Supply Chain Vulnerability Analysis

[[1]](#footnote-1)

***Abstract*— Supply chain attacks have become a notable menace to the software sector, resulting in extensive harm and penetration. These attacks capitalize on vulnerabilities within the security protocols of third-party suppliers, resulting in widespread security breaches that can have extensive repercussions for both the suppliers and cloud service providers catering to them and their clients. A considerable amount of investigation has been carried out regarding the origins of these attacks, yielding detailed accounts of significant incidents such as the SolarWinds breach. Nevertheless, there is still a necessity for comprehensive visualization tools to aid in predicting future vulnerabilities based on past patterns. To address this gap, this research introduces a distinctive approach to monitoring the evolution of supply chain attacks targeting major third-party suppliers. We also incorporate dependency-oriented vulnerability assessments to enhance threat identification and mitigation**

***Index Terms*— *supply chain attacks, vulnerabilities, third-party suppliers, visualization, npm aduit, mitigation***

# I. INTRODUCTION

T

HE software supply chain (SSC) represents a sophisticated and intricate web of entities, including companies, individuals, and organizations, engaged in the creation and dissemination of software products, with a primary goal of leveraging existing software components. The SSC assumes a crucial position in the realm of software application development, distribution, and utilization on a global scale, a significance that has been magnified by the escalating complexity characterizing software development endeavours in recent times. Leveraging technological progress, enterprises are deeply interconnected in their endeavors related to product development and services on a worldwide scale, underscoring the pivotal role of software in fostering operational diversification. Software development, in particular, is experiencing a paradigm shift with an emphasis on the recyclability of code emerging as a new trend that streamlines the development process [1].

A software vulnerability is a defect in software that could allow an attacker to gain control of a system. Understanding software vulnerabilities is fundamental to managing modern security threats. An attacker can exploit a software vulnerability to steal or manipulate sensitive data, join a system to a botnet, install a backdoor, or plant other types of malwares. In addition, the attacker could use one network host to break into other hosts on the same network after penetrating one [2].

## A. Supply chain attacks: A Growing Threat

One notable and relatively recent software vulnerability pertains to the emergence of supply chain attacks, a trend that has garnered noticeable attention within the software industry. Specifically, a software supply chain attack materializes when cyber attackers tamper with the code embedded in third-party software components to infiltrate the 'downstream' applications reliant on them. These perpetrators exploit compromised software to abscond with data, sabotage targeted systems, or infiltrate other segments of the victim’s network through lateral movement.

The industry has observed a surge in cyber-attacks exploiting deficient supply chain methodologies. Presently, the primary risk related to supply chains, faced by organizations, emanates from open-source software. The open-source community offers numerous modules and packages that are extensively embraced by businesses globally, including those integrated into your supply chain.

Nevertheless, the drawback of open-source lies in its inherent vulnerability. This is largely due to the fact that it is authored by individuals who might lack the necessary expertise or financial resources to ensure its complete security. Another concern associated with open-source code pertains to ownership. Once a package is shared with the community, it becomes challenging to ascertain its rightful owner and the party accountable for its upkeep.

This loophole in your security framework arises because the open-source packages you incorporate could have undisclosed dependencies. This was precisely the case with NotPetya: a variant of conventional malware, NotPetya managed to breach systems worldwide by exploiting a widely utilized open- source accounting software. Consequently, it spread rapidly, causing turmoil in Ukraine and various prominent nations like the U.K., France, Germany, Russia, and the U.S [3], [4].

The study, entitled "The State of Supply Chain Defense 2023 Annual Global Insights Report" and released by BlueVoyant on December 11, indicated that the average number of supply chain failures grew from 3.29 in 2022 to 4.16 in 2023, with a projected 26% increase in supply chain breaches impacting organizations between 2022 and 2023 [5]. Gartner predicts that by 2025, 45% of organizations will have experienced a software supply chain attack [6].

Organizations persist in their inadequate preparedness to fully grasp the extent and characteristics of threats emanating from third-party providers. A notable obstacle lies in the assurance of timely mitigation of threats by suppliers in the supply chain, along with the effective dissemination of information pertaining to vulnerabilities and security concerns. The prevailing constraints encountered by organizations have been noted to extend beyond the year 2022. Of significant concern is the dearth of comprehension within firms regarding the pivotal role that third-party vendors and customers can assume in enhancing their cybersecurity readiness. The endeavor to enhance the security posture of third-party vendors has ascended to the second rank among primary challenges, marking an advancement from its previous third-place position in 2022. The challenge of conforming to regulatory requirements and guaranteeing compliance with third-party cybersecurity standards, formerly occupying the second position in terms of obstacles, has now shifted to the third spot [7].

The study also identified advancements in monitoring and reporting methodologies. Forty-seven percent of managers reported conducting monthly reviews of supply chain providers, an increase of 5% from the data recorded in 2022. Additionally, 44% of respondents indicated that they provide regular updates to senior management at least once a month on supply chain security risks, a significant increase from the 38% reported last year [7].

There is evidence that the importance of third-party cyber risk management in organizations is growing. Eighty-five percent of participants agreed that they have increased their financial resources allocated to supply chain or third-party online security in the past 12 months, a slight increase from the previous year. Conversely, only 6% of respondents reported that the risk management budget decreased during that period, 2% more than the data provided in 2022 [7].

In a supply chain assault, the assets of the supplier that are the target of the assault may encompass software, hardware, data, processes, and individuals. The majority of the 24 supply chain assaults (spanning from January 2020 to early July 2021) scrutinized in the 2021 ENISA Threat Landscape for Supply Chain Attacks report are supply chain assaults relating to software. More specifically, 62% of these 24 supply chain assaults have employed malicious software; 50% have been linked to well-known APT groups (such as APT 29), and 58% have been directed towards the theft of clients' data. This study concentrates on the examination and identification of software supply chain assaults (for example, the SolarWinds assault) due to their prevalence and extensive repercussions [8].

The well-documented 2013 breach targeting the prominent US retail entity Target serves as a prominent illustration of lateral movement. In this instance, threat actors compromised systems at Target’s provider of heating and ventilation services, subsequently leveraging the trusted status of the application to breach the retailer’s confidential information [9].

Any other ‘upstream’ part of an organization’s supply chain can be targeted, including application developers, publishers of off-the-shelf software like SolarWinds, API providers, and the open-source community.

SolarWinds, a SaaS provider headquartered in Austin, Texas, is the developer of Orion, software designed to manage and monitor business infrastructure. In 2020, cybercriminals buried a backdoor within a plugin distributed as an Orion update. Approximately 18,000 customers, including technology firms and government agencies, had their security compromised by the malicious update. However, those responsible focused on targeted follow-up attacks, compromising high profile organizations including cybersecurity firm FireEye. U.S. intelligence services suspect the involvement of Russian hackers [10]

## B. Elements and Methods of Supply Chain Attacks

Elements of a supply chain include third-party software used to interface with enterprise software, open-source platforms, cloud services, and past and present suppliers and vendors that have access to a company’s data and systems. Cyberattacks are typically directed at these third-party points of contact, as these are usually the weakest security elements in an enterprise’s supply chain [11].

Although such attacks can take various forms, the two most common ones with respect to a detrimental effect on the cloud infrastructure and services are:

**Malicious Code or Hardware Insertion:** Attackers insert malicious code or hardware during the manufacturing or distribution process. This can be done by compromising the systems of a trusted vendor or by intercepting and tampering with physical components during transit. Once the compromised component is integrated into the cloud service infrastructure, it can serve as a backdoor for attackers to gain unauthorized access, exfiltrate sensitive data, or launch further attacks [11].

**Compromise of Third-Party Service Providers:** Cloud services often rely on third-party vendors for functionalities such as authentication, encryption, or content delivery. If one of these vendors is compromised, it can have a ripple effect on the security of the entire cloud service. Attackers can exploit vulnerabilities in the compromised vendor’s systems to gain unauthorized access to the cloud service and its data, bypassing any security measures implemented by the organization [11].

## C. The Cloud and Its Associated Risks

The cloud leverages software capabilities for agility, setting it apart from traditional data center resource provisioning. The components of cloud services, particularly Software-as-a- Service (SaaS) applications, are composed of loosely coupled services. These software programs are often developed by a large, dispersed group of developers worldwide. Cloud Service Providers (CSPs) typically do not build software from the ground up but rather assemble cloud applications using components from global vendors [12].

Recent studies have indicated that at least 80% of a typical SaaS application consists of assembled parts, each presenting varying levels of risk. In 2015, for instance, 1.8 billion vulnerable open-source components were downloaded, with at least 26% of the most common open-source components containing high-risk vulnerabilities. The risks associated with multi-cloud systems appear to be continually increasing [12].

Cloud risks are intricately linked to the processes, procedures, and practices used to ensure the integrity, security, resilience, and quality of cloud services. Assessing and managing these risks can be challenging, as significant portions of computing services are controlled by external cloud providers [12].

The cloud supply chain represents a complex system involving multiple organizations and processes in delivering a cloud service. The cloud not only inherits risks from its underlying architecture—such as the internet and web services—but also introduces an additional layer of complexity. This complexity multiplies uncertainties within the cloud service delivery framework, creating potential weak links in the supply chain.

Third parties and supply chains introduce considerable vulnerabilities into any cloud environment.

The Wiz Research team conducted an extensive analysis of permissions granted to third-party vendors in cloud environments, focusing on direct exposure via IAM roles. Their research examined the permissions provided to over 1,300 AWS accounts across more than 40 popular third-party vendors. The key findings reveal that 82% of companies provide highly privileged roles to third-party vendors, and 76% have roles that allow for full account takeover. Additionally, over 90% of cloud security teams were unaware of the high permissions granted to these vendors. The study also found that the majority of these permissions are unnecessary, as vendors do not actually need them, and customer teams are often unaware they have granted such access. This research underscores the minimizing risks associated with third-party vendors in cloud environments, highlighting the need for customers to carefully scrutinize every requested permission and for vendors to adopt a "less is more" approach. Reducing permissions minimizes liability and helps prevent vendors from becoming targets for adversaries [13].

According to IBM’s 2022 Cost of a Data Breach Report, 19% of breaches were caused by supply chain compromises. The average total cost of a third-party breach was $4.46 million, which is 2.5% higher than the average cost of other types of breaches. Additionally, identifying and containing third-party breaches took an average of 26 days longer compared to the global average for other kinds of breaches [14].

## D. Motivation and Contribution

While extensive research has been conducted on the detection and prevention of supply chain attacks, this paper primarily focuses on providing visualizations of a detailed software vulnerability dataset. This dataset encompasses a large number of vulnerabilities spread across various common third-party vendors, offering both historical context and potential future trends. By analyzing this dataset, the aim is to reveal patterns and insights that can help in understanding the evolution of software vulnerabilities over time.

