API Drift Detection Enhancing Data Protection

Anoop Gupta
Capital One Services LLC
Richmond, VA, 23238, US

ABSTRACT

The prevalence of data breaches in Application Programming Interfaces (APIs) has been highlighted in a recent Gartner study. The study predicts that APIs will become increasingly vulnerable to data leakage as the volume of data transmitted across organizational boundaries grows. APIs undergo multiple lifecycle phases and are handled by various engineers before they are deployed in production environments, which increases the likelihood of security breaches. This paper provides a comprehensive analysis of how to detect data leakage by identifying API drift and how to address such drift by pinpointing the exact location of the leakage. [1]

Keywords

API Security, Cybersecurity, Data Loss Prevention, API Drift, API Gateway, API Lifecycle, API Governance, API Discovery.

1. INTRODUCTION

As enterprises experience growth, the demand for Application Programming Interfaces (APIs) to serve the needs of customers is on the rise. This heightened need is being driven by the increasing complexity introduced by the widespread adoption of cloud computing and microservices architecture. This rapid proliferation of APIs led to the API Sprawl problem. This phenomenon results in the development of numerous shadow, dormant, redundant, and duplicate APIs, which in turn pose significant challenges in managing and maintaining consistent security policies across the entire enterprise. As a result, the overall security posture of the organization is impacted, necessitating a focus on effective API governance and management strategies. [2][3]

The risk of security breaches is a major worry, especially when it involves unauthorized access to sensitive data through rogue APIs. Such exposure can result in data breaches and financial losses, damaging the company's reputation. Both unauthorized (rogue) and authorized APIs can be detected using out-of-band network discovery with an API Gateway, which acts as a centralized entry point for APIs, or with a centralized API Management system that utilizes a cataloging solution.

It's important to address various issues, such as inconsistencies in API design, documentation standards, authentication and authorization requirements, data quality, and third-party contracts to ensure security. These inconsistencies in APIs are commonly known as API Drift and should be carefully managed to mitigate potential security vulnerabilities.

2. API DRIFT IDENTIFICATION

Drift detection is a crucial process that involves continuously monitoring and analyzing the operational state of a system to identify any deviations from the expected state. These discrepancies may arise from a variety of factors, such as changes in data, environment, or system behavior. It is imperative for API producers to rigorously adhere to the specifications set during the design phase, including data formats, protocols, and expected behaviors, to minimize the occurrence of such deviations. By continuously monitoring and ensuring compliance with these specifications, API producers can effectively mitigate the impact of drift and maintain the integrity and reliability of their systems.

2.1 Attack Surface

An API's attack surface refers to the total scope of potential vulnerabilities and weaknesses that could be targeted and exploited by malicious attackers. This encompasses all technical aspects of the API implementation as well as the surrounding infrastructure that supports it. Since APIs have direct access to critical data and services, they pose a significant risk to organizations if not properly secured. It's important to note that while web applications have historically been the primary targets for cyber attacks, APIs now constitute a staggering 90% of their overall attack surface, making them a crucial focus for security measures and risk mitigation strategies.

2.2 Authorization Drift

Access to the APIs is regulated by authorization policies that may be stored either in a centralized system such as API Gateway or with the API Producers. Authorization involves the process of determining a user's access rights. This access control can be implemented in various ways based on the specific application's business rules. In the context of APIs, each request to a protected endpoint needs to be evaluated to ascertain whether the client is authorized to access and execute the requested action. Usually, this evaluation is performed at the code level and involves verifying that the client possesses the necessary permissions to access the resource provided by the API endpoint.

Fine-grained access control methods such as Relationship-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and Policy-Based Access Control (PBAC) are designed to provide precise control over who can access specific resources within an API. By implementing these access control methods, organizations can effectively mitigate the risk of critical data breaches that may occur due to unauthorized access to API endpoints. These methods allow for granular control over access permissions based on factors such as user roles, attributes, and defined policies, thereby enhancing the overall security posture of the API infrastructure.

Over time, the authorization configurations for APIs may change, leaving the API data vulnerable to security breaches and exposing customers' data.

2.3 OpenAPI Specification Drift

The OpenAPI specifications play a crucial role in providing a well-structured and standardized framework for defining APIs. These specifications enable engineers to gain comprehensive insights into the functionality of the API, including its endpoints, request and response formats, authentication methods, and more. When coupled with API Management, the API undergoes a thorough producer lifecycle, which encompasses planning, design, review, development, testing, and deployment phases. This ensures that the API is developed, tested, and deployed in a controlled and efficient manner.

