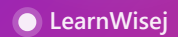


Sicherheitsarchitektur: Identität, SSO, Autorisierung, Audit und sicheres Deployment

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Integration von Identitäten, Berechtigungsmodell, Audit-Trails, sicheres HTML, sichere Konfiguration. Was das für ein Wisej.NET-Team auf Expertenniveau bedeutet – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Integration von Identitäten, Berechtigungsmodell, Audit-Trails, sicheres HTML, sichere Konfiguration.

1. Das Problem im Produktivbetrieb

Eine serverseitige Architektur verkleinert viele Angriffsflächen im Browser, doch anspruchsvolle Systeme brauchen trotzdem eine Integration von Identitäten, Autorisierung auf Service-Ebene, Audit-Trails, sicheres Rendering und eine sichere Betriebskonfiguration.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
OIDC/SSO-Grenze	Die Stelle, an der ein externer Identity Provider der Anwendung eine verifizierte Identität übergibt. Alles jenseits dieser Grenze liegt in der Verantwortung der Anwendung.
Claims-Mapping	Die Übersetzung providerspezifischer Claims in die eigenen Rollen und Berechtigungen der Anwendung, damit der Autorisierungscode nie vom Format eines bestimmten Identity Providers abhängt.
Berechtigungsservice	Ein einzelner Service, der beantwortet, ob der aktuelle Benutzer eine Aktion ausführen darf, sodass die Regel an einer Stelle liegt statt auf jedem Bildschirm.
Autorisierung auf Service-Ebene	Berechtigungen werden innerhalb der Anwendungsservices durchgesetzt, nicht nur in der UI, damit eine ausgeblendete Schaltfläche nie der einzige Schutz einer Operation ist.
Audit-Log	Ein manipulationssicherer Nachweis, wer wann was bei wichtigen Aktionen getan hat, damit sich Sicherheitsvorfälle untersuchen und Compliance-Anforderungen belegen lassen.
AllowHtml-Review	Die Prüfung jeder Stelle, an der ein Control rohes HTML zulässt, um sicherzustellen, dass von Benutzern gelieferte Inhalte bereinigt werden – denn ungeprüftes HTML ist ein Cross-Site-Scripting-Risiko.
Umgang mit Secrets	Connection Strings, Schlüssel und Tokens gehören in eine sichere Konfiguration oder einen Vault, niemals in die Versionsverwaltung oder in Einstellungen, die der Client sehen kann.
Sichere Header	HTTP-Antwortheader wie Content Security Policy und Strict Transport Security, die die Angriffsfläche der Anwendung im Browser verkleinern.

3. Der rote Faden der Lektion

Authentifizierung ist keine Autorisierung

Ein angemeldeter Benutzer darf nicht automatisch einen Arbeitsauftrag freigeben, einen anderen Mandanten einsehen, Daten exportieren oder eine Administrationsaufgabe ausführen. Die Berechtigungsprüfung gehört in den Service, der die Aktion ausführt.

Der Session-Kontext verbindet Identität und UI

Nach der Authentifizierung sollte die Anwendung die Identität in einen Session-Kontext übertragen: Benutzer-ID, Anzeigename, Mandant, Rollen, Berechtigungen, Kultur, Theme und Korrelations-IDs. Dieser Kontext hilft den Bildschirmen, sich anzupassen, während die Services weiterhin die Regeln durchsetzen.

Ein Identitäts- und Autorisierungsmodell für das Unternehmen

Autorisierung im Unternehmen hat mehr Dimensionen als einen Rollennamen. Dieselbe Person kann in einem Mandanten Disponent sein, in einem anderen ein Auditor mit reinen Leserechten und Arbeitsaufträge nur unterhalb einer Kostengrenze freigeben dürfen. Ein Modell, das das ausdrücken kann, beginnt mit drei Ebenen: der Identität, die sagt, wer der Benutzer ist; der Mitgliedschaft, die sagt, welche Mandanten und Rollen er hat; und den Berechtigungen, also den feingranularen Aktionen, die eine Rolle gewährt, etwa `ApproveWorkOrder` oder `ExportData`.

