(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 118540169 B
(45)授权公告日 2024.10.29

(21)申请号 202411011776.3
(22)申请日 2024.07.26
(65)同一申请的已公布的文献号
申请公布号 CN 118540169 A
(43)申请公布日 2024.08.23
(73)专利权人 成都云祺科技有限公司
地址 610041 四川省成都市高新区云华路333号8栋4-5层
(72)发明人 谢卓伟 王功喜 姜咏杰 郑相琴 罗凯 刘帅
(51)Int.Cl.
H04L 9/40(2022.01)
H04L 9/32(2006.01)
(56)对比文件
CN 106790238 A,2017.05.31
CN 110858836 A,2020.03.03
CN 112612508 A,2021.04.06
CN 115357282 A,2022.11.18
审查员 张焕娜

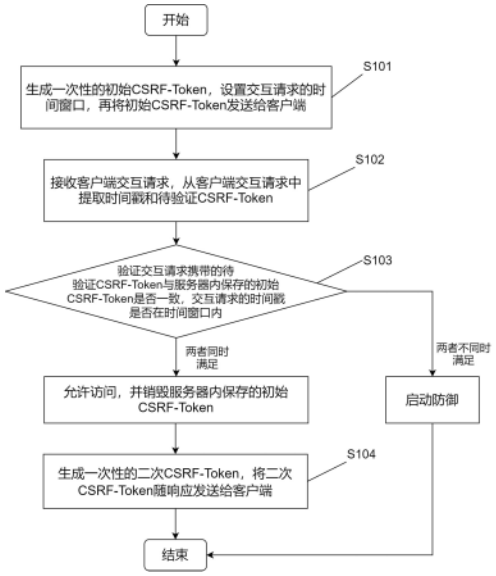
权利要求书2页 说明书9页 附图6页

(54)发明名称

API安全实现方法、系统、介质及API框架实现方法

(57)摘要

本发明涉及一种API安全实现方法、系统、介质及API框架实现方法,属于计算机技术领域。用于服务器端方法包括:验证配置及发送步骤,提取步骤;验证步骤;重新发送CSRF-Token步骤。用于客户端方法包括:接收初始CSRF-Token步骤;发送交互请求步骤;接收响应及二次CSRF-Token步骤。本发明方法基于一次性的CSRF-Token和时间戳验证用户交互请求,相比于现有技术中仅用时间戳进行验证,可以有效地防御重放攻击,且配置方便,不增加额外开销,交互及验证过程几乎无感。



CN 118540169 B

1.一种API框架实现方法,其特征在于,用于服务器端,包括步骤;

API框架创建步骤,在服务器创建具有统一API入口和路由池的API框架,API框架内设有若干相互隔离的主版本目录,每个主版本目录内设有若干相互隔离的次版本目录;路由池保存有请求匹配路径与指定版本目录之间的映射关系规则;

版本调用请求接收步骤,API入口接收到客户端版本调用请求,版本调用请求包括URL路径和指定的HTTP头,或者URL路径和查询参数;版本调用请求还包括时间戳和待验证CSRF-Token;

解析路由步骤,API框架解析版本调用请求,再按照路由池映射关系规则将版本调用请求路由到目标版本目录;

响应步骤,根据路由结果,API框架调用目标版本目录处理客户端版本调用请求,再形成响应发回客户端;

其中,所述API框架创建步骤,还包括:

API框架初步创建步骤,在服务器端创建API框架,所述API框架具有统一的API入口,用于接收和响应外部请求;

版本目录创建步骤,在API框架内创建若干相互隔离的主版本目录,每个主版本目录内创建通用配置信息目录和次版本目录,通用配置信息目录与次版本目录之间相互隔离,其中,通用配置信息目录用于存放所有次版本以及修订版本共享的配置信息,次版本目录内设有第一控制器,该控制器负责处理客户端请求并返回响应,包含业务逻辑和数据访问逻辑,版本调用请求的路径是解析到控制器;

版本控制设置步骤,在API入口路径中设置版本控制,用于支持操作不同版本的API;

路由池设置步骤,在API框架内创建路由池,在路由池内保存若干请求匹配路径,将请求匹配路径与指定版本目录之间一一对应,形成映射关系规则;

其中,对于API框架的API调用防护还包括步骤:

验证配置及发送步骤,生成一次性的初始CSRF-Token,设置交互请求的时间窗口,再将初始CSRF-Token发送给客户端;

提取步骤,接收客户端交互请求,从客户端交互请求中提取时间戳和待验证CSRF-Token;

验证步骤,验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内;若两者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若两者不同时满足,则启动防御;

重新发送CSRF-Token步骤,生成一次性的二次CSRF-Token,将二次CSRF-Token随响应发送给客户端。

2.根据权利要求1所述的API框架实现方法,其特征在于,所述验证配置及发送步骤,还包括:

所述初始CSRF-Token是以隐蔽的方式发送给客户端,发送的通道为加密通道;

所述重新发送CSRF-Token步骤,还包括:

所述二次CSRF-Token是以隐蔽的方式发送给客户端,发送的通道为加密通道。

3.根据权利要求1所述的API框架实现方法,其特征在于,在所述验证配置及发送步骤之前,还包括:

安全字典设置步骤,为每个API端点设置安全字典,安全字典规定服务器仅接收offset和limit两个参数,且每个参数的值为正整数;

所述验证配置及发送步骤,还包括:将初始CSRF-Token与指定API端点的请求关联;

所述提取步骤,还包括:接收客户端交互请求,从客户端交互请求中提取时间戳、CSRF-Token、API入参和API入参值;

所述验证步骤,还包括:验证交互请求携带的CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内,API入参和API入参值是否符合安全字典的规定;若三者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若三者不同时满足,则启动防御。

4.根据权利要求1所述的API框架实现方法,其特征在于,所述版本目录创建步骤,还包括:

通用配置信息目录设置步骤,所述通用配置信息目录内创建有配置文件目录、说明目录和语言目录;

业务逻辑扩展目录设置步骤,所述次版本目录内创建有控制器目录和业务逻辑扩展目录,所述业务逻辑扩展目录内创建有修订版本目录,修订版本目录存放有第二控制器。

API安全实现方法、系统、介质及API框架实现方法

技术领域

[0001] 本发明属于计算机技术领域,涉及一种API安全实现方法、系统、介质及API框架实现方法。

背景技术

[0002] 重放攻击(Replay Attack),也称为复制攻击或回放攻击,是一种网络攻击技术,攻击者截获有效的通信数据,然后选择时间点重新发送这些数据,以尝试获取未经授权的访问权限或执行未授权的操作,是一种威胁性较大的攻击方式。

[0003] API(应用程序编程入口)是软件系统之间进行交互的重要方式,在一些应用场景中,客户有通过API与开发商的产品或服务进行交互完成自身业务的需求。客户调用API的活动是频繁的,API对外又常是暴露的,在该交互过程中进行重放攻击的防御对于API的安全来说至关重要。

