Daniel Alfocea

WORKSHOP PRESENCIAL PARA EQUIPOS TÉCNICOS

Desarrollo seguro

Encontrar y cerrar los fallos de seguridad que más llegan a producción en webs y APIs

Un día para que el equipo explote, arregle y pruebe los fallos que más se repiten en aplicaciones web y APIs: autorización rota, inyección, secretos expuestos y dependencias que nadie mira. El último bloque lo dedicáis a buscarlos en vuestro repositorio.

5 a 8 h	6	6	1
presenciales, según el equipo	laboratorios prácticos	bloques de contenido	bloque sobre vuestro código

danielalfocea.com/formacion/workshop-desarrollo-seguro

yo@danielalfocea.com

EL PROBLEMA QUE YA TENÉIS

Conocéis el OWASP Top 10 y el código sigue teniendo los mismos fallos

Casi todos los desarrolladores han visto alguna vez la lista de OWASP. Aun así, lo primero de la edición 2025 es el control de acceso roto. Rara vez es que el equipo no lo conozca. Lo que pasa es que **nadie comprueba en el servidor que el objeto que piden es de quien lo pide**.

Mientras tanto, el riesgo se ha ido a lo que no escribís vosotros: paquetes que se instalan con sus propios scripts, acciones de CI que se referencian por una etiqueta que alguien puede reescribir y tokens que siguen vivos años después de filtrarse.

En este workshop no repasamos la lista. Cada participante explota el fallo, prueba el arreglo que parece obvio, ve que no basta y deja un test que lo demuestra. Después repite la búsqueda en su propio código.

Las cinco zonas donde se rompe una aplicación

01 **Diseño**
Qué puede salir mal, dibujado en un folio antes de escribir código.

02 **Autorización**
Quién puede leer o modificar cada objeto, comprobado en el servidor.

03 **Entradas**
Por dónde entra un dato en el que no confías y en qué acaba convertido: SQL, HTML, una URL o un objeto.

04 **Identidad y secretos**
Tokens, sesiones, credenciales y lo que acaba escrito en los logs.

05 **Cadena de suministro y CI**
Dependencias, lockfiles, pipelines y los gates que bloquean el merge.

Cinco zonas, con un control probado en cada una. Os las lleváis a vuestro repositorio al terminar el día.

RESULTADO DEL WORKSHOP

Cinco cosas que sabréis hacer al terminar el día

Cada una la demostráis durante el día con un exploit, un arreglo y un test.

01 **Hacer un threat model en quince minutos**
Responder a las cuatro preguntas del Threat Modeling Manifesto sobre un flujo real y marcar las fronteras de confianza antes de ponerte a discutir controles.
Evidencia: ficha de una página sobre el checkout de la aplicación de prácticas.

02 **Cerrar la autorización por objeto y por propiedad**
Distinguir BOLA, BFLA y asignación masiva, y aplicar la comprobación en la capa de datos en lugar de en la interfaz.
Evidencia: el exploit que leía pedidos ajenos devuelve 404 y un test lo fija.

03

Tratar cada entrada según su destino

Consultas parametrizadas, escape por contexto, allowlist de destinos para SSRF y nada de pickle o yaml.load con datos de fuera.

Evidencia: los payloads de inyección y SSRF del laboratorio dejan de funcionar.

04

Mantener los secretos fuera del código y de los logs

Validar tokens con una lista cerrada de algoritmos, sacar los secretos del repositorio, rotar lo filtrado y limpiar lo que se registra.

Evidencia: un token alg=none rechazado y un log sin contraseñas ni datos de tarjeta.

05

Poner gates de CI que bloqueen de verdad

Lockfiles con hashes, SBOM, SAST, SCA, secret scanning y acciones fijadas por SHA, con una decisión clara sobre qué rompe el build.

Evidencia: un pull request malicioso contra el pipeline queda bloqueado antes de ejecutarse con secretos.

CASOS REALES

Incidentes de 2025 que se explican con lo que practicáis ese día

Todos son públicos y los documentó quien los descubrió o el propio fabricante. Ninguno necesitó una técnica exótica.

