# Manipulation durch Mitwirkung: Sicherheitsrisiko Social-Engineering im Open-Source-Kontext

Fokusbericht 01

20.01.2026

Sophia Schulze Schleithoff & Judith Fassbender

PROTOTYPE FUND

# Inhalt

# 1 Einleitung

Open-Source-Softwarekomponenten finden zunehmend Verbreitung in Anwendungen aller Art — von Softwarebibliotheken, hin zu Benutzer*innenoberflächen oder kompletten Anwendungen. Eine im Jahr 2025 veröffentlichte Studie von Black Duck Software, Inc. zeigt beispielsweise, dass Open-Source-Komponenten in 97% der untersuchten Softwareprojekte verschiedener Branchen vorkommen. Die Anzahl von Open-Source-Komponenten pro Projekt hat sich darüber hinaus in den letzten vier Jahren durchschnittlich verdreifacht.[1] Eine Begleiterscheinung dieses Erfolgs ist, dass Open-Source-Softwarekomponenten ein immer attraktiveres Ziel für Angriffe werden. Zudem werden die Folgen eines erfolgreichen Angriffs durch die zunehmende Verbreitung immer gravierender. Entwickler*innen von Open-Source-Software beheben kritische Sicherheitslücken zwar heute schneller als in den vergangenen Jahren — auch im Vergleich zu proprietärer Software;[2] gleichzeitig ist jedoch auch die Zahl gezielter Angriffe und riskanter Sicherheitslücken in Open-Source-Software erheblich gestiegen.[3]

Zuletzt wurden im Entwicklungsprozess von Open-Source-Softwareprojekten vermehrt Social-Engineering-Angriffe bekannt. Dabei handelt es sich um Angriffe, die gezielt die offenen Kollaborationsprozesse dieser Projekte ausnutzen, um Vertrauen in deren Entwickler*innen-Community zu gewinnen und unbemerkt Schadsoftware oder Sicherheitslücken einzufügen. Gerade Infrastrukturkomponenten, wie sie der Prototype Fund schwerpunktmäßig fördert, sind aufgrund ihrer potenziell weiten Verbreitung als Grundbausteine prädestiniert für solche Angriffe. Gelingt es Angreifer*innen, Lücken in Infrastrukturkomponenten einzubauen, werden potenziell Softwareprojekte, die diese kompromittierten Bausteine integrieren, ebenfalls Teil des Angriffsziels — in solchen Fällen spricht man von Supply-Chain-Angriffen. Komponenten, die in Softwareprojekten verbaut werden, wie etwa Bibliotheken, Protokolle oder APIs, werden Dependencies genannt. Je mehr Projekte auf einer kompromittierten Dependency aufbauen, desto größer der potenzielle Schaden; je mehr Dependencies in einem Projekt verbaut sind, desto größer die Angriffsfläche des Projekts.

Besondere Aufmerksamkeit erlangte 2024 ein Social-Engineering-Angriff auf das freie xz-Utils-Paket, mit dem sich Daten komprimieren lassen und das Teil der meisten Linux-Distributionen ist. Bekannt geworden war der Fall, nachdem ein Entwickler bei Microsoft eine Backdoor in dem Softwarepaket entdeckt hatte, die ein*e Entwickler*in mit zentraler

---

[1] Black Duck Software, Inc. (2025). *2025 Open Source Security and Risk Analysis Report*. https://www.blackduck.com/content/dam/black-duck/en-us/reports/rep-ossra.pdf.

[2] Snyk Ltd. (2024, S. 10). *The State of Open Source*. https://snyk.io/de/lp/state-of-open-source-2024/

[3] Sonatype Inc. (2024). *2024 in Open Source Malware Report*. https://www.sonatype.com/resources/whitepapers/2024-open-source-malware-threat-report

Synopsis Inc. (2024). *2024 Open Source Security and Risk Analysis Report*. https://static.carahsoft.com/concrete/files/1617/1597/8665/2024_Open_Source_Security_and_Risk_Analysis_Report_WRAPPED.pdf

Stellung im xz-Utils-Projekt zuvor selbst in dieses eingebracht hatte. Analysen der Interaktionen im Projekt zeigten, dass dem Angriff mehrere Jahre strategische Mitarbeit im xz-Utils-Projekt vorausgegangen waren, die dazu dienten, Vertrauen aufzubauen und eine zentrale Position in der Governance des Projekts zu erlangen. Die in Open-Source-Projekten übliche prinzipielle Offenheit gegenüber neuen Beitragenden diente so als Einfallstor für einen Angriff, der hunderte Millionen Computer weltweit hätte betreffen können.[4] Das Bundesamt für Sicherheit in der Informationstechnik (BSI) stufte den Vorfall auf Stufe „3/Orange" von vier möglichen Stufen ein, die wie folgt spezifiziert wird: „Die IT-Bedrohungslage ist geschäftskritisch. Massive Beeinträchtigung des Regelbetriebs."[5] Der xz-Utils-Fall zeigt, wie die kollaborative Arbeit in der Entwicklung von (Freier-)Open-Source-Software (FOSS) unter bestimmten Umständen als Einfallstor für Angreifende dienen kann. In diesem Bericht wird es daher darum gehen, wie unterschiedliche Akteur*innen Maßnahmen ergreifen, um Sicherheitslücken zu schließen und welche Hürden in diesem Zusammenhang überwunden werden müssen. Auf dieser Basis werden im Fazit zudem Handlungsempfehlungen diskutiert.

## Überblick und Vorgehen des Berichts

Vor dem Hintergrund des steigenden Risikos für Social-Engineering-Angriffe beleuchten wir im Folgenden, auf welchen Ebenen Maßnahmen von welchen Akteur*innen ergriffen werden, um mit diesem Risiko umzugehen. Um die Thematik weiter in den Kontext des Prototype Fund einzuordnen, haben wir Interviews mit geförderten Projektverantwortlichen geführt und einen Austauschworkshop mit Personen aus dem Umfeld des Chaos Computer Clubs durchgeführt.

In Abschnitt 2 führen wir in den **Kontext von Social-Engineering-Angriffen** ein. Am Beispiel des xz-Utils-Angriffs zeigen wir auf, inwiefern die kollaborative Arbeitsweise in Open-Source-Projekten eine Angriffsfläche für gezielte Manipulation bietet. Darauf folgt Abschnitt 3, in dem wir konkrete Maßnahmen auf drei unterschiedlichen Ebenen behandeln: die **Projektebene** (3.1), die sicherheitsorientierte **Gestaltung von Softwareentwicklungsplattformen** (3.2) am Beispiel von GitHub und die **Policyebene** (3.3), auf der es um übergeordnete Maßnahmen in der Gesetzgebung und Förderung geht. Im letzten Abschnitt (4) ziehen wir ein **Fazit, wie die Ebenen miteinander im Einklang stehen** und an welchen Stellen sich besonderer Handlungsbedarf und Fragen auftun.

## Interviews und Workshop

Für die Einbettung des Themas in den Kontext des Prototype Fund haben wir im August und September 2025 Interviews mit sechs Projektverantwortlichen und Maintainer*innen von durch den Prototype Fund geförderten Projekten geführt und qualitativ ausgewertet. Bei den Projekten handelte es sich vorwiegend um Projekte mit Sicherheitsfokus, mit Kernteams von 3-10 Personen. Abgefragt wurden die Beschaffenheit und Verbindung zu Beitragendencommunity, etwaige Veränderungen in der Wahrnehmung der

[4] Roose, Kevin (03.04.2024). Did One Guy Just Stop a Huge Cyberattack?, *The New York Times*. https://www.nytimes.com/2024/04/03/technology/prevent-cyberattack-linux.html

[5] Bundesamt für Sicherheit in der Informationstechnik (2024). *Kritische Backdoor in XZ für Linux*. https://www.bsi.bund.de/SharedDocs/Cybersicherheitswarnungen/DE/2024/2024-223608-1032.pdf?__blob=publicationFile&v=5

Sicherheitslage für die Projekte nach dem xz-Utils-Fall, der Umgang mit Vertrauens- und Sicherheitsfragen und das Ideal von Communityarbeit und Sicherheit.

Im September 2025 haben wir zudem unter dem Titel „*A conversation on approaches to prevent social engineering security attacks*“ einen Workshop auf der Chaos Computer Club Veranstaltung InselChaos 2025 auf Rügen durchgeführt. Mit den über zehn Teilnehmenden, wurden die Passung formalisierter Ansätze, Erfahrungen im Umgang mit Social-Engineering-Angriffen sowie Bedürfnisse nach neuen beziehungsweise anderen Ansätzen diskutiert. Unter den Teilnehmenden waren Forschende, Mitarbeiter*innen von Stiftungen mit FOSS-Fokus, Entwickler*innen und Systemadministrator*innen aus unterschiedlichen Unternehmen, Nutzer*innen von Open-Source-Software-Komponenten und Verwaltungsmitarbeiter*innen. Während des Workshops wurde eine Mitschrift erstellt und für diesen Bericht qualitativ ausgewertet.

Folgende Aspekte wurden ausgewertet: Maßnahmen die in den Projekten ergriffen werden; benötigte Unterstützung und Bedürfnisse, übergeordnete Aussagen zum Problem von Social-Engineering-Angriffen.

# 2 Social-Engineering als Angriffsmuster in der kollaborativen Softwareentwicklung

Das Vorgehen der Angreifenden im xz-Utils-Projekt zeigt, wie moderne Social-Engineering-Angriffe im Kontext der Softwareentwicklung ablaufen können und wie typische Entwicklungsvoraussetzungen in Open-Source-Projekten, aber auch aktuelle politische und technische Entwicklungen, sie begünstigen.

