Mejoras sobre Directadmin

A veces hay que tunear o mejorar el software contenido y hay que hacerlo basándose en el paradigma del panel.

- Whitelist dinámica de rspamd via email en Directadmin
- Tuning de rspamd en DirectAdmin — Guía post-instalación
- Bug template rspamd_settings.conf — Whitelist per-user ineficaz
- CSF + ModSecurity en nginx: MODSEC_LOG y regex custom
- Arquitectura antispam en DirectAdmin: Exim ESF + rspamd
- DirectAdmin startips — supervisor / auto-restart de IPs virtuales

Whitelist dinámica de rspamd via email en Directadmin

Qué hace

Permite a cualquier usuario autenticado del servidor gestionar la whitelist de rspamd enviando un email. No requiere acceso al panel ni SSH.

Enviar un email a `whitelist@<cualquier-dominio-del-servidor>` con la acción en el asunto:

Asunto del email	Acción
`add ejemplo.com`	Añade el dominio completo ejemplo.com — todos los correos de @ejemplo.com pasan
`add user@ejemplo.com`	Añade solo esa dirección concreta — otros de @ejemplo.com siguen filtrados
`ejemplo.com`	Añade ejemplo.com (add implícito)
`del ejemplo.com`	Elimina ejemplo.com de la whitelist
`del user@ejemplo.com`	Elimina esa dirección de la whitelist
`remove ejemplo.com`	Igual que `del` (también `delete`, `rm`)
`list`	Responde con la whitelist completa actual

El sistema responde por email confirmando cada operación: dominio añadido, eliminado, ya existente, bloqueado, o error de formato.

El email debe enviarse desde un cliente de correo autenticado (Thunderbird, Outlook, webmail). Correos no autenticados son ignorados silenciosamente.

Dominios vs direcciones

El sistema distingue entre dos tipos de whitelist:

- **Dominio completo** (`add ejemplo.com`): todos los correos desde cualquier cuenta @ejemplo.com pasan sin penalización
- **Dirección concreta** (`add user@ejemplo.com`): solo los correos de esa cuenta pasan; el resto de @ejemplo.com sigue siendo evaluado normalmente

Esto es útil para proveedores masivos (Gmail, Outlook, Proton, etc.) donde no queremos whitelistear todo el dominio sino solo una cuenta concreta de confianza.

Restricciones de seguridad

- **Solo SMTP AUTH:** si el email no viene de un usuario autenticado del servidor, se descarta
- **Mass providers bloqueados como dominio:** no se puede whitelistear gmail.com, outlook.com, proton.me, etc. como dominio completo — pero SÍ se puede whitelistear una dirección concreta como `contacto@proton.me`
- **Límite:** máximo 500 entradas por servidor (dominios + direcciones combinados)
- **Validación estricta:** el dominio/email debe tener formato válido
- **Respuesta informativa:** el sistema responde indicando exactamente qué hizo o por qué rechazó la petición

Proveedores masivos bloqueados

gmail.com, googlemail.com, yahoo.com, yahoo.es, yahoo.fr, hotmail.com, hotmail.es, outlook.com, outlook.es, live.com, msn.com, aol.com, mail.ru, yandex.ru, yandex.com, protonmail.com, proton.me, icloud.com, me.com, mac.com, gmx.com, gmx.es, gmx.de, zoho.com, tutanota.com, tuta.com, tuta.io, fastmail.com, mail.com, email.com, web.de, freenet.de, t-online.de, libero.it, virgilio.it, laposte.net, free.fr, orange.fr, 163.com, 126.com, qq.com

Si se intenta añadir uno de estos como dominio, el sistema lo rechaza y sugiere usar la dirección completa (`add usuario@proveedor.com`).

Cómo funciona internamente

```
Email autenticado → whitelist@dominio.com
        ↓
Exim router (whitelist_pipe_router)
  · condición: $authenticated_id no vacío
  · domains = +local_domains (solo dominios hospedados)
        ↓
Exim transport → pipe a /usr/local/bin/rspamd_whitelist.sh
  · pasa AUTH_USER via variable de entorno
        ↓
Script: parsea Subject, valida, determina tipo (dominio/dirección)
        ↓
```

```
  Si dominio → /etc/rspamd/local.d/maps/whitelist_dynamic.map
  Si dirección → /etc/rspamd/local.d/maps/whitelist_dynamic_addr.map
        ↓
rspamd recarga automáticamente (~10 segundos)
        ↓
  Multimap WHITELIST_FROM_DYNAMIC (dominios, score -100.0)
  Multimap WHITELIST_ADDR_DYNAMIC (direcciones, score -100.0)
        ↓
Respuesta por email al usuario confirmando la operación
```

Ficheros involucrados

Fichero	Propósito	Persistencia
`/usr/local/bin/rspamd_whitelist.sh`	Script principal	Manual (no gestionado por DA)
`/etc/exim.routers.pre.conf`	Router Exim custom	Sobrevive `rewrite_confs` (include en template DA)
`/etc/exim.transports.pre.conf`	Transport Exim custom	Sobrevive `rewrite_confs` (include en template DA)
`/etc/rspamd/local.d/multimap.conf`	Definición de los símbolos WHITELIST_*_DYNAMIC	`local.d/` = override oficial de rspamd
`/etc/rspamd/local.d/maps/whitelist_dynamic.map`	Lista de dominios (uno por línea)	Fichero de datos, no config
`/etc/rspamd/local.d/maps/whitelist_dynamic_addr.map`	Lista de direcciones exactas (una por línea)	Fichero de datos, no config
`/var/log/rspamd_whitelist.log`	Log de operaciones	Fichero de log

Persistencia ante rebuilds de DA

Los ficheros `.pre.conf` son puntos de extensión del template de Exim en DirectAdmin. El template fuente (`custombuild/configure/exim/exim.conf`) contiene:

```
.include_if_exists /etc/exim.routers.pre.conf     # línea 519
.include_if_exists /etc/exim.transports.pre.conf   # línea 773
```

Estas directivas `.include_if_exists` están en el template, no solo en el fichero generado. Esto significa que:

- `build rewrite_confs` regenera `/etc/exim.conf` pero **no toca** los `.pre.conf`
- `build exim_conf` idem

- `build exim` (rebuild completo del binario) idem

Los ficheros `.pre.conf` son del usuario, no de DA. Son el único mecanismo ligero para añadir routers y transports custom sin mantener una copia completa de exim.conf.

Para rspamd, `local.d/` es el mecanismo oficial de override. DA no gestiona estos ficheros.

