Decomposing and Measuring Trust in Open-Source Software Supply Chains

Lina Boughton
The College of Wooster

Courtney Miller
Carnegie Mellon University

Yasemin Acar
University of Paderborn

Dominik Wermke
North Carolina State University

Christian Kästner
Carnegie Mellon University

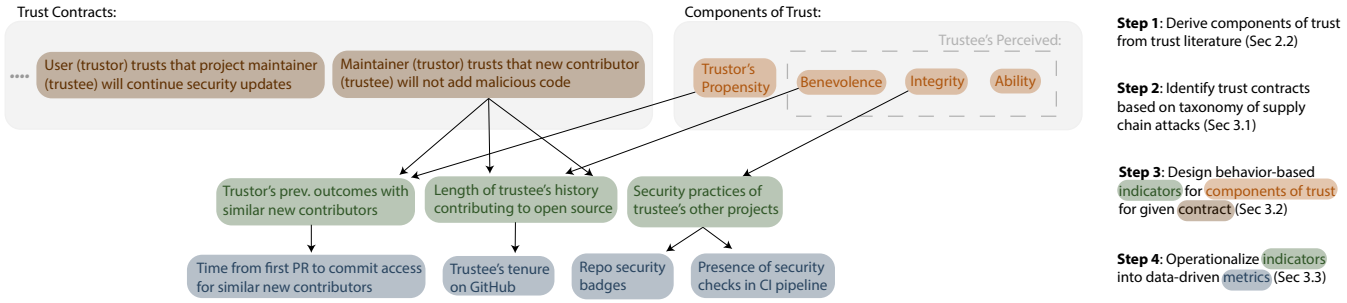

Figure 1: Example illustrating how metrics can be derived from components of trust for a given contract

ABSTRACT

Trust is integral for the successful and secure functioning of software supply chains, making it important to measure the state and evolution of trust in open source communities. However, existing security and supply chain research often studies the concept of trust without a clear definition and relies on obvious and easily available signals like GitHub stars without deeper grounding. In this paper, we explore how to measure trust in open source supply chains with the goal of developing robust measures for trust based on the behaviors of developers in the community. To this end, we contribute a process for decomposing trust in a complex large-scale system into key trust relationships, systematically identifying behavior-based indicators for the components of trust for a given relationship, and in turn operationalizing data-driven metrics for those indicators, allowing for the wide-scale measurement of trust in practice.

ACM Reference Format:
Lina Boughton, Courtney Miller, Yasemin Acar, Dominik Wermke, and Christian Kästner. 2024. Decomposing and Measuring Trust in Open-Source Software Supply Chains. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Digital innovation has been rapidly accelerated by the increased use of open source software which serves as digital infrastructure [5, 7]. Software engineering teams can reap the benefits by building software out of layers of existing reusable components such as libraries, frameworks, and cloud infrastructure. The use of these digital infrastructure components creates software supply chains [2] in which a software artifact includes components, i.e., dependencies, that are created and maintained by others, that in turn often have their own dependencies, making the chain.

Because of the interdependent and community-focused nature of open source supply chains, *trust* is an integral part of their successful and secure functioning. But sometimes that trust can be exploited to turn a dependency into a deliberate attack vector potentially putting all direct and indirect users of the dependency at risk [11, 15]. Due to attacks, or fear of such attacks, developers often adopt mitigation strategies to reduce their risk, for example, some companies create private mirrored and internally validated versions of the open source dependencies they rely on [19]. Individual risk mitigation strategies can be costly and time-consuming and have little to no benefit to the broader community relying on those same dependencies, reducing efficiency from a community perspective [19]. In other words, a lack of trust in open source supply chains is cost-ineffective and can undermine and threaten open source and the benefits it provides if developers feel the need to divert resources to reduce their private risk rather than supporting and reinforcing the open source communities they benefit from.

