OWASP GLOBAL AppSec SF'26 November 2-6

25 years of open source security

THE HIDDEN EFFORT CURVE OF REMEDIATION

BRUCE FRAM · MICHAEL CARTSONIS

15,000 findings across 400 open-source projects

Welcome to the

TAYLOR SWIFT
THE ERAS TOUR

OWASP AppSec edition

ERA 1
Discovery
Find the bugs

ERA 2
Scoring
Rank the bugs

ERA 3
AI Automation
More findings

ERA 4
Remediation
Validate and ship the fixes

I started this because I am curious (and a nerd)

1

EARLY 2026 – Mythos Moment

One project - Firefox

AI experiment on public fixes showed a way to estimate remediation effort

2

NEXT

20 More projects + expert review

Smart people challenged the results; patterns started to appear

3

TODAY

200+ projects

A more detailed, more carefully checked method for enterprise decisions

The question for enterprise teams: how can you allocate resources to take the most risk off the table?

Dataset is Enterprise Focused

223 projects

≈15,200 vulnerability records

≈11,650 effort estimates

≈3,600
unknown effort

BREADTH

15 programming languages

6 application architecture categories

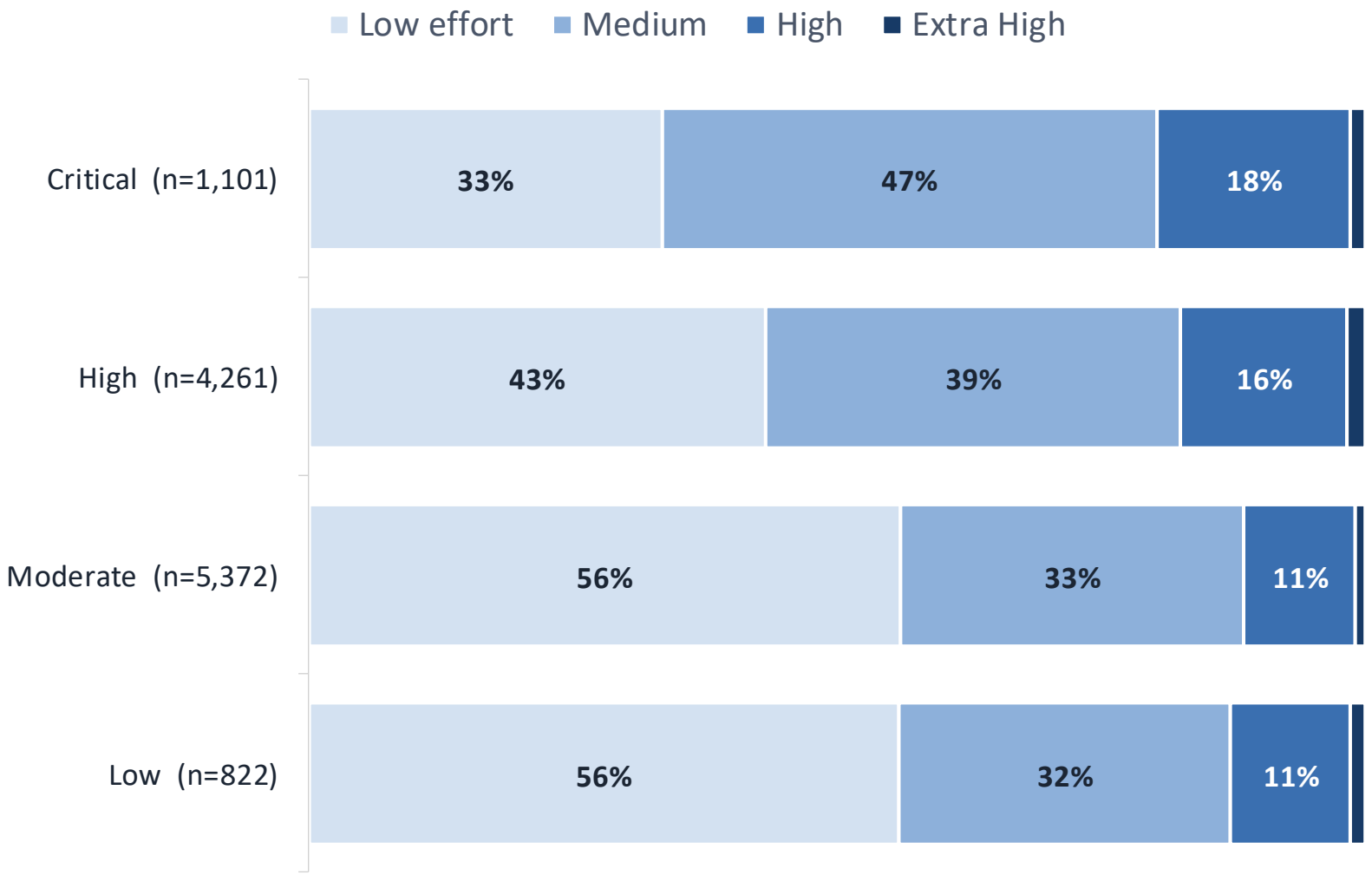
3 deployment categories

PROJECT EXAMPLES

Firefox · GitLab · Mattermost · Keycloak · Apache Tomcat · PostgreSQL · Django · Bouncy Castle · curl · Jenkins

