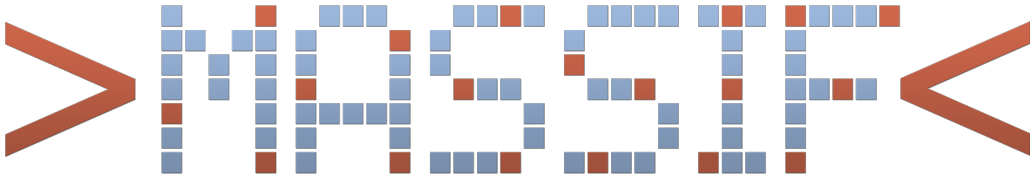

MAnagement of Security information and events in Service InFrastructures

MASSIF
FP7-257475

D4.2.3 - Predictive Security Analyser

Activity	A4	Workpackage	WP4.2
Due Date	Month 24	Submission Date	2013-01-31
Main Author(s)	Jürgen Repp (Fraunhofer), Roland Rieke (Fraunhofer)		
Contributor(s)	Rodrigo Diaz (Atos), Maria Zhdanova (Fraunhofer)		
Version	v1.0	Status	Final
Dissemination Level	PP	Nature	P
Keywords	predictive security analysis, analysis of process behavior, security monitoring, security assessment tools		
Reviewers	Luigi Coppolino (Epsilon) Romain Giot (FT)		

Part of the Seventh
Framework Programme
Funded by the EC - DG INFSO

Version history

Rev	Date	Author	Comments
V0.1	2012-06-04	Roland Rieke (SIT)	initial version
V0.9	2013-01-22	Jürgen Repp, Roland Rieke (SIT)	review version
V1.0	2013-01-31	Pedro Soria (Atos)	final review and official delivery

Glossary of Acronyms

ACL	Allegro Common Lisp
CLIM	Common Lisp Interface Manager
EPC	Event-driven Process Chain
IDMEF	Intrusion Detection Message Exchange Format
MOTIF	Graphical user interface specification and widget toolkit for building X Window applications
NIST	National Institute of Standard and Technologies
JDK	Java Development Kit
PL	Preamble Language
PN	Product Net
PNML	Petri Net Markup Language
PSA	Predictive Security Analyser
SIEM	Security Information and Event Management
XML	Extensible Markup Language
XSD	XML Schema Definition

Executive Summary

This deliverable describes the implementation of a new generation Predictive Security Analyser (PSA). The PSA takes as an input: (a) a process model, (b) the security requirements of the process, and (c) real-time events from the process execution. If a critical state or a process anomaly is detected, the PSA generates an alert that is disseminated to the MASSIF framework for further processing.

Security analysis for event-driven processes requires a process model that fits to the incoming events. Specific security properties can be used to restrict the allowed process behaviour further. Unfortunately, in most practical cases, a model of the control flow of a process, such as a semi-formal workflow specification, is not available. So the necessary model has to be constructed from artifacts, e.g., documentations of parts of the process with varying degree of details, interviews with involved personnel, and probably an analysis of historic event logs.

In order to assist the persons in charge of tool adaptation with regard to the specification of application processes, the PSA supports the use of process specifications that have been generated by external tools. Specifically, it is possible to use a Petri net specification that has been generated by the ProM [15] tool[1].

The PSA implements the techniques and methods developed in the previous work within WP4.2 [9, 12, 2], and complements the semi-automatic uncertainty management tool from D4.1.3 [10]. It can perform a dynamic runtime analysis and an intensive simulation analysis of the project scenarios. Thus, it will assist and support the advanced SIEM framework – developed in the MASSIF project – in making important decisions concerning countermeasures and reactions against upcoming security threats.

Besides the use within the MASSIF framework [1, 11, 7], we consider the PSA as a component that implements a meta-system to be used within an open architecture of a generic security strategy measurement and management system improving cyber-security in Future Internet [3, 4, 14, 13].

[1] http://www.promtools.org/prom6/

Contents

List of Figures

List of Tables

1 Introduction

This deliverable is the result of task 4.2.3 as part of work package 4.2. The outcome of this task - the PSA - is a software tool for dynamic runtime security analysis and intensive simulation analysis of the project scenarios. The prototype presented here implements advanced techniques for the evaluation of security-related events and their interpretation with respect to the monitored processes and their required security properties. These techniques should enable methodologies for performing dynamic process analysis at runtime, detecting any potential violation of the security properties.

This final prototype of the PSA delivered in D4.2.3 incorporates the solutions from D4.1.3 [10], namely (1) mapping incoming events to formal representations, and (2) reasoning about security with uncertain and incomplete knowledge about processes and their relation to these events.

Specific solutions within the prototype D4.2.3 address the issue of *prediction of close-future process actions*. The analytical approach taken for the prediction of close-future actions within a process is sketched in Figure 1.1.

