Zero Trust is Not Enough: Mitigating Data Repository Breaches

J.S. Hurley
ODNI, Bethesda, USA

john.s.hurley@odni.gov

Abstract: Successful mission operations depend on the ability of an organization to collect, manage, analyze, and secure its data. Traditional network frameworks have become less appealing because they rely on a "trust but verify" paradigm that does not stand up well against the advanced tools and techniques of modern cyber attackers. The Zero Trust Framework has emerged as a logical replacement but unfortunately does not adequately address the trustworthiness of data in data repositories. This broader view is important because data repositories have become arguably, the most prominent means for data sharing across many sectors around the globe. Unfortunately, data repositories are also undergoing widespread malware attacks and, in some cases, data critical to national security can be impacted. In this study, we propose a potential framework that relies more on data lineage, end-to-end metadata, and the use of machine learning tools and techniques to reduce and possibly mitigate the problems with data repositories that Zero Trust Frameworks fail to address.

Key Terms: Zero Trust, Data Repository, Frameworks, Machine Learning.

1. Introduction

Senior leaders must make decisions that can immensely impact operations and Mission success. As might be expected, decisions and how they are implemented can have significant implications to Mission success. Hence, the accuracy, accessibility, and availability of the data are crucial in the decision-making process. Data scientists facilitate an improved and informed decision-making process through the unique additional insight they provide. A data repository is an important element of a data scientist's toolbox wherein select datasets can be placed at an acknowledged centralized, isolated location, made available to users to validate existing and also generate new results. Data repositories have the additional benefit of serving as the primary mechanism in which data scientists facilitate collaboration and information sharing between users.

Application stacks are a dominant way that organizations carry out many of their key business functions. Third-party apps have become a staple of how we interact with data and services in both cloud and mobile-driven environments. There is general consensus that third party apps have immense benefits. However, forethought must also be given to the potential risks of third-party apps (Wang, et al., 2020).

Data repositories serve as the central point in which large amounts of computer code are stored and managed. As a result, security of the repository is a primary requirement and concern. However, herein lies a major inherent conflict. Although security is a necessary requirement, too many restrictions can be counterproductive because the key to collaborative and open-source development depends on the accessibility to data repositories. Recent data repository breaches seem to be signaling a growing and troubling trend. According to Sonatype, the volume of malicious attacks on open-source code repositories over the last three years has hit triple digit growth (Scott & Spitse, 2022)! It is reported that an almost 700% rise in attacks designed to plant malware in software components was detected. Almost every business relies in some way or another on access to open-source data. Compromised or lost credentials or breach of the underlying service could allow attackers to modify a code base without permission. Attacks can be subtle or disguised, so it is important to review any commits, changes or pull requests to understand security implications. The manner in which credentials are injected, managed, and stored into the computing environment is also critical. For example, the Uber hacks were all done by stealing server passwords that Uber had accidentally posted on GitHub repositories (twice!) (Fair, 2018).

Most of the recently recorded attacks on data repositories have largely been accomplished without exposing the passwords of users. Attackers have gained access through "access" tokens that authorize the sharing of specific user account information (Curwin, et al., 2023). The latest attacks on data repositories seem to have a common thread, i.e., the attacks seem to be coming primarily through third party access provided by OAuth, the open standard for token-based authentication and authorization. Through OAuth request links, recipients can be tricked into illicit grants, e.g., consent phishing emails that can enable access to attackers via API resources. In this case, targeted users unknowingly grant permissions that allow attackers to make API calls on their behalf through attacker-controlled apps. Unfortunately, access to cloud SaaS environments is obtained with relative ease due to end user-granted permissions occurring without much scrutiny. Permissions can be granted by end

users simply via a permissions request submitted from the third-party app. Similar problems can also occur through browser extensions via application performance interfaces (APIs).