[0004] 目前,传统技术是利用时间戳验证的方式来进行防御,客户端在生成请求时,加入当前的时间戳;服务器接收到请求,并从中提取时间戳,服务器检查时间戳与服务器当前时间的差异,服务器会设置一个时间窗口,例如5分钟或更短的时间,以确定请求是否在合理的时间内发起,如果时间戳显示请求是在时间窗口之外发送的(即与服务器当前时间的差异超出了设定的阈值),服务器将拒绝该请求,通过一个时间敏感的元素来进行安全验证,可以一定程度的防御重放攻击。但是,从客户端发出请求到服务器,始终有一定时间差,就算时间窗口设置很短,为1秒钟,攻击者也可能在该时间内并发很多请求,这就造成了时间戳验证方式无法完全防御重放攻击,给API的安全带来挑战。

[0005] 因此,如何确保API交互过程的安全性,有效防御重放攻击,保护企业和用户的资产免受损失,是当前急需解决的技术问题。

发明内容

[0006] 本发明为了解决上述背景技术中的技术问题,提供一种API安全实现方法、系统、介质及API框架实现方法。

[0007] 本发明解决上述技术问题的技术方案如下:

[0008] 第一个方面,提供了一种API安全实现方法,用于服务器端,所述方法包括步骤:

[0009] 验证配置及发送步骤,生成一次性的初始CSRF-Token,设置交互请求的时间窗口,再将初始CSRF-Token发送给客户端;

[0010] 提取步骤,接收客户端交互请求,从客户端交互请求中提取时间戳和待验证CSRF-Token;

[0011] 验证步骤,验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内;若两者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若两者不同时满足,则启动防御;

[0012] 重新发送CSRF-Token步骤,生成一次性的二次CSRF-Token,将二次CSRF-Token随

响应发送给客户端。

[0013] 第二个方面,提供了一种API安全实现方法,用于客户端,所述方法包括步骤:

[0014] 接收初始CSRF-Token步骤,接收并保存服务器端的初始CSRF-Token;

[0015] 发送交互请求步骤,向服务器端发送交互请求,交互请求携带有初始CSRF-Token和时间戳;

[0016] 接收响应及二次CSRF-Token步骤,接收服务器端随响应的二次CSRF-Token,保存二次CSRF-Token;其中所述服务器端用于生成一次性的初始CSRF-Token、一次性的二次CSRF-Token和设置交互请求的时间窗口;所述服务器端还用于验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内;若两者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若两者不同时满足,则启动防御。

[0017] 第三个方面,提供了一种API安全实现系统,用于服务器端,所述系统包括:

[0018] 验证配置及发送模块,用于生成一次性的初始CSRF-Token,设置交互请求的时间窗口,再将初始CSRF-Token发送给客户端;

[0019] 提取模块,用于接收客户端交互请求,从客户端交互请求中提取时间戳和待验证CSRF-Token;

[0020] 验证模块,用于验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内;若两者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若两者不同时满足,则启动防御;

[0021] 重新发送CSRF-Token模块,用于生成一次性的二次CSRF-Token,将响应和二次CSRF-Token发送给客户端。

[0022] 第四个方面,一种计算机可读存储介质,其上存储有计算机程序,该程序被处理器执行时实现上述的API安全实现方法。

[0023] 第五个方面,提供了一种API框架实现方法,用于服务器端,利用上述的API安全实现方法进行API框架的API调用防护,还包括步骤;

[0024] API框架创建步骤,在服务器创建具有统一API入口和路由池的API框架,API框架内设有若干相互隔离的主版本目录,每个主版本目录内设有若干相互隔离的次版本目录;路由池保存有请求匹配路径与指定版本目录之间的映射关系规则;

[0025] 版本调用请求接收步骤,API入口接收到客户端版本调用请求,版本调用请求包括URL路径和指定的HTTP头,或者URL路径和查询参数;版本调用请求还包括时间戳和待验证CSRF-Token;

[0026] 解析路由步骤,API框架解析版本调用请求,再按照路由池映射关系规则将版本调用请求路由到目标版本目录;

[0027] 响应步骤,根据路由结果,API框架调用目标版本目录处理客户端版本调用请求,再形成响应发回客户端。

[0028] 本发明实施例提供的一种API安全实现方法、系统、介质及API框架实现方法,基于一次性的CSRF-Token和时间戳验证用户交互请求,相比于现有技术中仅用时间戳进行验证,可以有效地防御重放攻击,且配置方便,不增加额外开销,交互及验证过程几乎无感。

附图说明

[0029] 为了更清楚地说明本发明实施例中的技术方案,下面将对实施例描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0030] 图1为本发明实施例提供的用于服务器端中使用双重验证的API安全实现方法流程示意图。

[0031] 图2为本发明实施例提供的用于服务器端中使用三重验证的API安全实现方法流程示意图。

[0032] 图3为本发明实施例提供的用于客户端中API安全实现方法流程示意图。

[0033] 图4为本发明实施例提供的用于服务器端中使用三重验证API安全实现系统结构示意图。

[0034] 图5为本发明实施例提供的用于服务器端中API框架实现方法流程示意图。

[0035] 图6为本发明实施例提供的API框架中主版本目录和次版本目录结构示意图。

[0036] 附图中,各标号所代表的部件列表如下:

[0037] 400、安全字典设置模块;410、验证配置及发送模块;420、提取模块;430、验证模块;440、重新发送CSRF-Token模块。

具体实施方式

[0038] 为了使本发明的目的、技术方案及优点更加清楚明白,以下结合附图及实施例,对本发明进行进一步详细说明。应当理解,此处描述的具体实施例仅仅用以解释本发明,并不用于限定本发明。

[0039] 重放攻击是一种通过截获并重新发送通信数据以获取未授权访问的网络攻击手段。在API交互中,传统的时间戳验证方法虽然能在一定程度上防御此类攻击,但由于客户端与服务器间存在时间差,这种方法并不能完全阻止攻击者在极短时间内发送大量并发请求。因此,开发更有效的技术以确保API交互的安全性,防止重放攻击,对保护企业和用户资产至关重要。

[0040] 针对上述问题,本发明实施例提供一种API安全实现方法,用于服务器端。图1为本发明实施例提供的用于服务器端中API安全实现方法流程示意图,如图1所示,本方法包括:

[0041] 步骤S101,生成一次性的初始CSRF-Token,设置交互请求的时间窗口,再将初始CSRF-Token发送给客户端。

[0042] 跨站请求伪造攻击(Cross-Site Request Forgery,通常缩写为CSRF)是一种网络攻击方式,它利用了Web应用程序对用户浏览器的信任。攻击者通过欺骗用户点击一个链接,或在用户不知情的情况下通过一些技术手段如JavaScript,让浏览器向一个受信任的网站发送非预期的请求。

[0043] CSRF-Token是一种专门防御CSRF的技术手段。CSRF-Token主要目的是在用户已经通过其他方式(如使用用户名和密码)登录后,保护用户在应用程序中的会话不被恶意利用。

