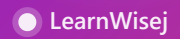


Lista de comprobación de bastionado y proyecto final

Guía de la lección · Guía del laboratorio / examen



Guía de la lección

HTTPS y cabeceras de seguridad, atributos de las cookies, ajustes de sesión, actualización de dependencias, limitación de frecuencia, bastionado del despliegue y la revisión de seguridad de AccessOps. Qué significa esto para un desarrollador que protege una aplicación Wisej.NET en producción, y cómo leer este módulo.



Enfoque en producción

HTTPS y cabeceras de seguridad, atributos de las cookies, ajustes de sesión, actualización de dependencias, limitación de frecuencia, bastionado del despliegue y la revisión de seguridad de AccessOps.

1. El problema en producción

AccessOps ya verifica las contraseñas contra un hash con sal, resuelve los permisos en un servicio, transmite los documentos solo a través de `DocumentService` y audita cada aprobación. Nada de eso sirve si la primera solicitud viaja en texto claro, si un script puede leer la cookie de sesión o si el build incluye un paquete con un aviso de seguridad publicado. El último tramo corresponde al host: el certificado, las cabeceras de respuesta, los atributos de las cookies, los límites de las solicitudes y la versión de todo lo que hay en el grafo de dependencias.

2. Conceptos clave

Concepto	Por qué importa
HSTS	Una cabecera <code>Strict-Transport-Security</code> que indica al navegador que rechace HTTP sin cifrar para este host durante un tiempo determinado, para que una visita posterior nunca pueda interceptarse en la redirección.
Content-Security-Policy	Una cabecera que enumera los orígenes desde los que una página puede cargar scripts, estilos y marcos; convierte un script inyectado en una solicitud bloqueada en lugar de código ejecutándose en una sesión activa.
frame-ancestors	La directiva CSP (<code>X-Frame-Options</code> es el equivalente antiguo) que impide que otro sitio incruste AccessOps en un marco y engañe a un responsable para que pulse Approve.
atributos de la cookie	<code>Secure</code> mantiene la cookie de sesión fuera de HTTP sin cifrar, <code>HttpOnly</code> la mantiene lejos de JavaScript y <code>SameSite</code> impide que viaje en una solicitud iniciada por otro sitio.
límites de las solicitudes	Topes a nivel de host en el tamaño del cuerpo y de las cabeceras que rechazan una subida sobredimensionada antes de que se ejecute una sola línea de código de AccessOps.
token bucket	Un contador que concede un intento solo mientras contiene un token y se rellena a un ritmo fijo, de modo que el tráfico puede tener un pequeño pico y después se ralentiza al mínimo.
auditoría de dependencias	Una comprobación repetible de cada paquete directo y transitivo, y del runtime, contra los avisos de seguridad publicados, registrada con una fecha y una decisión.

3. Desarrollo de la lección

HTTPS es tarea del host; HSTS, del navegador

TLS termina fuera de tu C#: en IIS con un certificado asociado, en Kestrel configurado con uno, o en el proxy inverso situado delante de ambos. Tu parte es hacer que HTTP sin cifrar no sirva para nada. Redirige cada solicitud HTTP en el host y envía después `Strict-Transport-Security` con un `max-age` largo para que el navegador deje de intentar por completo el esquema inseguro. Añade `includeSubDomains` solo cuando todos los subdominios de AccessOps sirvan realmente TLS, y trata `preload` como una decisión que no puedes deshacer rápidamente. Un detalle del despliegue pill a más de uno: detrás de un proxy, la solicitud llega a la aplicación sin cifrar, así que, salvo que esté habilitada la gestión de las cabeceras reenviadas, la aplicación cree que es insegura y no marca nada como `Secure`.

