PATTERN DI SICUREZZA E AUTENTICAZIONE · MODULO 7

Progettazione per la produzione e capstone

Guida della lezione · Guida al lab / esame

LearnWisej

Guida della lezione

Due punti di ingresso con un'unica policy sulle risorse, il comportamento in scala e in caso di guasto tra più nodi, timeout e responsabilità della revoca, prove degli incidenti correlate, e un rilascio e un rollback provati per il capstone di AccessOps. Che cosa significa per uno sviluppatore che mette in sicurezza i percorsi di accesso, identità e autorizzazione di un'app Wisej.NET di produzione — e come affrontare questo modulo.

Focus di produzione

Due punti di ingresso con un'unica policy sulle risorse, il comportamento in scala e in caso di guasto tra più nodi, timeout e responsabilità della revoca, prove degli incidenti correlate, e un rilascio e un rollback provati per il capstone di AccessOps.

1. Il problema in produzione

Ogni modulo precedente ha messo in sicurezza un percorso. In produzione AccessOps serve contemporaneamente il personale attraverso una route Windows intranet e i partner attraverso una route HTTPS pubblica, su più nodi, accanto a una directory, a un identity provider e a un archivio delle credenziali, ciascuno dei quali si guasta con i propri tempi. Un rilascio è sicuro solo quando entrambe le route arrivano alla stessa decisione lato server, quando né la perdita di un nodo né una scheda obsoleta possono trasformarsi in un accesso, e quando il team è in grado di dimostrarlo in una revisione.

2. Concetti chiave

Concetto	Perché è importante
Policy condivisa sulle risorse	Un unico controllo al confine del servizio su grant, tenant e record che entrambe le route chiamano, così che un partner e un membro del personale vengano rifiutati dallo stesso codice prima che venga eseguito qualsiasi repository.
Matrice delle route	Una tabella per ogni punto di ingresso con target framework, modalità IIS, challenge, principal dell'host, principal dell'app, identità di esecuzione e responsabile della revoca; un revisore la controlla riga per riga.
Affinità di sessione	Wisej.NET mantiene ogni sessione nella memoria di un nodo, quindi ogni richiesta di una sessione deve raggiungere lo stesso nodo; perdere quel nodo significa perdere la sessione.
Autorità condivisa e persistente	Grant, revoche, handle di passaggio dell'accesso, challenge e credenziali delle passkey vivono nell'archivio dell'applicazione, non nello stato di sessione, così che un riavvio o un failover non possano dimenticare una decisione.
Chiavi condivise di protezione dei cookie	Le chiavi di ASP.NET Core Data Protection e il nome dell'applicazione devono coincidere su ogni nodo che convalida i cookie di autenticazione o di correlazione di un altro nodo.
Primo confine che fallisce	Il primo componente che ha emesso un rifiuto (ingress, policy, token del sistema operativo, route di callback o convalida della passkey), da cui parte un'indagine.
Condizione di arresto	Un risultato concordato prima del rollout, come una lettura tra tenant diversi, che blocca la promozione e fa scattare il rollback provato.

3. Il filo della lezione

Due route, un'unica policy

La route intranet gira su IIS con l'accesso anonimo disabilitato e l'autenticazione Windows abilitata. Il browser negozia sul nome host canonico e Wisej.NET espone quel principal dell'host, la cui identità è una `WindowsIdentity`, come `Application.User`. AccessOps associa i gruppi della directory a un grant nel proprio archivio e stabilisce un principal applicativo che porta i claim di account, tenant e grant, come in *Confini di identità in Wisej.NET*. Il token Windows effettivo è un terzo fatto: esiste solo dentro una chiamata circoscritta a `WindowsIdentity.RunImpersonated` per l'unica condivisione di file che richiede i diritti propri dell'utente (*Impersonificazione e delega di Windows*), e altrove il codice viene eseguito con l'identità del pool.