[0044] 可以理解的是,此处提到的一次性的初始CSRF-Token,是指该CSRF-Token在服务

器里生成是唯一的,也只给客户端发送一次。因此,可以确保客户端获得的是唯一的CSRF-Token。

[0045] 可以理解的是,CSRF-Token的生成方式多样,可以采用随机值生成,使用随机数生成器创建一个不可预测的Token值;可以基于会话生成,Token的生成与用户的会话(Session)相关联,可能包含会话ID或其他会话特定的信息;还可以使用区块链技术生成,确保Token的唯一性和不可篡改性。因此,本发明实施例对此不作具体限定。

[0046] 步骤S102,接收客户端交互请求,从客户端交互请求中提取时间戳和待验证CSRF-Token。

[0047] 步骤S103,验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内;若两者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若两者不同时满足,则启动防御。

[0048] 值得说明的是,对于具体的验证过程,可以先进行CSRF-Token验证,再进行时间戳验证;也可以同时进行CSRF-Token验证和时间戳验证;还可以先进行时间戳验证,再进行CSRF-Token验证。因此,本发明实施例对此不作具体限定。不过,Token是唯一的,时间戳是允许一定的误差范围内允许访问的,作为优选,先验证CSRF-Token的正确性,再验证时间戳,更适应程序逻辑。

[0049] 此外,如果检测到CSRF攻击或时间戳异常,服务器可以立即采取防御措施,具体的防御措施包括但不限于拒绝请求、限制用户会话或提醒用户,及其它们的结合。因此,本发明实施例对此不作具体限定。

[0050] 我们还可以增加一些监控分析措施,如服务器可以利用日志监控和记录所有接收到的请求及其验证结果,便于事后审计和监控潜在的异常行为。

[0051] 步骤S104,生成一次性的二次CSRF-Token,将二次CSRF-Token随响应发送给客户端。

[0052] 现有技术中,客户端在发送请求时附加一个精确的时间戳,服务器接收请求后会立即比对时间戳与服务器当前时间,通过设置极短的时间窗口(如1秒内)来验证请求的时效性,以此机制抵御重放攻击。一方面,由于客户端到服务器的网络传输存在固有延迟,攻击者可能在时间窗口期间发送大量并发请求,这使得仅依赖时间戳的验证方法面临较大挑战;另一方面,在交互场景中,用户往往会进行频繁的操作,实时性强,设计的防御措施,需要平衡安全性与用户体验,避免因安全措施而影响用户的正常交互。

[0053] 而本发明的实施例中,针对交互场景,一次性CSRF-Token为每个请求提供了一个额外的安全层,即使攻击者能够截获请求,没有有效的Token也无法重复使用该请求;时间戳与服务器当前时间的比较,确保了请求在合理的时间窗口内发出,减少了请求被延迟或重放的风险,采用这种CSRF-Token和时间戳双重验证方式可以有效防御重放攻击。另外,实施这种验证方法简化了配置过程,能够无缝地提升Web框架和API平台的安全性;由于CSRF-Token和时间戳的生成、验证过程都非常快速,对系统性能的影响微乎其微,用户在进行交互时,不会感受到任何延迟或中断,验证过程对他们来说是透明的,从而保证了流畅的交互体验,可以在不牺牲用户体验的前提下,增强了API的安全性,有效平衡了安全措施、应用性能以及用户体验。

[0054] 基于上述实施例,该方法中,所述步骤S101,还包括:所述初始CSRF-Token是以隐

蔽的方式发送给客户端,发送的通道为加密通道;

[0055] 步骤S102,还包括:所述二次CSRF-Token是以隐蔽的方式发送给客户端,发送的通道为加密通道。

[0056] 值得说明的是,本实施例中提到的隐蔽的发送方式是多种多样的,包括但不限于:在HTML表单中使用隐藏的<input>字段来包含CSRF-Token,这样当用户提交表单时,Token会随表单数据一起发送;将CSRF-Token作为URL的查询参数附加在请求的URL上;通过HTTPOnly的Cookie来存储CSRF-Token,这样Token不会被客户端脚本访问,隐蔽性较好;使用自定义的HTTP请求头来发送Token,这需要客户端和服务器端的配合,并且不会暴露给最终用户。因此,本发明实施例对此不作具体限定。

[0057] 值得说明的是,本实施例中提到的加密通道是多种多样的,包括但不限于:使用HTTP Secure(HTTPS)协议来加密客户端和服务器之间的所有通信,确保数据在传输过程中的安全性;通过SSL(Secure Sockets Layer)或其继任者TLS(Transport Layer Security)协议来实现数据传输的加密;在企业环境中,客户端通过VPN(虚拟私人网络)连接到内部网络,然后再与服务器通信,VPN提供了加密的通信隧道;使用加密的API网关或服务,这些服务在将请求转发到后端服务之前,对请求进行加密处理。因此,本发明实施例对此不作具体限定。

[0058] 本实施例提供的方法,通过这些隐蔽的发送方式和加密通道,CSRF-Token的安全性得到了进一步加强,减少了Token在传输过程中被截获的风险,从而进一步防止了重放攻击。

[0059] 基于上述实施例,如图2所示,本方法包括:

[0060] 步骤S201,为每个API端点设置安全字典,安全字典规定服务器仅接收offset和limit两个参数,且每个参数的值为1到100之间的正整数;

[0061] 步骤S202,生成一次性的初始CSRF-Token,设置交互请求的时间窗口,将初始CSRF-Token与指定API端点的请求关联,再将初始CSRF-Token发送给客户端;

[0062] 步骤S203,接收客户端交互请求,从客户端交互请求中提取时间戳、CSRF-Token、API入参和API入参值;

[0063] 步骤S204,验证交互请求携带的CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内,API入参和API入参值是否符合安全字典的规定;若三者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若三者不同时满足,则启动防御;

[0064] 步骤S205,生成一次性的二次CSRF-Token,将二次CSRF-Token随响应发送给客户端。

[0065] 值得说明的是,在步骤S201中,offset是指偏移量,表示从查询结果的开始位置跳过多少条记录;limit是限制数量,表示计划从查询结果中获取的记录数量。当确定每个API入口应该传的类型和值后,可以避免非法字符,进一步保证了API的安全;同时,在API入口就可以拦截,避免在每个业务逻辑中单独进行判断,影响系统开销。

[0066] 还值得说明的是,在步骤S202中,将初始CSRF-Token与指定API端点的请求关联是指,服务器生成的CSRF-Token与用户会话相关联,而不是与特定的API端点永久绑定。当用户与应用程序进行交互,特别是通过表单或请求与API端点进行交互时,CSRF-Token用来确

保这些请求是从用户的浏览器发出的,而不是来自恶意的第三方网站。CSRF-Token的目的是为用户在会话期间发出的所有请求提供一层额外的安全性。

[0067] 本实施例提供的方法,使用安全字典来验证API请求参数,它确保了只有符合预定规则的参数被接受和处理,结合CSRF-Token和时间戳可以有效提升整体的安全性;同时,验证参数和值对用户是透明的,不会直接影响用户的操作流程或感知到的响应时间,这种方法对于安全性的提升是较大的,可对系统开销和用户体验的影响是微小的,较好的实现了安全性与性能的平衡。

