API Security: Threats, Best Practices, Challenges, and Way forward using AI

A Joint Whitepaper by CSIRT-Fin, CERT-In and Mastercard

August 2023

Table of Contents

Introduction 3
API Usage Landscape 3
API Threat Landscape 3
API Threat Landscape in Indian Financial sector 4
Type of API Attacks 5
Limitations of traditional methods 9
API Gateways and their Limitations: 9
WAF's and their Limitations: 10
AI & API Security 10
AI vs Rule based Security 11
How AI Based API security works: 12
Challenges in using AI Based models in API Security: 13
Current to Future 14
Conclusion 15
Appendix 16

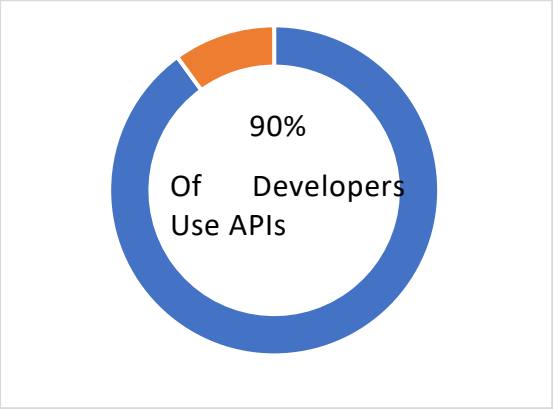
Introduction

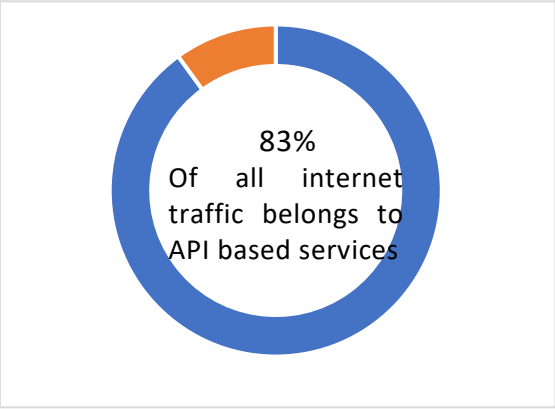
API Usage Landscape

An Application Programming Interface (API) is a data connection allowing data to be shared with other applications. They can be viewed as digital middlemen between organisations / enterprises and platforms that need to access data for driving innovation, increasing reach, discover new business models, increase partner network, etc.

APIs let your product or service communicate with other products and services without having to know how they have been implemented. This can simplify app development, integration of multiple product functionalities, saving time, money while providing a seamless user experience. While designing new tools and products—or managing existing ones—APIs provide flexibility, ease of usage, simplify design and administration.

With the speed of digital transformation, APIs are playing a central role in both mobile commerce and the internet of things (IoT). The usage of APIs has increased significantly during the past few years. Akamai estimates that roughly 83% of internet traffic is being driven by APIs. Further, according to Slashdata survey, which offers several granular insights into how developers use APIs, nearly 90% of developers are using APIs in some capacity[1]. This increasing dependence on API, in turn, is leading to API security becoming a serious concern as it not only expands the attack surface but also introduces new security risks. In addition, there could be severe consequences for consumers, businesses, and third-party providers in case of API breaches.