Source: AppSecAI Open Source Remediation Dataset, October 31, 2026. Engineer-days are modeled from patch evidence, not timesheets.

Key Finding: Severity and effort are not related



LOW OR MEDIUM EFFORT

80% of Critical

83% of High

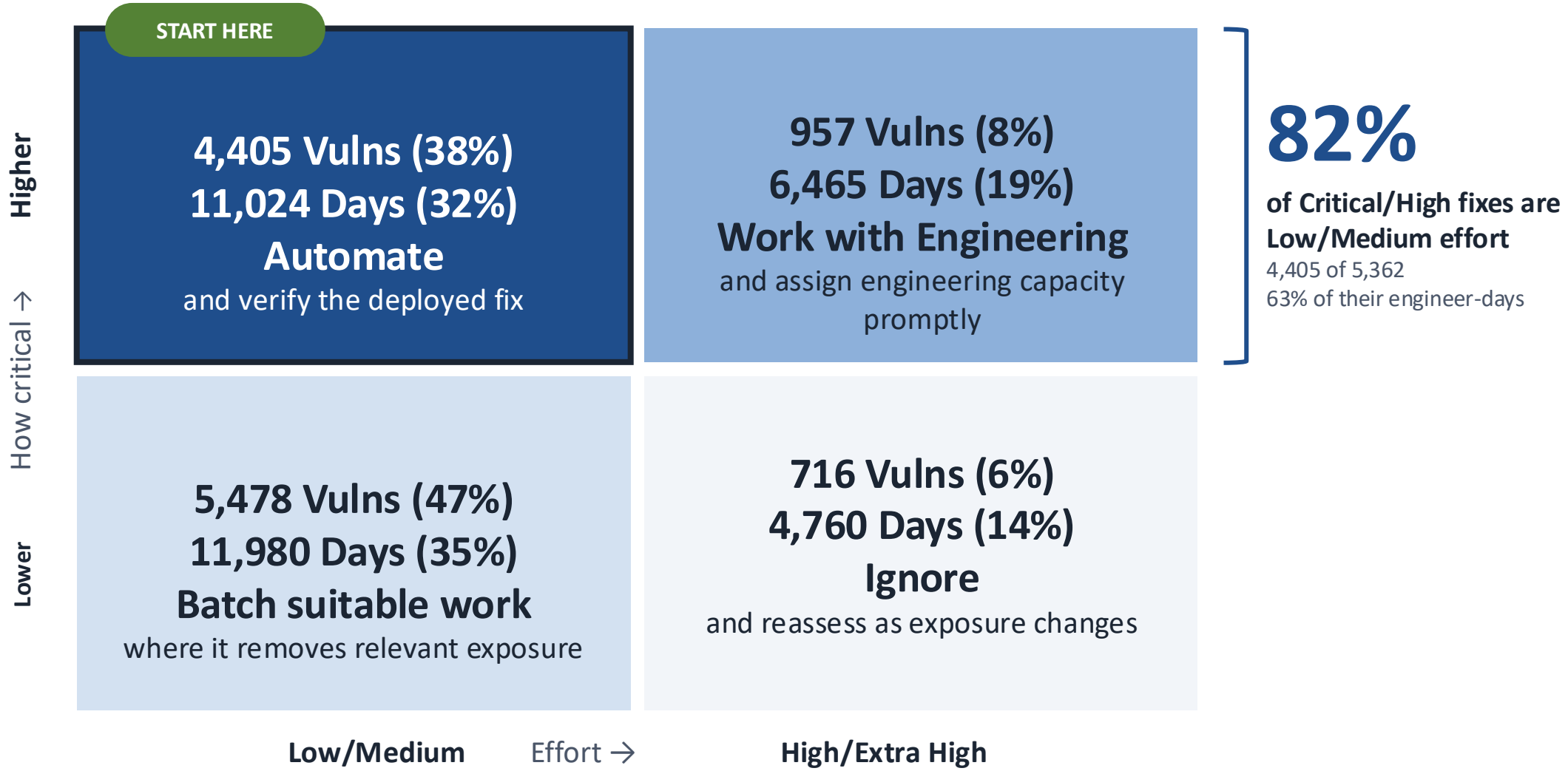
CORRELATION

0.11 Spearman ρ, severity vs. effort

0 = no relationship, 1 = perfect. Severity explains about 1% of the variation in effort ranking.