Instalación en un servidor nuevo

Prerequisitos

- DirectAdmin con Exim y rspamd
- rspamd con multimap configurado (al menos `local.d/multimap.conf` existente)

Pasos

1. Script:

```
# Subir rspamd_whitelist.sh al servidor
scp rspamd_whitelist.sh root@servidor:/usr/local/bin/
chmod +x /usr/local/bin/rspamd_whitelist.sh
```

2. Map files y log:

```
# Map de dominios
touch /etc/rspamd/local.d/maps/whitelist_dynamic.map
chown root:mail /etc/rspamd/local.d/maps/whitelist_dynamic.map
chmod 664 /etc/rspamd/local.d/maps/whitelist_dynamic.map

# Map de direcciones
touch /etc/rspamd/local.d/maps/whitelist_dynamic_addr.map
chown root:mail /etc/rspamd/local.d/maps/whitelist_dynamic_addr.map
chmod 664 /etc/rspamd/local.d/maps/whitelist_dynamic_addr.map

# Log
touch /var/log/rspamd_whitelist.log
chown mail:mail /var/log/rspamd_whitelist.log
chmod 644 /var/log/rspamd_whitelist.log
```

3. Exim router (prepend a `/etc/exim.routers.pre.conf` — ANTES de cualquier otro router):

```
# Dynamic whitelist via email
whitelist_pipe_router:
  driver = accept
  local_parts = whitelist
  domains = +local_domains
  condition = ${if !eq{$authenticated_id}{}}
  transport = whitelist_pipe_transport
```

4. Exim transport (crear o añadir a `/etc/exim.transports.pre.conf`):

```
# Dynamic whitelist pipe transport
whitelist_pipe_transport:
  driver = pipe
  command = /usr/local/bin/rspamd_whitelist.sh
  user = mail
  environment = AUTH_USER=$authenticated_id
```

5. rspamd multimap (añadir a `/etc/rspamd/local.d/multimap.conf`):

```
WHITELIST_FROM_DYNAMIC {
    type = "from";
    filter = "email:domain";
    map = "/etc/rspamd/local.d/maps/whitelist_dynamic.map";
    symbol = "WHITELIST_FROM_DYNAMIC";
    score = -100.0;
    description = "Dynamic whitelist — added via email by authenticated users";
}

WHITELIST_ADDR_DYNAMIC {
    type = "from";
    filter = "email:addr";
    map = "/etc/rspamd/local.d/maps/whitelist_dynamic_addr.map";
    symbol = "WHITELIST_ADDR_DYNAMIC";
    score = -100.0;
    description = "Dynamic whitelist (address) — specific sender addresses added via email";
}
```

6. Verificar y reiniciar:

```
exim -bV                    # Syntax check Exim
rspamadm configtest         # Syntax check rspamd
systemctl restart exim
systemctl restart rspamd
systemctl is-active exim rspamd
```

Verificación

```
# Comprobar que el router es el primero
exim -bP router_list | head -5

# Comprobar que rspamd carga los símbolos
rspamadm configdump multimap | grep -A3 "WHITELIST.*DYNAMIC"

# Test de routing (debe resolver al pipe transport)
exim -bt whitelist@undominiodel.servidor
```

Operaciones de mantenimiento

Ver whitelist actual:

```
echo "=== Dominios ===" && cat /etc/rspamd/local.d/maps/whitelist_dynamic.map
echo "=== Direcciones ===" && cat /etc/rspamd/local.d/maps/whitelist_dynamic_addr.map
```

Ver log de operaciones:

```
cat /var/log/rspamd_whitelist.log
```

Añadir/eliminar manualmente:

```
# Añadir dominio
echo "dominio.com" >> /etc/rspamd/local.d/maps/whitelist_dynamic.map

# Añadir dirección
echo "user@dominio.com" >> /etc/rspamd/local.d/maps/whitelist_dynamic_addr.map

# Eliminar (dominios)
sed -i '/^dominio\.com$/d' /etc/rspamd/local.d/maps/whitelist_dynamic.map
```

```
# Eliminar (direcciones)
sed -i '/^user@dominio\.com$/d' /etc/rspamd/local.d/maps/whitelist_dynamic_addr.map
```

rspamd recarga el map file automáticamente en ~10 segundos.

Vaciar whitelist:

```
> /etc/rspamd/local.d/maps/whitelist_dynamic.map
> /etc/rspamd/local.d/maps/whitelist_dynamic_addr.map
```

Trampas conocidas

- **Lock file:** NO usar `/var/lock/` ni `/run/lock/` — están montados con `noexec` y bash no puede abrir file descriptors ahí. El script usa el LOG file como target de `flock` (advisory lock, no interfiere con appends)
- **Permisos map files:** deben ser `root:mail 664`. Si se crean como root durante tests, el transport Exim (que corre como `user = mail`) no podrá escribir. Verificar con `ls -la /etc/rspamd/local.d/maps/whitelist_dynamic*.map`
- **El directorio `/etc/rspamd/local.d/maps/` es `root:root 755`:** mail no puede crear ficheros nuevos ahí. El script escribe directamente al map file (que sí tiene permiso de grupo). La operación `del` usa redirección en memoria en vez de fichero temporal
- **Tests manuales como root:** crean lock/map/log files con ownership `root:root`. Tras tests manuales, verificar y corregir permisos: `chown root:mail /etc/rspamd/local.d/maps/whitelist_dynamic*.map`

Servidores desplegados

Servidor	Fecha	Estado
kvm456	2026-03-11	Activo, probado (v3: dominios + direcciones + respuesta email)
srv120	2026-03-11	Activo
srv121	2026-03-11	Activo
dar	2026-03-11	Activo

No aplica: titrit (no recibe correo), amazzal (no recibe correo)

Tuning de rspamd en DirectAdmin — Guía post-instalación

Por qué hace falta

DirectAdmin instala rspamd via CustomBuild pero lo entrega **sin configuración operativa**:

- **Bayes:** backend SQLite (lento, sin concurrencia), `per_user=true` (cada usuario necesita 200+ muestras propias — inalcanzable en hosting compartido), sin autolearn, sin entrenamiento
- **Thresholds:** defaults muy permisivos (reject=15, add_header=6)
- **ESF (Easy Spam Fighter):** otorga -60 puntos acumulados a cualquier correo con SPF+DKIM+rDNS válidos — blinda spam reenviado via Google Groups
- **Sin phishing.conf:** OpenPhish desactivado por defecto desde rspamd 3.14.3
- **Sin url_suspect.conf:** patrones de URL ofuscada pueden generar falsos positivos en footers HTML

Resultado: rspamd funciona solo con reglas estáticas, Bayes nunca contribuye, y un spam con autenticación válida llega al INBOX.

Ficheros a crear

La receta completa son **8 ficheros** en `/etc/rspamd/local.d/` (que DA nunca toca) + 1 en ESF:

1. `/etc/rspamd/local.d/actions.conf` — Thresholds globales

```
# Thresholds globales rspamd
# reject = null → NUNCA rechazar correo (solo marcar)
# add_header = 5 → marca como spam a partir de score 5

reject = null;
add_header = 5;
```

```
rewrite_subject = 8;
greylist = null;
```

Decisión clave: `reject = null`. NUNCA rechazar correo por score rspamd en hosting compartido. El riesgo de rechazar un falso positivo legítimo es mayor que el coste de marcar. El usuario puede revisar su carpeta spam.

2. `/etc/rspamd/local.d/classifier-bayes.conf` — Bayes con Redis

```
# Bayes con Redis — obligatorio para rendimiento y concurrencia
# per_user = false: pool global, obligatorio para <50 usuarios

backend = "redis";
autolearn = true;

autolearn {
    spam_threshold = 8.0;
    ham_threshold = -1.0;
    min_balance = 0.9;
    min_learns = 100;
}

per_user = false;

# Conexión Redis — DA usa socket, NO localhost:6379
servers = "/var/lib/rspamd/.redis/redis.sock";
```

Trampas:

- **`per_user = false` es obligatorio** para servidores <50 usuarios. Con `per_user=true`, cada usuario necesita 100+ spam y 100+ ham propios — nunca se alcanza
- **Redis en DA no es el estándar.** El servicio se llama `redis-rspamd.service`, no `redis-server`. El socket está en `/var/lib/rspamd/.redis/redis.sock`
- **Verificar Redis:** `systemctl is-active redis-rspamd.service`

3. `/etc/rspamd/local.d/groups-override.conf` — Neutralizar MAILLIST

```
# Neutralizar símbolo MAILLIST — beneficia spam via Google Groups
symbols {
    "MAILLIST" {
        weight = 0.0;
        description = "Neutralized - mailing lists benefit spam forwarding";
    }
}
```

