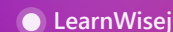


Il modello di sicurezza di Wisej.NET

Guida della lezione · Guida al lab / esame



Guida della lezione

Che cosa il browser può e non può vedere in una UI lato server, il confine di sessione, il modello delle minacce e dove si colloca ciascun controllo di sicurezza. Che cosa significa per uno sviluppatore che mette in sicurezza un'app Wisej.NET di produzione — e come affrontare questo modulo.



Focus di produzione

Che cosa il browser può e non può vedere in una UI lato server, il confine di sessione, il modello delle minacce e dove si colloca ciascun controllo di sicurezza.

1. Il problema in produzione

AccessOps è un portale operativo interno: reparti, dipendenti, documenti riservati, richieste di approvazione e una console di amministrazione. Chiunque possa raggiungerlo lavora già per l'azienda, ed è esattamente per questo che il lavoro sulla sicurezza viene rimandato — e per questo un unico portale finisce per contenere fasce retributive, lettere disciplinari e la traccia delle approvazioni di ogni eccezione che l'azienda abbia mai concesso. Prima che esista una sola riga di codice di autenticazione ti servono tre risposte: che cosa contiene il browser, dove passano i confini, e quali delle cose che potrebbero andare storte vale la pena contrastare per prime.

2. Concetti chiave

Concetto	Perché è importante
modello di UI lato server	I controlli sono oggetti C# nel processo server; il browser ne tiene uno specchio renderizzato e un canale per gli eventi, quindi il tuo codice e le tue query non vengono mai scaricati.
payload di stato dei widget	I valori delle proprietà che Wisej.NET serializza perché il browser possa disegnare un controllo. Tutto ciò che assegni viaggia in rete, anche sui controlli che l'utente non può vedere.
confine di fiducia	Una linea dove i dati passano tra parti con un diverso livello di fiducia. Tutto ciò che la attraversa viene ricontrollato dalla parte più affidabile, perché l'altra parte può mentire.
confine di sessione	La frontiera tra un browser e il suo stato sul server in <code>Application.Session</code> . Le decisioni nascono dallo stato sul lato server di questo confine, mai da un valore del browser.
punto di ingresso	Qualsiasi via d'accesso raggiungibile da un attaccante: la pagina di login, i gestori di evento, gli upload, i download, gli handler HTTP personalizzati, le chiamate di interop e il prompt dell'assistente IA.
STRIDE	Sei categorie per dare un nome alle minacce — spoofing, manomissione, ripudio, divulgazione di informazioni, denial of service, elevazione dei privilegi — così che una revisione sia sistematica.
posizionamento dei controlli	Scegliere il confine su cui una regola può essere fatta rispettare. Una regola che esiste solo nella UI è un suggerimento; la stessa regola dentro un servizio è un controllo.

3. Il filo della lezione

Che cosa riceve davvero il browser

Un'applicazione Wisej.NET è un normale programma .NET. I controlli che metti su una finestra sono oggetti C# del namespace `Wisej.Web` che vivono nel processo server, e il browser ne tiene uno specchio renderizzato più un canale per gli eventi. Un utente clicca; il browser segnala il clic; il server solleva il gestore nella sessione che possiede il widget; il tuo codice gira accanto al database; le proprietà modificate vengono serializzate indietro. Nessun assembly, query LINQ, regola di validazione o stringa di connessione viene mai inviato al client.

Ciò che invece viaggia in rete è ogni valore di proprietà che assegni, perché il browser non può disegnare ciò che non gli è stato comunicato: il testo di una label, gli elementi di una `ComboBox`, le celle di ogni riga caricata in una `DataGridView`, un tooltip, un `Tag` parcheggiato su un nodo dell'albero per comodità. Impostare `Visible` a false su una colonna cambia ciò che viene disegnato, non ciò che è stato inviato. Lo stesso vale per ciò che fai uscire deliberatamente dal processo: i file inviati con `Application.Download`, i file accettati da un controllo `Upload`, gli handler HTTP personalizzati che registri e qualsiasi JavaScript che chiedi al client di eseguire, con i suoi argomenti.

Ne segue la prima regola di AccessOps: è una query a decidere che cosa lascia il server, non una proprietà del designer. La griglia dell'elenco del personale riceve una proiezione delle colonne che ogni utente può vedere; la fascia retributiva è una seconda chiamata, verificata.

```
// ✗ l'intero record viaggia in rete
gridEmployees.DataSource = _repo.GetAllEmployees();
gridEmployees.Columns[5].Visible = false; // Stipendio

// ✓ è il server a decidere che cosa lascia il processo
gridEmployees.DataSource =
    _employees.GetDirectory(_session.CurrentUserId);
```

Quattro confini, un'applicazione

Il primo confine sta tra il browser e la sessione Wisej.NET. Tutto ciò che arriva attraverso di esso è un'affermazione: quale widget è stato cliccato, quale riga della griglia è quella corrente, che cosa è stato digitato, quale id di documento cita un link. Un'affermazione non è un fatto, e la sessione è il primo punto in cui un fatto può esistere.

