Eymia Labs

INFORME DE PENTEST · EJEMPLO

Informe de Pruebas de Penetración: Aplicación Web y API

EJEMPLO — datos ficticios. Este documento es una muestra del formato de entrega de Eymia Labs. "Tienda Demo S. A.", sus sistemas, personas, dominios, credenciales y hallazgos son inventados. Los dominios usan el sufijo reservado `.example`. No describe a ningún cliente real.

CLIENTE	Tienda Demo S. A. (ficticio)
SISTEMA EVALUADO	Tienda en línea "Tienda Demo" (Next.js) y API de comercio (Node.js / NestJS)
TIPO DE SERVICIO	Pentest web y API, caja gris
REFERENCIA	EYM-PT-2026-000 (ejemplo)
VENTANA DE PRUEBAS	5 al 16 de octubre de 2026
FECHA DEL INFORME	21 de octubre de 2026
CLASIFICACIÓN	CONFIDENCIAL. Uso exclusivo del cliente
PREPARADO POR	Eymia Labs

EJEMPLO — DATOS FICTICIOS · DOCUMENTO DE MUESTRA, NO CORRESPONDE A NINGÚN CLIENTE

Control de versiones

EJEMPLO — DATOS FICTICIOS.

VERSIÓN	FECHA	AUTOR	DESCRIPCIÓN
0.9	19/10/2026	Eymia Labs	Borrador para revisión interna
1.0	21/10/2026	Eymia Labs	Entrega al cliente
1.1	06/11/2026	Eymia Labs	Anexo A: resultados del retest

Distribución

NOMBRE	ORGANIZACIÓN	ROL
[Gerente general]	Tienda Demo S. A.	Destinatario del resumen ejecutivo
[Jefe de tecnología]	Tienda Demo S. A.	Contacto técnico principal
Consultor principal	Eymia Labs	Autor

Las notificaciones de hallazgos críticos se enviaron por el canal seguro acordado: EYM-01 el 7/10/2026 y EYM-02 el 8/10/2026.

1. Resumen ejecutivo

EJEMPLO — DATOS FICTICIOS.

1.1. Riesgo general: ALTO

Entre el 5 y el 16 de octubre de 2026 evaluamos la tienda en línea de Tienda Demo S. A. y la API que la alimenta, con autorización escrita del cliente y cuentas de prueba de cliente y de operador.

En pocas palabras: un atacante sin relación con la empresa, con solo crear una cuenta gratuita, podía convertirse en administrador de la tienda. Con ese acceso podía ver y modificar los pedidos y los datos personales de todos los clientes, cambiar precios y desactivar productos. Además, cualquier cliente registrado podía ver los pedidos de otros clientes, incluidos nombre, dirección y teléfono.

Esto expone a la empresa a fraude directo, a la pérdida de confianza de sus clientes y a incumplimientos de la Ley Orgánica de Protección de Datos Personales (LOPDP), que exige medidas de seguridad adecuadas para los datos que trata.

La buena noticia: las dos fallas más graves tienen correcciones acotadas, de horas a pocos días de trabajo, y no requieren rediseñar el sistema. En el retest del 6 de noviembre (Anexo A), 5 de los 7 hallazgos estaban corregidos, incluidos los dos más graves.

1.2. Hallazgos por severidad

SEVERIDAD	CANTIDAD	DISTRIBUCIÓN
CRÍTICA	1	
ALTA	2	
MEDIA	3	
BAJA	1	
INFORMATIVA	0	
Total	**7**	

1.3. Resumen de hallazgos

ID	HALLAZGO	SEVERIDAD	CVSS 4.0	ESFUERZO DE CORRECCIÓN
EYM-01	Tokens JWT firmados con secreto débil y sin expiración	CRÍTICA	9.3	Bajo
EYM-02	Acceso a pedidos de otros clientes (IDOR / BOLA)	ALTA	8.6	Bajo
EYM-03	CORS refleja cualquier origen y permite credenciales	ALTA	7.1	Bajo
EYM-04	Clave administrativa del buscador expuesta en el JavaScript público	MEDIA	6.9	Bajo
EYM-05	Sin límite de intentos en el inicio de sesión	MEDIA	6.9	Bajo
EYM-06	Enumeración de usuarios en inicio de sesión y recuperación	MEDIA	6.9	Bajo
EYM-07	Cabeceras de seguridad HTTP ausentes	BAJA	2.1	Bajo

