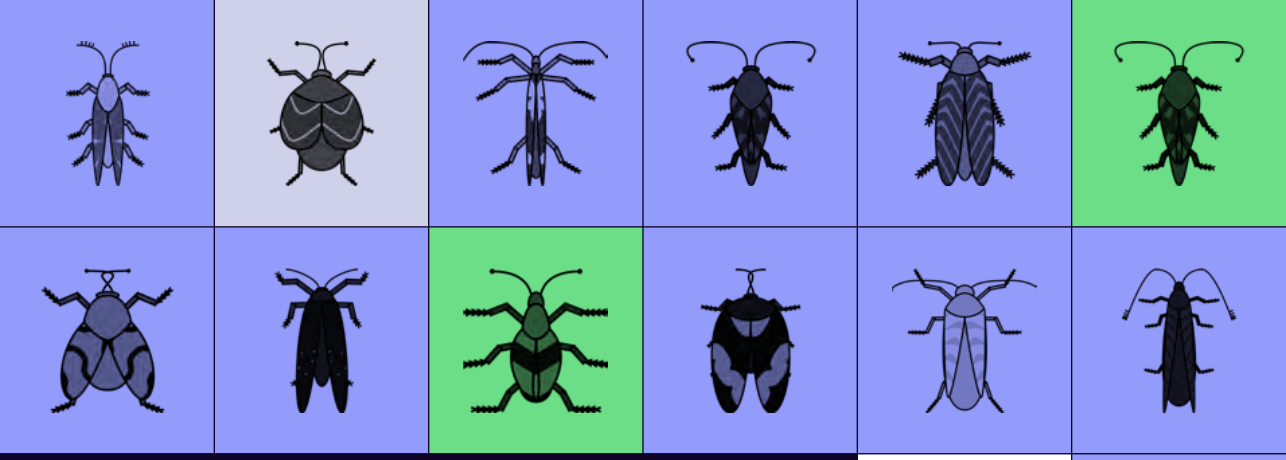


REPORT

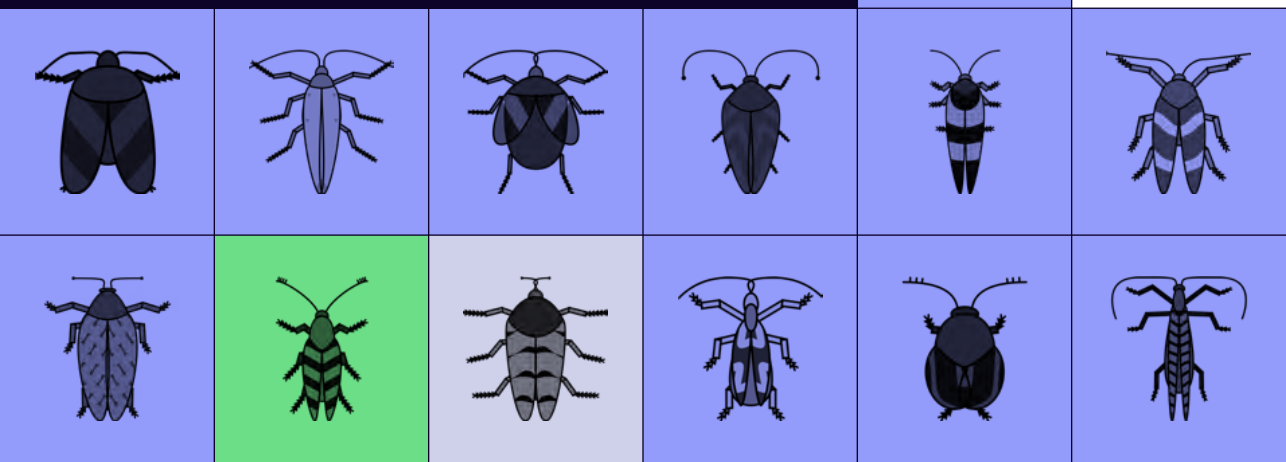
Bug Bounties und FOSS: Chancen, Risiken und der Weg nach vorn

Ryan Ellis
Associate Professor, Northeastern University

Jaikrishna Bollampalli
Kandidat für den Master Of Science in Software Engineering Systems, Northeastern University

HERAUSGEGEBEN VON

Sovereign Tech Resilience










Bug Bounties und FOSS: Chancen, Risiken und der Weg nach vorn

Ryan Ellis
Associate Professor, Northeastern University

Jaikrishna Bollampalli
Kandidat für den Master Of Science in Software Engineering Systems, Northeastern University


„Doch ich tue immer,
was ich kann,

um zu helfen.“

Log4j: Open Source-Sicherheit unter dem Mikroskop

Im Code von Log4j saß eine „tickende Zeitbombe“, die jahrelang niemand bemerkte.[1] Bis Ende 2021 wurde das beliebte Open Source-Logging-Tool gern und viel genutzt: Apple, Google, Microsoft, Amazon und unzählige andere hatten es in ihre Applikationen integriert.[2] Doch weder die Entwickler*innen, die die Software in ihre Builds eingebaut hatten, noch die Milliarden von Nutzer*innen, die nun die daraus resultierenden Applikationen mit Log4j nutzten, waren sich bewusst, dass sich schon Jahre zuvor – genauer gesagt 2013 – durch ein einfaches Update ein gravierender Fehler eingeschlichen hatte.[3] Die Schwachstelle saß dort fast acht Jahre lang, ohne dass sie jemand bemerkt hatte.[4]

Am 21. November 2021 meldete ein Forscher einer kleinen Gruppe von Freiwilligen, die Log4j betreuen, per E-Mail die schlechte Nachricht: Es gab einen erheblichen Bug in ihrem Code.[5] Wie viele Open Source-Projekte wurde Log4j trotz seiner Bedeutung und Allgegenwart in kommerziellen Produkten von einer relativ kleinen Gruppe von Freiwilligen entwickelt und betreut.[6] Sie stellten es dem Rest der Welt zur Verfügung, um es nach Belieben und mit wenigen Einschränkungen zu nutzen.[7] Die neu identifizierte Schwachstelle war gravierend: Sie würde es einem Angreifer ermöglichen, anfällige Systeme auszunutzen und beliebigen Code einzuschleusen.[8] Gary Gregory, ein Mitglied des Teams, das jahrelang an der Wartung von Log4j mitarbeitete, erinnerte sich an seine Reaktion auf den ersten Bericht: „Ich dachte nur, ‚Oh, Mist.‘ Einige von uns waren weniger davon überrascht, dass es ein Sicherheitsproblem gab, sondern eher davon, wie schlimm es war.“[9] Jen Easterly, Leiterin der US-amerikanischen Cybersecurity and Infrastructure Security Agency (CISA), bezeichnete es als „einen der gravierendsten Fehler“, den sie je entdeckt hatte.[10]

Als damit begonnen wurde, die Schwachstelle schnell zu beheben, folgten weitere schlechte Nachrichten: Die Schwachstelle wurde bereits in freier Wildbahn ausgenutzt.[11] Der bisher unbekannte Fehler war kein Geheimnis mehr: Milliarden Maschinen waren potenziell gefährdet.[12] Das Log4j-Team informierte möglichst schnell die Öffentlichkeit und veröffentlichte ein Update, das den Fehler beheben

1 Die Beschreibung der Log4j-Schwachstelle als „tickende Zeitbombe“ stammt aus Daniel Stenberg, zitiert in William Turton, Jack Gillum und Jordan Roberston, „Inside the Race to Fix a Potentially Disastrous Software Flaw“, *Bloomberg. 13. Dezember 2021.*

2 Google schätzt, dass Log4j bereits im Dezember 2021 in zehntausenden Softwarepaketen und Projekten enthalten war. Siehe James Wetter und Nicky Ringland, „Understanding the Impact of the Apache Log4j Vulnerability“, *Google Security Blog.* 17. Dezember 2021. Online abrufbar: https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html. Das U.S. Cyber Safety Review Board stellte fest, dass Log4j in „Millionen von Systemen“ integriert worden war. U.S. Cyber Safety Review Board, „Review of the December 2021 Log4J Event“. 11. Juli 2022, S. *ii.*

3 Eine detaillierte Timeline der Entstehung der Schwachstelle, der Entdeckung und der Reaktion sind zu finden in: Cyber Safety Review Board, „Review of the December 2021 Log4J Event“. Christian Grobmeier, Vice President der Apache Software Foundation, die die Entwicklung und Wartung von Log4j überwachte, bemerkte während der aktuellen Krise von Log4j, dass „im Grunde genommen die halbe Welt, vielleicht sogar mehr“ es nutzt. Zitiert in Turton, Gillum und Roberston, „Inside the Race to Fix a Potential Disastrous Software Flaw“.

4 Cyber Safety Review Board, „Review of December 2021 Log4J Event“.

5 Ebd., 1–2.

6 Log4j wurde von einer Gruppe von Freiwilligen betreut, die unter der Apache Software Foundation arbeiten. Siehe Turton, Gillum und Roberston, „Inside the Race to Fix a Potentially Disastrous Software Flaw“; und Patrick Howell O’Neill, „The Internet Runs on Free Open-Source Software. Who Pays to Fix It?“, *MIT Technology Review, 17. Dezember 2021.*

würde. Die Auswirkungen des Fehlers waren potenziell katastrophal. Joe Sullivan, Chief Security Officer bei Cloudflare, bemerkte, dass es „kaum ein Unternehmen gegeben hat, das nicht gefährdet war“.[13]

Nun kam der schwierige Teil. Die Entwicklung eines Patches unter Zeitdruck war in der Tat schwierig, aber die Bereitstellung der Lösung – die dann auch von den unzähligen Anwendungen, die auf Basis von Log4j gebaut worden waren, übernommen werden sollte – ging mit ganz eigenen Herausforderungen einher. Es war nicht klar zu erkennen, welche Anwendungen Log4j tatsächlich nutzten, es gab keine Masterliste oder eine einfache Möglichkeit, diese zu verfolgen.[14] Die meisten Nutzer*innen hatten keine Ahnung, ob es in ihren Anwendungen eingesetzt wurde.[15] Eine koordinierte Reaktion folgte schnell: Regierungen und Technologieunternehmen veröffentlichten eine Flut von Warnungen und Anweisungen, und Tausende von Sicherheitsexpert*innen arbeiteten daran, die potenziellen Auswirkungen zu mildern, indem sie über formelle und informelle Kanäle Tipps, Workarounds und Informationen austauschten.[16] Exploits und Angriffe folgten der öffentlichen Bekanntmachung der Schwachstelle, doch das sorgfältige und koordinierte Handeln von Freiwilligen, Technologieunternehmen und Regierungen schien Worst-Case-Szenarien zu entschärfen.[17]

Die Geschichte von Log4j ist so etwas wie eine Parabel für Open Source-Projekte und Open Source-Sicherheit. Log4j zeigt uns in vielerlei Hinsicht das Versprechen, das Open Source-Software bietet: Ein kleines, von Freiwilligen geführtes Projekt brachte ein nützliches Tool hervor, das in Milliarden von Geräten genutzt wurde und einen unermesslichen Mehrwert bei minimalen Entwicklungskosten bot. In Bezug auf die Reichweite ist es ein beeindruckender und unbestrittener Erfolg. Doch die Ereignisse von 2021 lassen diesen Erfolg in einem anderen Licht erscheinen. Viele Kommentator*innen – in der Branche, in der Regierung, in der Open Source-Community u. a. – nannten den Bug als Beispiel für die Mängel und Versäumnisse in der Open Source-Sicherheit: Ein kritischer Fehler in einem der Bausteine unserer gemeinsamen digitalen Infrastruktur blieb jahrelang unentdeckt und schuf ein systemisches und potenziell katastrophales Risiko. Die detaillierte Analyse des Vorfalls durch das U.S. Cyber Safety Review Board sah die Schuld zum Teil in der Anreizstruktur und Organisation von Open Source-Projekten. Sie fanden heraus:

7 Ebd.

8 Später bezeichnete das National Institute of Standards and Technology (NIST) den Fehler als „kritische Schwachstelle“ und gab ihm die höchstmögliche Punktzahl nach dem Common Vulnerability Scoring System (CVSS). Ebd., 1–2.

9 Zitiert aus Turton, Gillum und Roberston, „Inside the Race to Fix a Potentially Disastrous Software Flaw“.

10 Zitiert aus O’Neill, „The Internet Runs on Free Open-Source Software“.

11 Cyber Safety Review Board, „Review of the December 2021 Log4J Event“, 2–4.

12 Cyber Safety Review Board, „Review of the December 2021 Log4J Event“, 2; O’Neill, „The Internet Runs on Free Open-Source Software“.

13 Zitiert aus Frank Bajakd, „‘The Internet’s on Fire’ as Tech Races to Fix Software Flaw“, *Associated Press,* 10. Dezember 2021.

14 Joseph Marks, „One Month In, There Aren’t Any Huge, Known Log4j Hacks“, „The Cybersecurity 202“ [Newsletter], *Washington Post,* 11. Januar 2022.

15 Ebd.

16 Cyber Safety Review Board, „Review of the December 2021 Log4J Event“, 3–9, 35–37.

17 Cyber Safety Review Board, „Review of the December 2021 Log4J Event,“; Marks, „One Month In, There Aren’t Any Huge, Known Log4j Hacks“.

> [Der Log4j]-Vorfall machte auch auf Sicherheitsrisiken aufmerksam, die in der auf Freiwilligenarbeit und knappen Budgets basierten Open Source-Community einzigartig sind. Diese Community verfügt nicht über ausreichende Ressourcen, um sicherzustellen, dass Code nach branchenweit anerkannten sicheren Coding-Praktiken entwickelt und von Expert*innen geprüft wird.[18]

Das Board wies auf die scheinbare Diskrepanz zwischen der Entwicklung von Code auf freiwilliger Basis und soliden Sicherheitspraktiken hin. Sicherheitsarbeit, so argumentierten sie, könne oft ohne angemessene Anreize und Unterstützung auf der Strecke bleiben. Diese Analyse wurde von anderen bestätigt. Die U.S. Federal Trade Commission (FTC) argumentierte, dass diese Schwachstelle ein strukturelles, endemisches Problem für das Open Source-Ökosystem war:

> [Log4j] ist einer von Tausenden nicht angekündigten, aber äußerst wichtigen Open Source-Diensten, die in beinahe unzähligen Internetunternehmen zum Einsatz kommen. Diese Projekte werden oft von Freiwilligen erstellt und betreut, die nicht immer über ausreichende Ressourcen und Personal für Incident Response und proaktive Wartung verfügen, obwohl ihre Projekte für die Internetwirtschaft von entscheidender Bedeutung sind.[19]

Patrick Howell O'Neill fasste im *MIT Technology Review* das Missverhältnis bei den Anreizen kurz und knapp zusammen und stellte fest, dass Log4j zwar als Freiwilligenprojekt gegründet wurde und im Wesentlichen kostenlos läuft, doch „Millionen- und Milliarden-Dollar-Unternehmen darauf angewiesen sind und jeden Tag davon profitieren".[20] Bei Problemen würde es auf eine Gruppe von Freiwilligen zurückfallen, die in ihrer Freizeit arbeiten, um ein praktikables Patch zu entwickeln. Chris Wysopal, CTO bei Veracode, sagte

18 Cyber Safety Review Board, „Review of the December 2021 Log4J Event", v.

19 U.S. Federal Trade Commission (FTC), „FTC Warns companied to Remediate Log4j Security Vulnerability", *FTC Technology Blog*. 2. Januar 2022.

20 O'Neill, „The Internet Runs on Free Open-Source Software".

21 Zitiert aus O'Neill, „The Internet Runs on Free Open-Source Software".

unverblümt: Die fehlende Finanzierung von Open Source-Projekten sei nichts weniger als „ein systemisches Risiko für die USA, für kritische Infrastrukturen, für Banken, für den Finanzsektor“.[21]

Der Log4j-Bug unterstreicht für viele das fatale Problem bei Open Source-Projekten: Freiwillige Projekte mit fehlenden Hilfsmitteln können Sicherheitsaspekte nicht richtig berücksichtigen oder priorisieren. Einer der Hauptvorteile von Open Source-Technologien, wie die einfache Wiederverwendbarkeit von Paketen, wird durch eine potenziell schwache Sicherheit untergraben. Wenn Pakete umgenutzt werden, führt dies zu einer weitreichenden Verbreitung von unsicherem Code. Durch Fehlanreize kann dann die Ausbreitung einer wackeligen Komponente aufgegriffen und übernommen werden, was katastrophalen Schaden anrichten kann.

In den darauffolgenden Monaten gab es erneute Investitionsaufrufe und zur Priorisierung der Open Source-Sicherheit.[22] Das waren zwar keine neuen Themen, aber sie wurden endlich in den Fokus gerückt und erlangten neue Aufmerksamkeit.[23] Doch der Weg nach vorn ist ungewiss: Wie die Open Source-Sicherheit am besten unterstützt werden kann, bleibt ein drängendes Anliegen und eine offene Frage.

Bounties und Open Source-Sicherheit: der Weg nach vorn

Auf den folgenden Seiten werden die Vor- und Nachteile einer möglichen Intervention beleuchtet: die Einführung von Bug Bounties für Open Source-Projekte.[24] Während Bounties für bestimmte Open Source-Projekte selektiv eingesetzt wurden, fehlt eine umfassende Eignungsanalyse zum Einsatz von solchen Programmen. Dieser Bericht versucht, diese Lücke zu schließen. Er bündelt Forschung zu Bug Bounty-Programmen, Open Source-Communities und Open Source-Sicherheit und untersucht, wie Bounties die Sicherheit verbessern und gleichzeitig unzählige potenzielle Risiken für Sicherheitsforschende, Open Source-Projekte und die breite Öffentlichkeit vermeiden können.

Der Bericht enthüllt eine Reihe wichtiger Erkenntnisse. Er zeigt, dass Bounties die Sicherheit von Open Source-Software unter bestimmten Umständen sinnvoll verbessern können: Für ausgereifte Open Source-Projekte können Bounties eine zusätzliche Sicherheitsebene bieten.

22 Siehe z. B. Executive Office of the President, „Readout of White House Meeting on Software Security“, 13. Januar 2022. Online verfügbar: https://www.whitehouse.gov/briefing-room/statements-releases/2022/01/13/readout-of-white-house-meeting-on-software-security; Responding *to and Learning from the Log4Shell Vulnerability,* U.S. Senate Committee on Homeland Security and Government Affairs [Hearing], 8. Feburar 2022; Brian Behlendorf, „Log4Shell Retrospective“, Open Source Security Foundation (Blog). 15. Dezember 2022. Online verfügbar: https://openssf.org/blog/2022/12/15/avoiding-the-next-log4shell-learning-from-the-log4j-event-one-year-later; U.S. Office of the National Cyber Director, „Fact Sheet: Office of the National Cyber Director Requests Public comment on Open-Source Software Security and Memory Safe Programming Languages“, 10. August 2023. Online verfügbar: https://www.whitehouse.gov/oncd/briefing-room/2023/08/10/fact-sheet-office-of-the-national-cyber-director-requests-public-comment-on-open-source-software-security-and-memory-safe-programming-languages.

23 Corin Faife, „White House Hosts Tech Summit to Discuss Open-Source Security after Log4j“, 13. Januar 2022. Online verfügbar: https://www.theverge.com/2022/1/13/22881813/white-house-tech-summit-apple-google-meta-amazon-open-source-security. Siehe auch Trey Herr, Prepared Remarks, United States Senate Committee on Homeland Security and Government Affairs. 8. Februar 2022. Online verfügbar: https://www.hsgac.senate.gov/wp-content/uploads/imo/media/doc/Testimony-Herr-2022-02-08.pdf.