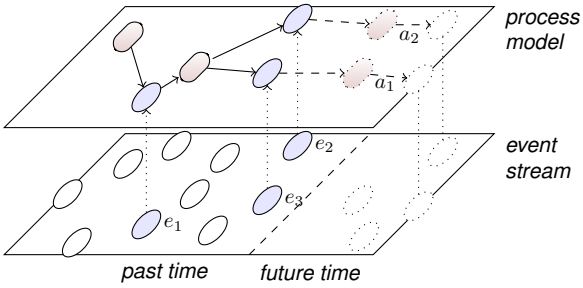

Figure 1.1: Prediction of close-future process actions

The blue ellipses in this figure denote the events and the red chamfered rectangles denote process actions. Process actions are not directly visible in the event stream. Therefore, the process state can only be reflected in the model by analysis of the events. In the example in Figure 1.1, the process execution trace and the current state of the process is computed using the measured events e_1, e_2, and e_3. The dotted arrows denote the synchronisation of the process model with the event stream and the solid lines denote the transitions within the process model. The process description is formalised in the process model. Therefore, the process behaviour can be computed starting with the current state of the process. This behaviour contains possible future actions to be predicted. The dashed arrows denote the predicted close-future process behaviour. Figure 1.1 shows possible close-future actions a_1 and a_2.

The novel methods implemented in this prototype have been described in D4.2.2 [12].

1.1 Analysis of Security Requirements

Besides issues like dependability, redundancy and fault tolerance, analysis of the four scenarios considered reveals the need for enhanced *security-related* features of future Security Information and Event Management (SIEM) platforms (cf. D.2.1.1 [8]). These features go beyond what is currently supported by existing solutions. Overall a lack of capability to model incidents at an abstract level is perceived. From the scenarios investigated, and the current SIEM limitations observed, the guidelines in Table 1.1 have been identified to be relevant for the work within this deliverable.

Guideline	Description
G.S.4. Predictive security monitoring	Predictive security monitoring allows to counter negative future actions, proactively. There is a crucial demand for early warning capabilities. Moreover, the limitations with regards to the Managed Enterprise Service point to the fact that dealing with unknown or unpredictable behaviour patterns is not sufficient in current SIEM solutions (cf. [8]).
G.S.5. Modelling of the events and their relation to other, possibly external, knowledge	A basic precondition of prediction and simulation as well as of attack analysis is the proper representation of the security requirements and any relevant information about the system as well as any knowledge about the actual and possible behaviour. When reasoning under incomplete information it is not only decisive to properly gather and describe the information valid, but it is also required to develop novel methods based on discernibility, probability or plausibility in order to reason about uncertainty.

Table 1.1: Guidelines concerning security addressed by this deliverable

In terms of *trustworthiness* considerations, we assume that data we use for analysis are already processed and provided by trustworthy MASSIF components.

In terms of *legal* considerations, we assume that data we use for analysis are already processed and provided in a form compliant with the legal requirements stated in [8]. With respect to the specific guideline *G.L.5* "Least Persistence Principle. Only data strictly needed for security guarantee must be kept, while unnecessary details must be deleted or made anonymous.", we provide technical means, which can be applied to fulfill the related *Requirement 5.1* "Mechanisms must be provided to filter data containing information not relevant to security processing" that is given in Annex B.1.5 of [8]. For this purpose, the tool provides a menu to define a schema mapping where all data containing information not relevant to security processing can be filtered out (cf. Figure 4.4).

1.2 Glossary adopted in this deliverable

As agreed by the MASSIF Consortium, the main reference of security glossary is provided by the National Institute of Standard and Technologies (NIST) [5].

1.3 Structure of the document

The remainder of this deliverable is organised as follows.

Some general information about the delivered prototype is summarised in Chapter 2. Technical and legal preconditions for using the tool are described in Chapter 3. The technological and implementation details and basic usage of the tool are described in Chapter 4. Chapter 5 provides some conclusive remarks and plans for future releases.

2 Prototype in a Nutshell

2.1 Purpose, Scope and Functionality

This deliverable is part of work package WP4.2. The particular objectives of this work package are:

- to specify an executable event-driven process model triggered by real-time events,
- to develop methodologies for performing dynamic predictive process analysis at runtime,
- to provide techniques, featuring the ability to perform intensive simulation analysis under given hypothesis, and
- to implement the provided techniques in an intelligent security event-processing engine.

This deliverable describes a prototype resulting from the Tasks T4.2.4 " Predictive Security Analyser". The purpose of this prototype is to provide an advanced tool for the evaluation of security-related events and their interpretation with respect to:

1. the known control-flow of the processes involved, and
2. the required security properties.

With respect to (1), the PSA analyses deviations from the given process specification. These deviations can be the result of: (a) an evolution in the process specification, (b) problems with the measurement (e.g., lost events) or (c) anomalies caused by attacker interactions. Specific reactions to (1) have been described in [10].

With respect to (2), the PSA specifically enables dynamic predictive process analysis at runtime and detection of potential violations of the specified security properties. This is described in Section 4.5.

In summary, the PSA comprises:

event pre-processing

- tool support for import of events from an XML stream,
- tool support for import of events from the MASSIF database,
- tool support for event abstraction in order to filter data containing information not relevant to security processing,
- tool support to map events to the corresponding process instance,

process specification and adaptation

- tool support for the specification of a formal security-aware process model,
- tool support for the import of process specifications from process discovery tools,
- tool support for uncertainty management (e.g., semi-automatic adaptation of process specifications to measured behaviour),

close-future process behaviour

- implementation of techniques for the computation of close-future process behaviour for Event-driven Process Chain (EPC) specifications,
- implementation of techniques for the computation of close-future process behaviour for Product Net (PN) specifications and imported Petri Net Markup Language (PNML) specifications,

security requirements specification and evaluation

- tool support for the specification of the required security properties that the corresponding process should fulfil (in form of monitor automata),
- implementation of techniques for an on-the-fly check of security requirements with respect to *current process behaviour*,
- implementation of techniques for an on-the-fly check of security requirements with respect to *close-future process behaviour*,

situational awareness and alarm generation

- visualisation of current process states with respect to security requirements in the security monitors,
- implementation of alarm generation on detection of critical states.

