Manipulating the Maintainers: Detecting Text-Based Social Engineering via Automated Sentiment Analysis

Benton M. Cesanek, MIDN
Dept. of Cyber Sciences
United States Naval Academy
Annapolis, United States
m260990@usna.edu

***Abstract*—This paper provides a conceptual overview of social engineering in cyberspace and specifically aims to explore how a lethal combination of social engineering techniques can be leveraged as attack vectors to target open source software maintainers, which when successful can lead to malicious code being committed into vulnerable repositories. The paper argues that automated sentiment analysis of open source message boards is a viable technique to detect and combat efforts to social-engineer repository maintainers when used in tandem with DEVSECOPS, ultimately protecting the integrity of the FOSS supply chain. The paper explores the process of fine-tuning AI through prompt engineering to correctly identify cases of malicious social engineering as found in the recent XZ Utils/liblzma backdoor, and presents the corresponding rates of detection accuracy and correct classification of potentially malicious messages from the XZ project forums.**[*]

***Index Terms*—Supply chain attack, Social engineering, Prompt engineering, Sentiment analysis, Large language model, Fine-tuning**

I. Introduction

According to a 2022 census study completed by The Linux Foundation and the Laboratory for Innovation Science at Harvard, "Free and Open Source Software (FOSS) constitutes 70-90% of any given piece of modern software solutions" [1], [2]. As FOSS, an umbrella term encompassing Open Source Software (OSS), appears in over 96% of all code bases and constitutes in some cases up to 99% of commercially available software, it represents one of the most important supply chains in a digitally dependent world with an estimated demand-side value of approximately $8.8 trillion USD [1]. The wide spread use, dependence upon, and inherent value of Open Source Software means that the security and safe guarding of the FOSS supply chain is not just of interest to a few individuals or select nation states, but rather an ongoing concern of global significance. Major distributions such as Red Hat, AWS, Fedora, Debian, Kali Linux, and Ubuntu represent just a select few of many software services and operating systems whose functionality and security are directly reliant upon the integrity of the millions of OSS projects in use today [3]. If attackers are able to disrupt the FOSS supply chain at the root, it would create an immediate ripple effect negatively impacting all parties both indirectly and directly dependent on OSS [4]. Ironically these OSS projects, which as mentioned constitute the backbone of modern infrastructure and digital industry, are often simply daytime or hobby projects for the communities and unpaid maintainers who are responsible for their longevity and development. As such, the vast majority of OSS projects and repositories lack specifically assigned security teams such as those utilized by developers such as Apple or Microsoft, making them and by extension the FOSS supply chain incredibly attractive targets for malefactors [5].

Exploitation of the FOSS supply chain is unfortunately not a novel occurrence by any means, a claim supported by the recent XZ Utils cyber attack and case study. On Mar 29, 2024, Microsoft software engineer Andres Freund uncovered a serious compromise (CVE-2024-3094) that had been committed to liblzma, a library which was part of the popular data compression OSS XZ Utils repository [6], [7]. This backdoor had been created by a user named Jia Tan. Over the course of a dedicated two-year long social engineering campaign from 2021-2023, Jia Tan had managed to gain primary control of the XZ Utils repository. He was able to accomplish this by targeting, pressuring, and social engineering historic maintainer Collin Lasse, who had been managing the repository since he created it in 2009, into surrendering maintainership of XZ Utils [8].

Once Collin Lasse was removed from the picture, Jia Tan utilized credibility he had built up over a series of previous legitimate commits to hide the fact that he was actually committing malicious software to libzma. By the time Andres Freund had discovered the XZ backdoor, which in his own words was largely by luck, it was just a month away from being incorporated into Fedora 40, RedHat, Ubuntu 20.04LTS, and other major OSS distros and services [3], [9]. Had it not been caught, Jia Tan's embedded liblzma exploit would have given remote access via OpenSSH to potentially 11 million SSH servers, jeopardizing and infecting the entire FOSS supply chain [3]. The only questions that remain are

[*] ***The views expressed in this document are those of the authors and do not reflect the official policy or position of the U.S. Naval Academy, Department of the Navy, the Department of Defense, or the U.S. Government.***

if XZ Utils was caught largely by coincidence then then what other similar cases are being missed, and what steps can be taking to better prevent another ”XZ Utils”.