## Neue Angriffsmuster in Open-Source-Projekten

Das angegriffene xz-Utils-Projekt, stellt eine Softwarebibliothek in der Form einer Sammlung von Open-Source-Tools bereit. Diese Softwarebibliothek ist ein Teil von vielen Linux-Betriebssystemen und dafür verantwortlich Daten zu komprimieren und zu dekomprimieren.[6] Wieso ein solcher Baustein für einen Angriff ausgewählt wurde erschließt sich nicht gleich. Das xz-Utils-Paket wurde im Sinne einer Supply-Chain-Attacke angegriffen, da es mit zwei kritischen Softwarekomponenten agiert, auf die durch den Angriff zugegriffen werden sollte. Die eine Komponente, OpenSSH, stellt verschlüsselte Verbindungen zu (remote) Servern her. Die andere Komponente, systemd, startet und stoppt Prozesse in Linux Betriebssystemen. Durch einen kombinierten Angriff auf beide Komponenten innerhalb der xz-Utils-Bibliothek, kann der Authentifizierungsprozess von verschlüsselten SSH-Verbindung umgangen werden.[7] Die Angreifenden hätten so

[6] Goodin, Dan (02.04.2024). The XZ Backdoor: Everything You Need to Know, *Wired*. https://www.wired.com/story/xz-backdoor-everything-you-need-to-know/

[7] Przymus, Piotr; Durieux, Thomas (2025, S. 92ff.). Wolves in the Repository: A Software Engineering Analysis of the XZ Utils Supply Chain Attack, *2025 IEEE/ACM 22nd International Conference on*

Schadcode auf Server und andere Computer hochladen und ausführen können, und diese so fernsteuern können. Die Schäden, die dadurch entstehen könnten sind umfangreich, Vermutungen reichen von der Einbringung weiter Schadsoftware bis hin zu Cryptodiebstahl.[8]

Im Zentrum des Angriffs auf xz-Utils stand dessen bis dahin einziger Maintainer, der als solcher alleinige Verantwortung dafür trug, Projektziele zu definieren, Beiträge zu verwalten und Beitragende zu koordinieren. Während die Art, auf die die Backdoor in xz-Utils eingeschleust wurde, technologischer Natur war, bestand der größte Teil an Vorbereitung aus vertrauensbildenden Maßnahmen innerhalb des Softwareprojekts. Dazu gehörten Codebeiträge, die Überprüfung der Beiträge anderer Entwickler*innen sowie die Übernahme nicht-technischer Aufgaben.[9] Der/die Angreifende nutzte insofern gezielt Dynamiken in Open-Source-Softwareprojekten aus. Man geht davon aus, dass die Beiträge von Kernentwickler*innen mit einer durch vorherige Beiträge aufgebauten Reputation schneller und häufiger angenommen werden als diejenigen von unbekannten Entwickler*innen.[10] Bereits nach einem Jahr erhielt er/sie auf diese Weise Schreibrechte für das Projekt. Kurze Zeit später wurde der/die Angreifende Maintainer*in, erhielt also noch weitreichendere Rechte innerhalb des Projekts. Der Schritt zur Übertragung von Maintainer*innen-Rechten wurde durch weitere Beitragende zum Projekt unterstützt, bei denen es sich mutmaßlich um sogenannte Sockenpuppen-Accounts handelte — dabei handelt es sich um zu diesem Zweck angelegte Fake-Accounts. Diese übten Druck auf den bis dahin alleinigen Maintainer aus[11] und machten sich zunutze, dass die Verantwortung für weitreichende Entscheidungen bei nur einer Person lag. Eine solche Unterbesetzung stellt in vielen Open-Source-Projekten ein indirektes Sicherheitsrisiko da. Open-Source-Projekte sind häufig Ein-Personen-Projekte.[12] Aufgefallen ist der Angriff bei Microsoft, mehr oder minder zufällig. Aufgrund einer ungewöhnlich hohen CPU-Auslastung in einem Teilprozess, stellte der bereits erwähnte Entwickler Nachforschungen an und konnte den Angriff so aufdecken.[13] Da die Backdoor verhältnismäßig schnell gefunden wurde und kein Schadensfall bekannt ist, geht man davon aus, dass der Angriff nicht erfolgreich war.[14]

---

*Mining Software Repositories (MSR)*, Ottawa, ON, Canada, S. 91-102. https://doi.org/10.1109/MSR66628.2025.00026

[8]Goodin, Dan (02.04.2024). Siehe Fußnote 6.

[9]Przymus, Piotr; Durieux, Thomas (2025, S. 92ff.). Siehe Fußnote 7.

[10]Bosu, Amiangshu; Carter, Jeffrey C. (2014). Impact of developer reputation on code review outcomes in OSS projects: an empirical investigation, *ESEM '14: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* 3, S. 1-10. https://doi.org/10.1145/2652524.2652544

[11]Prymus & Durieux (2025, S. 94f.). Siehe Fußnote 7.

[12]Bressers, Josh (28.08.2025). *Open Source is one person*, Open Source Security. https://opensourcesecurity.io/2025/08-oss-one-person/

[13]Prymus & Durieux (2025, S. 94f.). Siehe Fußnote 7.

[14]Prymus & Durieux (2025, S. 98). Siehe Fußnote 7.

Bemerkenswert ist der Angriff auch deshalb, weil dafür erhebliche zeitliche Kapazitäten, strategisches Kalkül und technische Kenntnisse erforderlich waren. Zwar konnte bislang nicht geklärt werden, wer hinter dem über mehrere Jahre sorgfältig vorbereiteten Angriff steckt; aufgrund des beträchtlichen Ressourceneinsatzes wird allerdings vermutet, dass staatliche Akteur*innen dafür verantwortlich sein könnten.[15] Auch andere Angriffe auf Open-Source-Softwarepakete werden staatlichen Akteur*innen zugeordnet. Ein Beispiel sind eine Reihe von 2025 bekannt gewordenen schädlichen Paketen, die im Verdacht stehen von der durch Nordkorea unterstützten Lazarus-Gruppe in Umlauf gebracht worden zu sein.[16] Nicht nur für kommerziell motivierte, sondern mutmaßlich auch für staatliche Akteur*innen scheinen Angriffe auf Open-Source-Projekte attraktiver zu werden. Sie verfügen über die erforderlichen Ressourcen für komplexe Social-Engineering-Angriffe auf Open-Source-Projekte. Nehmen diese Projekte wie xz-Utils eine zentrale Rolle in Softwarelieferketten ein, kann sich der Einsatz dieser Ressourcen lohnen, um politische Ziele zu verfolgen.

Auch technische Entwicklungen befördern Social-Engineering-Angriffe. Auffällig ist am Verhalten des/der Angreifer*in auf xz-Utils, dass neben Beiträgen zum Code des Projekts in weitaus größerem Umfang nicht-technische Aufgaben wie Dokumentation, Übersetzungen und Community-Management übernommen wurden. Der Account, von dem die Backdoor in das Projekt eingefügt wurde, verwendete dabei regelmäßig Software, um sprachliche Fehler im Code oder in dessen Dokumentation zu identifizieren sowie um Übersetzungen automatisiert anzufertigen. Die Verfügbarkeit zunehmend leistungsfähiger KI-Anwendungen ermöglicht es, auch mit vergleichsweise geringem Ressourceneinsatz in hoher Frequenz zu Open-Source-Projekten beizutragen, um Reputationsmechanismen auszunutzen — etwa durch automatisiert generierte Codebeiträge, bei denen allerdings offen bleibt, inwiefern diese einen qualitativ angemessenen Beitrag zu Projekten leisten. Dadurch werden Social-Engineering-Angriffe auf Open-Source-Entwicklungspraktiken auch für mehr Akteur*innen ohne Zugang zu personellen Ressourcen in größerem Umfang leichter umsetzbar.[17]

## Offenheit vs. Sicherheit?

Warum offene und kollaborative Entwicklungsprozesse von Open-Source-Software durch Social-Engineering-Angriffe unter Druck geraten könnten bzw. eine Abwägung zwischen Offenheit und Sicherheit vor gezielter Manipulation in Open-Source-Projekten erforderlich sein könnte, ist erklärungsbedürftig. Eine verbreitete Auffassung ist, dass Open-Source-Entwicklungspraktiken ein Vorteil für deren Sicherheit sind. Dafür spricht, dass der veröffentlichte Code für die Allgemeinheit überprüfbar ist und deshalb Sicherheitslücken leicht erkannt und behoben werden können, oder in den Worten von Eric Raymond: „Given

---

[15]Greenberg, Andy; Burgess, Matt (03.04.2024). The Mystery of ‘Jia Tan,’ the XZ Backdoor Mastermind, *Wired*. https://www.wired.com/story/jia-tan-xz-backdoor/.

[16]Sonatype Inc. (08.07.2025). Open Source Malware Index Q2 2025: Data exfiltration remains a leading threat, *Sonatype Blog*. https://www.sonatype.com/blog/open-source-malware-index-q2-2025

[17]Przymus & Durieux (2025, S. 97). Siehe Fußnote 7.

enough eyeballs, all bugs are shallow."[18] Hinzu kommt, dass die in Open-Source-Projekten üblichen offenen Kommunikationswege und kurzen Veröffentlichungszyklen dazu beitragen, dass Sicherheitslücken unmittelbar nach Bekanntwerden gemeldet und geschlossen werden können.[19]

Ob die Vorteile des offenen und kollaborativen Entwicklungsprozess zum Tragen kommen, hängt allerdings davon ab, ob der Code tatsächlich überprüft wird und ob Sicherheitslücken gemeldet und geschlossen werden können. Eine implizite Annahme im Argument für die Sicherheit von Open-Source-Software ist, dass sich eine ausreichend große Zahl an Personen mit den notwendigen Ressourcen findet, um den veröffentlichten Code im Interesse aller zu analysieren. Die Tatsache, dass die Backdoor im xz-Utils-Paket aufgedeckt wurde, scheint diese Annahme zu bestätigen. Allerdings ist davon auszugehen, dass ein großer Teil der Nutzenden von Open-Source-Software nicht motiviert oder aus unterschiedlichen Gründen nicht in der Lage ist, Sicherheitsüberprüfungen durchzuführen. Eine Umfrage von Wen et al. (2019) unter Open-Source-Softwareentwickler*innen ergab beispielsweise, dass nur zwei Drittel sich zutrauen, ihre Software effektiv gegen das Ausnutzen eines Sicherheitsproblems zu schützen.[20] Werden Open-Source-Softwarekomponenten eingebunden, müssen diese regelmäßig aktualisiert und auf ihre Sicherheit hin überprüft werden, was häufig nicht der gängigen Praxis entspricht. Die bereits erwähnte Studie von Black Duck Software, Inc. (2025) zeigt, dass die meisten Codebasen auf veralteten Komponenten aufbauen, für die bereits neuere Versionen zur Verfügung stehen.[21]

Zu denjenigen, die motiviert, mit entsprechenden Ressourcen ausgestattet und dazu in der Lage sind, Sicherheitslücken zu finden und Software zu entwickeln, zählen auch Personen, die diese für ihre eigenen Zwecke ausnutzen wollen. Ihnen bieten sich eine Vielzahl von Angriffsmöglichkeiten und das nicht nur im sichtbaren Softwarecode selbst, sondern auch beim Kompilieren desselben. Wie Ken Thompson 1984 in seinem wegweisenden Vortrag *Reflections on Trusting Trust* aufgezeigt hat, ist deshalb letztlich Vertrauen in die Entwickler*innen von Open-Source-Software und deren Motivation, im Interesse aller zu handeln, zentral.[22] Erschwert wird der Aufbau von Vertrauen dann, wenn, wie in vielen Open-Source-Softwareprojekten üblich, Personen aus verschiedenen Kontexten mit losen Verbindungen ehrenamtlich und mit begrenzten zeitlichen und

---

[18] Raymond, Eric S. (2001, S. 19). *The Cathedral and the Bazaar*, Revised Edition, O'Reilly: Sebastopol, CA.

[19] Payne, Christian (2002, S. 66). On the security of open source software, *Information Systems Journal* 12(1), S. 61-78. https://doi.org/10.1046/j.1365-2575.2002.00118.x

[20] Wen, Shao-Fang; Kianpour, Mazaher; Kowalski, Stewart (2019). An Empirical Study of Security Culture in Open Source Software Communities, *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, S. 863-870. https://doi.org/10.1145/3341161.3343520

[21] Black Duck Software, Inc. (2025, S. 25). Siehe Fußnote 1.

