Software INTERNATIONAL Concepts

Worldwide
Managed Services for
OpenVMS and Rdb

Rdb Security

Keeping the bad guys out and the auditors happy

Bryan Holland

Software Concepts International, LLC
402 Amherst Street, Suite 300
Nashua, NH 03063, USA

Phone: 603-879-9022
e-mail: holland@sciinc.com
www.sciinc.com

Agenda

- Why Security
- Basics of Security
- Protection
- Encryption
- Auditing

About SCI

- Located in Nashua, NH (USA)
  (7 exits North of Oracle Rdb NEDC)
- Provider of (remote) Managed Services
  - Rdb database administration
  - OpenVMS system administration
- Worldwide support
- Established in 1987 (22+ years)

Evolution of a database…

1. Logical/Physical database design
2. Performance
3. Security

Why security, why now?

- Computing has become much more "connected" and decentralized
  (risks have expanded)
- Privacy Laws are more stringent
  (and punitive)
- Security breaches are frequent news stories
  (and nobody wants *their* name in *that* story)

Sampling of laws…

- Health Insurance Portability and Accountability Act (HIPAA)
- Sarbanes-Oxley Act (SOX)
- Payment Card Industry (PCI)
- European Union Data Protection Directive [European Union]

(over 50 countries have privacy laws)

Australian Privacy Laws

- Information Privacy Principles (IPPs) [Government]
- National Privacy Principles (NPPs) [private sector]
- Part IIIA of the Privacy Act
- Etc…

You are responsible.

- Businesses are responsible for maintaining “adequate” levels of protection and control for access to and destruction of personal data.
- In the event of a security breach, you may have to disclose the extent of that breach.

Steps to a secure db…

1. Secure the physical environment
2. Secure the Network
3. Secure the Operating System
4. Secure the Application
5. Secure the database
6. Audit, audit, audit…

The Basics

- Avoid the use of “generic” or shared accounts.
- Identify tables and columns that contain personal, financial or confidential data
- Identify all db access methods used:
  - Local applications, invoked directly by users?
  - OLTP servers (ACMS, Tuxcedo)
  - SQL/Services, JDBC, OCI
  - Remote Servers

The Basics

- Disable or remove access methods NOT used by your application
- Disable or remove inactive accounts

Think “roles”, not people…

- Base security definitions on *roles* (functions) rather than *people* (accounts/uic)
  This provides greater flexibility and requires less maintenance.
- Create VMS identifiers for each role and grant to them specific accounts that perform those roles.
- Grant access to objects via identifiers, *not* account/uic.

Authentication

- How do users authenticate to the database?
  - As individual users?
    - Security & auditing may be enforced/implemented at the database level.
  - As an “application” user?
    - Security & auditing is largely the responsibility of the application.
    - If the application functions are well defined and isolated by user, the db security model can restrict access.

Privileges

- Goal: Users should be granted the least privilege/access required to perform their work.

VMS Override Privileges

Certain OpenVMS privileges (SYSPRV, BYPASS) override database protections.

Therefore:

- Carefully limit who has these privileges
- Use “compensating controls” to limit risk.

Database Override Privileges

- The database privilege, DBADM, overrides all database data access.
- The database privilege, RMU$ALL, overrides all RMU protections

Therefore:

  - Carefully limit who has these privileges
  - Use “compensating controls” to limit risk.

Protecting objects

Ownership: Files and directories should be owned by an identifier, not an account (uic).

Access: Grant access to objects (files, database, tables, etc.) using ACLs. (Avoid “world” or “public” access to objects.)

What to protect:

- All database files and their directories
- RMU Commands
- DML (SQL) access
- Service configuration files

Protecting Database files:

- Root file (read to directory)
- Storage Areas
- Snapshot Files
- Row Cache Backing files
- RUJ files
- AIJ files
- Backup files (database & AIJ)
- Sort/work directories

Rdb File protections

	Privileges	
Object	Privilege needed	
Root file directory	read	
Root file	none	
Storage area file directory	read	
Storage area	none	
Snapshot file directory	read	
Snapshot file	none	
Row Cache Backing file dir	none	
Row Cache Backing file dir	none	
RUJ directory	READ+WRITE	
RUJ files	O:RW set by Rdb	
AIJ directory	READ	
AIJ files	none	
DB & AIJ backup directory	none	DBA needs R+W
DB & AIJ backup files	none	DBA needs ALL

Service configuration files:

- SQL Services configuration files
- OCI configuration files
- JDBC configuration/startup files.

