



Tehnologii Java

Curs -

Cristian Frăsinaru

`acf@infoiasi.ro`

Facultatea de Informatică

Universitatea "Al. I. Cuza" Iași





Securitatea aplicatiilor



Cuprins

- Aspecte legate de securitate
- Securizarea aplicațiilor desktop
- Securizarea nivelului Web
- Securizarea nivelului de logică

Introducere

Securitatea aplicatiilor

Securitate software = Protecția informațiilor împotriva accesului neautorizat

Perspectiva desktop

- *Ce fel de cod* este executat de client ?
- *Ce fel de operații* dorește să execute ?

Perspectiva Web

- *Cine* accesează sistemul ?
- *Ce fel de operații* dorește să execute ?

Securitatea aplicațiilor Java SE

- *SecurityManager*

- `checkRead(String file)` throws `SecurityException`,...

- *Codebase*

- Semnături digitale

- Permisuni

- File, Socket, Net, Security, Runtime, Property, AWT, Reflect, Serializable

```
grant signedBy "Duke" codeBase "file:/C:/somepath/api/" {  
    permission java.io.FilePermission "/tmp/*", "read,write";  
};
```

Securitatea aplicațiilor Java EE

- Prevenirea accesului neautorizat la funcții ale sistemului sau la datele gestionate de acesta
- Evidența acțiunilor utilizatorilor
- Interoperabilitate la nivel de aplicație sau container
- Transparență în utilizare
- Ușurință în administrare

Ce trebuie să securizăm ?

- Nivelul Web
- Nivelul de logică a aplicației
- Nivelul serviciilor
- Nivelul de transport
- Nivelul mesajelor
- Nivelul datelor

Cuvinte cheie

- **Authentication**
- **Authorization (Access Control)**
 - **Confidentiality (Data Privacy)**
 - **Non-repudiation**
- **Data integrity**
- **Auditing**
- **Quality of Service (QoS)**

Mecanisme de implementare

- La nivelul aplicației (*Application-Layer Security*)
Securitate oferită de containere prin mecanisme:
 - *descriptive*: fișiere configurare, adnotări
 - *programatice*
- La nivelul de transport (*Transport-Layer Security*)
Comunicare **point-to-point** folosind algoritmi de criptare, chei publice, certificate
HTTPS - SSL (Secure Sockets Layer)
- La nivelul mesajelor (*Message-Layer Security*)
Comunicare **end-to-end** în care aspectele legate de securitatea sunt conținute în mesaj și aplicate selectiv conținutului acestuia.

Realms, Users, Groups, Roles

- **Realm (Domeniu)** = bază de date formată din utilizatori și grupuri
- **User (Utilizator)** = identitate definită la nivelul serverului de aplicații
- **Group (Grup)** = mulțime de utilizatori
- **Role (Rol, Funcție)** = Mulțime de permisiuni ce poate fi asignată sau nu utilizatorilor
- **Credential** = Date necesare autentificării
- **Principal** = Entitate care poate fi autentificată

Subject, Principal

- **java.security.Principal**

- Abstracțiune ce identifică folosind un nume o **identitate** ce participă într-un sistem (persoană, companie, etc.)

- **javax.security.auth.Subject**

- Informațiile ce conțin attribute legate de securitate necesare autentificării
publicCredentials, privateCredentials
- Un subiect poate conține mai multe **identități**
principals

javax.security.auth.Subject

A Subject represents a grouping of related information for a single entity, such as a person. Such information includes the Subject's identities as well as its security-related attributes (passwords and cryptographic keys, for example).

Subjects may potentially have multiple identities.

Each identity is represented as a Principal within the Subject. Principals simply bind names to a Subject. For example, a Subject that happens to be a person, Alice, might have two Principals: one which binds "Alice Bar", the name on her driver license, to the Subject, and another which binds, "999-99-9999", the number on her student identification card, to the Subject. Both Principals refer to the same Subject even though each has a different name.

A Subject may also own security-related attributes, which are referred to as credentials. Sensitive credentials that require special protection, such as private cryptographic keys, are stored within a private credential Set. Credentials intended to be shared, such as public key certificates or Kerberos server tickets are stored within a public credential Set. Different permissions are required to access and modify the different credential Sets.