[22] Thompson, Ken (1984). Reflections on Trusting Trust, *Communications of the ACM* 27(8), S. 761 - 763. https://doi.org/10.1145/358198.358210

finanziellen Ressourcen zusammenarbeiten. Insofern birgt die Offenheit von Open-Source-Softwareentwicklung auch eine Angriffsfläche für gezielte Manipulation.

Eine bessere Finanzierung von Open-Source-Softwareentwickler*innen, wie sie insbesondere im Zusammenhang mit dem xz-Utils-Angriff immer wieder gefordert wurde,[23] ist insofern nur ein — wenn auch überaus wichtiger — Teil der Lösung.

# 3 Maßnahmen zur Risikoreduktion

Instrumente, die zum Schutz vor Social-Engeneering-Angriffen erprobt werden, sind Governance-Mechanismen in Open-Source-Projekten, das Design von Softwareentwicklungsplattformen sowie die Förderung und Regulierung von Softwareprodukten. Im folgenden Abschnitt werden diese näher ausgeführt.

## 3.1 Projektebene: Governance in Open-Source-Projekten

Die Governance von Open-Source-Projekten, d. h. die Erwartungen, Regeln, Strukturen und Praktiken nach denen Entwickler*innen zusammenarbeiten, hat einen entscheidenden Einfluss darauf, wie Vertrauen hergestellt und Offenheit für neue Beiträge aufrechterhalten wird. Um die Gestaltung einer entsprechenden Governance zu fördern wird auf Best Practices als vorbildliche Verfahrensweisen zurückgegriffen.

Spezifische Best Practices für den Umgang mit Social-Engineering spielen in allgemeinen Leitfäden für Sicherheit in Open-Source-Projekten, wie sie beispielsweise von der Open Source Security Foundation (OpenSSF),[24] der Linux Foundation[25] und von GitHub[26] veröffentlicht wurden, keine zentrale Rolle. Diese Leitfäden enthalten allerdings auch Empfehlungen, die vor Social-Engineering-Angriffen schützen können. Weitere Empfehlungen in Reaktion auf den xz-Utils-Angriff haben die OpenSSF und die OpenJS Foundation in einem Blogpost veröffentlicht.[27]

---

[23] z. B. Hodgson, Matthew (04.04.2024). *Open Source Infrastructure must be a publicly funded service*, Matrix Blog. https://matrix.org/blog/2024/04/open-source-publicly-funded-service/

[24] Boehm, Mirko; Carter, Hilary; Osborne, Cailean (2025). *Pathways to Cybersecurity Best Practices in Open Source*, The Linux Foundation. https://www.linuxfoundation.org/hubfs/LF%20Research/lfr_cra_031725a.pdf?hsLang=en

[25] OpenSSF Best Practice Working Group (14.06.2023). *Concise Guide for Developing More Secure Software*. https://best.openssf.org/Concise-Guide-for-Developing-More-Secure-Software

[26] GitHub (o. D.). *Security Best Practices for your Project*, Stand: 26.04.2025. https://opensource.guide/de/security-best-practices-for-your-project/

[27] Bender Ginn, Robin; Arasaratnam, Omkhar (15.04.2024). Open Source Security (OpenSSF) and

Inwieweit die Best Practices zur Anwendung kommen, lässt sich anhand von empirischen Untersuchungen der Governance in Open-Source-Projekten beurteilen. Welche Rolle Sicherheit in der Open-Source-Governance von Open-Source-Software in der Praxis spielt, haben beispielsweise Wen et al. 2019 anhand einer Umfrage unter Entwickler*innen von Softwareprojekten auf GitHub untersucht. Demnach sahen Open-Source-Entwickler*innen Softwaresicherheit mehrheitlich als wichtig für den Erfolg ihrer Projekte an, vertrauten der Governance ihrer Community in dieser Hinsicht und erklärten, sich an Richtlinien für sichere Softwareentwicklung zu halten. Jedoch neigten Entwickler*innen auch dazu, ihren eigenen Anteil am Aufbau und Erhalt von Sicherheit gering einzuschätzen und fürchten Zielkonflikte zwischen ihrer Arbeit und Sicherheitsanforderungen.[28]

Im Folgenden wird ein Überblick über von Open-Source-Organisationen empfohlene Best Practices zum Umgang mit Social-Engineering-Angriffen in der Open-Source-Softwareentwicklung gegeben, sowie Studienergebnisse zur Implementierung von Leitlinien und technischen Sicherheitsmaßnahmen dargestellt.

### 3.1.1 Regeln und Strukturen

Zentral ist zunächst die Empfehlung, Angriffe zu dokumentieren und relevante Informationen zu veröffentlichen. Ziel ist dabei, sowohl die Projekt-Community als auch andere Open-Source-Entwickler*innen — insbesondere solche, die auf dem betroffenen Projekt aufbauen — transparent über Vorgänge und Risiken zu informieren. Das genaue Vorgehen sollte in einer Sicherheitsrichtlinie festgehalten und an prominenter Stelle veröffentlicht sein. Außerdem wird dazu geraten, die Kontaktinformationen der im Projekt mit Sicherheitsfragen befassten Personen offen zur Verfügung zu stellen.

In Bezug auf Rollen und Verantwortlichkeiten empfehlen die Best-Practice-Leitfäden grundsätzlich verschiedene Personen an Entscheidungen z. B. über die Annahme von Codebeiträgen zu beteiligen, um Risiken zu reduzieren. Die Zuständigkeit für Sicherheitsfragen sollte ausdrücklich einer Person bzw. einem Team übertragen werden. Eine weitere Best Practice für individuelle Entwickler*innen ist es, regelmäßig das eigene Verhältnis zu anderen am Projekt beteiligten Personen sowie deren Aktivität zu überprüfen, um deren Vertrauenswürdigkeit besser einschätzen zu können.

#### Implementierung von Regeln und Strukturen in Open-Source-Projekten

Explizite Regeln und Strukturen für IT-Sicherheit gibt es in Open-Source-Projekten häufig nicht. In der erwähnten Studie von Wen et al. (2019) wurden 254 Entwickler*innen befragt. Es zeigte sich, dass nur etwa die Hälfte der Projekte über allgemeine Richtlinien und Anleitungen für den Umgang mit sicherheitsrelevanten Themen wie Meldeverfahren für Schwachstellen, IT-Sicherheitstests und sicheren Entwicklungspraktiken verfügte. Zudem zeigte die Studie, dass in vielen Projekten auch keine Personen vorhanden waren, die

---

*OpenJS Foundations Issues Alert for Social Engineering Takeovers of Open Source Projects, OpenSSF Blog*. https://openssf.org/blog/2024/04/15/open-source-security-openssf-and-openjs-foundations-issue-alert-for-social-engineering-takeovers-of-open-source-projects/

[28]Wen et al. (2019, S. 867). Siehe Fußnote 20.

speziell mit diesen Themen befasst waren. Spezifische Kommunikationskanäle wie Mailinglisten oder Foren für sicherheitsrelevante Fragen bestanden in 40% der Projekte.[29] Zu ähnlichen Ergebnissen kamen Wermke et al. 2022 in einer Interviewstudie, ebenfalls mit Entwickler*innen von Softwareprojekten auf GitHub.

Wie Rollen und Verantwortlichkeiten von Beitragenden verteilt sind und wie Beiträge neuer Entwickler*innen behandelt werden, unterschied sich in den Open-Source-Projekten nach Angaben der interviewten Entwickler*innen stark und unabhängig von deren Größe. Während manche Projekte dezentrale Strukturen beschrieben, in denen Beitragenden bereits nach kurzer Mitarbeit weitreichende Rechte gewährt werden, definierten andere Projekte ihr Kernteam mit der Befugnis Codereviews durchzuführen und Beiträge anzunehmen deutlich hierarchischer.

Ein weiteres Ergebnis der Studie von Wermke et al. ist, dass die meisten Open-Source-Communities ihren Umgang mit Vorfällen, bei denen die Sicherheit des Projekts oder das Vertrauen zu anderen Entwickler*innen aus der Projekt-Community beeinträchtigt werden, pragmatisch und erst bei Bedarf festlegen. Mehr als die Hälfte der interviewten Entwickler*innen hatte eigenen Angaben zufolge noch keine solchen Vorfälle erlebt. Projekte, in denen es bereits zu Sicherheitsvorfällen gekommen war und die infolge dessen über spezifische Strategien für den Umgang mit ihnen verfügten, waren häufig größer. Entwickler*innen kleinerer Projekte berichteten dagegen, sofern bereits relevante Erfahrungen vorlagen, von dynamischen Einzelfallentscheidungen.[30]

### 3.1.2 Technische Sicherheitsmaßnahmen

Empfehlungen für die Umsetzung technischer Sicherheitsmaßnahmen umfassen Maßnahmen, die dazu dienen, die Identität von Beitragenden zu überprüfen und nur als ausreichend vertrauenswürdig eingestuften Personen Bearbeitungsrechte zu übertragen. Eine Best Practice für die Authentisierung, d.h. den Identitätsnachweis, einzelner Beitragender in Open-Source-Projekten ist die (kryptographische) Signatur von Codebeiträgen. Entwickler*innen können ihre Autorenschaft an bestimmten Beiträgen durch die Public-Key-Verschlüsselungsverfahren PGP, SSH und S/MIME nachweisen.

Zudem wird eine effektive Zugriffskontrolle empfohlen, um sicherzustellen, dass Beitragende in Code-Repositorien nur solche Änderungen vornehmen können, zu denen sie gemäß der im Projekt zugewiesenen Rollen und Verantwortlichkeiten autorisiert sind. Mittel für eine solche Zugriffskontrolle sind u. a. eine granulare Rechtevergabe, Multi-Faktor-Authentifizierung für Personen mit erweiterten Rechten sowie das Einstellen von Schutzregeln für wichtige Branches, also für Arbeitsbereiche im Versionskontrollsystem Git. Mögliche Regeln sind beispielsweise, dass Codebeiträge signiert sein müssen, dass

[29] Wen et al. (2019, S. 867f.). Siehe Fußnote 20.

[30] Wermke, Dominik; Wöhler, Noah; Klemmer, Jan H.; Fourné, Marcel; Acar, Yasemin; Fahl, Sascha (2022). Committed to Trust: A Qualitiative Study on Security & Trust in Open Source Software Projects, *43rd IEEE Symposium on Security and Privacy*. https://doi.org/10.1109/SP46214.2022.9833686

ein Schreibschutz besteht oder alle Anmerkungen zu einem Beitrag als geklärt markiert sein müssen, bevor Änderungen übernommen werden können.

