International Journal of Systems and Software Security and Protection
Volume 15 • Issue 1 • January-December 2024

Organizational Influence on Security Development in Open-Source Software Projects

Roland Robert Schreiber
University Bamberg, Germany

ABSTRACT

Increasing technological complexity, intensified competition, and security requirements have driven open-source software (OSS) projects to become a crucial part of organizations' software development. This study focuses on the OSS project TensorFlow (TF) and uses a case study to examine how organizations and their associated developers collaborate to identify, fix and prevent security vulnerabilities. Social Network Analysis (SNA) of archived security data from software repositories is used to gain insight into security activities. The study examines the internal structure and evolution of security code collaboration, organizational networks, and top organizational contributors to TF. It also examines productivity, homophily, development diversity, and turnover rates among developers across various software releases. The in-depth insights from this research enhance our understanding of collaborative patterns in OSS communities within open software ecosystems, particularly in the security context.

KEYWORDS

Diversity, Evolution, Open source, Organizational Influence, Productivity, Security, Social Network Analysis, Software Development Project, Structural, TensorFlow, Vulnerabilities

INTRODUCTION

Open-source software (OSS) is increasingly prevalent in today's digital landscape, and developers around the world contribute their expertise to a global pool of knowledge. There is a diverse group of collaborators, which includes volunteers, in-house developers, developers from partner firms, university representatives, and even competitors from various companies, who work together (Bengtsson & Kock, 2000).

This collaborative approach creates a challenge: ensuring security and maintaining trust in the software. The impact of vulnerabilities can be massive, as in the cases of Heartbleed CVE-2014-0160, WannaCry CVE-2017-0144, and the Equifax data breach CVE-2017-5638, the latter affecting half the U.S. population (Durumeric et al., 2014; Christensen & Liebetrau, 2019; Dong et al., 2019).

Organizational influencers are crucial in ensuring that a project's security is taken seriously and that appropriate measures are taken to maintain it. Numerous famous firms support different OSS projects and cooperate with others to find, fix, and prevent security vulnerabilities (Capiluppi et al., 2012). WebKit, OpenStack, and CloudFoundry are all examples of modern secure software in the open-source arena with this type of coopetition (Nguyen Duc et al., 2017).

DOI: 10.4018/IJSSSP.356659

Table 1. OSS ecosystems

Project	Domain	Examples of companies participating in OSS ecosystems
Blink	Web browsing	Alphabet, Opera, Intel and Samsung
CloudStack	Cloud computing	Citrix, SunGard AS and ShapeBlue
Eclipse	Software development	Actuate, CA, IBM, Alphabet, Oracel, SAP and Red Hat
Hadoop	Distributed computing	Facebook, Twitter, LinkedIn, Jive and Microsoft
Linux	Operating system	Fujitsu, HP, IBM, Intel, Samsung, Hitachi and Red Hat
OpenStack	Cloud computing	Rackspace, Canonical, IBM, HP, Vmware and Citrix
TF	Machine learning	Alphabet, Dropbox, Airbus, Uber, Deepmind and JD.com
WebKit	Web browsing	Apple, Nokia, Alphabet, Samsung, Intel and BlackBerry

To build more secure OSS ecosystem software, it is essential to understand the structure and collaboration of developers in security practices (Jabangwe et al., 2018; Wang & Nagappan, 2021). These OSS developments in projects are tremendously dependent on the use of robust software libraries, effective interactions between people, and the critical human factors that are integral to successful software projects (Oliveira et al., 2018). Challenges are continually growing and include the development of key novel products, large geographic distances between participants across different time zones, collaboration issues, security issues, and open software standards (Ouriques et al., 2019).

This paper analyzes developers' interactions when addressing security vulnerabilities and the importance of key organizations in building a secure software ecosystem, TensorFlow (TF). We also consider the social relationships between developers and their organizational affiliations in the context of security-related activities. Using the case study approach, we explore the interaction pattern (Basili et al., 1999); that is, how organizations and developers collaborate to find and fix security vulnerabilities in a complex open-source ecosystems such as TF (Abufouda & Abukwaik, 2017; McClean et al., 2021; Schreiber & Zylka, 2020). This is the first research into security collaboration networks built on organizations' interactions towards fixing security vulnerabilities. The study uncovers novel insights into the TF ecosystem, providing a deeper understanding of the informal security coordination structures and their evolution. Moreover, the paper provides a theoretical basis with which to formulate recommendations for future research. Our results are relevant for practitioners as well as academics who are looking for an overview of TF OSS ecosystem security collaboration, insights into software development in the ecosystem and the role of social networks, and opportunities to participate to research in this specialized area.

The next sections of this paper are organized as follows. The Background section describes the theoretical background and related work. In Research Questions, we formulate the research questions (RQs) and the research focus. The Methodology section describes the case study methodology, including our data collection and research method. In Results, we present and evaluate the results of the TF OSS project security case study. The Limitations section discusses the limitations and threats to the validity of this study, and the Conclusion concludes the paper.

BACKGROUND

A software ecosystem is the interaction of a set of actors around a common technological foundation used for software solutions or services (Manikas & Hansen, 2013). Table 1 provides several examples.

OSS ecosystems are self-organized and involve dynamic processes in which volunteers and different organizations worldwide contribute to software solutions (Gerber et al., 2010). Furthermore,

these new ecosystems within organizations have a significant impact on OSS software development (Sarker et al., 2011; Zhou et al., 2017). They are becoming increasingly important and provide essential software components and infrastructure, including operating systems, libraries, component stores, and entire platforms (Ghafele & Gibert, 2014).

In the coopetition paradigm, competitors can be involved in both cooperative and competitive relationships simultaneously. The firms involved benefit from cooperation and gain competitive edges in these networks that serve their own reliable software products or services in a symbiotic way (Barbosa & Alves, 2011; Morgan & Finnegan, 2014).

Software forges are web-based collaborative platforms to ease distributed development, documenting, and source code sharing (Cosentino et al., 2017). GitHub is currently one of the most popular software development platforms, offering a range of advanced tools and features to assist developers in creating and managing software projects (Oliveira et al., 2023). This platform provides free hosting, a user-friendly interface, and a powerful version control system, as well as a range of social features which promote collaboration and community building within the development community (Guendouz et al., 2015; Squire, 2014). It also supports the version control system git, including features such as social interactions like pull requests support and profiles, along with an impressive GitHub Application Programming Interface (API) to access metadata for its hosted software projects (Rashid & Prakash, 2022).

Vulnerabilities are especially critical in these complex ecosystems (Peng et al., 2022; Tahaei & Vaniea, 2019; Wu et al., 2024; Zimmermann et al., 2019). The volunteer collaborators in these OSS ecosystems work together to fix bugs or security issues. Recent studies have explored the evolution of dependency networks in various OSS ecosystems, revealing important structural and security-related dynamics that differ across projects (Decan et al., 2019). Moreover, the modern landscape of OSS security is increasingly complex, with rising concerns related to performance and governance challenges (Zhao et al., 2021). In this context, security patch management has become a critical focus area, with empirical studies showing how OSS projects manage and apply security patches to mitigate vulnerabilities (Alfadel et al., 2023; Delicheh et al., 2024).

The social aspects of security are the main focus of investigation in this study. Social network analysis (SNA) of OSS ecosystems has the potential to reveal hidden structural security issues, top influencers, and collaboration patterns, knowledge of which can help ensure success (Fischbach et al., 2009; Šmite et al., 2017).

There is a large body of research on the static characteristics of communication networks and the structural characteristics of developer collaboration networks in OSS ecosystem development activities (Çaglayan & Bener, 2016; Crowston et al., 2012; El Mezouar et al., 2019; Gharehyazie et al., 2015; Jermakovics et al., 2013; Joblin et al., 2017b; Zhang et al., 2014). The software development in large OSS ecosystems is a globally-distributed, knowledge-intensive process that involves complex and self-organized interactions among members (Herbsleb & Mockus, 2003; Hinds & Lee, 2009; Behfar et al., 2018; Shah, 2006).