[0068] 如图3所示,本发明实施例提供一种API安全实现方法,用于客户端,所述方法包括步骤:

[0069] 步骤S301,接收并保存服务器端的初始CSRF-Token;

[0070] 步骤S302,向服务器端发送交互请求,交互请求携带有初始CSRF-Token和时间戳;

[0071] 步骤S303,接收服务器端随响应的二次CSRF-Token,保存二次CSRF-Token;其中所述服务器端用于生成一次性的初始CSRF-Token、一次性的二次CSRF-Token和设置交互请求的时间窗口;所述服务器端还用于验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内;若两者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若两者不同时满足,则启动防御。

[0072] 本实施例提供的方法,在整个过程中,客户端依赖服务器对携带的CSRF-Token和时间戳进行严格验证,确保了只有符合要求的请求才能被服务器接受,从而有效防御重放攻击,同时为用户提供了无缝且安全的操作体验。

[0073] 如图4所示,本发明实施例提供一种API安全实现系统,用于服务器端,所述系统包括:

[0074] 验证配置及发送模块410,用于生成一次性的初始CSRF-Token,设置交互请求的时间窗口,再将初始CSRF-Token发送给客户端;

[0075] 提取模块420,用于接收客户端交互请求,从客户端交互请求中提取时间戳和待验证CSRF-Token;

[0076] 验证模块430,用于验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内;若两者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若两者不同时满足,则启动防御;

[0077] 重新发送CSRF-Token模块440,用于生成一次性的二次CSRF-Token,将响应和二次CSRF-Token发送给客户端。

[0078] 作为优选,如图4所示,在所述验证配置及发送模块410之前,还包括:

[0079] 安全字典设置模块400,用于为每个API端点设置安全字典,安全字典规定服务器仅接收offset和limit两个参数,且每个参数的值为1到100之间的正整数;

[0080] 所述验证配置及发送模块410,还用于将初始CSRF-Token与指定API端点的请求关联;

[0081] 所述提取模块420,还用于接收客户端交互请求,从客户端交互请求中提取时间戳、CSRF-Token、API入参和API入参值;

[0082] 所述验证模块430,还用于验证交互请求携带的CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内,API入参和API入参值是否

符合安全字典的规定;若三者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若三者不同时满足,则启动防御。

[0083] 本实施例提供的系统,通过这种基于一次性CSRF-Token和时间戳的验证形式,能够以一种几乎无感的方式,为用户提供了一个更加安全和可靠的交互环境。

[0084] 在本实施例中,提供一种计算机可读存储介质,其上存储有计算机程序,该程序被处理器执行时实现上述实施例所述API安全实现方法。

[0085] 如图5所示,本发明实施例提供一种API框架实现方法,用于服务器端,利用上述实施例的API安全实现方法进行API框架的API调用防护 ,包括步骤:

[0086] 步骤S501,生成一次性的初始CSRF-Token,设置交互请求的时间窗口,再将初始CSRF-Token发送给客户端;

[0087] 步骤S502,在服务器创建具有统一API入口和路由池的API框架,API框架内设有若干相互隔离的主版本目录,每个主版本目录内设有若干相互隔离的次版本目录;路由池保存有请求匹配路径与指定版本目录之间的映射关系规则;

[0088] 步骤S503,API入口接收到客户端版本调用请求,版本调用请求包括URL路径和指定的HTTP头,或者URL路径和查询参数,版本调用请求还包括时间戳和待验证CSRF-Token;

[0089] 步骤S504,从客户端交互请求中提取时间戳和待验证CSRF-Token;

[0090] 步骤S505,验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致,交互请求的时间戳是否在时间窗口内;若两者同时满足,则允许访问,并销毁服务器内保存的初始CSRF-Token;若两者不同时满足,则启动防御;

[0091] 步骤S506,API框架解析版本调用请求,再按照路由池映射关系规则将版本调用请求路由到目标版本目录;

[0092] 步骤S507,根据路由结果,API框架调用目标版本目录处理客户端版本调用请求,再形成响应和一次性的二次CSRF-Token发回客户端。

[0093] 而本发明的实施例中,所搭建的API框架本身具有高度模块化、灵活路由机制和稳定安全的API入口等特点,在此基础上配合特定的版本调用请求,即选择URL路径搭配指定的HTTP头或查询参数,一方面通过HTTP头或查询参数传递版本信息可以在不更改URL的情况下,动态地指定和切换所需版本,另一方面也提供了额外的扩展性,允许客户端根据需要请求特定版本的特定功能。因此,既可以满足开发商的版本迭代与维护需求,又满足客户通过API灵活调取所需版本代码需求。

[0094] 基于上述实施例,该方法中,所述步骤S502,包括:

[0095] 步骤S5021,在服务器端创建API框架,所述API框架具有统一的API入口,用于接收和响应外部请求。

[0096] 步骤S5022,在API框架内创建若干相互隔离的主版本目录,每个主版本目录内创建通用配置信息目录和次版本目录,通用配置信息目录与次版本目录之间相互隔离,次版本目录内设有第一控制器。

[0097] 值得说明的是,在每个主版本目录内,创建一个通用配置信息目录。这个目录用于存放所有次版本以及修订版本共享的配置信息,如数据库连接信息、API密钥、日志配置等,这种集中管理便于维护和更新。同时,确保通用配置信息目录与次版本目录之间相互隔离,可以有效避免配置信息的冲突和误用。

[0098] 此外,在每个次版本目录内,需要设置控制器。控制器是API框架中的核心组件,负责处理客户端请求并返回响应。控制器包含业务逻辑和数据访问逻辑。本实施例中,版本调用请求的路径是解析到控制器,控制器可以接收请求和输出响应。

[0099] 步骤S5023,在API入口路径中设置版本控制,用于支持操作不同版本的API。

[0100] 可以理解的是,此处版本控制并不是指整体的方法,此处版本控制是在API的URL中实现版本控制的一种技术手段,更具体的,它可以明确区分API的URL路径中不同版本的API,以便于管理和调用。

[0101] 对于用户来说,在API入口路径中设置版本控制可以通过URL路径明确地选择他们想要使用的API版本;对开发商来说,在API入口路径中设置版本控制可以更容易地管理和部署不同版本的API。

[0102] 步骤S5024,在API框架内创建路由池,在路由池内保存若干请求匹配路径,将请求匹配路径与指定版本目录之间一一对应,形成映射关系规则。

[0103] 本实施例提供的方法,利用版本目录的隔离确保了配置信息的集中管理和安全性,统一的API入口和版本控制机制为整个框架提供了清晰的结构和灵活性,而控制器的设置和路由池的创建则确保了请求能够被高效且准确地处理,进而实现了一个更加高效、更安全、可扩展且易于维护的API多版本控制框架。

[0104] 基于上述实施例,该方法中,所述步骤S5022,还包括:

[0105] 步骤S50221,所述通用配置信息目录内创建有配置文件目录、说明目录和语言目录;

