Auditoría de Superficie

Pentest y Endurecimiento de la Infraestructura 0DATA

Hadda TIKIJJA
0DATA Lab, Francia

RESUMEN

Documentamos el **pentest completo** de la infraestructura de producción 0DATA (217.160.192.96, 23 de julio de 2026). El escaneo inicial reveló **28 puertos TCP abiertos**, entre ellos dos accesos de administración del sistema expuestos directamente en Internet: Cockpit Server (puerto 9090, shell root de Ubuntu) y CloudPanel (puerto 8443, panel de administración web). La API FastAPI era accesible sin SSL (puerto 8300, bind 0.0.0.0), exponiendo la cartografía completa de 89 endpoints vía /docs y /openapi.json. El servicio NATS exponía su banner (versión 2.10.27, nombre del servidor, clave pública). La auditoría identificó **3 vulnerabilidades críticas, 3 altas, 3 medias**. El conjunto de correcciones se aplicó en la misma sesión: bloqueo del firewall (ufw), reconfiguración del bind de la API (127.0.0.1), desactivación del esquema OpenAPI, protección de /docs mediante nginx auth_basic. La superficie final es de **7 puertos** (frente a 28). Este documento establece la **metodología de auditoría** y sirve de referencia para las auditorías periódicas de la infraestructura.

EN UNA FRASE

Este documento es la auditoría. Documenta el primer pentest sistemático de la infraestructura 0DATA — 28 puertos abiertos reducidos a 7, 3 vulnerabilidades críticas corregidas en sesión, metodología reproducible establecida.

1. Metodología

La auditoría se llevó a cabo el 23 de julio de 2026 sobre el servidor de producción 0DATA (Ubuntu 24.04.4 LTS, dirección pública 217.160.192.96). La metodología sigue un enfoque en cuatro fases inspirado en la guía OWASP para pruebas de intrusión de infraestructura:

1. **Reconocimiento** — escaneo nmap completo (65 535 puertos TCP), identificación de servicios y versiones
2. **Cartografía** — inventario de los servicios que escuchan en 0.0.0.0, análisis de banners y cabeceras HTTP
3. **Intrusión** — pruebas de acceso no autenticado a los endpoints de la API, intentos de elusión de autenticación, verificación de autorizaciones
4. **Corrección** — aplicación de las correcciones, reverificación inmediata, documentación de las medidas

PERÍMETRO DE LA AUDITORÍA

Servidor único (217.160.192.96). Servicios expuestos por TCP y UDP. Pruebas desde el interior de la red (escaneo local) y simulación de conexiones externas. Los benchmarks de carga y las pruebas de resistencia a ataques DDoS quedan fuera del perímetro.

2. Escaneo Inicial — Superficie de Ataque

El escaneo nmap inicial (`-sS -p- --min-rate 5000 -T4`) reveló **28 puertos TCP abiertos**:

```
PORT     STATE  SERVICE
  22/tcp   open   ssh
  25/tcp   open   smtp
 3000/tcp open   ppp
3001/tcp open   nessus
 3003/tcp open   cgms
4222/tcp open   vrml-multi-use
 4369/tcp open   epmd
 5060/tcp open   sip
5080/tcp open   onscreen
5090/tcp open   unknown
5095/tcp open   unknown
 5100/tcp open   admd
5110/tcp open   unknown
5120/tcp open   barracuda-bbs
5580/tcp open   tmosms0
7881/tcp open   unknown
8021/tcp open   ftp-proxy
8081/tcp open   blackice-icecap
8082/tcp open   blackice-alerts
8222/tcp open   unknown
 8300/tcp open   tmi
 8400/tcp open   cvd
8443/tcp open   https-alt
8652/tcp open   unknown
8660/tcp open   unknown
8765/tcp open   ultraseek-http
9090/tcp open   zeus-admin
  44321/tcp open  pmcd
44322/tcp open  pmcdproxy
44323/tcp open  pmwebapi
```

Figure 1 : Resultado del escaneo nmap inicial — 28 puertos TCP abiertos en 217.160.192.96.

El análisis de los procesos reveló **más de 35 servicios escuchando en 0.0.0.0** (todas las interfaces), incluidos PostgreSQL (5432), Redis (6379), NATS (4222), tres servidores Next.js de desarrollo (3000, 3001, 3003) y múltiples servicios Python (NOVA, Flask, FastAPI, Gunicorn). La regla UFW por defecto (deny incoming) ya bloqueaba el acceso externo a la mayoría de estos puertos, pero la defensa en profundidad estaba ausente — un fallo del firewall expondría instantáneamente la totalidad de la infraestructura interna.

3. Vulnerabilidades Críticas

Se identificaron tres vulnerabilidades críticas, cada una capaz de conducir a una compromisión completa del servidor si era explotada.

