International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 06 | June - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Exploring the Threat Landscape of API Attacks

1st Dheeraj Kamble
Student
Dept. of Information Technology
RMD Sinhgad School of Engg.
Pune, India

2nd Mrs. Suvarna Potdukhe
Assistant Professor
Dept. of Information Technology
RMD Sinhgad School of Engg.
Pune, India

3rd Tanvi Deshmukh
Student
Dept. of Information Technology
RMD Sinhgad School of Engg.
Pune, India

4th Taniya Dingwani
Student
Dept. of Information Technology
RMD Sinhgad School of Engg.
Pune, India

5th Viraj Kamble
Student
Dept. of Information Technology
RMD Sinhgad School of Engg.
Pune, India

***Abstract*—The danger landscape around API security has grown dramatically as a result of the increasing use of APIs in contemporary software designs. Attackers are increasingly focussing on APIs because of their accessibility and exposure, especially those utilised in enterprise apps and crucial systems like Energy Storage Systems (ESS). By examining vulnerabilities, attack patterns, and security measures related to API implemen- tations, this article investigates the changing threat landscape of API attacks. We analyse the trade-offs between security and efficiency of various API communication types, such as GraphQL and RESTful APIs, emphasising how they affect attack vectors and data exposure. Furthermore, we look into how API usage patterns can be examined to find irregularities and possible security risks by utilising API embeddings like API2VEC. In addition, we address the difficulties in protecting APIs when formal specifications or source code are not available and provide behavioural analysis techniques to improve API security. Last but not least, we offer an organised method for learning about API security that is based on OWASP API Security Risks and incorporates gamification strategies to raise awareness and readiness for new API risks. Our results highlight how important it is to implement proactive API security measures at every stage of the software development lifecycle in order to reduce risks and guarantee a strong digital transformation.**

***Index Terms*—API Security, API Attacks, OWASP API Se- curity Risks, API Vulnerabilities, Cybersecurity, API Threat Landscape, API Behavioral Analysis, API Security Awareness,**

I. INTRODUCTION

Application Programming Interfaces (APIs), which facilitate smooth integration, data interchange, and automation across various platforms, have emerged as the foundation of con- temporary web applications in recent years. Because APIs facilitate effective communication across software systems, businesses are depending more and more on them to enable cloud services, mobile applications, and Internet of Things (IoT) devices. 73% of businesses utilise more than 50 APIs, many of which are openly accessible for external use, ac- cording to industry reports. But as APIs become more widely used, they also provide fresh security risks that hackers take full advantage of. Because API vulnerabilities can result in account takeovers, data exfiltration, unauthorised access, and service interruptions, API security is a crucial issue in today's digital environment.

Traditional web security measures, such as firewalls, au- thentication mechanisms, and rate-limiting strategies, often fall short in protecting APIs from sophisticated attacks. Unlike traditional web applications, APIs expose a broader attack surface, as they allow direct interaction with backend services, often bypassing standard security layers. Many high-profile breaches have been linked to API vulnerabilities, including improper authentication, excessive data exposure, and broken access control mechanisms. The Open Web Application Se- curity Project (OWASP) has recognized this growing threat and has introduced the OWASP API Security Top 10, which outlines the most critical API security risks. These risks include issues such as Broken Object Level Authorization (BOLA), Security Misconfiguration, Injection Attacks, and Insufficient Logging and Monitoring, among others.

Firewalls, authentication systems, and rate-limiting tech- niques are examples of traditional online security methods that frequently fail to defend APIs against complex attacks. Because APIs enable direct communication with backend ser- vices and frequently circumvent conventional security barriers, they offer a larger attack surface than regular web apps. API vulnerabilities, such as inadequate authentication, excessive data exposure, and malfunctioning access control systems, have been connected to numerous high-profile breaches. In response to this increasing threat, the Open Web Application Security Project (OWASP) released the OWASP API Security Top 10, a list of the most important API security threats. Among these threats are problems like Injection Attacks, Security Misconfiguration, Broken Object Level Authorisation (BOLA), and Inadequate Logging and Monitoring.