[0106] 步骤S50222,所述次版本目录内创建有控制器目录和业务逻辑扩展目录,所述业务逻辑扩展目录内创建有修订版本目录,修订版本目录存放有第二控制器。

[0107] 为了便于理解,如图6所示,我们展示了API框架中主版本目录(V1)和次版本目录(V1.0)结构,同时我们对主版本目录和次版本目录中子目录进行解释:

[0108] (1)配置文件目录(config):在通用配置信息目录中创建配置文件目录,允许集中存储所有API版本共用的配置文件,如数据库配置、服务器端点等;

[0109] (2)说明目录(description):包含API使用的文档说明,可能包括API的使用方法、参数说明、版本更新日志等,有助于用户和开发人员理解API的工作原理和使用方式;

[0110] (3)语言目录(lang):用于存放多语言支持的资源文件,使得API能够根据请求的语言偏好返回相应语言的响应;

[0111] (4)控制器目录(controller):在次版本目录中创建控制器目录,用于存放处理特定业务逻辑的控制器代码;

[0112] (5)业务逻辑扩展目录(logic):提供了一个扩展点,允许开发者根据需要添加额外的业务逻辑处理模块。

[0113] 此外,我们是在业务逻辑扩展目录内创建修订版本目录,用于存放第二控制器,提供修订版本。另外,图6中,app是指应用程序(Application),代表着版本管理这部分内容,也即根目录;alarm通常指的是警报或通知系统,可以将配置信息目录和次版本目录设此目录下,用于在特定提醒。

[0114] 此外,也可以根据具体的使用场景,增加一些功能目录,比如在配置文件目录中增加日志目录、环境目录等;在次版本目录内增加服务目录(service)、验证目录(validate)

等。本发明实施例对此不作具体限定。

[0115] 不同于之前的实施例方法,本实施例允许开发者在不影响主业务逻辑的情况下,对特定功能进行扩展和修订,进一步提高了API的灵活性和可维护性,同时通过在业务逻辑扩展目录中创建第二控制器,实现了控制器的模块化,使得不同的业务逻辑可以由不同的控制器处理,有助于降低代码的耦合度。另外,清晰的目录也使得管理更加条理化和专业化。

[0116] 本发明实施例的计算机存储介质,可以采用一个或多个计算机可读的介质的任意组合。计算机可读介质可以是计算机可读信号介质或者计算机可读存储介质。计算机可读存储介质例如可以是——但不限于——电、磁、光、电磁、红外线、或半导体的系统、装置或器件,或者任意以上的组合。计算机可读存储介质的更具体的例子(非穷举的列表)包括:具有一个或多个导线的电连接、便携式计算机磁盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、光纤、便携式紧凑磁盘只读存储器(CD-ROM)、光存储器件、磁存储器件、或者上述的任意合适的组合。在本文件中,计算机可读存储介质可以是任何包含或存储程序的有形介质,该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。

[0117] 计算机可读的信号介质可以包括在基带中或者作为载波一部分传播的数据信号,其中承载了计算机可读的程序代码。这种传播的数据信号可以采用多种形式,包括但不限于电磁信号、光信号或上述的任意合适的组合。计算机可读的信号介质还可以是计算机可读存储介质以外的任何计算机可读介质,该计算机可读介质可以发送、传播或者传输用于由指令执行系统、装置或者器件使用或者与其结合使用的程序。

[0118] 计算机可读介质上包含的程序代码可以用任何适当的介质传输,包括——但不限于无线、电线、光缆、RF等等,或者上述的任意合适的组合。

[0119] 可以以一种或多种程序设计语言或其组合来编写用于执行本发明操作的计算机程序代码,所述程序设计语言包括面向对象的程序设计语言—诸如Java、Smalltalk、C++、Ruby、Go,还包括常规的过程式程序设计语言—诸如“C”语言或类似的程序设计语言。程序代码可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中,远程计算机可以通过任意种类的网络——包括局域网(LAN)或广域网(WAN)—连接到用户计算机,或者,可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。

[0120] 以上所述实施例仅表达了本发明的几种实施方式,其描述较为具体和详细,但并不能因此而理解为对本发明专利范围的限制。应当指出的是,对于本领域的普通技术人员来说,在不脱离本发明构思的前提下,还可以做出若干变形和改进,这些都属于本发明的保护范围。因此,本发明专利的保护范围应以所附权利要求为准。

