ECOM EXPERTS

Sec Admin (ecom_secadmin)

Informe para el proveedor de hosting - version 1.7.0

Manual completo · 12/10/2026

Contenido

1. Por qué recibís este documento

2. Qué hace, en una página

3. Ficheros y permisos que necesita

Base de datos

4. Tráfico de red

4.1 Saliente

4.2 Entrante

5. Lo que puede llamaros la atención en los registros

5.1 Respuestas 403 en el back-office

5.2 logs/blocked.log creciendo

5.3 Un .php servible dentro de modules/

5.4 Correos salientes

6. Lo que el módulo NO hace

6 bis. No depende de Apache ni del .htaccess

7. Qué necesitamos del hosting (si venís de un incidente)

8. Si tenéis que desactivarlo de urgencia

9. Contacto

Qué es: un módulo de PrestaShop que restringe el acceso al **back-office** de la tienda. Nada más. **Versión:** 1.2.0 · **PrestaShop:** 1.7.8, 8.x y 9.x · **PHP:** 7.4 a 8.3 **Autor:** Ecom Experts · **Licencia:** AFL-3.0

---

1. Por qué recibís este documento

El módulo hace tres cosas que, vistas desde fuera y sin contexto, **pueden parecer sospechosas** en una revisión de seguridad:

1. Devuelve **HTTP 403** a peticiones que llegan al back-office.
2. Hace **una petición saliente** a un servicio de geolocalización de IPs.
3. Tiene **un fichero PHP accesible por URL** dentro de `modules/`.

Las tres son el funcionamiento normal y están explicadas abajo, con el fichero y la línea. Este documento está para que no haya que adivinarlo.

El módulo no ejecuta programas, no abre sockets, no descarga ni escribe código, y no toca ningún fichero del núcleo de PrestaShop. El apartado 6 lo detalla.

---

2. Qué hace, en una página

Protege **solo el back-office** (`/admin.../`). El front —la tienda, el catálogo, el carrito, el pago— **no se toca en absoluto**: ni una capa del módulo actúa sobre él.

Capa	Qué corta
Geobloqueo	Peticiones al back-office desde fuera de la UE-27
Lista negra	IPs y rangos que el comercio marca
Programas automáticos	Clientes que no son un navegador (`python-requests` , `curl` , `sqlmap` ...) **solo en el back-office**
Lista blanca de módulos	Entrar al gestor de módulos exige IP autorizada
Huella de sesión	Cierra la sesión si cambia el navegador a mitad (cookie robada)
Cookie de confianza	Firmada con HMAC, atada al empleado y a la IP
Aviso de acceso sospechoso	**No corta nada**: solo manda un aviso cuando una sesión parece robada

La última capa es **solo observación**. Distingue si una sesión nació de un login con contraseña o apareció ya validada (cookie reutilizada) y avisa únicamente con pruebas fuertes: la misma sesión en dos países en pocos minutos, una sesión reutilizada desde una IP de datacenter, o una petición autenticada hecha con un

programa en vez de un navegador. No bloquea, no cierra sesiones y no modifica la petición. Guarda el identificador de sesión **siempre hasheado** y no lo incluye en los avisos.

El **Webservice** (`/api/`) **se vigila pero NUNCA se bloquea**: solo deja constancia y manda un aviso. Decisión expresa del comercio, para no cortar sus integraciones.

---

3. Ficheros y permisos que necesita

Todo vive dentro de `modules/ecom_secadmin/` . **No escribe fuera de su propia carpeta.**

Ruta	Para qué	Permiso
`modules/ecom_secadmin/logs/blocked.log`	Una línea por bloqueo, en formato para fail2ban	**escritura**
`modules/ecom_secadmin/.htaccess`	Reabre dos ficheros que PS9 bloquea (ver 5.3)	lectura
`modules/ecom_secadmin/.disable`	Fichero de rescate, **lo crea el comercio a mano** por FTP	lectura

Es el único permiso de escritura que necesita: la carpeta `logs/` del propio módulo. No necesita escribir en `/var` , `/tmp` , `config/` , `override/` ni en la raíz.

Base de datos

Cinco tablas con el prefijo de la tienda:

```
<prefijo>ecomsec_log              registro de accesos y bloqueos
<prefijo>ecomsec_whitelist        IPs autorizadas
<prefijo>ecomsec_blacklist        IPs bloqueadas
<prefijo>ecomsec_geo_cache        caché IP → país (24 h)
<prefijo>ecomsec_notify_throttle  control de envío de correos
```

Crecen poco: el registro se purga solo una vez al día (90 días por defecto) y los accesos de una sesión se anotan una sola vez, no una por clic.

