SECURITY PATHFINDER

Risk Management & Action Framework

Inherent Risk | Access Controls | Detection | Forensic Analysis

CLASSIFICATION: CONFIDENTIAL

Prepared: May 22, 2026

Executive Overview

This document provides a structured action framework produced from the Security Pathfinder output. It addresses the full risk lifecycle across four pillars: identifying inherent risks, controlling who can access those risks, detecting when they are exploited, and forensically investigating what occurred and why.

PURPOSE	Enable security and database administration teams to systematically reduce exposure in the SA schema by working through a prioritized sequence of control tightening, risk reduction, detection implementation, and forensic readiness.

How the Framework Is Structured

The Pathfinder output identifies methods, processes, and code that have been modified or newly created to access data tables in the SA schema. From that output, four sequential work streams are defined:

1. Inherent Risk Identification — cataloging every code path, stored procedure, dynamic SQL call, and external access point that touches SA tables.
2. Access Control Reduction — shrinking the population of principals with privileges to execute those risk items.
3. Inherent Risk Reduction — eliminating or hardening the risky code paths themselves, with special focus on dynamic access originating outside the system boundary.
4. Detection & Forensic Investigation — instrumenting the environment to catch exploitation in real time and reconstruct what any actor did, how they did it, and what they were attempting to accomplish.

01 Inherent Risk Identification

Cataloging every modified or created code path touching SA tables

Definition of Inherent Risk

An inherent risk, in the context of this framework, is any method, process, stored procedure, function, trigger, dynamic SQL block, or external integration that has been created or modified and that has the ability to read, write, modify, or delete data in the SA schema tables. The Pathfinder identifies these items as the raw attack surface.

Risk Categories Identified by Pathfinder

- Modified stored procedures — existing procedures altered to expand their data access or bypass row-level security.
- Newly created code objects — functions, views, or procedures that did not exist before the review period.
- Dynamic SQL generation — code that builds and executes SQL strings at runtime, making static analysis difficult.
- External access paths — integrations, APIs, linked servers, or ODBC/JDBC connections that reach SA tables from outside the primary application boundary.
- Privilege escalation vectors — code paths that change execution context (e.g., EXECUTE AS, impersonation chains) to gain elevated access.

KEY RISK	Dynamic SQL generated outside the system boundary is the highest-severity category because it can bypass all static controls and is the hardest to detect after the fact.

Inherent Risk Register — Action Items

Action Item	Owner / Role	Priority	Notes
Export full Pathfinder output to risk register	Security Team Lead	**Critical**	*Baseline inventory, date-stamped*
Tag each item: Modified vs. New, Static vs. Dynamic	DBA / Security Analyst	**Critical**	*Enables triage prioritization*
Map each risk item to SA table(s) it can access	DBA	**High**	*Defines blast radius per item*
Assign risk severity score (High/Med/Low)	Risk Manager	**High**	*CVSS-style or internal rubric*
Document all external dynamic SQL entry points	Security Architect	**Critical**	*Priority for Sections 02 & 03*

02 Access Control Reduction

Reducing the number of principals with privileges over inherent risk items

The Control Problem

An inherent risk only becomes an active threat when a principal has the privilege to invoke it. The control issue is therefore: who has execute, alter, or indirect access to each item in the inherent risk register. Reducing this population is the fastest and most effective near-term risk reduction available.

Step 1 — Enumerate All Privileged Principals

For every item in the inherent risk register, identify all database logins, application service accounts, AD groups, and linked-server credentials that hold any of the following permissions:

- EXECUTE on the procedure, function, or object
- ALTER on the object (can modify the code itself)
- CONTROL or db_owner membership (implicit access to all objects)
- IMPERSONATE or EXECUTE AS rights that chain to a privileged account
- Membership in roles that carry any of the above

Step 2 — Apply Least Privilege

Review each principal's business justification for access. Remove privileges that are not operationally required. Prefer role-based grants over direct user grants so that access can be managed in one place. Document every removal with the approver and date.

Step 3 — Segregation of Duties

Ensure that no single account has both the ability to modify a risk-item and the ability to execute it in production. In particular:

- Developers must not hold production EXECUTE rights on objects they can ALTER.
- Service accounts must not hold DDL rights (CREATE, ALTER, DROP).
- DBAs who perform emergency access must do so through a break-glass process with automatic alerting.

