THREAT HUNTER RESEARCH

API Security Program White Paper (2022)

A practical white paper on API asset management, defense in depth and security across the API lifecycle.

Original publication: 2022-08-04
Research team: Threat Hunter Research Team
Source report: 22 pages

Original report text

This text version is reconstructed based on the 22-page original PDF, retaining the report narrative, chapters and research scope; the cover, repeated table of contents and purely decorative pages are not repeated. Localized figures are placed in context throughout this web edition, with the original PDF retained for reference.

Preface

In the era of digital economy, data has become an important factor of production, and the ability to control and utilize data elements has become the core driving force for economic growth. Data has become an important asset of enterprises because of its extremely high monetization value. At the same time, attacks and defenses around data have become more and more intense. Data security is an indispensable and important component of network security.

Driven by emerging technologies such as cloud computing, big data, the Internet of Things, artificial intelligence, and 5G, as well as the epidemic factors in recent years, most companies are actively promoting digital and online transformation. Digitization and onlineization have led to an explosion of APIs connecting data and applications. Enterprises integrate data resources through API capabilities and provide them to users, partners, internal employees, etc., allowing data to flow in multiple parties and improving enterprise production efficiency with the help of cloud-based technology. APIs will play an increasingly important role in digital transformation. Data exchange and business logic implementation through APIs have become the most common method. Each API may become an attack surface. As APIs increase, vulnerabilities will also increase, making APIs a key target for attackers.

The 2022 national-level offensive and defensive drills have added new attack and defense points for data leakage, indicating that data security protection has gradually moved from the implementation of regulatory regulations to specific actual offensive and defensive operations. Although data leakage caused by intrusion into the tow library has become very difficult as the security level of network boundaries has increased, data leakage incidents caused by API security issues have occurred frequently in recent years. It can be seen that API security is a common but seemingly unknown challenge. OWASP compiles the API Security Top 10 issues every year with new changes, which are worthy of every enterprise's attention and updated response strategies. From the traditional WAF to the WAAP solution proposed by Gartner, API security issues have become a new focus of the industry.

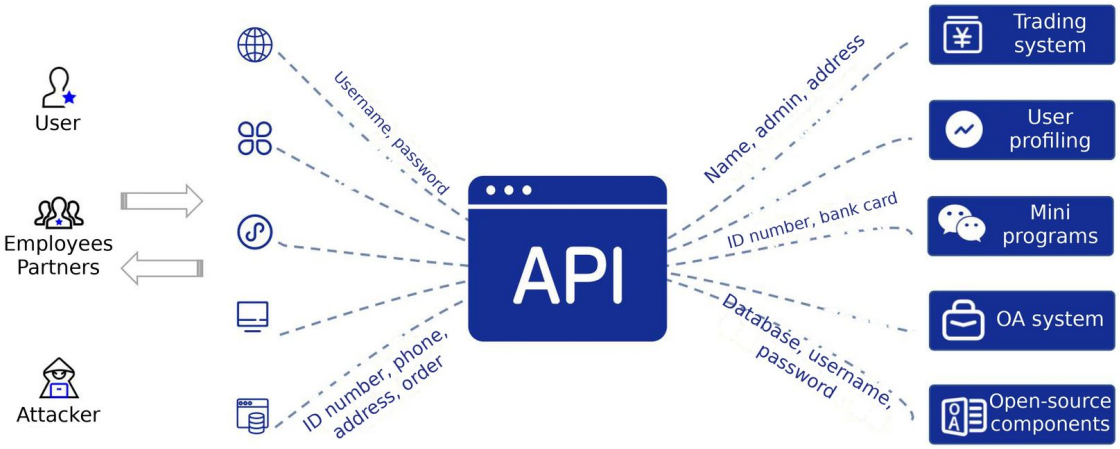
1. What is API

1.1. Definition of API

API (Application Programming Interface) is a computing interface that defines the data interaction method and function type between software. With the popularity and development of the Internet, APIs have expanded from interfaces called internally by early software to interfaces providing external services on the Internet. By calling the API, the caller can obtain various services provided by the interface without accessing the source code or understanding the details of the internal working mechanism.

The API we are discussing so far refers more to Web API, which is different from the API exposed by the operating system or library to applications running on the same machine. A Web API is a

programming interface consisting of one or more publicly exposed endpoints that point to a defined request-response messaging system, typically represented as JSON or XML.



APIs sit between users, employees, partners and attackers on one side and business systems on the other, carrying personal data in both directions

Source: Threat Hunter original report, page 4. Redrawn in English from the original figure.

1.2. Type of API

Web API is defined as HTTP-based and there are four main types of Web API seen today:

RESTful API: Dating back to Roy Fielding's PhD thesis in 2000, representational state transfer is the most common type of Web API, often using JSON (JavaScript Object Notation) to process data. RESTful APIs are easily consumed by modern front-end frameworks (e.g., React and React Native) and facilitate the development of web and mobile applications. They became the de facto standard for any Web API, including those used for B2B, and are now the dominant application style.

SOAP API: SOAP uses verbose Extensible Markup Language (XML) for remote procedure calls (RPC). It is currently rarely used and can still be seen in some older systems.

GraphQL API: The newer GraphQL standard provides database access through a single POST endpoint (usually /graphql). It is mostly used in data scenarios with graph structures. Practical applications are currently relatively rare.

gRPC API: A newer high-performance binary protocol based on HTTP/2.0, mainly used in scenarios with high concurrent requests from a large number of users.

1.3. Summary

In today's application-driven world, an essential element of innovation is APIs. From banking, retail, and transportation to the Internet of Things, self-driving cars, and smart cities, APIs are a key part of modern mobile, SaaS, and web applications. Enterprises can see the use of APIs everywhere in customer-facing, partner-facing, and internal applications. Akamai's statistical report states that "API requests account for 83% of all application requests, and the number of API request hits is expected to reach 42 trillion in 2024."

Essentially, an API exposes the application's logic and sensitive data, such as personally identifiable information, and because of this, it is increasingly being targeted by attackers.

