MODÈLES DE SÉCURITÉ ET D'AUTHENTIFICATION · MODULE 7

Conception pour la production et projet de synthèse

Guide de la leçon · Guide du lab / de l'examen

LearnWisej

Guide de la leçon

Deux points d'entrée avec une seule politique de ressources, le comportement à l'échelle et en cas de défaillance sur plusieurs nœuds, les délais et la responsabilité de la révocation, des preuves d'incident corrélées, et une release avec retour arrière répétée pour le projet de synthèse AccessOps. Ce que cela signifie pour un développeur qui sécurise les chemins de connexion, d'identité et d'autorisation d'une application Wisej.NET en production — et comment aborder ce module.

Objectif en production

Deux points d'entrée avec une seule politique de ressources, le comportement à l'échelle et en cas de défaillance sur plusieurs nœuds, les délais et la responsabilité de la révocation, des preuves d'incident corrélées, et une release avec retour arrière répétée pour le projet de synthèse AccessOps.

1. Le problème en production

Chaque module précédent a sécurisé un chemin. En production, AccessOps sert à la fois le personnel par une route Windows intranet et les partenaires par une route HTTPS publique, sur plusieurs nœuds, à côté d'un annuaire, d'un fournisseur d'identité et d'un magasin d'identifiants qui tombent chacun en panne à leur propre rythme. Une release n'est sûre que si les deux routes aboutissent à la même décision côté serveur, si la perte d'un nœud ou un onglet périmé ne peut pas devenir un accès, et si l'équipe peut le prouver lors d'une revue.

2. Concepts clés

Concept	Pourquoi c'est important
Politique de ressources partagée	Une seule vérification de l'octroi, du tenant et de l'enregistrement à la couche service, que les deux routes appellent, si bien qu'un partenaire et un membre du personnel sont refusés par le même code avant qu'aucun repository ne s'exécute.
Matrice des routes	Un tableau, par point d'entrée, du framework cible, du mode IIS, du challenge, du principal de l'hôte, du principal applicatif, de l'identité d'exécution et du responsable de la révocation ; un relecteur le vérifie ligne par ligne.
Affinité de session	Wisej.NET garde chaque session dans la mémoire d'un seul nœud ; chaque requête d'une session doit donc atteindre le même nœud, et perdre ce nœud, c'est perdre la session.
Autorité partagée et durable	Les octrois, les révocations, les handles de transfert de connexion, les challenges de clés d'accès et les identifiants vivent dans le magasin de l'application, pas dans l'état de session, si bien qu'un redémarrage ou une bascule ne peut pas oublier une décision.
Clés partagées de protection des cookies	Les clés de Data Protection d'ASP.NET Core et le nom d'application doivent être identiques sur chaque nœud qui valide les cookies d'authentification ou de corrélation d'un autre nœud.
Première frontière en échec	Le premier composant qui a émis un refus (ingress, politique, jeton du système d'exploitation, route de callback ou validation de la clé d'accès), là où commence une investigation.
Condition d'arrêt	Un résultat convenu avant le déploiement, comme une lecture entre tenants, qui bloque la promotion et déclenche le retour arrière répété.

3. Le fil de la leçon

Deux routes, une politique

La route intranet s'exécute sur IIS avec l'accès anonyme désactivé et l'authentification Windows activée. Le navigateur négocie sur le nom d'hôte canonique et Wisej.NET expose ce principal de l'hôte, dont l'identité est une `WindowsIdentity`, sous la forme d'`Application.User`. AccessOps associe les groupes de l'annuaire à un octroi dans son propre magasin et établit un principal applicatif portant les claims de compte, de tenant et d'octroi, comme dans *Frontières d'identité dans Wisej.NET*. Le jeton Windows effectif est un troisième fait : il n'existe qu'à l'intérieur d'un appel délimité à `WindowsIdentity.RunImpersonated`, pour l'unique partage de fichiers qui exige les propres droits de l'utilisateur (*Emprunt d'identité Windows et délégation*), et partout ailleurs le code s'exécute sous l'identité du pool.