Because of the key role trust takes in open source supply chains, it is important for us to understand and monitor the state of trust in open source, and to detect when and where trust changes to then explore interventions. In this study, we explore how to *measure trust* in open source supply chains. However, trust is a complex, nuanced, and multifaceted concept, which makes operationalizing

and measuring trust difficult [6, 10]. Existing security and supply chain research often studies the concept of trust without a clear definition [8, 21], existing survey and interview work on trust often relies on the participants' intuitive understanding of trust [27], and measures of trust often rely on obvious and easily available signals like GitHub stars without deeper grounding [22, 25].

Our goal is to develop robust measures for trust. Ideally, our measures will be grounded in how people actually behave, i.e., observing *revealed preferences* identified from behavior rather than relying on *stated preferences* from asking directly about the abstract concept of trust – therefore we need to define trust and identify how behaviors relate to trust. We approach measuring trust systematically by defining and decomposing trust into facets that are amendable to concrete measurement. Our work is grounded in the formalized definition of trust and the components of trust that influence whether trust is established that we present, with is sourced from decades of research on organizational trust and interpersonal trust in fields like organizational psychology and sociology. To that end, we accomplish this goal in two high-level parts. First, we begin by decomposing the high-level concept of trust in open source supply chains into concrete trust relationships and their corresponding trust contracts. Second, we demonstrate how we can identify details of developer behavior that can serve as indicators for the components of trust for a given trust contract, and how in turn those indicators can be operationalized to design concrete data-driven metrics that can serve as signals trust in practice, as we illustrate in Figure 1. We believe that this decomposition will allow the community to design better measures, perform longitudinal studies, identify problem areas and explore interventions.

In this paper, we contribute a process for decomposing trust in a complex large-scale system into key trust relationships and contracts, systematically identifying behavior-based indicators for the components of trust for a given contract, and in turn operationalizing data-driven metrics for those indicators, allowing for the wide-scale measuring of trust in practice. To demonstrate how this process can be used to measure trust in open source supply chains, we completed the literature analysis and the identification of contracts based on an existing taxonomy of supply chain attacks. We believe this process is applicable to other areas of interpersonal and organizational trust in computing.

2 DEFINING TRUST AND ITS COMPONENTS (RELATED WORK)

Defining trust is a prerequisite for designing and assessing trust metrics. Related work studying trust in the context of software and security usually does so without providing a formalized definition of trust, instead relying on the participant's internal implicit and intuitive definition of trust or on actions that the researchers associate with trust. For example, one interview study asked respondents how they "see the role on trust in the Linux Kernel community" which participants might interpret in many different ways (to which more than half the responses included the statement 'yes') [3], and another interview study asked participants about trust (without using the term in the interview guide) when using open source by asking about decisions regarding dependency selection[8].

Although the concept of trust is often used intuitively in software and security research, there is a long history of social sciences research identifying definitions, models, and theories of trust. This research provides a foundation for a more concrete and nuanced understanding of trust and the relevant factors for establishing and observing trust. While there is a separate line of research in trust in automation [12, 16], here we build in particular on research on organizational trust.

2.1 Defining Trust

We build on the definition of trust by Mayer et al. [18] and define trust as *the willingness of the trustor (the person doing the trusting) to take risks by being vulnerable to the trustee (the person being trusted).* Trust requires risk; for trust to be established, the trustor must accept the risk of an undesirable event possibly occurring, making themselves vulnerable to the trustee's actions. For the trustor to accept this vulnerability they must (1) perceive that the event that is at risk of occurring is undesirable; and (2) believe it is possible for this event to occur [13]. Trust requires the trustor to believe the trustee will act in a way that aligns with their best interest. In cases where the trustor does not believe this enough to accept the aforementioned vulnerability, the trustor *distrusts* the trustee [23]. Distrust indicates the trustor is not willing to fully accept being vulnerable to the actions of the trustee, which typically manifests in attempts to mitigate the risk of unfavorable event occurrence. For example, if an organization is interested in adopting an open source library for use in one of their software products but they have some distrust surrounding potential security vulnerabilities in the library, they might (a) adopt the dependency but create a private fork which they internally audit or (b) accept the opportunity cost of not adopting it. Conversely, observed mitigations could be interpreted as a sign of distrust.