It is important to note that new vulnerabilities can be introduced any time a change is made to a codebase. As a result, it is prudent to check for net vulnerabilities before merging a pull request. However, even this can be challenging in containerized applications where vulnerabilities can enter any container layer from the operating system to the application. Pre-existing container and dependency vulnerabilities in the target branch must be known in order to identify new vulnerabilities in a pull request. A comparison between the pull request and target branch images can provide a view into new vulnerabilities that might be introduced in containers and dependencies (Kaur, et al., 2021). Additionally, compromises of packages or libraries can lead to downstream infections. *Of particular concern is that containers and the microservices they provide have increased the surface area almost exponentially. Container image vulnerabilities, unauthorized access, and weak firewalls are a few of the challenges that containers face* (White, 2022).

Zero Trust is a security framework that continuously authorizes, authenticates, and validates users in order to be granted and maintain access to data and applications (Raina, 2022). Zero Trust assumes that networks can be local, cloud-based, or some combination, i.e., there is no traditional network edge. NIST Special Publication (SP) 800-207 is currently the de facto standard for Zero Trust Architectures for both industry and government agencies (Rose, et al., 2020). *Without question, Zero Trust provides significant benefits and performs admirably, in protecting against many cyberattacks, reducing attack surfaces, and simplifying infrastructure requirements. In addition, Zero Trust Architectures (ZTAs) are designed to better meet many new business requirements driven by digital transformation that can increase risk. This benefit of zero trust belies the fact that traditional perimeters are complex and can increase risk. Zero Trust, however, does have its challenges, e.g., Zero Trust must be slowly phased into infrastructure because it is much more restrictive and can require an entirely different business model than what might already be in place* (Chimakurthi, 2022).

In this study, the focus will be on the design of a potential framework that relies on the use of machine learning tools and techniques to reduce and possibly mitigate some of the problems with current and future breaches that can plague data repositories. First, the study begins with a discussion of data repositories and the value they have in the decision-making process. Next, the concept of Zero Trust and how it significantly improves a number of past issues with regards to security are considered. A series of recent breaches will then be discussed in detail while sharing suspected ways that attackers gained unauthorized entry into some data repositories. The Results and Discussion session will then follow with a proposed framework that utilizes various data science techniques to address the possibility of mitigating the breaches in data repositories. Specifically, the focus will be on how data science technologies can be used to determine if OAuth apps are either “risky” or “allowed”. The issues of commits and pull requests (PRs) are important but will be considered only briefly in this work. In those discussions, the context will be on how pre-existing container and dependency vulnerabilities in the target branch can be passed on through pull requests and commits. More extensive investigations will be the subject of future work.

2. Data Repositories

Data repositories represent a large database infrastructure in which datasets are collected, managed, and stored for analysis, reporting, and sharing. Data repositories provide immense value due to their ability to store, make accessible, organize, and archive datasets in a logical manner. From a scientific viewpoint, data repositories are critical because they enable data sharing, one of the primary mechanisms for advancing knowledge in a timely manner. Data repositories have added benefits, including serving as the centralized location that makes it possible to pull in data from multiple sources and improve the quality of preserved and aggregated data. Some of the most prominent repositories are placed on open-source end-to-end software development platforms such as GitHub, Bitbucket, and GitLab. As data grows, the database management system must be scalable to accommodate the inherent slowing down of the system. Users typically interact with code repositories through a web application or command-line utilities such as Git.

3. Zero Trust

Zero Trust is a strategic approach that eliminates implicit trust while continuously validating most stages of a digital interaction. Zero Trust is designed to enable digital transformation and protect modern environments by leveraging network segmentation; preventing lateral movement; providing Layer 7 threat prevention; simplifying granular “least access” policies; and using strong authentication methods (Souppaya, et al., 2022).