2.2 List of Components and their actual Release Numbers

Main components:

- MASSIF PSA Event Adapter, v1.0
- PSA prototype, v3.0

2.3 Prototype Availability

The prototype is distributed as a gzipped tar archive which contains the PSA lisp runtime image, shared libraries used by the lisp system, shell scripts for starting PSA, and one jar file with the event adapter.

2.4 Build Procedure

To build PSA a license of the Lisp development environment Allegro Common Lisp (ACL) 9.0 for Linux is needed. PSA includes lisp compiler functionality but generation of runtime images is not possible without a valid license. Within the MASSIF project a runtime image for 64bit Linux systems will be available.

3 Prototype Deployment

3.1 Pre-requisites

Java 6 and the Graphical user interface specification and widget toolkit for building X Window applications (MOTIF) must be installed. A free version is part of most distributions or is available from www.openmotif.org. The MOTIF library libXm.so.3 must exist in standard library path or libXm.so.3 has to be in a path specified by $LD_LIBRARY_PATH. If libXm.so.4 is installed a link from libXm.so.3 to libXm.so.4 must exist. For motif installation on RedHat Linux see:
http://www.franz.com/support/documentation/9.0/doc/faq/

Firefox or Mozilla has to be installed for online help, because the tool uses the Mozilla remote control feature. The default browser is Firefox and can be changed in the *shvt>options>misc* menu. To achieve improved graph visualization graphviz [1] can be used to determine positions for graph drawing *shvt>options>Graph Visualization*.

3.2 Installation Procedure

To get the current version of the PSA please contact Jürgen Repp (juergen.repp@sit.fraunhofer.de).

Installation of PSA from tar archive:

```
cd [install-dir]
tar -zxvf psa-[version].tar.gz
```

3.3 How to Verify the Installation

Start PSA:

```
[install-dir]/psa/spsa
```

The PSA start window shown in Figure 3.1 will be displayed. The PSA tool selection menu on the right side in this Figure can be opened by clicking on PSA in the start window.

[1] http://www.graphviz.org/

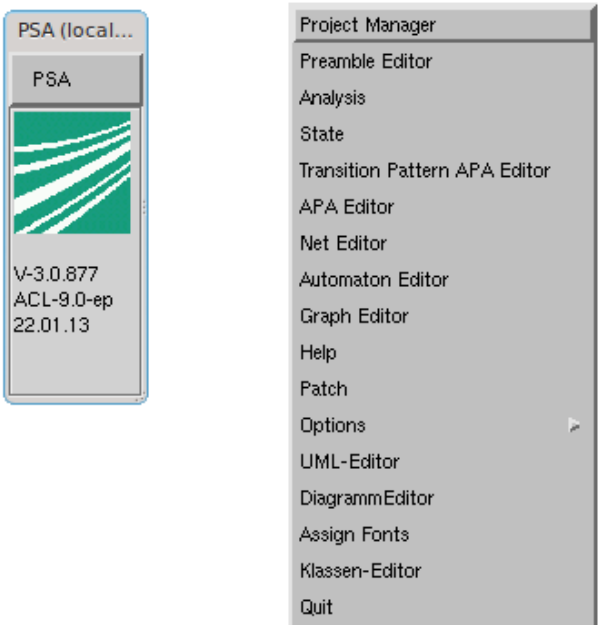

Figure 3.1: Starting PSA

3.4 Licensing

There are two PSA versions available:

- Standard runtime (without Lisp compiler)
- Dynamic runtime (with Lisp compiler - speed up factor of approx. 6 compared with the standard runtime version)

To use the dynamic runtime a valid dynamic runtime license is needed. During the MASSIF evaluation of PSA such licenses can be used restricted to the time of the PSA evaluation.
The code of PSA is proprietary and owned by Fraunhofer SIT. Access rights to the MASSIF consortium are provided for using it during the project execution.

3.5 Prototype Usage

PSA can be started by using the command `spsa` in the installation directory. The event adapter is started by executing:

```
java -jar Adapter.jar -plugA:XMLTag [event file] \\
     1000 1 -plugB:SHVT localhost 9999
```

Currently only the file interface is implemented for the event adapter. The interface to the MASSIF system will be implemented for the final version of PSA (D4.2.3). The usage of the prototype is described in detail in Section 4.5. Note that the event adapter can only be started after the PSA server process was

started. During the processing of the event stream the PSA model will be adapted. The changed Preamble Language (PL) source files will be loaded into memory. The user must explicitly save the buffers, if the changes shall be used in future simulations (Command *File>Save* in project manager).

4 Architecture Prototype Design

4.1 Prototype Context

As a part of the MASSIF SIEM, the PSA monitors workflow execution. Based on the monitoring results, PSA creates a model of the current system state and predicts future states of the system and checks possible violations of security requirements in this simulation. Details of the process modelling and monitoring are described in deliverables [12] and [10]. D4.2.2 [12] describes the transformation of process models to PSA models, while D4.1.3 [10] describes the handling of incomplete specifications. In the current version, treatment of processes running independently was improved. A process instance represents one specific instance of a process that is currently executing. Process instances share a common possible control flow given by the process specification but they contain specific runtime information related to that instance. The standard way of computing possible continuations would compute all possible interleavings of process instances and thus result in exponential state space explosion of the behaviour graph. To avoid this problem the approach taken in PSA is to run simulation on an abstraction level which is identical to all processes instances. So we can reuse the graph of predicted possible behaviour for every instance and thus reduce complexity significantly. However, to be able to identify the process instance at the user interface we have to create and maintain relations from the process instances in the system to the graph of predicted behaviour.