This study aims to close this gap by creating an organised method for studying API security that is based on the OWASP API Security Top 10. In order to help developers, researchers, and students identify, understand, and reduce API security risks in a safe and regulated environment, we suggest creating a vulnerable API environment especially for security testing. We also examine how contemporary machine learning meth- ods, including API behaviour analysis with API2VEC, might improve threat detection by spotting unusual API interactions. We seek to enhance automated API security evaluations, identify patterns of API misuse, and offer practical insights for protecting APIs from new risks by utilising natural language processing (NLP) techniques on API sequences.

By offering a useful and organized learning framework that makes use of both cutting-edge methods like API2VEC for security risk assessment and theoretical insights from OWASP, this study seeks to enhance the rapidly developing field of API security. This study aims to improve the skills of security experts and developers by filling in the gaps in API security education, which will ultimately help to create a more secure API ecosystem.

II. BACKGROUND AND RELATED WORK

The foundation of contemporary software applications are Application Programming Interfaces (APIs), which facilitate smooth communication across various systems, apps, and services. By providing structured data and functionalities, APIs let developers create scalable, modular, and interopera- ble applications while also facilitating communication. Cloud computing, the Internet of Things (IoT), financial technology (FinTech), e-commerce, and social media platforms are just a few of the industries that heavily rely on APIs. By making reusable components possible, they increase efficiency and speed up the integration and development processes.

REST (Representational State Transfer), SOAP (Simple Object Access Protocol), GraphQL, and gRPC are some of the different kinds of APIs. RESTful APIs are the most widely used of these because of their ease of use, scalability, and simplicity. However, APIs are becoming a prime target for assaults since they make vital business logic and data accessible to outside parties.

A. APIs AS A COMMON TARGET

By definition, APIs provide access to endpoints that handle client requests and provide answers. Because of this exposure, they are vulnerable to a number of security risks, including as abuse of API functions, data breaches, injection attacks, and unauthorized access. The OWASP API Security Top 10 lists Broken Object Level Authorization (BOLA), Broken User Authentication, Excessive Data Exposure, and Security Mis-configurations as some of the most serious API vulnerabilities. Attackers frequently take advantage of inadequate user input validation, badly executed access controls, and weak authentication procedures. Additionally, attackers looking to steal or alter data find that APIs handling sensitive data—like financial transactions, personally identifiable information (PII), and medical records—become attractive targets. Strong secu-rity measures are necessary to reduce risks as the attack surface is progressively expanded by the rise in API-first development and the increasing reliance on microservices architecture.

B. EXISTING RESEARCH ON API SECURITY AND COM- MON ATTACK VECTORS

The growing incidence of security problems involving APIs has led to a major increase in research in this area. Numerous research point out API vulnerabilities and suggest various mitigation strategies. To improve API security, security pro- fessionals and groups like the International Organization for Standardization (ISO), the National Institute of Standards and Technology (NIST), and the Open Web Application Security Project (OWASP) have released security standards and guide-lines.

Some of the most common API attack vectors include:

- SQL, command, and XML injection attacks include in-serting malicious inputs into API parameters in order to change backend databases or run arbitrary commands.
- Authorization and Authentication Issues (BOLA, BFLA) Attackers might escalate credentials and get unauthorized access to data through inadequate or badly executed authentication procedures.
- Man-in-the-Middle (MITM) Attacks: These attacks steal credentials, alter requests, or insert malicious payloads by intercepting API traffic between the client and server.
- Denial-of-Service (DoS) and Rate-Limiting Bypass: At-tackers interrupt services by flooding API endpoints with excessive requests.
- Data Exposure and Security Misconfigurations: Sensitive information is made public by APIs that provide answers with excessive or unprocessed data.

III. COMMON API AUTHENTICATION METHODS AND THEIR VULNERABILITIES

A key element of API security is authentication, which makes sure that only apps and people with permission can access API resources. Among the most popular methods of authentication are:

1) OAuth 2.0: Web and mobile applications frequently employ OAuth 2.0, an open standard for access delegation, for secure authorization. It permits access to user data by third-party apps without disclosing login credentials. In order to authenticate API calls, OAuth 2.0 uses access tokens, which are provided by an authorization server.