So far, research has mainly concentrated on social project structure and clique analysis, which encompasses the subjects of core periphery and clusters (Concas et al., 2017; Joblin et al., 2017a; Schreiber, 2023; Yu et al., 2016). Other findings have shown that cohesive social ties among network members lead to increased productivity (Lee & Cunningham, 2013; Singh et al., 2011). There are also publications that address prediction in support of various software engineering processes, such as the bug triad, defect prediction, and leadership change forecasting, and build failure prediction (Jeong et al., 2009; Bird et al., 2009; Meneely et al., 2008; Howison et al., 2006; Wolf et al., 2009). Researchers have also used social network metrics to predict new vulnerabilities, explore the impact of human factors on security vulnerabilities, and monitor vulnerabilities (Shin et al., 2011; Zimmermann et al., 2010; Meneely et al., 2014; Meneely & Williams, 2009; Meneely & Williams, 2010; Zafar, 2022; Trabelsi et al., 2015). In addition, research on open-source secure software development shows that

Table 2. RQs

No.	RQs
RQ1	How does security development collaboration evolve over time in TF OSS ecosystems?
RQ2	Which organizations contribute to the security activities of TF OSS ecosystems?
RQ3	Which organizations collaborate in the ecosystems for security issues?

it is a team effort and that collaborative developer networks exist (Meneely et al., 2014; Meneely & Williams, 2010; Shin et al., 2011; Sureka et al., 2011; Trabelsi et al., 2015).

What is lacking, however, is research on the dynamic aspects of social networks in the context of software security developers' activities in ecosystems throughout the life cycle. A deeper understanding of these developer security interactions is very helpful for building and supporting more secure software in ecosystems. There is also a lack of work on the formation of subgroups, productivity, participation of organizations, and their behavior within software ecosystem communities to address security issues (Herbold et al., 2021; Schreiber & Zylka, 2020).

The TF OSS ecosystem was selected as a case study to investigate collaboration networks due to its highly developed and complex structure, as well as its extensive use in the field of machine learning (ML). TF embodies a robust and mature ecosystem, featuring approximately 3300 developers, 108 subprojects, over 3 million lines of code, and 161,000 dependent repositories. It includes standard elements and libraries to help develop and train large-scale ML models using industry-wide and open-source standards. These attributes provide a rich context for examining the interactions among developers and organizations, particularly in addressing software security vulnerabilities. Google Brain initiated the creation of the TF OSS ecosystem in November 2015, and it was released under the Apache 2.0 open-source license (Abadi et al., 2016). This move aimed to accelerate its evolution by obtaining support from the larger community. Today it has many supporters, including Alphabet, Airbus, Dropbox, Uber, Deepmind, and JD.com, among others (Tensorflow, 2024). In addition, TF is a highly popular and dominant ML framework which is widely used in commercial production (Dinghofer & Hartung, 2020; Han et al., 2020).

RQS

Our research focuses on the collaborative relationship in coding TF security between developers and companies. The detailed questions related to this focus are listed in Table 2.

METHODOLOGY

Our case study employs combined quantitative and qualitative analysis of mined software repositories and uses methods from SNA on publicly accessible data from the TF OSS ecosystems (see Figure 1).

Data Collection

The entire TF project consists of three main projects—tensorflow (basic library of TF), docs (documentation), and community (project for documents used by TF developers)—and 108 subprojects. The git version system documents all of the commits made within the ecosystem, including any comments or details, and records them in the changelog documentation. We began by inspecting the different modules in separate subprojects within the TF ecosystem. We then mined the security-relevant source code and logs of the core development project tensorflow to obtain a deeper insight into the communities. Security issues were selected from the changelogs using regular

Figure 1. Overview of the study design

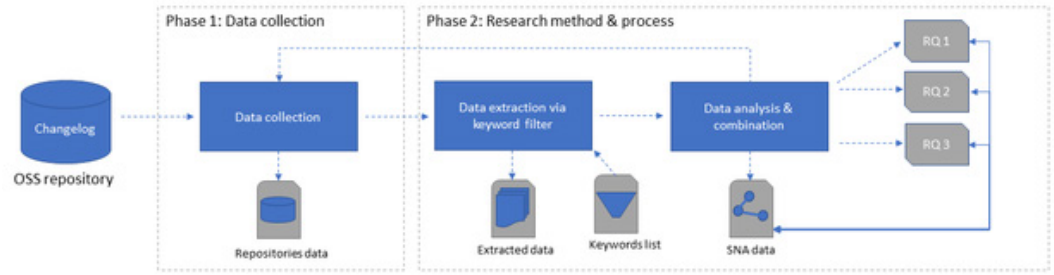


Table 3. Regular expression used to filter the security-related issues

Regular Expression
denial of service | \bXXE\b | remote code execution | \bopen redirect | OSVDB | \bvuln | \bCVE\b | \bXSS\b | \bReDoS\b | \bNVD\b | malicious | x−frame−options | attack | cross site | exploit | malicious | directory traversal | \bRCE\b | \bdos\b | \bXSRF \b | \bXSS\b | clickjack | session.fixation | hijack | \badvisory | \binsecure | security | \bcross−origin\b | unauthori[zls]ed | infinite loop | advisory | poison | exploitable | crack | hack

expressions for filtering (see Table 3) (Zhou & Sharma, 2017). We then connected the developers and companies that worked on the same files and created a social network based on the security collaboration patterns.

Data Analysis

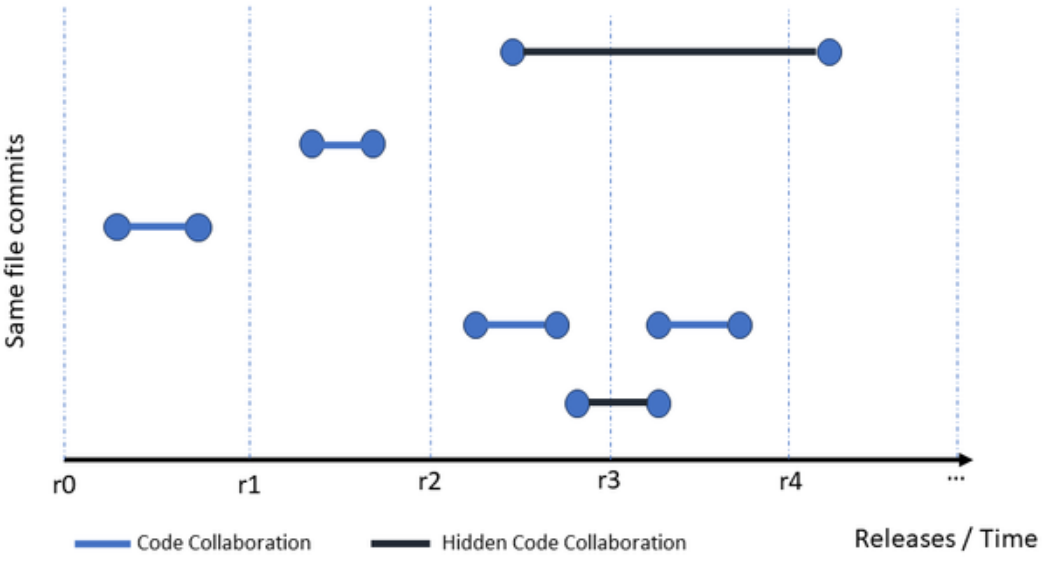
During this research, we constructed the collaboration network through different stages of the main project of TF (tensorflow). Our materials from the extracted data spanned the period from January 1, 2016 to March 20, 2023. Initially, the developers were identified by their email address and were assigned to a company based on information from the changelog records. Subsequently, a node in SNA represented a developer or a company, depending on the evaluation. When different developers edited a file together, we considered that to be collaboration, which was presented as an edge between different nodes.

The different developers, A, B, C, and D, were employed by various software development firms, such as Company 1, Company 2, and Company 3. Two developers were considered to have a collaborative relationship only if they both committed source code in the same file and in the same release (see Figure 2).

Based on this method and the extracted relationships, we were able to construct the social network at a clearly defined point in time. The construction process of the network of developers and companies is illustrated in Figure 3, providing an example of different release versions and individual developers. We gathered the complete social network for SNA at different evolution stages by time. In addition, we applied the SNA over all releases to remove the restrictions and cover all existing security code collaborations.

We constructed different networks of developers and companies for each time slice to analyze the development of social networks in the TF ecosystem. That enabled us to evaluate how the development of the collaboration has evolved and to uncover intriguing patterns. We conducted an analysis of social network data and visualizations using Gephi (v0.10) and several plugins, including MultiMode Networks Transformation, Node Color Manager, and Groups by partition. In addition, we employed various SNA methods to delve further into the constructed networks (Howison et al., 2011). Social networks can be described through structural and rational characteristics. Structural

Figure 2. Discovering code collaboration



properties, such as density and a network's size, correspond to the morphology of relationships between different actors. The rational dimension examines the connections between pairs of individuals and is characterized by homogeneity and intensity. Table 4 provides an overview of the fundamental social network characteristics and their different dimensions that were observed in our case study.

Social network theories, utilizing the basic properties described above, provide additional perspectives on involved social ties in ecosystems (see Table 5).

RESULTS

An Overview of the Developer Collaboration Over Releases

In this section, we present the results for RQ1, which yields an overview of developer collaboration evolution across the entire TF project and, specifically, in security development collaboration. We obtained archival data from the source version control system git, which spanned over eight years (2015-2023) of the ecosystem's existence. Our research primarily focused on the main versions of the TF OSS.

