API 安全建设白皮书

目录

前言

数字经济时代，数据成了重要生产要素，对数据要素的掌控和利用能力，已成为经济增长的核心驱动力。数据因其变现价值极高使其成为企业的重要资产，与此同时围绕数据的攻防也变得越来越剧烈，数据的安全是网络安全不可或缺的重要组成部分。

在云计算、大数据、物联网、人工智能、5G 等新兴技术的推动下，伴随着近些年的疫情因素，大部分企业都在积极推进数字化和在线化转型。数字化和在线化使得连接数据和应用的 API 爆炸式增长。企业通过 API 的能力将数据资源整合，提供给到用户、合作伙伴、内部员工等多方使用，让数据在多方流动起来，并借助云智物大移的技术提高企业的生产效率。API 在数字化转型中扮演的角色将愈发重要，通过 API 来进行数据交换和实现业务逻辑成为最常见的方式，每个 API 都有可能成为一个攻击面，API 增多，漏洞也会增多，API 也因此成为攻击者的重点攻击对象。

2022 年国家级攻防演练新增了对于数据泄漏的攻防点，说明数据的安全保护逐步从监管法规落实到具体的攻防实战中来。虽然入侵拖库带来的数据泄漏随着网络边界安全水位增高，难度已经非常大了，但近些年因 API 安全问题导致的数据泄漏事件却频频发生，可以看到 API 安全是一个常见但似乎又不为人熟知的挑战。OWASP 每年整理 API Security Top 10 问题都会有新的变化，值得每个企业关注并更新应对策略。从传统的 WAF 到 Gartner 提出的 WAAP 方案，API 的安全问题成为行业关注的新焦点。

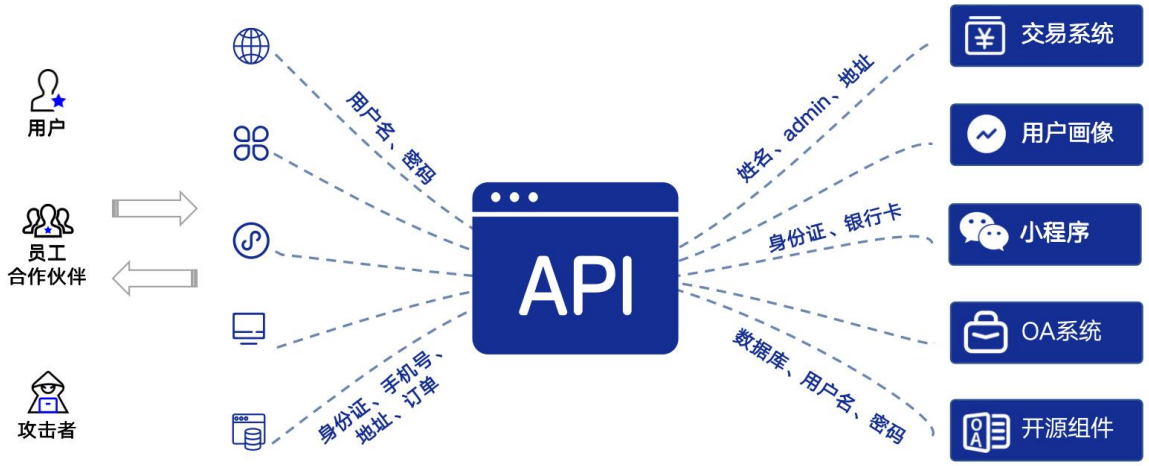
1. API 是什么

1.1. API 的定义

API（Application Programming Interface，应用程序接口）是一种计算接口，定义了软件之间的数据交互方式、功能类型。随着互联网的普及和发展，API 从早期的软件内部调用的接口，扩展到互联网上对外提供服务的接口。调用者通过调用 API，可以获取接口提供的各项服务，而无须访问源码，也无须理解内部工作机制的细节。

目前我们讨论的 API 更多是指 Web API，不同于由操作系统或库公开给在同一台机器上

运行的应用程序的 API。Web API 是一种编程接口，由一个或多个公开暴露的端点组成，指向已定义的请求-响应消息系统，通常以 JSON 或 XML 表示。



1.2. API 的类型

Web API 被定义为基于 HTTP，今天看到的四种主要类型的 Web API ：

RESTful API：可以追溯到 Roy Fielding 在 2000 年的博士论文，代表性状态传输是最常见的 Web API 类型，通常使用 JSON（JavaScript 对象表示法）来处理数据。RESTful API 很容易被现代前端框架（例如，React 和 React Native）使用，并促进 Web 和移动应用程序的开发。它们成为任何 Web API 的事实上的标准，包括用于 B2B 的那些，是目前的主流应用风格。

SOAP API： SOAP 使用详细的扩展标记语言（XML）进行远程过程调用（RPC)。目前使用比较少，在一些老旧的系统能还能看到。

GraphQL API： Facebook 开发的新 GraphQL 标准通过单个 POST 端点（通常是/graphql）提供数据库访问，多用于具有图结构的数据场景，实际应用目前比较少见。

gRPC API：一种新的、Google 开发的基于 HTTP/2.0 的高性能二进制协议，主要用于一些海量用户的高并发请求的场景。

1.3. 小结

在当今应用程序驱动的世界中，创新的一个基本元素就是 API。从银行、零售、运输到物联网、自动驾驶汽车和智慧城市，API 是现代移动端、SaaS 和 web 应用程序的关键部分，企业在面向客户、面向合作伙伴和机构内部的应用程序中随处可见API 的使用。Akamai 的统计报告指出“API 请求已占所有应用请求的 83%，预计 2024 年 API 请求命中数将达到 42 万亿次”。从本质上讲，API 暴露了应用程序的逻辑和敏感数据，如个人身份信息，正因为如此，它越来越多地成为攻击者的目标。

