计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(10):70−74 [doi: 10.15888/j.cnki.csa.006568]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

# 柔性微服务安全访问控制框架①

刘一田[1], 林亭君[2], 刘士进[1]

[1](南瑞集团 (国网电力科学研究院) 有限公司, 南京 211106)
[2](河海大学 能源与电气学院, 南京 210003)
通讯作者: 刘一田, E-mail: liuyitian@sgepri.sgcc.com.cn

**摘　要**: 微服务架构实现了应用服务的业务解耦和技术栈分离, 但更多的微服务也增加了进程间无状态服务调用频度, 如何在保证服务性能的同时确保无状态服务之间的安全访问控制是微服务安全架构面临的关键问题. 本文设计了一种柔性微服务安全访问控制框架, 结合微服务 API 网关、轻量级微服务访问令牌构建方法以及柔性适配的微服务安全控制策略等特征, 提高了微服务的柔性安全控制能力, 经试验分析, 代价更小, 并在实际项目中验证了框架及方法的有效性.

**关键词**: 微服务 API 网关; 服务访问令牌; 柔性安全访问控制策略

引用格式: 刘一田,林亭君,刘士进.柔性微服务安全访问控制框架.计算机系统应用,2018,27(10):70–74. http://www.c-s-a.org.cn/1003-3254/6568.html

## Flexible Microservice Security Access Control Framework

LIU Yi-Tian[1], LIN Ting-Jun[2], LIU Shi-Jin[1]

[1](NARI Group Corporation (State Grid Electric Power Research Institute), Nanjing 211106, China)
[2](College of Energy and Electrical Engineering, Hohai University, Nanjing 210003, China)

**Abstract**: The microservice architecture facilitates the service decoupling of application services and the separation of the technology stack. However, more microservices also increase the frequency of stateless service invocation across processes. How to ensure secure service access control between stateless services while ensuring service performance is a key issue for the microservices security architecture. In this study, we design a flexible microservice security access control framework. Combining the features of microservice API gateway, the lightweight microservice token construction mechanism and the flexible adaptation of microservices security control strategy, we improve the flexible security control ability of microservice. After the experimental analysis, the cost is smaller, and the validity of the framework and the method is verified in the actual project.

**Key words**: microservice API gateway; service access token; flexible safety access control strategy

微服务架构和其开源技术栈日益成熟, 已逐步成为互联网及企业级项目中的主流技术架构, 随着基于微服务架构的领域服务设计细化, 产生了更多的独立微服务进程, 进程间的微服务调用频度更加频繁, 无状态的微服务调用请求每次都需要鉴权, 在大量用户并发访问时会引发鉴权服务的性能瓶颈, 如何在保证服务性能的同时确保无状态服务之间的安全访问控制是微服务安全架构面临的关键问题.

目前主流的四种解决方案及相应问题包括: (1) 单点登录. 每次服务访问前必须与认证服务交互, 这会产

① 基金项目: 南瑞集团有限公司 (国网电力科学研究院有限公司) 科技项目“柔性微服务框架关键技术研究与应用”
Foundation item: Tech Project of NARI Group Corporation (State Grid Electric Power Research Institute) “Research and Application of Key Technology in Flexible Microservice Framework”
收稿时间: 2018-02-26; 修改时间: 2018-03-19; 采用时间: 2018-03-26; csa 在线出版时间: 2018-09-28

生大量琐碎的网络流量和重复的工作, 当微服务规模增大时影响比较显著. (2) 分布式会话. 将用户认证的信息存储在共享存储中, 且通常由用户会话作为查询键来实现的分布式哈希映射, 当用户访问微服务时, 用户数据可以从共享存储中获取. 这种方案的缺点在于共享存储需要一定保护机制, 需要通过安全链接来访问, 实现复杂度高, 性能有明显损耗. (3) 客户端令牌. 令牌在客户端生成, 由身份验证服务进行签名, 并且必须包含足够的信息, 以便可以在所有微服务中建立用户身份. 令牌会附加到每个请求上, 为微服务提供用户身份验证, 这种解决方案的安全性相对较好, 但身份验证注销时需要使用短期令牌和频繁检查认证服务, 性能略有损耗. (4) 客户端令牌和 API 网关相结合. 所有服务请求首先路由到 API 网关, API 网关将请求的原始用户令牌转换为内部会话令牌, 有效地避免了身份注销时令牌问题. 上述方案中相比第四种方案有明显的优势, 但认证时服务鉴权的粒度和频度不能平衡, 缺乏柔性适配的安全访问控制策略.