Das Lab macht daraus eine Berechtigungsmatrix: Die Zeilen sind Berechtigungen, die Spalten Rollen, und jede Zelle ist gewährt, verweigert oder an Bedingungen geknüpft. Die Matrix ist das Artefakt, das Product Owner, Sicherheitsprüfer und Entwickler gleichermaßen lesen können, deshalb sollte sie vor dem Code existieren. Auch Workflow-Schritte werden darauf abgebildet. Eine Eskalation einreichen, sie freigeben und sie schließen kann jeweils eine andere Berechtigung erfordern, und die Matrix macht das explizit.

Denken Sie das Modell vom Mandanten her. Eine Berechtigung wird immer innerhalb eines Mandantenkontexts ausgewertet, sodass ein Benutzer mit einer Rolle in Mandant A in Mandant B nichts erhält. Der Aufruf `PermissionService.Demand` im Lab prüft zuerst die Mandantensperre und fragt erst danach den Rollenspeicher ab.

Authentifizierungs-Gates, Session-Kontext und Services

Ein externer Identity Provider, erreicht über eine OpenID-Connect- oder SSO-Anmeldung, teilt der Anwendung mit, wer der Benutzer ist. Dort endet seine Aufgabe. Die Anwendung überträgt die eingehenden Claims anschließend in ihren eigenen Session-Kontext: Benutzer-ID, Anzeigename, Mandant, Rollen, Berechtigungen und eine Korrelations-ID. Da dieses Mapping einmal bei der Anmeldung läuft und in der Session gespeichert wird, können Bildschirme den Kontext günstig lesen und Services sich darauf verlassen, ohne die Claims erneut auszuwerten.

Das Gate selbst muss vor jedem Einstiegspunkt stehen, nicht nur vor der Startseite. Hintergrundjobs, WebMethod-Aufrufe, Postback-Handler und Diagnoseseiten müssen alle wissen, wer aufruft. Ein Service sollte die Identität nie aus einem UI-Control, einem Query-String oder einem versteckten Feld lesen, denn all das kann der Benutzer verändern. Er liest den Session-Kontext, und dieser wurde aus verifizierten Claims aufgebaut.

Gestalten Sie den Kontext so, dass er sich aktualisieren lässt. Rollen ändern sich, während Sessions offen sind, und eine lang laufende Session muss Berechtigungen neu laden können, ohne eine neue Anmeldung zu erzwingen. Die Review-Frage für jede Servicemethode ist einfach: Welche Zeile beweist, dass der Aufrufer hier sein darf?

Audit-Trails für sensible Commands

Ein Audit-Trail beantwortet, wer was an welchem Datensatz in welchem Mandanten wann und mit welchem Ergebnis getan hat. Er wird von dem Service geschrieben, der den Command ausführt, und zwar innerhalb derselben Operation, sodass eine Aktion nicht gelingen kann, ohne aufgezeichnet zu werden. Der Bildschirm für das Audit-Log im Lab zeigt jedes `Demand`, ob gewährt oder verweigert, und jeden sensiblen Command wie eine Freigabe, einen Export oder eine Änderung von Berechtigungen.

Ablehnungen sind genauso wichtig wie Erfolge. Eine fälschlich aktivierte Schaltfläche, über die ein Benutzer **ExportData** anfordern kann, wird vom Service abgefangen: Er lehnt mit einem benannten Fehler ab und schreibt einen Eintrag „verweigert“. Wiederholte Ablehnungen für ein Konto sind ein Signal, dass jemand die Anwendung abtastet – und sie sind nur sichtbar, wenn Ablehnungen protokolliert werden. Der Eintrag sollte die Korrelations-ID enthalten, damit er mit den strukturierten Logs aus Modul 11 verknüpft ist.

- Akteur, Mandant und gegebenenfalls ein Kennzeichen für Identitätswechsel (Impersonation).
- Name des Commands, Zieldatensatz und die relevanten Werte vorher und nachher.
- Ergebnis: gewährt, verweigert oder fehlgeschlagen, mit dem benannten Grund.
- Zeitstempel und Korrelations-ID, geschrieben in derselben Transaktion wie die Änderung.

Eine Sicherheits-Checkliste für die gesamte Angriffsfläche

Serverseitiges Rendering beseitigt viele Risiken im Browser, doch eine Wisej.NET-Anwendung liefert trotzdem HTML aus, nimmt Uploads an, setzt Cookies, schreibt Logs, liest Secrets und steht hinter einem Proxy. An jeder dieser Stellen kann aus einem kleinen Fehler ein Sicherheitsvorfall werden. Die Checkliste zur Sicherheitshärtung im Lab ist das Dokument, das diese Stellen in Review-Punkte verwandelt, die bei jedem Release geprüft werden, statt nur gelegentlich daran zu denken.