Il secondo separa una sessione dal processo che le ospita tutte. Le sessioni condividono la memoria, quindi un campo statico, un servizio singleton o una cache senza una chiave per utente sono visibili a chiunque abbia effettuato l'accesso a quel server. L'identità verificata va quindi in `Application.Session` e mai in un "utente corrente" statico. L'elaborazione in background avviata con `Application.StartTask` sopravvive alla richiesta che lo ha avviato, quindi deve portare con sé l'identità ricevuta invece di andare a recuperare la sessione che risulta corrente quando termina.

Il terzo e il quarto guardano verso l'esterno, al database e ai servizi esterni: l'identity provider che aggiungerai nel Modulo 4, l'archivio dei documenti, il relay di posta. Dai al login del database solo i diritti di cui AccessOps ha bisogno e parametrizza ogni comando, così che il nome di un reparto digitato in una casella di ricerca non possa diventare SQL. Assegna un responsabile a ogni confine, perché un controllo che non sta su nessuno di essi non ha nulla da fermare.

Un modello delle minacce per AccessOps

Un modello delle minacce è fatto di tre elenchi e una tabella. Prima gli asset: le anagrafiche dei dipendenti, l'archivio dei documenti, le decisioni di approvazione, l'audit trail che le dimostra e le credenziali che proteggono tutto questo. Poi i punti di ingresso: la pagina di login, ogni gestore di evento, il controllo `Upload`, ogni download, qualsiasi handler HTTP personalizzato e l'assistente IA, il cui prompt è input non attendibile come qualsiasi altro. Poi le minacce, chiamate per categoria e non per paura.

- Spoofing — qualcuno accede come responsabile finanziario: autenticazione e identità di sessione nel Modulo 2, login federato nel Modulo 4.
- Manomissione ed elevazione — un non-manager approva la propria richiesta: autorizzazione dentro il servizio, Modulo 3, non un pulsante nascosto.
- Divulgazione di informazioni — un documento raggiunge qualcuno fuori dal suo reparto, oppure un termine di ricerca con injection restituisce righe proibite: Modulo 5.
- Ripudio — nessuno riesce a dimostrare chi ha approvato che cosa, né quando: l'audit trail e i correlation ID, Modulo 6.

- Denial of service — la pagina di login diventa un endpoint per tirare a indovinare, oppure un upload enorme riempie il disco: Moduli 2 e 7.

Valuta ogni riga due volte, per probabilità e per impatto. Un pubblico interno cambia questi numeri senza azzerarli: un collega curioso è più probabile e meno dannoso di un estraneo, un dipendente in uscita che si porta via l'archivio dei documenti è sia plausibile sia grave. La colonna più utile è l'ultima, quella che indica il modulo che implementa la mitigazione — una riga senza nulla accanto è un rischio accettato, che deve dirlo per iscritto, oppure una lacuna che hai trovato gratis.

Mettere ogni controllo dove può essere fatto rispettare

Un controllo conta solo su un confine che è in grado di sorvegliare. Rimuovere la scheda Admin, disabilitare il pulsante Approve, filtrare una griglia sul reparto corrente — ognuna di queste cose avviene sul lato non attendibile del confine di sessione, quindi ognuna può essere aggirata chiedendo un evento che la UI non offre più. Vale la pena farle, perché una UI che propone lavoro che l'utente non può svolgere è una cattiva UI, ma sono esperienza utente, non sicurezza.

AccessOps mette quindi ogni decisione che conta in un servizio sul lato server di quel confine:

`DocumentService` decide se questo chiamante può leggere questo documento, `ApprovalService` se questo chiamante può approvare questa richiesta, ed entrambi leggono l'identità dalla sessione, non da un parametro fornito dalla schermata. La UI riflette queste decisioni e non le prende mai. In più rende le regole testabili: un servizio può essere chiamato con argomenti ostili in uno unit test.

La mappa di posizionamento dei controlli del lab ha una riga per controllo: la regola, il confine su cui si trova, la classe che la fa rispettare e il modo in cui dimostreresti che funziona. Trasforma il diagramma in un piano che i sei moduli rimanenti spuntano riga per riga.