1.4. Recomendaciones para la gerencia

1. **Esta semana:** corregir EYM-01 y EYM-02 y cerrar todas las sesiones activas. Son las únicas fallas que permiten acceso masivo a datos personales.
2. **En 30 días:** corregir EYM-03 a EYM-06 y rotar la clave del buscador.
3. **De forma permanente:** incluir pruebas de autorización en las pruebas automatizadas y escaneo de secretos en el pipeline de despliegue, para que estas clases de falla no vuelvan a aparecer.

1.5. Fortalezas observadas

- Las consultas a la base de datos usan un ORM con parámetros; no encontramos inyección SQL en los puntos probados.
- Los datos de tarjeta los procesa íntegramente la pasarela de pagos; la API no almacena números de tarjeta.
- El sitio fuerza HTTPS con TLS 1.2 y 1.3.
- Las contraseñas se almacenan con bcrypt.

2. Alcance y metodología

EJEMPLO — DATOS FICTICIOS.

2.1. Alcance autorizado

#	ACTIVO	TIPO	ENTORNO
1	`https://tienda.example`	App web Next.js 15 (App Router)	Staging, réplica de producción
2	`https://api.tienda.example/v1`	API REST NestJS, 46 endpoints documentados (OpenAPI)	Staging
3	`https://admin.tienda.example`	Panel administrativo	Staging

Cuentas de prueba: 2 clientes (`cliente.a@tienda.example`, `cliente.b@tienda.example`) y 1 operador. No se entregó cuenta de administrador.

Fuera de alcance: pasarela de pagos de terceros, servicio de correo, infraestructura del proveedor de nube, denegación de servicio e ingeniería social.

Origen de las pruebas: 2 direcciones IP fijas comunicadas al cliente antes del inicio.

2.2. Metodología

Seguimos estos estándares:

- **OWASP Web Security Testing Guide (WSTG) v4.2:** recolección de información, configuración, identidad, autenticación, autorización, sesiones, validación de entrada, lógica de negocio y lado del cliente.
- **OWASP ASVS 5.0, nivel 2:** como lista de verificación de controles.
- **OWASP API Security Top 10 (2023):** con énfasis en BOLA (API1), autenticación rota (API2) y autorización a nivel de función (API5).
- **CVSS 4.0 (base, CVSS-B):** para el puntaje técnico. La severidad final considera además el impacto de negocio, y cuando difieren se explica en el hallazgo.

Fases: preparación y verificación de alcance; reconocimiento; pruebas manuales con proxy de interceptación; pruebas automatizadas con límite de velocidad; validación con prueba de concepto mínima; documentación; reunión de cierre; retest.

Datos personales: para cada prueba de concepto se accedió como máximo a 3 registros. Los datos que aparecen en este informe están enmascarados. La evidencia se guarda cifrada y se elimina 30 días después del retest.

2.3. Escala de severidad

SEVERIDAD	CVSS 4.0	SIGNIFICADO PARA EL NEGOCIO	PLAZO SUGERIDO
CRÍTICA	9.0 – 10.0	Compromiso total o masivo, explotable con facilidad	24 – 72 horas
ALTA	7.0 – 8.9	Acceso a datos o funciones sensibles	7 días
MEDIA	4.0 – 6.9	Facilita ataques o expone información limitada	30 días
BAJA	0.1 – 3.9	Defensa en profundidad	90 días
INFORMATIVA	0.0	Buenas prácticas sin riesgo directo	A criterio

2.4. Limitaciones

La evaluación se realizó en un entorno de staging durante 8 días hábiles. Un pentest es una muestra acotada en tiempo y alcance; no garantiza la ausencia de otras vulnerabilidades. Las diferencias entre staging y producción se verificaron con el cliente el 5/10/2026.

3. Hallazgos detallados

EJEMPLO — DATOS FICTICIOS. TODAS LAS PETICIONES, RESPUESTAS, TOKENS Y DATOS SON SIMULADOS.

