计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(4):347−353 [doi: 10.15888/j.cnki.csa.009011]

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

基于口令签名和 OAuth2.0 协议的强身份认证方案①

郝　恬, 左黎明, 陈艺琳

(华东交通大学 理学院, 南昌 330013)
通信作者: 左黎明, E-mail: limingzuo@126.com

摘　要: 为了解决网络应用身份认证问题, OAuth2.0 协议在实际生产环境中得到了非常广泛的应用. 但很多系统在设计时不合理使用 OAuth2.0 标准、产生很多安全漏洞. 分析了近年来关于 OAuth2.0 协议出现的安全问题, 包括中间人攻击, 授权劫持漏洞和 CSRF 漏洞, 针对这些安全问题提出了一种基于口令的 Schnorr 数字签名和 OAuth2.0 的强身份认证方案. 最后对该方案进行安全性分析, 结果表明该方案具有良好的安全性且易于使用.

关键词: OAuth2.0 协议; 数字签名; Schnorr 签名

引用格式: 郝恬,左黎明,陈艺琳.基于口令签名和 OAuth2.0 协议的强身份认证方案.计算机系统应用,2023,32(4):347–353. http://www.c-s-a.org.cn/1003-3254/9011.html

Strong Identity Authentication Scheme Based on Password Signature and OAuth2.0 Protocol

HAO Tian, ZUO Li-Ming, CHEN Yi-Lin

(School of Science, East China Jiaotong University, Nanchang 330013, China)

Abstract: In order to solve the identity authentication problem of network applications, the OAuth2.0 protocol has been widely used in the actual production environment. However, many systems use the OAuth2.0 standard unreasonably in their design, which results in many security flaws. This study analyzes the security problems of OAuth2.0 protocol in recent years, including the man-in-the-middle attack, authorization hijacking vulnerability, and CSRF vulnerability, and the study proposes a password-based Schnorr digital signature and OAuth2.0 strong identity authentication scheme for solving these security problems. Finally, the security of the scheme is analyzed. The results show that the scheme has excellent security and is easy to use.

Key words: OAuth2.0; digital signature; Schnorr signature

互联网已经深入到生活的各个方面, 企业所需的各种网络系统迅猛增加, 若是为每个系统单独设置登录策略, 不仅会给运营者维护系统增加困难, 而且用户也会产生账号口令过多记不住的困扰, 此时单点登录应运而生. 单点登录 (single sign on, SSO)[1] 实现了登录一次, 就可以访问其他相互信任的应用系统的功能.

目前, 使用最广泛且最健全的单点登录协议是 OAuth 协议[2]. OAuth 协议即开放授权协议, 为了解决无须共享密码的情况下, 从第三方应用程序安全地访问受保护数据资源的问题而提出, 广泛应用于第三方网络应用的用户身份验证和服务器授权. OAuth 协议发展至今共经历了 OAuth1.0、OAuth1.0a、OAuth2.0 三个版本, OAuth1.0a 针对 OAuth1.0 存在的会话固化攻击, 进行安全升级, 但 OAuth1.0a 实用性较差且技术复杂, 故逐渐被 Auth2.0 所取代. 2019 年朱博昌[3] 对国内基于 OAuth2.0 协议的授权登录应用现状的研究表

① 基金项目: 江西省教育厅科技项目 (GJJ200626, GJJ210625)
收稿时间: 2022-08-06; 修改时间: 2022-09-07, 2022-09-27; 采用时间: 2022-09-30; csa 在线出版时间: 2022-12-23
CNKI 网络首发时间: 2022-12-27

明, 在生活服务、休闲娱乐、新闻媒体、网络科技、学习教育等领域的系统应用大多都接入了开放平台, 其中开放平台的提供方大多是类似 QQ、微信和微博的社交类平台. 同年刘奇旭等人[4] 设计实现了一种面向 OAuth2.0 授权服务 API[5] 的账号劫持威胁检测框架 OScan, 并利用 OScan 对 Alexa 排名前 10 000 的网站中真实部署 的 3 853 个 OAuth2.0 授权服务 API 进行了大规模检测, 发现 360 个 OAuth2.0 API 脆弱性调用, 确认 101 个脆弱性调用可以最终导致账号劫持, 涉及 80 个知名网站和 10 个知名 OAuth2.0 服务提供商. 2021 年 Munonye 等人[6] 提出了一种基于机器学习的 OAuth 授权过程中潜在漏洞挖掘技术, 同年阿里云漏洞库[7] 批露了这一年由于 OAuth2.0 协议不恰当使用所导致的安全问题 (漏洞编号 CVE-2021-21291、CVE-2021-21411、CVE-2021-29437 、CVE-2021-41580).

