National Centre of Excellence
CYBERSECURITY TECHNOLOGY AND ENTREPRENEURSHIP

इलेक्ट्रॉनिकी एवं सूचना प्रौद्योगिकी मंत्रालय
MINISTRY OF ELECTRONICS AND INFORMATION TECHNOLOGY

DSCI
PROMOTING DATA PROTECTION
A nasscom Initiative

Application Programming Interfaces

API SECURITY

Contributor

Prof. Sonali Bhutad, Assistant Professor, SAKEC

Table of
CONTENTS

1

Introduction

APPLICATION PROGRAMMING INTERFACES (APIS) PROVIDE THE FOUNDATION FOR COMMUNICATION BETWEEN VARIOUS SOFTWARE SYSTEMS IN TODAY'S NETWORKED DIGITAL ENVIRONMENT.

Application Programming Interfaces (APIs) provide the foundation for communication between various software systems in today's networked digital environment. APIs make it easier to share information and services, which makes it possible for a variety of applications to connect and work together smoothly. But there are also serious security issues brought about by the widespread use of APIs. Strong API security is essential for safeguarding private information, upholding confidence, and guaranteeing the seamless functioning of online services. Let us understand these terms of API security:

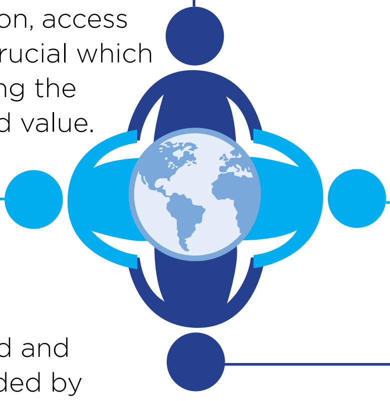
Importance of API security in modern digital eco-systems

Safeguarding Privat Information

To protect sensitive data from unwanted access and modification, effective API security features like encryption, access limits, and data masking are crucial which leads to enhancing trust among the customers by sustaining brand value.

Preserving the Business Continuity via Innovation

Attacks can be avoided and business continuity can be guaranteed by putting security measures like rate limitation, IP white listing, and intrusion detection systems into place. Innovations in business processes such as supply chains will help scalability by providing different services to global customers.



Stopping Data Breach an Unauthorized Access

These dangers can be reduced and unwanted access can be avoided by putting best practices like thorough logging and monitoring, frequent security testing (like penetration testing), and robust authentication (like OAuth 2.0) into effect.

Maintaining Adherence to Regulations

Implementing strong API security practices, such as data anonymity, audit trails, and secure authentication systems.

Safeguarding Private Information

APIs frequently deal with sensitive data, such as financial information, intellectual property, and personal data. Any compromise or breach could have serious repercussions, including identity theft, monetary loss, and harm to one's reputation

Stopping Data Breach and Unauthorized Access

Attackers looking to obtain unauthorized access to systems and data may find APIs to be a tempting target. Security defenses can be breached by taking advantage of common weaknesses including incorrect error handling, broken authentication, and a lack of rate limiting.

Preserving the Business Continuity via Innovation

The operation of many services and applications depends on APIs. An API attack or security breach can cause interruptions, and monetary damages by interfering with business processes.

Maintaining Adherence to Regulations

Organizations can meet compliance standards and show their dedication to protecting user data. Strict guidelines for data protection and privacy are enforced by legal frameworks such as the California Consumer Privacy Act (CCPA), the General Data Protection Regulation (GDPR), and the Health Insurance Portability and Accountability Act (HIPAA). To avoid steep fines, APIs handling regulated data must abide by these rules.