Despite the inclusion of robust processes to align API implementation with the prescribed OpenAPI specification and comprehensive reviews, developers may inadvertently deviate from the specified standards for various reasons. These reasons may include tight product modifications over time, project deadlines, evolving requirements, miscommunication, or a lack of awareness about the defined standards. Therefore, it is essential for teams to continuously monitor and enforce adherence to the OpenAPI specifications to ensure the consistency, quality, and security of the developed APIs. [4][5][6]

2.4 Data Attribute Drift

API Sprawl refers to the excessive proliferation of APIs within an organization, which can lead to the emergence of competing standards. This situation is common in large enterprises that have numerous APIs and multiple versions. When this happens, both the consumers and producers of these APIs may deviate from established enterprise data standards. As a result, non-compliance with data standards and enterprise security policies may occur, potentially exposing sensitive data to security risks.

One of the challenges posed by API Sprawl is the lack of consistent attribute naming patterns among the APIs. This can lead to conflicting standards for the same underlying attribute, making it difficult to enforce governance and security measures, such as encryption or tokenization, across the enterprise. Consequently, data privacy issues may arise due to the challenges in governing these attributes.

2.5 Data Quality Drift

APIs are made up of multiple endpoints, with each endpoint exposing various data attributes that consumers can access. These data attributes may have different data validations and types, and it's common for an API to deviate from the initial design specifications over time, leading to potential issues. One frequent issue that arises is data type drift, where integers are incorrectly interpreted as strings. This can lead to problems when the backends consume this data without proper validation, potentially causing errors or unexpected behavior in the application.

In addition, inadequate validation, such as through regular expressions or range checks, for an attribute can also cause problems. When consumers and producers operate based on undocumented specifications, it can result in unauthorized access to the data, potentially compromising data security and integrity.

2.6 Data Contract Drift

Large organizations develop APIs not just for their internal use cases but also to integrate with third-party systems. When customer data is shared outside the company, it's essential to have clear agreements with external partners. These agreements define the specific data shared with partners and establish rules for managing the shared data.

Sometimes, API providers might accidentally send more information to partners than intended, leading to data leaks on ungoverned data attributes.

3. API DRIFT DETECTION

Despite efforts to implement the shift left approach, which involves capturing and registering specifications early in the design phase and establishing governance and standardization processes, organizations continue to experience persistent drifts. Detecting drift in APIs in a timely manner is crucial, as doing so enhances security posture and fosters trust from third-party entities during the integration process.

3.1 Detection Methods

As the demand for business operations grows, the volume of API transactions also increases, posing a significant challenge in effectively monitoring and identifying deviations at such a large scale. Implementing non-intrusive, decoupled, and scalable techniques becomes essential to address this growing complexity and ensure efficient management of API transactions and security. Each drift detection method can be implemented as a standalone microservice that is designed to operate independently. This microservice is responsible for identifying any drift in the system and perhaps taking appropriate corrective actions to remedy the detected drift. By encapsulating drift detection and remediation within a loosely coupled microservice architecture, a more modular and scalable approach to managing system drift can be achived.

3.1.1 Out-of-band API Traffic Scanning

By employing out-of-band scanning techniques and intercepting network traffic, it is possible to identify APIs that are accessible on the Internet. This process can reveal API endpoints that may be exposing sensitive data and are vulnerable to unauthorized access.