24 Schlüsselbeispiele für Versuche, Open Source-Projekte und Bug Bounties zu vereinen, sind das kollaborative Internet Bug Bounty-Programm, das Open Source Vulnerability Reward Program von Google und das Bug Resilience Project des Sovereign Tech Fund.

Es gibt klare Vorteile: Bounties können die Anzahl und Qualität von Fehlerberichten verbessern, sie können dazu beitragen, talentierte Forschende anzuziehen, sie können Projekten helfen, Expert*innen zu binden und das Abwandern der Community zu schmälern. Sie können Mechanismen zur Rechenschaftspflicht und Werkzeuge bereitstellen, die ansonsten fehlen. Dies sind potenziell wichtige und signifikante Vorteile. Aber es gibt auch Einschränkungen und Nachteile. Bounty-Programme können die Sicherheit in mehrfacher Hinsicht gefährden. Investitionen in sie könnten die Sicherheit untergraben, indem sie die Aufmerksamkeit auf unterbesetzte oder schlecht verwaltete Projekte lenken; die Gegenseitigkeit untergraben – einen Schlüsselwert, der Open Source-Communities belebt; neue finanzielle Belastungen schaffen, die nicht tragbar sind; und indem Anstrengungen und Ressourcen von den eigentlichen Ursachen der Schwachstelle weggelenkt werden. Diese Schlussfolgerungen weisen einen Weg nach vorn – einen Weg, um Open Source-Bounty-Programme zu entwerfen und zu implementieren, die Vorteile für Sicherheitsforschende, Open Source-Projekte und die breite Öffentlichkeit bieten. Dieser Weg ist nicht ohne Hindernisse und Fallstricke, doch mit der gebotenen Vorsicht gibt es Möglichkeiten für sinnvolle Verbesserungen.

Aufbau des Berichts

Der Bericht gliedert sich in vier Abschnitte. Der erste, **„Methoden: Open Source-Herausforderungen aufzeigen, Bounties dokumentieren“** [→p. 12], gibt einen Überblick über die Methodik des Berichts. Der Bericht nutzt gemischte Methoden und greift auf frühere Forschungsergebnisse zurück, um die Herausforderungen von Open Source-Projekten, eine Übersicht zu Bounty-Programmen und die potenziell bereichernde Vereinigung zwischen beiden zu beleuchten. Dieser Abschnitt gibt einen Überblick darüber, wie komplementäre neue qualitative und quantitative Daten erfasst, gesammelt und analysiert wurden. Abschnitt zwei, **Open Source-Sicherheit: strukturelle Herausforderungen** [→p. 17], denen Open Source-Projekte mit Blick auf Sicherheitsherausforderungen gegenüberstehen. In diesem Zusammenhang stellt der Bericht fest, dass die Fähigkeit, Fehler innerhalb eines Projekts zu erkennen und schnell zu beheben, eng mit der allgemeinen Fähigkeit zur Projektwartung verknüpft ist. Dieser Abschnitt beleuchtet die Vielfalt innerhalb des Open Source-Ökosystems – Projekte unterschiedlicher Größe, Form und Organisation prägen die Landschaft. In diesem Abschnitt wird auch die schwankende Performance in den einzelnen Projekten hervorgehoben – während Sicherheitsherausforderungen zu Recht viel Aufmerksamkeit geschenkt wird, gibt es bemerkenswerte Erfolge, die hervorzuheben sind. Der dritte Abschnitt des Berichts, **Bug Bounties und die laufende Neugestaltung der Sicherheitsaufgaben** [→p. 29], bietet einen ersten Einblick in Bounties, woher sie kommen und wie sie funktionieren. Gestützt auf einen früheren Bericht von Ryan Ellis und Yuan Stevens, *Bounty Everything: Hackers and the Making of a Global Bug Marketplace*, zeigen wir Ihnen in diesem Kapitel die Vor- und Nachteile auf,

die mit dem Aufstieg von Bounty-Plattformen verbunden sind.[25] Und zum Schluss, **Open Source Sicherheits-Bounties: der Weg nach vorn** (→p. 35), schließt mit einer detaillierten Betrachtung der Risiken und Vorteile, die mit der Bereitstellung von Bounties für Open Source-Projekte verbunden sind. Dieser Abschnitt zeigt einen Weg nach vorn auf, indem er darauf eingeht, wie und wann Bounties für ausgewählte Open Source-Projekte sinnvoll erweitert werden könnten. Der Bericht schließt mit einem Überblick über bewährte Verfahren und umsetzbare Empfehlungen. In **zwei Anhängen** (→p. 46) wird die quantitative Datenanalyse im Zusammenhang mit der Bewertung der Projektwartung und der Beziehung zwischen Wartung und Sicherheit vertieft.

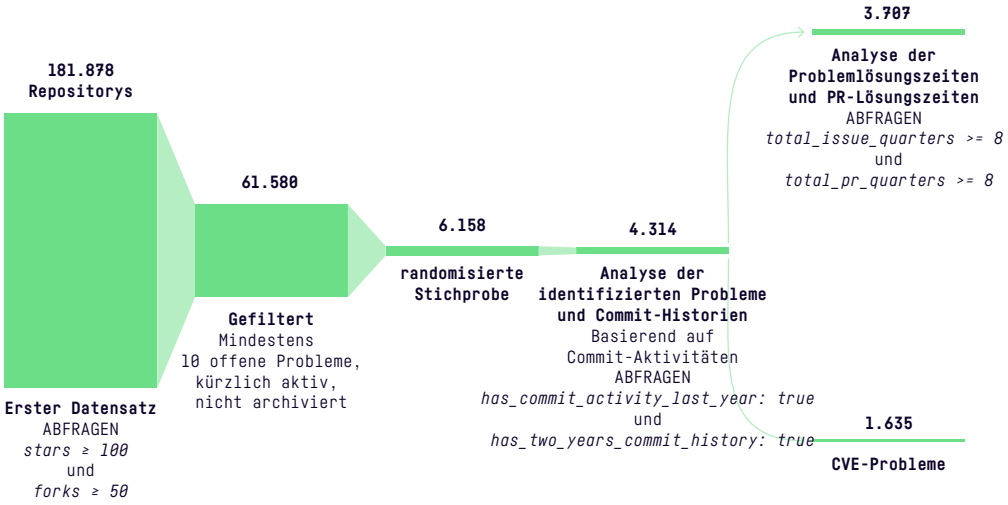
25 Ryan Ellis und Yuan Stevens, *Bounty Everything: Hackers and the Making of a Global Bug Marketplace*, Data & Society Research Institute. 2022. Online verfügbar: https://datasociety.net/library/bounty-everything-hackers-and-the-making-of-the-global-bug-marketplace.

Methoden: Open Source-Herausforderungen aufzeigen, Bounties dokumentieren

Dieser Bericht fasst Daten zum Open Source-Ökosystem und Bug Bounties zusammen. Er verknüpft neu erhobene qualitative und quantitative Daten mit vorhandenen Datensätzen, um diese zu erweitern und auf frühere Analysen aufzubauen. Die Forschung basiert im Wesentlichen auf zwei Datensätzen: quantitativen Open Source-Projekt- und Bounty-Daten sowie qualitativen Interviewdaten mit Schlüsselbeteiligten an Open Source-Projekten und Bounty-Programmen. Durch die Kombination von qualitativen und quantitativen Daten entsteht ein umfassender Überblick, der sowohl die expansive als auch die vielfältige Natur des Open Source-Ökosystems mit detaillierten, persönlichen Beobachtungen festhält. Diese Datensätze ergänzen sich: Sie verfügen sowohl über eine umfassende Bandbreite als auch über einen klaren Fokus, die für die Herausforderungen der Open Source-Sicherheit von Nutzen sein können. Gemessen an der Größe des Open Source-Ökosystems wurde anhand verschiedener Filter eine repräsentative und praxisnahe Stichprobe erstellt. Es folgt eine ausführliche Erörterung des methodischen Ansatzes. Zur Analyse und zu den Ergebnissen geht es in Abschnitt II weiter unten: Open Source-Sicherheit: strukturelle Herausforderungen.

GitHub-Daten: Analyse der Wartung und Sicherheit

Die Daten zu Open Source-Projekten wurden von GitHub durch gezielte Selektion und nachfolgende Filterrunden erhoben. Diese Daten wurden gesammelt, gespeichert und analysiert, um einen Einblick in die Dynamik von Open Source-Projekten zu erhalten und eine Analyse von verschiedenen Domänen im Zusammenhang mit der allgemeinen Wartung und Sicherheit zu ermöglichen (z. B. Untersuchung und Abfrage des breiten Leistungsspektrums im Zusammenhang mit der Identifizierung, Überprüfung und Abschließung von Problemen in einem Projekt). Bei der ersten Auswahl ging es darum, auf GitHub gehostete Projekte zu identifizieren, die zumindest minimal signifikant und aktiv waren, um die Herausforderungen in den Bereichen Wartung und Sicherheit (und ggf. die Beziehung zwischen den beiden) zu bewerten. Unter Verwendung der Abfragen *Stars ≥ 100* und *Forks ≥ 50* haben wir 181.878 erste populäre Repositorys identifiziert. Zur Eingrenzung des Datensatzes auf die relevantesten Repositorys haben wir zusätzliche Filter angewendet, um die aktuell aktiven zu erfassen. Die Kriterien umfassten Repositorys mit mindestens 10 offenen Problemen, einen kürzlich erfolgten Commit innerhalb des letzten Jahres und einen Status, der angibt, dass das Repository nicht archiviert wurde. Dieser Filterprozess reduzierte den Datensatz auf etwa 61.000 populäre und aktive Repositorys.



↑ Abb. 1 Prozess der gezielten Auswahl und anschließenden Filterrunden von GitHub-Repositorys

Aufgrund der großen Anzahl identifizierter Repositorys haben wir eine Stichprobe durchgeführt, um die Größe des Datensatzes unter Wahrung der statistischen Signifikanz zu verwalten. Aus einer 10 %igen randomisierten Stichprobe der gefilterten Repositorys ergab sich eine Teilmenge von 6.158 Repositorys. Dieser Ansatz ermöglichte eine überschaubare, aber repräsentative Stichprobe, die die Anwendbarkeit der gezogenen Schlussfolgerungen sicherstellte.

Als Nächstes sammelten wir detaillierte Daten zu den identifizierten Problemen und Commit-Historien der Repositorys. Diese Phase der Analyse konzentrierte sich auf das Sammeln von Daten und die Entwicklung von Metriken im Zusammenhang mit Wartungsaktivitäten, einschließlich Problemlösungszeiten, Commit-Häufigkeiten und Aktivität der Mitwirkenden. Damit sollte ein tieferes Verständnis der Entwicklungsdynamik der einzelnen Repositorys erlangt und Muster identifiziert werden, die entweder auf eine dauerhafte Wartung oder auf einen möglichen Rückgang hindeuten könnten. Die in dieser Phase gesammelten umfangreichen Daten bildeten die Grundlage für spätere Analysen der Wartungshistorie und -trends.

Bei der Betrachtung von Trenddaten haben wir darauf geachtet, dass partielle oder unvollständige Daten (z. B. Projekte mit nur einem oder zwei Monaten an Daten) die Analyse nicht verzerren. Dazu wendeten wir zusätzliche Kriterien an, um sicherzustellen, dass die ausgewählten Repositorys über einen längeren Zeitraum Entwicklungsaktivitäten und einen Verlauf von Problem- und Pull Request-Lösungen aufweisen. Zuerst haben wir nach Repositorys mit einer Commit-Aktivität innerhalb des letzten Jahres und einem Commit-Verlauf von mindestens zwei Jahren gefiltert. Als Nächstes haben wir den Datensatz gefiltert, indem wir uns auf Repositorys mit einem Verlauf der Problemlösung konzentrierten, insbesondere solche mit

mindestens acht Quartalen an Daten. Dieser Filterschritt war entscheidend für die Analyse langfristiger Wartungstrends. Um den Datensatz weiter zu verfeinern, wendeten wir einen Filter für Repositorys mit einem Verlauf der Pull Request-Lösung an, für die ebenfalls mindestens acht Quartale an Daten benötigt werden. Durch diese Schritte wurde der Datensatz auf 3.707 Repositorys reduziert, wodurch sichergestellt wurde, dass die ausgewählten Repositorys konsequent aktiv waren, gut gepflegt wurden und eine angemessene Aufzeichnung von Entwicklungs- und Lösungspraktiken aufwiesen.

Diese 3.707 Repositorys wurden analysiert und anhand von Metriken klassifiziert, um Trends und Verteilungen im Zusammenhang mit Wartungsaktivitäten zu identifizieren. Wartungsmetriken enthalten: Berechnung der mittleren Problemlösungszeit, der mittleren Pull Request-Lösungszeit und der relevanten Steigungen, die mit diesen Kennzahlen verknüpft sind (um eine Verbesserung oder einen Leistungsabfall im Laufe der Zeit anzuzeigen).

Um unser Verständnis der Sicherheitspraktiken zu verbessern, führten wir eine gezielte Analyse der Schwachstellen durch, die in Stichproben-Repositorys gemeldet wurden. Nachdem wir den Datensatz basierend auf den letzten Commit-Aktivitäten auf 4.314 Repositorys gefiltert hatten, konzentrierten wir uns auf das Sammeln detaillierter Problemdaten und das Extrahieren von Common Vulnerabilities and Exposures (CVE)-Informationen. Dieser Schritt beinhaltete die Verarbeitung von JSON-Dateien, die Ausgabe- und Pull-Anforderungsdaten für jedes Repository enthielten. Wir haben diese mit CVE-Daten verglichen, die aus dem offiziellen CVE GitHub-Repository heruntergeladen wurden, um Schwachstellen zu identifizieren, die mit den Repositorys in unserem Datensatz verbunden sind. Der Prozess begann mit dem Extrahieren von CVE-Nummern aus Ausgabetiteln unter Verwendung eines regulären Ausdrucks, gefolgt von der Erstellung von Dateipfaden, um die entsprechenden CVE-JSON-Dateien zu finden. Diese Dateien wurden dann analysiert, um wichtige Informationen wie Veröffentlichungs- und Aktualisierungsdaten abzurufen. Für jede Ausgabe, die mit einem CVE verknüpft ist, haben wir verschiedene Metriken berechnet, darunter die Verzögerung zwischen der CVE-Veröffentlichung und der Ausgabeerstellung sowie die Zeit, die für die Auflösung des CVE seit der Veröffentlichung und dem Aktualisierungsdatum benötigt wird. Diese detaillierte Analyse erlaubte es uns, die Reaktionsfähigkeit der Repositorys auf Sicherheitslücken zu quantifizieren, ein nützlicher Proxy für die Sicherheitsposition dieser Open Source-Projekte.

Nach der Extraktion von CVE-Daten identifizierten wir etwa 1.600 Repositorys mit zugehörigen CVE-Informationen. Für diese Repositorys haben wir eine Sekundäranalyse durchgeführt, um spezifische Metriken zum Umgang mit Schwachstellen zu erheben. Die Sekundäranalyse umfasste die Berechnung der Schlüsselstatistiken für jedes Repository, einschließlich des Medians, des Mittelwerts, des Höchstwerts und des Mindestwerts der CVE-Lösungszeiten ab Veröffentlichungsdatum sowie der Gesamtanzahl der CVEs pro Repository. Repositorys ohne CVEs wurden herausgefiltert, um eine gezielte Analyse auf diejenigen zu gewährleisten, die aktiv mit Schwachstellen umgehen.

HackerOne Data: das Internet Bug Bounty-Programm

Die Bug-Bounty-Plattform HackerOne hostet ein Programm namens Internet Bug Bounty (IBB) für ausgewählte Open Source-Projekte.[26] Mit einigen Einschränkungen konnten wir anhand der Daten des Internet Bug Bounty analysieren, wie Bounty-Programme in Open Source-Projekten eingebunden wurden. Wir haben alle verfügbaren öffentlichen Berichte des HackerOne IBB zusammengestellt. Dieser Datensatz enthält detaillierte Berichte über Schwachstellen und Sicherheitsprobleme, die von den Forschenden des Programms eingereicht wurden. Die Zusammenstellung dieser Berichte dient der Analyse von Auswirkungen und Trends von Sicherheitsbeiträgen in verschiedenen Open Source-Projekten im Rahmen des IBB. Vor allem diese Daten sind hilfreich, um aufzuzeigen, wie Forschende im Bereich Sicherheit mit verschiedenen Open Source-Repositorys interagieren, die in den Anwendungsbereich des Programms fallen.

Interviewdaten: Open Source-Maintainer*innen und Bug Bounty-Teilnehmende verstehen

Quantitative Daten wurden mit 62 Interviews gemischt, die mit zentralen Teilnehmenden von Open Source-Projekten und Bug Bounties geführt wurden. Die Stichprobe umfasste 22 speziell für dieses Projekt durchgeführte Interviews mit Fokus auf Open Source-Projekten und Open Source-Sicherheit sowie 42 zuvor durchgeführte Interviews mit Fokus auf Bug Bounties. Die Interviewdaten waren entscheidend: Sie beleuchteten und erschwerten in einigen Fällen die in den quantitativen Daten beobachteten Trends. Halbstrukturierte Gespräche mit Kennzahlen sowohl in Open Source-Projekten als auch in Bounty-Programmen boten wertvolle Erkenntnisse, die sonst nicht sichtbar gewesen wären.

Im Sommer 2024 führte das Projektteam Interviews mit 22 Maintainer*innen und zentralen Personen, die an Open Source-Projekten teilnehmen und sich mit Open Source-Security beschäftigen. Die Befragten wurden auf drei Arten identifiziert. Zunächst wurden Open Source-Projekte, die im oben beschriebenen GitHub-Datensatz enthalten sind, analysiert, um die besten fünf Prozent und die untersten fünf Prozent aller Projekte nach der mittleren Problemlösungszeit (d. h. wie lange ein Projekt zur Lösung eines eingereichten Problems benötigt hat) zu ermitteln. Diese Maßnahme wurde als grober, aber nützlicher Proxy für die Wartungskapazität oder -fähigkeit ausgewählt. Durch die Befragung von Personen an beiden Enden des Spektrums sollte sichergestellt werden, dass unterschiedliche Perspektiven, sowohl in Bezug auf Erfolg als auch auf Frustration oder Schwierigkeiten, in den Interviewdaten erfasst werden. Aus der GitHub-Datenstichprobe wurden Kontaktinformationen für wichtige Teilnehmende und Maintainer*innen extrahiert und Interviewanfragen an identifizierte potenzielle Probanden gesendet.

26 Derzeit umfasst das IBB-Programm 21 Projekte, darunter curl, Ruby, Rust, libssh, Django und weitere. Siehe HackerOne „Internet Bug Bounty“. Online verfügbar: https://hackerone.com/ibb?type=team.

Darüber hinaus wurde eine gezieltere Öffentlichkeitsarbeit durchgeführt, die sich auf Personen konzentrierte, die direkt an Open Source-Sicherheit arbeiten. Dabei wurden über den HackerOne-Datensatz Personen identifiziert, die mit den Repositorys des aktuellen Internet Bug Bounty-Programms verbunden sind. Die manuelle Identifizierung zusätzlicher möglicher Interviewpartner*innen erfolgte durch Überprüfung der GitHub-Daten, einschließlich der in den zugehörigen Sicherheitsrichtlinien und Sicherheitsteams angegebenen Kontakte und Mitarbeitenden. Die Schneeballstichprobe führte zur Identifikation weiterer potenzieller Probanden.

