چارچوب مجوزدهی به برنامه‌های اینترنتی

استاندارد OAuth2.0

گروه مهندسی اینترنت

کسری دولت‌خواهی، مینا رستم‌پور

فایل نمونه


بسم الله
الرحمن
الرحيم

عنوان : استاندارد OAuth 2.0
نویسندگان : گروه مهندسی اینترنت (IETF)
مترجمان: کسری دولت‌خواهی، مینا رستم‌پور
مشخصات نشر : تهران: راه پرداخت، ۱۴۰۱.
مشخصات ظاهری : ۱۶۷ ص.؛ ۵/۲۱×۵/۱۴ س‌م.
شابک: ۷-۴۷-۷۷۰۲-۶۲۲-۹۷۸
وضعیت فهرست نویسی : فیپا
یادداشت : عنوان اصلی: کتاب حاضر ترجمه فایل الکترونیکی با فرمت HTMLبه آدرس الکترونیکی https://datatracker.ietf.org/doc/html/rfc6749 است.
عنوان دیگر : چارچوب مجوزدهی به برنامه‌های اینترنتی.
موضوع : شبکه‌های کامپیوتری - تدابیر ایمنی
موضوع: وبگاه‌ها - کنترل دستیابی
موضوع : وب - سرورها - کنترل دستیابی
موضوع : حفاظت داده‌ها
شناسه افزوده : رستم‌پور، مینا، ۱۳۷۰-
شناسه افزوده : دولت‌خواهی، کسری، ۱۳۶۷-
شناسه افزوده : گروه مهندسی اینترنت
رده بندی کنگره: TK۵۱۰۵/۵۹
رده بندی دیویی : ۰۰۵/۸
شماره کتابشناسی ملی : ۸۹۵۷۴۷۵

چارچوب مجوزدهی به برنامه‌های اینترنتی

استاندارد OAuth2.0

گروه مهندسی اینترنت

کسری دولت‌خواهی، مینا رستم‌پور

۱

انتشارات راه‌پرداخت

INDEPENDENTLY PUBLISHED

عنوان : استاندارد OAuth 2.0

ناشر: راه پرداخت

نویسنده: گروه مهندسی اینترنت (IETF)

مترجمان: کسری دولت‌خواهی، مینا رستم‌پور

صفحه‌آرا: علیرضا کیوان

ناظر چاپ: قادر شهبازی

نوبت چاپ: اول ۱۴۰۱

شمارگان: ۱۰۰۰ نسخه

شابک: ۹۷۸-۶۲۲-۷۷۰۲-۴۷-۷

تلفن: ۰۲۱-۴۴۴۴۳۹۶۶

دورنگار: ۸۹۷۸۴۹۰۲

ایمیل : publisher@way2pay.press

وب‌سایت : way2pay.press

لیتوگرافی: هنر اشکان

چاپ و صحافی: واژه

فروشگاه انتشارات راه‌پرداخت نشانی: تهران، جنت‌آباد جنوبی، خیابان لاله غربی، روبه‌روی پاساژ سمرقند، خیابان حدیث، کوچه حدیث دوم، پلاک ۸

@way2paypress Way2Pay.press @way2paypress

فهرست

آشنایی با OAuth

فصل صفر خارج از متن اصلی استاندارد OAuth 2.0 تدوین شده است و هدف آن ارائه تصویری کلی از کتاب پیش رو است. این استاندارد با نگارشی دقیق، علمی و شبه حقوقی نوشته شده است؛ بنابراین لازم بود خارج از چارچوب استاندارد بخش‌هایی با تفسیر بیشتر به آن افزوده شود.