本文设计的柔性微服务安全访问控制框架在上述第四种解决方案的基础上进行了改进和提升. 引入支持常规安全策略和自定义安全策略的安全访问控制策略模板, 更好地实现柔性安全访问控制, 并通过统一配置服务和消息总线服务实现安全策略的即时发布和应用. 通过服务容器化后的容器安全策略加固服务边界; 在微服务 API 网关中采用分层拦截过滤的方式, 通过前置过滤、路由、后置过滤和异常处理等环节结合安全策略更好地安全访问控制和容错; 在服务访问令牌方面, 采用 RFC 7519[1]标准协议格式, 结合安全访问控制策略, 动态生成轻量级服务访问令牌, 在增强安全访问控制的易操作性和性能的同时, 提升了微服务安全访问控制的柔性. 考虑到基于容器的微服务构建正逐步成为微服务架构落地部署的主要模式, 框架采用容器化安全访问控制策略以增强微服务安全访问控制能力。

## 1 框架设计

本文设计的柔性微服务安全访问控制框架结构如图 1 所示. 分别从服务安全、数据安全、虚拟化安全等多维度增强微服务柔性安全控制, 其主要具备如下特征.

(1) 框架设计高可用. 整个系统无核心单点, 易运维, 易部署.

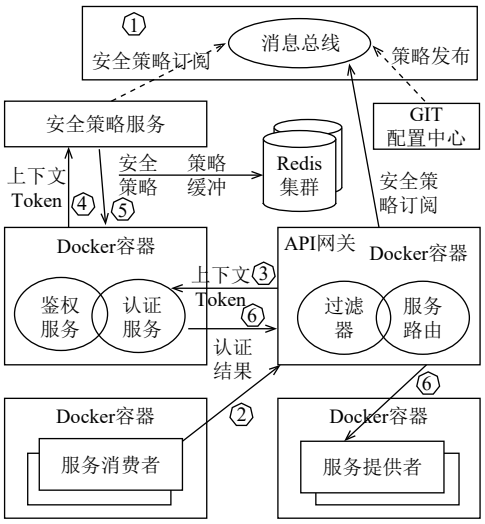
(2) 安全访问控制策略灵活和自发现. 支持策略模板、模板继承和覆盖.

(3) 框架的微服务 API 网关采用 HTTPS 协议路由请求, 并根据安全访问控制策略审计和过滤不安全的服务访问, 屏蔽不符合安全访问控制策略的请求. 在请求高并发时, API 网关采用令牌桶算法进行限流, 确保系统的可用性和性能.

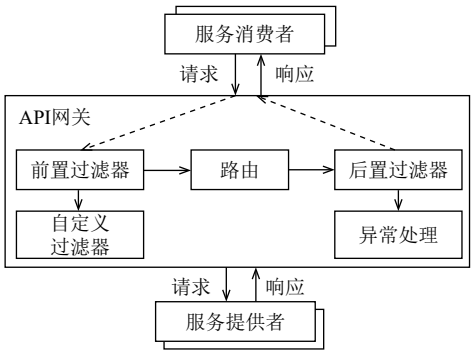
(4) 服务访问令牌轻量高效, 实现无状态会话机制. 结合服务访问令牌和请求刷新令牌兼顾服务安全和性能.

(5) 服务访问控制操作管理更精细, 支持为每个服务 API 提供按需认证方式.

(6) 基于虚拟宿主环境内部安全策略构建安全信任边界[2]. 使用基于 Docker 容器宿主操作系统的脚本语言快速创建和配置私有网络, 在网络安全级别上执行强大的安全策略.




(a) 柔性微服务安全框架总体结构




(b) API网关内部安全过滤结构

图 1 柔性微服务安全框架结构

柔性微服务安全框架执行流程如下.

(1) GIT 配置中心统一维护安全策略配置, 配置的变更自动发布到消息总线的约定主题上, 配置中心支持集群、负载均衡及高可用. 安全策略服务定时从消息总线订阅安全策略配置. 安全策略配置中以正则表达式描述了请求 URL 规则的安全策略映射.

(2) 服务消费者请求首先经由微服务 API 网关, 在 API 网关前置过滤器中以 POST 方式提交用户认证信息到认证服务.