Published or CVSS severity, 11,556 estimated records. Spearman ρ ≈ 0.11 (severity vs. modeled days); numeric CVSS vs. days Pearson r ≈ 0.10 (n=10,172) [S1, S3].

82% of Critical/High Fixes Are Low/Medium Effort



Box percentages are shares of all 11,556 fixes and 34,229 modeled engineer-days. A practical application of the findings, not an optimization algorithm validated by the study [S3].

What is Effort?

1. **Match finding to fix**
2. **Measure retained implementation changes**
3. **Score D1 · D2 · D3**
4. **Allocate shared work once**
5. **Estimate effort band and engineer-days**

D1 scored 1–5
Implementation footprint

D2 scored 1–5
Reasoning complexity

D3 scored 1–5
Validation difficulty

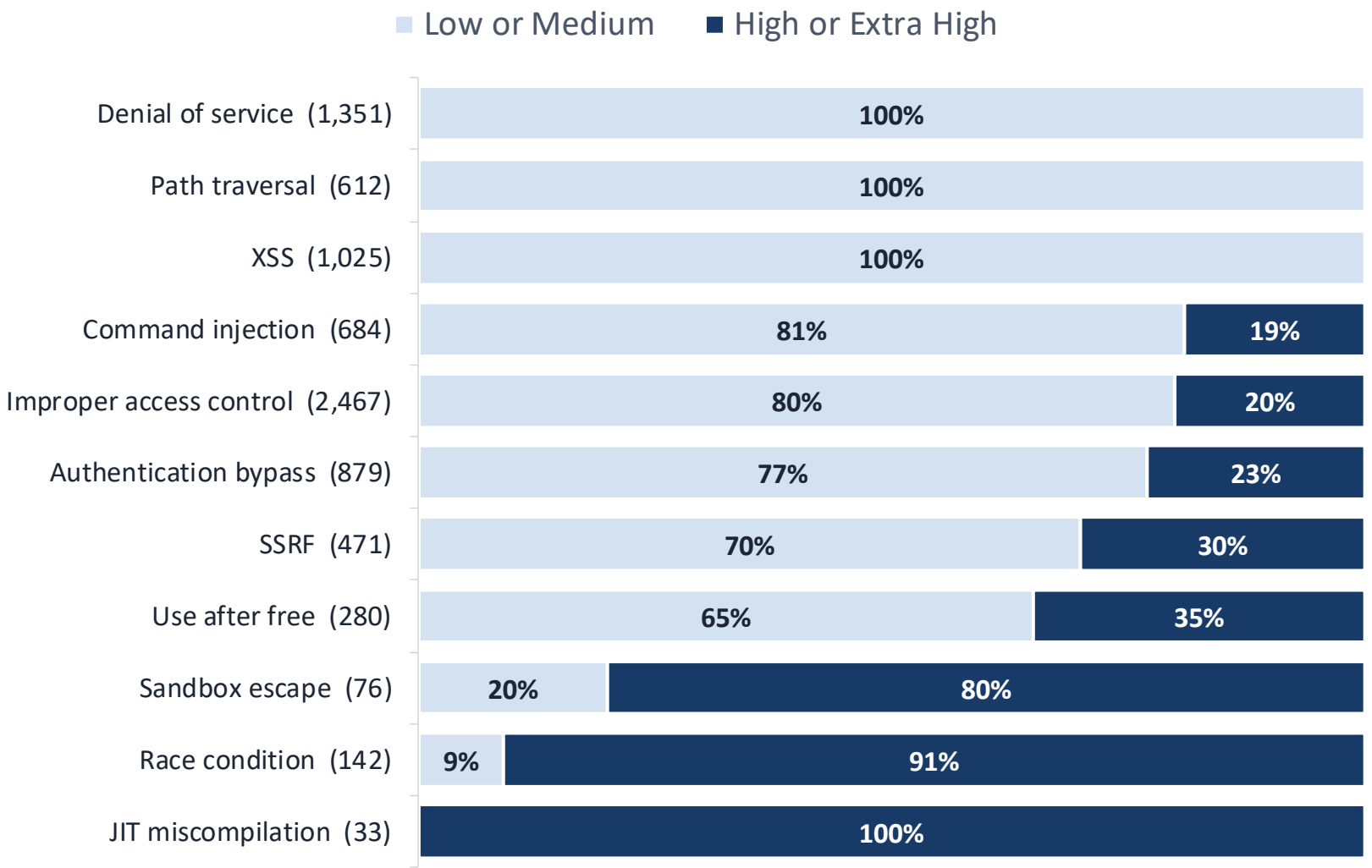
COMBINED SCORE → EFFORT BAND

Low	Medium	High	Extra High
3–7	8–9	10–11	12–15

Severity is not an input.

