MiP

PUG CHALLENGE

Layered Defence

A real-world approach to OpenEdge database and data security

Pieter J. Meyer

SPEAKER

Pieter J. Meyer

32+ years Progress OpenEdge experience,
from Progress v6 to OpenEdge 12.8

28+ years at MIP
responsible for the oversight of
Software Development,
Security and
Technical Operational Standards
across all teams



Head of DevOps
MIP Holdings (Pty) Ltd

pm@mip.co.za
linkedin.com/in/pietermeyer1

PUG CHALLENGE

Is your security strategy truly protecting your information

— or —

is it just an auditor-pleaser ?



MiP

PUG CHALLENGE

Data Security

Protecting the records themselves

Confidentiality of the information at rest and in transit

Survives a perimeter breach – the attacker gets the file, not the content

Owned by: encryption, key management, access to the data itself

Database Security

Protecting the infrastructure

Integrity of the environment the data lives in

Who may connect, with what rights, and what gets recorded

Owned by: hardening, authentication, auditing, backup and recovery

WHAT WE'LL COVER

PUG CHALLENGE

[01]

The Gold Coins

Securing the data

Transparent Data Encryption

HSM key management

[02]

The Bank Vault

Securing the database

Hardening (4GL and SQL)

OE Auditing

[03]

The Armoured Car

Securing the continuity

Secure backups

OE Replication & After-Imaging

Data in transit

MiP

PUG CHALLENGE

Security is broad.

This talk stays inside the OpenEdge database.

Outside the scope:

- **Network and infrastructure** – firewalls, segmentation, WAF, OS patching
- **Identity and access** – MFA, SSO, directory integration, password policy
- **Application security** – code review, input validation, session handling
- **Compliance mapping** – POPIA and GDPR clause by clause

All of it matters. None of it helps once the database file is in someone else's hands.

PUG CHALLENGE

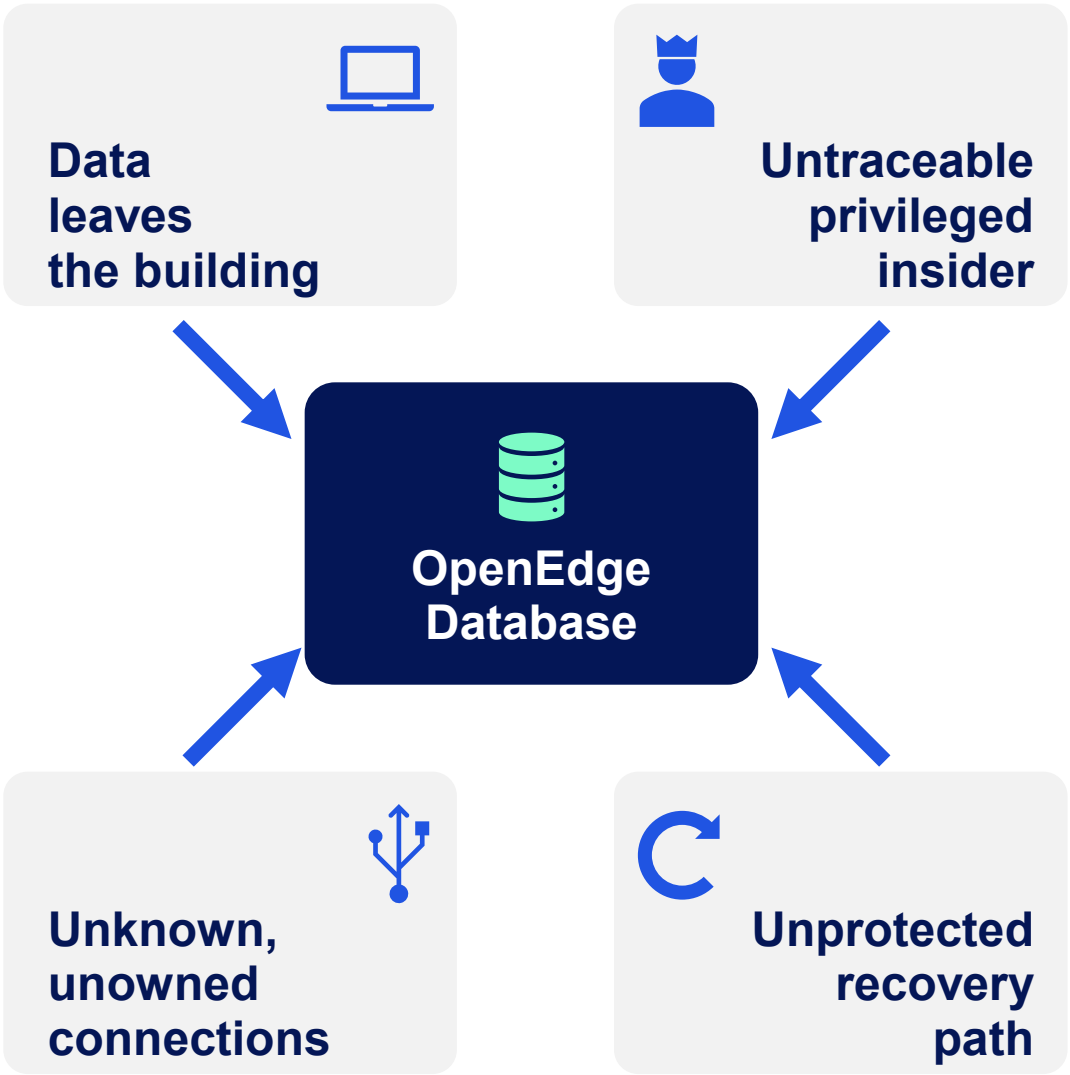
What are we actually defending against?

A copy of the database or a backup leaving the building – laptop, tape, cloud bucket, dev refresh

A privileged insider doing something nobody can later prove or disprove

A SQL or 4GL connection nobody knew existed and nobody owns

A disaster where the recovery path is itself unprotected



PUG CHALLENGE

COMMON MYTHS THAT LEAVE GAPS

[01]

"The firewall handles it"

Stops connections, not copies.

A non-production refresh copy on a secondary server.

[02]

"Encryption is too slow"

Database server engine level

[03]

"We have application security"

Authentication for default interface only.

[04]

"We back up nightly"

Untested means unproven.