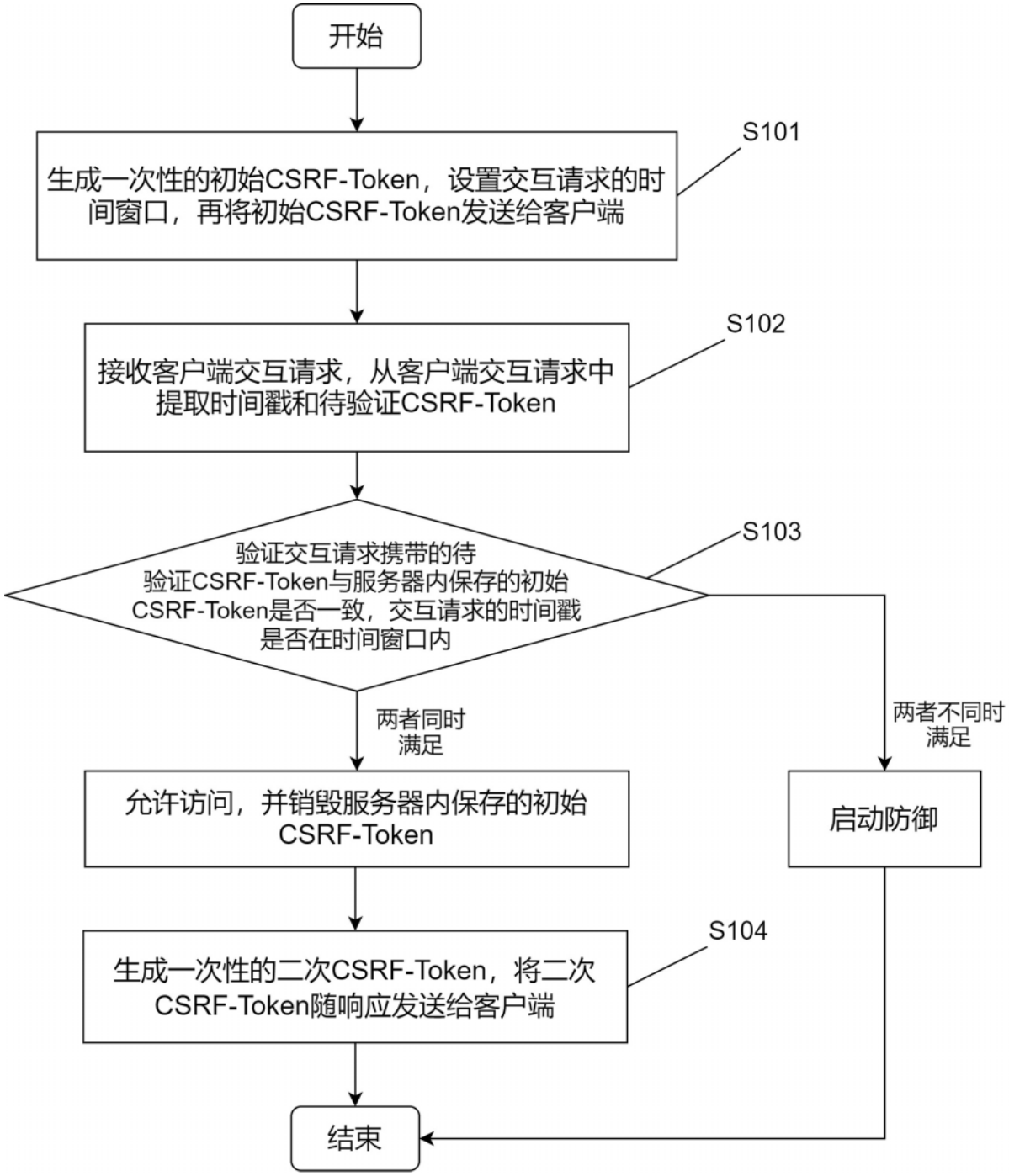

开始
生成一次性的初始CSRF-Token，设置交互请求的时间窗口，再将初始CSRF-Token发送给客户端
S101
接收客户端交互请求，从客户端交互请求中提取时间戳和待验证CSRF-Token
S102
验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致，交互请求的时间戳是否在时间窗口内
S103
两者同时满足
两者不同时满足
允许访问，并销毁服务器内保存的初始CSRF-Token
启动防御
生成一次性的二次CSRF-Token，将二次CSRF-Token随响应发送给客户端
S104
结束

图 1

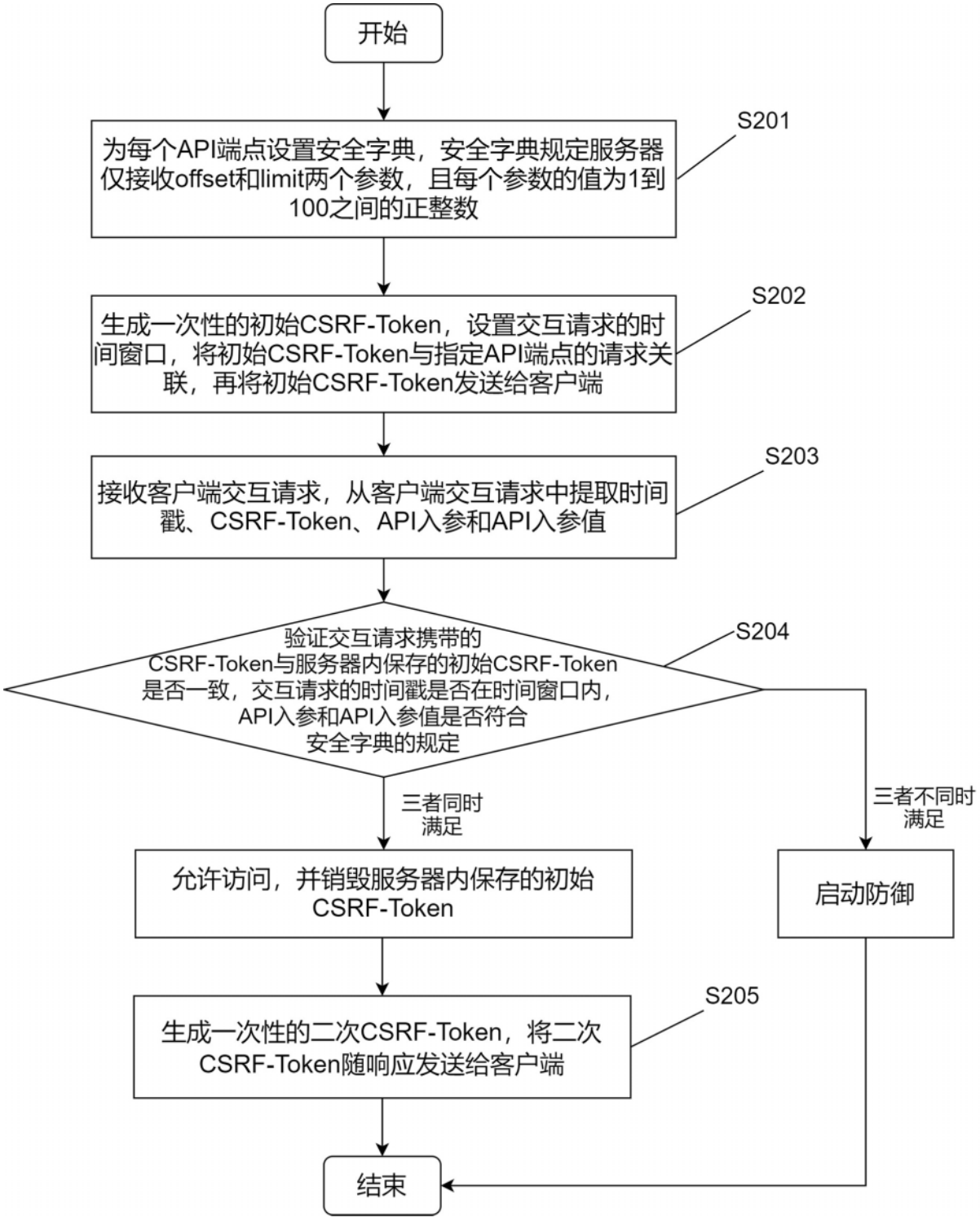

开始
为每个API端点设置安全字典，安全字典规定服务器仅接收offset和limit两个参数，且每个参数的值为1到100之间的正整数
S201
生成一次性的初始CSRF-Token，设置交互请求的时间窗口，将初始CSRF-Token与指定API端点的请求关联，再将初始CSRF-Token发送给客户端
S202
接收客户端交互请求，从客户端交互请求中提取时间戳、CSRF-Token、API入参和API入参值
S203
验证交互请求携带的CSRF-Token与服务器内保存的初始CSRF-Token是否一致，交互请求的时间戳是否在时间窗口内，API入参和API入参值是否符合安全字典的规定
S204
三者同时满足
三者不同时满足
允许访问，并销毁服务器内保存的初始CSRF-Token
启动防御
生成一次性的二次CSRF-Token，将二次CSRF-Token随响应发送给客户端
S205
结束

