PeteFinnigan.com Limited
Oracle Security

Security Design For Your Database Applications

Least privilege, data and ownership

Legal Notice

Security Design For Your Database Applications

Published by
PeteFinnigan.com Limited
9 Beech Grove
Acomb
York
England, YO26 5LD

Limit of Liability / Disclaimer of warranty. This information contained in this material is distributed on an “as-is” basis without warranty.
Whilst every precaution has been taken in the preparation of this material, neither the author nor the publisher shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions or guidance contained within this course.

TradeMarks. Many of the designations used by manufacturers and resellers to distinguish their products are claimed as trademarks. Linux is a trademark of Linus Torvalds, Oracle is a trademark of Oracle Corporation. All other trademarks are the property of their respective owners. All other product names or services identified throughout the material are used in an editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this material.

Pete Finnigan – Who Am I?

- Oracle Security specialist and researcher
- CEO and founder of PeteFinnigan.com Limited in February 2003
- Writer of the longest running Oracle security blog
- Author of the Oracle Security step-by-step guide and more recently "Oracle Expert Practices"
- Member of the OakTable
- Speaker at various conferences
  - UKOUG, PSOUG, BlackHat, more..
- Published many times, see
  - www.petefinnigan.com for links
- Influenced industry standards
  - And governments

Agenda

- The problem space
- Exploring technical details
- Possible technical solutions
- Privilege analysis
- Design of Users / Schemas
- Deployment and development

The Problem

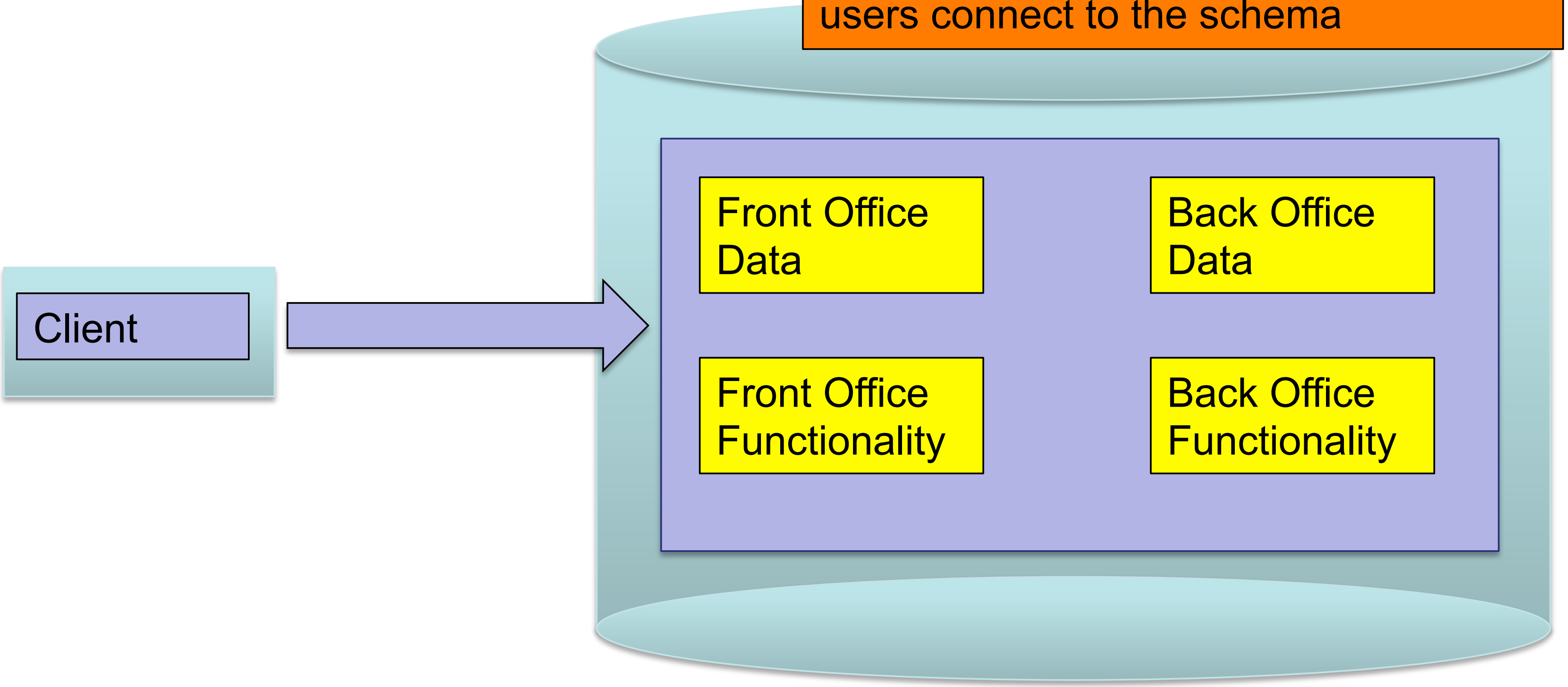
- Applications are often not designed for security of data
- Reviews performed over the years often show that the biggest issue is a lack of security of data
- Canned applications are hard to fix as vendors do not want changes
- Working (functionally) internal applications are hard to fix again because change is not wanted
- Fixing is hard / impossible? when the application exists
- The problem: Change?; lack of security design?