EYM-01 Tokens JWT firmados con secreto débil y sin expiración

Severidad	CRÍTICA
CVSS 4.0	9.3 · `CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:L/SC:N/SI:N/SA:N`
Activo	`https://api.tienda.example/v1` (todas las rutas autenticadas)
Categoría	OWASP API2:2023 Broken Authentication · WSTG-SESS-10
CWE	CWE-1391 (uso de credenciales débiles), CWE-613 (expiración de sesión insuficiente)
Estado	Corregido en retest

Descripción

La API emite tokens JWT firmados con HMAC-SHA256 (HS256) usando un secreto corto y presente en diccionarios públicos de contraseñas. Con un token propio (obtenido al registrar una cuenta gratuita), el secreto se recuperó por fuerza bruta fuera de línea en menos de un minuto. Conociendo el secreto, es posible firmar tokens con cualquier identificador de usuario y cualquier rol, incluido `admin`, que la API acepta como válidos.

Además, los tokens no incluyen el campo `exp`: un token robado o filtrado sigue siendo válido indefinidamente, y el cierre de sesión no lo invalida.

El registro es abierto y gratuito, por lo que el requisito de tener una cuenta no se considera un privilegio previo (PR:N).

Impacto de negocio

Cualquier persona en internet puede obtener control administrativo total de la tienda: ver y exportar los datos personales de todos los clientes, modificar pedidos, cambiar precios y desactivar productos. Es una brecha de datos personales con obligación de notificación según la LOPDP y un riesgo directo de fraude.

Pasos de reproducción

1. Registrar una cuenta en `https://tienda.example/registro` e iniciar sesión.

2. Copiar el token del encabezado `Authorization` de cualquier petición a la API.
3. Ejecutar un ataque de diccionario fuera de línea contra la firma HS256 del token con una lista pública de contraseñas comunes.
4. Con el secreto recuperado, generar un token con `"role": "admin"` y un `sub` arbitrario.
5. Usar ese token contra `GET /v1/admin/customers`.

Evidencia (simulada)

Contenido decodificado del token emitido a la cuenta de prueba. No contiene `exp`:

```
{ "alg": "HS256", "typ": "JWT" }
{ "sub": "usr_8f2c1a", "email": "cliente.a@tienda.example", "role": "customer", "iat":
1791100800 }
```

Secreto recuperado (enmascarado): `t**********26` (14 caracteres; nombre de la empresa seguido del año).

Petición con un token forjado con rol de administrador:

```
GET /v1/admin/customers?limit=3 HTTP/1.1
Host: api.tienda.example
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c3JfMDAwMDAwIiwicm9sZSI6ImFkbWluIn0.
<firma-ficticia>
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "total": 18342,
  "items": [
    { "id": "usr_1a…", "name": "M*** P***", "email": "m***@correo.example", "phone": "+593
9** *** *12" },
    { "id": "usr_1b…", "name": "J*** R***", "email": "j***@correo.example", "phone": "+593
9** *** *47" },
    { "id": "usr_1c…", "name": "A*** C***", "email": "a***@correo.example", "phone": "+593
9** *** *05" }
  ]
}
```

Recomendación

1. Generar un secreto nuevo de al menos 256 bits aleatorios o, preferiblemente, migrar a firma asimétrica (ES256 o RS256) con la clave privada en un gestor de secretos.
2. Fijar una expiración corta para el token de acceso (15 minutos) con token de renovación rotativo y revocable.
3. Aceptar solo el algoritmo esperado al verificar.

4. Tras el cambio, todos los tokens anteriores quedan inválidos; comunicar a los usuarios que deberán iniciar sesión de nuevo.

```
// auth.module.ts (NestJS)
JwtModule.registerAsync({
  inject: [ConfigService],
  useFactory: (cfg: ConfigService) => ({
    privateKey: cfg.getOrThrow('JWT_PRIVATE_KEY'), // desde el gestor de secretos
    publicKey: cfg.getOrThrow('JWT_PUBLIC_KEY'),
    signOptions: { algorithm: 'ES256', expiresIn: '15m', issuer: 'api.tienda.example',
audience: 'tienda.example' },
    verifyOptions: { algorithms: ['ES256'], issuer: 'api.tienda.example', audience:
'tienda.example' },
  }),
});
```

Referencias: OWASP API Security Top 10 2023 API2; OWASP JWT Cheat Sheet; OWASP ASVS 5.0 (tokens autocontenidos); CWE-1391; CWE-613; RFC 8725 (JWT Best Current Practices).