BROKEN OBJECT LEVEL AUTHORIZATION

McHire expuso más de 64 millones de solicitudes de empleo

Una cuenta de pruebas con usuario y contraseña 123456 daba acceso al panel. Desde ahí, cambiar un identificador secuencial en la petición devolvía los datos de otros candidatos. Se comunicó el 30 de junio de 2025 y se corrigió al día siguiente.

CONTROL QUE SE PRACTICA

Comprobación de propiedad en cada consulta, identificadores no predecibles y cuentas de prueba fuera de producción.

INSECURE DESIGN

Saltarse el middleware de Next.js con una cabecera

En marzo de 2025, CVE-2025-29927 permitió saltar el middleware enviando la cabecera x-middleware-subrequest. Las aplicaciones que solo comprobaban la autorización ahí se quedaron abiertas.

CONTROL QUE SE PRACTICA

Autorizar también junto a los datos, y no en una única capa que el atacante puede rodear.

INSECURE DESERIALIZATION

React2Shell: ejecución remota sin autenticar, CVSS 10.0

CVE-2025-55182, publicado el 3 de diciembre de 2025, permitía ejecutar código en el servidor a través de la deserialización de React Server Components.

CONTROL QUE SE PRACTICA
Un inventario de dependencias para saber en horas qué servicios están afectados, y no deserializar formatos con objetos a partir de entradas externas.

SOFTWARE SUPPLY CHAIN FAILURES

Shai-Hulud 2.0 se ejecutaba antes que los tests

Entre el 21 y el 24 de noviembre de 2025, paquetes npm comprometidos llevaban código en preinstall que robaba credenciales de nube y tokens de npm y GitHub en el momento de instalar.

CONTROL QUE SE PRACTICA
npm ci con lockfile, scripts de instalación desactivados en CI y publicación protegida con trusted publishing y WebAuthn.

CI/CD

tj-actions/changed-files volcó secretos en los logs de miles de repositorios

En marzo de 2025 alguien reescribió las etiquetas de la acción para que apuntaran a un commit malicioso. La usaban más de 23.000 repositorios, y los que la referenciaban por etiqueta ejecutaron el código nuevo sin tocar nada.

CONTROL QUE SE PRACTICA
Acciones y tareas fijadas por SHA completo, permisos mínimos del token de CI y secretos fuera de los pull requests externos.

PROGRAMA COMPLETO

Lo que se enseña, bloque por bloque

El orden sigue el camino de una petición: se diseña el flujo, alguien pide un objeto, envía datos, se autentica y el cambio pasa por el pipeline antes de llegar a producción. Las categorías son las del OWASP Top 10:2025 y el OWASP API Security Top 10 2023, y cada control se cruza con su requisito de ASVS 5.0.

01 Threat modeling ligero

Antes de hablar de controles, hay que saber qué proteges y de quién.

- Las cuatro preguntas: en qué trabajamos, qué puede salir mal, qué vamos a hacer y si lo hicimos bien.
- Fronteras de confianza: navegador, API, base de datos, servicios de terceros y metadatos de nube.
- Quién puede hacer esto, por dónde entra el dato no confiable y hasta dónde llega el daño.
- Cómo usar ASVS 5.0 como lista de requisitos y no como lectura de fondo.

> **Caso real:** CVE-2025-29927 en Next.js: la autorización vivía en una sola capa.

02 Autorización: el primer puesto del Top 10

Que el usuario esté autenticado no dice nada sobre qué puede tocar.

- BOLA e IDOR: leer o modificar objetos ajenos cambiando un identificador.
- BFLA y BOPLA: funciones de administración expuestas y propiedades que el cliente no debería poder escribir.
- Asignación masiva: el campo role o el precio que llegan en el cuerpo de la petición.
- Flujos de negocio sensibles: importes calculados en el cliente y pasos que se pueden saltar.

> **Caso real:** McHire, 2025: más de 64 millones de solicitudes accesibles cambiando un número.

03 Inyección y SSRF

El mismo dato es inofensivo o peligroso según dónde termine.

- SQL y NoSQL: por qué escapar a mano falla y qué hace una consulta parametrizada.
- XSS almacenado y el token guardado en localStorage.
- SSRF: la vista previa de un enlace que acaba pidiendo credenciales al servicio de metadatos de la nube.
- Allowlist de destinos, resolución de DNS y redirecciones, que es donde fallan los filtros por texto.

