文章编号:1671-4598(2018)12-0244-05 DOI:10.16526/j.cnki.11—4762/tp.2018.12.054 中图分类号:TP309 文献标识码:A

安全可扩展的 SaaS 服务开放平台框架设计

陆良伟，黄晓芳
（西南科技大学 计算机科学与技术学院，四川 绵阳 621010）

摘要：基于 SaaS 服务对开放平台的需求，结合 SaaS 服务的特点，提出了一种适用于 SaaS 服务的安全可扩展的开放平台解决方案，解决了 SaaS 服务在开放过程中的安全性、稳定性及性能等方面的问题；首先，通过对 Oauth2.0 协议两种授权模式的改进，提高了授权的安全性，保障了合法用户对资源的访问权限；其次，设计了基于令牌桶算法的限流策略和服务熔断机制，提高开放平台的稳定性，同时，制定可组合的负载策略为开放平台自身提供了负载均衡的能力，通过二级软负载的方式提升了开放平台的性能；最后，采用松耦合，多模块自由聚合的基础功能模块设计和动态可扩展的注册表机制，极大地提高的平台的扩展性。

关键词：Oauth2.0 协议；安全性；稳定性；扩展性；开放平台

Design of an Open Platform Framework for Secure and Scalable SaaS Services

Lu Liangwei, Huang Xiaofang
(School of Computer Science and Technology, Southwest University of Science and Technology, Mianyang 621010, China)

Abstract: Based on the requirements of SaaS services for open platforms, combined with the characteristics of SaaS services, a secure and scalable open platform solution for SaaS services is proposed, which solves the security, stability and performance of SaaS services in the open process. Aspects of the problem. Firstly, through the improvement of the two authorization modes of the Oauth 2.0 protocol, the security of the authorization is improved, and the access rights of the legitimate users to the resources are guaranteed. the access rights of legitimate users to resources. Secondly, the traffic limiting mechanism based on the token bucket algorithm and the service fuse mechanism are designed to improve the stability of the open platform. At the same time, the development of a loadable strategy can provide load balancing capability for the open platform itself. The way to improve the performance of the open platform. Finally, the loosely coupled, multi-module freely aggregated basic functional module design and dynamically scalable registry mechanism greatly enhance the scalability of the platform.

Keywords: Oauth2.0; security; stability; scalability; open platform

0 引言

开放是目前互联网的发展趋势，自 2007 年 5 月 FaceBook 正式开放其应用程序编程接口以来，各大平台也相继逐渐的开放了各自的 API（Application Programming Interface），建立了自己的开放平台。通过开放平台，不仅释放了平台的创造力，还能吸引第三方应用的用户，同时又丰富了用户体验，实现了三方共赢。SaaS（Software as a Service，软件即服务）服务作为一种新兴的软件模式在不断的发展和壮大，服务供应商提供给客户的服务种类也变得越来越丰富，其多租户（Multi-tenancy）[1-2]的特性，对于 SaaS 服务开放平台的安全性、并发性以及稳定性等方面也提出了越来越高的要求。

国内外著名的互联网企业都开放了其自己的开放平台，Google 的 OpenSocail 就是其开放的第一步，FaceBook 提出的 F8 开放标准，阿里巴巴集团的“大淘宝战略”也推出 TOP 开放平台，百度的“百度搜索开放平台”也开放了其搜索引擎，从技术架构上看，这些企业的开放平台都有其各自的技术特点和不同的框架。学术上也对开放平台的解决方案进行了许多的研究，朱蔚恒和周伟等人提出了一种开放平台架构模型，但是并未对稳定性和并发性作出深入的研究[3]。周巧也对开放平台系统进行了设计[4]，但是侧重于安全方面，对于性能及稳定性则没有深入讨论。虽然，现有开放平台有一些成熟的框架，但是在身份认证授权、灵活可扩展性及并发性能的处理方面，仍有待完善的地方，同时对于一些特定应用类型的开放平台框架，并不适用于 SaaS 服务开放平台。

