On Categorizing Open Source Software Security Vulnerability Reporting Mechanisms on GitHub

Sushawapak Kancharoendee[†], Thanat Phichitphanphong[†], Chanikarn Jongyingyos[†], Brittany Reid[*], Raula Gaikovina Kula[§], Morakot Choetkiertikul[†], Chaiyong Ragkhitwetsagul[†], Thanwadee Sunetnanta[†]
[†]*Faculty of Information and Communication Technology, Mahidol University, Thailand*
[*]*Graduate School of Science and Technology, Nara Institute of Science and Technology (NAIST), Japan*
[§]*Graduate School of Information Science and Technology, Osaka University, Japan*
{sushawapak.kan, thanat.phi, chanikarn.jon}@student.mahidol.ac.th, brittany.reid@naist.ac.jp,
raula-k@ist.osaka-u.ac.jp, {morakot.cho, chaiyong.rag, thanwadee.sun}@mahidol.ac.th

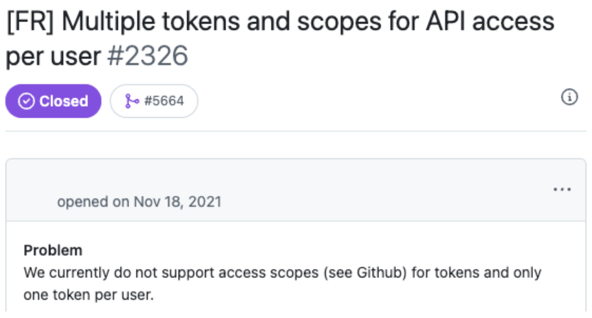
***Abstract*—Open-source projects are essential to software development, but publicly disclosing vulnerabilities without fixes increases the risk of exploitation. The Open Source Security Foundation (OpenSSF) addresses this issue by promoting robust security policies to enhance project security. Current research reveals that many projects perform poorly on OpenSSF criteria, indicating a need for stronger security practices and underscoring the value of SECURITY.md files for structured vulnerability reporting. This study aims to provide recommendations for improving security policies. By examining 679 open-source projects, we find that email is still the main source of reporting. Furthermore, we find that projects without SECURITY.md files tend to be less secure (lower OpenSSF scores). Our analysis also indicates that, although many maintainers encourage private reporting methods, some contributors continue to disclose vulnerabilities publicly, bypassing established protocols. The results from this preliminary study pave the way for understanding how developers react and communicate a potential security threat. Future challenges include understanding the impact and effectiveness of these mechanisms and what factors may influence how the security threat is addressed.**

***Index Terms*—open-source, security policy, OpenSSF**

I. Introduction

Open-source software has become essential in modern development, enabling developers to reuse and modify code. However, with its growing use, the risk of vulnerabilities has increased [1]. As security threats rise, developers' security practices are essential [2]. Effective management is key, especially for open-source projects, where developers' responses to vulnerabilities are vital [3]. Publicly disclosing vulnerabilities without fixes heightens the risk of exploitation. To address this, platforms like GitHub provide security measures, such as a `SECURITY.md` file [4] for reporting vulnerabilities and GitHub Advisories [5] for documenting security issues.

To address bugs or discuss their projects, developers use GitHub Issues, a task-tracking feature available to all GitHub users for open-source projects. However, the traditional practice of reporting security-related issues through GitHub Issues can pose risks of vulnerability being known by attackers, as it makes vulnerabilities publicly known before they are fixed, giving attackers more time to exploit them. For this reason, some GitHub projects request that developers avoid reporting


[FR] Multiple tokens and scopes for API access per user #2326

Closed · #5664

opened on Nov 18, 2021

Problem
We currently do not support access scopes (see Github) for tokens and only one token per user.

Fig. 1. A security issue created in open source project

vulnerabilities through Issues. For example, Figure 1 shows a security issue in an open-source project posted publicly as a GitHub issue, detailing a problem with token handling. By publically posting this issue, the unaddressed token leak could allow attackers to perform unauthorized actions beyond the intended scope before the developers can fix the issue.

