Plotting OSS-based Supply Chain attack strategies and the defense failure

Arpita Dinesh Sarang[1][0000−0003−4461−711X], Sang-Hoon Choi[2][0000−0002−9549−0887], and Ki-Woong Park[3][0000−0002−3377−223X] *

[1] SysCore Lab, Department of Information Security, and Convergence Engineering for Intelligent Drone, Sejong University, Seoul 05006, South Korea
`arpita.sarang@sju.ac.kr`

[2] SysCore Lab, Sejong University, Seoul 05006, South Korea
`csh0052@gmail.com`

[3] Department of Information Security, and Convergence Engineering for Intelligent Drone, Sejong University, Seoul 05006, South Korea
`woongbak@sejong.ac.kr`

Abstract. The supply chain attack, which targets open-source software, is currently the most discussed cyberattack. This is due to the recent open-source XZ utils and PyPI projects based attacks where the attackers employed similar strategies. A backdoor was injected in the libraries developed by these projects for future exploitation when software users installed them on systems. These attacks were executed by the trusted developers of these projects and influenced by unknown promoters. Therefore, there is a need to scrutinize Open Source Software(OSS) Security to protect the legacy of Open Software development for the future. This paper provides an overview of these two attack strategies through their case studies, which aid in the creation of a generic attack framework for supply chain attacks on OSS. We present the existing detection methods for OSS security in this work and their lacunas taking into account at various stages of a supply chain attack. This makes it necessary to build platforms that allow the current OSS security detection modules to be utilized in sequence. Therefore, we introduce an OSS security Detection platform that does not completely address the limitations of the detection technique, but when combined, they can maximize the effectiveness of an OSS security quality check.

Keywords: Supply Chain Attack · Linux distribution · Cyber Security · Supply Chain Attack as a Service(SCAaaS) · Open Source Software(OSS) Security · Software Bill of Materials(SBOM)

1 Introduction

A software-based supply chain attack is a cyber-attack that distributes software or its library with a backdoor that can exploit the critical data on systems they

* Corresponding author.

get installed on [1]. These backdoors are injected during the development process of the software or its library. They are induced by the developers with malicious intent so strategically that they remain undetected even after distribution. This cyber-attack has been in action since the first "Ken Thompson Hack" in 1984, which warned about the upcoming attacks through the great speech on " Reflections on Trusting Trust" [2]. Such attacks have been planned meticulously for many years and malicious code for them is induced periodically to the software. A global report highlighted that a total of 210,031 software packages were affected by supply chain attacks over the past two years [3]. The number of supply chain incidents on open-source software surged to 245,032 in 2023—a 178% increase from 2022 [4].

Open Source Software(OSS) is the potential target for the software-based supply chain attack. These software are widely consumed by the technical community due to the benefits they provide such as free license, customizability, availability, and so on. Also, the software developers take great interest in developing these software projects by contributing valuable suggestions through comments and code fixes on the software project development open discussions and support emails. The core project developers and maintainers consider the suggestions and notice the valuable contributions to OSS development. In this development process the core developer, maintainers or the developers outside the project comment or promote fixes cannot be trusted as they may have malicious intent. A little ignorance in evaluating development due to trust in these entities can lead to a Supply Chain Attack through OSS. An enhanced taxonomy for Open source software-based supply chain attacks (OSSCAs) that listed some affected vectors, include users, systems, project maintainers, and root nodes [5]. A multi-year social engineering initiative, the XZ Util attack on February 24, 2024 [6], and PyPI attack for extraction of the web browser and cryptocurrency data on March 29, 2024 were well-known OSS security breaches for the first half of the year. Due to the prolonged and organised nature of attacks, they are challenging to detect. Attackers typically introduce backdoors by embedding granular program blocks. By the time exploitation is discovered, it is often too late for the victims.

In this paper, we discuss the two famous OSS-based Supply Chain attacks i.e., the XZ Util and the PyPI Project attack with relevant case studies. These recent attacks on the OSS supply chain allowed us to study their attack strategies closely and draw a generic attack framework for OSS-based Supply Chain Attacks. With the advantage of this attack framework built based on the current attack strategies, we were motivated to study the existing OSS security mechanisms for the detection and their lacunas for such attacks. This assists in understanding why these existing defense mechanisms failed and where is the necessity for upgrade them.