(3) 认证服务首先和数据库信息匹配, 匹配成功后调用安全策略服务获取请求 URL 对应的安全访问控制策略, 根据安全策略逻辑动态生成加密签名的服务访问令牌并返回给 API 网关, API 网关将服务访问令牌添加到请求头中并路由到服务调用者.

(4) 服务提供者在接收到带有服务令牌请求头的服务调用后, 直接由目标服务根据约定的加密私钥对令牌进行验签和校验, 校验通过后则确认令牌有效则正常处理请求并返回响应, 否则返回禁止访问的响应信息.

(5)API 网关根据返回的认证结果做下一步的后置过滤处理或异常处理, 并将处理结果返回给请求的服务.

(6) 每一个微服务都构建为独立的 Docker 镜像, 通过 Kubernetes[3]创建 Docker 容器, 并为所有容器批量基于 Docker 镜像中 linux 内核操作系统安全策略构建自定义安全策略描述文件, 实现微服务宿主安全.

## 2 框架实现

柔性微服务安全访问控制框架采用可扩展的安全访问控制策略模板, 除了常规安全策略外, 支持 JAVA BEAN 和安全策略表达式的定义扩展. 策略模板的变更通过 GIT 配置中心发布, 以发布/订阅模式发送到消息总线的指定主题, 安全访问控制策略服务订阅后即时更新安全策略. 认证授权服务根据访问 URL 匹配的安全访问控制策略生成轻量级服务访问令牌, 服务提供者接收到访问令牌后直接解析鉴权, 并进行相应的路由或异常处理. 根据上述框架的设计, 需要实现五个主要关键技术组件: (1) 高可用的统一配置管理中心和消息总线. (2) 灵活可复用的安全访问控制策略模板. (3) 轻量级服务访问令牌. (4) 便捷的微服务 API 授权管理. (5) 服务容器化, 应用容器安全策略.

(1) 高可用统一配置中心和消息总线

框架通过基于 GIT 服务器的 GITLAB[4]配置管理视图编辑安全访问控制策略配置, 配置完成后提交到高可用统一配置中心, 配置管理中心采用分布式一致性 Raft[5]算法以保障统一配置中心集群环境下负载均衡、统一配置的一致性及高可用, 实现配置管理集群实例在节点故障重启时对请求透明化. 集群中的节点分为 Leader、Follower、Candidate 三种角色, 由 leader 响应客户端请求并确保响应结果的一致性. 统一配置管理节点集群首先选举一个 Leader 节点, 由 Leader 节点对外提供服务, 当配置更新时, Leader 节点发出命令, 并在确保集群中多数节点都已完成命令操作后, 发布最终确认配置值到消息总线. 如果一个 Follower 在选举超时的时间周期里没收到 leader 的信息, 就进入新的选举周期, 自身转成候选人 Candidate 角色, 给自己投票, 发起选举, 并重写产生 Leader 以继续提供高可用一致性服务. 如图 2 所示.

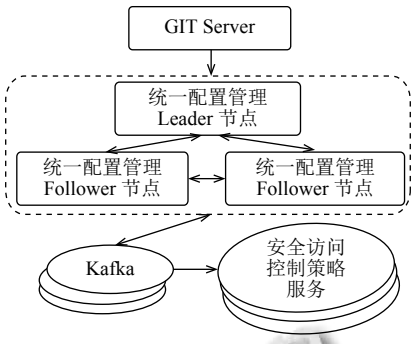



图 2 高可用统一配置中心和消息总线结构

为了实现在分布式环境下的即时消息通知和高性能传输, 框架采用高性能分布式消息总线 Apache Kafka[6]实现安全访问控制策略的发布/订阅, 使用 3 个服务器节点组成集群以保障消息总线的高可用, 通信模式采用发布/订阅模式, 使消息按照指定主题分发, 安全策略访问控制服务订阅消费.

(2) 灵活的安全访问控制策略模板

安全访问控制策略访有三种类型: 基于身份的安全策略、基于规则的安全策略和综合访问控制方式. 访问控制安全策略原则集中在主体、客体和安全控制规则集三者之间的关系. 框架中策略描述文件以 YAML[7]格式进行描述, 包括相互关联的 4 部分: 入站、路由、出站、异常处理. 这四部分中分别提供了不同的安全策略, 入站部分的安全策略包括基于服务访问令牌进行授权访问、请求参数加密、请求头部设置关联标识、目录级安全控制、属性安全控制等策略; 路由部分安全策略包括服务监测和锁定控制; 出站部分的安

全策略包括跨域策略、缓存策略、跳转策略等. 安全策略以 URL 正则表达式的方式定义了服务和安全策略的映射, 并支持自定义安全策略的处理 Bean 注入. 表 1 中的安全策略描述了入站的令牌生成表达式、令牌验证表达式对令牌及状态进行认证鉴权后的处理方式.

