University of Milan
Department of Information Technology

The first International Conference on Open Source Systems

Using Open Source Middleware for Securing e-Gov Applications

Authors: Claudio Agostino Ardagna, Ernesto Damiani, Fulvio Frati and Martin Montel

OSS 2005 Genova, 11th - 15th July 2005

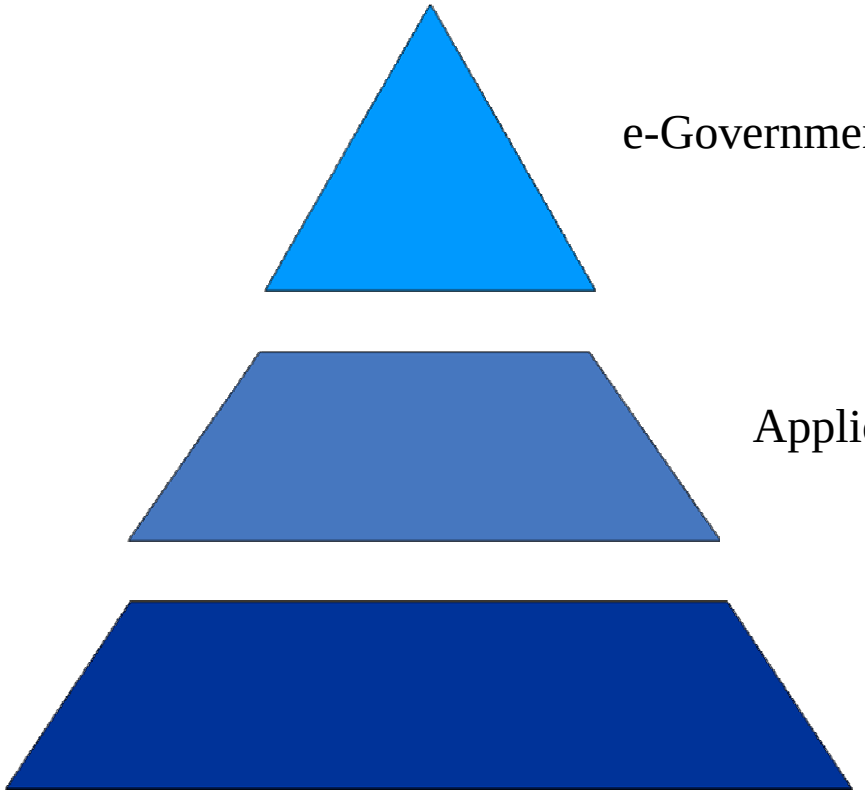
Outline

- e-Gov: state of the art
- Open Source e-Gov Environment
- Implementing JBoss security environment
- Advanced security implementation

e-Gov state of the art - 1

- e-Services fundamental role in modern economy
- Increasing interest in e-Gov services
- Definition of e-Gov services' layered structure

e-Gov state of the art - 2



e-Gov state of the art - 3

- Problems
  - budget limitations
  - hybrid implementations rely on proprietary middleware horizontal functionalities
- Solution
  - provide a complete *Open Source e-Gov Environment*

Open Source e-Gov environment

- Open Source service components
- Open Source Application Servers
  - JBoss
  - JOnAS
- Open Source Operating System
  - Linux

Open Source Application Server - Characteristics

- Provide an environment over which applications are deployed
- Avoid proprietary lock-in at middleware layer
- Developers focus on business activities

Open Source Application Server - Functionalities

- Web Server support
- Transaction Manager
- Security Manager
- Cache Manager
- Persistence Manager
- Clustering Manager

JBoss/JOnAS

- Open Source Application Servers
- Fully J2EE specifications compliant
- EJB/Web Container
- Web Services support through AXIS integration

- Access Control Management
  - JBossSX: component that handles security (role-based)
  - JAAS (Java Authentication and Authorization Service): standard modules for file, DB, LDAP - based security information

Application Server Security – JOnAS

- Based on security environments (*realms*)
  - Memory realm
  - Datasource realm
  - LDAP realm

Implementing JBoss security environment

- Security Domain Configuration (*jboss-web.xml*, *jboss.xml*)
- Authentication process configuration (*login-config.xml*)
- Authorization process configuration (*ejb-jar.xml*)

Implementing JBoss security environment

- Security Domain Configuration (*jboss-web.xml*, *jboss.xml*)
- Authentication process configuration (*login-config.xml*)
- Authorization process configuration (*ejb-jar.xml*)

```
<security-domain>
  java:/jaas/MySecurity
</security-domain>
```