没有安全的 API，快速创新将是不可能的。

2. API 的安全挑战

从 API 的发展过程可了解到，API 安全问题一直伴随着 API 技术的发展而不断变化。API 安全是从安全的角度关注 API 领域的安全问题和这些问题的解决方案，关注的安全领域与传统的 Web 安全比较接近，但又不同于 Web 安全。传统 Web 安全更多的是关注 Web 应用程序的安全性，以服务器端应用程序安全为主，其漏洞表现形式主要为 SQL 注入、XSS、CSRF 等。而新形势下的 API 由于承载了业务逻辑和数据流动，随着微服务和云计算的发展，模块之间越来越独立，每个模块可以根据请求需要动态扩容，原来的服务器边界被打破；攻击面不断扩大的情况下带来了新的安全管理问题和安全技术问题，面临的外部环境比传统的 Web 安全更为复杂。

从云管端的角度来看，API 安全在服务器端包含 API 服务及其运行环境（与传统 Web 安全相似）的安全，管道侧包括 API 消息传输的安全，终端包含 API 客户端应用程序、IoT 设备的安全、监管政策的安全风险等。从安全场景分类的角度来看，API 安全包含了网络安全、Web 应用安全、安全开发、监管合规多个方面。在网络层面，API 安全主要关注客户端与 API 服务器端之间的通信安全；在 Web 应用层面，重点关注 API 客户端与 API 服务器端之间的协议规范、账号的安全、应用安全审计、常见的 API 漏洞以及如何通过 API 安全设计规避这些安全问题；在安全开发层面，从 API 生命周期的角度，结合 SDL 或 DevSecOps 模型来综合

管理 API 开发过程的安全性；在监管合规层面，需要结合法律法规和行业监管要求，考虑 API 数据隐私保护和合规性设计。

2.1. API 防护缺失已成业务和数据安全最大风险敞口

2021 年 IBM Security X-Force 报告中指出，其分析的数据安全事件中有三分之二是由不安全的 API 造成的。据 Gartner 提供的数据显示，到 2025 年，由于 API 的爆炸式增长超过了 API 管理工具的能力，将有 50%的企业出现 API 安全防护缺位，并且有 90%的企业仅能为其公开发布的 API 进行保护，而其它 API 则不受监控，并且大部分企业缺乏 API 安全的实践经验。Gartner 因此预测，到 2022 年，API 滥用将成为导致企业 Web 应用程序数据泄露最常见的攻击媒介，甚至在 2024 年 API 安全问题引起的数据泄露风险将翻倍。

下图是近些年一些典型的因 API 漏洞导致的攻击事件：

事件主体	事件经过
Facebook	第三方应用通过 API 获取 5 千万用户数据， 并用以政治广告投放
Linkedin	因为 API 滥用导致泄漏 7 亿用户的姓名、邮件、手机号码、行业等信息
微博	通讯录匹配查询 API 被撞库，导致 5 亿用户信息泄漏
美国邮政 UPS	因 API 认证漏洞导致 6000 万用户信息泄漏
淘宝	两个 API 的逻辑漏洞导致 11 亿的用户购物信息泄漏
攻防演练企业	OA 系统任意用户登录漏洞、ajax.do 文件上传漏洞、任意文件上传漏洞导致靶标系统被攻破等

API 是数据交互最重要的传输方式之一，也因此成为攻击者窃取数据的重点攻击对象。与此同时，由于 API 防护的缺失，企业对外暴露了哪些 API、对谁开放 API、API 通信中哪些敏感数据在流动等问题都未得到应有的重视。攻击者可以通过 API 认证授权漏洞、数据过度暴露、数据可遍历、安全配置缺陷等攻击 API 进行数据窃取和业务攻击。

2.2. API 面临的主要安全问题

2.2.1.API 资产不可见

大部分企业并没有把 API 资产纳入到资产盘点的范畴，未做好全面的资产梳理工作；而且 API 随着业务的变更也在持续的迭代更新，导致企业对 API 进行安全测评时遗漏了部分资产或长期未对相关应用进行维护。另一方面，API 上流动的数据以及关联的账号，大部分企业也没有将其纳入到资产进行统一的管理和维护，一旦某类 API 框架型漏洞爆发或被黑客入侵时无法及时定位到相关应用节点，将错过最佳的应急响应时间。

资产的可见性是安全的基础，API 由于数量大、更新快、且关联有敏感数据和账号的变更，导致 API 资产很难通过有限人力以静态的方法来完成持续有效的梳理。

2.2.2.攻击面增加

随着云计算技术的广泛应用，越来越多的业务迁移上云，云原生的开发基于微服务架构和 k8s 等弹性扩容模式，相较于传统数据中心的单点调用，API 成为模块间通讯的标准，业务逻辑分散在多个微服务模块，无论东西向和南北向都有无数的 API，每个 API 可能成为一个攻击面，从而导致需要防范的攻击面比原来要大很多。

API 常见的攻击面：

攻击面	说明
认证授权存在漏洞引入攻击面	● 某些 API 在设计之初对身份认证的设计存在不足或者缺失，导致攻击者可以进行未授权或者越权攻击，可以通过 API 任意访问访问数据 ● 权限设计不合理，致使用户 A 可以访问到属于同一角色的用户 B 的数据，出现水平越权访问的攻击；或者普通权限 A 用户可以操作管理员 B 用户的功能，出现垂直越权的攻击 ● 密码安全性校验不足，攻击者可利用弱密码发起攻击，进行账号的破解
输入参数校验不严引入攻击面	在 API 设计和迭代的过程中，研发人员对 API 的入参缺少校验或者校验不严格，可能会被攻击者利用构造的输入来进行注入类攻击如 SQL、XSS、SSRF，或者利用参数遍历与用户身份进行组合带来越权类攻击
设计不合理引入攻击面	● 数据权限，某些 API 在设计时为兼容多个功能会将过多的数据杂糅到一起返回至前端，或者将脱敏和明文数据一起返回，然后由前端去筛选相关的数据。这导致 API 返回过多的数据，攻击者可通过流量拦截等手段获取 API 原始返回的数据，从而存在数据泄漏的隐患

