SICHERHEITS- UND AUTHENTIFIZIERUNGSMUSTER · MODUL 7

Design für den Produktivbetrieb und Abschlussprojekt

Lektionsleitfaden · Lab- und Prüfungsleitfaden

LearnWisej

Lektionsleitfaden

Zwei Einstiegspunkte mit einer Ressourcenrichtlinie, Skalierungs- und Fehlerverhalten über mehrere Knoten, Timeouts und die Zuständigkeit für den Entzug, korrelierte Nachweise bei Vorfällen sowie ein geprobtes Release und Rollback für das AccessOps-Abschlussprojekt. Was das für Entwicklerinnen und Entwickler bedeutet, die die Anmelde-, Identitäts- und Autorisierungspfade einer Wisej.NET-Anwendung im Produktivbetrieb absichern – und wie Sie dieses Modul lesen.

Fokus: Produktivbetrieb

Zwei Einstiegspunkte mit einer Ressourcenrichtlinie, Skalierungs- und Fehlerverhalten über mehrere Knoten, Timeouts und die Zuständigkeit für den Entzug, korrelierte Nachweise bei Vorfällen sowie ein geprobtes Release und Rollback für das AccessOps-Abschlussprojekt.

1. Das Problem im Produktivbetrieb

Jedes frühere Modul hat einen Pfad abgesichert. Im Produktivbetrieb bedient AccessOps Mitarbeitende über eine Windows-Route im Intranet und Partner über eine öffentliche HTTPS-Route gleichzeitig, auf mehreren Knoten, neben einem Verzeichnisdienst, einem Identitätsanbieter und einem Credential-Speicher, die jeweils nach eigenem Zeitplan ausfallen. Ein Release ist nur sicher, wenn beide Routen dieselbe serverseitige Entscheidung erreichen, wenn weder ein Knotenausfall noch ein veralteter Tab zu einem Zugriff werden kann und wenn das Team das in einem Review beweisen kann.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Gemeinsame Ressourcenrichtlinie	Eine einzige Prüfung von Grant, Mandant und Datensatz in der Service-Schicht, die beide Routen aufrufen, sodass ein Partner und ein Mitarbeiter vom selben Code abgelehnt werden, bevor ein Repository läuft.
Routenmatrix	Eine Tabelle pro Einstiegspunkt mit Ziel-Framework, IIS-Modus, Challenge, Host-Principal, Anwendungs-Principal, ausführender Identität und Zuständigkeit für den Entzug; ein Reviewer prüft sie Zeile für Zeile.
Session-Affinität	Wisej.NET hält jede Session im Speicher eines Knotens; jede Anfrage einer Session muss also denselben Knoten erreichen, und fällt dieser Knoten aus, ist die Session verloren.
Dauerhafte gemeinsame Instanz	Grants, Entzüge, Übergabe-Handles der Anmeldung, Passkey-Challenges und Credentials leben im Speicher der Anwendung, nicht im Session-Zustand, sodass ein Neustart oder Failover keine Entscheidung vergessen kann.
Gemeinsame Schlüssel zum Cookie-Schutz	Die Data-Protection-Schlüssel von ASP.NET Core und der Anwendungsname müssen auf jedem Knoten übereinstimmen, der Authentifizierungs- oder Korrelations-Cookies eines anderen Knotens validiert.
Erste scheiternde Grenze	Die früheste Komponente, die eine Ablehnung ausgegeben hat (Ingress, Richtlinie, Betriebssystem-Token, Callback-Route oder Passkey-Validierung) – dort beginnt eine Untersuchung.
Abbruchkriterium	Ein vor dem Rollout vereinbartes Ergebnis, etwa ein mandantenübergreifender Lesezugriff, das die Freigabe blockiert und das geprobte Rollback auslöst.