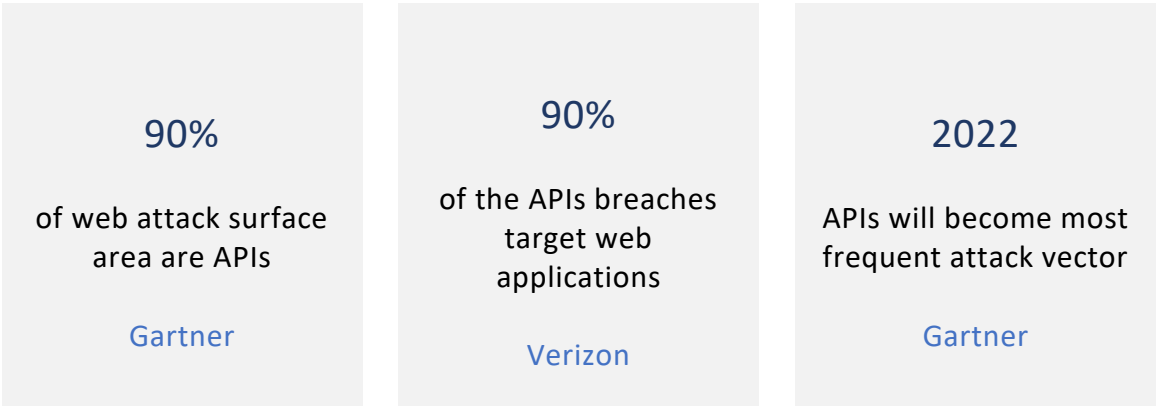




API Threat Landscape

With an exponential growth in the number of API calls, there is an aggressive increase in abuse of these APIs as well. A report published by *Salt Security* states that around 91% of the respondent organizations in their survey have experienced an API security related incident in the year 2020[1]. Other studies from Salt Security state that API attacks increased by over 600% from 2021 to 2022. *Gartner* predicts that 90% of web-enabled applications will have broader attack surfaces due to exposed API's. The latest study from *Imperva* claims that vulnerable APIs are costing organizations between $40 and $70 billion annually[2].

The shift from monolithic architectures to microservices in clouds and containers has revolutionized development cycles but at the same time increased the vulnerabilities exposed to the internet. The use of Kubernetes and other microservices is now a crucial component of APIs. Over 3,80,000 vulnerable Kubernetes API servers were recently discovered by a study, which is concerning because the Kubernetes API server is an essential control plane component for container deployment[2].



source: APIsec- Best Practices for API security[3]

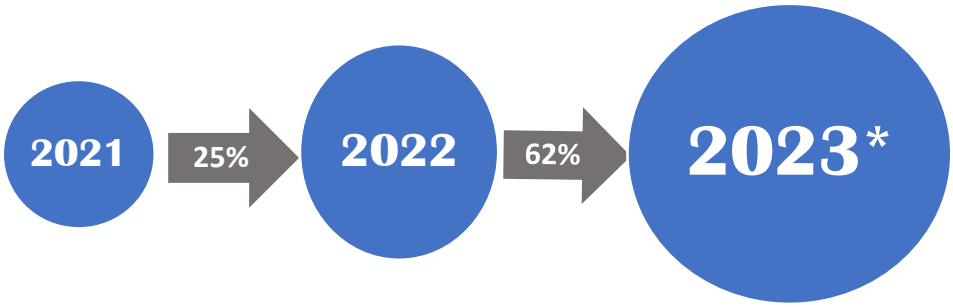
In fact, because of their direct access to extremely sensitive data and functionality, APIs are frequently cited as one of the primary security concerns that organizations face.

API Threat Landscape in Indian Financial sector

APIs are changing the landscape of financial services and playing a critical role in the rise of Fintech and Open Banking. Open Banking provides third-party financial service providers open access to consumer's banking and financial data transactions from banks, non-bank financial institutions as well as FinTech services through the use of APIs. Today, banks are in a position to provide better customer experience and develop new revenue streams by relying on banking APIs. APIs have opened doors to technologies such as P2P payments and cryptocurrency exchanges. However, with this rise of digitization and API usage in the financial sector along with the availability of sensitive customer information,

the financial sector is also becoming a preferred target for API attacks. Indian Financial Sector since 2021 has observed a consistent rise in API attacks over the last few years.

