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

The 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 endeavors in recent times [1].

A software vulnerability is a defect in software that could allow an attacker to gain control of a system. 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 [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. 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. 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, 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 [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 [8].

The well-documented 2013 breach targeting the prominent US retail entity Target serves as a prominent illustration of lateral movement [9]. SolarWinds, a SaaS provider, 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 [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 [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 [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 [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].

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 remainder of this paper is organized as follows.

Section 2 introduces related work. Section 3 describes

the vulnerability dashboard. An added functionality in the npm audit tool, is proposed in Section 4. Section 5 proposes a ML model for prediction of attack severity. Section 6 summarizes the results and concludes the

paper along with future scope.

# 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. 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. This study the pressing need for future research efforts to focus on the development of more comprehensive frameworks of risk assessment 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 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 [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.

## 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].

**Vendor Project:** A third-party entity linked to the vulnerability, typically an external individual or organization offering a service or technology under a contract.

**Product:** Products provided by the vendor which are related to the vulnerability, they allow understanding security weak points of various products.

**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, the software affected, and the results of the attack that could happen.

**Publication Date:** The public release date of a vulnerability is important to realize exactly when a risk was open to the world.

**CVSS:** Common Vulnerability Scoring System represents an openly available framework that scores and enumerates attributes of security vulnerabilities in information systems.

**CWE:** CWE is Common Weakness Enumeration for the weakness and vulnerabilities classified in hardware and software literature.

**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.

**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.

**Cloud Component:** Specify which cloud component is affected by the vulnerability so that mitigation strategies are put in place for that component only.

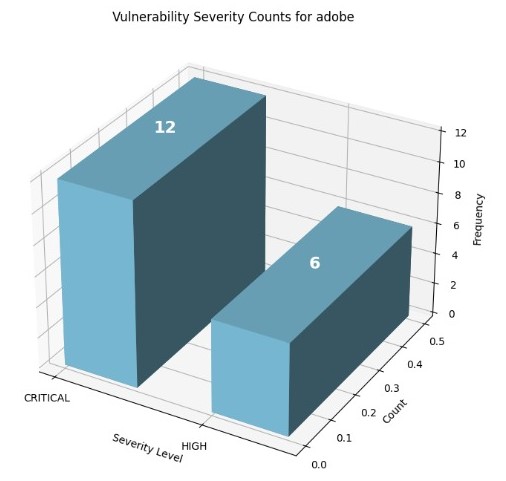
**Category of vulnerability:** This involves the class or grouping to which vulnerability belongs.

## B. Data Visualization

Data visualization is essential for interpreting and sharing cybersecurity data. 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.



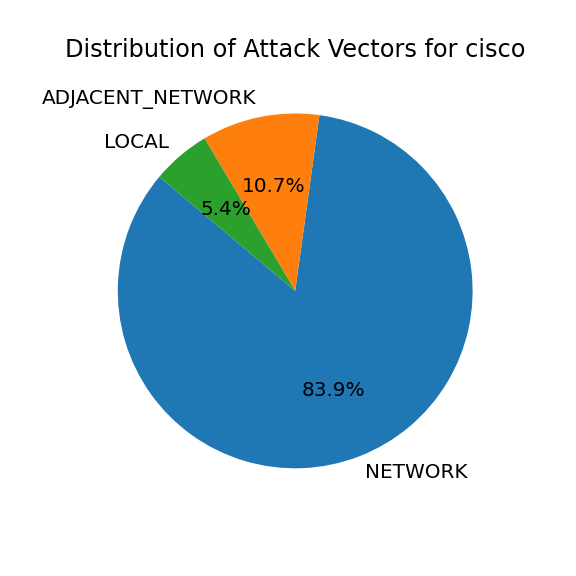
**Fig. 1.** Graph for Vulnerability Severity Counts

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.