4. `/etc/rspamd/local.d/multimap.conf` — Penalizaciones y whitelists

```
# Penalizar mailing lists de dominios sospechosos
FOREIGN_MAILLIST {
    type = "header";
    header = "List-Id";
    map = "/etc/rspamd/local.d/maps/foreign_maillist_domains.map";
    regexp = true;
    score = 4.0;
    description = "Foreign mailing list domain in List-Id";
}

GOOGLE_GROUPS_FOREIGN {
    type = "header";
    header = "X-Beenthere";
    map = "/etc/rspamd/local.d/maps/foreign_maillist_domains.map";
    regexp = true;
    score = 3.0;
    description = "Google Groups with foreign domain in X-Beenthere";
}

# Whitelist de dominios confianza
WHITELIST_FROM {
    type = "from";
    filter = "email:domain";
    map = "/etc/rspamd/local.d/maps/whitelist_from.map";
    symbol = "WHITELIST_FROM_TRUSTED";
    score = -100.0;
    description = "Whitelisted trusted sender — domain match";
```

```
}
```

Trampas multimap:

- `filter = "email:domain"` **es OBLIGATORIO** para whitelists por dominio. Sin él, rspamd matchea la dirección completa contra el literal del mapa — nunca coincide
- **Dominios sin `@` en el mapa** — `castris.com`, NO `@castris.com`
- **NUNCA usar `prefilter=true` + `action="no action"` como whitelist** — no funciona de forma fiable. Usar `score = -100.0` es la forma correcta
- `regexp:///path` como protocolo de mapa no funciona — usar `map = "/path"; regexp = true;` como campos separados

5. `/etc/rspamd/local.d/maps/foreign_maillist_domains.map`

```
# Dominios de mailing lists sospechosos (regex)
/go1001000\.com/
/googlegroups\.com/
```

Añadir nuevos dominios según aparezcan en spam. rspamd recarga mapas automáticamente (~10s).

6. `/etc/rspamd/local.d/maps/whitelist_from.map`

```
# Dominios de confianza (uno por línea, sin @)
castris.com
aichadigital.es
tabratino.com
```

7. `/etc/rspamd/local.d/phishing.conf` — Activar OpenPhish

```
# OpenPhish desactivado por defecto desde rspamd 3.14.3
# Activar explícitamente para detección de phishing
openphish_enabled = true;
```

8. `/etc/rspamd/local.d/url_suspect.conf` — Desactivar word_dot

```
# El patrón word_dot genera falsos positivos en footers HTML
# (ej: "reservados. skyp" → URL_OBFUSCATED_TEXT score 9.0)
# Afecta correos WHMCS, PHPMailer, cualquier footer con "frase. Dominio"
checks {
    obfuscated_text {
        patterns_enabled {
            word_dot = false;
        }
    }
}
```

Contexto: `URL_OBFUSCATED_TEXT` es un prefilter con score 5.0 hardcoded en `url_suspect.lua`. Ni `groups-override.conf` ni `override.d/` pueden cambiar el peso. La única solución es desactivar el patrón causante.

9. `/etc/exim.easy_spam_fighter/variables.conf.custom` — Reducir scores ESF

```
# Reducir scores ESF de -60 acumulado a -15
# El default blinda spam con autenticación válida (SPF+DKIM+rDNS = -60)
EASY_SPF_PASS == -5
EASY_DKIM_PASS == -5
EASY_FORWARD_CONFIRMED_RDNS == -5

# No rechazar correo por score ESF (nunca)
EASY_HIGH_SCORE_DROP == 9999

# Escanear correos hasta 15MB (default 200K omite adjuntos)
EASY_SPAMASSASSIN_MAX_SIZE == 15M
```

```
# Penalizaciones reducidas (evitar falsos positivos por forwarding)
EASY_DKIM_FAIL == 0
EASY_NO_REVERSE_IP == 30
EASY_SPF_FAIL == 50
EASY_SPF_SOFT_FAIL == 10
```

Trampas ESF:

- **Usar `==` para redefinir macros**, no `=`. Con `=` simple, Exim falla: "macro already defined (use ==)"
- **`variables.conf.custom` se incluye via `.include_if_exists`** al final de `variables.conf`. DA regenera `variables.conf` pero NUNCA toca el `.custom`

DNS resolvers — CRÍTICO para servidores de correo

rspamd consulta listas de reputación (Spamhaus ZEN, SURBL, URIBL) via DNS. Si el resolver está bloqueado, todas estas listas devuelven "not found" y el spam pasa sin penalización.

Resolvers válidos para DNSBL:

Resolver	IP	Spamhaus
Verisign	64.6.64.6	
Neustar	199.85.126.10	
Neustar Ultra	156.154.70.2	

NUNCA usar en servidores de correo:

Resolver	IP	Spamhaus
Google	8.8.8.8	Bloqueado
Cloudflare	1.1.1.1	Bloqueado
Quad9	9.9.9.9	Bloqueado
OVH	213.186.33.99	Bloqueado

Verificación:

```
# Debe devolver 127.0.0.2, 127.0.0.4 o 127.0.0.10
dig +short 2.0.0.127.zen.spamhaus.org
```

```
# Si devuelve NXDOMAIN o 127.255.255.254 → resolver bloqueado
```

Síntomas de DNSBL ciegas en rspamd: símbolos `DBL_BLOCKED_OPENRESOLVER`, `URIBL_BLOCKED`, `SURBL_BLOCKED` en las cabeceras X-Spamd-Result.

Cambiar DNS en Ubuntu (netplan):

```
# Editar /etc/netplan/*.yaml → nameservers: [64.6.64.6, 199.85.126.10, 156.154.70.2]
netplan apply
systemctl restart systemd-networkd
systemctl restart systemd-resolved
# Verificar: dig +short 2.0.0.127.zen.spamhaus.org
```

Atención: `netplan apply` puede no aplicar DNS — requiere reiniciar `systemd-networkd` y `systemd-resolved` por separado.

Entrenamiento masivo de Bayes

Sin entrenamiento, Bayes no contribuye al scoring (necesita mínimo 200 muestras de cada clase).

```
# SPAM — desde carpetas spam de todos los usuarios
# IMPORTANTE: usar SIEMPRE controller socket, NO worker 11333
find /home/*/Maildir/.INBOX.spam/cur/ -type f 2>/dev/null | head -500 | \
  xargs -I{} rspamc --connect /var/run/rspamd/rspamd_controller.sock learn_spam {}

find /home/*/Maildir/.Junk/cur/ -type f 2>/dev/null | head -500 | \
  xargs -I{} rspamc --connect /var/run/rspamd/rspamd_controller.sock learn_spam {}

# HAM — desde INBOX y Sent de todos los usuarios
find /home/*/Maildir/cur/ -type f 2>/dev/null | head -500 | \
  xargs -I{} rspamc --connect /var/run/rspamd/rspamd_controller.sock learn_ham {}

find /home/*/Maildir/.Sent/cur/ -type f 2>/dev/null | head -500 | \
  xargs -I{} rspamc --connect /var/run/rspamd/rspamd_controller.sock learn_ham {}
```

Trampas rspamc:

- `rspamc learn_spam` **contra el worker (puerto 11333)** da "HTTP 500 invalid command". HAY QUE usar el controller socket

- `rspamc stat` **contra puerto default (11333)** da "Connection refused" en DA — usar siempre: `rspamc --connect /var/run/rspamd/rspamd_controller.sock stat`
- `redis-server` / `redis` no existen como servicio en DA — es `redis-rspamd.service`
- `exim -bP macro` (singular) no existe — el comando es `exim -bP macros` (plural)

