A Real-Time Approach to Detecting API Abuses Based on Behavioral Patterns

Sameeraa Prinakaa
Dept. of Computer Science Engineering
PES University
Bengaluru, India
sprinakaaa@gmail.com

Bavanika V
Dept. of Computer Science Engineering
PES University
Bengaluru, India
vbavanika028@gmail.com

Sanjana S
Dept. of Computer Science Engineering
PES University
Bengaluru, India
shetty.sanjana476@gmail.com

Sneha Srinivasan
Dept. of Computer Science Engineering
PES University
Bengaluru, India
snehapesu@gmail.com

Sarasvathi V
Dept. of Computer Science Engineering
PES University
Bengaluru, India
sarsvathiv@pes.edu

2024 8th International Conference on Cryptography, Security and Privacy (CSP) | 979-8-3503-8632-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/CSP62567.2024.00012

***Abstract*—Over the past two decades, the internet has undergone a remarkable expansion, giving rise to the development of web-based software solutions. This evolution has led to widespread web Application Programming Interface (API) usage, facilitating remote software communication. However, the increased frequency of API calls has unfortunately resulted in a corresponding surge in API misuse instances. Monitoring API traffic becomes imperative to ensure data security, service availability, and system integrity. This paper introduces an API abuse detection system within a Log Storage Application utilizing Spring Boot with OAuth2.0 and JSON Web Token (JWT) based authentication. The proposed system effectively safeguards against critical security threats, such as Economical Denial of Service (EDoS) attacks (OWASP top 4), Unauthorized Access attempts (OWASP top 3), Brute Force Broken User Authentication, and Behavioral API abuses, with alerts promptly sent to the server. The prevalence of API usage has created numerous attack surfaces, offering a rich landscape for further research opportunities.**

***Keywords*—API abuses, JWT, OAuth2.0, EDoS, stolen token, brute force broken user authentication, OWASP, third-party service**

I. Introduction

The rapid shift toward digital transformation has led to an increased number of cyber risks and threats on a global scale. The rising popularity of APIs makes them an attractive target for hackers and malicious users [1].

A. All about API

An API comprises commands, functions, protocols, and objects, enabling communication with external systems through the execution of standard operations [2]. APIs are versatile, user-friendly, and effective. As a connection between different parts like modules, software, and developers, APIs play a crucial role in contemporary mobile apps, SaaS, and web applications [2]. Web APIs substantially cut down on software development expenses by offering diverse services [3].

B. Recent API Abuses

In the words of Tim Cook, "The advancement of technology has been a double-edged sword. While it has enhanced our lives in terms of convenience, made us more vulnerable to cyber-attacks."

According to the Salt Security 2023 report [4], the volume of API traffic has increased from 2.73 million to 26.46 million between December 2020 and June 2022. However, this rapid growth has inadvertently opened up a vast attack surface for APIs, escalating the security risk to unprecedented levels. Astonishingly, many companies remain oblivious to the pressing need for fortifying the security of their API endpoints, and often just rely on basic authentication.

Moreover, the raw statistics as mentioned in [4] highlight the urgency for increased API security, with a staggering 94% of companies reporting security issues in their production APIs over the past year. The most common issues include vulnerabilities (41%) and authentication problems (40%). Even more concerning is the fact that 31% have faced attacks with sensitive data exposure or privacy incidents, while 17% experienced a full-blown security breach [4]. The toll of these incidents isn't merely financial; it also affects the company's reputation.

In 2018 and 2022, Google Fi Service encountered significant breaches characterized by API vulnerabilities, resulting in unauthorized access and the compromise of sensitive data for millions of users [8]. Facebook, in 2020, faced a breach of its Photo API, exposing the private information of 6.8 million users due to a Facebook login bug. This incident resulted from a bug in the login feature, enabling third-party services to access shared photos [9]. T-Mobile, a major corporation, faced security challenges in 2018, 2022, and 2023, linked to vulnerable APIs. These breaches led to substantial costs of $350

979-8-3503-8632-5/24/$31.00 ©2024 IEEE
DOI 10.1109/CSP62567.2024.00012

million and compromised the data of 37 million users [10]. Similarly, in 2022, Twitter faced a security incident where the private data of 5.4 million users was compromised. This breach resulted from a vulnerability in their API, specifically exploited through a Broken Object Level Authorization [11]. These incidents emphasize the need for stronger mechanisms to ensure the security of APIs.

