# In War and Peace: The Impact of World Politics on Software Ecosystems

Raula Gaikovina Kula
Nara Institute of Science and Technology, Japan
raula-k@is.naist.jp

Christoph Treude
University of Melbourne, Australia
christoph.treude@unimelb.edu.au

## ABSTRACT

Reliance on third-party libraries is now commonplace in contemporary software engineering. Being open source in nature, these libraries should advocate for a world where the freedoms and opportunities of open source software can be enjoyed by all. Yet, there is a growing concern related to maintainers using their influence to make political stances (i.e., referred to as protestware). In this paper, we reflect on the impact of world politics on software ecosystems, especially in the context of the ongoing War in Ukraine. We show three cases where world politics has had an impact on a software ecosystem, and how these incidents may result in either benign or malignant consequences. We further point to specific opportunities for research, and conclude with a research agenda with ten research questions to guide future research directions.

## CCS CONCEPTS

**• Software and its engineering → Software libraries and repositories**; **• Social and professional topics**; **• Security and privacy → Social aspects of security and privacy**;

## KEYWORDS

Libraries, Software Ecosystem, Protestware, Supply Chain Attacks

**ACM Reference Format:**
Raula Gaikovina Kula and Christoph Treude. 2022. In War and Peace: The Impact of World Politics on Software Ecosystems. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3540250.3560882

## 1 INTRODUCTION

Contemporary software is not built in isolation. With the emergence of third-party libraries, and massive software repositories, now more than ever developers are able to quickly adopt functionality into their applications, avoiding the time-consuming and error-prone task of writing and testing from scratch.

The success of these libraries is made possible through the concept of Open Source Software (OSS). OSS allows software to be free for anyone to use, which is incorporated into everything from independent projects to mainstream, proprietary consumer software. Being OSS, libraries are typically maintained by one or a handful of volunteers, and they benefit from having further volunteer contributors, as generally the developers in a software ecosystem should be able to help each other by pointing out issues and contributing new features. Libraries harness the power of distributed peer review and transparency of process, promising higher quality, better reliability, greater flexibility, and lower cost. Projects build reputation over time, with developers gaining trust in using these libraries. Each ecosystem has their own culture of maintainers, who are empowered to approve and publish contributed code changes. Evidence of the impact of libraries is seen with the growing size of NPM with more than one million packages [23], for example, and the growing support for these ecosystems, e.g., GitHub's 2020 acquisition of NPM.[1]

Problems arise when a maintainer feels empowered to sabotage their own projects, thus weaponizing their library as protestware, e.g., with the intention to make users of their library aware of some political stance, or situation. Responses from practitioners to such protestware have been mixed. According to a developer blog that was later republished by the IEEE Computer Society [12], practitioners have expressed their concerns, stating that *"It's ill-considered and user-hostile, and can trivially go wrong. Weaponizing open source to inject malware, no matter how well intentioned, is still injecting malware".* Others question whether this is Open Source anymore, as a OSS licence clearly states that there should be "*No Discrimination Against Persons or Groups. The license must not discriminate against any person or group of persons*" [18]. The official statement from the Open Source Initiative (OSI) community lays the responsibility to maintainers, stating that "*Protest is an important element of free speech that should be protected. Openness and inclusivity are cornerstones of the culture of open source, and the tools of open source communities are designed for global access and participation. Instead of malware, there are so many outlets for open source communities to be creative without harming everyone who happens to load the update. Longer term, the downsides of vandalizing open source projects far outweigh any possible benefit, and the blowback will ultimately damage the projects and contributors responsible. Use your power, yes—but use it wisely*" [25].

The purpose of this article is to point the software engineering research community to open questions regarding how researchers can investigate, address, and regulate such kinds of protestware. In light of the War in Ukraine, we present three motivating scenarios where world politics has had impact on software ecosystems, highlighting the side affects, and then present an agenda on how

[1] https://github.blog/2020-03-16-npm-is-joining-github/

to dissect and respond to such behaviour during software engineering practices. Another blog [7] raises growing concerns within an ecosystem: "*Protestware can deliver similar anti-war messages, but within the open-source community there are worries that the possibility of sabotage — especially if it goes further than simple anti-invasion messaging and starts destroying data — can undermine the open-source ecosystem. Although it is less well known than commercial software, open-source software is enormously important to running every facet of the internet*". Our vision is that the lines of research outlined in this article can contribute towards building resilient and yet open software ecosystems.