While trust requires the acceptance of a risk that makes the trustor vulnerable to an undesirable event occurring to them, trust is not only necessary but beneficial in many situations [20]. Without trust, everybody interacting with the same organization incurs costs of mitigating risks. Mitigation costs may be shared, but that might again require trust. In communities with trust where trustees' actually act in the trustors' best interest, these extra costs can be avoided. For example, in the context of open source supply chains, trust is beneficial because it allows developers to rely on open source artifacts built and maintained by others and while they are making themselves vulnerable to the strangers code, the trust allows them to build on that code instead of writing and verifying every single update themselves, rapidly speeding up software production times and lowering upfront costs.

2.2 Components of Trust

Mayer et al. [18] further established a framework of what influences trust relationships. They identify that *the willingness of the trustor to take risks by being vulnerable to the trustee is influenced by the trustor's* ***propensity*** *and the trustee's perceived* ***ability***, ***benevolence***, *and* ***integrity*** [18]. We refer to these four factors that influence trust as the *components* of trust. The trustor's *propensity* is their general willingness or tendency to trust, i.e., the likelihood that they will choose to trust in any given situation. The trustee's *perceived*

ability relates to their skills, competencies, and characteristics that enable them to have influence over some specific domain. The trustee's *perceived benevolence* is their desire to do good, aside from egoistic motives, by demonstrating caring and goodwill. Finally, the trustee's *perceived integrity* is extent to which they follow a set of principles that the trustor is accepting of. Understanding the four components of trust is useful in designing measurements of trust because it allows for the creation of explicit measures that focus on individual aspects that influence whether trust is built.

2.3 Contractual Trust

In addition to disentangling the components of trust, we also consider different *contracts*, a concept explored under the name *contractual trust* [10, 13, 23]. A trust contract focuses the trust relationship on a specific aspect of a specific relationship, that is, a trustor believes the trustee will complete a specific action [10, 23]. For example, an case of general trust would be *'Julie trusts the developer of a dependency,'* whereas an case of contractual trust would be *'Julie trust the developer of the dependency not to introduce a backdoor'* or *'Julie trust the developer to write secure code.'* Contractual trust is particularly relevant to trust in open source software supply chains because it allows us to identify and enumerate the many different trust relationships and their corresponding trust contracts within the larger umbrella of the open source software supply chain.

3 DECOMPOSING AND MEASURING TRUST IN SUPPLY CHAINS

Previous studies on trust in software and security usually investigate the generic concept 'trust' in open source communities without specifying *who* they are trusting about *what* and *why*. That is, they do not identify a contract and do not consider which component of trust most influence those decisions.

We expect that a more specific and decomposed notion of trust will allows us to define multiple measures of distinct components of trust. We expect such measures to be more reliable and to enable pointing trust issues to specific relationships or concerns. In open source supply chains, there is not just one person trusting another but, instead, there is a complex network of interconnected parties engaging in many different trust contracts. Some of these trust contracts could include the user of an open source project trusting that the project's maintainers have not intentionally included malicious code in the project, or the maintainers of a project trusting that a new contributor has good intentions and is not planning on injecting malicious code into the project as soon as they get commit access. By only asking generalized questions about trust, we miss the opportunity to learn about the various trust relationships in this complex network.

To decompose trust specifically for software supply chains, we proceed in two steps: Decomposition by contract and decomposition by trust component.

3.1 Identifying Trust Contracts

We first identify relevant trust contacts for key trust relationships whose violation can lead to common supply chain attacks. We identify these trust contracts by working backwards from attacks to contracts that must be violated for those attacks to happen. Specifically we develop the following process: we systematically analyze all attacks in the recent taxonomy of supply chain attacks [15]; then, for each attack, we identify the potential trust relationships involved and the expectations associated with that trust. Often, multiple contracts relate to an attack and a trust contract may be relevant for multiple attacks. This way, we explicitly trace from an attack and corresponding threat to trust relationships and contracts, as illustrated with an excerpt in Table 2.