---

4. Tráfico de red

4.1 Saliente

Una sola llamada, y solo para traducir una IP a un país:

Destino	Cuándo	Qué envía
`http://ip-api.com/json/<ip>` (por defecto)	Primera vez que se ve una IP en el back-office	Solo la IP a consultar
`https://ipapi.co/` o `https://ipinfo.io/`	Solo si el comercio elige ese proveedor	Ídem

- **Se cachea 24 h por IP** (`ecomsec_geo_cache`), así que una misma IP no genera más de una consulta al día. En una tienda normal son **unas pocas llamadas diarias**.
- Si el destino no responde, el módulo no se cuelga: tiene tiempo de espera y actúa según lo configurado.
- **No envía datos de clientes, ni de pedidos, ni de la tienda.** Solo la IP.

Si el comercio activa los avisos por WhatsApp, hay además llamadas a su proveedor (`https://mensajero.gmartos.es` o `https://waba-v2.360dialog.io`), **solo cuando se manda un aviso**, y una consulta a `http://ip-api.com/json/<ip>?fields=status,as,hosting,proxy` para saber si una IP es de un datacenter — cacheada también 24 h por IP, y desactivable desde la pantalla del módulo.

Hay además una comprobación de versión del módulo contra `https://modules.gmartos.es/api.php`, **una vez al día como mucho**, que envía el nombre y la versión del módulo y la URL de la tienda. Se puede bloquear sin que el módulo deje de proteger nada.

4.2 Entrante

El módulo no abre ningún puerto ni escucha nada. Solo responde a peticiones HTTP que ya iban a la tienda.

---

5. Lo que puede llamaros la atención en los registros

5.1 Respuestas 403 en el back-office

Son del módulo. El cuerpo es una página pequeña (≈350 bytes) que dice «403 · Acceso restringido». Si veis muchos, es que **está trabajando**: alguien está intentando entrar desde una IP o un cliente no autorizados.

```
header('HTTP/1.1 403 Forbidden');          ecom_secadmin.php:860
```

5.2 `logs/blocked.log` creciendo

Una línea por bloqueo, pensada para que **fail2ban** pueda banear a nivel de cortafuegos:

```
[2026-10-05 18:20:30] ecom_secadmin DENY GEO ip=45.155.205.233 country=RU
```

Filtro sugerido:

```
[Definition]
failregex = ^.*ecom_secadmin DENY \S+ ip=<HOST>\b
```

Si queréis, podéis montar la jaula vosotros; el módulo funciona igual sin ella, pero con fail2ban el atacante deja de llegar al servidor en vez de recibir un 403.

5.3 Un `.php` servible dentro de `modules/`

`modules/ecom_secadmin/whitelist_add.php` es el enlace de «autorizar esta IP» que llega por correo al dueño de la tienda cuando se bloquea un acceso.

- Lleva una **firma HMAC-SHA256** de la IP y una caducidad; sin firma válida responde 403.
- **Caduca a las 24 horas.**
- Lo único que hace es insertar una fila en `ecomsec_whitelist`. No ejecuta nada.

PrestaShop 9 deniega por defecto todos los `.php` de `modules/` (su propio `modules/.htaccess` incluye `\.php`; en 8.2 no lo incluía). Por eso el módulo trae un `.htaccess` en su carpeta que **vuelve a cerrar todo y reabre solo ese fichero y el cron**. No toca el `.htaccess` del núcleo ni el de la raíz.

5.4 Correos salientes

Avisos al dueño de la tienda cuando algo se bloquea, por el mismo sistema de correo que ya usa PrestaShop (no abre su propio SMTP).

Tope de 6 correos por hora en total, configurable, y `0` los desactiva del todo. Al llegar al tope manda **un único resumen** en vez de uno por IP: con una botnet de mil IPs salen 7 correos, no mil.

---

6. Lo que el módulo NO hace

Verificado sobre el código, no es una declaración de intenciones:

- **No ejecuta programas.** Cero `exec`, `shell_exec`, `system`, `passthru`, `popen`, `proc_open`. Comprobadlo: `grep -rn "eval(\|exec(\|shell_exec\|passthru\|system(\|popen(\|proc_open" modules/ecom_secadmin/` El único `base64_decode` del módulo está en `classes/EcomSecadminCookie.php:126` y sirve para leer su propia cookie firmada.
- **No descarga ni escribe código.** Nada que baje un fichero y lo incluya o lo ejecute.
- **No abre sockets** ni conexiones salientes que no sean la consulta de geolocalización del apartado 4.1.
- **No crea tareas programadas.** Trae un `cron.php` opcional que el comercio puede programar si quiere, protegido con un token; si no lo programa, la limpieza se hace sola en la primera visita del día al back-office.
- **No modifica el núcleo de PrestaShop**, ni usa `override/`, ni toca `config/`.
- **No toca el front.** Ninguna capa actúa sobre la tienda, el carrito ni el pago.
- **No lee ni envía datos de clientes ni de pedidos.**

---

6 bis. No depende de Apache ni del `.htaccess`

Esto importa si delante hay Nginx, o si vuestro panel expone Apache por otro puerto.