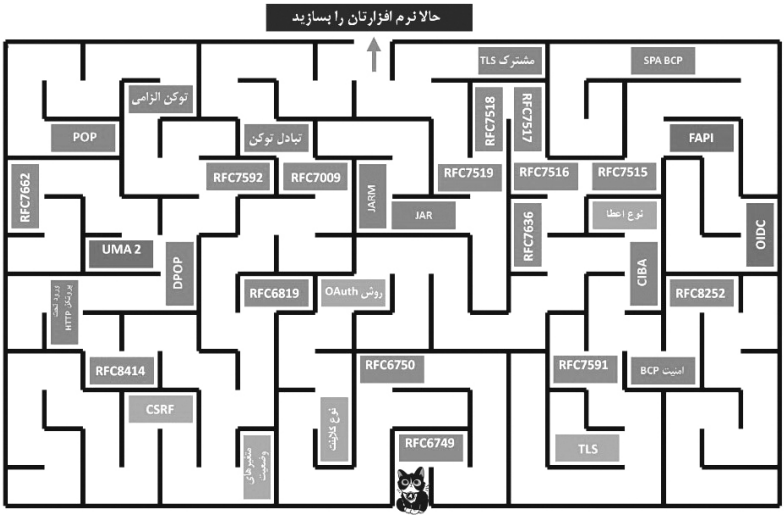
آمادگی برای مرور چارچوب OAuth

برای جواب به این سؤال که OAuth چیست باید کمی مقدمه‌چینی کنیم. اما اگر در یک کلام بخواهیم بگوییم OAuth یک استاندارد اینترنتی است. OAuth 2.0 ایمن‌ترین استاندارد به‌اشتراک‌گذاری داده در بازار است.

این استاندارد به کاربران اینترنت این اجازه را می‌دهد که اطلاعات کاربری خود را بدون اشتراک‌گذاری نام کاربری و گذرواژه‌شان به‌صورت امن با ارائه‌دهندگان خدمات دیگر به اشتراک بگذارند. چارچوب مجوز OAuth برای برنامه شخص ثالث، دستورالعملی با ساختاری پیچیده و تودرتو است و واژگان آن باتوجه‌به ماهیتش بسیار بادقت انتخاب شده‌اند، اما طوری بیان نشده‌اند که بتوانیم گام‌به‌گام با آن پیش برویم و OAuth را یاد بگیریم. در واقع، این سند ما را وارد مازی پیچیده می‌کند که باید به طریقی راه خودمان را از میان آن پیدا کنیم. دشوارترین بخش OAuth همین است که مخاطب باید بفهمد کوره‌راه‌های این مسیر چگونه با هم ارتباط پیدا می‌کنند.

متناسب با این ویژگی، ما هم در این فصل رویکردی متفاوت اتخاذ کردیم و قصد نداریم سند را موبه‌مو مرور کنیم.

در این فصل سعی شده تا در مورد امنیت APIها از طریق فعال‌سازی چارچوب مجوز OAuth برای برنامه شخص ثالث سخن بگوییم. برای توضیح ابتدا با استانداردهای اینترنتی آشنا شویم. در ادامه از ضرورت وجود OAuth صحبت می‌کنیم، سپس تاریخچه پیدایش آن را بررسی کرده و در انتها جریان‌های اصلی پروتکل آن را بررسی خواهیم کرد.



شکل ۱. مسیر تودرتوی چارچوب OAuth

استانداردهای اینترنت

استانداردهای اینترنت مجموعه اصولی از استانداردها در مهندسی شبکه‌های کامپیوتری است که ابعاد یک فناوری یا روش مناسب برای اینترنت را تشریح می‌کند. این استانداردها تعامل بین سخت‌افزار و نرم‌افزار از منابع مختلف را در راستای ایجاد شبکه اینترنت امکان‌پذیر می‌کنند و در واقع زبان ارتباطات جهانی بین اجزای مختلف اینترنت هستند. بسیاری از این استانداردها سال‌ها قبل‌تر از ظهور اینترنت شکل گرفته بودند، چیزی در حدود سال ۱۹۷۰ میلادی در زمانی که حتی کامپیوترهای شخصی اختراع نشده بودند. امروزه این استانداردها توسط کارگروه مهندسی اینترنت یا IETF تدوین و ابلاغ می‌شوند.

کارگروه مهندسی اینترنت بالارده‌ترین سازمان استانداردهای اینترنت است که استانداردهای باز را از طریق فرآیندهای باز توسعه می‌دهد. IETF یک جامعه باز بین‌المللی بزرگ متشکل از طراحان، اپراتورها، تولیدکنندگان و محققان شبکه است که به تکامل معماری اینترنت و عملکرد روان آن می‌پردازند. کارهای فنی IETF در گروه‌های کاری انجام

می‌شود که بر اساس موضوع در چندین حوزه سازماندهی شده‌اند. IETF کار خود را در سال ۱۹۸۶ میلادی آغاز کرده است. استانداردهای اینترنت و سایر اسنادی که توسط IETF منتشر می‌شوند - و الزاماً استاندارد نیستند - توسط عبارت RFC به همراه یک کد چهاررقمی نام‌گذاری می‌شوند. RFC از سرواژه‌های عبارت «درخواست برای نظر» ساخته شده است.

