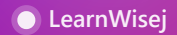


Auditing, Secrets und Konfiguration

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Audit-Trails für sensible Befehle, Korrelations-IDs, Verwaltung von Secrets, Umgebungskonfiguration, Logging ohne Datenabfluss. Was das für Entwickler bedeutet, die eine Wisej.NET-App im Produktivbetrieb absichern – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Audit-Trails für sensible Befehle, Korrelations-IDs, Verwaltung von Secrets, Umgebungskonfiguration, Logging ohne Datenabfluss.

1. Das Problem im Produktivbetrieb

AccessOps authentifiziert inzwischen Personen, löst ihre Berechtigungen auf und schützt seine Dokumente – und nichts davon hilft an dem Morgen, an dem jemand fragt, wer Antrag 4471 genehmigt hat, von welcher Adresse aus und ob vorher jemand abgewiesen wurde. Die vorigen fünf Module haben entschieden, was erlaubt ist; dieses entscheidet, was sich im Nachhinein nachvollziehen lässt. Außerdem schließt es die Lücke, die aus einem kleinen Vorfall einen großen macht: ein Datenbankpasswort oder ein OIDC-Client-Secret in einer Datei, die vor zwei Jahren eingecheckt wurde.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Audit-Trail	Ein geschäftlicher Nachweis jedes sensiblen Befehls – wer gehandelt hat, woran, wann und von wo –, aufbewahrt für eine festgelegte Frist, weil sich jemand dafür wird verantworten müssen.
Append-only-Speicher	Eine Audit-Tabelle, in die das Anwendungskonto nur per <code>INSERT</code> schreiben darf. Wenn der Code, der die Historie schreibt, sie auch umschreiben kann, beweist die Historie nichts.
Manipulationsnachweis	Eine lückenlose Sequenz plus eine Hash-Kette, bei der der Hash jeder Zeile ihre eigenen Felder und den vorherigen Hash abdeckt, sodass eine bearbeitete Zeile die Kette bricht.
Korrelations-ID	Ein Bezeichner, der beim Start eines Befehls erzeugt und auf jeder Logzeile und jeder Audit-Zeile wiederholt wird, die dieser Befehl verursacht, sodass sich ein einzelner Klick später rekonstruieren lässt.
Logging-Scope	Ein umgebender Eigenschaftsblock – <code>ILogger.BeginScope</code> oder ein Serilog-Enricher –, der die Korrelations-ID und den Akteur an alles anhängt, was darin protokolliert wird.
Secret-Speicher	Der Ort, an dem ein Credential außerhalb des Builds lebt: lokal User Secrets, in einem Container Umgebungsvariablen, im Produktivbetrieb ein verwalteter Vault – oder eine Managed Identity, bei der überhaupt nichts gespeichert wird.
Schwärzung	Eine Regel dafür, was nie in einer Log-Senke landet – Passwörter, Tokens, Cookies, Connection-Strings und personenbezogene Daten –, durchgesetzt durch strukturiertes Logging, nicht durch Erinnerungsvermögen.

3. Der rote Faden der Lektion

Der Eintrag, der den Streit überdauert

Zwei verschiedene Dinge werden „Logging“ genannt. Eine Logzeile dient der Diagnose und darf gesampelt oder nach einer Woche wegrotiert werden. Eine Audit-Zeile ist ein Beweismittel: Sie hat eine Aufbewahrungsfrist, einen Verantwortlichen und einen Leser, der kein Entwickler ist. AccessOps braucht einen `AuditService` als einzigen Einstiegspunkt für die zweite Art, aufgerufen von jedem sensiblen Befehl – einen Antrag genehmigen, eine Berechtigung erteilen oder entziehen, ein vertrauliches Dokument herunterladen und jeder Anmeldeversuch aus Modul 2. Nichts anderes schreibt in die Audit-Tabelle.

Eine Zeile beantwortet sechs Fragen: den stabilen Bezeichner des Akteurs, den Aktionsnamen (`Approval.Approve` , `Document.Download`), Zieltyp und Ziel-ID, den UTC-Zeitstempel der Serveruhr, Client-Adresse und User-Agent sowie die Korrelations-ID. Halten Sie auch das Ergebnis fest, denn eine Ablehnung ist oft interessanter als ein Erfolg. In Wisej.NET stammen Netzwerkadresse und Browser-User-Agent des Clients aus dem Anwendungskontext der jeweiligen Session und nicht aus irgendetwas, das die Seite herschickt; erfassen Sie sie daher einmal, wenn der `SessionContext` nach dem Login erzeugt wird. Nehmen Sie auch den Akteur aus diesem verifizierten Kontext – nie aus einem Control oder einem vom Aufrufer gelieferten Parameter.