MiP

PUG CHALLENGE

[01]

The Gold Coins

IN THIS LAYER

- Transparent Data Encryption
- HSM key management

Securing the Data

MiP

PUG CHALLENGE

Your data is more exposed than you think

Every `.d` dump, `.bup` and AI extent is a full copy of production

OS-level access bypasses every `_User` entry you have

Cloud, SAN snapshots and offsite tape multiply the number of copies

customer.d — 418,229 records

```
$ cat /backup/0204/customer.d

"10492" "Mrs A Botha"
  "8203145112083“ “2001 Penny Lane”
  "4539 8812 7745 9001" 48210.55
"10493" "T Nkosi"
  "9108230045098“ “19 Main Street”
  "5221 0043 9910 7788" 7725.00
"10494" "D Pillay"
  "7604015508082“ “7a 25th Avenue”
  "4024 0071 2250 3391" 119640.80

-- 3 of 418,229 records --
```

No login. No _User check. Full production data.

PUG CHALLENGE

What TDE is

At-rest protection

Encrypts blocks as they are written – database, AI, BI, backups

Transparent to ABL and SQL – no code change

Granular: choose tables, indexes and LOBs to encrypt

License prerequisites apply

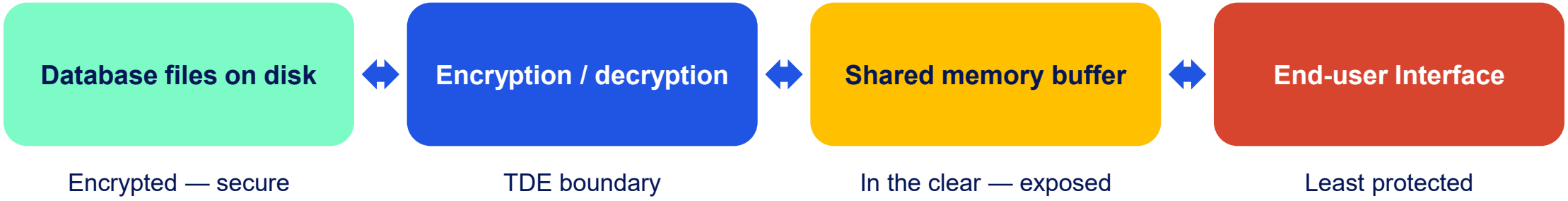
What TDE is not

Misconceptions to kill early

Not protection against an authorised user reading through the app

Not a substitute for access control

Not in-transit encryption



PUG CHALLENGE

Keystore

The database itself

Administration

- Cipher choice
- Passphrase handling
- Manual vs auto-start

Rotation — keeping the keys honest

- Rotate periodically
- Who holds the passphrase

Considerations

- Determine the trade-off
- Rotation of keys on secondary environments

Policies

The objects inside it

Administration

- Cipher choice (AES-CBC)
- Object-level policy (table, index, lobs)

Encrypt

- Update existing objects

Considerations

- Review the defaults and change
- Verify periodically

PUG CHALLENGE

Ciphers

One ID space covers the master key, object, binary-dump and BI/AI ciphers — AES CBC 256 (ID 3) is the practical default.

ID	Cipher	Mode	Size	Key type	DB Master key	Object	Binary dump	BI/AI
0	NULL	NULL	–	–	–	X	–	–
1	AES	CBC	128	binary	X	X	–	–
2	AES	CBC	192	binary	X	X	–	–
3	**AES**	**CBC**	**256**	**binary**	**X**	**X**	**–**	**–**
4	DES	CBC	56	binary	X	X	–	–
5	DES3	CBC	168	binary	X	X	–	–
6	DES	CBC	56	PBE	X	–	X	–
11	AES	CBC	128	PBE	X	–	X	–
12	AES	CBC	256	PBE	X	–	X	–
—	AES	ECB	128	PBE	–	–	–	X
—	AES	ECB	256	PBE	–	–	–	**X**

Source: OpenEdge supported ciphers

PUG CHALLENGE

Define

Type II Area

```
# vi <db-name>-tde.st

e "Encryption Policy Area":61,32;64 /dbfiles/db/sports_61.d1
```

Add Extent

Add Off-Line/On-Line

```
prostrct add <db-name> <db-name>-tde.st
```

TDE - DATABASE KEYSTORE

Enable

Activate feature

```
proutil <db-name> -C enableencryption
                  [-Cipher cipher-number]
                  [-Autostart {admin|user}]
                  [-biencryption {enable|disable}]
                  [-aiencryption {enable|disable}]
                  [-Passphrase] [[-userid userid] [-password password]]
```

Manage

Passphrase & start-up mode

```
proutil <db-name> -C epolicy manage autostart { admin | user | disable }
                  [-Passphrase] [[-userid userid] [-password password]]
```

Verify

View status

```
proutil <db-name> -C epolicy view autostart
                  [-Passphrase] [[-userid userid] [-password password]]
```

TDE - OBJECT POLICIES

PUG CHALLENGE

Enable

Enable encryption

```
proutil <db-name> -C epolicy manage encrypt
                  [ table / index / lob ] {value}
                  [-Cipher cipher-number]
                  [-Passphrase] [[-userid userid] [-password password]]
```

Update

Update the object

```
proutil <db-name> -C epolicy manage update
                  [ table / index / lob ] {value}
                  [-Cipher cipher-number]
                  [-Passphrase] [[-userid userid] [-password password]]
```

Disable

Disable encryption

```
proutil <db-name> -C epolicy manage cipher
                  [ table / index / lob ] {value}
                  -Cipher 0
                  [-Passphrase] [[-userid userid] [-password password]]
```

Verify

View or scan

```
proutil <db-name> -C epolicy manage [ view / scan ]
                  [ table / index / lob ] {value}
                  [-Passphrase] [[-userid userid] [-password password]]
```

PUG CHALLENGE

Backup

Backup .ks file separate

```
cp <db-path>/<db-name>.ks <db-backup>.ks

probkup <db-name> <db-backup>.bup [options] -newinstance –Passphrase
```

Restore

Copy .ks file and restore

```
cp <db-backup>.ks <db-path>/<db-name>.ks
## Same Key – DR/Testing
prorest <db-name> <db-backup>.bup
## New Key – Secondary/Non-Production
prorest <db-name> <db-backup>.bup -newinstance
proutil <db-name> -C epolicy manage keystore rebind -Passphrase
```

Dump & Load

Binary dump and load

```
proutil db-name -C dump [owner-name.]<table-name> <path>
                [-Cipher cipher-id]
                [-Passphrase] [[-userid userid] [-password password]]

proutil db-name -C load <path>/<table-name>.bd
                [-Cipher cipher-id]
                [-Passphrase] [[-userid userid] [-password password]]
```

PUG CHALLENGE

What TDE costs, and where the risk moves

Performance: measured overhead in your own environment, not the marketing number

Start-up: the auto-start keystore convenience vs the security it gives away

Backups, AI and BI inherit encryption – and inherit the key dependency with it.
(No O/S level file compression.)