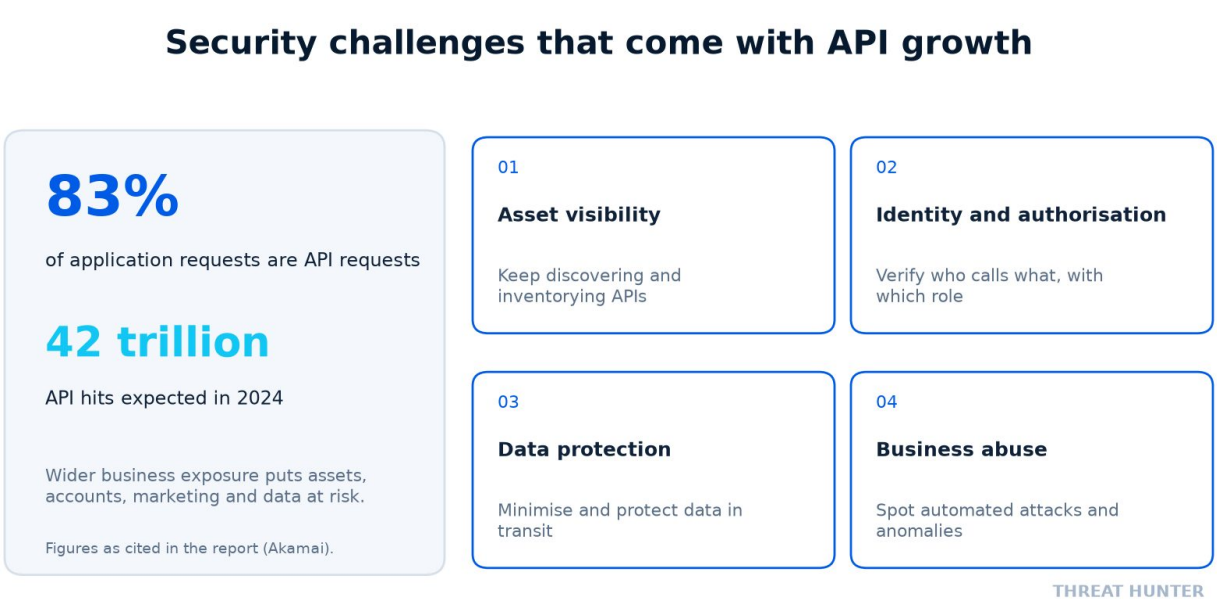
Without secure APIs, rapid innovation will be impossible.

2. API security challenges

From the development process of API, we can know that API security issues have been changing with the development of API technology. API security focuses on security issues in the API field and solutions to these issues from a security perspective. The security field it focuses on is close to traditional Web security, but different from Web security. Traditional web security pays more attention to the security of web applications, focusing on server-side application security. The main forms of vulnerabilities are SQL injection, XSS, CSRF, etc. In the new situation, APIs carry business logic and data flow. With the development of microservices and cloud computing, modules are becoming more and more independent. Each module can be dynamically expanded according to request needs, and the original server boundaries are broken. As the attack surface continues to expand, it brings new security management issues and security technology issues, and the external environment it faces is more complex than traditional Web security.

From the perspective of the cloud and pipe end, API security includes the security of API services and their operating environment (similar to traditional Web security) on the server side, the pipeline side includes the security of API message transmission, and the terminal includes the security of API client applications, IoT devices, and regulatory policy security risks. From the perspective of security scenario classification, API security includes network security, web application security, security development, and regulatory compliance. At the network level, API security mainly focuses on the communication security between the client and the API server; at the Web application level, it focuses on the protocol specifications between the API client and the API server, account security, application security audits, common API vulnerabilities and how to avoid these security issues through API security design;

At the security development level, from the perspective of the API life cycle, SDL or DevSecOps models are combined to comprehensively manage the security of the API development process; at the regulatory compliance level, it is necessary to consider API data privacy protection and compliance design in conjunction with laws, regulations and industry regulatory requirements.



API security challenges: APIs already carry most application requests, and security has to cover asset visibility, identity and authorisation, data protection and business abuse

Source: Threat Hunter original report, pages 5 to 7. Summary graphic drawn in English from the report text.

2.1. Lack of API protection has become the biggest risk exposure for business and data security

The 2021 IBM Security X-Force report stated that two-thirds of the data security incidents it analyzed were caused by insecure APIs. According to data provided by Gartner, by 2025, due to the explosive growth of APIs exceeding the capabilities of API management tools, 50% of enterprises will have a lack of API security protection, and 90% of enterprises can only protect their publicly released APIs, while other APIs are not monitored, and most enterprises lack practical experience in API security.

Gartner therefore predicts that by 2022, API abuse became the most common attack vector leading to data leakage in enterprise web applications, and the risk of data leakage caused by API security issues will even double by 2024.

The following figure shows some typical attacks caused by API vulnerabilities in recent years:

Data leaks caused by API vulnerabilities

Organisation	What happened
Facebook	A third-party app obtained data on 50 million users through an API and used it for political ad targeting
LinkedIn	API abuse leaked the names, emails, phone numbers and industries of 700 million users
Weibo	A contact-matching API was hit by credential stuffing, leaking data on 500 million users
UPS (US)	An API authentication flaw leaked data on 60 million users
Taobao	Logic flaws in two APIs leaked shopping data on 1.1 billion users
Companies in red-team drills	Arbitrary-login flaws in OA systems, ajax.do file-upload flaws and arbitrary file-upload flaws let target systems be breached

THREAT HUNTER

Typical data leaks caused by API vulnerabilities in recent years

Source: Threat Hunter original report, page 6. Table typeset in English from the original.

Subject of the incident: A third-party application on a major social network obtained data on 50 million users through an API and used it for political advertising. A professional networking platform leaked the names, emails, phone numbers, industries and other information of 700 million users due to API abuse. A microblogging platform's address-book matching API was hit by credential stuffing, resulting in the leakage of 500 million user information. The U.S. Postal Service caused the leakage of 60 million user information due to API authentication loopholes. Logic loopholes in two APIs led to the leakage of 1.1 billion users' shopping information. Enterprise OA System arbitrary user login vulnerabilities, ajax.do file upload vulnerabilities, arbitrary file upload vulnerabilities that lead to the target system being compromised, and other APIs are one of the most important transmission methods for data interaction, and have therefore become key targets for attackers to steal data. At the same time, due to the lack of API protection, issues such as which

APIs are exposed to the outside world, who is open to the API, and what sensitive data is flowing in API communication have not received due attention. Attackers can attack APIs through API authentication and authorization vulnerabilities, excessive data exposure, data traversability, and security configuration flaws to conduct data theft and business attacks.