## 2 MOTIVATING CASES

In this section, we discuss three cases where protestware has had an impact on a software ecosystem. Specifically, we focus on the War in Ukraine, and three different effects that resulted from maintainers making a political stance. The first case involves the malignant effects of protestware. The second case is a benign case where protestware had no malicious intent to harm the users of a library. In the last case, we take a look at a case where sanctions were placed on accounts that were related to world politics.

### 2.1 Case 1: Malignant Protestware

A developer of the JavaScript library node-ipc [8], which is used by the popular vue.js framework, deliberately introduced a critical security vulnerability that, for some netizens, would destroy their computers' files. The library is fetched about a million times a week from the NPM registry, and is described as an "inter-process communication module for Node, supporting Unix sockets, TCP, TLS, and UDP". Seemingly, the maintainer intentionally changed his code to overwrite the host system's data, then changed the code to display a message calling for world peace, as a protest against Russia's invasion of Ukraine. GitHub declared this a critical vulnerability, which was tracked as CVE-2022-23812 [1].

The malicious code was intended to overwrite arbitrary files dependent upon the geo-location of the user's IP address, attacking software in specific locations. Concretely, the affected versions 10.1.1 and 10.1.2 of the library check whether the host machine has an IP address in Russia or Belarus, and if so overwrites every file it could with a heart symbol. Version 10.1.3 was released soon after without this destructive functionality, while 10.1.1 and 10.1.2 were removed from the NPM registry.

There was a strong response from the community, including frustrations that led to insightful discussions. One example from a contributor on the GitHub Discussions channel is shown below [2]:

> *I'm very happy to see that the principles and character of many in tech (FOSS especially) remain clear enough to recognize how completely wrong this was. Of course, if the marketplace of current things keeps hammering away at this, it will benefit a small number of corporate giants (misplaced trust/safety). I hope we all start seeing these patterns as we grapple with a general blurring of lines between tools for marketing and weaponry. It's essential to ask: what's the outcome and who benefits? I like to ask the faux ideologues "who agrees with you?" "Isn't it strange how well aligned you are with a small number of very visible, influential, and powerful organizations?" "What's the fight and who is on which side, again?" It's about competency, not power. Power feeds and is fueled by egocentrism (plainly, weak vanity). Competency comes from discovering your natural gifts and applying them.*

Another user from that Github Discussion quoted how this affected the Open Source Community [2]:

> *The trust factor of open source, which was based on goodwill of the developers is now practically gone, and now, more and more people are realizing that one day, their library/application can possibly be exploited to do/say whatever some random dev on the internet thought was 'the right thing to do'.*

The maintainer defended his module on GitHub, saying "this is all public, documented, licensed and open source". Earlier, there were more than 20 issues flagged against node-ipc about its behavior. Some of the comments referred to the creation as *"protestware, while others might call it malware"*.

### 2.2 Case 2: Benign Protestware

We present two cases where the protestware does not have malicious intent, but aims at increasing awareness. For the first case, the same maintainer of the node-ipc library then created the peacenotwar library [9]. As explained by the maintainer, it serves as a non-violent protest against Russia's aggression. Instead of malicious deletion of files, the module adds a message of peace on users' desktops [10]. The maintainer was quoted in the README file[2]:

> *I pledge that this module, to the best of my knowledge and skills, does not do any damage to anyone's data. If you do not like what this module does, please just lock your dependencies to any of my work or other's which includes this module, to a version you have code reviewed and deemed acceptable for your needs. Also, please code-review your other modules for vulnerabilities.*

When accepted, this code was included:

```
  variable "putin_khuylo"{
  description = "Do you agree that Putin doesn't
   respect Ukrainian sovereignty and territorial
   integrity? More info: https://en.wikipedia.org/
   wiki/Putin_khuylo!"
  type        = bool
  default     = true}
```

Another example is given by a maintainer of the terraform modules for AWS who added their own protest in the licence file, with "Additional terms of use for users from Russia and Belarus" [11]:

> *By using the code provided in this repository you agree with the following:*
> *- Russia has illegally annexed Crimea in 2014 and brought the war in Donbas followed by full-scale invasion of Ukraine in 2022.*
> *- Russia has brought sorrow and devastations to millions*

[2]https://github.com/RIAEvangelist/peacenotwar

*of Ukrainians, killed hundreds of innocent people, damaged thousands of buildings, and forced several million people to flee.*

These two libraries are examples of protestware that are not intended to be malignant, compared to the case described in the previous section.

