Lecture 8:

Role-based Access Control

Access – a specific type of interaction between a subject and an object that result in the flow of information from one to the other

Access control – the process of limiting access to the resources of a system only to authorised programs, processes or other systems

Lecture outline

• RBAC principles
• Modelling languages for RBAC
  – SecureUML
  – UMLsec
• Different security design perspectives
  – Transformation from SecureUML to UMLsec
• Model-driven security
TR.1: A class with a stereotype `<<secureUML.resource>>` is transformed to an activity partition in the UMLsec model.

Operations of this class become actions belonging to this partition:
- each operation becomes a value the UMLsec associated tag `{protected}`
• **TR.2:** A relationship with a stereotype <<assignment>> relationship used to connect users and their roles is transformed to an associated tag {role}

{role= (Bob, FootballFederationEmployee)}
{role= (John, TeamRepresentative)}
{role= (Karl, TeamRepresentative)}

• **TR.3:** A class with the stereotype <<secuml.roles>> is transformed to the UMLsec activity partition
  - The attributes of an association class that connects the <<secuml.roles>> class with <<secuml.resource>> class, become actions in the corresponding activity partition
• **TR.4:** The association class with the stereotype \(<\text{secuml.permission}>\) defines the role value for the associated tag \(\{\text{right}\}\)
  
  - The value of \(\text{right}\) can be determined from the authorisation constraint defined for the attribute of the SecureUML association class.

\[
\text{right} = (\text{FootballFederationEmployee, setTimePlace}) \\
\text{right} = (\text{TeamRepresentative, viewTimePlace})
\]

**TR.5:** Received activity diagram is annotated with the \(<\text{rbac}>\) stereotype
The UMLsec model

Finish the transformation manually

- Define initial and final activity nodes
- Identify logical sequence of activities
  - Specify missing control flows
  - Identify missing conditions
- Define missing and assembly existing association tags
The UMLsec model

- Static Security model – SecureUML
- Dynamic Security model – UMLsec
Two approaches complement each other by providing different viewpoints to the secure software design.
Lecture outline

• RBAC principles
• Modelling languages for RBAC
  – SecureUML
  – UMLsec
• Different security design perspectives
  – Transformation from SecureUML to UMLsec

**Model-driven security**

Model-driven development

- Modelling language
- Modelling tool
- Model (consisting of diagrams)
- Code (transformed model)

Transformation rules
Model-driven development

Model (consisting of diagrams) → Transformation rules → Code (transformed model)

Modelling language → Model → Modelling tool

CREATE SEQUENCE User_SEQ;
CREATE SEQUENCE MeetingAgreement_SEQ;
CREATE SEQUENCE RequiredMaterial_SEQ;
CREATE SEQUENCE secuml.resource_SEQ;

CREATE TABLE User(
    unique-name varchar (255),
    id_User integer,
    PRIMARY KEY(id_User));

CREATE TABLE MeetingAgreement(
    place varchar (255),
    time varchar (255),
    id_MeetingAgreement integer,
    fk_Userid_User integer NOT NULL,
    PRIMARY KEY(id_MeetingAgreement),
    FOREIGN KEY(fk_Userid_User) REFERENCES User (id_User));

CREATE TABLE RequiredMaterial(
    id_RequiredMaterial integer,
    fk_MeetingAgreementid_MeetingAgreement integer NOT NULL,
    PRIMARY KEY(id_RequiredMaterial),
    FOREIGN KEY(fk_MeetingAgreementid_MeetingAgreement)
    REFERENCES MeetingAgreement (id_MeetingAgreement));

CREATE TABLE User_MeetingAgreement(
    fk_Userid_User integer,
    fk_MeetingAgreementid_MeetingAgreement integer,
    PRIMARY KEY(fk_Userid_User, fk_MeetingAgreementid_MeetingAgreement),
    FOREIGN KEY(fk_Userid_User, fk_MeetingAgreementid_MeetingAgreement)
    REFERENCES MeetingAgreement (id_MeetingAgreement));
Model Driven Security

UML (modelling languages) → Data Model

MagicDraw (modelling tool) → Security Model

SecureUML (modelling languages) → Security Model

Data Model → Transformation rules → Code (model transformed to SQL)


Security Model

```
<secuml.constraint>
FootballFederationEmployeeAuthConstraint
(sec: FootballFederationEmployeeAuthConstraint(self.id = "Y"))

<secuml.permission>
FootballFederationEmployeePermission
- enterAgreementDetails : Insert
- changeMeetingInfo : Update
- deleteMeetingDetails : Delete

<secuml.role>
FootballFederationEmployee

- assignedFFEmployee : 1
- organisedMeeting : 1

<secuml.constraint>
TeamRepresentativeAuthConstraint
(sec: TeamRepresentativeAuthConstraint(self.id = "Y"))

<secuml.permission>
TeamRepresentativePermission
- getAgreementInformation : Select
```

