云应用程序编程接口安全研究综述：威胁与防护

陈 真*①③ 乞文超① 贺鹏飞① 刘林林② 申利民①③

①(燕山大学信息科学与工程学院 秦皇岛 066004)

②(中国科学院文献情报中心 北京 100190)

③(河北省计算机虚拟技术与系统集成重点实验室 秦皇岛 066004)

摘 要：云时代，云应用程序编程接口(API)是服务交付、能力复制和数据输出的最佳载体。然而，云API在开放服务和数据的同时，增加了暴露面和攻击面，攻击者通过数据劫持和流量分析等技术获取目标云API的关键资源，能够识别用户的身份和行为，甚至直接造成背后系统的瘫痪。当前，针对云API的攻击类型繁多，威胁与防护方法各异，缺乏对现有攻击和防护方法的系统总结。该文梳理了云API安全研究中云API面临的威胁和防护方法，分析了云API的演化历程和类别划分；讨论了云API的脆弱性以及云API安全研究的重要性；提出了云API安全研究框架，涵盖身份验证、云API分布式拒绝服务(DDoS)攻击防护、重放攻击防护、中间人(MITM)攻击防护、注入攻击防护和敏感数据防护6个方面相关研究工作综述。在此基础上，探讨了增加人工智能(AI)防护的必要性。最后给出了云API防护的未来挑战和发展趋势。

关键词：云应用程序编程接口；云API脆弱性；云API安全；云API攻击；云API防护

中图分类号：TN915.08; TP309 **文献标识码**：A **文章编号**：1009-5896(2023)01-0371-12

DOI: 10.11999/JEIT211185

A Survey for Cloud Application Programming Interface Security: Threats and Protection

CHEN Zhen①③ QI Wenchao① HE Pengfei① LIU Linlin② SHEN Limin①③

①(*School of Information Science and Engineering*, *Yanshan University*, *Qinhuangdao* 066004, *China*)

②(*National Science Library*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)

③(*Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province*, *Yanshan University*, *Qinhuangdao* 066004, *China*)

Abstract: In the cloud era, cloud Application Programming Interface (API) is the best carrier for service delivery, capability replication and data output. However, cloud API increases the exposure and attack surface of cloud application while opening up services and data. Through data hijacking, traffic analysis and other technologies, attackers can obtain the key resources of the target cloud API, so as to identify the identity and behavior of users, or even directly cause the paralysis of the underlying system. Currently, there are many types of attacks against cloud APIs, and their threats and protection methods are different. However, the existing researches lack a systematic summary for cloud API attack and protection methods. In this paper, a detail survey on the threats and protection methods faced by cloud API is conducted. Firstly, the evolution and the classification of cloud API are analyzed. The vulnerability of cloud API and the importance of cloud API security research are then discussed. Furthermore, a systematical cloud API security research framework is proposed, which covers six aspects: identity authentication, cloud API Distributed Denial of Service (DDoS) attack protection, replay attack protection, Man-In-The-Middle (MITM) attack protection, injection attack protection and sensitive data protection. In addition, the necessity of Artificial Intelligence (AI) protection for cloud API is discussed. Finally, the future challenges and development trends of cloud API protection are presented.

收稿日期：2021-10-28；改回日期：2022-04-29；网络出版：2022-05-08

*通信作者： 陈真 zhenchen@ysu.edu.cn

基金项目：国家自然科学基金(62102348, 61772450)，河北省自然科学基金(F2019203287)，河北省教育厅高等学校科技计划(QN2020183)

Foundation Items: The National Natural Science Foundation of China (62102348, 61772450), The Natural Science Foundation of Hebei Province (F2019203287), The Science and Technology Research Project of Hebei University (QN2020183)

Key words: Cloud Application Programming Interface (API) ; Cloud API vulnerability; Cloud API security; Cloud API attack; Cloud API protection

1 引言

云应用程序编程接口(Application Programming Interface, API)是一种预先定义的网络接口，通过公共互联网和虚拟私有云网络(Virtual Private Cloud, VPC)为应用程序、开发人员提供访问一组例程和数据的能力，而无需访问源码和理解API内部工作机制。更宽泛地说，云API是一组协议、功能、机制、工具和定义，用于跨不同领域共享和开发新的服务，进而使分布在不同的领域的服务更具延展性和伸缩性。

数字化时代，云API已成为智能交互、能力开放和数据传输的最佳载体。2020年《中国人工智能API经济白皮书》[1]指出“云API在产业链的快速拆分重组、行业分层细化、加速融合创新的过程中发挥了极大作用”。云API支持服务集成、应用程序开发以及不同服务和产品之间的通信，不再需要为每个服务单独开发新的基础设施。这主要得益于其调用灵活，传输时延短，既降低了云API提供者与使用者的交流沟通成本，又可以快速迭代，更新或重建业务。

云API在带来便利的同时安全风险也很突出，其更广泛的暴露面让针对云API的攻击成为主流，网络安全行业也未能及时更新来保障爆炸式发展的云API的安全，到目前为止对于云API的安全意识和防范措施还有着明显的缺失。云API扩大了攻击者的攻击范围，却常常未受到传统安全防御的足够保护。典型的云API防护仍致力于通过身份验证解决方案和云API网关来限制对云API的访问。这类防护措施还没有从老旧的访问控制模型进化到全面的解决方案，要想实现对数据、应用和系统的全方位防护，仍需要解决包括云API特定登录攻击、拒绝服务攻击、重放攻击在内的多种威胁。

当前基于云API的攻击类型繁多，威胁与防护方法各异，目前还没有一个关于现有攻击防护方法的系统性总结。针对该问题，本文通过与传统Web威胁相对比，进一步阐述了基于云API的特定攻击原理与对抗防御方法，并对未来技术发展方向进行了展望。本文的贡献主要有：

(1)讨论了云API的脆弱性，为云API安全的相关研究提供了理论基础。

(2)与传统Web场景进行对比分析，对多种类型的云API威胁与防护进行归纳总结，方便后续研究。

(3)在传统云API防护的基础上，讨论人工智能(Artificial Intelligence, AI)防护的重要性，提出未来的改进方向。

本文在第2节介绍云API的演变过程与分类，第3节给出云API的脆弱性以及安全研究的必要性，第4节对每种类型的威胁与防护进行分类分析，第5节着重介绍AI防护，最后一节对云API安全未来的挑战与可能的方向进行总结。

2 云API演化历程及分类

2.1 云API演化历程

在早期进行数据处理时，开发者为了实现数据资源之间的交互，就曾试图设计可公开访问的“接入点”，这也是API概念的由来。但是直到分布式系统的普及，网络的全方位覆盖，这些基础概念才发挥出了它本来的效力。在1996年面向服务架构(Service Oriented Architecture, SOA)被提出之后[2]，API收获了来自各种大小网站的关注。在2000年左右，API以一种相对正式的形式发布，随后很多技术先驱着手为商业交易和服务管理定义了早期的API，从此开始了一场长达数十年之久的变革[3]。

软件工程过去50年的发展历史表明，云API是解决软件危机问题、促进软件产业工程化和产业化的切实可行的途径。如图1所示，云API自经历了服务电商、社交、云计算、移动时代之后，已经深刻改变了软件的形态、开发范式和运行模式[4]。随着SOA理念的深入发展，云API作为实现SOA的新技术逐渐取代了传统基于简单对象访问协议的Web服务技术，并且凭借着轻量级、易访问和易组合等优势将迄今为止一直隐藏在企业与组织背后的深层计算能力与大数据提供给合作伙伴和对其感兴趣的用户，促进了云API经济的形成与繁荣。当前，云API已经逐步成为面向服务体系结构系统开发和应用的事实标准。