Die Interviews wurden offen und via Videokonferenz-Software geführt.[27] Professionelle Protokolle wurden erstellt und allen Befragten zur Verfügung gestellt. Alle Teilnehmenden durften ihr Transkript nach eigenem Ermessen überprüfen und bearbeiten und erhielten die Möglichkeit, ihren echten Namen oder ein Pseudonym für dieses Projekt zu verwenden. Für jedes Interview wurden Interview-Memos erstellt und Transkripte ausgewertet.

Diese Open Source-Sicherheitsinterviews wurden durch frühere Interviewrunden ergänzt, die Ryan Ellis und Yuan Stevens in Vorbereitung auf ihren früheren Bericht *Bounty Everything* durchgeführt hatten. Dazu gehörten 42 Interviews mit wichtigen Teilnehmenden an Bounty-Programmen, die zwischen 2019 und 2021 durchgeführt worden waren. Diese Interviewprotokolle wurden überprüft und anhand der besonderen Fragen, die in diesem Bericht aufgeworfen wurden, überarbeitet. Diese Interviews lieferten nützliche Informationen über die Funktion, den Nutzen und die Risiken von Bounty-Programmen im Allgemeinen. Die Bounty-Interviews wurden analog zu den Open Source-Interviews durchgeführt: Alle Interviews wurden transkribiert, von den Teilnehmenden überprüft und anschließend ausgewertet. Zwei Interviewpartner*innen, Katie Moussouris (CEO Luta Security) und Jack Cable (CISA), Cybersecurity-Fachleute mit großer Erfahrung sowohl in Bug Bounty-Programmen als auch in Open Source-Security, waren in beiden Runden dabei.

27 Zwei Interviews wurden auf Wunsch der Befragten nach einem anderen Protokoll durchgeführt. Diese Interviews wurden per E-Mail geführt und waren dadurch etwas formeller und strukturierter.

Open Source-Sicherheit: strukturelle Herausforderungen

Die Bedeutung von Open Source-Technologie liegt auf der Hand. Nahezu jede Anwendung, jedes Mobilgerät und jedes digitale Netzwerk, das wir nutzen, ist auf Open Source-Komponenten angewiesen.[28] Open Source-Technologien sind grundlegend: Sie sind die Bausteine für viele Technologien und Dienstleistungen – kommerzielle, gemeinnützige und staatliche –, denen wir in unserem täglichen Leben begegnen.

Open Source-Software steht Closed Source- oder proprietärer Software gegenüber.[29] Ihr zugrunde liegt die Vereinbarung, Quellcode zur Überprüfung, Änderung und Freigabe zur Verfügung zu stellen. Kritisch ist, dass unter einer typischen Open Source-Lizenz alle nachfolgenden Modifikationen unter den gleichen Bedingungen zur Verfügung gestellt werden müssen. Mit anderen Worten, zusätzliche Iterationen müssen ebenfalls zur Überprüfung, Änderung und Freigabe zur Verfügung gestellt werden. Open Source als Innovation ist eine clevere Umkehrung typischer Eigentumsregime und Vorstellungen. Während geistiges Eigentum in der Regel auf Exklusivität und Kontrolle aufgebaut ist, geht es bei Open Source explizit darum zu teilen – Arbeit zu verteilen und zur Verfügung zu stellen, damit andere sie nutzen und ausbauen können, wie sie es für richtig halten.[30]

Freiwilligenarbeit steht im Zentrum des Open Source-Ökosystems. Während einige Open Source-Teilnehmende durch ihren Arbeitgeber, Stipendien oder andere Mittel finanziert werden, ist Freiwilligenarbeit die Norm. Als solche arbeiten die Teilnehmenden an Projekten und Aufgaben, die sie interessieren.[31] Freiwillige fühlen sich aus vielen Gründen zu Open Source-Projekten hingezogen: Neugierde, die Möglichkeit, mit einer gleichgesinnten Gemeinschaft zusammenzuarbeiten, der Drang, neue Fähigkeiten zu erlernen, der Wunsch, andere zu befähigen, und vielleicht vor allem ein Do-it-yourself-Spirit, der sie inspiriert, etwas Neues in der Welt zu schaffen. Chris, ein langjähriger Mitarbeiter von MDN Web Docs (ehemals Mozilla Developer Network), konnte sich noch sehr gut an den Moment erinnern, als er erkannte, dass „all diese kleinen Softwareteile, die wir benutzen … von einer ganzen Reihe von Freiwilligen gemacht werden“.[32] Die Arbeit vieler, wie er es ausdrückte, habe uns „mehr gebracht, als wir erahnen können“.[33]

28 Die Bedeutung und Allgegenwart von Open Source-Technologien wird allgemein anerkannt, siehe Executive Office of the President, „Securing the Open-Source Software Ecosystem“. 4. Januar 2024. Eine aktuelle Studie kam zu dem Schluss, dass Open Source-Komponenten in 96 % der Codebasen zu finden sind. Manuel Hoffman, Frank Nagle und Yanuo Zhou, „The Value of Open Source Software“, Harvard Business School Working Paper Series, 2024.

29 Es gibt keine einheitliche Definition von „Open Source“, aber generell sind die GNU General Public License und ihre Anforderungen eine gute Basis. Thomas Haigh und Paul E. Ceruzzi, *A New History of Modern Computing* (Cambridge, MA: MIT Press, 2021). Wichtige Debatten zwischen Freier Software, Open Source und anderen konkurrierenden Geschichten, Visionen und Projekten existieren, würden aber den Rahmen dieses Berichts sprengen. Für tiefgehende und nützliche Diskussionen über Geschichte, Organisation und Bedeutung von Freier und Open Source Software siehe Steven Weber, *The Success of Open Source,* Cambridge, MA: Harvard UP, 2004); Christopher M. Kelty, *Two Bits The Cultural Significance of Free Software* (Durham, NC: Duke UP, 2008); E. Gabriella Coleman, *Coding Freedom: The Ethics and Aesthetics of Hacking* (Princeton, NJ: Princeton UP, 2013); und Christopher Tozzi, *For Fun and Profit: A history of the Free and Open Source Revolution* (Cambridge, MA: MIT Press, 2017).

30 Siehe Coleman, *Coding Freedom.*

31 Weber beschreibt dies als Idealtyp, oft genug wahr und angestrebt, aber mit bemerkenswerten Ausnahmen (wie weiter unten beschrieben). Weber, The *Success of Open Source,* 62.

32 Chris M., Interview, 2024.

33 Ebd.

Gegenseitigkeit ist ein zentraler Wert. Während Projekte frei wiederverwendet werden können und bei vielen, wenn auch nicht bei allen, die Entwicklung auf freiwilliger Basis erfolgt, werden gegenseitige Beiträge geschätzt und gesucht. Sarah, Mitglied des Sicherheitsteams von Django, erinnerte sich, dass sie sich anfänglich für Open Source-Projekte interessierte, unter anderem, um „einigen Paketen, die ich wirklich bewundere und auf die ich vertraue, ein wenig zurückzugeben“.[34] In Gesprächen mit Teilnehmenden und Maintainer*innen sprachen sie ganz klar über ihren Wunsch und ihre Hoffnung, dass die Nutzer*innen auch ihren Teil zu den Projekten, die sie verwenden, beitragen würden.

In der Praxis wird das Open Source-Ökosystem durch Projekte definiert, die in Zielsetzung, Größe und Organisation sehr unterschiedlich sind. Es beschreibt große, komplizierte Projekte, die jahrzehntelange Arbeit mit etablierten Verhaltenskodizes und Mitwirkendenrichtlinien beinhalten, eine definierte Gruppe von Kern-Maintainer*innen, spezialisierte Sicherheitsteams, Tausende von Mitwirkenden und Millionen von Nutzer*innen; und es umfasst kleine Projekte, die von einer Handvoll Einzelpersonen mit wenigen (wenn überhaupt) bedeutsamen Mitwirkenden erstellt und verwaltet werden. Nadia Eghbals Überblick über die Herausforderungen bei der Schaffung nachhaltiger Open Source-Projekte, *The Making and Maintenance of Open Source Software*, bietet eine grobe Typologie von Projekten. Sie identifiziert Föderationen, definiert durch eine hohe Anzahl von Mitwirkenden (meist in Untergruppen gegliedert) und einen großen Pool von Nutzer*innen; Clubs mit etwa der gleichen Anzahl an Mitwirkenden und Nutzer*innen; Spielplätze, Kleinprojekte mit einem bzw. einer Maintainer*in und wenigen Nutzer*innen; und schließlich Stadien, definiert durch eine kleine Anzahl von Mitwirkenden und viele passive Nutzer*innen.[35] Dennis, ein Haupt-Mitwirkender an Managram, einem Open Source-Betriebssystem auf Mikrokernelbasis, fasste die Breite des Ökosystems gut zusammen und reflektierte darüber, wie große und beliebte Projekte, wie Debian, sich effektiv zu einer „Mini-Regierung“ mit einem engagierten Verwaltungspersonal, Lenkungsausschüssen und allen entsprechenden Regeln und Prozessen entwickeln, während andere Projekte fast ausschließlich vom Engagement von zwei oder drei Maintainer*innen abhängen.[36] Diese unterschiedlichen Arten von Projekten spielen alle eine Rolle im Open Source-Ökosystem, obwohl Dennis bemerken musste, dass „[…] keiner mit dem Ziel beginnt, sich in einer Mini-Regierung zu engagieren. Wenn du in einer Regierung mitmachen willst … gehst du in die Politik.“[37]

Sicherung der Gemeingüter: kollaboratives Debugging in Open Source-Projekten

Sowohl für Open Source- als auch für proprietäre Projekte ist das Finden und Beheben von Fehlern eine ständige Herausforderung. Ungepatchte Bugs können Angreifer*innen neue Wege eröffnen und die Sicherheit untergraben.[38]

34 Sarah B., Interview, 2024.

35 Nadia Eghbal, *The Making of Maintenance of Open Source Software* (San Francisco; Stripe, 2020), 56–64.

36 Dennis B., Interview, 2024.

37 Ebd.

Vor Jahrzehnten schätzte Fred Brooks in seiner klassischen Abhandlung über Softwaredesign und -management, *The Mythical Man-Month*, dass rund 50 % der Zeit, die für die Entwicklung eines kommerziellen Softwareprogramms aufgewendet wird, für das Finden und Beheben von Bugs genutzt wird.[39] Doch trotz aller Bemühungen finden sie ihren Weg in Software-Anwendungen, die nicht ausreichend getestet wurden und es trotzdem auf den Markt geschafft haben. „In der täglichen Anwendung im realen Leben wird sich dann jeder Fehler zeigen", wie Brooks schmerzlich bemerkte.[40] Den Nutzer*innen entgehen keine Fehler, die Entwickler*innen irgendwie verpasst haben.

Jahre später hat Eric Raymond die Beobachtungen von Brooks im Kontext von Open Source-Software neu formuliert.[41] Für Raymond und andere überbrückt Open Source die Distanz zwischen Nutzer*innen und Entwickler*innen. Indem Projekte unterschiedlichen Nutzer*innen zugänglich gemacht werden, können neue Schwachstellen gefunden und behoben werden – der Einsatz, die Erfahrung und die Weisheit der Masse wird sich aktiv zunutze gemacht. Wie Raymond es ausdrückte: „Mit genug Augenpaaren werden alle Bugs sichtbar."[42] Ein großer Pool von Nutzer*innen und Entwickler*innen kann für Raymond zu einer effizienteren Erkennung und Behebung von Schwachstellen führen.[43] Für Brooks war das Auffinden und Melden von Bugs durch Nutzer*innen ein Beweis für die inhärente Unvollständigkeit und Unvollkommenheit der Softwareentwicklung; für Raymond war das kollaborative Debugging eine der Stärken der Open Source-Entwicklung.

In der Praxis bleibt das Finden und Beheben von Bugs in Open Source-Projekten jedoch ein hartnäckiger Reibungspunkt. Bugs bleiben oft monate- oder jahrelang nach ihrer ersten Meldung und Veröffentlichung unbearbeitet oder ungelöst. Eine Überprüfung der Problemlösungszeit und der CVE-Daten für unser GitHub-Sample ist hier aufschlussreich. Die Lösung der Mehrzahl der Probleme (51 %) mit einer assoziierten CVE-Zahl dauerte mehr als drei Monate, während über ein Drittel (34,2 %) mehr als sechs Monate zur Lösung benötigte **(siehe Abbildung 1)**. Nur eine begrenzte Anzahl von Problemen, nämlich 28,5 % wurde bereits innerhalb des ersten Monats nach der Veröffentlichung des CVE gelöst. Es liegt auf der Hand, dass, wie bei allgemeinen Problemen, die meisten Sicherheitsprobleme anhalten und nur langsam gelöst werden.

38 Bugs sind jedoch nicht immer ein Sicherheitsproblem. Viele Schwachstellen bereiten den Nutzer*innen lästige Schwierigkeiten, ohne die Sicherheit wirklich zu beeinträchtigen. Für die Zwecke dieses Berichts wird die allgemeinere Bezeichnung „Bug" austauschbar mit der spezifischeren und engeren Kategorie oder Teilmenge von Bugs verwendet, die Sicherheitslücken oder Schwachstellen beschreiben.

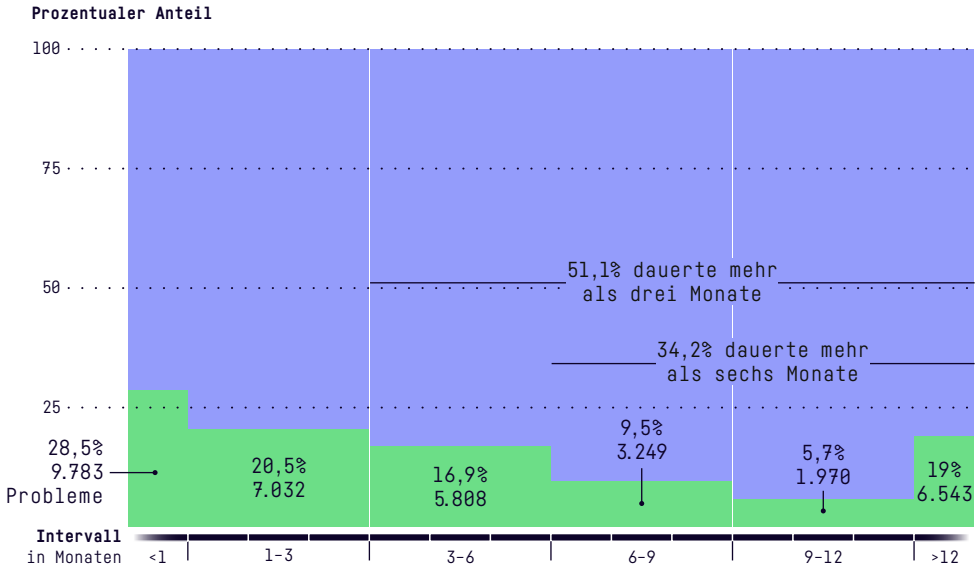
39 Frederick P. Brooks, Jr., *The Mythical Man-Month: Essays on Software Engineering*, Jubiläumsausgabe (Reading, MA: Addison-Wesley, 1995), 20.

40 Brooks, *The Mythical Man-Month*, 69.

41 Eric S. Raymond, *The Cathedral and the Bazaar* (2000). Online verfügbar: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html.

42 Raymond bezeichnete diese Beobachtungen als „Linus's Law" und nahm damit Bezug auf die Linux-Entwicklung. Raymond, *The Cathedral and the Bazaar*.

43 Ebd.



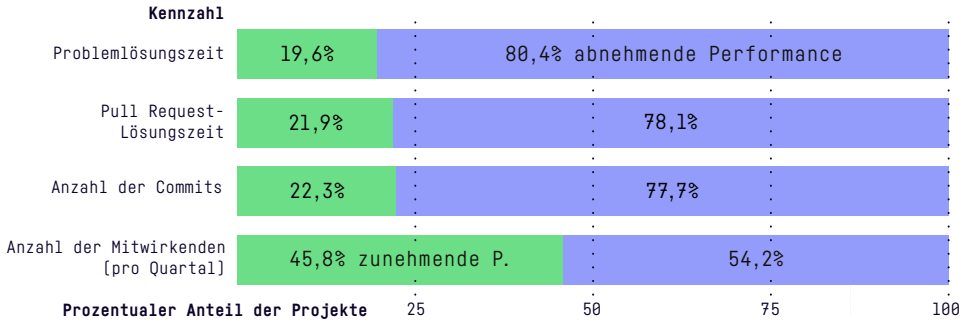
Sicherheit garantieren: Zeit, Vernachlässigung und das Problem der Popularität

Open Source-Sicherheitsherausforderungen folgen weitgehend den allgemeinen Herausforderungen bei der Pflege von Open Source-Projekten (wie unten ausführlich erläutert). Dieselben Schwierigkeiten, die die tägliche Wartung eines Open Source-Projekts erschweren, wirken sich auch auf die Sicherheit aus. Ein genauer Blick auf einen Querschnitt von Projekten zeigt: Ein gut gepflegtes Projekt ist oft auch ein sicheres Projekt. Ebenso haben Projekte, die mit der täglichen Wartung zu kämpfen haben, in der Regel Probleme, Fehler rechtzeitig zu erkennen und zu beheben.

Open Source-Projekte zu pflegen, ist eine klare und anhaltende Herausforderung. Eine Prüfung der zusammengestellten GitHub-Daten unterstreicht die signifikante Variation innerhalb von Open Source-Projekten – einige sind gut gepflegt, andere befinden sich in einem vergleichsweise vernachlässigten Zustand (für eine detaillierte Analyse der Daten siehe Anhang A: Messung der Open Source-Wartung). Während die durchschnittliche Zeit zur Lösung von Problemen bei Projekten in den Stichprobendaten etwas mehr als 20 Tage betrug, kann es bei einigen Projekten mehrere Jahre dauern, bis Probleme behoben sind. Auch die Daten zu den Lösungszeiten von

↑ Abb. 2 Lösungszeit für Probleme mit assoziiertem CVE

Intervall läuft vom ursprünglichen Veröffentlichungsdatum des CVE bis zum Datum, an dem das damit verbundene Problem durch ein Projekt behoben wurde. Die Daten stammen aus 1.824 einzigartigen Repositorys und umfassen 34.385 gelöste Probleme und 3.446 offene Probleme. Siehe Abschnitt I: Methoden: Open Source-Herausforderungen aufzeigen, Bounties dokumentieren für eine ausführliche Erörterung der Datenerhebung und der Stichprobe.



↗ **Abb. 3** Analyse der Performance von Open Source-Projekten im Zeitverlauf

Daten aus 3.707 Open Source-Repositorys. Projekte mit zunehmender oder abnehmender Performance werden durch die Analyse der Steigungen, die mit verschiedenen Kennzahlen verknüpft sind, ermittelt.

Pull Requests folgen einem ähnlichen Muster: signifikante Schwankungen mit extremen Unterschieden zwischen den Projekten. Eine Analyse eines breiten Satzes von Kennzahlen im Zusammenhang mit der Wartung, einschließlich der Häufigkeit der Commit-Aktivitäten, der Problemlösungszeit, der Zeit für die Lösung von Pull Requests und anderer, zeigt signifikante Unterschiede zwischen den Projekten.

