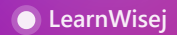


Das Sicherheitsmodell von Wisej.NET

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Was der Browser in einer serverseitigen UI sehen kann und was nicht, die Session-Grenze, das Bedrohungsmodell und wo jede Sicherheitskontrolle ihren Platz hat. Was das für Entwickler bedeutet, die eine Wisej.NET-App im Produktivbetrieb absichern – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Was der Browser in einer serverseitigen UI sehen kann und was nicht, die Session-Grenze, das Bedrohungsmodell und wo jede Sicherheitskontrolle ihren Platz hat.

1. Das Problem im Produktivbetrieb

AccessOps ist ein internes Betriebsportal: Abteilungen, Mitarbeitende, vertrauliche Dokumente, Genehmigungsanträge und eine Admin-Konsole. Wer es erreichen kann, arbeitet ohnehin schon für das Unternehmen – genau deshalb wird die Sicherheitsarbeit immer wieder verschoben, und genau deshalb enthält am Ende ein einziges Portal Gehaltsbänder, Abmahnungen und die Genehmigungshistorie jeder Ausnahme, die das Unternehmen je gewährt hat. Bevor auch nur eine Zeile Authentifizierungscode existiert, brauchen Sie drei Antworten: was der Browser in der Hand hält, wo die Grenzen verlaufen und welche der möglichen Fehlentwicklungen es wert sind, zuerst technisch abgesichert zu werden.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
serverseitiges UI-Modell	Controls sind C#-Objekte im Serverprozess; der Browser hält nur ein gerendertes Abbild und einen Event-Kanal, daher werden Ihr Code und Ihre Abfragen nie heruntergeladen.
Widget-Zustandsdaten	Die Eigenschaftswerte, die Wisej.NET serialisiert, damit der Browser ein Control zeichnen kann. Alles, was Sie zuweisen, geht über die Leitung – auch bei Controls, die der Benutzer nicht sieht.
Vertrauensgrenze	Eine Linie, an der Daten zwischen Parteien mit unterschiedlichem Vertrauensniveau wechseln. Alles, was sie überquert, prüft die vertrauenswürdigeren Seite erneut, denn die andere Seite kann lügen.
Session-Grenze	Die Grenze zwischen einem Browser und seinem eigenen Serverzustand in <code>Application.Session</code> . Entscheidungen beruhen auf dem Zustand auf der Serverseite dieser Grenze, nie auf einem Wert aus dem Browser.
Einstiegspunkt	Jeder Weg hinein, den ein Angreifer erreichen kann: die Login-Seite, Event-Handler, Uploads, Downloads, eigene HTTP-Handler, Interop-Aufrufe und der Prompt des KI-Assistenten.
STRIDE	Sechs Kategorien zur Benennung von Bedrohungen – Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege –, damit ein Review systematisch abläuft.
Platzierung von Kontrollen	Die Wahl der Grenze, an der eine Regel durchgesetzt werden kann. Eine Regel, die nur in der UI existiert, ist ein Vorschlag; dieselbe Regel in einem Service ist eine Kontrolle.

3. Der rote Faden der Lektion

Was der Browser tatsächlich erhält

Eine Wisej.NET-Anwendung ist ein gewöhnliches .NET-Programm. Die Controls, die Sie auf einem Fenster platzieren, sind C#-Objekte aus dem Namespace `Wisej.Web`, die im Serverprozess leben; der Browser hält ein gerendertes Abbild davon und einen Kanal für Events. Ein Benutzer klickt, der Browser meldet den Klick, der Server löst den Handler in der Session aus, der das Widget gehört, Ihr Code läuft direkt neben der Datenbank, und die geänderten Eigenschaften werden zurückserialisiert. Keine Assembly, keine LINQ-Abfrage, keine Validierungsregel und kein Connection-String wird je an den Client ausgeliefert.