II. Background & History

The FOSS supply chain is a critical component of modern software development. Version Control Systems (VCS) evolved from centralized configuration management tools such as Subversion (SVN) to a distributed version control system such as GIT. The creation and revision of modern software follows a variety of development life-cycle processes, which when chosen for use by the software architect or maintainer leverages VCS systems. The development life cycle processes such as Waterfall, Iterative design and development, or Agile development at their heart use VCS. Development Operations (DEVOPS) enforces a production process of Development, Testing and Production which then moves to Operational and Deployment phases of the software [10] and as shown in Figure I. The driving principle of OSS and DEVOPS is that the more eyes that are reviewing the software the easier it is to detect defects. This software development paradigm assumes honest brokers in the development of the software. The issue of dishonest brokers in the development of the software gave rise to Development Security Operations (DEVSECOPS). In [10], describes the integral concept of secure software development while emphasizing integration of security in the development process of commercial software to combat malefactors.

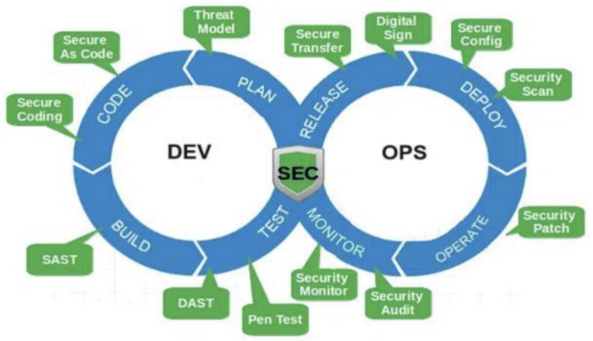

Fig. 1. Development Security Operations as defined by [10]. A multi-process of development and operations utilizing security as a gating function as the cycle moves to operations and production of commercial software.

The processes labeled as Plan, Code, Build and Test lead to a security gate (SEC) which then checks to see if designated security requirements are met. When pertaining to OSS, the concept and application of DEVSECOPS is not immediately apparent nor practical given the voluntary and community driven nature of OSS development. According to [11]–[13] attacks against the OSS supply chain are increasing in both frequency and sophistication. This is further supported by data sourced from Sonatype: Software Supply Chain Management which indicates that between 2020-2024 the number of malicious packages detected in OSS repositories increased from 929 to 459,070 or by a factor of 494%. As of this moment, these packages are more likely than not the result of traditional attack methods, and addressing the sharp increase in malicious software detected in the FOSS supply chain is something that could be addressed by a more cognizant effort of developers and maintainers in the OSS community to actually implement DEVSECOPS across their repositories.

The resounding shortcoming of the current model of DEVSECOPS is that while it accounts for the technical nature of protecting the actual integrity of OSS itself, the cycle offers no real method of defending the maintainers and developers themselves, including against social engineering. This is especially problematic when considering the fact that project maintainers and developers are the ones responsible for implementing security protocols and solutions like DEVSECOPS within their repositories. As is described in [12]:

> ***”Furthermore, attackers may become maintainers themselves through social engineering... [12]”***,
> Ohm, M. et.al.

When Jia Tan joined XZ Utils in 2021, he did just that [12], replacing an unprotected maintainer Collin Lasse via social engineering, thus putting himself in a position from which he could easily implement vulnerabilities into the XZ Utils repository. While DEVSECOPS was not a system of development prevalent within XZ Utils, it would not have done much good even if it was used considering it was Collin Lasse who was targeted initially and DEVSECOPS only prioritizes OSS integrity. As such, the events and social engineering based approach of the ZX Utils attack to the following conclusions:

1) *If the maintainer is responsible for secure development practices within their, yet they themselves are vulnerable to social engineering attacks and replacement, then integrity of the repository cannot be guaranteed unless the maintainer is protected.*
2) *A method of defending the maintainer against text-based social engineering attacks must be implemented across the FOSS supply chain in the form of a new hybrid DEVSECOPS process.*

III. Research Question & Hypotheses

The research question and subsequent hypotheses are defined as follows:

> **Research Question**: Can Generative Artificial Intelligence (AI) when directed to perform sentiment analysis successfully detect text-based social engineering attacks against a software maintainer?

