HOW-TO GUIDE

How to Audit SQL Server Login Failures with Extended Events

Use Extended Events to capture SQL Server login failures with a low-overhead session, verify the evidence, and cleanly remove the audit when you are done.

Databases - August 02, 2026

Quick version

If you need to audit SQL Server login failures, the fastest safe approach is to create a small Extended Events session that captures the error_reported event, filters for the login failure error numbers you care about, writes to an event file, and then queries the file for the failed logins, client application, host, and timestamp.

This is usually preferable to broad tracing because it is lighter, easier to scope, and simpler to clean up. It also gives you evidence you can use to distinguish bad passwords, missing logins, disabled accounts, application misconfiguration, or connectivity issues. If you are also investigating broader authentication or access control patterns, the same evidence can complement How to Secure SQL Server with Row-Level Security and Auditing when you need to correlate failed access attempts with higher-level security controls.

At a practical level, you will:

- Create a narrowly scoped session.
- Reproduce or observe the failure.
- Read the captured events.
- Validate that the audit contains what you expected.
- Remove the session and files when finished.

Prerequisites and scope

Before you start, confirm a few operational details so the audit remains safe and useful.

- You need permission to create Extended Events sessions and read the target directory.
- The SQL Server service account must be able to write to the chosen file path.
- The target path should be on local storage or another reliable location with enough space for the expected volume of events.
- If you are auditing a busy instance, define a capture window and a retention plan so the event file does not grow unchecked.
- If you need to audit across availability scenarios, confirm where the session will run and whether a failover changes the file path or access model. For that kind of operational context, it can help to compare the results with your availability troubleshooting workflow, such as the one used in the SQL Server Always On Availability Groups Troubleshooting Guide.

This guide focuses on login failures captured on the database engine. It does not cover password policy design, directory service troubleshooting, or application credential management in depth.

What to capture

For login failure auditing, the most practical event is error_reported filtered to common authentication failure numbers. The exact error you capture depends on the failure mode, but a useful starting point is to look for the errors that indicate login rejection rather than general connection noise.

A good capture should answer these questions:

- Which login name failed?
- From which host or client application did the attempt originate?
- When did it happen?
- What was the error number and message text?
- How often is the failure repeating?

In many environments, the event data alone is enough to distinguish a typo from a recurring application problem. If the same login is failing repeatedly from the same host and application name, you likely have a configuration issue rather than an isolated user mistake.

Create the Extended Events session

The following example creates a lightweight session that writes login failure events to a file. Adjust the file path to a location that the SQL Server service account can write to.

Why this pattern works

error_reported is broad, so the filter is what keeps the session focused. In this example, the filter narrows the capture to a specific login failure error number and excludes low-severity noise. The action list adds context so you can connect the failure to a host, app, and session.

Keep the action list small. Every extra field increases overhead slightly and can make the event file harder to read. Capture enough to identify the source, but not so much that the session becomes a general-purpose telemetry stream.

Generate or wait for a login failure

Once the session is running, reproduce the failure in a controlled way or wait for the event to occur naturally.

Useful examples include:

- An application using an incorrect password.
- A disabled login being used by a service.
- A deployment that still references an old credential.
- A scheduled job running under an account that no longer has access.

Avoid intentionally creating failures on production systems unless you have a controlled maintenance window or a clear operational need. For most environments, you can validate the session with a non-production login or by reproducing a known existing issue.

If you are troubleshooting a larger authentication problem where the failures appear alongside deadlocks, retries, or timeouts, keep the scope tight. Capture one symptom class at a time so the evidence stays easy to interpret. A separate workflow such as SQL Server Deadlock Troubleshooting with Extended Events is better suited to concurrency diagnostics than a login audit session.

Read the captured events

You can query the event file directly with sys.fn_xe_file_target_read_file and extract the useful fields from the XML payload.

Expected output

A typical result set should show one row per failed login event. The most useful columns are the event timestamp, error number, message text, server principal or username, client host, and application name.

If the message text is generic, use the action fields to narrow the source. For example:

- A single host and application name often points to a service or scheduled task.
- Many hosts with the same login can indicate an application-wide credential issue.
- A missing or null username may suggest the failure happened before authentication completed.

If you need to prove that the audit is working before relying on it, compare the event time with the observed login failure and check that the same source appears in the output.

Validate the audit before you trust it

Validation is the step that separates a useful audit from a false sense of coverage. Do not skip it.

Check the following:

- The session is running.
- The event file is being written.
- The filter is capturing the failure you meant to collect.
- The action list provides enough context to identify the source.
- The file can be read by the account performing the investigation.

You can confirm the session state with:

If the session exists but is not active, the failure may be in the start command, permissions, or file path. If the file exists but no rows appear, your filter may be too narrow or the target error number may not match the failure mode you are investigating.

Common adjustments

A few small changes make the audit more practical in real environments.

Capture a broader set of authentication failures

If you are not sure which error number to target, temporarily widen the filter and review the messages before narrowing the session again. This is useful when you are first identifying the dominant failure mode in a new environment.

Add duration or frequency context

If the same failure repeats many times, keep the file target and add a short observation window. Repetition often indicates an application retry loop, stale secret, or service account problem rather than a one-time user error.

Scope by environment or server role

On instances with multiple workloads, avoid leaving a broad audit session active indefinitely. A login failure audit is best used as a diagnostic or controlled monitoring tool, not as a permanent catch-all telemetry source.

Rollback and cleanup

When you are done, stop and remove the session, then delete or archive the event files according to your retention policy.

After dropping the session, remove the .xel files if they are no longer needed. If you are keeping them for incident review, move them to a secured location and make sure access is limited to the people who need to investigate the issue.

Operational boundaries and caveats

Extended Events is a good fit for login failure auditing when you need low-overhead, targeted evidence. It is not a replacement for a complete security program, and it should not be used as the only source of truth for account abuse or brute-force attempts.

Use caution when:

- The instance is already under heavy load.
- The failure rate is very high and could generate a large event file.
- The investigation involves credentials, secrets, or regulated audit data.
- The login issue may be caused upstream by a proxy, gateway, or application pool rather than by SQL Server itself.

If the event file does not show the expected login source, verify whether the failure is really occurring at the engine, or whether the client is failing before the connection is established. In those cases, network traces, application logs, or identity provider logs may be needed alongside the SQL Server audit.

Final takeaway

To audit SQL Server login failures with Extended Events, keep the session narrow, write to an event file, verify the output against a known failure, and remove the session when you are done. That workflow gives you practical evidence without turning authentication monitoring into a noisy always-on trace.

Use this guidance together with PostgreSQL RBAC and PostgreSQL index bloat to connect the workflow with related operational context already available on the site.