Verificación post-training:

```
rspamc --connect /var/run/rspamd/rspamd_controller.sock stat | grep -E "learned|messages"
# Debe mostrar >200 spam y >200 ham
```

Procedimiento de aplicación

Pre-checks

```
systemctl is-active rspamd
systemctl is-active redis-rspamd.service
ls -la /etc/rspamd/local.d/
cat /etc/exim.easy_spam_fighter/variables.conf.custom 2>/dev/null || echo "NO EXISTE"
# Contar correos disponibles para training
find /home/*/Maildir/.INBOX.spam/cur/ -type f 2>/dev/null | wc -l
find /home/*/Maildir/.Junk/cur/ -type f 2>/dev/null | wc -l
```

Orden

1. `mkdir -p /etc/rspamd/local.d/maps`
2. Crear los 8 ficheros rspamd (secciones 1-8 arriba)
3. `rspamadm configtest` — verificar sintaxis
4. `systemctl reload rspamd`
5. Crear/actualizar `variables.conf.custom` (sección 9)
6. `exim -bV` — verificar sintaxis Exim
7. `systemctl restart exim`
8. Entrenamiento masivo de Bayes
9. Verificar: `rspamc stat`, enviar correo de prueba

Verificación post-aplicación

```
# rspamd
systemctl is-active rspamd
```

```
rspamadm configtest
rspamc --connect /var/run/rspamd/rspamd_controller.sock stat

# Exim
systemctl is-active exim
exim -bP macros | grep -E "EASY_(SPF|DKIM|FORWARD)"
# Debe mostrar -5 en cada macro

# Test correo: enviar email y verificar cabeceras X-Spamd-Result
```

Rollback

Todos los ficheros son independientes. Para revertir cualquier cambio, borrar el fichero y `systemctl reload rspamd`:

```
# Rollback total rspamd
rm -f /etc/rspamd/local.d/actions.conf
rm -f /etc/rspamd/local.d/classifier-bayes.conf
rm -f /etc/rspamd/local.d/groups-override.conf
rm -f /etc/rspamd/local.d/multimap.conf
rm -f /etc/rspamd/local.d/phishing.conf
rm -f /etc/rspamd/local.d/url_suspect.conf
rm -rf /etc/rspamd/local.d/maps/
systemctl reload rspamd
```

Para ESF, editar `variables.conf.custom` y eliminar las líneas añadidas, luego `systemctl restart exim`.

Servidores aplicados

Servidor	Fecha	Bayes (spam/ham)	Estado
kvm456	2026-03-04	1.035 / 873	Completo
dar	2026-03-04 (alineado 2026-03-23)	Trained	Completo
srv120	2026-03-07	567 / 448	Completo
srv121	2026-03-07	158 / 1.597	Autolearn activo (spam bajo)

Ficheros que DA gestiona — NO TOCAR

Fichero	Motivo
`/etc/rspamd/users.d/<user>.conf`	Settings per-user generados por DA
`/etc/rspamd/directadmin-users.conf`	Config global usuarios DA
`/etc/exim.easy_spam_fighter/variables.conf`	Macros ESF (el `.custom` SÍ es seguro)
`/etc/rspamd/local.d/dkim_signing.conf`	Puede ser gestionado por DA

Bug template rspamd_settings.conf — Whitelist per-user ineficaz

El problema

Las whitelists de SpamAssassin que los usuarios configuran desde el panel de DirectAdmin (E-Mail → SpamAssassin Setup → Whitelist) **no funcionan** con rspamd. Los remitentes whitelisteados siguen llegando a la carpeta spam.

Afecta a **todas las versiones de DirectAdmin con** `spamd=rspamd` (verificado hasta DA 1.696).

Causa raíz (doble)

El template `/usr/local/directadmin/data/templates/rspamd_settings.conf` genera bloques `_whitelist` en `/etc/rspamd/users.d/*.conf` con dos defectos:

1. Priority incorrecta

El bloque `_whitelist` tiene `priority = 4` hardcoded, pero el bloque `_prefs` usa `priority = medium` (= 5 en rspamd). rspamd selecciona el bloque de mayor prioridad → `_prefs` SIEMPRE gana, la whitelist NUNCA se aplica.

2. Falta `apply.actions`

Incluso si la prioridad fuera correcta, el bloque whitelist solo tiene `want_spam = yes` sin un bloque `apply { actions { ... } }`. rspamd sigue añadiendo cabecera `X-Spam-Status: Yes` y el filtro de dominio de Exim sigue enviando a la carpeta spam.

Contraste con blacklist

Los bloques `_blacklist` del mismo template **sí funcionan** porque:

- Usan `priority = high` (= 10)
- Incluyen `apply { actions { ... } }`

Cómo verificar

```
# Ver qué settings block aplica rspamd para un usuario
grep "settings.lua" /var/log/rspamd/rspamd.log | grep USUARIO

# Siempre mostrará "settings_id: USUARIO_prefs" — nunca "USUARIO_whitelist"
```

Solución: template custom

Paso 1: Crear copia custom del template

```
mkdir -p /usr/local/directadmin/data/templates/custom/
cp /usr/local/directadmin/data/templates/rspamd_settings.conf \
   /usr/local/directadmin/data/templates/custom/rspamd_settings.conf
```

Paso 2: Editar la copia custom

En `/usr/local/directadmin/data/templates/custom/rspamd_settings.conf`, buscar los bloques que contienen `whitelist` y cambiar:

Antes (roto):

```
|*if whitelist_count>"0"|
|WHITELIST_ID| {
priority = 4;
|CUSTOM12|
|RCPT|
|whitelist_from_list|
want_spam = yes;
|CUSTOM13|
}
```

Después (corregido):

```
|*if whitelist_count>"0"|
|WHITELIST_ID| {
priority = 6;
|CUSTOM12|
|RCPT|
|whitelist_from_list|
want_spam = yes;
apply {
actions {
"add header" = 9999;
"rewrite subject" = null;
}
}
|CUSTOM13|
}
```

Hay **dos bloques** whitelist en el template (uno con `|whitelist_from_list|` y otro con listas adicionales). Aplicar el cambio a ambos.

Paso 3: Regenerar users.d para usuarios existentes

El template custom se aplica cuando:

- Un usuario cambia sus preferencias de spam en el panel
- Se ejecuta un rebuild que afecta a users.d

Para forzar la regeneración de un usuario específico, el usuario debe ir a E-Mail → SpamAssassin Setup y guardar (aunque no cambie nada).

Para **usuarios que ya tenían whitelist activa**, hay que parchear manualmente su fichero `users.d/*.conf`:

```
# Ejemplo para usuario "alufasa" en kvm456
# En /etc/rspamd/users.d/alufasa.conf, buscar el bloque _whitelist y cambiar:
# priority = 4 → priority = 6
# Añadir apply.actions tras want_spam = yes
```

Cadena de prioridades correcta

```
blacklist (high=10) > whitelist (6) > prefs (medium=5) > default
```

Persistencia

El template custom en `data/templates/custom/` sobrevive updates de DirectAdmin. Es el mecanismo oficial de override.

Referencia: post en el foro DA

Este bug fue reportado en el foro oficial de DirectAdmin:

Bug: rspamd_settings.conf template — per-user whitelist never applied (priority 4 < prefs medium)

El bug original de blacklist (mismo template) fue reportado en 2019 y corregido. La whitelist nunca recibió el mismo fix.

Alternativa: whitelist dinámica via email

Independientemente de este bug, existe un sistema de whitelist dinámica gestionado via email que **SÍ funciona correctamente** — usa multimap con score -100 en vez del mecanismo de settings per-user. Ver la página "Whitelist dinámica de rspamd via email en DirectAdmin" en esta misma wiki.