Recent studies showed that only a small proportion of open-source projects have defined security policies [6]. However, defining security policy in the projects is considered an important practice for developers to improve the project's security [2]. These highlight a gap in the security practices of the open-source projects. However, no study has focused on analyzing the characteristics and effectiveness of the security policies, including the extent to which developers' practices align with the provided guidelines. The related work in this study examines security policies in open-source projects on GitHub. Ayala et al. [6] analyzed popular GitHub repositories, finding that only a small portion had implemented security policies, with many lacking structured security measures. This highlights the importance of workflows and security policies in open-source projects. The GitHub security policy feature guides vulnerability reporting, while security issues serve as a feature for bug tracking and project discussions. Bühlmann et al. [7] studied security issues, noting that they comprise a small yet growing portion of issues and require more discussion time than non-security issues. Zahan et al. [8] used the OpenSSF Scorecard to assess security practices in the npm and PyPI ecosystems. The study identified gaps in areas such as Code Review, Maintenance, Licensing, Branch Protection, and Security Policy. The study also recommended practices for enhancing such areas. Additionally, a relationship between security practices and vulnerability levels was identified [2].

Despite increased attention on security policies for open-source projects, no research has yet examined the actual content of these policies. Our study addresses this gap by analyzing specific elements within security policies to propose guidelines for improvement. By identifying practices that could improve OpenSSF Scorecard scores if properly implemented, we set the stage for further guidance on defining effective security policies.

This motivated us to investigate the characteristics of security policies in open-source projects, aiming to provide software practitioners with guidelines for enhancing security policies. This paper presents our preliminary investigation into GitHub projects' security policies, focusing on two main aspects. First, we examine how developers report vulnerabilities by comparing the practices outlined in security policies with actual developer actions. To do this, we analyze the content of security policies to identify vulnerability reporting processes and manually review developer practices. Second, we conduct a preliminary assessment of security practices' quality using the Open Source Security Foundation's (OpenSSF) Scorecard [9]. Through a comparative study, we evaluate security practices in projects with and without defined security policies to determine whether the presence of a security policy strengthens overall security practices. Thus, we define the following research questions for our preliminary investigation:

1) **RQ1:** What are the reporting mechanisms in security policies?
2) **RQ2:** Do the developer's practices align with the security policy?
3) **RQ3:** Do projects with a security policy differ in OpenSSF Scorecard scores compared to those without one?

To address our research questions, we analyzed 679 PyPi packages hosted on GitHub. We focused on PyPI ecosystem for Python language since Python is a widely used programming language in software development, as indicated in the 2024 IEEE Spectrum survey [10]. We examined how developers report security vulnerabilities by reviewing security policy instructions and analyzing security issues in these projects. We then compared Scorecard scores between projects with and without a defined security policy. Our study shows that most PyPi package maintainers recognize the need for private channels like email or GitHub advisories to report vulnerabilities, as outlined in many security policies. However, external contributors often bypass these policies and submit security issues publicly. Projects with established security policies show notable improvements in the Scorecard's Dependency Update Tool and Maintained Practices scores. This underscores the importance of improving public communication strategies or refining security policies to enhance developer awareness and adherence to security practices.

II. Background

This section covers background and related work, starting with an overview of GitHub security policies and the OpenSSF Scorecard for evaluating security practices in open-source projects.

A. Security Policies in GitHub Repositories

GitHub provides a feature for creating a security policy within a repository through a dedicated `SECURITY.md` file. This file guides software practitioners, such as contributors and users, on handling security concerns, including the appropriate channels for reporting vulnerabilities in a project. Maintainers can add the security policy to the repository's root, `doc`, or `.github` folder. When creating a security policy using GitHub's security feature, a guideline suggests including key details to improve its usefulness.[1] These details include the *Supported Versions* section, listing the project versions that receive security updates, and the *Reporting a Vulnerability* section, which provides instructions for reporting vulnerabilities. This section can also specify communication channels for updates on reported vulnerabilities, outline steps taken upon vulnerability acceptance or rejection, and include additional information about the repository's security practices.

GitHub also provides features that enable users to report vulnerabilities, through the GitHub Advisory (GHSA) [5]. GitHub Advisory is a database that publishes information on security vulnerabilities that impact open-source projects. It includes security-related details, such as severity, affected versions, and recommended fixes.

B. OpenSSF Scorecard

The OpenSSF Scorecard is an automated tool developed by the Open Source Security Foundation (OpenSSF) to assess security practices in open-source software repositories [11]. The Scorecard provides a security practices score for each project, based on 18 criteria that collectively reflect the robustness of security practices in maintaining a project repository. These scores help software practitioners identify and address potential security risks in their projects, offering insights to improve overall security practices.