ایجاد یک استاندارد دو مرحله کلی دارد، استاندارد پیشنهادی و استاندارد اینترنت. این مراحل به‌اصطلاح سطوح بلوغ نامیده می‌شوند و این فرآیند شکل‌گیری «مسیر استاندارد» نامیده می‌شود. یکی از شاخصه‌های اصلی که میزانِ بلوغ یک پیشنهاد را مشخص می‌کند، اقبال و پذیرش عمومی در جوامع اینترنتی است. هر مشارکتی در IETF در ابتدا به‌عنوان یک پیش‌نویس اینترنت یا Internet Draft شروع می‌شود، پس از مدتی در صورت پذیرفته شدن توسط اعضا ممکن است به یک RFC یا درخواست برای نظر ارتقا پیدا کند و این RFC در نهایت پس از چندین بازنگری ممکن است به یک استاندارد اینترنتی تبدیل شود. RFC بخشی از یک پیشنهاد است که در مسیر استاندارد قرار دارد. در مرحله اول، استاندارد پیشنهاد می‌شود و در پی آن سازمان‌ها و افراد تصمیم می‌گیرند که آیا این استاندارد پیشنهادی را اجرا کنند یا خیر. پس از رعایت معیارهای RFC 6410 (دو پیاده‌سازی مجزا، استفاده گسترده، بدون اشتباه و ...) RFC می‌تواند به استاندارد اینترنت ارتقا پیدا کند.

استاندارد پیشنهادی

استاندارد پیشنهادی سندی پایدار است، مشکلات شناخته شده را حل کرده، بازنگری‌های قابل توجهی از جامعه دریافت کرد و از نظر جامعه ارزشمند تلقی می‌شود. معمولاً برای تعیین یک سند مشخص به‌عنوان استاندارد پیشنهادی، اجرایی بودن یا تجربه عملیاتی ضروری نیست.

استانداردهای پیشنهادی از کیفیتی برخوردارند که می‌توان پیاده‌سازی‌های مختلف آنها را در اینترنت اجرایی کرد. با این حال، مانند تمام اسناد فنی استانداردهای پیشنهادی ممکن است در صورت بروز مشکلات یا شناسایی راه‌حل‌های بهتر بازنگری شوند.

بسیاری از استانداردهای پیشنهادی هم اکنون در اینترنت اجرایی شده و به طور گسترده

به‌عنوان پروتکل‌های پایدار استفاده می‌شوند. حرکت از سوی «استاندارد پیشنهادی» به سمت تبدیل‌شدن به «استاندارد اینترنت» در دنیای واقعی بسیار نادر است و اکثر پروتکل‌های محبوب IETF در حد استاندارد پیشنهادی باقی می‌مانند.

پیش‌نویس استاندارد

پیش از سال ۲۰۱۱ این سطح بلوغ استفاده می‌شد اما در اکتبر ۲۰۱۱، سند RFC 6410 سطح بلوغ دوم و سوم را در یک استاندارد پیش‌نویس ادغام کرد. اما هنوز استانداردهای قدیمی پیش‌نویس در لیست RFC‌ها وجود دارند.

استاندارد اینترنت

یک استاندارد اینترنت درجه بالایی از بلوغ فنی دارد و باور عمومی این است که پروتکل یا سرویس مشخص‌شده مزایای قابل‌توجهی برای جامعه اینترنتی در پی دارد. به‌طورکلی استانداردهای اینترنت قابلیت تعامل سیستم‌ها در اینترنت را از طریق تعریف پروتکل‌ها، قالب‌های پیام و زبان‌ها فراهم می‌کنند. اساسی‌ترین استانداردهای اینترنت آن‌هایی هستند که پروتکل اینترنت را تعریف می‌کنند.