C. API Security

As technology continues to evolve, so too must our efforts to secure it. API security is necessary to safeguard confidential data and prevent unauthorized access. For this, several methods are used, including JWT, OAuth 2.0 and identity-based access control. OAuth 2.0 solves the issue of allowing access, using an open standard token-based protocol that makes secure authorization easier between different systems. Exchanging a temporary token with a predefined expiry enables third-party apps to access a user's resources without disclosing credentials [12]. JSON Web Tokens (JWT) works together with OAuth 2.0 to provide a framework for data transfer in JSON format between parties involved. Base64-encoded JSON objects make up JWTs, which are mostly used for stateless authentication and information transmission. They provide an effective and safe way for entities to exchange information [12].

API cyber-attacks work stealthily, and the uniqueness of API access patterns adds to the challenge of detecting them through traditional methods [1]. Existing detection methods primarily focus on network traffic or system calls, lacking efficiency in detecting anomalies at the application layer. Conventional approaches, as mentioned in [5] such as Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) rely on rule-based and signature-based approaches, but fall short in identifying subtle attacks within API traffic. This highlights the urgent need for detecting malicious behaviors specific to API traffic, emphasizing the distinct features and behaviors inherent in API communication [5].

Hence, the primary contributions are summarized as follows:

- Implementation of API Abuse Detector which is capable of detecting attacks such as EDoS, Stolen Tokens, and Unauthorized Access Attempts through behavioral analysis.
- Compilation of a comprehensive list of API endpoints and their schemas, followed by behavioral analysis aligned with the MITRE ATT&CK framework for detecting API attacks.
- A simple Log File Storage application is built using Spring Boot framework with basic functionalities upon which the API Abuse Detector has been tested.
- This detector monitors all endpoints for security breaches and alerts the server when attacks are detected.
- Basic authorization and authentication are implemented using JWT. Their tokens and expiry are monitored.

The rest of this paper is organized as follows: Section II reviews existing research efforts towards detecting API abuses in various frameworks. Section III describes the System Framework, highlighting application features and services. Section IV describes various API abuses. Section V embarks on the Detection Mechanism followed by concluding remarks and references.

II. Related Work

Addressing the upsurge of API-specific attacks, this literature survey covers a range of topics related to API security, including anomaly detection, traffic analysis, misuse detection, vulnerability assessment, and defense mechanisms. Hence, a concise overview of all referenced papers is provided below.

The authors in [13] have used a Support Vector Machine (SVM) model with a linear kernel to classify attacks. Certain features such as token size, bandwidth consumption, HTTP response codes and number of consecutive requests [13] were chosen to classify the API traffic. The results obtained: F1-score of 0.964 with a 7.3 % false positive rate [13]. Due to the limited availability of API-specific datasets, the authors created a synthetic dataset inspired by real-world API data. They utilized a Gaussian distribution outlier detection technique to generate a labeled dataset for training purposes [13]. The authors in [5] emphasized the need for implementing an IDS specific to the application architecture. They proposed API Traffic Anomaly Detection (API-TAD) [5], designed to detect anomalies in API traffic at both a general level and an application-specific level [5].

In the first large-scale empirical study on API misuses, Yang Jingbo and others leveraged GumTree, extracting one million bug-fixing operations from 528,546 historical commits on GitHub (2011 to 2018) [14]. Their findings reveal 51.7% as API misuses, with a user study confirming 7,541 potential misuses in Apache projects, of which 57 have been confirmed and fixed [14]. In [15], authors used a standard Transformer model and a target-combination Transformer model to acquire API usage information from a named API call sequence extracted from the program code [15]. The authors claim that better convergence and performance are observed in the models based on the Transformer. Another innovative approach involves constructing a fine-grained API-constraint knowledge graph [16] to detect API misuses directly against API caveats, achieving high accuracy compared to existing pattern-based detectors [16]. A Vulnerable Academic Information System (VAIS) [17] was created utilizing OWASP API Security Risks [17] and implemented with a containerization deployment strategy and gamification technique [17]. This system offers a practical learning environment for API security [17].