4.2 Prototype Component Structure

PSA components shown in Figure 4.1 are extended by the following features:

- Semi automatic adaptation of process models to workflows of the "real" system during runtime (Process Modeller and PSA Core).
- Handling of events that are not described correctly by the Event Type Interface (20 in Figure 4.1).
- Estimation of current system states in the case of incomplete monitoring data (PSA Core).
- Generation of alarms (independent of the monitoring process described in [12]) if problems can't be solved (PSA Core, Security Alarm Interface).
- User controlled change of the current state of simulation (Process Modeller and PSA Core).
- Process identification together with process oriented security monitoring.

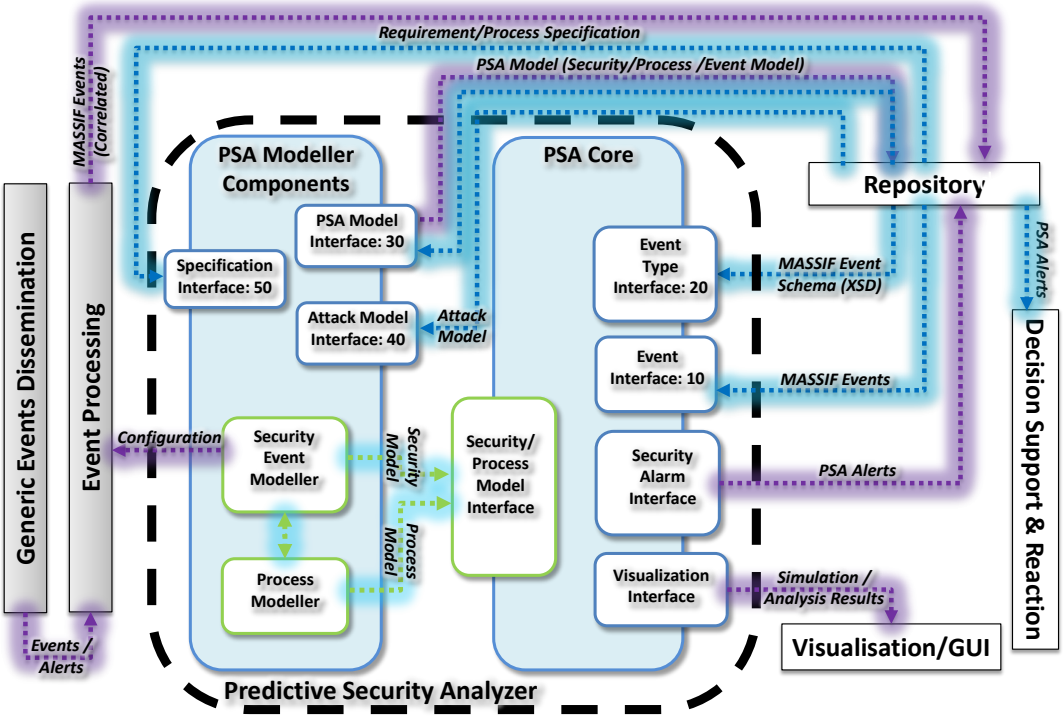

Figure 4.1: PSA components

4.3 PSA Data Flow

Figure 4.2 depicts PSA data flows during the model development and the simulation process. The dashed arrows denote data flows during runtime, while the solid arrows denote data flows during the development process. The quality of the monitoring, of the simulation, and of the prediction process relies on the precision of the operational process model. Normally detailed modelling of the behaviour of several similar processes in one PSA model causes a state space explosion. The presented approach tries to decrease this problem by generalizing the process model and elimination of process specific information in the mapping defined for the event stream. The information required to identify a process will be included in the Mapping Model and Process Identification Model to compile the functions for runtime mapping and process identification which will be applied to events received from the event stream. If such a generalization is not possible, the already presented approach for modelling without process identification still can be used.