### 2.3 Case 3: Developer Sanctions

The third case is not related to a single library maintainer, but affects a software ecosystem more broadly. We report two instances. The first is the decision of MongoDB not to sell its products to Russian buyers [33], using the following statement:

*The breadth of the new sanctions from the US and internationally is unprecedented, and MongoDB has taken action to comply with them. We will not sell our cloud services to customers in Russia and Belarus and we will not sell any more MongoDB software to customers in Russia or Belarus.*

MongoDB's decision appears to hinge on an interpretation of Western sanctions that say SaaS subscriptions represent a sale. In order to comply with sanctions, MongoDB decided it should not continue to offer its wares, not even as a service. Interestingly, Oracle also stated on Twitter[3] that it is ending support for Russian users, Even Microsoft joined Apple and SAP in suspending sales in Russia.[4]

In another instance of developer sanctions, at first GitHub suspended Russian accounts. According to a blog post from a developer [5], "suspending an account" on GitHub meant deleting all activity for a user—which results in (1) every pull request from the suspended account being deleted, (2) every issue opened by the suspended account being deleted, and (3) every comment or discussion from the suspended account being deleted. In effect, the user's entire activity and history evaporated. The bigger issue was that GitHub had not provided any warning to the library maintainers. As explained by a maintainer in a blog post [5]:

*I recently took over as a lead maintainer for two popular projects in the Apple developer community, Quick and Nimble. I just released version 5.0 of Quick a few days ago. During the week leading up to the release, I was reviewing and merging many pull requests. But when it came time to write the release notes, I noticed very bizarre behavior. Mysteriously, some pull requests were deleted. Poof. Gone. Then I realized that an entire contributor's presence had disappeared — all of their comments on issues were missing, all of the issues they opened were gone, all of the pull requests they opened had vanished. Every piece of activity related to the user was gone.*

GitHub later reached out to this maintainer, letting them know that it had restored the missing pull requests, issues, comments, etc. from the Russian developers whose accounts had been suspended. User profiles were also restored, although GitHub did not specifically mention that the accounts had been suspended [4].

[3]https://tinyurl.com/8admb86d
[4]https://www.siliconrepublic.com/enterprise/microsoft-sales-russia-ukraine-cyberattacks

## 3 OTHER MAINTAINER STANCES

The War in Ukraine is not the first time that open-source maintainers have used their open source libraries as a platform for protest. For example, Faker.js[5] and Colors.js[6] created problems for users of Amazon's Cloud Development Kit. Big companies, critics have long said, benefit from open source ecosystems without adequately compensating developers for their time. In turn, developers responsible for the software are unfairly strained. The maintainer of these two JavaScript libraries with more than 21,000 dependent apps and more than 22 million weekly downloads, intentionally released an update that produced an infinite loop that caused dependent apps to spew gibberish, prefaced by the words "Liberty Liberty Liberty". His stance was made clear in the README file: "Take this as an opportunity to send me a six-figure yearly contract or fork the project and have someone else work on it" [3].

These cases show that protestware is not only relevant in the context of political conflicts, but has a long history of capturing and communicating the opinions of open source maintainers.

## 4 RESEARCH AGENDA

In this section, we present our research agenda. We have presented three motivating scenarios where world politics has had an impact on a software ecosystem. These cases form initial evidence on the current state of practice, and how the nature of open source software and protestware will affect the software supply chain, trust, and resilience within an ecosystem. In this context, a software supply chain attack occurs when a compromised library distributes malicious code to applications that depend on it.[7]

As discussed in Section 3, there is anecdotal evidence from blogs and developer discussions on the topic, yet there have been very few research studies that articulate the impact of protestware on software engineering. From these motivating cases, the following questions about the possible implications have emerged. We discuss each of these potential implications below in turn, and suggest ten research questions that could form the basis of future research projects.