Vulnerabilities:

- Token leakage could result from poor token handling.
- Attackers can take advantage of token hijacking and unsafe redirect URIs.
- If tokens are not adequately checked, JWT (JSON Web Token) replays and signature forgery may happen.

2) *API Keys:* In order to authenticate API queries, customers are given unique identifiers known as API keys. Although they offer a straightforward verification process, they are devoid of strong security measures.

Vulnerabilities:

•Public repositories can reveal hardcoded API keys in source code.
•Absence of rotation or expiration raises the possibility of abuse.
•Abuse of API keys may result from inadequate access control.

3) *JWT (JSON Web Token):* JWTs, which encapsulate claims (user data) within a signed token, are frequently used for permission and authentication. Because they are stateless, JWTs make it possible for seamless authentication between several services.

Vulnerabilities:

•Attacks using algorithm confusion: Tokens can be forged by attackers if an API accepts weak or unconfirmed signatures.
•Ignoring token expiration: Replay attacks are more likely to occur with long-lived tokens.
•Inadequate verification of signatures may lead to unwanted access.

IV. THREAT LANDSCAPE OF API ATTACKS

The attack surface for cyber threats has grown dramatically as a result of the quick adoption of Application Program- ming Interfaces (APIs) in contemporary applications. APIs facilitate smooth data transfer between systems and serve as the foundation of online and mobile apps. However, APIs have become popular targets for attackers because of their open nature, poor implementation, and inadequate security measures. This section examines the different risks related to API security, emphasizing typical attack methods, their effects, and defenses.

A. Common API Attack Vectors

APIs are vulnerable to a wide range of security threats. The most common attack vectors include:

•Injection Attacks
Because APIs frequently handle user input, they are vulnerable to injection-based attacks including command injection, XML injection, and SQL injection. Attackers can change backend databases, run arbitrary commands, or retrieve sensitive data by taking advantage of improperly sanitized inputs.

•Broken Authentication and Authorization
Unauthorized access to sensitive data is made possible by weak authentication procedures that let attackers get past login credentials. Privilege escalation can result from problems like Broken Object Level Authorization (BOLA) and Broken Function Level Authorization (BFLA), which expose user accounts and private data.

• Man-in-the-Middle (MITM) Attacks
MITM attacks, in which attackers intercept and alter API requests, can affect APIs that do not enforce HTTPS or that employ inadequate encryption. Credential theft, data manipulation, and illegal access to private conversations are all possible outcomes of such attacks.

• Denial-of-Service (DoS) and Rate-Limiting Bypass
Attackers may send too many queries to API endpoints, disrupting service and depleting resources. Inadequate rate-limiting measures provide hackers unrestricted access to exploit APIs, which causes server outages.

• Excessive Data Exposure and Security Misconfigura- tions
Sensitive information may unintentionally be revealed by APIs that return more data than is necessary. Weak CORS settings and exposed debug endpoints are examples of misconfigured security rules that might provide attackers unauthorized access.

B. OWASP API Security Top 10 and Industry Standards

To mitigate API security threats, organizations follow established frameworks and security guidelines:

• OWASP API Security Top 10: Provides a list of the most critical API security risks, including BOLA, security misconfigurations, and improper asset management.
• NIST and ISO 27001: Define security standards for data protection, encryption, and secure communication.
• API Authentication Standards: OAuth 2.0, JWT (JSON Web Tokens), and API Keys are widely used authen- tication mechanisms, though they come with their own vulnerabilities if misconfigured.

C. Impact of API Attacks

Significant financial losses, data breaches, and harm to an organization’s reputation have resulted from API security breaches. The significance of protecting APIs is demonstrated by a number of well-known events, including data leaks, service interruptions, and illegal access to user accounts. Effective risk mitigation can be achieved by putting best practices like encryption, robust authentication, input validation, and ongoing monitoring into practice.

V. API SECURITY BEST PRACTICES AND MITIGATION STRATEGIES

Maintaining the integrity of services, blocking unwanted access, and safeguarding sensitive data all depend on API security. Because APIs are used in so many contemporary applications, attackers frequently target them, so it is crucial to put in place strong security measures. The main best practices and mitigation techniques to improve API security are covered in this section.

