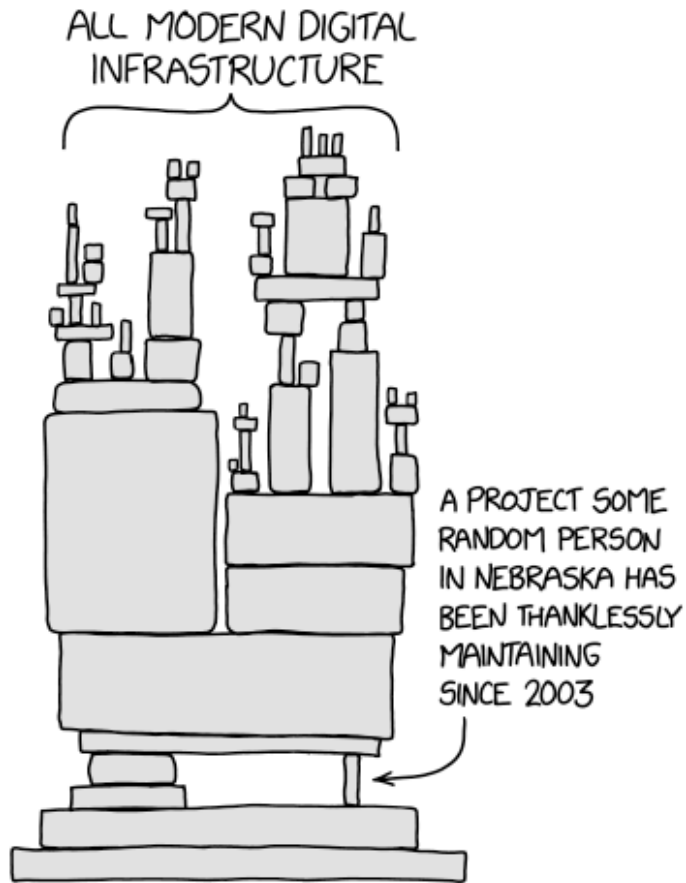




Finally. *Trust* in Open Source

Leveraging AI and Automation to Automatically Remediate Risks and Other Discovered Issues in the Software Supply Chain



<https://xkcd.com/2347/>

The Problem with AI

1. Dependency Explosion

AI can create risk faster than ever before.

2. Vulnerability Explosion

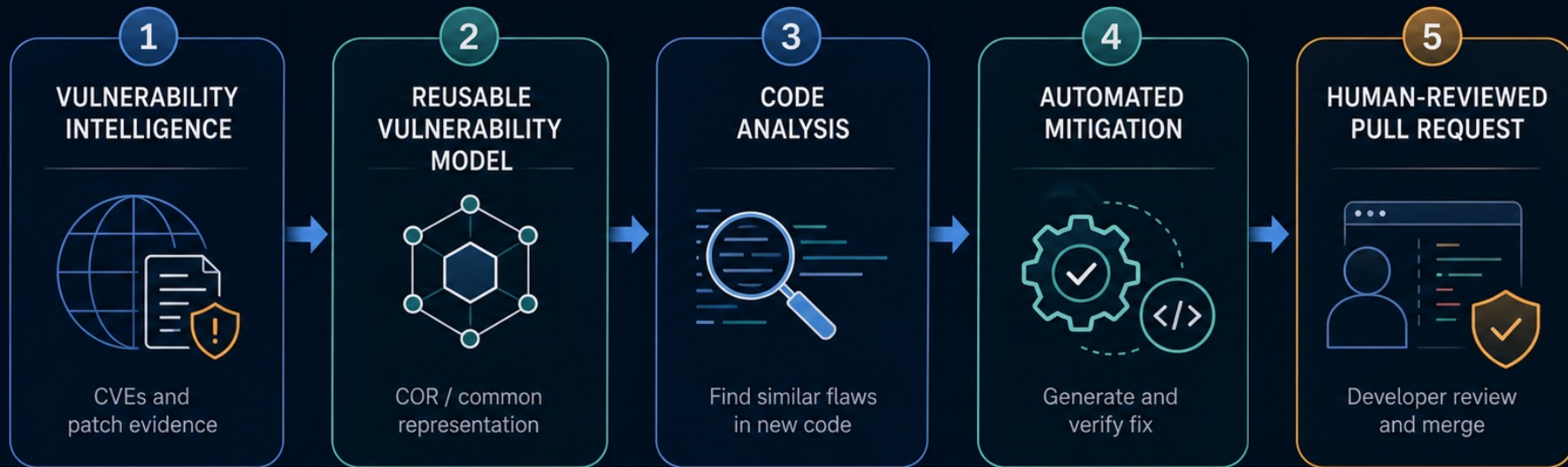
AI is discovering problems faster than ever before.