**Fig. 2.** Pie Chart for Distribution of Attack Vectors

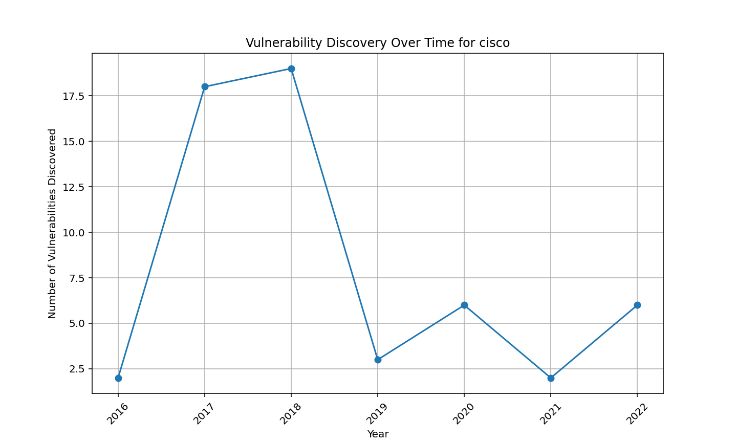
In that network vulnerabilities are remotely exploitable, not requiring physical access to the specific device, the potential for exposure from every point on the globe to possible actors represents a serious threat. This need emerges considering that network vulnerabilities are always present.

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.

### 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.



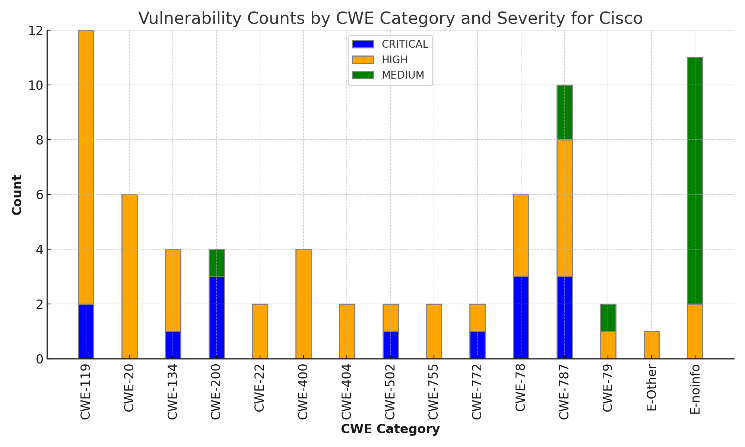
**Fig. 3.** Trend Analysis of Vulnerabilities over Time

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.

### 4) Vulnerability Counts by CWE and Severity

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

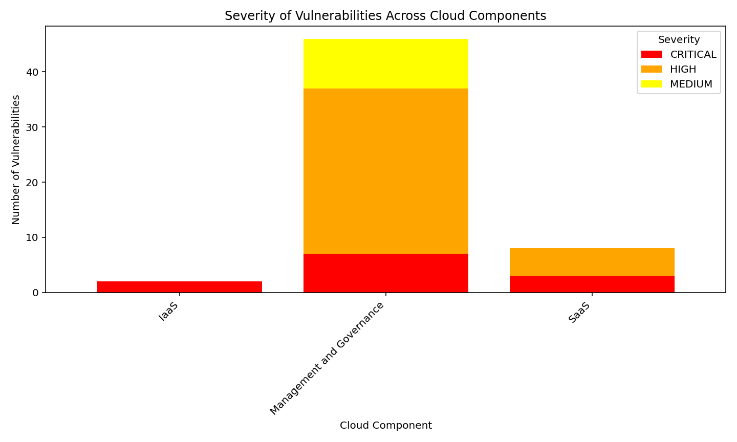


**Fig.4.** Graph for Vulnerability Counts by CWE Category

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

A bar chart showing the vulnerability distribution by severity ranking across the different components of cloud systems can be drawn as shown.



**Fig. 5.** Graph for Severity of Vulnerabilities across Cloud Components

The sheer complexity of securing IT environments has risen already because too many organizations have embraced cloud solutions [20]. Prioritization by severity can be accomplished using visualization tools. 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].

**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 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, 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**

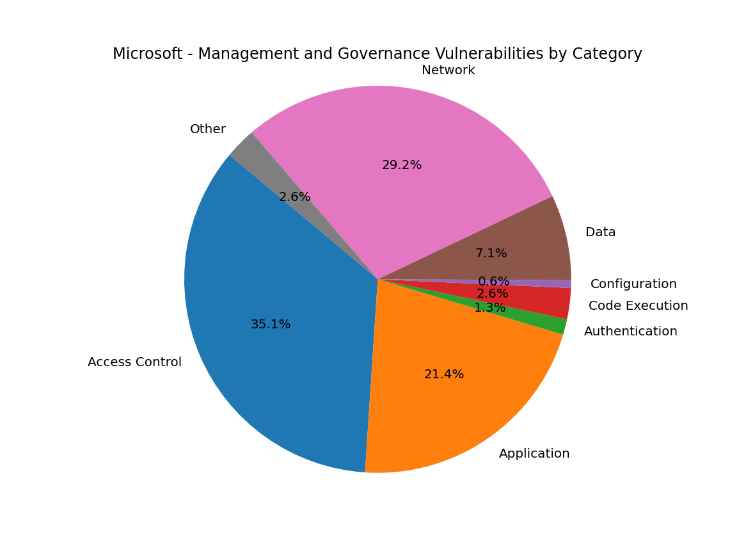
This layer deals with orchestration and management across cloud services, security policies, and compliance. In this case, the vulnerabilities to management and governance may source from the following areas: not transparent across cloud environments, misalignment of business and IT teams, and poor security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



