Access Control for Enterprise Apps

Dominic Duggan
Stevens Institute of Technology
Based on material by Lars Olson and Ross Anderson
SQL ACCESS CONTROL
App vs Database Security

- Multiple users for Apps (A)
- Apps have elevated privileges (B)
SQL grant Syntax

grant privilege_list on resource
to user_list;

• Privileges: select, insert, etc.
• Resource: table, database, function, etc.
• Individual users
• User group
Example

• Alice owns a database table of employees:
  – name varchar(50),
  – ssn int,
  – salary int,
  – email varchar(50)
Example

• Bob: read-only access

  grant select on employee to bob;

• Carol: read-only access to public info

  grant select (name, email)
    on employee to carol;

  – not implemented in PostgreSQL
  – not implemented for select in Oracle
  – implemented in MySQL
View-Based Access Control

- Carol: read-only access to public info

```sql
create view employee_public
    as select name, email
    from employee;

grant select
    on employee_public to carol;
```
Row-Level Access Control

• Employees can access their own record:

```sql
create view employee_Carol as
    select * from employee
    where name='Carol';
grant select on employee_Carol to carol;
```

• Employees can update their e-mail addresses:

```sql
grant update(email)
    on employee_Carol to carol;

– (Or create yet another new view...)
```
Delegating Policy Authority

grant privilege_list on resource to user_list with grant option;

• Alice:
  grant select on table1 to bob
  with grant option;

• Bob:
  grant select(column1) on table1 to carol
  with grant option;
SQL revoke Syntax

revoke privilege_list on resource from user_list;

• Griffiths-Wade:
  – Sequences of grant / revoke operations
  – ACLs should be indistinguishable from a sequence in which the grant never occurred
  – Cascading revocations
Disadvantages to SQL Model

• Too many views to create
  – Many users, each with their own view
  – View redefinitions
  – Fine-grained policies each require own view
  – Complicated policy logic
  – Update anomalies
VIRTUAL PRIVATE DATABASES
Virtual Private Databases

• Security model for Oracle
• Policies: user-defined functions that return where condition
• Applications can define “context,” e.g. for RBAC
Features

• Functions executed each time table is accessed.
• Multiple functions can be attached to a table.
• Different functions can be defined depending on:
  – Operation (read vs. write)
  – Columns being accessed
Simple Policy

• Two users, Alice and Bob
• Alice creates a table:

```
create table data(
  a int primary key,
  b varchar2(50));
insert into data values(1, 'hello');
insert into data values(2, 'world');
commit;
```

• Alice wants to limit Bob’s access to the row where a=1
Simple Policy

• Alice wants to limit Bob’s access to the row where a=1

• Three steps:
  – Grant Bob access to the table:
    grant select on data to bob;
  – Create a policy function
  – Attach the policy function to the table
create or replace function testFilter
(p_schema varchar2, p_obj varchar2)
return varchar2 as
begin
  if (SYS_CONTEXT('userenv', 'SESSION_USER') = 'BOB') then
    return 'a = 1';
  else
    return '';
  end if;
end;
Simple Policy

execute dbms_rls.add_policy(
    object_schema => 'alice',
    object_name => 'data',
    policy_name => 'FilterForBob',
    function_schema => 'alice',
    policy_function => 'testFilter',
    statement_types => 'select, update, insert',
    update_check => true);

Logging Policy

create or replace function
testLogging(p_schema varchar2, p_obj varchar2)
return varchar2 as
begin
    insert into alice.logtable values(
        sysdate,
        SYS_CONTEXT('userenv', 'SESSION_USER')
        || ',' ||
        SYS_CONTEXT('userenv', 'CURRENT_SQL'));
    commit;
    return '';
end;
/

Reflective Policy

• Table for policy (for table data)
  
  create table userperms (  
    username varchar2(50),  
    a int references data);  