OAuth2.0 协议的不正确使用导致了很多安全问题. 本文分析了目前基于 OAuth2.0 的第三方认证协议在应用中存在的安全问题, 包括中间人攻击、授权劫持漏洞、CSRF 漏洞, 并针对这些问题提出了对应的解决方案, 最终提出一种基于口令的 Schnorr 数字签名和 OAuth2.0 的强身份认证协议. 加入数字签名, 保障了该协议的消息完整性和来源可靠性, 同时加入时间戳作为新鲜因子, 可有效预防重放攻击, 此外, 通过对跳转地址及 state 参数进行校验以抵抗 OAuth2.0 协议普遍存在的授权劫持和 CSRF 攻击. 基于口令的 Schnorr 数字签名方案在客户端根据账户和口令生成密钥, 可避免证书存储所带来的复杂性, 综合考虑实际应用场景, 其在 APP 端具有很好的易用性.

1 第三方认证方案及其攻击

1.1 OAuth2.0 协议

OAuth2.0 (开放授权) 是一个开放标准, 允许用户授权第三方网站访问他们存储在另外的服务提供者上的信息, 而不需要将用户名和密码提供给第三方网站或分享他们数据的所有内容. 图 1 展示了 OAuth2.0 协议的抽象授权流程. 客户端先向资源所有者发送授权请求, 资源所有者若同意请求, 则向客户端发送授权许可, 客户端凭借资源所有者的授权许可, 向授权服务器请求访问令牌; 客户端通过访问令牌访问资源服务器的受保护资源信息.

OAuth2.0 针对不同的授权情况提出了 4 种授权模式: 授权码模式、隐式流模式、密码模式、客户端凭证模式. 本文将主要针对授权码模式进行分析和优化.

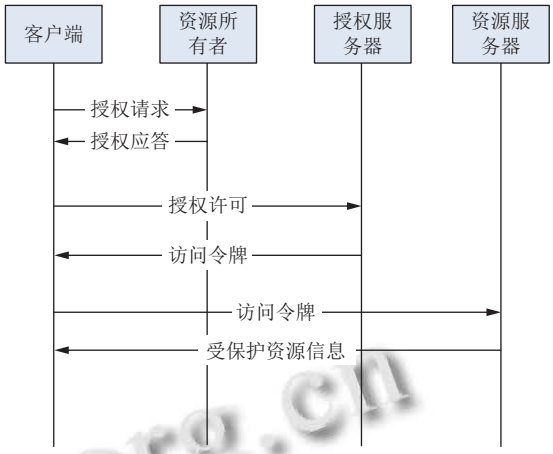

图 1 OAuth2.0 抽象授权流程图

1.2 第三方认证方案的攻击

OAuth2.0 的漏洞主要来自两个方面, 一是协议本身的缺陷, 二是开发人员不正确的使用协议构造方案.

1) 中间人攻击

OAuth2.0 协议在数据交互过程中使用 HTTPS 协议, 针对 HTTPS 协议, 最常见的中间人攻击[8] 就是 SSL 劫持攻击[9] 和 SSL 剥离攻击[10]. SSL 劫持攻击即 SSL 证书欺骗攻击, 攻击者将自己接入客户端和服务端之间, 将服务器端的证书和公钥均替换为自己的, 对于客户端来说, 在校验证书过程中会提示证书错误, 但由于很多用户安全意识不高, 会选择继续操作, 从而导致攻击者窃取了客户端和服务器端的传递数据.

SSL 剥离攻击又称 HTTP 降级攻击, 攻击者将自己接入客户端和服务器端之间, 在客户端向服务器端发出请求时阻碍 HTTP 向 HTTPS 的路由[11], 从而攻击者 (充当客户端) 和服务器端建立 HTTPS 链接且客户端和攻击者 (充当服务器端) 建立 HTTP 链接. 攻击者就实现了直接获取客户端提交给服务器端的明文数据, 并且可以直接修改从服务端收到的响应包返回给客户端.

