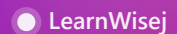


Le modèle de sécurité de Wisej.NET

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



Guide de la leçon

Ce que le navigateur peut voir et ne pas voir dans une UI côté serveur, la frontière de session, le modèle de menaces et l'endroit où se place chaque contrôle. Ce que cela implique pour un développeur qui sécurise une application Wisej.NET de production — et comment aborder ce module.



Enjeux en production

Ce que le navigateur peut voir et ne pas voir dans une UI côté serveur, la frontière de session, le modèle de menaces et l'endroit où se place chaque contrôle.

1. Le problème en production

AccessOps est un portail d'exploitation interne : départements, employés, documents confidentiels, demandes d'approbation et console d'administration. Tous ceux qui peuvent y accéder travaillent déjà pour l'entreprise, et c'est précisément pour cela que le travail de sécurité est repoussé — et qu'un seul portail finit par contenir les grilles de salaires, les courriers disciplinaires et la trace de chaque dérogation jamais accordée. Avant d'écrire la moindre ligne de code d'authentification, il vous faut trois réponses : ce que détient le navigateur, où passent les frontières, et lesquels des incidents possibles méritent d'être traités en premier.

2. Concepts clés

Concept	Pourquoi c'est important
modèle d'UI côté serveur	Les contrôles sont des objets C# dans le processus serveur ; le navigateur détient un miroir rendu et un canal d'événements, si bien que votre code et vos requêtes ne sont jamais téléchargés.
état sérialisé des widgets	Les valeurs de propriétés que Wisej.NET sérialise pour que le navigateur puisse dessiner un contrôle. Tout ce que vous affectez circule sur le réseau, y compris sur les contrôles que l'utilisateur ne voit pas.
frontière de confiance	Une ligne où les données passent entre des parties de niveaux de confiance différents. Tout ce qui la franchit est revérifié par le côté le plus fiable, car l'autre côté peut mentir.
frontière de session	La limite entre un navigateur et son propre état serveur dans <code>Application.Session</code> . Les décisions reposent sur l'état situé côté serveur de cette limite, jamais sur une valeur venue du navigateur.
point d'entrée	Toute voie d'accès qu'un attaquant peut atteindre : la page de connexion, les gestionnaires d'événements, les uploads, les téléchargements, les handlers HTTP personnalisés, les appels d'interop et le prompt de l'assistant IA.
STRIDE	Six catégories pour nommer les menaces — usurpation d'identité, falsification, répudiation, divulgation d'informations, déni de service, élévation de privilèges — afin qu'une revue soit systématique.
placement des contrôles	Choisir la frontière à laquelle une règle peut être appliquée. Une règle qui n'existe que dans l'UI est une suggestion ; la même règle dans un service est un contrôle.

3. Déroulé de la leçon

Ce que reçoit réellement le navigateur

Une application Wisej.NET est un programme .NET ordinaire. Les contrôles que vous placez sur une fenêtre sont des objets C# de l'espace de noms `Wisej.Web` qui vivent dans le processus serveur, et le navigateur en détient un miroir rendu ainsi qu'un canal pour les événements. Un utilisateur clique ; le navigateur signale le clic ; le serveur déclenche le gestionnaire dans la session qui possède le widget ; votre code s'exécute à côté de la base de données ; les propriétés modifiées sont sérialisées en retour. Aucun assembly, aucune requête LINQ, aucune règle de validation ni aucune chaîne de connexion n'est jamais envoyé au client.

Ce qui circule en revanche sur le réseau, c'est chaque valeur de propriété que vous affectez, car le navigateur ne peut pas dessiner ce qu'on ne lui a pas transmis : le texte d'un label, les éléments d'un `ComboBox` , les cellules de chaque ligne chargée dans un `DataGridView` , une infobulle, un `Tag` posé par commodité sur un nœud d'arbre. Passer `Visible` à false sur une colonne change ce qui est dessiné, pas ce qui a été envoyé. Il en va de même pour ce que vous faites délibérément sortir du processus : les fichiers envoyés avec `Application.Download` , les fichiers acceptés par un contrôle `Upload` , les handlers

HTTP personnalisés que vous enregistrez, et tout JavaScript que vous demandez au client d'exécuter, avec ses arguments.

D'où la première règle d'AccessOps : c'est une requête qui décide de ce qui quitte le serveur, pas une propriété du designer. La grille de l'annuaire reçoit une projection des colonnes que tout utilisateur peut voir ; la fourchette de salaire relève d'un second appel, contrôlé.

```
// ✗ the whole record crosses the wire
gridEmployees.DataSource = _repo.GetAllEmployees();
gridEmployees.Columns[5].Visible = false;    // Salary

// ✓ the server decides what leaves the process
gridEmployees.DataSource =
    _employees.GetDirectory(_session.CurrentUserId);
```

Quatre frontières, une application

La première frontière se situe entre le navigateur et la session Wisej.NET. Tout ce qui arrive en la franchissant est une affirmation : quel widget a été cliqué, quelle ligne de la grille est la ligne courante, ce qui a été saisi, quel id de document un lien mentionne. Une affirmation n'est pas un fait, et la session est le premier endroit où un fait peut exister.