Sorgen Sie dann dafür, dass sich die Zeile kaum ungeschehen machen lässt. Das Datenbank-Login der Anwendung erhält `INSERT` und sonst nichts, sodass ein Bug oder eine eingeschleuste Anweisung nicht stillschweigend einen ganzen Dienstag löschen kann. Geben Sie jeder Zeile eine lückenlose Sequenznummer und einen Hash über ihre eigenen Felder plus den Hash der vorherigen Zeile; ein geplanter Durchlauf berechnet die Kette neu und nennt die erste Zeile, die nicht mehr passt. Das ist ein Manipulations*nachweis*, kein Schutz – eine Änderung wird sichtbar, nicht unmöglich. Schreiben Sie die Audit-Zeile schließlich in derselben Transaktion wie die fachliche Änderung, sodass eine Genehmigung ohne zugehörigen Eintrag gar nicht entstehen kann.

```
using (_log.BeginScope(new Dictionary<string, object>
    { ["CorrelationId"] = ctx.CorrelationId, ["UserId"] = ctx.UserId }))
using (var tx = _db.BeginTransaction())
{
    _approvals.Approve(requestId, ctx);           // die fachliche Änderung
    _audit.Record(AuditEntry.For(ctx,             // Akteur, Adresse, Korrelation
        "Approval.Approve", "ApprovalRequest", requestId));
    tx.Commit();                                 // beides oder keins
}
```

Eine ID, eine Geschichte

Die Korrelations-ID wird dort erzeugt, wo ein Benutzerbefehl beginnt – am Anfang eines schlanken Event-Handlers, bevor irgendein Service aufgerufen wird. Eine `Guid` genügt; wenn Sie bereits mit `System.Diagnostics.Activity` arbeiten, verwenden Sie den Trace-Bezeichner der aktuellen Activity, damit die ID zu allem anderen passt, was das W3C-Trace-Format spricht. Legen Sie sie in den befehlsbezogenen Kontext, der auch den Akteur und die Client-Adresse trägt, und übergeben Sie diesen Kontext an jeden Service. Greifen Sie aus einem Service heraus nicht zurück in die Session:

Hintergrundverarbeitung, die mit `Application.StartTask` gestartet wurde, kann die Session überleben, die sie gestartet hat.

Die ID an Logzeilen anzuhängen, ist nicht Aufgabe des Aufrufers. Öffnen Sie einen Logging-Scope um den Befehl – `ILogger.BeginScope` mit einem Eigenschafts-Dictionary oder Serilog's `LogContext` mit einem Enricher –, und jede darin geschriebene Zeile trägt automatisch die Korrelations-ID und den Akteur, einschließlich der Zeilen aus Bibliotheken unter Ihnen. Verwenden Sie strukturiertes Logging, damit die ID in der Senke eine benannte Eigenschaft ist; so wird „zeig mir alles zu Korrelation 7f3c“ zu einer Abfrage statt zu einem grep. Und machen Sie sie sichtbar: Wenn ein Befehl fehlschlägt, beenden Sie den freundlichen Satz mit der ID als Referenz.

Secrets gehören in die Umgebung, nicht ins Repository

AccessOps hält inzwischen drei Credentials: den Datenbank-Connection-String, das OIDC-Client-Secret aus Modul 4 und den Signaturschlüssel für die ablaufenden Freigabelinks aus Modul 5. Keines davon gehört in eine versionierte Datei. Verwenden Sie in der Entwicklung .NET User Secrets – das Projekt erhält eine `UserSecretsId`, und die Werte liegen außerhalb des Solution-Ordners, sodass ein Klon des Repositorys keine Credentials enthält. Stellen Sie sie im Deployment als Umgebungsvariablen bereit oder lesen Sie sie über einen Konfigurationsprovider aus einem verwalteten Vault. Noch besser ist eine Managed Identity:

Die App authentifiziert sich als sie selbst und erhält ein kurzlebiges Token, es gibt also kein gespeichertes Secret, das durchsickern könnte.

Das funktioniert, weil die ASP.NET-Core-Konfiguration geschichtet ist und spätere Quellen gewinnen: `appsettings.json`, dann die Umgebungsüberlagerung, dann User Secrets in der Entwicklung, dann Umgebungsvariablen, dann ein Vault-Provider. Die eingecheckten Dateien enthalten Struktur und harmlose Standardwerte; die Werte, die nach Secret aussehen, kommen pro Umgebung hinzu. Die Wisej.NET-eigene `Default.json` konfiguriert die Hülle – Theme, Session-Timeout, Startfenster – und ist ebenso wenig ein Secret-Speicher wie `Web.config`.