由于不同业务类型对应的开放平台有其自身的架构特点，对于 SaaS 服务来说，常见的开放平台框架并不适用，

收稿日期:2018-04-17； **修回日期**:2018-05-15。
基金项目:国家青年基金(15zg2140)。
作者简介:陆良伟(1994-)，男，四川宜宾人，硕士研究生，主要从事 SaaS 服务开放平台安全性方向的研究。
黄晓芳(1977-)，女，四川绵阳人，研究生导师，副教授，主要从事信息安全、协议分析、数字签名等方向的研究。

需要提供一个适用于 SaaS 服务的开放平台框架。因此，基于 SaaS 服务对开放平台的需求，结合 SaaS 服务的特点，提出了一个安全可扩展的 SaaS 服务开放平台框架，并对其在安全性、稳定性等方面进行了分析。

1　开放平台技术研究

开放平台就是通过把网站的服务封装成一系列计算机易识别的数据接口开放出去，提供给第三方开发者使用，这种行为就叫做 OPEN API，提供 OPEN API 的平台本身就被称为开放平台。SNS 领域最具代表性的开放平台就是 FaceBook 的 F8 标准[5]，而国内电商平台中最具代表性的开放平台则是阿里巴巴的 TOP（Taobao Open Platform）[6]，国内搜索服务最具代表性的开放平台则是百度搜索开放平台[7]。通过研究发现，它们虽然在技术架构上差异很大，但是涉及到的核心技术，则都包含了如下两个方面：

1）Oauth2.0[8]协议授权，主要是为应用提供一种标准的方式去访问受保护的资源。对于开放平台来说，这里的授权访问包括两个方面，一是客户端直接与服务端进行交互，没有用户参与的授权；另一种则是需要用户授权，客户端才能访问用户受保护的资源。Oauth 协议为不同情况下的授权都提供的解决方案。

2）OPEN API 架构风格，主要包含了 RPC（Remote Procedure Call）和 RESTful（Representational State Transfer）[9]。RPC 即远程过程调用，它是一种通过网络向远程计算机程序请求服务，像调用本地服务一样调用服务。RESTful 则是一种资源定位及资源操作轻量级的 WEB 服务架构风格，基于这个风格设计的软件更加简洁，更有层次，也更利于实现缓存等机制，所以这也是目前最流行的 API 风格。

由于开放平台是将服务接口直接对第三方的开发者提供，接口都是暴露在公网上，同时开发者的数量和 API 的调用频率等都有差异，开放平台在架构设计的时候，需要关注以下两个方面：

1）安全性。开放平台的开放 API 由于是对外提供给第三方开发者调用，所以必须是暴露在公网上，这导致的直接的问题就是开放平台的安全性问题，谷歌的开放 API 就曾经因为安全问题而在发布不久后就下架[10]。Oauth2.0 自出现后，得到广泛关注，研究表明其框架在执行过程中存在令牌泄露，钓鱼攻击等威胁[11-13]，采用该协议的服务端存在 63.6%存在安全漏洞，作为客户端的网站也有 90.2%存在安全漏洞[14]。

2）稳定性。开放平台为不同的商家提供服务，API 被大量不同的第三方进行调用，所以开放平台的稳定性直接决定了为第三方提供的服务质量。如果因为某一方的调用而影响其他用户的调用，对用户而言是不合理的。随着用户数量的增长，开放平台还必须支撑高并发情况下的 API 调用，保证在高并发情况下服务的稳定，对于并发超过平台上限的时候，还必须具备一定的限流策略，从多个方面保证平台的稳定性。

2　框架设计

通过对开放平台的研究，再结合 SaaS 服务对开放平台的需求，在保证安全性和性能以及稳定性的前提下，设计了一种安全可扩展的 SaaS 服务的开放平台框架，并对其做了性能和稳定方面的改进，为 SaaS 服务在搭建开放平台的时候提供一个参考依据。