3. Der rote Faden der Lektion

Zwei Routen, eine Richtlinie

Die Intranet-Route läuft unter IIS mit deaktiviertem anonymem Zugriff und aktivierter Windows-Authentifizierung. Der Browser verhandelt über den kanonischen Hostnamen, und Wisej.NET stellt diesen Host-Principal, dessen Identität eine `WindowsIdentity` ist, als `Application.User` bereit. AccessOps ordnet die Verzeichnisgruppen einem Grant in seinem eigenen Speicher zu und etabliert einen Anwendungs-Principal mit Claims für Konto, Mandant und Grant, wie in *Identitätsgrenzen in Wisej.NET*. Das effektive Windows-Token ist ein dritter Fakt: Es existiert nur innerhalb eines begrenzten Aufrufs von `WindowsIdentity.RunImpersonated` für die eine Dateifreigabe, die die eigenen Rechte des Benutzers braucht (*Windows-Impersonation und Delegierung*); überall sonst läuft der Code als Pool-Identität.

Die Partner-Route ist öffentliches HTTPS. Ein Partner meldet sich mit der lokalen Anmeldung aus *Eigene Anmeldung und Session-Sicherheit* oder über den Anbieter an, mit den Cookie- und OpenID-Connect-Handlern von ASP.NET Core, registriert vor `app.UseWisej()` (*Föderation mit OpenID Connect*); ein verifiziertes Konto kann dann über die Erweiterung WebAuthn einen Passkey registrieren (*Passkeys mit der Erweiterung WebAuthn*). Beide Pfade enden gleich: Beim Laden einer Seite rotiert

`Application.RefreshSessionId()` die Session-ID, und der Anwendungs-Principal wird gesetzt. Ab dann können die Services die Routen nicht mehr unterscheiden, und jeder geschützte Lesezugriff durchläuft eine einzige Prüfung.

```
public Record Read(string tenantId, int recordId)
{
    var principal = Application.User as ClaimsPrincipal;
    var grant = _grants.Find(principal?.FindFirst("grant")?.Value); // dauerhafter Speicher
    if (grant == null || grant.Revoked || grant.ExpiresAt ≤ _clock.UtcNow
        || !grant.CanRead || grant.TenantId ≠ tenantId)
    {
        _audit.Deny("policy", grant?.Id, tenantId, recordId);
        return null;                                  // das Repository läuft nie
    }
    return _records.Get(tenantId, recordId);
}
```

Das Diagramm der Vertrauensgrenzen markiert, wo jede Identität entsteht, was das Vertrauen herstellt, wie lange sie lebt und wer sie entzieht: Der Verzeichnisdienst besitzt die Gruppenmitgliedschaft, der Anbieter das externe Konto des Partners, der Credential-Speicher die Passkeys und AccessOps den Grant. Einen Partner beim Anbieter zu deaktivieren beendet keinen bereits ausgestellten Grant; die Admin-Konsole entzieht deshalb auch den Grant.

Skalierungs- und Fehlerverhalten

Legen Sie fest, was Sie getestet haben: `Wisej-4` und `Wisej-4-WebAuthn` 4.1.7, den zur Laufzeit passenden OpenID-Connect-Handler und die Laufzeit selbst. Rufen Sie hinter dem Load Balancer `app.UseForwardedHeaders()` auf und vertrauen Sie nur den bekannten Proxys, denn kanonisches Schema und kanonischer Host fließen in die OIDC-Redirect-URI und in die Origin-Prüfung von WebAuthn ein. Wisej.NET verlangt Session-Affinität – ein Sticky-Cookie, einen Hash der Client-IP oder eine andere konsistente Route –, und manche Load Balancer brauchen für den WebSocket den TCP-Modus. Veröffentlichen Sie mit `"secure": true` in Default.json und halten Sie `.json`-Dateien vom statischen Dateiserver fern.