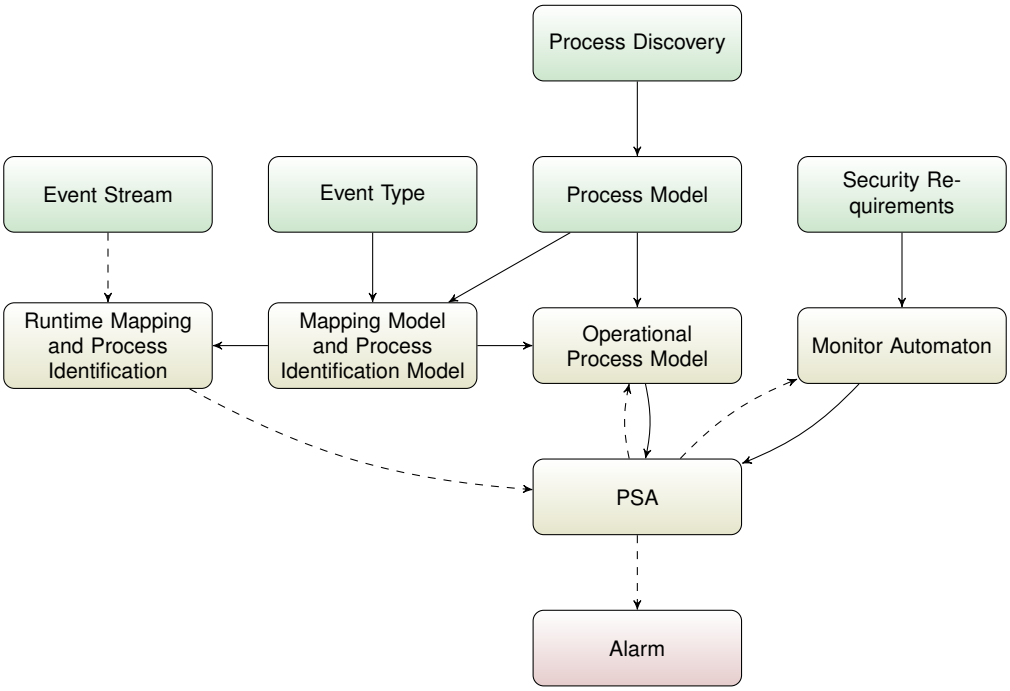

Figure 4.2: PSA data flow

4.4 Graphical User Interface

The graphical user interface for defining abstractions of the system events to derive corresponding PL definitions was described in [12]. The example XSD file used in section 4.4 in this deliverable is extended to enable handling of several process instances by one shared reachability graph. Thus several users can be monitored in parallel.

4.4.1 Specifying Process Instances

To identify a certain process instance a process id must be extracted from a received event. Listing 4.1 shows the XSD definition, which describes the format of the example events. `EventType` is the Extensible Markup Language (XML) type used for a single event. The attribute `id` is defined for this type to enable identification of a certain process. This new attribute can be mapped to PL values using the mechanisms described in section 4.3.1 of [12]. In our example we can use the integer value without any mapping. The value of the attribute `id` will not be added to the PL data assigned to an interpretation variable or state component for the continuation of the simulation. This value will be used only to identify the current state in the computed system behaviour (reachability graph) for the continuation of the prediction from this state. To take advantage of this new feature an appropriate abstraction level which omits all specific data of process instances, which causes the described state space explosion has to found. The corresponding elements and attributes of the received events should not be mapped do PL data used in the simulation.

To take advantage of this new feature an appropriate abstraction level for other values used in the

simulation has to be found. E.g. it makes no sense to use this feature if other values directly correspond to the defined process id. If more than one XML element is used as a process id, they will be combined to a list in the order of their occurrence in the XML event. Listing 4.2 shows a part of a possible event stream for this XSD definition with two process instances.

Listing 4.1: XSD schema for example events

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://namespaces.atos.com/AtosEvent" elementFormDefault="qualified">
<xsd:simpleType name="CheckLogin">
<xsd:restriction base="xsd:token">
<xsd:enumeration value="User_prompted_for_login_details" />
<xsd:enumeration value="Login_Data_Delivered" />
<xsd:enumeration value="Authenticate_user" />
<xsd:enumeration value="Login_with_admin_privileges_Attempted" />
<xsd:enumeration value="User_Authenticated" />
<xsd:enumeration value="Check_admin_connection_time" />
<xsd:enumeration value="Logon_Successful" />
<xsd:enumeration value="Admin_at_unusual_time" />
<xsd:enumeration value="Admin_behaviour_ok" />
<xsd:enumeration value="Admin_at_unusual_location" />
<xsd:enumeration value="Admin_behaviour_ok" />
<xsd:enumeration value="Event_logged_illegal_user" />
<xsd:enumeration value="Session_disconnected" />
</xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="EventType">
  <xsd:sequence>
    <xsd:element name="Eventname" type="CheckLogin" />
  </xsd:sequence>
  <xsd:attribute name="id"
                 type="xsd:integer" />
</xsd:complexType>

<xsd:element name="Event" type="EventType" />
</xsd:schema>
```

Listing 4.2: Example events