2.1　整体框架结构

安全可扩展的开放平台框架由平台核心模块，辅助可配置的独立模块，及服务注册表和服务群组成，框架整体结构如图 1 所示。

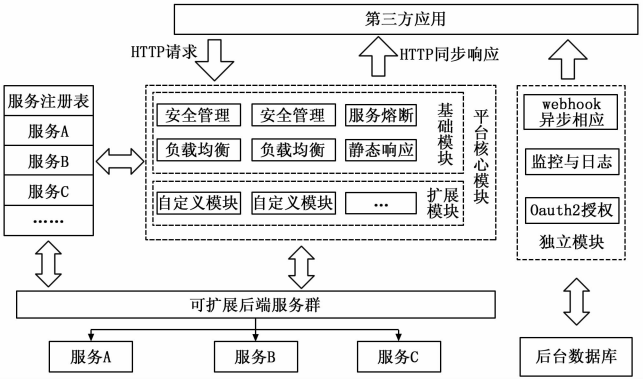

图 1　开放平台框架结构

平台核心模块主要由安全管理，流量管控，服务熔断，服务路由，负载均衡以及静态响应模块和可插拔的自定义模块组成，相较于传统的开放平台，提高了其扩展性，模块之间的松耦合也使得系统变得灵活可变。

其中安全管理模块负责开放平台的安全把控，通过对 Oauth2.0 授权流程的改进，增强了系统的安全性；流量管控模块负责在高并发情况下根据一定的策略对外部请求进行限流，保证了系统在高并发下达到系统上限时的稳定性；服务熔断模块的功能类似于电路总的保险丝，对下游的服务起到保护作用；服务路由模块负责对外部合法的请求进行转发，根据负载均衡模块提供的负载策略转发到下游服务；静态响应模块则负责在下游服务挂掉之后进行及时的响应。辅助可配置的独立模块则包含 webhook 响应，监控日志和授权组成，为核心模块提供安全保障和日志审计等功能。服务注册表用于维护下游 API 服务的基本信息，下游服务启动时，通过向开放平台发送自身的基本注册信息存储在注册表中，服务路由模块根据注册表中服务的信息，来对外部的请求进行转发。可扩展的后端服务群则是 SaaS 实际对外提供的服务 Restful API 集合，为租户提供实际的服务。

通过该框架设计，对于 SaaS 服务来说，第三方应用不用直接与具体服务的 API 进行通信，而是首先通过开放平

台这个中间层，按照一系列的校验规则和路由策略，最终才到达后端服务，进行具体的业务处理。

2.2 关键模块及流程

开放平台自身通过维护一个注册表，来保证和后端的服务连接，当外部的 HTTP/HTTPS 请求进入开放平台后，根据请求的 URL 去注册表中找到对应的可用服务，然后将请求转发到具体的服务中。在转发以前，开放平台还会对请求进行一系列的安全校验以及管控，从而对内部服务达到一种保护和隔离的作用。

2.2.1 改进的 Oauth2.0 授权设计

如图 1 所示，系统的安全管理模块中设计独立授权服务模块，通过对 Oauth2.0 的研究，在标准 Oauth2.0 的基础上对其进行了改进，来对客户端进行授权，当客户端通过服务器的身份验证之后，为客户端颁发访问资源的令牌。客户端带着已授权的访问令牌，对 API 进行请求访问，开放平台首先会对请求进行安全性的校验，包括验证调用者的身份，访问资源的权限。对于不符合条件的请求，直接返回，不进行路由，避免攻击者恶意调用 API。同时对于安全级别要求较高的资源，校验调用者是否有足够的权限对其进行访问。框架中包含了两种类型的授权：

1）没有用户参与，只对客户端访问 API 的情况进行授权。在 Oauth2.0 的客户端授权模式（client _ credentials）的基础上，对其做了改进，引入了 HMAC[15-16] 摘要计算，避免了客户端在申请访问令牌 token 的时候在信道中直接传输客户端密钥 SecretKey，有效的解决了密钥泄露的风险，增加了授权流程的安全性，改进后的授权流程如下图 2 所示。