Ein ausgefallener Knoten verliert seine Sessions; nichts, was überleben muss, darf also in `Application.Session` liegen. Teilen Sie den Data-Protection-Schlüsselring und den Anwendungsnamen über alle Knoten, damit ein von Knoten A geschriebenes Korrelations- oder Authentifizierungs-Cookie auf Knoten B validiert. Ein Callback des Anbieters, der auf Knoten B landet, hat zwei sichere Ausgänge: Der Handler validiert ihn, und das einmal verwendbare Übergabe-Handle im gemeinsamen Speicher setzt die Anmeldung dort in einer neuen Session fort, oder das Handle fehlt oder ist bereits verbraucht, und der Benutzer erhält einen einfachen Fehler und eine neue Anmeldung. Es vergibt nie zweimal einen Grant und startet nie aus dem eingehenden Code einen neuen Versuch. Timeouts gehören in dieselbe Tabelle: der Wisej-`sessionTimeout`, der absolute Ablauf des Grants, die `ExpireTimeSpan` des Host-Cookies, die Transaktion beim Anbieter und das Timeout der WebAuthn-Zeremonie, jeweils mit einer Zuständigkeit für den Entzug.

Eine Vorfallmatrix nach der ersten scheiternden Grenze

Beginnen Sie bei der ersten Komponente, die Nein gesagt hat. Ein 401 am Ingress verweist auf die Challenge, das IIS-Statustripel, `setspn -Q` und `klist`. Eine authentifizierte Anfrage, gefolgt von einer fachlichen Ablehnung, verweist auf einen Richtlinien-Trace: Grant, Mandant, Datensatz und null Repository-Aufrufe. Ein Fehler bei einer entfernten Datei nach lokalem Erfolg verweist auf das effektive Token. Ein Callback, der die Anwendung nie erreicht, verweist auf die kanonische URL, die exakte Redirect-URI und die Forwarded Headers; ein gefälschter endet in `OnRemoteFailure`. Eine nach der Abfrage abgewiesene Passkey-Antwort verweist auf RP-ID, Origin, Challenge, `ClientData.Type` und den Speicher.

Jede Zeile gibt eine Ereigniskategorie aus einer Allowlist mit einer Korrelations-ID und einer opaken Konto-ID aus; weil der Logging-Scope von ASP.NET Core pro WebSocket-Verbindung gilt, öffnet AccessOps pro Klick einen eigenen `ILogger.BeginScope`. Session-IDs, Codes, Tokens und Cookies erreichen nie den geschützten Audit-Speicher, der eine festgelegte Aufbewahrungsfrist hat.

Release, Abbruchkriterien und Rollback

Bevor Sie die Partner-Route freigeben, führen Sie die Negativmatrix mit 12 Fällen plus einem Normalfall pro Grenze aus und rollen Sie dann für einen Test-Mandanten aus. Schreiben Sie zuerst die Abbruchkriterien:

- ein mandantenübergreifender Lesezugriff oder eine Ablehnung, die erst nach dem Lauf des Repositorys erfolgt;
- ein wiederholter Callback oder eine wiederholte Assertion, die als neue Anmeldung akzeptiert wird;
- ein gelöschtes Credential oder ein entzogener Grant, das bzw. der weiterhin eine geschützte Aktion öffnet;
- Anmeldematerial in einem Log, einer ausgeblendeten Grid-Spalte oder `Application.Browser.UserData`.

Das Rollback schaltet die Partner-Route oder die neue Funktion am Ingress ab, während die getestete Intranet-Basis weiterläuft. Grants und Entzüge leben im dauerhaften Speicher, nicht im Build; das Zurückrollen von Code kann einen entzogenen Grant also nicht reaktivieren, und das Runbook beweist das, indem es danach den Fall des veralteten Tabs wiederholt. Das Runbook nennt die zuständige Person und die genaue Aktion für den Notfallentzug, und ein ausfallender Anbieter erhält begrenzte Wiederholungen und eine einfache Fehlermeldung. Fälle, die IIS, die Domäne oder einen echten Anbieter brauchen und auf Fixtures oder im PolicyHarness liefen, werden mit einem genauen Plan für den erneuten Lauf gekennzeichnet; das mündliche Review verteidigt die Nachweise dann in zehn Minuten.