2.2 云API应用体系结构

为清晰地讨论云API的安全问题，将云API的应用体系结构分为应用层、API层、数据处理层、数据资产层和统一资源管理层。其中，应用层主要包括智能设备、云/Web应用、移动应用；API层按照服务对象可划分为服务提供商内部的API、对外开放API、服务提供商之间的API；数据处理层主要是对数据的建模，如特征提取、行为分析等建模方法；数据资产层由具体的产品数据构成，主要

包括各类算法以及用户位置、昵称等云数据资产；统一资源管理层体现了云API开放性、虚拟化的特点，对不同来源的云数据实施资源管理、审计核查、模型维护。其总体结构如图2所示。云API的安全管理涉及以下4个关键环节。

云API的应用层场景十分丰富。服务提供商利用云API对外发布可嵌入其他智能设备的组件、构建低耦合的Web应用和开发跨平台的移动应用。例如百度通过云API提供语言翻译服务，阿里使用云API支持在线支付服务。在应用层智能设备的管理、应用数据的安全上传和服务化软件开发的新技术接入等过程中，通常采用云API的加密与认证、API流量的机器识别来保障云API的数据完整性和算法安全性。

如何保障API的数据传输、防范深度威胁是API层安全管理面临的主要问题。云API作为大量业务的接入点，面临着各类入侵风险，如应用协议漏洞、自动化工具造成的爬虫攻击、API身份攻击。实时监控云API入侵事件，加强对云API的漏洞扫描，防止云API滥用现象的发生。

数据处理层与数据资产层的安全管理涉及云API敏感数据识别、敏感数据脱敏与防护，API资产自动识别、梳理与分组，防止因语义异常或资源耗尽引起的数据泄露问题。

统一资源管理层是服务提供商通过云API向合作伙伴开放共享已有的数据资产，构建统一的资源管理平台，实现用户体验的优化。在此过程中，由于云API面向的是不同的服务提供商，在防护过程中要注意内部员工的越权行为、云账号行为审计，API异常访问控制等。

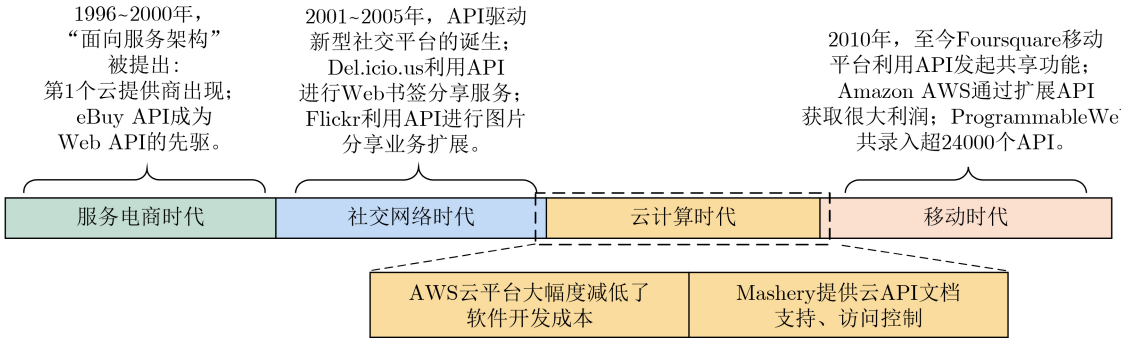


图 1 云API演化历程

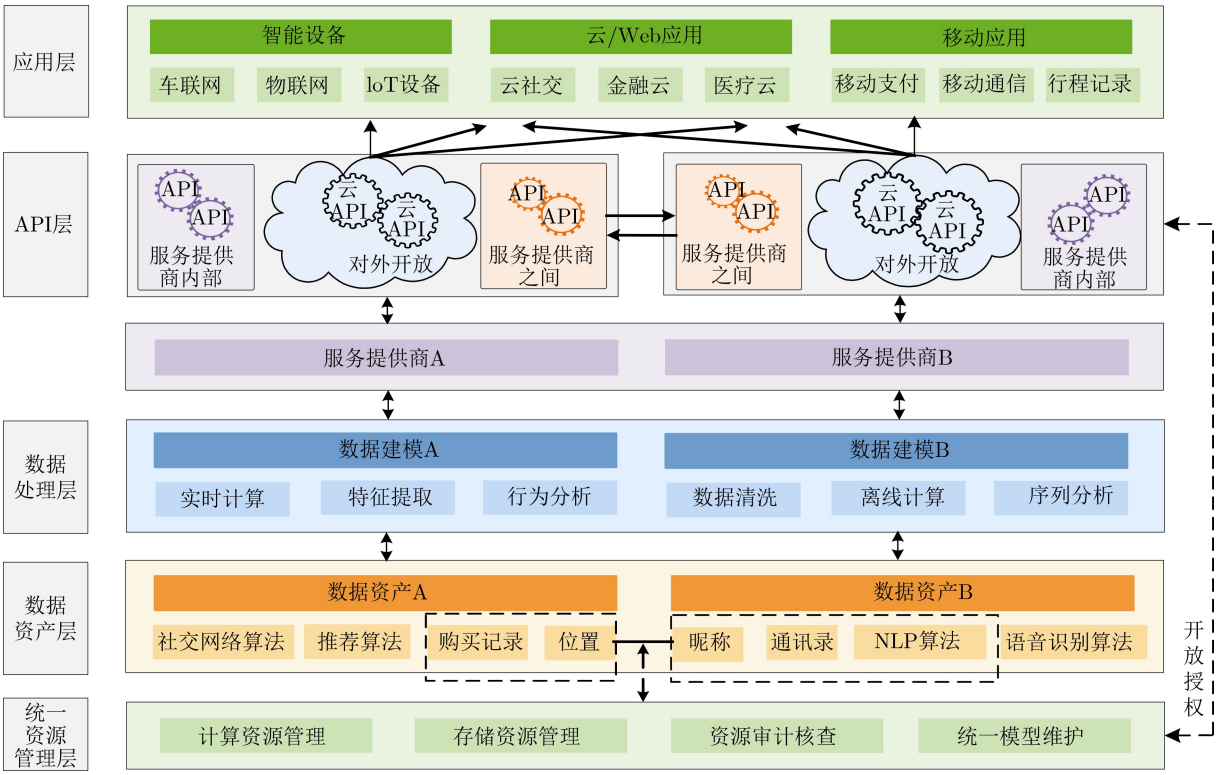


图 2 云API应用体系结构

2.3 云API分类

2.3.1 基于功能分类的云API

云API按照功能可划分为两类[5]：一类是用于提供计算服务的云API，这包括像阿里支付云API、携程票务云API，应用程序通过调用云API来请求原始软件处理业务；一类是用于提供数据服务的云API，其中包括如百度地图云API、墨迹天气云API等允许用户访问符合其权限的数据。

2.3.2 基于应用范围分类的云API

云API按照应用范围大致可以划分为3类[6]：私有API(Private API)、合作API(Partner API)和公有API(Public API)。这3类云API的特点如表1所示。