**Fig. 6.** Pie chart for vulnerabilities by category

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.

*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.

*Management and governance:* Loopholes in the network of management interfaces can be exploited to obtain unauthorized access and control over cloud resources.

**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.

*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.

*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.

*Management and Governance:* Unauthorized changes in cloud configurations would disrupt the service management due to the vulnerability of code execution in the management interfaces.

**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.

*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.

*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.

*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.

**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.

*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.

*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.

*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.

**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.

*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.

*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.

*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.

**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.

*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.

*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.

*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.

**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.

*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.

*SaaS:* May introduce access control vulnerabilities in SaaS to sensitive user data and application functionalities, hence causing data breaches or user privacy compromises

*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.

**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.

*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.

*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.

*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.

**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.

*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.

*SaaS:* Authentication vulnerabilities can be exploited to gain unauthorized access to applications and user data in Software as a Service, thus compromising the security of applications and users' privacy.

*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.

**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.

*PaaS:* DoS vulnerabilities can be triggered with malicious traffic against platform services, in turn affecting the availability and performance of applications on top.

*SaaS:* DoS vulnerabilities can make the application unavailable to users by simply flooding it with traffic.

*Management and Governance:* DoS vulnerabilities in the management interfaces will hamper cloud management services and may also impact the administrators from monitoring the environment.

**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.

*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.

*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.

# 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 [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.

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 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.

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.

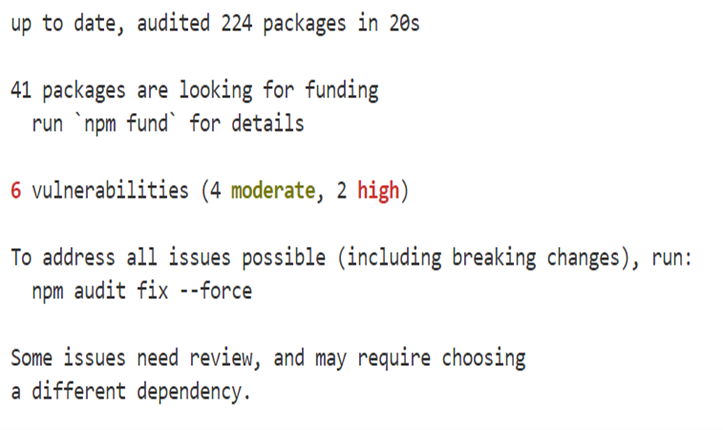
However, the visual analysis of the NPM audit data in detecting underlying patterns, enhancing understandability among a larger client base, etc., are some features that this tool has not yet implemented under its umbrella of functionalities.

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

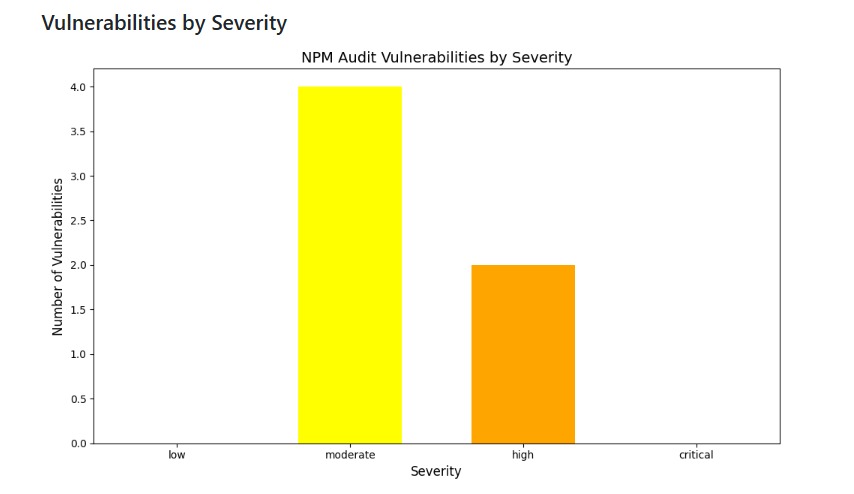
The base of this visualization is NPM, a package manager that scans the package.json file in real-time. The uploading of the package.json file connected with the project will trigger this NPM audit feature, and such a process will generate an audit dataset. By further parsing this dataset, key-value 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.