Schema changes: every new table, field or LOB needs the TDE policy applied

Recovery: an encrypted backup you cannot unlock is a lost backup



Software keystore

The default starting point

- Keystore file sits beside the database
- Passphrase and OS permissions only
- A single point of compromise
- DB hardening steps adds a secondary level

HSM keystore

Moving keys out of software

- Keys held in tamper-resistant hardware/location
- The DBA never holds key material
- Meets rules a key file cannot
- HSMs validated, cost/effort
- Connectivity dependencies

KEY MANAGEMENT

Separation grows with distance — so does dependency

O/S level

Keystore file on the database server

WHAT YOU GAIN

Nothing extra to buy or run
No start-up dependency

WHAT IT COSTS

File permissions are the only guard
The DBA can reach the key

Hardware level

Appliance HSM in your own data centre

WHAT YOU GAIN

Key never leaves the network
Meets rules a key file cannot

WHAT IT COSTS

Cost, and specialists to run it
Database start-up needs the HSM

Cloud level

Managed KMS or cloud HSM service

WHAT YOU GAIN

Someone else runs and audits it
Scales across sites for free

WHAT IT COSTS

Start-up depends on the connectivity
Key material sits outside your estate

Closer to the database: simpler, weaker separation

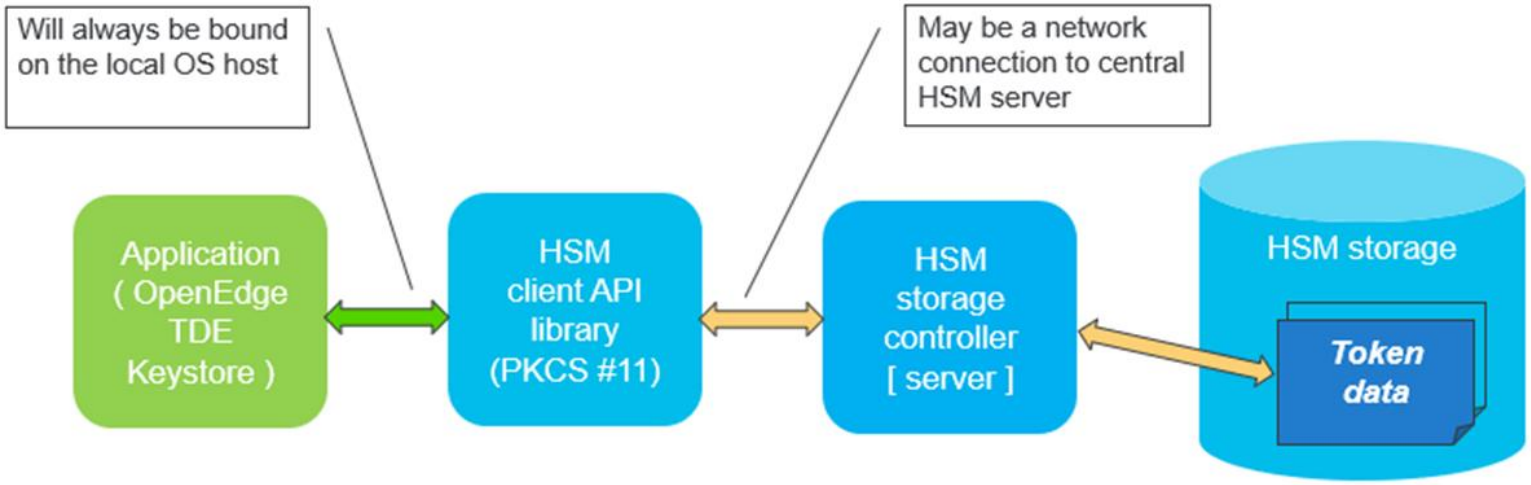
Further away: stronger separation, more dependencies

Hardware-backed key management

How OpenEdge talks to the HSM

What changes operationally:

- start-up,
- failover,
- DR site



What happens when the HSM is unavailable – plan this before you need it

PUG CHALLENGE

[01]

Encrypt the copies

Protects every copy at rest.

- Database extents
- Backups
- BI and AI files
- Binary dumps

Costs less than you expect.

Zero code change.

Minimum maintenance overhead.

[02]

Mind the keys

Keystore sits on the database volume.

Rotate keys on a schedule.

HSM secures the key itself.

DB Hardening adds a secondary access level.

[03]

Test the unlock

Restore to a clean host, keystore and all.

Test with the same key — DR and testing.

Test with a re-generated key — non-production.

PUG CHALLENGE

[02]

The Bank Vault

IN THIS LAYER

- Hardening (4GL and SQL)
- OE Auditing

Securing the Database

MiP

PUG CHALLENGE

What database security actually covers

Authentication: who is allowed to connect, and how they prove it

Authorisation: what they may do once connected – tables, fields, utilities

Configuration: the defaults that ship open and stay open

Accountability: what is recorded, and whether it can be tampered with



Every connection shows its credentials at the door

HARDENING

PUG CHALLENGE

[01]

Authenticate

Enable _User accounts

No shared accounts *

[02]

Authorise

Least privilege, ABL and SQL

[03]

Close defaults

Ports, utilities, shipped accounts

MiP

TWO FRONT DOORS

PUG CHALLENGE

4GL / ABL security

The door most teams look at

`_User`, domains, client-principal

Only as strong as the database or application security configured

SQL security

The door most teams forget

SQL default rights granted to connecting users

Grants that ignore your ABL rules

ODBC/JDBC clients bypass the app entirely

ONE BROKER OR TWO

OpenEdge defaults to one joined door

```
-ServerType both -S <port>
```

Split the doors

```
-ServerType 4GL -S <port-4gl>
-ServerType SQL -S <port-sql>
```