2) 针对 OAuth2.0 的授权劫持漏洞

根据 OAuth2.0 的授权流程, 授权服务器会把授权码和访问令牌转发到 redirect_url 对应的地址, 如果攻击者伪造 redirect_url 的地址为自己构造的地址, 并诱导用户发送该请求, 之后获取到的授权凭证将会返回给攻击者伪造的回调地址. 攻击者使用返回的授权凭证登录用户账号, 从而导致授权劫持[12].

OAuth2.0 的授权劫持漏洞可以细分为 redirect_url 未校验和校验可绕过两种情况. 若 redirect_url 未校验, 攻击者可以将回调地址修改为任意指定的 URL, 并配

合跨站请求伪造 (cross-site request forgery, CSRF) 进行 token 欺骗. 如果 redirect_url 做了校验, 但是校验不严格, 也很容易被攻击者利用造成授权劫持. 比如 2020 年发生的安全漏洞 CVE-2020-7741, 由于跳转的 URL 没有进行正确的转义处理和校验, 导致被注入 JavaScript 脚本 redirect_url=javascript:alert(document.cookie), 可诱导用户泄露其 access_token.

3) 针对 OAuth2.0 的 CSRF 漏洞

CSRF[13], 跨站请求伪造, 是一种对网站的恶意攻击. 在用户不登出受信任网站并在本地生成 cookie 的情况下, 访问危险网站 B, 从而实现身份盗用.

在 OAuth2.0 授权码模式中, 由于客户端与认证服务器交互多次, 若交互过程中 code 参数没有和当前客户端的状态绑定, CSRF 漏洞就很容易在此发生, 在乌云网上找到相关案例: wooyun-2014-054785、wooyun-2014-054888 、wooyun-2014-055473. 针对 OAuth2.0 协议的 CSRF 攻击主要流程如下.

登录网站 A 时, 可以申请绑定网站 B 的账号, 在网站 A-网站 B 的 OAuth2.0 认证流程中申请授权码的请求报文如图 2, 图 3 为返回报文.

```
https://openapi.baidu.com/oauth/2.0/authorize
?response_type=code
&client_id=foRRWjPq8In3SIhmKQwlPep3
&redirect_url=http://www.renren.com/bind/baidu/baiduLoginCallBack
```

图 2 申请授权码的请求报文

```
http://www.renren.com/bind/baidu/baiduLoginCallBack
?code=f056147c661d0b9fbb6cd305567cb994
```

图 3 申请授权码的返回报文

当攻击者获取到自己的 code 之后, 再精心构造一个 Web 页面, 此页面可以触发客户端向认证服务器发起令牌请求的申请, 而这个请求中的 code 正是攻击者自己的 code 值, 攻击者将此 Web 页面发布在网络上, 诱骗已经登陆网站 A 的普通用户来点击, 从而在普通用户的浏览器上触发申请令牌的请求, 但此时获得的 Access_Token 及进一步获取的用户信息均是攻击者的, 此时就将攻击者的网站 B 的账号与普通用户的网站 A 的账号绑定, 当攻击者再次通过自己的百度账号登录网站 A 时, 进入的其实是普通用户的网站 A 账号.

由于 state 参数在 OAuth2.0 授权流程中为非必选项, 所以很多开发人员会忽略这个参数, 进而无法保证 code 参数与当前客户端的状态绑定, 导致 CSRF 漏洞.

2 基于口令签名和 OAuth2.0 协议的强身份认证方案

2.1 Schnorr 签名算法简介

Schnorr 签名算法[14] 最初是由德国数学家和密码学家 Schnorr 在 1990 年提出, 主要分为 3 个过程: 初始过程、数字签名的生成过程、数字签名的验证过程. Schnorr 签名算法的安全性基于离散对数[15]问题, 具有安全性高、验证签名快等优点.

(1) 初始过程

1) 随机选取两个大素数, 且满足$p \geqslant 2^{512}$, $q \geqslant 2^{160}$, $q|(p-1)$, 在整数域上选取g, 使得$g^q = 1 \bmod p$.

2) 用户 A 随机选取整数x $(1 < x < q)$, 并计算$y = g^x \bmod p$, 其中, x作为用户私钥, y作为用户的公钥对外公开.

(2) 数字签名的生成过程

对于待签名的消息m, 进行如下操作.

1) 选择随机数$1 < k < q$, 计算$r = g^k \bmod p$.

2) 利用标准哈希函数进行哈希计算得$e = H(r,m)$.