Die Mehrzahl der Open Source-Projekte befindet sich im Niedergang und folgt einem ähnlichen Verlauf: einem Wartungs- und Aktivitätsrückgang im Laufe der Zeit. Eine Prüfung der gesammelten Projektdaten ergibt ein einheitliches Bild **[siehe Abbildung 3]**. Die meisten Open Source-Projekte brauchen zunehmend länger, um auf eingereichte Anfragen zu reagieren. Ebenso weisen die meisten von ihnen immer längere Zeitabstände zwischen dem Einreichen eines Pull Requests und seiner Lösung auf. Was die Aktivitäten anbelangt, so ist bei den meisten Projekten mit zunehmendem Bestehen eine Abnahme der Zahl der aktiven Mitwirkenden zu verzeichnen. Auch gibt es immer weniger neue Mitwirkende, die etwas beitragen möchten. Und es überrascht nicht, dass auch die Anzahl der Commits schrumpft.

Es ist kein Geheimnis, dass die Pflege von Open Source-Projekten häufig mit Schwierigkeiten einhergeht. Maintainer*innen und Mitwirkende weisen immer wieder auf die gleichen Probleme hin, die eine effektive Verwaltung erschweren: einen Mangel an Zeit und Ressourcen, zu wenige Teilnehmende und wertvolle Beiträge sowie eine überwältigende Flut von Beiträgen und Aufmerksamkeit. Diese Probleme verhindern, dass Maintainer*innen ihre Projekte auf dem neuesten Stand halten und auf ihre Nutzer*innen angemessen reagieren können.

Alltag und Leben: Wenn das Hobby immer als Letztes kommt

In Gesprächen mit Open Source-Maintainer*innen wird immer wieder deutlich: Es fehlt einfach die Zeit, um sich ihren Projekten widmen zu können. Für viele ist die Pflege eines Open Source-Repositorys eine Herzensangelegenheit – zumindest

zu Beginn. Doch die Belastungen des Alltags lassen oft nicht genug Raum für diese ehrenamtliche Arbeit. Für viele, lassen die Verpflichtungen eines Vollzeit-Jobs, Familie oder andere Interessen nicht viel Zeit für ihr Open Source-Projekt. Johann, Forschungswissenschaftler am Institut Pasteur, erlebte, wie das Leben durch seine ehrenamtlichen Projekte schnell zu voll wurde. Arbeit und Familie standen für ihn an erster Stelle: Da blieb wenig Zeit für sein Hobby und seine ehrenamtlichen Projekte. Mit einer gewissen Selbstironie erzählte er:

> Manchmal gehen Nachrichten unter. Ich vergesse zu antworten oder markiere sie und vergesse dann, sie auf meine To-do-Liste zu setzen. Am Ende vergesse ich sie einfach. Und es gibt Nachrichten, die einen Monat lang unbeantwortet bleiben. Oder ich baue keine Patches ein … ich verfolge keine Beiträge, weil ich sehr [wenig] Zeit [habe].[44]

Er sprach für viele Open Source-Teilnehmende, als er bemerkte, dass „dieses Hobby der Open Source-Projekte, sehr oft das Erste ist, was sie [aufgeben]“.[45] Mark, der Erfinder und Maintainer eines beliebten Gaming-Add-Ons, schloss sich Johanns Aussage an und stellte fest, dass das Projekt als Hobby fast immer „am Ende“ seiner Aufgabenliste landete.[46] Dennis, der bereits erwähnte Maintainer von Managram brachte es auf den Punkt: „Leider ist im richtigen Leben viel los.“[47]

Einige, wie Uday, der Entwickler und Maintainer des Pakets RNSwipeButton, kennen dieses Problem gut.[48] Vor vier Jahren startete er dieses Projekt, weil es etwas war, das er brauchte – anstatt darauf zu warten, dass jemand anderes den gewünschten Code schreiben würde, tat er es einfach selbst. Im Laufe der Zeit gewann das Projekt an Nutzen und wurde von anderen übernommen und in neuen Kontexten genutzt. Aber jetzt beginnt sein eigenes Interesse daran zu schwinden. Uday will sich auf etwas Neues konzentrieren, etwas anderes. Aber er fühlt sich verantwortlich gegenüber denen, die seine Arbeit nutzen und darauf aufgebaut haben. Sie zählen darauf, dass er das Projekt auch weiterhin pflegen und auf dem neuesten Stand halten wird. Er glaubt, dass er anderenfalls ihre Arbeit gefährden würde. Dennis fasste diese gemeinsame Entwicklung folgendermaßen zusammen: „[Es] fängt immer damit an, dass entweder ich etwas brauche, es nicht verfügbar ist und ich es dann selbst schreibe, oder ich denke: ‚Hey, das könnte Spaß machen und ich möchte dem Projekt helfen, weil ich es verwende‘.“[49] Aber mit der Zeit, so überlegte er auch, wandelte sich dieser spontane Einstieg in die Open Source-Technologie in ein Gefühl der Verantwortung um. Es ist nicht einfach: Maintainer*innen wie Uday sind hin- und hergerissen zwischen der Unterstützung des Projekts und der Community, die sie gefördert haben, und dem Wunsch, sich aus dem Projekt zurückzuziehen.

44 Johann, Interview, 2024.

45 Johann, Interview, 2024.

46 Mark W., Interview, 2024.

47 Dennis B., Interview, 2024.

48 Uday K., Interview, 2024.

49 Dennis B., Interview, 2024.

Der Alltag und die fehlende Zeit, sich wieder dem anspruchsvollen Hobby zu widmen, das gepflegt werden will, schafft Schwierigkeiten für Open Source-Projekte. Wenn die Wartung verständlicherweise zuletzt kommt, reagieren Projekte langsam auf Pull Requests und den Haufen an Problemen, die auf eine Lösung warten. Für die Sicherheit bedeutet dies: Neue Fehler werden nur langsam erkannt und behoben.

Die Gemeinschaft des Einen: das Problem der Vernachlässigung

Bei vielen Projekten werden diese Herausforderungen durch Vernachlässigung noch erschwert. In diesen Fällen ist die Gemeinschaft von Mitwirkenden nicht zustande gekommen. Vielmehr trägt innerhalb dessen, was Eghbal als Spielplatz- und Stadion-Projekte beschreibt, ein*e einzelne*r Maintainer*in oder eine kleine Gruppe von Maintainer*innen die volle Last, das Projekt mit wenig produktiver Unterstützung durch eine größere Community von Nutzer*innen zu betreiben.[50] Selbst bei Projekten, die eine große Nutzerbasis gefunden und entwickelt haben, fehlt es oft noch an nützlichen Beiträgen. Dennis, der Managram-Maintainer, bemerkte schnell, dass viele wichtige und beliebte Projekte tatsächlich von der Arbeit von ein oder zwei Maintainer*innen abhängen: „Es ist schon beängstigend, dass wir als Gemeinschaft … auf ein paar Leute angewiesen sind, um solche Kern-Infrastrukturen aufrechtzuerhalten.“[51] Nutzer*innen mögen das Projekt vielleicht und verwenden es, aber in den Augen der Maintainer*innen investieren sie selten auch ihre Zeit und Mühe, um das Projekt zu erweitern oder zu unterstützen.

Maintainer*innen beschreiben dies als einen Mangel an Gegenseitigkeit. Oftmals sind Nutzer*innen eifrig dabei, ein Open Source-Projekt zu verwenden, vielleicht sogar ein oder zwei Probleme zu melden, aber sie wollen sich nicht oder nur zögerlich dauerhaft einbringen. Jan, ein Mitwirkender von BigBlueButton, einem Open Source-Programm für virtuelle Klassenzimmer, bemerkte mit einiger Verärgerung:

> Eine Sache, die ich wirklich nicht mag und was ich in mehreren [Open Source]-Projekten erlebt habe, ist, dass Leute Feature Requests mit einem Anspruch stellen, wie zum Beispiel die Entwicklung von neuen Features, aber [sich] weigern, selbst daran zu arbeiten … Und sie denken, dass sie das Recht haben, nach Features zu fragen und diese dann auch entwickelt werden müssen, nur weil sie danach fragen.[52]

50 Eghbal, *The Making and Maintenance of Open Source Software.*

51 Dennis B., Interview, 2024.

52 Jan K., Interview, 2024.

Jan und andere glauben, dass es einfacher ist, sich mit einer Bitte um eine neue Funktion zu melden oder auf ein mögliches Problem hinzuweisen, als selbst die Ärmel hochzukrempeln und zu versuchen, ein Feature zu entwickeln oder zu beheben. Chris, der MDN-Maintainer, versucht immer, potenziell nützliche Mitwirkende ausfindig zu machen und die Community zu vergrößern, um herauszufinden, wer wirklich Interesse am Projekt hat, im Gegensatz zu jenen, die mal vorbeischauen und „eine schnelle PR einreichen und wieder verschwinden."[53]

Das Problem der Vernachlässigung belastet Open Source-Projekte. Das Fehlen aktiv beitragender Nutzer*innen und die mangelnde Gegenseitigkeit unter den Nutzer*innen erschweren eine gute Projekt-Performance: Die Beiträge konzentrieren sich auf eine*n einzige*n oder einen kleinen Kreis überlasteter Maintainer*innen.

Popularität und ihre Auswirkungen

Ironischerweise sind in manchen Fällen zu viele Mitwirkende das Einzige, was schlimmer ist als zu wenig Mitwirkende.[54] Nicht alle Beiträge sind nützlich oder willkommen, manchmal sind sie mehr Aufwand als Hilfe. Schlecht formulierte Pull Requests, ausschweifende oder unklare Probleme, belästigende Kommentare und auch einfach nur Spam überlasten die Maintainer*innen mit bereits knappen Ressourcen und lenken sie von der Kernprojektarbeit ab.

Mark, der Haupt-Maintainer eines kleinen Projekts, das ein Add-on für Rollenspiele bereitstellte, sprach vielen aus der Seele, als er bemerkte, dass die schlechte Qualität einiger Probleme und Pull Requests eine Belastung für das Projekt darstellte.[55] Diese Anfragen sind zeitaufwändig und nur teilweise projektbezogen. Uday unterstrich diesen Punkt und bemängelte, dass oft Leute ein Thema als Problem einreichen, das eigentlich ein Feature Request ist – eine Störung, die ihn von echten Problemen ablenke, die seine Aufmerksamkeit benötigten.[56]

Für Alex, ein langjähriger Mitarbeiter von Rust, ist die Zugänglichkeit von Open Source-Projekten Fluch und Segen zugleich. Für ein populäres Projekt wie Rust sei die Flut von Issues und Pull Requests ein echtes „Firehouse", eine endlose Auflistung von Dingen, um die es sich zu kümmern gilt.[57] Alex reflektierte über die Arbeit an einem beliebten Projekt und stellte fest, dass die Nutzer*innen ständig eingreifen wollen:

> Sie sagen dann: „Das funktioniert nicht richtig. Bitte repariert es. Ich brauche diesen Bug, ich brauche dieses Feature, bla, bla, bla." Das kann mit der Zeit sehr belastend werden, vor allem, wenn dies über eine Zeitspanne von • • •

53 Chris M., Interview, 2024.

54 Eghbal diskutiert ausführlich die Herausforderung der Beiträge und das Problem der benötigten Aufmerksamkeit. Daher sollten wir das Problem der Open Source-Wartung als eines von zu viel Engagement der Community betrachten, anstatt zu wenig. Siehe Eghbal, *the Making and Maintenance of Open Source Software.*

55 Mark W., Interview, 2024.

56 Uday K., Interview, 2024.

57 Alex C., Interview, 2024.

> einem Jahrzehnt geschieht ...Einige Leute sind auch einfach super unhöflich. Sie fragen: „Warum hast du es denn nicht *so* gemacht? Das ist lächerlich. Alle hätten es so gemacht.“ Das kann mit der Zeit sehr niederschmetternd sein ...[58]

Für Alex sind solche fordernden und unhöflichen Äußerungen „einer der großen Nachteile von Open Source“.[59] In einer offenen Community, in der alle Beiträge willkommen sind, sind einige Beiträge bestenfalls kontraproduktiv, aber schlimmstenfalls unangebracht und beleidigend. Für ihn war klar: Solche Interaktionen sind einer der Hauptgründe, warum er „irgendwann bei Rust ausgebrannt“ sei.[60]

Chris, der oben genannte Maintainer, der an MDN Web Docs arbeitet, stimmte mit vielen anderen darin überein, dass das hohe Volumen an eingehenden Anfragen zeitaufwändig und ermüdend war. Er meinte auch, „dann gibt es eine Menge Leute ..., die helfen wollen, aber keine Ahnung haben, wie sie helfen können“.[61] Andererseits möchte er aber auch nicht die breite Beteiligung oder die Beiträge ganz ausschließen. Er möchte, dass das Projekt offen und zugänglich bleibt: Man weiß nie, wann neue Mitwirkende tatsächlich zu produktiven und langfristigen Projektmitgliedern heranwachsen könnten. Alex ist sich trotz seiner Bedenken in diesem Punkt sicher: Open Source offen zu halten, einfachen Zugang zu haben, trotz der Nachteile, ist wichtig; nur so können auch neue, frische Talente in das Projekt kommen. Diese Flut von Neueinsteiger*innen, die ihren Beitrag leisten wollen, ist wichtig und entspricht dem, was viele Teilnehmende als Hauptvorteil von Open Source-Projekten sehen: ein Ökosystem, das für alle offen ist und das jeden befähigen kann.

Es müssen Wege gefunden werden, um die Aufmerksamkeit zu steuern – um Aufgaben zu begrenzen, die die Energie aufsaugen und Maintainer*innen von positiven Beiträgen abhalten. Bei vielen Projekten wurden Schritte unternommen, um die Qualität und nicht nur die Quantität der Beiträge zu verbessern. Leitlinien, ein gewisses Maß an formeller oder informeller Überprüfung und andere Praktiken können helfen, aber diese Schritte beinhalten immer ein gewisses Maß an Kompromissen: Barrieren können immer auch neue oder wiederkehrende Mitwirkende ausgrenzen. Doch für viele Projekte ist es das Risiko wert.

Ein gut gepflegtes Projekt ist ein sicheres Projekt

Herausforderungen bei der Wartung können schnell überlasten und die Sicherheit untergraben. Die quantitative Analyse der GitHub-Daten ist aufschlussreich: Projekte, die sich mit wesentlichen Wartungsaktivitäten auskennen, darunter vor allem effizientes Management von Pull Requests und ein

58 Alex C., Interview, 2024.

59 Alex C., Interview, 2024.

60 Alex C., Interview, 2024.

61 Chris M., Interview, 2024.

hoher Grad an kontinuierlicher Commit-Aktivität (Engagement für das Projekt), sind auch in der Lage, Sicherheitsprobleme im Repository zu identifizieren und zu beheben (für eine detaillierte Übersicht über die quantitative Analyse siehe Anhang A: „Analyse der Wartung und Sicherheit“). Diese Praktiken sind fest verankert: Wartung und Sicherheit gehen Hand in Hand. Die Aufgaben der Projektwartung – Beantwortung von Problemen, rechtzeitige Genehmigung von Pull Requests, Suche und Pflege von aktiven und produktiven Beiträgen – schaffen auch ein sicheres Projekt. Ohne richtige Wartung leidet die Sicherheit unweigerlich. Ohne ausreichende Zeit für die Arbeit an einem Projekt, bei Vernachlässigung der Nutzer*innen oder einer Fülle von kontraproduktiver Aufmerksamkeit warten gemeldete Probleme und Pull Requests unendlich lang auf eine Antwort. Das Projekt ist unzureichend gewartet, zwar noch am Leben, aber bereits angezählt. Auch hier bleiben Fehler unidentifiziert oder ungelöst.

„Es nervt …“: die einzigartigen Herausforderungen der Open Source-Sicherheit

Die allgemeinen Herausforderungen bei der Aufrechterhaltung eines Open Source-Projekts werden durch die zusätzlichen einzigartigen Herausforderungen der Sicherheitsarbeit verschärft. Freiwilligenarbeit ermöglicht es Interessierten, die Aspekte eines Projekts, auf die sie sich konzentrieren möchten, selbst auszuwählen.. Sicherheit ist für viele einfach nicht attraktiv. Aaron, ein Haupt-Mitwirkender, der sowohl für Rack als auch Ruby on Rails Sicherheitsarbeit leistet, erklärte unverblümt, warum die Mitwirkenden Sicherheitsarbeit meiden: „Weil es einfach langweilig ist! Es macht keinen Spaß.“[62] Er erzählte, wie auch er erst nach und nach Sicherheitsaufgaben übernahm:

> Ich mochte [Sicherheitsaufgaben] nicht wirklich. Ich war nicht besonders interessiert daran oder hatte Lust darauf. Aber ich weiß nicht … Ich dachte mir: „Ja, irgendjemand muss es ja tun und ich kann es tun.“ Und so übernahm ich erste Aufgaben. Und so kam es dazu, dass ich in Sicherheitsteams involviert war.

Der Antrieb der Freiwilligen ist das, was sie interessant und erfüllend finden. Sicherheit ist das oft nicht. Bei der Arbeit an Sicherheitsthemen fallen häufig einige der zentralen individuellen Vorteile weg, die Teilnehmende an der Arbeit an Open Source-Projekten schätzen: Zusammenarbeit und Feedback. Aaron ergänzte noch:

> [Sicherheit] ist keine besonders spannende Aufgabe. Ich versuche, zu beschreiben, warum.
> • • •

62 Aaron P., Interview, 2024.

> Zuerst müssen Sie jeden Sicherheitsbericht, der Sie erreicht, ernst nehmen und gründlich prüfen. Und leider müssen Sie dies im Grunde im Geheimen tun, denn wenn es eine echte Sicherheitslücke gibt, wollen Sie sicherstellen, dass sie behoben wird oder Sie eine Lösung dafür haben, bevor die Allgemeinheit davon Wind bekommt. Und das macht keinen Spaß, weil es bedeutet, dass man nicht wirklich Feedback von der Community oder anderen Leuten bekommen kann. Man arbeitet also ziemlich isoliert an seinen Aufgaben.

Die Art von Sicherheitsproblemen erfordert ein gewisses Maß an Isolation – diese Probleme können nicht immer offen gelöst werden. In einigen Fällen filtern Projekte diese Probleme in ein anderes Triage- und Tracking-System, und nur bestimmte autorisierte Mitwirkende sind in der Lage, an Sicherheitsproblemen zu arbeiten. Auf diese Weise wird die Zusammenarbeit erschwert oder stark eingeschränkt.

Zudem ist die Sicherheitsarbeit nicht nur hoch belastet, sondern auch mit möglichen zeitraubenden Umwegen gefüllt. Aaron fuhr fort:

> Zweitens … Sie müssen diese [Probleme] ernst nehmen, aber gleichzeitig bekommen Sie auch eine Menge Junk-Berichte. Sie versuchen also …, die Spreu vom Weizen zu trennen. Und oft bekommt man diese Mistberichte, das ist einfach nur nervig.

Der Antrieb der Freiwilligen ist das, was sie interessant und erfüllend finden. Angesichts dieser Worte ist es nicht verwunderlich, warum viele auf Sicherheitsarbeit verzichten.