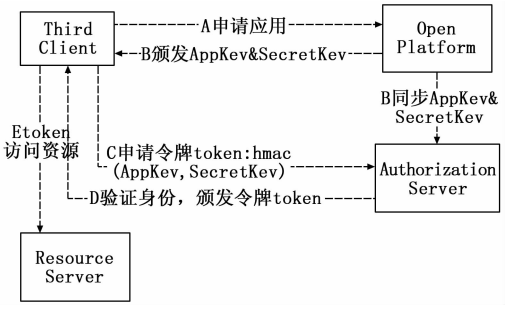

图 2 改进后的客户端授权流程

客户端想要访问开放平台的资源之前，需要到开放平台申请应用，开放平台对其进行身份认证，信息检查等之后，为客户端颁发应用 ID 和密钥，也就是 AppKey 和 SecretKey。当客户端想要申请令牌的时候，就对 AppKey 和 SecretKey 以及其他附加参数做 HMAC 摘要计算，并以“算法”＋“空格”＋“AppKey”＋“:”＋“摘要值”的形式传递给授权服务器（Authorization Server）进行令牌申请。授权服务器通过解析出 AppKey，再去寻找其对应的 SecretKey，并以相同的方式计算出 HMAC 摘要，如果比较两个值一致，则颁发令牌，否则不颁发。由于在传输过程中传递的是 hamc 值，并不是客户端密钥，所以即使信息被截取，对于攻击者而言也是没有意义的，提高了授权流程中的安全性。

2）有用户参与的授权，当客户端需要访问用户受保护的资源时，需要得到用户自己的授权，从而授权服务器才能给客户端颁发令牌，也就是标准 Oauth2.0 中的授权码模式（authorization _ code）。通过对标准授权码模式的授权流程的安全性分析，采用了基于信任机制的改进授权流程[17]，其授权流程如图 3 所示。

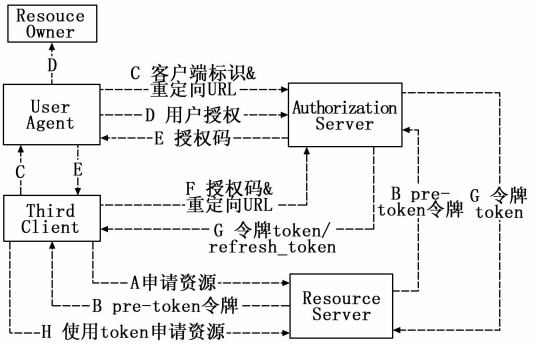

图 3 改进的授权码模式授权流程

相比于 Oauth2.0 中的标准授权码模式流程中，图 3 中的授权流程引入了资源服务器（Resource Server）和授权服务器（Authorization Server）之间的同步信任机制，也就是在原有流程中增加了同步信任表，该信任表与安全节点相互配合可以防止客户端与授权服务器端，用户通信时被监听盗取令牌。当客户端申请资源的时候，资源服务器为其颁发一个 pre－token 令牌，同时授权服务器也会同步得到这个 pre－token。当客户端得到用户授权后，去授权服务器申请资源的时候，会先去访问安全节点，安全节点检查是否存在 pre－token 与用户对应，如果存在，则表明这个用户确实有资源请求的要求，允许该客户端访问授权节点，进而颁发访问令牌 token。

2.2.2 高并发限流及熔断机制的设计

系统中流量管控模块设计了基于令牌桶算法[18]的限流策略，来实现对高并发下外部请求的流量控制。具体的设计思路就是根据令牌桶算法的特点，以一个恒定可配置的速度往桶里放入令牌，当 HTTP 请求到达时，如果该请求需要被处理，就从桶中取出一块令牌，路由的时候首先判断当前请求是否具有令牌，有则路由到下游 API 服务，没有则不进行路由。如果桶里没有令牌可取，那么后面的请求则需要排队等待，直到桶里有令牌才能进行后续操作。其流程图如图 4 所示。