In addition to this analysis, the paper explores the results and usage of the common vulnerability detection tool, npm audit. Although npm audit is already a powerful tool for identifying vulnerabilities in Node.js packages, this paper proposes enhancing its functionality by adding a new feature—a vulnerability details visualizer. This visualizer will present detailed information about identified vulnerabilities in a more accessible and interpretable format. By doing so, it will simplify the analysis process, enabling third-party vendors to better understand the specifics of each vulnerability and to implement appropriate measures to counteract and resolve these issues effectively.

Moreover, the study encompasses the creation of a predictive model for evaluating the seriousness of identified vulnerabilities, leveraging historical data and machine learning methodologies to anticipate the potential consequences of new vulnerabilities. In addition to assessing severity, the model will offer recommendations for detection and mitigation strategies, aiming to raise awareness among vendors and assist them in adopting proactive measures to mitigate the adverse impacts of supply chain attacks in computing ecosystems.

By integrating intricate visual representations with predictive analyses, this strategy aims to enrich the comprehensive comprehension and handling of software vulnerabilities. This will empower third-party vendors to enforce more efficient security protocols, nurturing a culture of proactive security and ultimately diminishing the likelihood and consequences of supply chain attacks. The primary objective of this manuscript is to contribute to the continuous endeavors aimed at enhancing software security and resilience amidst the evolving landscape of cyber threats.

The research focuses majorly on providing descriptive and comprehensive vulnerability visualizations strategies, aiming to improve awareness and understandability about the trends of the same, serving as a pivotal precautionary tool.

The section of vulnerability visualization creates a cohesive vulnerability data dashboard, based on a dataset which contains vulnerability information that occurred in a variety of vendors in the year span of 2014-2022.

The next section aims to enhance the functionality of a common node package tool, NPM-audit. The research wishes to promote the addition of visualizations of the raw vulnerability data NPM audit fetches based on the package.json file

Lastly, we conclude with a ML model severity predictor for any vulnerability based on certain specific and easily accessible and known input parameters, informing vendors on the importance on the immediate remediation of the same.

# II. Related Works

## A. Risk assessment strategies and current limitations in Security Control

The recent evolution of cyber assaults against the software supply chain has revealed that cyber actors are increasingly focusing their attacks on open-source projects. Recent statistics show that attacks increased by 430% in the past year and again by 650% in 2021, turning the targets to open-source projects. The expansive use of open-source components within third-party and proprietary software makes the need to resolve these vulnerabilities extremely urgent. Since the software supply chain attacks are such complex criminal operations, a thorough risk assessment alone cannot allow one to ignore the

holistic approach to this issue, which is significantly underexplored in the literature. An important point that should be factored in more detail when assessing the impact of those vulnerabilities is their exploitability and the degree of damage each vulnerability might cause—one which could be aided by relying on the resources provided by others, e.g., the U.S. National Vulnerability Database (NVD) and Software Composition Analysis (SCA). When it comes to resources in the form of publicly available knowledge repositories, however, NVD, GitHub Security Advisories, and the Sonatype OSS index are quite useful in aiding the known vulnerabilities process. But depending on one tool for vulnerability detection can miss a few vulnerabilities that might result in disaster within the supply chain of software. It concludes by underlining the pressing need for future research efforts to focus on the development of more comprehensive frameworks of risk assessment, the strengthening of security within open- source projects, and the smooth integration of robust vulnerability management practices across the entire software supply chain [15].

The weaknesses within the software supply chain by more specifically examining the SolarWinds breach and, simultaneously, providing plausible solutions. The article recommends the adoption of the NERC CIP standard in many configurations through management and vulnerability assessment, such as CIP-010-3, and in personnel training with the inclusion of system security augmentation, such as CIP- 004-6. The important steps taken include keeping the software patched up and disabling USB ports that will mitigate vulnerabilities such as use-after-free and privilege escalation in the identified solutions of CVE-2020. Future approaches underline the necessity of living up to SOC2 standards—not only managing vendor risk but also coming from internal governance—reinforced with penetration testing practices, all well-customized for the industry. It further affirms that improved detection capabilities in Cyber Hygiene and CVE solutions will help towards the adoption and paradigm of tools that detect traffic anomalies and tag packets in the flow with maximal efficacy to identify and mitigate software supply chain attacks in real time [16].

# III. Vulnerability Visualization

In 2022, the number of Known Exploited Vulnerabilities more than doubled by the end of the year, from [311 CVEs to](https://vulncheck.com/blog/2022-cisa-kev-review) [868](https://vulncheck.com/blog/2022-cisa-kev-review) [17]. This research began with a careful data collection process. Researchers drew on an extensive data set from the CISA Known Exploited Vulnerabilities catalog for 2022[18]. This catalog is a compilation of documented security vulnerabilities that have been successfully exploited, as well as vulnerabilities associated with ransomware campaigns. This dataset includes detailed information of the security vulnerabilities from 125 unique vendors across a period of 8 years from 2014 to 2022. The dataset brings a more up-to-date view, digging deeper into the product of the vendors, a description of what makes the specific vulnerability risky in nature, the required action to be taken for safety measures, and the severity of each vulnerability.

## A. Dataset Explanation

This dataset comprises 12 significant key attributes.

**CVE Identifier:** An alphanumeric string that designates a Publicly Disclosed vulnerability, following the format outlined in the CVE Record Format established by CNAs [19]. A CVE record is categorized as Reserved, Published, or Rejected, serving as a distinct identifier for each vulnerability [19].

**Vendor Project:** A third-party entity linked to the vulnerability, typically an external individual or organization offering a service or technology under a contract. Under IT, these parties usually provide technology that an organization uses to store, process and transmit data and thus make operations easier.

**Product:** Products provided by the vendor which are related to the vulnerability; they allow understanding security weak points of various products and, hence perform vulnerability management in a more focused way.

**Vulnerability Name:** This is a descriptive title for the vulnerability where it is identified with its type and characteristics. This helps in understanding the scope of the vulnerability and its possible consequences.

**Short description**: Briefly describe a little about the vulnerability: type, for example, buffer overflow or SQL injection, nature, the software affected, and the results of the attack that could happen, like data exposure or remote code execution. Additionally, it can briefly describe how the vulnerability is currently being exploited, so organizations understand it quickly and focus on mitigating the vulnerability.

**Publication Date:** The public release date of a vulnerability is important to realize exactly when a risk was open to the world. It functions as a referential date by which professionals and organizations can keep themselves updated and act promptly on known, exploited vulnerabilities.

**CVSS:** Common Vulnerability Scoring System represents an openly available framework that scores and enumerates attributes of security vulnerabilities in information systems. The base, temporal, and environmental enumerations combine separately to produce a single score, ranging from 0 (least severe) to 10 (most severe).

**CWE:** CWE is Common Weakness Enumeration for the weakness and vulnerabilities classified in hardware and software literature. It gives a language—a common one— to communicate the causes of security vulnerabilities in code, design, or system architecture. Each entry of CWE represents an individual kind of vulnerability.

**Attack Vector:** Describes how an attacker could possibly exploit the vulnerability. This will greatly contribute to determining the accessibility and exploitation possibilities and give vital insights on how to develop mitigation strategies. These might serve as vulnerability vectors that inform mitigation strategies and attack surface assessment but also incident response planning by pre-empting and mitigating possible attack vectors.

**Severity Levels:** The severity levels are normally divided in line with the degree of consequences that the vulnerability can cause to the impacted systems and the extended organizational environment. These are generally based on factors such as exploitability, impact, and scope, further classified into low, medium, high, and critical, respectively. This is important stratification to organizations, as it supports prioritization in addressing vulnerabilities first; hence, it will deal with the ones that pose a larger risk and potential impact. Other than that, with complexity, this will aid organizations in developing all-rounded, sound cybersecurity approaches to protecting their systems and data.

**Cloud Component:** Specify which cloud component is affected by the vulnerability so that mitigation strategies are put in place for that component only. This aids in effective communication with the cloud service providers on the identified vulnerabilities and the measures put in place to secure these vulnerabilities on the affected cloud components. Moreover, it aids in ensuring compliance standards of risk management regulatory that require specific securing actions on certain cloud components.

**Category of vulnerability:** This involves the class or grouping to which vulnerability belongs. Categorization is extremely helpful and has many applications in terms of grounding remedial efforts by performing actions that a vulnerability type builds and may have in potential impact. Other than that, it enables security training initiatives to educate stakeholders on several vulnerability types, together with effective defense options.

## B. Data Visualization

Data visualization is essential for interpreting and sharing cybersecurity data, especially in handling vulnerabilities. Through converting unprocessed data into visual representations like charts, graphs, and maps, data visualization simplifies intricate information, emphasizes trends, and uncovers insights that may go unnoticed. Proficient visualization enables cybersecurity experts to promptly understand the extent and consequences of vulnerabilities, prioritize mitigation actions, and make well-informed choices to improve their security stance.

### 1) Vulnerability Severity Counts

The chart gives a summary of the quantities pertaining to vulnerabilities, based on their severity levels: High, Medium, and Critical. We label vulnerabilities against their respective CVSS scores. It is precisely these scores that offer a tremendously useful quantifiable measure of the severity of the vulnerability from 0 to 10, hence making the assessment process more organized.

The vulnerabilities with a High Severity rating, thus part of the 7.0 to 8.9 severity spectrum in the CVSS scale, are those considered critical vulnerabilities whose exploitation can result in severe harm. The Medium Severity vulnerabilities have an intermediate score ranging between 4.0 and 6.9, with moderate levels of vulnerabilities that may be dependent on user interaction or other prerequisites to happen. Critical severity vulnerabilities are rated in the 9.0 to 10.0 range and represent very high vulnerabilities: easily exploitable, potentially very harmful if exploited.

An approach treating the vulnerabilities based on their criticality forms the basis for prioritizing these vulnerabilities for remediation. High- and critical-severity vulnerabilities are those which require the quickest possible fixes to cut down on the exposure to possible threats against systems, security, and functionality, and concerned networks linked with them.

## 2) Distribution of Attack Vectors

The chart, in its vulnerability distribution by attack vectors, represents an important dispelling of confusion between Network and Local vulnerabilities.

In that network vulnerabilities are remotely exploitable, not requiring physical access to the specific device, the potential for exposure from every point on the globe to possible actors represents a serious threat. The prevalence of network vulnerabilities indicates the importance of implementing an efficient security measure in a network, invasion detection/prevention systems, firewalls, and secure network configuration. This need emerges considering that network vulnerabilities are always present.

In marked contrast, local vulnerabilities involve either physical or logical access to a device and present an entirely different set of issues and concerns. While the risks posed by local vulnerabilities are relatively lower, this is still pretty significant in an environment where insider threats are a

primary concern. The identification of local vulnerabilities gives reason for implementing a general solution of endpoint protection that includes antivirus protection, access controls, and regular security assessments, hence minimizing the risks due to potential insider threats or physical breaches which could utilize the identified vulnerability.

### 3) Vulnerability Discovery Over Time

This line chart will show clearly the progress of the vulnerability discovered over time, thereby helping in visualizing the trends in vulnerability detection within a given period. This analytics feature will not only convey to users the dynamics of the threat landscape and its evolution over various time periods but also enlighten them about periodicity and oscillations in vulnerability disclosure.