The remainder of this paper is as follows: Section 2, provide insight based on previous detection techniques for OSS based Supply Chain Attack. Section 3, describe the OSS-based Supply Chain Attacks in detailed case studies. Section 4, conveys the Generic framework based recent attacks. Section 5, discusses the prevention and detection strategies and their lackings. Finally, Section 6, concludes our Learnings and future works.

2 Related Work

OSS projects are the most targeted entity for the software-based supply chain attack. Developers are welcomed to the software development environment of these projects based on their development skills, not their good or bad intentions. This allows the malware developer to participate in the development process easily. This is a cyber-security challenge that OSS development projects began to face due to rising supply chain attacks. This victimizes all the OSS users trusting and utilizing the software. When it comes to OSS the well-known ones are downloaded from trusted sites as they are free and can be more customised to use. For studies conducted on OSS security. Detection of supply chain attacks on such software needs to be reviewed thoroughly.

Table 1 summarizes the types of detection techniques introduced by the previous study. These studies have limitations and can fail in cases mentioned in section 3. The Detection techniques are based on various aspects of the OSS code as discussed. The Query-based detection type performs a scan on OSS code for malicious queries and performs a match malicious query database [16]. In [17] and [18], the library-based detection type scan the OSS project dependency libraries if they are outdated or malicious. Whereas, in [19], [20], [21] and [22]

Table 1: OSS Supply Chain Attack Detection Techniques

Related Work	**Technique Type**	**Limitations**
[16]	Query-based detection	It is dependent on source data integrity, normal queries with malicious arguments will be ignored, cannot detect the behavior of queries outside database.
[17], [18]	Library-based detection	Some resource might ignore the less vulnerable packages.
[19], [20], [21], [22]	Vulnerability-based detection	It can prove a package to be less vulnerable due to low vulnerability detected when compared with datasets available even though it is critically malicious.
[23]	Combination of different detection modules	Combining different modules for detection makes this technique type accurate can differ, third party contracts for different applications are risky due to delayed updates.
[24], [25]	Third party based detection	It depends on application detection capability and compatibility with OSS not causing delays.
[26], [27], [28]	Stakeholder-based detection	It is a quality analysis therefore, cannot be measured and trusted.
[29]	Network-based detection	This detection fails detect malicious packets when network traffic use encryption techniques, or the malicious traffic trigger not passed.

the Vulnerability-based detection type detects and matches with the Vulnerability database for flaws in OSS. For, instance SBOM is majorly consumed by vulnerability assessment tools. These three types of detection techniques depend majorly on data resource integrity. In [23], combination of different detection module types is to combine different detection tools available for attack-based malware detection. Third-party-based detection in [24], [25] is implementing the monitoring, protocol-based authentication techniques for security of OSS project development. While the Stakeholder-based detection in [28] is trying the detect the developer's intentions in the project. The Network-based detection in [29] of malicious packages or activity during network communication with other resources. Due to unpredictable patterns and the non-generalized design of Supply chain attacks, the detection techniques are not well defined. They depend on previous attack analysis outcomes and fail to detect new attacks that are polymorphic in nature and should be detected before distribution of the OSS stage.

3 Case studies

We conducted these case studies based on various analysis and resources to outline their attack strategies.

3.1 XZ Utils Project attack

XZ Utils Project(Backdoor target): In 2005, Lasse Collin launched Project LZMA Utils with a small team of engineers. The goal of the project was to create a library for data compression that would support the Linux operating system. This library uses the Lempel–Ziv–Markov chain algorithm (LZMA) for compression and decompression. The xz and lzma file formats can be compressed and decompressed using it. Other software also utilised this library to compress data. Later on, the project became known as XZ utils. Many developers and programming experts contributed to this project, since it's open source, by offering their insightful comments and useful code snippets. Lasse Collins had the authority to approve and merge the code blocks to the project since he was its primary developer and maintainer. Under Lasse Collin's guidance, several iterations of the XZ Utils were introduced throughout the year. These included the compromised versions 5.6.1, which was issued on March 9, 2024, and 5.6.0, which was released on February 24, 2024 [8]. The malicious file was present in these version's release tarballs. Through this version's backdoor, an attacker can command the system of the victim with administrator rights. Version 5.4.6 is presently the most stable version of this project that is available.