如图 4 所示，令牌桶中当前持有的令牌数量为 x，并且以每秒 n 个令牌的速率往桶中放入令牌，这个速率支持自定义配置。当每个外部请求到达流量控制模块，需要从令牌桶中取出一块令牌，然后路由模块才会将这些持有令牌的请求路由到下游 API 服务中。如果没有令牌，则不进行

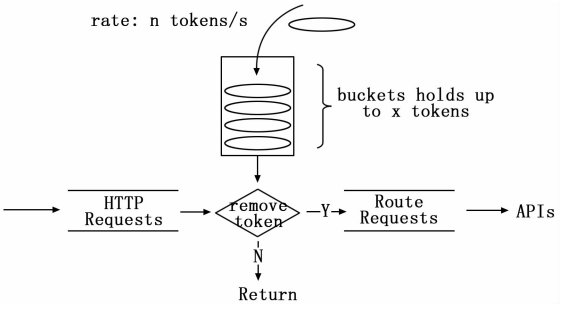

图 4　基于令牌桶算法的限流策略

路由，直接返回。

当开放平台某个 API 请求过于集中而导致无法响应或是响应缓慢，如果没有设计合理的处理机制，最终将会导致整个下游 API 不可用。本框架中服务熔断模块设计了熔断保护机制，在外部请求和下游 API 之间起到了“保险丝”的作用，也提升了系统的稳定性。其工作流程如图 5 所示。

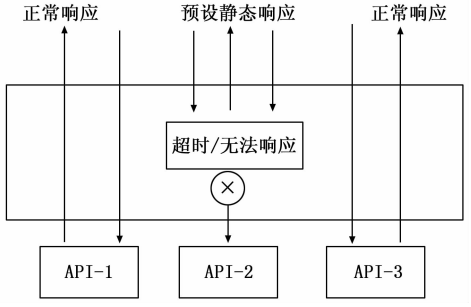

图 5　服务熔断处理机制

图中外部请求进入开放平台对 API－1，API－2，API－3 进行访问时，API－2 由于多次调用均失败（出现超时或者无法响应），这时系统就会对 API－2 进行服务熔断，及时返回预设的静态响应结果。API－1 和 API－3 由于没有被熔断，则返回正常的响应结果。为了保证服务熔断后能够重新连接并正常访问，该模块还设计了心跳检测机制，每隔一个时间间隔对 API－2 进行检测，如果能够正常响应，那么就关闭熔断，以保证 API－2 能够重新进行访问。

在负载均衡模块中，设计多组合方式的负载策略，该策略通过可配置的一个或者多个负载均衡策略的组合，选择相对合适的服务实例，对请求进行转发，实现平台内部的二级软负载。设计中默认的负载均衡策略包括：

1）选择一个并发请求最小的 Server 进行转发。

2）根据响应时间为每个服务的实例分配一个权重，响应时间越长，权重越小，被选中的可能性也越小。

3）以轮询的方式进行选择，对于同一个服务的每个实例，分配一个索引 index，然后对 index 进行轮询，选择对应索引的服务进行转发。

4）随机选择一个服务实例，进行转发。

配置多种负载均衡策略，为开放平台自身提供负载能力，降低了平台外部一级负载的压力，同时还支持设计自定义的负载策略，扩展负载均衡策略库，提高开放平台的性能。

2.2.3　基于注册表的可扩展性设计

可扩展性主要通过两个方面的设计来实现，一个是核心模块的可扩展性，另一个则是 API 服务的横向可扩展性。通过这两个方面的灵活设计，保证了开放平台在后续迭代和升级中可以灵活的进行扩展。

