[60] J. Manyika. An overview of bard: an early experiment with generative ai. Technical report, Google AI, 2023.

[61] M. Melo and G. Aquino. The pathology of failures in iot systems. In *Computational Science and Its Applications–ICCSA 2021: 21st International Conference, Cagliari, Italy, September 13–16, 2021, Proceedings, Part IX 21*, pages 437–452. Springer, 2021.

[62] National Research Council et al. *Software for dependable systems: Sufficient evidence?* National Academies Press, 2007.

[63] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 736–747, Raleigh North Carolina USA, Oct. 2012. ACM.

[64] C. Nissen, J. E. Gronager, R. S. Metzger, and H. Rishikof. Deliver uncompromised: A strategy for supply chain security and resilience in response to the changing character of war. Technical report, MITRE CORP MCLEAN VA, 2018.

[65] M. Ohm, H. Plate, A. Sykosch, and M. Meier. Backstabber's knife collection: A review of open source software supply chain attacks. In C. Maurice, L. Bilge, G. Stringhini, and N. Neves, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 23–43, Cham, 2020. Springer International Publishing.

[66] C. Okafor, T. R. Schorlemmer, S. Torres-Arias, and J. C. Davis. Sok: Analysis of software supply chain security by establishing secure design properties. In *Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*, SCORED'22, page 15–24, New York, NY, USA, 2022. Association for Computing Machinery.

[67] OpenAI. Gpt best practices. https://platform.openai.com/docs/guides/gpt-best-practices, 2023.

[68] OpenAI. Openai platform. https://platform.openai.com/docs/api-reference/chat, 2023. Accessed: 2023-07-05.

[69] OpenAI. Openai platform - gpt-3.5 models. https://platform.openai.com/docs/models/gpt-3-5, 2023. Accessed: 2023-06-27.

[70] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. How can i improve my app? Classifying user reviews for software maintenance and evolution. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 281–290. IEEE, 2015.

[71] I. Pashchenko, H. Plate, S. E. Ponta, A. Sabetta, and F. Massacci. Vulnerable open source dependencies: counting those that matter. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, Oulu Finland, Oct. 2018. ACM.

[72] K. Pedersen. Barriers for post mortem evaluations in systems development. In *UKAIS Conference, Glasgow, UK. ; Conference date: 19-05-2010*, 2004.

[73] H. Petroski et al. *Design paradigms: Case histories of error and judgment in engineering*. Cambridge University Press, 1994.

[74] S. Pichai. An important next step on our ai journey. https://blog.google/technology/ai/bard-google-ai-search-updates/, 2023. Accessed: 2023-07-03.

[75] S. E. Ponta, H. Plate, and A. Sabetta. Detection, assessment and mitigation of vulnerabilities in open source dependencies. *Empirical Software Engineering*, 25(5):3175–3215, Sept. 2020.

[76] P. Ranade, A. Piplai, A. Joshi, and T. Finin. Cybert: Contextualized embeddings for the cybersecurity domain. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 3334–3342. IEEE, 2021.

[77] D. Robbins. Gentoo linux security announcement 200312-01. https://archives.gentoo.org/gentoo-announce/message/7b0581416ddd91522c14513cb789f17a, 2003.

[78] I. Sarwar, A. Samad, and S. Mumtaz. Object oriented software modeling using nlp based knowledge extraction. *European Journal of Scientific Research*, 35:22–33, 01 2009.

[79] J. Schulman, B. Zoph, C. Kim, J. Hilton, J. Menick, J. Weng, J. F. C. Uribe, and L. Fedus. Introducing chatgpt. https://openai.com/blog/chatgpt, 2023. Accessed: 2023-07-03.

[80] E. Schwartz. [aur-general] acroread package compromised. https://lists.archlinux.org/pipermail/aur-general/2018-July/034152.html, 2018.

[81] Security Technical Advisory Group. Software Supply Chain Best Practices. Technical report, Cloud Native Computing Foundation, May 2021.

[82] I. Sommerville. *Software Engineering*, volume 137035152. Pearson Education, 2015.

[83] Sonatype. State of the software supply chain. https://www.sonatype.com/resources/state-of-the-software-supply-chain-2021, 2021.