"Services" can be configured to impersonate other users. Therefore, protecting the configuration of these services is an important part of database security.

Protecting RMU

- Certain RMU commands have the ability to do great good…or harm
- Others provide the ability access sensitive information

Protecting access to RMU clearly makes sense!

Protecting RMU

Show privileges

```
$ rmu show/privilege <root>
```

Set privileges

```
$ rmu/set priv <root> –
  /acl=(id=<identifier>,
  access=<privilege>) <root>
```

Protecting SQL access

- Database
- Tables
  - SELECT access to a table grants retrieval to *all* columns in that table.
- Columns (update or reference)
- Views (may restrict select of columns – or rows)
- Functions, procedures, modules, sequences

Encryption…

- Imagine…
  1. If the media containing your Rdb backups was stolen.
  2. Disks containing your Rdb databases failed – and were sent for repair – you don't have control of the media.
  3. Your system administrator (or DBA) were to take disk-images of your Rdb database offsite (where auditing and security controls don't exist)

Encryption on OpenVMS

- OpenVMS provides full encryption services
  - Used by DCL ENCRYPT and BACKUP commands
  - Used by RMU/BACKUP
  - Can be called by applications

Why encrypt?

- Provides a way to make the data unusable without a valid “key”.
- Can be used to protect data from their maintainers (if the key is not known to the DBA or System admin, the files are not usable when offsite).

What to encrypt?

- Rdb (v7.2) database backups
- Rdb (v7.2) AIJ backups
- Rdb columns via user-written functions based on VMS encryption services
  (see RDB_CYPHER.B32 from SQL$SAMPLE as a starting point)
- VMS (v8.3) backups (BACKUP/ENCRYPT)
- VMS (v8.3) “sensitive files” (DCL ENCRYPT)

Future versions of Rdb may include additional encryption options.

Encryption examples:

```
$ rmu/backup/encrypt=(value="Please don't share this")-
   <root> <backup-file>
```

Problem: How do you protect the key (not include in a command procedure) when invoking from batch?

Solution:

```
The Security officer uses the VMS "encrypt" utility to
   create a shared key:
$ ENCRYPT/CREATE_KEY/SYSTEM DBBACKUP_092 –
"Please don't share this")

$ RMU/BACKUP/ENCRYPT=NAME=DBBACKUP_092 –
<root> <backup-file>
```

About keys…

- If you loose the key, you loose your data! (There is no “back door” to the encryption services)
- Don’t store your keys in command procedures (along with the access control strings you use to remotely access systems)
  - Use “named keys” instead.

Auditing

Uses the OpenVMS Audit Server

- Saves “audit” events in the VMS audit file in binary format.
- Sends audit ALERTS to security operator terminals.

Where is the VMS audit log?

```
$ SHOW AUDIT/JOURNAL
```

Auditing

What is the current state of db auditing?

```
$ rmu/show audit <root> -
   /rmu/prot/daccess=(database,table,column)
$!– or –
$ rmu/extract/item=security <root>
```

Changes to auditing are performed with:

```
$ rmu/set audit <root> auditing attributes
```

Auditing RMU

Current state of RMU auditing:

```
$ RMU/SHOW AUDIT <root>/RMU
```

Enabling auditing of RMU access:

```
$!-- Audit RMU commands that attach to a database
$ RMU/SET AUDIT/TYPE=AUDIT/ENABLE=RMU <root>
$!–- Backups, analyze, verify not worth ALARMS…
$ RMU/SET AUDIT/TYPE=ALARM/DISABLE=RMU <root>
```

Auditing Protection changes

If someone is given access to your most sensitive data, wouldn't you want to be the first to know?

Current state of protection auditing:

```
$ RMU/SHOW AUDIT <root>/PROTECTION
```

Enabling auditing on protection changes:

```
$!-- Audit changes to database protections
$ RMU/SET AUDIT/TYPE=AUDIT/ENABLE=PROTECTION <root>
$!– Hopefully infrequent, and may be critical – ALARM
$ RMU/SET AUDIT/TYPE=ALARM/ENABLE=PROTECTION <root>
```

Auditing Auditing

If auditing were suddenly stopped,
wouldn't you want to be the first to know?