CONTROL ACTION	Conduct a quarterly re-certification of all accounts with privileges touching SA schema objects. Any account not re-certified within the window is automatically de-provisioned.

Access Control — Action Items

Action Item	Owner / Role	Priority	Notes
Query sys.database_permissions for all SA object grants	DBA	**Critical**	*Run immediately; baseline snapshot*
Identify accounts with db_owner or CONTROL SERVER	Security Team Lead	**Critical**	*Escalation risk — review first*
Remove all direct grants; convert to role-based grants	DBA / IAM Team	**High**	*Reduces management complexity*
Remove developer accounts from production execution roles	IAM Team	**High**	*Enforce dev/prod segregation*
Implement break-glass process for DBA emergency access	Security Architect	**High**	*With automatic alert to CISO*

Action Item	Owner / Role	Priority	Notes
Schedule quarterly access re-certification	Compliance / Risk	**Medium**	*Calendar entry with escalation path*
Disable or scope all service account excess privileges	DBA / App Team	**High**	*Principle of least privilege*

03 Inherent Risk Reduction

Eliminating or hardening risky code paths — especially dynamic external access

Goal: Shrink the Attack Surface

Where access control reduces who can reach a risk item, risk reduction removes or hardens the item itself. This is more durable because it eliminates the vulnerability regardless of future access changes. The priority sequence is: (1) external dynamic access, (2) internal dynamic SQL, (3) overly permissive static code.

Priority 1 — External Dynamic Access

Dynamic SQL executed from outside the application's internal boundary (e.g., from external APIs, linked servers, ad-hoc query tools, or BI platforms that connect directly to the SA schema) represents the highest inherent risk because:

- It is not subject to application-layer input validation.
- It is the most commonly exploited vector for SQL injection and data exfiltration.
- Its execution is harder to attribute to a specific user or workflow.

Reduction steps for external dynamic access:

5. Identify all external connection strings that target SA schema objects directly.
6. Route all external data requests through a defined API or service layer that enforces parameterized queries.
7. Revoke direct table SELECT/INSERT/UPDATE/DELETE from external service accounts; grant access to views or stored procedures only.
8. Block ad-hoc query tool connections to the SA schema at the network or firewall level where possible.
9. Implement IP allowlisting for any remaining legitimate external SQL connections.

Priority 2 — Internal Dynamic SQL

For dynamic SQL that originates within the application, apply the following hardening sequence:

- Convert to parameterized stored procedures wherever the business logic permits.
- Where dynamic SQL is unavoidable, implement strict input validation and a whitelist of allowable table/column names before string concatenation.

- Wrap dynamic execution in sp_executesql with typed parameters rather than EXEC(@string) with open concatenation.
- Add server-side logging of every dynamic SQL statement executed (see Section 04).

Priority 3 — Overly Permissive Static Code

Review each static stored procedure and function in the risk register for scope creep — code that accesses more tables or columns than its documented purpose requires. Refactor to narrow access. Apply column-level security or row-level security (RLS) policies on sensitive SA tables.

REDUCTION GOAL	Target: zero direct external connections to SA schema tables within 90 days. All remaining dynamic SQL wrapped in sp_executesql with typed parameters within 180 days.

Risk Reduction — Action Items

Action Item	Owner / Role	Priority	Notes
Enumerate all external connection strings to SA schema	DBA / Network	**Critical**	*Starting point for remediation*
Route external access through API/service layer	App Architect	**Critical**	*90-day target*
Revoke direct table rights from external service accounts	DBA	**Critical**	*Replace with view/proc grants*
Network-block ad-hoc query tool direct connections	Network / DBA	**High**	*Firewall or SQL listener config*
Convert internal dynamic SQL to sp_executesql + params	Dev Team	**High**	*180-day target; track by object*
Apply RLS policies on highest-sensitivity SA tables	DBA / Security	**High**	*Define sensitivity tiers first*
Refactor overly broad stored procedures	Dev Team	**Medium**	*Scope to documented purpose only*
Implement IP allowlisting for remaining external SQL	Network / Security	**High**	*Interim control while rerouting*

04 Detection

Identifying in real time when inherent risk items are utilized

Detection Philosophy

Detection does not prevent an event, but it limits the dwell time between exploitation and response, and it generates the evidence needed for forensic investigation. Effective detection for SA schema access requires multiple layers: database-native auditing, application-layer logging, network monitoring, and behavioral analytics.