Las cabeceras son los controles más baratos de este curso, y se establecen una vez para todo el pipeline y no pantalla por pantalla. `X-Content-Type-Options: nosniff` impide que un navegador suponga que un documento subido es en realidad HTML. `Referrer-Policy: strict-origin-when-cross-origin` impide que un identificador de documento se filtre a los logs de otro sitio. `frame-ancestors 'none'` rechaza sin más la inclusión en marcos. Establécelas donde pasen todas las respuestas, incluido el handler de enlaces compartidos firmados del Módulo 5: una URL aparte, y la que la gente olvida.

Content-Security-Policy requiere cuidado aquí, porque el navegador está ejecutando el cliente de Wisej.NET. Ese cliente, los recursos de su tema y los scripts de tus widgets vienen del mismo origen, así

que `default-src 'self'` los cubre; el motor de diseño escribe estilos en línea, así que `style-src` suele necesitar `'unsafe-inline'`; y la interoperabilidad que construye código de cliente en tiempo de ejecución te obliga a ampliar `script-src`. Despliega primero como `Content-Security-Policy-Report-Only`, recorre todas las pantallas, lee las infracciones y después aplícala.

```
app.UseForwardedHeaders();           // the proxy terminated TLS, not us
app.UseHttpsRedirection();
app.UseHsts();                       // Strict-Transport-Security
app.Use(async (ctx, next) =>
{
    var h = ctx.Response.Headers;
    h["X-Content-Type-Options"] = "nosniff";
    h["Referrer-Policy"] = "strict-origin-when-cross-origin";
    h["Content-Security-Policy-Report-Only"] = AccessOpsCsp;
    await next();
});
// the Wisej.NET handler runs after this, so every response carries the headers
```

Cookies, límites de sesión y un botón que no es un endpoint

Una vez que el inicio de sesión tiene éxito, la cookie de sesión es la credencial, así que recibe los tres atributos. `Secure` significa que nunca viaja sin TLS. `HttpOnly` significa que, aunque una inyección de script tenga éxito en un widget personalizado, no podrá leerla. `SameSite` con valor `Lax` o `Strict` significa que una solicitud iniciada por otro sitio no la lleva: la defensa más barata contra una aprobación falsificada. En ASP.NET Framework son ajustes de los elementos de cookie y de estado de sesión de `Web.config`; en ASP.NET Core son opciones de la política de cookies. Verifícalos en las herramientas de desarrollo del navegador, no releendo la configuración.

La duración de la sesión tiene dos mitades. Wisej.NET lee sus propios ajustes del archivo de configuración JSON de la aplicación, junto a la página de inicio, y ahí es donde va la expiración de la sesión del framework: el último recurso que recupera una sesión abandonada y su estado del lado del servidor. No es el cierre por inactividad que construiste en el Módulo 2, porque el cliente mantiene viva su conexión y un navegador abandonado puede parecer tráfico: la inactividad debe medirse desde el último evento real del usuario, y el cierre de sesión explícito debe liberar el `SessionContext` de `Application.Session` para que el botón Atrás lleve a una página de inicio de sesión. Los límites de las solicitudes son el tercer ajuste: limita en el host el tamaño del cuerpo y de las cabeceras para que una subida sobredimensionada se rechace antes de entrar en `DocumentService`.

La limitación de frecuencia se divide por una línea importante. El handler de enlaces compartidos es una URL real, así que un middleware puede limitarlo directamente: un limitador token bucket particionado por dirección del cliente que responde 429 sin revelar si el enlace existía. El botón de inicio de sesión no es una URL: todos los eventos de Wisej.NET, desde el desplazamiento de una cuadrícula hasta un clic en Sign in, llegan al mismo endpoint de la aplicación dentro de una sesión existente, así que un middleware no puede distinguirlos. Por eso la limitación del inicio de sesión vive en el servicio, como un bucket indexado por el nombre de usuario enviado y la dirección del cliente y comprobado antes de verificar el hash. Un rechazo devuelve el mismo mensaje genérico que una contraseña incorrecta, se audita con su ID de correlación y,

cuando AccessOps se ejecuta en varios servidores, se contabiliza en un almacén compartido y no en un campo estático.