Network complexity, overlays and dynamic IPs in combination with the limitations of traditional firewalls can cause problems with traditional security approaches (Fearn, 2020). Zero Trust Architectures (*ZTAs), however, fall short in dealing with some issues. In particular, one of the biggest challenges of zero trust is that it can be very complex to implement, given that every application, device, and user must be authenticated and authorized. For organizations with large numbers of users, this requirement can be especially challenging because of the layer of complexity added. Next, a discussion of some of the recent data repository attacks and resulting challenges are considered.*

4. Attacks

Credentials used or misused in the network account for more than 80% of all cyber attacks. Attention to greater account integrity; adherence to organizational rules; avoidance of high-risk shadow IT services; password security; integrity of accounts; adherence to organizational rules; and avoidance of high-risk shadow IT services should all, at a minimum be actuated (Raina, 2022). On August 3rd, software developer Stephen Lacy discovered a data repository attack involving bad actor cloning and adding malicious code to more than 35,000 GitHub repositories, while still keeping the code's original source code. Almost 40% of the infected repositories originated from a single organization, referred to as a "redhat-operator-ecosystem" on the site. Also, the code could download additional malware from a third-party site that allowed further exploitation of any environment or application that was using the malicious cloned code originally introduced to the GitHub repositories. The weaponized code could potentially lead developers to accidentally download cloned code repositories which contained malicious code (Powell, 2022).

In 2022, the identification and authentication giant, Okta, reported that it was informed by GitHub of "suspicious" activity in its code repositories. Okta concluded that hackers gained unauthorized access to copy code repositories. The breach incidents reinforced how crucial logs are to be put in place and continuously monitored to detect if any of Okta's systems were inappropriately accessed and if any data had been exfiltrated (Page, 2022). Undiscovered flaws and vulnerabilities on open-source GitHub repositories are increasingly becoming a major concern. Delayed or inconsistent code commits along with improper scanning also create risk as repositories age. The identity and access management platform, Okta, reported on December 2022 that its source code repositories had been inappropriately accessed and copied. Similar breaches and unauthorized access to code bases has also been reported by password manager, LastPass, in which a backup of its customer vault data was copied, impacting over 33 M registered users (Kapko, 2023).

5. Results and Discussion

It is important to reiterate that two primary mechanisms that can contribute to breaches on data repositories are the subject of discussion in this study. Third-party access (via the authorization and authentication standard, OAuth) and when changes, commits, or pull requests are made to a codebase. It is a main premise of this paper that a new framework that incorporates the impact of both possibilities must be introduced and implemented to reduce and possibly mitigate the challenges of data repository breaches.

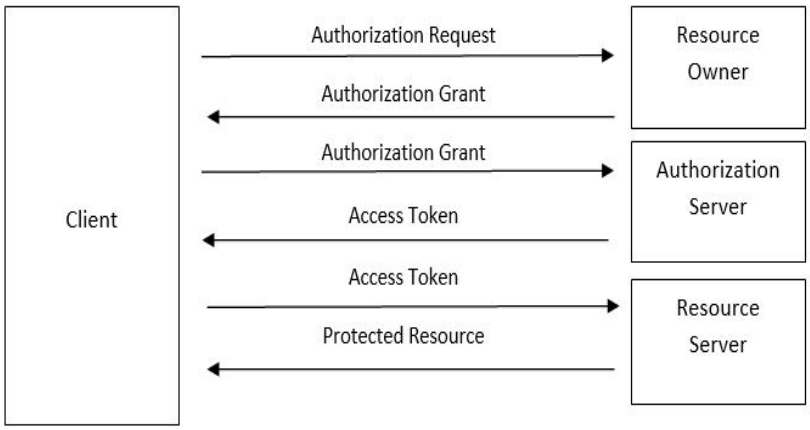

Figure 1: Access Token Issuance Process, (Shah, 2022)

In Figure 1., the process in which access tokens are generated and passed on to the third-party apps is shown. In this process, we must pay attention to possible malicious and "risky" third-party apps that could be introduced and how they might be detected and mitigated. It is important for users to review processes and procedures within their environments. Hence, filters should be set to Permission levels of high severity and uncommon community apps that are potentially very risky should not be allowed in the system. Under Permissions, all options should be selected that are particularly risky in a specific context. In particular, email access permissions should be carefully reviewed, especially when full access to all mailboxes is allowed (Curwin, et al., 2023). In Table 1., some key tests to determine if OAuth apps are risky or suspicious, or be allowed, or require further testing are shown below.