PUG CHALLENGE

Authenticate

Enable OE database security by creating _User records

No shared accounts *

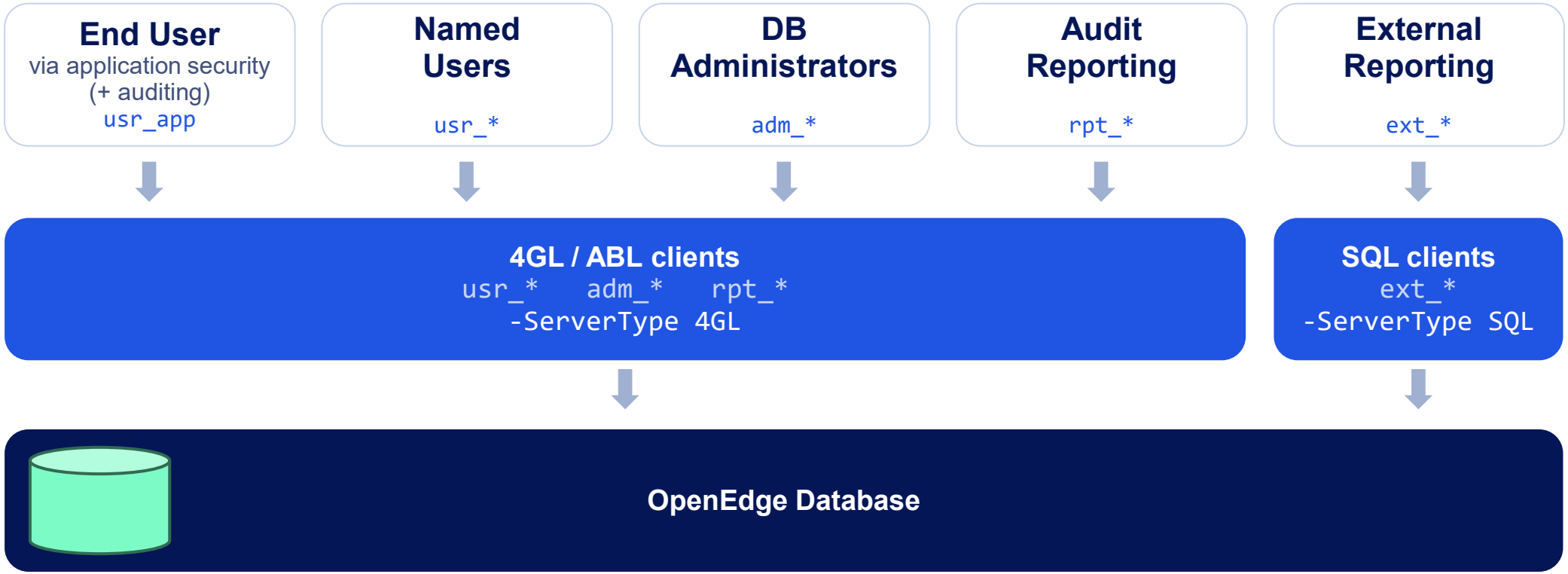
```
  +-------------------------------------------------------------------+
  | User ID        Domain Name    User Name         SQL Only  Password |
  + ------------  -------------  ---------------  --------  ----------+
  | adm_barry                     Barry Allen      no                   |
  | adm_bruce                     Bruce Wayne      no                   |
  | adm_clark                     Clark Kent       no                   |
  |                                                                    |
  | ext_ods_1                     ODS User 1       yes                  |
  | ext_ods_2                     ODS User 2       yes                  |
  |                                                                    |
  | rpt_dam                       Report Auditing  no                   |
  |                                                                    |
* | usr_app                       PAS Application  no                   |
  | usr_carol                     Carol Danvers    no                   |
  | usr_jean                      Jean Grey        no                   |
  | usr_jennifer                  Jennifer Walters no                   |
  |                                                                    |
  +-------------------------------------------------------------------+
```

PUG CHALLENGE

Authorise

The application connects once, on an embedded account.

Everyone else connects as themselves.



PUG CHALLENGE

Authorise 4GL

Configure security

Revoke all. Enable only required.

```
+------------------- Table Name: "order_line" ----------------------+
|    Can-Read: !,adm_*@*,usr_*@*,ext_*@*                            |
|   Can-Write: !,adm_*@*,usr_*@*                                    |
|  Can-Create: !,adm_*@*,usr_*@*                                    |
|  Can-Delete: !,adm_*@*,usr_*@*                                    |
|    Can-Dump: !,adm_*@*                                            |
|    Can-Load: !,adm_*@*                                            |
|                                                                   |
|  Examples:                                                        |
|    *                      - All users (login Ids) are allowed access.  |
|    <user>,<user>,etc.     - Only these users have access.         |
|    !<user>,!<user>,*      - All except these users have access.   |
|    acct*                  - Only users that begin with "acct" allowed. |
|  Do not use spaces in the string (they will be taken literally).  |
+-------------------------------------------------------------------+
```

PUG CHALLENGE

Authorise SQL

Remove defaults.
SYSPROGRESS

```
GRANT RESOURCE,DBA TO adm_barry; COMMIT;
GRANT RESOURCE,DBA TO adm_bruce; COMMIT;
GRANT RESOURCE,DBA TO adm_clark; COMMIT;
```

Restrict access.

```
REVOKE ALL ON "SYSPROGRESS"."SYSBIGINTSTAT"         FROM PUBLIC GRANTED BY ANY_USER; COMMIT;
REVOKE ALL ON "SYSPROGRESS"."SYSCHARSTAT"           FROM PUBLIC GRANTED BY ANY_USER; COMMIT;
REVOKE ALL ON "SYSPROGRESS"."SYSCHKCONSTR_NAME_MAP" FROM PUBLIC GRANTED BY ANY_USER; COMMIT;
```

Schema changes:
Re-apply the SQL security.

```
REVOKE ALL ON "PUB"."_ActAILog"  FROM PUBLIC GRANTED BY ANY_USER; COMMIT;
REVOKE ALL ON "PUB"."_ActBILog"  FROM PUBLIC GRANTED BY ANY_USER; COMMIT;
REVOKE ALL ON "PUB"."_ActBuffer" FROM PUBLIC GRANTED BY ANY_USER; COMMIT;
```

```
GRANT SELECT ON "PUB"."customer"   TO ext_ods_1; COMMIT;
GRANT SELECT ON "PUB"."order"      TO ext_ods_1; COMMIT;
GRANT SELECT ON "PUB"."order_line" TO ext_ods_1; COMMIT;
```

```
GRANT SELECT ON "PUB"."customer"   TO ext_ods_2; COMMIT;
GRANT SELECT ON "PUB"."order"      TO ext_ods_2; COMMIT;
GRANT SELECT ON "PUB"."order_line" TO ext_ods_2; COMMIT;
```

```
REVOKE RESOURCE,DBA FROM 'sysprogress’; COMMIT;
DROP USER 'sysprogress’; COMMIT;
```

