PATRONES DE SEGURIDAD Y AUTENTICACIÓN · MÓDULO 7

Diseño para producción y proyecto final

Guía de la lección · Guía del laboratorio / examen

LearnWisej

Guía de la lección

Dos puntos de entrada con una sola política de recursos, el comportamiento ante la escala y los fallos entre nodos, los tiempos de espera y quién es responsable de la revocación, la evidencia correlacionada de incidentes, y un release y una reversión ensayados para el proyecto final de AccessOps. Qué significa esto para un desarrollador que protege los caminos de inicio de sesión, identidad y autorización de una aplicación Wisej.NET en producción — y cómo leer este módulo.

Enfoque de producción

Dos puntos de entrada con una sola política de recursos, el comportamiento ante la escala y los fallos entre nodos, los tiempos de espera y quién es responsable de la revocación, la evidencia correlacionada de incidentes, y un release y una reversión ensayados para el proyecto final de AccessOps.

1. El problema en producción

Cada módulo anterior protegió un camino. En producción, AccessOps atiende a la vez al personal por una ruta Windows de intranet y a los socios por una ruta HTTPS pública, en varios nodos, junto a un directorio, un proveedor de identidad y un almacén de credenciales, cada uno de los cuales falla a su propio ritmo. Un release solo es seguro cuando ambas rutas llegan a la misma decisión del lado del servidor, la pérdida de un nodo o una pestaña obsoleta no pueden convertirse en acceso, y el equipo puede demostrarlo en una revisión.

2. Conceptos clave

Concepto	Por qué importa
Política de recursos compartida	Una única comprobación en el límite del servicio de concesión, tenant y registro a la que llaman ambas rutas, de modo que a un socio y a un empleado los deniega el mismo código antes de que se ejecute ningún repositorio.
Matriz de rutas	Una tabla por punto de entrada con la plataforma de destino, el modo de IIS, el desafío, el principal del host, el principal de la aplicación, la identidad de ejecución y el responsable de la revocación; un revisor la comprueba fila a fila.
Afinidad de sesión	Wisej.NET mantiene cada sesión en la memoria de un nodo, así que todas las peticiones de una sesión deben llegar al mismo nodo; perder ese nodo es perder la sesión.
Autoridad compartida y duradera	Las concesiones, las revocaciones, los identificadores de entrega del inicio de sesión, los desafíos de passkey y las credenciales viven en el almacén de la aplicación, no en el estado de la sesión, para que un reinicio o una conmutación por error no puedan olvidar una decisión.
Claves compartidas de protección de cookies	Las claves de Data Protection de ASP.NET Core y el nombre de la aplicación deben coincidir en cada nodo que valide las cookies de autenticación o de correlación de otro nodo.
Primer límite que falla	El primer componente que emitió una denegación (punto de entrada, política, token del sistema operativo, ruta del callback o validación de la passkey), donde empieza una investigación.
Condición de parada	Un resultado acordado antes del despliegue, como una lectura entre tenants, que bloquea la promoción y desencadena la reversión ensayada.

3. Desarrollo de la lección

Dos rutas, una política

La ruta de la intranet se ejecuta en IIS con el acceso anónimo deshabilitado y la autenticación de Windows habilitada. El navegador negocia sobre el nombre de host canónico y Wisej.NET expone ese principal del host, cuya identidad es un `WindowsIdentity`, como `Application.User`. AccessOps asigna los grupos del directorio a una concesión de su propio almacén y establece un principal de la aplicación con claims de cuenta, tenant y concesión, como en *Límites de identidad en Wisej.NET*. El token efectivo de Windows es un tercer hecho: solo existe dentro de una llamada acotada a `WindowsIdentity.RunImpersonated` para el único recurso compartido de archivos que necesita los derechos propios del usuario (*Suplantación y delegación de Windows*), y en el resto el código se ejecuta como la identidad del grupo.