Servidores con template custom desplegado

Servidor	Fecha	Usuarios parcheados manualmente
kvm456	2026-03-19	alufasa, amenocalbus, audionica, elayudante, gestemalisal, lopeza, regfiltersl
srv120	2026-03-19	tmar
srv121	2026-03-19	neolystic
dar	2026-03-19	(ninguno con whitelist activa)

Nota sobre `want_spam = yes`

`want_spam = yes` en rspamd settings NO es una whitelist por sí solo. Solo indica que el bloque de settings se aplica también a mensajes clasificados como spam. Sin `apply.actions`, rspamd sigue marcando el correo como spam. Es una trampa sutil de la documentación de rspamd.

CSF + ModSecurity en nginx: MODSEC_LOG y regex custom

El problema

En servidores DirectAdmin con **nginx + ModSecurity 3** (no Apache), CSF/LFD no detecta los bloqueos de ModSecurity. Un usuario puede ser bloqueado repetidamente por ModSecurity (HTTP 406) sin que LFD lo registre ni tome acción (ban temporal, notificación, etc.).

Consecuencia: ModSecurity bloquea requests pero LFD es **ciego** a esos eventos. Si una IP llega a `csf.deny`, es por otro mecanismo (DA BFM, manual), nunca por `LF_MODSEC`.

Causa raíz (triple)

1. `MODSEC_LOG` apunta al fichero equivocado

El default de CSF es:

```
MODSEC_LOG = "/var/log/httpd/error_log"
```

Pero en DirectAdmin con nginx + ModSecurity 3, **ModSecurity corre en nginx, no en Apache**. Los eventos de "Access denied" se escriben en los **error.log per-domain de nginx**, no en httpd:

```
/var/log/nginx/domains/dominio.com.error.log
```

2. CSF no tiene regex para nginx-modsec3

El fichero built-in `RegexMain.pm` solo tiene regex para el formato ModSecurity v2 (Apache). El formato nginx-modsecurity3 tiene campos extra (`PID#TID: *CONN`) que la regex no matchea.

Formato Apache-modsec2:

```
[date] [error] [client IP] ModSecurity: Access denied ...
```

Formato nginx-modsec3:

```
YYYY/MM/DD HH:MM:SS [error] PID#TID: *CONN [client IP] ModSecurity: Access denied ...
```

3. Logs distribuidos por dominio

Los "Access denied" de nginx-modsec3 se escriben en los error.log per-domain de nginx, no en un fichero centralizado. CSF necesita leer TODOS los error.log de dominio.

Solución

Paso 1: Corregir `MODSEC_LOG` en `csf.conf`

```
# En /etc/csf/csf.conf, cambiar:
MODSEC_LOG = "/var/log/nginx/domains/*.error.log"
```

CSF soporta globs — se evalúan al iniciar LFD. Cada `*.error.log` de cada dominio será monitoreado.

Consecuencia: tras crear un nuevo dominio, hay que reiniciar LFD (`csf -ra`) para que monitorice su error.log nuevo.

Paso 2: Crear regex custom para nginx-modsec3

En `/usr/local/csf/bin/regex.custom.pm`, añadir **dentro de la sección de reglas** (antes del `return ()` final):

```
# nginx-modsecurity3 format
# YYYY/MM/DD HH:MM:SS [error] PID#TID: *CONN [client IP] ModSecurity: Access denied ...
if (($globlogs{MODSEC_LOG}{$lgfile}) and ($line =~ /^\S+ \S+ \[\S+\] \S+ \*\d+ \[client
(\S+)\] ModSecurity:(( \[[^\]]+\])*)? Access denied/)) {
    my $ip = $1;
    my $domain = "";
    if ($line =~ /\[hostname "([^"]+)"\]/) {$domain = $1}
    my $ruleid = "unknown";
    if ($line =~ /\[id "(\d+)"\]/) {$ruleid = $1}
    $ip =~ s/^::ffff://;
    return ("mod_security v3 (id:$ruleid domain:$domain) triggered by", $ip, "modsecnginx",
"5", "80,443", "7200", "0");
```

```
}
```

Qué hace:

- Matchea el formato nginx-modsec3
- Extrae IP, dominio (hostname), y rule ID
- Retorna al LFD con trigger type "modsecnginx", 5 hits para ban, puertos 80/443, ban temporal de 2 horas

Paso 3: Verificar sintaxis y reiniciar

```
# Verificar que el Perl compila
perl -c /usr/local/csf/bin/regex.custom.pm

# Reiniciar CSF + LFD
csf -ra
```

Paso 4: Verificar que LFD monitoriza los ficheros

```
# Debe mostrar líneas de "watching" para *.error.log
grep "watching" /var/log/lfd.log | grep nginx | tail -10
```

Persistencia

- `regex.custom.pm` **sobrevive upgrades de CSF** — es el lugar oficial para customizaciones
- `csf.conf` también sobrevive upgrades, pero verificar tras actualizaciones mayores de CSF
- **Los globs en** `MODSEC_LOG` **solo se evalúan al iniciar LFD** — tras crear un nuevo dominio, reiniciar LFD con `csf -ra`

Diagnóstico

Si sospechas que LFD no está detectando bloqueos ModSecurity:

```
# 1. Verificar que hay eventos ModSecurity en los logs nginx
grep "ModSecurity: Access denied" /var/log/nginx/domains/*.error.log | tail -5
```

```
# 2. Verificar que MODSEC_LOG apunta a nginx
grep "MODSEC_LOG" /etc/csf/csf.conf

# 3. Verificar que LFD registra triggers modsec
grep "modsec" /var/log/lfd.log | tail -10

# 4. Si no hay nada en lfd.log pero sí hay eventos en nginx:
# → MODSEC_LOG incorrecto o regex no matchea
```

Servidores aplicados

Servidor	Fecha	Estado
srv120	2026-03-05	Activo
srv121	2026-03-05	Activo
kvm456	2026-03-05	Activo
amazzal	2026-03-05	Activo
dar	2026-03-05	Activo
titrit	N/A	No tiene modsec en nginx

