Thank you sponsors!



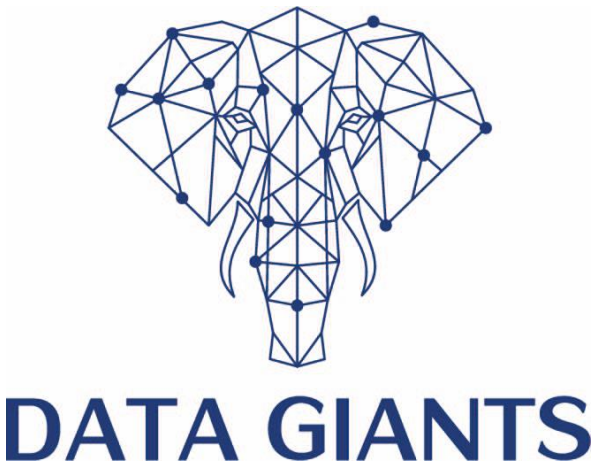
























Database Development with a Security-First Mentality

Data Saturday Holland 2026

DATA SATURDAYS HOLLAND

Late summer 2025...



Be aware of what you share

Introduction



Danny de Haan

Solutions Engineer

redgate

CIS Center for Internet Security®

thatinfradba.com | thatinfradba@pm.me | /dannydh | /ThatInfraDba



Schedule

- Why Security First
- Principles of Secure Development
- Core Security Features
- Continuous Monitoring & Risk Mitigation
- Security as a Lifestyle



The Stakes

- Data breaches are costly
- Compliance Requirements
  - GDPR
  - NIS2
  - DORA
  - ISO 27001
  - PCI-DSS
  - HIPAA



The Stakes

- Data breaches are costly
- Compliance Requirements
- Security as a business enabler



Common Pitfalls

- Security as an afterthought
- Over-privileged accounts
- Weak authentication
- Lack of monitoring & auditing



Mindset vs. Mentality in Security

- Mindset → “we patch because we must”
- Mentality → “we design securely because it’s who we are”
- Why it matters: Security isn’t just a checklist → it’s a discipline that must be lived, not just followed.



Principles of Secure Development



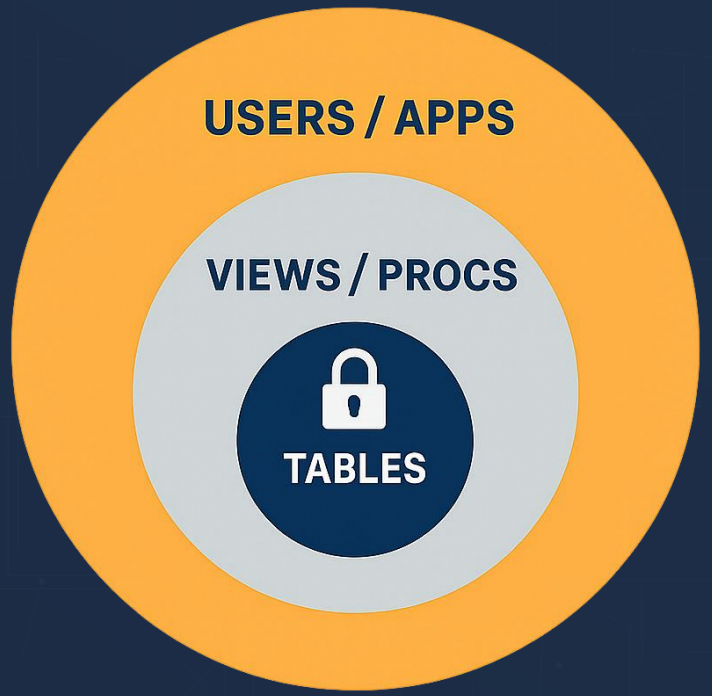
Security in the blueprint

- Would you build a house and add locks later?
- Or design it with doors, locks, and alarms from the start?
- Security must be part of the blueprint, not an afterthought.



Foundations of Secure Data Architecture

- Least privilege
- Defense in Depth
- Separation of Concerns



Three Layers Between Users and Data

Users & apps
read & write data

GATEKEEPERS
Procedures

- Parameterized input only
- Validation and auditing in one place
- Users get EXECUTE, nothing else

WINDOWS
Views

- Expose only the columns needed
- Filter rows by user context
- Hide hashes, mask identifiers

THE VAULT
Tables

- Raw data, including PII
- Classify sensitive columns
- Restrict direct access: DBAs and trusted services only



Why Procedures Work as Gatekeepers

Mechanism	How it grants access	Watch out for
Ownership chaining	Procedure and table share an owner, so SQL Server skips the table permission check. Callers need only EXECUTE.	Breaks on dynamic SQL. Keep cross-database chaining off.
EXECUTE AS	The procedure runs under another principal's security context.	Impersonate a low-privilege user, never dbo or OWNER by habit.
Module signing	Sign the procedure with a certificate; a user created from that certificate carries the permissions.	ALTER drops the signature, so re-sign in your deployment. Works for dynamic SQL and cross-database calls.