表 1 安全访问控制策略模板描述

```
---
        policies:
          inbound:
            set-var:
              token: @(expression)
            send-request:
              response-var-name: tokenstate
              timeout: 20
              ignore-error: true
              set-url:
              set-method: POST
              set-header: Authorization
              set-body: @($"token={(string)context.var["token"]}")
              choose:
                condition:
                @((bool)((IResponse)context.var["tokenstate"])["active
                "] == false)
                f-val:
                    return-response:
                        set-status: 401
                        set-header: invalid_token
          route:
            when:
          outbound:
            forward-request:
              follow-redirects: true
            cache: true
          exception:
            …
```

(3) 轻量级服务访问令牌

框架采用的轻量级服务访问令牌支持无状态的会话应用, 使用签名和加密来验证和保护会话内容. 服务访问令牌由三部分组成: 头信息（header）, 消息体（payload）和签名（signature）, 头信息指定了该令牌使用的签名算法, 如表 2 所示. 消息体包含了访问者的凭证等授权信息. 由于客户端数据可能被篡改, 因此, 框架采取密钥签名技术验证令牌的有效性, 防止恶意攻击, 框架提供了基于国密 SM3 的签名算法和 SM2 加密算法实现[8]. 为了避免 CSRF 和 XSS 攻击, 同时避免频繁更新令牌的性能损耗, 框架会定时监测令牌有效期, 并在令牌过期后的下次服务请求时通知服务消费者, 服务消费者通过刷新令牌申请新的服务访问令牌以确保安全服务访问控制有效性.

表 2 框架轻量级服务访问令牌格式描述

```
header:
{"alg": "SM3","typ": "JWT"},
payload:
{
  "sub": "piadmin",
  "scopes": [
   "ROLE_ADMIN"
  ],
  "iss": "http://pi6000.com",
  "iat": 1472033308,
  "exp": 1472035208
},
signature:
41rxtplFRw55ffqcw1.mPSWSgQgsR0NLndFcMPh7LSQt5mkYqROQ
```

(4) 便捷的微服务 API 授权管理

微服务 API 授权管理组件列出给出了所有可用的服务及操作资源, 选中左侧的服务操作, 在右侧授权设置中选择授权策略或自定义策略, 如图 3(a) 所示, 支持设置服务中的 URL 查询参数和请求头等特殊参数以更好地满足个性化服务认证鉴权需求. 自定义授权配置如图 3(b) 所示, 允许指定服务访问角色、身份令牌来源、身份验证表达式及秒级令牌有效期, 其中身份令牌来源指定令牌以何种方式传递给客户端, 包括请求头授权、Cookie 授权、Ajax 授权等三种方式, 令牌验证表达式默认分别对应不同身份令牌进行配对使用, 并允许自定义令牌验证表达式以个性化授权认证方式.

(5) 服务容器化, 应用容器安全策略

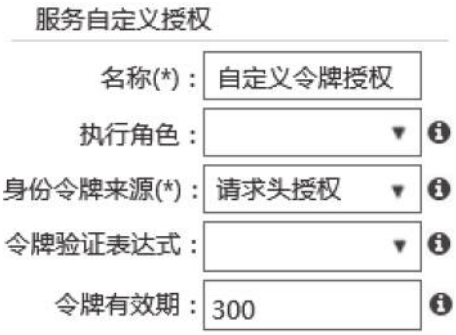
微服务的安全防火墙依赖其所宿主的虚拟化容器及网络环境, 在部署微服务容器的同时, 需要做好已知服务宿主容器安全攻击的防御工作, 从而保护微服务及数据的安全性. 容器安全始于宿主机的操作系统, 框架选择使用最小 Linux 发行版 Alpine Linux 作为微服务容器镜像构建的宿主操作系统, 并采用强制访问控制模式, 从操作系统角度应用最小访问原则, 使用按需定义的 Grsecurity[9]安全策略, 通过支持自动学习训练生成的基于路径的访问控制列表, 限制宿主机访问权限, 并通过 IP 限流等网络安全策略加固微服务安全边界.