3. Breakdown in Workflows

Humans cannot decide or act fast enough to keep up.



Automated CVE Discovery & Patching



Uncovering Risk

Package | ring@0.17.14

https://bpt-beta.darksky.technology/project/prj_d46eed42781248c5bc8b69

BULLETPROOF TRUST

Contributors

Name	Contributions	Risk	Country
Alfonette Alve	8	High	China
Tarrel Antonyo	6	Medium	USA
Coleen Bitania	2	Low	USA
Noelany Krystyne	4	Low	USA
Mikel Tahje	1,714	Low	USA
Bryse Chubby	5	High	USA
Willanna Naksh	5	Low	USA
Nyshay Sherkia	21	Low	USA
Janiqua Jocilynn	39	Low	USA
Modupe Rafelita	18	Medium	USA
Zyriana Gaither	20	Low	USA

Rows per page: 100 1-100 of 341

Contributor Intelligence

Nationality | Location | Associations | Connections

Contributors

All Contributors • Alfonso Alve

Overview

Alfonso Alve

Current Location: China, Asia

Contributor ID: cntr_174b8ad3fd2e43a19d00bfa7f186423f

Contributions: 8

Risk: High

Locations: 1

Organizations: huawei technology

Alerts

Contributions from Banned Entities

This package has contributions from the following entities of concern: Huawei Technology.

These contributions were found in pkg:cargo/ring@0.17.14.

Details

Using packages with contributions from entities on banned lists can expose organizations to legal and compliance risks, as it may violate trade regulations and sanctions, leading to potential fines and legal consequences. Dependencies on these packages may lead to supply chain vulnerabilities, as these entities might not adhere to security best practices, increasing the risk of introducing security flaws into your

Contributions by Location

Current City Current Location Previous Locations

Map showing contributions by location, highlighting China.

Contributor Network

Network diagram showing connections between contributors.

Other Package Contributions

Filter

- e2fsprogs@1.44.1-lubuntu4
- e2fsprogs@1.45.5-2ubuntu1.2
- e2fsprogs@1.47.0-2
- e2fsprogs@1.47.0-2+b2
- e2fsprogs@1.47.0-2+b2
- e2fsprogs@1.47.0-2.4-exp-lubuntu4.1
- e2fsprogs@1.47.1
- e2fsprogs@1.47.1-r0
- e2fsprogs@1.47.3
- e2fsprogs@1f66b8f236f63e75e2c00c1b89ea
- e2fsprogs@2bad80a3131b4d4d44a70b007
- e2fsprogs@28d22c467547f1f1f6db3131549ee
- e2fsprogs@2a214d1bb0ad550abdcce7f63812e
- e2fsprogs@2a0a607543861c0acd7ed8d4d42
- e2fsprogs@2a0d086b8ec78075767673b7e0012
- e2fsprogs@2c7d4f0f67688abdbb411a11b

1-20 of 957

Package File Contributions

crypto/x509/x_name.c



Contributor Intelligence

Nationality | Location | Associations | Connections

Scanners



SAST

Static App Scans



License

© + License Scans



Quality

Document + Code



OpenSSF

Scorecard Score



FOCI

Sanctioned Entities



Vuln.

30+ Databases

Repo Intel and Trends



Package Info

Popularity, Maturity, Bugs, etc.