Implementing JBoss security environment

- Security Domain Configuration (*jboss-web.xml*, *jboss.xml*)
- Authentication process configuration (*login-config.xml*)
- Authorization process configuration (*ejb-jar.xml*)

```
<application-policy name="MySecurity">
 <authentication>
  <login-module code= "org.jboss.security.
   auth.spi.DatabaseServerLoginModule"
   flag="required">
    <module-option name="dsJndiName">
      java:/UserDS
    </module-option>
    <module-option name="principalsQuery">
      SELECT Password FROM Operators
      WHERE OperatorID=?
    </module-option>
    <module-option name="rolesQuery">
      SELECT Role, 'Roles' FROM
      OperatorRoles WHERE OperatorID=?
    </module-option>
  </login-module>
 </authentication>
</application-policy>
```

Implementing JBoss security environment

- Security Domain Configuration (*jboss-web.xml*, *jboss.xml*)
- Authentication process configuration (*login-config.xml*)
- Authorization process configuration (*ejb-jar.xml*)

```
<ejb-jar>
 <assembly-descriptor>
  <security-role>
   <role-name>Standard</role-name>
  </security-role>
  <security-role>
   <role-name>Admin</role-name>
  </security-role>
  <method-permission>
   <role-name>Admin</role-name>
   <method>
    <ejb-name>Operator</ejb-name>
    <method-name>*</method-name>
   </method>
  </method-permission>
  <method-permission>
   <role-name>Standard</role-name>
    <method>
     <ejb-name>OperatorSession</ejb-name>
     <method-name>getOperator</method-name>
    </method>
   </method-permission>
 </assembly-descriptor>
</ejb-jar>
```

Advanced Security Implementation

JBoss and Single Sign-On

Single Sign-On (SSO)

- Provide centralized authentication to a single server
- Advantages of SSO solution
  - Reduction of authentication time to secondary domains
  - Security improvement
  - Reduction of user profiles management costs and time
  - Usability improvement: single login interface

Central Authentication Service (CAS)

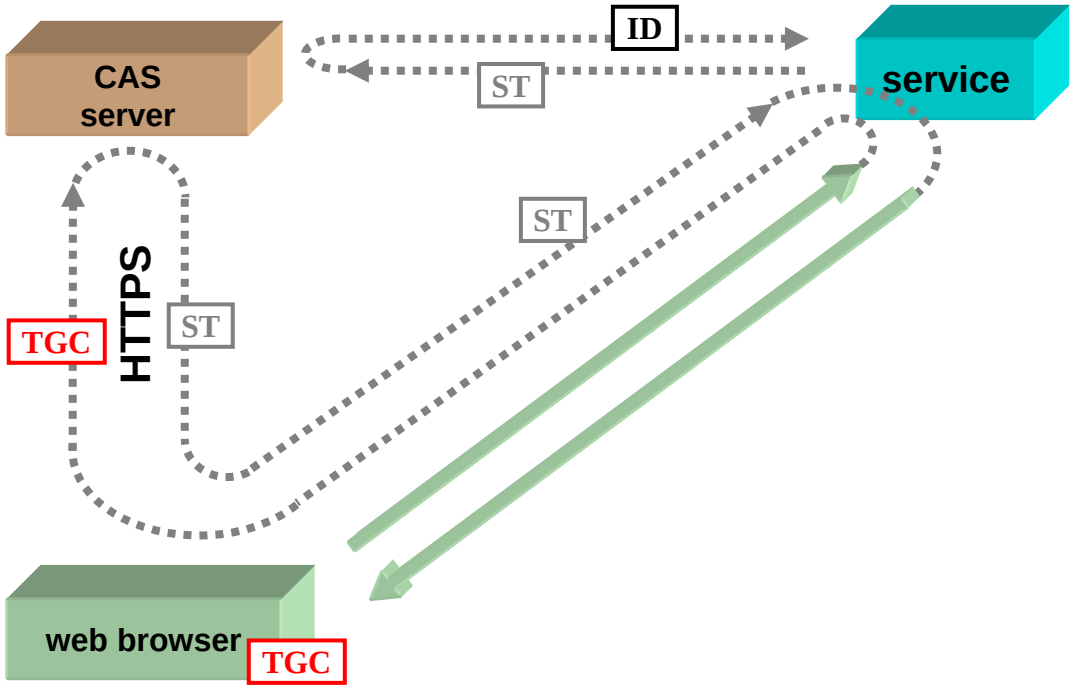
- SSO Open Source framework
- Developed by Yale University
- Run over any servlet engine
- Secure, flexible, reliable

Central Authentication Service (CAS) - Security

- Passwords used for authentication process only through encrypted channels
- Transparent re-authentication
- Applications know User Identities through opaque cookies
  - User Identities shared only between CAS server and services