[84] Sonatype. State of the Software Supply Chain. Technical Report 8th Annual, Sonatype, 2022.

[85] Synopsys. 2023 OSSRA Report. https://www.synopsys.com/software-integrity/engage/ossra/rep-ossra-2023-pdf.

[86] The Linux Foundation. SLSA: Supply-chain levels for software artifacts. https://slsa.dev, 2022. Accessed: 2022-04-30.

[87] The Recorded Future Team. What is open source intelligence and how is it used? = https://www.recordedfuture.com/open-source-intelligence-definition, 2022. Accessed: 2023-06-21.

[88] N. Vasilakis, A. Benetopoulos, S. Handa, A. Schoen, J. Shen, and M. C. Rinard. Supply-Chain Vulnerability Elimination via Active Learning and Regeneration. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1755–1770, Virtual Event Republic of Korea, Nov. 2021. ACM.

[89] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[90] K. Vijayakumar and C. Arun. Automated risk identification using NLP in cloud based development environments. *Journal of Ambient Intelligence and Humanized Computing*, 2017.

[91] K. Vivek. Is software reuse leading to dependency hell? = https://www.linkedin.com/pulse/software-reuse-leading-dependency-hell-vivek-kant, Sept. 2022.

[92] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.

[93] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. *arXiv preprint arXiv:2303.07839*, 2023.

[94] M. Yao. Top 6 nlp language models transforming ai in 2023. *TOPBOTS*, 2023.

[95] N. Zahan, T. Zimmermann, P. Godefroid, B. Murphy, C. Maddila, and L. Williams. What are Weak Links in the npm Supply Chain? In *International Conference on Software Engineering (ICSE)*, 2022.

[96] K. Zetter. 'google' hackers had ability to alter source code. https://www.wired.com/2010/03/source-code-hacks/, 2010.

[97] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro. Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. *ACM Computing Surveys*, 54(3):1–41, 2022.

[98] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen. A survey of large language models, 2023.

[99] M. Zimmermann, C.-A. Staicu, and M. Pradel. Small World with High Risks: A Study of Security Threats in the npm Ecosystem. In *USENIX Security Symposium*, 2019.

APPENDIX

Table 9 presents the finalized prompts utilized to query the Language Learning Models (LLMs) across various dimensions. These prompts were derived using a range of prompt engineering techniques, as detailed in Table 6.

Table 10 gives the full set of solutions/learnings proposed by GPT for the four articles discussed in detail in §5.2.

We wondered whether software supply chain reporting quality has improved over the years. If this were the case, we would expect to see an increase in LLM performance for newer articles. Figure 6 shows no such trend.

Figure 7, Figure 8, and Figure 9 show the ground truth for various dimensions. The ground truth for the dimension "Impact" is not presented as the disagreements among the raters were not resolved. In total, there were 65 articles analyzable for the "Intent", "Nature" and "Impacts" dimensions. For "Type of Compromise", there were analyzable articles. The failures that were not included were the ones with not functioning URLs and PDF formats, and where the manual labeling of the type of compromise by CNCF was not in the taxonomy.

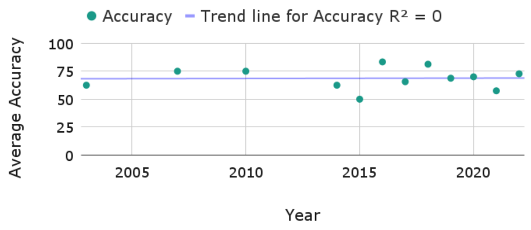

Figure 6: The average accuracy of the articles for all the dimensions over the years. The graph shows no specific trend.

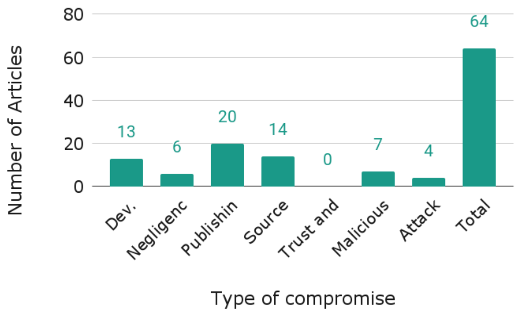

Figure 7: Categorization of articles for the dimension- "Type of Compromise" by CNCF catalog.