La route dei partner è HTTPS pubblica. Un partner accede con l'accesso locale di *Accesso personalizzato e sicurezza della sessione* oppure tramite il provider, con gli handler cookie e OpenID Connect di ASP.NET Core registrati prima di `app.UseWisej()` (*Federazione con OpenID Connect*); un account verificato può poi registrare una passkey attraverso l'estensione WebAuthn (*Passkey con l'estensione WebAuthn*). Entrambi i percorsi finiscono allo stesso modo: al caricamento di una pagina, `Application.RefreshSessionId()`

ruota l'ID di sessione e viene impostato il principal dell'app. Da quel momento i servizi non sanno più distinguere le route, e ogni lettura protetta passa per un unico controllo.

```
public Record Read(string tenantId, int recordId)
{
    var principal = Application.User as ClaimsPrincipal;
    var grant = _grants.Find(principal?.FindFirst("grant")?.Value); // archivio persistente
    if (grant == null || grant.Revoked || grant.ExpiresAt ≤ _clock.UtcNow
        || !grant.CanRead || grant.TenantId ≠ tenantId)
    {
        _audit.Deny("policy", grant?.Id, tenantId, recordId);
        return null;                                  // il repository non viene mai eseguito
    }
    return _records.Get(tenantId, recordId);
}
```

Il diagramma dei confini di fiducia indica dove nasce ogni identità, che cosa stabilisce la fiducia, quanto dura e chi la revoca: la directory gestisce l'appartenenza ai gruppi, il provider gestisce l'account esterno del partner, l'archivio delle credenziali gestisce le passkey e AccessOps gestisce il grant. Disabilitare un partner presso il provider non chiude un grant già emesso, quindi la console di amministrazione revoca anche il grant.

Comportamento in scala e in caso di guasto

Fissa le versioni che hai provato: `Wisej-4` e `Wisej-4-WebAuthn` 4.1.7, l'handler OpenID Connect allineato al runtime e il runtime stesso. Dietro il load balancer chiama `app.UseForwardedHeaders()` fidandoti solo dei proxy noti, perché lo schema e l'host canonici alimentano il redirect URI OIDC e il controllo dell'origine di WebAuthn. Wisej.NET richiede l'affinità di sessione — un cookie sticky, un hash dell'IP del client o un altro instradamento coerente — e alcuni bilanciatori richiedono la modalità TCP per il WebSocket. Pubblica con `"secure": true` in Default.json e tieni i file `.json` fuori dal server di file statici.

Un nodo perso perde le proprie sessioni, quindi nulla che debba sopravvivere può vivere in `Application.Session`. Condividi tra i nodi il key ring di Data Protection e il nome dell'applicazione, così che un cookie di correlazione o di autenticazione scritto dal nodo A venga convalidato sul nodo B. Un callback del provider che arriva sul nodo B ha due esiti sicuri: l'handler lo convalida e l'handle di passaggio monouso nell'archivio condiviso riprende l'accesso in una nuova sessione su quel nodo, oppure l'handle manca o è già stato consumato e l'utente riceve un semplice errore e un nuovo accesso. Non concede mai due volte né avvia un nuovo tentativo a partire dal codice in arrivo. I timeout appartengono alla stessa tabella: il `sessionTimeout` di Wisej, la scadenza assoluta del grant, l'`ExpireTimeSpan` del cookie dell'host, la transazione del provider e il timeout della cerimonia WebAuthn, ciascuno con un responsabile della revoca.

Una matrice degli incidenti basata sul primo confine che fallisce

Parti dal primo componente che ha detto no. Un 401 sull'ingress rimanda al challenge, alla terna di stati di IIS, a `setspn -Q` e a `klist`. Una richiesta autenticata seguita da un rifiuto di business rimanda a una traccia della policy: grant, tenant, record e un conteggio delle chiamate al repository pari a zero. Un errore su un file remoto dopo un successo in locale rimanda al token effettivo. Un callback che non raggiunge

mai l'app rimanda all'URL canonico, al redirect URI esatto e alle intestazioni inoltrate; uno contraffatto finisce in `OnRemoteFailure`. Una risposta della passkey rifiutata dopo il prompt rimanda a RP ID, origine, challenge, `ClientData.Type` e all'archivio.