2.2. Main security issues faced by APIs

2.2.1. API assets are not visible

Most enterprises have not included API assets in the scope of asset inventory and have not done a comprehensive asset sorting work; and

API continues to be iteratively updated as the business changes, resulting in enterprises omitting some assets when conducting security assessments on APIs or failing to maintain related applications for a long time. On the other hand, most companies do not include the data flowing on APIs and associated accounts into assets for unified management and maintenance. Once a certain type of API framework vulnerability breaks out or is hacked, the relevant application nodes cannot be located in time, and the best emergency response time will be missed.

The visibility of assets is the basis of security. Due to the large number of APIs, rapid updates, and changes in sensitive data and accounts associated with them, it is difficult to complete continuous and effective sorting of API assets through static methods with limited manpower.

2.2.2. Increased attack surface

With the widespread application of cloud computing technology, more and more businesses are migrating to the cloud. Cloud-native development is based on elastic expansion models such as microservice architecture and k8s. Compared with single-point calls in traditional data centers, APIs have become the standard for inter-module communication. Business logic is scattered in multiple microservice modules. There are countless APIs in both east-west and north-south directions. Each API may become an attack surface, resulting in a much larger attack surface that needs to be guarded against.

Common attack surfaces for APIs:

Attack surface It shows that there are loopholes in authentication and authorization, which introduces the attack surface

- Some APIs have insufficient or missing identity authentication design at the beginning of the design, which allows attackers to conduct unauthorized or unauthorized attacks, and can access data at will through the API
- The permission design is unreasonable, causing user A to access the data of user B who belongs to the same role, resulting in a horizontal unauthorized access attack; or a user with ordinary permissions A can operate the functions of the administrator B, resulting in a vertical unauthorized access attack
- Insufficient password security verification, attackers can use weak passwords to launch attacks to crack accounts. Insufficient verification of input parameters introduces attack surfaces. During the process of API design and iteration, developers lack verification or lax verification of API input parameters. Attackers may use constructed inputs to carry out injection attacks such as SQL, In order to be compatible with multiple functions during design, too much data will be mixed together and returned to the front end, or desensitized and plaintext data will be

returned together, and then the front end will filter the relevant data. This causes the API to return too much data, and attackers can use traffic interception and other means to

Obtain the original data returned by the API, thereby exposing the risk of data leakage

- For login scenarios, the API displays too detailed error information when error prompts. Attackers can use the prompt information to carry out credential stuffing and account scanning attacks. The API does not encrypt the transmission data and directly transmits it in plain text. The attacker can directly obtain the interactive format and data of the API through network sniffing and other means, analyze the obtained data, and carry out the next attack.
- API The design did not take into account business-side logic issues such as replay logic, frequency limits, transaction integrity logic, etc., allowing attackers to modify amounts and tamper with transactions through replay, modify parameters, etc.
- There are logic bugs in code implementation, allowing attackers to use bugs to carry out attacks
- Security configuration flaws introduce attack surfaces
- Allow web servers to browse any directory, do not turn off HTTP header configuration, and do not turn on some authentication and authorization configuration switches
- The default password of the business system has not been modified, allowing attackers to use the default password to log in. Deploying the internal system on the public network uses vulnerable and outdated components to introduce attack surfaces. Introducing open source or third-party plug-ins, modules, frameworks, etc. during the development process. When the third-party software or modules referenced have security issues, it will inevitably lead to vulnerabilities, malicious code, and "backends" in the code.

Security risks such as "gates" are introduced into the API interface

- The version of open source or third-party components used is too low, and there are loopholes that can be attacked

2.2.3. API attacks are more subtle

The API needs to be open for users to use. It has the characteristics of openness and carrying business logic. Attackers can call the API just like normal users. As a result, the attacker's traffic is hidden in the traffic of normal users, and their attack behavior will be more covert and harder to detect.

Some common attacks:

- High-frequency access behavior: The API interface does not limit the access frequency of the API interface at the beginning of the design, so that attackers can access a large number of API interfaces in a short period of time, and complete attacks such as promotion abuse and malicious registration in a short period of time, and may even

brings CC attacks.

- Large amounts of data download behavior: The API interface does not limit the number of user downloads within a certain period of time, the size of the download content, etc. As a result, attackers can obtain large amounts of data through multiple downloads, which can easily cause the leakage of a large amount of sensitive data.
- Web crawler behavior: The openness of the API interface. If no anti-crawler security policy is set, the attacker can use the proxy IP or modify the User-Agent request header to hide the identity,

obtain the internal system account of the enterprise through information collection, and use the web crawler to crawl account permissions and all API interface data open to the public network, resulting in a large amount of data leakage.

- Dynamic proxy IP low-frequency crawler behavior: If the API has frequency limits and anti-crawling strategies, attackers will also use a large number of dynamic proxy IPs in a low-frequency and slow manner to bypass existing defense measures, traverse and crawl data, conduct malicious registration, or market fraud.
- Interface abuse: If the API does not verify the user's identity, lacks whitelist check, and lacks the logic of verification code for human-machine verification, attackers will use the enterprise's API to jump to URLs, send arbitrary SMS and other functions to complete the attack, consuming the enterprise's SMS expenses and bringing negative social impact to the enterprise.

2.2.4. Regulatory Compliance Challenges

In recent years, with the continuous deepening of cyberspace governance at the national level, meeting compliance requirements has become a necessary condition for every enterprise to conduct normal business. Since 2016, China has successively introduced a series of laws and regulations to supervise data security issues, from the implementation of the "Cybersecurity Law" in June 2017 to 2021, which is known as the "First Year of Data Security".