Application Access and Secrets

Instead of	Do this
App connects as sa or db_owner	**One low-privilege principal per app, with EXECUTE on its own schema**
Passwords in connection strings and config files	**Managed identity (Azure, Arc) or gMSA on-premises; other secrets in Key Vault**
One shared account for every service	**A principal per service, so audit logs name the real caller**
Credentials committed to Git	**Secret scanning at pull request; rotate anything that leaked**
Encrypt=False or TrustServerCertificate=True	**Encrypt=True (Strict on TDS 8.0) with a trusted certificate**



Core Security Features



SQL Server Security Features

Authentication Who are you?

Authorization What may you do?

Encryption Can stolen data be read?

Row-Level Security Which rows can you see?

Data Masking Which values are hidden?

Auditing

Records what happened at every layer

Each layer assumes the one above it can fail: defense in depth, from the login to the column.



SQL Server Security Features

- Authentication
  - Windows vs SQL Logins vs Managed Identity*
  - Multi-factor authentication*
  - Contained users

* Requires Azure Arc



SQL Server Security Features

- Authentication
- Authorization
  - Role-based access control
  - Custom roles
  - Avoid direct user permissions



RBAC Demo



SQL Server Security Features

- Authentication
- Authorization
- Encryption
  - Transparent Data Encryption (TDE)
  - Always Encrypted
  - Column-level encryption



SQL Server Security Features

- Authentication
- Authorization
- Encryption
- Row-Level Security
  - Fine-grained access
  - Predicate functions



RLS Demo



SQL Server Security Features

- Authentication
- Authorization
- Encryption
- Row-Level Security
- Data Masking*
  - Protect sensitive fields
  - Masked vs unmasked views

* Not a security boundary



SQL Server Security Features

- Authentication
- Authorization
- Encryption
- Row-Level Security
- Data Masking
- Auditing
  - SQL Server Audit
  - Extended Events
  - Centralized log collection
  - Ledger Tables



Auditing Demo



Continuous Monitoring & Risk Mitigation



Continuous Monitoring & Risk Mitigation

STEP 1
Capture

- SQL Server Audit for logins, role and permission changes
- Extended Events for errors and failed queries
- Write audit files where admins can't wipe them

STEP 2
Collect

- Forward every instance to one SIEM (Sentinel, Splunk)
- Correlate with app and OS logs
- Keep retention in line with compliance

STEP 3
Alert

- Alert rules for the signals on the next slide
- Dashboards for trends, alerts for action
- Tune rules to avoid alert fatigue

STEP 4
Respond

- A runbook per alert type
- A named owner and escalation path
- Review each incident and update the rules



What to Watch

Signal	Audit action group or tool	Alert when
Failed logins	FAILED_LOGIN_GROUP	A spike from one source or against one account
New privileged members	SERVER_ROLE_MEMBER_CHANGE_GROUP DATABASE_ROLE_MEMBER_CHANGE_GROUP	Anyone joins sysadmin or db_owner
Permission changes	DATABASE_PERMISSION_CHANGE_GROUP	A GRANT or DENY lands outside a release
Audit tampering	AUDIT_CHANGE_GROUP	Always: an audit stopped or altered
Unusual data access	Defender for SQL, Extended Events	Bulk reads on classified columns



Hardening the Instance

Disable sa

Rename or disable it; admins sign in with Windows or Entra ID logins

Switch off unused features

xp_cmdshell, CLR, OLE Automation and ad hoc distributed queries stay off

Patch on a schedule

Apply each cumulative update within an agreed window

Least-privilege service accounts

gMSA or virtual accounts, never a domain admin

Close trust shortcuts

Cross-database ownership chaining off; TRUSTWORTHY OFF on user databases

Remove what you don't need

Unused logins, orphaned users, sample databases and open endpoints



Regular Security Assessments

- SQL Vulnerability Assessment → Built-in scanner in SSMS & Azure; baseline misconfigurations
- Microsoft Defender for SQL → Threat detection, anomalous access alerts
- CIS Benchmarks → Industry baseline for SQL Server hardening
- Penetration testing → External validation, at least annually
- Permission audits → Review role membership & orphaned users quarterly



Security as a Lifestyle



Security as Fire Prevention

- Firefighting = reacting once the blaze has started
- Prevention = sprinklers, alarms, safe wiring built in
- Security must be prevention, not just emergency response