The periods of time associated with higher numbers of vulnerability discoveries may have peak years in which the majority are premiums, which can be linked to a number of factors: from amplified research through improved detections to heightened threat activity in the cyber domain. The opposite—an overall decreasing trend of vulnerability discovery—might point to increased security practices, reduced new vulnerabilities, or a shift in the cyber domain strategies of malicious actors.

These trends in the discovery of vulnerabilities may be used by any given organization for the estimation and preemption of upcoming security challenges and, therefore, marshal resources effectively for strategic deployment. For instance, if the rate of discovery for vulnerabilities keeps increasing constantly, an organization may decide to increase its security research and development investments in order to align with the emerging threats and vulnerabilities. Using historical data on vulnerability discoveries, any organization can make very informed decisions on financial planning and resource allocation for security research, development, and mitigation efforts, enhancing its overall cybersecurity posture and resilience against threats and cyber-attacks.

It may increase the need and vulnerability disclosures. Regulatory pressure on organizations to adhere to more stringent security regulations and compliance standards, enforcing enhanced defenses against cyber threats and vulnerabilities, is a possibility with a rise in vulnerability disclosure. This therefore places critical importance on the need for further security research and development investments for the evolving cybersecurity challenges and threats in today's fast-changing digital landscape.

### 4) Vulnerability Counts by CWE and Severity

The bar chart expresses the count of vulnerabilities, differentiated by their CWE categories and colored by severity.

Knowing the exact CWE categories helps to recognize common developer mistakes and provides a lead for secure development. Color differentiation by severity within each CWE category helps recognize which weaknesses are more critical.

Knowing the trending CWE categories and their severity distribution enables organizations to focus on the most critical ones and the root cause of problems. Secure coding practices, frequent reviews of code, and a security training framework will go a long way in reducing the presence of such common CWEs to a large extent. Knowledge of common CWE categories benefits the secure software development process by lowering the introduction of new vulnerabilities. The potentials of developers who are taught about common vulnerabilities and techniques for secure coding get reduced to a considerable extent.

### 5) Severity of Vulnerabilities of Cloud Components

The sheer complexity of securing IT environments has risen already because too many organizations have embraced cloud solutions. If a trend in adoption by many organizations currently characterizes this area, the potential overlooked attack paths within an organization's IT infrastructure are worrisome [20].

Prioritization by severity can be accomplished using visualization tools. This is more important in complex settings, including public and hybrid clouds or large enterprise IT infrastructures. Effective prioritization would be required as resources would always be at a premium while organizations would be faced with high-risk vulnerabilities needing to be mitigated without much delay [21]. A bar chart showing the vulnerability distribution by severity ranking across the different components of cloud systems can be drawn as shown.

**IaaS (Infrastructure as a Service)**

IaaS sells virtualized computing resources that enable customers, via the Internet, to rent and manage important components of infrastructure without having to own physical hardware. The main thing on offer in this case is virtual machines, storage, and networking resources—users retain control over operating systems, applications, and configurations [22]. Some vulnerabilities include misconfigured VMs, cloud storage problems, no cloud infrastructure visibility, shared tenancy issues, and insider threats from employees at cloud providers.

Mitigation Strategies: The state-of-the-art access controls and multi-factor authentication should be ensured against those vulnerabilities. NSGs should also be working on both incoming and outgoing traffic to VMs. VM disks and data at rest have to be encrypted. Monitoring, logging, and security incident response should be established. Regular security assessments and penetration testing are also highly recommended [23].

**PaaS (Platform as a Service)**

It is the platform that PaaS offers, including not only the infrastructure—virtual machines and storage—but also development and management tools that operate at a high level of abstraction from the underlying infrastructure. Basically, the role of PaaS would be to provide a platform for creating, testing, and running applications. As designed, it is between Infrastructure as a Service and Software as a Service, so PaaS will be perfect for network applications, APIs, and general software solutions [22]. Examples of vulnerabilities include platform vulnerabilities in the underlying infrastructure, application vulnerabilities in custom code and configurations, insecure interfaces and API, misconfigured system settings, and exposure of sensitive data and secrets [24].

Mitigation Strategies: Secure coding practices, regular application testing, code repositories, and setting up container image security, combined with the visibility and control of cloud use by a Cloud Access Security Broker, workload security by means of Cloud Workload Protection Platforms, and threat modeling for risk identification and mitigation are recommended for mitigation against these PaaS vulnerabilities [25].

**SaaS (Software as a Service)**

It is a service that provides full software applications over the Internet, accessed by users through web browsers only [30]. The software is maintained and hosted by the service provider. This ranges from a very wide spectrum of applications from empowerment tools down to very specialized business software tools [22]. Some of the vulnerabilities include unauthorized access and account takeover, insider threats from employees at the SaaS provider, data breaches and loss, and compliance and audit failures [24].

Mitigation Strategies: Regarding SaaS vulnerabilities, proper identity and access management controls should be in place; the data at rest and in transit needs to be encrypted; configurations should securely follow vendor-recommended best practices; incident response and disaster recovery plans should be framed in a proper way; and compliance can be assured through regular audits [24].

**Management and Governance**

This layer deals with orchestration and management across cloud services, security policies, and compliance. In this case, the vulnerabilities to management and governance may source from the following areas: not transparent across cloud environments, misalignment of business and IT teams, and poor security policies and controls [26].

Mitigation Strategies: The strategies that offset at the management and governance layer for mitigating vulnerabilities are aimed at developing a comprehensive strategy and roadmap on cloud security, baseline setting, security requirements, and implementation on cloud services; implementation of the CSPM; ensuring the identification of risks, clarity in roles and responsibilities, and accountability of cloud security; provisions for cloud security training and awareness among employees [26].

### 6) Vulnerabilities by Category for Each Cloud Component

Technical analysis of the data from the CISA Known Exploited Vulnerabilities catalog allows the trending of clear lines regarding the severity of the vulnerabilities and their distribution across different cloud components. All these trends are important to understand for formulating pinpointed security measures. Control is ensured through strong security measures at the network and data levels in the IaaS environment, by secure configuration and following secure development practices in PaaS, through stringent application security in SaaS and by a comprehensive security approach in management and governance.

The pie charts display the distribution of vulnerability categories for each cloud component. The vulnerability categories across different impacted cloud components:

There exist 11 key divisions according to the dataset within cloud elements that, while interconnected, jointly influence the entirety of cloud architecture. These divisions encompass Network, Code Execution, Application, Data, Identity and Access Management (IAM), Configuration, Access Control, Cryptography, Authentication, Denial of Service (DoS), and Hardware.

**Network**

*IaaS:* Some of the vulnerabilities that threaten an IaaS network include but are not limited to those arising out of unpatched network hardware, poorly configured networks, and a lack of proper monitoring of such networks. Such flaws may lead to network sniffing, man-in-the-middle attacks that have been manipulated, and even illegal access. Some of the ways to mitigate these vulnerabilities include the use of intrusion detection systems, coming up with secure designs for the networks, and ensuring network devices are updated in good time.

*PaaS:* Weaknesses in network configuration with PaaS may reveal the underlying infrastructure and give access to unauthorized entities. Combat associated risks by using secure network protocols, segmented networks, and periodic assessment of network vulnerability.

*SaaS:* Since applications based on SaaS suffer, like most network solutions, from failures that may lead to access by applications and compromise other applications' integrity, resulting in a leak of data and service interruption, the point is to apply a set of safe network configurations, apply network security monitoring constantly, and encrypt data at transfer.

*Management and governance:* Loopholes in the network of management interfaces can be exploited to obtain unauthorized access and control over cloud resources. Risk mitigation can be done by network segmentation, secure management interfaces, login controls, and VPNs.

**Code execution**

*IaaS:* Code execution vulnerabilities in IaaS environments expose virtual machines, containers, and other infrastructure components to compromise, possibly leading to system compromise and review data breaches. Mitigations for this risk include hardening configurations, intrusion detection systems/intrusion prevention systems, and security at the hypervisor layer, in addition to runtime security monitoring.

*PaaS:* PaaS has code execution vulnerabilities, and attacks in it may compromise the tools and services of the platform, impacting the security of apps developed and deployed on that platform, etc. Mitigating techniques include the implementation of container security solutions, regular code reviews, and following secure development processes with secure coding practices and vulnerability assessment.

*SaaS:* This would be unauthorized activities against SaaS applications through the exploitation of vulnerabilities within the code, which could lead to data and application functionality compromise. To that effect, the web application firewalls should be in place, security controls implemented at an application layer, regular penetration tests carried out, and security patches to the application applied on time.

*Management and Governance:* Unauthorized changes in cloud configurations would disrupt the service management due to the vulnerability of code execution in the management interfaces. The mitigation strategies to the risks would be the protection of the management apps, setting up the least privilege access controls, monitoring for suspicious activity, and finally multi-factor authentication to add an extra layer of security.

**Application**

*IaaS:* Combined with the base vulnerabilities of IaaS, application vulnerabilities can increase the exposure of data and services to threats such as code injection, application servers that are unpatched, insecure APIs, and so on. This can be enhanced with the additional use of web application firewalls, secure coding techniques, application security testing at regular intervals, and runtime application self- protection.

*PaaS:* Weak application security and unsecure development environments may be a result of the exploitation of application vulnerabilities in PaaS. This could threaten the integrity of the application. It practices vulnerability scanning, secure coding, and secure development lifecycle processes—this includes source code reviews, threat modeling, and secure configuration and change management.

*SaaS:* SaaS application vulnerabilities cause service disruption, illegal access, and data breaches. It not only puts user information at risk but also opens a door for an attack on application availability. Introduce stringent controls for application security; execute frequent security audits and penetration testing. Provide multi-factor authentication for added protection.

*Management and Governance:* Severe vulnerabilities in management interfaces may further induce misconfigurations into the application, resulting in a disruption in cloud operations, thereby affecting the security of the cloud as a whole. Mitigation strategies for the threat include monitoring threats deeply with the help of SIEM systems, following secure development and performing regular security assessment, including penetration testing.

**Data**

*IaaS:* Vulnerabilities in data within IaaS expose insecure storage and transmission, with poor encryption that may lead to unauthorized access and problems of compliance with regulations. In this regard, strong cryptographic standards such as AES-256 should be used at rest and transit. It provides secure data storage solutions in terms of encrypted cloud storage, yet at the same time, it also provides granular data access controls, which limit this to only valid users. Again, DLP solutions are there to prevent accidental leakage of data.

*PaaS:* The data vulnerabilities within PaaS may be a result of insecure development methods, weak isolation of data, or poor security around the APIs. This can result in a data breach and potential regulatory fine. These risks can be hugely mitigated through secure management of data along with periodic security audits, data encryption, and robust mechanisms for authentication around APIs.