EYM-02 Acceso a pedidos de otros clientes (IDOR / BOLA)

Severidad	ALTA
CVSS 4.0	8.6 · `CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N`
Activo	`GET` y `PATCH /v1/orders/{orderId}`
Categoría	OWASP API1:2023 Broken Object Level Authorization · WSTG-ATHZ-04
CWE	CWE-639 (omisión de autorización mediante clave controlada por el usuario)
Estado	Corregido en retest

Descripción

La API verifica que el usuario haya iniciado sesión, pero no que el pedido solicitado le pertenezca. Los identificadores de pedido son números secuenciales, por lo que se pueden recorrer. El mismo problema afecta a `PATCH`, que permite cambiar la dirección de envío de un pedido ajeno mientras está en estado "pendiente de despacho".

Impacto de negocio

Cualquier cliente registrado puede leer nombre, dirección, teléfono y productos comprados de todos los pedidos de la tienda, y desviar envíos a otra dirección. Es exposición masiva de datos personales y una vía de fraude logístico.

Pasos de reproducción

1. Iniciar sesión como `cliente.a@tienda.example` y crear un pedido. Anotar su número (por ejemplo, 100231).
2. Iniciar sesión como `cliente.b@tienda.example` en otro navegador.
3. Como cliente B, solicitar `GET /v1/orders/100231`.
4. Como cliente B, enviar `PATCH /v1/orders/100231` con una nueva dirección de envío.

Evidencia (simulada)

```
GET /v1/orders/100231 HTTP/1.1
Host: api.tienda.example
Authorization: Bearer <token de cliente.b>
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": 100231,
  "customerId": "usr_8f2c1a",
  "shipping": { "name": "Cliente A (prueba)", "address": "Av. Ejemplo 123", "phone": "+593
9** *** *00" },
  "items": [{ "sku": "DEMO-001", "qty": 1, "price": 49.90 }],
  "status": "PENDING_SHIPMENT"
}
```

```
PATCH /v1/orders/100231 HTTP/1.1
Host: api.tienda.example
Authorization: Bearer <token de cliente.b>
Content-Type: application/json

{ "shipping": { "address": "Calle Desvío 456" } }
```

```
HTTP/1.1 200 OK
{ "id": 100231, "shipping": { "address": "Calle Desvío 456" }, "status": "PENDING_SHIPMENT"
}
```

Recomendación

Filtrar siempre por el propietario en la consulta (no después) y devolver 404 para no confirmar la existencia de pedidos ajenos. Aplicar el mismo patrón a todos los recursos con dueño (direcciones, carrito, facturas). Considerar identificadores no secuenciales (UUID) como defensa adicional, no como control principal.

```
// orders.service.ts (NestJS + Prisma)
async findForUser(orderId: number, userId: string) {
  const order = await this.prisma.order.findFirst({
    where: { id: orderId, customerId: userId },
  });
  if (!order) throw new NotFoundException();
  return order;
}

// orders.controller.ts
@Get(':id')
findOne(@Param('id', ParseIntPipe) id: number, @CurrentUser() user: AuthUser) {
  return this.orders.findForUser(id, user.id);
}
```

Agregar una prueba automatizada que verifique que el cliente B recibe 404 al pedir un pedido del cliente A.

Referencias: OWASP API Security Top 10 2023 API1; OWASP Authorization Cheat Sheet; OWASP ASVS V4.2.1; CWE-639.

EYM-03 CORS refleja cualquier origen y permite credenciales

Severidad	ALTA
CVSS 4.0	7.1 · `CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:H/VI:L/VA:N/SC:N/SI:N/SA:N`
Activo	`https://api.tienda.example/v1`
Categoría	WSTG-CLNT-07 · OWASP API8:2023 Security Misconfiguration
CWE	CWE-942 (política CORS permisiva con dominios no confiables)
Estado	Corregido en retest

Descripción

La API copia el valor del encabezado `Origin` de la petición en `Access-Control-Allow-Origin` y responde además con `Access-Control-Allow-Credentials: true`. La cookie de sesión del flujo de pago se emite con `SameSite=None; Secure`. En consecuencia, cualquier sitio web puede hacer peticiones autenticadas a la API en nombre de un cliente que lo visite y leer las respuestas.