Was dagegen über die Leitung geht, ist jeder Eigenschaftswert, den Sie zuweisen, denn der Browser kann nicht zeichnen, was man ihm nicht mitgeteilt hat: der Text eines Labels, die Einträge einer `ComboBox`, die Zellen jeder Zeile, die Sie in ein `DataGridView` geladen haben, ein Tooltip, ein der Bequemlichkeit halber an einem Baumknoten abgelegtes `Tag`. `Visible` bei einer Spalte auf false zu setzen, ändert, was gezeichnet wird, nicht, was gesendet wurde. Dasselbe gilt für alles, was Sie bewusst aus dem Prozess hinausgeben: Dateien, die mit `Application.Download` gestreamt werden, Dateien, die ein `Upload`-Control annimmt, eigene HTTP-Handler, die Sie registrieren, und jedes JavaScript, das der Client samt Argumenten ausführen soll.

Daraus folgt die erste AccessOps-Regel: Eine Abfrage entscheidet, was den Server verlässt, nicht eine Designer-Eigenschaft. Das Verzeichnis-Grid erhält eine Projektion der Spalten, die jeder Betrachter sehen darf; das Gehaltsband ist ein zweiter, geprüfter Aufruf.

```
// ✗ der komplette Datensatz geht über die Leitung
gridEmployees.DataSource = _repo.GetAllEmployees();
gridEmployees.Columns[5].Visible = false;    // Gehalt

// ✓ der Server entscheidet, was den Prozess verlässt
gridEmployees.DataSource =
    _employees.GetDirectory(_session.CurrentUserId);
```

Vier Grenzen, eine Anwendung

Die erste Grenze liegt zwischen dem Browser und der Wisej.NET-Session. Alles, was über sie hereinkommt, ist eine Behauptung: welches Widget angeklickt wurde, welche Grid-Zeile die aktuelle ist, was eingetippt wurde, welche Dokument-ID ein Link nennt. Eine Behauptung ist keine Tatsache, und die Session ist der erste Ort, an dem eine Tatsache existieren kann.

Die zweite trennt eine Session von dem Prozess, der alle Sessions beherbergt. Sessions teilen sich den Speicher, daher ist ein statisches Feld, ein Singleton-Service oder ein Cache ohne benutzerbezogenen Schlüssel für alle sichtbar, die auf diesem Server angemeldet sind. Die verifizierte Identität gehört deshalb in `Application.Session` und niemals in einen statischen „aktuellen Benutzer“. Hintergrundverarbeitung, die mit `Application.StartTask` gestartet wird, überlebt den Request, der sie ausgelöst hat; sie muss also die Identität mitführen, die ihr übergeben wurde, statt beim Abschluss nach der gerade aktuellen Session zu greifen.

Die dritte und vierte Grenze zeigen nach außen, zur Datenbank und zu externen Diensten: dem Identity Provider, den Sie in Modul 4 hinzufügen, dem Dokumentenspeicher, dem Mail-Relay. Geben Sie dem Datenbank-Login nur die Rechte, die AccessOps braucht, und parametrisieren Sie jeden Befehl, damit ein in ein Suchfeld eingetippter Abteilungsname nicht zu SQL werden kann. Benennen Sie für jede Grenze einen Verantwortlichen, denn eine Kontrolle, die auf keiner Grenze sitzt, hat nichts aufzuhalten.

Ein Bedrohungsmodell für AccessOps

Ein Bedrohungsmodell besteht aus drei Listen und einer Tabelle. Zuerst die Assets: Mitarbeiterdatensätze, der Dokumentenspeicher, Genehmigungsentscheidungen, der Audit-Trail, der sie belegt, und die Zugangsdaten, die all das schützen. Dann die Einstiegspunkte: die Login-Seite, jeder Event-Handler, das `Upload`-Control, jeder Download, jeder eigene HTTP-Handler und der KI-Assistent, dessen Prompt eine nicht vertrauenswürdige Eingabe ist wie jede andere. Dann die Bedrohungen, benannt nach Kategorie statt nach Bauchgefühl.

- Spoofing – jemand meldet sich als Finanzleiter an: Authentifizierung und Session-Identität in Modul 2, föderiertes Login in Modul 4.
- Tampering und Elevation – jemand ohne Führungsrolle genehmigt den eigenen Antrag: Autorisierung im Service, Modul 3, kein versteckter Button.
- Information Disclosure – ein Dokument erreicht jemanden außerhalb seiner Abteilung, oder ein eingeschleuster Suchbegriff liefert verbotene Zeilen: Modul 5.

