**OAuth 2.0：通往地狱之路**

译者：[HorseLuke](http://user.yeeyan.org/u/50978) 原作者：Eran Hammer

近日OAuth 2.0协议领导者之一Eran Hammer从OAuth工作组退出。他在博客中详细阐述了OAuth 2.0并没有达到两个最重要的目标-安全和互操作性，认为问题核心在于web和企业两个世界之间存在不可调和的冲突，同时直言将OAuth带入IETF是个巨大的错误

人们常说，通往地狱之路，常由善意铺就[2][3]。好吧，我说的就是[Oauth 2.0](http://tools.ietf.org/html/draft-ietf-oauth-v2" \t "_blank)。

上个月我做出了一个痛苦的决定，彻底和Oauth 2.0标准断绝关系。我辞去首席作者和校订者，**从规范中删掉自己的名字**，然后离开了工作组。从一份呕心沥血3年之久的文档和超过24份的草稿中去掉自己的名字是多么的不容易；而决定离开一个我领导5年之久为之努力的领域则是多么的痛心。

我做出这个极端决定，并不是单一问题或者事件可以解释的。这是一次由数千次刀割所引发的死亡。随着工作接近尾声，我越来越深思究竟我们在干什么。到了最后，我得出一个结论，Oauth 2.0是个烂协议，烂得和WS-\*有得一拼，烂得我不想再和它产生任何关联。这是我事业生涯中最大的一次职业挫败。

无论是邮件列表、会议讨论、特别设计委员会还是秘密渠道，无数次艰难的争论和妥协，最终产出的规范却在两个最重要目标上双缺失——安全性和互操作性。实际上，有一份折中方案将它从“协议”重新命名为“框架”，[而另一份则添加了一个免责声明，警告这份规范并不能阐述互操作性的实现。](http://tools.ietf.org/html/draft-ietf-oauth-v2-30#section-1.8)

和[Oauth 1.0](http://tools.ietf.org/html/rfc5849)相比，Oauth 2.0规范更复杂，缺乏互操作性，实用性打折扣，更加不完整，最重要的是，更加不安全。

说得更明确些，Oauth 2.0如果在一个对互联网安全有深入了解开发者手上，实现结果大抵是安全的。然而，就这两年经历的情况来看，Oauth 2.0在大部分开发者手中明显出现了不安全的实现结果。

**我们怎么到达这里的？**

这个问题的核心在于，**互联网（web）**和**企业（enterprise）**这两个世界之间有着强烈且难以弥合的冲突。IETF的OAuth工作组始于强烈的互联网驱动气势。然而一年之后随着工作一拖再拖，互联网工作者已从最初的1.0社区一个个离去，结果工作组剩下的大部分都是企业人员…...然后还有我。

互联网社区一直在寻找与1.0非常相似的协议，并且在某些缺乏的方面进行小改善：简化签名，增加轻量级识别层，标记native application，增加更多流程以适应新的client类型，还有增强安全性。而企业社区则在寻找一种可以最小化改动已有系统的框架，对某些人来讲，还试图寻求一种通过定制获利的新来源。举个例子理解这种分歧之深——在早期会议中，互联网工作者想实现一种在浏览器内client的流程优化，而企业人员则想实现基于安全断言标记语言（SAML）的流程。

由上述冲突产生的规范成为了一种**基于委员会设计**、且更多是为了服务企业方的东拼西凑妥协物。更确切来说，它并非用于满足企业所有直接的需求，它给予的是几乎无限制的可扩展性。正是这种可扩展性和必须要求的灵活性，把这些协议给毁了。**现在几乎啥东西都不用费什么努力，就可以宣称达到“Oauth 2.0兼容”标准了。**

**深入了解[4]**

要明白OAuth 2.0的问题，你需要了解它相比OAuth 1.0在核心架构上有哪些变化:

**无绑定token（Unbounded tokens）** - 在1.0中，client若要访问受保护资源（protected resource），则必须提供两组凭据：token凭据（the token credentials，即Access Token和Access Token Secret）和client凭据（the client credentials，即常说的应用APP KEY和APP SECRET）。在2.0中，client凭据已经不再被使用了，意味着token已经不再绑定于任何一个特定的client类型或者实例。这样的结果，既削弱了access token作为一种认证（authentication）手段的作用，同时也增加了出现安全问题的可能性。[5]

**无记名token（Bearer tokens[8]）** - 2.0在协议层去掉所有签名和加密要求，而仅仅依赖于传输层的TLS[6]。这意味着[2.0的token在本质上处于更低安全系数的情况](http://hueniverse.com/2010/09/oauth-bearer-tokens-are-a-terrible-idea/" \t "_blank)。任何想提高token安全性的行为必然要求额外的规范，而从目前提案[展示](http://openid.net/specs/draft-jones-json-web-token-07.html" \t "_blank)的情况来看，工作组仅仅着眼于企业应用范畴。