Reachability



PACKAGE PRESENT

Assigned Vulnerable

Your Application



The package is in your SBOM

Detected by scanners and listed as vulnerable



No insight into how it is used

Static presence does not equal exploitability



Unclear risk without context

Requires analysis to determine real impact

SIGNAL

Needs investigation

VS



ACTUAL REACHABLE BEHAVIOR

In Your Application Context

Your Application



A valid path exists

Dependency and code analysis confirm reachability



The vulnerable behavior can be triggered

Real exploit conditions are present

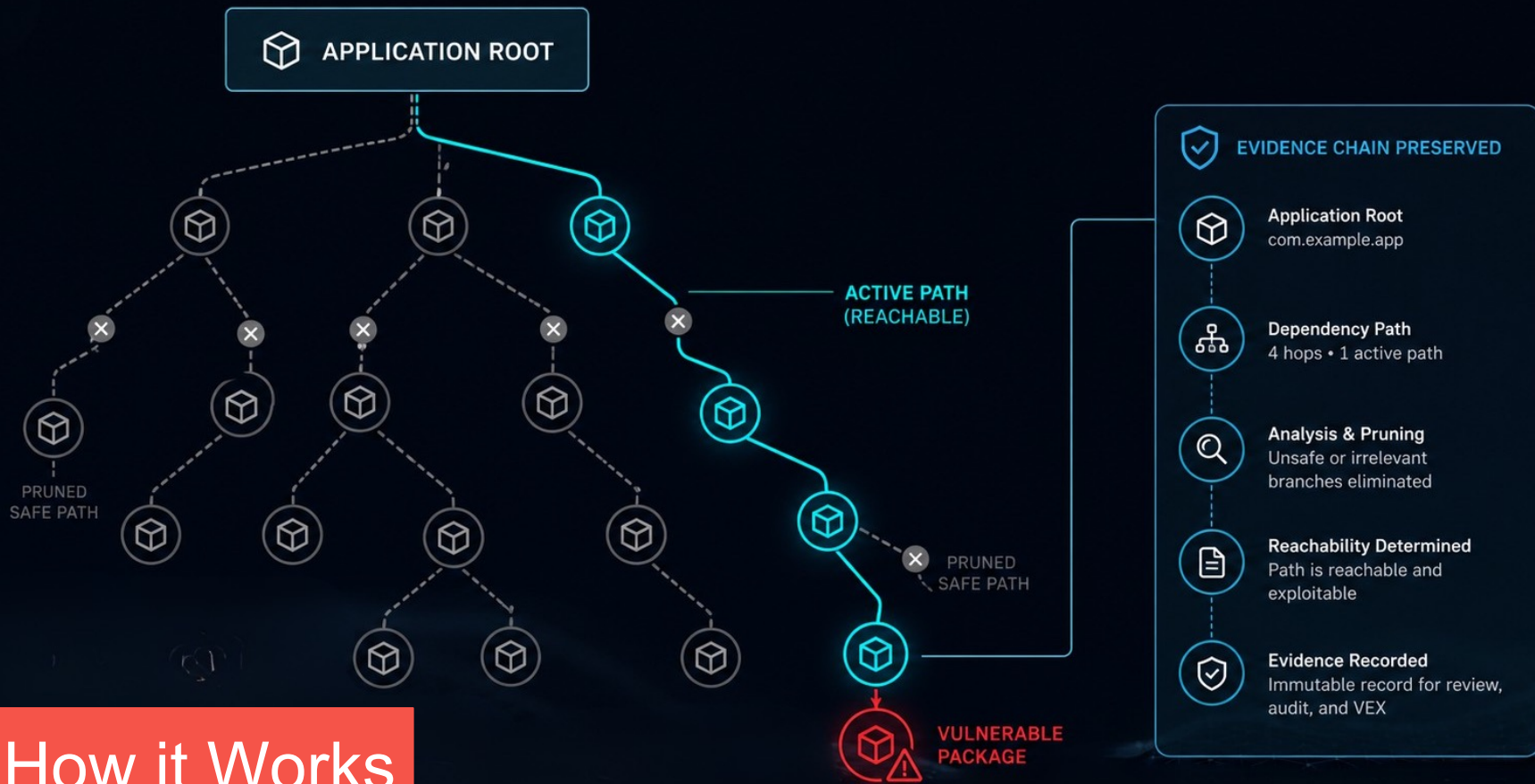


This is real risk

Actionable, priority, and defensible

REAL RISK

Requires action



How it Works

Real World Example

				Download CSAF VEX
				Reset
				Status
s-webpki@0.101.7	7.50	-	Pending Analysis	
hssl@0.10.78	8.70	-	Pending Analysis	
nl@0.0.5	8.70	-	Not Affected	
@8.3.2	7.50	0.00	Pending Analysis	
-xml-builder@1.1.5	8.70	-	Pending Analysis	
-uri@3.1.0	7.50	0.00	Pending Analysis	
Rows per page: 10				1-10 of 39