	● 对于登录场景类，API 在出错提示时对错误信息展示的过于详细，攻击者可以利用提示信息来进行撞库扫号攻击 ● API 未对传输数据进行加密设计而直接进行明文传输，攻击者可通过网络嗅探等手段直接获取 API 的交互格式以及数据，通过对获取的数据进行分析，并进行下一步的攻击 ● API 设计时没有考虑到重放逻辑、频率限制、事务完整性逻辑等业务侧逻辑的问题，导致攻击者可以通过重放、修改参数等方式来完成金额修改，篡改交易的攻击 ● 代码实现上存在逻辑 bug，导致攻击者可利用 bug 来进行攻击
安全配置缺陷引入攻击面	● 允许 web 服务器任意目录浏览、未关闭 HTTP 标头配置、没有打开一些认证授权配置开关 ● 未修改业务系统的缺省口令，导致攻击者可使用缺省的口令来进行登录 ● 将内部系统部署在公网
使用易受攻击和过时的组件引入的攻击面	● 开发过程中引入开源或第三方插件、模块、框架等，引用的第三方的软件或模块存在安全问题时，势必会导致代码中的漏洞、恶意代码、“后门”等安全隐患被引入至 API 接口中 ● 使用的开源或第三方组件的版本过低，存在漏洞可以被攻击

2.2.3.API 攻击更加隐蔽

API 是需要开放给用户来使用的，具备开放性和承载业务逻辑的特点，攻击者可以和正常用户一样来调用 API，导致攻击者的流量隐藏在正常用户的流量里面，其攻击行为会更加隐蔽，更难发现。

一些常见的攻击行为：

- 高频访问行为：API 接口在设计之初未对 API 接口访问频率做限制，使攻击者在短时间内可以访问大量 API 接口，很短的时间就可以完成如营销作弊、恶意注册等攻击，甚至可能

带来 CC 攻击。

- 大量数据下载行为：API 接口未对用户某个时段内的下载次数、下载内容大小等做限制，导致攻击者可以通过多次下载达到获取大量数据的目的，容易造成大量敏感数据泄漏。
- 网络爬虫行为：API 接口的开放性，如果没有设置反爬虫安全策略，则攻击者可以使用代理 IP 或修改 User-Agent 请求头隐匿身份，通过信息收集获取企业内部系统账号，利用网络爬虫爬取账号权限以及开放在公网上所有的 API 接口数据，导致大量数据泄漏。
- 动态代理 IP 低频爬虫行为：如果 API 有频率限制和反爬策略，攻击者会还会利用大量的动态代理 IP，低频慢速的方式来绕过现有的防御措施，遍历爬取数据、进行恶意注册或者营销作弊。
- 接口滥用：如果 API 没有对使用者的身份进行校验，缺少白名单检查、缺失验证码进行人机校验的逻辑，攻击者会利用企业的 API 可以任意跳转 URL、任意短信发送等功能来完成攻击，消耗企业的短信费用，给企业带来负面的社会影响。

2.2.4.监管合规性挑战

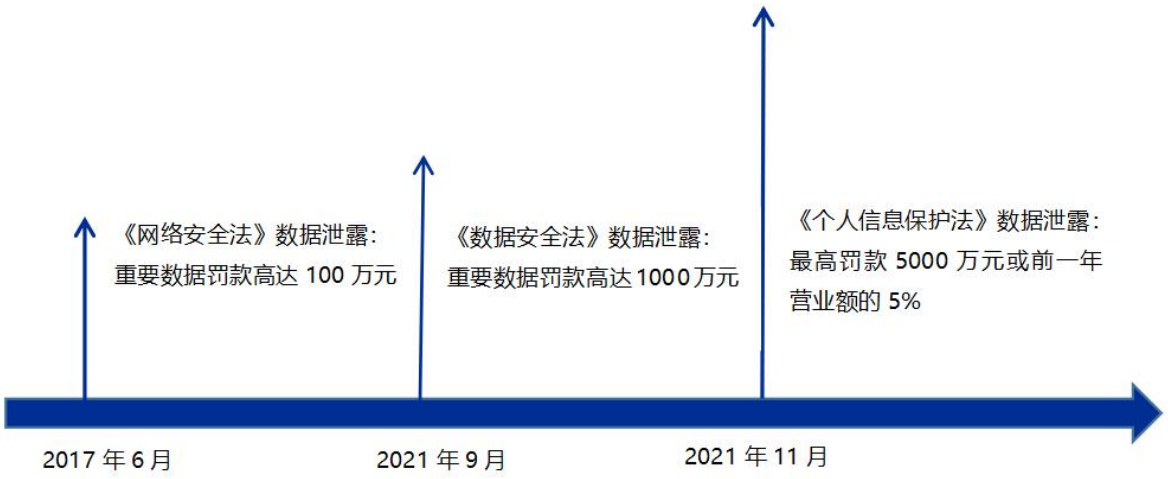
近几年，随着国家层面网络空间治理的不断深入，满足合规性要求成为每一个企业正常业务开展的必要条件。自 2016 年以来，我国陆续出台了一系列的法律法规来监管数据的安全问题，从 2017 年 6 月《网络安全法》的落地实施，再到被称为“数据安全元年”的 2021 年。在 2021 年，《数据安全法》《个人信息保护法》等法律法规陆续正式实施，国家网信办发布《数据出境安全评估办法(征求意见稿)》并公开征求意见。从这一系列法律法规中可以看出，我国对企业的数字安全监管确实在走向更加严格和规范化，聚焦的内容更加细化，处罚的力度也逐渐加强。对企业而言，围绕数据的全生命周期从数据采集、存储、访问、使用、销毁构建数据安全的体系成为重点的安全建设工作。