如表1所示，Private API一般部署在VPC网络，不对外开放使用，所以在内部调用期间很少会有攻击者发现这类API，在一定程度上回避了来自外部的攻击，只需要关注来自企业内部员工的身份验证问题，具备良好的安全性能。Partner API是指在服务提供商之间合作使用的云API，只需要有效的访问控制和授权机制就可以以较低的成本开发和维护。Public API提供给任何想要访问应用的用户使用，虽然云API所具备的功能受限，但是攻击者可以通过宽泛的访问机制来对后端系统造成潜在的安全威胁。本文介绍的云API安全问题虽然涵括了这3大类，但是更多的是指安全性能一般的Partner API和Public API所面临的问题。

3 云API脆弱性及安全研究的重要性

3.1 云API的脆弱性分析

开放Web应用程序安全项目(Open Web Application Security Project OWASP)[7]发表的10大网络应用安全风险评议名单表明：10大漏洞中有9个都包含云API组件。由此可见，云API在面对外来的干扰与冲击时所展现出来的其本身的脆弱性。在分析云API脆弱性的时候，可以从外界扰动和云API自身两方面来进行展开。

3.1.1 基于外界扰动的脆弱性分析

为了展现云API在面对外界扰动时表现出的脆弱性，Bozkurt等人[8]集中在可扩展标记语言(eXtensible Markup Language, XML)文件和简单对象访问协议(Simple Object Access Protocol, SOAP)上设置突变信息，以检验云API应对扰动的能力。Espinha等人[9]在云API收到的正常响应中添加消息扰动，这些扰动包括空响应、错误响应、字段移除等，以测试云API是否能做出正常的操作。文献[10]提出一种基于差异流量分析的云API脆弱性识别方法，通过比较正常调用和异常调用返回值的差异判断该云API是否具有脆弱性。

3.1.2 基于云API自身的脆弱性分析

对于云API自身而言，云API的脆弱性可以分为3部分。

首先，如图3所示，提供广泛服务的大型云平台通常具有大量的第三方集成，这些集成严重依赖云API收集数据，并将其无缝提供给用户。以多云环境、容器化架构为代表的云服务平台意味着数个云API在通信过程中将持有更多的统一资源标识(Uniform Resource Identifier, URI)、方法、标识等参数，攻击者在捕获其中一个云API时就有可能威胁到整个云平台，因此云API的可攻击域和攻击面更广。

其次，如图4所示，在传统的Web场景中，用户通过浏览器访问用户界面(User Interface, UI)来向服务器请求页面和数据，大多数的逻辑处理是由应用服务器本身执行的，攻击者只能通过比如向页面上的表单提交恶意输入来完成对浅层超文本标记语言(Hyper Text Markup Language, HTML)页面的操纵，在一定程度上保护了后端服务器的安全。而大多数的移动应用程序都将业务逻辑从后端服务器转移到了客户端设备本身，这使得应用程序在云API的支撑下，直接实现对数据和服务的调用，下游服务器转而作为应用程序提取数据的代理，不再单独进行数据处理。这样就导致了云API对后端层的调用粒度边界从相对安全的内部层转移到暴露在互联网上的客户端移动应用，当攻击者绕过用户身份验证后，就可以利用云API直接进入下游应用程序，对数据资产造成更深层次的攻击。

最后，云API的设计过程本身就是一个博弈的过程，一方面，云API有着存储、计算和访问敏感资源的能力，而另一方面，云API强调了数据资产的开放性和可用性，云API在为客户端提供更广泛

表 1 基于应用范围分类的云API特点比较

API类别	部署方式	应用范围	访问机制	延展性	安全性能
Private API	VPC网络	服务提供商内部	有效	一般	高
Partner API	VPC网络/公共互联网	服务提供商之间	有效	好	一般
Public API	公共互联网	任何用户	不足	好	低、易受攻击

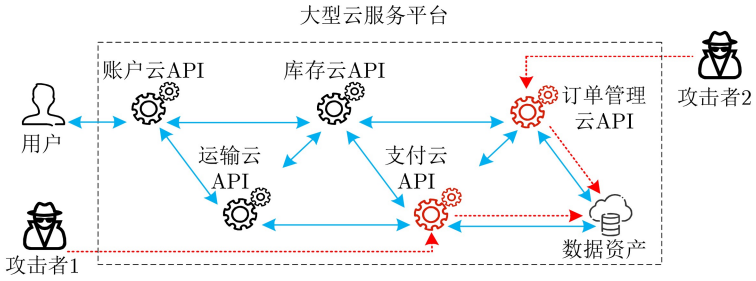
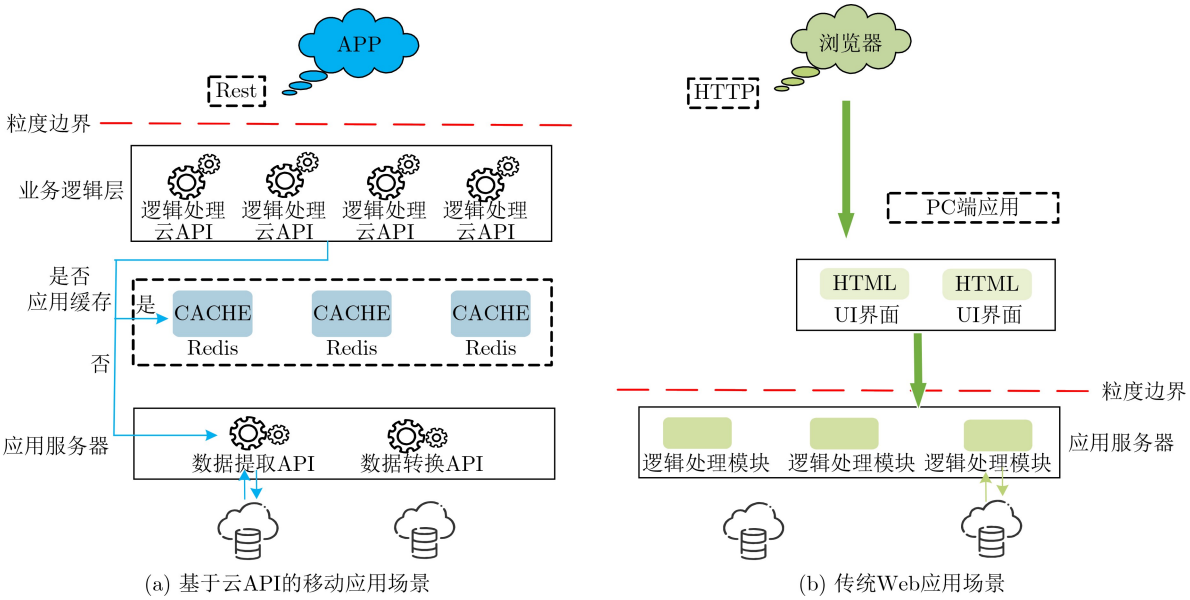


图 3 云API扩大了攻击域与攻击面



(a) 基于云API的移动应用场景　　(b) 传统Web应用场景

图 4 基于云API的移动应用场景与传统Web应用场景对比

的访问权限的同时，也暴露了更多的潜在敏感数据，所以如何平衡数据保护和数据开放之间的关系尤为重要。

3.2 云API安全重要性

随着云API数量的急剧增多，云API使用不规范现象时有发生，包括：(1)以第三方超出业务需要，大量私自缓存用户数据的云API数据滥用现象；(2)以接口权限漏洞被攻击者窃取，隐私数据采集未被用户授权的云API数据泄露现象。一方面，云API通过共享应用价值来提高开发效率，另一方面，云API引发了关于安全性和隐私问题的讨论，因此确保高效的访问控制和安全防御是十分重要的[11]。