La ruta de socios es HTTPS pública. Un socio inicia sesión con el inicio de sesión local de *Inicio de sesión propio y seguridad de la sesión* o a través del proveedor, con los controladores de cookies y de OpenID Connect de ASP.NET Core registrados antes de `app.UseWisej()` (*Federación con OpenID Connect*); una cuenta verificada puede registrar después una passkey mediante la extensión WebAuthn (*Passkeys con la extensión WebAuthn*). Ambos caminos terminan igual: en una carga de página,

`Application.RefreshSessionId()` rota el ID de sesión y se asigna el principal de la aplicación. A partir de ahí los servicios no distinguen las rutas, y cada lectura protegida pasa por una única comprobación.

```
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);
}
```

El diagrama de límites de confianza marca dónde se origina cada identidad, qué establece la confianza, cuánto dura y quién la revoca: el directorio es responsable de la pertenencia a grupos, el proveedor de la cuenta externa del socio, el almacén de credenciales de las passkeys y AccessOps de la concesión. Deshabilitar a un socio en el proveedor no termina una concesión ya emitida, así que la consola de administración también revoca la concesión.

Comportamiento ante la escala y los fallos

Fija lo que probaste: `Wisej-4` y `Wisej-4-WebAuthn` 4.1.7, el controlador de OpenID Connect acorde con el runtime y el propio runtime. Detrás del equilibrador de carga, llama a `app.UseForwardedHeaders()` confiando solo en los proxies conocidos, porque el esquema y el host canónicos alimentan la URI de redirección de OIDC y la comprobación de origen de WebAuthn. Wisej.NET exige afinidad de sesión —una cookie de afinidad, un hash de la IP del cliente u otra ruta coherente— y algunos equilibradores necesitan el modo TCP para el WebSocket. Publica con `"secure": true` en Default.json y mantén los archivos `.json` fuera del servidor de archivos estáticos.

Un nodo perdido pierde sus sesiones, así que nada que deba sobrevivir puede vivir en `Application.Session`. Comparte el conjunto de claves de Data Protection y el nombre de la aplicación entre nodos para que una cookie de correlación o de autenticación escrita por el nodo A se valide en el nodo B. Un callback del proveedor que llega al nodo B tiene dos resultados seguros: el controlador lo valida y el identificador de entrega de un solo uso del almacén compartido reanuda allí el inicio de sesión en una sesión nueva, o el identificador falta o ya se consumió y el usuario recibe un fallo sencillo y un inicio de sesión nuevo. Nunca concede dos veces ni empieza un intento nuevo a partir del código entrante. Los tiempos de espera pertenecen a la misma tabla: el `sessionTimeout` de Wisej, la caducidad absoluta de la concesión, el `ExpireTimeSpan` de la cookie del host, la transacción del proveedor y el tiempo de espera de la ceremonia WebAuthn, cada uno con un responsable de la revocación.

Una matriz de incidentes ordenada por el primer límite que falla

Empieza por el primer componente que dijo que no. Un 401 en el punto de entrada apunta al desafío, al trío de estados de IIS, a `setspn -Q` y a `klist`. Una petición autenticada seguida de una denegación de negocio apunta a una traza de la política: concesión, tenant, registro y cero llamadas al repositorio. Un

fallo de un archivo remoto tras un éxito local apunta al token efectivo. Un callback que nunca llega a la aplicación apunta a la URL canónica, a la URI de redirección exacta y a los encabezados reenviados; uno falsificado termina en `OnRemoteFailure`. Una respuesta de passkey rechazada después del aviso apunta al ID de RP, al origen, al desafío, a `ClientData.Type` y al almacén.

Cada fila emite una categoría de evento de una lista de permitidos con un ID de correlación y un ID de cuenta opaco; como el ámbito de registro de ASP.NET Core es por conexión WebSocket, AccessOps abre su propio `ILogger.BeginScope` en cada clic. Los IDs de sesión, los códigos, los tokens y las cookies nunca llegan al almacén de auditoría protegido, que tiene un periodo de retención declarado.