La route partenaires est en HTTPS public. Un partenaire se connecte avec la connexion locale de *Connexion personnalisée et sécurité de session* ou via le fournisseur, avec les handlers de cookies et OpenID Connect d'ASP.NET Core enregistrés avant `app.UseWisej()` (*Fédération avec OpenID Connect*) ; un compte vérifié peut ensuite enrôler une clé d'accès via l'extension WebAuthn (*Clés d'accès avec l'extension WebAuthn*). Les deux chemins se terminent de la même manière : lors d'un chargement de page, `Application.RefreshSessionId()` fait tourner l'ID de session et le principal applicatif est défini. À partir

de là, les services ne peuvent plus distinguer les routes, et chaque lecture protégée passe par une seule vérification.

```
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);
}
```

Le diagramme des frontières de confiance indique où naît chaque identité, ce qui établit la confiance, combien de temps elle vit et qui la révoque : l'annuaire est responsable de l'appartenance aux groupes, le fournisseur du compte externe du partenaire, le magasin d'identifiants des clés d'accès, et AccessOps de l'octroi. Désactiver un partenaire chez le fournisseur ne met pas fin à un octroi déjà émis ; la console d'administration révoque donc aussi l'octroi.

Comportement à l'échelle et en cas de défaillance

Épinglez ce que vous avez testé : `Wisej-4` et `Wisej-4-WebAuthn` 4.1.7, le handler OpenID Connect correspondant au runtime, et le runtime lui-même. Derrière l'équilibreur de charge, appelez `app.UseForwardedHeaders()` en ne faisant confiance qu'aux proxys connus, car le schéma et l'hôte canoniques alimentent l'URI de redirection OIDC et la vérification d'origine WebAuthn. Wisej.NET exige une affinité de session — un cookie d'affinité, un hachage de l'IP cliente ou une autre route cohérente — et certains équilibreurs ont besoin du mode TCP pour le WebSocket. Publiez avec `"secure": true` dans Default.json et tenez les fichiers `.json` hors du serveur de fichiers statiques.

Un nœud perdu perd ses sessions ; rien de ce qui doit survivre ne peut donc vivre dans `Application.Session`. Partagez le trousseau de clés de Data Protection et le nom d'application entre les nœuds, pour qu'un cookie de corrélation ou d'authentification écrit par le nœud A soit validé sur le nœud B. Un callback du fournisseur qui arrive sur le nœud B a deux issues sûres : soit le handler le valide et le handle de transfert à usage unique du magasin partagé reprend la connexion dans une nouvelle session sur ce nœud, soit le handle est absent ou déjà consommé et l'utilisateur reçoit un simple échec et une nouvelle connexion. Il n'accorde jamais deux octrois et ne démarre jamais une nouvelle tentative à partir du code entrant. Les délais ont leur place dans le même tableau : le `sessionTimeout` de Wisej, l'expiration absolue de l'octroi, l'`ExpireTimeSpan` du cookie de l'hôte, la transaction du fournisseur et le délai de la cérémonie WebAuthn, chacun avec un responsable de la révocation.

Une matrice d'incidents indexée sur la première frontière en échec

Partez du premier composant qui a dit non. Un 401 à l'ingress renvoie au challenge, au triplet de statut IIS, à `setspn -Q` et à `klist`. Une requête authentifiée suivie d'un refus métier renvoie à une trace de politique : octroi, tenant, enregistrement et un nombre d'appels au repository égal à zéro. Un échec sur un

fichier distant après une réussite locale renvoie au jeton effectif. Un callback qui n'atteint jamais l'application renvoie à l'URL canonique, à l'URI de redirection exacte et aux en-têtes transférés ; un callback forgé se termine dans `OnRemoteFailure`. Une réponse de clé d'accès rejetée après l'invite renvoie au RP ID, à l'origine, au challenge, à `ClientData.Type` et au magasin.

Chaque ligne émet une catégorie d'événement issue d'une liste d'autorisation, avec un ID de corrélation et un ID de compte opaque ; comme la portée de journalisation d'ASP.NET Core est propre à chaque connexion WebSocket, AccessOps ouvre son propre `ILogger.BeginScope` à chaque clic. Les ID de session, codes, jetons et cookies n'atteignent jamais le magasin d'audit protégé, qui a une durée de conservation définie.

