FIELD NOTES FROM A PRESALES SOLUTIONS ARCHITECT

Security Alerts That Never Arrive — a Dead maldet Monitor, Exim -t and a Hostname With No SPF

A field note about a hosting server whose malware scanner was "set up" for months and never sent a single alert — three silent failures in a row, and the end-to-end test I now run before I call any alert path working.

Daanial Farrukh · 24 November 2026 · https://quabin.com/server-security-alerts-never-arrive-maldet-exim-spf/

We were closing a firewall ticket on a customer's hosting VM. The boot problem was fixed, the reboot was proven, the sites were back.

"While you're in there," the customer said, "the malware scanner is running. It has been installed for a long time. We get alerts if anything is found."

"When did you last get one?"

"We haven't. Nothing has been found."

"How do you know nothing has been found?"

"Because we would get an alert."

I asked one more question.

"Has anyone ever received a test alert from this server?"

Silence. Then: "We assumed it works. The scanner says it is running."

That's the trap. **An alert path that has never delivered a message is not quiet. It is unproven.** "No alerts" and "alerts are broken" look exactly the same from the inbox.

So we traced it — from the scanner to a human inbox, one hop at a time. It failed at every hop.

What an Alert Path Actually Is

People think of an alert as one thing. It is a chain, and every link fails on its own:

Detection → Alert config → Local mail binary → MTA (Exim) → Sender authentication (SPF/DMARC) → Recipient's mail filter → Inbox

Each link can break without telling anyone:

- The detector can be dead, so there is nothing to alert on.
- The alert can be switched off, or pointed at a placeholder address.

- The local mail command can hand the MTA a message with no recipient.
- The MTA can send a message the receiving side is told to reject.

On this server, four links were broken at the same time: three failed silently, and one had never been switched on. The scanner was Linux Malware Detect (maldet), on Ubuntu with cPanel and Exim. The lessons apply to any tool that sends mail through the local MTA — fail2ban, lfd, cron, backup scripts, monitoring agents.

Why Nobody Sees It in Testing

Because nobody tests the whole path. The usual "test" is one of these:

- systemctl status on the scanner — proves a process started, not that it detects anything.
- A look at the scanner's config file — proves the setting exists, not that mail leaves the box.
- A look at the scanner's own log — and here is the part that trips people up. **The tool logs that it handed off the alert. It does not know if anyone received it.**

A tool that says "sent" has only proven that it ran a command. The rest of the chain is somebody else's problem — and nobody owns it.

How to Actually Fix This

Here is the order I worked in. Each step starts read-only; the change is labelled.

Step 1: Prove the Detector Is Actually Running

maldet has two modes: a daily cron scan, and a real-time monitor that watches folders with inotify. On this server the daily cron scan was working. The real-time monitor was not.

```
systemctl status maldet
```

The service showed as failed. The reason was in its log: the monitor needs inotifywait, and it was not installed.

```
command -v inotifywait
```

No output means the binary is missing. The monitor cannot start without it, no matter what the config says. The fix (a state change):

```
apt install inotify-tools
```

```
systemctl restart maldet
```

Then check it again. It should be active and enabled, and the monitor should list the paths it watches. On this server it came up in "users" mode, watching the account web roots, public_ftp, /dev/shm, /var/tmp and /tmp.

```
systemctl is-active maldet
```

Plain takeaway: a scanner can be "installed" for a year with its main feature dead. Check the service, not the package.

Step 2: Check That Alerts Are Switched On — and Pointed at a Real Person

The alert settings live in /usr/local/maldetect/conf.maldet. Read them before changing anything:

```
grep -E '^(email_alert|email_addr|quarantine_hits)=' /usr/local/maldetect/conf.maldet
```

On this server they were still the install defaults:

```
email_alert="0"
email_addr="you@domain.com"
quarantine_hits="0"
```

Alerts off. A placeholder address. Detections logged but not quarantined. Nobody had ever changed them.

The change: back up the file, then set email_alert="1", put a real monitored mailbox in email_addr, and set quarantine_hits="1" so a hit is moved aside, not only reported. Restart maldet afterwards. If quarantine ever catches a false positive, maldet --restore <file> puts it back.

Plain takeaway: defaults are written for the vendor's test, not for your operations.

Step 3: Make Sure the Local Mail Command Keeps the Recipient

This is the failure that fooled everyone, because the scanner's log said the alert was sent.

The server had no mail command installed. In that case maldet falls back to calling sendmail directly, which on a cPanel server is Exim. The fallback passes the address on the command line together with -t:

```
sendmail -t admin@example.com
```

Here's why that fails. With -t, Exim takes the recipients from the message headers. On this server, that meant the address on the command line was dropped. The message had no To: header, so Exim found no recipients at all — and maldet still logged the alert as sent.

The proof is in the Exim log:

```
grep "no recipients found in headers" /var/log/exim_mainlog
```

If this returns lines at the times your tool claims it sent an alert, the message never left the box.

The fix (a state change) was to install a proper mail command, so maldet uses it instead of the broken fallback:

```
apt install bsd-mailx

systemctl restart maldet
```

After that, Exim accepted the message with a real recipient and tried to deliver it. Which led straight to the next failure.

Plain takeaway: "sent" in an application log means "I ran a command". Only the MTA log tells you what happened next.