*SaaS:* Data vulnerabilities within SaaS could be opening up user data due to insecure configuration, lack of multi-factor authentication, and vulnerabilities within an application; hence, unauthorized access to the information in data leaks and breaches only act to compromise user privacy. Mitigation strategies would involve strong data encryption, strict data access policies, regular security audits, penetration testing, and enforcing multi-factor authentication for user accounts.

*Management and Governance:* Data vulnerabilities to management interfaces may expose sensitive configuration data and access credentials, which could result in unauthorized changes of cloud configurations and give the overall disruption to cloud security. Therefore, mitigation provisions involve data encryption for sensitive management data, secure access controls to the data, periodic security audits, strong passwords, and possibly multi-factor authentication for access to management consoles.

**IAM (Identity and Access Management)**

*IaaS:* IAM vulnerabilities in IaaS may lead to access by unauthorized users to VMs, storage resources, or infrastructure components; privilege escalation; spoofing of identity; or just overall compromise of the IaaS environment. Introduce multi- factor authentication for accounts belonging to users, implement least privileges, have regular IAM audits, and consider PAM solutions in order to get better control over privileged accounts.

*PaaS:* IAM vulnerabilities in PaaS may make the platform services available to unauthorized users. This could potentially make development tools and sensitive data at risk and deployed applications dangerous. Enhance security through a

strong authentication mechanism, role-based access control, and proactive monitoring of IAM configuration settings and all user actions within the environment.

*SaaS:* IAM vulnerabilities in SaaS could be the door to unauthorized clients' sensitive data or user accounts, thus exposing them to data breaches or compromising their privacy and functional disruption in apps. Enhance the security by mandating multi-factor authentication for all user accounts, putting in place stringent password policies, and periodically reviewing and updating access controls within the SaaS application.

*Management and Governance:* IAM vulnerabilities exposed in management interfaces can facilitate unauthorized access to configuration settings and tools in the cloud, security policy manipulation, changes to cloud resources, and be an agent for overall disruption of cloud security. Introduce robust IAM policies with event logs, monitoring of management console activity, Role-Based Access Control, and frequent IAM audits necessary for addressing access control weaknesses.

**Configuration**

*IaaS:* It is possible for misconfiguration in IaaS to expose different areas to a lot of threats, including unauthorized access and data breaches on virtual machines, storage resources, and network infrastructure. This comprises default configurations of known vulnerabilities, insecure settings inside security groups, and poor management. Mitigation strategies involve frequent configuration audits, automated configuration management tools, observing best practices like disabling not in use services, and applying security patches in time.

*PaaS:* There might be several kinds of misconfigurations within a PaaS environment setup that may expose platform services and APIs, which potentially endangers an application's security. It can also bring about inefficient usage of resources and thereby performance issues. Mitigation involves the use of configuration management tools for uniform configuration, sticking to secure practices specific to the PaaS platform, and regular audits with remediation.

*SaaS:* SaaS misconfiguration can expose user data and affect the execution functionality of applications, leading to data leakage and unauthorized access. This includes weak access settings and poor configuration for data encryption. Some of the mitigation techniques to counter these weaknesses include setting guidelines for secure configurations, periodic security assessments, and configuration checking using automated means that provide real-time monitoring.

*Management and Governance:* Misconfigurations to the management interface could have severe impacts, as this is where access and security settings for the entire cloud environment are kept. Whatever errors done here could lead to unwanted changes that may cause disturbances to the cloud services. Mitigation involves the implementation of configuration management tools for consistent configurations that allow continuum monitoring of activities in enforcing secure practices like strong passwords and least privilege access.

**Access Control**

*IaaS:* Access control vulnerabilities to IaaS mean unauthorized access to virtual machines, storage resources, and network infrastructure. This allows attackers to do a lot of actions, such as data theft, malware deployment, disruption of services, or full control over the infrastructure. Such common vulnerabilities include weak access control lists, less implementation of RBAC, and poor privileged account management. These can be mitigated by designing RBAC, periodic reviewing of access control lists, access management tools automation, and possibly multi-factor authentication.

*PaaS:* This type of access control vulnerability may result in unauthorized access to platform services, APIs, and the underpinning infrastructure, hence compromising the security of applications by probable data breaches. The mitigants will involve strong access control mechanisms through RBAC, IAM integrations, API keys, access control auditing, and enforcement of secure configurations on platforms.

*SaaS:* May introduce access control vulnerabilities in SaaS to sensitive user data and application functionalities, hence causing data breaches or user privacy compromises. It involves robust access controls, security assessments, and automated access management tools inside the SaaS application.

*Management and Governance:* Weak access control in the management interfaces may expose an entry point for unauthorized users into cloud configuration settings and utility tools, thus putting the security of the cloud into jeopardy. The mitigation strategies include designing a strict access control policy, enabling activity monitoring over time, and periodic access audits of all management interfaces.

**Cryptography**

*IaaS:* Cryptographic vulnerabilities in IaaS can give hackers an upper hand in gaining unauthorized access by exploiting weaknesses. Weak encryption algorithms and old cryptographic protocols have associated risks. Tremendous vulnerability has, in theory, directly decrypted data in a strong case; this weakens the whole security of the data. As a countermeasure, always use a powerful encryption algorithm such as AES 256 for data at rest and in transit. Embed secure key management, such as rotating keys and rest encryption. Follow up regularly for cryptographic protocol upgrade to accommodate newly identified vulnerabilities.

*PaaS:* Cryptographic flaws in PaaS can result in the insecure exposure of the security of platform services and applications. Weakly encrypted data inside the platform or in the deployed applications may be leaked. Enforce good encryption practices

applicable to data at rest and in transit inside a PaaS environment. Proper key management should be in place for the encryption keys. Finally, cryptographic assessments should be performed at regular intervals for detecting and rectifying vulnerabilities.

*SaaS:* Cryptographic vulnerabilities in SaaS may make unauthorized access to users' data more likely. In this case, the attackers may take advantage of weak encryption in the SaaS application and jeopardize data integrity and privacy. This can be mitigated with strong industry-standard encryption of data both at rest and in transit, and by enforcing good key management practices. Key reviews shall be performed on a regular basis to detect and contextualize the vulnerabilities.

*Management and Governance:* Cryptic vulnerabilities in the management interfaces may reveal highly sensitive data, including configuration settings and credentials. Weak encryption may facilitate the compromise of cloud governance by attackers. To this end, ensure the use of strong encryption standards on the data in management interfaces. Enforce safe key management practices for any keys in use for the purpose of encryption. Subject the management interfaces to regular cryptographic assessments for the identification and mitigation of vulnerabilities.

**Authentication**

*IaaS:* Authentication weaknesses in Infrastructure as a Service can leads to unauthorized access, identity theft, and privilege escalation, which risks the security of an infrastructure. Enforce strong password policies, multi-factor authentication, and secure session management.

*PaaS:* Authentication vulnerabilities in Platform as a Service may lead to unauthorized access to the services on offer at the platform and development environments, hence identity spoofing with possible data breaches. This can be mitigated by putting in place a strong authentication mechanism, regular security assessments, and session management.

*SaaS:* Authentication vulnerabilities can be exploited to gain unauthorized access to applications and user data in Software as a Service, thus compromise ng the security of applications and users' privacy. This, therefore, puts organizations at the forefront in enforcing MFA, stronger password policies, and implementing good session management practices.

*Management and Governance:* Authentication vulnerabilities in the management interface can lead to unauthorized access to management controls, which could further work to the detriment of overall cloud governance. Hence, tight authentication policies, continuous monitoring, and adoption of secure session management practices become quite critical in this regard.

**Denial of Service (DoS)**

*IaaS:* DoS attacks can swamp cloud resources with huge traffic, making them unavailable to valid users, hence service disruption. In an IaaS set up, DoS attacks may target a virtual machine, storage resources, and network infrastructure. The outcome will affect user access and areas of application functionality. Rate limiting mechanisms should hence be applied with anti-DoS protection tools. This could be helped by redundancy and load balancing in the IaaS infrastructure to prevent them being vulnerable to DoS attacks by distributing the traffic to it.

*PaaS:* DoS vulnerabilities can be triggered with malicious traffic against platform services, in turn affecting the availability and performance of applications on top. Mitigation strategies include rate limiting on platform services and anti- DoS tools custom-built for PaaS environments. Autoscaling measures can lend a hand in the automatic provisioning of resources during high traffic periods.

*SaaS:* DoS vulnerabilities can make the application unavailable to users by simply flooding it with traffic. Strategies for mitigating this include rate-limiting mechanisms in the SaaS application itself, along with anti-DoS tools offered by the SaaS provider.

*Management and Governance:* DoS vulnerabilities in the management interfaces will hamper cloud management services and may also impact the administrators from monitoring the environment. Some mitigation measures include rate limiting on management interfaces and anti-DoS protection for management consoles. The redundancy of management services can be done by deploying on multiple servers or regions to make sure access to them in case DoS attacks happen.

**Hardware**

*IaaS:* Depending on the type of vulnerability and the cloud service model, hardware vulnerabilities in cloud environments can lead to different security risks, such as unauthorized access to data, disruption of services, or even erosion of security. The presence of any hardware vulnerabilities in IaaS is of great concern since a customer does not have any direct control over the physical hardware. Problems can result from physical tampering and side-channel attacks right down to hardware failures. IaaS providers, hence, must ensure that the systems are physically securely held in areas with restricted access, they carry out periodic updates on the firmware, and, if possible, use HSMs in the encryption process.

*PaaS:* Since PaaS is vulnerable to hardware, its exploitation could mean infrastructure disruption and service breach. PaaS providers must therefore abide by best practices for secure management of hardware and its regular firmware updating, with relevant physical security measures in place within its data centers.

*SaaS:* Well, much like IaaS, hardware vulnerabilities in SaaS might expose it to physical tampering and such hardware failures. The customer needs to select trusted SaaS vendors who have security measures in place and ask what their processes are regarding physical security. Besides, keeping current on the latest firmware, remembering updates, and possible vulnerabilities keeps data secure.

*Management and Governance:* Hardware vulnerabilities in the Management Interfaces can be a significant risk to the security and availability of cloud management services. Vulnerabilities that were exploited by the attackers could be used for gaining unauthorized access to the management tools, therefore disrupting cloud operations. Mitigating this are the measures of physical security, updating the firmware regularly, and using the best practices for securely managing and configuring hardware used for the cloud.

# IV. NPM Aduit

In the backdrop of vendor specific vulnerabilities and their effect on the cloud, the need to drive down to the specific project-based vulnerabilities are of high stake.

Specialized tools for software supply chain security are crucial in safeguarding the DevSecOps pipeline against attacks on the supply chain. This tool plays a vital role in identifying and addressing vulnerabilities such as dependency issues, open- source concerns, misconfigurations, insider threats, and license compliance gaps. Apart from risk detection, these tools enforce security protocols and ensure compliance across the software development lifecycle (SDLC) by scrutinizing each component, especially those obtained from external sources.