由于云API在面对外界干扰时所表现出的脆弱性以及目前有效的安全机制缺乏的现状，云API的安全状态和市场需求之间还存在着很大的差距，恶意威胁行为比以往任何时候都更加瞄准云API。因此，如何确立一种有效的云API安全机制显得尤为重要，本文的目的在于讨论当今云API生态系统所面临的威胁和攻击，并提供云API的防护方法。

4 云API威胁与防护

一个全面的云API安全方法应该着眼于云API价值链中的所有环节：从使用云API的用户和应用场景，到构建云API的开发团队，一直到在后端系统中公开数据和服务的云API提供者，任何一个环节的缺失都可能导致整个项目的瘫痪。由于云API提供全通道访问，云API安全方法也应该是全通道安全。安全体系结构应该足够灵活，几乎实时地防止、检测和响应所有形式的云API威胁。图5给出了云API在整个项目节点中可能遭遇的威胁，本节将分别针对云API的6种不同类型的威胁与防护具体展开。

4.1 身份验证

用户身份验证是验证支撑证据(如用户标识或密码)以获取网络资源的过程，这一过程保证了用户的真实性。实际上，包括云API安全在内的所有安全项目都是从身份验证展开的。一些攻击者操控自动化恶意软件通过一系列的自动化决策进行自繁殖，根据被感染云API的参数进行智能调整。除了利用漏洞以外，它还可能找到并学习特定设备特定

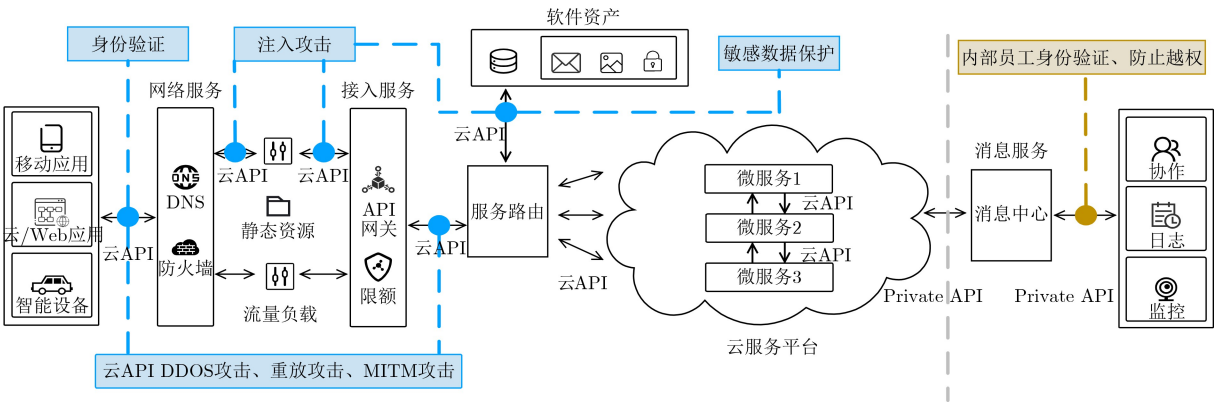


图 5 云API安全研究框架

时间使用的凭证，以更静默的方式发起攻击，减少被检测和识别的概率。

4.1.1 基于密码的云API身份验证

每个被成功授权的用户都被分配了一个由数字或字符串标识的账户，登录时系统需要测试用户输入密码与预期密码是否匹配。在密码传输过程中，Setiadi等人[12]提到可以使用抗攻击性强的安全哈希算法1(Secure Hash Algorithm 1, SHA1)将密码进行加密，来降低密码暴露的风险。但是由于弱密码现象的存在，即便按照Sklavos等人[13]提到的将密码经过单向哈希函数运算来存储在内存中，仍然会被攻击者通过不同类型的自动攻击[14]来破解用户的账户。

4.1.2 基于密钥和令牌的云API身份验证

为了避免用户弱密码现象的出现，可以使用对称算法对密码进行加密，在对称密钥交换过程中可通过密钥交换协议(Diffe-Hellman, DH)来保证其安全性，DH协议产生的密钥可用于后续的加密、进一步的密钥管理。另外也可以使用安全性更高的非对称算法在认证服务器上为用户生成临时云API密钥[15]，代替密码实现关联、标识客户端应用程序的作用。除此以外，云API令牌也可以代替用户密码访问资源。单点登录(Single Sign On, SSO)机制[16]将登录认证和业务系统分离，登录通过后，SSO颁发的令牌将在全局API中使用，所有相互信任的子系统都能免登录访问资源。OAuth 2.0机制在两个独立的系统之间完成了令牌授权，由用户决定是否把自己在某个服务商上面的资源授权给第三方访问，比如常见的微信登录、GitHub登录。

4.1.3 基于多因素的联合云API身份验证

用户认证信号涉及3大类因素：基于秘密信息、基于私人物品、基于生物特征。密码的另一种替代方法是双因素身份验证[17]，区别于传统静态信息验证，它并行校验两种不同类别的因素，将自然变量结合一定的加密算法组合出一组动态密码。除了用户需要参与的部分，Van Oorschot[18]还定义了一些隐式认证信号比如历史IP地址记录、设备指纹特征、页面缓存等信息来辅助识别用户身份。在面对攻击者大批量的自动攻击时，Nokovic等人[19]提出了一种多级云API身份认证机制，系统在机器学习模型中输入风险级别的数据，根据每次用户登录时收集的用户相关属性进行评估，在动态更新学习数据的同时提高风险判别的准确度。

4.2 云API DDoS攻击与防护

当云API端点被开放给外部世界，开发人员使用这些云API来搭建新的应用程序，这些使用了云API的应用程序又逐渐被用户熟知和采纳，云API的整体流量必然会显著增加。此外，服务器在响应云API请求的时候会大额消耗网络、内存、CPU等资源，同时还要应对来自多个客户端应用程序的资源争夺现象，其中很可能会掺杂着来自恶意程序的不必要的负载，这些都有可能影响整个系统的可用性。

分布式拒绝服务(Distributed Denial of Service, DDoS)通过远程连接利用恶意程序，对目标服务器性能或网络带宽造成影响。云API DDoS攻击不同于网络或传输层DDoS攻击的是，后者可以通过限制速率来实现检测和阻止，而云API DDoS的执行通常会由多个发送流量的客户端发起，由于每个攻击者都发送正常的流量，如果不对单个云API的流量逐一进行分析，这些攻击就很难被检测到。

4.2.1 云API DDoS攻击的表现形式

云API DDoS攻击的具体表现形式可以分为两类[20]：基于网络的攻击方法和基于参数缺陷的攻击方法。

(1)基于网络的攻击方法：攻击者控制多个客户端发送大量的云API请求来挤占受害服务器的网络资源，造成合法用户的请求没有足够的带宽支

持，从而让合法用户的服务请求无法被处理。现有的攻击者甚至可以检测限速控制，调整流量速率，使其保持在节流限制之下，以逃避检测。

(2)基于参数缺陷的攻击方法：攻击者利用受害服务器提供的云API服务在数据处理上的缺陷，发送大量的畸形数据或无效数据引发服务器处理错误，大量占用系统资源。

4.2.2 基于流量限制的云API DDoS防护

在面对云API DDoS攻击时，将整体流量限制在服务器的可承受范围内是基本原则。流量限制策略大致可以分为2种[21]：

(1)流量尖峰拦截策略：可以捕获到流量的突然变化，并在较小的时间内平滑流量。峰值的截断值应该基于后端服务的容量来计算，峰值的数值过大可能会起不到防御的作用而数值过小会影响整个云API的服务质量，一个合适的参数才能对云API的流量行为起到合理的保护作用。