Shifting towards API-related challenges, a framework called RestMule is introduced to handle service policies, such as request rate limits and multi-page responses, generating resilient clients for intensive data-fetching scenarios [18]. The authors in [19] emphasize the need for the automatic

performance evaluation of API Gateways in microservices architectures. They introduce AGE as a tool for deploying multiple API Gateway scenarios and providing a comparative performance indicator for a defined workload and infrastructure [19]. In the context of vulnerability assessment, the proposed method in [20] automatically acquires the WebAPI reference and iteratively examines WebAPI requests and responses to evaluate vulnerabilities to assess vulnerabilities, contributing to the prevention of API security risks outlined by OWASP [20].

In [21], CSRFSolver is introduced as an API-level framework aimed at mitigating CSRF vulnerabilities. It encompasses a CSRF detector and CSRF defender [21], with performance assessed against Cisco Webex public URL APIs, demonstrating its superiority over existing methods [21].

The authors in [22] examine the Spring Security Framework and OAuth2 for securing microservice APIs, conducting a Proof of Concept (POC) for an Inventory Management System [22]. Their study demonstrates the framework's efficacy through security tests, including unit testing and manual techniques. Lastly, a study on future API security emphasizes robust API management and security programs, stressing the integration of security standards into API management [23]. This paper extends this discourse by proposing a detection mechanism for API abuses.

III. System Framework

The application utilized for the detection of API abuses is a Log Storage System. It enables registered users or organizations to store, share, and analyze their log files and has been developed with the Spring Boot framework and Postgres database. The JWT token-based authentication with OAuth 2.0 helps in authorization. The API endpoints and their functionality are as below:

1) *File management service*
   - ***Endpoints:***
     - ***POST /api/files/upload:*** Uploads a file to storage.
     - ***GET /api/files/{fileId}:*** Retrieves details of a specific file.
     - ***GET /api/files/list:*** Retrieves a list of files for a user.
     - ***DELETE /api/files/{fileId}:*** Deletes a specific file.

2) *User account service*
   - ***Endpoints:***
     - ***POST /api/users/register:*** Creates an account for a new user.
     - ***POST /api/users/login:*** Authenticates a user and generates an access token.
     - ***GET /api/users/profile:*** Retrieves the user's profile information.
     - ***PUT /api/users/profile:*** Updates the user's profile information.

3) *Sharing service*
   - ***Endpoints:***
     - ***POST /api/sharing/share:*** Shares a file with another user.
     - ***GET /api/sharing/{fileId}/shared-users:*** Retrieves a list of users with whom a file is shared.
     - ***DELETE/api/sharing/{fileId}/unshare/{userId}:*** Revokes sharing access for a specific user.

4) *Log analysis service*
   - ***Endpoints:***
     - ***POST /api/analysis/file:*** Shares a file to be analyzed.

For log analysis and features such as password reset, third-party APIs have been integrated to simplify the process. When users access the relevant API endpoints, the system internally engages with these third-party APIs, operating on a pay-per-use model. This can be identified as a potential vulnerability in the system.