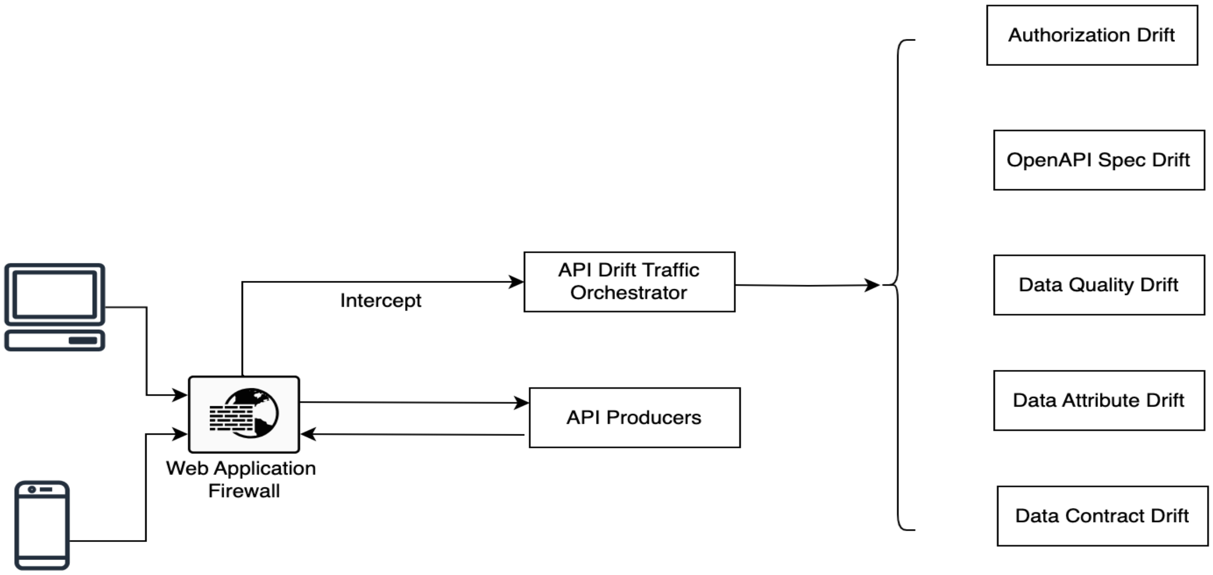

Fig 1: API Traffic Scanning

3.1.2 API Gateway

The API intermediary, also referred to as an API Gateway, serves as the primary entry point for all incoming and outgoing API traffic within an organization. It plays a crucial role in scanning and identifying any deviations or drifts in the API traffic, ensuring that all data transmission remains secure and compliant with organizational standards.

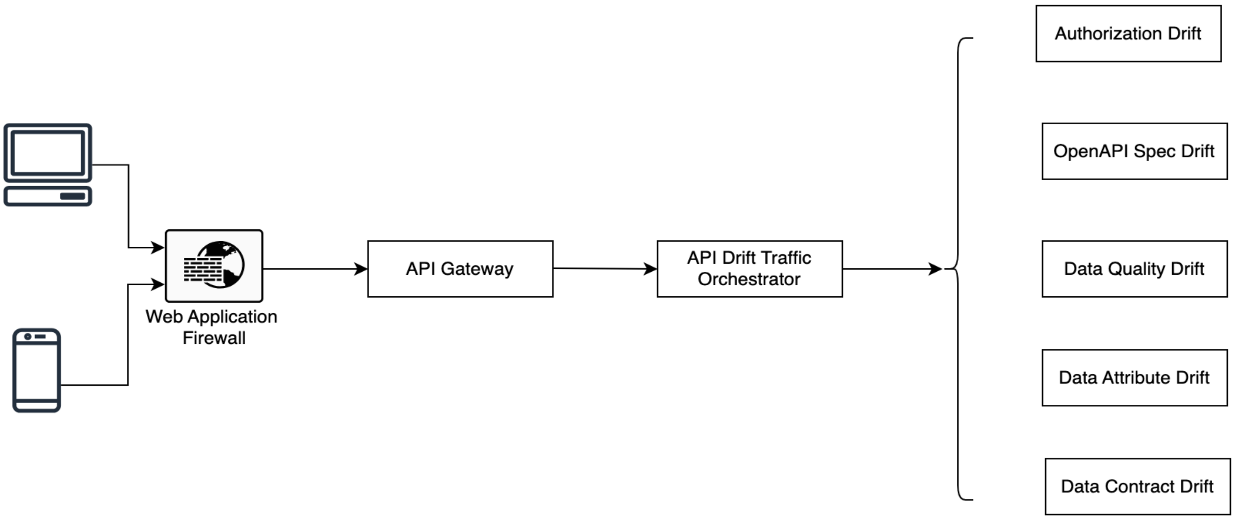

Fig 2: API Gateway Asynchronous Scanning

3.1.3 Centralized Drift Service

API providers have the capability to utilize centralized drift services in order to transmit API traffic asynchronously for scanning purposes.

4. MACHINE LEARNING

The identification of certain deviations, as mentioned earlier, can be facilitated by following the correct procedures. However, the utilization of machine learning for the purpose of identifying non-standardized data attributes has the potential to yield significant advantages. This involves the input of an extensive and continually growing repository of data attributes and their respective properties, in addition to domain and sub-domain information. By leveraging machine learning models, it becomes possible to determine whether there exists a more appropriate attribute for a given purpose and to subsequently apply the relevant governance and security policies based on the associated risk profile.