根据设计模式中单一职责的原则，每一个模块仅负责一个功能，各个模块之间没有相互依赖关系，它们都是独立的存在，降低了模块之间的耦合程度。通过各个模块的聚合，使得它们之间又相互协作，实现开放平台对于不同情境下的不同需求，保证了核心模块的可扩展性。另一方面，设计了一个长度可变的注册表，由于注册表中包含了每个 API 服务的名称，地址，多实例部署模式下各个实例对应的索引等信息，通过维护这个注册表，开放平台就能将外部请求路由到具体的服务中去。其中同步下线服务（cancel（）方法），同步注册服务（register（）方法），同步续约服务（renew（）方法）三个方法代表了三个交互行为，通过这三个方法来维护后端服务和开放平台关联关系。当开放平台需要新增后端服务的时候，通过 register（）方法即可将服务添加到注册表中进行维护即可；当需要从注册表中解除关联关系，通过 cancel（）方法就能从开放平台注册表中下线该服务；同时还采用了心跳机制通过 renew（）方法来检测服务是否续约，以决定其是否满足继续存留在注册表中的条件。注册表的动态可变特性，满足了 API 服务的横向可扩展性。

3　安全可扩展开放平台安全性及性能分析

3.1　安全性分析

开放平台将一系列的服务数据接口对外开放，接口暴露在外部导致会存在诸多的安全性问题。因此，为保证开放平台安全性，本框架设计了安全管理模块，用于过滤那些恶意的无效的请求，防止恶意调用 API 造成平台接口的安全问题。对于外部请求做了严格的权限控制，只有当第三方调用者被授权之后，开放平台才会对这些请求进行路由。

在安全管理模块的基础上，设计了独立的授权模块，通过改进 Oauth2.0 中的一些安全问题，针对客户端授权和用户授权两种类型的授权流程进行了安全改进。在客户端授权流程中，根据 HMAC 算法的特点，加入了 Hmac-SHA512 摘要算法，对请求 token 时候的敏感数据做了加密处理，以“算法”＋“空格”＋“AppKey”＋“:”＋“摘要值”最为最终进行传递的数据，保证了客户端在申请 token 的时候不会暴露密钥，避免了客户端密钥被窃取的风险；在用户授权流程中，采用改进的 Oauth2.0 授权流程，引入了信任机制和安全节点，实现可信授权及节点的安全控制，防止了钓鱼攻击和信道监听而导致的安全隐患。

同时，实现监控模块，对第三方的调用记录进行实时

的审计与监督，一旦发现异常的调用，可以及时采取相应的措施进行限制和禁用。因此，本框架从授权认证、接口信息加密处理及审计记录多个维度保证了开放平台的安全性问题。

3.2 稳定性与性能分析

为保证高并发的开放平台系统稳定性，从外部请求和内部服务两个方面来进行了改进。对于外部采取了限流的措施，通过限流管理模块，设计了基于令牌桶算法的限流策略，当外部请求的速率大于放入令牌桶中令牌的速率，导致桶中无令牌可用时，就进行流量限制，避免因过载而导致平台崩溃。为了防止个别 API 响应缓慢或无法提供服务时，由于大量的超时等待而导致整个开放平台堵塞，以及后端 API 来不及处理和响应大量的请求的情况出现，设计了服务熔断模块和负载均衡模块。多服务实例部署的方式能防止个别服务实例 down 掉的时候其他实例可以正常运行，以保证平台内部服务可以稳定的运行。

熔断机制在外部请求和 API 之间起到了保险丝的作用，对下游 API 进行保护的同时，当 API 无法响应或者响应超时的时候，实时的为客户端返回预设的静态响应，避免了客户端一直等待，对系统造成堵塞，影响开放平台整体的性能。载均衡模块设计了策略库，提供了默认的几种负载均衡策略，同时还支持扩充策略库。当外部请求进入的时候，负载均衡模块选择合适的负载策略，将请求分发到下游 API 服务实例上，避免了由于压力过大而出现某个服务实例崩溃的情况。

3.3 平台扩展性分析

对于一个 SaaS 服务的开放平台来说，必须要具备良好的扩展性，才能更好的支撑面向多租户的 SaaS 服务。针对传统 SaaS 开放平台扩展性不强的缺陷，在设计开放平台框架的时候，充分考虑了平台的可扩展性。根据设计模式中低耦合高内聚的设计原则，设计了多模块可组合的开放平台功能组件，保证了开放平台基础功能的可扩展性。通过自定义模块，可以根据实际需求，丰富开放平台的基础模块；为保证下游 API 能够方便的进行横向扩展，设计了注册表来维护下游 API 服务实例与开放平台的关联，使得下游 API 服务可以自由的进行扩展，这为 SaaS 扩展自身的业务提供了极大的便利。