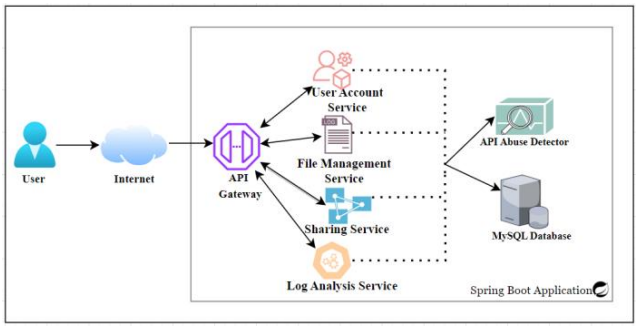

Fig. 1. System Architecture

IV. API Abuse

A. Economical Denial of Service (OWASP API4: Unrestricted Resource Consumption) [7]

- **Scenario 1:** When an attacker continuously triggers the reset password function [6], it results in excessive requests to the third-party API service. The substantial volume of spam can potentially lead to an EDoS.
- **Scenario 2:** If a suspicious user repeatedly conducts log analysis on the same log file and surpasses a predefined threshold, performing multiple analyses beyond this threshold could cause EDoS. This is because the third-party API is essential for each analysis, and the cumulative effect of excessive requests may result in economic disruptions.

B. Unauthorized Access Attempts

- **Scenario 1: Stolen Token -** In JWT authentication, the reception of an expired token could indicate a potential token theft attempt. Therefore, our system, upon detecting an expired token, transitions into a suspect mode. In this mode, the system conducts a thorough investigation into

the client's request and behavior to enhance security measures.

- **Scenario 2: Brute Force Attack -** Consistent requests to API endpoints from a client using various incorrect tokens may indicate a potential brute force attack.
- **Scenario 3: Whitelisted IP Address Manipulation (API5: Broken Function Level Authorization) [7] -** In organizational settings where IP whitelisting is implemented to secure API endpoints, API abuse occurs when a malicious user who is an insider can attempt to manipulate their IP addresses.

C. Brute Force Broken User Authentication

API abuse occurs when attackers systematically attempt to infiltrate user accounts by making repetitive and rapid login attempts, employing various credentials to compromise the system's security.

V. Detection Mechanisms

The API Abuse Detector embedded into the Log Storage system starts by discovering and profiling all the API endpoints gathering information on Methods Type (MT), interaction with User Input (UI), and Sensitivity of Data (SD). Using equation (1), each endpoint is then assigned a Vulnerability Score (VS), where w1, w2, and w3 are the weights associated to each parameter.

$$VS = (SD \times w_1) + (UI \times w_2) + (MT \times w_3) \quad (1)$$

During the detection of the attack, the detector utilizes VS to alert the severity of the data breach. A high emergency alert indicates the presence of highly sensitive data, such as addresses or passwords, while a medium alert signifies moderately sensitive and a low-level alert for minimal risk of sensitive data leakage.

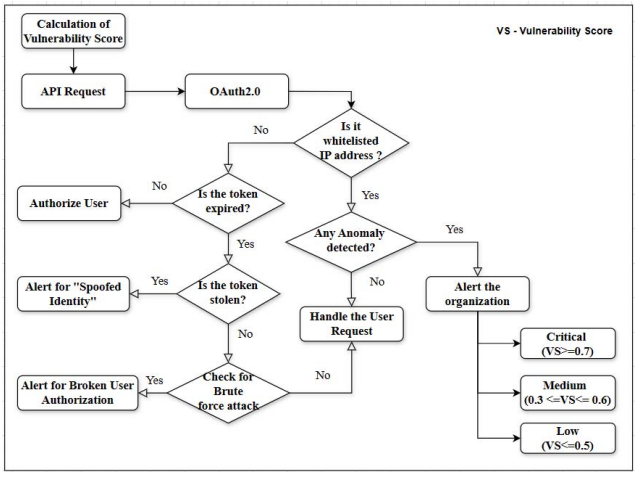

Fig. 2. Detection of Unauthorized Access Attempts

Subsequently, the detector monitors all API endpoints that invoke third-party APIs. An established threshold for enumeration acts as a trigger, placing the system in a suspect mode for deeper scrutiny once surpassed. Upon receiving an API request, the initial classification determines whether it originates from an approved whitelist of APIs. If affirmative, the system cross-references it with stored instances of malicious or abnormal behavior, transitioning to suspect mode if any are identified. For requests outside the whitelisted IPs, checks for token expiration and brute force authorization precede approval or diversion to suspect mode.