Impacto de negocio

Basta con que un cliente con sesión iniciada abra un enlace a una página controlada por un atacante (por ejemplo, desde una campaña de correo falsa) para que esa página lea su perfil, direcciones e historial de pedidos y modifique datos de su cuenta, sin que el cliente lo note.

Pasos de reproducción

1. Enviar a la API una petición con `Origin: https://atacante.example` y observar los encabezados de respuesta.
2. Alojar en un dominio de prueba controlado por el equipo de pruebas una página que haga `fetch('https://api.tienda.example/v1/me', { credentials: 'include' })`.
3. Con la sesión de la cuenta de prueba iniciada, visitar la página y comprobar que lee la respuesta.

Evidencia (simulada)

```
GET /v1/me HTTP/1.1
Host: api.tienda.example
Origin: https://atacante.example
Cookie: td_session=<sesión de cliente.a>
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://atacante.example
Access-Control-Allow-Credentials: true
Vary: Origin
Content-Type: application/json

{ "id": "usr_8f2c1a", "email": "cliente.a@tienda.example", "addresses": [ … ] }
```

Recomendación

Reemplazar el reflejo del origen por una lista explícita de orígenes permitidos. Si se usan cookies entre subdominios, preferir `SameSite=Lax` cuando el flujo lo permita.

```
// main.ts (NestJS)
app.enableCors({
  // Lista cerrada: nunca reflejar el encabezado Origin recibido
  origin: ['https://tienda.example', 'https://admin.tienda.example'],
  credentials: true,
  methods: ['GET', 'POST', 'PATCH', 'DELETE'],
});
```

No usar expresiones regulares laxas (por ejemplo, que acepten `tienda.example.atacante.example`) ni aceptar el origen `null`.

Referencias: OWASP WSTG-CLNT-07; OWASP HTML5 Security Cheat Sheet (CORS); PortSwigger Web Security Academy: CORS; CWE-942.

EYM-04 Clave administrativa del buscador expuesta en el JavaScript público

Severidad	MEDIA
CVSS 4.0	6.9 · `CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:L/SI:L/SA:N`
Activo	`https://tienda.example/_next/static/chunks/*.js`
Categoría	WSTG-INFO-05 · OWASP Top 10 2021 A05
CWE	CWE-798 (credenciales embebidas), CWE-540 (información sensible en código fuente)
Estado	Corregido en retest (clave rotada)

Descripción

El bundle de JavaScript que se descarga cualquier visitante contiene la clave administrativa del servicio externo de búsqueda de productos. La clave se definió como variable `NEXT_PUBLIC_SEARCH_ADMIN_KEY`; en Next.js, todo lo que lleva el prefijo `NEXT_PUBLIC_` se incrusta en el código del navegador. La clave permite crear, modificar y borrar índices, no solo buscar.

El CVSS refleja que el impacto recae en un sistema posterior (el servicio de búsqueda). Para el negocio, la severidad es relevante porque el buscador es la vía principal de descubrimiento de productos.

Impacto de negocio

Un atacante puede vaciar el índice (la búsqueda deja de mostrar productos), alterar nombres y precios mostrados en los resultados o insertar enlaces hacia sitios fraudulentos dentro de la tienda.

Pasos de reproducción

1. Abrir `https://tienda.example` y descargar los archivos de `/_next/static/chunks/`.
2. Buscar cadenas asociadas a la configuración del buscador.
3. Consultar con esa clave el endpoint de administración de índices del proveedor de búsqueda (solo lectura de la configuración; no se modificó ningún índice).

Evidencia (simulada)

```
$ grep -o 'SEARCH_ADMIN_KEY[^,]*' 4127-3f9a1c.js
SEARCH_ADMIN_KEY:"sk_adm_7Q2…(enmascarado)…9xL"
```

```
GET /indexes/products/settings HTTP/1.1
Host: search.proveedor.example
Authorization: Bearer sk_adm_7Q2…9xL
```

```
HTTP/1.1 200 OK
{ "uid": "products", "permissions": ["search", "documents.add", "documents.delete",
"indexes.delete"] }
```