(2)配额限制策略：云API请求数量会根据最初的应用程序或用户、发起地区等各种因素进行限制，在限制内的云API请求将会被成功路由到目标系统，超过限额的将被拒绝。配额限制值控制着一段持续时间内云API调用的次数，应该通过后端服务的总体容量和预期负载来谨慎推算。

4.2.3 基于流量验证的云API DDoS防护

在实际的多客户端的云API DDoS攻击中，单个实施攻击的客户端流量在正常的配额之内不会被策略阻止，但所有实施攻击的客户端的流量聚合仍然会对系统造成实质的损害。云API流量突然爆发时及时关闭应用程序是可以解决的，但是会导致所有正常用户断开与服务器的连接，这并不理想。因此，可以让平台去对超出阈值的错误率、同一IP反复发送请求等异常行为模式进行识别[22]，也可以通过更复杂的算法去验证客户端用户的属性，比如用户特征、常驻位置、常用设备等。结合Netacea[23]描述的历史IP分析、源验证来监测正常流量数据和访问记录，或者运用Harguindeguy[24]提到的学习每个API随时间推移的正常行为模式来标记有异常流量的API，同时可以考虑多个级别的上下文来进行云API访问推算。

4.3 重放攻击与防护

传统的重放攻击是指攻击者拦截并发送一个受害服务器已经接受过的包，来达到欺骗系统的目的[25]。在云API的交互过程中使用超文本传输安全协议(Hyper Text Transfer Protocol over Secure-socket, HTTPS)进行数据加密可以有效防止明文数据被监听，但是攻击者仍可以盗取云API请求或包含用户安全信息的语句，在不对这段信息进行解密的情况下，直接把这段信息原封不动地反复反馈给服务器，攻击者就能以受害客户端的身份与服务器交流。

重放攻击的危害体现在对通信双方认证安全性的破坏。此外，在攻击者反复发送窃取到的信息的过程中，阻碍了双方正常应答的同时也挤占了服务器端有限的网络资源，也算是另一种程度上的可用性攻击。

4.3.1 基于“抗重放因子”的抗重放方案

对于云API的重放攻击，可以选择在网络通信的过程中，在请求参数中添加一个“抗重放因子”[26]来判断API请求的实时性，防止攻击者重复发送窃取到的API请求。至于“抗重放因子”的选择，首先需要确保其不能随意被攻击者篡改，或者被篡改之后也可以通过一些技术如数字信封、完整性校验[27]检测出来，其次需要保证它在允许的范围内不会重复。“抗重放因子”包括：随机数、时间戳、流水号[28]。文献[29]提到将“随机数”和“时间戳”组合使用，只需要保存某个很短时间段内的所有随机数，这样降低了时间戳同步的精确度。除此之外，挑战-应答机制和一次性口令机制[30]在实际中使用得也很广泛，这一类动态访问机制对于银行支付、网银转账等即时性强、安全需求高的云API有很强的抗重放能力。

4.3.2 抗重放方案比较

各种防护方案都有其各自的优缺点(见表2)，实现方式不同，效率和难易程度上略有差异，需要根据实际应用场景选择合适的抗重放方案。

4.4 MITM攻击与防护

中间人(Man-In-The-Middle, MITM)攻击作为黑客常用的攻击手段，至今仍然具有极大的扩展空间。在传统Web场景中，MITM攻击范围广泛，以往猖獗一时的域名系统(Domain Name System, DNS)欺骗、会话劫持(Server Message Block, SMB)技术都是典型的MITM攻击手段。随着微服务架构的流行，云API流量的增长，在黑客技术越来越多地运用于以获取经济利益为目标的情况下时，MITM攻击转而成为对网银、网游等网络交易云API破坏性极强、威胁性极大的一种攻击方式。

4.4.1 MITM攻击的攻击方式

传统Web场景中，MITM攻击分为两步：会话劫持和数据篡改[31]。攻击者利用通信协议的缺陷和双方在通信过程中产生的信息差，恶意引导两个端点的通信路径。对比简单的窃听攻击，MITM攻击有着拦截、修改、替换数据的能力。云API场景中，攻击者在占据通信信道后会捕获身份凭证来进行未经授权的访问。

4.4.2 MITM攻击的攻击类别

不管是提供给企业内部使用的Private API，还是开放给外部程序使用的其他类型的云API，都有可能遭受到不同类型的MITM攻击。对于Private API来说，局域网通信中的攻击者会利用地址解析协议(Address Resolution Protocol, ARP)的漏洞修改本地ARP缓存表，将恶意主机的MAC地址与目标主机IP相关联，造成ARP缓存中毒。对于其他类型的云API来说，网络中分布的恶意DNS服务器，攻击者伪造的DNS响应，都会使用户绕过正确的IP地址进入攻击者预设的陷阱IP，从而导致DNS缓存中毒[32]。除此之外，大部分云API都会采用安全套接字协议(Secure Sockets Layer, SSL)保障数据的安全传输，攻击者可能在这一过程中劫持或伪造SSL证书，当用户忽略浏览器的安全警告接受伪造的证书时，攻击者就能操控通信双方信息的加、解密。

4.4.3 MITM防护方案比较

云API防护的主要原则不外乎对通信双方认证过程中的传输介质进行认证以及对认证过程本身的真实性进行认证。对于不同类型的API可能遭受的MITM攻击，表3从现有MITM攻击防护方法中选取了代表性的方法，并从多个角度进行了分析对比。对于Private API来说，MITM攻击主要利用通信协议的无连接性来进行，从而在防护过程中应该考虑在局域网通信过程中额外增加认证机制，比如分发密钥[33]、发行票据[34]、添加“等待”条目等[35]。对于其他类型的API来说，MITM攻击则是通过干扰云API的DNS响应，或伪造SSL证书来截获网络通信内容，因此在认证过程可以增加其他辅助信息如中央审计日志[36]、内置证书公钥[37]，或者对数字签名采用不同的加密方式来保证认证过程的真实性。

4.5 注入攻击与防护

在传统的Web场景中，数据传送到远程系统(服务器)更多地依赖统一资源定位(Uniform Resource Locator, URL)和表单提交的功能，只有很少的数据呈现。然而云API通过使用更开放的HTTPS协议实现了数据参数化。因此，当数据发送至云API时，攻击者通过基于查询参数、存取内容、脚本内容等各类参数攻击，有针对性地注入恶意内容。尤其是当客户端提供的数据没有经过云API验证、过滤或清洗，直接用来进行数据库查询、对象关系映射等功能性操作时，很可能会给云API带来巨大的危害。

4.5.1 注入攻击的攻击类别

注入攻击按照注入的内容可大致分为以下3类：

(1)脚本注入：攻击者通过提交系统要求以外的参数来诱使解释器执行非预期的命令。尽管云API没有一个明确可见的位置来接受用户输入，但是攻击者可以使用HTTP头字段作为入口点来注入恶意代码。