目前，大多数企业在数据采集、存储、数据库访问方面做了比较全面的安全建设，在数据的流动访问方面的安全建设的意识也正逐步萌芽和发展；而 API 作为连接数据与应用的主要通道，成为了数据传输中最薄弱的环节之一，传统的 WAF、IPS 类安全设备关注点不在数据安全，API 这个点目前的安全防护比较薄弱，所以 API 很容易成为攻击者眼中窃取数据的头号目标。

金融行业标准 JR/T 0185-2020《商业银行应用程序接口安全管理规范》中，更是从 API

类型与安全设计、开发、部署、集成、运维等生命周期角度，对 API 的管理提出多方面的合规性要求。这些标准或规范为企业的 API 安全实践提供方向性指引，同时也为 API 的合规提供可落地标准。企业完成了此类合规的挑战，才能更好地开展业务。

下图所示为这几年来的数据相关法规的处罚细化说明：



2.3. 小结

从攻击视角来看，当越来越多的企业通过 API 对外开放业务能力，意图共建生态时，账号、营销、数据方面的安全漏洞可以被攻击者直接快速获利，而且当前企业在这块的安全防护意识才刚刚开始萌芽，这种新型的攻击面充满诱惑。攻击者的动机越强、攻击手段越多、造成的危害越强，给企业的防御带来的挑战就会越大。

3. API 全生命周期安全防护

由于云计算的快速发展，越来越多的企业将应用和数据迁移至云端，并暴露核心业务能力和流程相关的 API 为外部合作伙伴提供服务。脱离了传统的内网或网络区域划分，云上应用的开发和集成、云端管理 API，被潜在的商业合作伙伴及攻击者使用，无形中使得 API 安全风

险增大。

对大多数企业而言，很难完全掌握系统全部 API；开发人员往往也只是熟悉自己开发的相关模块，且很多技术开发人员认为采纳新的、酷的技术更重要，在技术路线上选择新的特性，忽视 API 是否被攻击。在这种缺少 API 安全性管理平台又未建立全面系统的 API 安全治理体系的情况下，API 安全风险更不可控。

3.1. API 安全设计的指导原则

安全架构设计有很多的安全设计原则，比如公开设计原则、权限最小化、开放最小化、默认不信任等。安全设计原则需要不断的学习和培训，让安全设计人员和研发都能融会贯通。API 作为业务系统新的边界，从设计的角度来保障好 API 这个新的边界，有两个基本的原则可以参考，分别是 5A 原则和纵深防御原则。

3.1.1.5A 原则

5A 原则是指 Authentication（身份认证）、Authorization（授权）、Access Control（访问控制）、Auditable（可审计性）、AssetProtection（资产保护）5 个部分的首字母缩写，其含义是当安全设计人员在做安全设计时，需要从这 5 个方面考量安全设计的合理性。如果某一个方面缺失，则在安全设计上是不全面的。

- **身份认证，解决“你是谁”的问题，**目的是为了知道谁在与 API 服务进行通信，是否是 API 服务允许的客户端请求。一些需要权限才能访问的 API 服务，需要知道谁在请求，是否允许请求，以保障 API 接口调用的安全性。认证方式主要有用户名/密码认证、动态口令、数字证书认证、生物特征认证等。对于安全性的要求比较高的业务，可使用双因子认证（2FA）或多因子认证（MFA）。常见的组合有用户名/密码+短信挑战码、用户名/密码+动态令牌、用户名/密码+人脸识别、人脸识别+短信挑战码等。认证通常融入单点登录 SSO 系统中，使用统一的入口来完成身份认证，减少攻击面。
- **授权，解决“你可以访问什么数据和资源”的问题，**通常发生在身份认证之后，即对服务来说，谁在请求我，这个请求是有权限的么？某些 API 只有特定的角色才可以访问，比如只有内网的 IP 才可以调用某些服务、只有管理员用户才可以调用删除用户的 API。

- **访问控制，解决“你所访问具体的数据和功能是否被允许”的问题，**通常发生在授权之后，很多情况下，对于某个角色的权限设置正确，但访问控制做的不一定正确，这也是存在很多越权操作的原因。访问控制是对授权后的客户端访问时的正确性验证。某个角色，对于不具备访问权限的 API 却可以直接调用，问题就出在访问控制上。授权和访问控制常常是一起来进行的，有两种方式可以参考：一是基于使用者身份代理的授权与访问控制，典型的以 OAuth 2.0 协议为代表，对于 API 的授权和可访问资源的控制依赖于使用者的身份，使用者可能是某个自然人用户，也可能是某个客户端应用程序，当得到使用者的授权许可后，即可访问该使用者授权的资源；另一类是基于使用者角色的授权与访问控制，典型的以 RBAC 模型为代表，对于 API 的授权和资源访问依赖于使用者在系统中被授予的角色和分配的权限，不同的角色拥有不同的权限，比如功能权限、数据权限，访问资源时依据此角色分配的权限的不同可以访问不同的资源。
- **可审计性，解决“你所做的操作能够被溯源”的问题，**目的是为了记录 API 调用时的关键信息，以便事后能够通过审计手段及时发现问题，并在发生问题时通过审计日志进行溯源，找出问题的发生点。一般记录 API 日志需要有：什么人（账号、UA 信息）、在什么时间、利用什么 IP（在什么地方）、调用了什么 API（做了什么）、操作的结果是什么、操作 API 时的 Referer 是什么等。
- **资产保护，解决“阻止 API 被滥用”的问题，**主要是指对 API 接口自身的保护，比如限速、限流，防止恶意调用，以及对 API 接口传输的敏感数据如身份证、手机号码、银行卡号等的保护。

3.1.2.纵深防御原则

纵深防御这个词来源于军事术语，是指在前方到后方之间，构建多道防线，达到整体防御的目的。在网络安全领域，纵深防御通常是指不能只依赖单一安全机制，建立多种安全机制，互相支撑以达到相对安全的目的。可以通过一个生活中的例子来理解纵深防御原则的基本含义，比如坐飞机的这个过程中，机场采用了如下这些防线：第一道防线是入口的防爆检查，可快速的检查乘客是否携带了防爆物；第二道防线是安全检查，检查是否携带了一些违禁品；第三道防线是上机前身份校验，确保人票一致性。这三层的防御就构成了纵深防御，确保是购买了机票的人在安全的情况下乘机。