PUG CHALLENGE

Recording what actually happened

Policy-driven: you choose the events, tables and fields that matter

Records the action, the actor, the before/after and the context

Tamper-evident – designed so the record cannot be quietly rewritten

The cost is storage and discipline, not application change

OE vs Application Auditing



Auditing writes down who did what, and when

PUG CHALLENGE

[01]

Define the policy

Events, tables, fields, users

Enable at database level

[02]

Capture the data

Written with the transaction

Sizing and growth

[03]

Report and archive

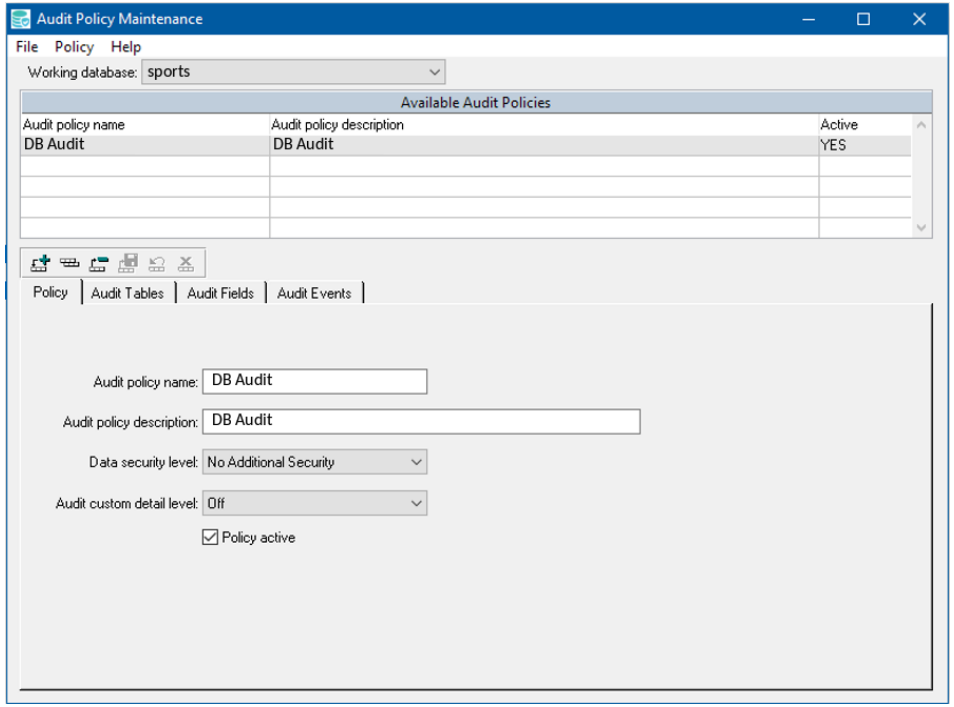
Query on demand

Archive as required

PUG CHALLENGE

Database events to be audited

- CUD of database user accounts
- CUD of SQL DBA database accounts
- CUD of database auditing configuration + enabling and disabling of auditing + runtime updates to auditing
- CUD of database roles
- CUD of database role assignments
- CUD of database tables, fields, indexes and database triggers
- CUD of database sequences
- Database index activation, deactivation, rebuilds and fixes
- Enabling and disabling of CDC
- Failed database user and SQL user login attempts
- Failed connection attempts
- etc.…



PUG CHALLENGE

Define

Type II Area

```
# vi <db-name>-dam.st

d "AuditIndex":81,1;64 /dbfiles/db/sports_81.d1
d "AuditData":80,64;512 /dbfiles/db/sports_80.d1
```