> **Sicherheitsregel**
>
> Halten Sie die Identitäten auseinander und entscheiden Sie auf dem Server. Der Host-Principal, der Wisej.NET-Anwendungs-Principal und das effektive Windows-Token sind drei verschiedene Fakten, und keines davon ist eine Berechtigung. Jeder geschützte Lesezugriff prüft den aktuellen Grant, den Mandanten und den Datensatz in der Service-Schicht, bevor das Repository läuft, und ein ausgeblendeter Button, ein Rollen-Claim oder ein abgemeldeter Tab beweist nichts, solange diese Prüfung nicht ablehnt.

Lernziele

Am Ende dieses Moduls können Sie:

- ✓ Die Windows-Route im Intranet und die HTTPS-Route für Partner mit einer gemeinsamen Mandanten- und Ressourcenrichtlinie entwerfen und darstellen, wo jede Identität entsteht, Vertrauen erhält, abläuft und entzogen wird.
- ✓ Skalierung und Fehlerfälle ausdrücklich machen: festgelegte Versionen, Forwarded Headers, WebSocket-Affinität, gemeinsame Schlüssel zum Cookie-Schutz und ein sicheres Ergebnis für einen Callback, der auf einem anderen Knoten landet.
- ✓ Eine Vorfallmatrix nach der ersten scheiternden Grenze aufbauen – Challenge am Ingress, Ablehnung durch die Richtlinie, effektives Token, Callback-Route, Passkey-Validierung – mit geschwärzten, korrelierten Audit-Ereignissen.
- ✓ Die Negativmatrix mit 12 Fällen ausführen, Abbruchkriterien für das Release und ein Rollback festlegen, das entzogene Grants entzogen hält, und das Release in einem mündlichen Review verteidigen.

Lab- und Prüfungsleitfaden

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

Praktisches Lab

Schließen Sie das AccessOps-Abschlussprojekt auf dem kumulativen Portal ab: Zeichnen Sie die Vertrauensgrenzen mit Verantwortlichen für IIS, den Verzeichnisdienst, den Identitätsanbieter, die Anwendung und den Credential-Speicher, schreiben Sie die Routenmatrix (Ziel-Framework, IIS-Modus, Challenge, Host-Principal, Anwendungs-Principal, ausführende Identität, Zuständigkeit für den Entzug) und führen Sie die Nachweismatrix mit 12 Fällen aus der Aufgabenstellung aus – Lesezugriff eines Readers im Intranet, Ablehnung nach Authentifizierung durch den Host, Mandant A fordert Datensatz 17 von Mandant B an, begrenzter Lesezugriff mit Zurücksetzen, zweiter Hop, Rotation der Session, entzogener Grant aus einem veralteten Tab, externe Rücksprung-URL und gefälschter POST, ein einziger Grant pro föderiertem Callback, nicht genehmigter Mandant oder ungültiger State, Passkey-Wiederholung, falscher Besitzer oder gelöschtes Credential sowie ein Callback nach Neustart oder auf dem zweiten Knoten –, und halten Sie für jeden Fall die Grenze fest, die abgelehnt hat, sowie den Nachweis, dass kein Datenzugriff und kein Grant folgte. Simulieren Sie einen Fehler wie einen falschen SPN, einen Knotenausfall während eines Callbacks oder einen verlorenen Authenticator, üben Sie das Runbook und präsentieren Sie die Erlaubnis- und Ablehnungs-Traces und das Rollback in einem zehnminütigen Review anhand der Checklisten für das Sicherheits-Review; Fälle, die IIS, die Domäne oder einen echten Anbieter brauchen und auf Fixtures oder im PolicyHarness liefen, werden mit einem genauen Plan für den erneuten Lauf gekennzeichnet.

Ausgangspunkt

Öffnen Sie das kumulative AccessOps-Portal, das Sie in den vorherigen sechs Modulen gebaut haben, mit seiner Windows-Route im Intranet, der Anmeldung für Partner, der OpenID-Connect-Übergabe, dem Passkey-Speicher und der gemeinsamen Datensatzrichtlinie, dazu das eigenständige PolicyHarness. Verwenden Sie zwei Mandanten (A und B), die beide einen Datensatz 17 enthalten, die reservierten Fixtures F01–F12 und entweder einen zweiten Knoten oder eine aufgezeichnete Übung mit Neustart und Umleitung.