Method: AppSecAI dataset scoring rules [S1, S2]. Thresholds and calibration in backup.

Effort profiles differ by vulnerability class



CLASS FRAMES THE QUESTIONS

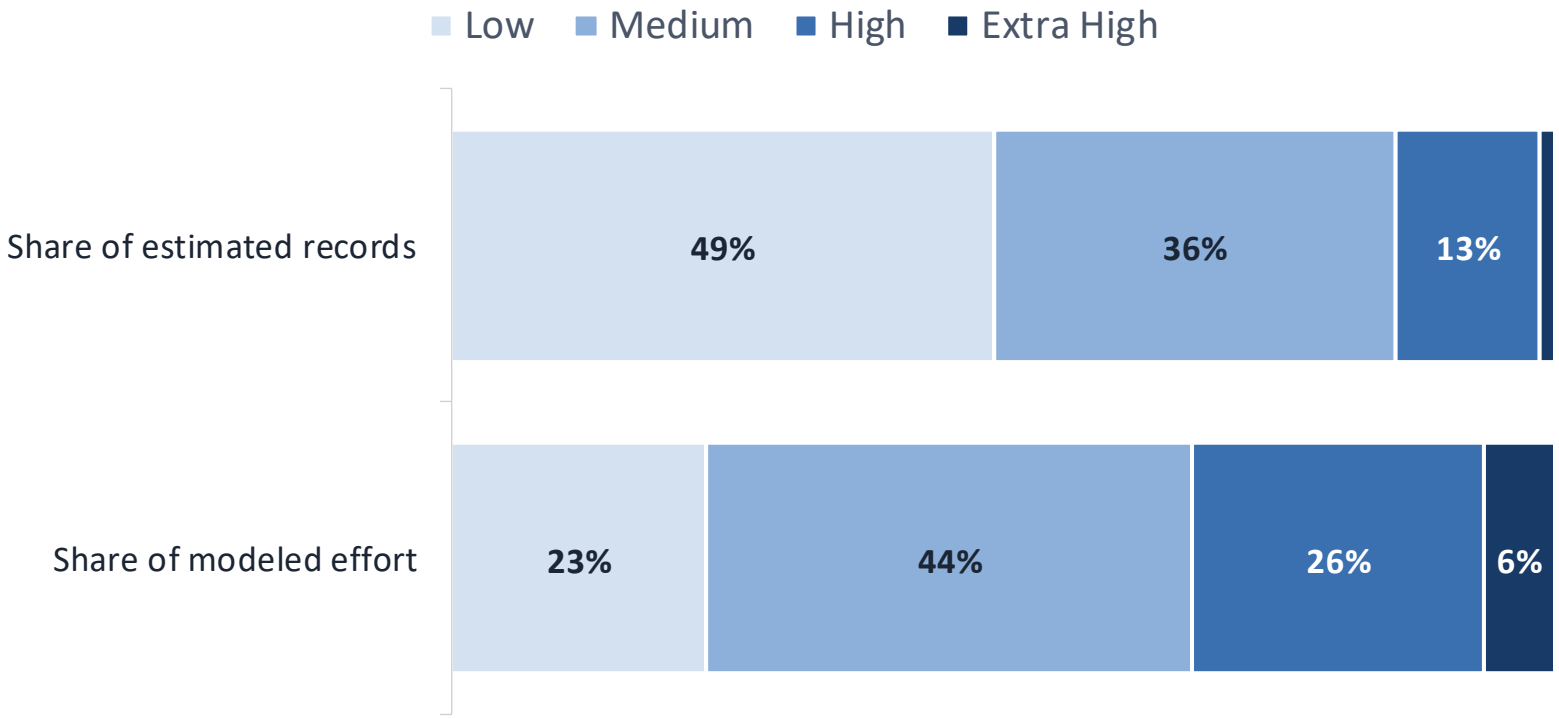
Input handling often has a reusable, validated fix pattern.

Boundary and state management (use after free, races, JIT) may need specialist reasoning and regression coverage.

Then inspect the actual patch and component before applying either expectation.