• Populate the table:
  
  insert into userperms values('BOB', 1);  
  insert into userperms values('ALICE', 1);  
  insert into userperms values('ALICE', 2);  
  commit;
create or replace function testFilter(p_schema varchar2, p_obj varchar2)
return varchar2 as
begin
  return 'a in (select a from alice.userperms ' || 'where username = ''' || SYSCONTEXT('userenv', 'SESSION_USER') || '''')';
end;
/

Fine-Grained Access Control

• Predicated grants

```
grant select on employee
  where (empid = userId())
to public
```

• VPD through app server filtering?
  – http://mattfleming.com/node/243
BEYOND ACCESS CONTROL
Principal B cannot read file F
Principal B can read contents of file F copied to file G

**Trojan Horse**

**Principal A**

Program Goodies

Trojan Horse

executes

read

write

File F

ACL

A:r
A:w

File G

B:r
A:w
MLS (Bell-Lapadula)

$L_{\text{Max}}(\text{General}) = \text{TopSecret}$

$L_{\text{Max}}(\text{Colonel}) = \text{Secret}$

$L_{\text{Current}}(\text{General}) = \text{Secret}$

$L_{\text{Max}}(\text{President}) = \text{Classified}$
Declassification: Intentional Leaks

Bob → Spreadsheet

Final Tax Form

explicit release

Preparer → WebTax

Proprietary Database

Tax Data

No explicit release
Multi-Level and Multi-Lateral

(TOP SECRET, {EUR,ASI,NUC})

(TOP SECRET, {EUR})  (SECRET, {EUR,ASI,NUC})

(TOP SECRET, {})  (SECRET, {EUR})

(SECRET, {})  (UNCLASSIFIED, {})

(TOP SECRET, {})
Clark-Wilson

• **Principles for data integrity**
  – Only access data through well-formed transactions
    • E.g. double-entry book-keeping (financial)
    • E.g. audit log (HPPA)
  – Separation of duties

• **Policy triples (S, TP, CDI)**
  – S = subject
  – TP = transformation procedure
  – CDI = constrained data item
BMA Security Model

• Decentralized
  – Patient record = the maximum set of health information with a single access control list
  – “Peer-to-peer” alternative to centralized databases
BMA Principle #1

• Access Control
  – Each identifiable record is marked with an ACL naming the people or groups of people who may read it and append data to it
BMA Principle #2

• Record Opening
  – Clinician can open a record with herself and patient on the ACL.
  – Where patient referred, can open record with herself, patient and referring clinician on ACL
BMA Principle #3

- Designated Control
  - One of the clinicians on the ACL must be marked as being responsible
  - Only she may alter the ACL
  - Only health professionals should be added to ACL
BMA Principle #4

• Consent and notification
  – Responsible clinician must notify the patient
    • of the names on his record’s ACL when it is opened,
    • of all additions to ACL and
    • whenever responsibility is transferred
BMA Policy

• Access control
• Record opening
• Designated control
• Consent and notification
• Persistence
• Attribution
• Information flow
• Aggregation control
• Trusted computing base
Relationship-Based Access Control (ReBAC)

• RBAC: Policies are sets
  – “Who are you?”

• ReBAC: Policies are relations
  – “Who do you know?”

• Scenario: Temporary access for consultation
Type Enforcement (SELinux)

Subject type can access object type to perform operations on objects
Type Enforcement Access Control

• All accesses must be explicitly granted in policy

• “Allow” rules specify:
  – Source type (domain type of process)
  – Target type (object type being accessed)
  – Object class
  – Permissions

• Example:
  allow user_t bin_t : file {read
Domain Transitions

- **Principle of Least Privilege:**
  - Any process must be able to access only such information and resources that are necessary to its legitimate purpose.

```
Euid: root
Type: passwd_t

allow passwd_t
shadow_t:
file { ... write ... }

allow passwd_t
shadow_t:
Object: /etc/shadow

Program: passwd

Euid: joe
Type: user_t

Program: bash

r-------- root root
Type: shadow_t
```
Conclusions

- Security is hard