### 4.1 Dissecting the status quo of Protestware

Evidence from this paper shows the thin line that exists between protestware and malware. As mentioned by the OSI community, protest is an important element of free speech, with openness and inclusivity being cornerstones of the culture of open source. However, vandalizing open source projects threatens any possible benefit, and might damage the projects and contributors responsible. Our first four research questions target the potential of protestware.

(1) How effective is protestware at communicating political messages?
(2) What is the immediate impact of protestware on a software ecosystem?
(3) Who is affected by protestware, and what relationship do they have with the protestware maintainer, if any?

[5]https://fakerjs.dev/
[6]https://colorjs.io/
[7]https://capec.mitre.org/data/definitions/437.html

(4) What mitigation strategies do projects use for securing or repairing their supply chains, and how effective are those mitigation strategies?

Through in-depth studies of cases such as the ones introduced in Section 2, we propose research questions aimed at understanding the status quo of protestware, to build the foundations for further work on increasing resilience and trust in software ecosystems. We propose to survey developers that rely on the modified projects to determine whether protestware managed to achieve its goals of political messaging. Maintainers interested in taking political action are confronted with dilemmas of instrumental vs. value rationality, i.e., selecting the most effective and efficient means to reach given ends (instrumental rationality) or acting without considering the foreseeable consequences in the service of conviction (value rationality) [24, 26]. Is the creation of protestware a legitimate means to pursue a political agenda? Answers to such questions lie beyond traditional software engineering research, but we can investigate whether the message is received as intended, and what its immediate consequences are. Who is affected? A small number of developers who have a working relationship with the acting maintainer (e.g., through reciprocal contributions to each others' code), or an entire ecosystem? Using case studies, we can further investigate which strategies other players in a software ecosystem have employed to work around protestware and/or mitigate its impact, with the ultimate goal of learning how to increase resilience in the future.

## 4.2 Increasing Supply Chain Resilience

In response to the attacks on supply chains, the Linux Foundation's partner group – Open Source Security Foundation (OpenSSF), Google, and Microsoft joined forces to work with security experts and use automated security testing to improve open-source security in a project called the Alpha-Omega Project [6]. This is a global effort to secure code, and has sparked other efforts such as weak links analysis [32]. Hence, our next three research questions are related to the supply chain.

(5) How effective are redundancies in supply chains at increasing resilience?
(6) How do changes which turn libraries into protestware differ from other software evolution?
(7) How accurately can we automatically detect protestware?

Efforts to increase supply chain resilience could form a spectrum from manual efforts to minimise the impact of protestware by reducing the reliance on external libraries [30] or introducing redundancies into supply chains [15] to automated tools for protestware detection and mitigation, similar to vulnerability detection [22]. To pave the way for automated detection and mitigation, we propose research questions related to unique characteristics of patches which turn a library into protestware (e.g., are such patches 'surprising' to a language model trained on past patches? [13]) as well as automated prediction, borrowing methods from the defect prediction literature [29].

## 4.3 Managing Trust and Responsibility

Prior work has established that the success of a library is based on the library itself, which involves the assumption of a module's functional and non-functional correctness. For instance, system maintainers need to trust the reliability of non-functional attributes such as security and stability of an adopted library [19]. This trust in components is well-known in other fields, such as Dependable and Secure Computing [16, 17, 31]. However, we argue that trust also falls back on the maintainer. The final three research questions are aimed at maintainers and their role in an ecosystem as a whole.

(8) What are the responsibilities of maintainers, as perceived by other stakeholders in a software ecosystem?
(9) How does protestware affect trust into a library and entire ecosystems?
(10) How is protestware regulated at ecosystem level?

Recent anecdotes, such as a Fortune-500 company unabashedly requesting a curl maintainer's immediate response about the Log4J vulnerability [27], highlight the current confusion about what exactly are the responsibilities of a great open-source maintainer [14]. The recent emergence of protestware only adds to this confusion—do maintainers act irresponsibly if they use open source for political action? We propose to capture the perceptions of different players in a software ecosystem about these responsibilities and how protestware can affect trust in libraries and ecosystems. It is important that software ecosystems are resilient to threats against their culture, thus becoming sustainable [21]. As mentioned in Section 1, open source licences contain statements about discrimination [18], but they are not explicit about protestware. We also propose to investigate how codes of conduct [28] at project and ecosystem level can play a role in regulating the management of protestware.