Release, conditions d'arrêt et retour arrière

Avant d'exposer la route partenaires, exécutez la matrice négative de 12 cas plus un cas normal par frontière, puis déployez sur un tenant de test. Écrivez d'abord les conditions d'arrêt :

- une lecture entre tenants, ou tout refus qui intervient après l'exécution du repository ;
- un callback ou une assertion rejoués acceptés comme une nouvelle connexion ;
- un identifiant supprimé ou un octroi révoqué qui ouvre encore une action protégée ;
- des éléments d'identification dans un journal, une colonne de grille masquée ou `Application.Browser.UserData`.

Le retour arrière désactive la route partenaires ou la nouvelle fonctionnalité à l'ingress, pendant que la référence intranet testée continue de fonctionner. Les octrois et les révocations vivent dans le magasin durable, pas dans le build ; revenir à l'ancien code ne peut donc pas réactiver un octroi révoqué, et le runbook le prouve en répétant ensuite le cas de l'onglet périmé. Le runbook nomme le responsable et l'action exacte pour une révocation d'urgence, et un fournisseur défaillant reçoit des nouvelles tentatives bornées et une simple erreur. Les cas qui exigent IIS, le domaine ou un fournisseur réel et qui ont été exécutés sur des fixtures ou dans le PolicyHarness sont étiquetés avec un plan précis de réexécution ; la revue orale défend ensuite les preuves en dix minutes.

Réflexe de sécurité

Gardez les identités séparées et décidez sur le serveur. Le principal de l'hôte, le principal applicatif Wisej.NET et le jeton Windows effectif sont trois faits différents, et aucun n'est une permission. Chaque lecture protégée vérifie l'octroi actuel, le tenant et l'enregistrement à la couche service avant que le repository ne s'exécute, et un bouton masqué, un claim de rôle ou un onglet déconnecté ne prouvent rien tant que cette vérification n'a pas refusé.

Objectifs d'apprentissage

À la fin de ce module, vous saurez :

- ✓ Concevoir la route Windows intranet et la route HTTPS partenaires avec une seule politique partagée de tenant et de ressources, et représenter où chaque identité naît, reçoit la confiance, expire et est révoquée.

- ✓ Rendre explicites l'échelle et les défaillances : versions épinglées, en-têtes transférés, affinité WebSocket, clés partagées de protection des cookies et issue sûre pour un callback qui arrive sur un autre nœud.
- ✓ Construire une matrice d'incidents indexée sur la première frontière en échec — challenge à l'ingress, refus de politique, jeton effectif, route de callback, validation de la clé d'accès — avec des événements d'audit masqués et corrélés.
- ✓ Exécuter la matrice négative de 12 cas, définir des conditions d'arrêt de la release et un retour arrière qui garde révoqués les octrois révoqués, et défendre la release lors d'une revue orale.

Guide du lab / de l'examen

Ce que vous allez construire dans le lab pratique, l'approche suggérée et les livrables attendus.

Lab pratique

Réalisez le projet de synthèse AccessOps sur le portail cumulatif : dessinez les frontières de confiance avec leurs responsables pour IIS, l'annuaire, le fournisseur d'identité, l'application et le magasin d'identifiants, rédigez la matrice des routes (framework cible, mode IIS, challenge, principal de l'hôte, principal applicatif, identité d'exécution, responsable de la révocation), et exécutez la matrice de preuves de 12 cas du cahier des charges — lecture intranet d'un Reader, refus après authentification par l'hôte, tenant A demandant l'enregistrement 17 du tenant B, lecture délimitée avec rétablissement, second saut, rotation de session, octroi révoqué depuis un onglet périmé, URL de retour externe et POST forgé, octroi unique par callback fédéré, tenant non approuvé ou state invalide, rejeu de clé d'accès, mauvais propriétaire ou identifiant supprimé, et callback après redémarrage ou sur un second nœud — en consignant pour chacun la frontière qui a refusé et la preuve qu'aucun accès aux données ni aucun octroi n'a suivi. Simulez une défaillance, comme un mauvais SPN, la perte d'un nœud pendant un callback ou un authentificateur perdu, répétez le runbook, et présentez les traces d'autorisation et de refus ainsi que le retour arrière lors d'une revue de dix minutes, au regard des checklists de revue de sécurité ; les cas qui exigent IIS, le domaine ou un fournisseur réel et qui ont été exécutés sur des fixtures ou dans le PolicyHarness sont étiquetés avec un plan précis de réexécution.