In 2021, laws and regulations such as the "Data Security Law" and the "Personal Information Protection Law" will be officially implemented one after another. The Cyberspace Administration of China released the "Measures for Data Transfer Security Assessment (Draft for Comments)" and publicly solicited opinions. It can be seen from this series of laws and regulations that China's digital security supervision of enterprises is indeed becoming more stringent and standardized, with more detailed focus and increasingly stronger penalties. For enterprises, building a data security system around the entire life cycle of data from data collection, storage, access, use, and destruction has become a key security construction task.

At present, most enterprises have made relatively comprehensive security construction in data collection, storage, and database access, and the awareness of security construction in data flow access is gradually sprouting and developing; and API, as the main channel connecting data and applications, has become one of the weakest links in data transmission. Traditional WAF and IPS security devices do not focus on data security. The current security protection of API is relatively weak, so API can easily become the number one target for data theft in the eyes of attackers.

The financial industry standard JR/T 0185-2020 "Commercial Bank Application Programming Interface Security Management Specification" proposes multi-faceted compliance for API management from the perspective of API types and life cycles such as security design, development, deployment, integration, and operation and maintenance.

Sexual Requirements. These standards or specifications provide directional guidance for enterprises' API security practices, and also provide implementable standards for API compliance. Only when enterprises complete such compliance challenges can they conduct business better.

The figure below shows a detailed description of the penalties imposed by data-related regulations in recent years:



Penalties for data leaks have risen with each new data law, from CNY 1 million in 2017 to CNY 50 million or 5% of annual turnover in 2021

Source: Threat Hunter original report, page 10. Redrawn in English from the original figure.

2.3. Summary

From an attack perspective, when more and more companies open their business capabilities to the outside world through APIs with the intention of building an ecosystem together, security vulnerabilities in accounts, marketing, and data can be directly and quickly profited by attackers. Moreover, the current security awareness of enterprises in this area has just begun to sprout. This new type of attack surface is full of temptations. The stronger the attacker's motivation, the more attack methods, and the greater the harm caused, the greater the challenge to the enterprise's defense will be.

3. API full life cycle security protection

Due to the rapid development of cloud computing, more and more enterprises are migrating applications and data to the cloud and exposing core business capabilities and process-related APIs to provide services to external partners. Breaking away from the traditional intranet or network zoning, the development and integration of cloud applications and cloud management APIs are used by potential business partners and attackers, which inadvertently increases API security risks.

For most enterprises, it is difficult to fully master all APIs of the system; developers are often only familiar with the relevant modules they have developed, and many technical developers believe that adopting new and cool technologies is more important, choosing new features on the technical route, regardless of whether the API is attacked. In the absence of an API security management platform and a comprehensive and systematic API security governance system, API security risks are even more uncontrollable.

3.1. Guidelines for API security design

There are many security design principles in security architecture design, such as open design principles, permission minimization, openness minimization, default distrust, etc. Safety design principles require continuous learning and training so that safety designers and R&D can master them. API is the new boundary of business systems. To ensure this new boundary of API from a design perspective, there are two basic principles that you can refer to, namely the 5A principle and the defense in depth principle.

3.1.1. 5A Principle

The 5A principle refers to the acronym for the five parts of Authentication (identity authentication), Authorization (authorization), Access Control (access control), Auditable (auditability), and Asset Protection (asset protection). Its meaning is that when security designers are doing security design, they need to consider the rationality of the security design from these five aspects. If one aspect is missing, the security design is incomplete.

- Identity authentication solves the problem of "who are you", the purpose is to know who is communicating with the API service and whether it is a client request allowed by the API service. Some API services that require permissions to access need to know who is requesting and whether the request is allowed to ensure the security of API interface calls. Authentication methods mainly include username/password authentication, dynamic password, digital certificate authentication, biometric authentication, etc. For businesses with relatively high security requirements, two-factor authentication (2FA) or multi-factor authentication (MFA) can be used. Common combinations include username/password + SMS challenge code, username/password + dynamic token, username/password + face recognition, face recognition + SMS challenge code, etc. Authentication is usually integrated into single sign-on SSO systems, using a unified entrance to complete identity authentication and reduce the attack surface.
- Authorization, which solves the problem of "what data and resources can you access", usually occurs after identity authentication, that is, for services, who is requesting me, does this request have permission? Some APIs can only be accessed by specific roles. For example, only intranet IPs can call certain services, and only administrator users can call APIs to delete users.
- Access control solves the problem of "whether the specific data and functions you access are allowed." It usually occurs after authorization. In many cases, the permissions for a certain role are set correctly, but the access control is not necessarily correct. This is also the reason why there are many unauthorized operations. Access control is the verification of the correctness of authorized client access. A certain role can directly call APIs that do not have access rights. The problem lies in access control. Authorization and access control are often carried out together. There are two methods for reference: one is authorization and access control based on user identity agent, typically represented by the OAuth 2.0 protocol. The authorization of API and the control of accessible resources depend on the identity of the user. The user may be a natural person user or a client application. After obtaining the user's authorization, the resource authorized by the user can be accessed; the other is authorization and access control based on user role, typically with Represented by the RBAC model, API authorization and resource access depend on the role and assigned permissions that the user is granted in the system. Different roles have different permissions, such as functional permissions and data permissions. When accessing resources, based on this

Different permissions assigned to roles can access different resources.

- Auditability solves the problem of "the operations you do can be traced to their source". The purpose is to record key information when calling the API, so that problems can be discovered in a timely manner through auditing methods afterwards, and when problems occur, audit logs can be used to trace the source and find out where the problem occurred. Generally, recording API logs requires: who (account, UA information), at what time, what IP was used (where), what API was called (what was done), what is the result of the operation, what is the Referer when operating the API, etc.
- Asset protection, to solve the problem of "preventing API abuse", mainly refers to the protection of the API interface itself, such as speed limit, current limit, preventing malicious calls, and the protection of sensitive data transmitted by the API interface, such as ID cards, phone numbers, bank card numbers, etc.