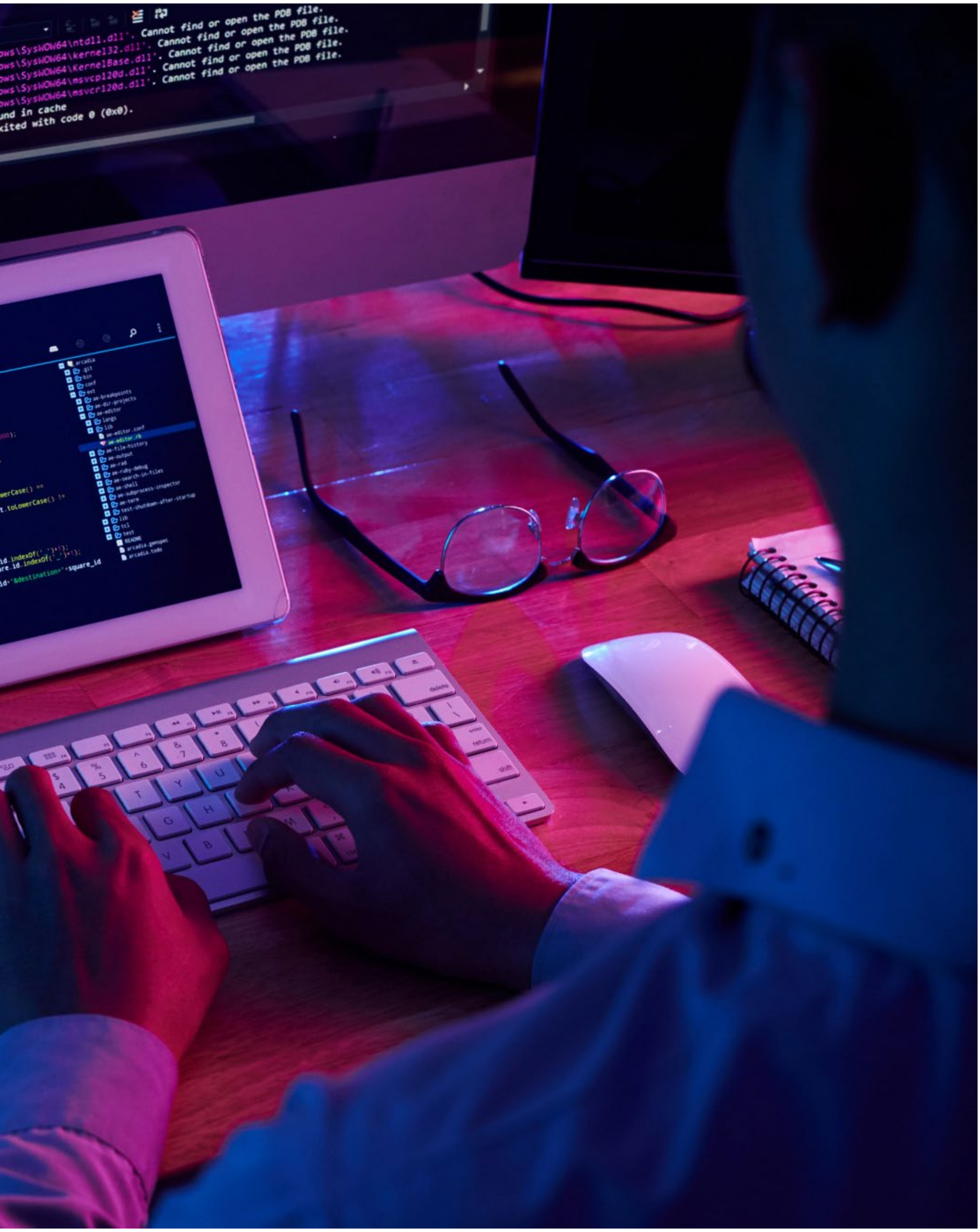
API Threat Landscape, Challenges and Evolving Risks

Escape's security research on the 2022–2024 API Security Threat Landscape emphasizes the increasing dangers and monetary consequences of data breaches with APIs. Since 2022, more than 190 million records have been compromised, potentially resulting in $31 billion in losses, with the most affected sectors being government, healthcare, and technology. Over 40% of breaches involve zero-day vulnerabilities, which are flaws that creators are unaware of but that attackers take advantage of. According to the OWASP API Security Top 10 for 2023, broken authentication and broken object level authorization are frequent attack vectors. Notable breaches that frequently result from security setup errors or exposed API tokens include those that impact Dell (49 million records), Trello (15 million users), and T-Mobile (37 million subscribers). The growing number of unmanaged and undocumented APIs, known as "API sprawl," has made the situation much worse.