Point de départ

Ouvrez le portail AccessOps cumulatif que vous avez construit au fil des six modules précédents, avec sa route Windows intranet, la connexion des partenaires, le transfert OpenID Connect, le magasin de clés d'accès et la politique d'enregistrements partagée, plus le PolicyHarness autonome. Utilisez deux tenants (A et B) contenant tous deux un enregistrement 17, les fixtures réservées F01–F12, et soit un second nœud, soit un exercice consigné de redémarrage et de réacheminement.

Approche suggérée

- ✓ Dessinez d'abord le diagramme des frontières de confiance : navigateur, IIS ou proxy, session Wisej.NET, fournisseur d'identité, magasin d'identifiants et backend des enregistrements, avec un responsable, une ancre de confiance, une durée de vie et une action de révocation pour chaque identité qui franchit une ligne.
- ✓ Remplissez une ligne de la matrice des routes par point d'entrée et épinglez les versions que vous avez testées ; notez ensuite la configuration épurée (en-têtes transférés, affinité, clés de Data Protection partagées, `"secure": true`) et la carte des rôles et des tenants.
- ✓ Exécutez les 12 cas dans l'ordre et remplissez chaque ligne de la matrice de preuves avec l'alias du test, le point d'entrée, le résultat attendu et le résultat obtenu, la frontière qui a refusé, l'étiquette de corrélation, le nombre d'appels au repository, et si le cas a été exécuté en réel, sur une fixture ou dans le PolicyHarness.
- ✓ Injectez une défaillance (un mauvais SPN, la perte d'un nœud pendant un callback du fournisseur ou un authentificateur perdu), suivez la matrice d'incidents depuis la première frontière en échec et chronométrez la reprise dans le runbook.

- ✓ Répétez le retour arrière : désactivez la route partenaires, confirmez que la référence intranet sert toujours les Readers, répétez le cas de l'onglet périmé pour montrer que l'octroi révoqué le reste, puis présentez la revue de dix minutes.

Livrables attendus

- ✓ Diagramme des frontières de confiance et matrice des routes
- ✓ Portail AccessOps cumulatif avec configuration épurée, carte des rôles et des tenants et tests de politique
- ✓ Matrice de preuves de 12 cas avec la frontière qui refuse et les étiquettes de fixtures
- ✓ Runbook de déploiement et de retour arrière avec responsables et conditions d'arrêt
- ✓ Checklist de revue de sécurité complétée et notes de revue orale notées selon la grille d'évaluation

Questions de révision

- ✓ Le callback d'un partenaire atteint le nœud B après que le nœud A est tombé en pleine connexion. Décrivez ce que le nœud B peut vérifier, ce qu'il ne peut pas vérifier, et pourquoi le résultat reste un octroi ou aucun.
- ✓ Votre retour arrière restaure le build et la configuration précédents. Quel état ne doit-il pas restaurer, et quelle ligne de preuve démontre qu'un octroi révoqué n'est pas revenu ?
- ✓ Le cas 5 a été exécuté sur une fixture, car le lab n'a pas de domaine. Que réexécuteriez-vous exactement sur la topologie réelle, et quel résultat y bloquerait la validation ?

Réflexe de sécurité

Gardez les identités séparées et décidez sur le serveur. Le principal de l'hôte, le principal applicatif Wisej.NET et le jeton Windows effectif sont trois faits différents, et aucun n'est une permission. Chaque lecture protégée vérifie l'octroi actuel, le tenant et l'enregistrement à la couche service avant que le repository ne s'exécute, et un bouton masqué, un claim de rôle ou un onglet déconnecté ne prouvent rien tant que cette vérification n'a pas refusé.