Parcheado, y demostrablemente

Ejecuta `dotnet list package --vulnerable --include-transitive` y léelo entero: el hallazgo habitual es el paquete que nunca elegiste, arrastrado tres niveles más abajo. Añade `--outdated` y `--deprecated` para lo que simplemente está desactualizado o abandonado. Los SDK recientes de .NET también comprueban los avisos durante la restauración, lo que convierte la auditoría en salida del build y no en una tarea que alguien tiene que recordar. Ponla en el pipeline para que un aviso nuevo haga fallar un build en lugar de aparecer en una prueba de penetración.

Los paquetes son solo una parte del grafo. El runtime de .NET, el servidor web y la imagen base del contenedor necesitan cada uno una cadencia de parches y una versión registrada. El propio Wisej.NET es una dependencia versionada: fíjalo, lee las notas de la versión antes de cambiar y vuelve a probar el tema y los widgets personalizados tras una actualización mayor. El JavaScript de terceros en la raíz web es invisible para el grafo de paquetes, así que enumera cada archivo a mano con su versión y su origen. Después deja la auditoría por escrito: fecha, herramienta, hallazgos, una decisión para cada uno (actualizar, mitigar o aceptar con caducidad) y la fecha de la próxima revisión.

La revisión previa a la publicación, módulo a módulo

El entregable del proyecto final es una revisión, no una funcionalidad. Toma el mapa de ubicación de controles del Módulo 1 y recorre el curso hacia delante, comprobando la aplicación y no la intención, y prueba como un atacante y no como un usuario: pide un documento de otro departamento como empleado normal, reutiliza un enlace compartido pasada su hora, llama a un método de aprobación como alguien que no es el responsable, envía veinte contraseñas erróneas en diez segundos.

- Módulo 2: bloqueo de cuenta, hash, identidad verificada, expiración por inactividad, cierre de sesión que libera el estado.
- Módulo 3: cada método de servicio protegido, reglas de departamento y de responsable por debajo de la interfaz, la solicitud falsificada rechazada.
- Módulo 4: PKCE, validación del token de ID y del nonce, asignación de claims, cierre de sesión que termina ambas sesiones.
- Módulo 5: validación de las subidas, descargas a través del servicio, enlaces firmados con caducidad, consultas parametrizadas, salida codificada.
- Módulo 6: filas de auditoría para los comandos sensibles, ID de correlación en los logs, ni secretos ni datos personales en ninguno de los dos.
- Módulo 7: HTTPS y HSTS, el conjunto de cabeceras, atributos de las cookies, límites de sesión y de solicitudes, limitación de frecuencia, una auditoría fechada.

Cada hallazgo recibe una gravedad, un responsable y una corrección o un riesgo aceptado con caducidad; una lista de comprobación con huecos sin explicar no está completa. La aprobación final significa una sola cosa: cada límite dibujado en el Módulo 1 tiene ahora un control que hace cumplir un servicio, no una pantalla.



Regla de seguridad

Nunca se confía en el navegador. Toda decisión importante —quién es el usuario, qué puede hacer, qué registro puede ver— se toma en el servidor, dentro de un servicio, y la interfaz solo la refleja. Si una regla solo existe como un botón oculto, no existe.



Objetivos de aprendizaje

Al terminar este módulo serás capaz de:

- ✓ Configurar HTTPS, HSTS, atributos seguros de las cookies y cabeceras de seguridad para un despliegue de Wisej.NET.
- ✓ Ajustar la configuración de sesión, los límites de las solicitudes y la limitación de frecuencia para resistir los abusos.
- ✓ Mantener parcheados las dependencias y el runtime con un proceso repetible.
- ✓ Realizar una revisión de seguridad previa a la publicación con una lista de comprobación y entregar el proyecto final de AccessOps.

Guía del laboratorio / examen

Qué construirás en el laboratorio práctico, el enfoque sugerido y los entregables requeridos.