**Fig. 7.** Command for Prompt

### 1) Vulnerability Severity Analysis

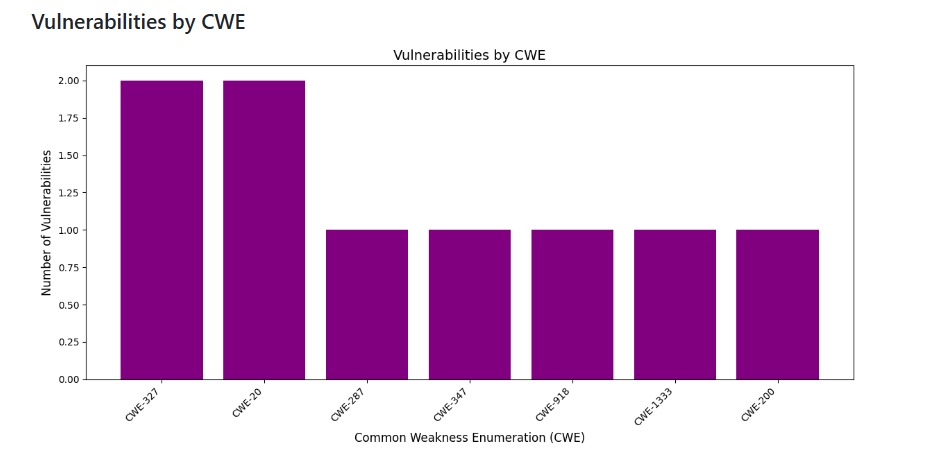


**Fig. 8.** Graph for Classification of Vulnerabilities by Severity

*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.

### 3) Classification According to CWE

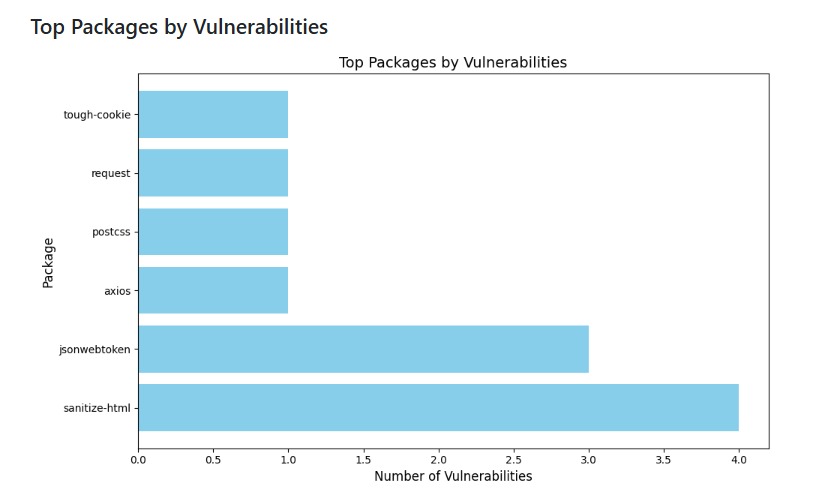


**Fig. 10.** Graph for classification according to CWE

*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.

### 2) Package Vulnerability Analysis

**Fig. 9.** Graph for Top Packages by Vulnerabilities

*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.

### 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]. These are the very reasons why data preprocessing needs to be carefully prepared for analysis.

Data cleaning is an intrinsic part of data science that ensures the information one is dealing with is correct and reliable. 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].

### 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].

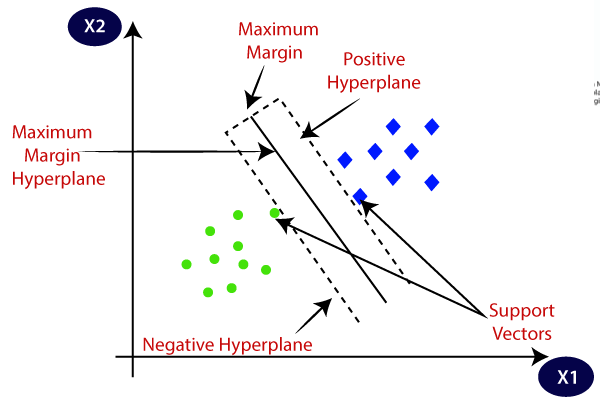
### 3) Data Transformation

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. 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. Support Vector Classifiers, or SVCs 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. 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:



**Fig. 11.** SVM Graph

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.