The report highlights automated security testing, robust access controls, and API discovery and inventory management as ways to reduce these risks. To find vulnerabilities before they are exploited, organizations are urged to implement role-based access control (RBAC), attribute-based access control (ABAC), and continuous monitoring. Because of their versatility, GraphQL APIs provide special dangers including recursive queries, schema leaking, and brute force attacks. As a result, they necessitate extra security measures like query depth limitation and authentication middleware enforcement. This report emphasizes the value of proactive API governance and security training, emphasizing that companies must give API security top priority in order to protect sensitive data, stay in compliance, and avoid expensive breaches.

Objective of Paper

The core functions and micro-service architecture of mobile applications, websites, applications, etc., are inseparable from the support of APIs. The scrum development model is a mainstream API development mode and in improving the speed and flexibility of innovation, API risks are underestimated which leads to API construction security ignorance. Therefore, to understand the API security posture this paper focuses on API security starting from API development phase till API testing.



2

API Security Trends for 2025

SECURE DESIGN IS ABOUT FORESEEING HAZARDS AND SUCCESSFULLY MITIGATING THEM, WHETHER THE GOAL IS TO PROTECT SENSITIVE DATA, STOP UNWANTED ACCESS, OR PRESERVE SYSTEM INTEGRITY.

Nowadays the security maintenance is shifted from reactive security measures to building robust systems from the ground up through secure design. By doing this, you can ensure that security is incorporated into the software as a fundamental feature rather than being handled as an afterthought. Secure design is about foreseeing hazards and successfully mitigating them, whether the goal is to protect sensitive data, stop unwanted access, or preserve system integrity. By ensuring that security policies are automated, version-controlled, and integrated into CI/CD pipelines, security as code implementation increases the scalability and effectiveness of risk management. In order to minimize manual intervention and human mistake, systems should also impose safe defaults, such as HTTPS and strong passwords. By providing the bare minimum of permissions to individuals and systems, the idea of least privilege lowers the possibility of unwanted access. Separation of roles also creates checks and balances by requiring several approvals for important acts, which helps guard against misuse or errors.

It is more difficult for attackers to take advantage of vulnerabilities when the attack surface is reduced by eliminating useless features, shutting down unused ports, and removing expired APIs. By guaranteeing that each access request is regularly validated, complete mediation stops unwanted access brought on by out-of-date permits. Additionally mistakes, particularly in authentication, should result in denial rather than access. When combined, these security guidelines provide a more effective, managed, and proactive defense against online attacks. Recognizing vulnerabilities in running applications is the proactive security measurement using Dynamic Application Security Testing (DAST) which plays a significant role in achieving secure software development by secure design principles and DAST must be integrated as DAST actively tests an application's attack surface during development and deployment.

Adoption of Zero Trust

The foundation of the Zero Trust paradigm is the idea "never trust, always verify." This method requires constant authorization and authentication for all users, devices, and applications, regardless of their location, based on the assumption that risks exist both inside and outside of conventional network borders. Rethinking how APIs are accessed and safeguarded in cloud-native environments is necessary to apply Zero Trust concepts to API security. This strategy is very important as APIs are now the main interface for data interchange and service integration in contemporary applications. Continuous authentication and authorization, micro-segmentation, least privilege access, encryption and data protection, and continuous monitoring and analytics are some of the ways that Zero Trust principles appear in the context of API security. A Zero Trust API security posture is facilitated by the use of mutual TLS (mTLS) for secure communication, OAuth 2.0 for authorization, and API gateways that allow robust authentication methods. Furthermore, the dynamic nature of cloud-native settings is particularly suited to the implementation of Zero Trust in API security. It is crucial to impose security standards at the API level as more and more apps are being controlled by platforms like Kubernetes and deployed in containers. In highly dispersed and scalable infrastructures, this granular approach to security aids organizations in keeping control over their data and services.

API Supply Chain Security

Assume that the online shop protects its consumers so well that it is extremely impossible for an attacker to compromise the system and obtain confidential customer information. Since the delivery service is a separate business, the attacker can now choose to target it rather than the store; they may have less stringent security measures than the online store, but even if they are successful, the objective is still accomplished because the delivery service now has all the required private data. This type of attack, which targets a weak point in the service's API ecosystem, is known as a "API Supply Chain Attack."

