

You've Generated Your SBOM. Now What?



IANAL



\$ whoami

Today's agenda

- 01 Why you need to care now
- 02 How to generate high quality SBOMS
- 03 Now what?
- 04 Discussion

EU Cyber Resilience Act (CRA)

Obligations **started** on September 11, 2026 with full force on December 11, 2027

What does this mean?

- Produce SBOMs from all your products
 - Archive said SBOMs for each release
- Actively monitor your product(s) for vulnerabilities
- Report actively exploited security issues



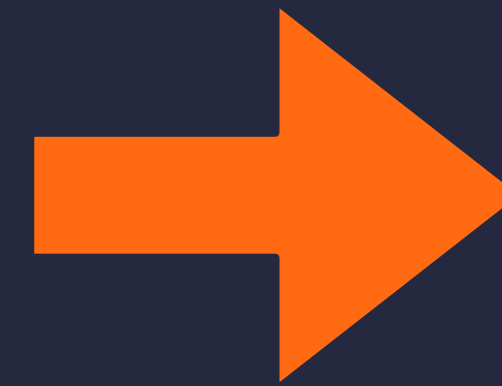
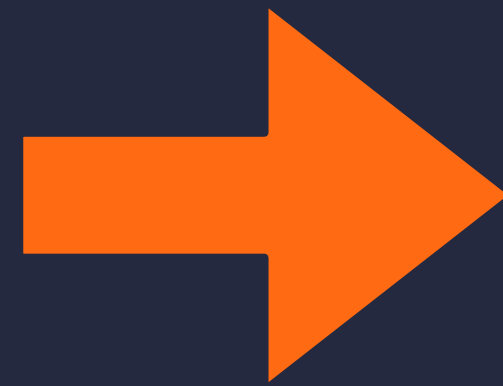
This is just the start.



How to generate high quality SBOMs?

How not to do it.

```
$ some-tool \  
-i requirements.txt \  
-o final.cdx.json
```





Creation

Augmentation

Enrichment

Signing



```
- uses: sbomify/sbomify-action@26.10.2
env:
  LOCK_FILE: requirements.txt
  OUTPUT_FILE: sbom.cdx.json
  ENRICH: true
  UPLOAD: false
```

Metric	Value
Overrides applied	2
Augmentation changes	8
Components enriched	12650
Sanitization fixes	422

Metric	Value
Components enriched	1773/2432
Descriptions added	1633
Publishers added	1423
Homepage URLs added	3071
Repository URLs added	1463
Distribution URLs added	1773