(2)SQL注入：攻击者在客户端输入数据中注

表 2 抗重放方案比较

抗重放方案	优点	缺点	适用通信单元数量		适用网络状况	
			少	多	不拥堵	无要求
随机数[25]	无需严格的时钟同步	内存占用大、查询开销大	√			√
时间戳[26]	内存占用少	严格的时钟同步		√	√	
流水号[27]	校验简单、内存占用较少	判断准确率较低	√		√	
一次性口令机制[28]	即用即更新、验证维持时间久	需要双方计数器同步、时钟同步		√		√
挑战-应答机制[30]	无需严格的时钟同步	信道占用大、验证维持时间短		√	√	

表 3 MITM防护方案比较

MITM防护方案	攻击目标类型	攻击场景	模型/方法	防护机制
Bruschi等人[33]	ARP缓存中毒	封闭	S-ARP	可信主机分发密钥
Limmaneewichid等人[38]	目标IP替换	封闭	P-ARP	哈希函数隐藏IP地址
Lootah等人[34]	ARP缓存中毒	封闭	T-ARP	集中发行票据认证
Trabelsi等人[35]	ARP请求应答超时	封闭	有状态ARP应答	添加“等待”条目
Ataullah等人[39]	ARP无状态性攻击	封闭	ES-ARP	广播ARP请求和应答
Ariyapperuma等人[40]	数据真实性受损	开放	DNSSEC	哈希函数加密数字签名
Kales等人[36]	恶意证书干扰	开放	伪造证书检验	补充中央审计日志
Soghoian等人[37]	恶意证书替换	开放	证书锁定	证书中的公钥提前内置

入查询语句，由于云API缺乏对输入语句的过滤，服务器端接收后会将攻击者的输入作为查询语句的一部分执行，从而破坏了原始的SQL查询逻辑。

(3)边界数据溢出攻击：它提供超出预期类型或范围过载的数据，比如具有级数过多的嵌套或过长的属性列表，导致系统崩溃从而获取对内存空间的访问。

4.5.2 注入攻击防护方案

在应对注入攻击时，接受输入参数的云API应提前将数据和查询或命令分开，Kingthorin[41]提到可以将注入攻击作为输入验证问题来处理，设置需要避开的潜在恶意值列表。此外，Zhong等人[42]提出让数据库以参数化的形式进行查询，执行时将参数里的敏感字符视为字段的值来处理。Rajaram等人[43]提到在云API实施通信之前，可以采用正则的方式对所有来自潜在不可信来源的云API数据做严格的合法性校验。

4.6 敏感信息泄露与防护

敏感信息包括但不限于：短信、通讯记录、通话日志、口令、凭据、用户数据(如姓名、住址、邮箱、电话)。对于云API而言，对外开放数据访问通道是必需的，但这样的数据访问通道在没有良好的安全防护的前提下，就会成为数据泄露的风险源头。

4.6.1 基于数据加密的敏感信息防护

传输过程中的数据包被拦截，就会导致数据泄露，存在很大的安全隐患。因此，可通过SSL、传输层安全性协议的加密方式来提高整体数据的安全性。整个系统的数据传输安全性与所采用的安全加密算法有一定的相关性。会话加密协议(Off-The-Record messaging, OTR)对保证即时通信API内容的安全性有着很大的作用，其选择使用DH密钥交换协议来建立共享密钥组，加密的时候会采用短期密钥来确保协议可以实现完美前向加密性，不足之处是很容易在认证过程中遭受MITM攻击。Yang等人[44]采用高级加密标准(Advanced Encryption Standard, AES)算法、RSA算法混合加密来提高数据传输的安全性，另外还利用云API网关作为用户与微服务模块之间的通信工具，解决了后端安全问题。

4.6.2 基于数据筛选的敏感信息防护

如何鉴别敏感数据并有选择性地加密云API包括两种已知的内容鉴别的方法：JSON-schema和数据的反序列化，在执行内容鉴别的过程中也可以额外构建关于数据屏蔽、数据加密是否到位的检查规则，在识别敏感数据的同时检测是否有暴露的风险。文献[45]引入了4条安全规则，违反这些规则可能造成攻击者劫持云资源窃取信息，其中为每条规则定义了一个模块化的属性检查器，在云服务上自动监视和检测违反规则的云API。文献[46]给出了敏感云API的定义，列出了与用户隐私相关的敏感云API库，并通过分析敏感云API库中的潜在威胁来判断是否存在敏感数据泄露。

5 AI驱动云API安全

Gartner[47]在2020年10大战略技术趋势报告中，明确指出在接下来的5年中，人工智能将广泛应用到网络攻防问题上。文献[48]将云API的复杂性分为两个方面，一方面是参数的复杂性，云API存在着大规模的参数和极大的相互依赖性；另一方面是代码库的复杂性，代码库应用了不同的框架，一个云API执行的业务规则可能会跨诸多类。随着云API数量的急剧增加，云API的复杂性也呈爆炸式增长，单方面限制对云API的访问已经不再能规避风险。每个新的云API都为数据和应用程序的安全威胁和攻击向量增加了新的维度，而传统安全模型主要依赖于认证、节流、通信安全，所以很难通过单一的传统方法来实现目前复杂云API的安全防护。与此同时，AI和机器学习(Machine Learning, ML)支持的云API安全能通过现有的基础安全特性跟踪流量来源，自动学习历史信息，以更灵活的方式应对目前的危机。

考虑到网络攻击的动态性和多样性，一个全面的云API安全解决方案不仅需要安全防护能力，还需要异常检测能力。到目前为止，一些研究已经提出应用不同的ML技术来检测云API网络流量中的异常。Kromkowski等人[49]提出了一个用于分析和检测网络流量数据异常情况的框架，通过实行二次验证策略，完成对流量数据的异常检测。另外，Baye等人[50]采用高斯分布离群检测技术来衡量云API流量数据的离散度，在标记异常流量后，使用带宽和令牌的请求数量等特定特征来检测与分类云API流量。目前流行的各种机器学习算法，如K最近邻法(K- Nearest Neighbor, KNN)、朴素贝叶斯(Naïve Bayes, NB)、决策树(Decision Tree, DT)等[51]，主要通过学习每个云API上下文信息中的正常行为模式来标记异常行为和恶意数据，在事先不了解攻击的情况下通过自动化的方式阻止多种情境下的云API攻击和恶意行为模式。如表4所示，与传统防护方案相比，AI防护在解决身份验证、云API DDoS攻击、敏感数据保护等各类安全问题时为云API添加了持续学习上下文的能力。总之，

表 4 两类防护方案对比

攻击类别	传统云API安全	AI驱动云API安全
身份验证	令牌、密钥	历史信息自学习
云API DDoS攻击	负载均衡、速率限制	流量数据计算、源验证
重放攻击	抗重放因子	暂无
MITM攻击	传输介质检验	暂无
注入攻击	参数化查询、正则化检验	暂无
敏感数据保护	加密(SSL, TLS)	敏感数据学习、提取