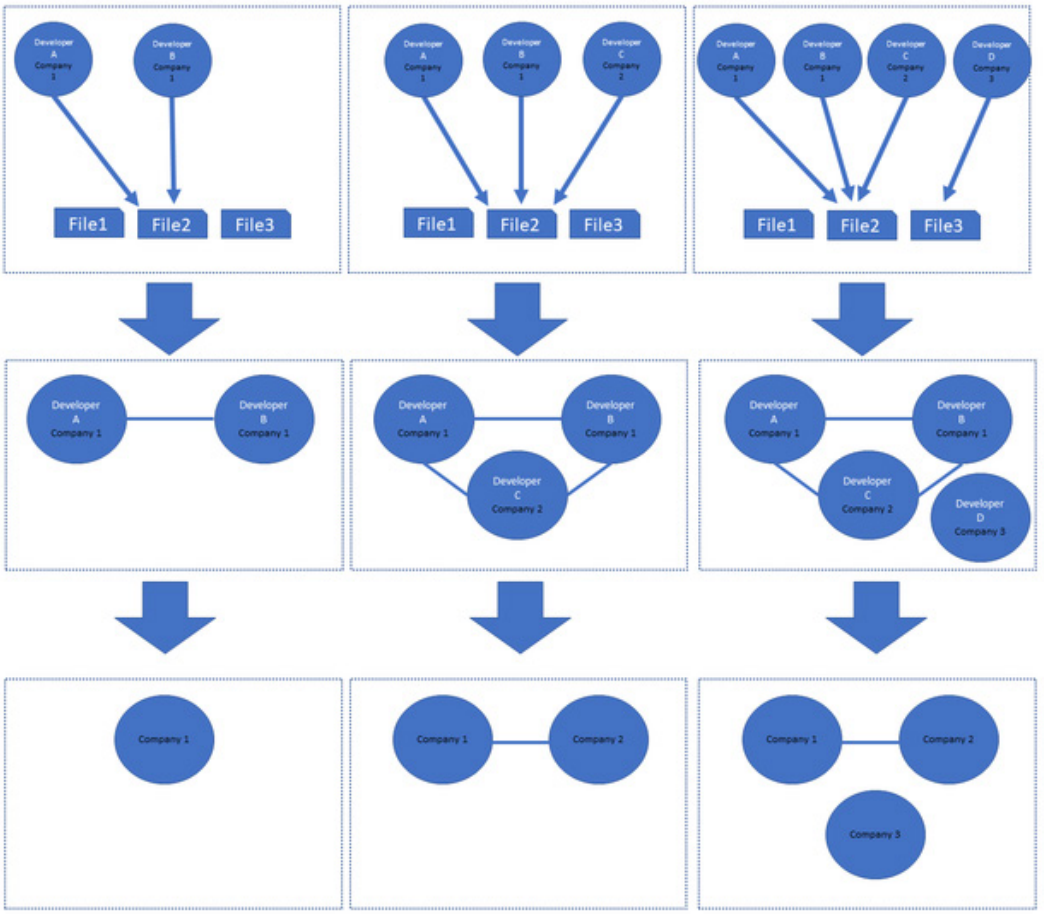
To obtain a better view of developer collaboration over time, we first analyzed the structural basic metrics of the social network: the sum of the relevant developers, network density, and number of communities. We also showed the number of commit activities, excluding merge commits, to represent the productivity of the developer community across the entire TF project.

Figure 4 shows the number of actively contributing developers (nodes) over the different TF releases. We can observe that the numbers of actors involved in development (Figure 4A) and security development activities (Figure 4B) increase over the releases.

Both network densities for software development and security are decreasing over the releases with increasing number of developers and collaborations in the TF ecosystem. Notably, there is a massive reduction in network density over the course of the releases in terms of normal software development, as shown in Figure 5.

The graph in Figure 6 illustrates a growing trend of community formation for both development types. After the first two releases, the network is always segregated, and many more connected

Figure 3. Construction process for the developer and company security network



subgraphs grow. In the normal TF development process, a significantly expanding giant component gradually emerges (Figure 6A), while the security developer network retains highly segregated nodes (Figure 6B).

The Top Organizations That Contribute to the TF Ecosystem

Addressing RQ2, we explored the relationship between developers and their affiliated organizations. Figure 7 shows that Simpson's Diversity Index increases for both general and security software development activities. The higher the diversity index, the more diverse the developer community and its affiliated organizations tend to be (Jiang et al., 2018). This means that the dominance by a few organizations decreases over the releases, and there is increasing diversity.

Figure 8 shows productivity with respect to the number of active contributors and the commits of the TF ecosystem. In both types of software development, productivity increases with the number of active contributors, and commit activity increases accordingly.

Furthermore, as Table 6 shows, our research unveils that six organizations significantly contribute to the development of TF security software.

Table 4. Basic network properties

Structural Characteristics	
Social Network Property	*Definition*
Density	This is the proportion of direct ties in a network relative to the total number of possible links (Wasserman & Faust, 1994).
Size	Size is the sum of important actors of a network (Wasserman & Faust, 1994).
Degree centrality	This is the number of direct ties a node has with other nodes (Scott, 2013).
Modularity	Modularity is a measure of the strength of the network's division into clusters (Blondel et al., 2008).
Diversity	Simpson's Diversity Index shows the diversity of the open-source ecosystem community's diversity based on developer-associated organizations (Jiang et al., 2018).
Rational characteristics	
Concept	*Definition*
Homogeneity	The homogeneity refers to the similarity of nodes (Schenk, 1995).
Intensity	Intensity is the strength of the relationship between nodes in a network (Scott, 2013).

Table 5. SNA theories

Theory concepts	*Definition*
Clique analysis	This analysis of social structure focuses attention on how connections of large social structures can be built up out of small and tight components (Kappelhoff, 1987).
Embeddedness	Trusting relationships between actors often develop through brokerage, and trust serves as the primary governing structure for cooperation, something on which transitivity has a structural effect (Grewal et al., 2006; Uzzi, 1997).
Structural holes	Individuals often have advantages or disadvantages based on their position within social structures. For example, a structural hole is a gap between two nodes which possesses complementary sources of information (Burt, 2009).
Strength of weak ties	This theory proposes that the most important links in a network are the weaker connections that link otherwise unconnected groups through a network bridge and allow distant clusters to access novel information (Granovetter, 1973).
Power-law distribution	Some actors possess a large number of incoming social ties, while other predominant actors have only a limited number ties (Barabási & Albert, 1999).

Organization Collaboration Over Releases

To address RQ3, we analyzed the developer and organizational collaboration networks in the TF ecosystem, in the security context, over the course of releases. The most significant coding collaborations are analyzed below.

The SNA visualizations in Figure 9 (9A) illustrate the general developer code collaboration using combinations of color for organization and location information. Each colored developer (node) represents one developer affiliated with one relevant organization for further security context investigations. A direct link between nodes shows that, in version v2.5.0, the source code in one file was edited together by different developers. The nodes highlighted in yellow are involved in security development activities. The Figure 9 (9B) SNA visualizations show the organizations' collaboration network.

Further visualizations in Figure 10 uncover the security-related collaborative coding network structure. Figure 10A illustrates developer code collaboration and Figure 10B illustrates code

Figure 4. Number of nodes over TF releases (4A) and number of nodes over TF releases regarding security development (4B)