La regola sulla sicurezza

Il browser non è mai considerato affidabile. Ogni decisione che conta — chi è l'utente, che cosa può fare, quale record può vedere — viene presa sul server, in un servizio, e la UI si limita a rifletterla. Se una regola esiste solo come pulsante nascosto, non esiste.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Descrivere che cosa riceve davvero un client Wisej.NET — widget renderizzati ed eventi, non il tuo codice né i tuoi dati — e che cosa viaggia comunque in rete.
- ✓ Individuare i confini di fiducia in un'app Wisej.NET: browser, sessione, processo server, database e servizi esterni.
- ✓ Costruire un modello delle minacce per un'applicazione gestionale: identità, autorizzazione, esposizione dei dati, injection, abuso della sessione e denial of service.
- ✓ Collocare ogni controllo di sicurezza sul confine in cui può essere fatto rispettare.

Guida al lab / esame

Che cosa costruirai nel lab pratico, l'approccio suggerito e i deliverable richiesti.



Lab pratico

Crea la soluzione AccessOps e scrivine il modello delle minacce: disegna i confini di fiducia, elenca gli asset (anagrafiche dei dipendenti, documenti, approvazioni), enumera i punti di ingresso (login, upload, download, endpoint personalizzati, l'assistente IA), valuta ogni minaccia e associa ogni mitigazione al modulo di questo corso che la implementa. Costruisci lo shell con una pagina di login segnaposto.

Punto di partenza

Parti da nient'altro che un progetto vuoto di applicazione web Wisej.NET chiamato AccessOps, la sua finestra di avvio generata e i file di configurazione predefiniti. In questo modulo crei lo shell — una pagina di login segnaposto e la navigazione per le aree reparti, dipendenti, documenti, approvazioni e amministrazione che il resto del corso riempirà — e il modello delle minacce scritto rispetto al quale verrà misurato ogni modulo successivo.

Approccio suggerito

- ✓ Crea la soluzione AccessOps e una pagina di login segnaposto con una casella per il nome utente, una per la password e un pulsante Sign in il cui gestore per ora non fa altro che impostare una label di stato, più schermate vuote per reparti, dipendenti, documenti, approvazioni e la console di amministrazione.
- ✓ Disegna il diagramma dei confini di fiducia con quattro regioni — browser, sessione Wisej.NET, processo server, e database e servizi esterni — e segna esattamente che cosa attraversa ogni linea, compresi upload, download, handler personalizzati e interop JavaScript.
- ✓ Scrivi l'inventario di asset e punti di ingresso: anagrafiche dei dipendenti, documenti, decisioni di approvazione, audit trail e credenziali in un elenco; la pagina di login, i gestori di evento di ogni schermata, il controllo Upload, i download, gli handler HTTP personalizzati e l'assistente IA nell'altro.
- ✓ Costruisci la tabella delle minacce passando ogni punto di ingresso attraverso spoofing, manomissione, ripudio, divulgazione di informazioni, denial of service ed elevazione dei privilegi, poi valuta ogni riga per probabilità e impatto e indica la mitigazione e il modulo che la realizza.
- ✓ Produci la mappa di posizionamento dei controlli: una riga per controllo con la regola, il confine su cui si trova, la classe che la farà rispettare e la prova che mostreresti a un revisore per dimostrare che funziona.

Deliverable richiesti

- ✓ Soluzione AccessOps con una pagina di login segnaposto
- ✓ Diagramma dei confini di fiducia
- ✓ Inventario di asset e punti di ingresso
- ✓ Tabella delle minacce con probabilità, impatto e mitigazione

- ✓ Mappa di posizionamento dei controlli di sicurezza

Domande di revisione per lo studente

- ✓ La griglia dei dipendenti è collegata ai record completi e la colonna dello stipendio è nascosta nel designer. Descrivi che cosa contiene il browser e indica la modifica che toglie davvero lo stipendio da ciò che viaggia in rete.
- ✓ Due persone hanno effettuato l'accesso ad AccessOps sullo stesso server. Spiega quali parti dello stato condividono per impostazione predefinita, e che cosa dovrebbe essere vero perché una delle due veda il reparto corrente dell'altra.
- ✓ Scegli la minaccia della tua tabella con l'impatto più alto e sostieni la tesi opposta: che cosa dovrebbe essere vero di AccessOps perché quella mitigazione sia una perdita di tempo per il team?



La regola sulla sicurezza

Il browser non è mai considerato affidabile. Ogni decisione che conta — chi è l'utente, che cosa può fare, quale record può vedere — viene presa sul server, in un servizio, e la UI si limita a rifletterla. Se una regola esiste solo come pulsante nascosto, non esiste.