

# Protección de datos, subidas, descargas y handlers

Guía de la lección · Guía del laboratorio / examen

● LearnWisej

## Guía de la lección

Autorizar subidas y descargas de archivos, handlers HTTP personalizados seguros, validación de la entrada, codificación de la salida, e inyección y riesgos entre sitios en una interfaz de usuario del lado del servidor. Qué significa esto para un desarrollador que protege una aplicación Wisej.NET en producción, y cómo leer este módulo.



### Enfoque en producción

Autorizar subidas y descargas de archivos, handlers HTTP personalizados seguros, validación de la entrada, codificación de la salida, e inyección y riesgos entre sitios en una interfaz de usuario del lado del servidor.

## 1. El problema en producción

El área de documentos confidenciales es la parte de AccessOps que de verdad quiere un atacante: cartas salariales, expedientes disciplinarios, contratos firmados, documentación del consejo. Todas las demás pantallas mueven filas entre cuadrículas; esta mueve bytes a través del límite hacia un navegador que no controlas, y de vuelta. La identidad y los permisos quedaron resueltos en los módulos anteriores, y este trata de los lugares por donde un archivo, una URL o una cadena los esquivan.

## 2. Conceptos clave

| Concepto                               | Por qué importa                                                                                                                                                                                                                 |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| descarga transmitida por el servidor   | Llamar a <code>Application.Download</code> desde código del servidor transmite un archivo al navegador durante un evento que ya has autorizado, así que no tiene una dirección que guardar en marcadores, compartir o adivinar. |
| exposición de archivos estáticos       | Todo lo que está bajo la raíz web se sirve antes de que se ejecute una sola línea de tu código, así que los permisos del archivo son exactamente «quien conozca la URL».                                                        |
| evento Uploaded                        | El control <code>Upload</code> lo lanza en el servidor con los flujos de los archivos entrantes: el único lugar donde se puede inspeccionar un archivo antes de que entre en tu sistema.                                        |
| comprobación de la firma del contenido | Comparar los primeros bytes de un archivo con la cabecera conocida del formato, porque la extensión y el tipo de contenido declarado los proporciona el cliente.                                                                |
| almacenamiento en cuarentena           | Guardar la subida aceptada fuera de la raíz web con un nombre generado hasta que el análisis salga limpio, y solo entonces vincularla al departamento propietario.                                                              |
| handler HTTP personalizado             | Una clase del servidor registrada en la aplicación que recibe solicitudes en bruto fuera del flujo de eventos, así que no tiene identidad de sesión y debe autenticar y validar todo por sí misma.                              |
| enlace firmado con caducidad           | Una URL que lleva un id de documento, una caducidad y un HMAC calculado sobre ambos con una clave guardada en el servidor, para que los valores no puedan editarse y el enlace deje de funcionar por sí solo.                   |

## 3. Desarrollo de la lección

### Una descarga sin dirección

Lo más seguro de una descarga en Wisej.NET es que no hay nada a lo que enlazar. El código del servidor abre un flujo, llama a `Application.Download` con ese flujo y el nombre de archivo con el que el navegador debe guardarlo, y los bytes viajan por la conexión existente como un adjunto. La transferencia ocurre dentro de una sesión ya autenticada, en un controlador de eventos que ya ha ejecutado tus comprobaciones: ninguna ruta en el marcado, ninguna URL en el historial, nada que reenviar.

Esa propiedad es fácil de echar a perder. Poner `/files/contract-4471.pdf` en un enlace entrega la función al servidor web, que lo sirve a cualquiera, y los nombres secuenciales invitan al siguiente a probar el 4472. En AccessOps los archivos viven fuera de la carpeta publicada con nombres que genera la aplicación, y la única entrada es `DocumentService.Download`.

El orden dentro de ese método de servicio importa. Carga el documento por id mediante una consulta parametrizada, tratando el id como entrada no fiable porque procede de una fila de cuadrícula en un navegador. Pregunta a las reglas de permisos del Módulo 3 —incluidas la pertenencia al departamento y el ámbito del responsable— si este llamante puede leer este registro, y lanza una excepción si no. Escribe la entrada de auditoría. Solo entonces abre el archivo almacenado y transmítelo. El controlador del botón

solo llama al servicio, así que la misma regla se cumple desde la cuadrícula, desde una pantalla de aprobación o desde un informe programado.

```
public void Download(int documentId)
{
    var doc = _repository.FindById(documentId);           // parameterised
    if (doc == null || !_permissions.CanRead(_session.User, doc))
        throw new SecurityException("Document not available.");

    _audit.RecordDownload(_session.User.Id, doc.Id);

    using var file = _storage.OpenRead(doc.StoredName); // outside the web root
    Application.Download(file, doc.DisplayName);
}
```