3) 计算$s = xe + k \bmod q$.

4) 将(e,s)作为生成的数字签名.

(3) 数字签名的验证过程

数字签名的接收方为了验证收到的消息m'和(e',s'), 进行如下操作.

1) 计算$r' = g^{s'} y^{-e'} \bmod p$.

2) 使用标准的哈希函数进行哈希计算得$H(r',m)$.

3) 若$H(r',m) = e$, 则验证通过, 否则验证失败.

2.2 基于口令的 Schnorr 签名方案密钥生成过程

基于口令的 Schnorr 签名方案 (PBS), 提出一种不同于传统签名方案的密钥生成方法, 图 4 展示了密钥生成过程.

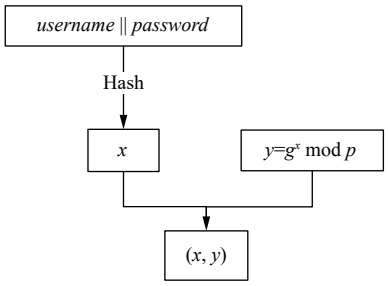

图 4 基于口令的 Schnorr 签名方案密钥生成过程

1) 先使用杂凑函数对字符串*username*||*password*进行压缩, 再进行数据类型转换得到整数 x.

2) 计算$y = g^x \bmod p$.

3) 密钥对是(x,y), 其中x是私钥, y是公钥.

2.3 强认证方案工作原理

网络应用的客户端[16] 一般分为 3 种: 基于桌面程序的客户端 (C/S 架构)、基于浏览器的客户端 (B/S 架构) 以及 APP 客户端程序 (APP/S 架构). 对于上述 3 种客户端架构, 我们均使用基于口令的 Schnorr 签名方案对用户身份进行认证.

图 5 展示了基于口令的 Schnorr 签名的第三方认证流程, 在每次客户端提交数据时进行数字签名, 并在服务器端进行验证. 数字签名并不会占用通信过程中很多资源, 但可有效保证消息完整性、不可抵赖性及来源可靠性. 对 redirect_url 参数进行过滤和校验, 首先检测 redirect_url 参数中是否有恶意字符和语句, 若有则拒绝访问. 限制只允许本网站内的链接进行跳转. 在请求 code 的过程中加入 state 参数, state 参数是客户端随机生成的一段字符串, 具有不可预测性, 且与当前会话具有关联性. 当客户端收到服务器端返回的 code 时, 先验证 state 参数是否与生成的 state 参数一致, 若不一致当作异常处理.

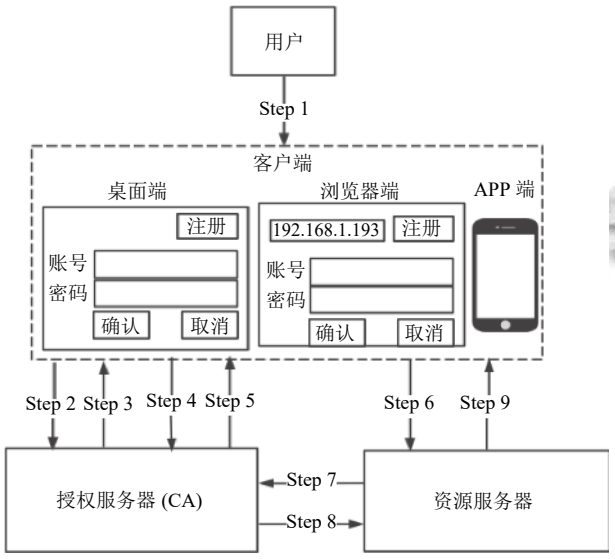

图 5 基于口令的 Schnorr 签名的第三方认证流程

Step 1. 用户访问客户端应用程序, 新用户可进行注册. 注册成功之后客户端会根据用户的账号和密码, 生成私钥. 客户端应用程序检测是否已获得授权许可, 若未授权则通过浏览器重定向到授权服务器, 启动身份验证流程.

Step 2. 用户携带参数向授权服务器请求授权码 code, 并使用私钥对参数拼接成的字符串进行签名, 将参数和签名均发送给授权服务器端.

Step 3. 授权服务器端验证签名, 若验证通过, 授权服务器会将授权码 code 和随机字符串 state 发送到客户端. 若验证失败, 授权服务器会返回错误码.