The following defines a series of hypotheses and null hypotheses to support the research question:

> $\mathbf{H}_1$: Is there a discernible pattern of a social engineering attacks which is a precursor of a cyber-security attack against a software repository maintainer that is detectable?
> $\mathbf{H}_2$: Can AI be used as a security gate in the DEVSECOPS construct to detect social engineering attacks?

The null hypotheses are stated as follows:

H$_{\emptyset_1}$: There is no discernible social engineering pattern that can be detected in a cyber security attack against the maintainer.

H$_{\emptyset_2}$: AI would not be suitable as a DEVSECOPS security gate in the DEVSECOPS development model.

IV. Methods

Sentiment analysis as a means to combat the social engineering of OSS maintainers is an approach with little to no research. As such, this paper examines whether a Large Language Model (LLM) performing automated sentiment analysis on a large text-based data set is capable of detecting instances of social engineering within the context of the XZ Utils Attack to a high degree of precision and accuracy. To begin, the large text-based data set for this paper was generated from the Tukaani XZ-dev forum availible for public view online [14]. A total of 699 messages dating from 2011-2025 were scraped from the forum, zipped, and then compiled into a working csv data set. As the XZ-dev forum is the main space for communications pertaining to XZ Utils, the compiled data set included known instances of social engineering pertaining to Jia Tan's campaign to remove Collin Lasse. Next, using manual review enhanced by documented timelines of Jia Tan's activities from 2021-2024, all 699 messages were examined for definite cases of social engineering [6], [15]. This review generated a total of three true positive instances of social engineering. The other 696 messages in the working data set either contained no instances of social engineering or not enough to be considered relevant or a good example.

ChatGPT 4o mini was selected as the AI model to run the sentiment analysis trials. This decision was made due to ChatGPT 4o's mainstream popularity and convenient browser based organizational features, which allowed for effective fine-tuning of a specific GPT via dedicated file integration and customized project instructions. Additionally, the chosen GPT was selected on additional premises of speed and cost efficiency. In order to avoid high rates of hallucination commonly associated with ChatGPT models, the project "Instructions" prompt was engineered in such a way as to instruct the GPT to take its time, double check its work, and re-familiarize itself with the provided content and files before returning an answer to the user.

Three prompts with varying degrees of rule and content specificity were manually created to further fine-tune the selected GPT. Fine-tuning the model was selected over training the model due to research time constraints. All three prompts contained identical flagging protocol for identifying True Positive and True Negative instances of social engineering. The prompts also all contained specific rules addressing the fact that the maintainer Collin Lasse would not be capable of social engineering himself, and as such an message generated by him was to be treated as a True Negative. Additionally, in order to enhance the fine-tuning of the model, the three prompts were supplemented with a Lexicoder Sentiment Dictionary and a Sarcasm Detection dictionary to ensure that the model had the maximum resources available to it to enhance its decision making process [16], [17].

V. Experiment

In order to test the research question of whether or not AI can detect text-based social engineering attacks against an OSS maintainer via sentiment analysis, the chosen ChatGPT 4o mini model was presented with the working csv dataset containing all 699 messages from the XZ-dev forum. This was conducted with the hope that upon iterating through the dataset, the model would be able to correctly flag and identify True Positive instances of social engineering. If the model flagged an incorrect number of messages within the dataset True Positive, then it was reset, asked to re-iterate through the dataset in order to re-familiarize itself, and was provided further prompting in order to enable it to perform better on the next trial. This process was repeated, with the three different prompts being used to help guide the model, until the it correctly flagged and identified the right number of True Positive instances of social engineering.