Year wise trend of API attacks in Indian Financial Sector



*Till June 30, 2023

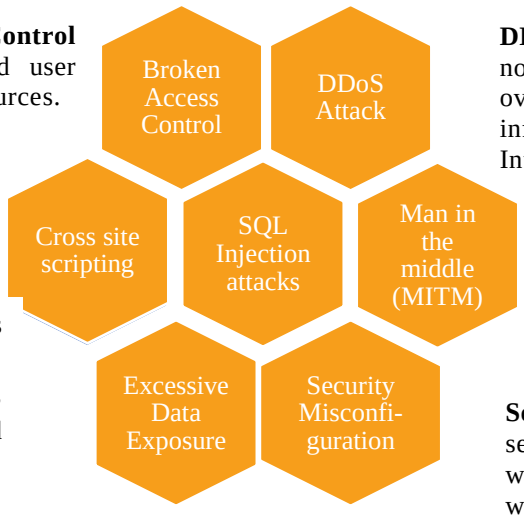
API related security incidents damage consumer trust as well as the organisations brand reputation.

Type of API Attacks

Broken Access Control allows an unauthorized user access to restricted resources.

Cross site scripting attacks are a type of injection, in which malicious scripts are injected into trusted websites

SQL Injection uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed



DDoS attack disrupts the normal traffic by overwhelming the target infrastructure with a flood of Internet traffic

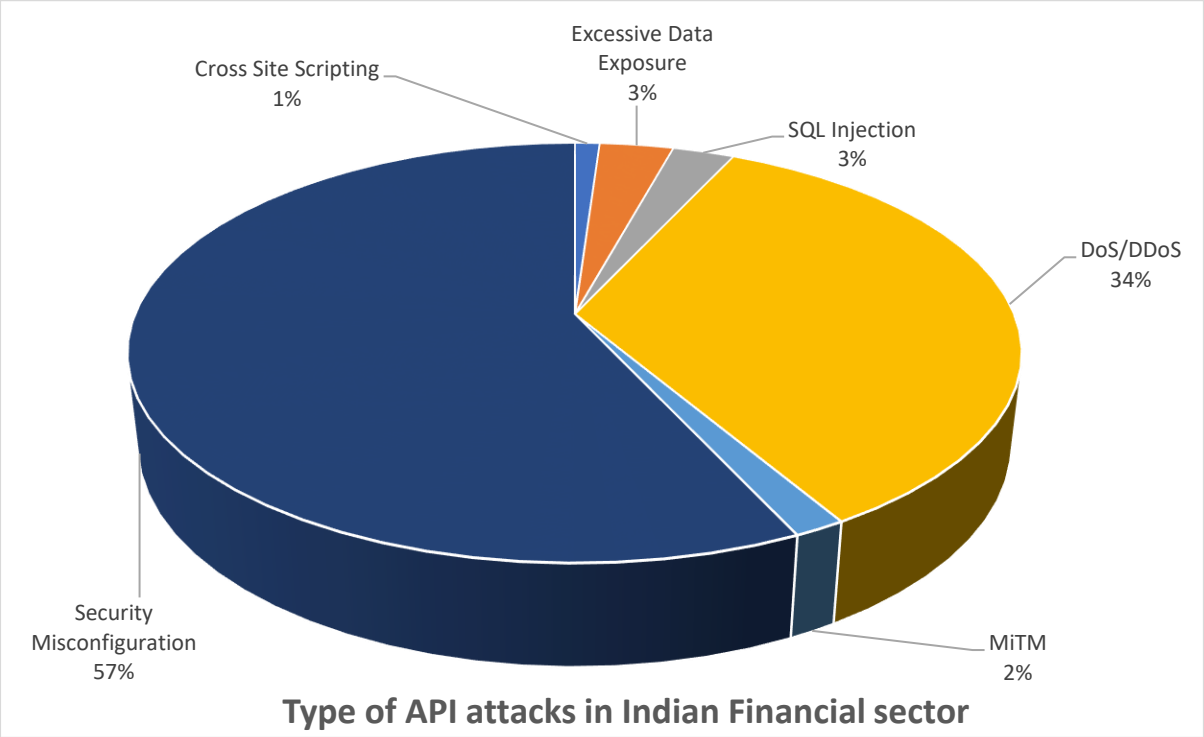
MITM is when a perpetrator is in between a user and an app -either to eavesdrop or to impersonate one of the parties.