Insgesamt scheint die Offenheit des Entwicklungsprozesses durch die empfohlenen Maßnahmen kaum eingeschränkt. Weder Dokumentation und Information noch eine klare technische und organisatorische Rollenverteilung unter wachsamen Entwickler*innen stehen einer offener Kollaboration grundsätzlich im Weg. Allerdings müssen diese Maßnahmen von einer ausreichend großen Zahl an Entwickler*innen umgesetzt werden. Verfügt ein Projekt nicht über die Ressourcen dafür, muss eine Abwägung getroffen werden: Eine Möglichkeit ist ein restriktiverer Umgang mit Beiträgen fremder Kontributor*innen. Das verspricht mehr Sicherheit, kann aber wertvolle, neue Community-Mitglieder abschrecken, etwa wenn Reviewprozesse zu langwierig sind.[31] Die Priorisierung offener Entwicklungsprozesse, in denen Kontributor*innen ohne entsprechende Sicherheitsmaßnahmen ein Vertrauensvorschuss gewährt wird, bringt dagegen Sicherheitsrisiken mit sich.

## Implementierung von technischen Sicherheitsmaßnahmen in Open-Source-Projekten

Auch technische Maßnahmen, die vor Social-Engineering-Angriffen schützen können, kommen in Open-Source-Projekten bislang nur teilweise zur Anwendung. Signierte Codebeiträge finden sich auf GitHub beispielsweise in 72% der Projekte, auf denen mindestens fünf weitere Projekte aufbauen und die somit als relevanter Teil der Softwarelieferkette gelten können. Allerdings sind in diesen durchschnittlich nur 5% der Beiträge signiert, wobei zwischen 2016 und 2023 aber ein Anstieg an Signaturen zu verzeichnen ist.[32] Eine Erklärung für die niedrige Verbreitung von signierten Beiträgen könnte sein, dass Entwickler*innen das Verhältnis zwischen deren Nutzen und den mit der Umsetzung verbundenen Kosten im Vergleich zu anderen Schutzmaßnahmen eher gering einschätzen.[33]

Zusammenfassend lässt sich sagen, dass die Governance von Open-Source-Projekten zur Vermeidung von Social-Engineering-Angriffen häufig schwach ausgeprägt ist, obwohl grundsätzlich ein Bewusstsein für die Bedeutung von Sicherheitsmaßnahmen besteht. Eine wichtige Rolle beim Aufbau von Governancestrukturen spielen insbesondere in kleinen und neuen Projekten der damit verbundene Aufwand sowie Konsequenzen für die Offenheit des Projekts. Letztlich besteht die Tendenz, die Offenheit gegenüber neuen Beitragenden höher zu gewichten als den Schutz vor etwaigen Sicherheitsrisiken. Das dürfte auch damit zusammenhängen, dass viele Open-Source-Entwickler*innen noch

---

[31] Przymus & Durieux (2025, S. 99). Siehe Fußnote 7.

[32] Holtgrave, Jan-Ulrich; Friedrich, Kay; Fischer, Fabian; Huaman, Nicolas; Busch, Niklas; Klemmer, Jan H.; Fourné, Marcel; Wiese, Oliver; Wermke, Dominik; Fahl, Sascha (2025, S. 8). Attributing Open-Source Contributions is Critical but Difficult: A Systematic Analysis of GitHub Practices and Their Impact on Software Supply Chain Security, *Network and Distributed System Security (NDSS) Symposium 2025*, San Diego, CA, USA. https://dx.doi.org/10.14722/ndss.2025.240613

[33] Ladisa, Piergiorgio; Plate, Henrik; Martinez, Matias; Barais, Olivier (2023, S. 1519). SoK: Taxonomy of Attacks on Open-Source Software Supply Chains, *2023 IEEE Symposium on Security and Privacy (SP)*, S. 1509–1526. https://doi.org/10.1109/SP46215.2023.10179304

keine Erfahrung mit Angriffen auf ihre Projekte gemacht haben. Ob diese Gewichtung den tatsächlichen Risiken gerecht wird, hängt von den spezifischen Eigenschaften eines Projekts ab. Von einem erhöhten Risiko ist beispielsweise dann auszugehen, wenn ein Open-Source-Projekt eine Grundlage für viele andere Projekte bietet, also ein relevanter Teil der Software Supply Chain und somit ein attraktives Ziel für Angreifende ist.

Insofern bieten existierende Best-Practice-Leitfäden zu Sicherheit in Open-Source-Softwareprojekten zwar einen Startpunkt für den Umgang mit Social-Engineering-Angriffen; jedoch fehlen bislang spezifische Anleitungen, die eine wirksame Unterstützung für Projekte mit unterschiedlich großen Communities, in verschiedenen Entwicklungsstadien und sowohl mit geringem als auch erhöhtem Angriffsrisiko bieten. Dazu ist erforderlich, dass Leitfäden neben Best Practices auch Hilfestellungen bei der Auswahl effizienter, den Gegebenheiten eines Open-Source-Projekts entsprechender Maßnahmen leisten. Ein Beispiel für diesen Ansatz sind die Empfehlungen der amerikanischen Cybersecurity and Infrastructure Security Agency (CISA) und der OpenSSF für Repositorien, in denen Softwarepakete verwaltet werden.[34] Darin sind entlang vier verschiedener Projektreifegrade angepasste Empfehlungen formuliert.

### 3.1.3 Projektpraxis in vom Prototype Fund geförderten Projekten

Um einen Eindruck zu bekommen, wie der Umgang mit dem Risiko von Social-Engineering-Angriffen in durch den Prototype Fund geförderten Projekten ist, haben wir Interviews mit sechs Projektverantwortlichen geführt. Im Folgenden fassen wir deren Umgangsweisen mit Sicherheitsrisiken und Einflussfaktoren aus den Interviews zusammen.

**Codereviews in unterschiedlichen Verfahren.** Die übergeordneten Maßnahmen, die in allen Projekten vorgenommen werden, sind rigorose Codereviews und ein restriktiver Umgang mit der Vergabe von Rechten. In einigen Interviews kam darüber hinaus zur Sprache, dass möglichst wenige Dependencies eingebaut werden, um Vulnerabilitäten an dieser Stelle so gering wie möglich zu halten, und dass Tools zur Verwaltung dieser eingesetzt werden — was jedoch nicht vor potenzieller Schadsoftware in Dependencies schütze. Zudem werden der Social-Engineering-Angriffe und der xz-Utils-Fall zumeist im Kontext von Softwarelieferkettenangriffen thematisiert.

In einigen Projekten existieren explizite Regeln dafür, wie mit Pull Requests umgegangen wird. In einem Projekt mit einer ausgeprägteren Organisationsstruktur müssen Codebeiträge von einem Security-Team überprüft werden; in einem anderen Projekt müssen die drei Maintainer*innen die Verantwortung für Codemerger, die sie nach einer Review durchführen, persönlich übernehmen, als handele es sich um von ihnen selbst geschriebenen Code.

In den meisten Fällen waren die Verfahrensweisen, die verfolgt werden, jedoch impliziter — sei es, weil eine Person einzige*r Maintainer*in ist und sich selbst Regeln geben würde, oder weil Maintainer*innen sich auf ihre langjährige Erfahrung verlassen und daher eine externe Anleitung für nicht notwendig halten. Ein*e Maintainer*in weist darüber hinaus

[34] Cable, Jack; Steindler, Zach (02.2024). *Principles for Package Repository Security*, OpenSSF Securing Software Repositories Working Group. https://repos.openssf.org/principles-for-package-repository-security

explizit darauf hin, dass das Wissen und die Kapazitäten der Projektverantwortlichen weitaus wichtiger seien als das Festschreiben von Regeln und dass bei Maßnahmen mehr Wert darauf gelegt werden sollte, diese Kapazitäten in der breiten Masse von Entwickler*innen auszubauen, um eine nachhaltige kulturelle Veränderung zu bewirken.

**Soziale Faktoren im Vertrauensaufbau.** Wiederholt wurden soziale und menschliche Aspekte als wichtiger Faktor in der Bekämpfung von Social-Engineering-Angriffen in den Projekten hervorgehoben. Zum einen wurde die Etablierung langfristiger Beziehungen zu Beitragenden unterstrichen. Dazu gehört im Blick zu behalten, wer die Beitragenden sind, wie lange diese beitragen, aber auch persönlicher Kontakt. Persönliche Treffen auf Konferenzen und Community-Events wurden mehrfach als wichtige Möglichkeiten genannt, Vertrauen aufzubauen. Aber auch niedrigschwellige Möglichkeiten, wie Telefonate, spielen eine Rolle. Gleichzeitig wurde betont, dass auch Personen, die mit entsprechenden Ressourcen und Fähigkeiten ausgestattet sind, selbst im persönlichen Kontakt eine falsche Identität vortäuschen könnten — wenn auch mit größerem Aufwand.

**Vorsichtiger Umgang mit Call-Outs.** Einige Interviewte berichteten von Erfahrung mit Beitragenden, deren Identität sie nicht kennen, wiesen jedoch darauf hin, dass es für deren Anonymität gute Gründe geben kann — etwa, wenn Beitragende für Organisationen tätig sind, die die betreffenden Softwarebausteine selbst nutzen, dies jedoch aus Sicherheitsgründen nicht öffentlich machen wollen. Nur ein*e Maintainer*in erwähnte die Praxis auf Personen hinzuweisen, die möglicherweise Angreifer*innen sein könnten. Die Person unterstrich gleichzeitig, dass dies als eine der letzten Maßnahmen betrachtet würde und nicht auf Grundlage bloßer Verdachtsfälle geschehen sollte.

**Menschliche Fehlbarkeit und Grenzen.** In Bezug auf Codereviews und das Finden von Fehlern wurde ebenfalls mehrfach auf menschliche Aspekte hingewiesen: dass man nach bestem Wissen und Gewissen arbeite, eine große Verantwortung für die Projekte und deren Nutzende empfinde, dies jedoch nicht davor schütze, fehlbar zu sein, auch mal einen schlechten Tag haben zu können. Niemand könne garantieren, dass so nicht auch sicherheitsrelevante Schwachstellen übersehen würden. Dazu wurde unterstrichen, dass das Schreiben von Schadcode ungleich viel weniger Aufwand und Zeit erfordere als dessen Überprüfung im Rahmen eines Reviews. Nicht zuletzt seien Codereviews, auch wenn diese zentral für die Projektsicherheit sind, keine besonders attraktive oder unterhaltsame Aufgabe, was deren Notwendigkeit nicht schmälert. Gegenüberstehend zu menschlichen Kapazitäten wurden automatisiert erstellte Codebeiträge, die minderwertig seien, gleichzeitig aber die Masse an zu sichtenden Code erhöhe, mehrfach als Problem erwähnt. Die Frage nach der Machbarkeit von Reviews wird durch den Einsatz von KI-generierten Beiträgen weiter verschärft.

**Ressourcenknappheit.** Gerade im Zusammenhang mit der notwendigen Zeit und Sorgfalt für Codereviews wurde wiederholt auf generelle Ressourcenknappheit verwiesen — insbesondere auf das Fehlen von Fördermöglichkeiten für explizite Sicherheitsaufgaben und Maintenance, im Gegensatz zur Förderung für neue Features.