04 Autenticación, tokens y deserialización

Un token firmado solo protege si el servidor decide cómo se valida.

- JWT: alg=none, confusión de algoritmos y la lista cerrada que lo evita.
- Sesiones, expiración, revocación y dónde guardar el token en el navegador.
- Deserialización: pickle, yaml.load y cualquier formato que reconstruye objetos.
- Fallos de autenticación: contraseñas por defecto, cuentas de prueba y límites de intentos.

> **Caso real:** React2Shell, CVE-2025-55182: ejecución remota sin autenticar por deserialización.

05 Secretos, datos sensibles, logs y errores

Lo que se filtra suele estar en texto plano en algún sitio que nadie mira.

- Secretos en el código y en el historial de git: detección, retirada y rotación.
- Datos de tarjeta y personales en reposo: qué no se guarda nunca y qué se cifra.
- Logs que registran contraseñas y respuestas de error que enseñan la pila.
- Modo debug en producción y gestión de condiciones excepcionales (A10:2025).

> **Caso real:** GitGuardian, 2026: 28,65 millones de secretos nuevos en GitHub público durante 2025, y más del 64 % de las credenciales válidas en 2022 seguían funcionando en enero de 2026.

06 Cadena de suministro y gates de CI

En el Top 10:2025 ocupa el tercer puesto, y casi todo es código que el equipo no ha escrito.

- Lockfiles y hashes: npm ci, pip --require-hashes y scripts de instalación desactivados.
- SBOM en CycloneDX o SPDX, escaneo de la imagen y despliegue por digest.
- Pipelines: pull requests que modifican el YAML, dependency confusion y acciones fijadas por SHA.
- SAST, DAST, SCA y secret scanning: qué encuentra cada uno y cuáles bloquean el merge.

> **Caso real:** Shai-Hulud 2.0 en npm, tj-actions/changed-files en GitHub Actions y num2words en PyPI, todos en 2025.

PRÁCTICA DELIBERADA

Seis laboratorios sobre una tienda online vulnerable

Los laboratorios usan ByteBazaar, una tienda en Flask con PostgreSQL y MongoDB que se levanta con Docker Compose. Es pública y de código abierto (github.com/cr0hn/vulnerable-python). Todas las prácticas siguen el mismo patrón: explotar, probar el arreglo que parece obvio, ver por qué falla, aplicar el bueno y dejar un test.

01

Los pedidos de otro cliente

Con una cuenta normal, cada participante lee pedidos ajenos cambiando el identificador de la URL. Después prueba a ocultar el enlace en la interfaz y comprueba que no sirve de nada.

Resultado: la consulta filtra por propietario y un test falla si alguien quita el filtro.

02

Hacerse administrador y pagar lo que uno quiera

El registro acepta un campo role enviado por el cliente y el checkout se fía del importe que llega del navegador.

Resultado: lista explícita de campos escribibles e importe recalculado en el servidor.

03

Búsquedas que devuelven toda la base de datos

Inyección SQL en el buscador de productos e inyección NoSQL en el filtro de etiquetas. El primer arreglo escapa comillas a mano y vuelve a caer.

Resultado: consultas parametrizadas en las dos bases de datos y payloads guardados como casos de test.

04

La vista previa que pide las llaves de la nube

Una función que muestra la previsualización de un enlace sirve para consultar el servicio de metadatos simulado y sacar credenciales de un almacén de secretos.

Resultado: allowlist de destinos, resolución validada y redirecciones bloqueadas.

05

Un token sin firma y unos logs que lo cuentan todo

Se forja un JWT con alg=none para entrar como otro usuario. Después se revisan los logs de la aplicación y aparecen contraseñas y datos de tarjeta.

Resultado: algoritmos permitidos fijados en el servidor y logs que ya no registran datos sensibles.

06

El pipeline que se deja engañar

Un pull request modifica el YAML del pipeline para leer secretos y una dependencia interna se suplanta desde el registro público. Se compara el pipeline inseguro con su versión endurecida en Azure DevOps.