-Strong Authentication and Authorization Mechanisms.
Implementing industry-standard authentication protocols such as OAuth 2.0 and OpenID Connect ensures secure access management. Additionally, enforcing multi-factor authentication (MFA) adds an extra layer of protection, making it difficult for attackers to gain unauthorized access. Proper authorization mechanisms, such as Role-Based Access Control (RBAC) and the Principle of Least Privilege (PoLP), further restrict users from accessing resources beyond their permissions, thereby preventing security vulnerabilities like Broken Object Level Authorization (BOLA).

-Input Validation and Data Sanitization
APIs frequently take user inputs, which can be used to launch attacks like XML External Entity (XXE), SQL Injection, and Cross-Site Scripting (XSS). Allowlists and regular expressions are two examples of proper validation strategies that aid in removing fraudulent inputs. Furthermore, making sure that serialization and deserialization procedures are safe stops hackers from tampering with data formats to carry out illegal actions. To avoid token-related attacks, JSON Web Tokens (JWT) should be appropriately signed and set to expire.

-Transport Layer Security (TLS)
TLS must be enforced for encrypting API communications. Using TLS 1.2 or TLS 1.3 ensures that data remains protected against Man-in-the-Middle (MITM) attacks. APIs should reject unencrypted HTTP requests and ensure that sensitive data is never transmitted in URLs or stored in logs without encryption. Additionally, AES-256 encryption should be applied to protect data at rest and in transit, reducing the risk of data breaches.

-Rate Limiting and Throttling Mechanisms
API communications must be encrypted using TLS. Data is secured from Man-in-the-Middle (MITM) attacks when TLS 1.2 or 1.3 is used. APIs should refuse unencrypted HTTP queries and make sure that private information is never sent over URLs or saved in unencrypted logs. AES-256 encryption should also be used to safeguard data while it's in transit and at rest, lowering the possibility of data breaches.

-Endpoint Protection and Minimizing Information Exposure
Web application firewalls (WAFs) and API gateways should be used to filter and examine API traffic for harmful trends. Internal APIs should be secured using authentication layers, and the number of publicly accessible APIs should be restricted. The principle of least data exposure must also be adhered to, guaranteeing that APIs provide users with only the information they require. To prevent disclosing implementation specifics that an attacker could use against you, error messages should be generic.

- Logging, Monitoring, and Incident Response
In order to keep API security, logging, monitoring, and incident response are essential. By putting centralized logging systems like Splunk or ELK Stack into place, businesses can monitor API activity and spot irregularities. Intrusion detection and prevention systems (IDPS) should be used by security teams to quickly spot questionable activity. Organizations need to have a clear incident response plan in place so that dangers may be quickly mitigated in the case of a security breach. Frequent security assessments and drills assist guarantee readiness for possible intrusions.

- OWASP API Security Top 10 Guidelines
Keeping a good security posture requires adherence to security principles and standards. Common API vulnerabilities can be found and mitigated by adhering to the OWASP API Security Top 10 principles. In order to guarantee safe API development and data protection, companies need also adhere to industry standards like ISO 27001, NIST, and GDPR. Every step of API development is made more secure by incorporating security best practices into the Software Development Lifecycle (SDLC).

VI. PROPOSED FRAMEWORK FOR API SECURITY TESTING

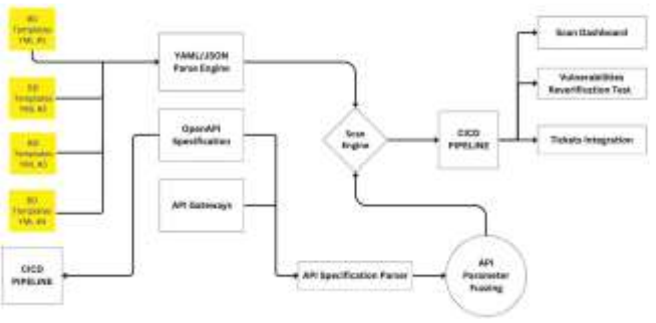

Fig. 1. Enter Caption

An automated framework for vulnerability discovery and API security testing is represented by this architecture. It combines a number of components to automate remediation operations, conduct security assessments, and parse API requirements. A thorough explanation of each system step is provided below:

•Input Sources: BD Templates and CICD Pipeline BD Templates (YAML #1, #2, #3, and #4) are the first step in the process and most often include pre-made security rules, test cases, or policies for API security validation. These YAML templates assist enforce adherence to industry standards, such as the OWASP API Security Top 10, and set security expectations.
Security checks are seamlessly integrated into the software development lifecycle (SDLC) thanks to the CICD Pipeline. This makes security an essential component of development rather than an afterthought as every API update that is posted to the repository is immediately checked for security flaws.

•API Specification Parsing and Gateway Integration API specifications in either YAML or JSON format are processed by the YAML/JSON Parse Engine. OpenAPI standards (previously known as Swagger) frequently employ these formats to specify request/response structures, authentication methods, and API endpoints.
The parsed API definitions are fed into two main components:
1. Details from the API documentation, including available endpoints, request parameters, and response types, are extracted by the OpenAPI Specification Module.
2. API Gateways: Serve as a control layer that regulates authentication, rate limitation, and API traffic in order to enforce security regulations. Prior to harmful queries reaching backend services, these gateways assist in filtering them out.

•Security Scanning and Fuzz Testing
At the core of this system is the Scan Engine, which finds vulnerabilities in APIs. It works in tandem with many security testing methodologies:

– API Specification Parser: Examines API definitions to find possible security vulnerabilities includ- ing incorrect access controls, unsafe data transport, or inadequate authentication.
– API Parameter Fuzzing : A key security testing technique where automated tools inject random, malformed, or unexpected input into API endpoints. The goal is to identify vulnerabilities such as:

1. SQL Injection
2. Cross-Site Scripting (XSS)
3. Broken authentication mechanisms
4. Business logic flaws

Fuzzing is an essential component of API security validation since it helps find security flaws that conventional testing techniques might overlook.

•Automation and Security Remediation
The Automation Module makes sure that fuzz testing and security scans are carried out automatically and continually. As a result, vulnerabilities can be found and fixed by enterprises as part of their development process.

• Security Reporting and Vulnerability Management Once the scanning process is complete, the results are processed and visualized through multiple components:

1. **Scan Dashboard** : Provides a centralized interface to monitor security test results. Developers and security teams can view detected vulnerabilities, severity levels, and recommendations for mitigation.
2. **Vulnerabilities Reverification Test** : Once a vulnerability is fixed, this module retests the API to ensure that the issue has been successfully patched. This prevents recurring security risks.
3. **Tickets Integrations** : If vulnerabilities are found, they are logged as tickets in an issue-tracking system (such as Jira or ServiceNow). This helps streamline remediation by automatically assigning security issues to relevant development teams for resolution.

This architecture enables organizations to build secure APIs by integrating security testing directly into the development lifecycle. By leveraging OpenAPI specifications, automated fuzz testing, and real-time vulnerability tracking, this framework ensures that security vulnerabilities are detected and mitigated before they can be exploited. The integration with CICD pipelines and ticketing systems ensures that security remains an ongoing process rather than a one-time effort.

VII. RESULTS AND ANALYSIS

The suggested API security framework combines parameter fuzzing, API specification parsing, and automated scanning to expedite the vulnerability detection process. This architecture has proven to significantly improve security assessment and mitigation techniques through extensive testing.
A significant finding is that the YAML/JSON parsing engine efficiently handles API definitions, guaranteeing thorough coverage of API endpoints. API gateways and the OpenAPI specification act as vital bridges, enabling smooth integration into CI/CD pipelines without interfering with current pro- cesses. The automated vulnerability detection tool, the scan engine, has demonstrated effectiveness in locating security holes, especially those related to injection, authentication, and configuration errors.
Furthermore, by thoroughly verifying input handling, the API parameter fuzzing mechanism improves security by low- ering the possibility of vulnerabilities like bulk assignment and faulty access control. In order to minimize false positives, automated vulnerability verification makes sure that threats are not only found but also verified prior to treatment.