Laboratorio práctico

Completa el proyecto final de AccessOps: impón HTTPS con HSTS y configura las cabeceras de seguridad, marca las cookies como Secure y HttpOnly con una política SameSite estricta, fija el tiempo de expiración de la sesión y los límites de tamaño de las solicitudes, añade limitación de frecuencia al inicio de sesión y a los enlaces compartidos, ejecuta una auditoría de dependencias y registra los resultados, y después recorre la lista de comprobación completa de este curso contra la aplicación y corrige cada hallazgo.

Punto de partida

Abre la solución AccessOps tal como está tras el Módulo 6: una página de inicio de sesión con bloqueo de cuenta, un `SessionContext` con la identidad verificada, un `PermissionService` que protege `DocumentService` y `ApprovalService`, inicio de sesión externo mediante OpenID Connect, un canal de subida y descarga con un handler de enlaces compartidos firmados y un `AuditService` que escribe filas marcadas con el ID de correlación. Todavía no se ha tocado nada de la configuración del host: la aplicación responde por HTTP sin cifrar en desarrollo, sus cookies no llevan atributos y ningún paquete se ha comprobado contra los avisos de seguridad.

Enfoque sugerido

- ✓ Asocia un certificado y redirige HTTP a HTTPS en el host, añade `Strict-Transport-Security` y confirma la gestión de las cabeceras reenviadas si hay un proxy inverso delante de AccessOps.
- ✓ Añade un único punto en el pipeline que establezca `X-Content-Type-Options`, `Referrer-Policy`, `frame-ancestors` y una Content-Security-Policy, empezando en modo report-only hasta que todas las pantallas de AccessOps y el handler de enlaces compartidos carguen sin infracciones.
- ✓ Marca la cookie de sesión como `Secure` y `HttpOnly` con una política `SameSite` estricta, fija la expiración de la sesión de Wisej.NET en el archivo de configuración de la aplicación y limita en el host el tamaño del cuerpo de las solicitudes.
- ✓ Pon un token bucket indexado por nombre de usuario y dirección del cliente dentro del servicio de inicio de sesión, y un limitador de middleware en la URL de los enlaces compartidos, auditando cada rechazo con su ID de correlación.
- ✓ Ejecuta `dotnet list package --vulnerable --include-transitive`, registra los hallazgos con una decisión y un responsable, y después recorre la lista de comprobación de los siete módulos contra la aplicación en ejecución y corrige lo que encuentre.

Entregables requeridos

- ✓ Configuración de HTTPS, HSTS y cabeceras de seguridad
- ✓ Ajustes de bastionado de cookies y sesión
- ✓ Limitación de frecuencia en el inicio de sesión y los enlaces compartidos

- ✓ Informe de auditoría de dependencias
- ✓ Lista de comprobación de la revisión de seguridad completada con las correcciones

Preguntas de repaso

- ✓ Un middleware puede limitar la frecuencia del handler de enlaces compartidos, pero no la del botón Sign in. ¿Qué diferencia hay entre ambas solicitudes en una aplicación Wisej.NET, y dónde tiene que vivir entonces el límite del inicio de sesión?
- ✓ Tu Content-Security-Policy está en modo obligatorio y la cuadrícula se muestra sin estilos. ¿Qué directiva es la causa probable, y cómo te lo habría advertido el modo report-only antes de que lo hiciera el despliegue?
- ✓ La auditoría de dependencias muestra un paquete vulnerable al que nadie en AccessOps hace referencia directa. ¿Cómo llegó al build, y cuáles son las dos formas honestas de cerrar el hallazgo?



Regla de seguridad

Nunca se confía en el navegador. Toda decisión importante —quién es el usuario, qué puede hacer, qué registro puede ver— se toma en el servidor, dentro de un servicio, y la interfaz solo la refleja. Si una regla solo existe como un botón oculto, no existe.