VEX Disposition

Real World Example

The screenshot shows the Bulletproof Trust web application interface. The top navigation bar includes the logo and a status indicator. The main content area displays a table of vulnerabilities. The table has columns for Severity, Identifier, and Package. The first three rows show vulnerabilities with a severity of HIGH. The first row is for GHSA-82j2-j2ch-gfr8, the second for GHSA-xp3w-r5p5-63rr, and the third for GHSA-gfxp-f68g-8x78. Each row has buttons for '1 alias' and '1 related'. The fourth row shows CVE-2026-41907. The fifth row shows GHSA-5wm8-gmm8-39j9. The table is filtered by 'HIGH' severity and 'ACTIVE' status. The page title is 'High Impact Vulnerabilities Detected'.

Severity	Identifier	Package
HIGH	GHSA-82j2-j2ch-gfr8 1 alias 1 related	rustls-w
HIGH	GHSA-xp3w-r5p5-63rr 1 alias 1 related	openssl
HIGH	GHSA-gfxp-f68g-8x78 1 alias	libyaml@
HIGH	CVE-2026-41907 1 alias 1 related	uuid@8.3
HIGH	GHSA-5wm8-gmm8-39j9 1 alias 1 related	fast-xml-builder@1.1.5

Edit Vulnerability GHSA-gfxp-f68g-8x78

Update the VEX fields for this vulnerability: set its current status, explain any mitigating justification, choose the planned responses, and capture supporting notes before saving.

Status

Not Affected

Justification

Responses

Notes

Path: cyclone-dx@0.1.0 -> spanned@0.1.0 -> serde_yaml@0.0.12 -> libyaml@0.0.5

Reasoning summary

This crate does not preserve external reachability to the vulnerable `serde_yaml/libyaml` parser path. Its only public parse endpoints are `json::Bom::from_str` and `xml::Bom::from_str`, which forward caller-controlled input to `spanned::from_pretty_json` and `spanned::from_pretty_xml` respectively. The vulnerable helper `spanned::from_pretty_yaml` exists only in the transitive `spanned` dependency and is neither imported, called, nor re-exported by `cyclone-dx`, so a downstream user would need a separate direct dependency on `spanned` to reach that signature.

Evidence

- Cargo.toml:6-18 declares dependencies on `quick-xml`, `serde`, `serde_json`, and `spanned`; `cyclone-dx` has no direct `serde_yaml` dependency.
- `src/lib.rs:64-68` imports `spanned::(Pretty, Spanned, from_pretty_json)` for the `json` module.
- `src/lib.rs:257-264` implements `json::Bom::from_str` by calling `from_pretty_json(s)`, not `YAML` parsing.
- `src/lib.rs:278-282` imports `spanned::(Pretty, Spanned, from_pretty_xml)` for the `xml` module.
- `src/lib.rs:559-566` implements `xml::Bom::from_str` by calling `from_pretty_xml(s)`, not `YAML` parsing.
- `/tmp/.tmp4WnjZH/lib.rs:118-126` defines the vulnerable helper `pub fn from_pretty_yaml(t: DeserializeOwned>(s: &str) -> Result<(T, Pretty), serde_yaml::Error> { serde_yaml::from_str(s)?; ... }`.
- Repository search in `cyclone-dx` found no references to `from_pretty_yaml` or `serde_yaml`; only `from_pretty_json/from_pretty_xml` call sites are present.
- cargo tree -e normal shows `serde_yaml` and `libyaml` only under the transitive dependency chain `cyclone-dx -> spanned -> serde_yaml -> libyaml`.