Each metric in the OpenSSF Scorecard is classified into one of four risk categories: Critical, High, Medium, and Low, based on its potential security implications. Metrics are scored from 0 to 10, reflecting how closely the project aligns with security best practices. If Scorecard cannot verify a particular practice, it assigns a score of -1, which indicates inconclusive evidence or a possible runtime error. This designation ensures that such cases do not negatively impact the overall project score. For example, the Scorecard evaluates metrics like Branch Protection, which checks if protective measures are in place for critical branches, and Dependency Update Tool,

[1]https://docs.github.com/en/code-security/getting-started/adding-a-security-policy-to-your-repository

assessing whether tools are used to keep dependencies up-to-date, which helps reduce vulnerability risks. Another metric, License, confirms whether the project has a well-defined, open-source license, contributing to transparency and legal security. These checks help developers understand security gaps and guide improvements.

III. Methodology

In this section, we explain our methodology, detailing the data collection process and our approach to answering the research questions.

A. Overview

Figure 2 shows an overview of the study workflow. To answer research questions, we collect PyPI packages from the security advisories in the GitHub advisory database. Our methodology examines how developers report vulnerabilities as outlined in security policies and compares this with developers' actual practices. We manually categorize the instructions provided in these policies and then identify any security issues that do not conform to them. To highlight the importance of effective security policies, we assess security practices using the OpenSSF Scorecard for both packages with and without a security policy. By comparing the scores, we identify specific practices that show significant improvements when a security policy is implemented.

B. Data collection

Our study incorporates three data sets: (1) GitHub repositories of the PyPi packages that appear in the GitHub advisory database, (2) security issues from GitHub repositories, and (3) security practice scores from the OpenSSF Scorecard. We gathered GitHub repositories of the PyPi packages with at least one GitHub advisory record from GitHub Advisory Database [5], which includes vulnerability advisories published before May 2024. The initial dataset contains 2,982 security advisories for PyPi packages. To ensure the accuracy of the collection process, we verified key repository values of the advisory, `source_code_location`, which links to the reported repositories.

In the GitHub security-related issues collection, we utilized WordNet [12] to identify synonyms and related words based on a core set of terms: ["vulnerability", "security", "risk", "CVE", "CWE"]. We collected the issues that have the labels containing these security-related keywords. The dataset of security issues includes the metadata of the issues, which comprises links to the issues, titles, labels with security keywords, submitter information, and submission dates. Lastly, we used the OpenSSF Scorecard tool to assess the security practices of collected repositories. The dataset includes scores for 18 security practices along with an overall security score, with each practice score ranging from -1 to 10.

C. Data processing

We began by performing data filtering to ensure the quality and relevance of the repositories analyzed. First, we excluded the OpenSSF Scorecard criterion rated as -1, as this rating indicates that the tool was unable to generate a score due to insufficient data or an internal error during the assessment process. Additionally, we excluded one repository that contained over 1,000 issue reports, treating it as an outlier due to its atypically high volume of issues. Following this initial filtering, we categorized the repositories into two groups. The first group included repositories with a defined security policy, identified by the presence of `SECURITY.md`, `security.md`, or `security.rst` files in either the top-level directory or the `.github` folder.

The second group comprised repositories without any security policy files. This classification allowed us to compare security practices between projects with and without explicit security policies in place. Note that the OpenSSF Scorecard considers the presence of a security policy under the "Security Policy" criterion. To ensure a fair comparison, we excluded the "Security Policy" score from our analysis, as the existence of a security policy directly impacts the overall OpenSSF score. In conclusion, our dataset comprises 679 repositories, with 303 containing the security policy and 376 without one.

D. Data Analysis

To address RQ1, we analyze the content specified in security policies. GitHub security policies typically guide developers on reporting vulnerabilities and security issues, detailing the reporting mechanisms available. The classification process was conducted collaboratively among the authors. We manually classified these mechanisms by examining the content to identify the specific methods or channels provided for issue reporting one by one. When discovering new reporting mechanisms, we expanded the methods and conducted the classification iteratively. To ensure consistency and minimize the influence of individual biases, we calculated Cohen's Kappa [13] for the reliability of the manual analysis, achieving an agreement score of 0.9. Our analysis revealed four distinct reporting mechanisms: reporting through email, external links (e.g., proprietary issue tracking systems), GitHub advisories, and GitHub Issue creation. The definition and example practices are presented in Table I. This classification allowed us to identify the most commonly used reporting methods.