Security misconfiguration is when security options are not defined in a way that maximizes security, or when services are deployed with insecure default settings

Excessive Data Exposure is when too much information passes from the API to the client, with the client filtering what information is displayed to the user

The above diagram highlights some of the most common API attacks. It should be noted that DDOS (Distributed Denial of Service) Attacks are intended to disrupt the normal working of the servers by overwhelming the infrastructure with a flood of incoming traffic. The Distributed part is where the attacker uses an array of sources which might be any network source capable of making requests to APIs and using them at the same time to overwhelm the infrastructure catering to the API. The devices usually carrying out the attack, themselves have been compromised allowing them to be controlled by a single attacker. DDOS attacks are very common and during one well-known incident, the system of the affected entity was hit by a volume of traffic ranging up to 1.35 terabits per second and the attack lasted for over 20 minutes which was launched by tens of thousands of unique end points orchestrated by more than a thousand different autonomous systems (ASNs).

Another common and well-known attack observed was a SQL Injection attack where the attacker injected malicious code into the system to extract or modify the database information, access sensitive data, execute admin tasks on the database, and issue commands to the underlying database operating system to steal thirty million credit card numbers. SQL Injection flaws are introduced when software developers create dynamic database queries constructed with string concatenation that includes user-supplied input. As user input is something that is a necessity for modern API and web applications, such attacks are becoming more common.



Best Practices for API security[4]

Some, and not exhaustive, of the best practices to protect against API attacks and cover some loopholes are:

1. **Authentication and Authorization**
   - Use strong authentication mechanisms such as API keys, OAuth, or JWT (JSON Web Tokens).
   - If using tokens like JWT, set appropriate expiration times and implement secure token management practices to prevent token misuse or replay attacks.
   - Implement granular access control to limit API access based on user roles and permissions.
   - Always validate user credentials and tokens before granting access to sensitive data.

2. **API Gateway and Firewall:**
   - Employ an API gateway for centralized security enforcement, monitoring, and management.
   - Implement web application firewall (WAF) to protect against common web threats.

3. **Data Protection and Secure Communication:**
   - Encrypt sensitive data using appropriate encryption algorithms and key management.
   - Apply data masking techniques to hide sensitive information in logs and responses.
   - Use secure communication protocols to prevent eavesdropping and man-in-the-middle attacks.
   - Employ secure headers and practices to prevent information leakage.

4. **Input Validation and Sanitization**
   - All user inputs should be validated and sanitized to prevent injection attacks (e.g., SQL injection, XSS) and parameter manipulation.

5. **Output Encoding**
   - Encode output to protect against HTML/JavaScript injection (XSS) and other data manipulation attacks.

6. **Rate Limiting and Throttling**
   - Implement rate limiting and throttling mechanisms to prevent abuse of the API and DDoS attacks by limiting the number of requests from a single client within a specific time frame.

7. **Error Handling and Logging**
   - Ensure proper error handling to avoid exposing sensitive information.
   - Implement comprehensive logging for monitoring and auditing purposes.

8. **CORS (Cross-Origin Resource Sharing)**
   - Configure CORS properly to restrict which domains can access the API from the client-side, thereby preventing unauthorized cross-origin requests.

9. **Secure Storage of Secrets**
   - Store API keys, credentials, and sensitive data securely using encryption and access controls.

10. **Regular Security Assessments**
    - Conduct regular security assessments of APIs such as penetration testing, security audits and code reviews to identify potential vulnerabilities and security flaws.

11. **Education and Documentation**
    - Clear documentation should be provided containing steps to use the API securely, including examples of proper authentication and authorization methods.

12. **Privacy Protection**
    - Minimize data collection and storage to only what is necessary.
    - Comply with relevant privacy regulations and obtain user consent for data processing.
    - Integrate privacy considerations from the initial stages of API development. Perform a Privacy Impact Assessment (PIA) to identify and mitigate potential privacy risks.

13. **Secure Development Lifecycle (SDLC)**
    - Integrate security considerations into the entire API development process.
    - Conduct security training for developers to raise awareness of secure coding practices.