Malicious Actors: For years, a number of players worked their part to create this backdoor in the data compression library of the Linux distribution. Some made significant contributions by changing code blocks. While others applied pressure to have the backdoor code added. In this analysis, we attempt to arrange the attack's chronology in order to obtain the event sequence for the supply chain attack carried out on the Linux distribution's XZ Utils project. Major elements of this attack are contributed by the attack contributors mentioned above. We draw a timeline for the attack events(see Fig. 1). Jia Tan(with developer

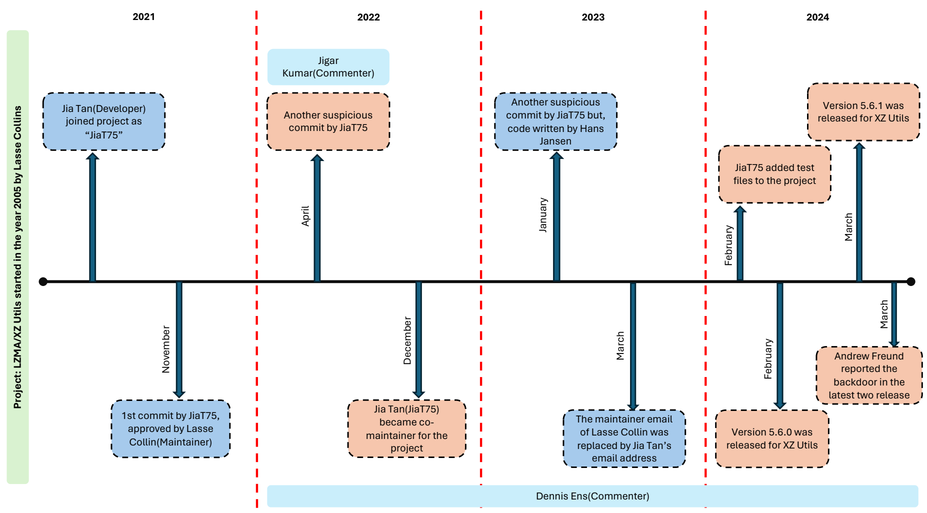
2021
2022
2023
2024
Project: LZMA/XZ Utils started in the year 2005 by Lasse Collins
Jia Tan(Developer) joined project as "JiaT75"
November
1st commit by JiaT75, approved by Lasse Collin(Maintainer)
Jigar Kumar(Commenter)
Another suspicious commit by JiaT75
April
December
Jia Tan(JiaT75) became co-maintainer for the project
Another suspicious commit by JiaT75 but, code written by Hans Jansen
January
March
The maintainer email of Lasse Collin was replaced by Jia Tan's email address
JiaT75 added test files to the project
February
February
Version 5.6.0 was released for XZ Utils
Version 5.6.1 was released for XZ Utils
March
March
Andrew Freund reported the backdoor in the latest two release
Dennis Ens(Commenter)

Fig. 1: Attack activity timeline for XZ Utils project

ID JiaT75) joined the XZ Utils project in 2021 and had a significant role in creating this backdoor for supply chain attacks in the XZ Utils library [9]. He provided Lasse Collin with various code blocks for XZ Util development that contained ignorant or obfuscated malicious code that initially was approved and integrated into the main project. Everything maintained on GitHub, including the "xz.tukaani.org" subdomain, was accessible to him. Lasse Collin was the only one with access to the main "tukaani.org" page [10]. JiaTan contributed a number of code blocks to the error-fixing and update of the XZ util project. In November 2021, JiaT75's first commit to xz utils was accepted and merged by Lasse Collin. There was a suspicious code edit in this commit that altered the secure "safe_fprintf()" to the "fprintf()" that code_1 indicated. This was done by employing an insecure function to perform a format string attack. An attacker may run arbitrary programs on the server, read variables off the stack, and cause segmentation faults or application crashes if they supply a format string made up of "fprintf()" conversion characters like %f, %p, %n, etc. as parameter values.