Add Extent

Add Off-Line/On-Line

```
prostrct add <db-name> <db-name>-dam.st
```

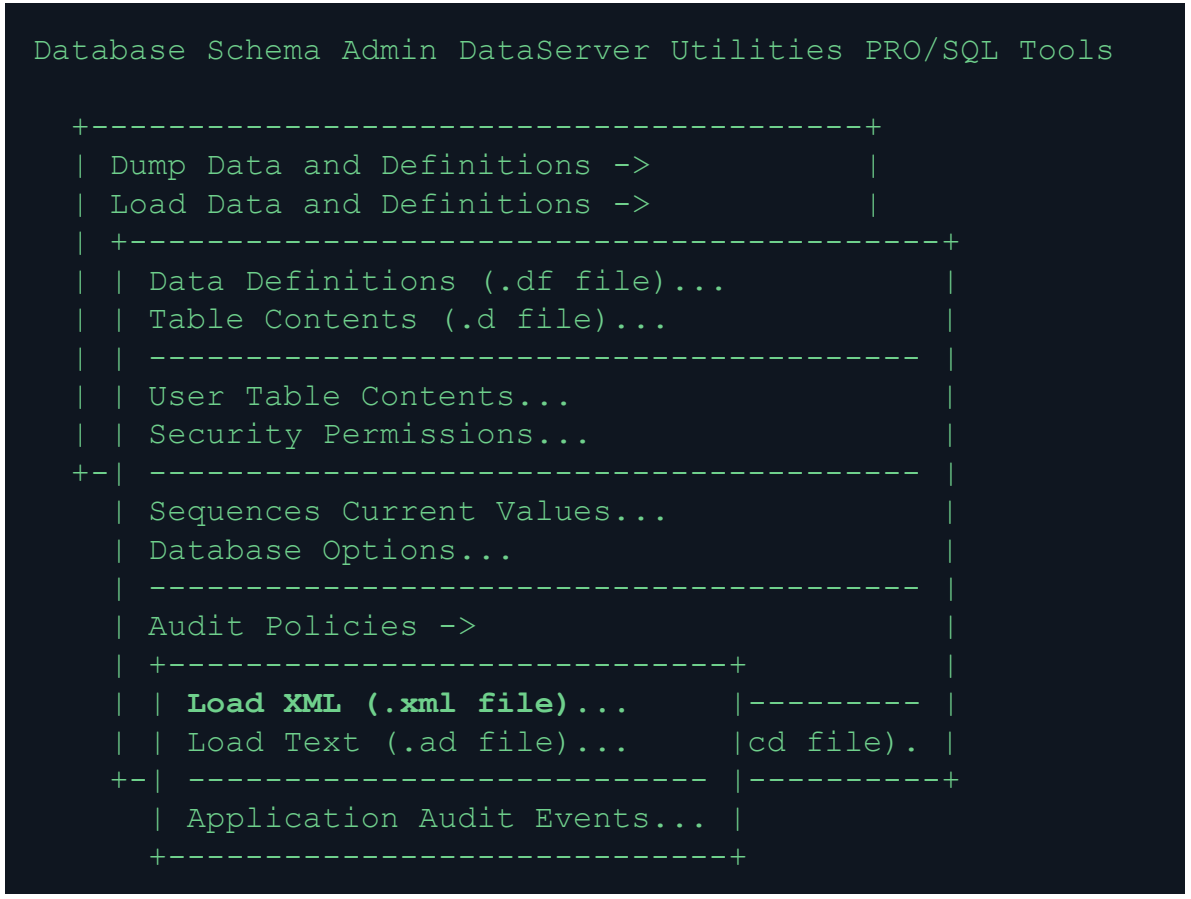
Enable

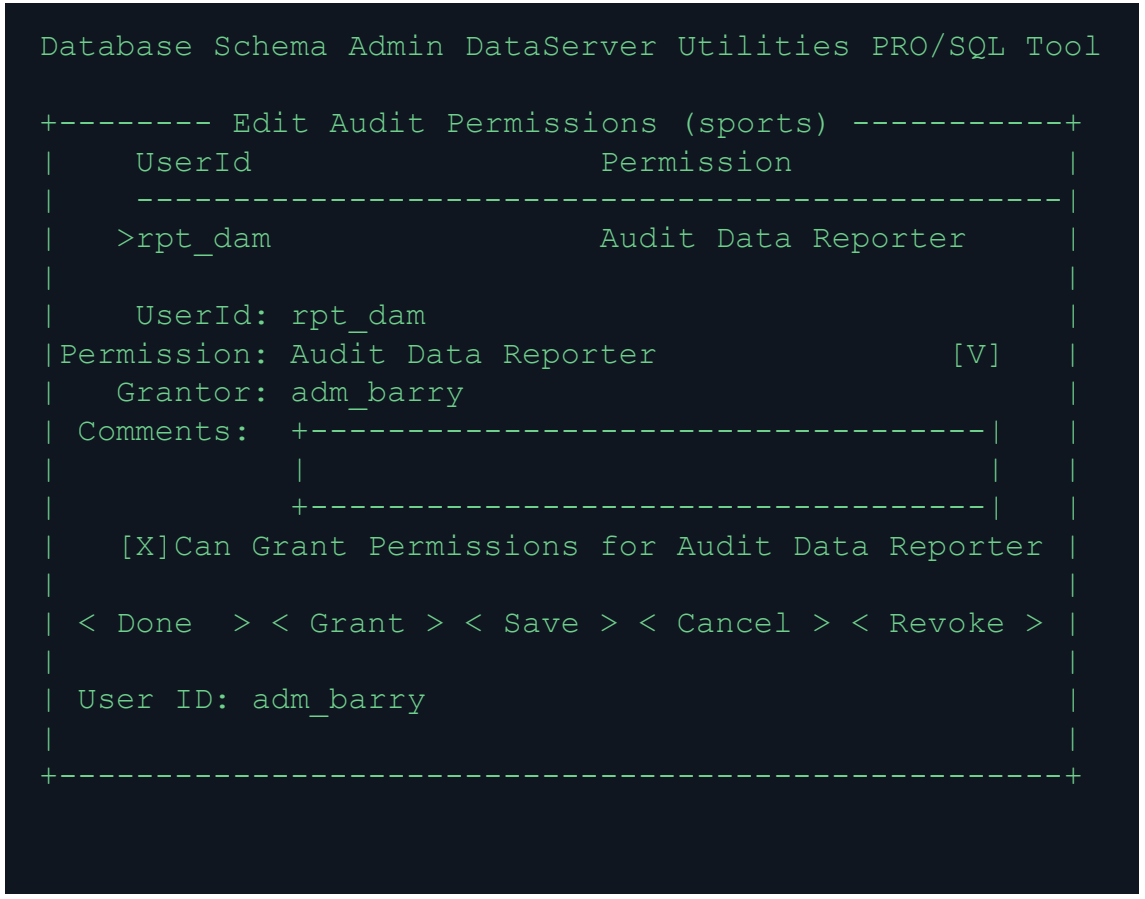
Activate feature

```
proutil <db-name> -C enableauditing area "AuditData" indexarea "AuditIndex"
```

PUG CHALLENGE

Policies





MiP

PUG CHALLENGE

Report

Audit report by rpt_dam user

Schedule report
Review report
Create alert mechanisms

Procedure - V:\shares\OE\DB\audit\dam.p

File Edit Search Compile Help

```
define variable cEvent as character no-undo.

for each _aud-audit-data
      by _aud-audit-data._audit-date-time desc:

    case _aud-audit-data._event-id:
        when 100   then cEvent = "User account created".          when 101   then cEvent = "User account updated".
        when 102   then cEvent = "User account deleted".          when 210   then cEvent = "SQL DBA created".
        when 211   then cEvent = "SQL DBA updated".               when 212   then cEvent = "SQL DBA deleted".
        when 400   then cEvent = "SQL table privilege created".   when 401   then cEvent = "SQL table privilege updated".
        when 402   then cEvent = "SQL table privelege deleted".   when 510   then cEvent = "Role definition created".
        when 511   then cEvent = "Role definition updated".       when 512   then cEvent = "Role definition deleted".
        when 515   then cEvent = "Role definition created".       when 516   then cEvent = "Role assignment updated".
        when 517   then cEvent = "Role assignment deleted".       when 5000  then cEvent = "Table created".
        when 5001  then cEvent = "Table trigger created".         when 5002  then cEvent = "Table field created".
        when 5004  then cEvent = "Table index created".           when 5010  then cEvent = "Table updated".
        when 5011  then cEvent = "Table trigger updated".         when 5012  then cEvent = "Table field updated".
        when 5014  then cEvent = "Table index updated".           when 5020  then cEvent = "Table deleted".
        when 5021  then cEvent = "Table trigger deleted".         when 5022  then cEvent = "Table field deleted".
        when 5024  then cEvent = "Table index deleted".           when 5025  then cEvent = "Table index field deleted".
        when 5030  then cEvent = "Sequence created".              when 5031  then cEvent = "Sequence updated".
        when 5032  then cEvent = "Sequence deleted".              when 10001 then cEvent = "Database auditing disabled".
        when 10010 then cEvent = "Audit policy runtime update".   when 10208 then cEvent = "Database index rebuild".
        when 10210 then cEvent = "Database index fix".            when 10215 then cEvent = "Database index activation".
        when 10216 then cEvent = "Database index deactivation".   when 10300 then cEvent = "Audit archive".
        when 10302 then cEvent = "Audit data load".               when 10304 then cEvent = "Audit policy load".
        when 10512 then cEvent = "User login failed".             when 10522 then cEvent = "SQL user login failed".
        when 10602 then cEvent = "Database connect failed".       when 10612 then cEvent = "SQL Database connect failed".
        when 11850 then cEvent = "Enable Change Data Capture".    when 11851 then cEvent = "Disable Change Data Capture".
    end case.

    disp _aud-audit-data._audit-date-time
         _aud-audit-data._event-id
         cEvent format "x(30)"
         _aud-audit-data._user-id
         _aud-audit-data._event-context format "x(20)" view-as fill-in size 30 by 1
        with scrollable size 200 by 10 view-as dialog-box three-d.
end.
```

