OpenSSL Foundation

The Security Response Process of the OpenSSL Library

Tomáš Mráz

Prague – 15 October 2026

Agenda

- Why do we need it
- Who are the key stakeholders
- Security Response Team
- Security Policy
- How we do it currently
- How AI changes things

Why do we need it?

It might seem obvious…

So the bad actors do not get to know the vulnerability before it is fixed.

But is it really what is happening with ubiquitous access to AI?

Key stakeholders

Our users

(in a broad sense everyone that depends on the security of our software)

Security researchers

(those who find the issues)

Security Response Team

(those who assess the severity)

Engineering Team

(those who develop, review and publish the fixes and advisories)

MITRE

(the CVE database)

Bad guys

(those who want to get benefit from exploiting unfixed vulnerabilities)

Security Response Team

A group designated by the Foundation &

Corporation directors

Mostly people with Committer status

They:

- Investigate and review the security reports
- Assess the severity of the reported issues
- Meet weekly online

  (the assessment usually happens at that time

  unless the report looks urgent)

Security Policy

The guiding document for the security response process

https://openssl-library.org/policies/general/security-policy/

How do you report an issue?

Report to openssl-security@openssl.org

Send one issue per email

The Security Response Team (SRT)

- Acknowledges the report
- Investigates it
- Decides whether it is a real security issue, bug or security hardening issue, or an invalid report.

(We usually ask you to create a public GitHub issue or PR if it is not a security issue.)

The threat model sets the baseline

RCE/DoS/Data Leaks: Attacker must be able to trigger the issue in otherwise working application by supplying untrusted input

- Private keys are considered trusted
- Configuration is considered trusted

Or a legitimate behavior of application causes data that is expected to be authenticated or encrypted to be unauthenticated or unencrypted

Or there are missing or incorrect authorization checks

Timing side-channels exploitable over remote link

Outside scope of our threat model

Same-system side channels and hardware flaws

Physical fault injection and physical side channel observation

Denial of Service in the OpenSSL command-line tool

Certain application API misuse

Private keys supplied by attackers

Mitigations may still be useful without a CVE

Severity reflects impact and real-world exposure

Critical	RCE or critical data leaks in common configurations & protocols
High	Lower exposure or exploitability
Moderate	Limited impact, local flaws or uncommon protocols
Low	Restricted impact or unlikely conditions

Critical:	immediate security release is triggered
High:	a security release is triggered within a month
Moderate:	grouped into a later release
Low:	grouped into a later release, might be developed in public

Severity levels – how users should react

Critical & High	Update as soon as possible
Moderate & Low	Schedule the updates without undue haste

Severity and CVSS

We do not assign CVSS scores to the issues and do not use CVSS for assessing the severity.

Examples:

CVE-2026-72897 – Out-of-Bounds Access After SSL_set_SSL_CTX() During a Handshake

Our severity rating: Low – Extremely unlikely configuration and application scenario, fixed, small value out of bounds write, extremely unlikely to be exploitable as DoS, even less possible to cause RCE

CVSS 3.1 score from NVD: 7.5 High

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Severity and CVSS

We do not assign CVSS scores to the issues and do not use CVSS for assessing the severity.

Examples:

CVE-2026-75803 – AEAD Forgeries with Empty Ciphertext When Using EVP_Cipher()

Our severity rating: Low – Extremely unlikely application scenario. We are not aware of any application passing empty ciphertexts to EVP_Cipher(), the API is documented as dangerous and not to be used.

CVSS 3.1 score from NVD: 9.1 Critical

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

The process

- Initial triage
- Investigation
- SRT assessment
- CVE assignment
- Fix
- Advisory
- Security release
  - Distributions and customers prenotification
  - Public prenotification
  - Release

Fixes must preserve compatibility

Fix the latest affected branch

Backport to older affected branches as applicable

Stable updates allow security fixes and bug fixes

Minor and patch releases must preserve API and ABI

The backports are usually straightforward unless they go as far as 1.0.2 extended support releases

Sometimes tests are not backported if they require testing infrastructure that is not present on older branches

Information in advisories

Issue summary

Impact summary

CWE (Common Weakness Enumeration) id

Description

FIPS impact

Vulnerable branches

Fixed versions

Reporters – Name (Association)

Fix developers – Name (Association)

Release preparation

Selected organisations (distributions, premium support customers): details about the vulnerabilities and draft fixes usually two weeks before release