Later, JiaT75 contributed a second suspicious commit to XZ Utils in code_2 in April 2022. The code that ignored the memory and CPU use indication was hardware-related. This was done with the goal of not notifying XZ Util's consumers about any CPU or memory-related issues. By December 2022, Lasse Collin designated JiaT75 as a reliable co-maintainer for the XZ util project. JiaT75 committed a third controversial code commit on January 7, 2023. The code block that was merged was originally composed by Hans Jansen, another developer, rather than JiaT75. This commit carried out an IFUNC()-based exploit using glibc. In order to safeguard the offset tables and procedure linkage tables, the glibc feature known as IFUNC (GNU indirect function) acts as a resolver, helping to replace the function addresses to default at startup. Since these tables can be modified at startup time, the function from the external library was used to replace the pointer to the RSA_Public_Decrypt. JiaT75 gained greater authority when the official email address for XZ Utils code for component testing infrastructure was updated from "lasse.collin@tukaani.org" to "jiat0218@gmail.com" in March 2023. Eventually, on February 23, 2024, he added the test files "good-large_compressed.lzma" and "bad-3-corrupt_lzma2.xz". These two malicious files are executed when the macro "build-to-host.m4". Additional small contributors Jigar Kumar and Dennis Ens made contributions to the project by pressuring the inclusion of malicious code as essential functionality and by causing pointless conflicts with the patch release schedule during the years 2022 to 2024 [11].

Microsoft developer "Andres Freund"'s suspicion on the unusual CPU usage patterns and Valgrind failures enabled him to track down this supply chain attack. This curious Microsoft engineer saw that the SSHD process was utilising a lot of CPU power, which was causing the Debian Linux distribution's SSH-based login to function slowly. This gave Andres the opportunity to investigate the reason for the delay, conduct the necessary analysis based on his expertise, and report the issue to the relevant Linux distribution authorities i.e., oss-security via mail on 29 March 2024 [7]. He notified them with the high-level dynamics of the attack that he had discovered during his analytical process. He discovered that the XZ utils package contained a backdoor committed to the open source linux distribution package code. The backdoor was archieved in the download package known as the tarballs and targeted the x86-64 linux.

3.2 PyPI Project attack

Pytoileur(Backdoor target): The supply chain attack's target programme is the PYPI project and its users, through introducing the package pytoileur. The purpose of this package is to automate and manage tasks in Python projects. It mostly automates the work of repetitive development. By offering a streamlined workflow for routine operations like configuring development environments, handling dependencies, and conducting tests, it improves productivity. Include linting and testing tools in your code to guarantee quality. Automate the deployment procedures. This package was actually trojonised with malicious code. That targeted at the user's crypto-based application data and web browser data.

Malicious Actors: On May 25, 2024, a developer named "PhilipsPY" joined the PYPI(Python Package Index) project. PYPI is an open source library that allows Python developers to reuse code; there are a lot of packages available for download and installation. He uploaded the malicious code-encoded in Pytoileur package. After joining the open community discussion on stackoverflow, "EstAYA G" (promoter) began responding to other users' questions about Python development. By responding to questions on stackoverflow, he promoted the installation of the pytoileur package as solution to their problems.

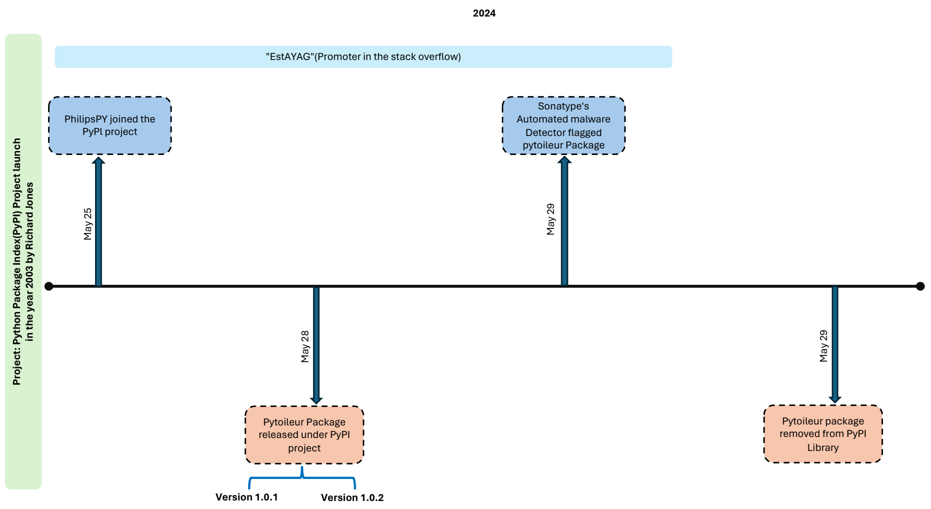
2024
"EstAYAG"(Promoter in the stack overflow)
Project: Python Package Index(PyPI) Project launch in the year 2003 by Richard Jones
PhilipsPY joined the PyPI project
May 25
Sonatype's Automated malware Detector flagged pytoileur Package
May 29
May 28
Pytoileur Package released under PyPI project
Version 1.0.1
Version 1.0.2
May 29
Pytoileur package removed from PyPI Library