PUG CHALLENGE

[01]

Lock the doors

Review shipped defaults.

Secure the 4GL with _User and domains.

Close SQL permissions — the most-missed gap.

Split the 4GL and SQL ports.

Update entity security restriction.

[02]

Audit on purpose

Audit what an investigation would need.

Audit the administrators too.

Audit data retention policies.

[03]

Report & Review

Run the reports on a schedule.

Report only accessed by a named owner.

Review, review, review.

PUG CHALLENGE

[03]

The Armoured Car

IN THIS LAYER

- Secure backups
- OE Replication & After-Imaging
- Data in transit

Securing the Continuity

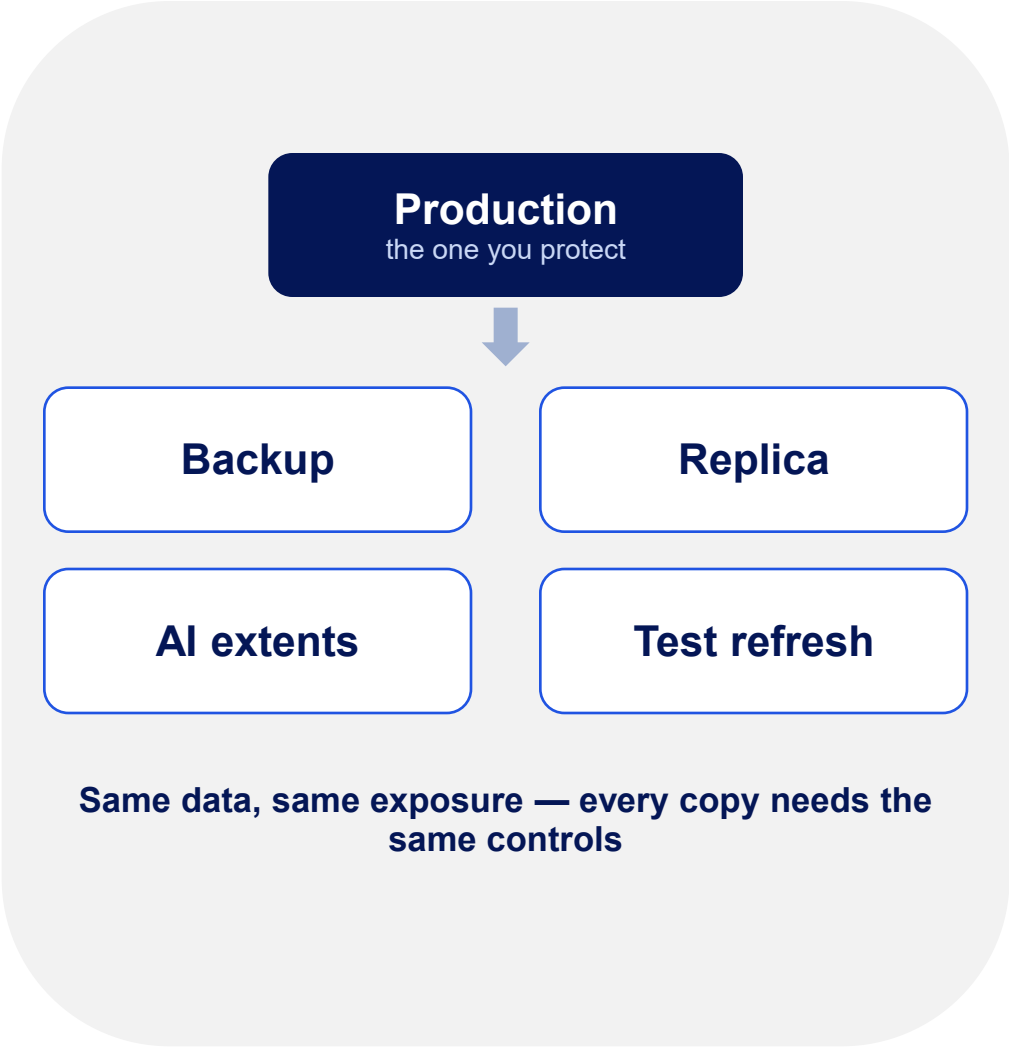
MiP

Availability is a security property

Confidentiality, integrity and availability – most estates score two out of three

Every backup, AI extent and replica is another copy that must be protected

A recovery you have never rehearsed is an assumption, not a plan



PUG CHALLENGE

[01]

Source

Backups

Afer-Image - ships notes as written

- Manual AI files

- Real-time Replication

[02]

Target

Warm standby

- Roll-forward

- near real-time replica

Secondary backups

[03]

Failover

Planned switchover vs disaster

The target becomes the source

Rehearse both, on a schedule

PUG CHALLENGE

The last line of defence

Backup TDE encrypted databases

A backup is a complete, portable copy of everything you hold

Online vs offline, full vs incremental – and what each exposes

Store backup and key separate

Validate with `proutil` – and with a scheduled restore test



A backup is the whole vault, in one box

PUG CHALLENGE

What it gives you

The transaction record

The evidence trail behind every committed change

Continuous record backups

Roll forward to any point in time after the last backup

The foundation replication is built on

What it costs

Operational discipline

Extents must be managed, archived and monitored – or the database stops

AI files are readable data: encrypt and protect them like the database

Monitor the AI activity and extents

Additional space for AI files and backups

PUG CHALLENGE

What it gives you

One source, up to two targets

A source database replicates to up to two target databases.

Targets can run on other servers, or in other data centres.

Each target stays identical to the source, transaction by transaction.

