collect information required for authorization/authentication abstract
configurations. The outcome is a set of abstract configurations in case the IT policy describes authentication or authorization rules.

Abstract Configurations are derived from IT Policies using models, languages and tools that depend on Semantic Web technologies. There are several aspects that motivate this decision. Specifically, the Semantic Web solutions support the construction of a concise and expressive representation of the elements of the policies.

**Enrichment**

IT policies that describe authorization and authentication rules have to be enriched before to refine them to technology-specific abstract configurations for Database Management Systems, File Systems and Web Applications. The enrichment of IT policies that contain authorization or authentication rules happens in two phases. The first phase, also called the first level of enrichment, enriches the IT Policy with further technology independent IT-level concepts.

## 5 Use of the tool

**Creating a new IT Policy**

As a first task, the plug-in requires to create an empty project (see Figure 3(a)). Once the project has been created, the user must create the file that will contain the policies.

At this point the plug-in will provide a window for choosing the meta-models used in the project (see Figure 3(b)). When the meta-models are imported in the tool, the user can insert the data about the IT policies that he wants to create. Then, the user will typically insert the elements of the model using the forms associated with the entities in the tabs.

After the creation of the file, the user can select the file and open it with the ITPolicy Editor. The editor permits to create/modify/delete new ITPolicies (and also Authorizations, Authentications etc.).

**Access control**

In order to create the authorization related to a business security requirements (for instance R2.1 - “Only Application Administrators shall access the Tomcat admin Web console”), the user can use the SecurityRule tab (See Figure 5).

![Figure 5: Selection of the SecurityRule tab.](image)
In the SecurityRule tab the user can insert all the data (e.g., name, privilege, subject) needed to define the authorization according to the IT Security meta-model. Figure 6 shows all the information needed to create the SystemAuthorization related to business security requirements R2.1.

The ITPolicy Tool permits to link ITElements (e.g., ITPolicySet) to Business Security Requirements in order to create the Policy chain. This functionality allows the user to select which is the business security requirements related to the specific ITElement, in this case to the ITSystemAuthorization defined above (see Figure 7).

Figure 7: Selection of the Business Security Requirement R2.1.
(a) The window used for the generation of the project.

(b) The window used to choose the project type.

Figure 3: Creation of new project.

(a) The window used to add information about the project.

Figure 4: Creation of new project.
(a) Insert name type and sign.

(b) Insert the subject of the authorization.

(c) Insert the privilege.

(d) Insert the object of the authorization.

Figure 6: SystemAuthorization “ITP30_SPDBAdmins_canAccess_DBMS_conf”. 

Data protection

To create the data protection requirement related to a Business Security requirement (for instance, R3.2 - “The confidentiality and integrity of cardholder data must be ensured wherever it is stored”), the user must create an ITEncryptedData as described in D2.5. To perform this task the user can use the Resource tab (see Figure 8).

![Figure 8: Selection of the resource tab.](image)

Figure 8: Selection of the resource tab.

Figure 9 shows the creation of an ITEncryptedData. After the creation of the element, as before, the user has to link the element to the Business Security Requirement R3.2 and also to the Functional representation of the element. In order to perform this activity the user can use the Business Security Requirements section (see Figure 10(a)) and the IT Functional Elements section (see Figure 10(b)).

![Figure 9: Information needed to define an ITEncryptedData.](image)
The views are quite similar, each of them allows the user to import a file containing the business security requirements or the IT Functional elements. The imported items are shown as a check-box list. Each item is composed by (a) the id and (b) a brief description.

To help the user during the creation and editing of data, a view has been introduced that allows the user to navigate between the entities entered above (see Figure 11).

As previously discussed, when the policy creation phase is complete, the user can use the plug-in not only to export the data into an XMI file, but it can also create the ontology, expressed in an OWL file. This feature of the plug-in allows the user to obtain a representation of the security policy that can be processed by a standard OWL reasoner. In Figure 12 we can see both aspects described above.