Table 1: OAuth Testing for Risk and Suspicious Apps

Testing (Phase 1)	Risky	Allowed
Email-related access users that have unnecessary permissions	x	
Apps authorized by external users that do not fit within organization's security standards and policies	x	
Apps authorized by a small number of users	x	
Apps with suspicious name, publisher, or URL	x	
Apps and Targeted apps that do not have current last authorization date	x	
Common or Often Used		x
Only Permissions linked to app's purpose allowed		x
Requires high privileges or administration consent	x	
Activities performed by the app are appropriate		x
Users consent to an app		x
If the OAuth 3rd party app satisfies the above allowed criteria, then they can be allowed. Otherwise, a second phase of testing should be implemented. In particular, an examination of the app's name, publisher, and URL in different app stores should be carried out with a focus on including:		
Testing (Phase 2)		
Low number of downloads	x	
Low rating or score or bad comments	x	
Suspicious publisher or URL	x	
Date of last update not recent	x	
Irrelevant permissions	x	
OAuth apps that fail to pass the second phase of testing should be disqualified and considered too risky		

If the app is determined to be risky, either manual or automatic remediation methods can be used to revoke the app or access privileges of the specific user of the app.

In the recently reported breaches, infections from third party apps were a primary contributor to the breaches. Immediate attention must be given because organizations are increasingly collecting, storing, and using data to gain critical insight to make the most effective business decisions. Data repositories have been viewed as a mechanism that could overcome many of the issues noted given their access and long-term storage capabilities that make them well-suited as a sustainable information infrastructure. In this study separate applications will be placed into clusters that can be categorized as "allowed" or "risky" before they are incorporated into the data repositories to reduce the likelihood of vulnerabilities.