Securizarea nivelului Web

Etapele



- Crearea utilizatorilor la nivelul serverului
- Crearea rolurilor de securitate
- Stabilirea mecanismului de autentificare
- Stabilirea constrângerilor de accesare ale componentelor Web
- Maparea utilizatorilor la roluri



Crearea utilizatorilor



GlassFish

- `http://localhost:4848` → consola de administrare
- Configuration → Security → Realms → file
- ManageUsers
 - `UserId`
 - `GroupList`
 - `Password`



Crearea rolurilor



web.xml

```
<web-app>
  ...
  <security-role>
    <description> Musafir </description>
    <role-name> guest </role-name>
  </security-role>

  <security-role>
    <description> Sef </description>
    <role-name> admin </role-name>
  </security-role>

</web-app>
```



Metode de autentificare



A web client can authenticate a user to a web server using one of the following mechanisms:

- HTTP Basic Authentication
- Form Based Authentication
- HTTP Digest Authentication
- HTTPS Client Authentication



Mecanismul de autentificare

NONE, DIGEST, CLIENT CERTIFICATE, BASIC, FORM

BASIC

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>file</realm-name>
</login-config>
```

FORM

```
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>file</realm-name>
  <form-login-config>
    <form-login-page>login.jsp</form-login-page>
    <form-error-page>error.jsp</form-error-page>
  </form-login-config>
</login-config>
```

Stabilirea constrângerilor

```
<security-constraint>
  <display-name>Constraint</display-name>

  <web-resource-collection>
    <web-resource-name>admin</web-resource-name>
    <description/>
    <url-pattern>/secureAdmin/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>

  <auth-constraint>
    <description/>
    <role-name>guest</role-name>
    <role-name>admin</role-name>
  </auth-constraint>

</security-constraint>
```

Maparea utilizatorilor la roluri

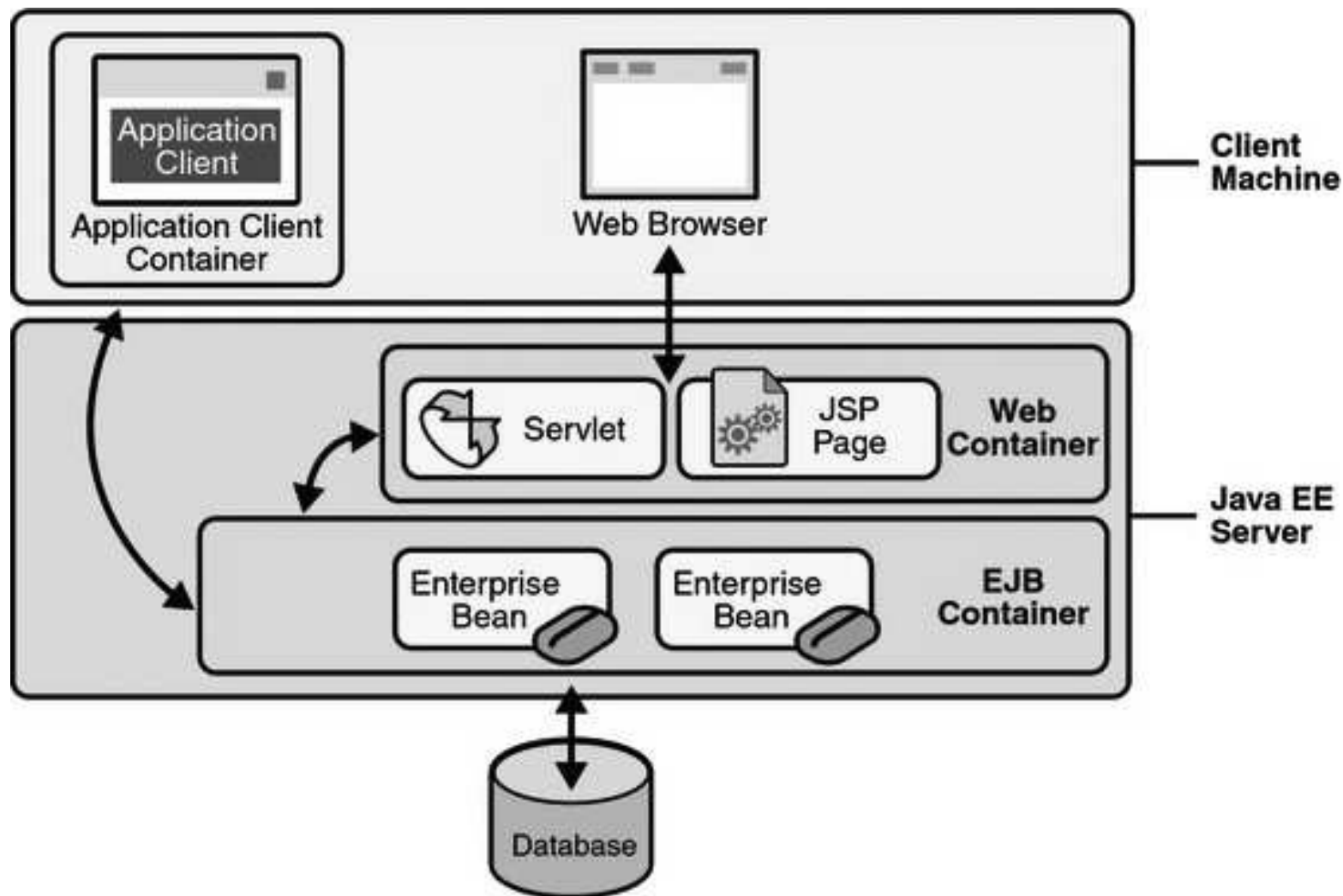
GlassFish: sun-web.xml