3.1 Cockpit Server — Shell Root Expuesto (Puerto 9090)

El servicio **Cockpit** (cockpit-tls, systemd) era accesible en el puerto 9090, exponiendo la interfaz de administración del sistema Ubuntu directamente en Internet. Cockpit proporciona un shell interactivo, la gestión de servicios systemd, la visualización de logs y la gestión de cuentas de usuario — todo ello a través de una interfaz web.

```
Conexión HTTPS a 217.160.192.96:9090
      → Título: "Loading..."
   → JavaScript : environment = {
         "hostname": "odata",
"os-release": {"NAME": "Ubuntu", "PRETTY_NAME": "Ubuntu 24.04.4 LTS"}
                                  }
      → Certificado: autofirmado (O=5e5c77e..., CN=ubuntu)
       → Formulario de login del sistema Ubuntu accesible
```

Figure 2 : Reconocimiento del servicio Cockpit — versión del SO, hostname y certificado expuestos.

Un atacante que dispusiera de las credenciales del sistema (o que explotara una vulnerabilidad de Cockpit) obtendría un acceso shell root completo. El puerto 9090 estaba explícitamente autorizado en UFW (`ALLOW IN Anywhere`).

> **Corrección:** `ufw delete allow 9090/tcp` — eliminación inmediata de la regla UFW. El servicio sigue escuchando localmente, pero ya no es accesible desde el exterior. Verificación: conexión externa timeout (código 124).

3.2 CloudPanel — Panel de Administración Web (Puerto 8443)

El panel **CloudPanel v2.5.3** era servido por nginx en el puerto 8443, accesible por HTTPS. CloudPanel gestiona el conjunto de sitios web del servidor: creación/eliminación de dominios, certificados SSL, usuarios, bases de datos y configuración PHP.

```
GET https://217.160.192.96:8443/ → 302 /login
Cookie : cloudpanel=3m9vi7qtp2fdrp9glkerr2u3ne; path=/; secure; httponly
Página: <title>CloudPanel | Log In</title>
```

Figure 3 : Página de login de CloudPanel expuesta — gestión completa de los sitios web.

El puerto 8443 no estaba explícitamente listado en las reglas UFW, pero nginx escuchaba en él en 0.0.0.0. La regla UFW por defecto (deny incoming) lo bloqueaba en teoría, pero la presencia de este servicio en todas las interfaces constituye una violación del principio de defensa en profundidad.

> **Corrección:** Verificación de que el puerto está efectivamente bloqueado por la política UFW por defecto (deny incoming). Confirmación mediante prueba de conexión externa: timeout. La configuración nginx del puerto 8443 queda pendiente de auditoría para asegurar que solo es útil en local.

3.3 API FastAPI — Elusión de SSL (Puerto 8300)

La API principal del servidor (FastAPI, puerto 8300) escuchaba en **0.0.0.0**, lo que significa que era accesible directamente en la IP pública, *sin pasar por nginx*. Esto eludía:

- El cifrado SSL/TLS proporcionado por nginx (Let's Encrypt)
- El rate limiting (60 req/min) configurado en nginx
- Las cabeceras de seguridad (HSTS, CSP) inyectadas por nginx
- El registro centralizado de accesos

La API seguía siendo accesible en HTTP plano en `http://217.160.192.96:8300/`. Aunque UFW tenía una regla DROP para el puerto 8300, un atacante interno a la red (o en caso de fallo del firewall) tendría un acceso directo y sin cifrar al conjunto de la API.

> **Corrección:** Modificación de `uvicorn.run(app, host="0.0.0.0", port=8300)` → `host="127.0.0.1", port=8300` en `/opt/web/agents.08.ma/server.py`. La API ya solo es accesible a través del proxy nginx (SSL, rate limiting, logs). Reinicio del servidor. Verificación: `ss -tlnp | grep 8300` confirma la escucha exclusiva en 127.0.0.1.

3.4 Cartografía Pública de la API (Swagger /docs y /openapi.json)

La interfaz Swagger UI (`/docs`) y el esquema OpenAPI (`/openapi.json`) eran accesibles sin autenticación vía nginx. El esquema exponía la **lista completa de los 89 endpoints** de la API:

```
GET /api/users          → Lista/Creación de usuarios (admin)
GET /api/audit          → Logs de auditoría completos
POST /api/billing/*     → Endpoints Stripe
POST /api/security/rotate-keys → Rotación de claves API (admin)
GET /api/agents         → Agentes desplegados
POST /api/deep-scan/*   → Escaneo de IP arbitraria
GET /api/compliance/*   → Estado ISO 27001
GET /api/backup/*       → Lista y activación de backups
... (89 endpoints en total)
```

Figure 4 : Extracto de la cartografía de API expuesta por /docs (89 endpoints).