Limitations of traditional methods

In both cases of API attacks discussed above (DDOS and SQL Injection), although there are traditional methods for the detection and blocking of these attacks, they are much more difficult to implement and not very effective at times when implemented.

In the case of usual traffic monitoring to check for DDOS attacks, it is very tough to identify and block such an attack as it is hard to distinguish the calls originating from compromised IP addresses and normal ones. In the case of SQL injection, a common prevention method is testing SQL Injection vulnerabilities while designing APIs. Yet this is not something trivial as there are huge, or potentially infinite, number of variants and semantic possibilities of SQL leading to attacks on various Web applications.

The traditional methods (WAF, API Gateways) of monitoring APIs and defending against application threats are still effective but relying on them to thwart today's sophisticated API attacks may not be sufficient.

API Gateways and their Limitations:

API gateways serve as the Central Access Management tool for organizations handling authentication and authorization of requests to access an API. Although it performs authorization and some basic security functions, it is primarily an API management and not a security tool. Although, these functionalities are crucial for safeguarding certain parts of an application, they are insufficient to defend against the sophisticated attacks (like the risks listed in the OWASP API Security Top 10).

As API gateways are part of an API management system that is dependent on the organization’s web application firewall (WAF), it may only secure “north/south” API traffic that passes through the WAF. However, API gateways cannot secure “east/west” API traffic — the traffic that makes up communication between servers, containers, and services — that does not move through the WAF. Because of this, the organization may be exposed to attacks like man in the middle (MITM) attacks, API injections (XSS and SQLi) (malicious code is inserted into a vulnerable software program to stage - an attack [e.g., cross site scripting (XSS) and SQL injection (SQLi)]) and DDoS attacks (where an attacker tries to overwhelm a web API).

WAF's and their Limitations:

Another most common security solution used for detecting malicious requests is Web Application Firewalls (WAFs). WAFs are proxy-based tools that inspect HTTP traffic to monitor, filter, and block malicious HTTP activity. WAFs are deployed at the forefront of the application, protecting it independently by analyzing the traffic's metadata and blocking known vulnerabilities using a wide range of pre-configured policies. WAFs are general-purpose security solutions, protecting any web application in the same manner, regardless of the application's functionality and purpose. In essence, WAF is only as effective as the policies it enforces.

The task of policy configuration (which was and sometimes even today is done manually) is automated by modern WAFs (aka 'NextGen WAF') which have the advanced capability of automatic policy creation. They have also started including features like input validation and content threat detection, addressing several common forms of technical API manipulations. To enable these protections, WAFs create dedicated policies by ingesting the API schema, such as OpenAPI Specifications. While these protections can be effective, they rely on having accurate schemas beforehand. However, outdated, incomplete, and inaccurate schemas-make these WAF protection irrelevant and noisy. Furthermore, WAFs require a lot of tuning and maintenance, so the variety and velocity of APIs makes them practically impossible to stay atop. Security teams are often unaware of changes being made to APIs - or even all the APIs the organization must begin with.

And even though we can configure WAF to adhere to strict API based rules to protect specific APIs, it still cannot protect them–against attacks that target the business logic itself. API documentations can provide technical guidance and standards for API calls, but they do not explain their exact functionalities (i.e., the business logic). These attacks target the functionalities of API like manipulating the call flow within the API. For example, a legitimate user tries to access the resources of other users after gaining credentials without their consent. Despite WAFs ability to protect API against a wide range of known technical attacks, they fail to assimilate enough data to stitch together the dynamic nature of API use over time, leading to their inability to identify the bad actors, across millions of users, who are trying to exploit the API vulnerabilities.

AI & API Security

The drawbacks of WAFs establishes the need for a protection tool which can understand the business needs and requirements of the API even with frequent changes in the API functionality and usage. The most obvious indicator for that is the user behavior of the

APIs. Understanding application behavior patterns requires a deeper grasp of the API functionality to determine a legitimate call.