Enrichment by Source

Metric	Value
deps.dev	962
ecosyste.ms	811

github.com/sbomify/sbomify-action

Transitive vs. primary dependencies

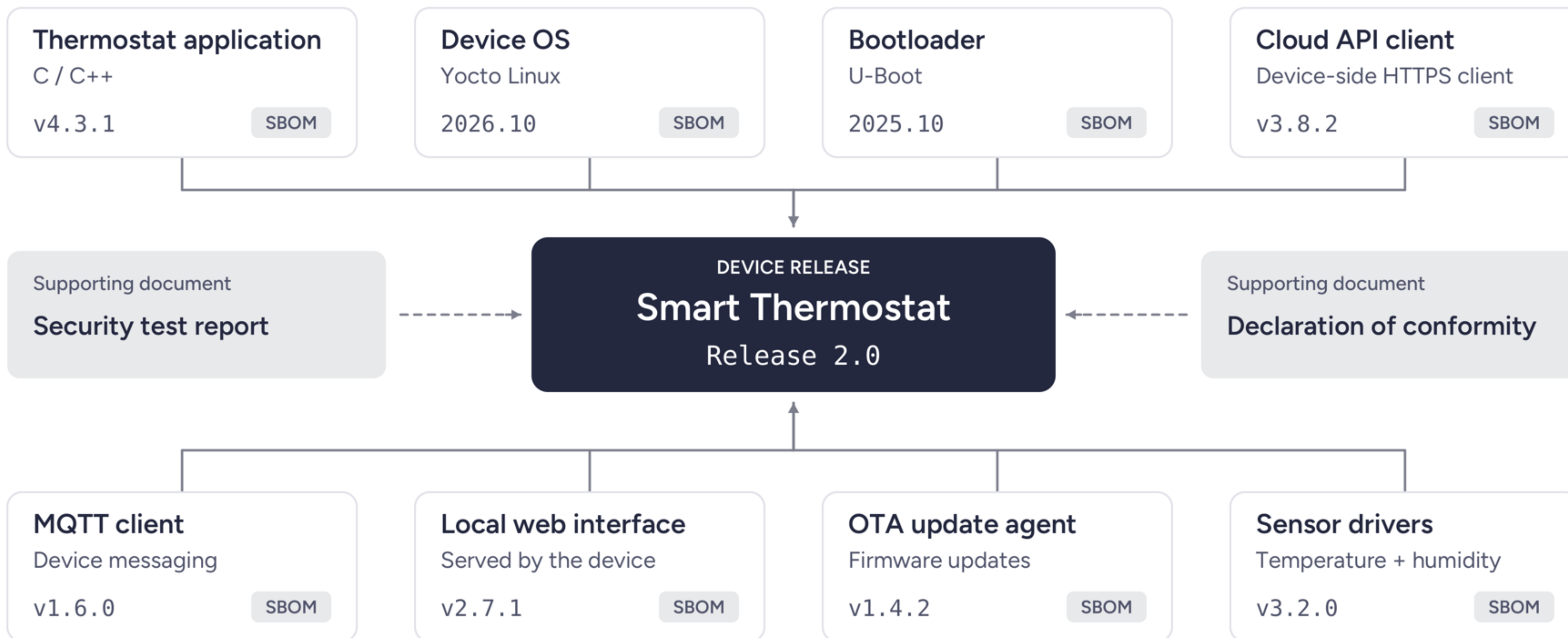
Now what?

A product is made up of components

- The real world is complicated
- Lots and lots of SBOMs
- Submodules, libraries, shared components

Many components, one product release

Software shipped in the thermostat, with its SBOMs and supporting product documents.



The release records the exact software components shipped in the device.

Illustrative components, versions and linked documents.

What is a product release?

- Release management is hard.
- Everyone does it differently

Release flow and ongoing monitoring

One product, two maintained release lines



Illustrative release policy. Superseded versions are not automatically outside CRA obligations.

CRA: Art. 13(8), Art. 13(10), Annex I Part II. Main requirements apply from 11 Dec 2027.

Vulnerability management?

- How do you monitor CVEs?

How do you communicate the security posture to customers?