Release, condiciones de parada y reversión

Antes de exponer la ruta de socios, ejecuta la matriz negativa de 12 casos más un caso normal por límite, y despliega después en un tenant de prueba. Escribe primero las condiciones de parada:

- una lectura entre tenants, o cualquier denegación que ocurra después de que se ejecutara el repositorio;
- un callback o una aserción reutilizados que se aceptan como un inicio de sesión nuevo;
- una credencial eliminada o una concesión revocada que todavía abre una acción protegida;
- material de credenciales en un registro, en una columna oculta de la cuadrícula o en `Application.Browser.UserData`.

La reversión desactiva la ruta de socios o la nueva funcionalidad en el punto de entrada mientras la referencia probada de la intranet sigue funcionando. Las concesiones y las revocaciones viven en el almacén duradero, no en la compilación, así que revertir el código no puede reactivar una concesión revocada; el runbook lo demuestra repitiendo después el caso de la pestaña obsoleta. El runbook nombra al responsable y la acción exacta para la revocación de emergencia, y un proveedor que falla recibe reintentos limitados y un error sencillo. Los casos que necesitan IIS, el dominio o un proveedor real y que se ejecutaron con fixtures o con el PolicyHarness se etiquetan con un plan exacto para volver a ejecutarlos; la revisión oral defiende después la evidencia en diez minutos.

Regla de seguridad

Mantén separadas las identidades y decide en el servidor. El principal del host, el principal de la aplicación Wisej.NET y el token efectivo de Windows son tres hechos distintos, y ninguno de ellos es un permiso. Cada lectura protegida comprueba la concesión vigente, el tenant y el registro en el límite del servicio antes de que se ejecute el repositorio, y un botón oculto, un claim de rol o una pestaña con la sesión cerrada no demuestran nada hasta que esa comprobación deniega.

Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Diseñar la ruta Windows de la intranet y la ruta HTTPS de socios con una política de tenant y de recursos compartida, y representar en un diagrama dónde se origina cada identidad, dónde se confía en ella, cuándo caduca y dónde se revoca.

- ✓ Hacer explícitos la escala y los fallos: versiones fijadas, encabezados reenviados, afinidad del WebSocket, claves compartidas de protección de cookies y un resultado seguro para un callback que llega a otro nodo.
- ✓ Construir una matriz de incidentes ordenada por el primer límite que falla —desafío en el punto de entrada, denegación de la política, token efectivo, ruta del callback, validación de la passkey — con eventos de auditoría ocultados y correlacionados.
- ✓ Ejecutar la matriz negativa de 12 casos, definir las condiciones de parada del release y una reversión que mantenga revocadas las concesiones revocadas, y defender el release en una revisión oral.

Guía del laboratorio / examen

Lo que construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.

Laboratorio práctico

Completa el proyecto final de AccessOps sobre el portal acumulativo: dibuja los límites de confianza con sus responsables para IIS, el directorio, el proveedor de identidad, la aplicación y el almacén de credenciales, escribe la matriz de rutas (plataforma de destino, modo de IIS, desafío, principal del host, principal de la aplicación, identidad de ejecución, responsable de la revocación) y ejecuta la matriz de evidencias de 12 casos del enunciado —lectura de un Reader en la intranet, denegación tras la autenticación del host, el tenant A solicitando el registro 17 del tenant B, lectura acotada con reversión, segundo salto, rotación de la sesión, concesión revocada desde una pestaña obsoleta, URL de retorno externa y POST falsificado, una sola concesión por callback federado, tenant no aprobado o state erróneo, reutilización de una passkey, propietario erróneo o credencial eliminada, y un callback tras un reinicio o en un segundo nodo—, anotando en cada uno el límite que denegó y la prueba de que no siguió ningún acceso a datos ni ninguna concesión. Simula un fallo, como un SPN erróneo, la pérdida de un nodo durante un callback o un autenticador perdido, practica el runbook y presenta las trazas de permiso y de denegación y la reversión en una revisión de diez minutos frente a las listas de comprobación de la revisión de seguridad; los casos que necesitan IIS, el dominio o un proveedor real y que se ejecutaron con fixtures o con el PolicyHarness se etiquetan con un plan exacto para volver a ejecutarlos.