在 API 安全设计中，可以在不同层面使用不同的安全技术，来达到纵深防御的目的。典型的场景如网银的转账业务，进入系统时需要进行登录进行身份认证，在后面的调用转账 API 时，仍需要再次输入密码，甚至对于大额转账还需要进行人脸认证。网银登录时的身份认证和转账时的身份认证，相互之间就构成了纵深防御原则。

5A 原则重点强调每一层安全架构设计的合理性，强调的是宽度；纵深防御是对同一问题从不同的层次、不同的角度做安全防护，强调的是深度。这两个原则相结合，共同将安全设计构成一个有机的防护整体。

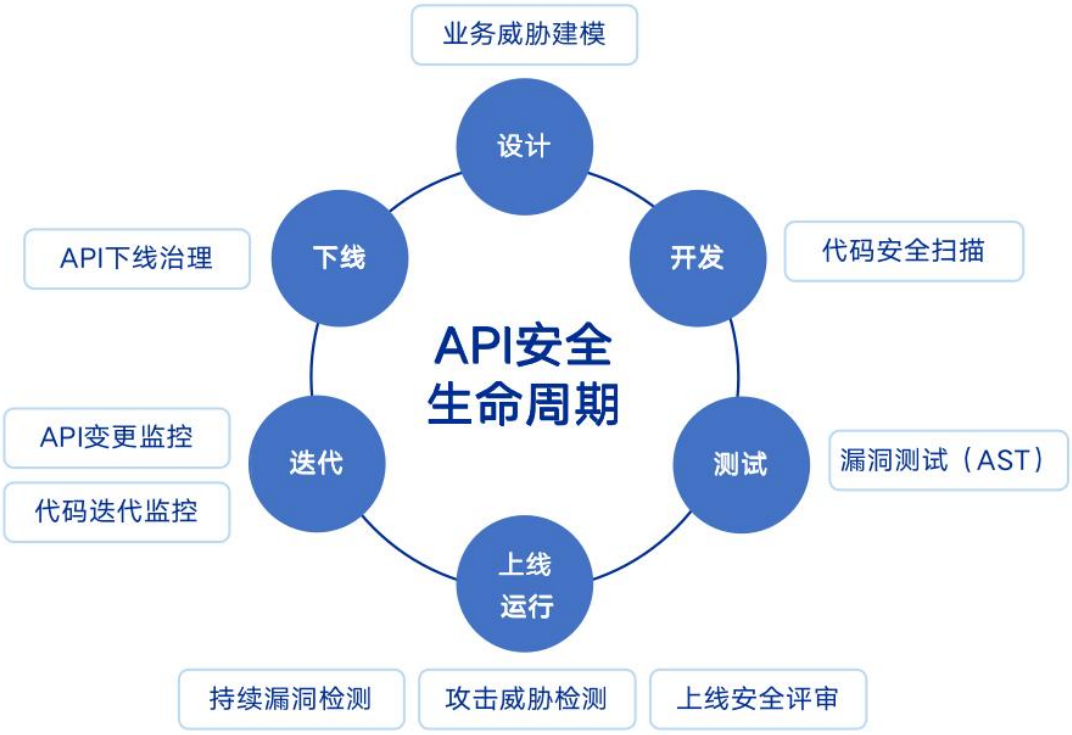
以下是将两个原则结合起来的 API 安全整体示意图：

3.2. API 生命周期的安全防护模型

从 API 面临的外部威胁来看，主要有如下方面：命令执行/SQL 注入/服务器接管等高危漏洞的风险、权限管控不完善的未授权和越权访问的风险、设计存在缺陷带来的数据批量泄漏的风险等。接入 WAF 可以解决部分命令执行/SQL 注入类问题，但目前市场上的 WAF 产品因 API 技术的特殊性对 API 安全的防护能力仍显不足，其主要表现在以下几个方面：一是认证和授权流程的绕过，API 的认证和授权流程很多互联网应用是基于 OAuth2.0 和 OpenIDConnect 去实现的，传统的安全防护产品难以检测业务流程绕过的威胁；二是数据格式难以识别，API 在交互过程使用的消息格式大多 JSON 格式、XML 格式、Protobuf 格式、JWT 格式等，在威胁检测时需要深入这些数据格式的数据结构内容去分析，传统的安全防护产品在此方面检测能力比较弱；三是流量控制能力难以满足业务需求，面对 API 层面的 CC 攻击、慢 BOT 攻击时，传统上使用的检测和防护策略，如访问频率限制、IP 黑名单设置、二次验证机制等难以对新型攻击起到很好的防御效果。

针对 API 存在威胁防护，使用 WAF 类产品只能覆盖其中的一小部分威胁，对业务而言，

从单点考虑 API 功能安全设计到通过对 API 生命周期来考虑 API 的安全，围绕设计、开发、测试、上线运行、迭代到下线的每一个环节加强安全建设就更加有必要。全生命周期来考虑 API 的安全性，通过安全左移方法和工具，综合性地融合管理手段和技术手段进行 API 安全治理，提高业务 API 的整体的安全性。



3.2.1.设计阶段：引入威胁建模

在设计之初以及开发过程中，根据业务特点对风险进行评估，做到可靠安全设计。安全工程师参与到对需求和设计方案的实现中来。资源允许的情况下，安全工程师在设计阶段对每个 API 进行威胁建模，线上的 API 风险将会大大减少。资源有限的情况下，建设自动化威胁建模的能力，采用“重点业务人工评审”和“非重点业务自动化威胁建模”相结合的方式，对核心高风险 API 如账号登录、文件上传、营销活动等进行覆盖。