Class supplies model defaults: these profiles describe modeled estimates, not an independent test of actual hours [S1, S2].

A small difficult tail consumes much of the effort



14% → 33%

High + Extra High: 14% of records, 33% of modeled work

1% → 6%

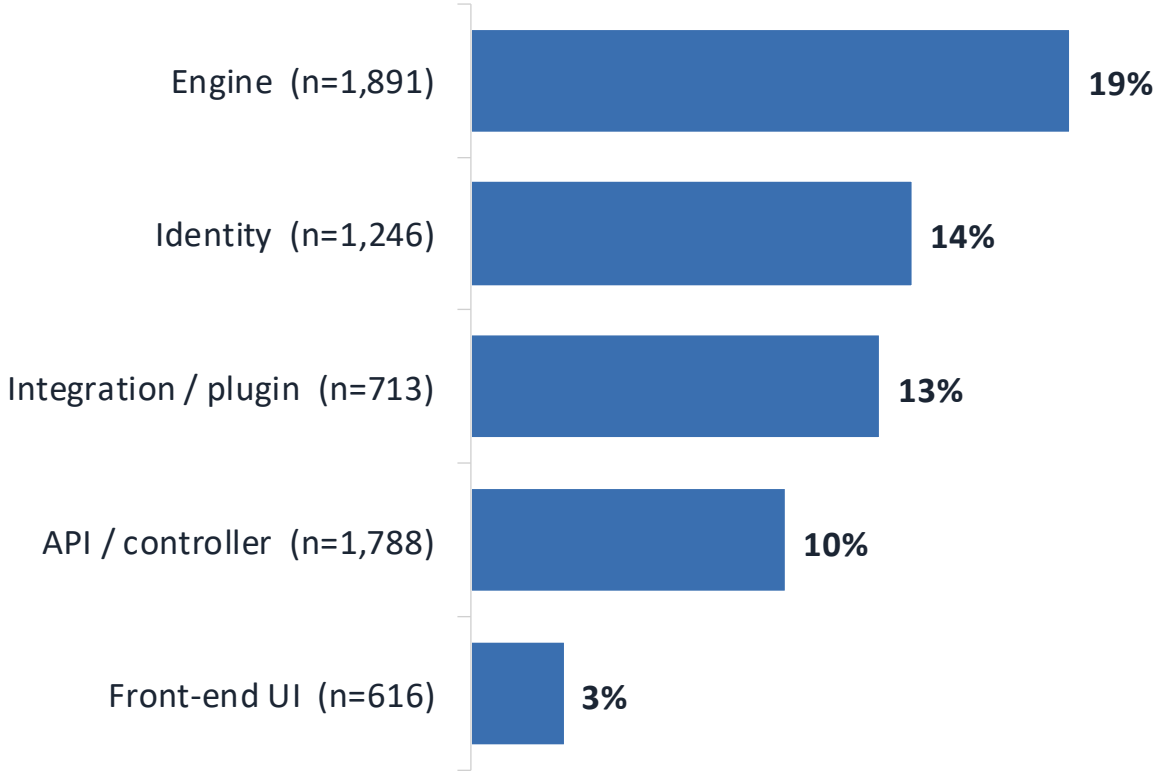
Extra High alone

Action: give difficult urgent work an owner, a reviewer, engineering time, and a containment decision. Keep modest material fixes moving.

11,652 estimated records, ≈34,476 modeled engineer-days. Historical modeled totals, not an annual staffing forecast [S1].

Type of Code has a big impact on Effort

HIGH OR EXTRA HIGH EFFORT, BY IMPLEMENTATION LAYER



FILES MATTER MORE THAN NAMES

Flowise component:
"Frontend Builder Web UI & Standard CRUD APIs"
14 records resolve to:

8 business-logic fixes

5 API / controller fixes

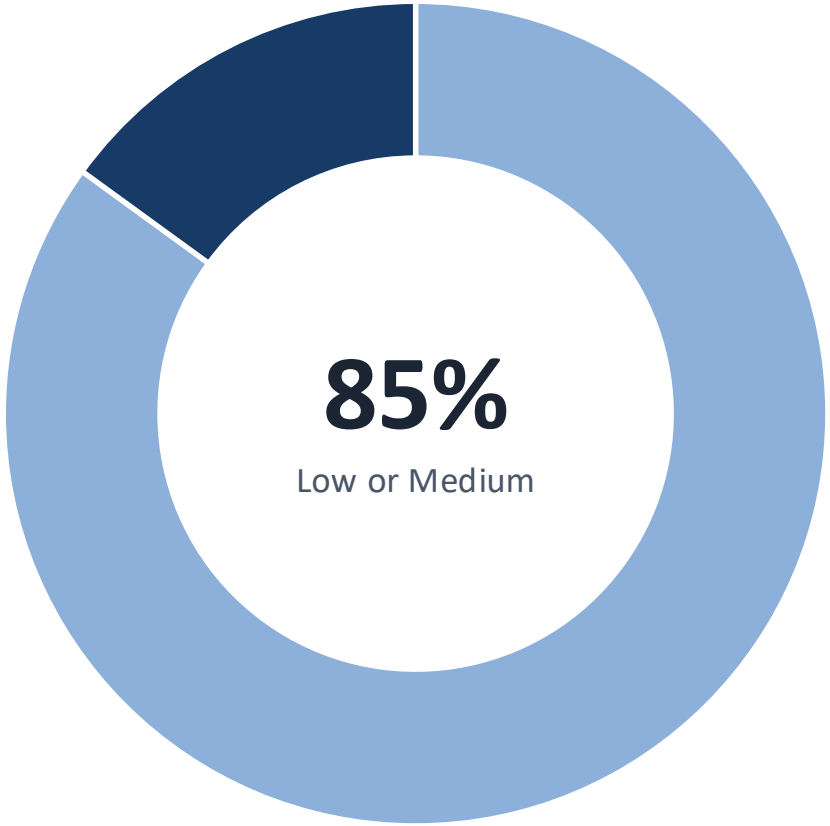
1 unclassified

0 front-end UI

Action: inspect implementation paths, pick the domain reviewer, and estimate the complete fix.

Descriptive group differences, influenced by project and class mix; 1,722 estimated records remain unclassified [S1, S2].

Anthropic-credited fixes look like everything else



Anthropic-credited records in this corpus

176 Anthropic-credited records with estimates, 27 projects

85% Low or Medium effort: Anthropic-credited

86% Low or Medium effort: the 11,442 records without credit

Project mix matters: Firefox's credited subset has a larger difficult tail than its noncredited subset.
Credit does not identify a particular model or prove autonomous discovery.

Credit data: Patrick Garrity (VulnCheck), Anthropic Credited CVEs repository (commit e8b88e2, October 31, 2026), and project advisories [S1, S4].

AI Discovery Creates a Human Review Queue

Anthropic coordinated disclosure

Snapshot October 2, 2026 (latest as of Nov 1)

29,439 candidate findings

6,157 reported to maintainers

516 **patched upstream**

≈2% of candidates patched upstream

OpenAI + Trail of Bits Patch the Planet

Snapshot October 30, 2026

2,055 reported issues

207 patches open upstream

472 **accepted upstream**

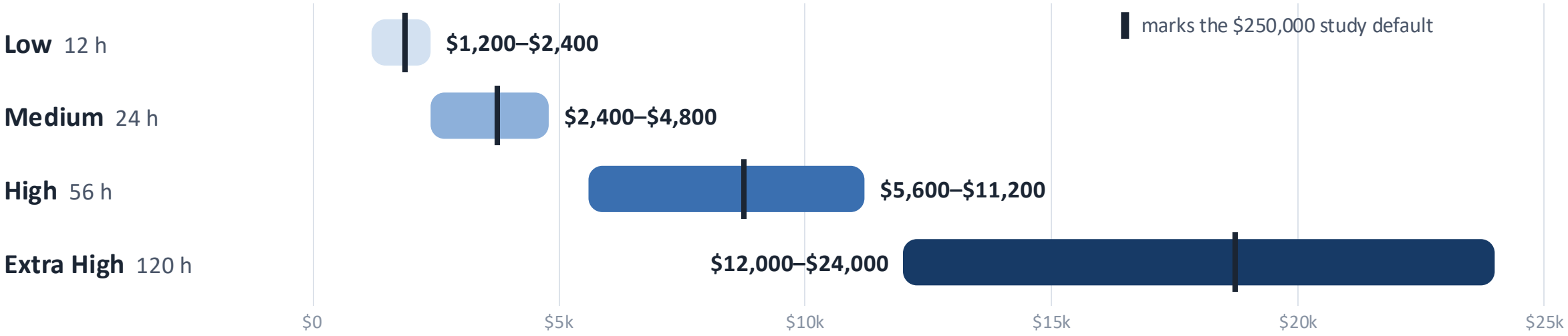
≈23% of issues accepted upstream

Review and remediation capacity still matter. Candidates, reports, upstream acceptance, and deployed fixes are different outcomes; not a race between labs.

Sources: red.anthropic.com/2026/cvd (no update since October 2) and trailofbits.com/patch-the-planet/dashboard (updated October 30); checked November 1, 2026 [S5]. Bars are scaled within each program.