While AUDIT class cannot be disabled, no audit records or alarms are produced while auditing is STOPPED.

The following does not do anything:

```
$ RMU/SET AUDIT/TYPE=AUDIT/[enable|disable] <root>
$! The above command has no effect because the AUDIT
$! Class is always enabled.
```

Auditing Auditing

The audit of audit changes (audit class) can be disabled by stopping all auditing:

```
$ RMU/SET AUDIT <root>/STOP ! Audits and alarms
$! Or just audits…
$ RMU/SET AUDIT <root>/STOP/TYPE=AUDIT
$! Or just alarms
$ RMU/SET AUDIT <root>/STOP/TYPE=ALARM
```

Discretionary Auditing

"Discretionary" does not mean "optional". It refers to the OpenVMS Discretionary Access Control (DAC) system. Essentially, everything that requires a privilege check can be audited with discretionary (DACCESS) auditing.

Since a privilege check is made for essentially all access to data, this provides a useful way to know who is doing what to your data.

Discretionary Auditing

```
$!-- Audit access to protected objects (such as
   databases, tables, columns…)
$ RMU/SET AUDIT <root> /TYPE=AUDIT/ENABLE=DACCESS
$!–- Enable ALARMS ONLY if you have
$!   specific requirements
$ RMU/SET AUDIT <root> /TYPE=ALARM/DISABLE=DACCESS
$
$!-- Define who gets audited (in this case PUBLIC)
$ RMU/SET AUDIT <root> /ENABLE=IDENT=("[*,*]")
$
$ RMU/SET AUDIT <root> /TYPE=AUDIT -
        /ENABLE=DACCESS=SCHEMA/PRIV=(ALL)
```

Discretionary Auditing

```
$!– Enable auditing for critical tables
$ RMU/SET AUDIT/TYPE=AUDIT -
      /ENABLE=DACCESS=TABLE=(<table-name-list>) –
      /PRIV=(priv-list) –
      <root>
$!– Enable auditing for critical columns
$ RMU/SET AUDIT/TYPE=AUDIT –
      /ENABLE=DACCESS=COLUMN=(<table-name.column>) –
      /PRIV=(priv-list) –
      <root>
$!-- Start audits & alarms
$ RMU/SET AUDIT/TYPE=AUDIT/START <root>
$ RMU/SET AUDIT/TYPE=ALARM/START <root>
```

Analyzing Audit data

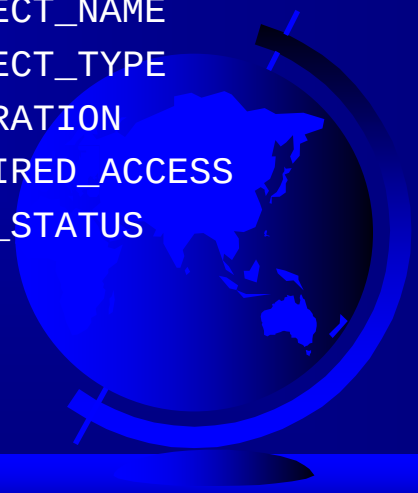
Audit records can be extracted from the audit journal and loaded into an Rdb database:

```
$ rmu/load/audit=database=<audited-db> -
   <db-to-load> <your-audit-table> <VMS-audit-file>
```

Note: <your-audit-table> will be created if it does not already exist

The “audit table”

```
Columns for table AUDIT_RECORDS (your-audit-table):
Column Name                    Data Type        Domain
-----------                    ---------        ------
AUDIT$EVENT                    CHAR(16)         AUDIT$EVENT
AUDIT$SYSTEM_NAME              CHAR(15)         AUDIT$SYSTEM_NAME
AUDIT$SYSTEM_ID                CHAR(12)         AUDIT$SYSTEM_ID
AUDIT$TIME_STAMP               CHAR(48)         AUDIT$TIME_STAMP
AUDIT$PROCESS_ID               CHAR(12)         AUDIT$PROCESS_ID
AUDIT$USER_NAME                CHAR(12)         AUDIT$USER_NAME
AUDIT$TSN                      CHAR(25)         AUDIT$TSN
AUDIT$OBJECT_NAME              CHAR(255)        AUDIT$OBJECT_NAME
AUDIT$OBJECT_TYPE              CHAR(12)         AUDIT$OBJECT_TYPE
AUDIT$OPERATION                CHAR(32)         AUDIT$OPERATION
AUDIT$DESIRED_ACCESS           CHAR(16)         AUDIT$DESIRED_ACCESS
AUDIT$SUB_STATUS               CHAR(32)         AUDIT$SUB_STATUS
continued…
```