Ein Secret, das Sie nicht in Ruhe rotieren können, werden Sie um drei Uhr morgens schlecht rotieren; das Rotationsverfahren ist hier deshalb ein Ergebnis. Halten Sie pro Secret fest: den Verantwortlichen, den Speicherort, das Ablaufdatum und die Schritte zum Ersetzen. Planen Sie für zwei gleichzeitig gültige Credentials – ein zweites Client-Secret, ein zweites Datenbank-Login –, damit die Rotation ein rollierender Neustart ist und kein Ausfall. Und wenn ein Wert einmal eingecheckt wurde, hilft es nicht, die Zeile zu löschen: Rotieren Sie ihn und widerrufen Sie den alten.

Protokollieren Sie den Vorfall, nicht die Person

Logs werden in Suchindizes kopiert, an Drittanbieter-Dienste übertragen, auf Support-Laptops geöffnet und weit länger aufbewahrt, als irgendjemand beabsichtigt. Sie müssen genug enthalten, um einen Vorfall zu rekonstruieren, und nichts, was für sich allein Schaden anrichtet. Protokollieren Sie nie ein Passwort (auch kein falsches), ein Token, einen Authorization-Header, ein Cookie, einen Connection-String oder den Inhalt eines vertraulichen Dokuments.

Personenbezogene Daten verlangen dieselbe Disziplin aus einem anderen Grund: Eine Audit-Zeile ist ein rechtmäßiger Nachweis mit Aufbewahrungsrichtlinie, während eine Debug-Zeile mit der Privatadresse eines Mitarbeiters eine unkontrollierte Kopie davon ist. Protokollieren Sie die stabile Benutzer-ID statt Name und E-Mail und die Dokument-ID statt eines Dateinamens, der selbst sensibel sein kann. Strukturiertes Logging hilft, weil Sie benannte Eigenschaften aus einer Allow-List protokollieren, statt ein ganzes Request-Objekt zu serialisieren und auf das Beste zu hoffen.

- Nie protokollieren: Passwörter, Tokens, Cookies, Connection-Strings, Dokumentinhalte.
- Per Referenz protokollieren: Benutzer-ID, Dokument-ID, Antrags-ID – keine Namen oder personenbezogenen Freitexte.
- In der Senke durchsetzen: eine Maskierungsrichtlinie plus ein Test, der sicherstellt, dass eine bekannte Secret-Markierung nie in der Ausgabe landet.

Schließen Sie den Kreis mit einem Review statt mit einer Regel: Gehen Sie die Logausgabe einer echten Genehmigung und eines echten Downloads von Anfang bis Ende durch und markieren Sie jedes Feld als notwendig, geschwärzt oder entfernt.



Sicherheitsgewohnheit

Dem Browser wird nie vertraut. Jede relevante Entscheidung – wer der Benutzer ist, was er tun darf, welchen Datensatz er sehen darf – wird auf dem Server in einem Service getroffen, und die UI spiegelt sie nur wider. Existiert eine Regel nur als versteckter Button, dann existiert sie nicht.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ für jeden sensiblen Befehl mit einem manipulationssicheren Audit-Trail festhalten, wer was wann und von wo getan hat.
- ✓ eine Korrelations-ID durch die Logs tragen, sodass sich eine einzelne Benutzeraktion über Services hinweg nachverfolgen lässt.
- ✓ Secrets mithilfe von Umgebungskonfiguration, Secret-Speichern und Managed Identities aus der Versionsverwaltung heraushalten.
- ✓ genug protokollieren, um einen Vorfall zu untersuchen, ohne Passwörter, Tokens oder personenbezogene Daten zu protokollieren.

Lab- und Prüfungsleitfaden

Was Sie im praktischen Lab erstellen, das vorgeschlagene Vorgehen und die geforderten Ergebnisse.



Praxis-Lab

Ergänzen Sie AccessOps um Auditing und Secret-Verwaltung: einen AuditService, der Genehmigungen, Berechtigungsänderungen und Dokument-Downloads mit Benutzer, Uhrzeit, Adresse und Korrelations-ID festhält, eine Append-only-Audit-Tabelle, eine pro Befehl erzeugte Korrelations-ID, die in jede Logzeile geschrieben wird, Connection-Strings und das OIDC-Client-Secret, die in die Umgebungsconfiguration verlagert werden, mit einem dokumentierten Rotationsverfahren, und ein Log-Review, das sensible Felder schwärzt.