Three categories of DevOps solutions contribute to upholding the integrity and security of the software supply chain: Software Composition Analysis (SCA), Static Application Security Testing (SAST), and Container Security [27]

Software development teams benefit significantly from utilizing Software Composition Analysis (SCA) tools to meticulously monitor and evaluate any open-source code integrated into a project from the perspectives of licensing compliance and security vulnerabilities. These tools are adept at identifying open-source code, both superficially and in- depth, along with their direct and indirect dependencies, applicable licenses, and the presence of any documented security vulnerabilities or potential threats. Several companies provide SCA suites, open-source tools, and associated services—usually driven by community initiatives [28]. In general, these tools are deployed during development and build phases to detect vulnerabilities coming from third-party sources.

Supply chain threats include dependency confusion attacks, the introduction of malicious code backdoors into open-source packages, and even the compromise of build pipeline infrastructure. Security risks with open-source libraries and ecosystems are the norm, posing a significant risk for developers. One of the reasons for that growth was the open design of NPM—millions of free, reusable software packages were available. While openness presents advantages, it also introduces vulnerabilities, as proven by previous cases of how single packages can cause disruptions or become targets to software running on many devices. This brings us to considering the different risks that third-party dependencies enable, and hence we understand how individual packages can have an impact on large fractions of the entire computing ecosystem.

Recently, NPM security has garnered considerable attention in the media, with a notable focus on NPM packages within the ecosystem rather than the NPM registry itself. The escalating security concerns affecting developers and software underscore the necessity of comprehending methods to prevent supply chain attacks and other vulnerabilities linked to the software development lifecycle.

Academic research published on June 7, 2019, shared interesting insights about this topic. Upon installing an average NPM package, users inadvertently place trust in 79 third-party packages and 39 maintainers, significantly enlarging the potential attack surface [29].

Researchers at ReversingLabs discovered that more than two dozen NPM packages, some of which dated back to at least December 2021, included code meant to collect form data from end users of the applications or websites that were hosting the malicious packages. Researchers do not know the full scope of the assault, but the malicious packages are presumably used by hundreds of downstream applications and websites. Researchers reported that the NPM modules identified as part of the campaign have been downloaded over 27,000 times in total [30].

Upon closer inspection, evidences have been discovered of a coordinated supply chain attack, with a large number of NPM packages containing jQuery scripts designed to steal form data from deployed applications that include them [30].

A study, conducted on 1.63 million packages in the NPM repository, reveals six indicators of weakness in the open- source software supply chain [31].

Acting on these findings can help package maintainers and users make better security decisions and therefore safeguard their software against potential supply chain attack [31].

There were 2,818 domains whose maintainers had expired. An intruder has the ability to acquire an expired domain and utilize it to access the accounts of the maintainers, unless it is secured by two-factor authentication (2FA) [31].

About 2% of the packages included installation scripts. Install scripts are executed automatically prior to, during, or subsequent to the installation of a package. If they are compromised, they have the potential to facilitate malicious activities on host devices, including the transfer of user data, downloading of malicious payloads, execution of reverse shells, and the deletion of files and directories [31].

59% of the packages were unmaintained for two years. Furthermore, 44% of the maintainers were not active for two years. Unmaintained packages have a higher chance of getting compromised without detection. Inactive maintainers may be targeted with account hijacking attacks without their knowledge [31].

A small proportion of the packages possessed an excessive number of maintainers, thereby elevating the probability of at least one of the maintainers' accounts being compromised.

Certain packages possessed an excessive number of contributors, rendering it challenging for maintainers to monitor all modifications. An attacker may employ social engineering to establish themselves as a dependable contributor to such packages prior to introducing malicious code [31].

Top 1% of maintainers were overloaded and owned an average of 180 packages. Since they are more likely to overlook changes to any particular packages, attackers have a greater incentive to target such maintainers [31].

A security audit involves evaluating package dependencies to identify security vulnerabilities, thereby safeguarding users from risks such as data loss, service disruptions, unauthorized access to sensitive data, and other related issues. The NPM- audit feature is a crucial component of the NPM (Node Package Manager) toolkit, specifically designed for Node.js to conduct security audits on package dependencies.

## A. NPM Audit: Empowering Developer Awareness and Security

The NPM audit is one of the major activities that help developers become aware of any known security vulnerabilities in their applications. It provides actionable guidance to eliminate the identified risks, thus helping developers take immediate actions to ward off any potential threats.

By reading information in the project's `package.json` and `package-lock.json` files, NPM-audit compares each declared dependency against databases of known vulnerabilities, such as the national vulnerability database maintained by the npm registry. This approach provides a means for developers to have their application dependencies free from known vulnerabilities [32]

This package.json file stands central to any Node.js and thus, by effect, to understanding and working with Node.js, NPM, or even modern JavaScript. The information contained is used for manifest purposes within applications, modules, packages, and other details.

Dependency Management: This is basically what a package.json file does: it keeps a record of all dependencies involved in the NodeJS project. Among these listed dependencies are runtime dependencies— dependencies an application needs to function—and development dependencies, which are used during the phases of building, testing, and maintenance of the project [33].

It typically contains metadata, such as the project name, version, description, author, and license, in the package.json configuration file. This information allows developers to know the purpose of a project and users to track changes and hence versions, besides complying with licensing requirements [33].

*Script Execution:* Most of the time, any NodeJS project defines custom scripts in package.json to automate different tasks, be it running tests, starting the development server, building the project, or deploying an application. These scripts simplify common development workflows and promote consistency across different environments [33]

Using NPM audit, it provides detailed reports to the developer about vulnerabilities that exist within dependencies. It rates these vulnerabilities further by severity, stating clearly remediation steps, like upgrading to a safe version, patching, or removing entirely vulnerable packages. This feedback enables the developers to address security issues immediately, hence reducing the chances of exploitation [33].

In case of no vulnerabilities, NPM returns exit code 0.

One of the major advantages of NPM audit is the network effect. The more developers run the tool and fix their packages, the more secure the overall situation of the NPM ecosystem will become.

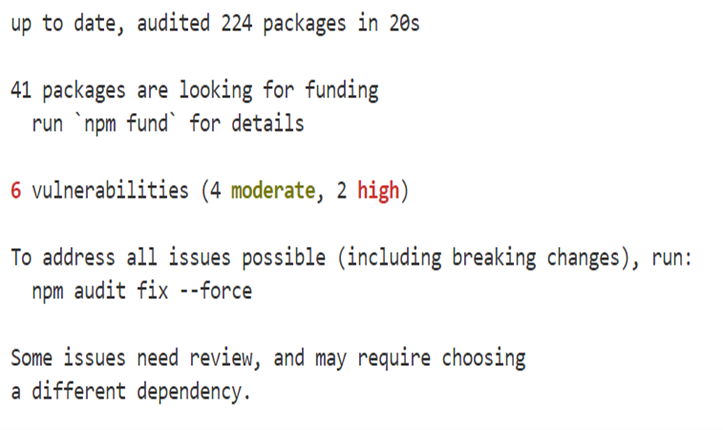
However, the visual analysis of the NPM audit data in detecting underlying patterns, enhancing understandability among a larger client base, etc., are some features that this tool has not yet implemented under its umbrella of functionalities.

A npm audit visualization dashboard requires the proper detection of the presence of packages with the highest count of vulnerabilities, their classification in accordance with common weakness enumeration, and the severity distribution assessment, among others. What we are trying to do in this work is provide a collocated NPM Audit Visualization Dashboard that will dynamically render such details based on the vulnerabilities checked in the uploaded package.json file.

## B. Overview of Functionality

The base of this visualization is NPM, a package manager that scans the package.json file in real-time. The uploading of the package.json file connected with the project will trigger this NPM audit feature, and such a process will generate an audit dataset. By further parsing this dataset, key-value pairs corresponding to several parameters can be obtained, which will then be used as the base for visualizations, thus giving full insight into the state of security related to dependencies to developers.

It will hence provide a better, more informed, and quantified decision and a comparison before adding any new package to the development pipelines.



### 1) Vulnerability Severity Analysis

*Graph Explanation:* The chart classifies the vulnerabilities into four classes of severity: low, moderate, high, and critical.

*Technical Specifications:* NPM audit describes vulnerabilities with a severity score. The graph consolidates the scoring to help in prioritizing remedial action. The company can start by addressing critical and high-severity vulnerabilities to minimize major risks so that this will keep the development environment of the project safe from any threats.

### 2) Package Vulnerability Analysis

*Graph Explanation:* This visualization displays which packages have been subject to the largest number of vulnerabilities.

*Technical Specifications:* Developers can use this to identify the weakest links amongst third-party/open-source packages by analysing the frequency of vulnerabilities in different packages. Understanding this forms a very critical basis while dealing with supply chain attacks and associated software vulnerabilities that can have high impacts on project confidentiality and financial stability.

### 3) Classification According to CWE

*Graph Explanation:* This classifier then classifies these detected vulnerabilities based on the CWE framework.

*Technical Details:* This facilitates the categorization of vulnerabilities by CWE categories, letting developers understand the risk exposure of their dependencies. A developer might, in such a case, make use of already developed detection and mitigation strategies against an issue if a vulnerability either falls into one of the top-10 CWEs or any other high-risk category. This gives a clear understanding not only of the nature of the vulnerabilities but also eases remediation with very specific and more relevant solutions.

### 4) Implementation Details

*Real-Time Data:* The visualizations are driven by dynamic information extracted from real-time results of npm audit, thus providing the most current security data to developers.

*Parsing of Audit Dataset:* Parsing of the audit dataset made available by npm audit is done for extracting details such as package names, severity levels, and CWE identifiers. This parsed information is further used in generating visualizations.

*Visual Representations:* There are already a number of important graphs on the dashboard, but many more can be implemented. All such visualizations are built to be user-friendly, insightful, and developer-friendly.

### 5) Advantages

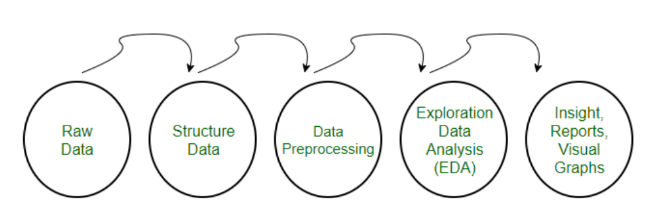
*Efficient Use of Resources:* Knowing how the severity is distributed would help companies use their resources efficiently by first tackling the most critical vulnerabilities.

*Identification of Vulnerable Components:* The package analysis graph assists developers in recognizing and addressing the most vulnerable dependencies, enhancing the overall security of the project.

*Risk Mitigation:* The CWE-based classifier provides a structured approach to vulnerability management, enabling developers to implement effective detection and mitigation strategies based on the type of vulnerabilities present.

# V. PREDICTIVE MODEL BASED ON MACHINE LEARNING

Machine learning models are algorithms that can identify patterns or make predictions on unseen datasets. Unlike rule-based programs, these models do not have to be explicitly coded and can evolve over time as new data enters the system.

 Scikit-learn is a machine learning toolkit that provides various tools to cater to different aspects of machine learning e.g. Classification, Regression, Clustering, Dimensionality reduction, Model selection, Preprocessing.