图 2

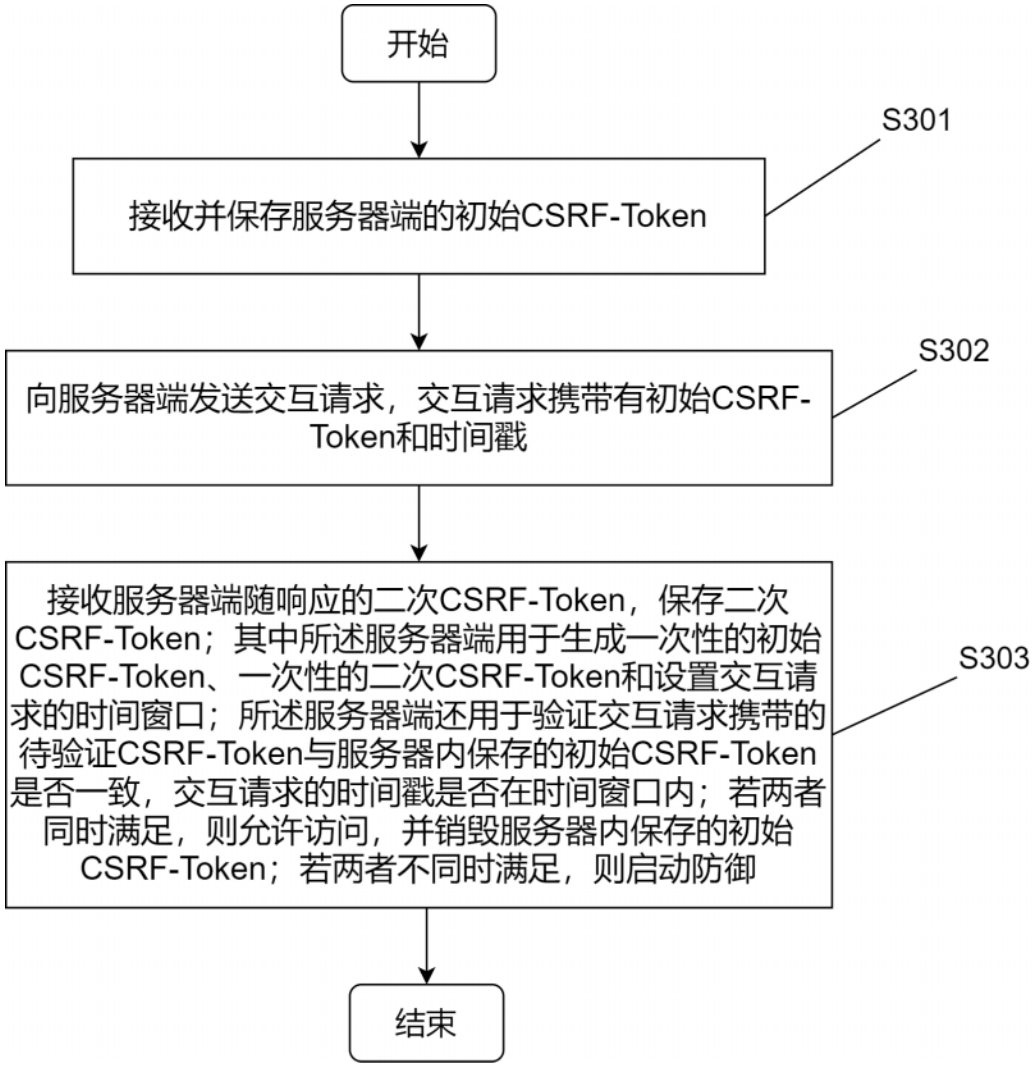

开始
S301
接收并保存服务器端的初始CSRF-Token
S302
向服务器端发送交互请求，交互请求携带有初始CSRF-Token和时间戳
S303
接收服务器端随响应的二次CSRF-Token，保存二次CSRF-Token；其中所述服务器端用于生成一次性的初始CSRF-Token、一次性的二次CSRF-Token和设置交互请求的时间窗口；所述服务器端还用于验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致，交互请求的时间戳是否在时间窗口内；若两者同时满足，则允许访问，并销毁服务器内保存的初始CSRF-Token；若两者不同时满足，则启动防御
结束