Ausgangspunkt

Sie öffnen die AccessOps-Solution in dem Stand am Ende von Modul 5: ein verifizierter `SessionContext`, der beim Login erzeugt wird, ein `PermissionService` und ein Autorisierungs-Guard in `DocumentService` und `ApprovalService`, OIDC-Anmeldung und die Dokumentenfunktion mit ihrer Upload-Pipeline und signierten Freigabelinks. Der Genehmigungsbildschirm, der Berechtigungseditor der Administration und die Dokumentenliste funktionieren alle – sie hinterlassen nur keinerlei Spur dessen, was sie getan haben, und der Datenbank-Connection-String sowie das OIDC-Client-Secret stehen immer noch in einer Konfigurationsdatei, die von der Versionsverwaltung erfasst wird.

Vorgeschlagenes Vorgehen

- ✓ Legen Sie die Audit-Tabelle mit lückenloser Sequenz, Akteur, Aktion, Zieltyp und Ziel-ID, UTC-Zeitstempel, Client-Adresse, User-Agent, Ergebnis, Korrelations-ID und einer Hash-Spalte an und geben Sie dem Datenbank-Login der Anwendung nur `INSERT` – kein `UPDATE`, kein `DELETE`.
- ✓ Schreiben Sie den `AuditService` als einzigen Schreiber dieser Tabelle: Er nimmt Akteur und Client-Details aus dem verifizierten `SessionContext`, berechnet den Hash jeder Zeile über ihre eigenen Felder plus den Hash der vorherigen Zeile und nimmt das Insert in dieselbe Transaktion auf wie die fachliche Änderung.
- ✓ Ergänzen Sie einen befehlsbezogenen Kontext, der am Anfang jedes Event-Handlers eine Korrelations-ID erzeugt, übergeben Sie ihn an `ApprovalService`, `DocumentService` und den Berechtigungseditor und öffnen Sie einen Logging-Scope, damit jede während des Befehls geschriebene Zeile die ID und den Akteur trägt.
- ✓ Verlagern Sie Connection-String, OIDC-Client-Secret und den Signaturschlüssel der Freigabelinks aus den versionierten Konfigurationsdateien in User Secrets für die Entwicklung und Umgebungsvariablen für das Deployment, und schreiben Sie dann das Rotationsverfahren mit Verantwortlichen, Ablaufdaten und der Überlappung zweier gültiger Credentials.
- ✓ Rufen Sie den `AuditService` beim Genehmigen, beim Erteilen und Entziehen von Berechtigungen und beim Dokument-Download auf – Ablehnungen ebenso wie Erfolge –, führen Sie dann eine Genehmigung und einen Download aus und prüfen Sie jede dabei entstandene Logzeile auf Passwörter, Tokens, Header, Connection-Strings und personenbezogene Daten.

Geforderte Ergebnisse

- ✓ AuditService und Append-only-Audit-Tabelle
- ✓ Korrelations-ID in jedem Befehl und jeder Logzeile
- ✓ In die Umgebungsconfiguration verlagerte Secrets
- ✓ Verfahren zur Rotation von Secrets
- ✓ Review der Log-Schwärzung

Wiederholungsfragen

- ✓ Ihre Audit-Tabelle ist für die Anwendung nur beschreibbar per Insert, und jede Zeile ist per Hash verkettet. Nennen Sie etwas, das ein Angreifer mit dem eigenen Datenbank-Login der Anwendung dem Trail trotzdem antun kann, und erklären Sie genau, wie der Prüfdurchlauf es sichtbar machen würde.
- ✓ Eine Genehmigung wird auf dem Genehmigungsbildschirm gestartet und in einem mit `Application.StartTask` gestarteten Hintergrund-Task abgeschlossen, nachdem sich der Benutzer abgemeldet hat. Woher bekommt dieser Task seine Korrelations-ID und seinen Akteur, und was geht kaputt, wenn er sie stattdessen aus der Session liest?
- ✓ Ein Entwickler entfernt das OIDC-Client-Secret aus der versionierten Konfigurationsdatei und pusht die Änderung. Warum ist das Secret trotzdem noch kompromittiert, und was muss geschehen, bevor der Vorfall tatsächlich abgeschlossen ist?



Sicherheitsgewohnheit

Dem Browser wird nie vertraut. Jede relevante Entscheidung – wer der Benutzer ist, was er tun darf, welchen Datensatz er sehen darf – wird auf dem Server in einem Service getroffen, und die UI spiegelt sie nur wider. Existiert eine Regel nur als versteckter Button, dann existiert sie nicht.