Zusammenfassend lässt sich für die Projekte sagen, dass Codereviews nach dem Vier-Augen-Prinzip, ein restriktiver Umgang mit der Rechtevergabe sowie die sorgfältige Integration von Dependencies eine zentrale Rolle spielen. In Bezug auf unterstützende

Maßnahmen werden insbesondere soziale, relationale und menschliche Aspekte gegenüber prozeduralen, regelbasierten und formalisierten Ansätzen hervorgehoben — etwa der Aufbau persönlicher Beziehungen in Persona, die Kapazitäten von Maintainer*innen sowie deren Fehlbarkeit. Damit diese Aspekte schützend wirken können, ist ein intaktes FOSS-Ökosystem in dem Offline-Treffen stattfinden können, von zentraler Bedeutung. Solche Treffen sowie Förderung für Maintenance und Sicherheitsmaßnahmen können jedoch nur mit entsprechender finanzieller Unterstützung realisiert werden. Diese spielt aber eine grundlegende Rolle für die Sicherheit der Projekte und des gesamten Ökosystems. Keine dieser Maßnahmen wird als Garant für vollständige Sicherheit verstanden, ein Restrisiko bleibt stets bestehen. Für eine sinnvolle Unterstützung müssen allerdings soziale und kulturelle Aspekte in der Zusammenarbeit berücksichtigt und aktiv eingebunden werden.

## 3.2 Plattformebene: Sicherheitsfokussierte Gestaltung von Softwareentwicklungs-plattformen am Beispiel von GitHub

Softwareentwicklungsplattformen wie GitHub, GitLab und Codeberg stellen die Infrastruktur dar, innerhalb derer Open-Source-Projekte entwickelt werden. Als solche ergänzen sie deren Governance und beeinflussen diese insbesondere durch ihre Moderationspraktiken, Reputationsmechanismen und technischen Sicherheitsfunktionen. Insofern sind auch Softwareentwicklungsplattformen relevant für die Abwägung zwischen Sicherheit und Offenheit in Open-Source-Projekten. Die größte Softwareentwicklungsplattform, auf der auch das Projekt xz-Utils gehostet wird, ist GitHub. Sie steht deshalb im Folgenden im Fokus.

### 3.2.1 Moderation

GitHub nimmt eine aktive Rolle bei der Moderation der Plattform ein, insbesondere in Fragen der Softwaresicherheit. Seit dem xz-Utils-Angriff hat das Unternehmen aktive Öffentlichkeitsarbeit geleistet, um den gewählten Moderationsansatz bekannt zu machen und mit Open-Source-Entwickler*innen zu diskutieren.[35] GitHub sieht es als seine Aufgabe, potenzielle Schäden aktiv zu begrenzen, indem es Moderation durch die Communities der auf GitHub gehosteten Projekte durch Funktionen der Plattform anregt und gleichzeitig auch selbst moderiert.[36] Im Fall des xz-Utils-Angriff bedeutete das, dass GitHub sowohl das Projekt als auch die Accounts aller Maintainer*innen nach Bekanntwerden der Sicherheitslücke sperrte. Der Account der Maintainer*in, der/die nicht an dem Angriff

[35]Tucker, Margaret (20.02.2025). Engaging with the developer community on our approach to content moderation, *GitHub Blog*. https://github.blog/news-insights/policy-news-and-insights/engaging-with-the-developer-community-on-our-approach-to-content-moderation/

[36]Tucker, Margaret; Coogan, Rose; Pace, Will (2024, S. 3, 7). Nuances and Challenges of Moderating a Code Collaboration Platform, *Journal of Online Trust and Safety* 2(4), S. 1-19. https://doi.org/10.54501/jots.v2i4.213

beteiligt war, wurde kurze Zeit später reaktiviert, die Person reaktivierte das Projekt. Doch auch danach nahm GitHub Änderungen im Repositorium durch das Schließen von Änderungsanfragen in Pull Requests vor, ohne transparent zu dokumentieren, dass die Schließungen nicht vom Projektmaintainer ausgingen.[37]

GitHub bezeichnet insbesondere die Deaktivierung von Projekten als äußerstes Mittel, dem gegenüber in der Regel die Moderation durch Projekt-Communities selbst oder andere Moderationsmethoden von GitHub wie das Ausblenden von individuellen Beiträgen, das Sperren von Nutzendenaccounts oder das Einschränken der Auffindbarkeit von Projekten vorzuziehen sei. Änderungen an Code oder sonstigen Inhalten innerhalb von Projekten nimmt GitHub eigenen Angaben zufolge grundsätzlich nicht vor.[38] Wie die Moderation im Fall des xz-Utils-Projekts zeigt, bedeutet das jedoch nicht, dass keinerlei Änderungen in Projekten vorgenommen werden. Für die Moderation in Projekten selbst bietet GitHub die Möglichkeit, Nutzendenaccounts von der Mitarbeit auszuschließen oder Kommentare auszublenden, zu ändern oder zu löschen.[39]

Eine wachsende Herausforderung für GitHub, deren Relevanz auch der xz-Utils-Angriff zeigte, ist der Umgang mit automatisiert generierten Inhalten. Das Unternehmen arbeitet deshalb zur Zeit am Ausbau der (teil-)automatisierten Plattformmoderation.[40] Dass die Automatisierung von Moderationsentscheidungen allerdings auch Risiken für Nutzende und ihre Projekte haben kann, zeigte sich beispielsweise, als der Account einer der GitHub-Gründer eigenen Angaben zufolge im Februar 2024 fälschlicherweise gesperrt wurde.[41]

Die Moderationsregeln anderer Softwareentwicklungsplattformen ähneln grundsätzlich denen von GitHub. Auch auf GitLab und Codeberg besteht beispielsweise die Möglichkeit, dass Plattformbetreiber*innen Beiträge sperren oder löschen, etwa wenn Nutzende sich in betrügerischer Weise als jemand anderes ausgeben[42] oder wenn Nutzendenaktivitäten den/die Plattformbetreiber*in direkt oder indirekt schädigen.[43] Zusätzlich können Projekt-Communities ihre Projekte z. B. durch den Ausschluss von Beitragenden moderieren. In welchem Ausmaß die Plattformbetreiber*innen von GitLab und Codeberg von ihren Moderationsrechten Gebrauch machen, wird nicht veröffentlicht.

---

[37]Collin, Lasse (o. D.). *XZ Utils backdoor*, Stand: 17.01.2025. https://tukaani.org/xz-backdoor/

[38]Tucker et al. (2024, 8ff.). Siehe Fußnote 35.

[39]GitHub, Inc. (o. D.). *Verwalten von Moderatoren in deiner Organisation*, Stand: 20.11.2025. https://docs.github.com/de/organizations/managing-peoples-access-to-your-organization-with-roles/managing-moderators-in-your-organization

[40]Tucker et al. (2024, S. 14f.). Siehe Fußnote 35.

[41]Goled, Shraddha (08.02.2024). When automation errror led to GitHub co-founder's account suspension, *Techcircle*. https://www.techcircle.in/2024/02/08/when-automation-error-led-to-co-founder-s-github-account-suspension

[42]GitLab (o.D.). GitLab Acceptable Use Policy, *The GitLab Handbook, Stand: 01.05.2025*. https://handbook. gitlab.com/handbook/legal/acceptable-use-policy/

[43]Codeberg (22.01.2022). *Codeberg's Terms of Use*. https://codeberg.org/Codeberg/org/src/branch/main/TermsOfUse.md

### 3.2.2 Reputationsmechanismen

Reputationsmechanismen, die Open-Source-Entwickler*innen in ihren Entscheidungen über die Zusammenarbeit mit anderen beeinflussen, werden von Softwareentwicklungsplattformen aktiv gestaltet. Hilfreich können diese Mechanismen sein, um beurteilen zu können, wie viel Erfahrung und Kompetenzen neue Beitragende zu einem Projekt mitbringen. Allerdings besteht das Risiko des sogenannten Reputation Farming, bei dem Akteure eine Reputation vortäuschen, indem sie entsprechende Indikatoren auf Entwicklungsplattformen gezielt manipulieren. Anders als im Fall des xz-Utils-Angriffs, bei dem die Akteure durch tatsächliche, kontinuierliche Mitarbeit über längere Zeit eine Reputation im Projekt aufbauten, sind die meisten Formen von Reputation Farming deutlich weniger aufwändig vorbereitet.

Auf eine Form von Reputation Farming machte beispielsweise die OpenSSF im Juni 2024 aufmerksam: Dabei tragen auffällige GitHub-Nutzendenprofile zu bereits gelösten Issues und Pull Requests in hochrangigen Projekten bei, um ihre Mitarbeit an diesen zu simulieren.[44] Auf ähnliche Weise können automatisiert erstellte Beiträge schnell und ohne größeren Ressourcenaufwand den Eindruck aktiver, produktiver Mitarbeit an einem Projekt erwecken. Eine andere Möglichkeit ist es, eigene Beiträge anderen Accounts mit hoher Reputation zuzuordnen oder Beiträge von Accounts mit hoher Reputation dem eigenen Account bzw. einem beliebigen Projekt zuzuordnen.[45] Zu Profilen, denen viele GitHub-Nutzende großes Vertrauen entgegenbringen und deren Beiträge sie deshalb nicht näher überprüfen, gehört Dependabot, ein Tool zur automatisierten Verwaltung von Softwareabhängigkeiten. 2023 wurde ein Fall bekannt, in dem versucht wurde, u. a. durch die Maskierung als Dependabot unbemerkt schädlichen Code in hunderte Projekte einzuschleusen.[46] Ein weiteres Einfallstor für Reputation Farming sind Markierungen von Projekten mit Sternen oder das Folgen von Accounts, um die Mitarbeit an erfolgreichen Projekten bzw. ein hohes Ansehen unter anderen Entwickler*innen vorzutäuschen. Insbesondere bei Sternmarkierungen konnte seit 2022 ein signifikanter Anstieg an mutmaßlichen Fälschungen — Sterne, die beispielsweise gekauft und durch Fake Accounts vergeben wurden — beobachtet werden.[47]

Softwareentwicklungsplattformen bieten bereits eine Reihe an Funktionen an, die einen gewissen Schutz vor Reputation Farming bieten. Bei GitHub gehört dazu neben der Unterstützung von Signaturen beispielsweise die Möglichkeit, weitere Beiträge zu gelösten Issues und Pull Requests zu unterbinden. Auch durch ihre eigenen Moderationspraktiken

---

[44] Gooding, Sarah (26.062024). OpenSSF Warns of Reputation Farming Leveraging Closed GitHub Issues and PRs, *Socket Blog*. https://socket.dev/blog/openssf-warns-of-reputation-farming-using-closed-github-issues-and-prs

[45] Tucker et al. (2024, 8ff.). Siehe Fußnote 35.