For RQ2, we examine repositories that specify reporting methods such as Email, External Links, and GitHub Advisory in their security policies. Specifically, we assess whether, after the declaration of these policies, any security-related issues were reported as GitHub Issues, which would be non-compliant with the stated reporting methods.

To address RQ3, we perform a statistical comparison of OpenSSF Scorecard scores between repositories with a security policy and those without one.

IV. Results

This section provides a detailed account of the results for each research question, along with a discussion that emphasizes the significance of our findings.

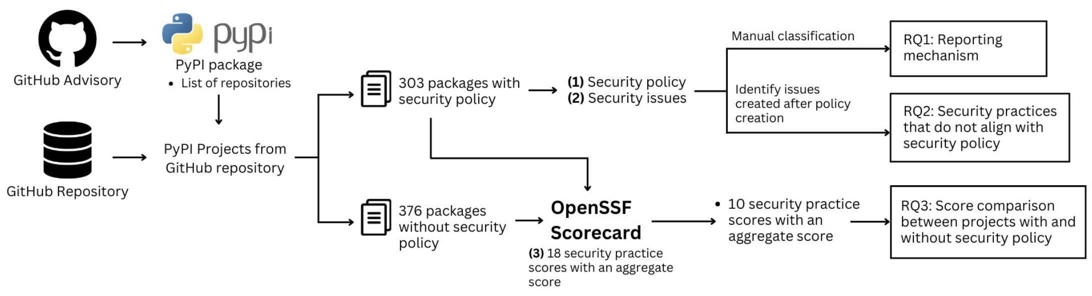

GitHub Advisory
PyPI package
List of repositories
GitHub Repository
PyPI Projects from GitHub repository
303 packages with security policy
376 packages without security policy
(1) Security policy
(2) Security issues
OpenSSF Scorecard
(3) 18 security practice scores with an aggregate score
Manual classification
Identify issues created after policy creation
10 security practice scores with an aggregate score
RQ1: Reporting mechanism
RQ2: Security practices that do not align with security policy
RQ3: Score comparison between projects with and without security policy

Fig. 2. Our research methodology

TABLE I
REPORTING MECHANISM DEFINITION

Reporting Mechanism	Definition	Example
Email	The project provides the process to report the vulnerabilities through sending an email	If you discover a security vulnerability or would like to report a security issue privately and securely, please email us at [redacted]
GitHub Advisory	The project provides the process to report the vulnerabilities by submitting a GitHub Advisory	To report a vulnerability head over to the Security Advisories page and click on "New draft security advisory".
GitHub Issue	The project provides the process to report the vulnerabilities by creating an issue	Please report a vulnerability as an issue; our team will evaluate it and address it in the proper time.
External link	The project provides another platform (e.g., bug bounty program) or external links for further information about the security policy	Please report sensitive security issues via [redacted] bug-bounty program by following this instruction, rather than GitHub.
Not mentioned	The project did not provide the reporting mechanism in the security policy	

A. *RQ1: What are the reporting mechanisms in security policies?*

Figure 3 shows a Venn diagram of the reporting mechanisms defined in security policies. Our analysis of 302 GitHub repositories from PyPi shows that most repositories use email (41.06%) as the preferred reporting method, followed by external links to other platforms (21.52%) and the GitHub Advisories feature (12.91%). Notably, 62 repositories (20.53%) provide more than one reporting mechanism. Only 0.99% use GitHub Issues, and 2.98% do not specify any reporting method. We categorize email, external links, or GitHub advisories as private channels for reporting security vulnerabilities in repositories. In answering RQ1, we found that most security policies define the private channels as reporting mechanisms, as the results show that 285 repositories (94.37%) define these channels in the security policies. This suggests that most

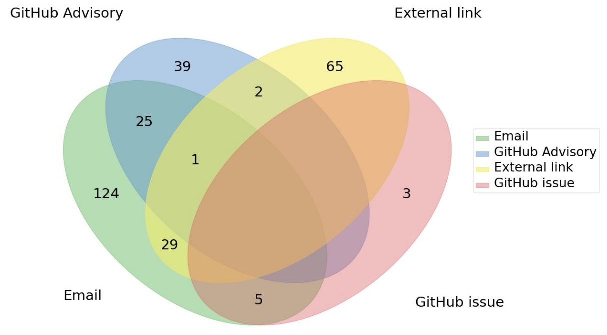


Fig. 3. The Reporting Mechanism Defined In Security Policy

project maintainers are aware of the risks of publicly disclosed security vulnerabilities and take steps to mitigate them.