## A. Data Preprocessing

Data preprocessing is a very necessary step in the preparation of raw data for use within the machine learning model [41]. Its objective is to simplify the data complexities used in machine learning algorithms. Real-world-based data normally is missing, and inconsistent. These are due to human mistakes, unexpected events, faults, and many other factors [40]. Algorithms cannot use noisy and incomplete data since they are not usually designed for processing with missing information, and the noise conceals the real pattern of data [40]. These are the very reasons why data preprocessing needs to be carefully prepared for analysis.

### 1) Data Cleaning

Data cleaning is an intrinsic part of data science that ensures the information one is dealing with is correct and reliable. In the field of data science, derivation of insight and prediction from big and complex data sets are major tasks. Therefore, the correctness and dependability of the data used badly affects the reliability of the findings.

Data cleaning is a process that involves searching for errors, inconsistencies, and inaccuracies in data systematically, correcting them, and enhancing its completeness and quality. It is an exercise quite rigorous and extremely important for improving the reliability of the analyses, promoting accurate modeling, and eventually making informed decisions based upon credible data of high quality [39].

We had to handle missing value and remove unwanted observations due to duplicate/insufficient data. The data cleaning procedure was carried out on the basis of the required parameters for Machine Learning. We also fixed structural errors due to inconsistencies in data formats, naming conventions, or variable types. Maintaining consistent formats, addressing issues with names, and making sure data is presented uniformly [39]. Correcting issues with the structure improves the consistency of data and makes it easier to conduct precise analysis and understanding.

### 2) Data Integration

Data Integration is one of the data preprocessing steps, which involves integration of data and its subsequent combination into some larger database, for example, a Data Warehouse. This step becomes very important, especially when dealing with real-world problem-solving scenarios [37]. This becomes difficult to manage because of the variation in data format, structure, and meaning from one data source to another. Another data source may utilize different data categories, naming conventions, and data models, all of which may cause complications in an attempt to combine the data into a single view [38].

### 3) Data Transformation

Data transformation is the process of transforming unprocessed data into a format or structure that is better suited for analysis, to enhance its quality, and to align it with the needs of a specific task or systems. It is the process of improving the quality of raw data and ensuring that it complies with the specifications of a specific task or system by transforming it into a more suited format or structure for analysis [39].

## B. Comparative analysis of Machine Learning Models

Each model comes with intrinsic strengths, and the usability of each comes down to certain contexts.

**Gradient Boosting** is especially strong when it comes to very high predictive accuracy and when it deals effectively with intricate feature interactions through its ensemble methodology.

**Multinomial Naive Bayes**, on the other hand, is appealing due to its simplicity and computational efficiency, and therefore its performance sparkles for large datasets and text categorization.

**Random Forest** is a robust model which provides the ability to fight overfitting and a way to extract feature importance so that one may easily understand the data given.

Our preferred model is the **SVC Support Vector Classifier**, being optimal in the implementation of accuracy and highly effective on many dimensions when the number of features is higher than the number of samples. This is possible in the SVC method because of its knack to support various kernel functions: linear, polynomial, RBF, and sigmoid functions. These enable the SVC to capture complex non- linear relationships that are prevalent in the data, hence improving its adaptability and effectiveness. These characteristics of SVC are able to avoid the problem of overfitting by identifying the best hyperplane with the largest margin between classes, ensuring generalizability for new and unseen data and staying performant overall.

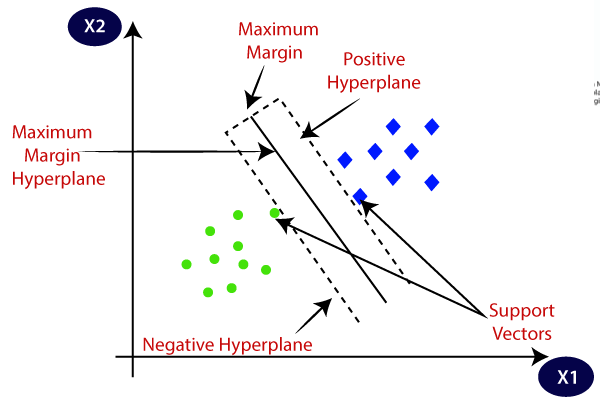
### C. Mathematical analysis of the SVM algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning [40].

Support Vector Classifiers, or SVCs (SVM is referred to as SVC in the implementation provided by scikit-learn), are a type of supervised machine learning algorithm that can be used for both classification and regression tasks. SVCs are a popular choice for many machine learning applications due to their ability to achieve high accuracy while still being computationally efficient [40].

SVMs work by mapping data points to a high-dimensional space and then finding the best boundary between the different classes. This boundary is known as a hyperplane, and the data points that lie on either side of it are said to be support vectors. The position of the hyperplane is determined by the support vectors. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane [40].

The below image shows a hyperplane segregating two classes under the SVM model:



**SVMs can be of two types:**

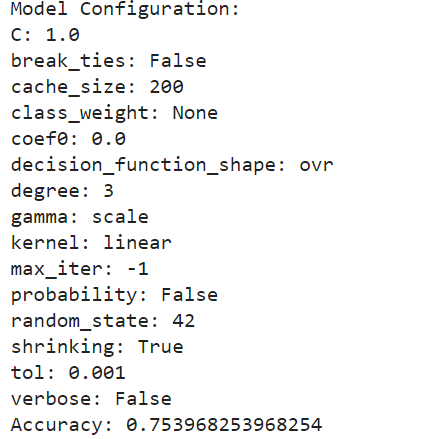
**Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier [40].

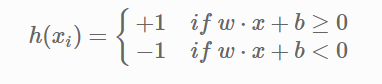
**Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier [40].

For the dataset we are using, the kernel set to linear gives highest precision and accuracy as the classification of the severity among the various classes do not involve very large levels of complexity and dispersion amongst themselves. Thus multiple (in our case 2) linear hyperplanes can be used to classify the severity data effectively.

The severity output can fall into any one of the three classes- medium, high or critical. Thus, two linear hyperplanes are required for this classification.





The hypothesis function h is defined as:

Any data point will either lie on the positive or the negative side of the hyperplane that separates the two classes. This point above or on the hyperplane will be classified as class +1, and the point below the hyperplane will be classified as class - 1.

The hypothesis function helps to determine the same [41].

NOTE:

|  |  |
| --- | --- |
| w | The weight vector normal to the hyperplane, which decodes the orientation of the hyperplane |
| x | The feature vector of the data point. |
| b | The bias term of the hyperplane, which shifts the hyperplane |

**Soft-margins:**

In a real time scenario, it might be possible that some data- points belonging to one hyperplane is placed on the incorrect side of the hyperplane or in other words is misclassified

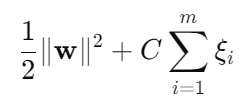
The soft-margin SVM allows some points to be on the wrong side of the hyperplane but tries to keep the number and degree of such errors small. It does this by introducing a concept called slack variables ξ, which measure how much a point is on the wrong side.

The goal of the soft-margin SVM is to find the best hyperplane by minimizing an objective function that balances two things:

*Maximizing the Margin:* Making the hyperplane as far away as possible from the nearest points (support vectors) of any class.

*Minimizing Misclassification:* Penalizing points that are on the wrong side of the hyperplane using the slack variables ξ.

The mathematical expression to minimize is:



1.    is the term that encourages a larger margin (a simpler and more generalizable model).
2. is the penalty term for misclassified points, where C is a parameter that controls the trade-off between margin size and misclassification penalty.

## D. Data Transformation

### 1) Encoding Categorical Values

Transforming categorical data into a numerical format is called encoding. It allows algorithms to straightforwardly identify and handle the data [42]. We use encoding for CWE and Attack Vector which are taken as input parameters for the Machine Learning Model. Since both of these parameters have varied categories, we used Label Encoding for data transformation. Label Encoding is a technique that is used to convert categorical columns into numerical ones so that they can be fitted by machine learning models which only take numerical data. It is an important preprocessing step in a machine-learning project [43]. The mathematical formula for label encoding a single categorical feature can be described as follows:

Given:

* A categorical feature *X* with *n* unique categories:

*{c1, c2, …, cn}*

* A function *f* that maps each category to a unique integer

The label encoding function *f* can be defined as:

*f (c i ) = i*

where:

* *ci* is the *i*-th category in the feature *X*
* *i* is the corresponding integer label for the category *ci*

with *i* ∈ *{1, 2, …, n}*

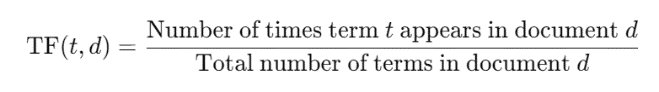
In practice, label encoding can be done using two different approaches: category codes approach and using sci-kit learn library approach.

### 2) Vectorization

Vectorization merely refers to the process of textual information being represented as numerical vectors. This is normally conducted after text cleaning. It might increase the speed of code execution and considerably reduce time used for training [44]. In the next section, a detailed explanation will be provided on the best ways to conduct vectorization. There exist mainly three major approaches to implement this on textual data: CountVectorizer, TF-IDF, and Word2Vec. We used TF-IDF vectorization for the input parameter short description of the Machine Learning Model. This parameter gives a brief overview of the nature, type, cause and impact of the vulnerability.

TF-IDF is the shortened form of Term Frequency–Inverse Document Frequency, a statistical measure that expresses the importance of a word in a document. This measure combines the two simplest yet important components of term frequency and inverse document frequency to arrive at a relevance score. Term Frequency describes the frequency of a word in a document [44]. The calculation involves the number of times the word occurs divided by the total amount of words within that same document. Inverse Document Frequency, on the other hand, is a measure of the amount of information the word conveys [45]. Most common words, such as "the," appear very frequently but do little when trying to get context or meaning from a sentence. For instance, such may be computed by applying an inverse logarithm to the document frequency, which will yield a proportion of documents that host a word [44].

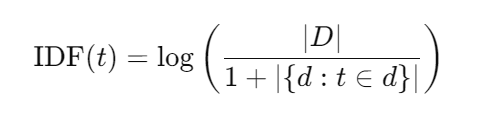
The mathematical formula for Term Frequency (TF):



Where:

* *t* is a term (word).
* *d* is a document.

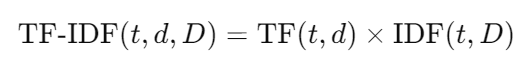
The mathematical formula for Inverse Document Frequency (IDF):



Where:

* ∣ *D* ∣ is the total number of documents in the corpus *D*.
* ∣ *{d : t ∈ d}*∣ is the number of documents where the term *t* appears.
* The 1*+* in the denominator is added to prevent division by zero if the term is not present in any document.

Once you have both TF and IDF, TF-IDF for a term *t* in a document *d* is computed as:



The increase in TF-IDF, which comes from the number of times a word appears in a document, is offset by the frequency of the word in the corpus. This means that greater weights are provided to words more unique for a given document. These are those terms whose use is not too diluted across many different documents.