Einige Projekte können diese Probleme lösen, indem sie bezahlte Mitarbeitende bei der Übernahme der wichtigen, aber ungeliebten Aufgabe des Sicherheitsmanagements unterstützen und Sicherheitsteams entwickeln, die eine Zusammenarbeit ermöglichen. Django sticht als gutes Beispiel heraus. Während der Großteil ihres Sicherheitsteams ehrenamtlich tätig ist, hilft eine kleine Anzahl von Auftragnehmenden bei der Bewältigung eingehender Sicherheitsprobleme und bei der Entwicklung von Korrekturen. So wird sichergestellt, dass Sicherheit eine aktive Priorität bleibt und die Freiwilligen sich auf Themen konzentrieren können, die sie spannend und interessant finden (und die Dinge, mit denen sie sich lieber nicht beschäftigen würden, abgeben können). Shai, Mitglied des Sicherheitsteams bei Django, beschreibt seine Rolle folgendermaßen:

> Ich versuche, dabei zu helfen, alle Probleme zu lösen, die ich kann, wenn ich Zeit und auch das Interesse dafür habe. Doch bei einigen der Probleme, die auftauchen, habe nichts beizutragen … Doch ich tue immer, was ich kann, um zu helfen.

Shai merkte an, dass viele Sicherheitsprobleme, die dem Projekt gemeldet werden, in der Tat überhaupt keine Sicherheitsprobleme sind: Sie sind Missverständnisse oder gar keine Problemmeldungen. Sich durch den Haufen eingehender Meldungen zu wühlen, ist zeitaufwändig und mühsam. Für Shai sind Auftragnehmende absolut essenziell: Sie haben die Zeit und den Anreiz, diese Berichte zu bearbeiten und das Ernsthafte vom Trivialen zu trennen. Shai ist sich über die Bedeutung der Auftragnehmenden im Klaren: „Wenn ich weiß, dass es Leute gibt, die dafür bezahlt werden, vertraue ich darauf, dass sie geduldiger sind als ich."[63]

Die Struktur des Sicherheitsteams von Django sorgt dafür, dass diese oft ungeliebte Arbeit der Open Source-Projekte für Freiwillige wie Shai interessant und spannend bleibt. Die Arbeit mit dem Sicherheitsteam – einer Gruppe von etwa 10 Personen – ermöglicht die Zusammenarbeit. Sicherheitsberichte werden vom regulären Issue Tracking-System isoliert und dieser kleinen Arbeitsgruppe zugestellt, die in der Lage ist, die Einreichungen privat zu diskutieren und zu prüfen. Und noch besser: Mithilfe der unschätzbaren bezahlten Auftragnehmenden können Freiwillige den Aspekt der Arbeit, die sie nicht genießen, abgeben und sich auf das konzentrieren, was sie spannend finden. Die Arbeit isoliert Shai nicht, sie bleibt erfüllend.[64]

Aber für Projekte mit knappen Ressourcen ist die Entwicklung eines Sicherheitsteams mit bezahlter Unterstützung möglicherweise nicht möglich. Bei einem rein freiwilligen Modell führt die Selbstauswahl dazu, dass einige Teilnehmenden die Last der Sicherheitsarbeit nicht auf sich nehmen.

63 Shai B., Interview, 2024.

64 Shai B., Interview, 2024.

Bug Bounties und die laufende Neugestaltung der Sicherheitsaufgaben

In gewisser Weise hat sich seit Fred Brooks Erkenntnissen vor Jahrzehnten wenig geändert: Software-Tests sind teuer, zeitaufwändig und unvollständig. Das Finden und Beheben von Bugs bleibt eine ständige Herausforderung. In den letzten zwei Jahrzehnten wurden Bug Bounty-Programme – Programme, die unabhängige Sicherheitsforschende, also „Hacker*innen" bezahlen, um neue Fehler zu finden und zu melden – eingeführt, um diesen Prozess zu verbessern. Bounty-Programme zielen darauf ab, die Masse zu nutzen, Hacker*innen für ihre Arbeit zu belohnen und gleichzeitig die Sicherheit zu einem vernünftigen Preis für Anbieter zu verbessern.

Da diese Programme zunehmend von Unternehmen und Regierungen übernommen werden, experimentieren auch Open Source-Projekte mit Bounties. Auf den ersten Blick scheinen sie eine ideale Lösung für die Verbesserung der Open Source-Sicherheit und die Bewältigung der oben skizzierten Herausforderungen zu bieten: Sie versprechen eine bessere Auseinandersetzung mit ansonsten vernachlässigten Open Source-Projekten und sie bieten eine Möglichkeit, Anfragen für beliebte Projekte besser zu verwalten. Allerdings sind Bounty-Programme nicht ohne Nachteile: In einigen Fällen können sie echte Risiken für Hacker*innen, Organisationen und die breite Öffentlichkeit schaffen.

Bounty Everything: der Aufstieg von Bug Bounty-Programmen

Bevor wir uns dem Nutzen und den Risiken der Integration von Bounties in Open Source-Repositorys zuwenden, folgt ein kurzer Überblick über Bug Bounty-Programme – deren Geschichte, Organisation und die damit verbundenen Vorteile und Gefahren. Diese Übersicht und Analyse verdanken wir in hohem Maße dem früheren Bericht von Ryan Ellis und Yuan Stevens, *Bounty Everything: Hackers and the Making of the Global Bug Marketplace.*[65]

Ein florierender Markt für Fehler: das Bounty-Ökosystem im Überblick

Bug Bounties sind ein florierendes Geschäft. Hunderte von Organisationen bieten jetzt Belohnungen für neuartige Bugs an. Jedes Jahr nehmen Tausende von Hacker*innen an diesen Programmen teil und verbringen unzählige Stunden damit, nach bisher unbekannten und unentdeckten Schwachstellen zu

65 Siehe Ellis und Stevens, *Bounty Everything für eine detaillierte Übersicht.*

suchen. Unternehmen zahlen Hacker*innen Millionen, oder in einigen Fällen, Dutzende von Millionen Dollar jedes Jahr über Bounty-Programme.

Bounty-Programme sind keine neue Erfindung. Netscape startete das erste weithin anerkannte Programm vor fast 25 Jahren.[66] Nach einer schlechten Presse in Bezug auf die Sicherheit ihres beliebten Webbrowsers entwickelte Netscape einen neuen Ansatz: Hacker*innen werden bezahlt.[67] Dieses erste Bounty-Programm war so etwas wie ein Sprung ins kalte Wasser. Es war ein Versuch, Sicherheitsforschende dazu zu bewegen, keine neuen Schwachstellen öffentlich zu melden, wo sie Schlagzeilen machten und das Vertrauen in Netscape untergruben.[68] Mit der Zeit setzte sich das Modell durch. Mozilla, der Nachfolger von Netscape, und Tech-Unternehmen wie Google, Facebook und Microsoft übernahmen dann alle irgendwann ein Bounty-Modell.[69] Das Aufkommen von Bounty-Plattformen – BugCrowd 2011 und HackerOne 2012 – entwickelte dann den Markt rasant weiter.[70] Plattformen rekrutierten neue Unternehmen und Organisationen und ermutigten sie, Bounty-Programme anzubieten. Gleichzeitig vermarkteten sie diese Programme an Hacker*innen als eine Möglichkeit für zusätzliches Einkommen oder als Vollzeit-Karriere.[71]

Genaue Angaben über Größe und Umfang des aktuellen Marktes sind schwer zu erhalten, aber die verfügbaren Zahlen sind dennoch erstaunlich. HackerOne, eine der größten Plattformen, hostet Dutzende von Bounty-Programmen für verschiedene Unternehmen und Organisationen und verarbeitet jeden Monat Tausende von neuen Berichten.[72] Bis heute haben sich über 2 Millionen Hacker*innen für die Teilnahme an auf der Plattform gehosteten Bug Bounty-Programmen registriert.[73] Seit HackerOne vor über einem Jahrzehnt ins Leben gerufen wurde, hat es über 300 Millionen US-Dollar an sogenannten Bounties (eine Art Kopfgeld) in mehr als 400.000 gültigen Einsendungen gezahlt.[74] Nicht nur Bounty-Plattformen sind erfolgreich, Google, Facebook und andere bieten ihre eigenen, eigenständigen oder Inhouse-Bounty-Programme an. Seit der Lancierung des Programms im Jahr 2011 hat Facebook (heute Meta) Hacker*innen Bounties in Höhe von über 16 Millionen US-Dollar gezahlt.[75] Google hat diese Zahl über die Laufzeit des Programms mehr als verdreifacht und zahlte über 58 Millionen US-Dollar an über 3.000 verschiedene Hacker*innen aus.[76]

Basierend auf den Plattformen und den erfolgreichen Programmen ist Bounty-Arbeit für einige zu einem Vollzeitjob geworden. BugCrowd berichtet, dass über 60 % der Hacker*innen, die sich an ihrer Plattform beteiligen, Bounties als Vollzeitbeschäftigung betrachten.[77]

66 Ellis und Stevens, *Bounty Everything*, 28–33.

67 Ebd.

68 Ebd.

69 Ellis und Stevens, *Bounty Everything*, 38–43.

70 Ebd.

71 Siehe Ellis und Stevens, *Bounty Everything*.

72 HackerOne, „Why HackerOne?“ Online verfügbar: https://www.hackerone.com/why-hackerone.

73 Unklar ist jedoch, wie viele dieser Accounts aktiv sind. Ebd.

74 HackerOne, „Why HackerOne?“; HackerOne, „Hackers Surpass $300 Million in All-time Earnings on the HackerOne Platform“, 26. Oktober 2023. Online verfügbar: https://www.hackerone.com/press-release/hackers-surpass-300-million-all-time-earnings-hackerone-platform#.

75 Neta Oren, „Looking Back at Our Bug Bounty Program in 2022“, 15. Dezember 2022. Online verfügbar: https://about.fb.com/news/2022/12/metas-bug-bounty-program-2022/.

76 Google, „Our Greatest Achievements (So Far)“, 8. August 2024. Online verfügbar: https://bughunters.google.com/about/key-stats.

77 BugCrowd, *Inside the Mind of a Hacker 2023. Online verfügbar:* https://www.bugcrowd.com/wp-content/uploads/2023/12/inside-the-mind-of-hacker.pdf. 8.

Dabei handelt es sich um eine globale Belegschaft. Hacker*innen aus Indien, den USA und Nepal stellen durch ihre Teilnahme auf der Plattform von BugCrowd die Top 3 Länder dar.[78]

Die Funktionsweise von Bounty-Programmen ist ziemlich einfach. Die Hacker*innen werden für das Finden und Melden von neuen Bugs bezahlt. Programme definieren dabei, um welche Assets und Bug-Kategorien es geht und wie viel für qualifizierende Bugs gezahlt wird. Eingereichte Fehlerberichte werden überprüft und, falls als relevant erachtet, bezahlt. Nichtqualifizierende Fehler, Berichte, die entweder nicht im Umfang enthalten, Duplikate, ungültig oder einfach nur rein informativ sind, werden nicht bezahlt.

Bounty-Programme sind unterschiedlich organisiert. Einige sind offen für alle Hacker*innen, während andere nur mit Einladungen funktionieren. Zudem laufen Programme entweder eigenständig oder über eine etablierte Bounty-Plattform. Bei Inhouse-Programmen zahlt der Anbieter oder die Organisation für Bugs in den eigenen Software-Anwendungen und kümmert sich zudem um alle Programmbereiche, von der Definition der Servicebedingungen über die Triage eingehender Berichte bis hin zur Zahlung. Bounty-Plattformen hosten Bounties für andere Organisationen gegen eine Servicegebühr oder Provision (die von ihnen angebotenen Dienstleistungen variieren, können aber nicht nur die Verwaltung der Zahlung, sondern auch die Überwachung der Triage, die Definition von Zielvorgaben und andere Programmelemente umfassen). Plattformen verleiten Hacker dazu, Bounty-Programme ausfindig zu machen, die sie – unter anderem über den exklusiven Zugang zu Live-Events und privaten Programmen für aktive und erfolgreiche Teilnehmende – hosten.[79]

Die Vorteile von Bounties: verbesserte Sicherheit und flexibles Arbeiten

Bounty-Programme bieten erhebliche Vorteile für Anbieter, Hacker*innen und damit auch für die breite Öffentlichkeit. Bounties sind im Grunde eine Möglichkeit, Hacker*innen dazu zu bewegen, bestimmte Assets zu überprüfen und zu testen. Für Anbieter sind diese Programme verlockend: Sie versprechen Expertenanalyse zu einem Bruchteil des Preises eines vertraglich vereinbarten Penetrationstests oder einer internen Revision. Pen-Test-Verträge werden unabhängig von den Ergebnissen bezahlt – ein Bericht, der ein Dutzend neue Mängel aufdeckt, kostet genauso viel wie ein Bericht, der zwei findet. Ebenso trägt die interne Sicherheitsarbeit mit Vollzeitmitarbeitenden sämtliche Kosten, die mit diesem Beschäftigungsmodell verbunden sind, nämlich definierte Leistungen, unabhängig vom Arbeitsergebnis. Bei Bounty-Programmen werden Hacker*innen erst bezahlt, wenn sie einen qualifizierenden Bug finden. Die Zeit, die sie damit verbringen, doppelte Fehler aufzuspüren, Sackgassen zu folgen, Berichte außerhalb des Umfangs einzureichen, wird nicht entlohnt.

78 BugCrowd, *Inside the Mind of a Hacker 2023*, 5. Dieser internationale Charakter des Arbeitskräftepools ist über Plattformen und Programme hinweg konsistent. Beispielsweise sind Indien, Nepal und Tunesien nach Herkunftsland die Top-3-Länder für das Facebook-Programm. Oren, „Looking Back at Our Bug Bounty Program in 2022".

79 Ellis und Stevens, *Bounty Everything*, 67–72.

Im Gespräch mit Hacker*innen schwärmen viele vom Nutzen der Bounty-Programme.[80] Sie sehen sie als eine Möglichkeit, Geld zu verdienen, indem sie etwas tun, das sie lieben. Für diejenigen, die in Ländern mit niedrigeren Jahresgehältern arbeiten, können Bounties einen erheblichen Vorteil bieten. Für einige sind Bounties ein Sprungbrett, ein Weg, um einen Einstieg in die stark umkämpfte Welt der Computer-Sicherheit zu finden. Für andere bietet die Flexibilität von Bounty-Programmen eine attraktive Work-Life-Balance – frei zu arbeiten, unabhängig von Ort, Zeit und Umfang.[81]

Die Kosten für die Erstellung sicherer Software zu senken, liegt nicht nur im Interesse von Anbietern und Hacker*innen, sondern auch im öffentlichen Interesse. Die Kosten für unsichere Software werden nicht von Softwareherstellern getragen. Im Gegenteil, der aktuelle Stand der Software-Haftung sorgt dafür, dass die Kosten der Unsicherheit zu einem großen Teil externalisiert und auf die Öffentlichkeit abgewälzt werden.[82] Interventionen wie Bounty-Programme, die die Sicherheit verbessern können, ohne die Kosten signifikant zu steigern, liegen klar im öffentlichen Interesse. Doch Bounty-Programme sind kein Allheilmittel, sondern bergen erhebliche, oftmals nicht anerkannte Risiken.

Prekäre Arbeit und prekäre Technologie: Schaffung und Verbreitung von Risiken

Der frühere Bericht *Bounty Everything* dokumentiert, wie Bug Bounty-Programme in einigen Fällen sogar Risiken für beteiligte Hacker*innen, Anbieter und die breite Öffentlichkeit schaffen können.[83] Bevor wir uns überlegen, wie oder ob Bounties sich für die Herausforderungen der Open Source-Sicherheit eignen, lohnt es sich, innezuhalten, um die heiklen Themen rund um Bounties im Allgemeinen zu betrachten.

Für Hacker*innen kann der Aufstieg von Bug Bounty-Programmen wie der Beginn eines neuen goldenen Zeitalters erscheinen. Nach Jahrzehnten feindlicher rechtlicher Drohungen oder Gleichgültigkeit von Seiten der Software-Anbieter scheinen Bounties eine versöhnende Geste zu sein: die Freiheit, zu experimentieren und zu probieren, mit dem Versprechen eines Gehaltsschecks. Doch da Bounties für viele zu einer Haupteinnahmequelle und für manche sogar zu einem Vollzeitjob geworden sind, ist die Realität komplexer. Bounty-Programme können die schlimmsten Aspekte der Gig Economy wahr werden lassen: Während Anbieter zu einem vergleichsweise niedrigen Preis Zugang zu qualitativ hochwertiger Arbeit erhalten, gehen die Auftragnehmenden ein erhebliches Risiko ein.[84] Katie Moussouris, Gründerin und CEO von Luta Security und Expertin für alles rund um Bug Bounty-Programme, sagte dazu: „Es ist spekulative Arbeit … Es ist der schlechteste Job in der Gig Economy. Uber- oder Lyft-Fahrer*innen werden bezahlt, wenn sie eine Fahrt annehmen

80 Siehe Ellis und Stevens, *Bounty Everything*.

81 Ebd.

82 Siehe Executive Office of the President, *National Cybersecurity Strategy, März 2003. Online verfügbar:* https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf. 20–21.

83 Ellis und Stevens, *Bounty Everything*.

84 Ebd.

und dich zum Flughafen bringen“, aber für Hacker*innen, die in Bounty-Programmen arbeiten, ist die Zahlung nie sicher.[85] Unentgeltliche Arbeit ist Standard. Da Hacker*innen nur bezahlt werden, wenn sie einen qualifizierenden Fehler finden, sind die Stunden und die langen Nächte, die sie ohne Erfolg auf der Suche nach einem neuartigen Bug verbringen, unbezahlte Arbeit.

In diesem Markt haben Bounty-Programme und -Plattformen eine beträchtliche Macht. Sie bestimmen, was als gültige Einsendung gilt, legen Preise fest und können allein entscheiden, wann oder ob ein Bounty bezahlt wird. Wenn sie mit einer Triage-Entscheidung nicht einverstanden sind – wenn ein Bounty-Programm feststellt, dass ein Bug lediglich informativ oder nicht im Umfang enthalten ist und somit nicht für eine Zahlung in Frage kommt –, haben Hacker*innen nur wenige Möglichkeiten, eine Bezahlung einzufordern.[86]

Für Softwarefirmen und -hersteller kann das Versprechen von Bounty-Programmen jedoch schnell ins Stocken geraten. Die Schaffung finanzieller Anreize, die die Öffentlichkeit dazu anregen, Mängel zu melden, kann zu einer Flut von letztlich nutzlosen Berichten führen. Mous-souris, die eine prägende Rolle bei der Etablierung von Bug Bounties für große Unternehmen und Regierungen gespielt hat, stellt fest, dass die weit verbreitete Kommerzialisierung von Bounties zu einem Zustrom von minderwertigen Berichten und Spam geführt hat. Sie analysierte diesen Trend:

> [Die Popularisierung von Bounty-Programmen] begann als ein Prozess, den ich „Beg Bounty“ nenne und bei dem [Teilnehmende] eine Menge von minderwertigen Berichten einreichen … Dann erhoffen sie sich eine Geldprämie, weil [es] für sie mit sehr geringen Kosten verbunden ist … Und diese „Beg Bounty Hunter“ nutzen diese Taktik, um Erträge zu generieren und es funktioniert tatsächlich.[87]

Bei schlechter Vorbereitung können Bounty-Programme Schwierigkeiten haben, aus all der Flut an Einsendungen herauszufiltern, was wichtig und wertvoll ist und was nicht. So kann es passieren, dass Unternehmen, die Bounties zur Verbesserung ihrer Sicherheit nutzen, sich durch einen Sumpf an qualitativ minderwertigen Berichten wühlen müssen und dabei ihr eingeplantes Bounty-Budget bereits für eine Vielzahl kleinerer Probleme aufbrauchen.