The integration of reasoning is planned to be relatively direct, thanks to the flexible architecture that characterizes Eclipse. We created a view that shows the messages returned by HermiT, to provide feedback to the user and inform him of possible conflicts within the model. The Eclipse architecture allows us to manage well this integration, something that would be harder if using closed-source or less flexible frameworks.
Harmonization

Consistency conflict

In order to show a Consistency conflict example we have introduced in the testbed the following authorizations.

<table>
<thead>
<tr>
<th>Sign</th>
<th>Name</th>
<th>Grantor</th>
<th>GrantedTo</th>
<th>ITPrivilege</th>
<th>ITSecurityObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative</td>
<td>SP_Admin-canAccess- Tomcat_Configuration</td>
<td>Angus</td>
<td>SP_Admin</td>
<td>canAccess</td>
<td>Tomcat_Configuration</td>
</tr>
<tr>
<td>Positive</td>
<td>SP_App_Admin-canAccess- Tomcat_Folder</td>
<td>Angus</td>
<td>SP_App_Admin</td>
<td>canAccess</td>
<td>Tomcat_Folder</td>
</tr>
</tbody>
</table>

In order to discover consistency conflicts the user can perform Consistency checking by clicking on the start Consistency check button (see Figure 13).

Figure 13: start Consistency check —— Minimization Reasoning service button

A wizard aids the user in the selection of which type of service to perform (see Figure 14). There are three types of service:

**Consistency Checking:** checks whether the policy is consistent or if it contains one or more conflicts, and returns an adequate boolean value;

**Consistency Checking + Explanation:** checks whether the policy is consistent or not; if not, it returns a list of explanations that identify the source of the inconsistencies;

**Consistency Checking + Repair:** checks whether the policy is consistent or not; if not, it returns a list of fixes that can remove the identified inconsistencies.

For the purpose of this example, we perform the following service, Consistency Checking + Explanation. Figure 15 shows the result of the check given by the reasoner, Figure 16 shows the result in a more user-friendly way. The conflict explanation is more detailed, the user can see all the relationships involved in the conflict.

Figure 14: Wizard
Minimization checks are special cases of Consistency checks. They do not identify misconfigurations or inconsistencies in the model, but identify redundant elements in the model, that can be removed with no functional impact. For instance, we have inserted into the testbed the following authorizations:

<table>
<thead>
<tr>
<th>Sign</th>
<th>Name</th>
<th>Grantor</th>
<th>GrantedTo</th>
<th>ITPrivilege</th>
<th>ITSecurityObject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>SP_Admin-canAccess-Tomcat_Configuration</td>
<td>Angus</td>
<td>SP_Admin</td>
<td>canAccess</td>
<td>Tomcat_Configuration</td>
</tr>
<tr>
<td>Positive</td>
<td>SP_App_Admin-canAccess-Tomcat_Configuration</td>
<td>Angus</td>
<td>SP_App_Admin</td>
<td>canAccess</td>
<td>Tomcat_Folder</td>
</tr>
</tbody>
</table>

As for the Modality Conflict the user can select to perform Redundancy checking by clicking on the start Minimization check service button (see Figure 17).
A wizard (like the one for modality checking) aids the user in the selection of which reasoning service (e.g., minimization checking, minimization checking + explanation) he wants to perform. For the purpose of this example we select to perform the minimization checking + explanation service. Figures 18 and 19 show the result of the check.

As for the modality conflict, the explanation about the minimization check is more detailed and allows the user to take into account the relationships involved in the conflict.
Figure 19: Minimization check result - ITExplanation.
Separation of Duty