Fig. 2: Attack activity timeline for PyPI project

The Project PYPI, included a package Pytoileur version 1.0.1 and version 1.0.2. This package was published on the same day by the author PhilipsPy in the PYPI library on 28 May 2024. We draw a timeline for the attack events(see Fig. 2). It had a total number of 264 downloads This package setup.py included a hidden malicous code. In its metadata, the package calls itself a "Cool package." The HTML webpage description refers to the package as a "API Management tool written in Python." The package is also trying to typosquat users of legitimate packages like "Pyston," as evidenced by the code's reference to "pystob," a now-defunct package (more on that later). The setup.py was injected with whitespaces after the print() statement in its code which can be avoided by the text editors those use the wordwrap. The whitespaces actually was hiding the code that executed the base64-encoded that connected to the remote server. Its payload stated that it connected "http: 51.11.140.144:8086/dl/runtime". Connecting to this external server it retrieved the windows binary called "runtime.exe". This file was undetected by the file scanning software and firewall. The "runtime.exe" was executed using the powershell and vbscript of the system. Once the execution of the "runtime.exe" is performed it dropped more suspicious file that modify the windows registry settings and deploy harmful payloads. This leads to the execution of the main.exe file which gathers information from the web browsers and the crypto currency services. webbrowsers such as Brave, firefox, google etc and well known crypto currency services such as the Coinbase, Binance, Paysafecard, crypto.com, exodus etc data were retrieved by this execution. Sonatype's automated malware detection engine for sonatype repository firewall detected and flagged the pytoileur package in PYPI repository on 28 May 2024. The Sonatype security researcher Jeff Thornhill noticed the line 17 whitespaces in the setup.py of the pytoileur package was hiding a code. This allowed him to further analyse this backdoor and track the issue as "SONATYPE-2024-1783" [12]. Later raised an alert to the PYPI admins and declared it a software based supply chain attack. The developer "PhilipsPY" and the promoter ID "EstAYA G" for pytoileur package were discontinued from development community [13]. Sonatype is a business that specialises in supply chain management software. The Sonatype company's automatic malware detection tool was one of their tools that assisted us in finding the backdoor in the earlier mentioned Python programme. This business develops tools and services for identification and the fixing of vulnerabilities in OSS with a focus on security of the OSS's weakness and danger. They support the creation of software security policies. They encourage community engagement and tool enhancement.

4 Generic framework

These days, supply chain attacks based on software have become commodified. On the dark web, configurations, and portions of code for exploiting software vulnerabilities during development are offered for sale. This commodification is majorly known as a Supply chain attack as a service(SCAaaS). Since supply chain attacks are becoming more accessible and commercialized, they can be easily replicated by targeting software development environments and taking advantage of the update process. Software libraries, pirated or open-source development tools, and their distribution networks are its primary targets. As a result, the frequency of attacks increased, having a widespread impact and making the process of detection and attribution more difficult. Based on the case studies this is a preliminary analysis.

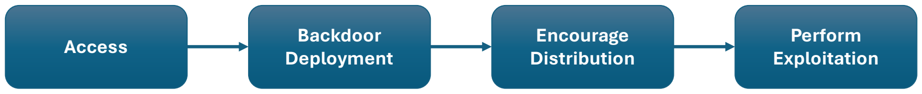

Fig. 3: OSS-based Supply Chain Attack strategy

There are two primary objectives of a supply chain attack: embed malicious code into software and to promote its use and distribution. As a result, there are two types of criminals involved in a supply chain attack: those who create and insert malicious code and those who promote or encourage on download of software that contains harmful code. These attackers work together to gain access to the target software project, deploy the backdoor code, encourage the software distribution of that project that includes the backdoor and finally exploit the backdoor to gain access to the system and the data (as shown in Fig. 3).