Database-Native Auditing

Enable SQL Server Audit (or the equivalent for your RDBMS) to capture the following event classes for all SA schema objects in the risk register:

- EXECUTE events on every stored procedure and function in the risk register.
- SELECT, INSERT, UPDATE, DELETE on the underlying SA tables.
- ALTER and CREATE events (schema change detection).
- LOGIN events for all accounts with SA schema privileges.
- EXECUTE AS and impersonation events.
- Failed permission attempts (indicates probing behavior).

Extended Events / Trace for Dynamic SQL

Because dynamic SQL may not surface through object-level audit alone, configure Extended Events to capture:

- sql_statement_completed events filtered to the SA database, with the statement text logged.
- rpc_completed events for all parameterized calls to SA schema objects.
- Threshold alerts when statement volume per session exceeds a defined baseline (bulk exfiltration indicator).

SIEM Integration & Alerting

Forward all database audit logs to the SIEM in real time. Define the following alert rules:

- Any execution of a risk-register object by an account not in the approved principal list.
- Any access to SA tables outside defined business hours by a service account (anomaly).
- Any DDL event (ALTER, DROP, CREATE) on SA schema objects outside a change window.
- Any access from an IP address not in the approved connection allowlist.
- Row-count thresholds: queries returning more than N rows from sensitive SA tables (exfiltration indicator — tune N to business baseline).

DETECTION TARGET	Mean Time to Detect (MTTD) for unauthorized SA schema access: under 15 minutes for critical objects, under 1 hour for high-risk objects.

Detection — Action Items

Action Item	Owner / Role	Priority	Notes
Enable SQL Server Audit on all SA risk-register objects	DBA	**Critical**	*Captures execute, DDL, DML events*
Configure Extended Events for dynamic SQL capture	DBA	**Critical**	*Required for dynamic SQL forensics*
Forward audit logs to SIEM in real time	Security / DBA	**Critical**	*No local-only log storage*
Build SIEM alert: execution by non-approved principal	Security Analyst	**Critical**	*Alert + ticket auto-creation*
Build SIEM alert: after-hours service account access	Security Analyst	**High**	*Anomaly baseline required*
Build SIEM alert: DDL outside change window	Security Analyst	**High**	*Change-window calendar feed*
Set row-count exfiltration threshold alerts	Security Analyst	**High**	*Tune to SA table baselines*
Test all alerts with red-team exercise	Security Team	**High**	*Validate before declaring live*
Define incident response runbook for each alert type	Security Lead / IR	**High**	*Include escalation chain*

05 Forensic Investigation

Identifying wrongdoing, reconstructing actions, and determining intent

Forensic Investigation Framework

When detection fires — or when a historical incident is suspected — the forensic investigation must answer five core questions: Who accessed the data? What data was accessed or modified? How did they access it (the code path used)? When did each action occur? And critically: What were they trying to accomplish?

Step 1 — Preservation (Act First)

Before any analysis, preserve all evidence in its original state to maintain chain of custody:

10. Immediately snapshot all relevant database audit logs and Extended Events trace files to immutable storage (write-once S3, WORM drive, or offline copy).
11. Capture a point-in-time database backup to preserve data state.
12. Preserve all SIEM log data for the suspected time window — extend the retention hold to prevent rotation.
13. Capture network flow records (NetFlow/IPFIX) for the IP addresses and ports involved.
14. Do NOT alter, remediate, or patch the affected objects until a forensic copy is secured.

Step 2 — Timeline Reconstruction

Build a unified chronological timeline using all available log sources. Each event in the timeline should record:

- Timestamp (UTC, sub-second where available)
- Principal (database login + AD identity if mappable)
- Source IP address and hostname
- Object accessed (table, procedure, function, or dynamic SQL text)
- Operation type (SELECT, EXECUTE, INSERT, ALTER, etc.)
- Row count or data volume affected
- Result (success or failure)

Correlate the database timeline with application logs, authentication logs (AD/LDAP), and network logs to confirm the full chain of events and confirm or rule out insider vs. external actor.