La deuxième sépare une session du processus qui les héberge toutes. Les sessions partagent la mémoire : un champ statique, un service singleton ou un cache sans clé par utilisateur est donc visible par toute personne connectée à ce serveur. L'identité vérifiée a par conséquent sa place dans

`Application.Session`, et jamais dans un « utilisateur courant » statique. Un traitement en arrière-plan lancé avec `Application.StartTask` survit à la requête qui l'a démarré ; il doit donc transporter l'identité qui lui a été confiée plutôt que d'aller chercher la session qui se trouve être courante au moment où il se termine.

La troisième et la quatrième sont tournées vers l'extérieur, vers la base de données et vers les services externes : le fournisseur d'identité que vous ajoutez au Module 4, le stockage des documents, le relais de messagerie. Ne donnez au compte de base de données que les droits dont AccessOps a besoin et paramétrez chaque commande, afin qu'un nom de département saisi dans une zone de recherche ne puisse pas devenir du SQL. Désignez un responsable pour chaque frontière, car un contrôle qui ne se trouve sur aucune d'elles n'a rien à arrêter.

Un modèle de menaces pour AccessOps

Un modèle de menaces, c'est trois listes et un tableau. D'abord les actifs : les dossiers des employés, le stockage des documents, les décisions d'approbation, la piste d'audit qui les prouve, et les identifiants qui protègent le tout. Ensuite les points d'entrée : la page de connexion, chaque gestionnaire d'événements, le contrôle `Upload`, chaque téléchargement, tout handler HTTP personnalisé, et l'assistant IA, dont le prompt est une entrée non fiable comme une autre. Enfin les menaces, nommées par catégorie plutôt que par crainte.

- Usurpation d'identité — quelqu'un se connecte en tant que responsable financier :
authentification et identité de session au Module 2, connexion fédérée au Module 4.

- Falsification et élévation — un non-manager approuve sa propre demande : autorisation à l'intérieur du service, Module 3, et non un bouton masqué.
- Divulgence d'informations — un document parvient à quelqu'un d'extérieur à son département, ou un terme de recherche injecté renvoie des lignes interdites : Module 5.
- Répudiation — personne ne peut prouver qui a approuvé quoi, ni quand : la piste d'audit et les identifiants de corrélation, Module 6.
- Dénier de service — la page de connexion devient un point d'essai de mots de passe, ou un upload surdimensionné remplit le disque : Modules 2 et 7.

Évaluez chaque ligne deux fois, en probabilité et en impact. Un public interne modifie ces chiffres sans les annuler : un collègue curieux est plus probable et moins grave qu'un attaquant externe, un employé sur le départ qui emporte le stockage des documents est à la fois plausible et grave. La colonne la plus utile est la dernière, qui nomme le module mettant en œuvre l'atténuation — une ligne sans module est soit un risque accepté, qui doit être consigné par écrit, soit une lacune que vous avez trouvée gratuitement.

Placer chaque contrôle là où il peut être appliqué

Un contrôle ne compte qu'à une frontière qu'il peut surveiller. Retirer l'onglet Admin, désactiver le bouton Approve, filtrer une grille sur le département courant — tout cela se passe du côté non fiable de la frontière de session, et peut donc être contourné en demandant un événement que l'UI ne propose plus. Ces mesures valent la peine, car une UI qui propose des actions que l'utilisateur ne peut pas réaliser est une mauvaise UI, mais elles relèvent de l'expérience utilisateur, pas de la sécurité.

AccessOps place donc chaque décision importante dans un service situé côté serveur de cette frontière : `DocumentService` décide si cet appelant peut lire ce document, `ApprovalService` si cet appelant peut approuver cette demande, et tous deux lisent l'identité dans la session, pas dans un paramètre fourni par l'écran. L'UI reflète ces décisions et ne les prend jamais. Cela rend aussi les règles testables : un service peut être appelé avec des arguments hostiles dans un test unitaire.

La carte de placement des contrôles du lab comporte une ligne par contrôle : la règle, la frontière sur laquelle il se trouve, la classe qui l'applique, et la façon dont vous prouveriez qu'il fonctionne. Elle transforme le diagramme en un plan que les six modules restants cochent ligne par ligne.