Recomendación

1. Rotar la clave de inmediato.
2. En el navegador, usar solo una clave de búsqueda de solo lectura, restringida al índice público.
3. Las operaciones de indexación se hacen en el servidor, con la clave administrativa en variables sin prefijo `NEXT_PUBLIC_`.
4. Agregar escaneo de secretos (por ejemplo, gitleaks) al pipeline para detectar claves en el código y en el bundle generado.

```
// app/api/reindex/route.ts (Next.js, se ejecuta solo en el servidor)
import 'server-only';
export async function POST(req: Request) {
  await requireAdmin(req);
  const key = process.env.SEARCH_ADMIN_KEY!; // sin NEXT_PUBLIC_
  // … llamada al proveedor de búsqueda
}
```

```
# .env
NEXT_PUBLIC_SEARCH_KEY=<clave de solo búsqueda>
SEARCH_ADMIN_KEY=<clave administrativa, solo servidor>
```

Referencias: Next.js Docs: Environment Variables; OWASP Secrets Management Cheat Sheet; CWE-798; CWE-540.

EYM-05 Sin límite de intentos en el inicio de sesión

Severidad	MEDIA
CVSS 4.0	6.9 · `CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N`
Activo	`POST /v1/auth/login`, `POST /v1/auth/forgot-password`
Categoría	OWASP API4:2023 Unrestricted Resource Consumption · WSTG-ATHN-03
CWE	CWE-307 (restricción inadecuada de intentos de autenticación excesivos)
Estado	Parcialmente corregido en retest

Descripción

El endpoint de inicio de sesión acepta un número ilimitado de intentos por cuenta y por dirección IP. Se enviaron 500 intentos fallidos contra la cuenta de prueba en 60 segundos, a un ritmo controlado y acordado, sin bloqueo, demora progresiva ni CAPTCHA. El intento 501, con la contraseña correcta, fue aceptado.

Impacto de negocio

Permite ataques automatizados de relleno de credenciales (contraseñas filtradas de otros sitios) y de adivinación de contraseñas, que suelen terminar en cuentas tomadas, compras con saldo o datos guardados y reclamos de clientes.

Pasos de reproducción

1. Enviar peticiones repetidas a `POST /v1/auth/login` con la cuenta `cliente.a@tienda.example` y contraseñas incorrectas.
2. Observar que todas reciben `401` sin variación ni bloqueo.
3. Enviar la contraseña correcta y observar `200`.

Evidencia (simulada)

```
Intentos: 500 · Duración: 60 s · Respuestas: 500 × 401 Unauthorized
Encabezados de límite (RateLimit, Retry-After): ausentes
Intento 501 (contraseña correcta): 200 OK
```

Recomendación

Aplicar límites por IP y por cuenta, con demora progresiva y bloqueo temporal, y registrar los eventos para alertas. Considerar un desafío (por ejemplo, Turnstile o CAPTCHA) tras varios fallos y ofrecer segundo factor a cuentas de operadores.

```
// app.module.ts (NestJS)
ThrottlerModule.forRoot([{ name: 'login', ttl: 60_000, limit: 5 }]),

// auth.controller.ts
@UseGuards(ThrottlerGuard)
@Throttle({ login: { ttl: 60_000, limit: 5 } })
@Post('login')
login(@Body() dto: LoginDto) { … }
```

Además del límite por IP, mantener un contador de fallos por cuenta (por ejemplo, en Redis) para frenar ataques distribuidos desde muchas IP.

Referencias: OWASP Authentication Cheat Sheet; OWASP Credential Stuffing Prevention Cheat Sheet; OWASP ASVS V2.2.1; CWE-307.

EYM-06 Enumeración de usuarios en inicio de sesión y recuperación

Severidad	MEDIA
CVSS 4.0	6.9 · `CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N`
Activo	`POST /v1/auth/login`, `POST /v1/auth/forgot-password`
Categoría	WSTG-IDNT-04
CWE	CWE-204 (discrepancia observable en la respuesta)
Estado	Corregido en retest

Descripción

Los mensajes de error distinguen entre un correo no registrado y una contraseña incorrecta. La recuperación de contraseña responde de forma distinta según el correo exista o no.

Aunque el puntaje CVSS 4.0 base es 6.9, la severidad de negocio es Media-baja por sí sola. Se mantiene en Media porque, combinada con EYM-05, facilita ataques dirigidos a clientes reales.