Ogni riga emette una sola categoria di evento presente nella allowlist, con un ID di correlazione e un ID dell'account opaco; poiché lo scope di logging di ASP.NET Core vale per l'intera connessione WebSocket, AccessOps apre un proprio `ILogger.BeginScope` per ogni clic. ID di sessione, codici, token e cookie non raggiungono mai l'archivio di audit protetto, che ha un periodo di conservazione dichiarato.

Rilascio, condizioni di arresto e rollback

Prima di esporre la route dei partner, esegui la matrice negativa di 12 casi più un caso normale per ogni confine, poi fai il rollout su un tenant di test. Scrivi prima le condizioni di arresto:

- una lettura tra tenant diversi, o qualsiasi rifiuto che avviene dopo che il repository è stato eseguito;
- un callback o un'assertion riprodotti accettati come un nuovo accesso;
- una credenziale eliminata o un grant revocato che apre ancora un'azione protetta;
- materiale delle credenziali in un log, in una colonna nascosta della griglia o in `Application.Browser.UserData`.

Il rollback disattiva sull'ingress la route dei partner o la nuova funzionalità, mentre la base intranet già provata continua a funzionare. Grant e revoche vivono nell'archivio persistente, non nella build, quindi riportare indietro il codice non può riattivare un grant revocato; il runbook lo dimostra ripetendo in seguito il caso della scheda obsoleta. Il runbook indica il responsabile e l'azione esatta per la revoca d'emergenza, e un provider che fallisce riceve tentativi limitati e un errore semplice. I casi che richiedono IIS, il dominio o un provider attivo e che sono stati eseguiti su fixture o nel PolicyHarness vengono etichettati con un piano preciso per rieseguirli; la revisione orale difende poi le prove in dieci minuti.

Regola di sicurezza

Tieni separate le identità e decidi sul server. Il principal dell'host, il principal applicativo di Wisej.NET e il token Windows effettivo sono tre fatti diversi, e nessuno di essi è un permesso. Ogni lettura protetta controlla il grant corrente, il tenant e il record al confine del servizio prima che il repository venga eseguito, e un pulsante nascosto, un claim di ruolo o una scheda disconnessa non dimostrano nulla finché quel controllo non rifiuta.

Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Progettare la route Windows intranet e la route HTTPS dei partner con un'unica policy condivisa su tenant e risorse, e rappresentare in un diagramma dove ogni identità nasce, riceve fiducia, scade e viene revocata.
- ✓ Rendere espliciti la scala e i guasti: versioni fissate, intestazioni inoltrate, affinità del WebSocket, chiavi condivise di protezione dei cookie e un esito sicuro per un callback che arriva su un altro nodo.

- ✓ Costruire una matrice degli incidenti basata sul primo confine che fallisce — challenge sull'ingress, rifiuto della policy, token effettivo, route di callback, convalida della passkey — con eventi di audit oscurati e correlati.
- ✓ Eseguire la matrice negativa di 12 casi, definire le condizioni di arresto del rilascio e un rollback che mantiene revocati i grant revocati, e difendere il rilascio in una revisione orale.

Guida al lab / esame

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

Lab pratico

Completa il capstone di AccessOps sul portale cumulativo: disegna i confini di fiducia con i responsabili per IIS, la directory, l'identity provider, l'app e l'archivio delle credenziali, scrivi la matrice delle route (target framework, modalità IIS, challenge, principal dell'host, principal dell'app, identità di esecuzione, responsabile della revoca) ed esegui la matrice delle prove di 12 casi indicata nelle istruzioni — lettura intranet del Reader, rifiuto dopo l'autenticazione dell'host, il tenant A che richiede il record 17 del tenant B, lettura circoscritta con ripristino, secondo hop, rotazione della sessione, grant revocato da una scheda obsoleta, URL di ritorno esterno e post contraffatto, un solo grant per callback federato, tenant non approvato o state errato, riproduzione della passkey, proprietario errato o credenziale eliminata, e un callback dopo un riavvio o su un secondo nodo — registrando per ciascuno il confine che ha rifiutato e la prova che non sono seguiti accessi ai dati né grant. Simula un guasto, come uno SPN errato, la perdita di un nodo durante un callback o un autenticatore smarrito, esercitati con il runbook, e presenta le tracce di consenso e di rifiuto e il rollback in una revisione di dieci minuti rispetto alle checklist di revisione della sicurezza; i casi che richiedono IIS, il dominio o un provider attivo e che sono stati eseguiti su fixture o nel PolicyHarness vengono etichettati con un piano preciso per rieseguirli.