If the source fails, a target becomes active, with limited downtime for users.

Allows for distributed read-only (R/O) query and reporting

What it costs

Operational discipline

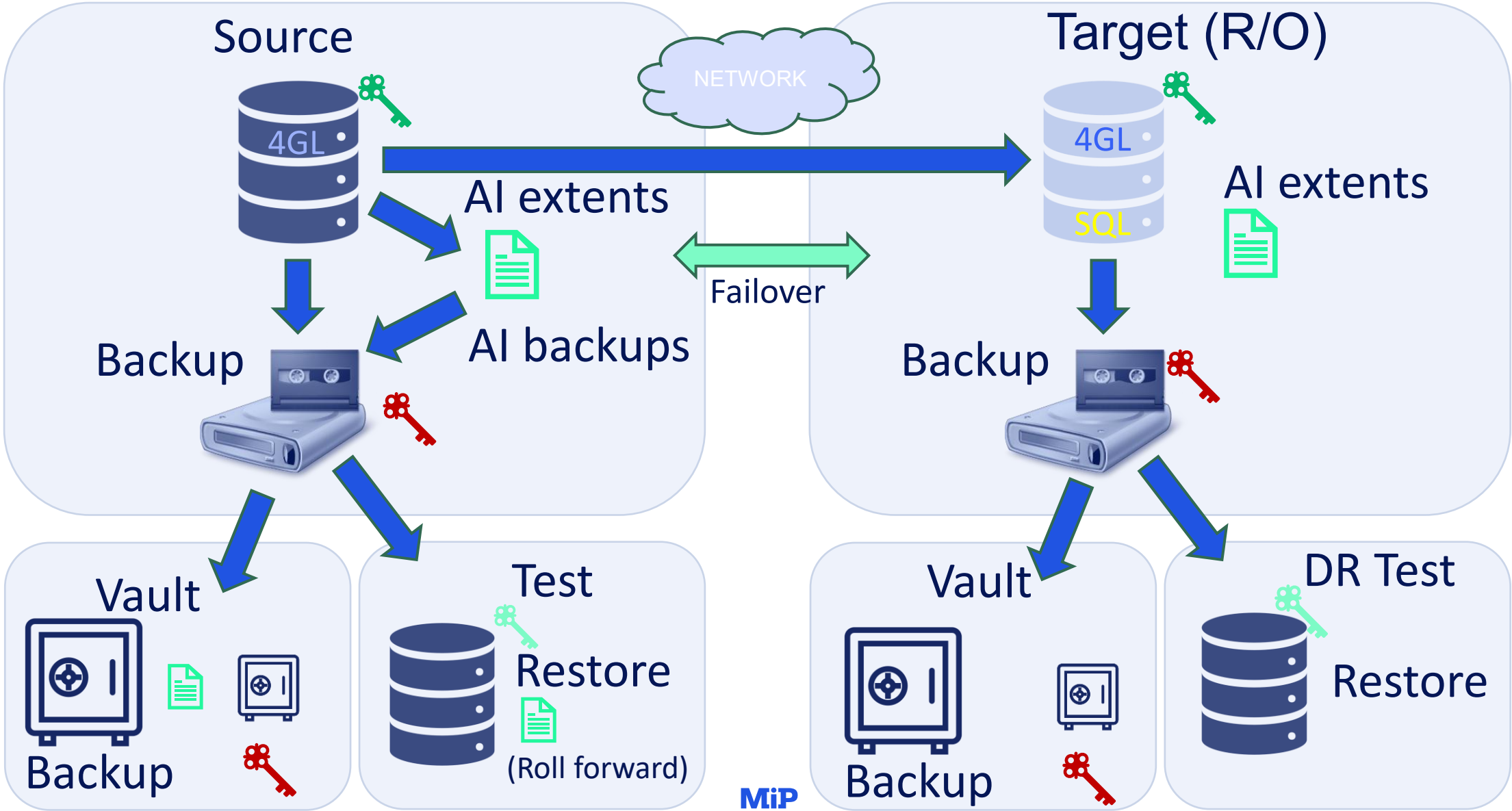
Replication rides on after-imaging — AI has to stay healthy.

A target is a full copy: same encryption, same access controls.

A slow or broken link shows up as replication lag.

Failover you have never rehearsed is an assumption, not a plan.

SECURE CONTINUITY – ULTIMATE TO STRIVE TOWARDS



PUG CHALLENGE

The exits

More paths than you think

Client to database (ABL, SQL, ODBC/JDBC)

Connections to PAS (Mobile App, API's)

Source to replication target

Backups and dumps to storage or cloud

Extracts to downstream systems

Map and secure every path you have.

The armoured car

Securing every route

TLS on OpenEdge connections.

HTTPS for PASOE and web services.

Encrypt files before they move, not after they land.

Password-protect and encrypt extracts.

Retire SSL and the old TLS versions.

PUG CHALLENGE

[01]

Rehearse the failover

Fail over on a schedule, not on the day.

Time the recovery and record the real RTO.

Secure the secondary like production.

Untested DR fails on the day.

[02]

Protect the copies

Extents, replicas and backups are all full copies.

Encrypt every copy; keep the keys somewhere else.

Test a restore with a re-generated key.

The copy nobody tracks is the one that leaks.

[03]

Watch the exits

TLS on every connection, replication included.

Encrypt extracts before they leave the estate.

Review who still receives a feed.

An extract in clear text undoes the rest.

PUG CHALLENGE

So where do you actually start on Monday morning?

MiP

PUG CHALLENGE

Start testing your own defences

Can you read production data from a raw file on disk today?

Who currently holds SQL DBA rights on your production database?

When did you last restore an encrypted backup, end to end?

When did you last fail over, and how long did it take?

If someone changed a record they should not have, could you prove it?



Try your own doors before someone else does

THE ROADMAP

PUG CHALLENGE

[01]

Find the copies

Inventory every copy you hold.

Cheap, and usually alarming.

[02]

Close the doors

Review defaults

Upgrade security

Fix any account with no owner.

[03]

Secure

Encrypt database.

Plan the keystore.

OE Replication

After-Imaging

[04]

Prove it

Audit and report.

Restore, fail over – on a schedule.

MiP

PUG CHALLENGE

Where to go next

Progress documentation – TDE, auditing, replication links

Progress KB articles

PUG Challenge sessions that go deeper on individual topics

OpenEdge forums

Reach out…



PUG CHALLENGE

Thank you

Pieter J. Meyer · pm@mip.co.za · www.mip.co.za

MiP