

Architecture de sécurité : identité, SSO, autorisation, audit et déploiement sécurisé

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

● LearnWisej

Guide de la leçon

Intégration de l'identité, modèle de permissions, pistes d'audit, HTML sûr, configuration sécurisée. Ce que cela signifie pour une équipe Wisej.NET avancée — et comment aborder ce module.



Enjeu de production

Intégration de l'identité, modèle de permissions, pistes d'audit, HTML sûr, configuration sécurisée.

1. Le problème en production

L'architecture côté serveur réduit de nombreuses surfaces d'attaque dans le navigateur, mais les systèmes avancés ont toujours besoin d'intégration de l'identité, d'autorisation au niveau des services, de pistes d'audit, d'un rendu sûr et d'une configuration d'exploitation sécurisée.

2. Concepts clés

Concept	Pourquoi c'est important
frontière OIDC/SSO	Le point où un fournisseur d'identité externe remet à l'application une identité vérifiée. Tout ce qui se trouve après cette frontière relève de la responsabilité de l'application.
correspondance des claims	Traduire les claims propres au fournisseur en rôles et permissions de l'application, pour que le code d'autorisation ne dépende jamais du format d'un fournisseur d'identité particulier.
service de permissions	Service unique qui indique si l'utilisateur courant peut effectuer une action, pour que la règle vive en un seul endroit plutôt que dans chaque écran.
autorisation au niveau des services	Appliquer les permissions dans les services applicatifs, et pas seulement dans l'UI, pour qu'un bouton masqué ne soit jamais la seule protection d'une opération.
journal d'audit	Trace infalsifiable de qui a fait quoi et quand pour les actions importantes, afin de pouvoir enquêter sur les événements de sécurité et prouver la conformité.
revue d'AllowHtml	Vérifier chaque endroit où un contrôle autorise du HTML brut pour s'assurer que le contenu fourni par l'utilisateur est assaini, car un HTML non revu constitue un risque de cross-site scripting.
gestion des secrets	Conserver les chaînes de connexion, les clés et les jetons dans une configuration sécurisée ou un coffre-fort, jamais dans le contrôle de code source ni dans des paramètres visibles côté client.
en-têtes de sécurité	En-têtes de réponse HTTP, comme la content security policy et la strict transport security, qui réduisent la surface d'attaque de l'application dans le navigateur.

3. Déroulé de la leçon

L'authentification n'est pas l'autorisation

Un utilisateur connecté n'est pas automatiquement autorisé à approuver un ordre de travail, à consulter un autre tenant, à exporter des données ou à exécuter une tâche d'administration. Le contrôle de permission a sa place dans le service qui exécute l'action.

Le contexte de session relie l'identité et l'UI

Après l'authentification, l'application doit projeter l'identité dans un contexte de session : ID utilisateur, nom affiché, tenant, rôles, permissions, culture, thème et ID de corrélation. Ce contexte aide les écrans à s'adapter, tandis que les services continuent d'appliquer les règles.

Un modèle d'identité et d'autorisation pour l'entreprise

L'autorisation en entreprise comporte plus de dimensions qu'un nom de rôle. Une même personne peut être dispatcher dans un tenant, auditeur en lecture seule dans un autre, et autorisée à approuver des ordres de travail uniquement en dessous d'un plafond de coût. Un modèle capable d'exprimer cela commence par trois couches : l'identité, qui dit qui est l'utilisateur ; l'appartenance, qui dit de quels tenants

et rôles il dispose ; et les permissions, qui sont les actions fines accordées par un rôle, comme

`ApproveWorkOrder` ou `ExportData` .

Le lab traduit cela en une matrice des permissions : les lignes sont les permissions, les colonnes sont les rôles, et chaque cellule est accordée, refusée ou conditionnelle. La matrice est l'artefact que les product owners, les relecteurs sécurité et les développeurs peuvent tous lire ; elle doit donc exister avant le code. Les étapes des workflows s'y rattachent aussi. Soumettre une escalade, l'approuver et la clôturer peuvent chacune exiger une permission différente, et la matrice le rend explicite.

Gardez le tenant au premier plan du modèle. Une permission est toujours évaluée dans un contexte de tenant ; un utilisateur qui détient un rôle dans le tenant A n'obtient donc rien dans le tenant B. L'appel `PermissionService.Demand` du lab vérifie d'abord la garde de tenant, et ne consulte qu'ensuite le magasin des rôles.

Contrôles d'authentification, contexte de session et services

Un fournisseur d'identité externe, atteint via une connexion OpenID Connect ou SSO, indique à l'application qui est l'utilisateur. C'est là que s'arrête le rôle du fournisseur. L'application projette ensuite les claims reçues dans son propre contexte de session : ID utilisateur, nom affiché, tenant, rôles, permissions et ID de corrélation. Comme cette correspondance s'exécute une fois à la connexion et est stockée dans la session, les écrans peuvent la lire à moindre coût et les services peuvent s'y fier sans refaire l'analyse des claims.

Le contrôle lui-même doit se trouver devant chaque point d'entrée, pas seulement devant la page principale. Les jobs en arrière-plan, les appels WebMethod, les questionnaires de postback et les pages de diagnostic ont tous besoin de savoir qui appelle. Un service ne doit jamais lire l'identité depuis un contrôle d'UI, une query string ou un champ masqué, car tous peuvent être modifiés par l'utilisateur. Il lit le contexte de session, et ce contexte a été construit à partir de claims vérifiées.