Beginnen Sie mit HTML. Jedes Control, das rohes HTML aus Benutzereingaben rendert, muss es bereinigen, und das **AllowHtml** -Review im Lab listet jedes solche Control auf. Uploads brauchen Größenbeschränkungen, Typprüfungen und einen Speicherort außerhalb des Web-Roots. Cookies sollten als Secure und HTTP-only markiert sein und eine SameSite-Richtlinie haben. Eine Content Security Policy begrenzt, welche Skripte laufen dürfen. Logs dürfen niemals Passwörter, Tokens oder vollständige personenbezogene Datensätze enthalten. Secrets liegen in einem Vault oder in geschützter Konfiguration, niemals in der Versionsverwaltung. Ein Reverse Proxy muss das ursprüngliche Schema und die Clientadresse weiterreichen, damit die Anwendung die echten Werte sieht.

Behandeln Sie die Checkliste wie lebenden Code: Ergänzen Sie einen Punkt, sobald ein Review eine Lücke findet, und halten Sie für jedes Release fest, wer welchen Punkt geprüft hat.



Expertenregel für den Produktivbetrieb

Hören Sie nicht auf, sobald der Bildschirm korrekt aussieht. Erklären Sie, wem der Zustand gehört, welche Sicherheitsfolgen es gibt, welche Fehlerpfade existieren und wie sich das System im Produktivbetrieb verhält, bevor Sie das Modul als abgeschlossen betrachten.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ ein Identitäts- und Autorisierungsmodell für Enterprise-Rollen, Mandanten und Workflows entwerfen.
- ✓ Authentifizierungs-Gates mit dem Session-Kontext und den Anwendungsservices verbinden.
- ✓ Audit-Trails für sensible Commands aufbauen.

- ✓ Sicherheits-Checklisten für HTML, Uploads, Cookies, CSP, Logs, Secrets und Reverse Proxys erstellen.

Lab- und Prüfungsleitfaden

Was Sie im Praxis-Lab bauen, das vorgeschlagene Vorgehen und die geforderten Ergebnisse.



Praxis-Lab

Ergänzen Sie eine Simulation der Enterprise-Authentifizierung, das Mapping von Claims auf Berechtigungen, die Durchsetzung von Mandantenrollen, Autorisierung auf Service-Ebene, ein Review für sicheres HTML, eine Freigabe für Exporte und Audit-Logging für sensible Aktionen.

Vorgeschlagenes Vorgehen

- ✓ Lesen Sie das Problem im Produktivbetrieb und benennen Sie das Architekturrisiko, bevor Sie die UI anfassen.
- ✓ Legen Sie zuerst die betroffene Grenze für Service, Command, Modell oder Komponente an oder aktualisieren Sie sie.
- ✓ Erstellen Sie den Wisej.NET-Bildschirm oder die Komponente nach den etablierten Projektstandards.
- ✓ Demonstrieren Sie auch einen Fehlerpfad, nicht nur den Happy Path.
- ✓ Halten Sie die Änderungen im Lernprotokoll fest und aktualisieren Sie die zugehörige Checkliste.

Geforderte Ergebnisse

- ✓ Design des Identitäts-Mappings
- ✓ Berechtigungsmatrix
- ✓ Implementierung des Berechtigungsservices
- ✓ Bildschirm für das Audit-Log
- ✓ Checkliste zur Sicherheitshärtung

Wiederholungsfragen

- ✓ Wo wird die Autorisierung durchgesetzt?
- ✓ Welche UI-Elemente rendern HTML?
- ✓ Kann eine aktivierte Schaltfläche die Regeln des Service umgehen?



Expertenregel für den Produktivbetrieb

Hören Sie nicht auf, sobald der Bildschirm korrekt aussieht. Erklären Sie, wem der Zustand gehört, welche Sicherheitsfolgen es gibt, welche Fehlerpfade existieren und wie sich das System im Produktivbetrieb verhält, bevor Sie das Modul als abgeschlossen betrachten.