SECURITY & AUTH PATTERNS · MODULE 7

Production Design and Capstone

Lesson Guide · Lab / Exam Guide

LearnWisej

Lesson Guide

Two entry points with one resource policy, scale and failure behavior across nodes, timeouts and revocation ownership, correlated incident evidence, and a rehearsed release and rollback for the AccessOps capstone. What this means for a developer securing the sign-in, identity and authorization paths of a production Wisej.NET app — and how to read this module.

Production focus

Two entry points with one resource policy, scale and failure behavior across nodes, timeouts and revocation ownership, correlated incident evidence, and a rehearsed release and rollback for the AccessOps capstone.

1. The production problem

Each earlier module secured one path. In production AccessOps serves staff through an intranet Windows route and partners through a public HTTPS route at once, on several nodes, beside a directory, an identity provider and a credential store, each failing on its own schedule. A release is safe only when both routes reach the same server-side decision, a node loss or a stale tab cannot become access, and the team can prove it in a review.

2. Key concepts

Concept	Why it matters
Shared resource policy	One service-boundary check of grant, tenant and record that both routes call, so a partner and a staff member are denied by the same code before any repository runs.
Route matrix	A table per entry point of framework target, IIS mode, challenge, host principal, app principal, execution identity and revocation owner; a reviewer checks it row by row.
Session affinity	Wisej.NET keeps each session in one node's memory, so every request of a session must reach the same node; losing that node loses the session.
Durable shared authority	Grants, revocations, sign-in hand-off handles, passkey challenges and credentials live in the application's store, not in session state, so a restart or failover cannot forget a decision.
Shared cookie-protection keys	ASP.NET Core Data Protection keys and the application name must match on every node that validates another node's authentication or correlation cookies.
First failing boundary	The earliest component that emitted a deny (ingress, policy, OS token, callback route or passkey validation), where an investigation starts.
Stop condition	A result agreed before rollout, such as a cross-tenant read, that blocks promotion and triggers the rehearsed rollback.

3. Lesson narrative

Two routes, one policy

The intranet route runs on IIS with anonymous access disabled and Windows authentication enabled. The browser negotiates on the canonical host name and Wisej.NET exposes that host principal, whose identity is a `WindowsIdentity`, as `Application.User`. AccessOps maps the directory groups to a grant in its own store and establishes an application principal carrying account, tenant and grant claims, as in *Identity Boundaries in Wisej.NET*. The effective Windows token is a third fact: it exists only inside a scoped `WindowsIdentity.RunImpersonated` call for the one file share that needs the user's own rights (*Windows Impersonation and Delegation*), and elsewhere code runs as the pool identity.

The partner route is public HTTPS. A partner signs in with the local sign-in from *Custom Sign In and Session Security* or through the provider, with ASP.NET Core's cookie and OpenID Connect handlers registered before `app.UseWisej()` (*Federation with OpenID Connect*); a verified account may then enroll a passkey through the WebAuthn extension (*Passkeys with the WebAuthn Extension*). Both paths end the same way: on a page load, `Application.RefreshSessionId()` rotates the session ID and the app principal is set. From then on the services cannot tell the routes apart, and every protected read goes through one check.

```
public Record Read(string tenantId, int recordId)
{
    var principal = Application.User as ClaimsPrincipal;
    var grant = _grants.Find(principal?.FindFirst("grant")?.Value); // durable store
    if (grant == null || grant.Revoked || grant.ExpiresAt ≤ _clock.UtcNow
        || !grant.CanRead || grant.TenantId ≠ tenantId)
    {
        _audit.Deny("policy", grant?.Id, tenantId, recordId);
        return null;                                  // repository never runs
    }
    return _records.Get(tenantId, recordId);
}
```

The trust-boundary diagram marks where each identity originates, what establishes trust, how long it lives and who revokes it: the directory owns group membership, the provider owns the partner's external account, the credential store owns passkeys, and AccessOps owns the grant. Disabling a partner at the provider does not end a grant already issued, so the admin console revokes the grant too.

Scale and failure behavior

Pin what you tested: `Wisej-4` and `Wisej-4-WebAuthn` 4.1.7, the OpenID Connect handler matching the runtime, and the runtime itself. Behind the load balancer, call `app.UseForwardedHeaders()` trusting only the known proxies, because the canonical scheme and host feed the OIDC redirect URI and the WebAuthn origin check. Wisej.NET requires session affinity — a sticky cookie, a client-IP hash or another consistent route — and some balancers need TCP mode for the WebSocket. Publish with `"secure": true` in Default.json and keep `.json` files out of the static file server.

A lost node loses its sessions, so nothing that must survive may live in `Application.Session`. Share the Data Protection key ring and application name across nodes so a correlation or authentication cookie written by node A validates on node B. A provider callback landing on node B has two safe outcomes: the handler validates it and the single-use hand-off handle in the shared store resumes sign-in in a new session there, or the handle is missing or already consumed and the user gets a plain failure and a fresh sign-in. It never grants twice or starts a new attempt from the incoming code. Timeouts belong in the same table: the Wisej `sessionTimeout`, the grant's absolute expiry, the host cookie's `ExpireTimeSpan`, the provider transaction and the WebAuthn ceremony timeout, each with a revocation owner.

An incident matrix keyed to the first failing boundary

Start from the first component that said no. A 401 at ingress points to the challenge, the IIS status triplet, `setspn -Q` and `klist`. An authenticated request followed by a business deny points to a policy trace: grant, tenant, record and a repository call count of zero. A remote file failure after a local success points to the effective token. A callback that never reaches the app points to the canonical URL, exact redirect URI and forwarded headers; a forged one ends in `OnRemoteFailure`. A passkey response rejected after the prompt points to RP ID, origin, challenge, `ClientData.Type` and the store.