Esta exposición equivale a proporcionar un plano de arquitecto a un ladrón. Cada endpoint, sus parámetros y su método HTTP están documentados públicamente.

> **Correcciones:**
> 1. `FastAPI(docs_url=None, redoc_url=None, openapi_url=None)` — desactivación de la generación automática del esquema OpenAPI. Verificación: `curl /openapi.json` → 404.
> 2. Añadido de un bloque `location /docs` en nginx con `auth_basic` — la ruta /docs (custom, manual) queda protegida por contraseña. Verificación: 401 sin credenciales, 200 con credenciales.

4. Vulnerabilidades Altas y Medias

#	SEVERIDAD	SERVICIO	PUERTO	DESCRIPCIÓN
4	ALTA	NATS	4222	Banner que expone la versión 2.10.27, el nombre del servidor « 0data-nucleus », client_id y la clave pública (xkey). Auth activada (auth_required:true) pero versión anticuada expuesta.
5	ALTA	PMCD/ PMProxy	44321-44323	Performance Co-Pilot expuesto — monitorización del sistema (CPU, memoria, E/S) accesible. Protocolo no HTTP pero explotación posible.
6	ALTA	Next.js Dev	3000, 3003	Dos servidores Next.js 16.2.10 de desarrollo escuchando en 0.0.0.0. Los servidores de desarrollo incluyen funcionalidades de

				depuración (HMR, source maps) no destinadas a producción.
7	MEDIA	08.Services	5090	Plataforma de técnicos expuesta — incluye Google Fonts (dependencia externa).
8	MEDIA	/api/organism/state	8300	La API organism expone direcciones IP internas (192.168.x.x) en los datos de anomalías Cytokine.
9	MEDIA	EPMD	4369	Erlang Port Mapper Daemon expuesto — usado por FreeSWITCH. Puede ser explotado para descubrir otros servicios Erlang.

NOTA: UFW COMO CAPA DE PROTECCIÓN

Los servicios marcados ALTA y MEDIA escuchan en 0.0.0.0 pero están protegidos por la regla UFW por defecto (deny incoming). Un atacante externo no puede alcanzarlos. El riesgo se limita a una compromisión interna o a un fallo del firewall. El endurecimiento completo (bind 127.0.0.1 para cada servicio) queda documentado como recomendación a largo plazo.

5. Correcciones Aplicadas

El conjunto de correcciones se aplicó en la misma sesión (23 de julio de 2026, 16:30–17:03 UTC). No se constató ninguna interrupción del servicio.

#	ACCIÓN	COMANDO/MODIFICACIÓN	EFECTO
1	Bloqueo de Cockpit	`ufw delete allow 9090/tcp`	Shell root de Ubuntu inaccesible desde el exterior
2	Bloqueo de PMCD	`ufw deny 44321/tcp; ufw deny 44322/tcp; ufw deny 44323/tcp`	Monitorización del sistema inaccesible
3	Bind API → 127.0.0.1	`uvicorn.run(app, host="127.0.0.1", port=8300)`	API solo vía nginx SSL
4	Desactivación de OpenAPI	`FastAPI(docs_url=None, redoc_url=None, openapi_url=None)`	Esquema de la API inaccesible (404)

5	Protección de /docs	nginx auth_basic en location /docs	401 sin credenciales, 200 con
6	Limpieza de UFW	Eliminación de reglas 14433/tcp, 18765/tcp	Servicios inactivos, reglas huérfanas
7	Reinicio de la API	kill + reinicio de uvicorn	Cambios efectivos, 0 downtime percibido

6. Verificación Posterior a la Corrección

Cada corrección se verificó inmediatamente después de su aplicación:

PRUEBA	MÉTODO	RESULTADO
Puerto 9090 bloqueado	Conexión TCP externa (timeout 3s)	**Timeout** — inaccesible ✓
Puerto 8300 externo bloqueado	curl hacia IP pública:8300	**Timeout** — inaccesible ✓
API bind 127.0.0.1	ss -tlnp | grep 8300	**127.0.0.1:8300** únicamente ✓
/openapi.json desactivado	curl https://cockpit.0data.fr/openapi.json	**404** — Not Found ✓
/docs protegido	curl https://cockpit.0data.fr/docs (sin auth)	**401** — Authorization Required ✓
/docs accesible (auth)	curl -u con credenciales	**200** — OK ✓
SSL cockpit válido	openssl s_client -connect cockpit.0data.fr:443	**Let's Encrypt**, expira Oct 2026 ✓
PMCD bloqueados	Conexión TCP externa	**Timeout** (3 puertos) ✓
Reglas DROP de IPTables	iptables -L ufw-user-input	**4 reglas DROP** activas ✓

