Autorización OAuth2

OAuth es un estándar abierto para la delegación de acceso, utilizado comúnmente como una forma para que los usuarios de Internet concedan a sitios web u otro tipo de aplicaciones acceso a su información en otros sitios web, pero sin darles las contraseñas. Este mecanismo es utilizado por compañías como Google, Facebook, Microsoft y Twitter para permitir a los usuarios compartir información sobre sus cuentas con aplicaciones de terceros o sitios web.

En general, OAuth proporciona a los clientes un "acceso delegado seguro" a los recursos del servidor en nombre de un propietario de recursos. Especifica un proceso para que los propietarios de recursos autoricen el acceso de terceros a sus recursos de servidor sin compartir sus credenciales. Está diseñado específicamente para trabajar con HTTP (Hypertext Transfer Protocol), OAuth esencialmente permite que los tokens de acceso sean emitidos a clientes de terceros por un servidor de autorización, con la aprobación del propietario del recurso. A continuación, el tercero utiliza el token de acceso para acceder a los recursos protegidos alojados por el servidor de recursos.

OAuth es un servicio que es complementario y distinto de OpenID. OAuth también es distinto de OATH, que es una arquitectura de referencia para la autenticación, no un estándar para la autorización. Sin embargo, OAuth está directamente relacionado con OpenID Connect (OIDC), ya que OIDC es una capa de autenticación construida sobre OAuth 2.0. OAuth también es distinto de XACML, que es un estándar de política de autorización. OAuth se puede utilizar en combinación con XACML donde se utiliza OAuth para el consentimiento de propiedad y la delegación de acceso mientras que XACML se utiliza para definir las políticas de autorización (por ejemplo, los administradores pueden ver documentos en su región).

Historia

OAuth comenzó en noviembre de 2006 cuando Blaine Cook estaba desarrollando la implementación de OpenID de Twitter. Mientras tanto, Ma.gnolia necesitaba una solución para permitir a sus miembros con OpenIDs autorizar a Dashboard Widgets a acceder a su servicio. Cook, Chris Messina y Larry Halff de Magnolia se reunieron con David Recordon para discutir el uso de OpenID con las API de Twitter y Ma.gnolia para delegar la autenticación. Concluyeron que no había estándares abiertos para la delegación de acceso a API.

El grupo de discusión OAuth fue creado en abril de 2007, para que el pequeño grupo de implementadores escribiera el proyecto de propuesta para un protocolo abierto. DeWitt Clinton de Google se enteró del proyecto OAuth y expresó su interés en apoyar el esfuerzo. En julio de 2007, el equipo redactó una especificación inicial. Eran Hammer se unió y coordinó las muchas contribuciones de OAuth creando una especificación más formal. El 4 de diciembre de 2007, se publicó el borrador final de OAuth Core 1.0.

En la 73ª Reunión del Grupo de Trabajo de Ingeniería de Internet (IETF) en Minneapolis

en noviembre de 2008, se celebró un BoF de OAuth para discutir la introducción del protocolo en el IETF para un trabajo de normalización adicional. El evento fue muy asistido y hubo un amplio apoyo para formar oficialmente un grupo de trabajo OAuth dentro de la IETF.

El protocolo OAuth 1.0 fue publicado como RFC 5849, una Solicitud informativa de Comentarios, en abril de 2010.

Desde el 31 de agosto de 2010, todas las aplicaciones de Twitter de terceros han sido requeridas para usar OAuth.

El marco OAuth 2.0 fue publicado como RFC 6749, y el uso de Token de portador como RFC 6750, ambos estándares de seguimiento de las solicitudes de comentarios, en octubre de 2012.

OAuth 2.0

OAuth 2.0 no es compatible con OAuth 1.0. OAuth 2.0 proporciona flujos de autorización específicos para aplicaciones web, aplicaciones de escritorio, teléfonos móviles y dispositivos de sala de estar. La especificación y RFCs asociados son desarrollados por el IETF OAuth WG; el marco principal se publicó en octubre de 2012.

Facebook 's Graph API sólo admite OAuth 2.0. Google apoya OAuth 2.0 como el mecanismo de autorización recomendado para todas sus API. Microsoft también admite OAuth 2.0 para varias API y su servicio Azure Active Directory, que se utiliza para proteger muchas API de Microsoft y de terceros.