VI. Analysis

From the experiment, the total number of iterations, trials, and flagged True Positive instances of social engineering were recorded in order to analyze the effectiveness of AI at being able to detect social engineering directed against the maintainer via sentiment analysis. The experiment took a total

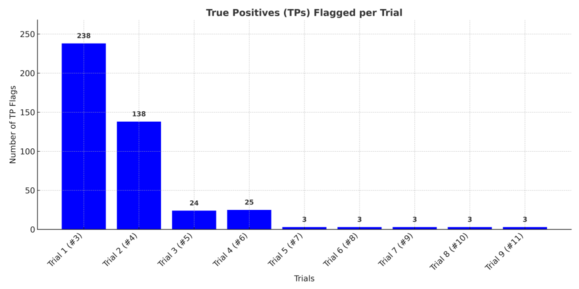

Fig. 2. Quantified predicted TP flags as returned per iteration and experiment trial

of 12 iterations through the data, 9 trials, and approximately 4 hours and 50 minutes for the model to correctly identify and flag the correct True Positive instances of social engineering. Figure II contains a visual indicating the number of predicted True Positive flags returned by the GPT model per iteration and experiment trial.

TABLE I
Confusion Matrix Initial Trial Quantified Results

	Predicted Yes	**Predicted No**	
Actual Yes	TP = 3	FN = 0	3
Actual No	FP = 235	TN = 461	696
	238	461	

As shown in Table I, the quantified results of respective flags returned from the first trial of the experiment indicate an inability for the model to correctly identify instances of social engineering after minimal tuning and exposure to the dataset.

- *Accuracy = 0.6638*
- *Precision = 0.0126*
- *False Discovery Rate = 0.9874*

These results while far from ideal, are unsurprising considering the incredibly limited degree of familiarity the model had with the data during the first trial.

TABLE II
CONFUSION MATRIX FINAL TRIAL QUANTIFIED RESULTS

	Predicted Yes	**Predicted No**	
Actual Yes	TP = 3	FN = 0	3
Actual No	FP = 0	TN = 696	696
	3	696	

The data from the final trial of the experiment presented in Table II indicates a complete positive flip in the models ability to successfully identify instances of social engineering directed against the maintainer with sentiment analysis.

- *Accuracy = 1.0000*
- *Precision = 1.0000*
- *False Discovery Rate = 0.0000*

While the ChatGPT model is evidently not capable of correctly identifying examples social engineering in a large dataset immediately, when given time to learn and familiarize with the tasks required of it such as executing sentiment analysis, the model was able to meet expectations and complete the task required of it without any error or instances of false discovery.

VII. Related Works

While there are a number of works in existence today that analyze the XZ-Utils Attack, none of them approach the concept of using automated sentiment analysis as a means to defend OSS maintainers against targeted social engineering attacks [4], [5], [18], [19]. The [5] is the closest comparable research to this paper, as both critically focus on and analyze Jia Tan's activities that proceeded his committing of the backdoor itself to the repository. However, [5] delves into the concept of ”software engineering” over social engineering, and does not present any solutions similar to automating sentiment analysis.

VIII. Conclusion

Upon examination of the results produced from the experiment, this paper is able to conclude that generative AI, specifically ChatGPT 4o mini, is capable of successfully detecting social engineering attack against a software maintainer when directed to use sentiment analysis. The 9th trial yielded a *False Discovery Rate = 0.0000* score indicating that there were no instances of False Positive identification of social engineering examples by the end of the expirement. Additionally, the messages from the XZ-dev forum which the GPT flagged, all contained a pattern of negative terminology surrounding the word ”maintainer”. This suggests support for $\mathbf{H}_1$, as the model was able to consistently identify similar patterns of language tone and contextual usage. By extension, $\mathbf{H}_2$ was also backed up by the *Precision = 0.0000* and *Accuracy = 0.0000* scores which indicate that the model was entirely successful at detecting text-based social engineering directed at the project maintainer by the end of the experiment, implying it would be a valuable DEVSECOPS security gate.

IX. Acknowledgments

The author would like to thank United States Naval Academy (USNA) professors Dr. Melaragno and Dr. Casey. In addition, I would like to thank Office of Naval Research (ONR) and LCDR Chris Scheider.

References

[1] M. Hoffmann, F. Nagle, and Y. Zhou, “The value of open source software,” *Harvard Business School Strategy Unit Working Paper*, no. 24-038, 2024.