Each row emits one allowlisted event category with a correlation ID and an opaque account ID; because the ASP.NET Core logging scope is per WebSocket connection, AccessOps opens its own

`ILogger.BeginScope` per click. Session IDs, codes, tokens and cookies never reach the protected audit store, which has a stated retention period.

Release, stop conditions and rollback

Before exposing the partner route, run the 12-case negative matrix plus one normal case per boundary, then roll out to a test tenant. Write the stop conditions first:

- a cross-tenant read, or any deny that happens after the repository ran;
- a replayed callback or assertion accepted as a fresh sign-in;
- a deleted credential or revoked grant that still opens a protected action;
- credential material in a log, a hidden grid column or `Application.Browser.UserData`.

The rollback switches the partner route or the new feature off at ingress while the tested intranet baseline keeps running. Grants and revocations live in the durable store, not in the build, so rolling back code cannot reactivate a revoked grant; the runbook proves it by repeating the stale-tab case afterwards. The runbook names the owner and exact action for emergency revocation, and a failing provider gets bounded retries and a plain error. Cases that need IIS, the domain or a live provider and ran on fixtures or the PolicyHarness are labeled with an exact rerun plan; the oral review then defends the evidence in ten minutes.

Security habit

Keep the identities apart and decide on the server. The host principal, the Wisej.NET application principal and the effective Windows token are three different facts, and none of them is a permission. Every protected read checks the current grant, the tenant and the record at the service boundary before the repository runs, and a hidden button, a role claim or a signed-out tab proves nothing until that check denies.

Learning objectives

By the end of this module you'll be able to:

- ✓ Design the intranet Windows route and the partner HTTPS route with one shared tenant and resource policy, and diagram where each identity originates, is trusted, expires and is revoked.
- ✓ Make scale and failure explicit: pinned versions, forwarded headers, WebSocket affinity, shared cookie-protection keys and a safe outcome for a callback that lands on another node.
- ✓ Build an incident matrix keyed to the first failing boundary — ingress challenge, policy deny, effective token, callback route, passkey validation — with redacted, correlated audit events.
- ✓ Run the 12-case negative matrix, define release stop conditions and a rollback that keeps revoked grants revoked, and defend the release in an oral review.

Lab / Exam Guide

What you'll build in the hands-on lab, the suggested approach, and the required deliverables.

Hands-on lab

Complete the AccessOps capstone on the cumulative portal: draw the trust boundaries with owners for IIS, the directory, the identity provider, the app and the credential store, write the route matrix (framework target, IIS mode, challenge, host principal, app principal, execution identity, revocation owner), and run the 12-case evidence matrix from the brief — intranet Reader read, host-authenticated deny, tenant A requesting tenant B's record 17, scoped read with reversion, second hop, session rotation, revoked grant from a stale tab, external return URL and forged post, single grant per federated callback, unapproved tenant or bad state, passkey replay, wrong owner or deleted credential, and a restart or second-node callback — recording for each the boundary that denied and proof that no data access or grant followed. Simulate one failure such as a wrong SPN, node loss during a callback or a lost authenticator, practise the runbook, and present the allow and deny traces and the rollback in a ten-minute review against the security review checklists; cases that need IIS, the domain or a live provider and ran on fixtures or the PolicyHarness are labeled with an exact rerun plan.

Starting point

Open the cumulative AccessOps portal you built through the previous six modules, with its intranet Windows route, partner sign-in, OpenID Connect hand-off, passkey store and the shared record policy, plus the self-contained PolicyHarness. Use two tenants (A and B) that both hold a record 17, the reserved fixtures F01–F12, and either a second node or a recorded restart-and-reroute exercise.

Suggested approach

- ✓ Draw the trust-boundary diagram first: browser, IIS or proxy, Wisej.NET session, identity provider, credential store and record backend, with an owner, a trust anchor, a lifetime and a revocation action on every identity that crosses a line.
- ✓ Fill one route-matrix row per entry point and pin the versions you tested; then write down the sanitized configuration (forwarded headers, affinity, shared Data Protection keys, `"secure": true`) and the role and tenant map.
- ✓ Run the 12 cases in order and fill each row of the evidence matrix with the test alias, entry point, expected and actual outcome, the boundary that denied, the correlation label, the repository call count and whether it ran live, on a fixture or in the PolicyHarness.
- ✓ Inject one failure (a wrong SPN, node loss during a provider callback or a lost authenticator), follow the incident matrix from the first failing boundary and time the recovery in the runbook.
- ✓ Rehearse the rollback: switch the partner route off, confirm the intranet baseline still serves Readers, repeat the stale-tab case to show the revoked grant stays revoked, then present the ten-minute review.

Required deliverables

- ✓ Trust-boundary diagram and route matrix

- ✓ Cumulative AccessOps portal with sanitized configuration, role and tenant map and policy tests
- ✓ 12-case evidence matrix with denying boundary and fixture labels
- ✓ Deployment and rollback runbook with owners and stop conditions
- ✓ Completed security review checklist and oral review notes scored against the rubric

Student review questions

- ✓ A partner's callback reaches node B after node A died mid sign-in. Walk through what node B can verify, what it cannot, and why the outcome is still one grant or none.
- ✓ Your rollback restores the previous build and configuration. Which state must it not restore, and which evidence row proves that a revoked grant did not come back?
- ✓ Case 5 ran against a fixture because the lab has no domain. What exactly would you rerun on the real topology, and what result there would block sign-off?

Security habit

Keep the identities apart and decide on the server. The host principal, the Wisej.NET application principal and the effective Windows token are three different facts, and none of them is a permission. Every protected read checks the current grant, the tenant and the record at the service boundary before the repository runs, and a hidden button, a role claim or a signed-out tab proves nothing until that check denies.