Public notice usually one week before release

Tree frozen 3 days before release

Release day

Passing CI and resolved release milestones

Release automation triggered

Release commits reviewed & approved

Fixes pushed from the private to the public repository

Release built, signed, artifacts attached to the release in GitHub

CVE information pushed to MITRE

Advisory and release information publicly announced

How AI changes things

We started receiving AI reports in 2024

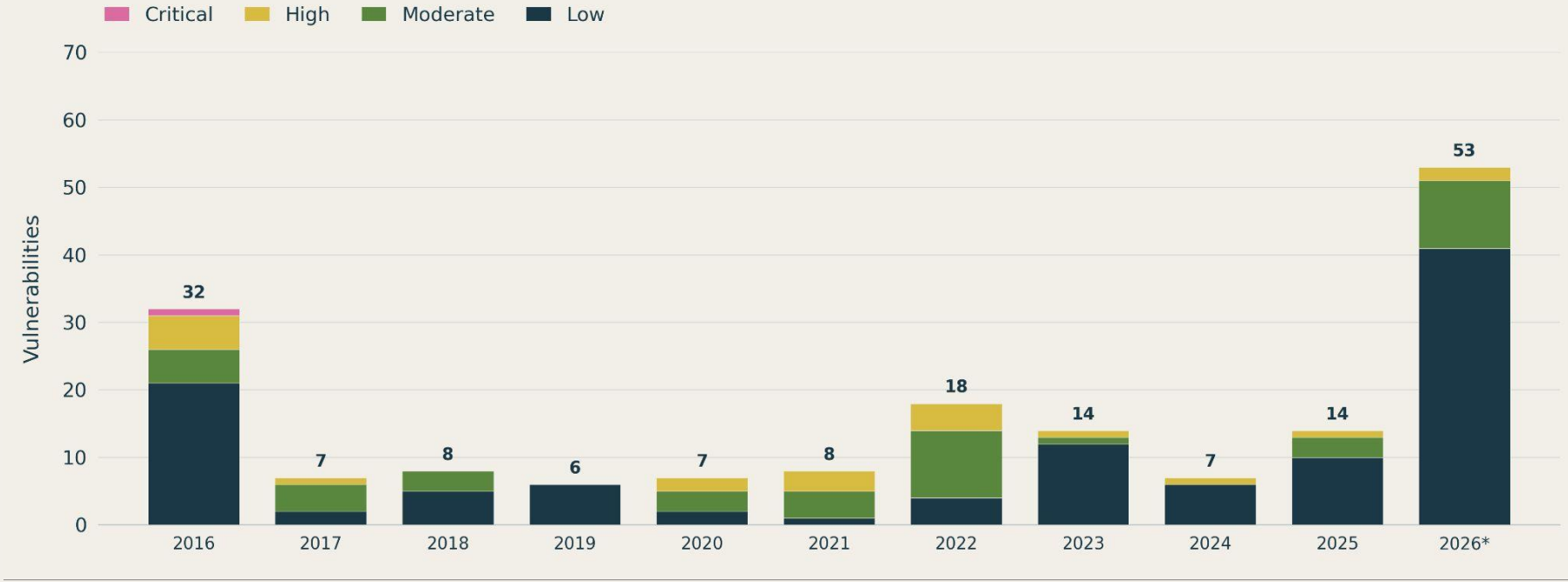
- Mostly slop

Second half of 2025

- First valuable reports from AISLE including PoC code and proposed fixes

In 2026 it exploded

How AI changes things



AI makes some things particularly easy

It is now very easy to ask AI to review open source code for vulnerabilities.

The number of reports multiplied since the availability of the new AI models in 2026.

The biggest challenges

- Duplicate reports (of both security and non-security issues)
- Invalid (or non-security), but seemingly plausible reports
- The embargo and release process carries big communication overhead

AI cannot replace human in the loop

At least not yet

AI can help with triage

> There can be an automated AI review of the reports, but somebody still needs to read and understand the review report.

AI can help with fixing

> Many of the reports generated by AI have proposed fixes. And we use the AI to write code. But still the author and two reviewers have to understand the fix.

The communication overhead in the process

Initial report acknowledgement

- Could be automated

Rejection/Severity & CVE assignment/Ask for reporter's credentials

- Not fully automatable
- Should we stop giving credits to duplicate reports?

Ask for draft patch and advisory review

- Probably could be automated but not easily

Security release process overhead

Not terrible but it doesn't allow us to release too often

(at most once per month)

Too many Low severity issues in a release

Handling them in public helps but we changed the policy to handle them under embargo by default recently.

Should we change it back?

More radical thoughts

Is the embargo process obsolete now?

Even if the more capable closed models block against misuse all the bad actors have already access to the open weight models that are more than capable to find and exploit these vulnerabilities if prompted properly.

However the Moderate & Low issues are either not exploitable in common applications or configurations or the impact is minor.

More radical thoughts

Shorten the prenotification for High & Critical

Do immediate release for High & Critical

Do Moderate & Low in public

Create a fixed release schedule for Moderate & Low

Thank you to our leading supporters!

PREMIER SUPPORTERS

CODE PROTECTORS

Questions & Discussion