Resultado: gates de SCA, SAST y secret scanning, dependencias resueltas desde el registro correcto y secretos fuera de los pull requests.

SOBRE VUESTRO CÓDIGO

El último bloque lo hacéis en vuestro repositorio

Lo que habéis practicado sobre ByteBazaar lo buscáis después en el código de cada participante. Así os vais con hallazgos propios y no solo con ejemplos de otra aplicación.

01

Elegir un flujo que importe

Uno que mueva dinero o datos personales. Le hacéis el threat model de quince minutos con las cuatro preguntas.

02

Buscar los patrones del día

Objetos cargados por id sin filtrar por usuario, SQL concatenado, peticiones a URLs que manda el cliente, pickle o yaml.load con datos de fuera. Un escáner de secretos recorre también el historial de git.

03

Revisar la cadena

Si el CI instala desde el lockfile, si las acciones o tareas están fijadas y qué scripts se ejecutan al instalar dependencias.

04

Escribir un test que falle

Para el hallazgo más grave, un test que falle hoy y pase cuando esté arreglado.

05

Decidir la primera acción

Cada participante cuenta en voz alta qué arreglará primero y por qué. Si no quiere, no enseña el código.

> Cada participante trabaja en su portátil, sobre su propio repositorio. El código no sale de la empresa y yo no me llevo ninguna copia. No es una auditoría: los hallazgos son del equipo y se quedan con el equipo.

CÓMO ES EL DÍA

Entre 5 y 8 horas, según el equipo

Alternamos explicación corta y práctica. Cada bloque termina con un control probado, y el último es entero para vuestro código.

Bloque 1	**Diseñar antes de atacar** Threat model ligero sobre el checkout de ByteBazaar. Se levanta la aplicación con Docker y se marcan las fronteras de confianza.
Bloque 2	**Quién puede tocar qué** BOLA, BFLA, asignación masiva y flujos de negocio. Laboratorios 1 y 2.
Bloque 3	**Datos que se convierten en código** Inyección SQL y NoSQL, XSS y SSRF hacia los metadatos de la nube. Laboratorios 3 y 4.
Bloque 4	**Identidad, secretos y lo que queda en los logs** JWT, sesiones, deserialización, secretos en git y gestión de errores. Laboratorio 5.
Bloque 5	**Lo que el equipo no escribe** Dependencias, lockfiles, SBOM, ataques reales a npm y PyPI y gates de CI. Laboratorio 6.
Bloque 6	**Vuestro código** Cada participante aplica lo del día a su repositorio: threat model, búsqueda de patrones, revisión de la cadena y un test que falla. Cerramos con la primera acción de cada uno.

MATERIAL QUE SE QUEDA EN LA EMPRESA

El workshop se acaba y los controles se quedan

El material está hecho para usarlo en el repositorio y en las revisiones de código de las semanas siguientes.

Ficha de threat model de una página
Las cuatro preguntas, fronteras de confianza y un ejemplo resuelto.

Checklist de autorización por endpoint
Objeto, función y propiedad, con el test que corresponde a cada caso.

Mapa de controles
OWASP Top 10:2025 y API Security Top 10 cruzados con requisitos de ASVS 5.0 y con cada laboratorio.

Pares de arreglos
El arreglo que parece correcto, por qué falla y el que funciona, para cada vulnerabilidad del día.

Pipeline de referencia
Azure DevOps con los gates del día. Los mismos controles se pasan a GitHub Actions o GitLab CI.

Política de dependencias
Lockfiles, hashes, scripts de instalación, SBOM y qué hacer cuando un paquete conocido aparece comprometido.

Aplicación y laboratorios
ByteBazaar y los enunciados, para repetir las prácticas o usarlas con quien se incorpore al equipo.

LO QUE SE SUMA AL CURSO

Cuatro cosas que se quedan con el equipo, sin coste añadido

Son lo que un equipo acaba pidiendo después del curso.

+1

Paquete de evidencia de cumplimiento

Lista de asistencia firmada, certificado individual con fecha, horas y contenidos, temario mapeado a las normas que os apliquen y una carta de una página para entregar al auditor.