Für Unternehmen bedeutet die Verbesserung der Sicherheit im Idealfall mehr als nur die Behebung eines einzigen Bugs: Es geht darum, nachhaltige Entwicklungspraktiken zu entwickeln, die das Auftreten und die Auswirkungen von Fehlern minimieren. Die Reaktion auf eingehende Berichte kann ein wichtiger Teil dieses Entwicklungsprozesses sein, aber wichtige Lehren aus Fehlerberichten zu ziehen – die Integration der Erkenntnisse in die Ursachenanalyse, die nicht nur das Problem beheben, sondern auch die Einführung neuer Fehler verhindern kann –, ist keine einfache

85 Katie M., Interview, 2024.

86 Ellis und Stevens, *Bounty Everything*.

87 Katie M., Interview, 2024.

Angelegenheit. Einige Unternehmen spielen weiter „Whack-a-Mole“, indem sie punktuelle Korrekturen vornehmen, die zwar das Problem beheben, aber die zugrunde liegenden Ursachen der Schwachstelle nicht angehen.

Dies birgt nicht nur Risiken für Hacker*innen und Unternehmen, sondern auch für die breite Öffentlichkeit. Bounty-Programme können hier sogar kontraintuitiv die Sicherheit untergraben. In einer Welt knapper Ressourcen könnten die Zeit und das Geld, die für das Aufspüren und Beantworten jedes neuen Fehlerberichts aufgewendet werden, besser dafür genutzt werden, sichere Software zu entwickeln und zu implementieren. Bounty-Programme können reifen Unternehmen helfen, die bereits in sichere Softwareentwicklungspraktiken investiert und diese eingeführt haben – sie bieten ihnen eine zusätzliche Sicherheits- und Schutzschicht. Aber für Unternehmen, die diese Investitionen noch nicht getätigt haben, erhalten Bounty-Programme eher eine Welt fehlerhafter Software und Bugs: Sie bieten eine Illusion der Sicherheit, während sie einen Zyklus der Softwareentwicklung aufrechterhalten, der Geschwindigkeit, Marktanteile und Kosten gegenüber Sicherheit bevorzugt. **88**

88 Ellis und Stevens, *Bounty Everything*.

Open Source Sicherheits-Bounties: der Weg nach vorn

Open Source-Sicherheit steht den allgemeinen Herausforderungen der Wartung gegenüber – mangelnde Betreuungszeit, zu wenig Aufmerksamkeit, zu viel Aufmerksamkeit – sowie weiteren Herausforderungen, die mit der als wünschenswert empfundenen Sicherheitsarbeit unter Open Source-Freiwilligen einhergehen. Bounties können in bestimmten Fällen helfen, einige dieser Probleme zu beheben. Dennoch gibt es viele Risiken: Bounties können die Herausforderungen bei Wartung und Sicherheit verschärfen, mit denen einige Projekte derzeit konfrontiert sind. So könnten Bug Bounty-Programme, wenn sie nicht umsichtig eingesetzt werden, mehr schaden als nützen. Eine transparente Darlegung dieser Chancen und Risiken kann helfen, einen Weg in die Zukunft zu skizzieren, der Fallstricke vermeidet und die Sicherheit nachhaltig verbessert.

Die Verwundbarkeit senken: Verbesserung der Sicherheit durch Open Source Bounties

Es eröffnen sich viele Möglichkeiten, die Open Source-Sicherheit zu verbessern, indem Projekte dazu befähigt werden, Fehler zu finden, zu beheben und schließlich wichtige Erkenntnisse aus ihnen zu gewinnen. Eine Reihe von Maintainer*innen, die mit der Integration von Bounties in ihre Projekte experimentiert haben, empfinden die Erfahrung als sehr positiv. Viele haben mit der von HackerOne unterstützten Internet Bug Bounty gearbeitet. In diesem Programm triagiert HackerOne eingehende Berichte nicht, sondern die Projekte bleiben selbst für die Verwaltung der Berichte zuständig. Das Finanzmodell unterscheidet sich von anderen Bounty-Programmen: Die IBB wird durch Spenden anderer Unternehmen finanziert – die Open Source-Projekte zahlen die Bounty nicht aus – und auch ein Teil des pro Bounty gezahlten Geldes fließt wieder in die Zielprojekte zurück.

Bounties können auf unterversorgte Open Source-Projekte aufmerksam machen und dazu beitragen, deren Teilnehmenden weiter zu motivieren. Daniel, der Gründer von curl, dem beliebten und weit verbreiteten Kommandozeilen-Tool, entschied sich für Bounties, um das Projekt so sicher wie möglich zu machen. Ein Gespräch mit einem Mitwirkenden führte ihn zu einem Bounty-Modell:

> Irgendwann erwähnte er, dass er nicht mehr bei curl mitarbeiten und sich stattdessen nach einer wirklichen Einnahmequelle umsehen würde. Er wollte in einigen Projekten, die ihm Bounties einbrachten, auf Bug-Jagd gehen.[89]

89 Daniel S., Interview, 2024.

So entschied er sich für Bounties, um Menschen wie diese weiter in seinem Projekt zu halten. Außerdem hoffte er, dass vielleicht ein Bounty-Programm hochqualifizierte Expert*innen, die sonst nicht mit curl beschäftigt waren, zur Prüfung anlocken würde.[90] Seiner Ansicht nach hat das Bounty-Programm Wunder gewirkt, da es „die Häufigkeit von [hochwertigen] Sicherheitsberichten gesteigert hat".[91] Daniel verglich die durch das Bounty-Programm generierten Erkenntnisse mit den aufwändigen Sicherheitsprüfungen, die curl in der Vergangenheit eingesetzt hatte, und die Entscheidung war eindeutig: Bounties lieferten mehr CVEs bei deutlich geringeren Kosten.[92]

Die positiven Erfahrungen, die Daniel mit der IBB gemacht hat, wurden von anderen bestätigt. Sarah von Django stimmte zu, dass Bounties dazu beitragen, die Leute im Projekt zu halten und neue Mitwirkende anzuziehen.[93] Alex, ein Mitwirkender bei Rust, meinte auch, dass Bounties eine positive Möglichkeit sein können, Teilnehmende für das Projekt zu gewinnen.[94] Shai, ein Mitglied des Sicherheitsteams bei Django, führte an, dass Bounties eine gute Möglichkeit seien, Teilnehmende zu würdigen und zu belohnen. Für Projekte, die unter mangelnder Aufmerksamkeit leiden, bieten sie eine mögliche Abhilfe: eine Möglichkeit, neue Talente für das Projekt zu gewinnen. Bei Projekten, die Gefahr laufen, Mitwirkende zu verlieren, kann der zusätzliche Anreiz von Bounties diese dazu motivieren, dabei zu bleiben.

Bounties können auch dabei helfen, schädliche Aufmerksamkeit zu bewältigen, die Projekte erhalten könnten – eine zentrale Herausforderung, die Wartung und Sicherheit untergraben kann. Aaron, ein Haupt-Mitwirkender bei Ruby on Rails und anderen oben vorgestellten Projekten, sprach über die erhöhte Rechenschaftspflicht aufgrund der Bounties. Die Nutzung der HackerOne-Plattform war ein nützlicher Weg, um den Missbrauch und die Belästigung, die teilweise mit einem Bounty-Programm einhergehen, zu bewältigen. Aaron beobachtete, dass, obwohl er es nicht verallgemeinern wollte, „die Leute in der Sicherheitsforschung oft nur Arschlöcher sind".[95] Wenn Geld auf dem Spiel steht, kann es zu einer gewissen Feindseligkeit kommen, da einige Hacker*innen schnell bezahlt werden wollen. Die Partnerschaft mit HackerOne ermöglichte es ihm, Hacker*innen negativ zu bewerten – durch Beurteilungen, die ihrem Ranking auf der Plattform schaden würden und potenziell folgenschwere Auswirkungen hätten. Hier führte die Bounty-Plattform eine gewisse Rechenschaftspflicht ein, die Aaron begrüßte.

Die Risiken von Open Source-Bounty-Programmen: Vorsicht ist geboten

Doch die Einführung von Bounty-Programmen in Open Source-Projekte ist nicht ohne Risiko: Bei manchen Projekten können Bounties die Situation verschlimmern. Die Testimonials von Maintainer*innen, die erfolgreich Bounty-Programme in Open Source-Projekte integriert haben, enthalten auch Worte zur Vorsicht. Bounty-Programme

90 Ebd.

91 Ebd.

92 Ebd.

93 Sarah B., Interview, 2024.

94 Alex C., Interview, 2024.

95 Aaron P., Interview, 2024.

können die Herausforderungen verschärfen, mit denen Projekte bereits konfrontiert sind – sie erhöhen die Verpflichtungen für ohnehin schon in Zeitnot geratene Maintainer*innen, lenken wenig hilfreiche und ablenkende Aufmerksamkeit auf das Projekt und untergraben die fragile Gegenseitigkeit, auf der ein Großteil des Open Source-Ökosystems aufbaut.

Bounty-Programme können die wenige kostbare Zeit, die Maintainer*innen haben, um an ihren Projekten zu arbeiten, auffressen, indem sie unnötige Aufmerksamkeit auf das Projekt lenken. Bounties sind kein Ersatz für eine sichere Entwicklung. Daniel, der Gründer von curl, ist begeistert von seiner Unterstützung für Open Source-Bounties, aber mit einigen Einschränkungen. Für neue Projekte, die gerade erst ihren Weg finden, könnten Bounties eine Katastrophe sein. Zu früh mit einem Bounty zu beginnen, könnte in seinen Worten „ziemlich beängstigend" sein.[96] Er findet, dass sie am besten funktionieren, wenn sie das letzte Teil des Sicherheitspuzzles sind, nicht das erste. Bevor man ein Bounty-Programm startet, ist es wichtig, zuerst über starke Sicherheitspraktiken zu verfügen. Diese Praktiken sind seiner Meinung nach auch weithin bekannt.

> [Um] eine gute Qualität und eine gute Sicherheit zu erreichen, ist es meistens nur eine Frage der Einhaltung all der Entwicklungspraktiken, die wir alle bereits kennen und die wir befolgen sollten. Wir alle wissen genau, wie man Dinge sicher macht, und so gehen Dinge meist schief, wenn wir die Regeln nicht befolgen. Wir folgen ihnen nicht, wir machen keine Tests, wir prüfen nichts und wir überspringen einfach Teile ...[97]

Da es keine angemessenen Sicherheitspraktiken gibt, bergen Open Source-Projekte viele Risiken. Das Projekt wird mit qualitativ minderwertigen Berichten überhäuft, die es unmöglich meistern kann. Katie Moussouris unterstrich dieses Thema: „Bug Bounties fangen idealerweise die Schwachstellen ab, die Sicherheits-Entwicklungsprozesse übersehen."[98] Aber wenn diese Prozesse noch nicht etabliert oder ausgereift sind, wird ein Bounty-Programm die Dinge exponentiell verschlimmern – Spam, kleinere Probleme und andere relevante und irrelevante Berichte werden das Projekt überfluten.

Bounty-Programme können dem Projekt kontraproduktive oder schädliche Aufmerksamkeit verleihen. Der Umgang mit der Menge an Berichten, die einem Bounty-Programm folgen, ist selbst für etablierte Projekte mit Sicherheitsteams, klaren Offenlegungsrichtlinien und Prozessoren sowie jahrelanger Erfahrung eine Herausforderung. Shai, Mitglied des Django Sicherheitsteams, erzählte von dem scheinbar endlosen Strom von E-Mails, in denen immer wieder dieselben Probleme, die nichts mit der Sicherheit zu tun hatten, gemeldet wurden.[99] Daniel stimmte zu,

96 Daniel S., Interview, 2024.

97 Daniel S., Interview, 2024.

98 Katie M., Interview, 2024.

99 Shai B., Interview, 2024.

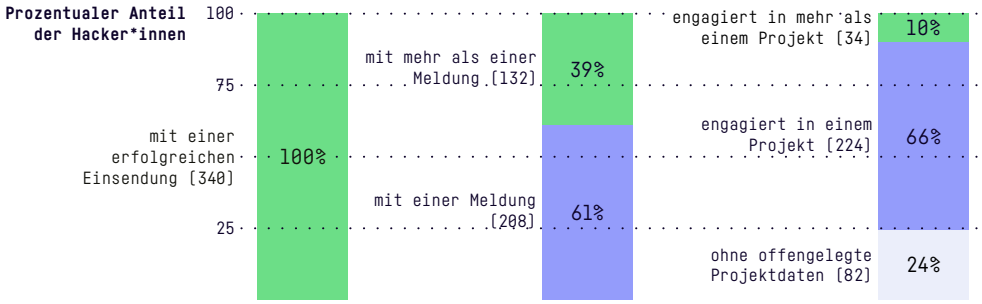
dass der Umgang mit „Mist"-Berichten damit einhergeht. In etablierten Projekten wie Django und curl können sich die Maintainer*innen nur dank ihrer umfangreichen Erfahrung und klaren Richtlinien durch den unvermeidlichen Müll wühlen, der dank des Bounty-Programms hereinströmt. Andere Projekte sind vielleicht nicht so gut aufgestellt. Die wenige Zeit, die Maintainer*innen zur Verfügung haben, um sich ihrem Projekt zu widmen, wird weiter vergeudet, indem sie sich der Arbeit des Triagierens von Fehlerberichten widmen. Diese Arbeit – Triage – ist nichts, das leicht auf eine Bounty-Plattform ausgelagert oder outgesourct werden kann. Die Projekte müssen in all ihren Anforderungen und Eigenheiten verstanden werden. Projekt-Maintainer*innen müssen diese Arbeit erledigen. Wenn Zeit bereits eine knappe Ressource ist, kann das Hinzufügen dieser neuen Aufgabe – das Verwalten von Fehlerberichten – zusätzlich zu ihrem bereits vollen und überfüllten Zeitplan die Wartung und damit auch die Sicherheit untergraben.

Bounty-Programme fördern die Teilnahme, aber sie werden wahrscheinlich nicht dazu beitragen, dass Projekte mit wenigen Mitwirkenden signifikant wachsen. Finanzielle Anreize lenken die Aufmerksamkeit auf ein Projekt, aber Bounties sind wohl kaum eine Abkürzung für das Wachstum einer nachhaltigen und unterstützenden Community. Aaron, der bereits erwähnte Mitwirkende an Rack und Ruby on Rails, befürchtete, dass Bounties „etablierteren Projekten" zwar helfen könnten, kleinere Projekte jedoch aufgrund von unproduktivem Engagement untergehen würden. Wie er bemerkte, „fischen die meisten Leute, die zum Projekt kommen … im Grunde nur nach Bounties".[100] Andere beobachteten, dass diejenigen, die über Bounty-Programme zum Projekt kommen, sich selten, wenn überhaupt, in anderen Rollen engagieren – sie wollen nur das Geld einkassieren.

Tatsächlich zeigt ein Blick auf die öffentlichen IBB-Daten, die auf HackerOne gehostet werden, dass die meisten Hacker*innen in Open Source-Bounty-Programmen unregelmäßige Mitwirkende an Projekten sind – sie reichen in der Regel nur einen erfolgreichen Bounty-Bericht ein. Eine deutliche Mehrheit (62,2 %) derer, die einen Bounty über das IBB-Programm erhalten haben, hat nur eine erfolgreiche Einsendung **(siehe Abbildung 4)**. Das sind keine Hacker*innen, die langfristig in das Fortbestehen und die Wartung des Projekts investieren werden.

Diese Art von lockerem, gewinnorientiertem Engagement zu fördern, kann das Prinzip der Gegenseitigkeit untergraben. Bounties können den frustrierenden Kreislauf der schädlichen Aufmerksamkeit verstärken, der verhindert, was viele Teilnehmende in Open Source-Projekten so schätzen: die Zusammenarbeit und Gemeinschaft. Anstatt Zeit und Energie in die Teilnahme am Projekt zu investieren, werden einige Teilnehmende, die dem Projekt dank eines Bounty-Programms begegnen, unweigerlich „Spaying and Praying" betreiben. Das heißt: Sie versenden Berichte von minderwertiger Qualität in der Hoffnung, vielleicht bezahlt zu werden. Diese Beiträge können mehr Schaden als Gutes bewirken. Zusätzlich können Bounties auch einen Keil durch die Community eines Projekts treiben. Sobald bestimmte Teilnehmende beginnen, für ihre

100 Aaron P., Interview, 2024.



↗ Abb. 4 Internet Bug Bounty-Programm von HackerOne

Daten aus 980 öffentlich zugänglichen Berichten (betrifft 24 Repositorys) im Zusammenhang mit der IBB auf der Plattform von HackerOne.

Beiträge bezahlt zu werden, kann bei anderen, die sich abgemüht haben und ihre Arbeit auf freiwilliger Basis spenden, ein schlechter Beigeschmack entstehen.

In einem kritischen Punkt zur Machbarkeit von Open Source-Bounty-Programmen war man sich einig: Die finanzielle Förderung ist ein zentraler Punkt. Einige Maintainer*innen konnten nur dank des einzigartigen Fördermodells der IBB Bounty-Programme unterstützen und waren sich einig, dass das Programm ohne externe finanzielle Unterstützung nicht möglich sei. Aaron merkte an, dass die Unterstützung der IBB es ermöglichte, die steigende Zahl der eingehenden Berichte zu finanzieren: „Ich weiß nicht, wie viele dieser ReDoS-Attacken wir ausgezahlt haben, aber es ist eine Menge. Und [ich bin] froh, dass wir nicht mein Geld ausgeben.“[101] Daniel stimmte dem zu. Zunächst experimentierte curl mit einem anderen Bounty-Modell, bevor er sich für die IBB entschied. Entscheidend war aber das Modell der finanziellen Unterstützung der IBB. Die finanziellen Auswirkungen einer hohen Anzahl von Meldungen seien nun nach seinen Worten kein Grund zur Sorge: „Es spielt keine Rolle, wenn uns viele Sicherheitsprobleme gemeldet werden. Die Höhe des Betrags, den wir bezahlen, ist für uns kein Problem.“[102] Ohne Förderung durch Programme wie die IBB sind Bounties für Open Source-Projekte weitgehend nicht tragbar.

Best Practices und der Weg nach vorn: eine Zukunft für Open Source Bounties

Open Source-Sicherheit ist eine große Herausforderung: Angesichts der Allgegenwärtigkeit von Open Source-Komponenten in kommerziellen Software-Anwendungen und der Bedeutung von Open Source-Software als eigenständige Werkzeuge ist es schwierig, über Software-Sicherheit zu sprechen, ohne sich mit Open Source-Sicherheit zu befassen. Tatsächlich sind die beiden jetzt untrennbar miteinander verbunden.

Bug Bounty-Programme können in bestimmten Fällen zur Verbesserung der Open Source-Sicherheit beitragen, aber es gibt gute Gründe, vorsichtig vorzugehen.

101 Aaron P., Interview, 2024.

102 Daniel S., Interview, 2024.

Bounty-Programme können die Sicherheit unabsichtlich untergraben. Wenn sie nicht sinnvoll eingesetzt werden, können sie die Herausforderungen verstärken, mit denen Open Source-Projekte bereits konfrontiert sind.

