A Secure Public Sector Workflow Management System

Maarten Rits, Andreas Schaad
SAP Research

Stefano Crosta, Jean-Christophe Pazzaglia
Eurecom
eJustice – Project Details

Project Management
- ONE North East (UK)

Project End Users
- Bundesverfassungsgericht (DE)
- Generalstaatsanwalt Zweibrücken (DE)
- Bundeskanzleramt der Rep. Österreich (AT)

Consortium Members
- SAP Research (FR)
- Unisys (BE)
- Thales Identification (FR)
- ZN Vision Technologies AG (DE)
- DFKI (DE)
- Institut Eurécom (FR)
- Infocamere (IT)
- Max-Planck-Institut für Informatik (DE)
- Institut für Rechtsinformatik (Universität des Saarland, DE)

Project Plan
- Duration: 24 Months
- Start: March 2004

eJustice Team
- Dr. Andreas Schaad, PL
- Cedric Hebert
- Maarten Rits
- Pascal Spadone
SAP Research focuses on workflow security

- Roles and rights management in workflow systems
- Security of cross-border workflow systems
- Workflow traceability, integrity and non-repudiation
- Judicial processes modeling and translation to workflows
eJustice Case Studies

- eJustice produced a set of case studies that show some interesting properties:
  - No central management authority
  - Objects to be protected are records

- These four case studies selected include:
  - The eRecht system & process
  - Judicial assistance in criminal matters
    - aka Arrest Warrant
    - aka Fiche Belge
    - aka Rogatory Letter
  - European orders for payment procedure
  - Service of documents in civil procedures
Tools & Technology used ...

- Maestro
- Nehemiah
- EureCA
Maestro – A workflow modeling language
Nehemiah – The execution engine
- Attribute Certificate in XML
- Self-contained (full chain of issuer and subject)
- Short-time certificates
- Extensible (Plugin Architecture)
- Flexible attribute structure:
  - Name
  - Value
  - Resource
  - Policy
  - Delegation
  - ...
- Customisable delegation scheme
The Border Control Scenario:

- Overview Case Study
- Security Analysis:
  - Consistent access control policies
  - Temporal provisioning of access rights
  - Enforcing access control on workflow tasks
During a border control, a police officer checks the passport of passengers.

If the officer has reasons to believe that the person is suspicious, he will submit an enquiry to the Schengen Information System (SIS).

If there is an alert, the alert-originating-country has to be informed. Then there will be:

- either an arrest by the border control officer,
- no arrest,
- or a European Arrest Warrant may be issued by a national judicial authority to require the arrest and return of the person.

If the border control officer arrests the individual, a further investigation has to be performed.

During this investigation the legal authorities in the current country can obtain more information from the originating country by using the rogatory letter mechanism.
We will present an analysis of the different access control requirements that we identified in the scenario:

- i) deriving consistent access control policies for workflow tasks
- ii) the temporal provisioning of access rights with certificates
- iii) enforcing access control on workflow tasks and associated data, with a focus on inter-organizational workflows
i) deriving consistent access control policies for workflow tasks

- Resource management (who *should* perform an activity) vs. Access Control (who is *allowed* to perform an activity)

- Multi-layered applications

- Needed privileges for higher-layer services dpt. lower-layers

- Authorization exceptions ➔ uncontrolled rights assignment

- Over-privileged systems
eJustice Prototype Architecture

- JAAS
- XACML
- J2EE
- Stored procedures

© SAP AG 2005, ACSAC2005 / Maarten Rits / 13
Excursus: Access Control Specification

Sample policy:

“the captain has the right to display SIS”

J2EE

```xml
<method-permission>
  <role-name>captain</role-name>
  <method>
    <ejb-name>BusinessEJB</ejb-name>
    <method-intf>Remote</method-intf>
    <method-name>displaySIS</method-name>
  </method>
</method-permission>
```

JAAS

```java
grant Principal reac.jaas.ac.REACPrincipal "captain" {
  permission reac.jaas.ac.REACPermission "displaySIS";
};
```

XACML

```xml
<policy>
  <target>
    <subject>
      <subjectmatch>
        <attribute="role" value="captain" match="string-equal"/>
      </subjectmatch>
    </subject>
    <action>
      <actionmatch>
        <attribute="method-name" value="displaySIS" match="string-equal"/>
      </actionmatch>
    </action>
  </target>
  <anyResource/>
  <rule effect="permit"/>
</policy>
```
1. Target the access controlling methods for every AC type