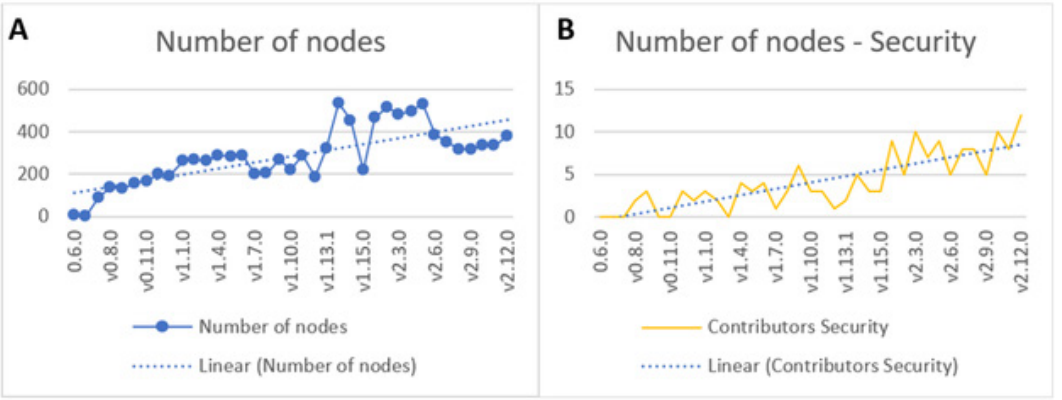


Figure 5. Network density over TF releases

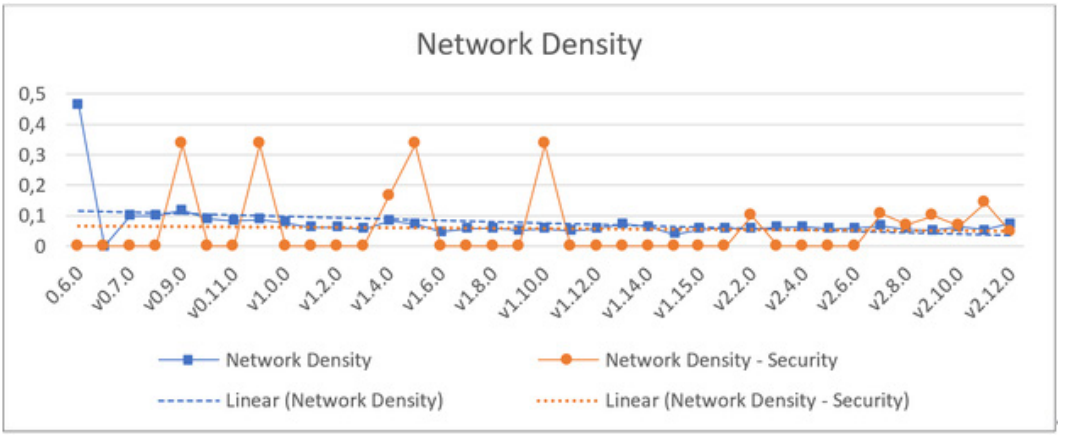


Figure 6. Number of communities over TF releases (6A) and number of communities regarding security (6B)

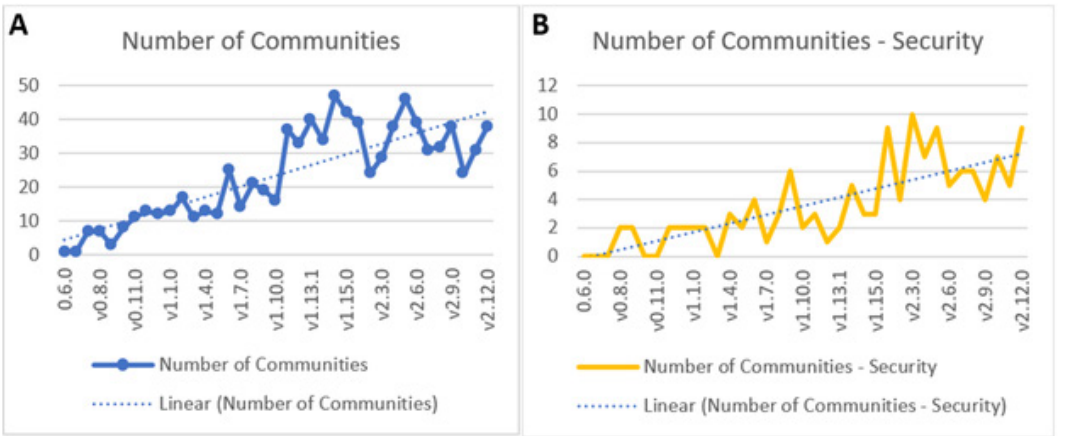


Figure 7. Simpson's diversity index over TF releases

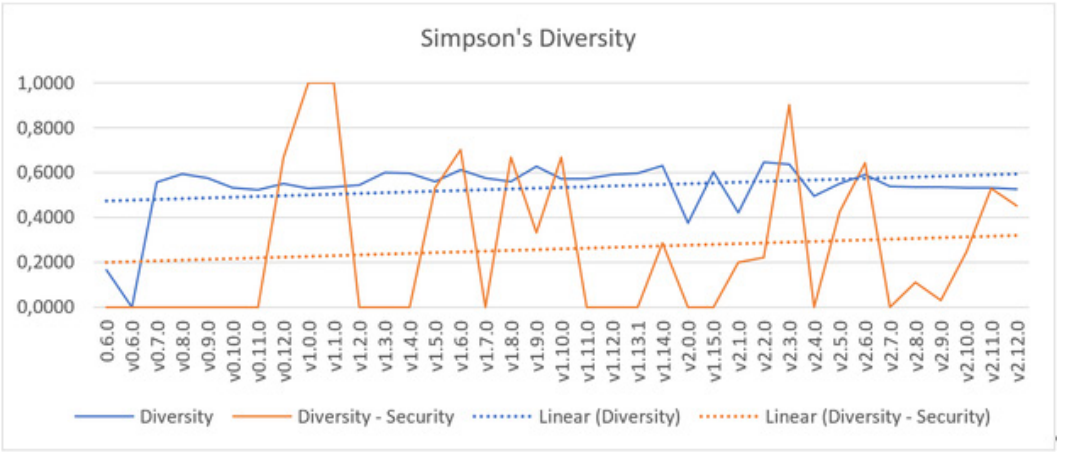
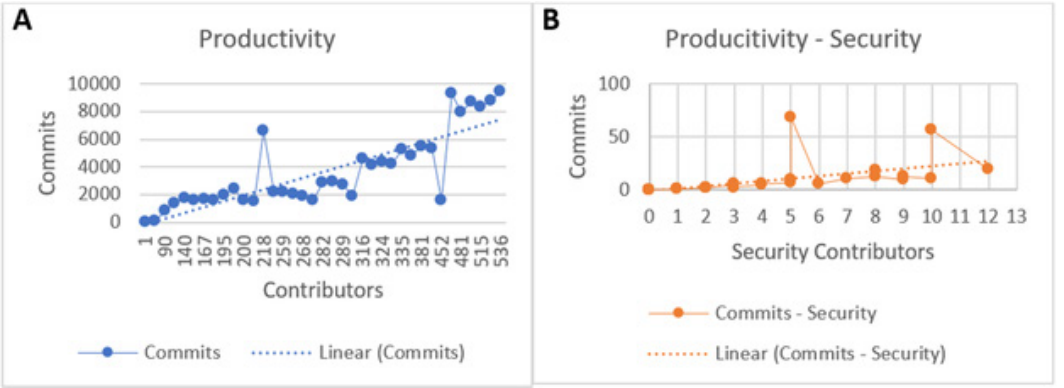


Figure 8. Productivity over TF releases (8A) and productivity over TF releases regarding security (8B)



collaboration among organizations. Note that there is no cooperation between the developers on the

Table 6. Top organizations contributing to the TF OSS ecosystem with security context

Organization	Firm description	Color
Alphabet	Alphabet (formerly Google) is a US-based multinational technology company which specializes in internet-related services and products, including a search engine, cloud computing, software, and hardware.	
TF	TF is an open-source community for ML and deep neural networks supported by Alphabet and other voluntary contributors.	
MobileIron	MobileIron is a US software company which provides unified endpoint and enterprise mobility management for mobile devices.	
Intel	Intel is a US-based multinational technology and software corporation.	
IBM	IBM is a US-based multinational technology, software, and consulting company.	
Nvidia	Nvidia is a US-based multinational technology and software company primarily for gaming and professional markets.	

Figure 9. Collaboration of network developers (9A) and organizations (9B) in version v2.5.0