Each layer of interdependencies generated by the complex web of third-party libraries, suppliers, and platforms that modern software development ecosystems rely on is difficult to monitor and secure; a single weak link in this chain can leave an entire system vulnerable to attack. Even when vulnerabilities are found, it frequently takes 60 to 150 days to fix them completely, leaving systems vulnerable for an excessive amount of time. Rapid remediation is hampered by

manual procedures, communication snags, and a lack of automation. Attackers have plenty of opportunity to take advantage of known vulnerabilities when patching or updating vulnerable components is delayed. Effective software supply chain risk management requires organizations to streamline procedures and expedite communication between the development and security teams.

For securing API Supply chains, evaluate third-party and vendor software, develop automatic dependency audits that scan the complete CI/CD environment, and update versions based on known vulnerabilities and patches. This involves examining the tools, scripts, and dependencies that support your pipeline for errors, improper access controls, and vulnerabilities.

Use of Artificial Intelligence (AI) and Machine Learning (ML) for API Security

Pattern identification linked to known attack vectors, anomaly detection by verifying the API's typical behavior, and behavioral analysis by observing interactions are all possible applications for AI-powered threat detection. AI is essential for responding to and reducing dangers; it is not just for detection. AI-powered automated response systems are capable of incident response, rate limitation, and real-time blocking of vulnerabilities. Furthermore, improved authorization and authentication will support behavioural biometrics and adaptive authentication.

API Security Platforms

The significance of API security solutions, which protect APIs against misuse, unauthorized access, and exploits, is described in the Gartner report. These solutions, which ensure API security throughout their lifecycle, include runtime protection, security posture management, and API discovery. Cloud-based or on-premises solutions can be used to implement API protection, helping organizations in reducing security risks and fixing vulnerabilities.

Important companies offering reviews and ratings that assist businesses in comparing solutions include Check Point, Imperva, Akamai, Google, Cequence Security, and others. It showcases top API security tools, including Imperva API Security, Akamai API Security, and Check Point CloudGuard WAF.

Increased Collaboration between Security and Development Teams

The strategy that integrates API development, security, and operations is called API DevSecOps. Offering the detection and mitigation of security issues, it seeks to guarantee that APIs are built securely, implemented effectively, and monitored effectively.

API DevSecOps puts security at the centre of the API creation process rather than as an afterthought, enabling businesses to produce trustworthy and safe APIs. The ability of API DevSecOps to find and fix vulnerabilities early in the development process is one of the main factors contributing to its importance. Organizations can identify possible vulnerabilities before they become serious security breaches by including security evaluations into the pipeline. This preventative measure lowers the possibility of expensive and destructive data breaches or cyber attacks.

Applying API DevSecOps frequently presents difficulties, such as handling the intricacy of striking a balance between security and functionality, blending in skill gaps in both development and security, integrating security seamlessly throughout all stages of development, modifying security procedures for legacy systems, allocating sufficient resources, and promoting a collaborative culture.

3

API Security Challenges & Case Studies

API security is a crucial component of cyber security since APIs serve as the link between various systems.

Common API Vulnerabilities

Broken Object Level Authorization

Object's sensitive fields are Improperly exposed

Broken User Authentication and Function Level Authorization

Broken Authentication API operations are improperly authorized, Function Level Authorisation. unauthorized access by API calls

Improper Asset Management

Private data is either not accessible or is not adequately safeguarded via APIs

Excessive Data Exposure

Availability of too much data leads to reconnaissance

Lack of Resources & Rate Limiting

No limitations on API request

Security Risks in Different API Architectures

The choice between GraphQL and REST APIs is crucial in the quick-paced world of web application development. Although each technology has advantages, there are also unique security risks. The following table shows vulnerabilities associated with each API architecture.