4 总结

根据 SaaS 服务的特点和对开放平台的需求，设计了一个安全可扩展的 OPBG 框架，为 SaaS 服务搭建开放平台提供一个可参考的解决方案。兼顾了开放平台所需要的稳定性和高并发性，同时根据 Oauth2.0 标准授权流程中存在的安全隐患，对授权流程做了改进，提高了开放平台的安全性保障，松耦合的模块设计也提高了开放平台的灵活性与可扩展性。为 SaaS 服务开放平台提供一个基础的解决方案，还有一些有待改进的地方，比如对于一些热点数据进行特殊的处理，提高系统的效率。安全管理方面，对于访问控制策略和其他的安全限制及保障都有待提高和研究。

参考文献：

[1] Kang S，Kang S，Hur S. A Design of the Conceptual Architecture for a Multitenant SaaS Application Platform [A]. Acis/jnu International Conference on Computers [C]. IEEE Computer Society，2011：462－467.

[2] 李 森. 浅析基于 SaaS 架构的多租户技术 [J]. 电子设计工程，2013，21 (20)：41－44.

[3] 朱蔚恒，周 伟，龙 舜. 开放平台解决方案及其安全策略研究 [J]. 计算机工程，2012，38 (12)：265－267.

[4] 周巧俊. RESTful Web 服务开放平台的设计与实现 [D]. 杭州：浙江大学，2016.

[5] Facebook. Facebook Developers [EB/OL]. (2018－05－15). http：//de velopers. facebook. com.

[6] 淘宝开放平台详细介绍 [EB/OL]. [2018－05－20]. http：//open. taobao. com/.

[7] 百度开放平台详细介绍 [EB/OL]. [2018－05－26]. http：//open. baidu. com/.

[8] Hardt D. The OAuth 2. 0 Authorization Framework [A]. Internet Engineering Task Force (IETF) [C]. 2012：4－70

[9] Fielding，Thomas R. Architectural styles and the design of network－based software architectures [D]. University of California，Irvine，2000：303.

[10] 高嘉阳. Web 开放平台安全机制的研究与设计 [D]. 北京：北京邮电大学，2009.

[11] Yan H，Fang H，Kuka C，et al. Verification for OAuth Using ASLan＋＋ [A]. IEEE，International Symposium on High Assurance Systems Engineering [C]. IEEE，2015：76－84.

[12] Pai S，Sharma Y，Kumar S，et al. Formal Verification of OAuth 2. 0 Using Alloy Framework [A]. International Conference on Communication Systems and Network Technologies [C]. IEEE Computer Society，2011：655－659.

[13] Bansal C，Bhargavan K，Maffeis S. Discovering Concrete Attacks on Website Authorization by Formal Analysis [A]. IEEE，Computer Security Foundations Symposium [C]. IEEE Computer Society，2012：247－262.

[14] 王焕孝，顾纯祥，郑永辉. 开放授权协议 OAuth2. 0 的安全性形式化分析 [J]. 信息工程大学学报，2014，15 (2)：141－147.

[15] Li J，Wu L，Zhang X. An efficient HMAC processor based on the SHA－3 HASH function [A]. IEEE，International Conference on Asic [C]. IEEE，2017：252－255.

[16] 须 磊. HMAC－SHA256 算法的优化设计 [J]. 价值工程，2012，31 (29)：202－204.

[17] 魏成坤，刘向东，石兆军. OAuth2.0 协议的优化方法 [J]. 计算机工程与设计，2016，37 (11)：2949－2955.

[18] 仲思惠. 基于令牌桶算法的流量控制服务的设计与实现 [D]. 大连：大连理工大学，2016.