在设计阶段的威胁建模，一般使用 ASTRIDE（隐私、欺骗、篡改、信息泄露、否认性、拒绝服务和特权提升)和攻击树模型作为常用的威胁建模技术指导原则。结合业界安全白皮书、历史上安全事件总结，根据业务 API 的需求和功能设计，基于上述 5A 的原则整理出来潜在攻击者可能进行攻击的目标和方式。

对于 API 而言，攻击者通常可能会有：恶意内部员工、外部攻击者，竞争对手、好奇者等，攻击路径可能是下班后通过内部系统 API 盗取数据、利用 API 的逻辑漏洞批量爬取数据、发起 BOT 攻击、借助手机号接码平台发起恶意注册、利用代理秒拨平台发起低频慢速的攻击等。

下面列表的一些安全检查表和最佳实践可以作为威胁建模阶段的一些参考：

OWASP REST 安全检查表	https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html
apisec 提供了 API 安全工具和资源	https://github.com/arainho/awesome-api-security
API 安全检查表	https://github.com/shieldfy/API-Security-Checklist
JWT 安全检查表	https://pragmaticwebsecurity.com/files/cheatsheets/jwt.pdf

3.2.2. 开发阶段：安全开发意识和规范培训，引入安全工具

比较理想的情况，同研发在设计阶段就威胁建模进行了讨论，大家对于 API 可能遇到的安全问题点都有了全面且清晰的理解；然而，在研发进行编码实现的环节，往往可能在实现上出现不完整，或者引入新的 bug。同时考虑到研发人员也在不断的变化，需要安全工程师提供 API 安全开发规范、安全开发插件、安全辅助包、敏感数据加解密工具等辅助性安全开发工具。一方面通过培训的方式加强全员对安全开发的意识和能力，一方面需要借助工具的方式来实现自动化的检查，提早发现 bug 和漏洞。

在 API 安全开发培训方面，可以从四个方面来考虑：

- API 安全管理框架和关键指标，企业在 API 安全这块整体的框架，定型和定量的指标，如上线后的漏洞数量等。
- 常见 API 安全问题，如 OWASP API Top 10 ，以及安全运营通过 SRC 以及业务中提炼出来的一些具有代表性的问题。通过问题案例的方式，更加能让大家理解和学习。
- 常见 API 安全技术与安全设计，整理收集业界和企业自身的一些优秀安全实践案例，能开拓研发的思维，在遇到类似问题时能想到更好的解决方法。
- API 安全编码案例，结合业界的优秀案例，基于企业自身的特点制定出来符合企业特色的安全编码的规范和案例。

下面列表的一些安全编码和开发规范，可以借鉴和参考：

OWASP 安全编码实践	https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated_content
腾讯的代码安全指南	https://github.com/Tencent/secguide
永安在线 API 安全开发规范	https://www.yazx.com/reportDetail/14ac36b8-f5c6-11ec-af75-00163e048a4c

在自动化工具方面，可以考虑如下几个方面的工具：

- 引入代码白盒代码扫描工具，嵌入到 CI/CD 或者研发流程中，发布到测试环境时就进行扫描。
- 引入 API 管理工具，对 API 的文档进行集中统一的管理，能够监控到 API 的变更迭代的过程数据。
- 引入 API 安全验证相关的工具，验证 API 安全实现的正确性，以保障验证工作尽可能做到全面，相关工具如 FuzzDB、个人数据隐私监测类工具。

3.2.3.测试阶段：漏洞加入测试流程，使用 AST 类工具提高覆盖率

在测试环节加入 API 安全相关的内容，针对自动化测试流程和人工测试，在 API 业务逻辑实现、稳定性、性能测试的基础上增加 API 安全的测试。落实 API 安全测试的目的是为了自动化扫描每一个 API，自动化 API 安全测试从请求输入与应答响应两部分去分析 API 是否存在漏洞。API 安全测试的常规内容主要包含 API 身份验证、授权、输入验证、异常处理、数据保护、安全传输以及 HTTP Header 安全性等。

API 数量多，且在不断的迭代更新，需要借助自动化工具来辅助进行安全测试。常用的工具有以下三类：SAST、DAST、IAST 工具，在测试阶段提前发现研发的安全漏洞。根据业务的特点，和供应商一起优化 IAST 工具，保障覆盖率的情况下，提高准确率，能提前发现许多输入处理不当导致的漏洞。

- 静态安全检测（Static Application SecurityTesting，SAST），其特点是分析应用程序的源代码或二进制文件，通过语法、结构、过程、接口等来发现应用程序的代码是否存在漏洞。

- 动态安全检测（Dynamic Application SecurityTesting，DAST），其特点是在应用程序的动态运行状态下，模拟黑客攻击行为，分析应用程序的响应，而确定应用程序是否存在漏洞。
- 交互式安全检测（Interactive ApplicationSecurity Testing，IAST），相当于是 DAST 和 SAST 结合的一种安全检测技术，通常会在应用程序中添加探针或 Agent 代理，收集应用程、Web 容器、JVM 中的执行日志和函数调用信息，结合请求输入与响应消息，分析应用程序中是否存在漏洞。

这三类工具中，IAST 使用过程稍显繁琐，但技术优势比较明显，漏洞检出率高于其他两类，同时漏洞误报率也低于其他两类，并可以快速定位代码片段和 API 接口，可以作为首选的自动化 API 安全测试工具，如果没有此类工具，则建议选择 DAST 类。

3.2.4. 上线运行阶段：借助网关、WAF 和流量审计工具提早感知攻击面

经过前期的开发和测试，API 终于到了上线的环节，可根据业务评估是否需要对上线环节进行安全评审，API 上线后，就进入了运行阶段。对外暴露的 API 承载了业务逻辑和数据流动，成为攻击者攻击的主要通道。因此，需要借助各种安全工具来保障运营阶段的 API 安全。