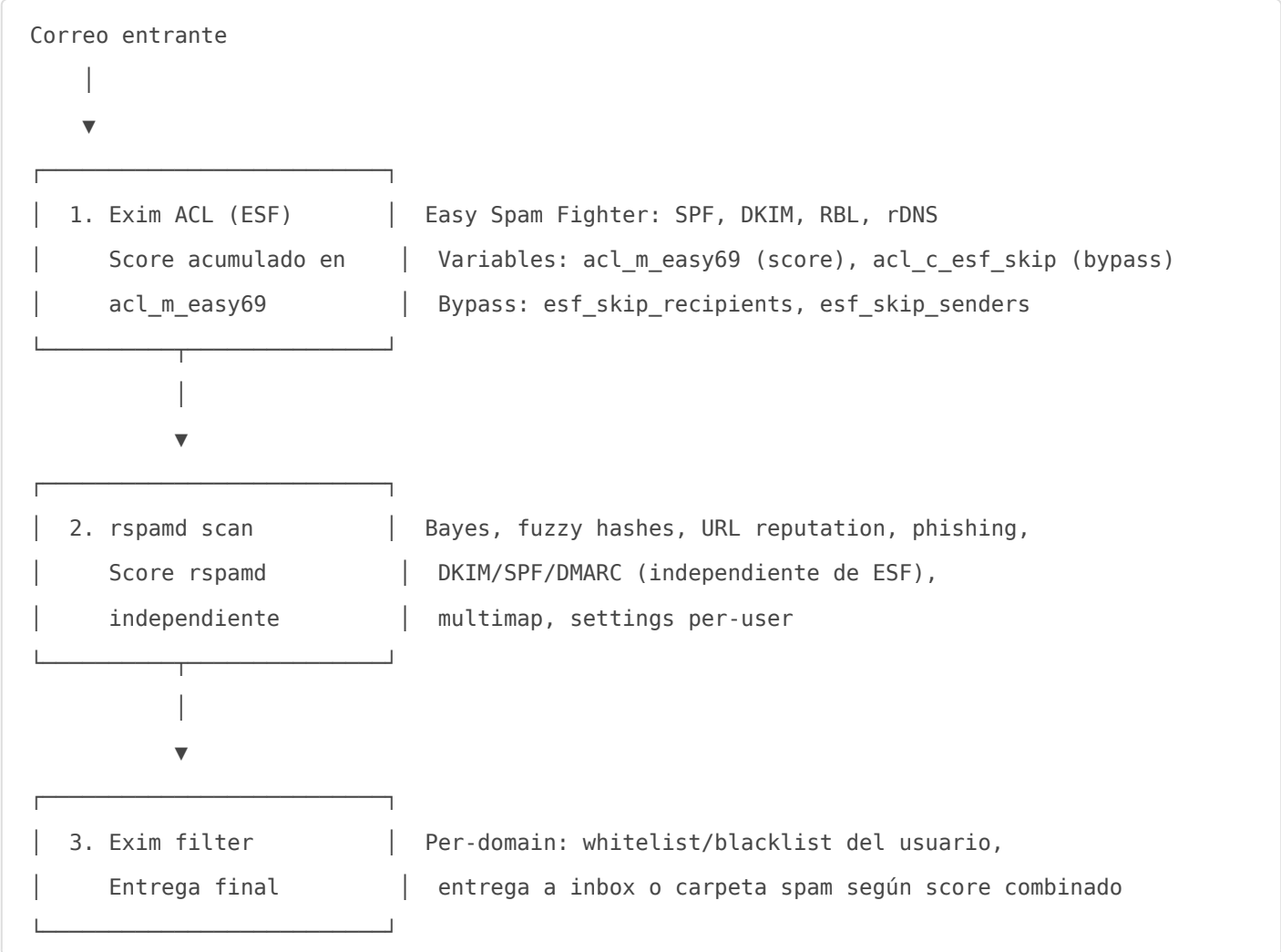
Checklist para nuevos servidores DA

- ☐ `MODSEC_LOG` en csf.conf apunta a `/var/log/nginx/domains/*.error.log`
- ☐ `regex.custom.pm` tiene la regex para nginx-modsec3
- ☐ `perl -c regex.custom.pm` compila sin errores
- ☐ `csf -ra` ejecutado
- ☐ Verificar con `grep "watching" /var/log/lfd.log | grep nginx`

Arquitectura antispam en DirectAdmin: Exim ESF + rspamd

Las tres capas

DirectAdmin con rspamd activo procesa cada correo entrante en tres capas secuenciales:



Punto clave: ESF y rspamd son sistemas **independientes**. ESF evalúa en la ACL de Exim (antes de aceptar el correo), rspamd evalúa el contenido (después de aceptar). Cada uno genera su propio score. La decisión final la toma el filtro de dominio de Exim combinando ambos.

Capa 1: Easy Spam Fighter (ESF)

ESF es un conjunto de ACLs de Exim mantenido por DirectAdmin. Evalúa:

Check	Macro (default)	Qué hace
SPF pass	`EASY_SPF_PASS` (-30)	Bonificación por SPF válido
DKIM pass	`EASY_DKIM_PASS` (-20)	Bonificación por DKIM válido
rDNS confirmado	`EASY_FORWARD_CONFIRMED_RDNS` (-10)	Bonificación por PTR válido
SPF fail	`EASY_SPF_FAIL` (100)	Penalización por SPF inválido
DKIM fail	`EASY_DKIM_FAIL` (100)	Penalización por DKIM inválido
Sin rDNS	`EASY_NO_REVERSE_IP` (100)	Penalización por sin PTR
RBL hit	Variable	Penalización por blacklist
Acumulado default	**-60**	SPF+DKIM+rDNS válidos

El problema del -60

Un correo spam reenviado via Google Groups pasa SPF (google.com) + DKIM (google.com) + rDNS → obtiene **-60 puntos ESF** antes de que rspamd lo evalúe. Si rspamd devuelve un score bajo (no action), ESF no marca el correo y se entrega al INBOX.

Solución aplicada: reducir a -15 total (-5/-5/-5) via `variables.conf.custom`. Ver la página "Tuning de rspamd en DirectAdmin" para detalles.

Dónde se configuran los overrides ESF

```
/etc/exim.easy_spam_fighter/variables.conf         ← DA regenera (NO TOCAR)
/etc/exim.easy_spam_fighter/variables.conf.custom   ← Override seguro (== para redefinir)
```

Capa 2: rspamd

rspamd evalúa el contenido del correo de forma independiente:

- Bayes (aprendizaje estadístico)
- Fuzzy hashes (contenido conocido como spam)
- URL reputation (Spamhaus DBL, SURBL, URIBL)
- Phishing detection (OpenPhish)
- DKIM/SPF/DMARC (evaluación independiente de ESF)

- Multimap (whitelists, blacklists, penalizaciones custom)
- Settings per-user (thresholds, whitelists del panel)

Ficheros de configuración:

Ruta	Gestión	Seguro para editar
`/etc/rspamd/local.d/*.conf`	Manual	✅ Sí
`/etc/rspamd/local.d/maps/*.map`	Manual	✅ Sí
`/etc/rspamd/users.d/*.conf`	DirectAdmin	❌ No
`/etc/rspamd/directadmin-users.conf`	DirectAdmin	❌ No

Punto de unión: la condición de escaneo rspamd

El módulo rspamd de Exim (`/etc/exim/rspamd/check_message.conf`) tiene esta condición antes de escanear:

```
warn  condition = ${if eq{$acl_c_rspamd}{1}}
      condition = ${if !eq{$acl_c_esf_skip}{1}}      ← bypass ESF = bypass rspamd
      condition = ${if < {$message_size}{EASY_SPAMASSASSIN_MAX_SIZE}}
      condition = ${if !eq{$acl_m_spam_user}{nobody}}
      set acl_m_rspamd_on = 1
```

Cuatro condiciones deben cumplirse para que rspamd escanee:

1. `acl_c_rspamd = 1` — rspamd habilitado globalmente
2. `acl_c_esf_skip ≠ 1` — el dominio/remitente NO está en `esf_skip_*`
3. `message_size < MAX_SIZE` — correo dentro del límite (default 200K, recomendado 15M)
4. `acl_m_spam_user ≠ nobody` — el usuario tiene `~/.spamassassin/user_prefs`

Si **cualquiera** falla, rspamd NO escanea y el correo pasa sin evaluación de contenido.

Mecanismos de bypass

`esf_skip_recipients` y `esf_skip_senders`

Son **bypass TOTALES** — desactivan tanto el scoring ESF como el escaneo rspamd.

```
/etc/virtual/esf_skip_recipients   ← dominios destino (uno por línea)
/etc/virtual/esf_skip_senders      ← dominios remitente (uno por línea)
```

Cuando un dominio está en estos ficheros, `acl_c_esf_skip` se pone a 1 y rspamd no escanea.

Cuándo se usan: Como medida de protección para dominios con historial de falsos positivos graves que generaron fricción con el cliente. Es una decisión operativa consciente, no un mecanismo de "no rechazar".

Riesgo: Los correos que pasan por bypass no tienen NINGÚN escaneo — ni ESF ni rspamd ni phishing ni malware.