GraphQL	REST API
✓ Insecure Endpoints	✓ Introspection Queries
✓ Over-fetching and Under-fetching	✓ N+1 Query Problem
✓ Inadequate Rate Limiting	✓ Data Over-fetching
✓ Injection Attacks	✓ Enumeration Attacks

Case Studies

Attackers may be able to assume the identity of the authorized user and access their account if a user's personal information is not safely secured in an API response that is returned to the user's browser or mobile device. Applications that rely on APIs frequently experience this problem which is described in the following examples

- **By taking advantage of a broken object-level authorization (BOLA)** bug in September 2019, Uber's API had a serious vulnerability that allowed hackers to access or alter data without the necessary authorization checks, potentially stealing customer data and taking over accounts. By submitting API requests with a user's phone number or email address, attackers were able to list all Uber user IDs. They could retrieve the user's location, payment details, and even access tokens for mobile apps by replaying the request after they obtained the user's ID. The user's account might be completely compromised with these access tokens, giving hackers the ability to track location, request rides, and more.

- **Through the cdn.polyfill.io domain,** Polyfill, a package that helps older browsers support new capabilities, is frequently used. After acquiring the domain and GitHub account in February 2024, the Chinese business Funnull altered Polyfill.js to introduce malicious malware into any website that used it. This hack permitted unauthorized code execution, stole confidential information, and sent users to fraud websites. This malicious code abused users' browsers to carry out a variety of malicious tasks. It was inserted within scripts from domains such as bootcdn[.]net, bootcss[.]com, and polyfill[.]io. These included stealing private data, sending visitors to phishing websites, and engaging in other illegal activities similar to previous Cross-Site Scripting (XSS) attacks.

4

API Security Framework and Best Practices

API Discovery & Inventory Management

Inventory all APIs

The process of locating and categorizing the APIs that are being used by a company is known as API discovery. The goal of API discovery is to compile a thorough list of all APIs, both internal and external. Following set of questions will help you to perform API inventory,

- How many APIs does your company now use?
- What happens to an earlier version of an API when a new version is launched?
- How do those modifications affect the functionality of your application?
- How often were they updated or modified?
- How do you locate an unofficial API that was made by a developer to cut down on rework?

Search for Zombie and Shadow APIs

A shadow API operates outside of the established IT governance, security procedures, and peer review systems of your company. Due to their lack of documentation, tracking, and accounting, these APIs are unsafe and poorly maintained.

Outdated, forgotten, or abandoned exposed APIs and API endpoints are known as zombie APIs. They were legitimate, authorised, and had a purpose in the past. However, as they are no longer required, they have been upgraded or replaced with a newer version, or they have just been dropped. Shadow APIs are unapproved and might not even be legitimate, in contrast to zombie APIs, which were once authorised and legitimate. Continuous API discovery by analyzing live traffic at internal and external endpoints helps in discovering the shadow and zombie APIs.

Authentication & Authorization

Enforce OAuth 2.0, OpenID Connect

The industry-standard protocols for user identification and authorisation are OpenID Connect (OIDC) and OAuth 2.0.

OpenID Connect (OIDC): Adds user authentication and Single Sign-On (SSO) features to OAuth 2.0. It allows you to store and retrieve your end users' authentication data. Additionally, it specifies a number of OAuth 2.0 scopes that allow apps to access user profile data.

OAuth 2.0 : Regulates and assigns permission to use a protected resource, such as your API service, web application, or native app. It uses scoped access tokens to provide API security.

Role-Based (RBAC) and Attribute-Based (ABAC) Access Controls

Role-based access control (RBAC) and attribute-based access control (ABAC) are two of the most popular techniques for protecting access to company resources. There are several significant distinctions between these kinds of access control models, which rely on the user's identity and the resources they are attempting to access. RBAC specifically grants rights depending on each user's function within an organization, whereas ABAC concentrates on attributes like location and time of day.

Data Protection & Encryption

Use Encryption for External and Internal Microservices

Encrypting microservices' data and inter-service communication can be done in several ways. The most popular method combines symmetric key cryptography with public key infrastructure (PKI). If encryption is not used properly, it might increase microservices' latency and complexity in troubleshooting. By utilizing standard encryption algorithms and mutual transport layer security (mTLS) between services, API data security can be maintained.