Aus der Verfahrensanalyse werden die folgenden fünf Empfehlungen abgeleitet. Sie liefern eine Reihe von Koordinaten, die helfen können, einen Weg nach vorn festzulegen. Bounty-Programme sind natürlich kein Allheilmittel. Sie sind nur eine von vielen möglichen Sicherheitsinterventionen. Wie bei vielen komplexen Problemen braucht es auch bei den Herausforderungen der Open Source-Sicherheit einen Querschnitt an Lösungsansätzen. Diese Empfehlungen sollen Stiftungen, politischen Entscheidungsträger*innen, Open Source-Projekten, Förderern und anderen helfen zu klären, wann und wie Open Source-Bounty-Programme eingesetzt werden und wann sie auf andere Tools und Ansätze zurückgreifen können.

Empfehlung Nr. 1: Investieren Sie in ganzheitliche Wartungsansätze

Open Source-Sicherheit ist eng mit der guten Wartung eines Projekts verbunden. Schlecht gewartete Projekte sind unweigerlich unsicherer. Investitionen in die Wartung und deren Verbesserung haben Spillover-Effekte, die auch die Sicherheit erhöhen. Um es klar zu sagen: Die meisten Projekte haben es schwer. Ihre Leistung nimmt im Laufe der Zeit bei fast allen Schlüsselkennzahlen ab – man braucht länger, um Probleme zu lösen, und länger, um auf Pull-Anfragen zu reagieren, die Zahl der Mitwirkenden nimmt ab und die Commit-Aktivität ist rückläufig. Die Entwicklung der meisten Open Source-Projekte ist nicht vielversprechend.

Bei diesen Projekten führt die Verbesserung der Basiswartungskapazität zu Sicherheitsgewinnen, die über das hinausgehen, was ein Bounty-Programm bieten kann. Projekten, die derzeit Schwierigkeiten haben, auf eingereichte Probleme zu reagieren, Pull Requests zu lösen oder aktive Mitwirkende anzuziehen, schadet ein Bounty-Programm wahrscheinlich mehr, als es ihnen nützt. Bounties bieten neue finanzielle Anreize, die wenig Aufmerksamkeit auf das Projekt lenken, während Maintainer*innen zusätzliche Arbeit meistern müssen, für die sie wahrscheinlich keine Zeit haben. Für Stiftungen, Konzerne, Regierungen und andere, die in Open Source-Sicherheit investieren wollen, lohnt es sich, weiterhin nach Möglichkeiten zur Unterstützung der Wartung zu suchen. Diese Interventionen können und sollten nicht nur als Investitionen in das Open Source-Ökosystem, sondern als Investitionen in die Sicherheit betrachtet werden.

Empfehlung Nr. 2: Bounties als letzte Wahl

Bug Bounty-Programme funktionieren am besten, wenn sie auf bereits existierende Sicherheitspraktiken aufbauen. Bounties sind kein Ersatz für sichere Entwicklungspraktiken und sie sollten nicht eingesetzt werden, bevor diese Praktiken umgesetzt wurden. Mit der Übernahme von Best Practices wie den

Prozessen rund um den Security Development Lifecycle (SDL) oder das Secure Software Development Framework (SSDF) können grundlegende Sicherheitsmaßnahmen gewährleistet werden.[103] Für ausgereifte Projekte können Bounty-Projekte eine zusätzliche Abwehrschicht und neue Erkenntnisse liefern, die in einem kontinuierlichen Verbesserungszyklus wieder in den Entwicklungsprozess integriert werden können.

Die Einführung eines Bug Bounty-Programms vor Erreichen der Projektreife untergräbt jedoch die Sicherheit. Wird ein Bounty-Programm zu früh gestartet, kann dies zu einer schier endlosen Flut von Meldungen über kleinere Probleme führen. Diese können die Projektbeteiligten überfordern. An unausgereiften Projekten Beteiligte haben nicht nur Mühe, eingereichte Fehler zu überprüfen und zu bearbeiten, sie tun sich auch schwer, sinnvolle Lehren und Erkenntnisse aus neuen Berichten zu ziehen. Oberflächliche Fehler können behoben werden, aber die Ursachen bleiben unbehandelt.

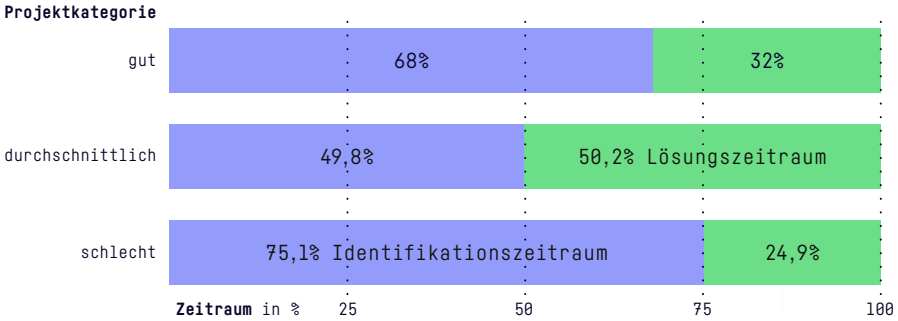
Empfehlung Nr. 3: Nutzen von Bounty-Programmen zur Verbesserung der Fehlersuche

Nicht alle ausgereiften Projekte ziehen den gleichen Nutzen aus einem Bug Bounty-Programm. Entscheidend ist, welche Projekte die attraktivsten Kandidaten für Bounty-Programme sein könnten. Ein Blick auf die untersuchten GitHub- und CVE-Daten zeigt, dass es gute Möglichkeiten gibt, die Sicherheit durch den Einsatz von Bounty-Programmen zu verbessern, um die erste Identifikation und Meldung von Schwachstellen an betroffene Repositorys zu beschleunigen.

Wie verschiedene Projekte auf bekannte Schwachstellen reagieren, ist ein nützlicher, wenn auch grober Indikator für die Sicherheitsleistung. Programme, die schnell bekannte Fehler beheben, schneiden insgesamt besser ab. Das Beheben eines bekannten Fehlers im Code eines Projektes folgt einer Zeitachse, die drei Hauptschritte umfasst: (1) die initiale Offenlegung und Veröffentlichung des Fehlers (CVE-Publikationsdatum oder *pd*); (2) die Identifikation des Fehlers im Code eines bestimmten Projekts (Identifikationsdatum oder *id*); und (3) die Bereitstellung eines Updates, das den Fehler durch das betroffene Projekt behebt (Lösungsdatum oder *rd*). Diese gesamte Zeitleiste, Behebungszeitraum, kann als Differenz zwischen dem Lösungsdatum (*rd*) und dem CVE-Publikationsdatum (*pd*) beschrieben werden.

Die Zeitleiste kann in zwei Segmente unterteilt werden: (1) Identifizierung des Fehlers in einem Projekt und (2) Erstellen und Bereitstellen des Updates. Dieses erste Segment – Identifizierung des Fehlers – ist der Zeitraum zwischen *pd* und *id* (Identifikationszeitraum = *id–pd*). Das zweite Segment – das Erstellen und Bereitstellen des

103 Einen Überblick über den sicheren Entwicklungs-Lebenszyklus-Ansatz finden Sie unter: Microsoft Security Engineering, „Security Development Lifecycle (SDL) Practices“. Online verfügbar: https://www.microsoft.com/en-us/securityengineering/sdl/practices. Einen Überblick über das sichere Software-Entwicklungs-Framework finden Sie unter: Murugiah Souppaya, Kare Scarfone und Donna Dodson, *Secure Software Development Framework (SSDF) Version 1.1: Recommendation for Mitigating the Risk of Software Vulnerabilities, NIST Special Publication 800-218. Februar 2022. Online verfügbar:* https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf.



↑ Abb. 5 Behebungszeitraum: Vergleich des Identifikationszeitraums und des Lösungszeitraums

Daten stammen aus ca. 1.824 Repositorys und 34.385 gelösten und 3.446 offenen Meldungen (Issues) (zum Zeitpunkt der Erhebung).

Updates – ist der Zeitraum zwischen *id* und *rd* (Lösungszeitraum = *rd–id*). Der gesamte Prozess, Behebungszeitraum, umfasst die gesamte Zeitachse (Behebungszeitraum = *rd–pd*).

Die Betrachtung der Zusammensetzung der Zeitleiste Behebungszeitraum zeigt Pain Points und Chancen zur Leistungssteigerung auf. Diese Analyse verwendet gleiche Quantile, um Projekte in „gute", „durchschnittliche" und „schlechte" Kategorien einzuteilen. Gute Projekte liegen in Bezug auf die mittlere CVE-Auflösungszeit im schnellsten Drittel (untere 33 %), durchschnittliche Projekte liegen im mittleren Drittel (33 % bis 67 %) und schlechte Projekte im langsamsten Drittel (Top 33 %). Bei der Untersuchung, wie verschiedene Projekte innerhalb dieser Kategorien ihre Zeit verteilen, stellen sich die Fragen: Wie viel Zeit verbringen sie damit, Schwachstellen in ihrem Repository zu identifizieren? Wie viel Zeit verbringen sie damit, Lösungen für gemeldete Mängel zu entwickeln? Projekte, die in die Kategorie „mittel" fallen, verbringen gleich viel Zeit mit der Identifizierung und Behebung **(siehe Abbildung 5)**, während Projekte der Kategorie „gut" und „schlecht" unverhältnismäßig viel Zeit damit verbringen, veröffentlichte Schwachstellen in ihren Projekten zu identifizieren.

Die Analyse zeigt, dass die durchschnittlichen Leistungserbringer ein Gleichgewicht zwischen Identifikation und Lösung wahren. Doch sowohl starke als auch unterdurchschnittliche Projekte sind deutlich weniger effizient und wenden deutlich mehr Zeit für die erste Identifikationsphase auf. Die Schlussfolgerung aus dieser Analyse ist eindeutig: Es besteht erheblicher Spielraum, um die Sicherheitsperformance von Projekten zu verbessern, indem in Tools und Prozesse investiert wird, die helfen können, Fehler in Open Source-Projekten schneller zu erkennen. Bounty-Programme sollten ausgereifte Projekte ansprechen, die unverhältnismäßig viel Zeit für die Identifikation aufwenden. Die Beschleunigung der ersten Identifikation kann die Exposition verringern und die Sicherheit erhöhen.

Empfehlung Nr. 4: Open Source-Bounty-Programme sollten ethische Praktiken übernehmen

Bounty-Programme sollten die Einzigartigkeit des Open Source-Ökosystems erkennen und darauf achten, das Prinzip der Gegenseitigkeit nicht zu untergraben, sondern es vielmehr zu verstärken. In jedem Fall sollten sie so konzipiert sein, dass die Risiken für die beteiligten Hacker*innen minimiert werden – unter anderem durch rechtliche Absicherung, transparente Zahlungs- und Prüfungsprozesse, das Angebot von Streitschlichtungsverfahren und weitere Maßnahmen.[104] Bounty-Programme bringen im Open Source-Kontext zusätzliche Risiken mit sich, die die Gegenseitigkeit stören und untergraben können. Open Source-Projekte setzen weitgehend auf Freiwilligenarbeit. Diese Beiträge werden oft im Geiste der Gegenseitigkeit angeboten: Sie sind frei für andere, aber die Hoffnung ist, dass auch andere ihre eigenen Beiträge anbieten, die dann frei verfügbar sind (typische Open Source-Lizenzen machen diese Verpflichtung explizit). Bounties bieten selektiv die Zahlung für einige identifizierte Teilmengen von Projektaktivitäten, während andere nicht kompensiert werden. Dies kann die Gegenseitigkeit untergraben und Groll innerhalb des Projekts zwischen denen, die bezahlt werden, und denen, die nicht bezahlt werden, hervorrufen.

Open Source-Projekte haben ähnliche Herausforderungen erfolgreich bewältigt. Da Sponsorenprogramme, Stipendien und Unternehmensunterstützung mit Open Source-Projekten interagieren, ist es üblich, dass bezahlte und unbezahlte Mitarbeitende zusammenarbeiten. Damit Bounties die Gegenseitigkeit nicht untergraben, ist es entscheidend, dass Bounty-Programme von der bestehenden Projekt-Community gestaltet und akzeptiert werden. Programme, die an das Projekt angelehnt sind – und nur mit Rücksprache mit den aktuellen Teilnehmenden verankert werden –, können und sollten Fragen der Gerechtigkeit und Gegenseitigkeit ansprechen. Diese Gespräche sollten der Annahme eines Bounty-Programms vorausgehen. Andernfalls droht eine Entfremdung der aktuellen Projekt-Community auf unerwünschte und sicherheitskontraproduktive Weise.

Empfehlung Nr. 5: Bounty-Finanzierung sollte gemeinschaftsorientiert sein und strukturelle Unterstützung fördern

Open Source Bug Bounty-Programme benötigen Ressourcen, die die Kapazitäten der meisten Open Source-Projekte übersteigen. Die Finanzierung dieser Programme sollte von der größeren Community von Nutzer*innen kommen, die Open Source-Technologien verwenden und davon profitieren. Trotz der Bemühungen von Unternehmen, Regierungen und anderen Organisationen, Open Source-Programmbüros einzurichten und in Open Source-Programme zu investieren, spüren die Mitwirkenden immer noch einen Mangel an Gegenseitigkeit von

104 Für eine detaillierte Erörterung, wie sich die generellen Risiken, die Bounty-Programme auf Forschende verlagern können, mindern lassen, siehe Ellis und Stevens, *Bounty Everything*.

Großanwender*innen, insbesondere profitablen Unternehmen. Bounty-Programme können eine Möglichkeit sein, durch die sich Nutzer*innen aktiv in Open Source-Programme einbringen und diese unterstützen können.

Um die unbeabsichtigten Folgen und die negativen Auswirkungen wirtschaftlicher Anreize zu vermeiden, sollten die Bounties mit allgemeinen Investitionen in die Aufrechterhaltung des Open Source-Projekts gekoppelt werden. Bounties schaffen neue Arbeit für Projekte: So müssen die Maintainer*innen eingehende Fehlerberichte aussortieren und neue Korrekturen entwickeln. Ohne diese neue Arbeit zu unterstützen oder zu entlohnen, kann der Nutzen der Bounty-Programme verloren gehen. Das Hinzufügen neuer Verantwortlichkeiten ohne neue Kapazitäten dürfte die Wartung oder Sicherheit eines Open Source-Projekts kaum verbessern. Die IBB ist ein nützliches Modell dafür, wie sichergestellt werden kann, dass Bounty-Programme auch Maintainer*innen und das größere Projekt unterstützen. In der IBB beteiligen sich Unternehmen und Organisationen an dem revolvierenden Bounty-Fonds; die Zahlungen erfolgen sowohl an die Hacker*innen, die das bisher unbekannte Problem identifiziert haben, als auch an das Projekt (die Auszahlung an das Projekt entspricht einem Prozentsatz der Bounty). Die Förderung erfolgt ohne Einschränkung und Ausrichtung – die Projekte können die Mittel nach eigenem Ermessen einsetzen. Dieses Modell zielt darauf ab, den neuen Aufwand für die Verwaltung der mit einem Bounty-Programm verbundenen Kosten durch einheitlichen allgemeinen Support zu minimieren oder auszugleichen.








Messung der Open Source-Wartung

Um die allgemeinen Wartungsherausforderungen von Projekten zu verstehen, analysierten wir die wichtigsten Wartungskennzahlen, einschließlich der mittleren Problemlösungszeit und der mittleren Pull Request-Lösungszeit (PR) für alle untersuchten GitHub-Projekte. Die Daten zeigen erhebliche Unterschiede in der Geschwindigkeit, mit der Projekte Probleme und PRs lösen.

Wir konzentrieren uns darauf, die Variabilität der Wartungsleistung in verschiedenen Open Source-Projekten zu verstehen, indem wir wichtige Wartungsfunktionen analysieren. Zu diesen Funktionen gehören Kennzahlen im Zusammenhang mit der Commit-Häufigkeit, der Problemlösung und der Pull Request-Handhabung, die kritische Indikatoren für die Projektwartung sind.

Kennzahlen

Vierteljährliches Commit-Gefälle Misst die Änderungsrate der Commit-Aktivität im Lauf der Zeit. Eine positive Neigung (↗) weist auf eine zunehmende Aktivität hin, während eine negative Neigung (↘) auf eine abnehmende Aktivität hinweist.

Problemlösungszeit-Gefälle Erfasst, ob Projekte im Laufe der Zeit Probleme schneller oder langsamer lösen. Ein negativer Wert (▶) bedeutet eine Verbesserung, ein positiver Wert (■) eine Verzögerung.

PR-Lösungszeit-Gefälle Ähnlich wie das Problemlösungszeit-Gefälle, aber für Pull Requests. Es spiegelt wider, ob PR im Laufe der Zeit effizienter gehandhabt werden.

Vierteljährliche mittlere Anzahl der aktiven Mitwirkenden Zeigt die durchschnittliche Anzahl der aktiven Mitwirkenden, die pro Quartal am Projekt teilnehmen. Dies ist ein Indikator für den Zustand und das Engagement der Community des Projekts.

Mittlere Problemlösungszeit
Die durchschnittliche Zeit, die benötigt wird, um Probleme in einem Projekt zu lösen. Tiefere Werte bedeuten schnellere Reaktionszeiten.

Mittlere PR-Lösungszeit Die durchschnittliche Zeit, die benötigt wird, um Pull Requests zu lösen. Schnellere PR-Lösungszeiten deuten auf ein reaktionsfähigeres Projekt hin.

Statistik-Glossar

Mittelwert Der Durchschnittswert für jede Metrik.

Median Der Median ist der Mittelwert in einem sortierten Datensatz. Er repräsentiert den Punkt, an dem die Hälfte der Datenpunkte unten und die Hälfte darüber liegen, wodurch er weniger empfindlich gegenüber Ausreißern im Vergleich zum Mittelwert ist.

Standardabweichung (std) Veranschaulicht, wie groß die Abweichung vom Mittelwert ist.

Min Der tiefste beobachtete Wert, der die beste Performance in dieser Kennzahl angibt.

25 % (1. Quartil) Der Wert, unterhalb dessen 25 % der Daten liegen.

50 % (Median) Der Mittelwert des Datasets. Projekte schneiden an diesem Punkt durchschnittlich ab.

75 % (3. Quartil) Der Wert, unterhalb dessen 75 % der Daten liegen.

Statistiken	Vierteljährliches Commit Gefälle	Problemlösungszeit Gefälle	PR Lösungszeit Gefälle	Vierteljährliche mittlere Anzahl aktiver Mitwirkender	Mittlere Problemlösungszeit (Tage)	Mittlere PR Lösungszeit (Tage)
Mittelwert	↘ -2,67	■ 14,72	■ 10,43	5,64	20,16	9,19
Median	↘ -1,12	■ 3,78	■ 0,79	3,00	9,00	1,00
Std	↗ 20,80	■ 46,86	■ 44,29	11,33	38,36	34,16
Min	↘ -335,50	▶ -647,15	▶ -807,00	1,00	0,00	0,00
25%	↘ -3,66	■ 0,36	● 0,00	2,00	4,00	0,00
50%	↘ -1,12	■ 3,78	■ 0,79	3,00	9,00	1,00
75%	↘ -0,10	■ 13,50	■ 6,96	5,00	21,00	4,50
Max	↗ 415,00	■ 879,00	■ 972,00	271,00	729,00	619,50
Varianz	432,63	▲ 2.195,52	1.961,72	128,35	1.471,28	1.167,19

↑ Abb. 6 Messung der Wartungsaktivität

Daten und Analysen stammen aus 3.707 aktiven Repositorys.

