Securing the software supply chain

Charles Zumbaugh
Dept. of Computer Science
Kansas State University
Manhattan, Kansas
cazumbaugh@ksu.edu

***Abstract*—Modern software development depends heavily on layers of abstraction such as libraries, frameworks, cloud infrastructure, build tooling, and other software built and maintained by others. This speeds innovation and makes the development of complex systems more accessible by allowing developers to build upon previous work. Given it's importance to modern development, the software supply chain represents an important attack vector. Attackers are increasingly targeting this ecosystem to mount attacks, and recent incidents highlight the scale of the fallout when these attacks are successful. This review discusses the primary attack vectors in the software supply chain, security measures and frameworks to defend against these attacks, and recent high-profile incidents.**

***Index Terms*—memory corruption, security, vulnerability, systems, malware**

I. Introduction

As computers become more embedded in everyday life, the security of these systems have become a critical concern. Cyber attacks against government and non-government organizations have increased in recent years, particularly against healthcare and large, multinational companies [1]. As personal data is increasingly being collected and stored on servers, the scale of these breaches is also increasing. For example, a recent breach of Change Healthcare in 2025 exposed the personal data of more than 192 million individuals in the United States [2]. The financial implications of security vulnerabilities are substantial, and IBM estimated the cost of a data breach in 2025 to be $4.4 million [3].

Modern software is built upon layers of reusable abstractions such as libraries, frameworks, cloud infrastructure, and build tools that are built and maintained by third parties. This ecosystem is known as the software supply chain and it involves numerous, globally-distributed participants building software used at various phases in the development process. The 2025 Open Source Security and Risk Analysis report found that 97% of evaluated codebases contained open source software, with 100% of codebases in the EdTech, internet, and mobile app sectors containing open source software [4].

Given the massive scale of open source software and the software supply chain in general, it is not surprising that attackers are increasingly targeting this ecosystem. Sonatype reported that 34,319 new open source malware packages were identified in quarter 3 of 2025, representing a 140% increase from quarter 2 2025 [5]. Recent attacks such as Log4j and SolarWinds highlight the importance of securing this ecosystem and the consequences of a successful attack. Aside from the economic damages associated with these attacks, distrust in the software ecosystem will undoubtedly slow technological innovation. Thus, it is essential to develop tools and frameworks to guard against malicious actors.

II. Software Supply Chain Attack vectors

Software supply chain attacks are considered to have three major attack vectors: dependencies, build infrastructure, and humans [6]. These attack vectors span the entire software lifecycle, and thus, security risks exist in each phase of this lifecycle. This review will focus primarily on software dependencies and artifact registries, cloud infrastructure, and build systems.

A. Dependencies

Vulnerabilities and malware included in open source and third party dependencies represent a critical security threat. In many cases, unintentional vulnerabilities are included in these dependencies. While such vulnerabilities are generally discovered, patched, and made public, developers and administrators may be slow to patch. For example, many systems were still vulnerable to attacks such as Heartbleed and Shellshock after the patch was released, in some cases years after [7]. A recent analysis of major package managers found that technical lag is common, with the majority of fixed version declarations being outdated and a significant number of flexible version declarations being outdated [8]. While there are likely many reasons for this, research has shown a strong presence of technical lag caused by the use of dependency constraints in the JavaScript package manager, npm, suggesting that developers may be reluctant to update dependencies due to backwards compatibility [9]. Maintaining software ecosystems is costly, especially if updates are frequent [10], but the technical lag generated by missing updates can result in an increased risk of critical vulnerabilities.

Attackers may also attempt to infect programs by typosquatting, the practice of uploading a package with a similar name as that of a very popular package (ex. reacte vs react). If a developer makes a typo while installing the package (ex. *npm install reacte*), the malicious software is installed. This practice is common in software libraries and other areas of the internet. While the effectiveness at tricking users varies based on how the fake name is composed. Users are more likely to correctly identify typosquatting that adds characters to the name, and are

more likely to be deceived by permutations and substitutions of characters [**?**].

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is.

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor

group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I: Table Type Styles

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 1: Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

tnotes in the abstract or reference list. Use letters for table footnotes.

REFERENCES

[1] H. Hammouchi, O. Cherqi, G. Mezzour, M. Ghogho, and M. El Koutbi, "Digging deeper into data breaches: An exploratory data analysis of hacking breaches over time," *Procedia Computer Science*, vol. 151, pp. 1004–1009, 2019.

[2] P. R. Clearinghouse. (2025) Data breaches. [Online]. Available: https://privacyrights.org/data-breaches

[3] IBM. (2025) Cost of a data breach report 2025. [Online]. Available: https://www.ibm.com/reports/data-breach

[4] B. Duck, "2025 open source security and risk analysis report," Black Duck, Tech. Rep., 2025.

[5] S. S. R. Team. (2025) Open source malware index q3 2025: High-severity attacks surge. [Online]. Available: https://www.sonatype.com/blog/open-source-malware-index-q3-2025

[6] L. Williams, G. Benedetti, S. Hamer, R. Paramitha, I. Rahman, M. Tamanna, G. Tystahl, N. Zahan, P. Morrison, Y. Acar *et al.*, "Research directions in software supply chain security," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 5, pp. 1–38, 2025.

[7] B. Hammi and S. Zeadally, "Software supply-chain security: Issues and countermeasures," *Computer*, vol. 56, no. 7, pp. 54–66, 2023.

[8] J. Stringer, A. Tahir, K. Blincoe, and J. Dietrich, "Technical lag of dependencies in major package managers," in *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, 2020, pp. 228–237.

[9] A. Zerouali, E. Constantinou, T. Mens, G. Robles, and J. González-Barahona, "An empirical analysis of technical lag in npm package dependencies," in *International conference on software reuse*. Springer, 2018, pp. 95–110.

[10] S. Berhe, M. Maynard, and F. Khomh, "Maintenance cost of software ecosystem updates," *Procedia Computer Science*, vol. 220, pp. 608–615, 2023.