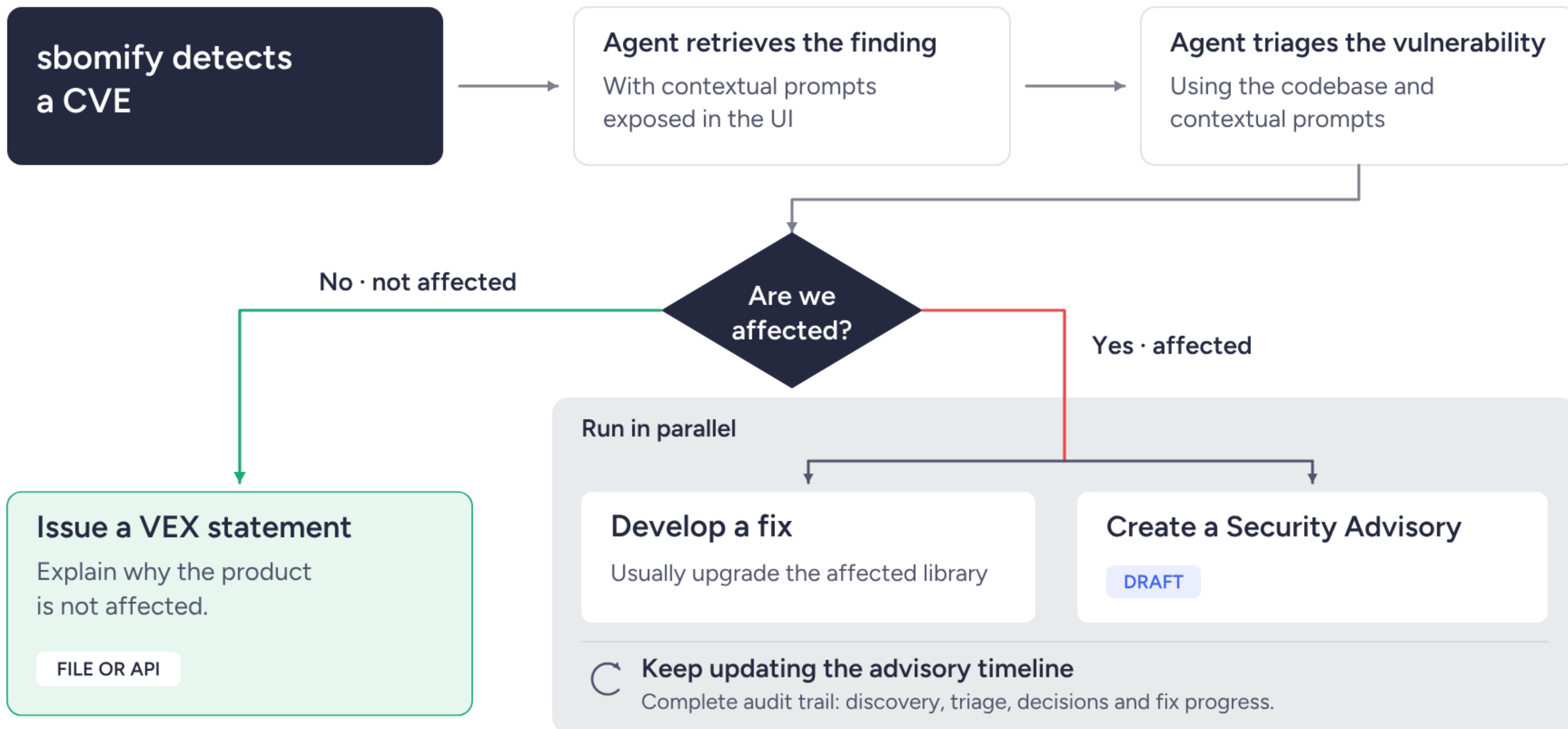
- Trust center?

Use the force, Luke!

- Use your agents for triaging

From CVE detection to action

Agent-assisted vulnerability triage



sbomify - Python Stack 5c79218, 5e110db AI agent

Reject

Approve

2 rulings · hides 2 open vulnerabilities in current and released versions, counted when sent · Viktor Petersson via "Agent IAP" · 5 minutes ago

Two new PyPI findings on **sbomify - Python Stack** (SBOM 5c79218, published upstream 2026-10-27), both unreachable in sbomify:

- **CVE-2026-114613 / GHSA-j586-36cw-2gc2 msgpack@1.2.1**: only used by **channels-redis** to decode sbomify's own channel-layer messages from its private Redis (`channels_redis/serializers.py` → `msgpack.unpackb`). Messages are produced server-side by `broadcast_to_workspace` (`sbomify/apps/core/utils.py`) with string keys; `WorkspaceConsumer.receive_json` (`sbomify/apps/core/consumers.py:179`) drops client input. No streaming **Unpacker** use.
- **CVE-2026-114605 / GHSA-45c3-73f7-pv4m dnspython@2.8.0**: only pulled in by **email-validator**, reached via pydantic **EmailStr**, which calls `validate_email(..., check_deliverability=False)` (`pydantic/networks.py:1303`). DNS code only runs in the deliverability branch (`email_validator/validate_email.py:191-203`). sbomify never imports `dns` or `email_validator` directly.

Fixed versions (msgpack 1.2.3, dnspython 2.9.0) can come in with normal dependency bumps.

Note: the fix-prompt endpoint returned 504 (gateway timeout) for both findings, so the purls are the standard PyPI purls for the reported package versions.