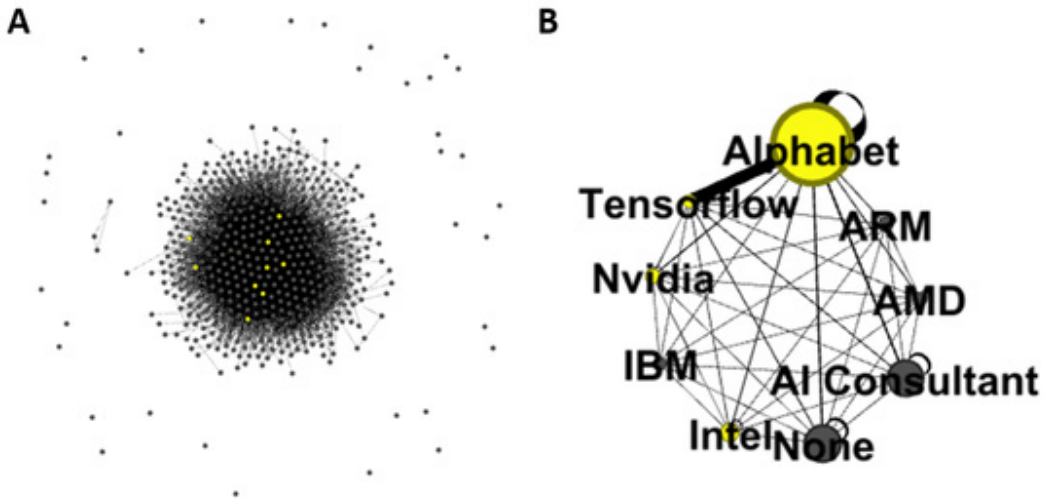
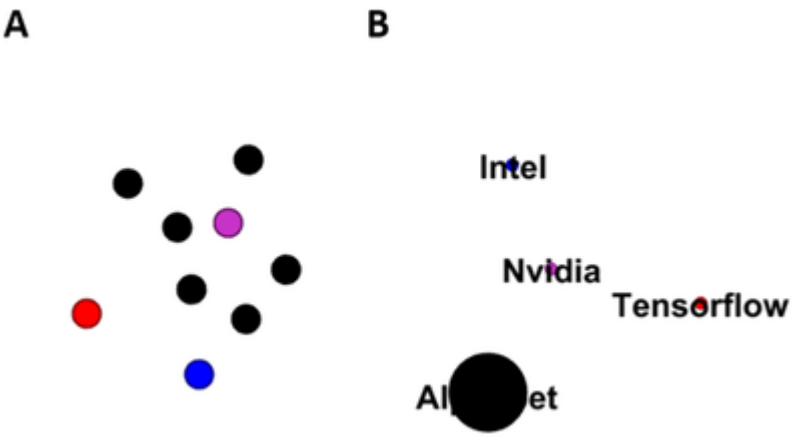


Figure 10. Focus on collaboration security network of developers (10A) and organizations (10B) in version v2.5.0



various security bugs.

While the first versions have a similar SNA structure as v2.5.0, a significant change occurs in later versions beginning with v2.8.0. Figure 11 shows another example: SNA visualizations of version v2.12.0. The Figure 11A SNA visualization is the entire TF developer collaboration network; the Figure 11B SNA shows the collaboration network of organizations.

The SNA visualization in Figure 12 focuses on the security development collaborative coding network structure. Figure 12A shows developer code collaboration and Figure 12B visualizes code collaboration among organizations. Note the increasing level of cooperation between developers to address the multiple security vulnerabilities.

To view the entire collaboration security network, we drew and then analyzed the collaboration security network of developers and organizations across all software releases using SNA (Figure 13). Figure 13A shows developer code collaboration and Figure 13B visualizes code collaboration among organizations. The topology of the developer network (Figure 13A) is characterized by the

Figure 11. SNA collaboration network of developers (11A) and organizations (11B) in version v2.12.0

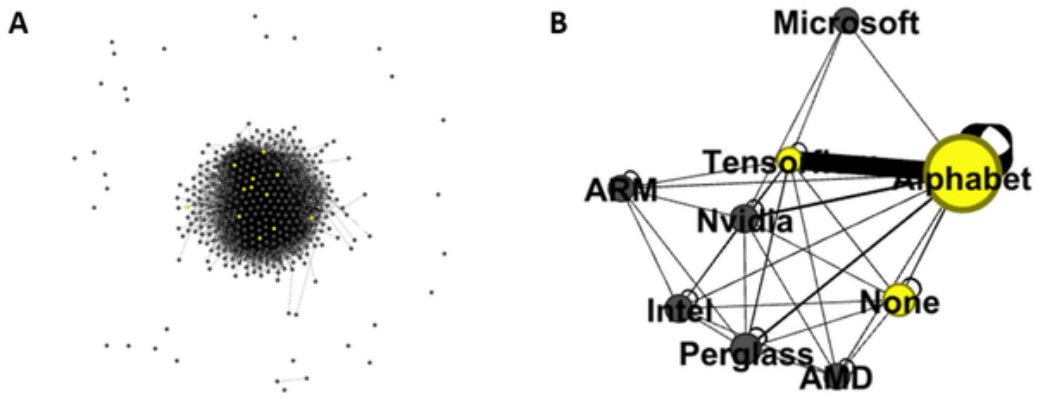
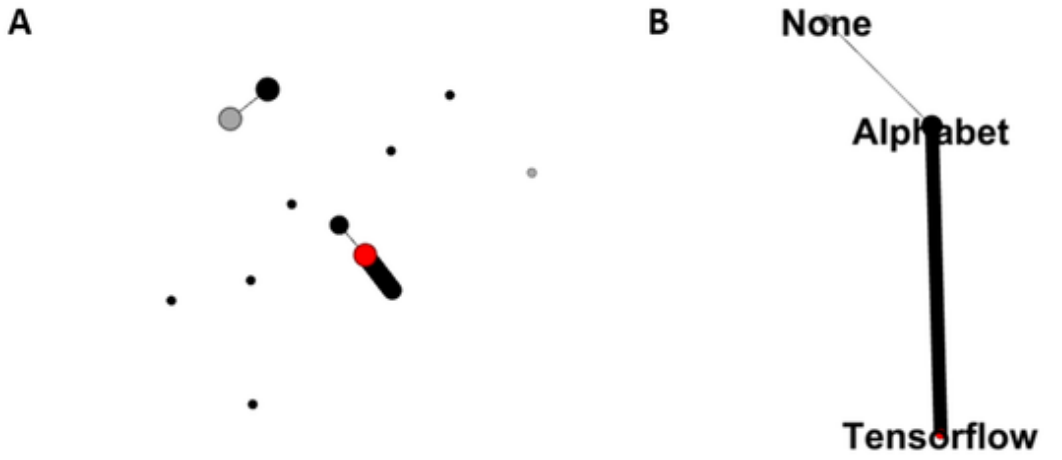


Figure 12. Focus on collaboration security network of developers (12A) and organizations (12B) in version v2.12.0



connection of various developers and it shows that the network is well-connected. The developers from the top organizations play an important role in making security-related changes and improvements to the TF OSS ecosystem, which shows a core-periphery network structure. These organizations, including Alphabet and TF itself, are especially pivotal in the initial stages of addressing security vulnerabilities, ensuring that critical issues are promptly identified and fixed. We can also see a power-law distribution over the entirety of TF OSS ecosystem versions. Alphabet and TF dominate the security network for the first versions. Gradually, new developers and thus new organizations join the TF software development, increasing the security. Many nodes connect only temporarily to close a specific relevant security gap and then disappear. Interestingly, our data indicate that as the diversity of contributors increases, the impact of these key organizations diminishes, suggesting a transition towards a more distributed and resilient security collaboration network. This is proven in the social network structure through the addition of a growing number of weak ties over the different versions.

Figure 14 shows the turnover of developers (A) and organizations (B) related to security collaborative coding. In Figure 14A, the green bars represent the number of developers who joined; the red bars show developers who left; and the yellow plot line represents the total number of developers involved in security-related coding in every TF ecosystem major release. The left *y*-axis corresponds

Figure 13. Focus on collaboration security network of developers (13A) and organizations (13B)

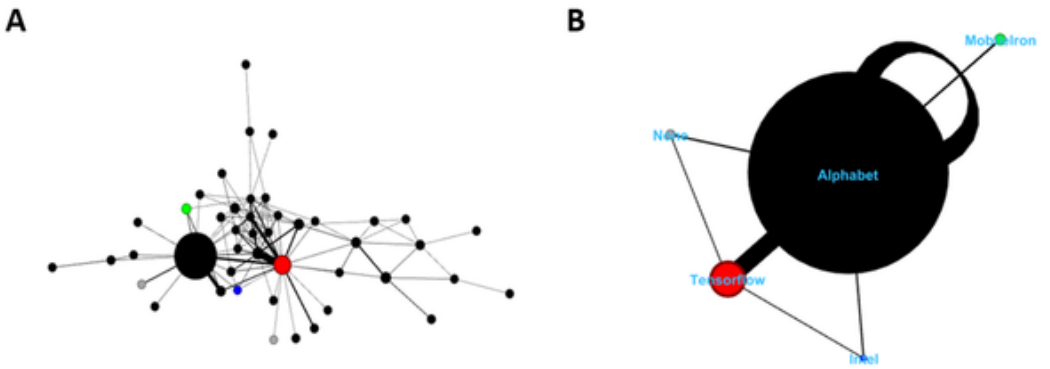
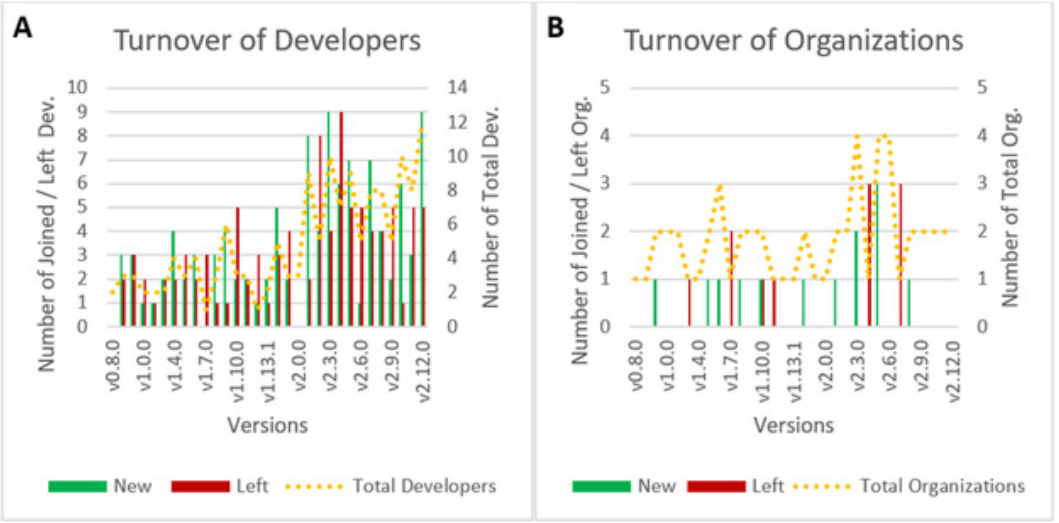