+2

Encuesta previa de 10 minutos

Antes del día, el equipo responde una encuesta corta sobre su stack y dónde se atasca. Con ella adapto los laboratorios a lo que usáis de verdad.

+3

Configuraciones en una rama de vuestro repositorio

Durante el bloque final dejo en una rama las configuraciones que hemos probado: tests de regresión, comprobaciones del pipeline y reglas de merge. Os las revisáis y las integráis como cualquier otro cambio.

+4

Laboratorios con soluciones

El repositorio de laboratorios con sus soluciones y una licencia de uso interno de 12 meses, para que quien entre al equipo no dependa de repetir el día.

CUMPLIMIENTO

Qué pide cada norma y qué os entrego como evidencia

El curso no cumple ninguna norma por sí solo. Sirve como evidencia documentada para el control de formación que vuestro auditor ya os pide, si la norma os aplica.

NORMA	QUÉ PIDE	QUÉ OS DEJO COMO EVIDENCIA
PCI DSS 4.0 · 6.2.2	Formación al menos cada 12 meses para quien desarrolla software a medida: diseño seguro, codificación segura y herramientas de testing, documentada.	Temario mapeado a esos puntos, asistencia firmada y fecha.
ISO 27001:2022 · A.6.3	Concienciación, educación y formación en seguridad adecuadas a la función de cada persona.	Certificado individual con fecha, horas y contenidos, y registro por persona.
ISO 27001:2022 · A.8.25 y A.8.28	Ciclo de vida de desarrollo seguro y codificación segura, incluida la formación de quien desarrolla.	Temario mapeado y las comprobaciones de pipeline que quedan en vuestra rama.
SOC 2 · CC1.4 y CC2.2	Programa de formación documentado, con registros de finalización por persona y profundidad según el rol.	Lista con nombre, fecha, contenido y rol.
DORA · art. 13(6)	Programas de formación en resiliencia operativa digital para el personal, proporcionados a su función (banca y seguros).	Registro de asistencia y temario del curso.

A QUIÉN VA DIRIGIDO

Para equipos que mantienen aplicaciones web y APIs en producción

Encaja especialmente con

- Desarrolladores backend y full stack que escriben endpoints, consultas y lógica de negocio.
- Tech leads que revisan pull requests y deciden qué bloquea un merge.
- Equipos de plataforma o DevOps que llevan el pipeline y las dependencias.
- Security champions que van a llevar estas prácticas al resto de la organización.
- Equipos de cualquier stack: los laboratorios están en Python, pero los controles son los mismos en otros lenguajes.

No es la formación adecuada si

- El equipo no escribe ni mantiene código de aplicación.
- Buscáis una introducción a la seguridad para perfiles no técnicos o para dirección.
- Esperáis una auditoría o un pentest de vuestras aplicaciones. El bloque final sobre vuestro código lo hace el propio equipo y no sustituye a una revisión externa.
- Necesitáis preparar una certificación concreta.

QUIÉN IMPARTE EL WORKSHOP

Seguridad y desarrollo contados por alguien que ha trabajado en los dos lados

Soy Daniel Alfocea y llevo veinte años en desarrollo y seguridad. En Banco Santander fui Principal Engineer del departamento de IA durante la puesta en marcha del área. En BBVA fui Tech Lead de ciberseguridad. En 42Crunch, en Reino Unido, fui responsable de uno de sus productos de seguridad de APIs.

He presentado investigación en RSA Conference y en OWASP AppSec EU, y he sido ponente habitual de RootedCON. El Instituto Nacional de Ciberseguridad (INCIBE) me ha llamado varias veces como juez de sus concursos. Fundé Navaja Negra y el capítulo de OWASP Madrid. Soy autor de herramientas de seguridad muy usadas, como GoLismero, Plecost y Dockerscan, y de LAPIS, una investigación publicada en arXiv sobre cómo describirle una API a un modelo con menos tokens. Con Alfonso Muñoz escribí *MCP Seguro. Guía práctica para desarrolladores, arquitectos software y tech leads*.

La aplicación de los laboratorios es mía y es pública. Junta los fallos que un equipo de desarrollo puede escribir cualquier semana, y en el workshop cada uno se arregla con un test que impide que vuelva.