3.1.2. Defense in depth principle

The term defense in depth comes from military terminology, which refers to the construction of multiple lines of defense from the front to the rear to achieve the purpose of overall defense. In the field of network security, defense in depth usually means that we cannot only rely on a single security mechanism, but establish multiple security mechanisms to support each other to achieve relative security. The basic meaning of the principle of defense in depth can be understood through an example in daily life. For example, in the process of flying, the airport adopts the following lines of defense: the first line of defense is the explosion-proof inspection at the entrance, which can quickly check whether passengers are carrying explosive-proof materials; the second line of defense is security inspection, which checks whether they are carrying some contraband; the third line of defense is identity verification before boarding the plane to ensure the consistency of tickets. These three layers of defense constitute defense in depth, ensuring that people who have purchased tickets can take the flight in safety.

In API security design, different security technologies can be used at different levels to achieve the purpose of defense in depth. Typical scenarios include the transfer business of online banking. When entering the system, you need to log in for identity authentication. When calling the transfer API later, you still need to enter the password again, and even face authentication is required for large transfers. The identity authentication when logging in to online banking and the identity authentication when transferring money constitute the principle of defense in depth.

The 5A principle emphasizes the rationality of the design of each layer of security architecture, emphasizing the width; defense in depth is to provide security protection for the same problem from different levels and different angles, emphasizing the depth. The combination of these two principles makes the security design an organic protective whole.

Here is an overall diagram of API security that combines the two principles:



Defence in depth for APIs: HTTPS on every client link, then rate limiting, authentication, authorisation and access control, masking and encryption, and log auditing

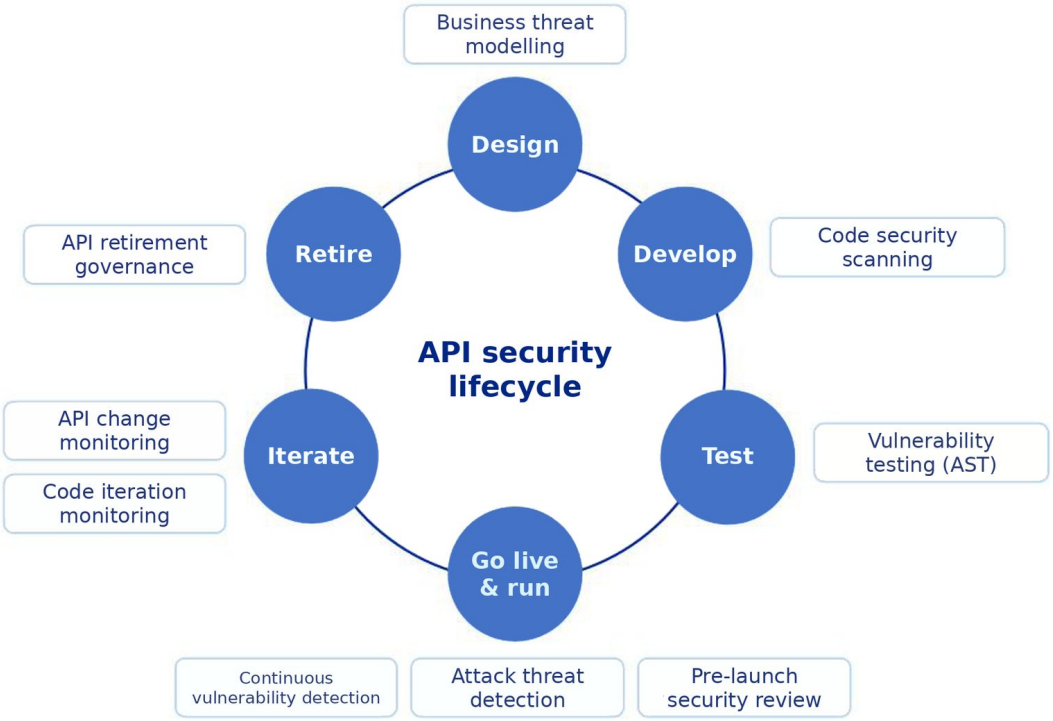
Source: Threat Hunter original report, page 13. Original figure with Chinese text replaced by English.

3.2. Security protection model of API life cycle

From the perspective of external threats faced by APIs, they mainly include the following aspects: the risk of high-risk vulnerabilities such as command execution/SQL injection/server takeover, the risk of unauthorized and unauthorized access due to imperfect permission control, the risk of batch data leakage caused by design flaws, etc. Accessing WAF can solve some command execution/SQL injection problems, but the WAF products currently on the market still have insufficient protection capabilities for API security due to the particularity of API technology. This is mainly reflected in the following aspects: First, the bypass of the authentication and authorization process. Many Internet applications of API authentication and authorization processes are based on OAuth2.0 and OpenIDConnect. Traditional security protection products are difficult to detect threats of business process bypass; second, the data format is difficult to identify, and most of the message formats used by APIs in the interaction process JSON format, XML format, Protobuf format, JWT format, etc. When detecting threats, it is necessary to in-depth analyze the data structure content of these data formats. Traditional security protection products have relatively weak detection capabilities in this aspect. Third, the traffic control capabilities are difficult to meet business needs. When facing CC attacks and slow BOT attacks at the API level, traditionally

The detection and protection strategies used, such as access frequency restrictions, IP blacklist settings, and two-step verification mechanisms, are difficult to protect against new attacks.

Threat protection exists for APIs. The use of WAF products can only cover a small part of the threats. For business, it is more necessary to consider API functional security design from a single point to consider API security through the API life cycle. It is even more necessary to strengthen security construction around every link from design, development, testing, production operations, iteration to decommissioning. Consider the security of APIs throughout the entire life cycle. Through shift-left security practices and tools, we comprehensively integrate management methods and technical means to conduct API security governance and improve the overall security of business APIs.



Security across the API lifecycle: design, development, testing, go-live and operation, iteration and retirement each get their own controls

Source: Threat Hunter original report, page 14. Original figure with Chinese text replaced by English.

3.2.1. Design Phase: Introducing Threat Modeling

At the beginning of design and during the development process, risks are assessed based on business characteristics to achieve reliable and secure design. Security engineers are involved in the realization of requirements and design solutions. If resources permit, security engineers can conduct threat modeling for each API during the design phase, and runtime API risks will be greatly reduced. When resources are limited, build automated threat modeling capabilities, using a combination of "manual review of key businesses" and "automated threat modeling of non-key businesses" to cover core high-risk APIs such as account login, file upload, marketing activities, etc.