Algorithm 1 Detection of EDoS

```
Input: api, userDetails[2], θ, counter[user], reqCounter
Output: Detection Result
if api ≠ 3rd-party API then
        // Continue processing for non-3rd party API
        HandleRequest(api)
end if
else
        // Case 1: Monitoring requests from specific users
        counter[user] += 1
        if counter[user] > θ then
            return "Economical DoS Alert!"
        end if
        // Case 2: Request frequency check with time 't'
        else
            reqCounter += 1
            if elapsed_time >= t then
                reqCounter = 0
            end if
            if reqCounter > θ:
                return "Economical DoS Alert!"
            end if
        end else
        ContinueProcessing(api)
end else
```

As depicted in Algorithm 1, the process begins by evaluating incoming API requests. The algorithm examines whether the request in turn calls for a third-party API service. If not, the system proceeds to handle the request. On the other hand, for requests involving third-party APIs, the algorithm checks whether the total number of calls exceeds a predetermined threshold (θ). If this threshold (θ) is surpassed, an alert is triggered, indicating a potential EDoS attack. This approach ensures the efficient handling of regular requests while actively identifying and alerting against abnormal traffic patterns indicative of EDoS threats. The time complexity of Algorithm 1 is primarily determined by the constant-time operations involved in checking conditions and performing arithmetic operations. Each step, including the comparison of API status, incrementing counters, and elapsed time checks, requires constant time. Therefore, Algorithm 1 exhibits a time complexity of O(1), ensuring efficient execution regardless of input size. Similarly, the space complexity remains at O(1), given that it solely relies on counters like 'counter[user]' and 'reqCounter', alongside other input parameters, without utilizing any additional storage space.

Algorithm 2 Threshold setter based on the Service

Input: *api*
Output: *Threshold value* ($\theta > \alpha > \beta$)
if *api is reset password* **then**
    return α
else if *api is analyze log* **then**
    return β
else
    return θ

Algorithm 2, evaluates the provided API and service inputs to determine an appropriate threshold value. If the requested service is "reset password," the algorithm sets the threshold to α. In the case of the "analyze log" service, the threshold is established at β. For any other service, a default threshold of θ is applied. This algorithm is designed to dynamically tailor the threshold based on the specific needs of different services within the system.

VI. Conclusion and Future Work

This paper proposes an API Abuse Detector, which does behavioral analysis to effectively identify various attacks such as EDoS, Brute Force Broken User Authentication, and Unauthorized access attempts. Timely alerts are notified via mail. The working of the proposed system has been rigorously tested on the Log Storage System which is built using Spring Boot framework and MySQL database.

The current system makes use of VS, which is calculated during the initialization phase and remains static throughout the detection process. This only notifies about the severity of the data breach but can be extended to being dynamic, using a heuristic approach where even the severity of the attack can be alerted.

The proposed approach is capable of detecting only the top 3 and top 4 OWASP attacks, but there's potential for expansion to cover zero-day attacks and additional OWASP vulnerabilities. Additionally, it can be extended to function as IPS.

Other future enhancements can include:

- **Utilization of ML techniques:** The current approach solely relies on behavioral analysis and detecting attacks, restricting its ability to make intelligent decisions. Therefore, integrating ML models for the detection and categorization of attacks would address this limitation and enable more intelligent decision-making.
- **Integration of Dashboards**: Integrate dashboards into the system to provide a visual representation of API traffic and potential attacks, enhancing overall situational awareness.

References

[1] Arshardh Ifthikar et al, "A Novel Anomaly Detection Approach to Secure APIs from Cyberattacks", 2021.

[2] Ronghua Sun, Qianxun Wang and Liang Guo, " Research Towards Key Issues of API Security," Cyber Security, Springer Nature Singapore, 2022, pp. 179-192.

[3] X. Wang, J. Liu, X. Liu, X. Cui and H. Wu, "A Novel Dual-Graph Convolutional Network based Web Service Classification Framework," IEEE International Conference on Web Services (ICWS), Beijing, China, 2020, pp. 281-288.