We provide a generic architecture for supply chain attacks that are OSS-based(see Fig. 4). This figure clarifies the trade-off that exists between the attacker, the vulnerability, and the exploitation of it. The attacking group's developers attempt to get associated with one of the targeted software's core development projects. The malicious developers begin injecting malicious code into the

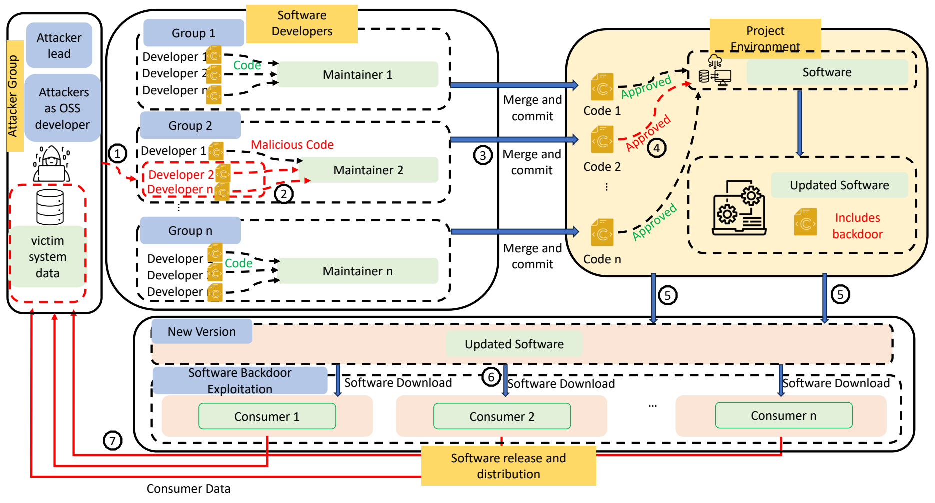

Fig. 4: OSS-based Supply Chain Attack framework

program's code as soon as they have access to it as authorized developers. The malicious code can be an argument, single-line code or a dependent executable file. To prevent being discovered, this code has been encoded or obfuscated. It is also possible to hide or remove logs produced by this type of programming. During development, the attacker group's developer deactivates alerts for administrative commands, memory utilization, or intrusion detectors. This is accomplished over a longer period by gradually incorporating the malicious code while remaining undetected by other legitimate developers or maintenance leads. Even if any potentially harmful code is discovered, the promoters defend it so well that others are led to believe that the code is necessary and urgent for the project. Since the maintainer or project lead must take the demands of the public into account, they approve the code because they think the developer is legitimate and has good intentions. Once the code is deployed into the software with all its dependencies satisfied, it becomes a backdoor for that software. Later when the software is released by the project authors on their official websites. The promoters working with the malicious developer and team initiate discussions suggesting to download and install the software as solution for most of the queries of users. The users of the OSS then start downloading and installing the dependent packages or software. Finally, when the package or the software is in use the injected backdoor gets executed on the user's system. This backdoor allows exploitation of user data by exposing it to the attacker groups.

To avoid OSS-based attacks, the development of software bill of materials(SBOM) for instance has made it possible to identify dependencies, which aren't directly part of an application but are installed or launched when the application is deployed as well as components that developers directly integrated into an application by importing them into its source code as a potential solution [15]. As a result, after researching current detection techniques, we suggest combining their use with development platforms during the software development lifecycle.

