1.简介（1. Introduction）

1. 在传统的客户端-服务器身份验证模式中，客户端请求服务器上访问受限的资源（受保护的资源）时，需要使用资源所有者的凭据在服务器上进行身份验证。资源所有者为了给第三方应用提供受限资源的访问权限，需要与第

OAuth通过引入授权层以及从资源所有者角色分离出客户端角色来解决这些问题。在OAuth中，客户端请求对受资源所有者控制且托管在资源服务器上的资源的访问权限，并授予一组不同于资源所有者所拥有的凭据。

作为使用资源所有者的凭据访问受保护资源的替代，客户端获得一个访问令牌————一个代表特定作用域、生命周期以及其他访问权限属性的字符串。访问令牌由授权服务器在资源所有者认可的情况下颁发给第三方客户端。客户

例如，终端用户（资源所有者）可以许可一个打印服务（客户端）访问她存储在图片分享网站（资源服务器）上的受保护图片，而无需与打印服务分享自己的用户名和密码，而是，她直接与图片分享网站信任的服务器（授权服务器）进

本规范是为RFC2616）协议设计的。在任何非HTTP协议上使用OAuth不在本规范的范围之内。

OAuth 1.0（RFC5849）作为一个指导性文档发布，是一个小的特设团体的工作成果。本标准化规范在OAuth 1.0的部署经验之上构建，也包括从更广泛的IETF社区收集到其他用户案例和可扩展性需
1.0的实现者应该理解本文档，而非对有关OAuth 2.0的结构和细节做任何假设。

1. Introduction

```
   In the traditional client-server authentication model, the client   requests an access-
restricted resource (protected resource) on the   server by authenticating with the server using
the resource owner's
   credentials.  In order to provide third-party applications access to   restricted resources,
the resource owner shares its credentials with   the third party.  This creates several problems
and limitations:
   o  Third-party applications are required to store the resource      owner's credentials for
future use, typically a password in
      clear-text.
   o  Servers are required to support password authentication, despite      the security
weaknesses inherent in passwords.
   o  Third-party applications gain overly broad access to the resource      owner's protected
resources, leaving resource owners without any      ability to restrict duration or access to a
limited subset of
      resources.
```

o Compromise of any third-party application results in compromise of the end-user's password and all of the data protected by that
password.

OAuth addresses these issues by introducing an authorization layer and separating the role of the client from that of the resource owner. In OAuth, the client requests access to resources controlled
by the resource owner and hosted by the resource server, and is issued a different set of credentials than those of the resource
owner.

Instead of using the resource owner's credentials to access protected resources, the client obtains an access token -- a string denoting a specific scope, lifetime, and other access attributes. Access tokens are issued to third-party clients by an authorization server with the
approval of the resource owner. The client uses the access token to access the protected resources hosted by the resource server.

For example, an end-user (resource owner) can grant a printing service (client) access to her protected photos stored at a photo- sharing service (resource server), without sharing her username and
password with the printing service. Instead, she authenticates directly with a server trusted by the photo-sharing service (authorization server), which issues the printing service delegation-
specific credentials (access token).

This specification is designed for use with HTTP ([RFC2616]). The use of OAuth over any protocol other than HTTP is out of scope.

The OAuth 1.0 protocol ([RFC5849]), published as an informational document, was the result of a small ad hoc community effort. This Standards Track specification builds on the OAuth 1.0 deployment
experience, as well as additional use cases and extensibility requirements gathered from the wider IETF community. The OAuth 2.0 protocol is not backward compatible with OAuth 1.0.

```
The two versions
   may co-exist on the network, and implementations may choose to   support both.  However, it
is the intention of this specification   that new implementations support OAuth 2.0 as specified
in this
   document and that OAuth 1.0 is used only to support existing   deployments.  The OAuth 2.0
protocol shares very few implementation   details with the OAuth 1.0 protocol.  Implementers
familiar with
   OAuth 1.0 should approach this document without any assumptions as to
   its structure and details.
```

Revision #4
Created Tue, Mar 24, 2020 11:48 PM by □□□
Updated Wed, Mar 25, 2020 12:29 AM by □□□