API tokenization and data masking

Secrets are frequently used as attack targets to gain unauthorized access to user accounts, project repositories, or system data, and they expose sensitive information when included in source code and configuration manifests. Attackers can also reverse engineer closed-source binary files to obtain secrets, however this is typically relevant in open-source software where the code is publicly available.

Threat Mitigation & Response

Install API Gateways and Enforce Security Policies

An intermediary between APIs and their users is an API gateway. Their main purpose is to secure, manage, and authenticate API calls. Request routing, rate limitation, caching, and even load balancing are all made possible via API gateways. Using a good API gateway is crucial for businesses that manage a lot of APIs. In order to specify how an API functions, security policies are essential. The framework establishes precise rules for responding to requests, implementing security audits, and specifying the many tiers of access controls. Sensitive information is shielded from tampering and unwanted access by an efficient security policy.

Implement Rate Limiting, Traffic Filtering, and Anomaly Detection

API may identify and stop unusually high traffic levels going to your application by using rate limitation for Web Application Firewall on Application Gateway. API can prevent several kinds of denial-of-service attacks, guard against clients that have inadvertently been configured to send a lot of requests in a short amount of time, or manage traffic rates to your website from particular regions by implementing rate restriction on Application Gateway WAF_v2. confirming that user-supplied data is secure and devoid of dangerous code by validating and cleaning API inputs to stop injection threats. Filtering API outputs also helps keep sensitive data from accidentally being made public.

API Logging and Incident Response Mechanisms

By putting logging in place, you can keep track of who has used your API, what they have done, and when. This helps you understand usage trends and spot any vulnerabilities in addition to assisting with incident investigation and tracing. API logging systems should record a variety of data, including as request and response metadata, user identities, IP addresses, timestamps, and error messages, in order to guarantee strong API security. Forensic analysis and event response may benefit greatly from this information.

Compliance & Regulatory Considerations

Businesses all across the world are juggling complex legal concerns to protect people's privacy and preserve confidence. The Digital Personal Data Protection (DPDP) Act, the General Data Protection Regulation (GDPR), and the Health Insurance Portability and Accountability Act (HIPAA) are three important pieces of law that stand out in this respect. Businesses must comprehend variations in scope and application in order to decide which regulations, depending on their region, industry, and data-driven operations, they should follow.

5

Methods of API Security Testing

A SHADOW API OPERATES OUTSIDE OF THE ESTABLISHED IT GOVERNANCE, SECURITY PROCEDURES, AND PEER REVIEW SYSTEMS OF YOUR COMPANY.

API Security Testing is a crucial component of ensuring the robustness and integrity of APIs. Organizations can proactively find and fix security flaws in their APIs by doing thorough testing. It assesses rate limitation, error handling procedures, input validation methods, authentication and authorization systems, and other crucial elements. API Security Testing helps businesses strengthen their APIs and protect sensitive data and resources by utilizing cutting-edge technologies, techniques, and industry best practices.

Security Testing Techniques

- **Parameter Tampering Test -** Ensures that parameters are validated and sense-checked by the API before being processed.
- **Command Injection -** Ensures that the API blocks requests that attempt to alter back-end databases or run OS commands on the server without disclosing any private data.
- **API Input Fuzzing -** A software testing method called "API input fuzzing" is submitting unexpected data to an API in order to identify errors and vulnerabilities.
- **Unhandled HTTP Methods -** An API server's failure to appropriately handle or respond to unexpected HTTP methods such as PUT, GET, POST, and DELETE ,which is sent in a request is known as unhandled HTTP methods in API security.

API Lifecycle Security Management

In the connected digital world of today, API administration is essential to guaranteeing scalability, security, and consistency. Strong governance frameworks are more important than ever as companies depend more and more on APIs to spur innovation. According to a Vanson Bourne survey from 2021, 93% of businesses acknowledge that APIs are critical to their operations, and 97% stress the importance of strategically integrating APIs and microservices to increase productivity and customer satisfaction. To preserve efficiency and security, however, the increasing complexity of API portfolios necessitates innovative approaches.