Step 4: Check How the Receiving Side Sees Your Server

The message now left the server — and the recipient's mail filter rejected it:

```
550 ... blocked
```

The sender was root@vps.example.com — the server's own hostname. That is normal for system mail. But look at what the parent domain had published:

```
dig +short TXT _dmarc.example.com
```

```
"v=DMARC1; p=reject; sp=reject; aspf=s; adkim=s"
```

That is a strict policy, and the parts that matter here are:

- sp=reject — the policy applies to every subdomain, including the server hostname, unless the subdomain publishes its own DMARC record.
- aspf=s — strict SPF alignment: the domain that passes SPF must match the From: domain exactly.

Now check the hostname itself:

```
dig +short TXT vps.example.com
```

No output. **The hostname had no SPF record at all.** So mail from root@vps.example.com could not pass SPF, could not pass DMARC, and the domain owner's own policy told every receiver to reject it. The recipient's filter did exactly what it was asked to do.

This is easy to miss because the main domain's mail works perfectly. The website's contact form, the staff mailboxes — all fine. Only the system mail from the hostname fails, and system mail is the one nobody reads until it matters.

The fix (a DNS change) was one TXT record on the hostname, allowing only the server's own IP:

```
vps.example.com.  TXT  "v=spf1 ip4:203.0.113.10 -all"
```

Then resend the test. This time the recipient's filter answered 250 OK queued, and Exim logged the message as completed. **The SPF record was the only change between the failure and the success**, so that is the proven cause.

A side effect worth noting: this one record fixed every other piece of system mail from that hostname too — cron output, firewall alerts, update notices. They had all been failing the same way.

Plain takeaway: if your domain publishes sp=reject, every hostname that sends mail needs its own SPF record — or its own DMARC record — or it is rejected by policy.

Step 5: The Verification — Prove the Whole Path, End to End

This step is where most tutorials stop and where mine starts.

Every fix above proves one link. None of them proves the chain. So the last step is to send a message through the exact same path the alert will use, and watch it arrive.

```
echo "alert path test" | mail -s "alert path test from $(hostname -f)" admin@example.com
```

Then read what Exim did with it:

```
grep "alert path test" /var/log/exim_mainlog
```

Take the message ID from that line and follow it:

```
exigrep <message-id> /var/log/exim_mainlog
```

What success looks like:

- An <= line from root@ your hostname, with the right recipient.
- An => line showing the remote server answered 250.
- A Completed line.
- And the message actually in the mailbox — open it, look at it, check the headers say SPF pass.

What failure looks like:

- no recipients found in headers — Step 3 is still broken.
- A 550 or 5xx from the remote side — read the text; with a strict DMARC domain it is almost always Step 4.
- A deferral that repeats — the message is stuck; check exim -bp for the queue.

This error is not a bug. It is the verification. A failed test today is the alert you would have missed next month.

The Bigger Lesson — a Checklist for Any Alert Path

I now run these six checks on every server I hand over, for every tool that is supposed to "email us if something happens":

1. **The detector is alive.** systemctl is-active <service> — and read why if it is not.
2. **Its dependencies exist.** For maldet's monitor: command -v inotifywait.
3. **Alerts are on and point to a real mailbox.** grep -E '^(email_alert|email_addr)=' /usr/local/maldetect/conf.maldet
4. **A real mail command exists.** command -v mail — don't rely on a raw sendmail fallback.
5. **The hostname can pass DMARC.** dig +short TXT $(hostname -f) must return an SPF record if the parent domain is strict.
6. **One message has been received by a human.** Not "sent". Received.

Absence of alerts is not health. It is either health, or a broken pipe — and only a test can tell you which.

I write about the questions I ask in real customer rooms — the ones that change architectures and surface the problems no one else noticed. If you found this useful, drop your email below and I'll send the next one when it's ready.

FAQs

Why am I not receiving maldet email alerts?

Check four things in order: the maldet service is active, email_alert is set to 1 with a real address in conf.maldet, a mail command is installed, and the server hostname has an SPF record if your domain's DMARC policy is strict. Then send a test message and read the Exim log to see where it stops.

Why does the maldet monitor fail to start?

The real-time monitor uses inotify and needs the inotifywait binary from the inotify-tools package. If command -v inotifywait prints nothing, install inotify-tools and restart maldet. The daily cron scan can keep working while the monitor is dead, which hides the problem.

What does Exim "no recipients found in headers" mean?

The message was submitted with sendmail -t, which tells Exim to take recipients from the To:, Cc: and Bcc: headers, and the message had none. The address on the command line was not used. Install a proper mail command so the tool builds a complete message instead.

Why is mail from my server's hostname rejected when the main domain works?

If the parent domain's DMARC record has sp=reject, the reject policy also covers subdomains such as the server hostname. Mail from root@ the hostname then needs its own SPF record that matches the hostname. Without one, SPF and DMARC fail and receivers reject the mail.

Is an SPF record on the server hostname safe to add?

Yes. A record such as v=spf1 ip4:<server IP> -all only authorizes that one server to send as the hostname and tells receivers to reject anything else. It does not change how the main domain's mail is authorized.

How do I test that server alerts really reach someone?

Send a message with the same mail command the tool uses, find it in /var/log/exim_mainlog, follow it with exigrep, and confirm a 250 answer and a Completed line. Then open the message in the mailbox. An alert path is only proven when a person has received a message through it.

Read online: https://quabin.com/server-security-alerts-never-arrive-maldet-exim-spf/