Max Der höchste beobachtete Wert, der die schlechteste Performance in dieser Kennzahl darstellt.

Varianz Der Grad der Verbreitung oder Variabilität der Daten. Eine höhere Varianz bedeutet, dass die Daten mehr gestreut sind, während eine geringere Varianz bedeutet, dass die Werte stärker um den Mittelwert gruppiert sind.

Statistische Erkenntnisse

1. **Vierteljährliches Commit-Gefälle** Im Durchschnitt ist die Commit-Aktivität mit einem Mittelwert von -2,67 (↘) leicht rückläufig. Die hohe Varianz von 432,63 weist auf eine signifikante Variabilität zwischen den Projekten hin. Während einige Projekte ihre Commit-Frequenz beschleunigen, erleben andere beträchtliche Verlangsamungen mit einer minimalen Steigung von -335,5 (↘) und einem Maximum von +415,0 (↗).

2. **Problemlösungszeit-Gefälle** Der Mittelwert von +14,72 (■) zeigt, dass bei Projekten im Durchschnitt die Zeit zur Lösung von Problemen zunimmt. Allerdings deutet die große Abweichung von 2.195,52 (▲) auf eine beträchtliche Diskrepanz hin: Einige Projekte verbessern sich, während andere mit erheblichen Verzögerungen bei der Lösung von Problemen konfrontiert sind. Die extreme Bandbreite von -647,15 (▶) bis +879,0 (■) spiegelt diese Herausforderung wider.

3. **PR-Lösungszeit-Gefälle** Ähnlich wie bei der Problemlösung dauert die Lösung bei Pull-Anfragen länger, mit einer mittleren Steigung von +10,43 (■). Die hohe Varianz von 1.961,72 unterstreicht erneut die große Leistungslücke zwischen den Projekten. Das Gefälle variiert von -807,0 (▶) bis +972,0 (■), was zeigt, dass einige Projekte eine stark verbesserte PR-Lösung haben, während andere deutlich kämpfen.

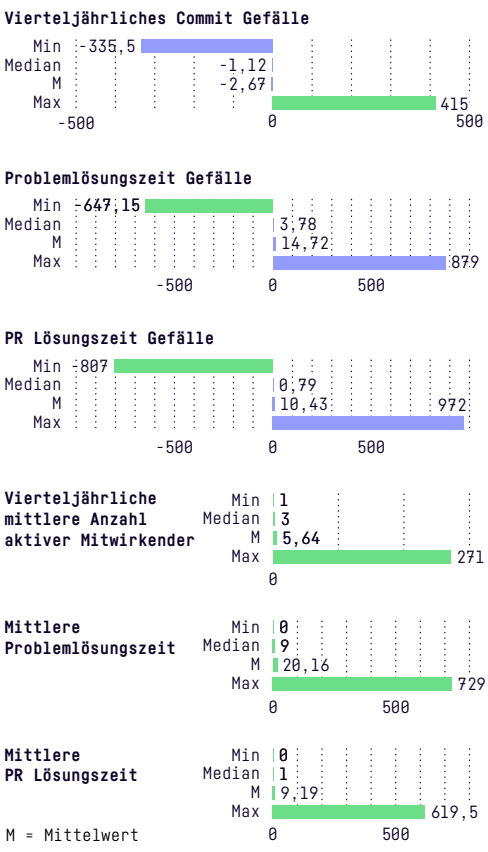
4. **Vierteljährliche mittlere Anzahl der aktiven Mitwirkenden** Die durchschnittliche Zahl der aktiven Mitwirkenden pro Projekt liegt bei 5,64

mit einer Abweichung von 128,35; dies deutet darauf hin, dass die meisten Projekte kleine Teams haben, während einige größere Projekte deutlich mehr aktive Mitwirkende haben. Diese Diskrepanz zeigt sich im Bereich von 1,0 bis 271,0 Mitwirkenden pro Projekt.

5. **Mittlere Problemlösungszeit** Die mittlere Problemlösungszeit beträgt 20,16 Tage, aber die Varianz von 1.471,28 zeigt beträchtliche Unterschiede. Einige Projekte lösen Probleme sehr schnell (mindestens 0,0 Tage), andere brauchen deutlich länger, maximal 729 Tage. Die Variabilität der Problemlösungszeiten deutet darauf hin, dass einige Projekte die Problemlösung effizient verwalten, während andere mit langen Verzögerungen kämpfen.

6. **Mittlere PR-Lösungszeit** Im Durchschnitt werden Pull Requests mit einer mittleren Lösungszeit von 9,19 Tagen schneller gelöst als Probleme. Die Varianz von 1.167.19 zeigt jedoch signifikante Inkonsistenzen zwischen den Projekten, wobei einige PRs fast sofort gelöst werden (Minimum 0,0 Tage) und es bei anderen bis zu 619,5 Tage dauern kann. Dies spiegelt die Variabilität in der Effizienz der verschiedenen Projekte wider.

Zusammengenommen unterstreichen diese Statistiken die Variabilität bei der Wartung und Durchführung von Open Source-Projekten. Während einige Projekte effiziente Lösungszeiten und konstante Mitarbeiteraktivitäten aufweisen, sind andere mit erheblichen Verzögerungen und Wartungsproblemen konfrontiert, was zur Gesamtkomplexität des Projektmanagements beiträgt.



↑ **Abb. 7** Visualisierung von Wartungskennzahlen und -trends

Daten und Analysen stammen aus 3.707 aktiven Repositorys.

Analyse der Wartung und Sicherheit

Gesammelte GitHub-Daten und CVE-Informationen können verwendet werden, um die Beziehung zwischen Wartungspraktiken und Sicherheitsleistung in Open Source-Projekten zu testen. Konkret möchten wir untersuchen, ob Projekte, die gut gewartet werden – solche mit regelmäßigen Commit-Aktivitäten, zeitnahem Pull Request Management (PR) und aktiver Problemlösung –, sicherer sind. Die Sicherheitsleistung wird mittels der mittleren CVE-Lösungszeit daran gemessen, wie schnell ein Projekt bekannte Schwachstellen erkennt und behebt.

Dafür haben wir die logistische Regression verwendet, um die Sicherheitseinstufung eines Projekts anhand seiner Wartungsmerkmale vorherzusagen. Dieser Ansatz ermöglicht es uns, die Beziehung zwischen wartungsrelevanten Kennzahlen (wie PR-Handling und Commit-Aktivität) und Sicherheitsergebnissen quantitativ zu bewerten.

Datenerfassung und Feature-Auswahl

Für diese Untersuchung sammelten wir zwei Haupttypen von Daten aus entsprechenden Open Source-Repositorys:

1. **Wartungsdaten** Dazu gehören Funktionen, die Aufschluss darüber geben, wie ein Projekt gewartet wird, einschließlich:

    - **PR-Lösungszeit-Gefälle** ein Maß dafür, wie sich die PR-Lösungszeiten im Laufe der Zeit ändern;

    - **Vierteljährliche mittlere Anzahl der Commits** Zählung der durchschnittlichen Anzahl der pro Quartal an das Repository abgegebenen Commits, woraus die Entwicklungstätigkeit des Projekts hervorgeht;

    - **Erweiterte vierteljährliche mittlere Anzahl der Commits** eine längerfristige Sicht auf die Commit-Aktivitäten;

    - **Anzahl der offenen Meldungen/ Issues** ein Maß für die Gesamtzahl der offenen Fragen, das einen Einblick in die Arbeitsbelastung oder den Rückstand bei der Bearbeitung von Meldungen des Projekts gibt;

    - **Stargazers Count** ein Proxy für das Interesse der Community an dem Projekt;

    - **Vierteljährliche mittlere Anzahl der aktiven Mitwirkenden** ein Maß dafür, wie viele aktive Mitwirkende in das Projekt involviert sind.

2. **Sicherheitsdaten** Darin enthalten ist die mittlere CVE-Lösungszeit, die als nützlicher Indikator für die Sicherheitsperformance des Projekts dient. Generell gelten Projekte, die CVEs schnell auflösen, als Projekte mit besseren Sicherheitspraktiken.

Projektklassifizierung

Um einen klaren Zusammenhang zwischen Wartung und Sicherheit herzustellen, kategorisierten wir als Nächstes Projekte basierend auf ihrer mittleren CVE-Lösungszeit unter Verwendung gleicher Quantile:

Gute Projekte Projekte, die in die unteren 33 % der mittleren CVE-Lösungszeiten fallen (d. h. diejenigen, die Sicherheitslücken am schnellsten beheben);

Durchschnittliche Projekte Projekte, die in die mittleren 33 % fallen;

Schlechte Projekte Projekte, die in die oberen 33 % der mittleren CVE-Auflösungszeiten fallen (d. h. diejenigen, die Schwachstellen am langsamsten beheben).

Diese Klassifizierung ermöglichte es uns, die Wartungspraktiken von Projekten über verschiedene Sicherheitsleistungsstufen hinweg zu vergleichen.

Modellierung mit logistischer Regression

Um die Sicherheitseinstufung (gut, durchschnittlich oder schlecht) vorherzusagen, haben wir ein logistisches Regressionsmodell verwendet. Die logistische Regression eignet sich für diese Analyse, da sie Wahrscheinlichkeiten für die Klassifizierung liefert und uns so abschätzen lässt, wie unterschiedliche Wartungsmerkmale dazu beitragen, dass ein Projekt in eine bestimmte Sicherheitskategorie fällt.

Der Prozess umfasste mehrere Schritte:

1. **Feature-Auswahl** Wir wählten wichtige Wartungsfunktionen (z. B. PR-Lösungszeit-Gefälle, vierteljährliche mittlere Anzahl der Commits und andere) als Prädiktoren im Modell aus.

2. **Kreuzvalidierung** Wir verwendeten 5-fach geschichtete Kreuzvalidierung, um die Ergebnisse des Modells zu bewerten. Dadurch wird sichergestellt, dass das Modell auf verschiedenen Teilmengen der Daten getestet wird, was das Risiko von Verzerrungen verringert und eine robustere Einschätzung seiner Ergebnisse ermöglicht.

3. **Kennzahlen** Wir haben die Ergebnisse des Modells anhand von drei Kennzahlen eingeschätzt:

    - **Genauigkeit der Kreuzvalidierung** der Anteil der korrekten Vorhersagen, die das Modell in verschiedenen Bereichen der Daten gemacht hat;

    - **Abdeckung** der Anteil des Datensatzes, der für jede Feature-Kombination verwendet wurde, um sicherzustellen, dass das Modell für einen großen Teil des Datensatzes ausgewertet wird;

    - **Verhältnis zwischen Genauigkeit und Abdeckung** Diese Kennzahl gleicht die Genauigkeit des Modells mit der Datenmenge ab, die es abdecken konnte. Ein höheres Verhältnis weist auf einen besseren Kompromiss zwischen Genauigkeit und Abdeckung hin.

Analyse der Auswirkungen von Funktionen

Um herauszufinden, welche Wartungsfunktionen den größten Einfluss auf die Sicherheitsperformance hatten, haben wir verschiedene Kombinationen von Funktionen anhand logistischer Regression bewertet. Wir konzentrierten uns sowohl auf Kombinationen aus einer Funktion als auch aus zwei Funktionen, um zu ermitteln, welche Kennzahlen am besten geeignet waren, um aussagekräftige Sicherheitsergebnisse zu liefern.

Die Top Ten der Kombinationen aus Funktionen basierend auf der Kreuzvalidierungs-Genauigkeit sind in **Abbildung 8** aufgeführt.

Wichtige Erkenntnisse

Die Analyse zeigt, dass bestimmte Wartungspraktiken stark mit einer besseren Sicherheitsleistung korrelieren:

- Das *PR-Lösungszeit-Gefälle* und die *vierteljährliche mittlere Anzahl der Commits* gehörten durchweg zu den leistungsstärksten Feature-Kombinationen. Diese Kennzahlen spiegeln sowohl ein effizientes PR-Management als auch regelmäßige

Funktions-Kombination	CV-Genauigkeit [%]	Abdeckung [%]	Verhältnis von Genauigkeit zu Abdeckung
PR-Lösungszeit-Gefälle + vierteljährliche mittlere Anzahl der Commits	39,63	99,94	0,40
PR-Lösungszeit-Gefälle + erweiterte vierteljährliche mittlere Anzahl der Commits	39,64	99,94	0,40
Vierteljährliche mittlere Anzahl der Commits + Mittlere PR-Lösungszeit	39,14	100,00	0,39
Erweiterte vierteljährliche mittlere Anzahl der Commits + Mittlere PR-Lösungszeit	39,31	100,00	0,39
Erweiterte vierteljährliche mittlere Anzahl der Commits	38,30	100,00	0,38
Erweiterte vierteljährliche mittlere Anzahl der Commits + Anzahl der Stargazers	36,97	100,00	0,37
Erweiterte vierteljährliche mittlere Anzahl der Commits + Anzahl der offenen Probleme	37,61	100,00	0,38
Vierteljährliche mittlere Anzahl der Commits	38,18	100,00	0,38
Vierteljährliche mittlere Anzahl der Commits + Anzahl der offenen Probleme	37,73	100,00	0,38
PR-Lösungszeit-Gefälle + vierteljährliche mittlere Anzahl der aktiven Mitwirkenden	39,55	99,94	0,40

↑ Abb. 8 Prüfen der Verbindung zwischen Sicherheit und Wartung

Daten und Analysen stammen aus 3.707 aktiven Repositorys. Tabelle fasst die Effektivität verschiedener Wartungskennzahlen zur Vorhersage von Sicherheitsergebnissen zusammen.

Entwicklungsaktivitäten wider, die für die Aufrechterhaltung des Projektzustands und der -sicherheit von entscheidender Bedeutung sind.

- Die *erweiterte vierteljährliche mittlere Anzahl der Commits* bietet eine längerfristige Sicht auf die Commit-Aktivität und tauchten häufig in den Top-Kombinationen auf, was die Bedeutung einer nachhaltigen Entwicklung im Lauf der Zeit unterstreicht.

- Die *Anzahl der offenen Probleme* und die *mittlere PR-Lösungszeit* tragen ebenfalls zur Sicherheitsleistung bei, insbesondere in Kombination mit regelmäßigen Commit-Aktivitäten.

Fazit

Die Ergebnisse der logistischen Regressionsanalyse weisen auf einen klaren Zusammenhang zwischen starken Wartungspraktiken und der Sicherheitsleistung hin. Projekte, die regelmäßig gewartet werden und eine hohe Commit-Aktivität und effiziente Bearbeitung von Pull Requests und Problemen aufweisen, haben tendenziell kürzere CVE-Lösungszeiten und sind insgesamt sicherer. Das *PR-Lösungszeit-Gefälle* und die *vierteljährliche mittlere Anzahl der Commits* haben sich als kritische Indikatoren für die Sicherheitsposition eines Projekts herausgestellt.

Diese Analyse stützt die Schlussfolgerung, dass *gut gewartete Projekte sicherer sind*, da robuste Wartungspraktiken dazu beitragen, dass Schwachstellen rechtzeitig erkannt und behoben werden.

Danksagungen
Die Autoren danken und würdigen Yuan Stevens für ihre Beiträge. Die frühere Interview-Runde mit Bug Bounty-Teilnehmenden wurde von Ryan Ellis und Yuan Stevens für ihren gemeinsamen Data & Society Report, *Bounty Everything: Hackers and the Making of the Global Bug Marketplace* (2022), durchgeführt. Dieser Bericht bildet die Grundlage für einen Großteil dieser Arbeit. Yuans Beiträge sind von unschätzbarem Wert.

Dieser Bericht wäre ohne die Teilnahme der Dutzenden Interviewpartner*innen, die freundlicherweise ihre Erfahrungen teilten, nicht möglich. Ihre Einblicke und ihr Engagement werden ebenfalls anerkannt und sehr geschätzt.

Die Unterstützung des Sovereign Tech Fund war entscheidend für die Konzeption und Umsetzung dieses Berichts. Paul Sharratt und Tara Tarakiyee lieferten hilfreiche Rückmeldungen, Anregungen und Beiträge, die zur Gestaltung und Verbesserung des finalen Dokuments beitrugen.

Darüber hinaus möchten sich die Autor*innen für die zusätzlichen Fördermittel der National Science Foundation bedanken. Dieses Material basiert auf Arbeiten, die von der National Science Foundation unter den Stipendiennummern 1915815, 1935520 und 2203175 unterstützt wurden. Alle in diesem Material zum Ausdruck gebrachten Meinungen, Ergebnisse und Schlussfolgerungen oder Empfehlungen sind die der Autor*innen und geben nicht zwingend die Ansichten der National Science Foundation wieder.

Autor*innen
Ryan Ellis
Jaikrishna Bollampalli

Sovereign Tech Fund
SPRIND GmbH
Bundesagentur für Sprunginnovationen

Lagerhofstraße 4
04103 Leipzig
Deutschland

Übersetzung
Supertext Deutschland GmbH

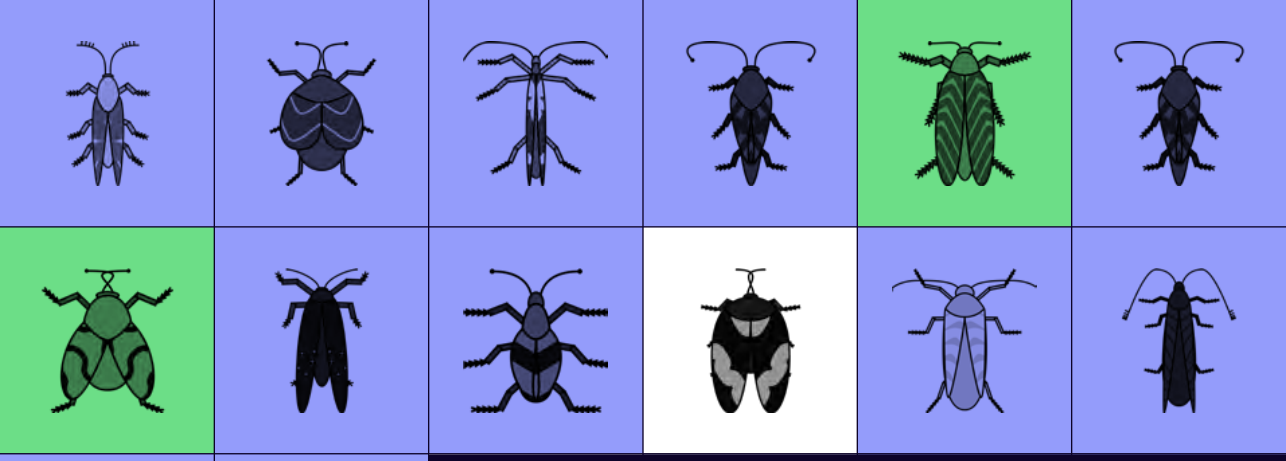
Gestaltung und Satz
Village One eG × Nina Bender

Schriften
Fabrikat Normal & Fabrikat Mono
von HvD Fonts









Dieser Bericht bündelt Forschung zu Bug Bounty-Programmen, Open Source-Communities und Open Source-Sicherheit und untersucht, wie Bounties die Sicherheit verbessern und gleichzeitig unzählige potenzielle Risiken für Sicherheitsforschende, Open Source-Projekte und die breite Öffentlichkeit vermeiden können.