Punto di partenza

Apri il portale AccessOps cumulativo che hai costruito nei sei moduli precedenti, con la sua route Windows intranet, l'accesso dei partner, il passaggio OpenID Connect, l'archivio delle passkey e la policy condivisa sui record, più il PolicyHarness autonomo. Usa due tenant (A e B) che contengono entrambi un record 17, le fixture riservate F01–F12 e un secondo nodo oppure un esercizio registrato di riavvio e reinstradamento.

Approccio suggerito

- ✓ Disegna per primo il diagramma dei confini di fiducia: browser, IIS o proxy, sessione Wisej.NET, identity provider, archivio delle credenziali e backend dei record, con un responsabile, un'ancora di fiducia, una durata e un'azione di revoca su ogni identità che attraversa una linea.
- ✓ Compila una riga della matrice delle route per ogni punto di ingresso e fissa le versioni che hai provato; poi scrivi la configurazione ripulita (intestazioni inoltrate, affinità, chiavi condivise di Data Protection, `"secure": true`) e la mappa di ruoli e tenant.
- ✓ Esegui i 12 casi in ordine e compila ogni riga della matrice delle prove con l'alias del test, il punto di ingresso, l'esito atteso ed effettivo, il confine che ha rifiutato, l'etichetta di correlazione, il conteggio delle chiamate al repository e se il caso è stato eseguito dal vivo, su una fixture o nel PolicyHarness.
- ✓ Introduci un guasto (uno SPN errato, la perdita di un nodo durante un callback del provider o un autenticatore smarrito), segui la matrice degli incidenti a partire dal primo confine che fallisce e cronometra il ripristino nel runbook.
- ✓ Prova il rollback: disattiva la route dei partner, conferma che la base intranet serve ancora i Reader, ripeti il caso della scheda obsoleta per mostrare che il grant revocato resta revocato, poi presenta la revisione di dieci minuti.

Deliverable richiesti

- ✓ Diagramma dei confini di fiducia e matrice delle route
- ✓ Portale AccessOps cumulativo con configurazione ripulita, mappa di ruoli e tenant e test della policy
- ✓ Matrice delle prove di 12 casi con il confine che ha rifiutato e le etichette delle fixture
- ✓ Runbook di deployment e rollback con responsabili e condizioni di arresto
- ✓ Checklist di revisione della sicurezza completata e note della revisione orale valutate secondo la rubrica

Domande di autovalutazione

- ✓ Il callback di un partner raggiunge il nodo B dopo che il nodo A si è fermato a metà dell'accesso. Ripercorri che cosa può verificare il nodo B, che cosa non può verificare, e perché l'esito è comunque un solo grant o nessuno.
- ✓ Il tuo rollback ripristina la build e la configurazione precedenti. Quale stato non deve ripristinare, e quale riga delle prove dimostra che un grant revocato non è tornato?
- ✓ Il caso 5 è stato eseguito su una fixture perché il lab non ha un dominio. Che cosa rieseguiresti esattamente sulla topologia reale, e quale risultato lì bloccherebbe l'approvazione?

Regola di sicurezza

Tieni separate le identità e decidi sul server. Il principal dell'host, il principal applicativo di Wisej.NET e il token Windows effettivo sono tre fatti diversi, e nessuno di essi è un permesso. Ogni lettura protetta controlla il grant corrente, il tenant e il record al confine del servizio prima che il repository venga eseguito, e un pulsante nascosto, un claim di ruolo o una scheda disconnessa non dimostrano nulla finché quel controllo non rifiuta.