[4] API Security in 2023 [online]. https://salt.security/api-security-trends

[5] M. Sowmya et al., "API Traffic Anomaly Detection in Microservice Architecture," IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW), Bangalore, India, 2023, pp. 206-213.

[6] Ben Solomon and Thomas Vissers, Announcing API Abuse Detection [online]. Available: https://blog.cloudflare.com/api-abuse-detection/

[7] OWASP Top 10 API Security Risks 2023 [online]. Available: https://owasp.org/API-Security/editions/2023/en/0x11-t10/

[8] Google Fi suffers data breach, customer info compromised [online]. Available: https://telecom.economictimes.indiatimes.com/news/google-fi-suffers-data -breach-customer-info-compromised/97517188

[9] Notifying our Developer Ecosystem about a Photo API Bug [online]. https://developers.facebook.com/blog/post/2018/12/14/notifying-our-developer-ecosystem-about-a-photo-api-bug/

[10] T-Mobile hacked to steal data of 37 million accounts in API data breach [online]. Available: https://www.bleepingcomputer.com/news/security/t-mobile-hacked-to-steal-data-of-37-million-accounts-in-api-data-breach/

[11] Twitter Data Breaches: Full Timeline Through 2023 [online]. Available: https://firewalltimes.com/twitter-data-breach-timeline/

[12] Prabath Siriwardena, "Advanced API Security OAuth 2.0 and Beyond", 2nd Edition, 2020.

[13] G. Baye, F. Hussain, A. Oracevic, R. Hussain and S. M. Ahsan Kazmi, "API Security in Large Enterprises: Leveraging Machine Learning for Anomaly Detection," International Symposium on Networks, Computers and Communications (ISNCC), Dubai, UAE, 2021, pp.1-6.

[14] X. Li, J. Jiang, S. Benton, Y. Xiong and L. Zhang, "A Large-scale Study on API Misuses in the Wild," 14th IEEE Conference on Software Testing, Verification and Validation (ICST), Porto de Galinhas, Brazil, 2021, pp. 241-252.

[15] J. Yang, J. Ren and W. Wu, "API Misuse Detection Method Based on Transformer," IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS), China, 2022, pp. 958-969.

[16] X. Ren et al., "API-Misuse Detection Driven by Fine-Grained API-Constraint Knowledge Graph," 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, VIC, Australia, 2020, pp. 461-472.

[17] M. Idris, I. Syarif and I. Winarno, "Development of Vulnerable Web Application Based on OWASP API Security Risks." International Electronics Symposium (IES), Surabaya, Indonesia, 2021, pp. 190-194.

[18] B. A. Sanchez, K. Barmpis, P. Neubauer, R. F. Paige and D. S. Kolovos, "RestMule: Enabling Resilient Clients for Remote APIs," IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), Gothenburg, Sweden, 2018, pp. 537-541.

[19] Moreira, A. Ribeiro and J. Silva, "AGE: Automatic Performance Evaluation of API Gateways," IEEE Symposium on Computers and Communications (ISCC), Gammarth, Tunisia, 2023, pp. 405-410.

[20] T. Taya et al., "An Automated Vulnerability Assessment Approach for WebAPI that Considers Requests and Responses," IEEE 24th International Conference on Advanced Communication Technology (ICACT), PyeongChang KwangwoonDo, Korea, Republic of, 2022, pp. 423-430.

[21] S. Wang, C. Ni, J. Wang and C. Nie, "Detecting and Defending CSRF at API-Level," IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Charlotte, USA, 2022, pp. 75- 80.

[22] Quy Nguyen and Oras Baker, "Applying Spring Security Framework and OAuth2 To Protect Microservice Architecture API," Journal of Software, 2012, pp. 257-264.

[23] Munsch, Alison and Peter Munsch. "The Future of API (Application Programming Interface) Security: The Adoption of APIs for Digital Communications and the Implications for Cyber Security Vulnerabilities." Journal of International Technology and Information Management, 2021.