In threat modeling in the design phase, STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege) and attack tree models are generally used as common threat modeling technical guiding principles. Combining industry security white papers and historical security incident summaries, according to the needs and functional design of business APIs, and based on the above 5A principles, we sorted out the possible targets and methods of attacks by potential attackers.

For APIs, attackers may usually include: malicious internal employees, external attackers, competitors, curiosity-driven attackers, etc. The attack path may be to steal data through internal system APIs outside working hours, use API logic loopholes to crawl data in batches, launch BOT

attacks, use SMS verification-code receiving services to initiate malicious registration, use proxy dialing platforms to launch low-frequency and slow attacks, etc.

The following list of security checklists and best practices can be used as a reference during the threat modeling phase:

- OWASP REST Security Checklist: https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html
- API security tools and resources: https://github.com/arainho/awesome-api-security
- API Security Checklist: https://github.com/shieldfy/API-Security-Checklist
- JWT Security Checklist: https://pragmaticwebsecurity.com/files/cheatsheets/jwt.pdf

3.2.2. Development stage: security development awareness and standard training, introduction of security tools

In an ideal situation, threat modeling will be discussed with R&D during the design phase, and everyone will have a comprehensive and clear understanding of the security issues that the API may encounter; however, during the coding and implementation phase of R&D, the implementation may often be incomplete or new bugs may be introduced. At the same time, considering that R&D personnel are constantly changing, security engineers are required to provide auxiliary security development tools such as API security development specifications, security development plug-ins, security auxiliary packages, and sensitive data encryption and decryption tools.

On the one hand, training is needed to enhance the awareness and ability of all employees on security development. On the other hand, tools are needed to implement automated inspections and detect bugs and vulnerabilities early.

In terms of API security development training, four aspects can be considered:

- API security management framework and key indicators, the company’s overall API security framework, stereotyped and quantitative indicators, such as the number of vulnerabilities after launch, etc.
- Common API security issues, such as OWASP API Top 10, and some representative issues extracted from security operations through SRC and business. Through problem cases, everyone can understand and learn more.
- Common API security technology and security design, sorting out and collecting some excellent security practice cases from the industry and enterprises themselves, which can expand R&D thinking and think of better solutions when encountering similar problems.
- API secure coding cases, combined with excellent cases in the industry, are based on the company's own characteristics to develop secure coding specifications and cases that meet the company's characteristics.

The following list of some secure coding and development specifications can be used for reference:

- OWASP Secure Coding Practices: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated_content
- An open-source code security guide (secguide) published by a major internet company

- Threat Hunter API Security Development Specification: https://www.yazx.com/reportDetail/14ac36b8-f5c6-11ec-af75-00163e048a4c

For automated verification, consider the following tools:

- Introduce a code white-box code scanning tool, embed it into the CI/CD or R&D process, and scan it when it is released to the test environment.
- Introduce API management tools to centrally and uniformly manage API documents, and monitor the process data of API changes and iterations.
- Introduce tools related to API security verification to verify the correctness of API security implementation to ensure that the verification work is as comprehensive as possible, such as FuzzDB and personal data privacy monitoring tools.

3.2.3. Testing phase: Vulnerabilities are added to the testing process and AST tools are used to improve coverage.

Add API security-related content to the testing process. For automated testing processes and manual testing, API security testing is added based on API business logic implementation, stability, and performance testing. The purpose of implementing API security testing is to automatically scan each API. Automated API security testing analyzes whether there are vulnerabilities in the API from both the request input and response parts. The general content of API security testing mainly includes API authentication, authorization, input validation, exception handling, data protection, secure transmission, and HTTP Header security.

There are a large number of APIs and they are constantly iteratively updated, so automated tools are needed to assist in security testing. Commonly used tools include the following three categories: SAST, DAST, and IAST tools, which can discover security vulnerabilities developed in advance during the testing phase. According to the characteristics of the business, we work with suppliers to optimize the IAST tool to ensure coverage, improve accuracy, and discover many vulnerabilities caused by improper input processing in advance.

- Static Application Security Testing (SAST) is characterized by analyzing the source code or binary files of the application and discovering whether there are vulnerabilities in the application code through syntax, structure, process, interface, etc.
- Dynamic Application Security Testing (DAST) is characterized by simulating hacker attack behavior in the dynamic running state of the application, analyzing the response of the application, and determining whether the application has vulnerabilities.
- Interactive Application Security Testing (IAST) is equivalent to a security detection technology that combines DAST and SAST. It usually adds probes or agents to the application, collects execution logs and function call information in the application, Web container, and JVM, and combines request input and response messages to analyze whether there are vulnerabilities in the application.

Among these three types of tools, IAST is slightly more cumbersome to use, but its technical advantages are more obvious. The vulnerability detection rate is higher than the other two categories, and the vulnerability false positive rate is also lower than the other two categories. It can quickly locate code fragments and API interfaces, and can be used as the preferred automated API security testing tool. If there is no such tool, it is recommended to choose the DAST category.

3.2.4. production operations phase: Use gateway, WAF and traffic auditing tools to sense the attack surface in advance

After early development and testing, the API has finally reached the online stage. According to the business assessment, a security review of the online stage can be carried out. After the API is online, it enters the operation stage. The externally exposed API carries business logic and data flow, becoming the main channel for attackers to attack. Therefore, various security tools are needed to ensure API security during the operation phase.