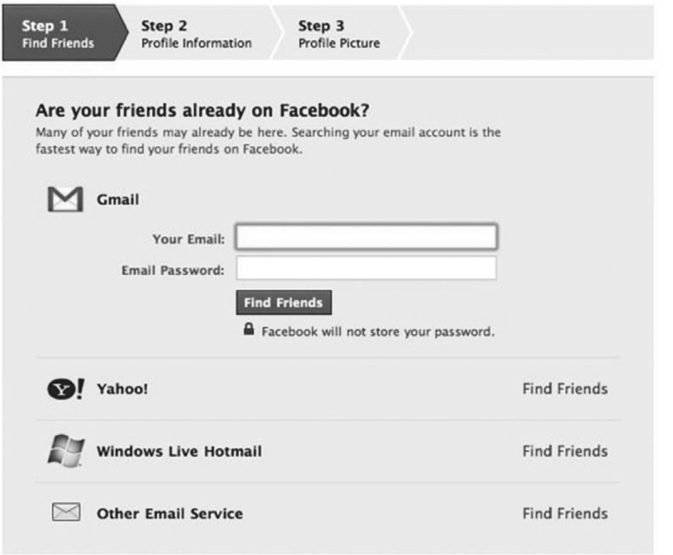
استانداردهای اینترنت تضمین می‌کند که سخت‌افزار و نرم‌افزار تولیدشده توسط تولیدکنندگان مختلف می‌توانند با هم به‌درستی کار کنند. استانداردها توسعه نرم‌افزارها و سخت‌افزارها را بسیار آسان‌تر می‌کند زیرا در هنگام اتصال شبکه می‌توان نرم‌افزار و سخت‌افزار را در یک لایه و در یک زمان توسعه داد. به طور معمول استانداردهایی که در ارتباطات داده استفاده می‌شود پروتکل نامیده می‌شوند.

اگر دنبال تعریف دقیق‌تری از استانداردهای اینترنت باشیم می‌توانیم از تعریف موجود در RFC2026 استفاده کنیم. طبق این تعریف:

«به‌طورکلی، استاندارد اینترنت اسناد مشخصه پایداری هستند که به‌خوبی درک شده‌اند. از طرفی از نظر فنی صلاحیت دارند و دارای پیاده‌سازی‌های متعدد هستند. همچنین مستقل بوده و درعین‌حال با تجربه‌های عملیاتی فعلی قابلیت تعامل دارند و از حمایت عمومی قابل‌توجهی برخوردارند. همچنین به‌وضوح در بخش‌هایی از اینترنت کارآمدی دارند.»

کتاب پیش رو ترجمه یکی از همین RFC هاست. استانداردی که تحت عنوان OAuth 2.0 شناخته می‌شود و به‌صورت رسمی RFC 6749 نام دارد. این استاندارد نحوه اشتراک گذاری امن اطلاعات تحت مالکیت یک شخص در بستر اینترنت را تشریح می‌کند. اما چرا این امر ضروری است این سؤالی است که در بخش بعد به آن پاسخ می‌دهیم.

برای تبیین ضرورت وجود OAuth کمی عقب گرد می‌کنیم و به نمونه‌ای برمی‌گردیم. زمانی وقتی در تارنمای فیس‌بوک ثبت‌نام می‌کردیم، فیس‌بوک از ما می‌خواست که نام کاربری و رمز عبور حساب ایمیلمان را وارد کنیم تا با دسترسی یافتن به فهرست مخاطبانمان در حساب کاربری ایمیل، هم از میان فهرست مخاطبان ما کاربران کنونی را برای دوستی به ما پیشنهاد بدهد و هم به ما این امکان را بدهد که برای بقیه دوستانمان دعوت‌نامه عضویت بفرستیم. با این کار فیس‌بوک به جز فهرست مخاطبانمان، به تمام محتوای حساب کاربری ایمیلمان نیز دسترسی پیدا می‌کرد. مسئله این بود که چگونه می‌توان طوری امکان دسترسی فیس‌بوک به فهرست مخاطبان را فراهم کرد که نیازی به اشتراک گذاشتن رمز عبور با آن نباشد و به متن ایمیل‌ها دسترسی پیدا نکند.