```
<security-role-mapping>
  <role-name>guest</role-name>
  <principal-name>ionescu</principal-name>
</security-role-mapping>

<security-role-mapping>
  <role-name>admin</role-name>
  <principal-name>popescu</principal-name>
</security-role-mapping>
```

Securizarea nivelului EJB

Imagine de ansamblu



EJBContext, SessionContext

The **EJBContext** interface provides an instance with access to the container-provided runtime context of an enterprise bean instance. This interface is extended by the **SessionContext**, **EntityContext**, and **MessageDrivenContext** interfaces to provide additional methods specific to the enterprise interface bean type.

The **SessionContext** interface provides access to the runtime session context that the container provides for a session bean instance. The container passes the **SessionContext** interface to an instance after the instance has been created. The session context remains associated with the instance for the lifetime of the instance.

Accesarea contextului de securitate

java.security.Principal getCallerPrincipal()

```
@Stateless public class EmployeeServiceBean
    implements EmployeeService{
    @Resource SessionContext ctx;
    @PersistenceContext EntityManager em;

    public void changePhoneNumber(...) {
        // obtain the caller principal.
        callerPrincipal = ctx.getCallerPrincipal();

        // obtain the caller principals name.
        callerKey = callerPrincipal.getName();

        // use callerKey as primary key to find EmployeeRecord
        EmployeeRecord myEmployeeRecord =
            em.findByPrimaryKey(EmployeeRecord.class, callerKey);

        // update phone number
        myEmployeeRecord.setPhoneNumber(...);
    }
}
```

Accesarea contextului de securitate

boolean isCallerInRole(String roleName)

```
@DeclareRoles("payroll")
@Stateless public class PayrollBean implements Payroll {
    @Resource SessionContext ctx;

    public void updateEmployeeInfo(EmplInfo info) {

        oldInfo = ... read from database;

        // The salary field can be changed only by callers
        // who have the security role "payroll"
        if (info.salary != oldInfo.salary &&
            !ctx.isCallerInRole("payroll")) {
            throw new SecurityException(...);
        }
        ...
    }
    ...
}
```

Specificarea permisiunilor

- `@RolesAllowed("list-of-roles")`
- `@PermitAll, @DenyAll`

```
@RolesAllowed("admin")
public class SomeClass {
    public void aMethod () {...}
    public void bMethod () {...}
    ...
}
@Stateless public class MyBean implements A extends SomeClass {
    @RolesAllowed("guest")
    public void aMethod () {...}