## 5 DISCUSSION AND CONCLUSION

Protests are a powerful way for people to make their voices heard. They can be used to call attention to injustices, and to demand change. In the world of open source software, protests take the form of "protestware". In highly inter-connected and large software ecosystems, where the average package can directly depend on more than five other packages (NPM [20]), the impact of protestware on the entire ecosystem can be devastating, especially if it is malignant in nature. Using the context of the ongoing War in Ukraine, we have argued how and why protestware does have an impact on software ecosystems, and we have outlined a comprehensive research agenda for understanding and addressing protestware and its implications on ecosystem resilience, trust, and responsibility.

To answer the research questions posed above, we need a systematic research approach. To confirm the anecdotal evidence, we need methods to define and detect different forms of protestware. Perceptions of developers as well as viewpoints of large Fortune-500 companies would need to be sought, e.g., via surveys or interviews. Furthermore, political stances can become a sensitive topic, so care needs to be taken in the design on how to effectively gather and interpret our results. Our hope is that answering these questions will help us understand how to sustain and build resilient software ecosystems.

## ACKNOWLEDGEMENT

This work is supported by Japanese Society for the Promotion of Science (JSPS) KAKENHI Grant Numbers 20K19774 and 20H05706.

## REFERENCES

[1] 2022. CVE-2022-23812. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-23812. (Accessed on 05/11/2022).

[2] 2022. Discussion on node-ipc. https://github.com/RIAEvangelist/node-ipc/discussions/505. (Accessed on 05/11/2022).

[3] 2022. Faker Protest message. http://web.archive.org/web/20210704022108/https:/github.com/Marak/faker.js/issues/1046. (Accessed on 05/11/2022).

[4] 2022. Github response to the war in Ukraine. https://github.blog/2022-03-02-our-response-to-the-war-in-ukraine/. (Accessed on 05/11/2022).

[5] 2022. GitHub suspending Russian accounts deleted project history and pull requests. https://www.jessesquires.com/blog/2022/04/19/github-suspending-russian-accounts/. (Accessed on 05/11/2022).

[6] 2022. GitHub suspending Russian accounts deleted project history and pull requests. https://openssf.org/community/alpha-omega/. (Accessed on 05/11/2022).

[7] 2022. n. https://www.bleepingcomputer.com/news/security/third-npm-protestware-event-source-polyfill-calls-russia-out/. (Accessed on 05/11/2022).

[8] 2022. node-ipc GitHub Repository. https://github.com/RIAEvangelist/node-ipc. (Accessed on 05/11/2022).

[9] 2022. peacenotwar GitHub Repository. https://github.com/RIAEvangelist/peacenotwar. (Accessed on 05/11/2022).

[10] 2022. peacenotwar message. https://github.com/medikoo/es5-ext/commit/28de285ed433b45113f01e4ce7c74e9a356b2af2. (Accessed on 05/11/2022).

[11] 2022. Terraform added Terms of Use. https://github.com/terraform-aws-modules/terraform-aws-ec2-instance/commit/6867788411a202b61187f9935e9eaa72a18f0bbe. (Accessed on 05/11/2022).

[12] Gerald Benischke. 2022. On the Weaponization of Open Source. https://www.computer.org/publications/tech-news/community-voices/on-the-weaponization-of-open-source. (Accessed on 05/11/2022).

[13] James Caddy, Markus Wagner, Christoph Treude, Earl T Barr, and Miltiadis Allamanis. 2022. Is Surprisal in Issue Trackers Actionable? *arXiv preprint arXiv:2204.07363* (2022). https://doi.org/10.48550/arXiv.2204.07363

[14] Edson Dias, Paulo Meirelles, Fernando Castor, Igor Steinmacher, Igor Wiese, and Gustavo Pinto. 2021. What Makes a Great Maintainer of Open Source Projects?. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 982–994. https://doi.org/10.1109/ICSE43902.2021.00093

[15] Abdelouahed Gherbi, Robert Charpentier, and Mario Couture. 2011. Software diversity for future systems security. *CrossTalk: The Journal of Defense Software Engineering* 24, 5 (2011), 10–13. https://doi.org/10.1.1.445.6492

[16] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. 2004. Propagation of Trust and Distrust. In *Proceedings of the 13th International Conference on World Wide Web* (New York, NY, USA) *(WWW '04)*. Association for Computing Machinery, New York, NY, USA, 403–412. https://doi.org/10.1145/988672.988727

[17] W. Hasselbring and R. Reussner. 2006. Toward trustworthy software systems. *Computer* 39, 4 (2006), 91–92. https://doi.org/10.1109/MC.2006.142

[18] Open Source Initiative. 2007. The Open Source Definition. https://opensource.org/osd. (Accessed on 05/11/2022).