Every API performs a specific task, hence the user behavior corresponding to that API is also unique. API calls and transactions help us model the intended flow and business logic for that API which further helps in detection of anomalous event. Such a data-driven approach helps identifying the threats without the need for documentation and considers frequent updates done to the API. Granular API level behavior models for each API which are learnt automatically thus serve as the ideal solution for rising landscape of sophisticated API attacks.

AI vs Rule based Security

Let us take an example of an API abuse called Penny drop attack in the context of banking industry. Penny drop APIs are used to verify bank account details automatically. These APIs allow financial institutions to validate the bank account information of consumers in a secure manner. The service provider performs a transaction for INR 1 into the client's bank account. Client information is confirmed based on the transaction response received. In a few cyber-attacks, it was observed that malicious actors exploited vulnerabilities in these APIs using bots to pose as legit clients and initiate multiple fraudulent transactions.

Individually observed, an API call originating is not malicious, so they lack certain suspicious signatures, hence do not get blocked by the security layer of WAF. These types of requests which attack the business logic while posing as legitimate requests are something WAF's struggle to detect. On the other hand, an AI model trained on general user behaviors of an API will be able to detect this behavior as deviating from the normal user behavior and flag these calls as malicious and block them if required.

Conventional / Rule Based	AI Based techniques
Detecting attacks/anomalous activity based on **handwritten rules**	Anomaly detection is a widely used field in AI and can be leveraged to **learn patterns** in API calls
Static in nature (might miss detection on slight modification)	More **flexible and can detect attack types** like previously observed attacks
Need to constantly update as the attack types keep **getting more complex**	Deep learning techniques can be used to keep updating the anomaly detection module to learn the **changing patterns in observed attacks**.

AI vs Rule based techniques

As the use of API has grown exponentially, the number of exposed flaws in business logic which can be exploited are also increasing rapidly. The number of endpoints which can be interacted with and exploited has also increased. APIs today are providing access to functionalities which were previously inside a monolith implying more flaws to be exploited.

Protecting APIs from threats requires analysis of all API traffic to gain the context needed to identify and stop attackers. A WAFs proxy architecture limits the ability to see the big picture - instead, WAFs provide protection one transaction at a time. Without broader context, and the ability to stitch together disparate activities from a single user, a platform cannot stop attacks in progress, for example.

In this context, an Artificial Intelligence based system which learns to distinguish anomalous behavior from general behavior using the data itself becomes the need. In both types of API attacks discussed above (DDoS, and SQL injection), we see that the advanced automated detection and blocking systems are more effective in detecting unusual behavior than normal. For example, a detection system considering IP profile like its demography, time of calls, usual endpoints, and usual traffic along with the normal behavior of overall traffic of the API and IP addresses consuming it, is more capable of detecting attacks originating from compromised set of IP addresses.

Similarly, in case of detecting SQL Injection attacks, using Machine learning models based on Natural language processing (NLP) can detect what a usual semantic pattern is and what an anomaly is - something that can assist in the automated detection of attacks.

How AI Based API security works:

At the heart of AI Based API security will be Machine learning Models trained on API data. The models are fed information about each API call's metadata, access tokens or cookies, and the timing and order of certain operations. What the model learns from this data is what a normal API user behavior looks like and how to distinguish it from that of a malicious user.

The model learns the complex correlations of features of a call like user device, time of call, request and response lengths, authentication, and authorization time both and aggregate level for a unique client with suspicious activity. Any user whose general usage differs from what the normal user does is marked an anomaly and is a candidate for threat. On the flagged candidate's other validation methods like NLP models on the string data like call URL's, content, and header features can be run to give a confidence score. By understanding feature importance, the call features which result in the user being flagged as malicious can also be pointed out.

The AI based State of the Art models are very powerful in learning complex relations between features which might indicate the call being an anomaly. While firewalls only used signature-based detection to detect if a single call has malicious signatures, AI models can do this in complement with a behavior detection model for overall call data of a unique user. Thus, in addition to detecting known attacks with attack

signatures, these models can detect client trying to abuse API by finding flaws in business logic. For example, in case of penny drop attack we discussed above, the model can identify a certain client (identified uniquely using his / her device, geolocation and other Identifying features) sending multiple account verification requests at regular intervals which is not what normal user behavior would look like and flag it as an anomaly.