```
<Event id="1"><Eventname>User_prompted_for_login_details</Eventname></Event>
<Event id="1"><Eventname>Login_Data_Delivered</Eventname></Event>
<Event id="2"><Eventname>User_prompted_for_login_details</Eventname></Event>
<Event id="1"><Eventname>Login_with_admin_privileges_Attempted</Eventname></Event>
<Event id="1"><Eventname>Admin_at_unusual_time</Eventname></Event>
<Event id="2"><Eventname>Login_with_admin_privileges_Attempted</Eventname></Event>
```

The next section will describe step by step how to handle the corresponding process definition in PSA. The PL implementation presented in [12] section 7.2 will be used for this purpose, since this model has been already introduced and can handle the events described by the changed XSD definition. The corre-

sponding project file will be available in the demo directory (Unusual_Behaviour/Unusual_Behaviour.prj).

4.5 Running an Example

Start the PSA by using the command *spsa* in the installation directory.

- Start project manager

  To load an existing project (a set of files comprising process specifications, event mappings, ...), the Project Manager in the PSA main menu has to be started. The Project Manager window contains two parts, one for the directory tree, once an existing project is loaded, and one that shows which action is being executed, and successful execution and error messages, respectively. When starting the Project Manager both window parts are empty.

- Read the project file `Unusual_Behaviour.prj` from the sub directory `Unusual_Behaviour` of the demo directory. You will see the project manager window shown in Figure 4.3. The presented XSD file is already included in this project and assigned to a preamble file. Also the mapping from XML data to PL data is defined.

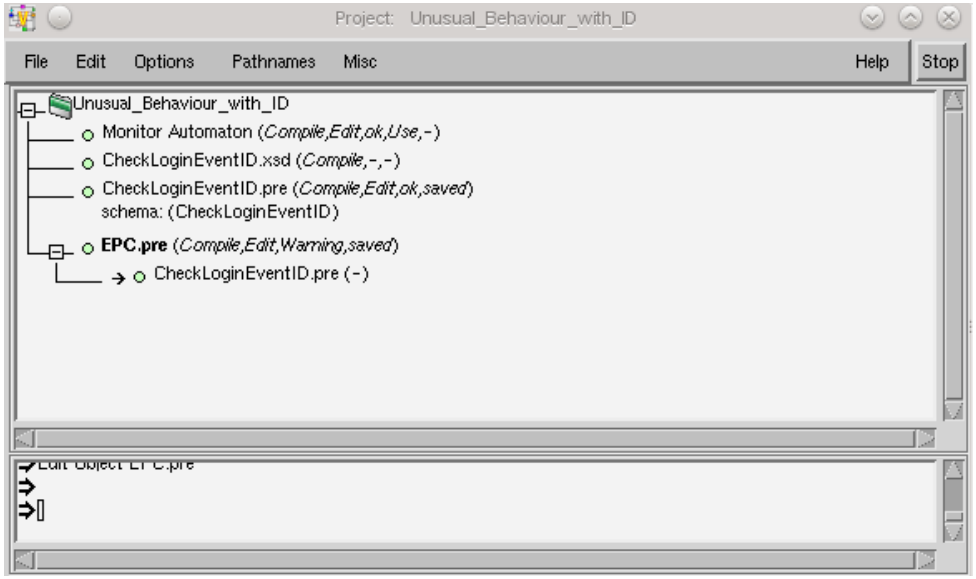

Figure 4.3: PSA project manager

- To check the mapping execute the command *Define Schema Mapping* in context menu of `CheckLoginEventID.pre`. The schema mapping shown in Figure 4.4 will be displayed. The red icon with the text `id` and the text "Use Number as an ID" indicates that the attribute `id` will be used as a process id. This option can be activated or deactivated by using the commands *Use as an ID* or *Do not use as an ID* in the context menu of an XML element.

- Preparing PSA Simulation

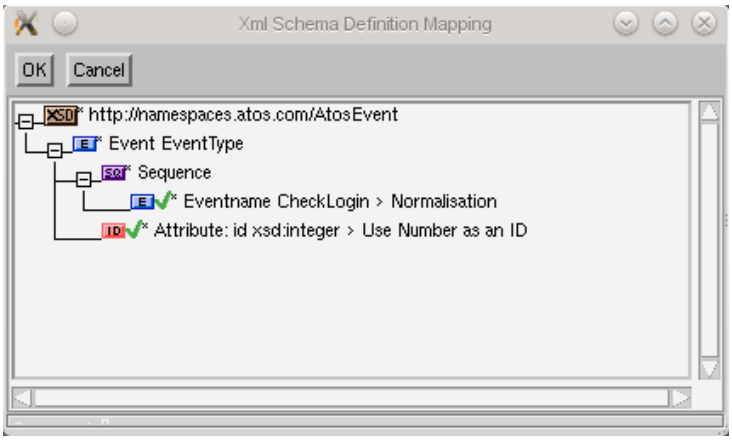

Figure 4.4: XSD definition mapping

The simulation window can be opened by executing the command *Analysis* in the context menu of the root node. The preferences have to be adjusted as shown in Figure 4.5 (Menu bar command *Options* > *Simulation*). The type has to be set to *MASSIF Event Bus Simulation* and *ExternalEvents* must be set to *Yes*. The default values for the interpretation variable and the state component for storing external events in the PSA model can be defined. The default values will fit to the presented example model.

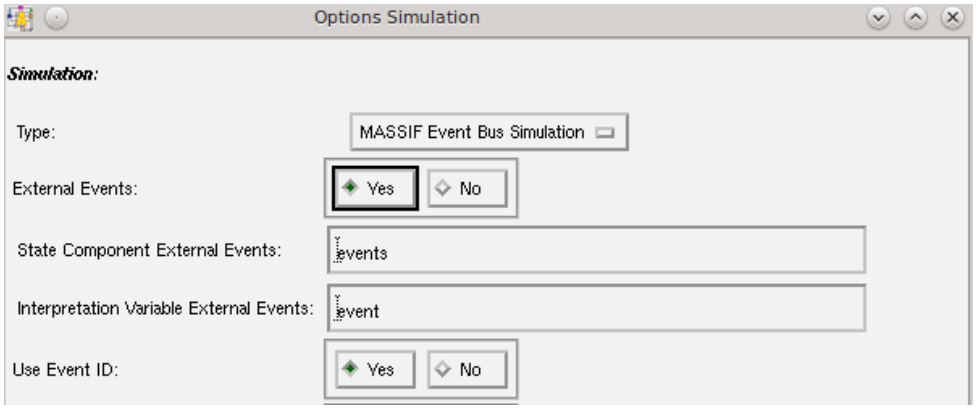

Figure 4.5: Simulation options

- Monitor Automaton

  For demonstration purposes, the simple monitor automaton shown in Figure 4.6 has been added to the project. The automaton is compiled automatically after executing the analysis command in the project manager and can be edited and compiled from the monitor automaton editor (Command *Edit* > *Monitor Automaton*). This automaton performs a state change only when the event `Admin_at_unusual_location` does occur.

- Now PSA simulation can be started by entering the command *PSA Simulation* at the prompt of the command pane of the simulation window. To initiate the sending of events the user should start the

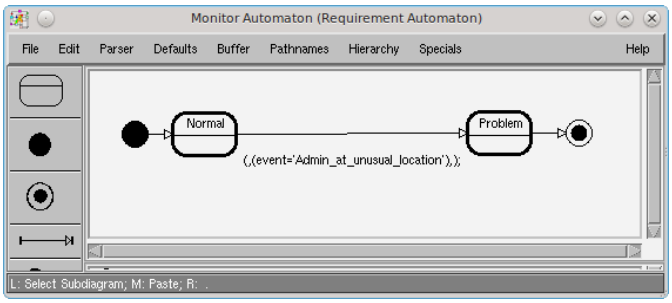

Figure 4.6: Monitor automaton

Event Interface at the command line after changing the working directory to the PSA installation directory.