شکل ۲. درخواست رمز عبور برای ورود به Facebook

در واقع اولین دلیلی که باعث به‌وجودآمدن استاندارد OAuth شد لزوم اشتراک‌گذاری امن منابع در سطح اینترنت بود. اما این منابع چه بودند و این اشتراک‌گذاری چرا ضروری بود؟ پلتفرم‌هایی مانند یاهو، توییتر و گوگل حجم بالایی از کاربران فعال اینترنت را پوشش می‌دادند. اکثر کاربران منابع خود را که شامل اطلاعات هویتی، عکس‌ها، تاریخچه جست‌وجو، شماره‌تلفن‌ها و غیره را در اختیار این پلتفرم‌ها گذاشته بودند. از طرفی شرکت‌های استارت‌آپی هرروزه در حال گسترش بودند و نیاز به بخشی از این اطلاعات داشتند. همکاری بین این شرکت‌های استارت‌آپی و پلتفرم‌های بزرگی که API‌های خدماتی خود را منتشر می‌کردند باعث بهبود کارایی برنامه‌ها می‌شد.

هنگامی که گوگل برای اولین بار Google Calendar API را منتشر کرد، این توانایی را برای توسعه‌دهندگان برنامه فراهم کرد تا تقویم گوگل کاربر را بخوانند و روی آن تغییر ایجاد کنند. برای مثال یک سرویس فروش بلیت هواپیما نیاز داشت تا بعد از رزرو بلیت، تاریخ پرواز را روی تقویم ما ثبت کند.

در ابتدا تنها راه‌حل این بود که کاربرها نام کاربری و گذرواژه خود را در اختیار این برنامه سوم‌شخص بگذارند (مانند برنامه فروش بلیت). این کار با ریسک‌های فراوانی همراه بود که در ادامه بیشتر به آنها می‌پردازیم. بعد از آن گوگل پروتکل اختصاصی Client Log in گوگل را توسعه داد و در اختیار توسعه‌دهندگان نرم‌افزار گذاشت.

پروتکل‌های اختصاصی مانند Client Login و پروتکل‌های استاندارد مانند HTTP Basic authentication منجر به این شد که برنامه‌های کوچک و بزرگ دیگر هم از کاربران برای دسترسی به داده‌هایشان گذرواژه درخواست کنند.

این امر فقط مختص نرم‌افزارهای نصب شده بر روی سیستم‌عامل نبود بلکه برنامه‌های آنلاین هم درخواست این نوع دسترسی را داشتند. فلیکر، سایت اشتراک‌گذاری عکس آنلاین که در مالکیت یاهو بود، یکی از این برنامه‌ها بود که درخواست دریافت گذرواژه حساب کاربری کاربران گوگل را داشت.

درخواست گذرواژه‌های کاربران گوگل توسط یاهو، هر دو شرکت را ترساند و منجر به توسعه پروتکل‌های اختصاصی جدیدی شد که این مشکل را در وب برطرف کرد.

با این پروتکل‌های جدید، مانند Google's AuthSub و Yahoo!'s BBAuth کاربران ضمن

ورود به حساب‌های خود، اجازه دسترسی به اطلاعات خود را صادر می‌کردند سپس برنامه شخص سوم یک رمز برای دسترسی به داده‌های کاربران دریافت می‌کرد.

درحالی‌که این امر برخی از مشکلات امنیتی را حل کرد، برای توسعه‌دهندگان هزینه‌هایی نیز ایجاد کرد. استارت‌آپ‌هایی که API‌های جدید می‌ساختند در توسعه برنامه‌های احراز هویت اختصاصی با مشکل مواجه می‌شدند.