## D. Implementation of the SVC algorithm in context to supply chain vulnerability prediction and mitigation

Algorithm: Train SVC Model for Vulnerability Classification

1. Load and Clean Dataset

   - Load dataset

   - Clean data to remove noise

2. Encode Features

   - Encode 'CWE\_category' using LabelEncoder

   - Encode 'Attack\_vector' using LabelEncoder

   - Encode 'Short\_Description' using TF-IDF Vectorizer

3. Split Dataset

   - Split dataset into training set (80%) and testing set (20%)

   - Define features X: 'CWE\_encoded', 'Vector\_encoded', 'Description\_tfidf'

   - Define target y: 'Severity'

4. Train Model

   - Initialize SVC model with linear kernel and OvR strategy

   - Train model on training set X\_train, y\_train

5. Evaluate Model

   - Calculate accuracy on testing set X\_test, y\_test

   - Print accuracy

6. Provide Mitigation Strategies

   - For each input CWE category:

   - Retrieve CWE link

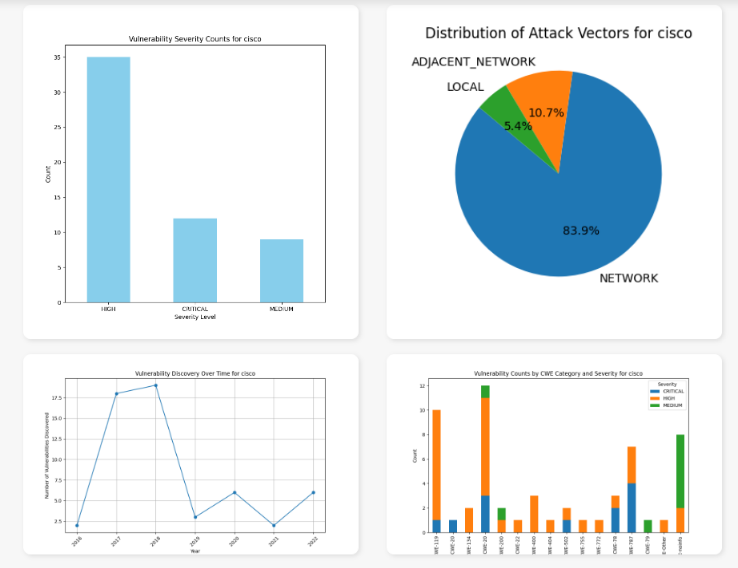
   - Fetch and display mitigation strategies from CWE website

End Algorithm

# VI. Results and discussion

The Vulnerability Dashboard provides insight into deep current vulnerabilities faced by a specific third-party vendor. That is very useful and powerful information in many ways for the given vendor, the client of that vendor, and any Cloud Service Providers to be careful with forming appropriate security measures vis-à-vis the threatening factors related to vulnerability. Identification and understanding of such vulnerabilities would accordingly help stakeholders make proper decisions for the protection of their systems within interdependent computing environments. Detailed dashboard reports enable proactive security defence strategies to ensure the management of any impending security threats before malicious entities have a chance to exploit them.

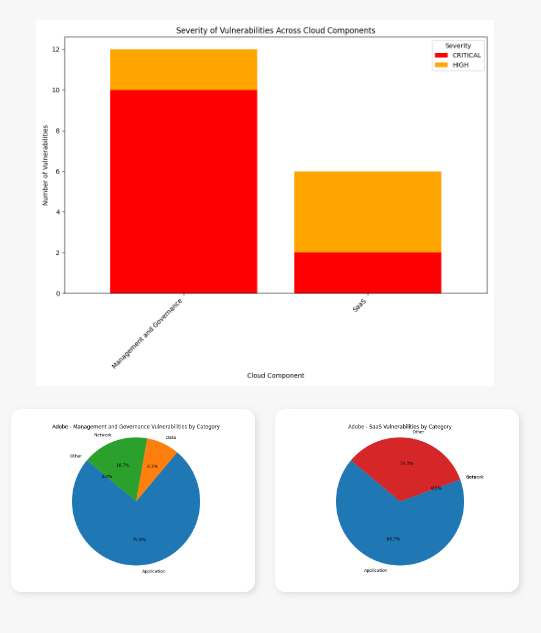
Consider the vulnerability analysis of cisco-



The vulnerability-based graphs provide us with a lot of vital information.

1. The severity and attack vector graphs indicate that the majority of attacks propagate through networks rather than being localized.
2. Most vulnerabilities are associated with either high or critical severity.
3. The time-based graphs clearly depict a peak in the number of attacks in 2019, followed by a declining trend in subsequent years.
4. The CWE classifier reveals that the majority of attacks fall under the CWE-20 category.

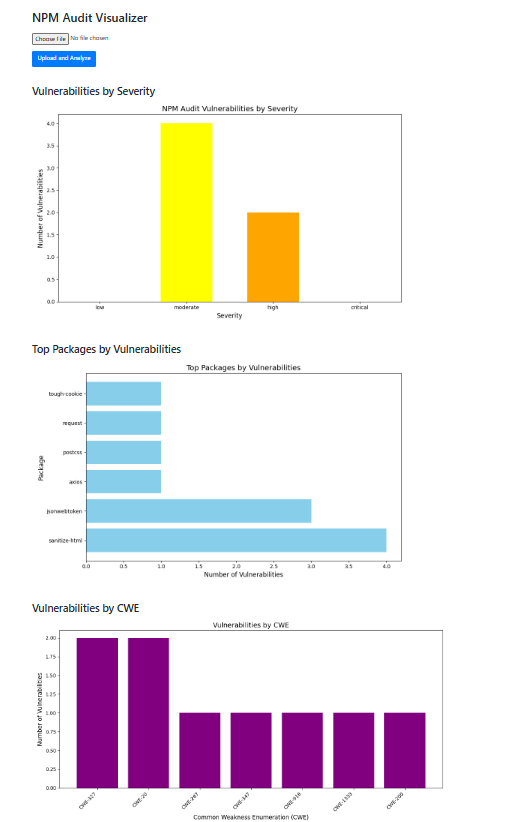
The cloud-based graphs classify which cloud component the attack is likely to effects based on the description and attack vector. The following are the graphs for adobe-



For adobe most of the critical attacks affect the management and governance sector of the cloud infrastructure and within it specifically the application security domain.

Cloud application security is the process of securing cloud-based software applications throughout the development lifecycle. It includes application-level policies, tools, technologies and rules to maintain visibility into all cloud-based assets, protect cloud-based applications from cyberattacks and limit access only to authorized users.

The npm audit visualization tool is a key tool in the analysis and representation of a project's vaguely structured and extended data on vulnerabilities. It uses visualization to transform these complex data sets into easily interpretable formats, such as charts and graphs, that contribute to the identification and assessment of security issues. It improves the efficiency of the mitigation process by providing a developer with a clear and concise depiction of vulnerabilities. Developers are allowed to identify critical vulnerabilities quickly and thus can set a priority of actions concerning mitigation measures, which helps in improving the security posture of the project.



The audit visualiser indicates the following:

1. Maximum vulnerabilities are found in the sanitize.html package.
2. In the given package.json file, out of 6 vulnerabilities aggregated in the 6 dependencies (within each package there can be multiple vulnerabilities, which are aggregated in terms of severity as a single count)-4 are moderate and 2 are high.
3. The most frequent CWE categories are placed in their descending order. In this case, the CWE-20 and 337 are the most common vulnerability type in the given project file with a count of 2 each.

The Severity Predictor is one of the better tools for analyzing vulnerability data in order to predict the potential severity of an attack. In that regard, the system uses sophisticated algorithms and machine learning techniques to analyze factors such as the nature of the vulnerability, its system impact, and the likelihood of its exploitation. Following up on these are mitigation strategies that are fine-tuned to the predicted severity of the attack. Such a targeted approach would ensure the implementation of the most effective security counters in line with the severity level anticipated, hence reducing the likelihood of successful attacks and reducing potential harm to the system.

# VII. Conclusion and Future Works

The general implementation can be broadly classified into three domains, which are independent but together cover a broad range of issues emerging under the domain of supply chain attacks.

**Vulnerability Visualization:** This module trends the vulnerabilities of the previous year that are related to the third-party vendor components. It comes with an elaborate dashboard, providing detailed textual descriptions of the associated vulnerabilities that the different vendors face, the level of threat they pose, which dimensions in the cloud they are in, and more. So, in identifying all trends, some patterns may be caught which in turn could be helpful for precluding future vulnerabilities.

**Severity Predictor:** This domain has an SVC model predicting the severity of newly identified vulnerabilities, accompanied by mitigation and detection strategy documentation coherent for each type of vulnerability. The ML model increases the speed and accuracy of finding critical vulnerabilities, thus enabling agile, faster, and more effective response measures.

**NPM Audit Visualizer:** This is a tool for visualizing vulnerabilities found during npm audits, presenting them in the form of graphs and charts. This graphical presentation of data makes large and complex data comprehensible, expanding the knowledge reach of such vulnerabilities to most firms.

These visualizations make planning, detection, and further analysis much easier to carry out and build upon.

This cohesive implementation is a proper warning system for CSPs and their third-party vendors within their domains in advance, so that they can perform mitigation procedures to alleviate the effects and protect the computing systems from possible threats.

It would associate more effective strategies if tied to a **real- time detector of such weak links** by understanding the trends of vulnerabilities through our research in various packages and third-party vendors. This detector would focus on common attack entry points and more vulnerable dependencies, providing a more effective mitigation approach. This proactive stance can significantly reduce the adverse effects of supply chain attacks by preventing them from reaching their full potential.

These attacks, however, are up and coming in recent history and pretty unexplored in the domain of vulnerabilities; hence, the future is indeed expected to bring new variable forms of attacks. Remediation and analysis on new forms of attacks are extremely critical with learning and updating strategies. This way, we will stay ahead of these evolving threats and be prepared with defense mechanisms that are strong and adaptable to the dynamic nature of supply chain vulnerabilities. Thus, this multi-pronged approach serves to strengthen current vulnerabilities detection and mitigation and provides a path forward to improve the state of research and development on protection against attacks on the supply chain. An approach such as this one is vital for the security and soundness of supply chains in this increasingly networked digital environment.

# References

[1] Md J. H. Faruk, M. Tasnim, H. Shahriar, M. Valero, A. Rahman, and F. Wu, "Investigating Novel Approaches to Defend Software Supply Chain Attacks," in *2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2022, pp. 283-283.

[2] "What is Software Vulnerability?", JFrog. [Online]. Available:<https://jfrog.com/devops-tools/article/software-vulnerability/>. [Accessed: Jul. 10, 2024].

[3] "PMC7338168," NCBI. [Online]. Available:<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7338168/>. [Accessed: Jul. 10, 2024].

[4]"Cloud Computing Supply Chain Attacks and the Open Source Effect," Voice&Data. [Online]. Available:<https://www.voicendata.com/cloud-computing-supply-chain-attacks-the-open-source-effect/>. [Accessed: Jul. 10, 2024].