Figure 14. Turnover of developers (14A) and organizations (14B) in security development activities



to the green and red bars, and the right *y*-axis corresponds to the yellow plot line. The number of leaving and joining developers increases over versions, and an uptrend of contributing developers can be detected. Conversely, there is a high degree of fluctuation, with an average turnover rate of about 45% of developers who join in one version and later withdraw.

Figure 14B focuses on the evolution of organization turnover. The green bars represent the number of organizations that have joined; the red bars show the organizations that have left; and the yellow plot line represents the total number of organizations that support the security-related coding in each major release. The number of companies leaving and joining stays relatively stable. There is an average turnover rate of approximately 10% for organizations that work on the current version but have decided to withdraw in the next version.

LIMITATIONS

This study, employing a case-study approach, aims to analyze the importance of top-influencing organizations in security development work within TF's software ecosystem (Runeson, 2012). The

concept of validity and threats distinguishes between different types of validity, such as construct validity, internal validity, external validity, and conclusion validity (Wohlin, 2012).

Construct validity pertains to the extent to which the operational measures utilized in a study accurately represent the intended objective of the RQs. One issue is that the collaboration was narrowed to working on the same source code files in the TF git repositories, excluding other software development activities. We also counted as collaboration only modifications of the same file. The individual developer was identified by that developer's company email address, which also determined company affiliation. Moreover, some third-party developers who were commissioned by a software supplier firm were not able to be assigned to the original company.

To obtain a comprehensive overview of developer collaboration over time, it is essential to apply and evaluate various SNA metrics, which can also provide a solid foundation for creating unique SNA visualizations; other types of visualizations and measures could be helpful. Concerning the internal validity of our study, the findings may be erroneous if we have misconstrued the SNA and other indicators. To minimize this limitation, the process was carried out with various measurements, and we implemented different methods. To ensure that the outcomes of this research paper are unbiased, we have made every effort to present the information in a clear and easily comprehensible manner.

The extent to which the findings of a study can be applied to other contexts is referred to as external validity (Yin, 2014). This research focuses on a single case study of a substantial OSS development ecosystem. Generalizing the results to other software development security activities ecosystems is possible in part. The external validity appears to be reasonable, but additional research is necessary to investigate other collaborations related to OSS security activities.

Conclusion validity concerns drawing accurate conclusions about the relationships in data and visualizations. In our research, we aimed to ensure the reproducibility of our results. We employed a combination of SNA methods, visualizations, and other measures to achieve a balanced and accurate interpretation of our findings. The researchers conducted thorough validation at each step of the case study and regularly performed assessments to maintain the integrity of the results.

CONCLUSION

This study examines coding activities in the TF OSS ecosystem through a longitudinal analysis of collaboration networks in normal and security code development. We analyzed the different structural evolutions of the normal and security code collaboration networks. The results are based on mining the TF software repository and using different SNA methods and theories related to our RQs.

The outcomes show that, with the expansion of the network size, the number of communities increases over the TF releases. That increase also increases the numbers of commits and the volume of changed source code lines. At the same time, network density is decreasing and modularity is increasing, which means that there are many growing connected clusters over time. The Simpson's Diversity Index shows that the diversity is also increasing over the project lifespan for normal and security development activities. As a result, with increasing diversity, developers, and commits, the entire TF project's productivity increases.

We focused further on examining the security code collaboration of organizations over TF OSS releases. Our findings reveal that there are six top influencing organizations contributing to the source code in the security context. These organizations play a critical role in the development and fixing of critical security issues in TF software. By analyzing the security code collaboration over the entire timeline and deepening the analysis using versions v2.5.0 and v2.12.0, we gained a better understanding of the TF OSS ecosystem evolution. We observed that while these top organizations are pivotal in addressing security vulnerabilities, their impact diminishes as the diversity of contributors increases. This pattern suggests that as more organizations and developers participate, the reliance on a few key players decreases, leading to a more distributed and resilient security collaboration network.

In addition, we investigated the turnover of developers and companies in the TF ecosystem involved in security software development activities. The number of developers leaving and joining the project increases over time with each release. In contrast to the high fluctuation rate of developers, there is an average low turnover rate of approximately 10% for organizations, meaning relative stability of collaborating companies. As a result, software safety is enhanced, continuous development is ensured, and security issues tend to be resolved more quickly.

While this study provides valuable insights into the security practices, collaboration patterns, and organizational involvement within the TF OSS ecosystem, we posit that some findings may be cautiously generalized to other OSS ecosystems. The strategies and findings discussed—particularly those related to managing security vulnerabilities, fostering effective collaboration, and leveraging organizational support—are not exclusive to TF. These insights can be adapted and applied to similar OSS projects, especially those with large-scale adoption, complex contributor networks, or significant security challenges.

These results help provide a better understanding of OSS project evolution and the collaborative patterns of software developers and organizations involved in security-related coding. They are essential results that deepen the understanding of ecosystems for practitioners as well as academics in the software engineering discipline. We have successfully integrated essential theories from SNA, open-source ecosystem development, software engineering, and software security, and have established clear theoretical references for each of these concepts.

Further steps are needed to strengthen and enhance our results. There is a need for significant case study research in the future which examines open coopetition, open-source security development activities in greater depth, and their impact on OSS ecosystems. Additionally, future research should consider conducting comparative studies involving other significant OSS projects to validate and extend the findings presented in this research.

CONFLICTS OF INTEREST

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

FUNDING STATEMENT

No funding was received for this work.

PROCESS DATES

September 13, 2024
Received: March 18, 2024, Revision: August 12, 2024, Accepted: August 27, 2024

CORRESPONDING AUTHOR

Correspondence should be addressed to Roland Robert Schreiber (Germany, roland-robert.schreiber@stud.uni-bamberg.de)

REFERENCES

Abadi, M., et. al. (2016). TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, Savannah, GA, 265–283.

Abufouda, M., & Abukwaik, H. (2017). On using network science in mining developers collaboration in software engineering: A systematic literature review. *International Journal of Data Mining & Knowledge Management Process*, *7*(5/6), 1–20. DOI: 10.5121/ijdkp.2017.7601

Alfadel, M., Costa, D. E., & Shihab, E. (2023). Empirical analysis of security vulnerabilities in Python packages. *Empirical Software Engineering*, *28*(3), 59. DOI: 10.1007/s10664-022-10278-4

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, *286*(5439), 509–512. DOI: 10.1126/science.286.5439.509 PMID: 10521342

Barbosa, O., & Alves, C. (2011). *A systematic mapping study on software ecosystems*. Citeseer.

Basili, V. R., Shull, F., & Lanubile, F. (1999). Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, *25*(4), 456–473. DOI: 10.1109/32.799939

Behfar, S. K., Turkina, E., & Burger-Helmchen, T. (2018). Knowledge management in OSS communities: Relationship between dense and sparse network structures. *International Journal of Information Management*, *38*(1), 167–174. DOI: 10.1016/j.ijinfomgt.2017.09.004

Bengtsson, M., & Kock, S. (2000). 'Coopetition ' in business networks—To cooperate and compete simultaneously. *Industrial Marketing Management*, *29*(5), 411–426. DOI: 10.1016/S0019-8501(99)00067-X

Bird, C., Nagappan, N., Gall, H., Murphy, B., & Devanbu, P. (2009). Putting it all together: Using socio-technical networks to predict failures. In *Proceedings of the 20th International Symposium on Software Reliability Engineering*, 109–119. IEEE Computer Society. DOI: 10.1109/ISSRE.2009.17

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, *2008*(10), P10008. Advance online publication. DOI: 10.1088/1742-5468/2008/10/P10008

Burt, R. S. (2009). *Structural holes: The social structure of competition*. Harvard University Press.