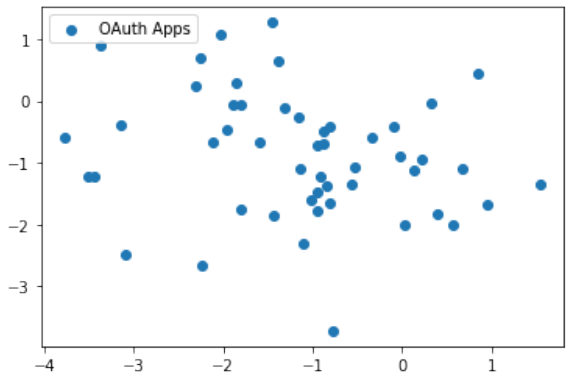

Figure 2: Sample Cluster 100 points of OAuth Apps vs Allowed Apps

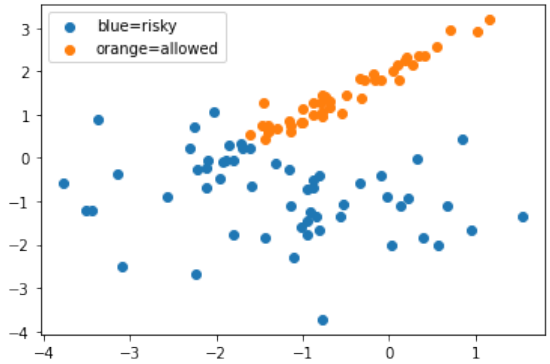

Figure 2. Sample Cluster 100 Points of OAuth Apps

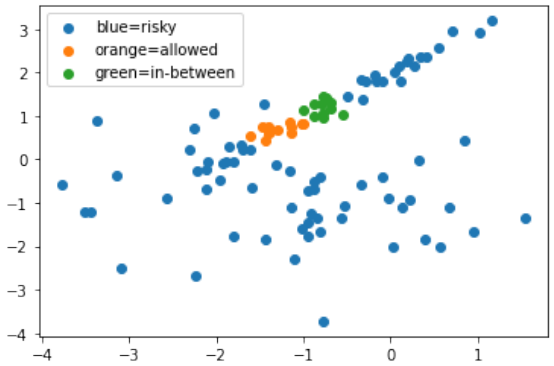

Figure 3. Agglomerative Clustering Applied for Risky vs Allowed Apps

Figure 4. DBScan Clustering Applied for Risky vs Allowed Apps

In Figure 2., there are 100 OAuth apps (points) randomly selected. In Figure 3., the points are categorized as either "risky" or "allowed" according to the criteria listed in Table 1. In Figure 3., the apps considered "risky" are colored blue (note: the blue points in Figure 1.are just arbitrary points not yet categorized) and those that are allowed are colored "orange". In this study, the Agglomerative clustering algorithm (ACA) was used based on the features provided in Table 1. to categorize the apps because it bases group objects in clusters in terms of their similarity. This algorithm is considered a greedy algorithm because of how it determines the appropriate mergers and splits. In Figure 2., the "risky" vs "allowed" apps are identified and separated to enable users to determine which apps should be accepted into the repositories. The ability to make this determination beforehand is considered critical to maintaining the desired security posture of the data repositories so that users and owners can feel confident that vulnerabilities are not being passed on. It is usually prudent to consider at least another clustering algorithm for comparison of the results. In this study, the density-based (DBscan) algorithm was selected for comparison vs the ACA algorithm. The DBScan representation is interesting because it identifies an "intermediate" class between the risky and allowed points. The ACA was chosen, to get a clear delineation between "risky" and "allowed" OAuth apps.

In addition to the OAuth risky third-party apps, the other challenge noted to impact data repository security, i.e., the way that pull requests (PRs), commits and other changes are allowed within data repositories (Yu, et al., 2018). Researchers have recently started to investigate the pull request workflow because the evaluation of PRs is an iterative, complex process involving many stakeholders (Gousios, et al., 2014), (Tsay & Dabbish, 2014), (Tsay, et al., 2014), (Tsay, et al., 2014). Unfortunately, the factors that can influence PR acceptance (decision to merge) and latency (evaluation time) are still not well understood. In addition, PR evaluations have not yet been fully examined in terms of continuous integration (CI) (Jiang, et al., 2017).

In Figure 3, shown below, a GitHub pull-request operation flow diagram is provided.

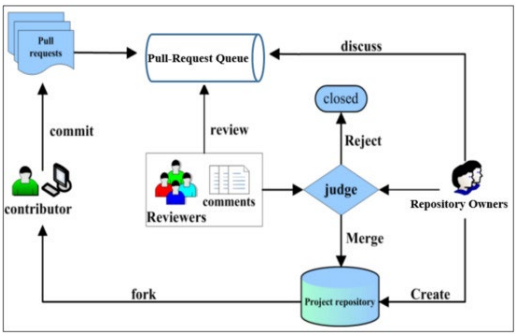

Figure 3. Overview of Pull-Request Mechanism in GitHub, (Salman, et al., 2022)

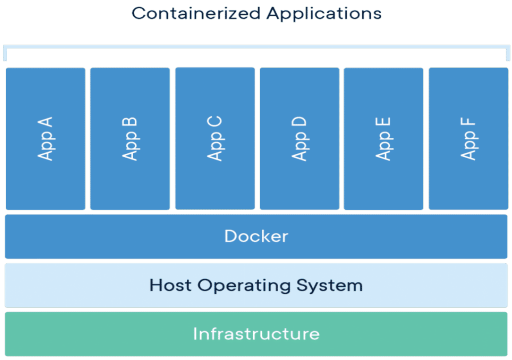

Figure 4. Docker Containerized Applications (See What is a Container? | Docker)

The continuous and parallel distributed development make managing changes in software development very difficult (Nerur, et al., 2005) (Dabbish, et al., 2012).

Containers enable code to be rolled out more quickly and can enable app development time to be shortened while using fewer storage and compute resources. Containers are usually accompanied by an orchestration tool, which essentially controls the container. Most of the orchestrator and container vulnerabilities correspond to the classic vulnerabilities usually found in applications and operating systems, including: access and authorization, API server access, image vulnerabilities, inter-container network traffic, and other (Mell & Pariseau, 2023).