Path: `spdx3@0.1.0 -> spanned@0.1.0 -> serde_yaml@0.0.12 -> libyaml@0.0.5`

Reasoning summary

This package does include the vulnerable `serde_yaml/libyaml` stack transitively via `spanned`, but its own externally reachable parsing surface does not forward caller-controlled input into that `YAML` path. The only text-

Cancel

Save

Pending Analysis

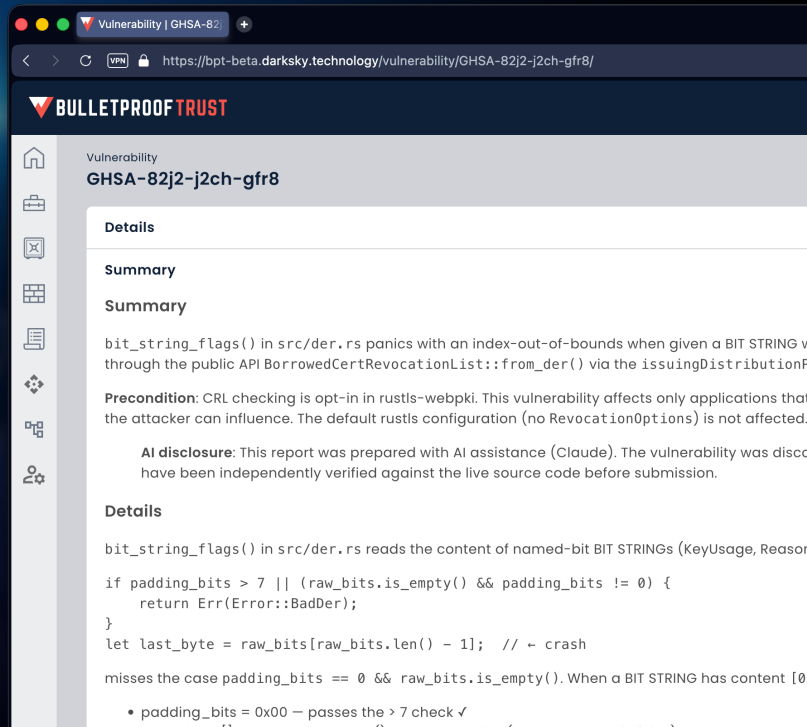
Real World Example

Example vulnerability

GHSA-82j2-j2ch-gfr8 in rustls-webpki@0.101.7

Point of the example

This is not manual ticket work. It is combinatorial path analysis, and it creates a serious engineering review burden without structured automation.



The screenshot shows a web browser window with the address bar displaying `https://bpt-beta.darksky.technology/vulnerability/GHSA-82j2-j2ch-gfr8/`. The page header features the "BULLETPROOF TRUST" logo. The main content area is titled "Vulnerability GHSA-82j2-j2ch-gfr8" and includes a "Details" section with a "Summary" subsection. The summary text states: "bit_string_flags() in src/der.rs panics with an index-out-of-bounds when given a BIT STRING v through the public API BorrowedCertRevocationList::from_der() via the issuingDistributionF". A "Precondition" note mentions that CRL checking is opt-in in rustls-webpki. An "AI disclosure" note states the report was prepared with AI assistance (Claude) and was independently verified. The "Details" section contains a code snippet for `bit_string_flags()` and a note that it "misses the case padding_bits == 0 && raw_bits.is_empty()". A bullet point at the bottom indicates "padding_bits = 0x00 - passes the > 7 check ✓".

Vulnerability | GHSA-82j2-j2ch-gfr8

<https://bpt-beta.darksky.technology/vulnerability/GHSA-82j2-j2ch-gfr8/>

BULLETPROOF TRUST

Vulnerability
GHSA-82j2-j2ch-gfr8

Details

Summary

Summary

bit_string_flags() in src/der.rs panics with an index-out-of-bounds when given a BIT STRING v through the public API BorrowedCertRevocationList::from_der() via the issuingDistributionF

Precondition: CRL checking is opt-in in rustls-webpki. This vulnerability affects only applications that the attacker can influence. The default rustls configuration (no RevocationOptions) is not affected.

AI disclosure: This report was prepared with AI assistance (Claude). The vulnerability was disc have been independently verified against the live source code before submission.