El OAuth 2.0 Framework y el uso de token de portador se publicaron en octubre de 2012.

Seguridad

El 23 de abril de 2009, se anunció una falla de seguridad de fijación de sesión en el protocolo 1.0. Afectaba al flujo de autorización de OAuth (también conocido como "OAuth de 3 patas") en OAuth Core 1.0 Sección 6. Se publicó la versión 1.0a del protocolo OAuth Core para solucionar este problema.

OAuth 2.0 no admite firma, cifrado, enlace de canal o verificación de cliente. Se basa completamente en TLS para cierto grado de confidencialidad y autenticación de servidor.

OAuth 2.0 ha tenido numerosas fallas de seguridad expuestas en las implementaciones. El propio protocolo ha sido descrito como inherentemente insegura por los expertos en seguridad y un principal contribuyente a la especificación declaró que los errores de ejecución son casi inevitables.

En enero de 2013, Internet Engineering Task Force publicó una serie de modelos de amenaza para OAuth 2.0. Entre ellos se encontraba uno llamado "Open Redirect"; En la

primavera de 2014, esto fue descrito bajo el nombre de "Covert Redirect" de Wang Jing.

Posiblemente el fallo de seguridad OAuth más devastador es la vulnerabilidad de phishing: cada sitio web que utiliza OAuth está visualmente (pero no técnicamente) pidiendo a los usuarios finales su nombre de usuario y contraseña de su identidad maestra, lo que impide que los usuarios comunes entiendan que no deberían escribir, los que se encuentran en el sitio web de un atacante que visualmente emula este proceso para robar credenciales. La autenticación tradicional de dos factores (utilizando contraseñas de una sola vez) no impide este ataque, ya que el sitio de phishing puede robarlo también y usarlo de inmediato (tenga en cuenta que los tokens de Factor Universal 2 no son vulnerables a este ataque de tipo específico).

En abril-mayo de 2017, alrededor de un millón de usuarios de Gmail (menos del 0,1% de los usuarios a mayo de 2017) fueron atacados por un ataque de phishing basado en OAuth, recibiendo un correo electrónico que supuestamente provenía de un amigo, empleador o amigo que quería compartir un documento en Google Docs. Aquellos que hicieron clic en el enlace dentro del correo electrónico fueron dirigidos a iniciar sesión y permitir que un programa de terceros potencialmente malintencionado llamado "Google Apps" acceda a su "cuenta de correo electrónico, contactos y documentos en línea". En "aproximadamente una hora", el ataque de phishing fue detenido por Google, quien aconsejó a aquellos que habían dado "Google Apps" acceso a su correo electrónico para revocar tal acceso y cambiar sus contraseñas.

No interoperabilidad

Debido a que OAuth 2.0 es más un framework que un protocolo definido, es menos probable que una implementación de OAuth 2.0 sea naturalmente interoperable con otra implementación de OAuth 2.0. Se requieren más perfiles de despliegue y especificación para cualquier interoperabilidad.

Usos

OAuth puede utilizarse como un mecanismo de autorización para consumir RSS / ATOM seguras. El consumo de RSS / ATOM feeds que requieren autenticación siempre ha sido un problema. Por ejemplo, un feed RSS de un sitio de Google seguro no se puede consumir con Google Reader. En su lugar, OAuth se puede utilizar para autorizar a Google Reader a acceder a la fuente RSS de ese sitio de Google.

OAuth y otras normas

OpenID vs. pseudo-autenticación usando OAuth

OAuth es un protocolo de autorización, en lugar de un protocolo de autenticación. El uso de OAuth por sí solo como método de autenticación puede denominarse pseudo-autenticación. Los siguientes diagramas destacan las diferencias entre el uso de OpenID (específicamente diseñado como un protocolo de autenticación) y OAuth para la

autenticación.