[46] GitHub, Inc. (o. D.). *Verwalten von Moderatoren in deiner Organisation*. Stand: 01.06.2025. https://docs.github.com/ de/organizations/managing-peoples-access-to-your-organization-with-roles/managing-moderators-in-your-organization

[47] He, Hao; Yang, Haoqin; Burckhardt, Philipp; Kapravelos, Alexandros; Vasilescu, Bogdan; Kästner, Christian (2024). *4.5 Million (Suspected) Fake Stars in GitHub: A Growing Spiral of Popularity Contests, Scams, and Malware*, Stand: 18.12.2024. https://doi.org/10.48550/arXiv.2412.13459

gehen Entwicklungsplattformen gegen Reputation Farming vor. Indem GitHub in seinen Nutzungsbestimmungen Aktivitäten wie das Fälschen von Nutzendenkonten, die automatisierte Vergabe von Sternen oder das automatisierte Folgen von Nutzenden sowie automatisierte, nicht-authentische Beiträge ausschließt, behält sich das Unternehmen beispielsweise vor, entsprechende Accounts und Beiträge von der Plattform zu entfernen.[48] Allerdings könnten diese Maßnahmen durch weitere ergänzt werden: Die durch Plattformen wie GitHub angebotenen Reputationssignale wie Sterne oder Follower könnten durch komplexere, weniger leicht manipulierbare Signale ersetzt werden. Ein Vorschlag dafür sind beispielsweise sogenannte Contributor Reputation Badges, die Faktoren wie Verbindungen zu anderen zentralen Entwickler*innen[49] oder Beitragsmuster im Zeitverlauf[50] berücksichtigen. Durch eine Umgestaltung des User Interface – etwa indem die Verlässlichkeit der Zuordnungen von Beiträgen zu bestimmten Accounts oder automatisiert erstellte Beiträge transparenter angezeigt werden — könnten Nutzende dabei unterstützt werden, verdächtige Aktivitäten zu erkennen.[51] Um umfassende Erkenntnisse über die Verbreitung der oben beschriebenen Varianten von Reputation Farming, deren Nutzung für Social-Engineering-Angriffe und die Gegenreaktionen von Softwareentwicklungsplattformen zu gewinnen, ist außerdem erforderlich, dass Informationen über gelöschte Repositorien und Nutzende zur Verfügung gestellt werden.

Insgesamt gewichten Softwareentwicklungsplattformen wie GitHub Sicherheit hoch und unterstützen Open-Source-Entwickler*innen dabei, Best Practices wie Codesignaturen und Rollenverteilungen technisch umzusetzen sowie Vertrauen zu anderen Entwickler*innen durch Reputationsmechanismen aufzubauen. Wie beschrieben schränken diese Maßnahmen die Offenheit von Projekten nicht zwingend ein, auch wenn die angebotenen Reputationsmechanismen effektiver gestaltet werden könnten. Potenziell einschränkend könnten dagegen die Moderationspraktiken von Softwareentwicklungsplattformen wirken, die die Vermeidung von Sicherheitsrisiken priorisieren und die Entscheidungen innerhalb von Entwickler*innen-Communities überlagern können. Das ist insbesondere dann der Fall, wenn Moderationsentscheidungen automatisiert getroffen werden. Für eine Beurteilung, in welchem Ausmaß es in der Praxis zu solchen Einschränkungen kommt und inwieweit dabei Aspekte wie das Sicherheitsrisiko eines Projekts berücksichtigt werden, ist es erforderlich, das Softwareentwicklungsplattformen transparent über ihre Moderationspraxis informieren.

---

[48] GitHub (o.D). *GitHub-Nutzungsbedingungen*, Stand: 16.11.2020. https://docs.github.com/de/ site-policy/github-terms/github-terms-of-service

[49] Hamer, Sivana; Imtiaz, Nasif; Tamanna, Mahzabin; Shabrina, Preya; Williams, Laurie (2025). *Trusting code in the wild*: Exploring contributor reputation measures to review dependencies in the Rust ecosystem, *IEEE Transactions on Software Engineering* 51(4), S. 1319-1333. https://doi.org/10.1109/TSE.2025.3551664

[50] Przymus und Durieux (2025, S. 99). Siehe Fußnote 7.

[51] Holtgrave et al. (2025, S. 13). Siehe Fußnote 31.

### 3.2.3 Bedürfnisse in durch den Prototype Fund geförderten Projekten

Sowohl in den Interviews, als auch im Workshop wird kaum auf die Plattformgestaltung und die Rolle von Plattformen in der Verhinderung von Social-Engineering-Angriffen eingegangen.

***Spezifische technische Funktionen.*** Einzelne Projektverantwortliche wünschen sich jedoch explizite Funktionen, die Prozesse auf Plattformen betreffen; diese betreffen hauptsächlich das Codemergen und Testläufe.

***Auffindbarkeit von Tools.*** In diesem Zusammenhang wird mehrfach auch angemerkt, dass man gar nicht wisse, ob es diese Funktionen nicht doch schon gebe; aber falls dem so sei, wisse man davon nicht. An dieser Stelle sowie in Bezug auf Leitlinien wird mehrfach darauf verwiesen, dass diese bekannt und auffindbar sein müssten, um Wirkung zu entfalten.

***Etablierung von Sicherheitspraktiken.*** Darüber hinaus müsste die Nutzung unterschiedlicher Tools, wie beispielsweise das Signieren von Codebeiträgen, weiter verbreitet sein, damit Softwarelieferketten sicherer werden; auch hier verweisen die Befragten auf kulturelle Aspekte in der Community.

## 3.3 Policyebene: Regulierung, Aufbau von Kapazitäten und Förderung

Nicht nur die Verbreitung von Open-Source-Software, auch das Bewusstsein für die gesellschaftliche Bedeutung ihrer Sicherheit hat in den vergangenen Jahren zugenommen. Das zeigt sich u. a. darin, dass Regulierung und Förderung in diesem Bereich — wenn auch langsam und in begrenztem Umfang — vorangetrieben werden. Insbesondere IT-Sicherheitsregulierung besteht in der Europäischen Union heute aus einem wachsenden Geflecht aus Gesetzen, Standards und Normen. Nachdem Open-Source-Software darin lange Zeit wenig Beachtung fand, ändert sich das zunehmend. Im Folgenden werden Bestrebungen zur Regulierung und Förderung dargelegt und ausgelotet, wie diese dazu beitragen, Social-Engineering-Angriffe in Open-Source-Softwareprojekten zu verhindern und inwieweit dabei die Offenheit von Governance-Strukturen in Open-Source-Projekten beeinflusst wird.

### 3.3.1 Gesetzgebung – Cyber Resilience Act

Einen Wendepunkt in der Regulierung von Open-Source-Software stellte der 2024 verabschiedete Cyber Resilience Act (CRA)[52] dar, der Sicherheitsanforderungen für kommerziell in der EU vertriebene Software- und Hardwareprodukte festlegt. Nachdem

[52]Cyber Resilience Act (2024). Verordnung (EU) 2024/2847 des Europäischen Parlaments und des Rates vom 23. Oktober 2024 über horizontale Cybersicherheitsanforderungen für Produkte mit digitalen Elementen, ABl. L 2024/2847 vom 20.11.2024. http://data.europa.eu/eli/reg/2024/2847/oj

zunächst befürchtet wurde, dass Open-Source-Entwicklung umfassend von den Regelungen des CRA betroffen sein würde,[53] bleiben Entwickler*innen der meisten Open-Source-Infrastrukturkomponenten von dem Gesetz zunächst unberührt. Nur die Hersteller*innen und — in abgeschwächter Form — sogenannte Verwalter*innen kommerziell vertriebener Software sind zur Einhaltung des CRA verpflichtet. Allerdings ist von einem indirekten Effekt des CRA auf Module auszugehen, die in kommerziellen Open-Source-Softwareprojekten zum Einsatz kommen; denn deren Hersteller*innen sind in Zukunft für alle Komponenten, die in ihren Software- und Hardwareprodukten zum Einsatz kommen, verantwortlich. Das bedeutet, dass sie bei der Auswahl von Open-Source-Softwaremodulen voraussichtlich höhere Ansprüche an deren Sicherheitsmaßnahmen wie die oben beschriebenen Governance-Strukturen stellen werden.

Diese Verantwortungszuweisung könnte einerseits dazu führen, dass Unternehmen, die von Open-Source-Softwarekomponenten profitieren, mehr in deren Sicherheit investieren. Andererseits besteht die Möglichkeit, dass in Projekten restriktivere Governance-Strukturen eingefordert werden, um Ressourcen zu sparen, die für offene Kollaborationsprozesse erforderlich sind. Zudem besteht das Risiko, dass gerade kleinere, neue Open-Source-Projekte den neuen Anforderungen, die der CRA indirekt an sie stellt, nicht gerecht werden können. Das hätte zur Folge, dass sie sich deutlich schwerer etablieren und mit größeren Projekten um Ressourcen konkurrieren können, die für den Aufbau einer offenen und sicheren Projekt-Governance notwendig sind. Insgesamt erscheint der CRA als potenziell zweischneidiges Schwert, dass dem FOSS-Ökosystem sowohl schaden als auch nutzen könnte.

### 3.3.2 Aufbau von Kapazitäten und ideelle Förderung

Flankiert werden die Verpflichtungen des CRA sowohl durch die Erarbeitung von Standards als auch durch Handlungsempfehlungen und Vorschriften zur Umsetzung. Wichtige Akteure sind in diesem Zusammenhang die Agentur der Europäischen Union für Cybersicherheit (ENISA) sowie das BSI, deren Aufgabe es ist, IT-Sicherheit auf europäischer Ebene beziehungsweise in Deutschland zu fördern. Unterstützung von Entwicker*innen bei der Verbesserung von Sicherheitspraktiken speziell für Open-Source-Softwareprojekte — insbesondere solche, die Infrastrukturkomponenten entwickeln — haben diese staatlichen Organisationen bisher jedoch weder in Bezug auf den CRA noch allgemein vertieft in den Blick genommen. Es überwiegt die Perspektive der Softwarenutzenden wie das BSI-Projekt *Codeanalyse von Open-Source-Software (CAOS)* zeigt. In dessen Rahmen werden Schwachstellen in Anwendungen identifiziert und gemeldet, die vermehrt in Behörden und darüber hinaus zum Einsatz kommen.[54] Es wird hier weniger die Perspektive derer, die diese Anwendungen entwickeln und verwalten, eingenommen.