The “audit table”

```
Columns for table AUDIT_RECORDS (your-audit-table):
Column Name                    Data Type        Domain
-----------                    ---------        ------
…continued
AUDIT$FINAL_STATUS             CHAR(32)         AUDIT$FINAL_STATUS
AUDIT$RDB_PRIV                 CHAR(16)         AUDIT$RDB_PRIV
AUDIT$VMS_PRIV                 CHAR(16)         AUDIT$VMS_PRIV
AUDIT$GRANT_IDENT              CHAR(192)        AUDIT$GRANT_IDENT
AUDIT$NEW_ACE                  CHAR(192)        AUDIT$NEW_ACE
AUDIT$OLD_ACE                  CHAR(192)        AUDIT$OLD_ACE
AUDIT$RMU_COMMAND              CHAR(512)        AUDIT$RMU_COMMAND
```

The “audit table”

```
Columns for table AUDIT_RECORDS (your-audit-table):
Column Name                    Data Type        Domain
-----------                    ---------        ------
…continued
AUDIT$FINAL_STATUS             CHAR(32)         AUDIT$FINAL_STATUS
AUDIT$RDB_PRIV                 CHAR(16)         AUDIT$RDB_PRIV
AUDIT$VMS_PRIV                 CHAR(16)         AUDIT$VMS_PRIV
AUDIT$GRANT_IDENT              CHAR(192)        AUDIT$GRANT_IDENT
AUDIT$NEW_ACE                  CHAR(192)        AUDIT$NEW_ACE
AUDIT$OLD_ACE                  CHAR(192)        AUDIT$OLD_ACE
AUDIT$RMU_COMMAND              CHAR(512)        AUDIT$RMU_COMMAND
```

Exporting audit data

For long term storage or for input into external auditing systems, the audit data can be unloaded into a portable format:

Create an XML document

```
$ RMU/UNLOAD <root> /RECORD=FORMAT:XML <your-audit-table> -
    <output-file>
```

Or create a CSV format file

```
$ RMU/UNLOAD <root> /RECORD=FORMAT:DELIMITED <your-audit-table> -
    <output-file>
```

Other audit trails…

- After Image Journals
- Rdb monitor logs
- OpenVMS accounting files
- Application/Service logs
  - SQLserver

Other audit trails…

- After Image Journals contain a complete record of all changes made to the database – including made the changes and when. Enabling the logminer feature provides additional information that is helpful in an audit.

Other audit trails…

- The Rdb monitor log files
  - Attach
    - time & status, Type of access (utility or application), PID, stream-id, username, process name, Image,
  - Detach time & status
  - This is very helpful in determining what processes where accessing a database during a certain time window – and the specific image that they were running

Other audit trails…

- VMS accounting files
  - When a process (or image) started and ended
  - Final completion status
  - Mode
  - Privilege masks
  - Remote node/user info
  - Input device (terminal, mailbox)
  - Queue info (BATCH)

Monitoring changes…

Once a secure environment has been established, monitor for changes.

Create security “reference files” -- files that contains the output from known security settings. This allows you to compare the current settings with the “verified settings”.

Monitoring changes
(using “reference files”)

Create reference files from the following:

- `$ rmu show/priv <root>`
- `$ rmu/extract/item=security <root>`
- `$ rmu/extract/item=protection <root>`
- `$ dir/sec <key files>`
- Copy of SQL configuration file
- Copy of OCI Configuration file

Software INTERNATIONAL Concepts

Worldwide Managed Services for OpenVMS and Rdb

Questions & Answers

Software Concepts International, Inc.
Worldwide Managed Services for OpenVMS and Rdb