`acl_m_spam_user = nobody`

Si `~/.spamassassin/user_prefs` no existe para un usuario DA, la ACL no puede resolver el usuario y rspamd no escanea. Esto afecta silenciosamente a usuarios que nunca han tocado la configuración de SpamAssassin en el panel.

Diagnóstico:

```
# Contar correos sin escanear por user_prefs faltante
grep "acl_m_spam_user=nobody" /var/log/exim/mainlog | wc -l

# Listar usuarios sin user_prefs
for d in /home/*/; do
    user=$(basename "$d")
    if [ ! -f "$d/.spamassassin/user_prefs" ]; then
        echo "$user"
    fi
done
```

Solución: Crear `user_prefs` vacío con propietario correcto:

```
for d in /home/*/; do
    user=$(basename "$d")
    uid=$(id -u "$user" 2>/dev/null) || continue
    prefs_dir="$d/.spamassassin"
    prefs_file="$prefs_dir/user_prefs"
    if [ ! -f "$prefs_file" ]; then
        mkdir -p "$prefs_dir"
        touch "$prefs_file"
```

```
        chown -R "$user:$user" "$prefs_dir"
    fi
done
```

`EASY_SPAMASSASSIN_MAX_SIZE`

Default: 200K. Correos con adjuntos (>200K) no se escanean. Recomendado: 15M.

```
# En variables.conf.custom
EASY_SPAMASSASSIN_MAX_SIZE == 15M
```

Alternativas a bypass total

Para casos donde se necesita proteger contra falsos positivos SIN perder el escaneo rspamd:

Mecanismo	Qué hace	rspamd escanea	Riesgo
`esf_skip_*`	Bypass total	No	Sin protección alguna
`whitelist_from.map` (score -100)	Whitelist rspamd	Sí	Bajo (detecta phishing/malware)
Whitelist dinámica (via email)	Whitelist rspamd	Sí	Bajo
Per-user threshold alto	Threshold permisivo	Sí	Medio (score alto pasa)

Recomendación: Para nuevos casos de "no rechazar", usar `whitelist_from.map` con score -100. rspamd escanea (phishing, malware, URLs), pero el score se neutraliza.

Diagrama de decisión: ¿por qué no se escaneó?

```
¿rspamd escaneó este correo?
    │
    ├── NO → ¿acl_c_esf_skip = 1?
    │         ├── SÍ → Dominio en esf_skip_recipients o esf_skip_senders
    │         └── NO → ¿message_size > MAX_SIZE?
    │                   ├── SÍ → Correo demasiado grande (subir MAX_SIZE)
    │                   └── NO → ¿acl_m_spam_user = nobody?
```

```
    │                          ├── SÍ → user_prefs no existe (crear)
    │                          └── NO → ¿acl_c_rspamd = 0?
    │                                    ├── SÍ → rspamd desactivado globalmente
    │                                    └── NO → Investigar ACLs
    │
    └── SÍ → ¿Por qué pasó?
              ├── Score bajo → Revisar Bayes, multimap, thresholds
              ├── Whitelist activa → Revisar whitelist_from.map + dynamic
              └── ESF score muy negativo → Reducir scores ESF (variables.conf.custom)
```

Diagnóstico rápido

```
# 1. ¿rspamd escaneó? Buscar cabeceras en el correo
grep "X-Spamd-Result" /path/to/email

# 2. ¿Cuántos correos no se escanearon esta semana?
# Por esf_skip:
grep "esf_skip" /var/log/exim/mainlog | wc -l

# Por user nobody:
grep "spam_user=nobody" /var/log/exim/mainlog | wc -l

# Por tamaño:
grep "too large for spam" /var/log/exim/mainlog | wc -l

# 3. ¿Qué dominios tienen bypass activo?
cat /etc/virtual/esf_skip_recipients
cat /etc/virtual/esf_skip_senders

# 4. Estado de Bayes (usar controller socket en DA)
rspamc --connect /var/run/rspamd/rspamd_controller.sock stat

# 5. Volumen spam vs ham
rspamc --connect /var/run/rspamd/rspamd_controller.sock stat | grep -E "Messages
scanned|Spam|Ham"
```

Resumen de configuración de referencia

Tras aplicar el tuning completo documentado en esta wiki:

Componente	Configuración	Fichero
ESF scores	-5/-5/-5 (total -15)	`variables.conf.custom`
ESF max_size	15M	`variables.conf.custom`
ESF high_score_drop	9999 (nunca)	`variables.conf.custom`
rspamd reject	null (nunca)	`actions.conf`
rspamd add_header	5 (marcar)	`actions.conf`
Bayes backend	Redis (pool global)	`classifier-bayes.conf`
Bayes autolearn	spam ≥8.0, ham ≤-1.0	`classifier-bayes.conf`
Phishing	OpenPhish activo	`phishing.conf`
URL suspect	word_dot desactivado	`url_suspect.conf`
Whitelist estática	score -100, filter=email:domain	`multimap.conf`
Whitelist dinámica	Via email, score -100	`multimap.conf`
DNSBL resolvers	64.6.64.6 / 199.85.126.10 / 156.154.70.2	netplan

DirectAdmin startips — supervisor / auto-restart de IPs virtuales

Fecha: 2026-04-30

- **Autor:** Abdelkarim Mateos
- **Estado:** plan de desarrollo, NO IMPLEMENTADO. Reservado para sesión futura.
- **Origen:** incidente Burcode 2026-04-30 (cloud500 IP virtual `87.98.230.68` caída → diagnóstico erróneo y reescritura de 178 A records → reversión).
- **Informe relacionado:** `informes/2026-04/2026-04-30-burcode-cloud500-fail500-dns-glue-muerto.md`
- **Memory:** `~/.claude/projects/-Users-abkrim-claude/memory/feedback-da-startips-no-protegido.md`

1. Problema

`startips.service` en DirectAdmin es `Type=oneshot`:

```
[Unit]
Description=Start the additional IPs
Wants=network-online.target
After=syslog.target network.target network-online.target
Requires=network.target

[Service]
Type=oneshot
ExecStart=/usr/local/directadmin/scripts/startips

[Install]
WantedBy=multi-user.target
```

El binario `/usr/local/directadmin/scripts/startips` recorre `/usr/local/directadmin/data/admin/ip.list` y aplica IPs adicionales con `ip addr add` sobre la interfaz primaria. Tras correr, el servicio queda `inactive (dead)` con `code=exited, status=0/SUCCESS`. **No hay supervisor.**

Si las IPs virtuales se caen tras boot por cualquier motivo:

- Reset/restart del subsistema de red (`systemctl restart networkd`, `netplan apply`, mantenimiento del proveedor).
- OOM killing de `networkd`.
- vRack/cloud-init re-aplicando configuración fresca.
- Cambio en `ens3` que limpia los aliases.

...las IPs no vuelven hasta el siguiente reboot completo o hasta que alguien lance `systemctl start startips.service` o `bash /usr/local/directadmin/scripts/startips` manualmente.

Impacto operativo:

- Dominios DA con virtual host ligado a IP virtual quedan inalcanzables.
- LE auto-renew falla para esos dominios (HTTP-01 challenge no responde).
- Diagnóstico desde fuera puede confundirse con "servidor antiguo desmantelado" (el operador o un agente terminan reescribiendo DNS por error — caso real Burcode 2026-04-30).

El operador reporta que es **problema recurrente** en su parque DA — segundo frente de trabajo abierto.