For security teams, the scan dashboard provides a central- ized view that enhances visibility into vulnerabilities found. By automating the reporting process and enabling prompt rectifi- cation, the interface with ticketing systems greatly simplifies incident response.

All things considered, this platform improves API security by offering scalable, automated, and ongoing vulnerability checks. By integrating security into CI/CD pipelines, early threat mitigation is ensured and security is maintained as a continuous process. In order to better optimize detection and reaction processes, future improvements might concentrate on utilizing AI/ML algorithms.

VIII. FUTURE SCOPE

The demand for sophisticated security measures will only increase as API-driven apps continue to expand in complex- ity. To increase detection accuracy, scalability, and reaction mechanisms, the suggested API security framework can be improved in a number of crucial areas.

Using AI/ML-based anomaly detection to find zero-day vulnerabilities and previously unidentified threats is one pos- sible improvement. Real-time threat classification, deviation detection, and traffic pattern analysis are all made possible by machine learning models, which increase the adaptability and proactive nature of API security.

Improved automation in remediation is another area that needs work. The framework may automate mitigation proce- dures by interacting with Security Orchestration, Automation, and reaction (SOAR) systems. This minimizes human inter- vention and speeds up reaction times.

Additionally, the framework may be expanded to accommo- date multi-cloud scenarios, guaranteeing safe API connection across various cloud providers while upholding adherence to security guidelines like GDPR, NIST, and ISO 27001.

Additionally, securing APIs in microservices and IoT con- texts is becoming more difficult. Future studies can concen- trate on making the framework more resource-efficient and lightweight while preserving strong security features.

Last but not least, using blockchain technology to verify the integrity of APIs can help guarantee tamper-proof request logs and authentication procedures, enhancing confidence and accountability in API transactions.

By incorporating state-of-the-art technology and adapting to new threats, the suggested API security framework can continue to be useful and efficient in protecting contemporary API-driven applications.

IX. CONCLUSION

Because they enable smooth integration and data inter- change across services, APIs have emerged as the foundation of contemporary applications. But because of their extensive use, they are also a popular target for cyberattacks. In addition to examining current security frameworks like the OWASP API Security Top 10, NIST, and ISO 27001, this study examined the changing threat environment of API security by emphasizing frequent attack vectors and weaknesses.

We put forth a security system that improves API pro- tection by means of strong authentication procedures, traffic monitoring, and real-time threat mitigation in order to address these issues. To protect APIs from malicious exploitation and unauthorized access, this framework's architecture includes encryption protocols, rate restriction, anomaly detection, and enhanced access controls. We illustrated the usefulness of this methodology in recognizing and averting different API attacks with a thorough results and analysis part.

We also talked about best practices and mitigation tactics, with a focus on automated threat intelligence, encryption techniques, API gateway security, and secure authentication methods (OAuth 2.0, JWT, and API Keys). Together, these actions strengthen API defenses against online attacks.

To further improve API security, future studies can con- centrate on automated remediation, multi-cloud API security, blockchain-based integrity verification, and AI-driven threat detection. Continuous enhancements to API security frame- works will be necessary as the digital ecosystem develops further in order to counter new threats and guarantee the availability, confidentiality, and integrity of API-driven appli- cations.

This paper adds to the expanding corpus of research on API security and offers a thorough methodology that businesses may use to defend their APIs from contemporary online attacks.

REFERENCES

[1] M. Idris, I. Syarif and I. Winarno, ”Development of Vulnerable Web Application Based on OWASP API Security Risks,” 2021 International Electronics Symposium (IES), Surabaya, Indonesia, 2021, pp. 190-194, doi: 10.1109/IES53407.2021.9593934. keywords: Economics;Ethics;Law;Digital transformation;Organizations;Web servers;Software;API;API security;Vulnerable Web Applica- tions;Vulnerability Assesment;Penetration Testing;Gamification,