A common class of constraints represented in security policies is *Separation of Duty* (SoD), described in D2.4. These constraints follow the common best practice for which sensitive combinations of permissions should not be held by the same individual in order to avoid the violation of business roles. In the testbed scenario, the rules involved in an SoD are the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP_App_Admin</td>
<td>SC_Admin</td>
</tr>
<tr>
<td>SP.Sys_Admin</td>
<td>SC_Admin</td>
</tr>
<tr>
<td>SP_DB_Admin</td>
<td>SC_Admin</td>
</tr>
<tr>
<td>SP_App_Admin</td>
<td>SC_Acc_Manager</td>
</tr>
<tr>
<td>SP_Sys_Admin</td>
<td>SC_Acc_Manager</td>
</tr>
<tr>
<td>SP_DB_Admin</td>
<td>SC_Acc_Manager</td>
</tr>
<tr>
<td>SP_App_Admin</td>
<td>SC_Batch</td>
</tr>
<tr>
<td>SP_Sys_Admin</td>
<td>SC_Batch</td>
</tr>
<tr>
<td>SP_DB_Admin</td>
<td>SC_Batch</td>
</tr>
<tr>
<td>SP.Batch</td>
<td>SC_Acc_Manager</td>
</tr>
<tr>
<td>SC_Admin</td>
<td>SC_Acc_Manager</td>
</tr>
<tr>
<td>SC_Admin</td>
<td>SC_Batch</td>
</tr>
<tr>
<td>SC_Admin</td>
<td>BP</td>
</tr>
<tr>
<td>SC_Acc_Manager</td>
<td>BP</td>
</tr>
<tr>
<td>SC_Batch</td>
<td>BP</td>
</tr>
</tbody>
</table>

For the purpose of this example we consider only the SoD constraint between the roles (a) *SP_App_Admin* and (b) *SC_Admin*. In order to show the violation of the SoD constraint above we have added the following *ITRoleAuthorization*:

<table>
<thead>
<tr>
<th>Sign</th>
<th>Name</th>
<th>Grantor</th>
<th>GrantedTo</th>
<th>enabledRole</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative</td>
<td>SP_App_Admin- SC_Admin</td>
<td>Angus</td>
<td>SP_App_Admin</td>
<td>SC_Admin</td>
</tr>
<tr>
<td>Positive</td>
<td>Angus- canEnable- SP_App_Admin</td>
<td>Angus</td>
<td>Angus</td>
<td>SC_Admin</td>
</tr>
<tr>
<td>Positive</td>
<td>Angus- canEnable- SC_Admin</td>
<td>Angus</td>
<td>Angus</td>
<td>SP_App_Admin</td>
</tr>
</tbody>
</table>

The first *ITRoleAuthorization*, in the table above, states the SoD constraint. The other *ITRoleAuthorizations* state that the user *Angus* can enable both the role *SP_App_Admin* and the role *SC_Admin*, thus a violation of the SoD constraint. In order to discover SoD constraint violation, the user can perform *SoD checking* by clicking on the *start SoD check service* button (see Figure 20).

![Figure 20: start SoD check service button](image)

As for the previous reasoning services, a wizard aids the user in the selection of which type of service to perform. Figures 21 and 22 show the result of the *SoD Checking + Explanation service*. 
Figure 21: Result of SoD checking.

Figure 22: Result of SoD checking.
**Enrichment**

After the Harmonization phase, the user can perform the Enrichment phase. For this example, we will focus on the case of the enrichment of an authorization related to an ITPolicySet Authorization. The user can select the ITElement to refine, in this case we will use the individual `itps_authz`. Figure 25 shows the list of candidates for the enrichment process. The IT Policy Tool identifies that the ITEElement is related to an ITPolicySet and enables the enrichment module dedicated. But the action used to define the policy is a generic `access` action, and it is not explicitly related to the type of the resource.

Figure 23: Selection of the element to enrich.
Figure 24: Selection of the element to enrich.
Figure 25: Selection of the element to enrich.
Refinement

The last step, in order to generate Abstract Configuration is to perform the refinement process. The user can start the refinement process by clicking the button in Figure 26.

Figure 26: Start refinement phase.

As for the enrichment process the user can select which element to refine. For this example, we will use the `itps_authz` (see Figure 28) which contains the authorization enriched before. After the selection, the IT Policy Tool retrieves all the information about the resource (e.g., node) where the security mechanisms have to be implemented (see Figure 28), if the user clicks on the Start Analysis button, the refinement process can start. The process is (semi-)automatic. The user has to aid the ITPolicy Tool only when some information is missing and has to be inserted manually.

Figure 27: Selection ITPolicy to refine.
Figure 28: View of the resources involved in the refinement process.