CVE-2026-114613

msgpack@1.2.1

Not affected · Code not reachable

sbomify at commit 5c79218 does not deserialize untrusted MessagePack data, so the leaking error path in the msgpack C extension cannot be driven by an attacker. msgpack 1.2.1 is present only as a dependency of channels-redis 4.3.0 (uv.lock); sbomify itself never imports msgpack. channels-redis uses it in channels_redis/serializers.py (MsgPackSerializer: `as_bytes = msgpack.packb`, `from_bytes = msgpack.unpackb`) to encode and decode channel layer messages that sbomify writes to its own private Redis (sbomify/settings.py CHANNEL_LAYERS, RedisChannelLayer with the default msgpack serializer). Those messages are built server side only, by `broadcast_to_workspace` in sbomify/apps/core/utils.py, as dicts with string keys, so `unpackb` never meets the non-string map key that triggers the `strict_map_key` rejection and the leak. WebSocket clients cannot inject channel layer messages: `WorkspaceConsumer.receive_json` in sbomify/apps/core/consumers.py only logs client input and never forwards it. The second issue in the advisory (stale parser stack references after `Unpacker.skip`) needs the streaming `Unpacker` API, which channels-redis and sbomify do not use. Writing crafted data into the channel layer would require write access to the Redis instance itself. The upgrade to msgpack 1.2.3 will still follow through the normal dependency updates.



An AI agent drafted this advisory

Only your workspace can see it. Read it through, then publish it to put it on your Trust Center. After that, every change an agent makes waits for your approval.

 Publish advisory

Product statuses

What your Trust Center and CSAF say about each product.

+ Add product

PRODUCT	STATUS	FIXED IN	WHAT USERS SHOULD DO
sbomify Action	Fixed	sbom-tools tools-26.10.0	Fixed in sbom-tools release tools-26.10.0 (commit a7fb4d4, PR https://github.com/sbomify/sbom-tools/pull/48), whose cdxgen and dotnet bundles ship brace-expansion 5.0.12 and fast-uri 3.1.8. sbomify Action fetches these bundles at runtime; no action release pins an older bundle set.

DETAILS

Status	Resolved
Severity	MEDIUM
CVSS	4.0
Type	CVE
Identifier	CVE-2026-102277
Created	22 Oct 2026
Last updated	26 Oct 2026

AFFECTED PRODUCTS

- sbomify Action
 - Before sbom-tools tools-26.10.0

Timeline

14 updates



Add an update...

Note only

 Post update

Supports Markdown: **bold**, *italic*, ``code``, lists, headings and [\[links\]\(https://example.com\)](https://example.com).

[Trust center](#)

sbomify

Security advisories, software bills of materials, and compliance artifacts for everything we ship, published as we publish them.

5

Advisories

3

Products

0

Compliance artifacts

25 Oct 2026

Latest advisory

Security advisories

Vulnerabilities we have disclosed, what they affect, and where the fix is.

[All advisories](#)

SBOMIFY-SA-2026-0005

Out-of-bounds write in the Brotli decoder in the sbomify Action image (CVE-2026-111497)

sbomify Action container images 26.7.0 to 26.10.1 ship Debian libbrotli1 1.1.0-2+b7, whose Brotli decoder has an out-of-bounds write (CVE-2026-111497) reachable through git's HTTPS transport. Fixed in sbomify Action v26.10.2.

✳ CVE-2026-111497

sbomify Action

Critical

✓ Resolved

25 Oct 2026

SBOMIFY-SA-2026-0004

Out-of-bounds write in the Brotli decoder in sbomify (CVE-2026-111497)

sbomify v26.2.0 to v26.9.0 bundle brotli 1.2.0, whose decoder has an out-of-bounds write reachable through Brotli-compressed responses to outbound HTTP calls. Fixed in sbomify v26.10.0 (brotli 1.2.1); app.sbomify.com was fixed on 2026-10-24.

✳ CVE-2026-111497

sbomify

Critical

✓ Resolved

24 Oct 2026

SBOMIFY-SA-2026-0003

Compromised eMudhra emSign intermediate CAs trusted by sbomify and sbomify Action (GHSA-6q48-6c7w-9gw2)

certifi, pip's vendored certifi, the sbomify image's OS CA bundle, sigstore and the JDK in sbomify Action's jvm tool bundle trusted eMudhra emSign roots whose intermediate CA keys were compromised, allowing TLS interception of outbound connections by an attacker holding those keys. Fixed in sbomify v26.10.0 and sbomify Action v26.10.1; the jvm tool bundle was fixed on 2026-10-26 for every Action release; app.sbomify.com is fixed.

sbomify

sbomify Action

Medium

✓ Resolved

23 Oct 2026

Questions?

301.vpetersson.com

viktor@sbomify.com

github.com/sbomify/sbomify

github.com/sbomify/sbomify-action

sbomify.com