Details

bit_string_flags() in src/der.rs reads the content of named-bit BIT STRINGS (KeyUsage, Reason

```
if padding_bits > 7 || (raw_bits.is_empty() && padding_bits != 0) {
    return Err(Error::BadDer);
}
```

let last_byte = raw_bits[raw_bits.len() - 1]; // ← crash

misses the case padding_bits == 0 && raw_bits.is_empty(). When a BIT STRING has content [0

- padding_bits = 0x00 - passes the > 7 check ✓

Real World Example

1,211

independent paths

from service root to a single vulnerable component



> 12 Hours

Example vulnerability

GHSA-82j2-j2ch-gfr8 in rustls-webpki@0.101.7

Point of the example

This is not manual ticket work. It is combinatorial path analysis, and it creates a serious engineering review burden without structured automation.

Real World Example

35

independent paths

from service root to a single vulnerable component



3 hours 30 minutes

Example vulnerability

RUSTSEC-2026-0097 in rand@0.9.2

Point of the example

This is not manual ticket work. It is combinatorial path analysis, and it creates a serious engineering review burden without structured automation.

Lessons Learned



1. REASONING QUALITY MATTERS

Deeper reasoning = better reachability dispositions and safer fixes



More analysis depth



Stronger evidence and confidence



Better dispositions and outcomes



2. LARGER LOCAL MODELS CAN WORK—JUST SLOWLY

Higher capability at the cost of runtime and resource demand



Longer run times



More compute and memory



Higher resource consumption



3. SMALLER MODELS BREAK DOWN

Shallower reasoning leads to gaps, missed paths, and risk



Incomplete analysis



Missed paths or false confidence



Higher risk, less trust

Optimization

EXAMPLE 1

Large Scale Analysis

rustls-webpki



1,211

Independent
Paths Analyzed



5h+

Analysis
Runtime



1

Reachable
Path Found



High complexity. High cost without automation.

EXAMPLE 2

Focused Analysis

smaller service



24

Independent
Paths Analyzed



12m

Analysis
Runtime



1

Reachable
Path Found



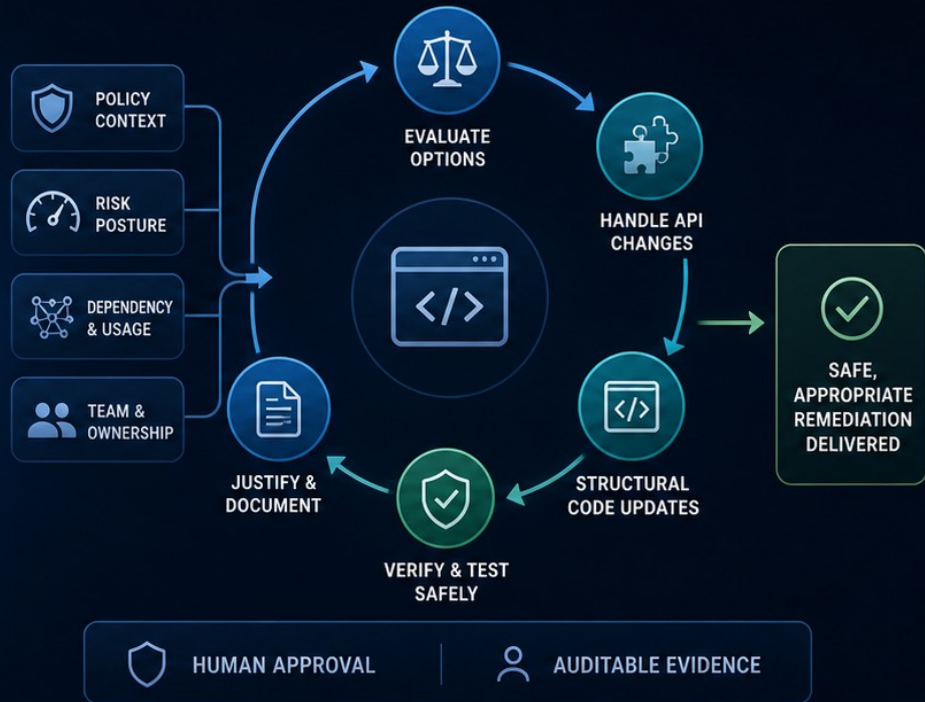
Lower complexity. Same outcome.

Why Automated Remediation?