CAS Flow

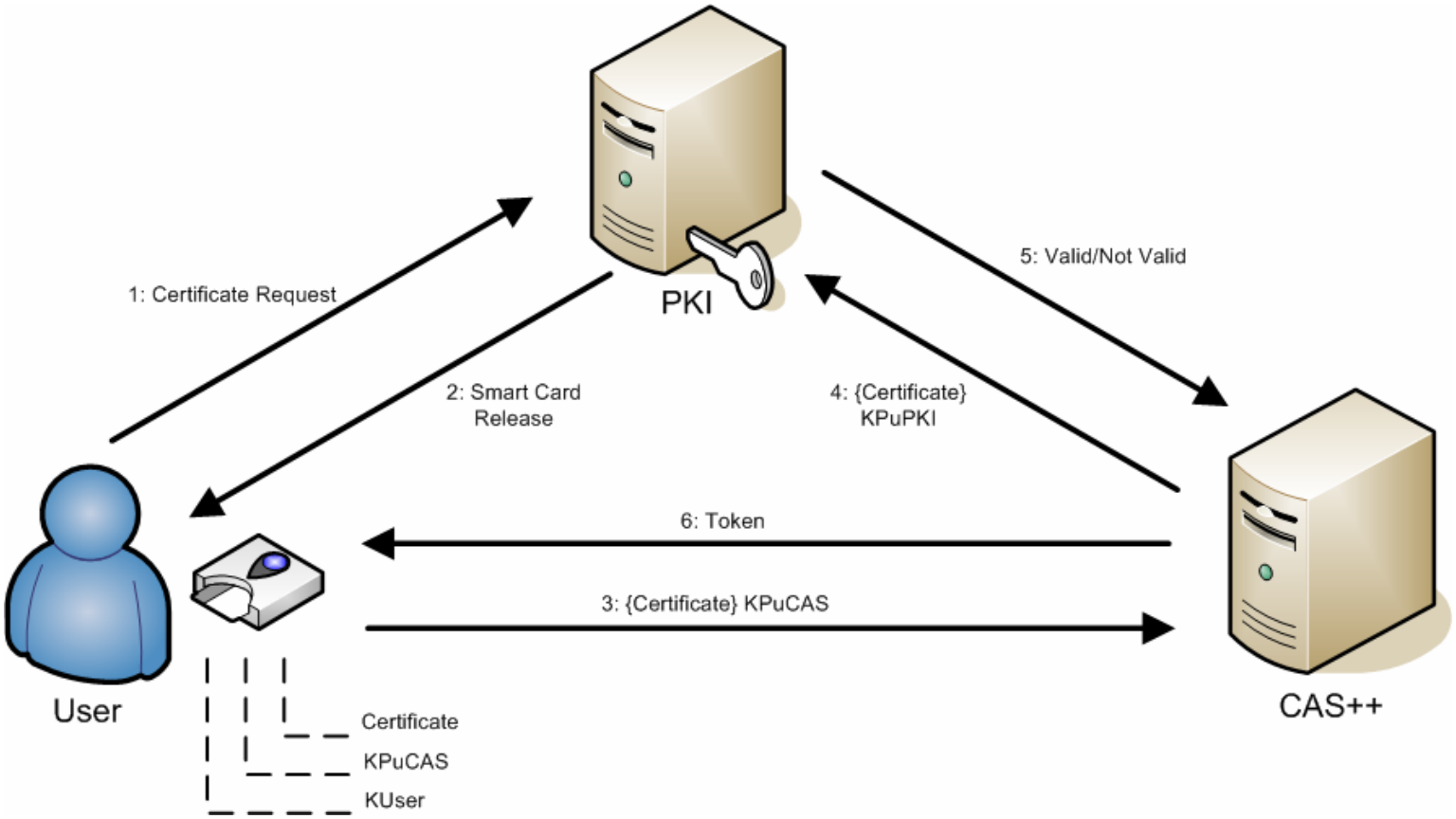
- TGC (Ticket Grant Cookie):
  - Identify the user
  - CAS usable only
- ST (Service Ticket)
  - Limited Time Validity
  - Authorize web browser access to service



Advanced Security Implementation (CAS++)

- SSO based on certificates
  - Improvement to CAS architecture (CAS++)
- Integration with Public Key Infrastructure (PKI)
- Integration with strong authentication mechanisms based on
  - smart card, and
  - fingerprint reader

Advanced Security Implementation - Flow



Thanks to

- Ernesto Damiani
- Fulvio Frati
- Martin Montel
- Software Engineering and Advanced Architectures Lab. (University of Milan)

Reference

- CAS

http://tp.its.yale.edu/tiki/tiki-index.php?page=CentralAuthenticationService

- Italian Project PEOPLE

http://www.progettopeople.it

Questions???