What is needed is a framework that makes it possible to verify the authenticity of an artifact (in this case a container image) and check for anomalous behavior (vulnerabilities) due to cyber attacks. The image contains code, runtime, system tools, system libraries and settings as shown in Figure 4. In this work, given the interest is in determining if a container image has a vulnerability, it is proposed that natural language processing (NLP) be applied to container image recognition to identify vulnerabilities. In this approach a standard "WordPiece" tokenizer that tracks fractions of words would be used and converted into select features of the image. The WordPiece approach could cluster traffic in an analytically meaningful way. The benefit is that no preprocessing would be required because raw input could be tokenized directly. Additionally, a key takeaway is that the proposed model would group the artifacts by behavior and automatically learn to characterize "normal" vs. altered behavior. In this approach, the system would learn enterprise normality. This approach also uses embeddings to identify adversarial bias in which a key takeaway might be the ability to tell the degree of threat just by its relative position in embedding space. Words generally arrange themselves spatially along a continuum of meaning. Embeddings can be used to answer complex what-if questions that are very hard to do in some other cases. What-if Analysis is employed using Embedding Vector Math. Connection embeddings show adversarial bias, wherein the difference between embeddings of two communicating containers is an embedding for their connection. In most cases, most of the suspicious traffic forms a cluster. The benefit is that the model can separate traffic into analytically meaningful types without supervision (i.e., no labels).

6. Conclusion

In this study, a couple of the factors have been reinforced as key contributors to the recent data repository breaches, including: the risk associated with suspicious OAuth apps and the challenging process of commits and pull requests. Most of the attention in this study was placed on the OAuth apps. In particular, there are a number of criteria used to determine when an app should be allowed or considered too risky. To date, there is very little effort to isolate and define the "bad" OAuth apps. A table was presented that does the isolation and specifically defines parameters that must be addressed before apps can be allowed into data repositories. In the study, 100 arbitrarily selected apps were successfully clustered based on pre-defined criteria using Agglomerative clustering and DBscan algorithms to define specifically those apps that are allowed and those that should be excluded. Zero trust architectures, without a doubt have improved the security posture of most current computer environments. However, some challenges still remain in terms of when it will be ready to address data repository vulnerabilities. A new approach introduced by PNNL to visualize normal enterprise behavior via

natural language processing applied to address the huge data in cyber logs because cyber log data is different and there are domain specifics that can help or hinder analysis. The approach is considered in this work to address the vulnerabilities in containers, OSs, and dependencies (Ranade, et al., 2021) and mitigate vulnerabilities passed on from container to that can get pulled into the host computer. Future efforts will focus on the commits and pull requests challenges and the development of a machine learning pipeline to include both the risky Auth apps and commits/pull request challenges. In addition, other machine learning algorithms will be considered to compare performance with the decision tree algorithm.

Bibliography

Chimakurthi, V., 2022. The Challenge of Achieving Zero Trust Remote Access in Multi-Cloud Environment. *ABC Journal of Advanced Research,* 9(2), pp. 1-13.

Curwin, D. et al., 2023. *Investigate and Remediate Risky OAuth Apps.* [Online]
Available at: https://learn.microsoft.com/en-us/defender-cloud-apps/investigate-risky-oauth
[Accessed 19 01 2023].

Dabbish, L., Stuart, C., Tsay, J. & Herbsleb, J., 2012. Social Coding in GitHub: Transparency and Collaboration i an Open Software Repository. *Association for Computing Machinery*, 12-17 02, pp. 1277-1286.

Fair, L., 2018. *FTC Addresses Uber's Undisclosed Data Breach in New Proposed Order.* [Online]
Available at: https://www.ftc.gov/business-guidance/blog/2018/04/ftc-addresses-ubers-undisclosed-data-breach-new-proposed-order
[Accessed 19 01 2023].

Fearn, N., 2020. *How to Apply Zero-Trust Models to Container Security.* [Online]
Available at: https://www.computerweekly.com/feature/How-to-apply-zero-trust-models-to-container-security
[Accessed 07 02 2023].