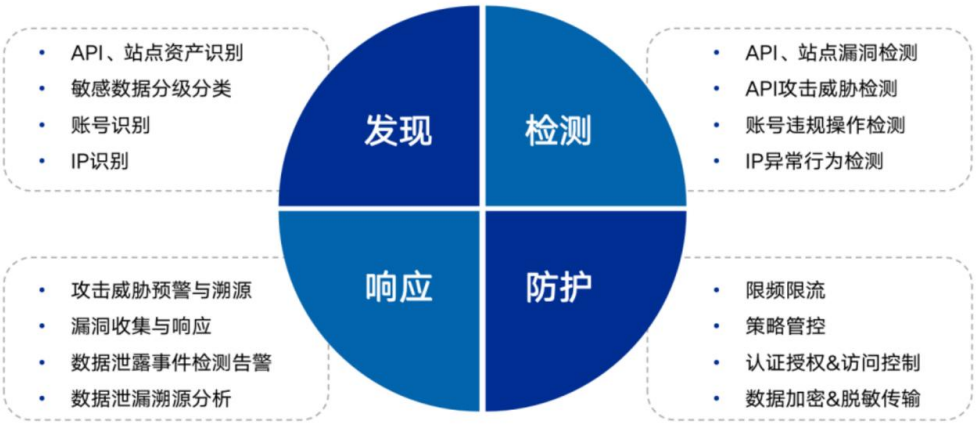
3.2.4.1. 上线阶段常用的安全工具

- 利用 WAF，阻断 SQL 注入、XSS 等攻击请求、防御 DDoS 防御、常规的 BOT 攻击。外部攻击流量过了 WAF 的流量检测，对恶意漏洞扫描的行为完成了过滤，对后端系统的危害性就会降低。同时，运维人员通过 WAF 的数据与日志，可以定向分析异常行为，在 WAF 上调整安全防护策略，达到快速阻止攻击的目的。
- 利用 API 网关，实现认证、授权、访问控制、数据脱敏，同时也能帮助安全团队来管理 API。虽然 API 网关产品中所具有的身份认证、访问控制、数据校验、限流熔断等功能，能有效地提高 API 的安全性。但是对内部开发人员来说，需要打通持续集成（CI/CD）与 API 网关的接口，发布前准备好 API 导入数据供 CI/CD 调用。这会增加开发人员的额外工作量，并影响发布进度；同时，当所有后端服务的流量都必须由 API 网关进行通信时，对原有通信性能的影响和 API 自身的稳定性，将是对推进此项工作的负责人员的最大挑战。可以根据业务的实际情况来评估是否需要使用 API 网关。

- 利用 API 安全审计工具，对 API 分类分级，标示出来高风险 API，如涉及敏感信息出站的、涉及资金的、涉及核心基础设施操作等；持续监测 API 数据暴露、越权、配置、设计不合理带来的漏洞；发现僵尸 API、影子 API，老版本、功能重复的 API；通过 UEBA 发现利用自身权限批量获取敏感数据的行为等；借助情报和机器学习模型来发现针对 API 的低频慢速的攻击。

3.2.4.2. 基于 P2DR 模型的自适应安全闭环

工具只是手段，在安全模型的指导下能充分利用其功效。在上线运营阶段需要基于动态自适应的 P2DR 安全模型，对 API 攻击威胁实现〝发现〞、〝检测〞、〝防护〞、〝响应〞的闭环，可以更好满足数据安全体系下的 API 安全防护需要。安全的前提是资产的可见性，资产可见后才能实现可控，可控包括两个方面，一个方面是资产漏洞风险的持续监测，一个方面是攻击威胁的感知和防护。



1）持续构建资产发现能力

API 发现能力是 API 提供者和攻击者之间的竞赛，要在攻击者之前发现 API，提前感知和了解到系统可能的攻击面。因为 API 是不断在研发迭代的，所以持续动态梳理 API 的能力显得至关重要。可以通过 API 安全网关、负载平衡或交换机镜像的网络流量中提取出来 API、API 上流动的数据、访问 API 时关联的账号和 IP 等资产。同时对发现的资产进行分级分类处理，分级分类可以让安全运营的人分优先级的来进行治理。

- API 可以根据业务场景，如登录、文件上传、文件下载、第三方开源组件等进行分级分类；也可以根据返回的数据敏感度的情况来进行分级分类。

- 流动的数据，可以根据国家的监管法规规范来进行分级分类，同时建立数据到 API 的映射关系，这样可以从容应对法规和监管的治理需要。
- 账号资产，可以根据权限级别、访问系统、活跃性等来进行分级分类。
- IP 资产，可以根据地域、类型、安全性等来进行分级分类。

API 的资产发现要全面，从 API 的应用场景的角度，围绕终端用户、合作企业、内部员工、以及开源组件和中间件四个场景能比较全面的覆盖 API 资产，借助负载均衡或者核心交换机的的流量来梳理 API 能够实现对上述四个场景比较全面的覆盖。

面向终端用户	面向合作企业	面向内部员工	面向开源组件和中间件
在互联网上给到用户使用的 APP、Web、小程序等用到的 API	在互联网上开放给到合作企业客户使用的业务 API	开放给到内部员工或者合作伙伴使用的应用系统上关联到的 API	使用到的 clickhouse、spring boot、k8s、hadoop、jenkins 等涉及到的 API

2）持续的风险检测能力

API 安全的风险检测包括了 API 自身的漏洞风险，也包括了攻击者对 API 进行攻击的风险、账号的违规操作行为风险、IP 的异常行为风险等。需要持续针对资产的漏洞风险、攻击行为和异常行为风险进行检测，在保障召回率的前提下，提高准确率。

漏洞风险的检测，有如下几个方面：

- 自身业务漏洞检测，需要持续监测 API 存在的授权、认证、数据过度暴露、配置、逻辑设计等方面的缺陷；以及关联账号的弱密码这类风险的监测。
- 第三方组件漏洞检测，企业引入的第三方开源组件或者中间件同样也会存在漏洞，需要持续对供应链上的 API 进行漏洞的监测。