Step 4. 用户携带 code 向授权服务器请求授权码 Access_Token, 客户端使用自己的私钥对参数拼接起来的字符串进行签名, 将参数和签名均发送给授权服务器.

Step 5. 授权服务器验证签名, 若验证成功, 便向客户端发送 Access_Token, 若验证失败, 则返回错误码.

Step 6. 客户端应用程序携带访问令牌 Access_Token, 发送给资源服务器请求受保护资源.

Step 7. 资源服务器向授权服务器验证 Access_Token 的正确性和是时效性.

Step 8. 返回 Access_Token 验证结果.

Step 9. 资源服务器返回用户请求的受保护资源.

2.4 实验仿真

实验中使用 Win10 操作系统, 其支持 C#开发环境. 在 C/S 架构下, 客户端开发基于 C#的 WinForm, 并在客户端使用 C#代码实现数字签名; 在 B/S 架构中, 浏览器充当客户端, 使用 JavaScript 实现 Schnorr 算法和 SM3 算法并对请求数据进行数字签名; 在 APP/S 架构中, 客户端基于开源移动开发框架 Apache Cordova 进行开发, 同时使用 JavaScript 语言实现数字签名.

PBS 生成公私钥对代码如下.

```
public string keygenerator(string username, string password){
    SM2 sm2 = new SM2();
    string message = username + password;
    Hash hh = new Hash();
    string x = hh.SHA1Encrypt(message);
    BigInteger xx = new BigInteger(x);
    ECPoint y = sm2.ecc_point_g.Multiply(xx);
    return (x);
    BigInteger x1 = y.X.ToBigInteger();
    BigInteger y1 = y.Y.ToBigInteger();
    return (x1+“, ”+y1);
    }
```

图 6 展示了客户端向服务端请求 code 的数据交互过程, 客户端携带参数 response_type||client_id||scope||redirect_url||state||time||sig, 其中 response_type 是授权模式, 对于授权码模式其值是 code; client_id 指系统应

用程序分配的 id 值; scope 指请求的令牌范围; redirect_url 指授权通过后响应发送到的 URL; state 是随机字符串, time 是获取当前的时间. 服务器端签名校验通过, 返回 code 值, 并返回客户端产生的随机字符串 state, sig 是客户端使用私钥对其他参数组成的字符串签名得到的值.

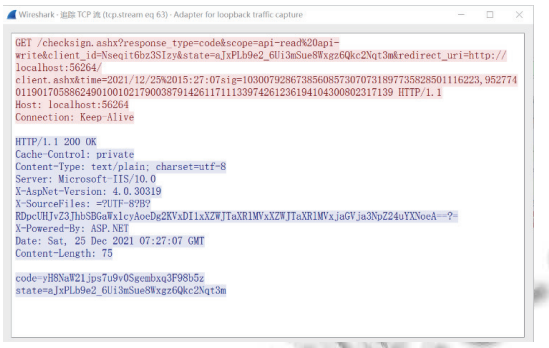

图 6 客户端请求 code 的数据交互流程

图 7 展示了使用 wireshark 抓包客户端向服务端请求 Access_Token 的数据交互过程, 客户端携带参数 grant_type||code|| redirect_uri||time||sig, 其中 grant_type 在这里默认为 authorizat_code. 服务端验证签名通过后, 会返回访问令牌 Access_Token、失效时间 expires_in 及访问范围 scope.

3 安全性分析

3.1 来源可靠性

当用户在客户端向服务器端发送请求时, 会用自己的私钥对数据进行签名, 服务器端用存储的公钥进行验证签名, 可以确保信息数据来自某位用户, 保证了消息来源的权威性.

当客户端-服务器端完成了生成签名、验证签名, 双方都不能否认自己做过此操作, 且授权服务器每次返回客户端所需要的数据时, 都会写入日志文件, 保证了双方通信当中的不可抵赖性[17]. 故该协议可保证消息来源的权威性和不可抵赖性, 进而保证了来源的可靠性.

3.2 消息完整性

客户端对要传递的数据进行签名, 交互的验证方在收到签名之后验证签名, 若攻击者对截获传递中的数据包并对其进行篡改, 签名也会随即变化, 无法通过验证, 因此该协议可防止数据被篡改, 保证消息的完整性. 图 8 使用 burpsuite 拦截客户端向服务器端发送的 code 请求包, 并将 client_id 修改为另一个用户的 client_id, 返回 error 并提示签名验证失败.