## 3 案例分析

为了对本文提出的柔性微服务安全访问控制框架的有效性和性能进行验证, 在一个真实的环境中进行了案例的部署和研究. 案例环境由 16 台 8 核 16G 内存的 PC 服务器 (HPDL 380 G4 378735-AA1) 组成, 其中数据库、认证鉴权微服务、应用微服务、安全访问控

制策略微服务、统一配置管理微服务、消息总线微服务节点比例为 1:1:2:1:3:1, 数据库和应用服务采用国家电网公司运检智能分析管控系统的测试数据库和经过拆分的微服务, 微服务实例以 Kubernetes 构建的 Docker 容器形式运行. 通过 loadrunner 模拟并发 800 用户的微服务访问, 通过观察柔性微服务安全访问控制框架的效率和稳定性, 评估服务访问令牌对应用的性能影响, 试验结果如表 3 所示.


(a) 微服务API授权策略定义




(b) 微服务API自定义授权

图 3 微服务 API 授权管理

表 3 服务访问令牌对微服务性能的影响

| 处理器数 | 每秒处理请求数 | CPU 占用 (%) |
|---|---|---|
| 4 | 3000 | 0.025 |
| 8 | 12 000 | 0.075 |
| 16 | 35 000 | 0.045 |

通过 BurpSuite[10]扫描微服务应用, 渗透攻击已发布的微服务接口. 验证结果表明, 设计的柔性微服务安全访问控制框架符合国家标准中信息系统安全等级保护第二级要素要求[11], 在大并发吞吐量时保证了微服务安全性、服务高可用和高效, 但也发现了一些问题, 针对小规模的应用现有框架的部署架构略显复杂, 服务访问令牌会在生产模式下会产生微小的性能损耗, 尚有提升空间. 目前, 柔性微服务监控框架已在国网运检智能分析管控系统中实现并应用, 取得了较好的应用效果.

## 4 结论与展望

本文研究了微服务安全访问控制的策略、令牌机制和容器化安全边界技术, 在此基础上, 设计了柔性微服务安全访问控制框架, 给出了高可用统一配置中心和消息总线、轻量级服务访问令牌、精细访问控制、服务容器化并应用 Grsecurity 策略增强微服务安全边界等创新点, 阐述了该框架的架构设计及关键实现技术. 最后, 以国网运检智能分析管控系统的微服务应用案例为背景, 给出了柔性微服务安全访问控制框架的应用验证评估, 验证结果、效率评估及生产运行实践表明, 该框架提升了分布式系统中微服务安全访问控制的灵活度、效率和柔性, 提高了电网信息系统的服务水平. 后续将针对遗留问题持续改进优化该框架.

## 参考文献

1 RFC7519. JSON Web Token (JWT). https://tools.ietf.org/html/rfc7519. [2015-05-01].
2 Sun YQ, Nanda S, Jaeger T. Security-as-a-service for microservices-based cloud applications. 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom). Vancouver, BC, Canada. 2015. 50–57.
3 Bernstein D. Containers and cloud: From LXC to docker to kubernetes. IEEE Cloud Computing, 2015, 1(3): 81–84.
4 Wildish T, Udwary D. Advanced git and gitlab. https://www.nersc.gov/assets/Uploads/Advanced-Gitlab.pdf. [2017-05-30].
5 Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference. Philadelphia, PA, USA. 2014. 305–320.
6 Apache Kafka Project. A distributed streaming platform. http://kafka.apache.org/intro.
7 Ben-Kiki O, Evans C, Net ID . YAML ain't markup language (YAML™) version 1.2. 2009. http://www.yaml.org/spec/1.2/spec.html. [2009-10-01].
8 孙荣燕, 蔡昌曙, 周洲, 等. 国密 SM2 数字签名算法与 ECDSA 算法对比分析研究. 网络安全技术与应用, 2013, (2): 60–62. [doi: 10.3969/j.issn.1009-6833.2013.02.021]
9 Grsecurity. Grsecurity adds confidence to containers. https://grsecurity.net, 2018.
10 俞诗源, 王誉天, 刘鑫. Burpsuite 工具在漏洞检测中的应用. 信息网络安全, 2016, (9): 94–97. [doi: 10.3969/j.issn.1671-1122.2016.09.019]
11 中华人民共和国国家质量监督检验检疫总局, 中国国家标准化管理委员会. GB/T 22239-2008 信息安全技术信息系统安全等级保护基本要求. 北京: 中国标准化出版社, 2008.