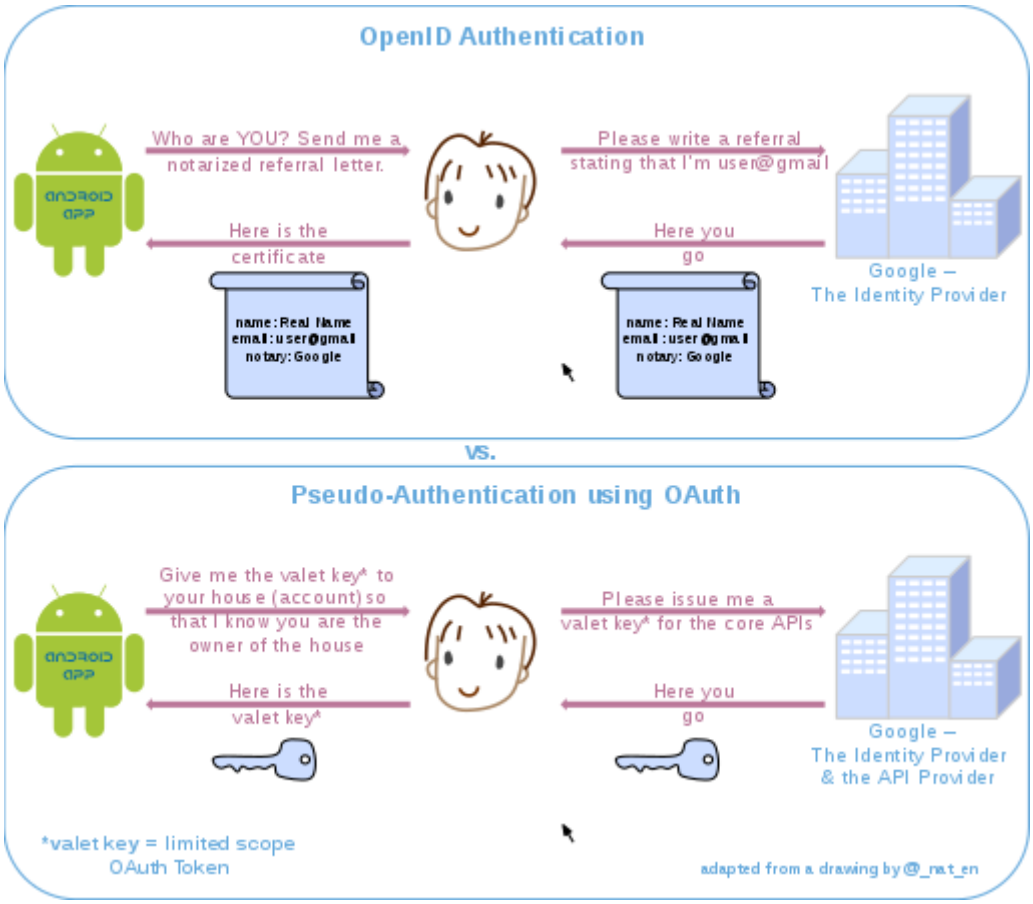
El flujo de comunicación en ambos procesos es similar:

1. El usuario solicita un inicio de sesión de recurso o sitio desde la aplicación.
2. El sitio ve que el usuario no está autenticado. Formula una solicitud para el proveedor de identidad, lo codifica y lo envía al usuario como parte de una URL de redireccionamiento.
3. El navegador del usuario solicita la URL de redireccionamiento para el proveedor de identidad, incluida la solicitud de la aplicación
4. Si es necesario, el proveedor de identidad autentica al usuario (tal vez pidiéndoles su nombre de usuario y contraseña)
5. Una vez que el proveedor de identidad esté satisfecho de que el usuario está suficientemente autenticado, procesa la solicitud de la aplicación, formula una respuesta y la devuelve al usuario junto con una URL de redireccionamiento a la aplicación.
6. El navegador del usuario solicita la URL de redireccionamiento que se remonta a la aplicación, incluida la respuesta del proveedor de identidad
7. La aplicación descodifica la respuesta del proveedor de identidad y continúa en consecuencia.
8. (OAuth solamente) La respuesta incluye un token de acceso que la aplicación puede utilizar para obtener acceso directo a los servicios del proveedor de identidad en nombre del usuario.