[19] Raula Gaikovina Kula, Daniel M. German, Takashi Ishio, and Katsuro Inoue. 2015. Trusting a library: A study of the latency to adopt the latest Maven release. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 520–524. https://doi.org/10.1109/SANER.2015.7081869

[20] Raula Gaikovina Kula, Ali Ouni, Daniel M German, and Katsuro Inoue. 2017. On the impact of micro-packages: An empirical study of the npm javascript ecosystem. *arXiv preprint arXiv:1709.04638* (2017). https://doi.org/10.48550/arXiv.1709.04638

[21] Raula Gaikovina Kula and Gregorio Robles. 2019. The Life and Death of Software Ecosystems. In *Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability*. Springer, 97–105. https://doi.org/10.1007/978-981-13-7099-1_6

[22] Zhen Li, Deqing Zou, Shouhuai Xu, Hai Jin, Hanchao Qi, and Jie Hu. 2016. Vulpecker: an automated vulnerability detection system based on code similarity analysis. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*. 201–213. https://doi.org/10.1145/2991079.2991102

[23] Suhaib Mujahid, Rabe Abdalkareem, and Emad Shihab. 2022. What are the characteristics of highly-selected packages? A case study on the npm ecosystem. *arXiv preprint arXiv:2204.04562* (2022). https://doi.org/10.48550/arXiv.2204.04562

[24] Guy Oakes. 2003. Max Weber on value rationality and value spheres: Critical remarks. *Journal of Classical Sociology* 3, 1 (2003), 27–45. https://doi.org/10.1177/1468795X03003001693

[25] Open Source Initiative Statement on Protestware. 2007. The Open Source Definition. https://blog.opensource.org/open-source-protestware-harms-open-source/. (Accessed on 05/11/2022).

[26] Mark R Rutgers and Petra Schreurs. 2006. The morality of value-and purpose-rationality: The Kantian roots of Weber's foundational distinction. *Administration & Society* 38, 4 (2006), 403–421. https://doi.org/10.1177/0095399706290632

[27] Daniel Stenberg. 2022. LOGJ4 SECURITY INQUIRY – RESPONSE REQUIRED. https://daniel.haxx.se/blog/2022/01/24/logj4-security-inquiry-response-required/. (Accessed on 05/11/2022).

[28] Parastou Tourani, Bram Adams, and Alexander Serebrenik. 2017. Code of conduct in open source projects. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 24–33. https://doi.org/10.1109/SANER.2017.7884606

[29] Zhiyuan Wan, Xin Xia, Ahmed E Hassan, David Lo, Jianwei Yin, and Xiaohu Yang. 2018. Perceptions, expectations, and challenges in defect prediction. *IEEE Transactions on Software Engineering* 46, 11 (2018), 1241–1266. https://doi.org/10.1109/TSE.2018.2877678

[30] Bowen Xu, Le An, Ferdian Thung, Foutse Khomh, and David Lo. 2020. Why reinventing the wheels? An empirical study on library reuse and re-implementation. *Empirical Software Engineering* 25, 1 (2020), 755–789. https://doi.org/10.1007/s10664-019-09771-0

[31] Zheng Yan and Christian Prehofer. 2011. Autonomic Trust Management for a Component-Based Software System. *IEEE Transactions on Dependable and Secure Computing* 8, 6 (2011), 810–823. https://doi.org/10.1109/TDSC.2010.47

[32] N. Zahan, T. Zimmermann, P. Godefroid, B. Murphy, C. Maddila, and L. Williams. 2022. What are Weak Links in the npm Supply Chain?. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE Computer Society, Los Alamitos, CA, USA, 331–340. https://doi.org/10.1109/ICSE-SEIP55303.2022.9794068

[33] Dimko Zhluktenko. 2022. Tweet. https://twitter.com/dim0kq/status/1502372427589996545. (Accessed on 05/11/2022).