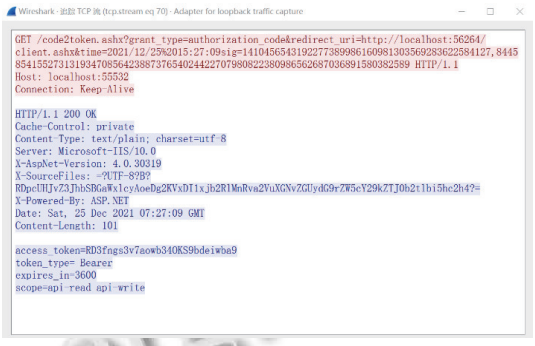

图 7 客户端请求 Access_Token 的数据交互流程

图 8 消息完整性测试包

3.3 抗重放攻击

在第三方认证协议中加入时间戳作为新鲜因子, 当验证签名时会先验证时间戳的新鲜性, 任何攻击者通过截获之前的数据包进行重放, 都不能通过验证, 有效防止了重放攻击. 图 9 使用 burpsuite 将事先拦截好的数据包进行重放, 由于此时时间戳已经失去新鲜性, 故不能验证通过.

3.4 可抵抗授权劫持漏洞

本文所提到的第三方认证协议对 redirect_uri 进行校验, 首先判断是否带有恶意字符和语句, 若有则拒绝

跳转; 进而通过增加网站跳转的判断条件, 限制仅允许本网站的链接进行跳转. 因此该协议可抵抗授权劫持漏洞. 图 10 使用 burpsuite 对向非本网站的地址进行跳转请求时进行抓包, 返回 error 并禁止用户非源跳转, 有效防止授权劫持漏洞.

3.5 可抵抗 CSRF 漏洞

在客户端向服务器端发送 code 请求包时, 加入 state 参数, 服务器端返回 code 时也会携带 state 参数, 以供客户端校验. 由于 state 参数具有不可预测性, 故不容易被猜解; state 参数与当前用户会话是相互关联的且 state 参数在使用之后会立即失效, 不会被攻击者二次利用. 图 11 使用 burpsuite 拦截客户端向服务器端发送的 code 请求包并将 state 参数修改为已失效的 state 参数, 返回 error 并提示 state 参数校验失败, 有效预防了 CSRF 攻击.

图 9 抗重放攻击测试包

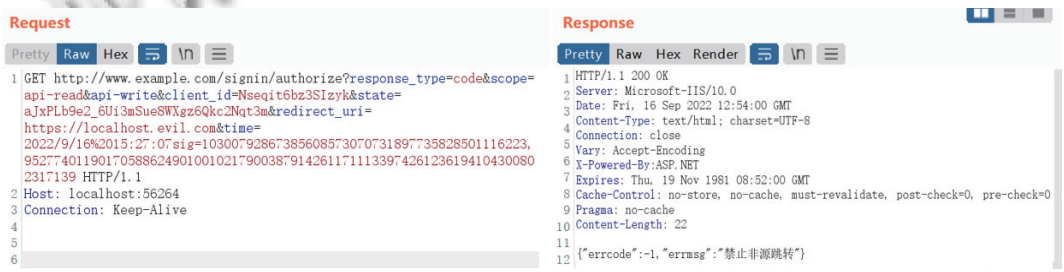

图 10 进行非源跳转时的交互包

图 11 失效 state 参数请求交互包

4 安全性比较

选取目前使用比较广泛的开放平台[18–21]提供方, 并通过这些开放平台提供的测试账号进行测试, 对其数据交互过程进行抓包, 分析其安全性. 表 1 针对近年来 OAuth2.0 协议在各个授权服务方应用现状进行比较, 若该授权服务方满足此安全选项, 则打勾, 没有则不填.

方案 1–4 分别为百度、GitHub、微信、微博的第三方认证协议, 方案 5 为基于口令的 Schnorr 数字签名的第三方认证方案. 5 个方案均对回调地址的传入内容进行校验, 并加入 state 参数, 故能抵抗授权劫持和 CSRF 攻击. 方案 1、方案 3 和方案 5 在认证过程中对数据均进行数字签名, 具有消息完整性、来源可靠性和抗重放攻击的特征[22].