- Repudiation – niemand kann belegen, wer was genehmigt hat und wann: Audit-Trail und Korrelations-IDs, Modul 6.
- Denial of Service – die Login-Seite wird zum Endpunkt fürs Passwortraten, oder ein übergroßer Upload füllt die Festplatte: Module 2 und 7.

Bewerten Sie jede Zeile zweimal, nach Eintrittswahrscheinlichkeit und nach Auswirkung. Ein internes Publikum verändert diese Werte, ohne sie aufzuheben: Eine neugierige Kollegin ist wahrscheinlicher und harmloser als ein Außenstehender, ein ausscheidender Mitarbeiter, der den Dokumentenspeicher mitnimmt, ist sowohl plausibel als auch gravierend. Die nützlichste Spalte ist die letzte, die das Modul nennt, das die Gegenmaßnahme umsetzt – eine Zeile ohne Eintrag ist entweder ein akzeptiertes Risiko, das dann auch schriftlich so festgehalten werden sollte, oder eine Lücke, die Sie gratis gefunden haben.

Jede Kontrolle dort platzieren, wo sie durchsetzbar ist

Eine Kontrolle zählt nur an einer Grenze, die sie überwachen kann. Den Admin-Tab entfernen, den Approve-Button deaktivieren, ein Grid auf die aktuelle Abteilung filtern – all das geschieht auf der nicht vertrauenswürdigen Seite der Session-Grenze und lässt sich umgehen, indem man ein Event anfordert, das die UI nicht mehr anbietet. Es lohnt sich trotzdem, denn eine UI, die Arbeit anbietet, die der Benutzer nicht ausführen darf, ist eine schlechte UI – aber es ist User Experience, keine Sicherheit.

AccessOps legt deshalb jede relevante Entscheidung in einen Service auf der Serverseite dieser Grenze: `DocumentService` entscheidet, ob dieser Aufrufer dieses Dokument lesen darf, `ApprovalService`, ob dieser Aufrufer diesen Antrag genehmigen darf, und beide lesen die Identität aus der Session, nicht aus einem Parameter, den der Bildschirm geliefert hat. Die UI spiegelt diese Entscheidungen wider und trifft sie nie selbst. Außerdem werden die Regeln so testbar: Ein Service lässt sich in einem Unit-Test mit feindseligen Argumenten aufrufen.

Die Platzierungsübersicht der Kontrollen im Lab hat eine Zeile pro Kontrolle: die Regel, die Grenze, auf der sie sitzt, die Klasse, die sie durchsetzt, und wie Sie belegt würden, dass sie funktioniert. So wird aus dem Diagramm ein Plan, den die übrigen sechs Module Zeile für Zeile abarbeiten.



Sicherheitsgewohnheit

Dem Browser wird nie vertraut. Jede relevante Entscheidung – wer der Benutzer ist, was er tun darf, welchen Datensatz er sehen darf – wird auf dem Server in einem Service getroffen, und die UI spiegelt sie nur wider. Existiert eine Regel nur als versteckter Button, dann existiert sie nicht.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ beschreiben, was ein Wisej.NET-Client tatsächlich erhält – gerenderte Widgets und Events, nicht Ihren Code oder Ihre Daten – und was dennoch über die Leitung geht.
- ✓ die Vertrauensgrenzen in einer Wisej.NET-App identifizieren: Browser, Session, Serverprozess, Datenbank und externe Dienste.
- ✓ ein Bedrohungsmodell für eine Geschäftsanwendung erstellen: Identität, Autorisierung, Datenpreisgabe, Injection, Session-Missbrauch und Denial of Service.

- ✓ jede Sicherheitskontrolle an der Grenze platzieren, an der sie durchsetzbar ist.

Lab- und Prüfungsleitfaden

Was Sie im praktischen Lab erstellen, das vorgeschlagene Vorgehen und die geforderten Ergebnisse.



Praxis-Lab

Legen Sie die AccessOps-Solution an und schreiben Sie ihr Bedrohungsmodell: Zeichnen Sie die Vertrauensgrenzen, listen Sie die Assets auf (Mitarbeiterdatensätze, Dokumente, Genehmigungen), erfassen Sie die Einstiegspunkte (Login, Uploads, Downloads, eigene Endpunkte, den KI-Assistenten), bewerten Sie jede Bedrohung und ordnen Sie jede Gegenmaßnahme dem Modul dieses Kurses zu, das sie umsetzt. Bauen Sie das Grundgerüst mit einer Platzhalter-Login-Seite.