Çaglayan, B., & Bener, A. B. (2016). Effect of developer collaboration activity on software quality in two large scale projects. *Journal of Systems and Software*, *118*, 288–296. DOI: 10.1016/j.jss.2016.03.055

Capiluppi, A., Stol, K.-J., & Boldyreff, C. (2012). Exploring the role of commercial stakeholders in open source software evolution. In Hammouda, I. (Ed.), *Open source systems long-term sustainability* (pp. 178–200). Springer. DOI: 10.1007/978-3-642-33442-9_12

Christensen, K. K., & Liebetrau, T. (2019). A new role for "the public"? Exploring cyber security controversies in the case of WannaCry. *Intelligence and National Security*, *34*(3), 395–408. DOI: 10.1080/02684527.2019.1553704

Concas, G., Marchesi, M., Monni, C., Orrù, M., & Tonelli, R. (2017). Software quality and community structure in java software networks. *International Journal of Software Engineering and Knowledge Engineering*, *27*(07), 1063–1096. DOI: 10.1142/S0218194017500401

Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2017). A systematic mapping study of software development with GitHub. *IEEE Access : Practical Innovations, Open Solutions*, *5*, 7173–7192. DOI: 10.1109/ACCESS.2017.2682323

Crowston, K., Wei, K., Howison, J., & Wiggins, A. (2012). Free/Libre open-source software development. *ACM Computing Surveys*, *44*(2), 1–35. DOI: 10.1145/2089125.2089127

Decan, A., Mens, T., & Grosjean, P. (2019). An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empirical Software Engineering*, *24*(1), 381–416. DOI: 10.1007/s10664-017-9589-y

Delicheh, H. O., Decan, A., & Mens, T. (2024). Quantifying security issues in reusable JavaScript actions in GitHub workflows. In *Proceedings of the IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)* (pp. 692–703). https://ieeexplore.ieee.org/document/10555637

Dinghofer, K., & Hartung, F. (2020). Analysis of criteria for the selection of machine learning frameworks. In *Proceedings of the 2020 International Conference on Computing, Networking and Communications (ICNC)* (pp. 373–377). IEEE. DOI: 10.1109/ICNC47757.2020.9049650

Dong, Y., Guo, W., Chen, Y., Xing, X., Zhang, Y., & Wang, G. (2019). Towards the detection of inconsistencies in public security vulnerability reports. *USENIX Security Symposium*, 869–885.

Durumeric, Z., Li, F., Kasten, J., Amann, J., Beekman, J., Payer, M., Weaver, N., Adrian, D., Paxson, V., Bailey, M., & Halderman, J. A. (2014). The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (pp. 475–488). ACM. DOI: 10.1145/2663716.2663755

El Mezouar, M., Zhang, F., & Zou, Y. (2019). An empirical study on the teams structures in social coding using GitHub projects. *Empirical Software Engineering*, *24*(6), 3790–3823. DOI: 10.1007/s10664-019-09700-1

Fischbach, K., Gloor, P. A., & Schoder, D. (2009). Analysis of informal communication networks–a case study. *Business & Information Systems Engineering*, *1*(2), 140–149. DOI: 10.1007/s12599-008-0018-z

Gerber, A., Molefe, O., & van der Merwe, A. (2010). *Documenting open source migration processes for re-use*. ACM. https://dl.acm.org/ft_gateway.cfm?id=1899512&type=pdf

Ghafele, R., & Gibert, B. (2014). Open growth. *International Journal of Open Source Software and Processes*, *5*(1), 16–49. DOI: 10.4018/ijossp.2014010102

Gharehyazie, M., Posnett, D., Vasilescu, B., & Filkov, V. (2015). Developer initiation and social interactions in OSS: A case study of the Apache Software Foundation. *Empirical Software Engineering*, *20*(5), 1318–1353. DOI: 10.1007/s10664-014-9332-x

Granovetter, M. S. (1973). The strength of weak ties. *American Journal of Sociology*, *78*(6), 1360–1380. DOI: 10.1086/225469

Grewal, R., Lilien, G., & Mallapragada, G. (2006). Location, Location, location: How Network Embeddedness affects project success in open source systems. *Management Science*, *52*(7), 1043–1056. DOI: 10.1287/mnsc.1060.0550

Guendouz, M., Amine, A., & Hamou, R. M. (2015). Recommending relevant open source projects on GitHub using a collaborative-filtering technique. *International Journal of Open Source Software and Processes*, *6*(1), 1–16. DOI: 10.4018/IJOSSP.2015010101

Han, J., Shihab, E., Wan, Z., Deng, S., & Xia, X. (2020). What do programmers discuss about deep learning frameworks. *Empirical Software Engineering*, *25*(4), 2694–2747. DOI: 10.1007/s10664-020-09819-6

Herbold, S., Amirfallah, A., Trautsch, F., & Grabowski, J. (2021). A systematic mapping study of developer social network research. *Journal of Systems and Software*, *171*, 110802. DOI: 10.1016/j.jss.2020.110802

Herbsleb, J. D., & Mockus, A. (2003). An Empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, *29*(06), 481–494. DOI: 10.1109/TSE.2003.1205177

Hinds, D., & Lee, R. M. (2009). Communication network characteristics of open source communities. *International Journal of Open Source Software and Processes*, *1*(4), 26–48. DOI: 10.4018/jossp.2009100102

Howison, J., Inoue, K., & Crowston, K. (2006). Social dynamics of free and open source team communications. *International Federation for Information Processing Digital Library*. *Open Source Systems*, *203*, 319–330. DOI: 10.1007/0-387-34226-5_32

Howison, J., Wiggins, A., & Crowston, K. (2011). Validity issues in the use of social network analysis with digital trace data. *Journal of the Association for Information Systems*, *12*(12), 2. DOI: 10.17705/1jais.00282

Jabangwe, R., Kuusinen, K., Riisom, K. R., Hubel, M. S., Alradhi, H. M., & Nielsen, N. B. (2018). Challenges and solutions for addressing software security in agile software development: A literature review and rigor and relevance assessment. [IJSSSP]. *International Journal of Systems and Software Security and Protection*, *9*(1), 1–17. DOI: 10.4018/IJSSSP.2018010101

Jeong, G., Kim, S., & Zimmermann, T. (2009). Improving bug triage with bug tossing graphs. In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering* (pp. 111–120). ACM. DOI: 10.1145/1595696.1595715

Jermakovics, A., Sillitti, A., & Succi, G. (2013). Exploring collaboration networks in open-source projects. In Petrinja, E., Succi, G., El Ioini, N., & Sillitti, A. (Eds.), *Open source software: Quality verification* (pp. 97–108). Springer., DOI: 10.1007/978-3-642-38928-3_7

Jiang, Q., Lee, Y. C., Davis, J. G., & Zomaya, A. Y. (2018). *Diversity, productivity, and growth of open source developer communities*. CoRR 2018, abs/1809.03725.

Joblin, M., Apel, S., Hunsen, C., & Mauerer, W. (2017a). Classifying developers into core and peripheral: An empirical study on count and network metrics. *Proceedings of ICSE*, *2017*, 164–174. DOI: 10.1109/ICSE.2017.23

Joblin, M., Apel, S., & Mauerer, W. (2017b). Evolutionary trends of developer coordination: A network approach. *Empirical Software Engineering*, *22*(4), 2050–2094. DOI: 10.1007/s10664-016-9478-9

Kappelhoff, P. (1987). Cliquenanalyse. Die Bestimmung von intern verbundenen Teilgruppen in Netzwerken. In Pappi, F. U. (Ed.), *Techniken Der Empirischen Sozialforschung–Methoden Der Netzwerkanalyse* (pp. 39–63). Pieper.

Lee, C., & Cunningham, P. (2013). Community detection: Effective evaluation on large social networks. *Journal of Complex Networks*, *2*(1), cnt012.

Manikas, K., & Hansen, K. M. (2013). Software ecosystems–A systematic literature review. *Journal of Systems and Software*, *86*(5), 1294–1306. DOI: 10.1016/j.jss.2012.12.026

McClean, K., Greer, D., & Jurek-Loughrey, A. (2021). Social network analysis of open source software: A review and categorisation. *Information and Software Technology*, *130*, 106442. DOI: 10.1016/j.infsof.2020.106442

Meneely, A., Tejeda, A. C. R., Spates, B., Trudeau, S., Neuberger, D., Whitlock, K., Ketant, C., & Davis, K. (2014). An empirical investigation of socio-technical code review metrics and security vulnerabilities. In Lanubile, F. (Ed.), *Proceedings of the 6th International Workshop on Social Software Engineering* (pp. 37–44). ACM. DOI: 10.1145/2661685.2661687

Meneely, A., & Williams, L. (2009). Secure open source collaboration. In Al-Shaer, E., Jha, S., & Keromytis, A. D. (Eds.), *Proceedings of the 16th ACM conference on Computer and communications security* (p. 453). ACM.