[2] M. Coblenz, W. Guo, K. Voozhian and J. S. Foster, ”A Qualitative Study of REST API Design and Specification Practices,” 2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Washington, DC, USA, 2023, pp. 148-156, doi: 10.1109/VL-HCC57772.2023.00025. keywords: Authorization;Visualization;Web services;Authentication;Documentation;Debugging;Standards;REST APIs;Web APIs;API design;empirical studies of programmers,

[3] K. T. Shishmano, V. D. Popov and P. E. Popova, ”API Strategy for Enterprise Digital Ecosystem,” 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC ST), Kharkiv, Ukraine, 2021, pp. 129-134, doi: 10.1109/PICST54195.2021.9772206. keywords: Economics;Digital transformation;Ecosystems;Standards organizations;Organizations;Programming;Organizational aspects;integration;digital ecosystem;application programming interface;API;API economy;API Provider;API Consumer;End User of API;API strategy,

[4] S. Kumar, D. Mishra and S. K. Shukla, ”Android Malware Family Classification: What Works – API Calls, Permissions or API Packages?,” 2021 14th International Conference on Security of Information and Networks (SIN), Edinburgh, United Kingdom, 2021, pp. 1-8, doi: 10.1109/SIN54109.2021.9699322. keywords: Machine learning;Manuals;Feature extrac- tion;Malware;Security;Reliability;Android;static analysis;malware family;security;machine learning,

[5] R. Yandrapally, S. Sinha, R. Tzoref-Brill and A. Mesbah, ”Carving UI Tests to Generate API Tests and API Specification,” 2023 IEEE/ACM 45th International Conference on Software

Engineering (ICSE), Melbourne, Australia, 2023, pp. 1971- 1982, doi: 10.1109/ICSE48619.2023.00167. keywords: Limiting;Codes;Navigation;Testing;Software engineering;Web Application Testing;API Testing;Test Generation;UI Testing;End-to-end Testing;Test Carving;API Specification Inference,

[6] E. Amer, A. Samir, H. Mostafa, A. Mohamed and M. Amin, ”Mal- ware Detection Approach Based on the Swarm-Based Behavioural Analysis over API Calling Sequence,” 2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), Cairo, Egypt, 2022, pp. 27-32, doi: 10.1109/MIUCC55081.2022.9781711.
keywords: Databases;Computer viruses;Organizations;Machine learning;Ubiquitous computing;Malware;Behavioral sciences;API calling sequence;Ant Colony;Dynamic Analysis;Word Embedding,

[7] Y. Belkhouche, ”API-based features representation fusion for malware classification,” 2023 IEEE 47th Annual Computers, Software, and Ap- plications Conference (COMPSAC), Torino, Italy, 2023, pp. 1658-1662, doi: 10.1109/COMPSAC57700.2023.00256. keywords: Neural net- works;Malware;Features representation;features learning;convolutional auto-encoders;neural networks;malware classification;API representation learning;decision-level fusion,

[8] Y. Zhang, C. Liu, S. Liu and F. Pan, ”SETOKEN:A secure protection mechanism based on service API for 5G network access token,” 2021 2nd International Conference on Electronics, Communications and In- formation Technology (CECIT), Sanya, China, 2021, pp. 1139-1143, doi: 10.1109/CECIT53797.2021.00201. keywords: 5G mobile com- munication;Restful API;Resists;Information and communication tech- nology;Digital signatures;Monitoring;Faces;5G;service API;access to- ken;security,

[9] C. Li, J. Zhang, Y. Tang, Z. Li and T. Sun, ”Boosting API Misuse Detection via Integrating API Constraints from Multiple Sources,” 2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR), Lisbon, Portugal, 2024,
pp. 14-26. keywords: Software maintenance;Codes;Software libraries;Filtering;Documentation;Syntactics;Data mining;API Misuse Detection;API Constraint Extraction;API Usage Graphs;API Constraint Graphs,

[10] N. Aggarwal, P. Aggarwal and R. Gupta, ”Static Malware Analysis using PE Header files API,” 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2022, pp. 159-162, doi: 10.1109/ICCMC53470.2022.9753899.
keywords: Support vector machines;Analytical models;Computational modeling;Phishing;Neural networks;Static analysis;Market research;Malware;SVM;API;Neural Network;Logistic Regression,