5 Proposed Platform

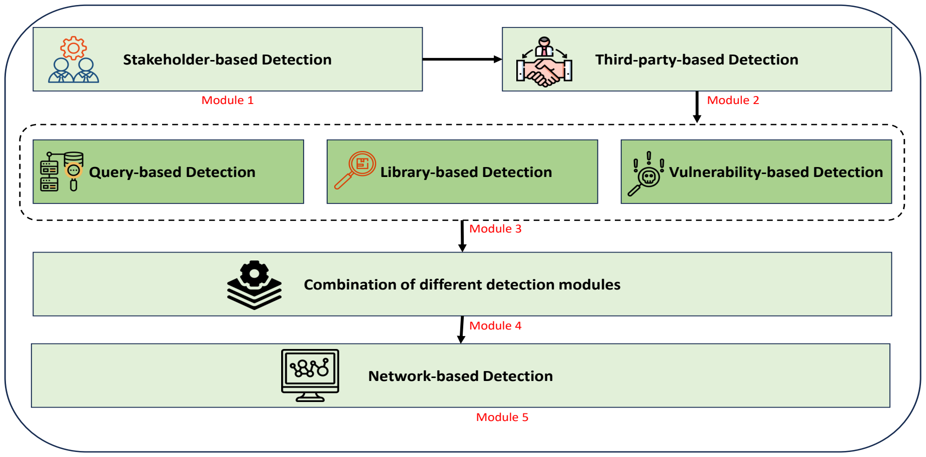
Stakeholder-based Detection
Module 1
Third-party-based Detection
Module 2
Query-based Detection
Library-based Detection
Vulnerability-based Detection
Module 3
Combination of different detection modules
Module 4
Network-based Detection
Module 5

Fig. 5: OSS-Security Quality testing Platform

Software security quality checks require detection prior to OSS software distribution. This makes the OSS development Lifecycle require a Security platform. The lack of data labeling for malicious semantics, the uncertainty of the attack design, and the irregular patterns of supply chain attacks based on OSS make the detection approaches in table 1 insufficient. We propose an OSS Security Quality testing platform using the prior study-based detection approach type in 5. This platform will provide software is secure quality tests based on all the detection approaches based on current standards by incorporating the best modules available. Module 1, The Stakeholder-based detection provides their rightful intent, background verifications, and control. Module 2, Third-party-based detection allows authentication, and control privileges and monitors the stakeholder's activity once they are part of the project. Module 3, is a combination of three different detection approaches Query-based, Library-based, and Vulnerability-based detection as they all focus on OSS semantics, dependency, and logic. After Semantic and logic check the OSS should go through Module 4, Combinations of different detection modules based on components of the OSS. For instance, if the OSS Utilizes outside resources the connectivity, data transfer and infrastructures should be analyzed well with security tools that are developed resource-based for security. Finally, Module 5, Network-based detection, scans for malicious package exchange by providing different combinations of benign and malicious parameters.

By using the OSS-Security Quality Testing Platform with the best detection techniques for each module, OSS development, and utilization, organizations can improve the security of their software and systems and increase their capacity for identifying and thwarting supply chain attacks on OSS.

6 Conclusion

The XZ Utils and PyPI Project attacks were multiyear strategized attacks. They followed the common attack strategy. This motivated us to create the Generic framework for the OSS-based Supply Chain Attack. These attacks remained undetected throughout the development and release process of the packages. The study detection strategies during the development process and after the release of the software is crucial which assisted us in finding the lacunas in these defences. The most critical lacuna for this strategy is ignorance based on trust. Protocols and Standards should be followed strictly to reduce the possibility of such attacks. Our proposed OSS Security Quality Testing platform is an attempt towards securing OSS. For Future works we plan to propose more detailed combinations of the modules in the platform and its feasibility as potential solution for OSS Security.

7 Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) (Project No. 2022-0-00701, 10%; Project No. RS-2023-00228996, 10%, Project No.RS-2022-00165794, 10%), the ICT R&D Program of MSIT/IITP (Project No. 2021-0-01816, 10%), and a National Research Foundation of Korea (NRF) grant funded by the Korean Government (Project No. RS2023-00208460, 60%).

References

1. Ladisa P, Plate H, Martinez M, Barais O. Taxonomy of attacks on open-source software supply chains. arXiv preprint arXiv:2204.04008. 2022 Apr 8.
2. Thompson K. Reflections on trusting trust. Communications of the ACM. 1984 Aug 1;27(8):761-3.
3. Statista Homepage, https://www.statista.com/statistics/1375128/supply-chain-attacks-software-packages-affected-global/
4. Statista Homepage, https://www.statista.com/statistics/1268934/worldwide-open-source-supply-chain-attacks/
5. Ladisa P, Plate H, Martinez M, Barais O. Sok: Taxonomy of attacks on open-source software supply chains. In2023 IEEE Symposium on Security and Privacy (SP) 2023 May 21 (pp. 1509-1526). IEEE.
6. Seah J. Sliced cables and cyber-attacks: How safe is our internet?. News Weekly. 2024 May(3164):22-3.
7. Openwall Homepage, https://www.openwall.com/lists/oss-security/2024/03/29/4
8. Security Week: Supply Chain Attack: Major Linux Distributions Impacted by XZ Utils Backdoor https://www.securityweek.com/supply-chain-attack-major-linux-distributions-impacted-by-xz-utils-backdoor/
9. Lins M, Mayrhofer R, Roland M, Hofer D, Schwaighofer M. On the critical path to implant backdoors and the effectiveness of potential mitigation techniques: Early learnings from XZ. arXiv preprint arXiv:2404.08987. 2024 Apr 13.
10. The Tukaani Project, https://tukaani.org/xz-backdoor/
11. Evan Boehs, https://boehs.org/node/everything-i-know-about-the-xz-backdoor