Punto de partida

Abre el portal acumulativo de AccessOps que construiste a lo largo de los seis módulos anteriores, con su ruta Windows de intranet, el inicio de sesión de socios, la entrega de OpenID Connect, el almacén de passkeys y la política de registros compartida, más el PolicyHarness autónomo. Usa dos tenants (A y B) que tengan ambos un registro 17, los fixtures reservados F01–F12 y, o bien un segundo nodo, o bien un ejercicio grabado de reinicio y redirección.

Enfoque sugerido

- ✓ Dibuja primero el diagrama de límites de confianza: navegador, IIS o proxy, sesión Wisej.NET, proveedor de identidad, almacén de credenciales y backend de registros, con un responsable, un ancla de confianza, una vida útil y una acción de revocación en cada identidad que cruza una línea.
- ✓ Rellena una fila de la matriz de rutas por punto de entrada y fija las versiones que probaste; escribe después la configuración saneada (encabezados reenviados, afinidad, claves compartidas de Data Protection, `"secure": true`) y el mapa de roles y tenants.
- ✓ Ejecuta los 12 casos en orden y rellena cada fila de la matriz de evidencias con el alias de la prueba, el punto de entrada, el resultado esperado y el real, el límite que denegó, la etiqueta de correlación, el número de llamadas al repositorio y si se ejecutó en real, con un fixture o en el PolicyHarness.
- ✓ Inyecta un fallo (un SPN erróneo, la pérdida de un nodo durante un callback del proveedor o un autenticador perdido), sigue la matriz de incidentes desde el primer límite que falla y cronometra la recuperación en el runbook.

- ✓ Ensaya la reversión: desactiva la ruta de socios, confirma que la referencia de la intranet sigue sirviendo a los Readers, repite el caso de la pestaña obsoleta para mostrar que la concesión revocada sigue revocada y presenta después la revisión de diez minutos.

Entregables requeridos

- ✓ Diagrama de límites de confianza y matriz de rutas
- ✓ Portal acumulativo de AccessOps con configuración saneada, mapa de roles y tenants y pruebas de la política
- ✓ Matriz de evidencias de 12 casos con el límite que deniega y las etiquetas de fixture
- ✓ Runbook de despliegue y reversión con responsables y condiciones de parada
- ✓ Lista de comprobación de la revisión de seguridad completada y notas de la revisión oral puntuadas según la rúbrica

Preguntas de repaso

- ✓ El callback de un socio llega al nodo B después de que el nodo A cayera a mitad del inicio de sesión. Explica paso a paso qué puede verificar el nodo B, qué no puede y por qué el resultado sigue siendo una concesión o ninguna.
- ✓ Tu reversión restaura la compilación y la configuración anteriores. ¿Qué estado no debe restaurar, y qué fila de evidencias demuestra que una concesión revocada no volvió?
- ✓ El caso 5 se ejecutó contra un fixture porque el laboratorio no tiene dominio. ¿Qué volverías a ejecutar exactamente en la topología real, y qué resultado allí bloquearía la aprobación?

Regla de seguridad

Mantén separadas las identidades y decide en el servidor. El principal del host, el principal de la aplicación Wisej.NET y el token efectivo de Windows son tres hechos distintos, y ninguno de ellos es un permiso. Cada lectura protegida comprueba la concesión vigente, el tenant y el registro en el límite del servicio antes de que se ejecute el repositorio, y un botón oculto, un claim de rol o una pestaña con la sesión cerrada no demuestran nada hasta que esa comprobación deniega.