El `.htaccess` de la carpeta del módulo sirve **solo** para cerrar sus dos PHP sueltos (`whitelist_add.php` y `cron.php`). Todo lo demás —el bloqueo de la zona de módulos, el geobloqueo, los programas automatizados, la lista negra, la huella de sesión y el aviso de acceso sospechoso— **es PHP** y se ejecuta dentro de un hook de PrestaShop (`actionDispatcherBefore`). Por tanto:

Montaje	¿Protege?
Apache con `.htaccess`	Sí
Apache con `AllowOverride None`	Sí (se pierde solo el cierre de esos dos PHP)
Nginx delante, PHP-FPM detrás	Sí
Nginx delante y alguien entra a Apache directo por 7080 / 7081	**Sí**

Ese último caso es el del incidente del 02/10/2026: la petición aparecía en el `error_log` de Apache y **no** en el `proxy_access_ssl_log` de Nginx, es decir, alguien llegaba al Apache de Plesk saltándose el Nginx. Como el bloqueo lo hace PHP, da igual por dónde entre.

Medido el 12/10/2026: las tiendas de prueba del módulo las sirve `php -S` , que **no lee** `.htaccess` **en absoluto**, y aun así el `POST` a `/improve/modules/import` con sesión y token válidos se corta y no aparece nada nuevo en `/modules/` .

Si de verdad queréis cerrar también los dos PHP sueltos con Nginx, al final del `.htaccess` del módulo está el `location` equivalente.

---

7. Qué necesitamos del hosting (si venís de un incidente)

Este módulo **reduce la superficie de ataque del back-office, pero no limpia un servidor ya comprometido.** Si hubo intrusión, hay cosas que solo podéis hacer vosotros:

1. **Matar los procesos sospechosos** del usuario de la suscripción. Un 403 de PHP no puede cortar la conexión saliente de un proceso que ya está corriendo.
2. **Bloquear el tráfico saliente** hacia la infraestructura del atacante. El `.htaccess` filtra entrada, no salida.
3. **Revisar las cuentas FTP/SFTP** y rotar sus contraseñas.
4. **Confirmar que no quedan ficheros ajenos** en el árbol (`.php` y binarios ELF nuevos o modificados fuera de las rutas conocidas).

Y una advertencia importante por si estáis endureciendo la configuración de PHP:

> `proc_open` **NO se puede deshabilitar en una tienda PrestaShop.** SwiftMailer lo usa para el transporte *sendmail*; deshabilitarlo tumba **todos los correos** y, como el aviso de pedido va en un hook de validación, **rompe la finalización de pedidos** con un *Fatal error: Call to undefined function proc_open()*.

> `exec`, `system`, `passthru`, `shell_exec` y `popen` normalmente sí se pueden. `proc_open` solo si antes se pasa el correo a SMTP y se prueba un envío real.

---

8. Si tenéis que desactivarlo de urgencia

Cuatro formas, de menos a más drástica. **Ninguna requiere tocar el código.**

1. **Fichero de rescate:** crear un fichero vacío llamado `.disable` en `modules/ecom_secadmin/`. Apaga el geobloqueo, la lista negra y el filtro de programas automáticos. Borrarlo lo reactiva.
2. **Autorizar una IP** por base de datos: `sql INSERT INTO <prefijo>ecomsec_whitelist (cidr, label, active, created_at) VALUES ('LA.IP.AQUI', 'Rescate hosting', 1, NOW());`
3. **Apagar el módulo** (con el prefijo real de la tienda, normalmente `ps_`): `sql UPDATE ps_module SET active = 0 WHERE name = 'ecom_secadmin';`
4. **Desinstalarlo** desde el back-office. Borra sus cinco tablas y sus entradas de menú.

Tras cualquiera de las dos últimas hay que **vaciar la caché** de PrestaShop.

---

9. Contacto

Ecom Experts — ecomyseo@gmail.com

Si algo de lo que veis en los registros no encaja con este documento, escribidnos antes de dar nada por hecho: puede ser un dato útil.