**token失效（Expiring tokens）** - 2.0的token存在过期时效，并且必须在过期时进行刷新。这对从1.0过来的client开发者而言是最大的改变，现在他们需要自行实现token状态管理。之所以有token过期一说，是因为要适应自编码token的实现——一种在服务器端无需进行数据库查找即可进行认证的加密token。正因为这种自编码token的存在，他们不可能被回收，只能强制为短时效存在，以此减少遭受危险时带来的损失。无论怎样从（server端）去掉签名所带来的好处，都要在（client端）实现token状态管理面前输了两回。

**准许类型（Grant types[7]）** - 在2.0中，授权准许（authorization grants）被用于交换获取access token。准许是一种抽象概念，代表最终用户已同意授权。它可以是用户在访问许可授权页中点击“Approve（同意）”后获取的一个code，也可以是用户实际的用户名和密码。之所以有这个概念，是为了实现多重认证流程。1.0着眼于用1个认证流程适应于多种client类型，而2.0则是明显为不同的client类型增加多个特殊化认证流程。

**犹豫不决的决策**

以上的那些改变，若在一个良好定义的协议上实施是可控管理的。然而由于工作组的组成性质，导致了问题（issue）要不陷在细节纠缠不清，要不就只能保持开放等待所有执行成员共同决定。以下仅是工作组无法达成共识的一小部分**例子**：

- 不强制要求token类型

- 不能对协定基于HMAC算法token类型目标统一意见

- 不要求实施token过期

- 没有关于token字段值长度的指导，其他值类似

- 没有严格要求注册登记流程

- 弱定义client类型

- 缺乏明晰的client安全属性

- 不强制要求准许类型

- 没有一个关于准许类型的适宜性或者适用性指导

- 没有一个实用的native application支持（倒是有许多空口说白话）

- 没有强制要求client认证机制

- 没有限制可扩展性

另一方面，OAuth 2.0为可扩展性定义4个新的登记值，以及通过URI增加额外扩展点，结果引发了一堆关于扩展性的提案。然而真正的问题在于，工作组还是不能为协议定义真正的安全属性。这很清晰地表现在安全考虑环节上，大多处于扑朔迷离的情况。这导致安全专家几乎没有有效的聚焦点可供分析。实际上，工作组推出过70多页用于描述[2.0威胁模型](http://tools.ietf.org/html/draft-ietf-oauth-v2-threatmodel" \t "_blank)的文档，它试图提供更多额外信息，但仍受困于一个相同的根本问题：没有一个实质的协议可供分析。

**现实**

在现实世界，Facebook仍在运行着一年半前的草案12，而且也完全没有理由要他们升级现有的实现形式。毕竟，一个升级过的、用于实现Facebook对接的2.0 client也不太可能复用于其它Oauth服务提供方，反之亦然。Oauth 2.0一点也没有提供代码复用的潜力。

Oauth 2.0提供的是一个关于授权协议的**蓝图**（a **blueprint** for an authorization protocol）。正如上述定义所说，（独立使用）它基本上毫无用武之地，必须要迁并入一个具体的工作解决方案中——而这正是企业化的方式，WS-\*的套路。2.0提供了一个全新的领域，用来推销咨询服务和整合方案。

然而互联网并不需要又一套安全框架。它需要的是一个简单、良好定义、和恰如其分合适的协议，以此提高安全性和增强互操作性。（可以说，）OAuth 2.0在寻求任何有意义的协议替代实现上失败了。

**升级，还是停留**

在最近几个月，许多人都问我，他们是否应该升级到Oauth 2.0，又或者应该推荐哪个协议版本给他们实现。我的答案是不能一概而论。

如果你现在用1.0用得很成功，忽略掉2.0吧。它并没有比1.0体现出更好的价值。（我猜测，用你们服务的client开发者现在已经很熟悉1.0的签名机制了。）

如果你在这个领域是个新来者，而且认为自己是个安全专家，务必要在仔细小心地审查Oauth 2.0的各种特性后再使用。如果你不是专家，要不就使用Oauth 1.0，要不就复制一个你信得过的Oauth 2.0服务方实现形式，来把事情做对（Facebook的API文档是一个很好的着手地方）。2.0对于大规模扩展（实施）相对较好，但是如果要运行大规模改造，你很可能还需要一些安全专家现场估算各种情况和解决方案。

**现在该咋办？**

我现在希望有人接手2.0，然后创作10页左右、抛弃企业用途并更适用于绝大部分互联网服务提供方的文档。版本2.1应该更接近于版本1.5。但是在IETF，这根本不可能发生。在那个社区氛围中，全都是企业级用例。如果你看看他们的其他努力（比如[OpenID Connect](http://openid.net/connect/" \t "_blank)，本来超级简单的提案最终变成了一大打复杂的规范）就会知道，他们没能力让事情简化。