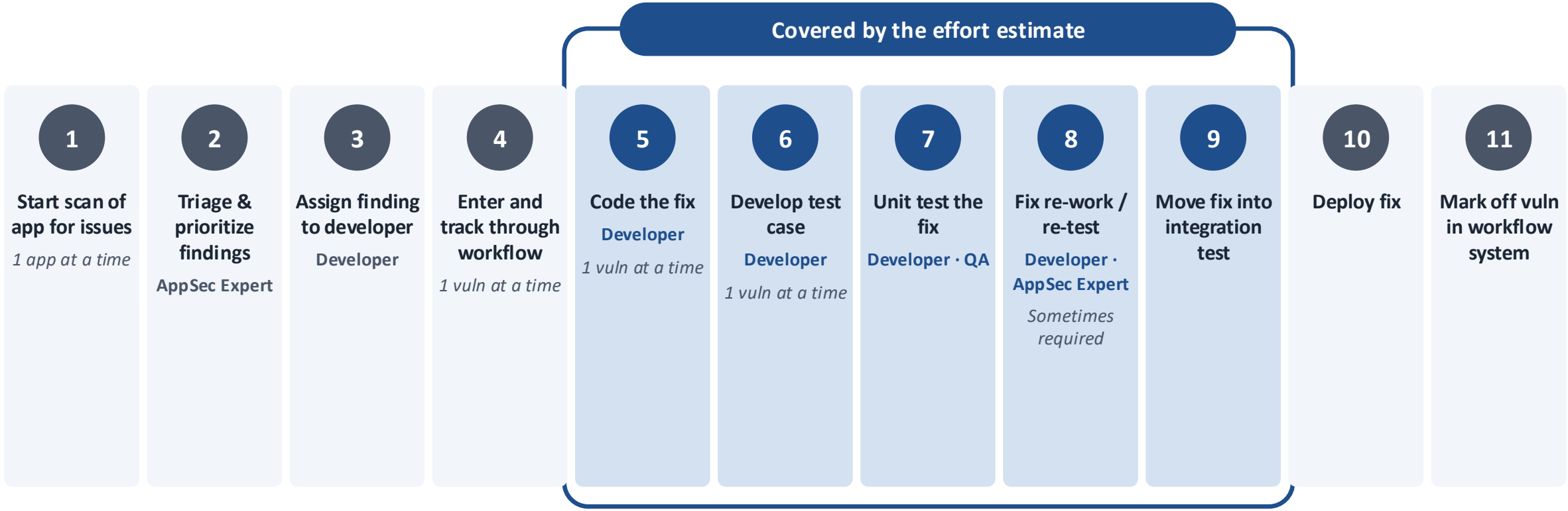
What the effort costs – median $2,500/fix

TYPICAL COST PER FIX BY EFFORT BAND: MEDIAN HOURS × $100–$200 PER HOUR



Modeled hours × rate ($156.25/h = $250k loaded FTE ÷ 1,600 h). Steps 5–9 only; 3,584 records without estimates are not costed. AppSecAI dataset, Oct 31, 2026 [S1].

Only Part of Process Covered by Cost and Hours



Steps 1–4: find, triage, assign, track. Not in the estimate.

Steps 5–9: implement, test, re-work, integrate. Modeled from the public patch.

Steps 10–11: deploy and close. Not in the estimate.

11+ manual steps to fix a vulnerability. Costs on the previous slide cover only the highlighted steps.

The estimate does not measure discovery, triage, full coordination, downstream deployment, or loss prevented [S1, S2].

Demonstration: analyze the data for your enterprise

Remediation Cost vs. Risk Profile

Published severity vs. the engineering cost to fix each vulnerability — priced records only. How the cost is calculated

FILTER ANALYSIS

Search by CVE, project, class, description...

All Projects | All Languages | All CWEs

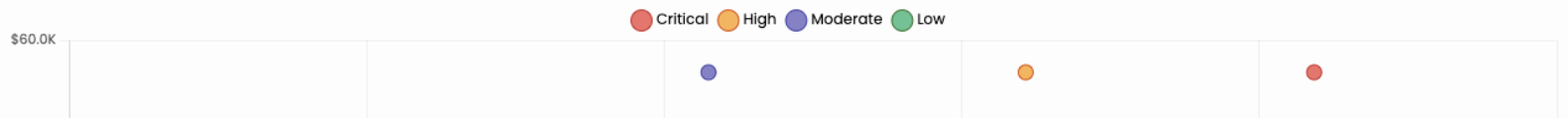
All Vuln Classes | All Severities | All Buckets | All Cost Bases | All Credits

2,853	$12.32M	$4,319	$ 250,000 /yr	$156.25/hr	28	+0.09
FILTERED RECORDS	FILTERED COST	AVG COST / CVE	ANNUAL FTE COST (EDITABLE)	HOURLY RATE	PROJECTS MATCH	NO CORRELATION (R)

Which commit is the fix, for the records shown: 33% of the engineer-days rest on fixes the project itself named (published). 52% rest on links we drew from a shared key, such as an issue or pull-request number (inferred-high), and 15% on a choice among a release's candidate commits made by a model, an analyst or a title match (inferred-medium or -low). Every record carries its label.