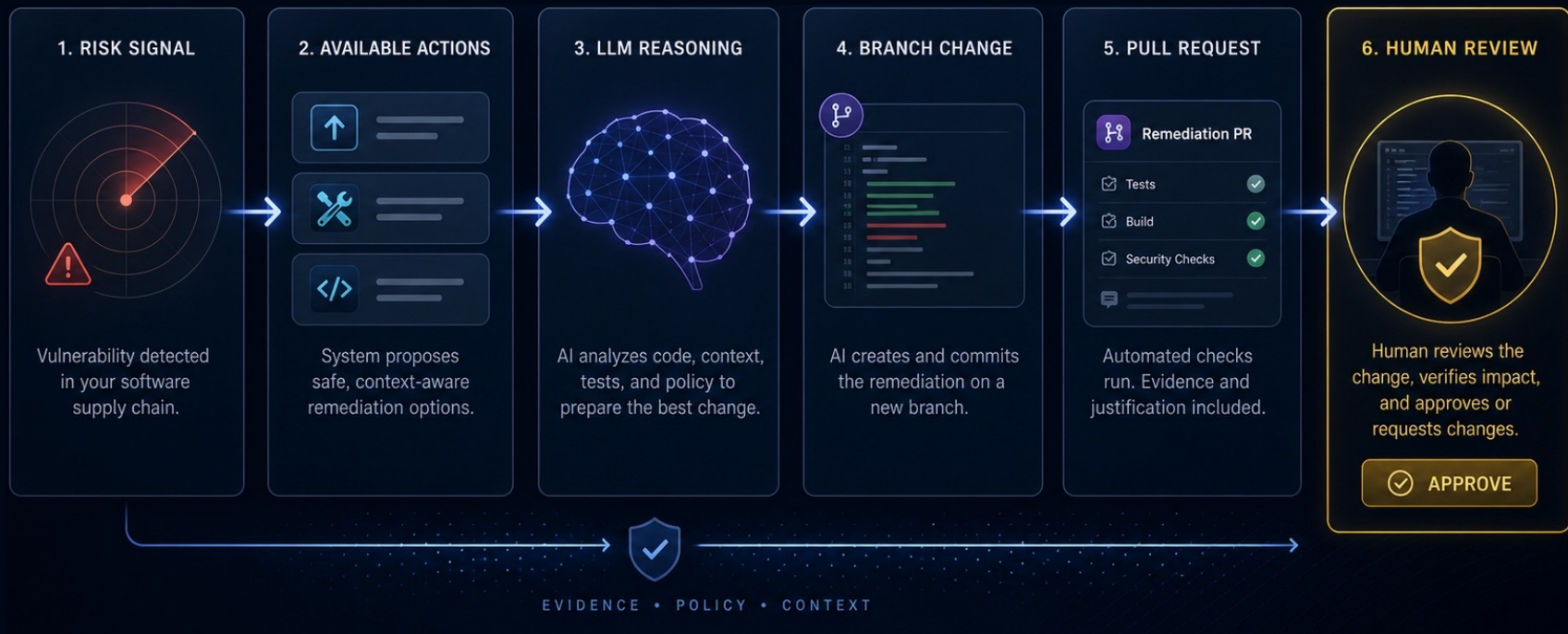


May not be the right fix
May introduce new risk
Doesn't address usage or code

vs.



How it Works



Other Use Cases



End-of-Life Component

No more upstream support.
Rising risk over time.



Action: Replace, Refactor,
or Remove



Vulnerable Component

Known vulnerability with potential
for exploitation.



Action: Patch, Upgrade,
or Refactor



Out-of-Date Component

Lagging behind current versions.
Increased exposure over time.



Action: Update to
Supported Version

Planned Work



1. REMEDIATION

- Propose and verify targeted code fixes
- Human-approved changes via PR workflow
- Reduce risk with evidence-backed action



2. INVESTIGATION

- Expand analysis to related components
- Trace impact, exploitability, and context
- Produce evidence for confident decisions



3. POLICY-GUIDED ACTION

- Apply plain-language policy to outcomes
- Recommend accept, mitigate, or defer
- Continuously improve with feedback



Finally. *Trust* in Open Source

INFO@DARKSKYTECHNOLOGY.COM