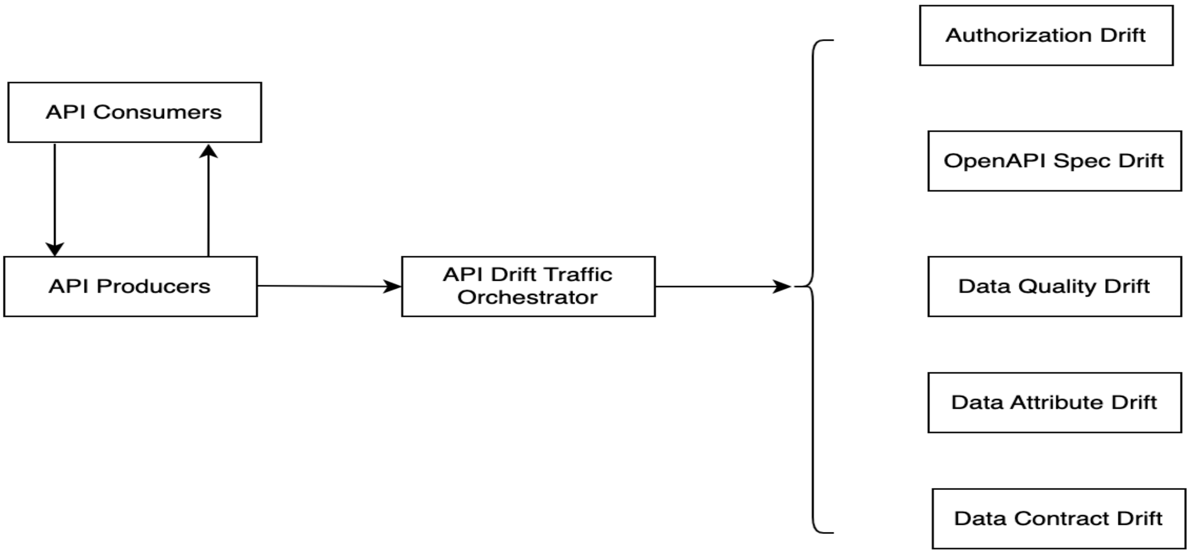

Fig 3: API Drift Centralized Service

5. DRIFT REMEDIATION

Addressing drift issues within a system can be quite challenging and requires thorough attention to detail. Remediation involves the meticulous process of identifying and understanding the potential risks posed by the drift, evaluating the severity of the issue, and determining its impact on the various components of the system. It also involves identifying the API producer and consumers, tracking occurrences and metrics to recognize patterns, and establishing a comprehensive process to address and remediate the vulnerability. A centralized solution, which consolidates and streamlines these processes, can offer an effective and efficient approach to resolving these complex challenges.

6. CONCLUSION

As awareness of the critical importance of API security continues to grow, incidents of data breaches involving APIs remain prevalent. This ongoing challenge underscores the need for application leaders to take proactive measures. By developing and implementing a comprehensive API security strategy, organizations can significantly enhance the protection of their APIs against potential threats and vulnerabilities.

One effective data protection strategy that can be employed is API Drift detection and remediation. This technique is designed to monitor and safeguard customer data by identifying any discrepancies or unauthorized changes in API behavior or integrations. API Drift detection utilizes various methodologies to track changes within the API ecosystem, allowing organizations to respond swiftly to emerging threats.

There are multiple solutions available for implementing drift detection techniques, each with its unique features and benefits. Application owners can choose from a range of options outlined in this paper, recent research and other industry papers, tailoring their approach to align with their specific architectural needs and organizational objectives. By selecting the most appropriate implementation, application leaders can ensure that their APIs are not only secure but also resilient against potential breaches, thereby reinforcing their overall data protection strategies.

Despite the increasing awareness of the importance of API security, data breaches involving APIs continue to happen. However, application leaders can take independent action by devising and implementing a robust API security strategy to safeguard their APIs from potential threats and breaches. API Drift detection and remediation is one such data protection strategy for safeguarding customer data.

7. REFERENCES

[1] McKinley Sconiers-Hasan, Application Programming Interface (API) Vulnerabilities and Risks, June 2024 https://insights.sei.cmu.edu/documents/5908/api-vulnerabilities-and-risks-2024sr004-1.pdf

[2] Gartner Reports, May 2024 https://www.gartner.com/en/documents/5471595

[3] Larry Ponemon, State of API Security, 2023 https://ponemonsullivanreport.com/2023/09/state-of-api-security-2023-global-findings/

[4] J Stimpson, Most APIs Suffer From Specification Drift, September 2024 https://nordicapis.com/most-apis-suffer-from-specification-drift/

[5] API Context, Bad API Specification Management Creates Application Risk and Challenge https://apicontext.com/bad-api-specification-management-creates-application-risks-and-challenges-new-white-paper/

[6] APIContext, API Drift https://apicontext.com/resources/api-drift-white-paper/