3.2 Identifying Indicators of Trust Components for Each Trust Contract

After identifying the trust relations and contracts, we now analyze each contract individually to identify possible details of developer behavior that could serve as *indicators* for the four components of trust introduced in Section 2.2. Specifically we develop the following process: for a given contract, we consider each of the four components of trust in turn, thinking through potentials details of developer behavior that could serve as indicators for each component within the context of the given contract. This process is then repeated for each trust contract. Similar to the previous step, this is a structured form of analysis, using the trust contracts and trust components as checklists to systematically identify indicators at every combination. While not every combination yields an indicator, with this structured process we found many more indicators than we would have with just an open brainstorming session. For some combinations, we identified multiple indicators.

As an example, consider the trust contract *the project maintainer (trustor) trusts the new contributor (trustee) will not add malicious code*:

- *Propensity:* The maintainer's *propensity to trust* could be observed indirectly by looking at their past actions, specifically seeing how they onboarded other contributors as maintainers with similar experience previously. If they have openly embraced new contributors in the past without much screening, this signals a higher propensity to trust.
- *Perceived integrity:* A new contributor who has been engaged in good security practices across multiple projects in the past may be perceived to have more integrity.
- *Perceived benevolence:* A new contributor with a long history of contributing to open source projects may be perceived as more benevolent.
- *Perceived ability:* For this contract, we do not think it makes sense to think of the new contributor's ability (e.g., ability to perform an attack if they wanted?). We do not expect to find an indicator for every contract-component combination.

3.3 Operationalizing Metrics for Indicators

Once we identify indicators of trust, we can explore ways to actually operationalize metrics for these indicators. The transparency of open source platforms like GitHub provides many opportunities to develop measures based on publicly available trace data.

In developing measures, we have a strong preference for observing behaviors rather than asking opinions. Analyzing behaviors is a way to identify *revealed preferences*, rather than relying on easily influenced stated preferences [26, 28].

Trust Relationship	Attack (Effect)	Associated Threats (Cause)	Trust Contracts
Project maintainer and new contributor	A malicious new contributor is added as a maintainer to project	Run a malicious build job (tampering with system resources) Compromise the maintainer system (exploit vulnerabilities, add malicious components to maintainer systems) Take over legit accounts (stealing account credentials)	*The project maintainer (trustor) trusts the new contributor (trustee) to...* … not include malicious dependencies … not steal vulnerable information … not add malicious code
Dependency user and maintainer	A developer adopts a dependency that contains malicious code	Mask legitimate packages (targeting name or URL resolution) ...	*The dependency user (trustor) trusts the maintainer (trustee) to…* … not intentionally include malicious code ...

Figure 2: Example entry from open source supply chain trust contracts table

While we leave operationalization to future work, we illustrate some possible operationalizatized metrics for the indicators above:

- *Propensity:* To operationalize the indicator of how quickly maintainers grant write access to repositories, we could measure the average time between similar contributor's first interaction with the project (e.g., first issue, first pull request) and when the contributor received write access to the repository (recognizable in GitHub by the fact that they can commit directly or close pull request opened by others). Notice that adding a contributor is a rare event and the first interaction and write-access action are only proxies for underlying phenomenon, hence we do not expect to provide accurate trust assessments for individual maintainers or projects, but we expect that this measure can characterize one aspect of trust *at an ecosystem level*, especially when observed longitudinally.
- *Perceived integrity:* To operationalize the indicator of whether new contributors engage in good security practices, we can look for various aspects in their repositories, such as badges indicating security practices, use of dependency monitoring tools, average lag of dependency updates, presence of known vulnerabilities in a dependency network, past reports of vulnerability, and so forth. Existing initiatives such as OpenSSF Scorecards can also be used where available.
- *Perceived benevolence:* To operationalize the indicator of a new contributors history, we could use straightforward analyzes of commit activities on GitHub through the GitHub API. We can easily measure their tenure as a maintainer, the number of maintained and contributed projects, the reputation of those projects in terms of stars, and so forth.