攻击行为和异常行为的检测，有如下几个方面：

- API 攻击威胁检测，攻击者利用 API 的逻辑漏洞来实现数据爬取、账号注册、爆破、撞库、短信轰炸、任意网址跳转、事务攻击等恶意的攻击行为， 在 API 运行时加强外部风险感知能力和风险阻断能力的建设尤为重要，这样才能够及时准确地感知到 API 的任何滥用情

况，并及时阻断攻击者的进一步行动。基于风险情报来监测攻击行为，在召回率和准确率上都会有比较好的表现。

- 账号违规操作行为检测，攻击者可以利用内鬼来盗取敏感数据，也可以借助撞库、账号暴力破解、社工钓鱼等方式获得敏感的账号，进而利用授权账号进行违规的操作。账号违规操作行为的检测，需围绕账号的历史行为以及账号和同组织的人在登录环境、登录 IP、获取数据的时间、访问站点和数据的情况等行为方面构建基线，基于 UEBA 的模型来监测账号违规操作的行为。
- IP 异常行为检测，攻击者的攻击都需要通过 IP 资源来完成，针对 IP 构建历史行为画像如数据访问、地理位置、API 访问序列、风险情报画像等，基于机器学习的方法检测 IP 的异常行为，如路径扫描、漏洞扫描、单一 API 请求过多、境外 IP 获取敏感数据等异常行为。基于 IP 风险情报来构建的检测模型，准确率高，可解释性强。

3）持续构建防护能力

- 在 5A 模型和纵深防御原则的指导下，借助 WAF 和 API 网关等安全产品，通过对认证授权、访问控制、信息传输保护、限流限速、旁路检测联动等方式持续构建防护能力，应对 DDoS、CC、BOT 攻击，防止资源消耗在无意义或恶意的 API 请求上。
- 数据安全防护能力，企业应对敏感信息的交互进行加密和脱敏处理，特殊的敏感信息需要被加密或做脱密处理，例如身份证号、银行卡号、手机号等，以减少敏感信息的泄漏风险。

4）持续构建响应能力

- 漏洞管理与响应，企业应构建漏洞管理与响应机制，借助 SRC、渗透测试、基于旁路流量的 API 安全工具持续挖掘、收集 API 相关漏洞，做好补丁管理，应对不断变化的攻击面及隐秘多变的攻击手段。
- 攻击威胁预警和溯源，企业需要借助风险情报和大数据分析技术，对 API 的访问日志进行分析，挖掘出里面的攻击流量，并对攻击流量进行解释和溯源，进而做到及时预防和响应。可从业务流量中细化攻击的行为结合攻击者历史情报信息对攻击者的资源、手法、意图、团伙情况、潜在目标进行关联跟踪分析，进而构建攻击者情报库，对攻击者进行持续跟踪。
- 数据泄漏预警和溯源，监控暗网和数据交易平台，监测是否存在企业数据信息贩卖的情况。

基于暗网监测数据，进行脏数据顾虑、二手信息过滤、结合数据售卖者的置信度对交易数据准确性和及时性进行分析，提炼出来准确的数据泄漏预警信息，及时预警数据泄漏的事件。同时，利用大数据日志平台，对泄漏的数据进行追踪溯源，找到泄漏的源头，针对性的来进行修复，避免大规模的数据泄漏事件发生。

对于 API 安全防护，持续自动化的防护是关键，整体可参考如下图所示的 Gartner 提出的 API 安全三步法，结合上面介绍的方法、手段和工具来实现 API 上线运行阶段的安全防护。

3.2.5.迭代阶段：利用安全工具及时审计 API 变更

随着业务需求变化，API 的实现逻辑发生改动是很常见的现象，API 实现逻辑发生变化，很容易引入新的风险，特别是当开发人员认为前期已经做过非常多的安全检测后，不太可能再出问题时。所以在这个阶段的安全也是需要关注的，借助安全工具可以感知到 API 的更新变化，提早进行治理。

- 利用 API 安全工具，及时感知 API 资产的变化，针对 API 出入站数据的变化，如新增，组合异常进行监控告警，通知安全工程师重新介入进行安全的评审。
- 监控代码仓库变更，当发现 API 方法代码发生变更时，通知安全工程师重新介入进行安全的评审。

3.2.6.下线阶段：及时下线僵尸影子 API

大量的 API 在完成自己的使命后，没有被及时下线，不仅会浪费系统资源，还会变成潜在的线上风险。例如一个 API 存在越权漏洞，但是因为没有流量一直没有被发现，直到有一天被攻击者利用。在这几个阶段，进行如下两个方面运营工作：

- 利用安全工具，维护动态 API 资产台账，识别哪些 API 是应该下线的，如僵尸 API，老版本 API、功能重复的 API，推动业务及时下线。
- 标记出来一年中只会使用几次的 API，如招聘 API、营销活动 API，这部分要单独维护。

3.3. 小结

围绕 API 全生命周期进行安全的设计和运营，需要企业的各个角色参与进来，投入的工作量是极大的。相比产品和研发人员，安全人员的占比很小，企业可根据自己的业务情况来调整在不同环节的投入情况。

为了追求高效的防御，建议从 API 的上线运行阶段入手，基于风险情报对 API 的流量进行分析，持续实现 API 和数据资产的梳理、漏洞的检测、攻击威胁的检测，一方面可以摸底清楚资产的情况，一方面可以及时预警风险并止损，从而给到安全人员有更多的战略时空资源来进行安全左移建设，逐渐实现 API 全生命周期的防护。

结束语

随着企业数字化和在线化进程推进，API 安全技术也得到大力发展，构建 API 安全需要包含 API 发现、敏感数据分级分类、账号资产发现和异常行为检测、API 身份与访问控制、API 消息安全、API 传输安全、API 威胁检测、API 安全审计、API 监测与跟踪、API 管理等多个方面。

为筑牢 API 安全基础，从实战化角度进行 API 安全防护体系的建设，需要将安全落实到 API 全生命周期的各个环节，进而构建具有更好可见性和可控性的 API 安全防护体系。