---

[53] Eclipse Foundation (17.04.2023). *Open Letter to the European Commission on the Cybr Resilience Act*. https://newsroom.eclipse.org/news/announcements/open-letter-european-commission-cyber-resilience-act

[54] Bundesamt für Sicherheit in der Informationstechnik (o.D.). *Codeanalyse von Open Source Software (Projekt CAOS)*. https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Studien/Projekt_P486/projekt_P486_node.html

In den USA, wo Soft Law wie Leitlinien und Standards etablierte Instrumente für IT-Sicherheitsregulierung sind, ist das anders.[55] Die Cybersecurity and Infrastructure Security Agency (CISA), das amerikanische Äquivalent zur ENISA, hat 2023 eine Roadmap für Open-Source-Softwaresicherheit verabschiedet und erarbeitet in Kooperation mit Open-Source-Communities und -Organisationen Best Practices.[56]

Ein wichtiger Partner für die CISA ist dabei die in erster Linie durch Unternehmen finanzierte Linux Foundation beziehungsweise deren OpenSSF.[57] Auch für die Umsetzung des CRA arbeiten die Linux Foundation und die OpenSSF seit Januar 2025 im Rahmen einer Initiative an Unterstützungsmaßnahmen für Entwickler*innen von Open-Source-Software wie Standards, Best Practices und Schulungsmaterialien.[58] Weitere Maßnahmen, um die Sicherheit von Open-Source-Software zu verbessern, entwickelt die OpenSSF im Rahmen ihres Projekts Alpha Omega. Dazu zählte in der Vergangenheit beispielsweise ein Mentorship-Programm für Nachwuchsentwickler*innen und Sicherheitsforschende[59] oder Codeanalysen mit dem Ziel, Schwachstellen zu identifizieren und zu melden. Die Identifikation von Schwachstellen wurde von Entwickler*innen jedoch gemischt aufgenommen, weil in den betroffenen Projekten häufig keine ausreichenden personellen Kapazitäten zur Verfügung standen, um die gemeldeten Schwachstellen zu beheben. Aus diesem Grund priorisierte Alpha Omega im Anschluss Maßnahmen, die insbesondere kleine Teams konkret bei der Betreuung ihrer Projekte unterstützen könnten, wie z. B. technische Mittel zur Automatisierung von Routineaufgaben.[60] Auf diese Weise könnten sie eine breitere Umsetzung der in Abschnitt 3.1 beschriebenen Best Practices ermöglichen.

Den Aufbau von Kapazitäten für die Etablierung von Sicherheitsstandards in Open-Source-Projekten unterstützen staatliche Akteure auf EU-Ebene und in Deutschland bisher kaum. Insofern nehmen sie keinen Einfluss auf die Offenheit von Governance-Strukturen in

---

[55]Fahey, Elaine (2024). The evolution of EU-US cybersecurity law and policy: on drivers of convergence, *Journal of European Integration* 46(7), S. 1073-1088. https://doi.org/10.1080/07036337.2024.2411240

[56]Cybersecurity and Infrastructure Security Agency (2023). *CISA Open Source Software Security Roadmap*. https://www.cisa.gov/sites/default/files/2024-02/CISA-Open-Source-Software-Security-Roadmap-508c.pdf

[57]OpenSSF (07.03.2024). OpenSSF and CISA Join Forces to Secure Open Source Software, *OpenSSF Blog*. https://openssf.org/blog/2024/03/07/openssf-and-cisa-join-forces-to-secure-open-source-software/

[58]OpenSSF (31.01.2025). Linux Foundation Europe and OpenSSF Launch Initiative to Prepare Maintainers, Manufacturers, and Open Source Stewards for Global Cybersecurity Legislation, *OpenSSF Blog*. https://openssf.org/press-release/2025/01/31/linux-foundation-europe-and-openssf-launch-initiative-to-prepare-maintainers-manufacturers-and-open-source-stewards-for-global-cybersecurity-legislation/

[59]Blume, Aaron; Orbe, Andres; Garcia, Glenda; Navani, Saumya (o.D.). Experiences from the Alpha-Omega Mentorship Program Mentees, *Alpha-Omega Blog,* Stand: 01.06.2025. https://alpha-omega.dev/blog/experiences-from-the-alpha-omega-mentorship-program-mentees/

[60]OpenSSF (2025, S. 18). *Alpha-Omega Annual Report 2024*. https://alpha-omega.dev/wp-content/uploads/sites/22/2025/01/Alpha-Omega-Annual-Report-2024_012925.pdf

Open-Source-Projekten. Als relevanter staatlicher Akteur tritt dagegen die US-amerikanische CISA auf. Sie arbeitet intensiv mit der Linux Foundation zusammen, eine nicht-staatliche Organisation, die die Perspektive von Open-Source-Entwickler*innen vertreten und selbst Sicherheitsstandards mit Blick auf offene Entwicklungsprozesse vorantreiben.

### 3.3.3 Finanzielle Förderung von Softwareinfrastruktur

Um sicherheitsrelevante Governance-Mechanismen und sonstige Best Practices so umzusetzen, dass der für Open-Source-Software charakteristische offene und kollaborative Entwicklungsprozess erhalten bleibt, ist eine gezielte Finanzierung unabdingbar. Auch in diesem Bereich sind ein langsam wachsendes Problembewusstsein und Interesse sowohl bei staatlichen Akteur*innen als auch bei Unternehmen zu beobachten. Das zeigt sich in neueren Fördermaßnahmen, wie beispielsweise der Sovereign Tech Agency (STA), die 2022 als Sovereign Tech Fund ins Leben gerufen wurde und durch das Bundesministerium für Wirtschaft und Energie finanziert wird. Ein weiteres Förderbeispiel, das durch Unternehmen gefördert wird, ist das Projekt Alpha Omega, das ebenfalls 2022 startete und von Unternehmen wie Amazon Web Services, Google und Microsoft getragen wird.

Um diese Ziele zu erreichen, wählen STA und Alpha Omega strategisch Projekte aus, die als besonders sicherheitsrelevant erachtet werden. Entscheidend dafür ist etwa, dass die Software, um deren Entwicklung es geht, in besonders vielen oder wichtigen anderen Softwareprojekten zum Einsatz kommt und dass die finanzierten Entwicklungstätigkeiten zu mehr Sicherheit beitragen. Unterstützt werden deshalb, anders als bei vielen anderen Fördermaßnahmen, nicht nur die Entwicklung neuer Features, sondern auch Tätigkeiten wie Codeüberarbeitung, Tests und Fehlerbehebung oder Dokumentation und Bildung.[61] Empfänger*innen der Förderung sind meist Vereine oder Stiftungen wie die OpenJS Foundation oder die Eclipse Foundation, die die Entwicklung großer Open-Source-Projekte unterstützen.

Auch fördern STA und Alpha Omega regelmäßig Best Practices zur Vermeidung von Social-Engineering. Unter den Förderprojekten der STA fallen insbesondere Projekte auf, die Open-Source-Communities durch technische Mittel dazu befähigen sollen, die Vertrauenswürdigkeit von Beiträgen zu überprüfen. Beispielsweise wurden mit Unterstützung der STA die Projekte Sequoia PGP[62] und Sigstore[63] weiterentwickelt, mit denen Entwickler*innen Codebeiträge und Softwarepakete signieren können. Den Umgang mit Schwachstellen durch automatisierte Erkennung und Managementsysteme zu unterstützen, steht im Fokus einer Förderung der Eclipse Foundation.[64] Ein Schwerpunkt

[61] Ruohonen, Jukka; Choudhary, Gaurav; Alami, Adam (2025). *An Overview of Cyber Security Funding for Open Source Software*, Stand: 29.04.2025. https://doi.org/10.48550/ arXiv.2412.05887

[62] Sovereign Tech Agency (o. D.). *Sequoia PGP (2022)*, Stand: 22.07.2025. https://www.sovereign.tech/tech/sequoia-pgp-2022

[63] Sovereign Tech Agency (o. D.). *Python Package Index*, Stand: 22.07.2025. https://www.sovereign.tech/tech/python-package-index

[64] Sovereign Tech Agency (o. D.). *Eclipse Foundation*, Stand: 22.07.2025. https://www.sovereign.

von Alpha Omega liegt dagegen in der Finanzierung von Personen, deren Aufgabe es ist, die Sicherheitskultur in ihren Projekten zu stärken, indem sie bestehende Sicherheitspraktiken in konkreten Open-Source-Projekten formalisieren, neue Prozesse etablieren und Projekt-Communities im Umgang mit Sicherheitsrisiken schulen. Im Jahr 2024 entfielen 63% der Fördergelder von Alpha Omega auf sogenanntes Sicherheitspersonal, z. B. in Projekten der Python Software Foundation. Diesen Einsatz von Fördergeldern hat Alpha Omega als den effizientesten identifiziert.[65]

Der Förderansatz der STA und von Alpha Omega trägt dazu bei, Sicherheit und Offenheit miteinander in Einklang zu bringen. In Übereinstimmung mit dem Ziel, kritische Infrastruktur zu finanzieren, werden in erster Linie große, etablierte Projekte finanziert, deren Entwicklung beispielsweise von einem Verein oder einer Stiftung unterstützt wird. Wie der xz-Utils-Angriff zeigt, können allerdings auch auf weniger etablierte Open-Source-Projekte mit kleinen Projekt-Communities so viele andere Softwareprojekte aufbauen, dass Angriffe auf ihre Sicherheit weitreichende Folgen haben. Ein echter Wandel hin zu mehr Sicherheit und Offenheit würde eine massive Steigerung der Finanzierung für Open-Source-Infrastrukturkomponenten voraussetzen. Gefordert werden deshalb neue Fördermaßnahmen mit ähnlicher Ausrichtung beispielsweise in Großbritannien[66] und auf europäischer Ebene.[67] Auch der Prototype Fund fördert seit 2024 ausschließlich Projekte mit Schwerpunkt auf Datensicherheit und Softwareinfrastruktur und geht damit, als bereits existierende Fördermaßnahme in eine ähnliche Richtung.

### 3.3.4 Übergeordnete Kommentare zu Herausforderungen aus Workshop und Interviews

Um übergeordnete Maßnahmen weiter in den Kontext des Prototype Fund einordnen zu können, fassen wir im Folgenden ein Stimmungsbild von den unterschiedlichen Akteuer*innen im Feld zusammen, das sich auf das FOSS-Ökosystem bezieht — von Maintainer*innen aus den Interviews und von Personen, die am Workshop teilgenommen haben.

***Unterschiedliche Betroffenengruppen entlang der Softwarelieferkette erfordern unterschiedliche Ansprachen und Maßnahmen.*** Aufgrund der heterogenen Teilnehmenden-Gruppe wird im Workshop insbesondere deutlich, wie viele Akteur*innen auf verschiedenen Ebenen die Thematik betrifft. Entlang der Softwarelieferkette

---

tech/tech/eclipse-foundation

[65]Alpha Omega (2025). *Alpha Omega 2024 Annual Report*. https://alpha-omega.dev/wp-content/uploads/sites/22/2025/01/Alpha-Omega-Annual-Report-2024_012925.pdf

[66]Milton, Tom; Osborne, Cailean; Pickering, Matt (17.04.2024). *A UK Open-Source Fund to Support Software Innovation and Maintenance*, Centre for British Progress. https://britishprogress.org/uk-day-one/a-uk-open-source-fund-to-support-software-innovati

[67]Gates, Nicholas; Tridgell, Jennifer; Torraco, Rosa Maria; Schwäbe, Carste; Reda, Felix; Hummler, Andreas; Streinz, Thomas; Nummelin Carlberg, Astor; Blind, Knut (2025). Funding Europe‘s Digital Infrastrucuture: A Study on the Economic, Legal, and Political Feasibility of an EU Sovereign Tech Fund (EU-STF), Open Forum Europe (Hrsg.). https://eu-stf.openforumeurope.org/wp-content/uploads/2025/08/EU-STF-Feasibility-Study_final.pdf

verändern sich die Betroffenen der Angriffe: von Maintainer*innen, die die Softwarebausteine verantworten, über Expert*innen-Nutzer*innen, die Softwarebausteine verbauen, bis hin zu Laien-Nutzer*innen, die die Software, in der die entsprechenden Bausteine verbaut sind, nutzen. Von Teilnehmenden wird unterstrichen, dass die Maßnahmen zur Verhinderung von Angriffen an die unterschiedlichen Betroffenengruppen angepasst werden müssten. Dabei sollten bestehende Unterschiede in Bezug auf Expertise und Verbindung zur Software entsprechend berücksichtigt werden.

