Building Practical Audit Trails

Design easy to implement and use audit trails
Legal Notice

Oracle Database Security Presentation

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

Copyright © 2015 by PeteFinnigan.com Limited

No part of this publication may be stored in a retrieval system, reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, scanning, recording, or otherwise except as permitted by local statutory law, without the prior written permission of the publisher. In particular this material may not be used to provide training or presentations of any type or method. This material may not be translated into any other language or used in any translated form to provide training or presentations. Requests for permission should be addressed to the above registered address of PeteFinnigan.com Limited in writing.

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
- 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

- History of this talk
- Design Must Come first
- What is practical?
- Background / Possible Solutions
- Solutions and Examples
History; 12c

- I wrote this presentation back in 2012 and presented it just once at a SIG
- I subsequently designed policy based auditing and VPS and a simple firewall
- This then became the basis of a one day class on the same subject
- 12c came along and unified audit was part of 12c. I was not in the beta
- The material here is relevant for 12c either using standard audit or unified audit
- The same nuances apply whether core audit or unified
- I am going to focus at a high level but also use core audit and not 12c Unified audit
Design Comes First – No Matter the solution

- Before we get started implementing
- Design must be the first step
- The solution implements the design
- Therefore until you know the design you cannot specify the right solution – right?
- The solution could be “free” or “commercial” solution or even a combination of both
- So often people buy products and implement out of the box with no internal requirements
“What do I want to know?”

- Start with “what do I want to know?”
  - Risk based
  - Based on regulations
  - Based on business needs
- Then design technical solutions to implement these requirements
- Include
  - Sizing, performance, tech data collect, transmit/transfer if relevant, raw trail storage, reports, alerts, management, issues management, more...
So What Is Practical In This Context?

- Quick to implement
- Cheap in software license / time
- Easy to maintain and extend
- Easy to configure in the first place
- Provides value vs risk
- On-going support
- Easy customisation
Fundamentals

- Do we satisfy auditors or compromise?
  - Risk vs cost (implement and TCO/ROI)
- Keep the raw trail or the reports?
- Trail to be kept off the server or local?
- Size of the storage required?
- Performance (depends on actions captured and design decisions)
- Re-Active “vs” Pro-Active audit
The Case For Free Core Solutions

- The free core solutions are worth using
- Simple to design
  - We may have to be creative
- DBA can implement and maintain
- Should be easy to extend
- If database used (aud$) reports are simple
- Can use database to manage
Possibilities to Audit an Oracle Database?

- Options – at a very high level
  - DAM – Database Activity Monitoring
  - IDS – Intrusion Detection System
  - IPS – Intrusion Prevention System
  - VPS – Virtual Patching Systems
  - Audit Log Monitoring
  - Centralised Audit – Audit Vault
  - Native database audit / OS audit features
- Solutions can be
  - Network / host
  - Hardware / software
  - Both / All
Address The Elephant In The Room

• It is: **Performance of database audit**
• Storage to a lesser extent; we need to consider
  • Do we store results or alerts/reports – regulations will impact this choice
• Design with performance in mind (often we can catch an issue (attack) without creating an extensive trail)
• Capture non-standard access – no performance issue
• Because of perceived performance problems sites do nothing in the core; yet use DML triggers in applications with horrendous performance results
Options Compared – Commercial / Free

<table>
<thead>
<tr>
<th>Option</th>
<th>Commercial</th>
<th>Free (Database)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-Day Detection</td>
<td>Yes (open to debate)</td>
<td>No</td>
</tr>
<tr>
<td>Detailed Reports</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Virtual Patch</td>
<td>Yes (most vendors)</td>
<td>No</td>
</tr>
<tr>
<td>Out of the box protection</td>
<td>Yes</td>
<td>No – well sort of</td>
</tr>
<tr>
<td>Fast deployment</td>
<td>No / Yes (depends)</td>
<td>Yes</td>
</tr>
<tr>
<td>Alerts and Escalation</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Storage</td>
<td>Yes / No (depends)</td>
<td>Yes</td>
</tr>
<tr>
<td>Management of rules/reports/storage</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Management of issues located</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Dashboard – Plasma screen!!!</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Real time monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security of Audit</td>
<td>Yes</td>
<td>Maybe</td>
</tr>
<tr>
<td>Support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>License cost</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Extra Hardware / Software</td>
<td>Yes</td>
<td>Maybe</td>
</tr>
</tbody>
</table>
Options Compared – Commercial / Free