Step 3 — Code Path Analysis

Identify exactly which code path was used. For each event in the timeline:

- Determine whether it was a static stored procedure call, a view, or a dynamic SQL string.
- If dynamic: reconstruct the full SQL statement from the Extended Events log.
- Map the code path back to the risk register entry — was this a known inherent risk item, or a previously unknown vector?
- Check for signs of obfuscation: encoded strings, multi-step chaining through temp tables, use of EXECUTE AS to mask the originating identity.

Step 4 — Data Impact Assessment

Determine what data was actually exposed or modified:

- For SELECT operations: determine which rows and columns were returned, and whether the result set was large enough to constitute exfiltration.
- For INSERT/UPDATE/DELETE: reconstruct the before-and-after state using transaction log analysis or CDC (Change Data Capture) if enabled.
- Cross-reference with data classification — which data sensitivity tier was touched (PII, financial, regulated, confidential)?

Step 5 — Intent Determination

This is the most analytical phase. Intent cannot always be proven from logs alone, but the following indicators are probative:

Indicators of Deliberate Exfiltration

- Queries structured to export entire tables or large datasets (no meaningful WHERE clause, SELECT *).
- Repeated access to the same high-value tables in a short window — more than would be explained by normal workflow.
- Access from a client tool (e.g., SSMS, ODBC driver) rather than the application service account — suggests manual human action.
- Export or output to a file, linked server, or external destination.
- Access occurring outside the actor's normal working hours or from an unusual location.

Indicators of Unauthorized Modification

- UPDATE or DELETE operations not tied to an application transaction (no corresponding application log entry).
- Changes to data that benefit the actor personally (e.g., financial records, access tables, approval records).
- Modification of audit or log tables themselves — a strong indicator of cover-up intent.
- DDL changes (ALTER, DROP) to risk-register objects outside a documented change window.

Indicators of Reconnaissance

- Large number of failed permission attempts before a successful access.
- Queries against system catalog tables (sys.tables, sys.columns, INFORMATION_SCHEMA) from an account that does not normally query metadata.
- Sequential access to many different tables in a short period — consistent with mapping the data landscape.

REPORTING	The forensic investigation report must include: a factual timeline, code path used, data impacted, a classification of likely intent (exfiltration / modification / reconnaissance / accidental), confidence level, and recommended escalation path (HR, legal, law enforcement).

Forensic Investigation — Action Items

Action Item	Owner / Role	Priority	Notes
Establish evidence preservation SOP before an incident	Security Lead / Legal	**Critical**	*Chain of custody documentation*
Configure CDC or temporal tables on highest-risk SA tables	DBA	**Critical**	*Enables before/after reconstruction*
Ensure Extended Events captures full dynamic SQL text	DBA	**Critical**	*Without this, dynamic SQL is a blind spot*

Action Item	Owner / Role	Priority	Notes
Build unified log correlation query (DB + AD + Network)	Security Analyst	**High**	*Pre-build for rapid deployment*
Define data classification tiers for SA schema tables	Data Governance	**High**	*Required for impact assessment*
Create forensic investigation report template	Security Lead	**High**	*Standardize findings documentation*
Train investigation team on intent-indicator framework	Security Lead	**Medium**	*Tabletop exercise recommended*
Establish legal/HR escalation path and trigger criteria	Legal / HR / CISO	**High**	*Pre-agreed before an incident occurs*

Implementation Roadmap Summary

The table below consolidates the recommended implementation sequence. Phases are designed so that each builds on the previous — you cannot effectively reduce risk items until you know what they are, and detection is only meaningful once controls are in place.

Phase	Work Stream	Key Deliverable	Target	Owner
Phase 1 0–30 days	Risk Identification	Complete risk register from Pathfinder output	Day 30	Security Lead
Phase 2 0–30 days	Access Control	Baseline privilege inventory; remove excess grants	Day 30	DBA / IAM
Phase 3 30–60 days	Detection	Audit logging live; SIEM alerts active; runbooks drafted	Day 60	Security Analyst
Phase 4 30–90 days	Risk Reduction — External	Zero direct external SA connections	Day 90	App Architect
Phase 5 90–180 days	Risk Reduction — Internal	Dynamic SQL converted to sp_executesql + params	Day 180	Dev Team
Phase 6 Ongoing	Forensic Readiness	CDC live; investigation SOP adopted; team trained	Continuous	Security Lead