DevSecOps

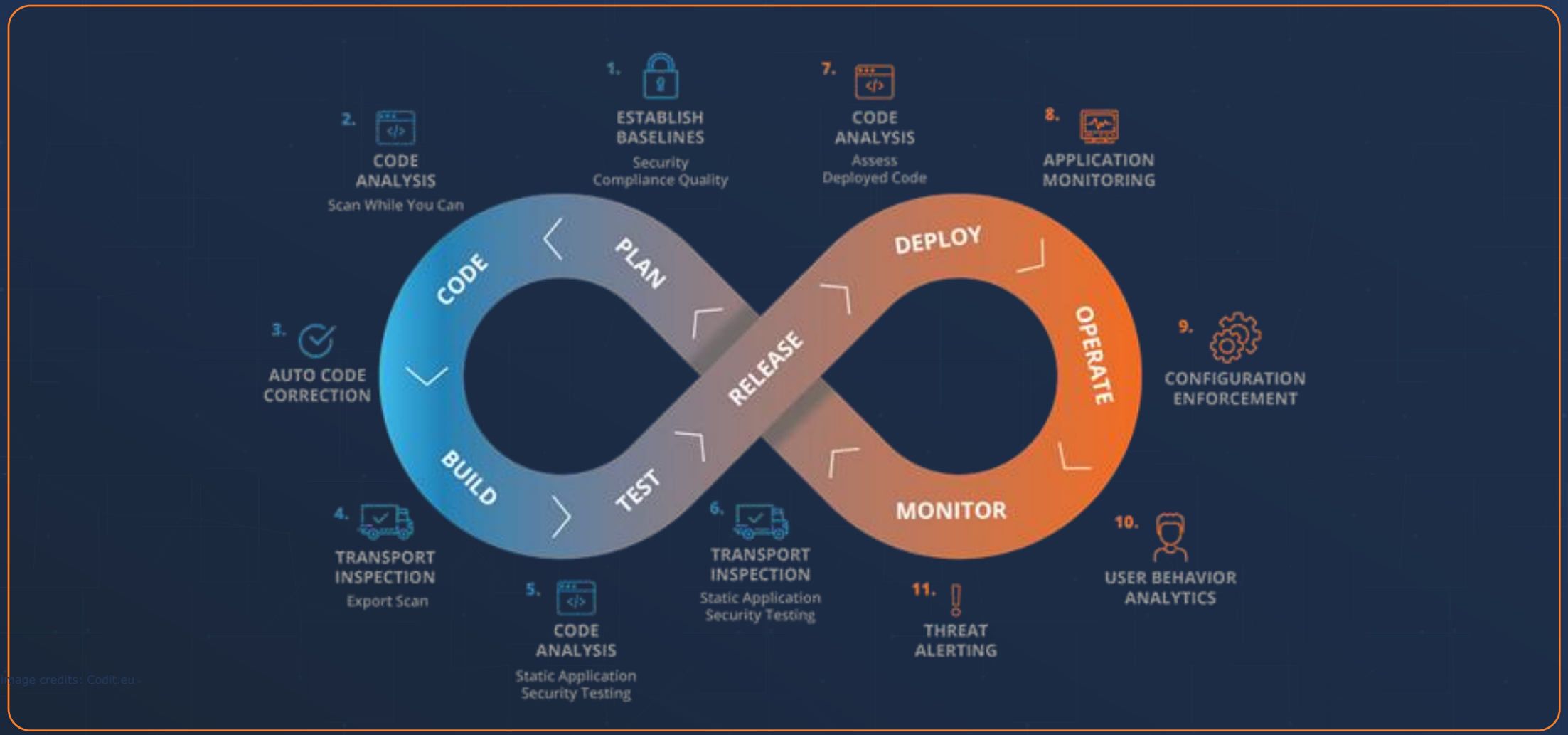


Image credits: Codit.eu

DevSecOps for Databases

- Source control everything → Schema, permissions, static data (state-based or migration-based)
- Secret scanning → Block connection strings & credentials at PR time
- Static analysis → T-SQL linting, SQL injection patterns, missing schema binding
- Deployment gates → Require review for permission changes; drift detection on prod
- Shift left → Run vulnerability assessment in CI, not after release



Key Takeaways

- **Build security in from the start**: lock the tables, give access through views and procedures
- **Use your DBMS' built-in features**: RBAC, encryption, RLS, masking and auditing are already in the box
- **Monitor continuously**: capture, collect, alert, respond
- **Stay compliant**: map each regulation to a feature you can prove



Thank you very much

Database Development with a Security-First Mentality

Feedback



Danny de Haan

thatinfradba.com | thatinfradba@pm.me | /dannydh

Thank you sponsors!