<table>
<thead>
<tr>
<th>Option</th>
<th>Commercial</th>
<th>Free (Database)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-Day Detection</td>
<td>Yes (open to debate)</td>
<td>No <em>(We can implement)</em></td>
</tr>
<tr>
<td>Detailed Reports</td>
<td>Yes</td>
<td>No <em>(We can write them)</em></td>
</tr>
<tr>
<td>Virtual Patch</td>
<td>Yes (most vendors)</td>
<td>No <em>(We can create simple)</em></td>
</tr>
<tr>
<td>Out of the box protection</td>
<td>Yes</td>
<td>No – well sort of</td>
</tr>
<tr>
<td>Fast deployment</td>
<td>No / Yes (depends)</td>
<td>Yes</td>
</tr>
<tr>
<td>Alerts and Escalation</td>
<td>Yes</td>
<td>No <em>(We can write and create)</em></td>
</tr>
<tr>
<td>Storage</td>
<td>Yes / No (depends)</td>
<td>Yes</td>
</tr>
<tr>
<td>Management of rules/reports/storage</td>
<td>Yes</td>
<td>No <em>(Harder to make simple)</em></td>
</tr>
<tr>
<td>Management of issues located</td>
<td>Yes</td>
<td>No <em>(harder again but can do)</em></td>
</tr>
<tr>
<td>Dashboard – Plasma screen!!!</td>
<td>Yes</td>
<td>No <em>(can create simple with SQL)</em></td>
</tr>
<tr>
<td>Real time monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security of Audit</td>
<td>Yes</td>
<td>Maybe <em>(not by default)</em></td>
</tr>
<tr>
<td>Support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>License cost</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Extra Hardware / Software</td>
<td>Yes</td>
<td>Maybe</td>
</tr>
</tbody>
</table>
## Local / Remote Comparison

<table>
<thead>
<tr>
<th>Option / Issue</th>
<th>Commercial</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trail storage</td>
<td>Remote</td>
<td>Local or Remote</td>
</tr>
<tr>
<td>Rule storage</td>
<td>Remote (but can be part local)</td>
<td>Local</td>
</tr>
<tr>
<td>Collection of raw date</td>
<td>Remote (can be local)</td>
<td>Local</td>
</tr>
<tr>
<td>Process / analysis</td>
<td>Remote</td>
<td>None</td>
</tr>
<tr>
<td>Alerts / issues</td>
<td>Remote</td>
<td>None</td>
</tr>
</tbody>
</table>
Comparison Conclusions

• Commercial gives some value “Out of the box”
• Commercial gives management, alerts, rules “Out of the box”
• Free core database options have no extra license cost; commercial can be expensive
  • Both would have ongoing costs to configure and maintain
• There are areas where commercial solutions have issues; network sniffing can miss something
• Both need to be customised – so does core solutions
• The same design process should be followed for any solution – “commercial or free”
Solution Requirements (assuming Core Database)

- Privileged user (SYSDBA/SYSOPER/SYSASM) access audit
- Startup/shutdown - Mandatory audit
- System activity - Logs – listener, alert,...
- Capture privilege use (DDL)
- Third party access and actions – proxy audit
- Data access / change
- Audit security controls
- Audit the audit itself
- Static data and configuration data
- Connections by all users
- Attacks (CPU’s, 0-days...)
- **All of these should not cause performance or functional issues**
Mechanical Process To Walk Through

- Take “What do I want to know” (Rules for set up)
- Design and deploy settings needed (+ code)
- Filter results / store / protect audit data and audit settings
- Produce reports – general reports
- Send alerts – These are specific “I really want to know”
- Manage raw data
- Manage extracted summary data
- Understand risk in solution choices – local vs remote
Security of Audit

- Protect the audit trail with audit - next
- Protect all audit trails – limit access
- Protect any audit functionality used – test it
- Control audit privileges granted – test
- Potential privileges – grant any privilege, grant any object privilege, alter user, create any procedure…
Audit of Audit

• Audit audit trail
• Audit FGA
• Audit triggers used
• Audit the audit system privileges
• Audit alter session, system, database
• Audit shadow tables used – if any
• Identify all storage used, controls, PL/SQL functions and audit them as well
Audit of Security Controls