3.2.4.1. Commonly used security tools during the online stage

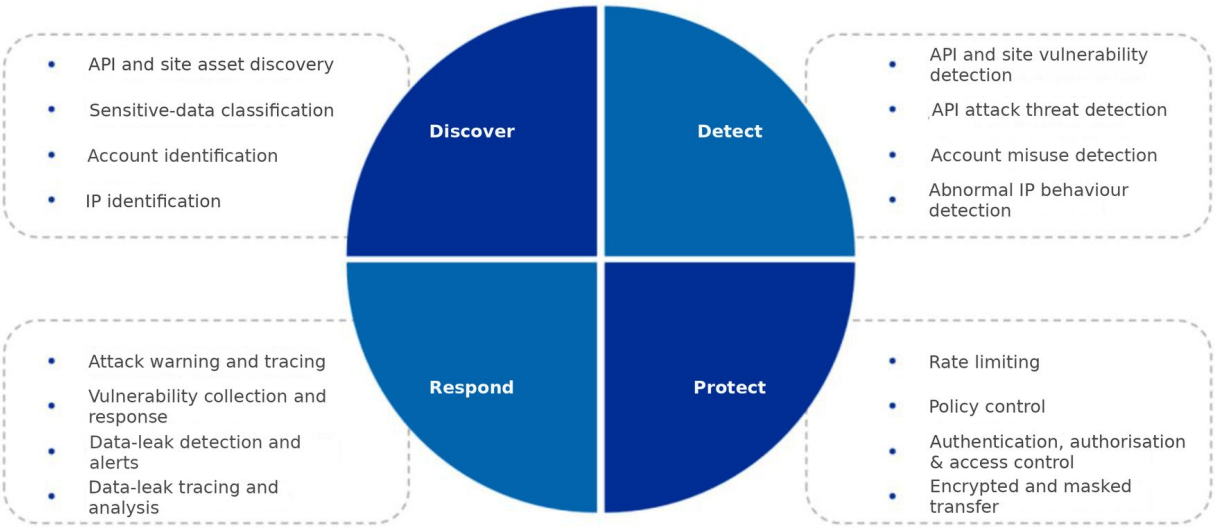
- Use WAF to block SQL injection, XSS and other attack requests, defend against DDoS defense, and conventional BOT attacks. External attack traffic has passed WAF's traffic detection, and the malicious vulnerability scanning behavior has been filtered, and the harm to the back-end system will be reduced. At the same time, operation and maintenance personnel can analyze abnormal behaviors through WAF data and logs, and adjust security protection strategies on WAF to quickly prevent attacks.
- Use API gateway to implement authentication, authorization, access control, and data desensitization, and also help the security team manage APIs. Although API gateway products have functions such as identity authentication, access control, data verification, current limiting and fusing, they can effectively improve the security of APIs. But for internal developers, they need to open the interface between continuous integration (CI/CD) and API gateway, and prepare API import data for CI/CD calls before publishing. This will increase the additional workload of developers and affect the release schedule; at the same time, when the traffic of all back-end services must be communicated by the API gateway, the impact on the original communication performance and the stability of the API itself will be the biggest challenge for those responsible for advancing this work. Yes

to evaluate whether to use an API gateway based on the actual business situation.

- Use API security audit tools to classify and classify APIs, and mark out high-risk APIs, such as those involving sensitive information outbound, involving funds, involving core infrastructure operations, etc.; continuously monitor vulnerabilities caused by API data exposure, unauthorized access, configuration, and unreasonable design; discover zombie APIs, shadow APIs, old versions, and APIs with duplicate functions; use UEBA to discover behaviors that use one's own permissions to obtain sensitive data in batches; use intelligence and machine learning models to discover low-frequency and slow attacks on APIs.

3.2.4.2. Adaptive safety closed loop based on P2DR model

Tools are just means, and their effectiveness can be fully utilized under the guidance of the security model. In the production operations stage, it is necessary to implement a closed loop of "discovery", "detection", "protection" and "response" for API attack threats based on the dynamically adaptive P2DR security model, which can better meet the API security protection needs under the data security system. The premise of security is the visibility of assets. Only when assets are visible can they be controllable. Controllability includes two aspects. One is the continuous monitoring of asset vulnerability risks, and the other is the perception and protection of attack threats.



Adaptive security loop based on the P2DR model: discover, detect, protect and respond

Source: Threat Hunter original report, page 18. Original figure with Chinese text replaced by English.

The first is to continuously build asset discovery capabilities. API discovery capabilities are a competition between API providers and attackers. They must discover APIs before attackers, and sense and understand the possible attack surface of the system in advance. Because APIs are constantly being developed and iterated, the ability to continuously and dynamically sort out APIs is crucial. Assets such as the API, the data flowing on the API, and the accounts and IPs associated when accessing the API can be extracted from the network traffic through the API security gateway, load balancing, or switch mirroring. At the same time, the discovered assets are classified and processed in a hierarchical manner. The hierarchical classification allows security operators to prioritize management.

- API can be classified according to business scenarios, such as login, file upload, file download, third-party open source components, etc.;

Classification can also be performed based on the sensitivity of the returned data.

- The flowing data can be graded and classified according to the country's regulatory regulations, and a mapping relationship between data and APIs can be established at the same time, so that the governance needs of regulations and supervision can be calmly dealt with.
- Account assets can be classified according to permission levels, access to the system, activity, etc.

- IP assets can be classified according to region, type, security, etc.

API asset discovery must be comprehensive. From the perspective of API application scenarios, API assets can be comprehensively covered around the four scenarios of end users, partner companies, internal employees, and open source components and middleware. Using load balancing or core switch traffic to sort out APIs can achieve comprehensive coverage of the above four scenarios.

For end users For partner companies For internal employees For open source components and middleware on the Internet APIs used by apps, Web, mini programs, etc. used by users are open on the Internet Business APIs used by customers of cooperative companies are open for internal employees or partners APIs associated with the application system used by clickhouse, springboot, k8s , hadoop, jenkins and other APIs involved. The second is the continuous risk monitoring capability. Risk monitoring of API security includes the vulnerability risk of the API itself, as well as the risk of attackers attacking the API, the risk of illegal operation of the account, the risk of abnormal behavior of the IP, etc. It is necessary to continuously detect the vulnerability risks, attack behaviors and abnormal behavior risks of assets, and improve the accuracy rate while ensuring the recall rate.

The detection of vulnerability risks includes the following aspects:

- To detect vulnerabilities in your own business, you need to continuously monitor API defects in authorization, authentication, data overexposure, configuration, logical design, etc., as well as monitor risks such as weak passwords associated with accounts.
- Third-party component vulnerability detection. Third-party open source components or middleware introduced by enterprises will also have vulnerabilities. APIs in the supply chain need to be continuously monitored for vulnerabilities.