Challenges in using AI Based models in API Security:

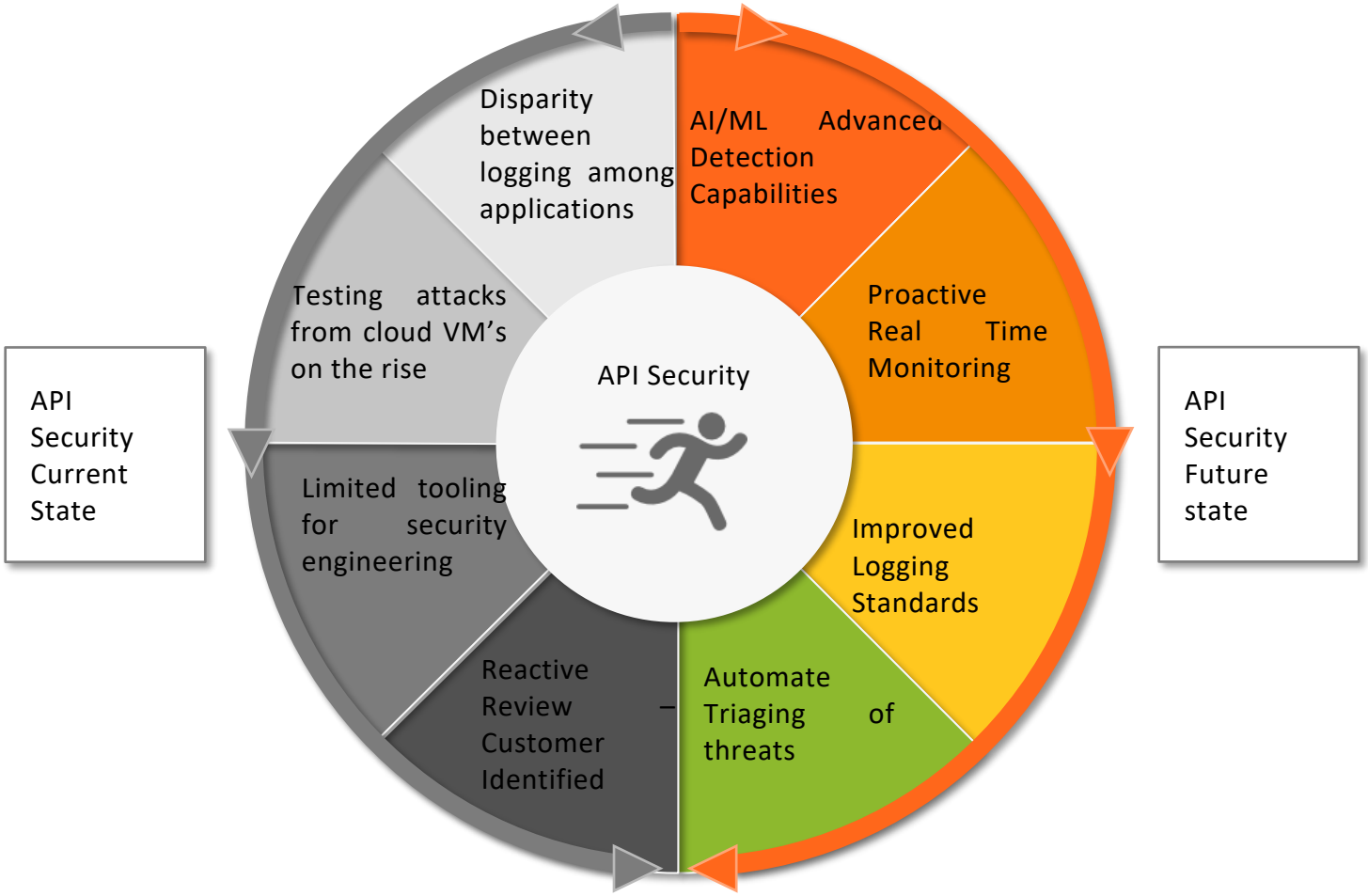
Along with the advantages of using AI in API Security, there are a few challenges. First and foremost are the modeling and validation challenges that are present in most AI-based modeling. Most of the deep learning models require a huge amount of data to be trained and are prone to overfitting for small samples. The quality of data these models are trained on is also a challenge. This comes as an issue for smaller or newly created APIs for which we do not have sufficient data to train an efficient model. Also, APIs with low traffic in general suffer from a lack of data which makes less effective models.