Gousios, G., Pinzger, M. & and v.Deursen, A., 2014. An Exploratory Study of the Pull Based Software Development Model. *International Conference on Software Engineering, ICSE*, pp. 345-355.

Jiang, J. et al., 2017. Analyzing Attributes for More Accurate Commenter Recommendation in Pull-Based Development. *Science Direct,* 84(Information and Software Technology), pp. 48-62.

Kapko, M., 2023. *Open-source Repository Risk Amplified on GitHub.* [Online]
Available at: https://www.cybersecuritydive.com/news/open-source-repository-risk/640333/
[Accessed 19 01 2023].

Kaur, B., Dugre', M., Hanna, A. & Glatard, T., 2021. An analysis of security vulnerabilities in container images for scientific data analysis. *Gigascience,* 10(6), pp. 1-7.

Nerur, S., Mahapatra, R. & Mangalaraj, G., 2005. Challenges of Migating to Agile Methodologies. *Communications ACM ,* Volume 48, pp. 72-78.

Page, C., 2022. *Okta Confirms Another Breach After Hackers Steal Source Code,* San Francisco: TechCrunch.

Powell, O., 2022. *GitHub Supply Chain Attack Could Affect 83 Million Developers.* [Online]
Available at: https://www.cshub.com/attacks/news/github-supply-chain-attack-could-affect-83-million-developers
[Accessed 12 01 2023].

Raina, K., 2022. *www.crowdstrike.com.* [Online]
Available at: www.crowdstrike.com/cybersecurity-101/zero-trust-security/
[Accessed 12 01 2023].

Rose, S., Borchert, O., Mitchell, S. & and Connelly, S., 2020. *Zero Trust Architecture.* [Online]
Available at: https://www.nist.gov/publications/zero-trust-architecture
[Accessed 12 01 2023].

Salman, H., Alshar, Z. & Seriai, A., 2022. *Automatic Identification of Similar Pull-Requests in GitHub's Repositories Using Machine Learning.* [Online]
Available at: https://www.mdpi.com/2078-2489/13/2/73
[Accessed 19 01 2023].

Scott, T. & Spitse, N., 2022. *Third Party Risk is Becoming a First Party Challenge.* [Online]
Available at: dhttps://www2.deloitte.com/ca/en/pages/risk/articles/reduce-your-third-party-risk.html
[Accessed 05 02 2023].

Shah, S., 2022. *How OAuth 2.0 Supports Third-Party Applications to Authorize HTTP Service?.* [Online]
Available at: https://www.einfochips.com/blog/the-use-of-oauth-2-0-framework-for-third-party-applications-to-authorize-http-service/
[Accessed 19 01 2023].

Souppaya, A., Scarfone, K., Symington, S. & Barker, W., 2022. *Implementing a Zero Trust Architecture,* Gaithersburg: NIST.

Tolbert, J., 2020. *A Look at NIST's Zero Trust Architecture.* [Online]
Available at: https://www.kuppingercole.com/blog/tolbert/a-look-at-nists-zero-trust-architecture
[Accessed 12 01 2023].

Tsay, J., Dabbish, L. & Herebsleb, J., 2014. *Let's Talk About It: Evaluating Contributions through Discussions in GitHub.* Kansas City, ACM.

Tsay, J. & Dabbish, L. H. J., 2014. *Infleuence of Social and Technical Factors for Evaluating Contribution in GitHub.* Kansas City, ACM.

Wang, Y. et al., 2020. *An Empirical Study of Usages, Updates and Risks of Third-Party Libraries in Java Projects.* Cyprus, IEEE International Conference on Software Maintenance (ICSM).

White, D., 2022. *How Zero Trust Models Work in Container Security.* [Online] Available at: https://thenewstack.io/how-zero-trust-models-work-in-container-security/ [Accessed 07 02 2023].

Yu, S. et al., 2018. *NBSL: A Supervised Classification Model of Pull Reques in GitHub.* Kansas City, IEEE International Conference on Communications (ICC).