Ausgangspunkt

Sie beginnen mit nichts als einem leeren Wisej.NET-Web-Application-Projekt namens AccessOps, seinem generierten Startfenster und den Standard-Konfigurationsdateien. In diesem Modul erstellen Sie das Grundgerüst – eine Platzhalter-Login-Seite und die Navigation für die Bereiche Abteilungen, Mitarbeitende, Dokumente, Genehmigungen und Administration, die der restliche Kurs füllt – sowie das schriftliche Bedrohungsmodell, an dem jedes spätere Modul gemessen wird.

Vorgeschlagenes Vorgehen

- ✓ Legen Sie die AccessOps-Solution und eine Platzhalter-Login-Seite an, mit einem Feld für den Benutzernamen, einem Passwortfeld und einem Sign-in-Button, dessen Handler vorerst nichts tut, außer ein Status-Label zu setzen; dazu leere Bildschirme für Abteilungen, Mitarbeitende, Dokumente, Genehmigungen und die Admin-Konsole.
- ✓ Zeichnen Sie das Diagramm der Vertrauensgrenzen mit vier Bereichen – Browser, Wisej.NET-Session, Serverprozess sowie Datenbank und externe Dienste – und markieren Sie darauf genau, was jede Linie überquert, einschließlich Uploads, Downloads, eigener Handler und JavaScript-Interop.
- ✓ Schreiben Sie das Inventar der Assets und Einstiegspunkte: Mitarbeiterdatensätze, Dokumente, Genehmigungsentscheidungen, Audit-Trail und Zugangsdaten auf die eine Liste; die Login-Seite, die Event-Handler jedes Bildschirms, das Upload-Control, Downloads, eigene HTTP-Handler und den KI-Assistenten auf die andere.
- ✓ Erstellen Sie die Bedrohungstabelle, indem Sie jeden Einstiegspunkt durch Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service und Elevation of Privilege führen; bewerten Sie dann jede Zeile nach Eintrittswahrscheinlichkeit und Auswirkung und nennen Sie die Gegenmaßnahme und das Modul, das sie liefert.
- ✓ Erstellen Sie die Platzierungsübersicht der Kontrollen: eine Zeile pro Kontrolle mit der Regel, der Grenze, auf der sie sitzt, der Klasse, die sie durchsetzen wird, und dem Nachweis, mit dem Sie einem Reviewer zeigen würden, dass sie funktioniert.

Geforderte Ergebnisse

- ✓ AccessOps-Solution mit einer Platzhalter-Login-Seite

- ✓ Diagramm der Vertrauensgrenzen
- ✓ Inventar der Assets und Einstiegspunkte
- ✓ Bedrohungstabelle mit Eintrittswahrscheinlichkeit, Auswirkung und Gegenmaßnahme
- ✓ Platzierungsübersicht der Sicherheitskontrollen

Wiederholungsfragen

- ✓ Ihr Mitarbeiter-Grid ist an vollständige Mitarbeiterdatensätze gebunden, und die Gehaltsspalte ist im Designer ausgeblendet. Beschreiben Sie, was der Browser in der Hand hält, und nennen Sie die Änderung, die das Gehalt tatsächlich von der Leitung nimmt.
- ✓ Zwei Personen sind auf demselben Server bei AccessOps angemeldet. Erklären Sie, welche Teile des Zustands sie standardmäßig teilen und was zutreffen müsste, damit eine von ihnen die aktuelle Abteilung der anderen sieht.
- ✓ Wählen Sie die Bedrohung in Ihrer Tabelle mit der größten Auswirkung und vertreten Sie die Gegenposition: Was müsste für AccessOps gelten, damit diese Gegenmaßnahme Zeitverschwendung für das Team wäre?



Sicherheitsgewohnheit

Dem Browser wird nie vertraut. Jede relevante Entscheidung – wer der Benutzer ist, was er tun darf, welchen Datensatz er sehen darf – wird auf dem Server in einem Service getroffen, und die UI spiegelt sie nur wider. Existiert eine Regel nur als versteckter Button, dann existiert sie nicht.