Another issue present in this domain is that of deep learning models being black boxes. As most of these models are trained on very large data to fit billions of parameters through complex optimization algorithms and all the parameters have a role in the final output, it is very difficult to know exactly how the model decides. We can provide the input to the model and get its output, but it does not explain how the model arrived at the particular output. In essence, the AI model is just a black box to us which gives us some results for input.

Although these are significant challenges to the use of AI-based models there are workarounds also. For training models with a lesser amount of data, we can use transfer learning. In transfer learning, we train a model on another general problem for which we have sufficient data and fine-tune the model on the smaller data for which the model needs to detect the anomaly. We can train a model on general API data which is not scarce and fine-tune the model on the API we need to detect anomalies. The data quality issues may be dealt with by reducing the disparity and improving logging standards. Careful data cleaning and feature engineering is the most straightforward solution to these challenges. For the black box problem, some algorithms give some insight into the working of the model. Algorithms like surrogate trees or heatmaps can give an estimate of which features are important to the algorithm while producing output for a data point.

Current to Future

AI/ML with Real time capabilities could be the future direction. In addition to this, improvement in logging standards and automation can fill the security gaps.



Conclusion

As the use of API has grown exponentially, the abuse of these APIs has also increased. The shift from monolithic architectures to microservices in clouds and containers has revolutionized development cycles but also increased the vulnerabilities exposed to internet. APIs today are providing access to functionalities which were previously inside a monolith implying more flaws to be exploited. Further, the number of endpoints which can be interacted with and exploited has also increased. In fact, because of their direct access to extremely sensitive data and functionality, APIs are frequently cited as one of the primary security concerns that organizations face today.

Some of the major types of attacks on APIs are Broken Access Control, SQL injection, DDoS attack, excessive data Exposure, etc. Although there are traditional methods for the detection and blocking of these attacks, they are much more difficult to implement and not very effective at times when implemented. Further, the conventional methods adhere to strict API based rules to protect specific APIs, thus, cannot protect them against attacks that target the business logic or functionalities of API like manipulating the call flow within the API.

The drawbacks of these conventional methods establish the need for advanced automated detection and blocking systems which are more effective in detecting unusual behavior than normal. These advanced protection tools should be capable of understanding the business needs and requirements of the API even with frequent changes in the API functionality and usage. In these cases, real-time monitoring using automated algorithms can detect the problem and reduce the potential damage much earlier than the scheduled user-generated reports.

In this context, an Artificial Intelligence based system which learns to distinguish anomalous behavior from general behavior using the data itself could be quite helpful. As, every API performs a specific task, hence the user behavior corresponding to that API is also unique. API calls and transactions could be used to build an AI model which understands the intended flow and business logic for that API and helps in detection of anomalous event. Such a data-driven approach helps identifying the threats without the need for documentation and considers frequent updates done to the API. Granular API level behavior models for each API which are learnt automatically thus serve as the ideal solution for rising landscape of sophisticated API attacks.

Using best practices and moving towards Artificial Intelligence to secure Application Programming Interface (API) used for Open Banking would help in comprehensively addressing the cyber risks in a timely manner and to mitigate cyber risk in interconnectedness of financial systems.

Appendix

[1] 20 Impressive API Economy Statistics | Nordic APIs |

[2] Understanding your API attack surface: How to get started | CSO Online

[3] https://www.apisec.ai/whitepaper/api-security-best-practices

[4] https://owasp.org/API-Security/editions/2023/en/0x11-t10/