Vulnerability Cost vs. Risk Distribution

REMEDIATION COST BY SEVERITY (HOVER FOR DETAILS, CLICK A POINT TO EXPLORE RECORD DETAILS BELOW)



Limitations and future work

1

Only part of cost is captured

Not measured: discovery, triage, full coordination, downstream deployment, or loss prevented.

2

Look at the aggregates – individual records vary

Some fix matches are inferred; software classifications need review

3

Estimates need local calibration

Engineer-days are modeled. Compare with your team's actual work before using them for commitments.

4

Some fixes are shared – not independent.

Completed public fixes across different time windows; shared fixes mean records are not independent patches.

FUTURE WORK

Examine the corpus – provide feedback

About one-third of estimated records have direct project-published fix attribution [S1, S2].

Your next step: Try fixing at scale with OWASP OASIS

OASIS

Open Automated Security Initiative for Software
An OWASP project

AI matches the velocity of the threat. Humans ensure the quality.

15+ years in AppSec and Bug Bounty
Previously ran Intel's Bug Bounty program



Chris Holt
Intigriti

1

Pipeline

AI generates candidate fixes at threat velocity across open source repositories

2

Community

AppSec experts validate fixes and build trusted relationships with maintainers

3

Narrative

Shared urgency around AI-enabled threats unites security and developer communities

4

Governance

Community-validated quality standards that mirror open source culture

Next week: join Team OASIS and validate one candidate fix in a codebase you know. Check the issue, patch, tests, and regression risk, and record the review effort.

owasp-oasis.org

https://www.owasp-oasis.org/ · Verify project status and participation instructions before the conference [S7].

Join the Remediation Era – See the data

1

Look at the data

Data and methodology at the final public link

2

Book signing

Outside the room after the session

3

Try it on your code

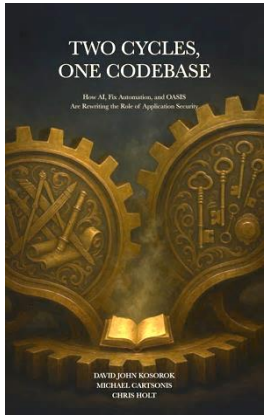
Contact Bruce to run the process on your internal repositories

www.appsecai.io/fixcost

Two Cycles, One Codebase

How AI, Fix Automation, and OASIS Are Rewriting the Role of Application Security

David Kosorok, Chris Holt, and Michael Cartsonis



Acknowledgments

Thank you to

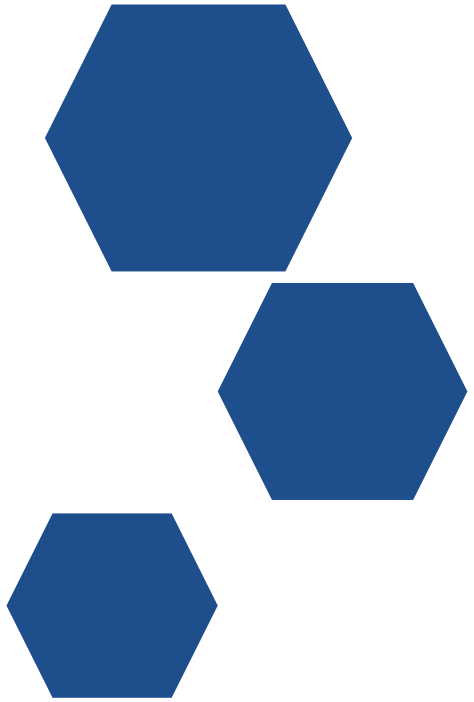
Eva Sarafianou and Mattermost

Vitor Meireles and GitLab

Keyfactor

Eric Bing · David Kosorok · Kevin Fealey

and the open source maintainers whose public fixes make this analysis possible.



Welcome to the



REMEDIATION ERA

ERA 1
Discovery
Toured

ERA 2
Scoring
Toured

ERA 3
AI Automation
Toured

ERA 4
Remediation
Now playing

Validate the fix. Ship the fix. Measure the work.