21

22
Insert

security constraint

```
CREATE OR REPLACE TRIGGER MeetingAgreement_sec_insert_trg
INSTEAD OF INSERT ON MeetingAgreement_v
REFERENCING NEW AS NEW
FOR EACH ROW
DECLARE
    ex_denied EXCEPTION;
BEGIN
    IF sec.is_role('FootballFederationEmployee') = 'Y' AND sec.FootballFederationEmployeeAuthConstraint(self.id)='Y'
    THEN
        INSERT INTO MeetingAgreement (place, time)
        VALUES (:NEW.place, :NEW.time);
    ELSE
        RAISE ex_denied;
    END IF;
EXCEPTION
    WHEN ex_denied THEN
        raise_application_error (-20000, 'Access denied!');
END;
```
**Insert**

security constraint

- Resource that needs to be secure
  - Place
  - Time

- Security action
  - Insert
Insert security constraint

Checking the Role and …

… the authorisation constraint
Model Driven Security

- **Insert values**
  - Place
  - Time

---

UML (modelling languages) → Data Model → Transformation rules → Code (model transformed to SQL)


SecureUML (modelling languages)
Model Driven Security

- UML (modelling languages)
- MagicDraw (modelling tool)
- SecureUML (modelling languages)
- Data Model
  - Transformation rules
  - Code (model transformed to SQL)
  - Security Model
  - Security Transformation rules
  - Security constraints (in PL/SQL)

```sql
CREATE SEQUENCE User_SEQ;
CREATE SEQUENCE MeetingAgreement_SEQ;
CREATE SEQUENCE RequiredMaterial_SEQ;
CREATE SEQUENCE security_resource_SEQ;

CREATE TABLE User(
  unique_name varchar (100),
  id_User integer,
  PRIMARY KEY(id_User));

CREATE TABLE MeetingAgreement(
  place varchar (255),
  time varchar (100),
  id_MeetingAgreement integer NOT NULL,
  PRIMARY KEY(id_MeetingAgreement),
  FOREIGN KEY(User_id_User) REFERENCES User(id_User));

CREATE TABLE RequiredMaterial(
  id_RequiredMaterial integer,
  MeetingAgreement_id_MeetingAgreement integer NOT NULL,
  PRIMARY KEY(id_RequiredMaterial),
  FOREIGN KEY(MeetingAgreement_id_MeetingAgreement) REFERENCES MeetingAgreement(id_MeetingAgreement));

-- Created common-sql.sql
CREATE OR REPLACE TRIGGER MeetingAgreement_sec_insert_trg
INSTEAD OF INSERT ON MeetingAgreement
REFERENCING NEW AS NEW
FOR EACH ROW
DECLARE
  ex_denied EXCEPTION;
BEGIN
  IF sec_is_role('FootballFederationEmployee') = 'Y' AND
     sec_FootballFederationEmployeeAuditConstraint(self.Id)= 'Y'
  THEN
    INSERT INTO MeetingAgreement (place, time ) VALUES (NEW.place, NEW.time);
  ELSE
    RAISE ex_denied;
  END IF;
EXCEPTION
  WHEN ex_denied THEN
    raise_application_error (-20000, 'Access denied');
END;
```
**Code (model transformed to SQL)**

```
CREATE SEQUENCE User_SEQ;
CREATE SEQUENCE MeetingAgreement_SEQ;
CREATE SEQUENCE RequiredMaterial_SEQ;
CREATE SEQUENCE sequence_resource_SEQ;

CREATE TABLE User
unique-name varchar(255),
ID_User integer,
PRIMARY KEY (ID_User);

CREATE TABLE MeetingAgreement
place varchar(255),
time varchar(255),
ID_MeetingAgreement integer,
R_ID Required Material integer NOT NULL,
PRIMARY KEY (ID_MeetingAgreement),
FOREIGN KEY (R_ID Required Material) REFERENCES Required Material (ID Required Material);

CREATE TABLE RequiredMaterial
required Material integer,
R_ID MeetingAgreement integer NOT NULL,
PRIMARY KEY (R_ID MeetingAgreement),
FOREIGN KEY (R_ID MeetingAgreement) REFERENCES MeetingAgreement (ID MeetingAgreement));
```

**Security constraints (in PL/SQL)**

```sql
BEGIN
IF sec_is_owner('FootballFederationEmployee') = 'Y' AND sec.is_FootballFederationEmployeeAuthConstraint('self.id') = 'Y'
THEN
  INSERT INTO MeetingAgreement (place, time)
  VALUES (:NEW.place, :NEW.time);
ELSE
  RAISE ex_denied;
END IF;
EXCEPTION
  WHEN ex_denied THEN
    raise_application_error(-20000, 'Access denied');
END;
```

**Database management systems (e.g., SQL*Plus)**

**Demo**
Scenario

- Increasing number of documents
- Need to secure the document content
- Display part of document to different stakeholders
{role= (Eve, Receptionist)}
{protected= (addPatientData)}
{right= (Receptionist, addPatientData)}

{role= (Ann, Nurse)}
{protected= (addComplainsObservations)}
{right= (Nurse, addComplainsObservations)}

{role= (John, Doctor)}
{protected= (displayRecord)}
{right= (Doctor, displayRecord)}

{protected= (addDiagnosis)}
{right= (Doctor, addDiagnosis)}
Message to take home

• RBAC principles
• Modelling languages for RBAC
  – SecureUML
  – UMLsec
• Different security design perspectives
  – Transformation from SecureUML to UMLsec
• Model-driven security
  – Transformation of SecureUML to code