6

Conclusion & Recommendations

```
nent as ArrowIcon } from "./assets/icons/arrow.svg";
onent as BoltIcon } from "./assets/icons/bolt.svg";
ponent as RightArrowIcon } from "./assets/icons/right-arrow.svg";

eState, useEffect, useRef } from "react";
ition } from "react-transition-group";
```

IT EMPLOYS AUTOMATIC PERCEPTION TO PREVENT HAZARDOUS SITUATIONS IN ADVANCE AND ENHANCES FORECAST ACCURACY THROUGH SELF-LEARNING CHANGES WHEN CONFRONTED WITH ENORMOUS NETWORK ATTACKS.

Network security requires the evaluation of security posture, and security decision-making heavily relies on API security posture. It employs automatic perception to prevent hazardous situations in advance and enhances forecast accuracy through self-learning changes when confronted with enormous network attacks. The number of APIs continues to grow, and API security research needs to continue to keep pace with the times and innovate various industries, enriching data and business functionality through value sharing and integration. In the process of opening to the outside world and integration in the form of API, unified security life-cycle management and fine-grained security control can help the organization

to achieve effective governance inside and outside. API security needs to be monitored throughout the entire process.

Access control for the service grid provides benefits by improving the authorization and authentication procedure. Accurately identifying every facet of API information, creating an API asset attribute library using deep learning and other techniques, automatically sorting API asset images, and supplying rich analytical data for API asset security analysis are all requirements for API security management. Through active scanning, passive traffic monitoring, and other means, the machine learning-based intelligent model enhances automation and resolves the issue of automated association between new API and current assets. Fine-grained security control, unified security life-cycle management, and API integration can all assist the company in achieving both internal and external governance. Hence it is imperative to monitor API security at every stage of the procedure.

References

1. https://escape.tech/resources/report-the-api-threat-landscape?ref=escape.tech
2. https://link.springer.com/chapter/10.1007/978-981-16-9229-1_11
3. https://learn-cloudsecurity.cisco.com/umbrella-library/cyber-threat-trends-report
4. https://www.practical-devsecops.com/api-security-management/
5. https://www.f5.com/go/ebook/forrester-report-the-eight-components-of-api-security/
6. https://www.imperva.com/resources/resource-library/reports/the-state-of-api-security-in-2024/
7. https://brightsec.com/blog/api-security/
8. https://www.csk.gov.in/documents/CIWP-2023-0001.pdf
9. https://www.cert-in.org.in/PDF/CIWP-2023-0001.pdf
10. https://www.ericsson.com/en/blog/2023/12/apis-and-network-security-three-key-lessons-from-enterprise

National Centre of Excellence
CYBERSECURITY TECHNOLOGY AND ENTREPRENEURSHIP

The National Centre of Excellence (NCoE) for Cybersecurity Technology Development has been conceptualized by the Ministry of Electronics & Information Technology (MeitY), Government of India, in collaboration with the Data Security Council of India (DSCI). Its primary objective is to catalyze and accelerate cybersecurity technology development and entrepreneurship within the country. NCoE plays a crucial role in scaling and advancing the cybersecurity ecosystem, with a focus on critical and emerging areas of security.

Equipped with state-of-the-art facilities, including advanced lab infrastructure and test beds, NCoE enables research, technology development, and solution validation for adoption across government and industrial sectors. By adopting a concerted strategy, NCoE aims to translate innovations and research into market-ready, deployable solutions—contributing to the evolution of an integrated technology stack comprising cutting-edge, homegrown security products and solutions.

DSCI
PROMOTING DATA PROTECTION
A nasscom Initiative

Data Security Council of India (DSCI) is a premier industry body on data protection in India, setup by nasscom, committed to making the cyberspace safe, secure and trusted by establishing best practices, standards and initiatives in cybersecurity and privacy. DSCI brings together governments and their agencies, industry sectors including ITBPM, BFSI, telecom, industry associations, data protection authorities and think-tanks for policy advocacy, thought leadership, capacity building and outreach initiatives. For more info, please visit www.dsci.in

DATA SECURITY COUNCIL OF INDIA

+91-120-4990253 | ncoe@dsci.in

https://www.n-coe.in/

4 Floor, NASSCOM Campus, Plot No. 7-10, Sector 126, Noida, UP -201303

Follow us on

@CoeNational

nationalcoe

nationalcoe

NationalCoE