Oracle
Fine Grained Access Control

by
Arup Nanda
• Oracle DBA for more than 10 years
• Written 50+ Articles
  – Oracle Magazine, Select Journal, DBAZine.com, SQLUpdate, Oracle Scene, TechJournal
• Presented at several conferences
  – Oracle World, IOUG Live, OraTechs, AOTC, VOUG, NYOUG
• Executive Director of Connecticut Oracle User Group
• Editor of Select Journal – the IOUG Publication
• Written the book Oracle Privacy Security Auditing, from Rampant TechPress
• Awarded DBA of the Year by Oracle.
# Hospital Database

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DrAdam</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>DrBarb</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>DrCharlie</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Doctor</th>
<th>Name</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Larry</td>
<td>Ego</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Bill</td>
<td>Control</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Scott</td>
<td>Fickleness</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Craig</td>
<td>LowVision</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>Lou</td>
<td>Greed</td>
</tr>
</tbody>
</table>
**Patient Application**

Dr. Adam
Doctor ID = 1

select * from patients
where doctor_id = <id of the doctor logged in>

<table>
<thead>
<tr>
<th>ID</th>
<th>Doctor</th>
<th>Name</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Larry</td>
<td>Ego</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Bill</td>
<td>Control</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Scott</td>
<td>Fickleness</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Craig</td>
<td>LowVision</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>Lou</td>
<td>Greed</td>
</tr>
</tbody>
</table>
Hospital Database

DOCTORS

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PATIENTS

<table>
<thead>
<tr>
<th>ID</th>
<th>Doctor</th>
<th>Name</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Select * from PATIENTS

Select * from PATIENTS
Where DOCTOR_ID = 1
Options

• Application Change
  • Add a predicate to each SQL statement
  • No security!

• Views
  • Automatic predicate
  • Selection on view; no access to base table
  • Too many views
  • Predicate has to be static
  • Difficult to determine accountability
A Third Option

- Automatic application of predicate
- User’s statement
  ```sql
  SELECT * FROM PATIENTS
  ```
- Transformed to
  ```sql
  SELECT * FROM PATIENTS
  WHERE DOCTOR_ID = <ID>
  ```
- Predicate generated by a user defined policy function.
Policy

<table>
<thead>
<tr>
<th>ID</th>
<th>Doctor</th>
<th>Name</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Larry</td>
<td>Ego</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Bill</td>
<td>Control</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Scott</td>
<td>Fickleness</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Craig</td>
<td>LowVision</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>Lou</td>
<td>Greed</td>
</tr>
</tbody>
</table>

select * from patients;

select * from patients where doctor_id = 1

Doctor ID = 1

Policy Function
where
doctor_id = 1
Policy Function

• Takes only two arguments
  – Table Owner
  – Table Name
• Must return a predicate that is to be applied, without the word WHERE.
• The predicate must be syntactically correct
  – Correct: doctor_id = (select doctor_id from doctors where doctor_name = USER)
  – Incorrect: doctor_id = (select USER from doctors)
Policy Function

create or replace function get_doctor_id
(
    p_schema_name in varchar2,
    p_table_name in varchar2
)
return varchar2
is
    l_doctor_id number;
begin
    select doctor_id into l_doctor_id
    from doctors
    where doctor_name = USER;
    return 'doctor_id = ' || l_doctor_id;
end;

returns the currently logged in username
Adding a Policy

```sql
begin
    dbms_rls.add_policy(
        object_schema => 'HOSPITAL',
        object_name   => 'PATIENTS',
        policy_name   => 'PATIENT_VIEW_POLICY',
        policy_function => 'GET_DOCTOR_ID',
        function_schema => 'HOSPITAL',
        statement_types =>
            'SELECT, INSERT, UPDATE, DELETE',
        update_check => true,
        enable => true
    );
end;
```

- **object_schema**: 'HOSPITAL',
- **object_name**: 'PATIENTS',
- **policy_name**: 'PATIENT_VIEW_POLICY',
- **policy_function**: 'GET_DOCTOR_ID',
- **function_schema**: 'HOSPITAL',
- **statement_types**: 'SELECT, INSERT, UPDATE, DELETE',
- **update_check**: true,
- **enable**: true

*the table on which the policy is defined*
*the owner and name of the policy function*
*Policy applied to all types of statements*
Query Transformation

Original Query

```
SELECT * FROM PATIENTS
```

Modified to

```
SELECT * FROM

(SELECT * FROM PATIENTS)

WHERE DOCTOR_ID = 1
```
Insert/Update Check

User DRADAM allowed to see only DOCTOR_ID = 1
He tries to insert a record with DOCTOR_ID = 2
ORA-28115: policy with check option violation
He issues
    update PATIENTS set DOCTOR_ID = 2;
ORA-28115: policy with check option violation, if
    update_check = TRUE
create or replace function get_doctor_id
(
    p_schema_name   in   varchar2,
    p_table_name   in   varchar2
)
return varchar2
is
    l_doctor_id   number;
begin
    if (p_schema_name = USER) then
        return null;
    end if;
    select doctor_id
    into l_doctor_id
    from doctors
    where doctor_name = USER;
    return 'doctor_id = '||l_doctor_id;
end;
Other Bypasses