Vorgeschlagenes Vorgehen

- ✓ Zeichnen Sie zuerst das Diagramm der Vertrauensgrenzen: Browser, IIS oder Proxy, Wisej.NET-Session, Identitätsanbieter, Credential-Speicher und Datensatz-Backend, mit einem Verantwortlichen, einem Vertrauensanker, einer Lebensdauer und einer Entzugsaktion für jede Identität, die eine Linie überquert.
- ✓ Füllen Sie pro Einstiegspunkt eine Zeile der Routenmatrix aus und legen Sie die getesteten Versionen fest; schreiben Sie dann die bereinigte Konfiguration (Forwarded Headers, Affinität, gemeinsame Data-Protection-Schlüssel, `"secure": true`) und die Zuordnung von Rollen und Mandanten auf.
- ✓ Führen Sie die 12 Fälle der Reihe nach aus und füllen Sie jede Zeile der Nachweismatrix mit Test-Alias, Einstiegspunkt, erwartetem und tatsächlichem Ergebnis, der ablehnenden Grenze, der

Korrelationsbezeichnung, der Anzahl der Repository-Aufrufe und der Angabe, ob der Fall live, auf einer Fixture oder im PolicyHarness lief.

- ✓ Speisen Sie einen Fehler ein (einen falschen SPN, einen Knotenausfall während eines Callbacks des Anbieters oder einen verlorenen Authenticator), folgen Sie der Vorfallmatrix ab der ersten scheiternden Grenze und messen Sie die Wiederherstellungszeit im Runbook.
- ✓ Proben Sie das Rollback: Schalten Sie die Partner-Route ab, bestätigen Sie, dass die Intranet-Basis weiterhin Reader bedient, wiederholen Sie den Fall des veralteten Tabs, um zu zeigen, dass der entzogene Grant entzogen bleibt, und präsentieren Sie dann das zehnminütige Review.

Geforderte Ergebnisse

- ✓ Diagramm der Vertrauensgrenzen und Routenmatrix
- ✓ Kumulatives AccessOps-Portal mit bereinigter Konfiguration, Zuordnung von Rollen und Mandanten und Richtlinientests
- ✓ Nachweismatrix mit 12 Fällen, ablehnender Grenze und Fixture-Kennzeichnung
- ✓ Runbook für Deployment und Rollback mit Verantwortlichen und Abbruchkriterien
- ✓ Ausgefüllte Checkliste für das Sicherheits-Review und nach dem Bewertungsschema bewertete Notizen zum mündlichen Review

Wiederholungsfragen

- ✓ Der Callback eines Partners erreicht Knoten B, nachdem Knoten A mitten in der Anmeldung ausgefallen ist. Gehen Sie durch, was Knoten B prüfen kann, was nicht, und warum das Ergebnis trotzdem ein Grant oder keiner ist.
- ✓ Ihr Rollback stellt den vorherigen Build und die vorherige Konfiguration wieder her. Welchen Zustand darf es nicht wiederherstellen, und welche Nachweiszeile beweist, dass ein entzogener Grant nicht zurückgekehrt ist?
- ✓ Fall 5 lief gegen eine Fixture, weil das Lab keine Domäne hat. Was genau würden Sie auf der echten Topologie erneut ausführen, und welches Ergebnis dort würde die Abnahme blockieren?

Sicherheitsregel

Halten Sie die Identitäten auseinander und entscheiden Sie auf dem Server. Der Host-Principal, der Wisej.NET-Anwendungs-Principal und das effektive Windows-Token sind drei verschiedene Fakten, und keines davon ist eine Berechtigung. Jeder geschützte Lesezugriff prüft den aktuellen Grant, den Mandanten und den Datensatz in der Service-Schicht, bevor das Repository läuft, und ein ausgeblendeter Button, ein Rollen-Claim oder ein abgemeldeter Tab beweist nichts, solange diese Prüfung nicht ablehnt.