表 1 各方案比较

方案	来源可靠性	消息完整性	抗重放攻击	抗授权劫持	抗CSRF
1	√	√	√	√	√
2				√	√
3	√	√	√	√	√
4				√	√
5	√	√	√	√	√

5 结论

针对目前第三方认证所存在的问题, 本文提出了一种基于口令的 Schnorr 签名方案的第三方认证方案. 通过对当前开放平台提供方进行安全性分析和比较, 基于口令的 Schnorr 数字签名的第三方认证协议具有消息完整性、来源可靠性、抗重放攻击、抗授权劫持等特点, 并实现该方案在 C/S、B/S、APP/S 三种架构下的安全控制访问. 接下来会将可证明安全理论作为研究方向, 继续深入.

参考文献

1 Hossain N, Hossain A, Hossain Z, *et al*. OAuth-SSO: A framework to secure the OAuth-based SSO service for packaged web applications. Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). New York: IEEE, 2018. 1575–1578. [doi: 10.1109/TrustCom/BigDataSE.2018.00227]

2 赵嘉熙, 邬卓恒. 系统科学视域下基于 OAuth2.0 的授权登录安全性提升方案研究分析. 网络安全技术与应用, 2022, (3): 14–16.

3 朱博昌. 基于 OAuth2.0 协议的授权登录国内应用现状研究. 现代信息科技, 2019, 3(20): 151–154.

4 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. 通信学报, 2019, 40(6): 40–50.

5 谈丽君. 基于 Robot Framework 的 API 接口自动化测试系统的设计与实现 [硕士学位论文]. 上海: 华东师范大学, 2022.

6 Munonye K, Péter M. Machine learning approach to vulnerability detection in OAuth2.0 authentication and authorization flow. International Journal of Information Security, 2022, 21(2): 223–237. [doi: 10.1007/s10207-021-00551-w]

7 阿里云漏洞库. https://avd.aliyun.com/. [2022-07-26].

8 康荣保, 张玲, 兰昆. SSL 中间人攻击分析与防范. 信息安全与通信保密, 2010, (3): 85–87, 90. [doi: 10.3969/j.issn.1009-8054.2010.03.035]

9 翟雪峰. SSL 的安全分析及被劫持的研究、实现 [硕士学位论文]. 成都: 四川大学, 2004.

10 知乎. 什么是 SSL 剥离攻击? https://zhuanlan.zhihu.com/p/405289913. (2021-08-31)[2021-12-26].

11 朱子豪. 基于动态信任评估的 MANET 安全路由协议研究 [硕士学位论文]. 南京: 南京信息工程大学, 2022. [doi: 10.27248/d.cnki.gnjqc.2022.001146]

12 毛剑, 韦韬, 陈昱, 等. 抗授权劫持攻击的安全电子交易方案. 武汉大学学报 (理学版), 2008, 54(5): 593–597. [doi: 10.14188/j.1671-8836.2008.05.027]

13 Barabanov AV, Markov AS, Tsirlov VL. Information security controls against cross-site request forgery attacks on software applications of automated systems. Journal of Physics: Conference Series, 2018, 1015(4): 042034.

14 Schnorr CP. Efficient identification and signatures for smart cards. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques. Houthalen: Springer, 1990. 688–689.

15 Huang Y, Su ZF, Zhang FG, *et al*. Quantum algorithm for solving hyperelliptic curve discrete logarithm problem. Quantum Information Processing, 2020, 19(2): 62. [doi: 10.1007/s11128-019-2562-5]

16 左黎明, 夏萍萍, 陈祚松. 基于国密 SM2 数字签名的网络摄像头保护技术. 信息网络安全, 2018, (5): 32–40. [doi: 10.3969/j.issn.1671-1122.2018.05.004]

17 左黎明, 胡凯雨, 张梦丽, 等. 一种具有双向安全性的基于身份的短签名方案. 信息网络安全, 2018, (7): 47–54. [doi: 10.3969/j.issn.1671-1122.2018.07.006]

18 百度开放平台. https://developer.baidu.com/. [2022-07-26].

19 GitHub 开放平台. https://github.com/settings/developers. [2022-07-26].

20 微信开放平台. https://open.weixin.qq.com/. [2022-07-26].

21 微博开放平台. https://open.weibo.com/development. [2022-07-26].

22 钱君生, 杨明, 韦巍. API 安全技术与实战. 北京: 机械工业出版社, 2021.

(校对责编: 孙君艳)