GARANTÍAS

Dos compromisos por escrito y un límite

Si las configuraciones no funcionan, las arreglo

Si las configuraciones que dejo en vuestro repositorio no funcionan, hago una sesión remota gratuita para arreglarlas.

Si os atascáis en 30 días, una hora gratis

Si a los 30 días el equipo se ha atascado aplicando algo del curso, os dedico una hora en remoto sin coste.

Esto no es una auditoría ni una promesa de resultados: no garantizo porcentajes de mejora ni que no haya incidentes.

ANTES DE RESERVAR

Preguntas que conviene resolver ahora

¿Cuánto cuesta?
Depende de la duración, entre cinco y ocho horas, y del tamaño del equipo. Tras la primera conversación te mando una propuesta cerrada.

¿En qué se diferencia del workshop de desarrollo seguro con IA?
Este no habla de asistentes de código. Va de las vulnerabilidades de la aplicación y de la cadena de suministro, las escriba una persona o una herramienta. Si el equipo ya trabaja a diario con Claude Code o GitHub Copilot, el otro workshop cubre los riesgos propios de esas herramientas.

¿Se adapta a nuestro lenguaje o stack?
Los laboratorios están en Python y el ejemplo de pipeline usa Azure DevOps. Los controles son los mismos en otros lenguajes y otros CI: autorizar junto a los datos, parametrizar consultas, fijar dependencias y bloquear el merge sin evidencia. No reescribo los laboratorios en otro lenguaje.

¿Usamos nuestro repositorio durante el día?
Sí, en el último bloque. Los laboratorios se hacen sobre ByteBazaar porque está preparada para romperse sin consecuencias y todo el grupo ve el mismo fallo. Después, cada participante aplica lo aprendido a su propio código, en su portátil.

¿Veis o os lleváis nuestro código?
No. El código no sale de la empresa y no me llevo ninguna copia. Si un participante quiere enseñarme algo, le ayudo con la duda, pero no es una auditoría ni se entrega un informe de hallazgos.

¿Qué nivel hace falta?
Saber leer Python básico, usar git y levantar contenedores con Docker. No hace falta experiencia previa en seguridad.

¿Se puede bonificar por FUNDAE?
Pregúntamelo al pedir fecha. Te paso la información que necesites para que vuestra gestoría o vuestro departamento de formación lo valoren.

¿Se puede impartir en remoto?
Sí, aunque en presencial funciona mejor para desatascar laboratorios y echar una mano en el bloque sobre vuestro código. El alcance es el mismo.

¿Garantiza que no tendremos incidentes de seguridad?
No. Se garantizan los seis laboratorios y la entrega del material. Que baje el riesgo depende de que el equipo aplique y mantenga los controles después.

¿Y si necesitamos seguimiento?
Existe un programa de 6 semanas de esta misma temática, con medición antes y después y revisión de vuestros PR. Tienes el detalle en el programa.

FORMATO

Un equipo, una sala y un mismo código de prácticas

Incluye la preparación, entre cinco y ocho horas presenciales según el equipo y todo el material de trabajo. Con grupos pequeños puedo revisar los laboratorios mientras los hacéis y echar una mano a cada participante en el bloque sobre su código. Con demasiada gente, la práctica se queda en demostración.

- Participantes del mismo equipo o de áreas que trabajan juntas.
- Lo imparto en español o en inglés.
- Sala, pantalla, wifi y portátiles a cargo de la empresa, con Docker instalado y acceso al repositorio de cada participante.
- Desplazamiento fuera de Madrid acordado antes de cerrar la fecha.
- Incluye el paquete de evidencia, las configuraciones en vuestra rama y los dos compromisos de arriba.
- Si buscáis una auditoría, un certificado de cumplimiento o una charla para escuchar, esto no es. Es un día de trabajo para un equipo que va a aplicar lo aprendido.

Escríbeme a **yo@danielalfocea.com**

En la primera conversación vemos si encajamos, cuántos vais a ser y qué fecha os viene bien.

Bonificable a través de FUNDAE.

CORREO

yo@danielalfocea.com

PROGRAMA EN LA WEB

danielalfocea.com/formacion/workshop-desarrollo-seguro