```
java -jar Adapter.jar \\
   -plugA:XMLTag demo/Unusual_Behavior/login_events_with_id.xml \\
   1000 1 -plugB:SHVT localhost 9999
```

The adapter will send test events from the file login_events_with_id.xml (see Listing 4.2).

- After the adapter is started the user can check the state of the simulation by executing the *Statistics* command in the command pane on the right side of the simulation window. Statistical information concerning the computation of the reachability graph and to the state of the Monitor automaton will be displayed as shown in Figure 4.7.

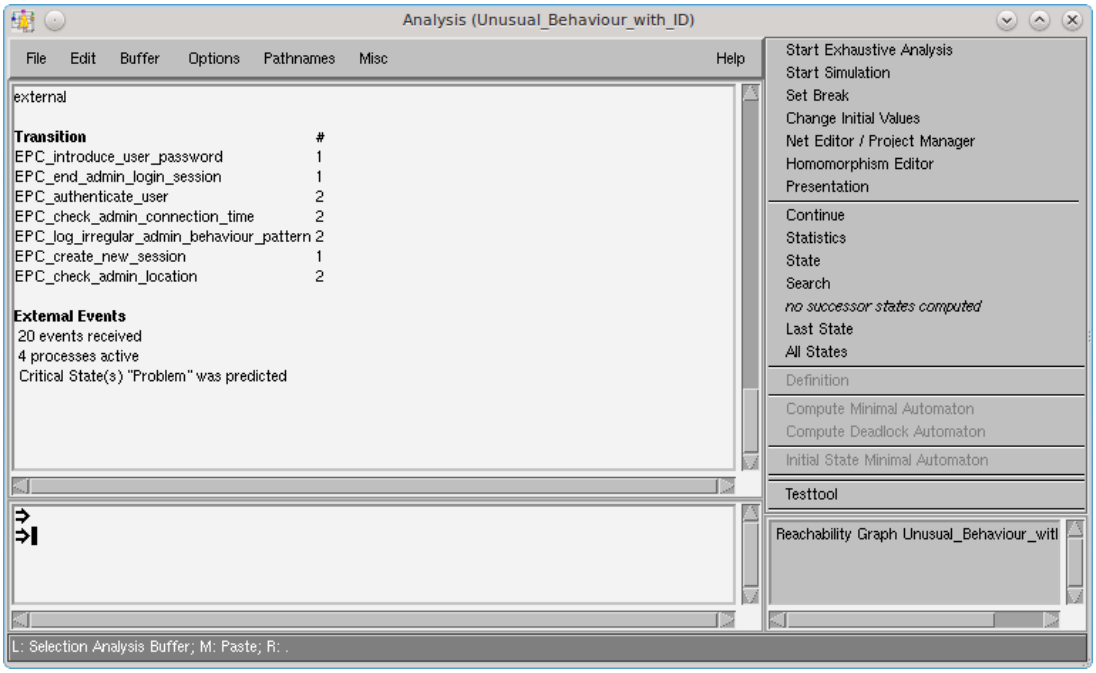

Figure 4.7: Simulation statistics

PSA provides capabilities to investigate the current state of the simulation by several commands available in the context menus of displayed object in the analysis window and in the monitor automaton editor. In contrast to running simulation without process identification the user will be asked to select an appropriate process id, if process related information (e.g. monitor automaton states) assigned to entities of the reachability graph will be selected. The statistics shows that the monitor automaton state `Problem` was predicted. This transition will produce an alarm in Intrusion Detection Message Exchange Format (IDMEF) format sent to the Event Interface (see 4.3). The id of the process responsible for the alarm is included as additional data in the alarm, despite the fact that this id does not occur in any state or interpretation variable binding of the system behaviour (reachability graph).

Listing 4.3: IDMEF alarm