### Una subida es una entrada no fiable que resulta ser un archivo

El control `Upload` envía los archivos seleccionados al servidor y lanza `Uploaded`, donde tu controlador recibe los flujos de los archivos. Todo lo que el evento te dice de ellos procede del navegador: el nombre, la extensión, el tipo de contenido declarado, el tamaño. Un cliente hostil puede enviar un ejecutable llamado `payslip.pdf`, o un nombre lleno de segmentos de directorio padre que se sale de la carpeta que tenías prevista.

Así que el controlador es un filtro con cinco comprobaciones, por orden. Aplica un tamaño máximo en el control y en los límites de solicitud del servidor. Comprueba la extensión con una lista de permitidas de lo que la función necesita, nunca con una lista de prohibidas de lo que temes. Lee los primeros bytes y confirma que coinciden con la firma de ese formato, para que un ejecutable renombrado se rechace aunque su extensión pareciera correcta. Escribe el archivo fuera de la raíz web con un nombre que genere tu código, conservando el nombre original del usuario como valor para mostrar. Por último, ponlo en cuarentena, ejecuta el análisis antimalware y añádelo a los documentos del departamento solo cuando el análisis salga limpio.

- Tamaño y extensión rechazados antes de leer el flujo hasta el final.
- Firma comprobada con los formatos permitidos, no con el tipo declarado.
- Nombre almacenado generado por el servidor; el nombre del cliente nunca es una ruta.

Cada rechazo es un mensaje con el que el usuario puede hacer algo —«documentos PDF y Word de hasta 20 MB»— más una entrada de auditoría que nombra al llamante, el archivo y el motivo. El silencio es peor que el rechazo.

### Los handlers están fuera de la sesión

A veces un enlace tiene que funcionar sin el portal: un proveedor necesita el contrato firmado, un responsable reenvía un documento a un auditor. Eso es un handler HTTP personalizado, una clase del servidor registrada en la aplicación que recibe la solicitud en bruto. No forma parte de una sesión de Wisej.NET, así que no se ha establecido nada sobre el llamante. Responda lo que responda el handler, se lo responderá a todo internet.

AccessOps emite un enlace compartido como un id de documento, una caducidad y una firma: un HMAC sobre el id y la caducidad con una clave guardada solo en el servidor. El handler interpreta el id como un entero, vuelve a calcular el HMAC a partir de los valores de la query string, compara ambos en tiempo fijo, comprueba la caducidad y solo entonces carga el registro y transmite el archivo. Editar el id o alargar la caducidad cambia la firma, así que la solicitud se rechaza. Los enlaces caducados y los falsificados reciben la misma respuesta breve, y ambos se registran con la dirección del llamante para que un barrido en busca de ids válidos se haga visible.

El resto de las reglas del handler se derivan del mismo hecho: valida cada entrada en lugar de confiar en un llamante que no puedes ver, no devuelvas detalles en la respuesta de error, mantén la vida corta —una hora en el laboratorio— y limita la frecuencia del endpoint para que los documentos no puedan enumerarse con calma.

### Inyección en una interfaz de usuario del lado del servidor

Los controles del lado del servidor eliminan buena parte del riesgo web habitual, porque tu C# establece propiedades en objetos en lugar de concatenar marcado. No eliminan la inyección, que sigue a los datos. Todas las consultas de AccessOps usan parámetros —EF Core parametriza LINQ automáticamente y el SQL en bruto recibe parámetros explícitos—, así que un cuadro de búsqueda que contiene una comilla y un punto y coma es un término de búsqueda, no una sentencia. Concatenar en SQL es el único atajo que este curso nunca permite.

La segunda vía es la salida. La mayoría de los controles de Wisej.NET renderizan un texto como texto, pero una propiedad a la que se permite llevar marcado lo renderizará, así que una etiqueta con `AllowHtml` activado y un nombre aportado por un empleado se convierte en un anfitrión de scripts. Mantén esas propiedades desactivadas para el contenido del usuario, o codifica el valor antes de asignarlo; los nombres de archivo que muestras tras una subida también vinieron del navegador.