***Auswirkungen des Balanceakts zwischen übermäßiger und angemessener Vorsicht für das FOSS-Ökosystem.*** Im Workshop kam immer wieder auf, dass möglicherweise betroffene Teilnehmende sich fortwährend die Frage stellen, ob sie kompromittierte Softwarebausteine nutzten, ohne dies zu bemerken. Damit ging auch die Frage einher, wo der Kipppunkt zwischen übermäßiger Vorsicht und angemessener Skepsis liege und wie dies die Zusammenarbeit beeinflusse. Auch in den Interviews wurde darauf hingewiesen, dass man versuche, Beitragenden nicht zu viel Skepsis entgegen zu bringen und darüber hinaus Reviews schnell durchführen wolle, um den sonst daraus resultierenden Frust zu vermeiden. Ein*e weitere*r Projektverantwortliche*r verweist im Interview darauf, dass mit Social-Engineering-Angriffen der Kern der gemeinsamen Arbeit an Projekten angriffen werde und zitiert den Debian-Developer Russ Allbery:„The hardest part about defending against social engineering is that it doesn't attack attack (sic) the weakness of a community.  It attacks its *strengths*: trust, collaboration, and mutual assistance.“[68]

Skepsis und Vertrauen stehen sich gegenüber und müssen immer wieder neu in Balance gebracht werden, während eine absolute Sicherheit unerreichbar ist. Auch wenn das Verantwortungsgefühl von Projektverantwortlichen immer wieder unterstrichen wird, ist nicht zuletzt in der Freiwilligenarbeit an FOSS-Projekten eine Atmosphäre, in der man gerne zusammenarbeitet, ein wichtiger Aspekt, der in der Entwicklung von Maßnahmen nicht übersehen werden sollte.

***Strukturelle Aspekte erschweren die Abwehr von Social-Engineering Angriffen.*** Sowohl im Workshop, als auch in den Interviews werden unterschiedliche strukturelle Aspekte erwähnt, die die Abwehr von Angriffen erschweren. Dazu gehört, dass es deutlich länger dauere Code zu überprüfen und Fehler oder Schadcode zu finden, als es dauere, diesen Code zu schreiben. Darüber hinaus sind vor allem staatlich finanzierte Angreifer*innen übermäßig gut mit Ressourcen ausgestattet. Andere Aspekte betreffen das FOSS-Ökosystem. Software würde immer komplexer und schwieriger nachzuvollziehen, aufgrund von vielen Dependencies, aber beispielsweise auch durch nicht nachvollziehbare Outputs von Algorithmen. Diese Komplexität erschwere auch das Überprüfen der Software zu Sicherheitszwecken. Außerdem würden Übersichtlichkeit, eine verlässliche Normierung von Software und verlässliche Institutionen fehlen. Diese Unsicherheit und immer weiter zunehmende Komplexität in der Infrastruktur potenziere sich so.

***Notwendigkeit von politischem und wirtschaftlichem Willen, Ressourcen aufzubauen.*** Der politische sowie ökonomische Wille, dem Risiko etwas entgegenzusetzen, wird als unzureichend bewertet. Dieser stehe weder in Relation zum bestehenden Risiko, noch zum

[68]Russ Allbery (o.D.). *Re: Validating tarballs against git repositoties*, List Debian, Stand: 19.11.2025. https://lists.debian.org/debian-devel/2024/03/msg00377.html

Nutzen von FOSS für Unternehmen und Gesellschaft. Mehrfach wurde unterstrichen, dass Unternehmen und andere Organisationen die Gefährdung durch Lieferkettenangriffe als deutlich zu gering einschätzten. Erst konkrete Vorfälle würden dazu führen, dass Maßnahmen ergriffen werden, außerdem würden keine ausreichenden Investitionen getätigt. Sowohl Unternehmen, die FOSS nutzen, als auch öffentliche Stellen sollten daher deutlich höhere Rückführungen von Ressourcen in FOSS als Infrastruktur tätigen. Entsprechend der vielen betroffenen Ebenen, seien wesentlich mehr und insbesondere koordinierte Aktionen notwendig, um bestehende Maßnahmen aufeinander abzustimmen und entsprechende Lücken zu schließen. Das gelte nicht zuletzt, weil sich die Grenzen von ehrenamtlicher Arbeit und intrinsischer Motivation zeigen, wenn es um die langfristige Sicherung kritischer Softwareinfrastruktur geht, und auch, um nachhaltige Governance und institutionelle Unterstützung sicherstellen zu können.

# 4 Fazit

Social-Engineering-Angriffe und in Konsequenz Angriffe auf Softwarelieferketten finden in einem komplexen sozialen Ökosystemen statt. Wir haben in diesem Bericht anhand des xz-Utils-Fall in das Thema eingeführt und Maßnahmen, die sich auf unterschiedlichen Ebenen des Ökosystems abspielen, beleuchtet — auf der Projektebene, in Bezug auf die Gestaltung von Plattformen zur kollaborativen Entwicklung von Software, so wie Regulierung und Förderung. Aus den Interviews mit Projektverantwortlichen und dem Workshop mit Communitymitgliedern wird weiter deutlich: Open-Source-Projekte sind soziale, vernetzte, aufeinander aufbauende, inhärent kooperative Projekte. Maßnahmen, die dies nicht beachten, gefährden unter Umständen wichtige Funktionen im FOSS-Ökosystem und versprechen weniger Erfolg, als kulturell sensitive Ansätze die das gesamte Ökosystem berücksichtigen. Ein gesundes FOSS-Ökosystem ermöglicht resilientere Projekte. Die Interviews verweisen beispielsweise darauf, dass persönliche Treffen auf Events Vertrauen schaffen. Gleichzeitig wird darauf hingewiesen, dass etablierte Institutionen fehlen, die strukturelle Probleme, wie beispielsweise fehlende Normierungen und die Koordination von Maßnahmen gegen Lieferkettenangriffe auf unterschiedlichen Ebenen koordinieren können. Zuletzt geht auch aus der Interviews hervor, dass Ressourcenbedarf für die Förderung zu Sicherheitsmaßnahmen besteht.

Daran anschließend stellt sich die Frage: Hängt am Ende doch alles von der Finanzierung ab? Theoretisch ja, und wie beschrieben gibt es Beispiele, die zeigen, dass es in dieser Hinsicht zu einem Umdenken sowohl bei staatlichen und philanthropischen als auch bei privatwirtschaftlichen Akteur*innen kommt. Allerdings ist fraglich, ob angesichts der neuen Herausforderungen durch automatisiert erstellte Open-Source-Beiträge sowie politisch und wirtschaftlich motivierten Angreifenden eine Finanzierung bereitgestellt werden kann, die ausreicht, um Offenheit und Sicherheit in der Breite zu ermöglichen. In Zukunft wird deshalb auszuhandeln sein, welche Einschränkungen offener Entwicklungspraktiken innerhalb von Open-Source-Softwareprojekten akzeptabel sind, um Sicherheit zu gewährleisten, ohne die verfügbaren personellen Ressourcen in der Projekt-Community zu überbeanspruchen. Eine wichtige Frage ist in diesem Zusammenhang, unter welchen Umständen es vertretbar ist, Beitragende aus Sicherheitserwägungen von der Mitarbeit

an einem Projekt auszuschließen. Insbesondere das Verhältnis zwischen den Moderationspraktiken von Projekt-Communities und Softwareentwicklungsplattformen gilt es dabei zu bewerten.

Zur Zeit werden mit Blick auf Sicherheit in erster Linie etablierte Open-Source-Projekte finanziert, die für besonders kritische Bestandteile der Softwarelieferkette erachtet werden. Zielgerichtete Unterstützung für den Ausbau von Sicherheitsmaßnahmen fehlt insbesondere für kleinere und neuere Projekte. Sie könnten vom Ausbau technischer Sicherheitsmaßnahmen wie z. B. leicht benutzbarer Software zum Signieren von Code profitieren. Neben der Entwicklung dieser Tools, ist es gleichermaßen wichtig, dass es gefördert wird, dass sich diese etablieren. Eine weitere Unterstützung könnten Best-Practice-Leitfäden und Standards darstellen, die dem Entwicklungsstand eines Projekts angepasst sind und sich auch mit geringen personellen Ressourcen umsetzen lassen. Die besonderen Herausforderungen von Open-Source-Entwicklung in Bezug auf Sicherheit, wie beispielsweise durch Social-Engineering-Angriffe, nehmen bisher jedoch nur wenige Organisationen in den Blick. Damit auch neue Ansätze für eine offene und sichere Softwareinfrastruktur eine Chance haben Verbreitung zu finden, muss ein Gleichgewicht gefunden werden, zwischen der Unterstützung etablierter Projekte einerseits und kleinerer, unbekannterer Projekte sowie dem FOSS-Ökosystem andererseits.

## Prototype Fund

Open Knowledge Foundation Deutschland e. V.
Singerstr. 109
10179 Berlin

info@prototypefund.de
www.prototypefund.de

Der Prototype Fund ist das erste niedrigschwellige Förderprogramm für freie Entwickler*innen, die in Deutschland innovative Open-Source-Software aus der Gesellschaft und für die Gesellschaft entwickeln. Die auch als „Software Sprint“ bekannte Maßnahme existiert seit 2017. Das Förderprogramm wurde neu aufgelegt und wird bis 2029 in vier Jahrgängen ungefähr 100 Softwareprojekte auf ihrem Weg von der Idee zum Softwareprototypen begleiten.

Nutzer*innen sind nicht nur Konsument*innen von Software sondern oft auch Expert*innen: Diese Schnittmenge nutzen wir und bieten der technisch versierten Zivilgesellschaft Zugang zu den Ressourcen und Prozessen, die nötig sind, um sich im Sinne des Gemeinwohls einzubringen. Gefördert werden ausschließlich Softwareprojekte mit gesellschaftlichem Mehrwert, die die Bedürfnisse der Nutzer*innen in den Mittelpunkt stellen und als Open-Source-Software frei verfügbar, nachhaltig zugänglich sowie anpassbar sind. Der Schwerpunkt der Förderung liegt auf Datensicherheit und Software-Infrastruktur.

## Ein Projekt der:

OKF OPEN KNOWLEDGE FOUNDATION DEUTSCHLAND

## Gefördert durch:

Bundesministerium für Forschung, Technologie und Raumfahrt

Förderkennzeichen: 16IS24086

## Danksagung

Wir bedanken uns bei unseren Interviewpartner*innen und den Teilnehmenden des Workshops für die Einblicke und ihre Großzügigkeit mit ihrer Zeit.

Januar 2026