我认为OAuth牌子已经在衰退。这个框架还会存在一段时间，并且在当前缺乏替代品的情况下，它还会得到更广泛的接纳。然而我们**更乐于看到接下来的几年里，Oauth 2.0会爆发各种重大的安全故障**，还有其缓慢但稳步的品牌贬值。这将成为又一个你所厌恶却陷于其中泥潭的协议。

与此同时，我希望众多新兴社区们提出一些骨子里更接近1.0而非2.0的东西。Oauth 1.0是所有关于小型互联网创业团队寻找快速解决已明确定义问题的用户方案。老实点说，我都不知道Oauth 2.0究竟要解决什么样的用例。

**写在最后**

对于一个曾经有前途的社区，结论却是悲哀的。OAuth曾作为小型、快速且有用的标准模范，在标准机构之外无需繁琐过程和法律花销中产生。

（然而现在，）我们的标准制定过程已经破碎地无法修复。这个结局直接来源于IETF的本性，还有监督这份工作时的某些特定性格。更直白地说，这并非由于某些坏人或者无能者导致的。恰恰相反，大家都非常有才华、聪明、和蔼可亲等等。然而大多数人表态只为自身企业领域服务，而剩下的我们已在事实上无法与之竞争。

将OAuth带入IETF是个巨大的错误。[另一种替代方案WRAP也不会有更好的结局](http://hueniverse.com/2009/11/wrap-and-the-demise-of-the-oauth-community/" \t "_blank)，但至少**不用花费3年时间**才得知这个下场。只要能坚持住，我尽可能地留在里面，去为我自认为最适合互联网的模式而战。可是就自己而言，我并没有从所做的决策中得到什么。到了最后，一个反对派的声音只能将事情延缓，但不能改变什么。

我失败了。

我们都失败了。



*插图翻译：*

*- 噢，老天他们扼杀了Oauth。*

*- 你妹的！*

[一些更多思考…...](http://hueniverse.com/2012/07/on-leaving-oauth/)

**译注：**

==========================

[1] 部分词汇并没有翻译，原因是开发者和架构师在交流OAuth协议时，基本倾向于直接使用英文词汇单词，而此文翻译也主要面向他们，故保留原意。以下给出这些词汇的意思：

token：访问令牌。可简单理解为打开家门的钥匙、门禁卡等。

server：OAuth服务器，服务器提供方。

client：客户端，和提交到IETF之前的“消费方”（Consumer）等义。可简单理解为直接向OAuth服务器发起请求的应用、客户端、甚至是浏览器等。

native application：原生应用。一般为手机APP和桌面程序。

[2]本文开始部分的翻译参照了开源中国的文章，在此致谢：[http://www.oschina.net/news/31399/oauth20-road-to-hell](http://www.oschina.net/news/31399/oauth20-road-to-hell" \t "_blank)

[3] 此话相当于“好心办坏事”。来源不可考：[http://zhidao.baidu.com/question/32462302.html](http://zhidao.baidu.com/question/32462302.html" \t "_blank)

[4] 关于“深入了解”部分，如果不了解OAuth协议的开发者，可参考以下给出新浪微博开放平台的调用用户信息API接口的请求例子（HTTP HEADER部分），再对照原文阅读：

【HTTP, Sina weibo OAuth 1.0a】

GET /users/show/123456.json?oauth\_consumer\_key=【应用APP KEY】&oauth\_nonce=【OAuth随机值】&oauth\_signature=【查询参数+应用APP SECRET+Access Token Secret三者共同进行签名后的值】&oauth\_signature\_method=HMAC-SHA1&oauth\_timestamp=【请求时间】&oauth\_token=【Access Token】&oauth\_version=1.0a HTTP/1.1

Host: api.t.sina.com.cn

【HTTPS, Sina weibo OAuth 2.0】

Host: api.weibo.com

GET /2/users/show.json?uid=123456 HTTP/1.1

Authorization: OAuth2 【Access Token】

[5] 可参照微软研究组发给IETF的信件：<http://www.ietf.org/mail-archive/web/oauth/current/msg09270.html>

[6] 常说的HTTPS

[7] 此处grant为名词

[8] 参见：<http://www.infoq.com/cn/news/2010/09/oauth2-bad-for-web>

==============

翻译存疑：

[1] addressing native applications：由于本人开发背景是web开发，故此处暂不能理解。暂时翻译为“标记native application”。

[2] “But most of them show up to serve their corporate overlords”。这句咋理解？

[3] “The web community was looking for a protocol very much in-line with 1.0, with small improvement in areas that proved lacking”。area指的是1.0这个协议呢，还是说在寻找中的协议呢？从工作经验看，似乎是后者？

[4] “produced outside standards bodies without all the process and legal overhead” 这话好想意译为“在标准机构之外野蛮生长”的-\_-||