Data Domains - Example

All data, front and back office are in the same schema

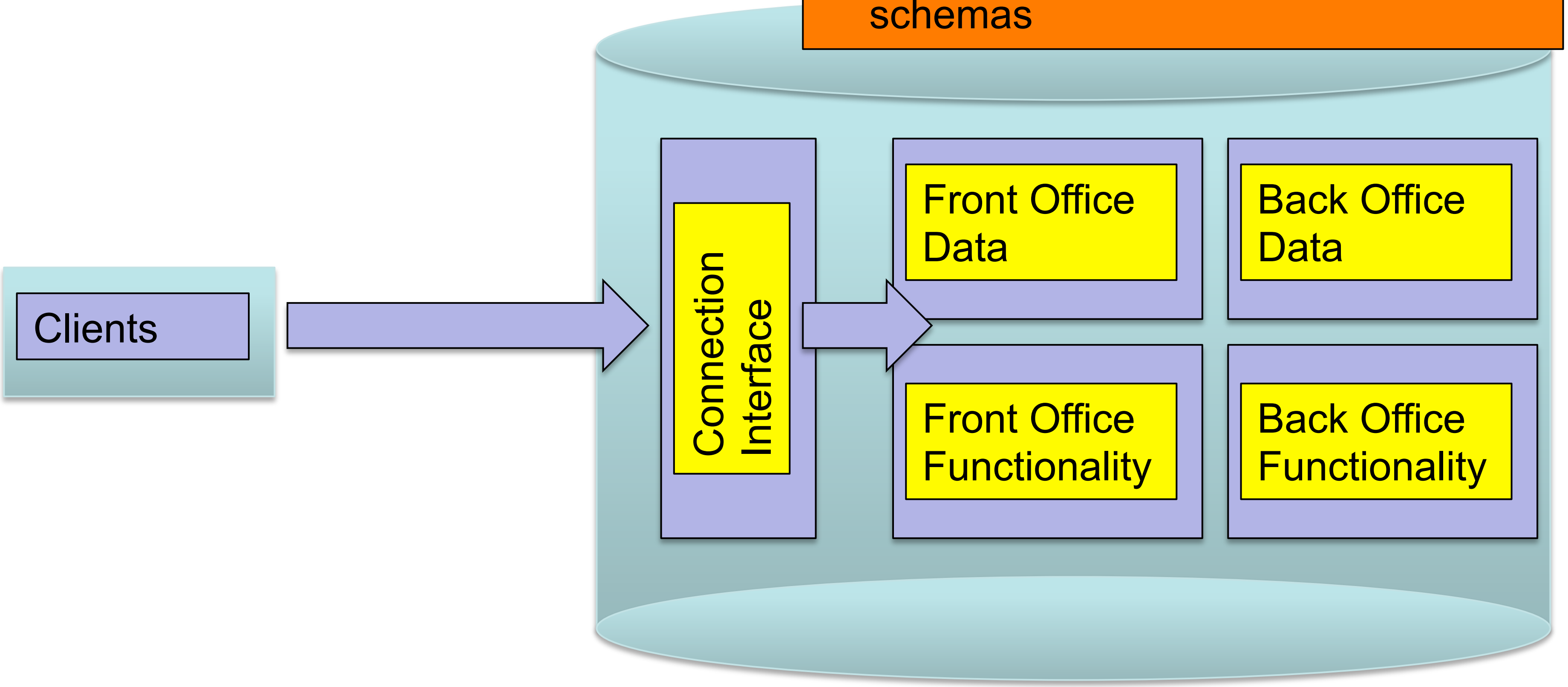
All functionality for front and back office are in the same schema