12. Sonatype, https://www.sonatype.com/blog/pypi-crypto-stealer-targets-windows-users-revives-malware-campaign
13. Black Hat Ethical Hacking, https://www.blackhatethicalhacking.com/news/
14. SNYK Security Database, https://security.snyk.io
15. America's Cyber Defense Agency, https://www.cisa.gov/sbom
16. Andreoli A, Lounis A, Debbabi M, Hanna A. On the prevalence of software supply chain attacks: Empirical study and investigative framework. Forensic Science International: Digital Investigation. 2023 Mar 1;44:301508.
17. Malka J. Increasing Trust in the Open Source Supply Chain with Reproducible Builds and Functional Package Management. In46th International Conference on Software Engineering (ICSE 2024)-Doctoral Symposium (DS) Track 2024.
18. Nahum M, Grolman E, Maimon I, Mimran D, Elyashar A, Brodt O, Elovici Y, Shabtai A. Ossintegrity: Collaborative Open Source Code Integrity Verification. Available at SSRN 4711134.
19. Singla T, Anandayuvaraj D, Kalu KG, Schorlemmer TR, Davis JC. An empirical study on using large language models to analyze software supply chain security failures. InProceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses 2023 Nov 30 (pp. 5-15).
20. O'Donoghue E, Reinhold AM, Izurieta C. Assessing Security Risks of Software Supply Chains Using Software Bill of Materials. In2nd International Workshop on Mining Software Repositories for Privacy and Security, MSR4P&S,(SANER 2024), Rovaniemi, Finland 2024 Mar.
21. Haque BM. An Analysis of SBOM in the Context of Software Supply-chain Risk Management (Master's thesis).
22. Mirakhorli M, Garcia D, Dillon S, Laporte K, Morrison M, Lu H, Koscinski V, Enoch C. A Landscape Study of Open Source and Proprietary Tools for Software Bill of Materials (SBOM). arXiv preprint arXiv:2402.11151. 2024 Feb 17.
23. Gokkaya B, Karafili E, Aniello L, Halak B. Global supply chains security: a comparative analysis of emerging threats and traceability solutions. Benchmarking: An International Journal. 2024 Mar 4.
24. Younis AA, Hu Y, Abdunabi R. Analyzing Software Supply Chain Security Risks in Industrial Control System Protocols: An OpenSSF Scorecard Approach. In2023 10th International Conference on Dependable Systems and Their Applications (DSA) 2023 Aug 10 (pp. 302-311). IEEE.
25. Nygård AR, Katsikas SK. Ethical hardware reverse engineering for securing the digital supply chain in critical infrastructure. Information & Computer Security. 2024 Jan 15.
26. Vashisth M, Verma SK. State of the Art Different Security Challenges, Solutions on Supply Chain: A Review. In2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA) 2023 Mar 14 (pp. 427-431). IEEE.
27. Turksonmez H, Ozcanhan MH. ENHANCING SECURITY OF RFID-ENABLED IOT SUPPLY CHAIN. Malaysian Journal of Computer Science. 2023 Jul 31;36(3):289-307.
28. Boughton L, Miller C, Acar Y, Wermke D, Kästner C. Decomposing and Measuring Trust in Open-Source Software Supply Chains. InProceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results 2024 Apr 14 (pp. 57-61).
29. Chauhdary SH, Alkatheiri MS, Alqarni MA, Saleem S. An efficient evolutionary deep learning-based attack prediction in supply chain management systems. Computers and Electrical Engineering. 2023 Jul 1;109:108768.