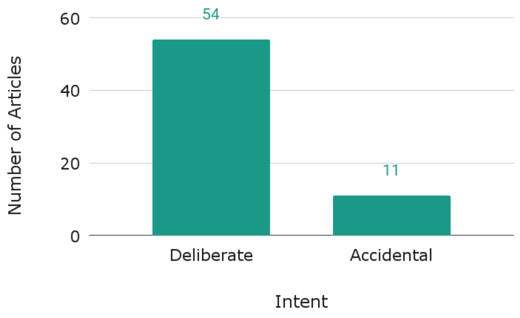

Figure 8: Categorization of articles for the dimension- "Intent" by raters.

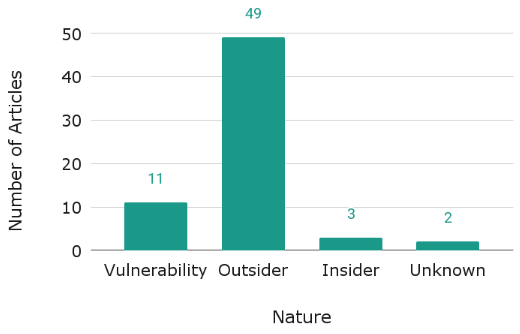

Figure 9: Categorization of articles for the dimension- "Nature" by raters.

Table 9: The final prompts for each dimension.

Dimension	Prompt
Type of compromise	Classify the attack from the following choices Choice 1: Dev Tooling- This occurs when the development machine, SDK, toolchains, or build kit has been exploited. These exploits often result in the introduction of a backdoor by an attacker to own the development environment. Choice 2: Negligence- Occurs due to a lack of adherence to best practices. TypoSquatting attacks are a common type of attack associated with negligence, such as when a developer fails to verify the requested dependency name was correct (spelling, name components, glyphs in use, etc). Choice 3: Publishing Infrastructure- Occurs when the integrity or availability of shipment, publishing, or distribution mechanisms and infrastructure are affected. This can result from a number of attacks that permit access to the infrastructure. Choice 4: Source Code- Occurs when a source code repository (public or private) is manipulated intentionally by the developer or through a developer or repository credential compromise. Source Code compromise can also occur with intentional introduction of security backdoors and bugs in Open Source code contributions by malicious actors. Choice 5: Trust and Signing- Occurs when the signing key used is compromised, resulting in a breach of trust of the software from the open source community or software vendor. This kind of compromise results in the legitimate software being replaced with a malicious, modified version. Choice 6: Malicious Maintainer- Occurs when a maintainer, or an entity posing as a maintainer, deliberately injects a vulnerability somewhere in the supply chain or in the source code. This kind of compromise could have great consequences because usually the individual executing the attack is considered trustworthy by many. This category includes attacks from experienced maintainers going rogue, account compromise, and new personas performing an attack soon after they have acquired responsibilities. Choice 7: Attack Chaining- Sometimes a breach may be attributed to multiple lapses, with several compromises chained together to enable the attack. The attack chain may include types of supply chain attacks as defined here. However, catalogued attack chains often include other types of compromise, such as social engineering or a lack of adherence to best practices for securing publicly accessible infrastructure components. Explain your answer using the given definitions and return the option. Use JSON format with the keys: 'explanation', 'choice' Based on the information provided in the Article delimited by triple backticks. Article: ```{article}```
Intent	Was the root cause of the compromise: Option 1: deliberate eg. cyberattack on a system, malicious attackers stealing information Option 2: accidental eg. Development incompetence or a bug/vulnerability found Explain your reasoning and select an option. Use a JSON format with the keys: 'Explanation', 'option' Based on the information provided in the Article delimited by triple backticks. Article: ```{article}```
Nature	Was the article about an attack or a vulnerability which was not exploited? If it was an attack, who was responsible for the attack? Choice 2: Outsider - An attack conducted by an individual or group outside the supply chain, such as a group of terrorists or malicious actors. Choice 3: Insider- attack by the developer/someone who was a part of the supply chain Choice 4: Unclear If it was a vulnerability, return choice 1: Vulnerability Explain your reasoning and chose an option. Use a JSON format with the key: 'Explanation', 'Option'. Based on the information provided in the Article delimited by triple backticks. Article: ```{article}```
Impacts	Classify the attack from the following choices, remember if it is one or more, choose option 5 Option 1: Performing data or financial theft- accessing, extraction, alteration, or destruction of data and/or identity/financial theft. Option 2: Disabling networks or systems- compromising core functionality, efficiency, or maintainability of the system. Software changes that lead to the product being unusable Option 3: Monitoring organizations or individuals- keeping track of activities performed by organizations or individuals Option 4: Causing physical harm or death. Option 5: All of the above/multiple choices- it is a vulnerability/exploit that can lead to various or all of the impacts from the list. Option 6: Unknown or unclear Explain your answer using the given definitions and return the option. Use JSON format with the keys: 'explanation', 'option' Based on the information provided in the Article delimited by triple backticks. Article: ```{article}```
Solutions/learnings	Suggest the solutions and learning from the failure. Use JSON format with the keys: 'Solutions/Learnings' Based on the information provided in the Article delimited by triple backticks. Article: ```{article}```