```
<IDMEF-Message>
  <Analyzer analyzerid="0" name="PSA" manufacturer="http://www.sit.fraunhofer.de"
    model="PSA" version="3.0.877"
    class="Concentrator" ostype="Linux" osversion="3.0.0-29-generic">
    <Node category="unknown"><name>pc-repp2.sit.fraunhofer.de</name></Node>
    <Process><name>psa</name><pid>29415</pid><path>/local/acl90/clim</path></Process>
  </Analyzer>
  <CreateTime ntpstamp="0xd4a8daff.0x00000000">2013-01-22T10:31:43+01:00</CreateTime>
  <Classification text="Critical State Predicted"/>
  <AdditionalData type="xsd:string" meaning="State">Problem</AdditionalData>
  <AdditionalData type="xsd:string" meaning="ProcessID">2</AdditionalData>
</IDMEF-Message>
```

5 Prototype Implementation

The main part of the prototype is implemented in Common Lisp using the dynamic object-oriented development system ACL 9.0. A lisp compiler is included in the runtime version of the prototype and so the recompilation of changed PSA models during runtime is supported. The graphical user interface of the prototype is implemented in Common Lisp Interface Manager (CLIM) which is currently based on MOTIF.

The event interface to the MASSIF system (see Figure 4.1) is implemented in Java to alleviate the integration process. The adapter requires Java 6 and runs with Sun Java Development Kit (JDK) and also open JDK.

A socket interface is used for communication between the adapter module and the PSA core. Thus the distribution of the components to different hosts is possible.

6 Conclusions

6.1 Self-evaluation and Assessment

The prototype has been tested without the MASSIF environment using a mockup implementation of the adapter of the Event Interface. An example process description from Olympic Games scenario has been used to create test data. The example was extended by process identification features presented in this deliverable.

PNML import was tested with several examples created by process discovery tools.

6.2 Roadmap

Access to the *repository storage layer* [6] via the *repository service layer* will be implemented (cf. task 5.3.4 “MASSIF integration”).

In cooperation with MASSIF partners PSA process models have to be developed. Based on these models, runtime monitoring and prediction has to be tested. Improvements to the user interface will be developed according to user needs that will be identified during the tool adaptation in work package 2.2.

Bibliography

[1] Luigi Coppolino, Michael Jäger, Nicolai Kuntze, and Roland Rieke. A Trusted Information Agent for Security Information and Event Management. In *ICONS 2012, The Seventh International Conference on Systems, February 29 - March 5, 2012 - Saint Gilles, Reunion Island*, pages 6–12. IARIA, 2012.

[2] Jörn Eichler and Roland Rieke. Model-based Situational Security Analysis. In *Workshop on Models@run.time*, volume 794, pages 25–36. CEUR, 2011.

[3] Andrew Hutchison and Roland Rieke. Management of Security Information and Events in Future Internet. In *2011 Workshop on Cyber Security and Global Affairs, Budapest*. 2011.

[4] Andrew Hutchison and Roland Rieke. Measuring Progress in Cyber-Security: An Open Architecture for Security Measurement Consolidation. In *2012 Workshop on Cyber Security and Global Affairs and Global Security Forum, Barcelona*. 2012.

[5] Richard Kissel. Glossary of key information security terms. NIST Interagency Reports NIST IR 7298 Revision 1, National Institute of Standards and Technology, February 2011.

[6] Igor Kotenko, Evgenia Novikova, Olga Polubelova, and Valerio Formicola. *D5.3.4 – Unified repository and visualization tools*. FP7-257475 MASSIF European project, January 2013.

[7] Elsa Prieto, Rodrigo Diaz, Luigi Romano, Roland Rieke, and Mohammed Achemlal. Massif: A promising solution to enhance olympic games it security. In Christos K. Georgiadis, Hamid Jahankhani, Elias Pimenidis, Rabih Bashroush, Ameer Al-Nemrat, Ozgur Akan, Paolo Bellavista, Jiannong Cao, Falko Dressler, Domenico Ferrari, Mario Gerla, Hisashi Kobayashi, Sergio Palazzo, Sartaj Sahni, Xuemin (Sherman) Shen, Mircea Stan, Jia Xiaohua, Albert Zomaya, and Geoffrey Coulson, editors, *Global Security, Safety and Sustainability & e-Democracy*, volume 99 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 139–147. Springer Berlin Heidelberg, 2012.

[8] MASSIF project consortium. *D2.1.1 - Scenario requirements*. FP7-257475 MASSIF European project, March 2011.

[9] Jürgen Repp and Roland Rieke. *D4.2.1 – Formal Specification of Security Properties*. FP7-257475 MASSIF European project, September 2011.

[10] Jürgen Repp, Maria Zhdanova, and Roland Rieke. *D4.1.3 – Methods and tools for reasoning about security with uncertain knowledge*. FP7-257475 MASSIF European project, September 2012.

[11] Roland Rieke, Luigi Coppolino, Andrew Hutchison, Elsa Prieto, and Chrystel Gaber. Security and reliability requirements for advanced security event management. In Igor Kotenko and Victor Skormin, editors, *Computer Network Security*, volume 7531 of *Lecture Notes in Computer Science*, pages 171–180. Springer Berlin Heidelberg, 2012.

[12] Roland Rieke, Jürgen Repp, and Maria Zhdanova. *D4.2.2 – Process Model and Dynamic Simulation and Analysis Modelling Framework*. FP7-257475 MASSIF European project, September 2012.

[13] Roland Rieke, Julian Schütte, and Andrew Hutchison. Architecting a security strategy measurement and management system. In *Proceedings of the Workshop on Model-Driven Security*, MDsec '12, pages 2:1–2:6, New York, NY, USA, 2012. ACM.

[14] Julian Schütte, Roland Rieke, and Timo Winkelvos. Model-based security event management. In Igor Kotenko and Victor Skormin, editors, *Computer Network Security*, volume 7531 of *Lecture Notes in Computer Science*, pages 181–190. Springer Berlin Heidelberg, 2012.

[15] W. M. P. van der Aalst, B. F. van Dongen, C. Günther, A. Rozinat, H. M. W. Verbeek, and A. J. M. M. Weijters. Prom: The process mining toolkit. In *BPM 2009 Demonstration Track*, volume 489, pages 1–4. CEUR, 2009.