Concevez le contexte pour qu'il puisse être rafraîchi. Les rôles changent pendant que des sessions sont ouvertes, et une session de longue durée doit pouvoir recharger les permissions sans imposer une nouvelle connexion. Une question de revue pour chaque méthode de service est simple : quelle ligne prouve que l'appelant a le droit d'être ici ?

Pistes d'audit pour les commandes sensibles

Une piste d'audit répond aux questions : qui a fait quoi, sur quel enregistrement, dans quel tenant, quand et avec quel résultat. Elle est écrite par le service qui exécute la commande, au sein de la même opération, de sorte qu'une action ne peut pas réussir sans être consignée. L'écran du journal d'audit du lab affiche chaque `Demand` , accordé ou refusé, et chaque commande sensible comme une approbation, un export ou un changement de permission.

Les refus sont aussi importants que les succès. Un bouton activé par erreur qui permet à un utilisateur de demander `ExportData` est intercepté par le service, qui refuse avec une erreur nommée et écrit une entrée de refus. Des refus répétés depuis un même compte signalent que quelqu'un sonde l'application, et ils ne sont visibles que si les refus sont journalisés. L'entrée doit porter l'ID de corrélation pour être reliée aux logs structurés du Module 11.

- Auteur, tenant et, le cas échéant, indicateur d'usurpation d'identité.

- Nom de la commande, enregistrement cible, et les valeurs avant et après qui comptent.
- Issue : accordée, refusée ou en échec, avec la raison nommée.
- Horodatage et ID de corrélation, écrits dans la même transaction que la modification.

Une checklist de sécurité pour toute la surface

Le rendu côté serveur élimine de nombreux risques côté navigateur, mais une application Wisej.NET envoie toujours du HTML, accepte des uploads, définit des cookies, écrit des logs, lit des secrets et se trouve derrière un proxy. Chacun de ces points est un endroit où une petite erreur devient un incident. La checklist de durcissement du lab est le document qui transforme ces points en éléments de revue vérifiés à chaque release plutôt que rappelés de temps en temps.

Commencez par le HTML. Tout contrôle qui affiche du HTML brut issu d'une saisie utilisateur doit l'assainir, et la revue d' `AllowHtml` du lab liste chacun de ces contrôles. Les uploads nécessitent des limites de taille, des contrôles de type et un stockage en dehors de la racine web. Les cookies doivent être marqués secure et HTTP-only, avec une politique same-site. Une content security policy limite les scripts autorisés à s'exécuter. Les logs ne doivent jamais contenir de mots de passe, de jetons ni de fiches personnelles complètes. Les secrets vivent dans un coffre-fort ou une configuration protégée, jamais dans le contrôle de code source. Un reverse proxy doit transmettre le schéma et l'adresse client d'origine pour que l'application voie les valeurs réelles.

Traitez la checklist comme du code vivant : ajoutez un élément chaque fois qu'une revue découvre une lacune, et consignez qui a vérifié chaque élément pour chaque release.



Réflexe de production avancé

Ne vous arrêtez pas quand l'écran paraît correct. Expliquez à qui appartient l'état, les implications de sécurité, les chemins d'échec et le comportement en production avant de considérer le module comme terminé.



Objectifs d'apprentissage

À la fin de ce module, vous saurez :

- ✓ Concevoir un modèle d'identité et d'autorisation pour les rôles, les tenants et les workflows d'entreprise.
- ✓ Intégrer les contrôles d'authentification au contexte de session et aux services applicatifs.
- ✓ Construire des pistes d'audit pour les commandes sensibles.
- ✓ Créer des checklists de sécurité pour le HTML, les uploads, les cookies, la CSP, les logs, les secrets et les reverse proxies.

Guide du lab / de l'examen

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



Lab pratique

Ajoutez une simulation d'authentification d'entreprise, la correspondance des claims vers les permissions, l'application des rôles par tenant, l'autorisation au niveau des services, la revue du HTML sûr, l'approbation des exports et la journalisation d'audit des actions sensibles.

Approche suggérée

- ✓ Lisez le problème en production et identifiez le risque d'architecture avant de toucher à l'UI.
- ✓ Créez ou mettez à jour d'abord l'encapsulation par un service, de commande, de modèle ou de composant concernée.
- ✓ Construisez l'écran ou le composant Wisej.NET en suivant les standards de projet établis.
- ✓ Ajoutez une démonstration du chemin d'échec, pas seulement du chemin nominal.
- ✓ Consignez ce qui a changé dans le journal d'apprentissage et mettez à jour la checklist concernée.

Livrables attendus

- ✓ Conception de la correspondance d'identité
- ✓ Matrice des permissions
- ✓ Implémentation du service de permissions
- ✓ Écran du journal d'audit
- ✓ Checklist de durcissement de la sécurité

Questions de révision

- ✓ Où l'autorisation est-elle appliquée ?
- ✓ Quels éléments de l'UI affichent du HTML ?
- ✓ Un bouton activé peut-il contourner les règles du service ?



Réflexe de production avancé

Ne vous arrêtez pas quand l'écran paraît correct. Expliquez à qui appartient l'état, les implications de sécurité, les chemins d'échec et le comportement en production avant de considérer le module comme terminé.