Impacto de negocio

Un atacante puede confirmar qué correos son clientes de la tienda, armar listas para phishing con la marca de la empresa y concentrar ataques de contraseñas solo en cuentas que existen.

Pasos de reproducción

1. Enviar `POST /v1/auth/login` con un correo inexistente y observar el mensaje.
2. Repetir con `cliente.a@tienda.example` y una contraseña incorrecta.
3. Comparar las respuestas. Repetir con `/v1/auth/forgot-password`.

Evidencia (simulada)

```
POST /v1/auth/login        { "email": "noexiste@correo.example", "password": "x" }
HTTP/1.1 404 Not Found     { "message": "Usuario no registrado" }

POST /v1/auth/login        { "email": "cliente.a@tienda.example", "password": "x" }
HTTP/1.1 401 Unauthorized  { "message": "Contraseña incorrecta" }
```

Recomendación

Responder con el mismo código, mensaje y tiempo aproximado en ambos casos.

```
// auth.service.ts
async validate(email: string, password: string) {
  const user = await this.users.findByEmail(email);
  // Comparar siempre para igualar tiempos, aunque el usuario no exista
  const ok = await bcrypt.compare(password, user?.passwordHash ?? DUMMY_BCRYPT_HASH);
  if (!user || !ok) throw new UnauthorizedException('Correo o contraseña incorrectos');
  return user;
}

// forgot-password: siempre 202 con el mismo mensaje
return { message: 'Si el correo está registrado, recibirás instrucciones.' };
```

Referencias: OWASP WSTG-IDNT-04; OWASP Authentication Cheat Sheet (Authentication and Error Messages); CWE-204.

EYM-07 Cabeceras de seguridad HTTP ausentes

Severidad	BAJA
CVSS 4.0	2.1 · `CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:A/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N`
Activo	`https://tienda.example`, `https://admin.tienda.example`
Categoría	WSTG-CONF-07, WSTG-CLNT-09 · OWASP Top 10 2021 A05
CWE	CWE-1021 (restricción inadecuada de capas de interfaz), CWE-693 (falla de mecanismo de protección)
Estado	Pendiente (riesgo aceptado temporalmente por el cliente hasta el 30/11/2026)

Descripción

Las respuestas del sitio y del panel administrativo no incluyen `Content-Security-Policy`, `Strict-Transport-Security`, `X-Content-Type-Options`, `Referrer-Policy` ni `Permissions-Policy`, y no restringen su inclusión en marcos (`frame-ancestors`). El panel puede cargarse dentro de un `iframe` en otro sitio.

Impacto de negocio

No es una vulnerabilidad explotable por sí sola, pero quita capas de defensa: facilita ataques de clickjacking contra operadores del panel y aumenta el daño de cualquier futura falla de inyección de scripts.

Pasos de reproducción

1. Solicitar `https://admin.tienda.example/` y revisar los encabezados de respuesta.
2. Cargar la URL en un `iframe` desde un archivo HTML local y comprobar que se muestra.

Evidencia (simulada)

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Powered-By: Next.js
Cache-Control: private, no-cache
```

Recomendación