ML可以将云API安全扩展到访问控制和通信安全之外，比如解决新的网络威胁、识别历史攻击行为、基于现有模式进行预测以确保云API安全。

6 结束语

虽然现在已围绕云API安全防护做了大量的研究工作，总结归纳了许多有效的解决方案，但网络攻击呈现出了多样化、规模化、复杂化等特点。攻防对抗的形势日趋严峻，未来仍然还有诸多挑战，比如：(1)微服务架构下由于云API特殊的内部不可见性，对于云API的识别和发现会更为困难；(2)研发运维一体化，高级管理账户云API更易暴露在攻击者面前，纵向越权攻击更易发生；各类服务间调用密集、依赖性增强，通过云API进行横向攻击变得更加频繁；(3)功能内容更加细化，攻防粒度也更为精细，大而全的规则不再适用，云API的防御规则也应该向着细而精的方向改进。

云API安全防护是一个很有前景的研究方向，仍处于成熟过程的早期阶段。现有方案的配置仍受到静态、不可变的限制，在挫败攻击者进行探查、发动攻击和窃取敏感信息方面仍有很多不足之处。最初云API安全防护主要解决通用的各类漏洞攻击，未来自动化攻击、AI类攻击将占据云API攻击的主流。相应地，安全防护措施也需要加强系统化、自动化、智能化能力，共建一个全面、高效、动态的防护方案，向着以体系对抗体系，以智能防护智能的方向推进。

参 考 文 献

[1] 艾瑞咨询有限公司. 2020年中国人工智能API经济白皮书[R]. 艾瑞咨询系列研究报告, 2020.
IResearch Consulting Croup. White paper on API economy of China's artificial intelligence[R]. IResearch Consulting Series Research Reports, 2020.

[2] TAN Wei, FAN Yushun, GHONEIM A, *et al*. From the service-oriented architecture to the Web API economy[J]. *IEEE Internet Computing*, 2016, 20(4): 64–68. doi: 10.1109/MIC.2016.74.

[3] ESPINHA T, ZAIDMAN A, and GROSS H G. Web API growing pains: Loosely coupled yet strongly tied[J]. *Journal of Systems and Software*, 2015, 100: 27–43. doi: 10.1016/j.jss.2014.10.014.

[4] BOUGUETTAYA A, SINGH M, HUHNS M, *et al*. A service computing manifesto: The next 10 years[J]. *Communications of the ACM*, 2017, 60(4): 64–72. doi: 10.1145/2983528.

[5] HUSSAIN F, HUSSAIN R, NOYE B, *et al*. Enterprise API security and GDPR compliance: Design and implementation perspective[J]. *IT Professional*, 2020, 22(5): 81–89. doi: 10.1109/MITP.2020.2973852.

[6] ARCURI A, FRASER G, and JUST R. Private API access and functional mocking in automated unit test generation[C]. 2017 IEEE International Conference on Software Testing, Verification and Validation, Tokyo, Japan, 2017: 126–137. doi: 10.1109/ICST.2017.19.

[7] OWASP. OWASP top ten 2017[EB/OL]. https://www.owasp.org/index.php/Top_10-2017_Top_10, 2017.

[8] BOZKURT M, HARMAN M, and HASSOUN Y. Testing Web services: A survey[R]. Technical Reports TR-10-01, 2010.

[9] ESPINHA T, ZAIDMAN A, and GROSS H G. Web API fragility: How robust is your mobile application?[C]. The 2nd ACM International Conference on Mobile Software Engineering and Systems, Florence, Italy, 2015: 12–21. doi: 10.1109/MobileSoft.2015.9.

[10] 刘奇旭, 邱凯丽, 王乙文, 等. 面向OAuth2.0授权服务API的账号劫持攻击威胁检测[J]. 通信学报, 2019, 40(6): 40–50. doi: 10.11959/j.issn.1000-436x.2019144.
LIU Qixu, QIU Kaili, WANG Yiwen, *et al*. Account hijacking threat attack detection for OAuth2.0 authorization API[J]. *Journal on Communications*, 2019, 40(6): 40–50. doi: 10.11959/j.issn.1000-436x.2019144.

[11] DIG D and JOHNSON R. How do APIs evolve? A story of refactoring[J]. *Journal of Software Maintenance and Evolution:Research and Practice*, 2006, 18(2): 83–107. doi: 10.1002/smr.328.

[12] SETIADI D R I M, NAJIB A F, RACHMAWANTO E H, *et al*. A comparative study MD5 and SHA1 algorithms to encrypt REST API authentication on mobile-based application[C]. 2019 International Conference on Information and Communications Technology, Yogyakarta, Indonesia, 2019: 206–211. doi: 10.1109/ICOIACT46704.2019.8938570.

[13] SKLAVOS N and KOUFOPAVLOU O. Implementation of the SHA-2 hash family standard using FPGAs[J]. *The Journal of Supercomputing*, 2005, 31(3): 227–248. doi: 10.1007/s11227-005-0086-5.

[14] GORSKI P L, ACAR Y, IACONO L L, *et al*. Listen to

developers! A participatory design study on security warnings for cryptographic APIs[C]. The 2020 CHI Conference on Human Factors in Computing Systems, Honolulu, USA, 2020: 1–13. doi: 10.1145/3313831.3376142.

[15] Angular University. JWT: The complete guide to JSON web tokens[EB/OL]. https://blog.angular-university.io/angular-jwt/, 2022.

[16] KARUNANITHI M D and KIRUTHIKA B. Single sign-on and single log out in identity[C]. The International Conference on Nanoscience, Engineering and Technology, Chennai, India, 2011: 607–611. doi: 10.1109/ICONSET.2011.6168044.

[17] FUJII H and TSURUOKA Y. SV-2FA: Two-factor user authentication with SMS and voiceprint challenge response[C]. The 8th International Conference for Internet Technology and Secured Transactions, London, UK, 2013: 283–287. doi: 10.1109/ICITST.2013.6750207.

[18] VAN OORSCHOT P C. Computer Security and the Internet: Tools and Jewels[M]. Cham: Springer, 2020: 1–25. doi: 10.1007/978-3-030-33649-3.

[19] NOKOVIC B, DJOSIC N, and LI W O. API security risk assessment based on dynamic ML models[C]. The 14th International Conference on Innovations in Information Technology, Al Ain, United Arab Emirates, 2020: 247–252. doi: 10.1109/IIT50501.2020.9298975.

[20] BERA P, SAHA A, and SETUA S K. Denial of service attack in software defined network[C]. The 5th International Conference on Computer Science and Network Technology, Changchun, China, 2016: 497–501. doi: 10.1109/ICCSNT.2016.8070208.

[21] DE B. API Management[M]. Berkeley: Apress, 2017: 15–28. doi: 10.1007/978-1-4842-1305-6_2.

[22] IMPERVA. Bot defense for API security data sheet[EB/OL]. https://resources.distilnetworks.com/data-sheets/bot-defense-for-apis, 2018.

[23] NETACEA. Bot detection and mitigation with machine learning[EB/OL]. https://www.netacea.com/bot-detection, 2018.

[24] HARGUINDEGUY B. Artificial intelligence and machine learning: A new approach to API security[EB/OL]. https://www.pingidentity.com/en/company/blog/posts/2018/artificial-intelligence-machine-learning-a-new-approach-to-api-Security.html, 2018.

[25] ZHU Minghui and MARTÍNEZ S. On the performance analysis of resilient networked control systems under replay attacks[J]. *IEEE Transactions on Automatic Control*, 2014, 59(3): 804–808. doi: 10.1109/TAC.2013.2279896.