Table 10: The solutions/learnings outputted by GPT for the four articles mentioned in §5.2.

{article}	**Solutions/Learnings from GPT**
Article 7	"Implement stronger password reset mechanisms that use secure random number generators instead of weak entropy.", "Regularly update and patch software dependencies, such as Archive_Tar, to prevent known vulnerabilities.", "Encourage companies and developers to migrate from PEAR to Composer, which has a more active contributors community and better security practices.", "Increase security contributions from companies relying on PHP ecosystems like PEAR to identify and fix vulnerabilities.", "Continuously monitor and scan code repositories for security vulnerabilities using tools like SonarCloud.", "Improve the validation and sanitization of user input to prevent potential attacks.", "Educate developers and administrators about secure coding practices and the importance of regularly updating software dependencies.", "Establish a responsible disclosure process to report vulnerabilities to maintainers and ensure timely patching.", "Enhance security awareness and training for developers and administrators to identify and mitigate potential threats.", "Implement access controls and monitoring mechanisms to detect and prevent unauthorized access to critical systems and networks.", "Regularly review and assess the security of third-party dependencies and libraries used in software development."
Article 35	"Implement stricter security measures when transferring ownership of npm packages to prevent malicious users from gaining control.", "Regularly audit and review the dependencies of popular npm packages to identify any potential security vulnerabilities.", "Encourage open source developers to collaborate and review each other's code to catch any malicious modifications.", "Consider implementing code signing or verification mechanisms to ensure that the code being uploaded in an npm module is equivalent to the code stored publicly in a git repository.", "Educate developers about the importance of verifying the integrity and authenticity of dependencies before including them in their projects.", "Encourage the use of tools that scan npm packages for potential security vulnerabilities, even though they may not be foolproof.", "Promote the use of Content Security Policy (CSP) to prevent malicious code execution in web applications.", "Encourage the use of Intrinsic or similar Node.js packages to whitelist and control access to sensitive resources and APIs."
Article 65	"The project should provide more information about the incident, including checksums of good or compromised versions of the software.", "The project should investigate and address the vulnerability that allowed the cracker to gain user-level access to the server.", "The project should take steps to ensure that future WordPress releases are not susceptible to similar security breaches.", "WordPress users should ask the project serious questions to ensure that they are protected from future incidents.", "The WordPress developers should conduct code audits to identify and fix any potential backdoors inserted with more care.", "Free software projects should be aware of the incentive for crackers to insert malware and take appropriate measures to protect their distributions.", "Projects should focus on securing their processes and servers to prevent attacks and minimize the impact of any potential breaches."
Article 67	"Perform forensic analysis to determine the cause of the remote exploit", "Install additional security measures such as intrusion detection systems and file integrity checkers", "Regularly monitor and analyze server logs for any suspicious activity", "Educate users about the importance of running security updates and syncing against trusted servers", "Consider implementing stronger access controls and authentication mechanisms", "Regularly backup critical data to minimize the impact of a compromise", "Collaborate with sponsors and infrastructure providers to ensure the security of donated servers", "Promptly remove compromised servers from rotations and rebuild them after forensic analysis", "Consider publicly identifying compromised servers to increase transparency and awareness", "Continuously improve security measures based on lessons learned from incidents"