[5] "Supply Chain Breaches Increased from 2022 to 2023," Supply Chain Brain. [Online].  Available:<https://www.supplychainbrain.com/articles/38672-supply-chain-breaches-increased-from-2022-to-2023>. [Accessed: Jul. 10, 2024].

[6] Snyk, "Software Supply Chain Report 2023," Snyk, 2023. [Online]. Available:  <https://go.snyk.io/2023-supply-chain-attacks-report.html>

[7] BlueVoyant, “The State of Supply Chain Defense: Annual Global Insights Report 2023”, 2023. [Online]. Available: <https://www2.bluevoyant.com/TheStateofSupplyChainDefense2023Report>

[8] "Most Supply Chain Attacks Target Suppliers’ Code," Trend Micro. [Online]. Available:<https://www.trendmicro.com/en_in/research/21/h/most-supply-chain-attacks-target-suppliers-code-enisa.html>. [Accessed: Jul. 10, 2024].

[9] "A Kill Chain Analysis of the 2013 Target Data Breach," Covert.io. [Online]. Available:<https://www.covert.io/research-papers/security/A%20Kill%20Chain%20Analysis%20of%20the%202013%20Target%20Data%20Breach.pdf>. [Accessed: Jul. 10, 2024].

[10] "Software Supply Chain Attacks: Everything You Need to Know," PortSwigger. [Online]. Available:<https://portswigger.net/daily-swig/software-supply-chain-attacks-everything-you-need-to-know>. [Accessed: Jul. 10, 2024].

[11] "The Impact of Supply Chain Attacks on Cloud Services," Cloud Tech Insight. [Online]. Available:<https://cloudtechinsight.com/the-impact-of-supply-chain-attacks-on-cloud-services/#:~:text=In%20conclusion%2C%20the%20potential%20impact%20of%20a%20supply,reputation%2C%20regulatory%20compliance%20issues%2C%20and%20increased%20cybersecurity%20risks>. [Accessed: Jul. 10, 2024].

 [12] "Assessing Cloud Risk: The Supply Chain Perspective," BCS. [Online]. Available:<https://www.bcs.org/articles-opinion-and-research/assessing-cloud-risk-the-supply-chain-perspective/>. [Accessed: Jul. 10, 2024].

[13] "Security Magazine - Companies Give Third Parties Access to All Cloud Data," Security Magazine. [Online]. Available:<https://www.securitymagazine.com/articles/94435-of-companies-give-third-parties-access-to-all-cloud-data#:~:text=The%20Wiz%20Research%20team%20conducted%20extensive%20research%20of,security%20risk%2C%20as%20well%20as%20serious%20privacy%20risk>. [Accessed: Jul. 10, 2024].

[14] "6 Tips for Understanding 3rd Party Risk in the Cloud," Cloud Security Alliance. [Online]. Available:<https://cloudsecurityalliance.org/blog/2023/01/11/6-tips-for-understanding-3rd-party-risk-in-the-cloud>. [Accessed: Jul. 10, 2024].

[15] B. Gokkaya, L. Aniello, and B. Halak, \*Software Supply Chain: Review of Attacks, Risk Assessment Strategies, and Security Controls\*. 2023, pp. 1-27.

[16] A. Kruti, U. Butt, and R. Bin Sulaiman, \*A Review of SolarWinds Attack on Orion Platform Using Persistent Threat Agents and Techniques for Gaining Unauthorized Access\*. 2023, pp. 1-6.

[17] "Known Exploited Vulnerabilities," Panorays. [Online]. Available:<https://panorays.com/blog/known-exploited-vulnerabilities/>. [Accessed: Jul. 10, 2024].

[18] "Known Exploited Vulnerabilities Catalog," CISA. [Online]. Available:<https://www.cisa.gov/known-exploited-vulnerabilities-catalog>. [Accessed: Jul. 10, 2024].

[19] "Common Vulnerabilities and Exposures," CVE. [Online]. Available:<https://www.cve.org/>. [Accessed: Jul. 10, 2024].

[20] "Link Springer Article," Springer. [Online]. Available:<https://link.springer.com/article/10.1007/s10207-023-00751-6>. [Accessed: Jul. 10, 2024].

[21] "How to Prioritize Vulnerabilities," Paladin Cloud. [Online]. Available:<https://paladincloud.io/unified-vulnerability-management/how-to-prioritize-vulnerabilities/>. [Accessed: Jul. 10, 2024].

[22] "Infrastructure as a Service," Dot Net Tutorials. [Online]. Available:<https://dotnettutorials.net/lesson/infrastructure-as-a-service/>. [Accessed: Jul. 10, 2024].

[23] "Design Strategy for Securing IaaS, PaaS, and SaaS," Whizlabs. [Online]. Available:<https://www.whizlabs.com/blog/design-strategy-securing-iaas-paas-and-saas/>. [Accessed: Jul. 10, 2024].

[24] "SaaS, PaaS, and IaaS: What are All the Risks?," Arrow. [Online]. Available:<https://www.arrow.com/globalecs/na/arrow-channel-advisor/saas-paas-and-iaas-what-are-all-the-risks/>. [Accessed: Jul. 10, 2024].

[25] "PaaS Security Threats, Solutions, and Best Practices," Spot.io. [Online]. Available:<https://spot.io/resources/cloud-security/paas-security-threats-solutions-and-best-practices/>. [Accessed: Jul. 10, 2024].

[26] "Cloud Security Architecture for IaaS, PaaS, and SaaS," BlueXP. [Online]. Available:<https://bluexp.netapp.com/blog/blg-cloud-security-architecture-for-iaas-paas-and-saas>. [Accessed: Jul. 10, 2024].

[27] "Software Supply Chain Security Tools," Snyk. [Online]. Available:<https://snyk.io/series/software-supply-chain-security/supply-chain-security-tools/>. [Accessed: Jul. 10, 2024].

[28] "An Open Guide to Evaluating Software Composition Analysis Tools," Linux Foundation Japan. [Online]. Available:<https://www.linuxfoundation.jp/wp-content/uploads/2020/11/An-Open-Guide-To-Evaluating-Software-Composition-Analysis-Tools_V2.pdf>. [Accessed: Jul. 10, 2024].

[29] NPM Security: Preventing Supply Chain Attacks," Snyk. [Online]. Available:<https://snyk.io/blog/npm-security-preventing-supply-chain-attacks/>. [Accessed: Jul. 10, 2024].

[30] "Dozens of Malicious Data Harvesting NPM Packages Found," Duo. [Online]. Available:<https://duo.com/decipher/dozens-of-malicious-data-harvesting-npm-packages-found#:~:text=Researchers%20have%20uncovered%20a%20software%20supply-chain%20attack%20involving,were%20embedded%20in%20various%20mobile%20applications%20and%20websites>. [Accessed: Jul. 10, 2024].

[31] "Data Study Reveals Predictors of Supply Chain Attacks in NPM Repositories," PortSwigger. [Online]. Available:<https://portswigger.net/daily-swig/data-study-reveals-predictors-of-supply-chain-attacks-in-npm-repositories>. [Accessed: Jul. 10, 2024].

[32] "OSV Scanner vs. npm Audit: A Detailed Comparison of SCA Tools," Jit.io. [Online]. Available: <https://www.jit.io/resources/appsec-tools/osv-scanner-vs-npm-audit-a-detailed-comparison-of-sca-tools>. [Accessed: Jul. 10, 2024].

[33] "The Basics of Package.json," NodeSource. [Online]. Available: <https://nodesource.com/blog/the-basics-of-package-json>. [Accessed: Jul. 10, 2024].

[34] "Data Preprocessing in Machine Learning," Simplilearn. [Online]. Available: <https://www.simplilearn.com/data-preprocessing-in-machine-learning-article>.

[Accessed: Jul. 10, 2024].

[35] "Data Preprocessing Guide," Neptune.ai. [Online]. Available: <https://neptune.ai/blog/data-preprocessing-guide> [Accessed: Jul. 10, 2024].

[36] "Data Preprocessing in Machine Learning using Python," GeeksforGeeks. [Online]. Available: <https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/>

[Accessed: Jul. 10, 2024].

[37] "Data Preprocessing Guide," V7 Labs. [Online]. Available: <https://www.v7labs.com/blog/data-preprocessing-guide> [Accessed: Jul. 10, 2024].

[38] "Data Integration in Data Mining," GeeksforGeeks. [Online]. Available: <https://www.geeksforgeeks.org/data-integration-in-data-mining/> [Accessed: Jul. 10, 2024].

[39] "Data Transformation in Machine Learning," GeeksforGeeks. [Online]. Available: <https://www.geeksforgeeks.org/data-transformation-in-machine-learning/> [Accessed: Jul. 10, 2024].

[40] "Support Vector Machine Algorithm," JavaTpoint. [Online]. Available: <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>.

[Accessed: Jul. 10, 2024].

[41] "Understanding the Mathematics Behind Support Vector Machines," Fritz.ai. [Online]. Available: <https://fritz.ai/understanding-the-mathematics-behind-support-vector-machines/> [Accessed: Jul. 10, 2024].

[42] "Encoding Techniques for Machine Learning in Alteryx: A Concise Guide," Alteryx Community. [Online]. Available: <https://community.alteryx.com/t5/Data-Science/Encoding-Techniques-for-Machine-Learning-in-Alteryx-A-Concise/ba-p/1203894> [Accessed: Jul. 10, 2024].

[43] "Categorical Data," DataCamp. [Online]. Available: <https://www.datacamp.com/tutorial/categorical-data>

 [Accessed: Jul. 10, 2024].

[44] "Vectorization in Machine Learning," Heartbeat by Comet. [Online]. Available: <https://heartbeat.comet.ml/vectorization-in-machine-learning-2e3bdce7dbe> [Accessed: Jul. 10, 2024].

[45] "Vectorization Techniques in NLP: A Guide," Neptune.ai. [Online]. Available: <https://neptune.ai/blog/vectorization-techniques-in-nlp-guide>

[Accessed: Jul. 10, 2024].

[46] "TF-IDF Vectorizer Scikit-Learn," Medium. [Online]. Available: <https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>

[Accessed: Jul. 10, 2024].

1. This paragraph of the first footnote will contain the date on which you submitted your paper for review, which is populated by IEEE. It is IEEE style to display support information, including sponsor and financial support acknowledgment, here and not in an acknowledgment section at the end of the article. For example, “This work was supported in part by the U.S. Department of Commerce under Grant 123456.” The name of the corresponding author appears after the financial information, e.g. *(Corresponding author: Second B. Author).* Here you may also indicate if authors contributed equally or if there are co-first authors.

   The next few paragraphs should contain the authors’ current affiliations, including current address and e-mail. For example, First A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

   Second B. Author Jr. was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

   Third C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba 305-0047, Japan (e-mail: author@nrim.go.jp).

   Mentions of supplemental materials and animal/human rights statements can be included here.

   Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org [↑](#footnote-ref-1)