The web application and back office users connect to the schema



Data Domains - Future

- The ideal scenario is to separate front and back office data and ideally also functionality.
- End users should not connect to any of the schemas directly
- Permissions NOW are needed between the users/schemas/schemas



Problem Space – At a Lower Level

- What are the core issues in application design
  - Shared schemas for multiple applications
  - Shared schemas for data / functionality in applications
  - End users log on to schemas
  - No privilege model – object owner is used or grants to all users or the owner has the DBA role or similar
  - SQL injection issues in application code
  - No data domains
  - No separation of critical functionality
  - No separation of critical data
  - No easy way to control functionality and data access if hacked

SoD and CoI

- (SoD) Separation of Duties:
  - This is the idea that more than one person **is required** to complete a task
  - In business the sharing of a task between two or more people is known as a **security control**.
- (CoI) Conflict of Interest:
  - A situation where a person or party has the potential to undermine their impartiality
- These issues can exist in the database layer, application layer or in the database code / data where the database provides controls for the application layer
- SoD Example: A user has the ability to create a loan and also approve it
- CoI Example: A batch user has application rights and also developer rights

Who Controls the Security of Data?

- Number of possible operating "modes"
  - Data and function in the database
  - Data in the database, function external (Java for instance)
  - All users have database accounts
  - All users share a single database account (possibly pooled)
  - All users connect to the schema (or one of the schemas)
- The database manages security (whether its good or bad!)
- All users have excessive data access because the application layers controls over the database access controls (if they exist)
- **The database must manage the security of the data as the data is in the database**
- Also the database must not relinquish security in favour of believing the application manages it
- It is OK for the application to manage security but in concert with the database – the database provides the controls for the application to use

How Many Security Issues Are Design Related?

- Applications are most often not designed with data security in mind – rarely some are in my experience
- An audit of an Oracle database will highlight parameters, settings, privilege grants, even code issues
- None of these are usually the major issue that causes concern that data can be stolen
- The core issue is always potential unauthorised access to data
- Strangely
  - Adding CPU / Security patches often will not add more protection to the actual data
  - Hardening of the database parameters will not in general add any more protection to the actual data
- Design choices represent the biggest problems for security of data

Lack of Granularity

- A lack of security granularity at many levels causes issues
- If schemas are shared
  - Function and data are together
  - Multiple applications in same schema
- There is no security separation in terms of functionality – i.e. everything is lumped together
  - In schemas (data)
  - In packages
- Privileges
  - If you do not separate functionality or even data it makes it hard or impossible to separate privileges
- Granularity can be fixed but its hard work

Possible Design Solutions - Helpers

- Over the years Oracle have provided many data security solutions such as (some free with EE, some cost options):
  - VPD – Virtual Private database
  - OLS – Oracle Label Security
  - FGA – Fine Grained Audit
  - DV – Database Vault
  - Data Masking
  - Data Redaction
  - TDE – Transparent Database Encryption
  - More..
- We can use these products and extensions but unless we get the basics right these are just duct tape
- Also use of an additional option requires more design, more protection

Design of Schemas

- Multiple schemas are necessary
- At least separate data from function
- Separate critical function from normal function
- Separate critical data from normal data
- Do not allow end user connections to schemas
- This all means that
  - We need grants of rights between schemas and users and schemas and schemas
  - Separation is now possible and controllable
  - Context based security is also now possible

Code Privilege - Invoker vs Definer Rights

- Background http://docs.oracle.com/database/121/DBSEG/dr_ir.htm#DBSEG658
- Note that direct grants are needed even for invoker rights at compilation time for static dependencies
- If you share definer and invoker then all code **decays** to definer
- Careful design is needed to use invoker rights
- Invoker implies end users need grants – example ALTER USER
- Invoker is not a global solution because of the needed isolation and also of limiting the grants needed