[26] GRUSCHKA N and LUTTENBERGER N. Protecting web services from DoS attacks by SOAP message validation[C]. The IFIP TC-11 21st International Information Security Conference, Karlstad, Sweden, 2006: 171–182. doi: 10.1007/0-387-33406-8_15.

[27] JENSEN M, GRUSCHKA N, and HERKENHöNER R. A survey of attacks on web services[J]. *Computer Science-Research and Development*, 2009, 24(4): 185–197. doi: 10.1007/s00450-009-0092-6.

[28] DE RYCK P, DESMET L, PIESSENS F, *et al*. Primer on client-side web security[M]. Cham: Springer, 2014: 105–109. doi: 10.1007/978-3-319-12226-7.

[29] 肖斌斌, 徐雨明. 基于双重验证的抗重放攻击方案[J]. 计算机工程, 2017, 43(5): 115–120,128. doi: 10.3969/j.issn.1000-3428.2017.05.019.

XIAO Binbin and XU Yuming. Scheme of anti-replay attacks based on two-factor authentication[J]. *Computer Engineering*, 2017, 43(5): 115–120,128. doi: 10.3969/j.issn.1000-3428.2017.05.019.

[30] 王育红, 夏安祥, 林国庆, 等. 抗重放攻击方案在工程中的应用[J]. 网络安全技术与应用, 2021(4): 8–10. doi: 10.3969/j.issn.1009-6833.2021.04.006.

WANG Yuhong, XIA Anxiang, LIN Guoqing, *et al*. Application of anti-replay attack scheme in engineering[J]. *Network Security Technology & Application*, 2021(4): 8–10. doi: 10.3969/j.issn.1009-6833.2021.04.006.

[31] CONTI M, DRAGONI N, and LESYK V. A survey of man in the middle attacks[J]. *IEEE Communications Surveys & Tutorials*, 2016, 18(3): 2027–2051. doi: 10.1109/COMST.2016.2548426.

[32] NAQASH T, UBAID F B, ISHFAQ A, *et al*. Protecting DNS from cache poisoning attack by using secure proxy[C]. 2012 International Conference on Emerging Technologies, Islamabad, Pakistan, 2012: 1–5. doi: 10.1109/ICET.2012.6375486.

[33] BRUSCHI D, ORNAGHI A, and ROSTI E. S-ARP: A secure address resolution protocol[C]. The 19th Annual Computer Security Applications Conference, Las Vegas, USA, 2003: 66–74. doi: 10.1109/CSAC.2003.1254311.

[34] LOOTAH W, ENCK W, and MCDANIEL P. TARP: Ticket-based address resolution protocol[J]. *Computer Networks*, 2007, 51(15): 4322–4337. doi: 10.1016/j.comnet.2007.05.007.

[35] TRABELSI Z and EL-HAJJ W. Preventing ARP attacks using a fuzzy-based stateful ARP cache[C]. 2007 IEEE International Conference on Communications, Glasgow, UK, 2007: 1355–1360. doi: 10.1109/ICC.2007.228.

[36] KALES D, OMOLOLA O, and RAMACHER S. Revisiting user privacy for certificate transparency[C]. 2019 IEEE European Symposium on Security and Privacy, Stockholm, Sweden, 2019. doi: 10.1109/EuroSP.2019.00039.

[37] SOGHOIAN C and STAMM S. Certified lies: Detecting and defeating government interception attacks against SSL (short paper)[C]. The 15th International Conference on Financial Cryptography and Data Security, Gros Islet, St. Lucia, 2011: 250–259. doi: 10.1007/978-3-642-27576-0_20.

[38] LIMMANEEWICHID P and LILAKIATSAKUN W. P-ARP: A novel enhanced authentication scheme for securing ARP[C]. The 2011 International Conference on Telecommunication Technology and Applications, Singapore, Singapore, 2011: 83–87.

[39] ATAULLAH M and CHAUHAN N. ES-ARP: An efficient and secure address resolution protocol[C]. 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science, Bhopal, India, 2012: 1–5. doi: 10.1109/SCEECS.2012.6184794.

[40] ARIYAPPERUMA S and MITCHELL C J. Security vulnerabilities in DNS and DNSSEC[C]. The 2rd International Conference on Availability, Reliability and Security, Vienna, Austria, 2007: 335–342. doi: 10.1109/ARES.2007.139.

[41] KINGTHORIN. OWASP SQL injection[EB/OL]. https://owasp.org/www-community/attacks/SQL_Injection#, 2021.

[42] ZHONG Weilin and REZOS. Code injection software attack[EB/OL]. https://owasp.org/www-community/attacks/Code_Injection, 2021.

[43] RAJARAM A K, BABU B C, and KUMAR R C K. API based security solutions for communication among web services[C]. The 15th International Conference on Advanced Computing, Chennai, India, 2013: 571–575. doi: 10.1109/ICoAC.2013.6922014.

[44] YANG Dawei, GAO Yang, HE Wei, *et al*. Design and achievement of security mechanism of API gateway platform based on microservice architecture[J]. *Journal of Physics*:*Conference Series*, 2021, 1738: 012046. doi: 10.1088/1742-6596/1738/1/012046.

[45] ATLIDAKIS V, GODEFROID P, and POLISHCHUK M. Checking security properties of cloud service REST APIs[C]. The 13th International Conference on Software Testing, Validation and Verification, Porto, Portugal, 2020: 387–397. doi: 10.1109/ICST46399.2020.00046.

[46] MENG Shanshan, YANG Xiaohui, SONG Yubo, *et al*. Android's sensitive data leakage detection based on API monitoring[C]. The International Conference on Cyberspace Technology, Beijing, China, 2014: 1–4. doi: 10.1049/cp.2014.1340.

[47] PANETTA K. Gartner top 10 strategic technology for 2020[EB/OL]. https://www.gartner.com, 2020.

[48] GRENT H, AKIMOV A, and ANICHE M. Automatically identifying parameter constraints in complex Web APIs: A case study at Adyen[C]. The IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice, Madrid, ES, 2021: 71–80. doi: 10.1109/ICSE-SEIP52600.2021.00016.

[49] KROMKOWSKI P, LI Shaoran, ZHAO Wenxi, *et al*. Evaluating statistical models for network traffic anomaly detection[C]. 2019 Systems and Information Engineering Design Symposium, Charlottesville, USA, 2019: 1–6. doi: 10.1109/SIEDS.2019.8735594.

[50] BAYE G, HUSSAIN F, ORACEVIC A, *et al*. API security in large enterprises: Leveraging machine learning for anomaly detection[C]. 2021 International Symposium on Networks, Computers and Communications, Dubai, United Arab Emirates, 2021: 1–6. doi: 10.1109/ISNCC52172.2021.9615638.

[51] SHI Yi, SAGDUYU Y E, DAVASLIOGLU K, *et al*. Active deep learning attacks under strict rate limitations for online API calls[C]. 2018 IEEE International Symposium on Technologies for Homeland Security, Woburn, USA, 2018: 1–6. doi: 10.1109/THS.2018.8574124.

陈 真：男，副教授，研究方向为服务计算、云计算等.

乞文超：女，硕士生，研究方向为云API安全、云API攻击与防护等.

贺鹏飞：男，硕士生，研究方向为云API推荐、数据挖掘等.

刘林林：男，助理馆员，研究方向为云监测、科技数据挖掘、Web安全等.

申利民：男，教授，研究方向为柔性软件、协同计算、信息安全等.

责任编辑：余 蓉