2. Run the program, execute one workflow step with WhiteBox test data

3. While running, XacT checks every method call against every target

4. If a match is detected, it extracts all roles allowed to perform this running method, using the appropriate policy-monitoring component
MonitoringAspect with AspectJ

```java
public aspect MonitoringAspect {
    private pointcut allMethods() : (call(* *(..)) || call(*.new(..))) && within(ejustice..*) && ! within(test.*);

    private pointcut mainMethod() : execution(static void main(String[])) && ! within(ejustice..*);

    before() : allMethods() {
        StackTraceElement[] trace = (new Throwable()).getStackTrace();
        Downcall dc = new Downcall(thisJoinPoint, trace);
        for (int i = 0; i < getTargets().size(); i++) {
            if (getTargets().get(i).getTarget().matchesDowncall(dc)) {
                System.out.println("-XacT- match found !");
                PolicyMonitor pm = PolicyMonitorFactory.getPolicyMonitor(getTargets().get(i).getAccessControlType());
                Policy pol = pm.inspectPolicy(dc);
                getReportGenerator().addDowncall(dc, pol); calculateAllowedRoles(pol.getAllowedRoles());
            }
        }
    }

    after() : mainMethod() {
        if (!finished)
            getReportGenerator().finishReport(allowedRoles);
        finished = true;
        System.exit(0);
    }

    after() throwing (Exception e): mainMethod() { // }
}
```
Excursus: Demo Result

Sample report:

```
<report
date_generated="7 Sep 2004 12:20:10 GMT">
  <downcall>
    <targetobject>
      <class fqname="reac.j2ee.beans.business.Business"/>
    </targetobject>
    <targetmethod fqname="reac.j2ee.beans.business.Business.doAction" scope="public">
      <returntype>
        <class fqname="java.lang.String"/>
      </returntype>
      <argument rank="0">
        <class fqname="int"/>
      </argument>
    </targetmethod>
    <stacktraceelement rank="0" filename="BusinessClient.java"
      linenumber="69" methodname="doAction" isnativemethod="FALSE">
      <class fqname="reac.j2ee.client.BusinessClient"/>
    </stacktraceelement>
    ...
  </downcall>
  <policy>
    <allowedroles>
      <role name="judicial authority"/>
      <role name="minister"/>
      <role name="officer"/>
      <role name="admin"/>
    </allowedroles>
  </policy>
  <downcall>
    ...
  </downcall>
  <allowedroles>
    <role name="officer"/>
    <role name="admin"/>
  </allowedroles>
</report>
```
ii) the temporal provisioning of access rights with certificates

- Different approaches may be envisaged to control the invocation of external applications:
  - engine limits itself to distribute tasks to users
  - more tightly coupled collaboration between the workflow engine and the applications
Tightly coupled collaboration

Requirements

- Interrogation of the system should be restricted to a small number of persons.
- Interrogation should be performed by a qualified workflow engine.
- Interrogation should be possible only when certain conditions are respected (time, presence of suspect, etc).
- The Border Control Department, but also the SIS organization, should be able to trace precisely the requests, in order to track possible malicious behaviors.
iii) Task Execution

- We introduce the new term of “task relevant data” as a subset of workflow relevant data, which is necessary for the execution of a certain task.

- We introduce the concept of the “Workflow Engine Guardian”
  - The guardian extracts the task relevant data and holds it back. Afterwards, it notifies all relevant users about the upcoming task and determines the real executor for the task.
  - In order to retrieve the task relevant data, a user has to provide an attribute certificate, which certifies that he is the executor of the task.
  - Solution using delegation in case the user is not allowed to access these task relevant data because of security policy (dynamic).
  - The guardian changes the task status to running and to completed.
iii) Task Execution

[Diagram of task execution process]

- Start
  - New task notification
  - Change task status to running
  - Change task status to complete
  - Executed by the workflow engine

- End
  - Executed by the guardian

- Extract TRC
  - Authenticate and authorize user
  - Notify about start of execution
  - Generate "executor certificate"
  - Forward notification

- Forward certificate
  - Verify user certificate
  - Send "executor certificate"

- Receive certificate
  - Verify "executor certificate"
  - Deliver TRC
  - Send certificate to prox

- Determine relevant users
  - Notify users

- Accept task (Y/N)
  - Send certificate to prox

- Execute task
  - Ask for TRE
  - Receive TRE
  - Execute task

- Executed by tasks: manage
- Executed by users
Questions

???