Réflexe sécurité

Le navigateur n'est jamais digne de confiance. Chaque décision importante — qui est l'utilisateur, ce qu'il peut faire, quel enregistrement il peut voir — est prise sur le serveur, dans un service, et l'UI se contente de la refléter. Si une règle n'existe que sous la forme d'un bouton masqué, elle n'existe pas.



Objectifs d'apprentissage

À la fin de ce module, vous serez capable de :

- ✓ Décrire ce que reçoit réellement un client Wisej.NET — des widgets rendus et des événements, pas votre code ni vos données — et ce qui circule malgré tout sur le réseau.

- ✓ Identifier les frontières de confiance d'une application Wisej.NET : navigateur, session, processus serveur, base de données et services externes.
- ✓ Construire un modèle de menaces pour une application métier : identité, autorisation, exposition des données, injection, abus de session et déni de service.
- ✓ Placer chaque contrôle de sécurité à la frontière où il peut être appliqué.

Guide du lab / de l'examen

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



Lab pratique

Créez la solution AccessOps et rédigez son modèle de menaces : tracez les frontières de confiance, listez les actifs (dossiers des employés, documents, approbations), recensez les points d'entrée (connexion, uploads, téléchargements, endpoints personnalisés, l'assistant IA), évaluez chaque menace et associez chaque mesure d'atténuation au module de ce cours qui la met en œuvre. Construisez le shell avec une page de connexion provisoire.

Point de départ

Vous partez d'un simple projet Wisej.NET Web Application vide nommé AccessOps, de sa fenêtre de démarrage générée et des fichiers de configuration par défaut. Dans ce module, vous créez le shell — une page de connexion provisoire et la navigation vers les espaces départements, employés, documents, approbations et administration que le reste du cours viendra remplir — ainsi que le modèle de menaces écrit à l'aune duquel chaque module suivant sera évalué.

Approche suggérée

- ✓ Créez la solution AccessOps et une page de connexion provisoire avec une zone de nom d'utilisateur, une zone de mot de passe et un bouton Sign in dont le gestionnaire se contente pour l'instant de mettre à jour un label d'état, ainsi que des écrans vides pour les départements, les employés, les documents, les approbations et la console d'administration.
- ✓ Tracez le diagramme des frontières de confiance avec quatre zones — navigateur, session Wisej.NET, processus serveur, et base de données avec les services externes — et indiquez-y précisément ce qui franchit chaque ligne, y compris les uploads, les téléchargements, les handlers personnalisés et l'interop JavaScript.
- ✓ Rédigez l'inventaire des actifs et des points d'entrée : dossiers des employés, documents, décisions d'approbation, piste d'audit et identifiants dans une liste ; la page de connexion, les gestionnaires d'événements de chaque écran, le contrôle Upload, les téléchargements, les handlers HTTP personnalisés et l'assistant IA dans l'autre.
- ✓ Construisez le tableau des menaces en passant chaque point d'entrée au crible de l'usurpation d'identité, de la falsification, de la répudiation, de la divulgation d'informations, du déni de service et de l'élévation de privilèges, puis évaluez chaque ligne en probabilité et en impact, et nommez l'atténuation ainsi que le module qui la met en œuvre.
- ✓ Produisez la carte de placement des contrôles : une ligne par contrôle, avec la règle, la frontière sur laquelle il se trouve, la classe qui l'appliquera, et la preuve de son bon fonctionnement que vous montreriez à un relecteur.

Livrables attendus

- ✓ Solution AccessOps avec une page de connexion provisoire

- ✓ Diagramme des frontières de confiance
- ✓ Inventaire des actifs et des points d'entrée
- ✓ Tableau des menaces avec probabilité, impact et atténuation
- ✓ Carte de placement des contrôles de sécurité

Questions de révision

- ✓ Votre grille des employés est liée aux dossiers complets des employés et la colonne du salaire est masquée dans le designer. Décrivez ce que détient le navigateur, et nommez la modification qui retire réellement le salaire de ce qui circule sur le réseau.
- ✓ Deux personnes sont connectées à AccessOps sur le même serveur. Expliquez quelles parties de l'état elles partagent par défaut, et ce qui devrait être vrai pour que l'une d'elles voie le département courant de l'autre.
- ✓ Choisissez la menace de votre tableau dont l'impact est le plus élevé et défendez la thèse inverse : que faudrait-il qu'il soit vrai d'AccessOps pour que cette atténuation soit une perte de temps pour l'équipe ?



Réflexe sécurité

Le navigateur n'est jamais digne de confiance. Chaque décision importante — qui est l'utilisateur, ce qu'il peut faire, quel enregistrement il peut voir — est prise sur le serveur, dans un service, et l'UI se contente de la refléter. Si une règle n'existe que sous la forme d'un bouton masqué, elle n'existe pas.