Meneely, A., & Williams, L. (2010). Strengthening the empirical analysis of the relationship between Linus' Law and software security. In Succi, G. (Ed.), *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (p. 1). ACM. DOI: 10.1145/1852786.1852798

Meneely, A., Williams, L., Snipes, W., & Osborne, J. (2008). Predicting failures with developer networks and social network analysis. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 13–23). ACM. DOI: 10.1145/1453101.1453106

Morgan, L., & Finnegan, P. (2014). Beyond free software: An exploration of the business value of strategic open source. *The Journal of Strategic Information Systems*, *23*(3), 226–238. DOI: 10.1016/j.jsis.2014.07.001

Nguyen Duc, A., Cruzes, D. S., Hanssen, G. K., Snarby, T., & Abrahamsson, P. (2017). Coopetition of software firms in open source software ecosystems. In Ojala, A., & Olsson, H. H. (Eds.), *Software business* (pp. 146–160). Springer. DOI: 10.1007/978-3-319-69191-6_10

Oliveira, E., Conte, T., Cristo, M., & Valentim, N. (2018). Influence factors in software productivity—A tertiary literature review. *International Journal of Software Engineering and Knowledge Engineering, 28*(11n12), 1795–1810.

Oliveira, G. P., Moura, A. F. C., Batista, N. A., Brandão, M. A., Hora, A., & Moro, M. M. (2023). How do developers collaborate? Investigating GitHub heterogeneous networks. *Software Quality Journal*, *31*(1), 211–241. DOI: 10.1007/s11219-022-09598-x

Ouriques, R. A. B., Wnuk, K., Gorschek, T., & Svensson, R. B. (2019). Knowledge management strategies and processes in agile software development: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, *29*(03), 345–380. DOI: 10.1142/S0218194019500153

Peng, J., Zhang, G., & Chiu, C.-H. (2022). An empirical investigation on vulnerability for software companies. *International Journal of Systems and Software Security and Protection*, *13*(1), 1–15. DOI: 10.4018/IJSSSP.304894

Rashid, E., & Prakash, M. (2022). An empirical analysis of inferences from commit, fork, and branch rates of top GitHub projects. *International Journal of Open Source Software and Processes*, *13*(1), 1–16. DOI: 10.4018/IJOSSP.300751

Runeson, P. (2012). *Case study research in software engineering: Guidelines and examples* (1st ed.). Wiley., https://onlinelibrary.wiley.com/doi/book/10.1002/9781118181034 DOI: 10.1002/9781118181034

Sarker, S., Ahuja, M., Sarker, S., & Kirkeby, S. (2011). The role of communication and trust in global virtual teams: A social network perspective. *Journal of Management Information Systems*, *28*(1), 273–309. DOI: 10.2753/MIS0742-1222280109

Schenk, M. (1995). *Soziale Netzwerke und Massenmedien: Untersuchungen zum Einfluss der persönlichen Kommunikation*. Mohr Siebeck.

Schreiber, R. R. (2023). Organizational influencers in open-source software projects. *International Journal of Open Source Software and Processes*, *14*(1), 1–20. DOI: 10.4018/IJOSSP.318400

Schreiber, R. R., & Zylka, M. P. (2020). Social network analysis in software development projects: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, *30*(03), 321–362. DOI: 10.1142/S021819402050014X

Scott, J. (2013). *Social network analysis* (3rd ed.). SAGE. DOI: 10.4135/9781529682557

Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, *52*(7), 1000–1014. DOI: 10.1287/mnsc.1060.0553

Shin, Y., Meneely, A., Williams, L., & Osborne, J. A. (2011). Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. *IEEE Transactions on Software Engineering*, *37*(6), 772–787. DOI: 10.1109/TSE.2010.81

Singh, P. V., Tan, Y., & Mookerjee, V. (2011). Network effects: The influence of structural capital on open source project success. *Management Information Systems Quarterly*, *35*(4), 813–829. DOI: 10.2307/41409962

Šmite, D., Moe, N. B., Šāblis, A., & Wohlin, C. (2017). Software teams and their knowledge networks in large-scale software development. *Information and Software Technology*, *86*, 71–86. DOI: 10.1016/j.infsof.2017.01.003

Squire, M. (2014). *Forge++: The changing landscape of FLOSS development*. In *Proceedings of the Annual Hawaii International Conference on System Sciences* (pp. 3266-3275). DOI: 10.1109/HICSS.2014.405

Sureka, A., Goyal, A., & Rastogi, A. (2011). Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis. In *Proceedings of the 4th India Software Engineering Conference (ISEC)* (pp. 195–204). ACM. DOI: 10.1145/1953355.1953381

Tahaei, M., & Vaniea, K. (2019). A Survey on developer-centred security. In *Proceedings of the 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (pp. 129–138). IEEE. DOI: 10.1109/EuroSPW.2019.00021

Tensorflow. (2024). *Why Tensorflow—Case studies*. https://www.tensorflow.org/about/case-studies

Trabelsi, S., Plate, H., Abida, A., Aoun, M. M. B., Zouaoui, A., Missaoui, C., Gharbi, S., & Ayari, A. (2015). Mining social networks for software vulnerabilities monitoring. In *Proceedings of the 7th International Conference on New Technologies, Mobility and Security (NTMS)* (pp. 1–7). IEEE. DOI: 10.1109/NTMS.2015.7266506

Uzzi, B. (1997). Social structure and competition in interfirm networks: The paradox of embeddedness. *Administrative Science Quarterly*, *42*(1), 37–69. DOI: 10.2307/2393808

Wang, S., & Nagappan, N. (2021). Characterizing and understanding software developer networks in security development. In *Proceedings of the IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)* (pp. 534–545). DOI: 10.1109/ISSRE52982.2021.00061

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge University Press. DOI: 10.1017/CBO9780511815478

Wohlin, C. (2012). *Experimentation in software engineering*. Springer. DOI: 10.1007/978-3-642-29044-2

Wolf, T., Schroter, A., Damian, D., & Nguyen, T. (2009). Predicting build failures using social network analysis on developer communication. In *Proceedings of the IEEE 31st International Conference on Software Engineering* (pp. 1–11). IEEE. DOI: 10.1109/ICSE.2009.5070503

Wu, S., Song, W., Huang, K., Chen, B., & Peng, X. (2024). Identifying affected libraries and their ecosystems for open source software vulnerabilities. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (pp. 1–12). DOI: 10.1145/3597503.3639582

Yin, R. K. (2014). *Case study research: Design and methods* (5th ed.). Sage.

Yu, Y., Wang, H., Yin, G., & Wang, T. (2016). Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment? *Information and Software Technology*, *74*, 204–218. DOI: 10.1016/j.infsof.2016.01.004

Zafar, H. (2022). Critical success factors for an effective security risk management program. [IJSSSP]. *International Journal of Systems and Software Security and Protection*, *13*(1), 1–26. DOI: 10.4018/IJSSSP.315765

Zhang, W., Nie, L., Jiang, H., Chen, Z., & Liu, J. (2014). Developer social networks in software engineering: Construction, analysis, and applications. *Science China. Information Sciences*, *57*(12), 1–23. DOI: 10.1007/s11432-014-5221-6

Zhao, Y., Liang, R., Chen, X., & Zou, J. (2021). Evaluation indicators for open-source software: A review. *Cybersecurity*, *4*(1), 20. DOI: 10.1186/s42400-021-00084-8

Zhou, M., Chen, Q., Mockus, A., & Wu, F. (2017). On the scalability of Linux kernel maintainers' work. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (pp. 27–37). ACM. DOI: 10.1145/3106237.3106287

Zhou, Y., & Sharma, A. (2017). Automated identification of security issues from commit messages and bug reports. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (pp. 914–919). ACM. DOI: 10.1145/3106237.3117771

Zimmermann, M., Staicu, C.-A., Tenny, C., & Pradel, M. (2019). Small world with high risks: A study of security threats in the npm ecosystem. In *Proceedings of the USENIX Security Symposium* (pp. 17). USENIX security.

Zimmermann, T., Nagappan, N., & Williams, L. (2010). Searching for a needle in a haystack: Predicting security vulnerabilities for windows vista. In P*roceedings of the Third International Conference on Software Testing, Verification and Validation* (pp. 421–428). IEEE.