• System Privilege
• EXEMPT ACCESS POLICY
• SYS and DBA roles have this by default.
OtherDependent Tables

Applied predicate
WHERE PATIENT_ID IN (SELECT PATIENT_ID
FROM PATIENTS)
Multiple Policies

- Table can have multiple policies of the same type.
- Each policy applied with AND

```sql
select *
from patients
where
  DOCTOR_ID = 1
  AND
  PROC_CODE != 'HIV'
  AND
  TREATED = TRUE
```

OR

```sql
select *
from patients
where
  policy1
  AND
  policy2
  AND
  policy3
```
Extending the Functionality

Table for Authorized User

Table: USER_AUTHORITY

- USERNAME - the name of the user
- DOCTOR_ID – the DOCTOR_ID this user is allowed to see

Policy Function Change

```sql
select deptno into l_doctor_id
from user_authority where username = USER;
l_ret := 'doctor_id = '||l_doctor_id;
```

Table TREATMENTS (PATIENT_ID, TRATMENT_ID)

```sql
l_ret := 'patient_id in (select patient_id from patients)';
```
VPD and Other Oracle Tools

VPD is applied in Conventional Modes only.

Export DIRECT=Y

EXP-00079: Data in table "PATIENTS" is protected. Conventional path may only be exporting partial table.

. . . exporting table PATIENTS 3 rows exported

SQL*Loader DIRECT=Y

SQL*Loader-951: Error calling once/load initialization

ORA-00604: error occurred at recursive SQL level 1

ORA-28113: policy predicate has error

Direct Mode Load

insert /*+ APPEND */ into EMP;
ERROR at line 1:
ORA-28115: policy with check option violation
Managing Policies

• View DBA_POLICIES
• Oracle Policy Manager
  – oemapp opm
• Applied Policies
  – V$VPD_POLICY
Refreshing a Policy

```
dbms_rls.refresh_policy (  
    object_schema => 'HOSPITAL',  
    object_name   => 'PATIENTS',  
    policy_name   => 'PATIENT_VIEW_POLICY');
```

Required when the parsed copy of the policy function needs to be changed.

Refreshing guarantees that. Recommended every time the policy or function is changed.

Not required in 9i
Dropping a Policy

dbms_rls.drop_policy ( 
    object_schema => 'HOSPITAL',
    object_name => 'PATIENTS',
    policy_name => 'PATIENT_VIEW_POLICY'
);

When the policy is not required anymore or the table should not be subjected to the restrictions.
Enabling / Disabling a Policy

```
dbms_rls.enable_policy (  
    object_schema => 'HOSPITAL'
    object_name => 'PATIENTS',
    policy_name => 'PATIENT_VIEW_POLICY',
    enable => TRUE
  );
```

When enabling a policy, just change parameter `enable` to TRUE and execute this function.
Troubleshooting

- Most errors produce trace files
- Debugging
  - `alter session set events
    '10730 trace name context forever, level 12';
  `Will produce the rewritten query in a trace file
- ORA-28110: Policy function or package has error
  - Recompile the package
- ORA-28112: failed to execute policy function
  - Some unhandled exception; check the trace file
- ORA-28116: insufficient privileges to do direct path access
  - Conventional or Exempt User
- ORA-28113: policy predicate has error
  - Check the trace file – SYNTAX Problem
Application Users

User: DrAdam

User: DrCharlie

Application Server

User: APPUSER

Empowering Intelligence
Client Identifier

- Introduced in Oracle 9i
- `dbms_session.set_identifier('<identifier>')`
- `CLIENT_ID` in `V$SESSION`
- `CLIENT_ID` in Auditing
- `sys_context('USERENV','CLIENT_IDENTIFIER')`
Application Context

Select USER from dual;
Select SYS_CONTEXT ('USERENV', 'CURRENT_USER') from dual;

set_app_ctx

<table>
<thead>
<tr>
<th>APP_CTX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTR1</td>
</tr>
<tr>
<td>ATTR2</td>
</tr>
</tbody>
</table>
Oracle 10g Enhancements

Relevant Columns

SELECT COUNT(*) FROM PATIENTS
SELECT PATIENT_ID FROM PATIENTS
SELECT SOCIAL_SEC_NO FROM PATIENTS

Another parameter

dbms_rls.add_policy (  
  ...
  sec_relevant_cols => 'PATIENT_ID'
)
Policy Types

- dynamic
- context_sensitive
- shared_context_sensitive
- static
- shared_static
Conclusion

• Different view – on user
• Predicate applied automatically
• Predicate user generated
• 10g enhancements
Thank You!

Questions?

arup@proligence.com