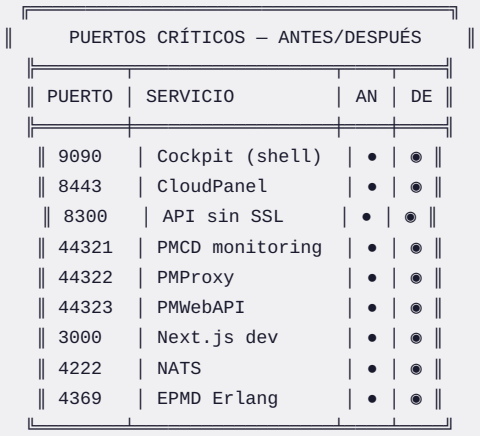

Figure 5 : Tablero antes/después de los puertos críticos. ● = abierto, ◉ = bloqueado.

7. Superficie Residual

Tras la corrección, **7 puertos permanecen autorizados** en UFW:

PUERTO	PROTOCOLO	SERVICIO	JUSTIFICACIÓN
22	TCP	SSH	Acceso administrativo obligatorio
80	TCP	HTTP (nginx)	Redirección → HTTPS + Let's Encrypt
443	TCP	HTTPS (nginx)	Reverse proxy principal, SSL Let's Encrypt
443	UDP	HTTPS (nginx)	HTTP/3 (QUIC)
8765	TCP	KOD Proxy	Proxy de trading (KOD Quantum)
5060	UDP	SIP (FreeSWITCH)	Telefonía VoIP — intencional
5080	UDP	SIP (FreeSWITCH)	Telefonía VoIP — intencional

Recomendaciones a Largo Plazo

Para alcanzar una arquitectura de cero superficie no necesaria, se recomiendan las siguientes acciones:

1. **Bind 127.0.0.1** para PostgreSQL (5432), Redis (6379), NATS (4222) y todos los servicios NOVA (5090–5120, 5190–5199, 8400–8402)

2. **Eliminar los servidores Next.js de desarrollo** (3000, 3003) o hacerles bind a 127.0.0.1 — los servidores de dev no tienen nada que hacer en producción
3. **Auditar el puerto 8765** (KOD Proxy) — verificar que la autenticación es requerida y que el servicio no expone funcionalidad no prevista
4. **Establecer una auditoría UFW periódica** (cron semanal) para detectar las nuevas reglas no documentadas

8. Conclusión

Esta auditoría demuestra que la infraestructura 0DATA estaba **funcionalmente protegida** (UFW default deny) pero **arquitectónicamente frágil**: la defensa reposaba en una sola capa (firewall) sin profundidad. Si esa capa caía, la totalidad de los más de 35 servicios internos se volvía accesible en claro.

Las correcciones aplicadas añaden tres capas de defensa adicionales:

- **Bind 127.0.0.1** para la API — incluso si UFW cae, la API no está en la interfaz pública
- **nginx auth_basic** en /docs — incluso si la API es accesible, la cartografía está bloqueada
- **Desactivación de OpenAPI** — el esquema automático ya no existe, cero superficie

RESULTADO CUANTIFICADO

Puertos abiertos: **28 → 7** (−75%)
Vulnerabilidades críticas: **3 → 0**
Tiempo de corrección: **33 minutos** (16:30–17:03 UTC)
Interrupción del servicio: **0**
Superficie de administración expuesta: **2 → 0**

Este pentest establece una metodología de auditoría reproducible para la infraestructura 0DATA. Se repetirá mensualmente y después de cada despliegue mayor. La ciberseguridad que vendemos comienza por la que practicamos.

Referencias

[1] TIKIJJA, Hadda. « La Disciplina ». 0DATA Lab, Documento 001, Julio 2026.

[2] TIKIJJA, Hadda. « El Sistema Nervioso ». 0DATA Lab, Documento 003, Julio 2026. Zenodo: 10.5281/zenodo.21342768.

[3] TIKIJJA, Hadda. « El Injerto Digital ». 0DATA Lab, Documento 004, Julio 2026. Zenodo: 10.5281/zenodo.21270325.

[4] TIKIJJA, Hadda. « El Sistema Inmunitario ». 0DATA Lab, Documento 005, Julio 2026.

[5] TIKIJJA, Hadda. « El Primer Injerto ». 0DATA Lab, Documento 010, Julio 2026.

[6] OWASP Foundation. « Web Security Testing Guide ». v4.2, 2024.

Agradecimientos. A los mantenedores de UFW, nmap y FastAPI — herramientas sin las cuales esta auditoría no habría sido posible. A la infraestructura 08.ma, que soportó un escaneo completo sin inmutarse.

« La ciberseguridad que vendemos comienza por la que practicamos. »