[2] L. F. O. C. I. Project), “A summary of census ii: Open source software application libraries the world depends on,” Mar 2022. [Online]. Available: https://www.linuxfoundation.org/blog/blog/a-summary-of-census-ii-open-source-software-application-libraries-the-world-depends-on

[3] A. Melaragno, C. Farid, and W. Casey, “The anatomy and implications of a software supply chain attack & panel discussions,” United States Naval Academy, 2024, controlled Unclassified Information. No public link available.

[4] S. Hamer, J. Bowen, M. N. Haque, R. Hines, C. Madden, and L. Williams, “Closing the chain: How to reduce your risk of being solarwinds, log4j, or xz utils,” *arXiv preprint arXiv:2503.12192v1*, Mar 2025. [Online]. Available: https://arxiv.org/abs/2503.12192

[5] P. Przymus and T. Durieux, “Wolves in the repository: A software engineering analysis of the xz utils supply chain attack,” *arXiv preprint arXiv:2504.17473v1*, Apr 2025. [Online]. Available: https://arxiv.org/abs/2504.17473

[6] E. Boehs. (2024) Everything i know about the xz backdoor. Accessed: 2025-04-30. [Online]. Available: https://boehs.org/node/everything-i-know-about-the-xz-backdoor

[7] SSH Communications Security. (2024) A recap of the openssh and xz liblzma incident. Accessed: 2025-04-30. [Online]. Available: https://www.ssh.com/blog/a-recap-of-the-openssh-and-xz-liblzma-incident

[8] J. Sherman. (2024) The backdoor in xz utils that almost happened. Accessed: 2025-04-30. [Online]. Available: https://www.lawfaremedia.org/article/backdoor-in-xz-utils-that-almost-happened

[9] S. C. S. S. Team), “A recap of the openssh and xz liblzma incident,” Apr 2024. [Online]. Available: https://www.ssh.com/blog/a-recap-of-the-openssh-and-xz-liblzma-incident

[10] B. A. Dapshima and S. K. Ahmad, “Evaluation and assessment of software security risks and vulnerabilities within the realm of secure devops,” *no. July*, 2024.

[11] P. Ladisa, H. Plate, M. Martinez, and O. Barais, “Sok: Taxonomy of attacks on open-source software supply chains,” in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 1509–1526.

[12] M. Ohm, H. Plate, A. Sykosch, and M. Meier, “Backstabber's knife collection: A review of open source software supply chain attacks,” in *Detection of Intrusions and Malware, and Vulnerability Assessment: 17th International Conference, DIMVA 2020, Lisbon, Portugal, June 24–26, 2020, Proceedings 17*. Springer, 2020, pp. 23–43.

[13] B. Chess, F. D. Lee, and J. West, “Attacking the build through cross-build injection: how your build process can open the gates to a trojan horse,” *Fortify Software*, pp. 24–25, 2007.

[14] T. M. List, “Xz-devel mailing list archive,” n.d. [Online]. Available: https://www.mail-archive.com/xz-devel@tukaani.org/maillist.html

[15] H. L. Team, “Where the wild things are: A complete analysis of jia tan's github history and the xz utils software supply chain breach,” Apr 2024. [Online]. Available: https://huntedlabs.com/where-the-wild-things-are-a-complete-analysis-of-jia-tans-github-history-and-the-xz-utils-software-supply-chain-breach

[16] S. N. Soroka, "Lexicoder sentiment dictionary (lsd)," n.d. [Online]. Available: https://www.snsoroka.com/data-lexicoder/

[17] E. Riloff, M. Sultan, and S. Amir, "News headlines dataset for sarcasm detection v2," n.d. [Online]. Available: https://nlds.soe.ucsc.edu/sarcasm2

[18] M. Lins, R. Mayrhofer, M. Roland, D. Hofer, and M. Schwaighofer, "On the critical path to implant backdoors and the effectiveness of potential mitigation techniques: Early learnings from xz," *arXiv preprint arXiv:2404.08987v1*, Apr 2024. [Online]. Available: https://arxiv.org/abs/2404.08987

[19] D.-L. Vu, T. Dunlap, K. Obermeier-Velazquez, P. Gibert, J. S. Meyers, and S. Torres-Arias, "A study of malware prevention in linux distributions," *arXiv preprint arXiv:2411.11017v2*, Nov 2024. [Online]. Available: https://arxiv.org/abs/2411.11017