2. Diseño propuesto

2.1 Componentes

A) Health-check periódico de IPs virtuales

Script en `~/utilidades/directadmin/check-virtual-ips.sh` (versionado en monorepo `gitlab.castris.com/root/utilidades`):

- Lee `/usr/local/directadmin/data/admin/ip.list`.
- Para cada IP, comprueba si está activa en la interfaz primaria (`ip -4 addr show <iface> | grep <ip>`).
- Si UNA IP de la lista NO está en la interfaz, lanza:
  1. `systemctl start startips.service` (idempotente — el servicio ya existe).
  2. Verifica que tras el restart la IP aparece.
  3. Si tras el restart sigue sin aparecer, alerta (mail/Telegram) y NO bucla.

B) Systemd timer para el health-check

En `/etc/systemd/system/startips-watchdog.{service,timer}`:

```
# startips-watchdog.service
[Unit]
Description=Watchdog for DirectAdmin virtual IPs
After=startips.service

[Service]
Type=oneshot
ExecStart=/usr/local/sbin/check-virtual-ips.sh
StandardOutput=journal
StandardError=journal
```

```
# startips-watchdog.timer
[Unit]
Description=Run virtual IP watchdog every 5 minutes

[Timer]
OnBootSec=2min
OnUnitActiveSec=5min
AccuracySec=30s

[Install]
WantedBy=timers.target
```

Frecuencia 5 min = compromiso entre detección rápida y carga mínima. Ajustable per-server.

C) Alertas

Para no llenar el inbox, política de "alerta solo si auto-fix falla":

- Si auto-restart resuelve → log en journald, sin alerta.
- Si auto-restart falla 2 ciclos consecutivos → mail al operador + Telegram (vía bot existente).
- Métrica opcional: contador en `/var/log/startips-watchdog.metrics` con histórico de detecciones.

D) (Opcional) Métricas a Zabbix

Si Castris tiene Zabbix, exponer:

- Item `startips.virtual_ips.expected` (entero, lectura de `ip.list`).
- Item `startips.virtual_ips.active` (entero, IPs activas en interfaz que están en `ip.list`).
- Trigger: si `expected != active` durante 2 ciclos → warning.

3. Plan de trabajo (otra sesión)

Fase 1 — POC manual (1-2 horas)

1. Escribir `check-virtual-ips.sh` con dry-run + apply.
2. Probar en cloud500 (caso conocido). Forzar caída con `ip addr del 87.98.230.68 dev ens3` y verificar detección + auto-fix.
3. Validar logs en journald.

Fase 2 — Empaquetado (1 hora)

1. Mover script a `~/utilidades/directadmin/`.
2. Crear systemd unit + timer.
3. Documentar en `~/utilidades/directadmin/README.md`.
4. Empaquetar en script de instalación que copie unit + binarios + habilite el timer.

Fase 3 — Despliegue flota DA Castris (30 min)

1. Auditar qué servidores DA tienen IPs adicionales:

   ```
   for s in $(sshctx list --panel directadmin --format raw); do
     n=$(ssh $s 'wc -l < /usr/local/directadmin/data/admin/ip.list')
     echo "$s: $n IPs"
   done
   ```

2. Desplegar el watchdog en los que tienen `>1 IP`.
3. Servidores conocidos con IPs adicionales: cloud500 (Burcode, 4 IPs). Resto pendiente auditoría.

Fase 4 — Despliegue clientes externos (variable)

Servidores semi-managed (Burcode, otros): pedir autorización antes de instalar.

4. Casos de uso

- **Burcode cloud500** (caso semilla del incidente).

- **Flota DA Castris** que tenga IPs virtuales: srv120, srv121, kvm456, dar, amazzal, titrit, bitatrader (auditar).
- **Cualquier DA futuro provisionado con IPs adicionales** (regla preventiva: instalar el watchdog en provisión, junto con CSF/DA).

5. Decisiones pendientes (para sesión de implementación)

5.1 Frecuencia del timer

Opciones:

- **5 min**: detección rápida, carga insignificante. Recomendado.
- **1 min**: detección casi inmediata pero ruido en journald.
- **10-15 min**: caída detectable pero ventana de fallo amplia.

5.2 Política de retry

Opciones:

- **Auto-restart 1 vez + alerta si falla**: simple, evita bucles.
- **Backoff exponencial**: 1 min → 5 min → 15 min → alerta.
- **Solo alerta, sin auto-restart**: dejar decisión al operador.

5.3 Alcance del watchdog

Opciones:

- **Solo IPs virtuales DA** (lo que propone este plan).
- **Extender a otros servicios cuya caída sea silenciosa** (ej. `dataskq` cron, `assp`, etc.). Probable scope-creep — empezar simple.

5.4 Forma de alerta

Opciones:

- **Mail al operador** (existing).
- **Telegram via bot existente** (más rápido).
- **Ambos** (recomendado para incidentes recurrentes).

6. Riesgos y mitigaciones

Riesgo	Mitigación
El watchdog se mete en bucle restartando startips si éste falla siempre	Implementar contador de retries, parar tras N fallos, alertar
Race condition con un cambio legítimo de IP del operador	El watchdog debe leer `ip.list` cada vez (no cachear); si se cambia `ip.list` está reflejado
El propio `startips.service` es buggy en DA y no funciona en algunos escenarios	Documentar exit code de startips; si falla repetidamente, alertar para fix manual
Ruido en journald	Log conciso (1 línea por chequeo OK, n líneas por incidente)

7. Anexos

7.1 Script de detección manual (referencia rápida)

```
#!/bin/bash
# /usr/local/sbin/check-virtual-ips.sh — POC v0
set -euo pipefail

IP_LIST="/usr/local/directadmin/data/admin/ip.list"
IFACE="$(ip -4 route show default | awk '{print $5; exit}')"
MISSING=()

while read -r ip; do
  [ -z "$ip" ] && continue
  if ! ip -4 addr show "$IFACE" | grep -qE "inet $ip(/| )"; then
    MISSING+=("$ip")
  fi
done < "$IP_LIST"

if [ "${#MISSING[@]}" -eq 0 ]; then
  exit 0  # OK
fi
```

```
logger -t startips-watchdog "IPs virtuales caídas: ${MISSING[*]}"
systemctl start startips.service

# verificar tras 2s
sleep 2
STILL_DOWN=()
for ip in "${MISSING[@]}"; do
  if ! ip -4 addr show "$IFACE" | grep -qE "inet $ip(/| )"; then
    STILL_DOWN+=("$ip")
  fi
done

if [ "${#STILL_DOWN[@]}" -eq 0 ]; then
  logger -t startips-watchdog "Auto-restart OK, IPs restauradas: ${MISSING[*]}"
  exit 0
fi

logger -t startips-watchdog "ALERTA: IPs siguen caídas tras restart: ${STILL_DOWN[*]}"
# TODO: enviar mail/Telegram
exit 1
```

7.2 Comandos de auditoría de la flota

```
# ¿Qué servidores DA tienen IPs adicionales?
for s in cloud500 srv120 srv121 kvm456 dar amazzal titrit bitatrader fail500; do
  n=$(ssh $s 'wc -l < /usr/local/directadmin/data/admin/ip.list 2>/dev/null')
  echo "$s: $n IPs"
done

# ¿Y tras un test de stress, qué dominios usan IPs no-primarias?
ssh <server> '
PRIMARY=$(ip route get 8.8.8.8 | awk "/src/ {print \$NF}")
for f in /usr/local/directadmin/data/users/*/domains/*.conf; do
  ip=$(grep "^ip=" "$f" | cut -d= -f2)
  if [ "$ip" != "$PRIMARY" ] && [ -n "$ip" ]; then
    dom=$(basename "$f" .conf)
    echo "  $ip → $dom"
  fi
done | sort | uniq -c | sort -rn
```

'

Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de Castris

Si necesitas soporte profesional puedes contratar con Castris soporte profesional.