Invoker - Inherit Privilege in 12c

- If a powerful user can be tricked into running invoker rights code
  - Or an existing definer rights procedure can be replaced with invoker rights code then privileges can be inherited including roles
  - Replacing a definer rights procedure with your own code can target only the direct grants
- Inherit is a back stop to prevent this but its default position is open for backwards compatibility
  - Better that users who run invoker code do not have excessive rights
- See Tim Halls example - https://oracle-base.com/articles/12c/control-invoker-rights-privileges-for-plsql-code-12cr1
- Revoke and control – no one has / will do?
- Note inherit is not trapped by an exception handler
- “Inherit any” is clearly a dangerous system privilege
- The invoker users can use their rights but grants are blocked in invoker code

Demo admin_grant.sql

With Admin / With Grant

- “with grant option”
  - Only works on users / PUBLIC and not roles
  - Revokes cascade
- “with admin option”
  - For roles or system privileges
  - Side effects:
    - Can revoke a role from someone else (not the owner), alter roles authorisation; drop the role
- “with grant” is needed for a view granted to a user that accesses a third users table
- The recipient of the “with grant” could grant on
- Careful review exposure of data via views is needed

Privilege Separation and Duplication

- One problem I see in all databases I review is cross over and duplication
  - Cross over occurs
    - Oracle granted to => customer granted to => Oracle …
    - Also consider incorrect customer => customer cross over
- Duplications are a bigger issue
  - Every single statement executed by the database incurs recursive SQL that in turn determines rights for the SQL to run
  - Duplications can be massive in some cases
  - Duplications cause confusion in management and maintenance and slow the database
- Both cause investigation of privilege to be complex
- Oracle are just as guilty with their own code

"High Risk / Low Risk" Code and data

- Not all code or data is the same
- For instance a procedure that creates users is more high risk
- For instance a procedure that decrypts data is more high risk
- Credit card details are higher risk than meaningless text
- Least privilege is not just about grants of privileges but also about access and classification of data and functionality
- We will look next at some detailed examples of these aspects

Existing Applications – Possible Solutions

- When some of the problems we have discussed appear in new applications/ database designs there are opportunities to change the design and make them more secure
- With existing applications (particularly wide spread COTS products) fixing issues is harder
- We can try to:
    - Add a layer between the database and application to filter or control actions or access to data.
        - This can be done with triggers (DML, DDL), views (+ functions), network controls and more
    - Add a layer in front of the application functionality in the database
        - For instance intercept all calls to specific procedures, manipulate the input and then call the original code
    - We can use layered features such as password controls
    - We can use cost options such as DV, OLS or VPD with EE

Context Based Security

- What is “context” in the context of Oracle Security?
- Privileges could be manage in the database or application layer and access could be based on
  - When – time based
  - Where – in which code and called from which other code
  - Why – permissions, users, roles set, IP Address, machine…
- A context should be set or tested to decide on whether privileges, resource or code path is allowed
- In Oracle could be done with VPD, OLS, DV, FGA, SAR
- But can also be for free with
  - PL/SQL Code, SYSTEM triggers, DML triggers, Views…
  - Role membership can be a control – even if role has no rights

Context – Use of a Privilege

- If system privilege needs to be used as part of an application then it must be properly controlled
- If this is a dangerous privilege such as ALTER USER, CREATE USER, GRANT.., ROLE… then much more care is needed
- Create a schema that “owns” the privilege – for example ALTER USER and importantly this schema does nothing else
- This schema then exposes the privilege via a PL/SQL API for use to limit its use to intended function only
- The schema should be locked
- Access to the API is controlled via grants and rules in the code
- A DDL trigger can be added to prevent direct use outside of the PL/SQL of ALTER USER

Demo: resource.sql – shows code before solution

Context Based Resource Management

- Similar to privilege control; Use a separate schema, grant for example a DIRECTORY to the schema. The PL/SQL control API must be secured
  - By grants to the right users
  - By code embedded in the PL/SQL that ensures that it is only used as planned
- But we cannot use an extra system trigger to control access to a DIRECTORY as we can with ALTER USER as there is no suitable system event
- We could use compensating controls
  - Audit on directory use
  - A system error trigger to detect error (i.e. a possible attack)
- Focus on limiting access to the DIRECTORY object in the schema and on locking and preventing direct schema access