While we prefer direct measures on trace data because they allow assess historical trends from past behavior, the decomposition into indicators can also help to design surveys that ask more nuanced questions than *'Do you trust in the software supply chain?'* We could specifically ask about the different indicators, such as *'Do you look at the previous projects of a new contributor and their security outcomes?'* Also with survey questions, we prefer to ask questions about behaviors rather than questions about opinions to get more reliable answers. Especially when such surveys are run yearly on representative samples of developers, we expect to see clear trends in how trust develops in the community.

4 FUTURE PLANS

In this paper, we propose an overall process of decomposing trust in terms of components and contracts, identifying indicators of trust for each combination of identified component and contract, and then developing measures for each. We completed the literature analysis and the identification of contracts based on an existing taxonomy of supply chain attacks and we identified a number of indicators, full results are available on Zenodo [1]. DOI 10.5281/zenodo.10372704 We believe this process is reproducible and applicable to other areas of interpersonal and organizational trust in computing. At the same time, we leave the actual operationalization and validation of measures and the assessment of trends in supply chain security for future work.

We hope that our work enables the following new directions:

- *Longitudinal observations of trust:* If metrics can be derived from trace data or surveys are conducted repeatedly, we can observe trends in trust at large across ecosystems. Instead of just observing how opinions change, we can track how behaviors change, e.g., whether developers start to engage in more risk mitigation strategies that raise the collective cost for development with open source dependencies.
- *Designing signals for better decision making:* If our metrics accurately reveal practices relating to trust, it can be worth to make these practices transparent to foster better collective decision making in communities. If our example metric for benevolence is commonly accepted, we could make it more visible, e.g., as a community badge or an assessment signal shown on GitHub. Such strategies to develop explicit signals to support decentralized coordination have been shown to be effective in open source communities [24].
- *Validated measures of trust:* We need to validate that our measures of trust align with developers' perceptions. Where revealed preferences derived from actions seem to contradict stated preferences, interesting research opportunities exist to explore origins of this mismatch. Ideally, once validated these metrics can be applied broadly by the research community, similar to how the field of psychology has validated surveys to measure and diagnose things like burnout using the Maslach Burnout Inventory [17], or depression using the Beck Depression Inventory [4].

- *Detecting areas of unwarranted trust:* Finally, we can compare trends in trust to actual attacks occurring. Are distrust and actions that developers take to mitigate risks calibrated to the actual level of risks faced? Is trust that they have in others actually warranted? Trust literature has a whole another dimension of *trustworthiness* and *unwarranted trust* [9, 13, 14] which provides an interesting foundation for exploring trust in the context of actual capabilities of actors.

