SECURITY & AUTH PATTERNS · MODULE 2

IIS and Browser Integrated Authentication

Lesson Guide · Lab / Exam Guide

LearnWisej

Lesson Guide

Choosing the IIS site boundary, Windows Authentication providers and challenge order, browser negotiation and intranet policy, SPNs, Kerberos evidence, extended protection and proxies, and diagnosing a failure by layer. 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

Choosing the IIS site boundary, Windows Authentication providers and challenge order, browser negotiation and intranet policy, SPNs, Kerberos evidence, extended protection and proxies, and diagnosing a failure by layer.

1. The production problem

Staff open the AccessOps intranet route and expect to be signed in silently with their Windows account, while partners reach a public sign-in page that must never be challenged for Windows credentials. When silent sign-in fails, the symptom is a prompt, a 401 loop or an error page, and the cause sits in IIS, the browser, the directory or a proxy before any Wisej.NET code runs. This module teaches you to configure that host boundary deliberately and to tell a host failure from an application deny.

2. Key concepts

Concept	Why it matters
site boundary	A separate IIS application or binding per entry point, so the Windows-only intranet route and the Anonymous public route each carry exactly one visible authentication policy.
Negotiate challenge	An HTTP 401 carrying `WWW-Authenticate: Negotiate` and/or `NTLM`; an eligible browser retries with an `Authorization` header, and Negotiate selects Kerberos or falls back to NTLM.
Kerberos evidence	A service ticket for the canonical `HTTP/` host name on the client, correlated with the server's logon record; the response header alone never proves which provider ran.
kernel-mode authentication	With `useKernelMode` the HTTP.sys driver performs the Windows handshake, so the account that decrypts tickets is normally the machine account unless `useAppPoolCredentials` says otherwise.
SPN	The service principal name (`HTTP/accessops.contoso.test`) that tells the domain which account a ticket is encrypted for; a missing or duplicate SPN silently pushes clients to NTLM or into a loop.
extended protection	Channel-binding and service-binding checks that tie the Windows handshake to the TLS connection and the expected SPN, which matters once TLS ends at a proxy.
host principal in Wisej.NET	On ASP.NET Core the Windows user reaches the app only as `Application.User`, the request's `HttpContext.User` captured by Wisej.NET; it proves who signed in, not what they may read.

3. Lesson narrative

One application per entry point

AccessOps has two front doors. The intranet route is for domain staff: in IIS you install the Windows Authentication role service, enable it on that application and disable Anonymous there, so every request is challenged before the ASP.NET Core Module hands it to the app. The public route serves the partner sign-in page and, in Module 5, the OpenID Connect callback. The partner's browser must reach those pages before the app knows who the partner is, so that application keeps Anonymous transport enabled and the Wisej.NET app enforces access itself. Sharing one application would either challenge the external callback or open the intranet screens.

Keep the routes in separate IIS applications or separate bindings with their own application pools. In the lab the intranet route is an isolated HTTPS application with a dedicated pool, whose binding, target framework and authentication state you record first. Inventory the other IIS modes rather than adding them: Basic sends a readily decodable credential and needs HTTPS everywhere, Digest is a legacy role service, and client certificate mapping still needs trust, revocation and account-mapping policy.

Reading the challenge and proving the provider

The first request receives a 401 with `WWW-Authenticate: Negotiate` and, if configured, `NTLM`. A browser that considers the host eligible retries with an `Authorization` header, IIS validates the exchange, and the follow-up returns 200. The lab fixture F01 records exactly this: a Negotiate challenge followed by 200 is a host challenge followed by authentication, and nothing more. Negotiate is a wrapper that can carry Kerberos or fall back to NTLM depending on the client, the SPN and the provider configuration, so a Negotiate header alone does not prove Kerberos.

To claim Kerberos you correlate two pieces of evidence: `klist` on the disposable client shows a service ticket for `HTTP/accessops.contoso.test`, and the server's logon record for the same moment names Kerberos as the authentication package. Without both, write the provider down as undetermined; a workgroup lab shows challenges but can never support a Kerberos claim. Record headers by scheme and status only, because `Authorization` values are credentials and must not land in a trace or a ticket.

Providers, kernel mode, SPN and the browser

The scoped configuration for the lab sets `<anonymousAuthentication enabled="false" />` and `<windowsAuthentication enabled="true" useKernelMode="true">` with the provider list cleared and rebuilt as Negotiate, then NTLM. Keep Negotiate first when you test Kerberos, and test the fallback you intend rather than inferring it from the order. The connection-persistence attributes `authPersistNonNTLM` and `authPersistSingleRequest` decide how long an authenticated connection is trusted. Per the pack and the Microsoft IIS documentation, kernel mode normally decrypts tickets with the machine account; when the HTTP SPN is registered to a custom pool identity, `useAppPoolCredentials` must match that topology. Query the SPN with `setspn -Q` and change SPNs only with the directory team.

Extended protection adds `tokenChecking`, `flags` and an SPN list under `extendedProtection`; decide first whether TLS ends at IIS or at a trusted intermediary, then test across that exact path. The browser has its own vote: Edge sends silent credentials only to hosts it treats as intranet or that administrators list in `AuthServerAllowlist`, so a domain-joined machine does not guarantee silent sign-in for every FQDN, profile or VPN state. NTLM and some Negotiate arrangements also need connection affinity, and with Kestrel's Negotiate handler (the `Microsoft.AspNetCore.Authentication.Negotiate` package, not part of the shared framework) Microsoft requires one-to-one persistent connection affinity behind a proxy. A direct test against the server is your baseline; verify the proxy path separately.