```
// next.config.js
const securityHeaders = [
  { key: 'Strict-Transport-Security', value: 'max-age=63072000; includeSubDomains; preload'
},
  { key: 'X-Content-Type-Options', value: 'nosniff' },
  { key: 'Referrer-Policy', value: 'strict-origin-when-cross-origin' },
  { key: 'Permissions-Policy', value: 'camera=(), microphone=(), geolocation=()' },
  { key: 'Content-Security-Policy', value: "frame-ancestors 'none'; base-uri 'self';
object-src 'none'" },
];
module.exports = {
  poweredByHeader: false,
  async headers() {
    return [{ source: '/:path*', headers: securityHeaders }];
  },
};
```

Desplegar después una CSP completa con nonces (soportados por el middleware de Next.js), primero en modo `Content-Security-Policy-Report-Only` para no romper el sitio.

Referencias: OWASP Secure Headers Project; OWASP HTTP Headers Cheat Sheet; OWASP Clickjacking Defense Cheat Sheet; Next.js Docs: Content Security Policy; CWE-1021.

4. Plan de remediación priorizado

EJEMPLO — DATOS FICTICIOS.

PRIORIDAD	ID	ACCIÓN	RESPONSABLE SUGERIDO	ESFUERZO ESTIMADO	PLAZO
1	EYM-01	Nuevo secreto o firma ES256, expiración de 15 min, renovación revocable, invalidar tokens actuales	Backend	1 – 2 días	72 horas
2	EYM-02	Filtro por propietario en pedidos, direcciones y facturas; pruebas automatizadas de autorización	Backend	1 día	7 días
3	EYM-03	Lista explícita de orígenes CORS; revisar `SameSite`	Backend	2 horas	7 días
4	EYM-04	Rotar clave; clave de solo búsqueda en el navegador; indexación en servidor	Frontend	4 horas	7 días (rotación: inmediata)
5	EYM-05	Límite por IP y por cuenta, bloqueo temporal, alertas	Backend / DevOps	1 día	30 días
6	EYM-06	Mensajes y tiempos uniformes	Backend	2 horas	30 días
7	EYM-07	Cabeceras de seguridad; CSP en modo reporte y luego obligatoria	Frontend / DevOps	4 horas + ajuste de CSP	90 días
—	Transversal	Escaneo de secretos y dependencias en CI; pruebas de autorización por rol	DevOps	2 – 3 días	90 días

Eymia Labs puede ejecutar o acompañar estas correcciones como servicio adicional ("encontramos y corregimos").

Anexo A. Resultados del retest

EJEMPLO — DATOS FICTICIOS.

Fecha del retest: 6 de noviembre de 2026 **Entorno:** staging, versión `v2.14.0` informada por el cliente **Alcance:** únicamente los hallazgos EYM-01 a EYM-07.

ID	HALLAZGO	SEVERIDAD ORIGINAL	ESTADO	VERIFICACIÓN REALIZADA
EYM-01	JWT con secreto débil y sin expiración	CRÍTICA	**Corregido**	Tokens firmados con ES256, `exp` de 15 min; un token forjado con el secreto anterior recibe `401`; los tokens antiguos fueron invalidados
EYM-02	IDOR en pedidos	ALTA	**Corregido**	`GET` y `PATCH` de pedidos ajenos devuelven `404`; se verificaron también direcciones y facturas
EYM-03	CORS con origen reflejado	ALTA	**Corregido**	`Origin: https://atacante.example` no recibe `Access-Control-Allow-Origin`; los orígenes legítimos funcionan
EYM-04	Clave del buscador en el bundle	MEDIA	**Corregido**	La clave anterior responde `401`; el bundle contiene solo una clave de búsqueda de solo lectura
EYM-05	Sin límite de intentos	MEDIA	**Parcialmente corregido**	Límite por IP activo (5/min, `429` con `Retry-After`); falta el contador por cuenta, por lo que un ataque distribuido desde muchas IP sigue siendo posible
EYM-06	Enumeración de usuarios	MEDIA	**Corregido**	Mensajes y códigos idénticos; diferencia de tiempo de respuesta no significativa en 50 muestras
EYM-07	Cabeceras ausentes	BAJA	**Pendiente**	Riesgo aceptado por el cliente hasta el 30/11/2026

Estado posterior al retest: riesgo general **BAJO**, con una corrección parcial (EYM-05) y un pendiente aceptado (EYM-07).

Recomendación: completar EYM-05 y EYM-07 antes del 30/11/2026. Eymia Labs verificará ambos sin costo adicional si se solicita dentro del plazo del retest.

Anexo B. Glosario para la gerencia

EJEMPLO — DATOS FICTICIOS.

TÉRMINO	SIGNIFICADO
API	La parte del sistema que entrega y recibe datos entre la tienda, la app y el servidor
CVSS	Escala internacional de 0 a 10 para medir la gravedad técnica de una vulnerabilidad
IDOR / BOLA	Falla en la que el sistema no verifica que un dato pertenezca a quien lo pide
JWT	"Pase" digital que la API entrega al usuario al iniciar sesión
CORS	Regla que decide qué otros sitios web pueden hablar con la API en nombre del usuario
Retest	Segunda revisión para confirmar que las correcciones funcionan

Fin del informe de muestra. EJEMPLO — datos ficticios. Eymia Labs · contacto@eymialabs.com