La tercera vía es la interoperabilidad con JavaScript. `Application.Eval` y `Application.Call` envían script al cliente, y construir ese script pegando una cadena del usuario es el mismo error que construir SQL a mano. Pasa los valores como argumentos a una función que ya existe en la página y deja que el código del lado del cliente los trate como datos.



#### 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:

- ✓ Autorizar cada descarga a través de un servicio y transmitir los archivos con `Application.Download` en lugar de exponer rutas o URL.
- ✓ Validar las subidas por tamaño, tipo y contenido, almacenarlas fuera de la raíz web y analizarlas o ponerlas en cuarentena.

- ✓ Escribir handlers HTTP y endpoints personalizados que autenticuen al llamante y validen cada entrada.
- ✓ Prevenir los riesgos de inyección y de scripts: consultas parametrizadas, salida codificada, HTML seguro en las etiquetas e interoperabilidad segura con JavaScript.

## Guía del laboratorio / examen

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



### Laboratorio práctico

Protege la función de documentos de AccessOps: subidas limitadas por tamaño y tipo con análisis del contenido, almacenadas fuera de la raíz web con un nombre aleatorio y vinculadas al departamento propietario; descargas que pasan por la autorización de DocumentService y por Application.Download; un handler personalizado para un enlace compartido firmado que caduca en una hora; y una revisión de que todas las consultas están parametrizadas y de que toda cadena proporcionada por el usuario se codifica antes de mostrarse.

## Punto de partida

Abres la solución AccessOps tal como quedó al final del Módulo 4: un inicio de sesión funcional con un proveedor de identidades externo, un SessionContext con la identidad verificada y un PermissionService que resuelve roles, claims y las reglas de ámbito por departamento y por responsable. La pantalla Documents ya enumera los documentos confidenciales del departamento actual y ya tiene un control Upload y un botón Download, y ambos hacen ahora lo ingenuo: escribir archivos en una carpeta bajo la raíz web y devolver el archivo cuyo nombre tenga la fila de la cuadrícula.

## Enfoque sugerido

- ✓ Sacar el almacén de documentos de la carpeta publicada a una ruta configurada fuera de la raíz web, dar a cada archivo almacenado un nombre generado por el servidor y conservar el nombre de archivo original del usuario en la base de datos solo como valor para mostrar.
- ✓ Reescribe el controlador Uploaded como un filtro: aplica el límite de tamaño, comprueba la extensión con una lista de permitidas, lee los primeros bytes y compáralos con la firma del formato; después escribe el archivo aceptado en la carpeta de cuarentena y vincúlalo al departamento propietario solo cuando el análisis sea correcto.
- ✓ Pon todas las descargas detrás de DocumentService: carga el registro por id con una consulta parametrizada, llama a la comprobación de permisos para el llamante y ese registro, escribe una entrada de auditoría y transmite después el archivo con Application.Download, dejando en el controlador del botón nada más que la llamada al servicio.
- ✓ Añade el handler de enlaces compartidos: genera una URL que lleve el id del documento, una caducidad a una hora vista y un HMAC sobre ambos; después haz que el handler vuelva a calcular la firma con una comparación en tiempo fijo, rechace los enlaces caducados o manipulados con la misma respuesta breve y registre cada rechazo.
- ✓ Revisa la función en busca de inyecciones: confirma que todas las consultas usan parámetros, confirma que ninguna cadena aportada por el usuario llega sin codificar a una propiedad que admita HTML ni a una llamada a Application.Eval, y termina con el intento de descarga falsificada desde una cuenta de otro departamento.

## Entregables requeridos

- ✓ Canal de subida validado con carpeta de cuarentena
- ✓ Descarga autorizada a través del servicio
- ✓ Handler de enlaces compartidos firmados y con caducidad
- ✓ Revisión de consultas parametrizadas y codificación de la salida
- ✓ Prueba con una solicitud de descarga falsificada

## Preguntas de repaso

- ✓ Tu controlador de subidas comprueba la extensión con una lista de permitidas y el archivo la supera. Explica qué puede enviar todavía un atacante y qué aporta la comprobación de la firma que la comprobación de la extensión no puede aportar.
- ✓ Un responsable reenvía un enlace compartido a alguien de fuera de la empresa antes de que caduque. ¿Qué partes de tu diseño limitan el daño y qué tendrías que añadir para revocar ese único enlace sin cambiar la clave para todos?
- ✓ Tu botón de descarga está oculto para los usuarios sin permiso de lectura, y la prueba de la solicitud falsificada sigue pasando. Explica con precisión qué línea de código hace que pase y por qué el botón oculto no tiene nada que ver.



### 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.