Demo: dacc.sql

Context Based Access to Data

- We can use cost options such as DV or OLS to protect data access but we also can use free solutions
  - SELECT blocking without VPD, DV, OLS is harder **(Extra License needed)**
  - There is no SELECT trigger apart from FGA (Fine Grained Audit) – not possible in my standard edition database
  - We could create a complex solution
    - Create a PL/SQL function that implements the access control logic and blocks access where necessary
    - Create a view on the table to be protected. My table is CREDIT_CARD
    - Call the access control function in the view to deny access or allow access
  - There are still issues though
    - The attacker could access the base table CREDIT_CARD directly instead and still read the data
    - We then need to have the view in a separate schema and lock the schema with the data (if we need to use synonyms)
    - The use of the view needs to be granted to any other user WITH ADMIN so even moving the data would not work.
    - The “with admin” undermines the base security
  - We could also limit DML more easily (Insert, Update and Delete) by creating DML triggers with rules to limit access

Protecting Code and Data

- We also have to think about protecting critical PL/SQL code
- Sometimes the IPR is the code itself or it contains risky data – i.e. encryption keys, magic values
- Obfuscate the code - better
  - Wrap the code - weaker
- We also can protect some of the data if needed
  - Obfuscate it
  - Encrypt it
  - Audit it for change and access
- Need to review code for issues, checksum the code
- We can also grant roles to PL/SQL in 12c

Privilege Analysis

- What is privilege analysis?
    - Locate all privileges for all users
    - Establish all grants and roles
    - Establish user type if possible
- Must be done for schemas, users, support, DBAs
- Four types of privileges
    - Compile time (create)
    - Rebuild/change time (alter)
    - Run time (object use and access)
    - System rights (support / maintenance)
- Database Vault in 12c includes privilege analysis tools
    - We can simulate with analysis tools and audit trails
    - Also locate duplicates and cross over

Demo: some of the tools

Privilege Analysis - Demo

- We can use:
  - use.sql
  - who_can_access.sql
  - who_has_priv.sql
  - find_all_privs.sql
  - get_tab2.sql
- We can also analyse role membership
- We should review hierarchical privileges

Design of Users

- What is a database user?
  - An account in the database that is used by **a real person**
    - A DBA
    - An end user
    - A support user
- Each user account should have its rights designed for ***least privilege***
  - The absolute rights necessary to do their assigned job/role (not database role)
- If a user/DBA/Support (a real person) connects to or uses an existing account such as SYSTEM or a schema then ***least privilege*** is impossible

Least Privilege

- One of the most common issues I see in customer databases is a complete lack of “least privilege”
- This must be combined with SoD (Segregation of Duties) and CoI (Conflict of Interest)
- Each person/process must have one suitable account for one job
- That account must have the minimum privileges necessary to do the intended job and only these privileges.

Creation Time Privileges

- Some rights are only needed at initial create time and never again
- Once an object exists it only needs to be updated for change control
- The “create rights” are not needed day-to-day
- Once an object exists it gets implicit ALTER and DROP for the owner
- If CREATE OR REPLACE are used then the privileges do not need to be re-granted
- Privileges can be revoked and granted back for change controls

Run Time Rights

- Run time rights fall into two main categories
- Data and functional access
    - Users and schemas need access to each others data and functions
- Support, DBA and Maintenance
    - Support need access to schemas data and functions – often only read and not change and no execute
    - DBAs need access to change structure and monitor the database
- Ensure all rights are classified, are necessary and understand when they are necessary

Developer and Third Party Access

- When third parties need access to maintain the software they have provided to you often the only option is to release the schema passwords
- When developers or release teams need to update or maintain your in-house applications often the only option is to release the schema passwords
- The solutions could be
  - Schema Access
  - DBA like powers – CREATE ANY PROCEDURE
    - This gets complex as other %ANY% rights are also needed
  - Grants for another schema require powerful privileges
- The better solution is proxy access as all actions work as though connected as the intended user
- Connecting to the schema (owner) should be avoided
- **Development should use the same model as run time (owner access at development time and proxy for deployment) then designed rights work in DEV and PROD**
- Audit can be enabled for all actions whilst connected and it cannot be disabled

Conclusions

- Understand data domains
- Understand how your schemas have been designed
- Think about the security of data not simple post hardening
- Do not have weak database rights and rely on the application
- Get creative if necessary

Questions?

Any Final Questions?

Security Design For Your Database Applications

Least privilege, data and ownership