Classes in the model: [‘CRITICAL’, ‘HIGH’, ‘MEDIUM’]

The hypothesis equation is defined as:

where:

* is the weight vector normal to the hyperplane, which decodes the orientation
* is the feature vector of the point
* is the bias term of the hyperplane

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].

**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 and Minimizing Misclassification.

The mathematical expression to minimize is:

* is the term that encourages a larger margin (a simpler and more generalizable model).
* 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].

### 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]. 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. 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].

The mathematical formula for Term Frequency (TF):

where *t* is a term (word) and *d* is a document

The mathematical formula for Inverse Document Frequency:

where:

* is the total number of documents in the corpus *D*.
* is the number of documents where the term *t* appears.
* The in the denominator is added to prevent division by zero if the term is not present in any document.

TF-IDF for a term in a document is computed as:

## 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:

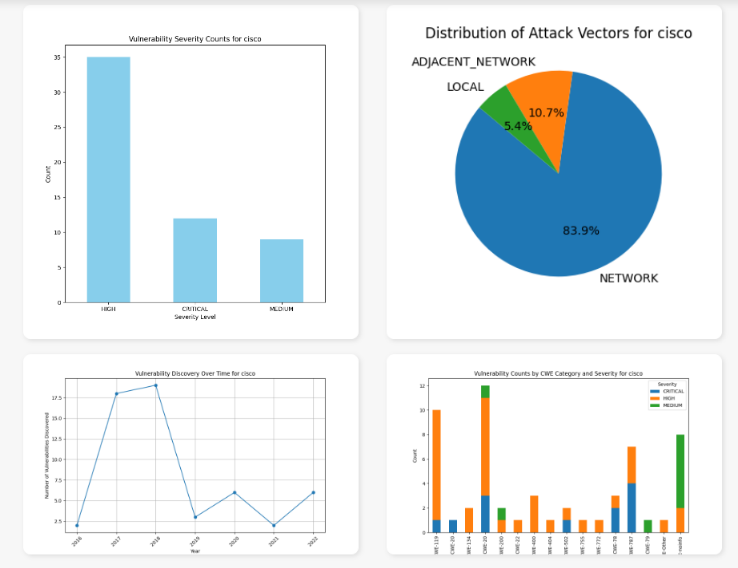
   - Retrieve CWE link

   - 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-

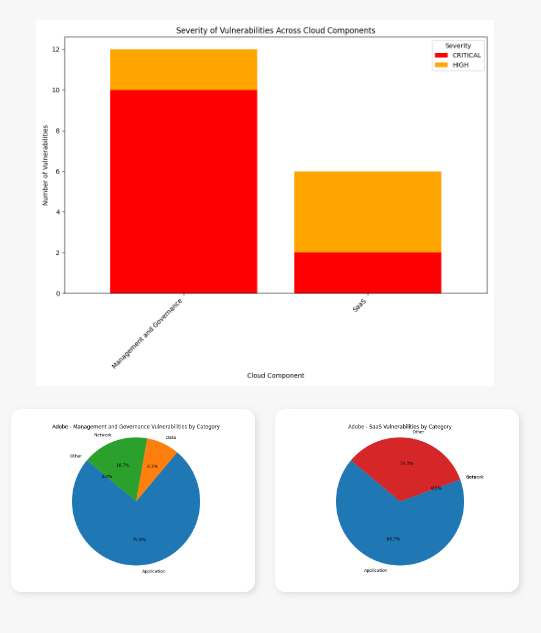


**Fig. 11.** Dashboard for Vulnerabilities

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

* The severity and attack vector graphs indicate that the majority of attacks propagate through networks rather than being localized.
* Most vulnerabilities are associated with either high or critical severity.
* The time-based graphs clearly depict a peak in the number of attacks in 2019, followed by a declining trend in subsequent years.
* 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-



**Fig. 12.** Dashboard for Cloud Components

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.

The Severity Predictor is one of the better tools for analyzing vulnerability data in order to predict the potential severity of an 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 system.

# VII. Future Works

The 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.

# 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. [↑](#footnote-ref-1)