Diagnosing by layer

Work outward in. A 401 before `Program.Main` belongs to the host challenge: compare what the browser saw with the IIS log fields `sc-status`, `sc-substatus` and `sc-win32-status`, then check the SPN, browser policy and proxy. A 500.19 right after you put authentication settings in `web.config` is an IIS configuration error, usually a section locked at a parent scope: read the reported file and section, apply the same values in the server-level location or through IIS Manager, and leave the code alone. A page that loads with a valid Windows identity but refuses a document is an application deny, which is the outcome you want for an account without the Reader grant. A page that works while a file share or SQL call fails is the second hop, the subject of Module 3.

Once IIS has authenticated the request, Wisej.NET captures `HttpContext.User`, and `Application.User` returns it unless the session has assigned its own principal. `Application.UserIdentity` exists only in the

.NET Framework build. The AccessOps intranet start-up therefore treats a missing Windows identity as a host fault and looks the grant up separately:

```
static void Main()
{
    // IIS answered the challenge; Wisej captured HttpContext.User.
    if (!Application.IsAuthenticated ||
        Application.User.Identity is not WindowsIdentity windows)
    {
        new HostFaultPage().Show();   // host layer, not an app deny
        return;
    }

    // The Windows account is evidence; the grant is the permission.
    var grant = AccessGrants.FindForAccount(windows.User?.Value);
    new DocumentsPage(grant).Show(); // DocumentService denies a null grant
}
```

The lab proves this split twice: on IIS in a Windows domain, a second account without Reader is denied after successful host authentication; on any developer machine, the PolicyHarness denies an authenticated identity whose grant cannot read, with zero repository reads. Save the original IIS settings so rollback is one step.

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:

- ✓ Choose the IIS authentication mode for each AccessOps entry point — Windows-only intranet, Anonymous transport for the public sign-in and callback — and keep the two in separate applications or bindings.
- ✓ Read a 401 Negotiate or NTLM challenge and prove whether Kerberos was used from correlated ticket and server logon evidence rather than the response header.
- ✓ Configure Windows Authentication providers, kernel mode, the HTTP SPN and extended protection, and explain how browser intranet policy and proxy connection affinity affect silent sign-in.
- ✓ Diagnose a failure by layer: host challenge, IIS configuration lock (500.19), application deny and second-hop access.

Lab / Exam Guide

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

Hands-on lab

Publish the AccessOps intranet route to an isolated HTTPS IIS application with a dedicated application pool, record the binding, target framework and authentication state, then apply the scoped windows-auth location settings to enable Windows Authentication and disable Anonymous. From a domain client, open the canonical FQDN as CONTOSO\reader1, record the challenge scheme, the follow-up outcome and the redacted application identity, query the HTTP SPN and compare the client ticket with server logon evidence, then sign in as a second account without Reader and capture an application deny after successful host authentication before restoring the original settings. The challenge, the negotiated provider and the SPN evidence can only be verified in the lab environment with IIS and a Windows domain (a workgroup shows challenges but cannot claim Kerberos, so record an undetermined result there); on a plain developer machine the PolicyHarness proves locally that an authenticated identity without the Reader grant is still denied with zero repository reads.

Starting point

Open the AccessOps solution from Module 1, with its `DocumentService`, read-counting repository and tenant A and B fixtures, and the self-contained PolicyHarness. You need an isolated HTTPS IIS application, two domain test accounts, Edge and permission to read the IIS logs; without a domain, run the IIS steps in a workgroup and the policy steps in the PolicyHarness.

Suggested approach

- ✓ Publish the intranet route to its own IIS application and pool, and save the current authentication settings, binding and target framework as the rollback baseline before you change anything.
- ✓ Apply the scoped location settings (Anonymous off, Windows on, kernel mode, providers cleared and rebuilt as Negotiate then NTLM); if IIS answers 500.19, set the same values at the server level or in IIS Manager instead of unlocking sections.
- ✓ Open `https://accessops.contoso.test` from a domain client as `CONTOSO\reader1` and record the 401 scheme, the follow-up status and the redacted identity the diagnostics view shows, never the `Authorization` value.
- ✓ Run `setspn -Q HTTP/accessops.contoso.test` and `klist`, match the client ticket with the server logon record, and write Kerberos, NTLM or undetermined from that evidence alone.
- ✓ Sign in as the second account without Reader, capture the `DocumentService` deny with zero repository reads, add the matching PolicyHarness case, then restore the saved settings and confirm the site is healthy.

Required deliverables

- ✓ Effective IIS authentication settings for the intranet and public routes
- ✓ Redacted challenge trace with scheme, status and follow-up outcome
- ✓ SPN and ticket evidence or an explicit undetermined-provider result

- ✓ Host-authenticated user denied by the AccessOps policy, with a PolicyHarness case for the same rule
- ✓ Layered diagnosis note separating IIS failure from application deny, with the rollback settings

Student review questions

- ✓ The browser shows a Negotiate challenge followed by 200 and the diagnostics view shows the right account name. What can you claim about Kerberos, and what evidence is still missing?
- ✓ A partner reports a Windows credential prompt on the public sign-in page. Which IIS setting or boundary would you inspect first, and why is the fix not in the Wisej.NET code?
- ✓ Silent sign-in works when you browse the server directly but loops through the load balancer. Which two properties of the proxy path would you compare, and what would each one break?

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.