The detection of attack behavior and abnormal behavior includes the following aspects:

- API attack threat detection, attackers use API logic vulnerabilities to achieve data crawling, account registration, blasting, credential stuffing,

Due to malicious attack behaviors such as SMS bombing, arbitrary URL redirection, and transaction attacks, it is particularly important to strengthen the external risk perception and risk blocking capabilities when the API is running, so that any abuse of the API can be detected promptly and accurately, and further actions of the attacker can be blocked in a timely manner. Monitoring attack behavior based on risk intelligence will have better performance in terms of recall rate and accuracy rate.

- Detection of illegal account operations. Attackers can use insiders to steal sensitive data, or obtain sensitive accounts through credential stuffing, account brute force cracking, social engineering phishing, etc., and then use authorized accounts to perform illegal operations. The detection of illegal account operations requires building a baseline based on the historical behavior of the account and the behavior of the account and people in the same organization in terms of login environment, login IP, time to obtain data, access to sites and data, etc., based on the UEBA model to monitor account illegal operations.
- IP abnormal behavior detection. Attackers need to complete their attacks through IP resources. Build historical behavior portraits for IP such as data access, geographical location, API access sequence, risk intelligence portraits, etc., and use machine learning-based methods to detect abnormal IP behaviors, such as path scanning, vulnerability scanning, excessive single API requests, sensitive-data access from foreign IP addresses and other abnormal behaviors.

The detection model built based on IP risk intelligence has high accuracy and strong interpretability.

The third is to continuously build protection capabilities. Under the guidance of the 5A model and the principle of defense in depth, with the help of security products such as WAF and API gateways, we continue to build protection capabilities through authentication and authorization, access control, information transmission protection, current and speed limiting, bypass detection linkage, etc. to respond to DDoS, CC, and BOT attacks and prevent resources from being consumed on meaningless or malicious API requests.

- Data security protection capabilities. Enterprises should encrypt and desensitize the interaction of sensitive information. Special sensitive information needs to be encrypted or decrypted, such as ID numbers, bank card numbers, phone numbers, etc., to reduce the risk of leakage of sensitive information.

The fourth is to continuously build response capabilities.

- Vulnerability management and response. Enterprises should build vulnerability management and response mechanisms, use SRC, penetration testing, and API security tools based on side-channel traffic to continuously mine and collect API-related vulnerabilities, and do patch management to deal with the changing attack surface and covert and changeable attack methods.
- For attack threat warning and traceability, enterprises need to use risk intelligence and big data analysis technology to analyze API access logs, dig out the attack traffic inside, and explain and trace the attack traffic to achieve timely prevention and response.

The attack behavior can be refined from the business traffic and combined with the attacker's historical intelligence information to understand the attacker's resources, techniques, intentions,

Perform correlation tracking analysis on gang situations and potential targets, and then build an attacker intelligence database to continuously track attackers.

- Data leakage warning and traceability, monitoring dark web and data trading platforms, and monitoring whether there is trafficking of corporate data information.

Based on dark web monitoring data, dirty data concerns, second-hand information filtering, and data sellers’ confidence are analyzed to analyze the accuracy and timeliness of transaction data, and accurate data leakage warning information is extracted to provide timely warning of data leakage events. At the same time, the big data log platform is used to trace the leaked data, find the source of the leak, and perform targeted repairs to avoid large-scale data leaks.

For API security protection, continuous automated protection is the key. Overall, you can refer to the three-step API security method proposed by Gartner as shown in the figure below, and combine the methods, means and tools introduced above to achieve security protection during the API production operations phase.



Three steps to API security: take inventory of API assets, detect API vulnerabilities and protect against API threats

Source: Threat Hunter original report, page 21. Original figure with Chinese text replaced by English.

3.2.5. Iterative phase: Use security tools to audit API changes in a timely manner

As business needs change, it is very common for API implementation logic to change. Changes in API implementation logic can easily introduce new risks, especially when developers believe that after a lot of security testing has been done in the early stage, problems are unlikely to occur again. Therefore, security at this stage also needs to be paid attention to. With the help of security tools, API updates and changes can be sensed and governance can be carried out in advance.

- Use API security tools to sense changes in API assets in a timely manner, monitor and alert for changes in API inbound and outbound data, such as new additions, and combined exceptions, and notify security engineers to re-intervene for security review.
- Monitor changes in the code warehouse. When changes are found in the API method code, notify the security engineer to re-intervene for security review.

3.2.6. decommissioning stage: retire zombie and shadow APIs promptly

A large number of APIs are not decommissioned in time after completing their missions, which will not only waste system resources, but also become potential online risks. For example, an API has an over-privilege vulnerability, but it was not discovered because there was no traffic until one day it was exploited by an attacker. During these stages, the following two aspects of operational work are carried out:

- Use security tools to maintain a dynamic API asset ledger and identify which APIs should be decommissioned, such as zombie APIs, old version APIs, and APIs with duplicate functions, to promote timely API decommissioning.
- Mark APIs that will only be used a few times a year, such as recruitment API and marketing activity API. This part must be maintained separately.

3.3. Summary

Security design and operation around the entire API life cycle requires the participation of various roles in the enterprise, and the amount of work invested is huge. Compared with product and R&D personnel, the proportion of security personnel is very small. Enterprises can adjust their investment in different links according to their own business conditions.

In order to pursue efficient defense, it is recommended to start from the production operations stage of the API, analyze the traffic of the API based on risk intelligence, and continuously implement the combing of API and data assets, vulnerability detection, and attack threat detection. On the one hand, the situation of the assets can be clearly understood, and on the other hand, risks can be warned in time and losses can be stopped. This gives security personnel more strategic time and space resources to carry out security shift construction, and gradually realizes the protection of the entire API life cycle.

Conclusion With the advancement of enterprise digitalization and online processes, API security technology has also been vigorously developed. Building API security needs to include API discovery, sensitive data classification, account asset discovery and abnormal behavior detection, API identity and access control, API message security, API transmission security, API threat detection, API security audit, API monitoring and tracking, API management and other aspects.

In order to build a solid foundation for API security and build an API security protection system from a practical perspective, it is necessary to implement security into all aspects of the API life cycle, and then build an API security protection system with better visibility and controllability.