RQ1 Summary: Our findings indicate that most project maintainers are aware of publicly disclosed security, since most security policy reporting mechanisms are private communication channels.

B. *RQ2: Do the developer's practices align with the security policy?*

We investigate the alignment between reporting mechanisms defined in security policies and actual practices. In repositories specifying Email, External Link, or GitHub Advisory as reporting channels, we found a significant decrease in security-related issues reported through GitHub Issues, suggesting that practitioners are generally adhering to the defined reporting mechanisms. However, 787 security-related issues were still reported through GitHub Issues across 58 repositories after private reporting methods were established, indicating non-compliance with the policies.

Table II shows that of 787 non-compliance issues, these are primarily occurring in repositories using Email as the reporting method. We also investigated the roles of submitters of these security issues identified by the `author_association` attribute of the analyzed repositories. This attribute includes four categories: *OWNER*, *CONTRIBUTOR*, and *MEMBER*, while *NONE* represents external contributors with no formal association to the project. The results indicate that most developers follow the defined reporting mechanism, as the results show a small number of security issues do not comply with the security policy. However, for these noncompliant security issues, the majority of them were created by external contributors (i.e., the `author_association` is *NONE*), who may either be unaware of the policy or intentionally disregard it.

TABLE II
NON-COMPLIANCE IN SECURITY-RELATED ISSUES BASED ON REPORTING MECHANISMS AND REPORTER ROLES

Reporting mechanism	**%**	**Submitter**	**%**
Email	39.52	NONE	44.34
External link	26.30	MEMBER	25.67
GitHub (GH) Advisory	21.09	CONTRIBUTOR	23.13
Email and External link	9.66	COLLABORATOR	6.73
Not mentioned	1.78	OWNER	0.13
Email & GH Advisory	0.76		
Email & GH issues	0.76		
GH Advisory & External link	0.13		

RQ2 Summary: Most developers follow the defined reporting mechanism, as a small number of security issues that do not comply with the policy are created by external contributors.

C. RQ3: Do projects with a security policy differ in OpenSSF Scorecard scores compared to those without one?

We used the OpenSSF Scorecard tool to assess 302 repositories with a security policy and 376 without one across 10 measurable security practices from OpenSSF Scorecard: Binary Artifacts, Branch Protection, CII Best Practices, Contributors, Dependency Update Tool, Fuzzing, License, Maintained, SAST, and Vulnerabilities.

Table III shows the comparison results. Repositories with a security policy in place show higher adherence to security practices across most metrics. Notably, the mean scores for eight practices such as Branch Protection, CII Best Practices, and Maintained, are significantly higher in repositories with a security policy. This is shown by a higher mean score with *p-value* less than 0.001 for the repositories with the security policy. This indicates that repositories with security policies are more proactive about implementing strong security practices.

RQ3 Summary: Repositories with security policies are more proactive in implementing security practices, as evidenced by the higher security practices score.

TABLE III
COMPARISON OF OPENSSF SCORECARD SCORES BETWEEN REPOSITORIES WITH AND WITHOUT SECURITY POLICIES

Security Practice	**With Policy**		**Without Policy**		**p-value**
	Mean	**SD**	**Mean**	**SD**	
Aggregate Score*	5.93	1.14	3.95	1.29	<0.001
Binary-Artifacts	9.50	1.73	9.64	1.61	0.26
Branch-Protection*	3.53	2.94	1.69	2.57	<0.001
CII-Best-Practices*	0.29	1.21	0.01	0.1	<0.001
Contributors*	9.48	1.97	8.25	3.37	<0.001
Dependency-Update-Tool*	7.02	4.58	3.19	4.67	<0.001
Fuzzing*	1.75	3.81	0.61	2.40	<0.001
License	9.45	1.71	9.24	2.29	0.20
Maintained*	7.49	4.12	4.04	4.62	<0.001
SAST*	3.57	4.44	1.06	2.84	<0.001
Vulnerabilities	7.45	3.9	7.43	3.95	0.95

* indicates practices where the score for repositories with a security policy is significantly higher than those without.