REFERENCES

[1] 2024. *Supplementary Material for Decomposing and Measuring Trust in Open-Source Software Supply Chains.* Zenodo. https://doi.org/10.5281/zenodo.10372704
[2] Christopher J Alberts, Audrey J Dorofee, Rita Creel, Robert J Ellison, and Carol Woody. 2011. A systemic approach for assessing software supply-chain risk. In *2011 44th Hawaii International Conference on System Sciences*. IEEE, 1–8.
[3] Maria Antikainen, Timo Aaltonen, and Jaani Väisänen. 2007. The role of trust in OSS communities—case Linux Kernel community. In *Open Source Development, Adoption and Innovation: IFIP Working Group 2.13 on Open Source Software, June 11–14, 2007, Limerick, Ireland 3*. Springer, 223–228.
[4] Aaron T Beck, Robert A Steer, Gregory K Brown, et al. 1987. *Beck depression inventory*. Harcourt Brace Jovanovich New York:.
[5] Paul Cormier. 2022. The state of enterprise open source: A red hat report. https://www.redhat.com/en/resources/state-of-enterprise-open-source-report-2022. Accessed: 2023-09-12.
[6] Graham Dietz and Deanne N Den Hartog. 2006. Measuring trust inside organisations. *Personnel review* 35, 5 (2006), 557–588.
[7] Nadia Eghbal. 2016. *Roads and bridges: The unseen labor behind our digital infrastructure*. Ford Foundation.
[8] Javad Ghofrani, Paria Heravi, Kambiz A Babaei, and Mohammad D Soorati. 2022. Trust challenges in reusing open source software: An interview-based initial study. In *Proceedings of the 26th ACM International Systems and Software Product Line Conference-Volume B*. 110–116.
[9] Russell Hardin. 2002. *Trust and trustworthiness*. Russell Sage Foundation.
[10] Katherine Hawley. 2014. Trust, distrust and commitment. *Noûs* 48, 1 (2014), 1–20.
[11] Trey Herr, William Loomis, Stewart Scott, and June Lee. 2020. Breaking trust: Shades of crisis across an insecure software supply chain. https://www.atlanticcouncil.org/in-depth-research-reports/report/breaking-trust-shades-of-crisis-across-an-insecure-software-supply-chain/. Accessed: 2023-09-12.
[12] Kevin Anthony Hoff and Masooda Bashir. 2015. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human factors* 57, 3 (2015), 407–434.
[13] Alon Jacovi, Ana Marasović, Tim Miller, and Yoav Goldberg. 2021. Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in AI. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 624–635.
[14] Kristiina Karvonen. 1999. Creating trust. In *Proceedings of the fourth Nordic Workshop on Secure IT systems (Nordsec'99)*. 21–36.
[15] Piergiorgio Ladisa, Henrik Plate, Matias Martinez, and Olivier Barais. 2023. Sok: Taxonomy of attacks on open-source software supply chains. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1509–1526.
[16] John D Lee and Katrina A See. 2004. Trust in automation: Designing for appropriate reliance. *Human factors* 46, 1 (2004), 50–80.
[17] Christina Maslach and Susan E Jackson. 1981. The measurement of experienced burnout. *Journal of organizational behavior* 2, 2 (1981), 99–113.
[18] Roger C Mayer, James H Davis, and F David Schoorman. 1995. An integrative model of organizational trust. *Academy of management review* 20, 3 (1995), 709–734.
[19] Courtney Miller, Christian Kästner, and Bogdan Vasilescu. [n. d.]. "We Feel Like We're Winging It:" A Study on Navigating Open-Source Dependency Abandonment. ([n. d.]).
[20] Barbara Misztal. 2013. *Trust in modern societies: The search for the bases of social order*. John Wiley & Sons.
[21] Vibha Singhal Sinha, Senthil Mani, and Saurabh Sinha. 2011. Entering the circle of trust: developer initiation as committers in open-source projects. In *Proceedings of the 8th Working Conference on Mining Software Repositories*. 133–142.
[22] Mahbubul Syeed, Juho Lindman, and Imed Hammouda. 2017. Measuring perceived trust in open source software communities. In *Open Source Systems: Towards Robust Practices: 13th IFIP WG 2.13 International Conference, OSS 2017, Buenos Aires, Argentina, May 22-23, 2017, Proceedings 13*. Springer, 49–54.
[23] Jonathan Tallant. 2017. Commitment in cases of trust and distrust. *Thought: A Journal of Philosophy* 6, 4 (2017), 261–267.
[24] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 2018. Adding sparkle to social coding: an empirical study of repository badges in the npm ecosystem. In *Proc. Int'l Conf. Software Engineering (ICSE)*. 511–522.
[25] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of social and technical factors for evaluating contribution in GitHub. In *Proc. Int'l Conf. Software Engineering (ICSE)*.
[26] Hal R Varian. 2012. Revealed preference and its applications. *The Economic Journal* 122, 560 (2012), 332–338.
[27] Dominik Wermke, Noah Wöhler, Jan H Klemmer, Marcel Fourné, Yasemin Acar, and Sascha Fahl. 2022. Committed to trust: A qualitative study on security & trust in open source software projects. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1880–1896.
[28] Morteza Zadimoghaddam and Aaron Roth. 2012. Efficiently learning from revealed preference. In *Internet and Network Economics: 8th International Workshop, WINE 2012, Liverpool, UK, December 10-12, 2012. Proceedings 8*. Springer, 114–127.