La diferencia crucial reside en que en el caso de uso de la autenticación OpenID, la respuesta del proveedor de identidad es una afirmación de identidad; Mientras que en el caso de uso de autorización de OAuth, el proveedor de identidad es también un proveedor de API y la respuesta del proveedor de identidad es un token de acceso que puede conceder a la aplicación acceso continuo a algunas de las API del proveedor de identidad en nombre del usuario. El token de acceso actúa como una especie de "tecla de valet" que la aplicación puede incluir con sus peticiones al proveedor de identidad, que demuestran que tiene permiso del usuario para acceder a dichas API.

Dado que el proveedor de identidad típicamente (pero no siempre) autentica al usuario como parte del proceso de concesión de un token de acceso de OAuth, es tentador ver una solicitud de token de acceso de OAuth satisfactoria como un método de autenticación. Sin embargo, debido a que OAuth no fue diseñado teniendo en cuenta este caso de uso, esta suposición puede llevar a fallas importantes de seguridad.

OpenID vs. pseudo-autenticación usando OAuth



OAuth y XACML

XACML es un framework de autorización de control de acceso basado en políticas y basado en atributos. Proporciona:

- Una arquitectura de control de acceso.
- Un lenguaje de políticas con el que expresar una amplia gama de políticas de control de acceso, incluidas las políticas que pueden utilizar los consentimientos manejados / definidos a través de OAuth.
- Un esquema de solicitud / respuesta para enviar y recibir solicitudes de autorización.

XACML y OAuth se pueden combinar para ofrecer un enfoque más completo de la autorización. OAuth no proporciona un lenguaje de políticas con el que definir políticas de control de acceso. XACML se puede utilizar para su lenguaje de políticas.

Donde OAuth se enfoca en el acceso delegado (yo, el usuario, concedo acceso de Twitter a mi muro de Facebook) y la autorización centrada en la identidad, XACML toma un enfoque basado en atributos que puede considerar los atributos del usuario, la acción, el recurso y el Contexto (quién, qué, dónde, cuándo, cómo).

Con XACML es posible definir políticas como:

- Los gerentes pueden ver documentos en su departamento
- Los administradores pueden editar los documentos que poseen en modo borrador

XACML proporciona un control de acceso más fino que OAuth. OAuth está limitado en granularidad a la funcionalidad gruesa (los ámbitos) expuestos por el servicio de destino. Como resultado, a menudo tiene sentido combinar OAuth y XACML juntos donde OAuth proporcionará el caso de uso de acceso delegado y la administración de consentimiento y XACML proporcionará las políticas de autorización que funcionan en las aplicaciones, procesos y datos.

Por último, XACML puede trabajar de forma transparente en múltiples pilas (API, SSO web, ESBs, aplicaciones domésticas, bases de datos...). OAuth se enfoca exclusivamente en aplicaciones basadas en HTTP.

Controversia

En julio de 2012, Eran Hammer renunció a su cargo de autor principal del proyecto OAuth 2.0, se retiró del grupo de trabajo de la IETF y retiró su nombre de la especificación. Hammer apuntó a un conflicto entre la web y las culturas empresariales, citando a la IETF como una comunidad que es "todo sobre casos de uso empresarial", que es "no es capaz de simplificarlo". Lo que ahora se ofrece es un modelo para un protocolo de autorización, dice, y "esa es la forma de la empresa", proporcionando una "nueva frontera para vender servicios de consultoría y soluciones de integración".

Al comparar OAuth 2.0 con 1.0, Hammer señala que se ha vuelto "más complejo, menos interoperable, menos útil, más incompleto y lo más importante, menos seguro". Explica cómo los cambios arquitectónicos para los tokens sin consolidar 2.0 de los clientes, eliminando todas las firmas y la criptografía a un nivel de protocolo y agregó tokens de expiración (porque los tokens no podían ser revocados), mientras que complica el procesamiento de la autorización. Numerosos artículos se dejaron sin especificar o fin finalizar en la especificación porque "como ha sido la naturaleza de este grupo de trabajo, ningún problema es demasiado pequeño para quedarse atascado o dejar abierto para que cada aplicación decida".

David Recordon más tarde también retiró su nombre de las especificaciones por razones no especificadas. Dick Hardt asumió el papel de editor, y el framework fue publicado en octubre de 2012.