V. CHALLENGS AND FUTURE OUTLOOK

Open-source projects play a vital role in software development, but sharing vulnerabilities before fixes are available poses a significant risk of exploitation. Consequently, open-source projects often establish their own security policies to address these concerns. Our study examined security reporting mechanisms in PyPi packages on GitHub, focusing on the alignment between security policies and actual reporting practices. Interestingly, email remains the primary means of reporting security issues. Our analysis reveals frequent non-compliance in reporting security issues, suggesting that current reporting methods may not be effective for all contributors. Effective security policies must cover key practices such as secure coding, vulnerability management, data handling, and access control. A crucial takeaway is that email communication requires robust security measures to protect sensitive information. The OpenSSF Scorecard analysis also shows that projects with well-defined security policies tend to adhere more closely to key practices, underscoring the positive impact of effective security policies on project security. At this stage, we observe various methods of reporting vulnerabilities, which warrants further investigation into the reasons behind these choices and their pros and cons.

Future work could focus on understanding the communities of open-source projects and their practices and compliance that they enforce within their communities. This includes examining software library ecosystems, where regulations in an open-source context remain unclear. Examining security policy and identifying best practices for security policies across diverse open-source ecosystems is also needed. This would involve tailoring security policy elements to varying project characteristics, such as project size, complexity, and contributor base, to provide actionable recommendations for software practitioners.

Moreover, in the era of AI, further research is needed to explore automated methods for detecting non-compliance with security policies and investigate communication strategies to enhance adherence among external contributors, such as tools for preventing vulnerabilities from being discussed publicly. Such agents would be critical in quickly raising awareness and mitigating potential threats while identifying false positives. Ultimately, these challenges and future directions offer new opportunities to keep the open-source community secure from security threats with rapid mitigation.

VI. ACKNOWLEDGMENT

This work is supported by JSPS KAKENHI JP20H05706, JP23K28065, and Daiichi-Sankyo “Habataku” Support Program for the Next Generation of Researchers, NAIST Senju Monju Project.

References

[1] E. S. Mahmoud Alfadel, Diego Elias Costa, “Empirical Analysis of Security Vulnerabilities in Python Packages,” *Empirical Software Engineering (ESME)*, 2023.

[2] N. Zahan, S. Shohan, D. Harris, and L. Williams, “Do software security practices yield fewer vulnerabilities?,” in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 292–303, 2023.

[3] G. A. A. Prana, A. Sharma, L. K. Shar, D. Foo, A. E. Santosa, A. Sharma, and D. Lo, “Out of sight, out of mind? how vulnerable dependencies affect open-source projects,” *Empirical Softw. Engg.*, vol. 26, July 2021.

[4] “Adding a security policy to your repository.” https://docs.github.com/en/code-security/getting-started/adding-a-security-policy-to-your-repository. (Accessed on 26/06/2024).

[5] “Github advisory database.” https://github.com/advisories. (Accessed on 26/06/2024).

[6] J. Ayala and J. Garcia, “An empirical study on workflows and security policies in popular github repositories,” in *2023 IEEE/ACM 1st International Workshop on Software Vulnerability (SVM)*, pp. 6–9, 2023.

[7] N. Bühlmann and M. Ghafari, “How do developers deal with security issue reports on github?,” in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, SAC ’22, (New York, NY, USA), p. 1580–1589, Association for Computing Machinery, 2022.

[8] N. Zahan, P. Kanakiya, B. Hambleton, S. Shohan, and L. Williams, “Openssf scorecard: On the path toward ecosystem-wide automated security metrics,” *IEEE Security & Privacy*, vol. 21, no. 6, pp. 76–88, 2023.

[9] “Opensff scorecard.” https://openssf.org/projects/scorecard/, 2015. (Accessed on 26/07/2024).

[10] “The top programming languages 2024 typescript and rust are among the rising stars.” https://spectrum.ieee.org/quantum-technology-sydney. (Accessed on 18/11/2024).

[11] “Opensff.” https://openssf.org/, 2015. (Accessed on 26/07/2024).

[12] “Sample usage for wordnet.” https://www.nltk.org/howto/wordnet.html. (Accessed on 08/09/2024).

[13] J. Cohen, “A coefficient of agreement for nominal scales,” *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.