• Use audit facilities to audit security controls implemented in the database
• Identify all security controls that are enabled
  • VPD, system privileges, roles, secure application roles, parameters, schemas
  • Data level controls and grants
• Enable audit controls on the security controls identified and privileges used
• The solutions can be varied but layered i.e.
  • E.g. : Parameter change: Audit ALTER SESSION, ALTER SYSTEM and use a system startup/shutdown trigger
Customising Identity Example

```sql
SQL> select client_identifier from v$session
    2     where sid=(select distinct sid from v$mystat);
CLIENT_IDENTIFIER
-------------------------------------------------------------

SQL> exec dbms_session.set_identifier('Hack the planet!');
PL/SQL procedure successfully completed.
SQL> select client_identifier from v$session
    2     where sid=(select distinct sid from v$mystat);
CLIENT_IDENTIFIER
-------------------------------------------------------------
Hack the planet!

SQL> alter user orascan identified by orascan;
User altered.
SQL> select client_id from dba_audit_trail
    2     where sessionid=1854567;
CLIENT_ID
-------------------------------------------------------------
Hack the planet!
```

Each user must have an identity that can be relied upon.
Must be set on logon (before logon ideally).
Blank identity is suspicious.
Needs to be set to a trusted non-guessable value by the client (application not your customer).
The value shown here is the only modifiable field that permeates to core audit.
Correlation

We can see correlation between core audit, logon audit, error audit, system triggers, custom code...

We could extend to others such as FGA, logs, more

We need identity to show who did it
SQL Injection and 0-Days - 1

Driven by simple DDL system error trigger

Audit table needs to be protected

Trigger could also detect ‘--’ or similar in code that is caught
Trigger can be much more sophisticated, parse out ‘--’ or special chars or certain views or...

Some errors do not fire the trigger such as 1756; the logging is different
Virtual Patching

- Detects an attack to an un-patched system and blocks the attack
- An error trigger can be extended
  
  If(ora_server_error(i)='1789') then
  log the event
  kill the session

- Detect use of vulnerable package – cpu fixed, dangerous packages (File, OS, Network...)
  Audit all on utl_file
  Audit all on xdb.xdb_pitrig_pkg

- Or create dummy package to allow actions to be blocked
Virtual Patch Dummy

Create a copy of a vulnerable package from a CPU or other attack.

Call through to the real package

Implement context based test for attacks and log or kill

Kill is not simple as cannot kill current session but possible via job or daemon. Can also take other actions, lock, rollback...

Murder is possible, suicide is not
As you can see, we can turn off audit that is enabled.
We captured the audit turn off by auditing “noaudit”.

At this point we would assume an attack so would need to detect this.

One way is to use the “dummy” trick which will capture the action.

A good solution is to have multiple audit for correlation and also detect changes to audit.

We can “poll” the settings and stability of the setup and also include a DDL trigger to prevent the change and also control privilege use.

We can also react by checking redo.
Trick Set Up

Does one of 4 things:
1) Allow context based core audit
2) Allow performance improvements if written to OS
3) Allows correlation with second source
4) Can be used for before and after but very limited

No additional Database DML
Trick 2

To improve performance we used a trick to create core audit from a trigger.

The trigger has little impact when the “when” clause is not TRUE

Writing to core audit saves database writes to a log table

When OS audit used best improvement

Can use utl_file and write before / after to a file and link file in client_id

In ideal world we need to retain the current “client id” for identity and add a file reference

Also the call to “dummy” needs to identify true source
Reporting And Alerts

• Starting with Audit in the database leads to simpler “to write” reports – we can use SQL
• Reports should be reports and alerts
  • Poll fast for changes to the security/audit
  • Poll fast for key issues and/or send alerts from triggers
  • Poll slower for summary of all checks
• Overnight reports on issues located
• Include sizing, management

Can use cron to poll, or dbms_scheduler. We can also get better performance with OS and SYSLOG audit but harder to write
Conclusions

• We must understand what is needed before deciding on a solution; people often decide on solutions first – this is wrong
• Practical for me means “risk” vs “cost” – i.e. Low Total cost
• Often risk can be mitigated with security on audit and audit on security and audit on audit
• Audit does not work fully unless identity is also included
• Audit does not work unless you can rely on the audit being not being changed – or at least you can detect the change and react
Questions?

Any Final Questions?
Building Practical Audit Trails

Design easy to implement and use audit trails