    @PermitAll()
    public void cMethod () {...}
    ...
}
```

Alte aspecte

Biometria



Metode de autentificare

- **Biometrice: "Cine sunt"**
- Cunoasterea secretului: "Ce stiu"
- Posesie personala: "Ce am"
- Sisteme combinate

Biometria reprezinta recunoasterea automata a indivizilor pe baza caracteristicilor biologice si comportamentale.

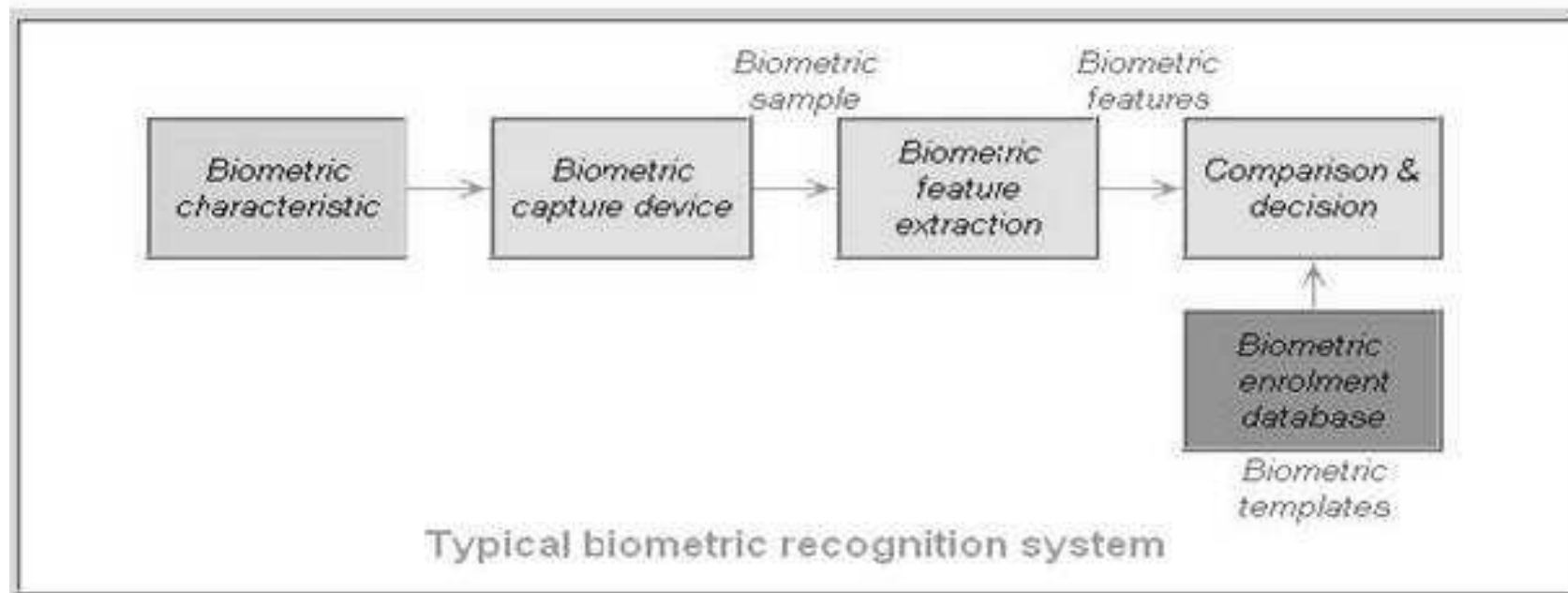
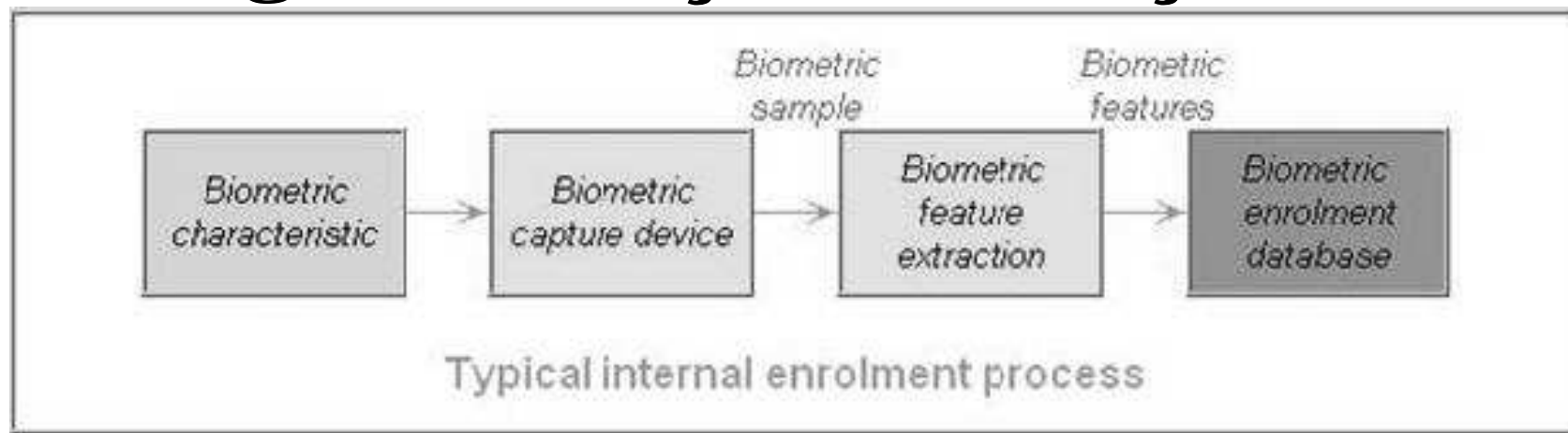


Trăsături biometrice

- **Unicitate:** sa nu mai apara la nici o alta persoana
- **Universalitate:** sa apara la toate persoanele sau la cat mai multe dintre ele
- **Permanenta:** sa nu se schimbe în timp
- **Comensurabilitate:** sa fie masurabile cu instrumente tehnice simple
- **Usurinta in folosire**

Amprenta digitala, Semnatura, Geometria fetei, Iris, Retina Geometria fetei / mainii/
degetului, Structura venoasa a mainii, Forma urechii, Voce, Miros, ADN

Inregistrarea și recunoașterea



JAAS