图 3

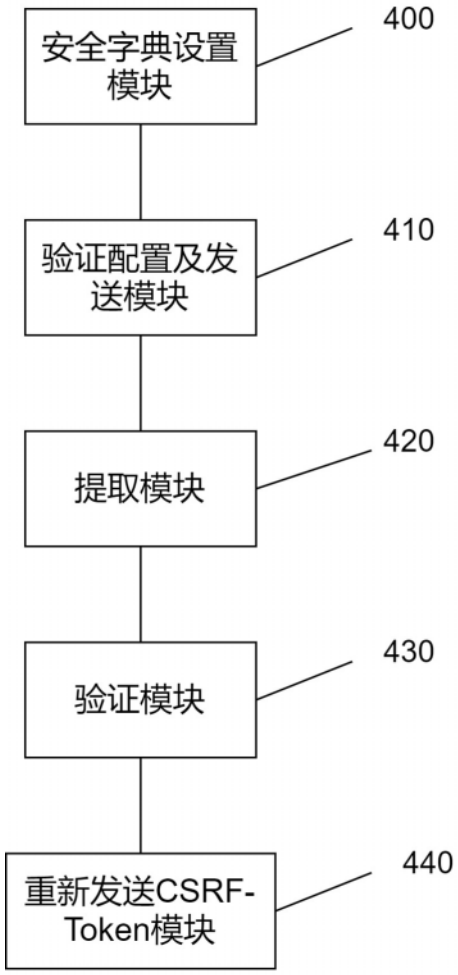

安全字典设置模块
400
验证配置及发送模块
410
提取模块
420
验证模块
430
重新发送CSRF-Token模块
440

图 4

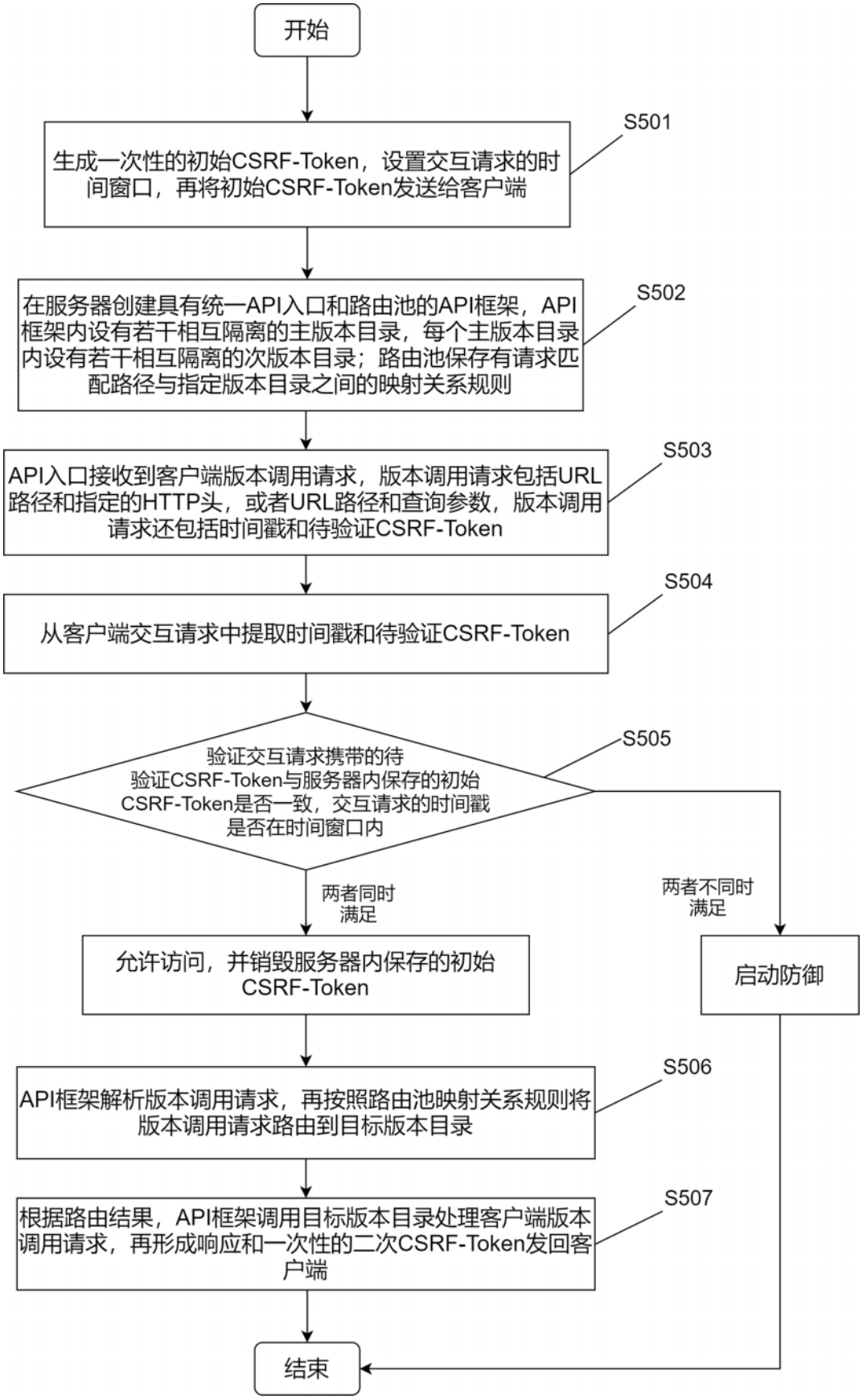

开始
S501
生成一次性的初始CSRF-Token，设置交互请求的时间窗口，再将初始CSRF-Token发送给客户端
S502
在服务器创建具有统一API入口和路由池的API框架，API框架内设有若干相互隔离的主版本目录，每个主版本目录内设有若干相互隔离的次版本目录；路由池保存有请求匹配路径与指定版本目录之间的映射关系规则
S503
API入口接收到客户端版本调用请求，版本调用请求包括URL路径和指定的HTTP头，或者URL路径和查询参数，版本调用请求还包括时间戳和待验证CSRF-Token
S504
从客户端交互请求中提取时间戳和待验证CSRF-Token
S505
验证交互请求携带的待验证CSRF-Token与服务器内保存的初始CSRF-Token是否一致，交互请求的时间戳是否在时间窗口内
两者同时满足
两者不同时满足
允许访问，并销毁服务器内保存的初始CSRF-Token
启动防御
S506
API框架解析版本调用请求，再按照路由池映射关系规则将版本调用请求路由到目标版本目录
S507
根据路由结果，API框架调用目标版本目录处理客户端版本调用请求，再形成响应和一次性的二次CSRF-Token发回客户端
结束

图 5

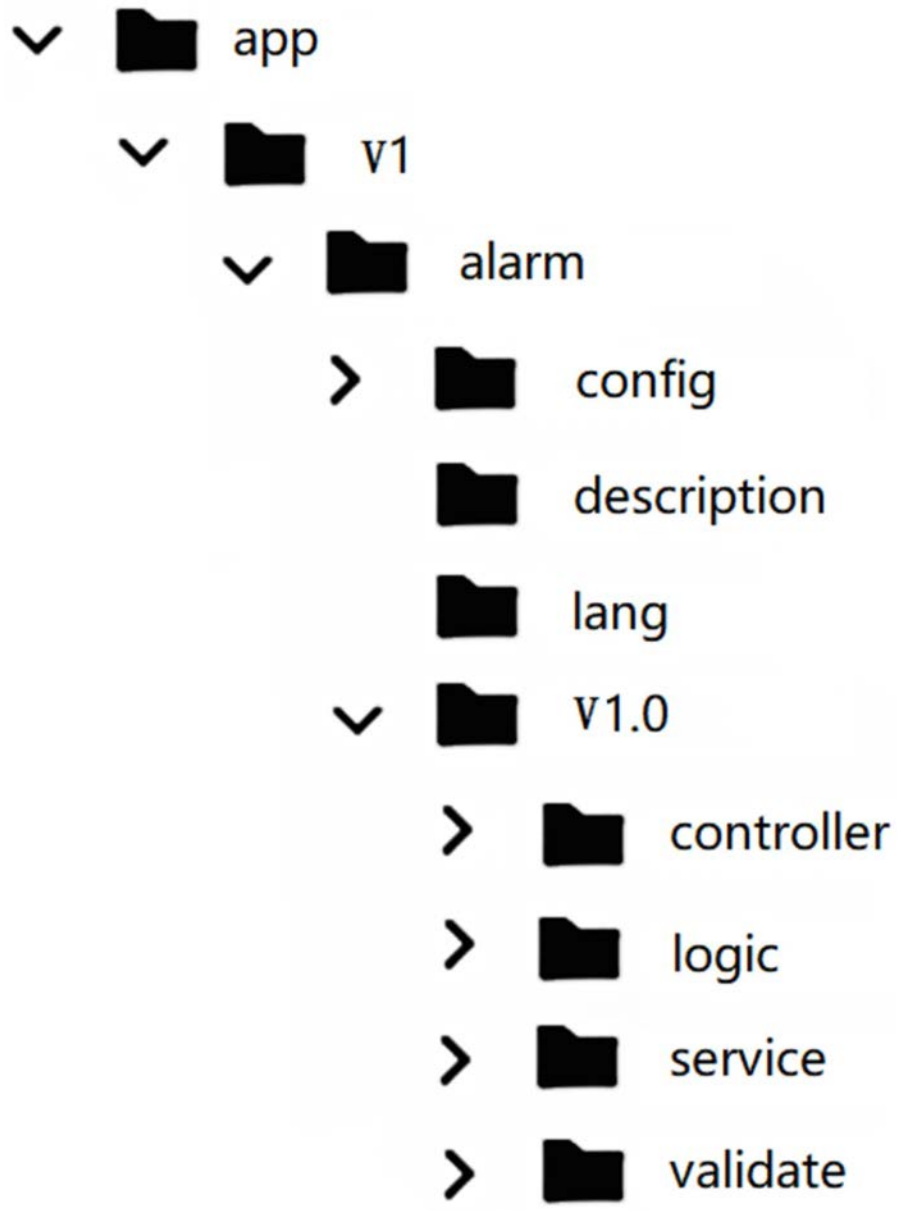

app
v1
alarm
config
description
lang
v1.0
controller
logic
service
validate

图 6