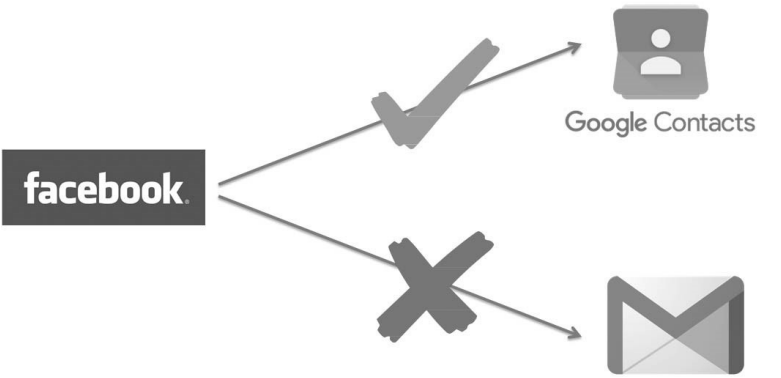
اما همیشه موضوع به این سادگی نبود. عموم اطلاعاتی که سرویس‌های که مانند توییتر نگهداری می‌کردند برای اکثر کاربران اهمیت ویژه‌ای نداشت. برای مثال عکس کاربری، آخرین توییت‌هایی که پسندیده شده بودند یا تصاویر گیف که کاربر از آنها استفاده کرده بود؛ اما در مورد سامانه‌های مالی یا سندهای اداری مهم این قضیه کمی تفاوت داشت. برای مثال کاربر نمی‌توانست گذرواژه حساب بانکی خود را در اختیار یک برنامه دیگر قرار دهد تا کاری مانند پرداخت‌ها قبوض به‌صورت خودکار صورت بگیرد.

در نتیجه استارت‌آپ‌ها و ارائه‌دهندگان اصلی API تصمیم گرفتند به‌جای مدل‌های موجود یک پروتکل استاندارد ایجاد کنند سازوکار فرآیندهای اعطای دسترسی آنلاین را بهبود بخشد.

پروتکلی مانند OAuth برای حل این مشکل ایجاد شد. در اواخر دهه ۲۰۰۰ میلادی، بسیاری از شرکت‌های نرم‌افزاری مشغول ساخت API بودند و تقریباً در همان زمان با مشکل مشابهی دست‌وپنجه نرم می‌کردند؛ ازاین‌رو در نهایت مؤسسان این شرکت‌ها گرد هم آمدند و چیزی را تشکیل دادند که در نهایت به چارچوب OAuth و ایجاد گروهی منجر شد که آن را بازنگری کرده و به‌روز می‌کند. OAuth راهی ایجاد کرد تا کاربر بتواند فقط اجازه دسترسی به بخشی از داده‌های یک حساب را به برنامه موردنظر خود اعطا کند، بدون اینکه رمز عبور را در اختیار آن برنامه بگذارد و یا به محتوای اطلاعات دیگر موجود در حساب کاربری دسترسی بدهد. در نتیجه، امروز دیگر به‌جای اینکه در برنامه‌ای مانند فیس‌بوک، رمز عبور ایمیلتان را وارد کنید، می‌توانید با استفاده حساب گوگل یا نظیر آن وارد برنامه شوید.

فراموش نکنیم که OAuth برای این طراحی نشده است که بگوید چه کسی به سایت وارد شده است. OAuth قرار است مجوزی برای دسترسی برنامه‌ها به داده‌ها و اطلاعاتی مشخص ایجاد کند. مواردی مثل دسترسی فیس‌بوک به مخاطبین گوگل یا دسترسی

وب‌سایت last.fm به تاریخچه آنچه در Spotify گوش داده‌ایم، همه در مورد دسترسی به API هستند و به شناسایی کاربر مربوط نمی‌شوند.



شکل ۳. تفویض اختیار دسترسی به ایمیل

پیدایش استاندارد OAuth

شکل‌گیری OAuth به نوامبر ۲۰۰۶ (آبان سال ۱۳۸۵) برمی‌گردد، زمانی که بلین کوک یکی از توسعه‌دهندگان توییتر در حال کار بر روی اجرای OpenID توییتر بود. OpenID یک فناوری است که به کاربران اجازه می‌دهد بدون ثبت‌نام در یک وب‌سایت با استفاده از اطلاعات خود در سایت‌هایی مانند گوگل (که یکی از سرویس‌دهندگان OpenID است) حساب کاربری مخصوص به خود را داشته باشند. او به دنبال راهی بود تا به‌وسیله OpenID و API توییتر بتواند احراز هویت یک کاربر را به برنامه‌های جانبی توییتر تفویض کند. او به همراه کریس مس ینا (خالق هشتگ)، دیوید رکوردون، لری هالف و دیگران جلسه‌ای برگزار کردند تا در مورد راه‌حل‌های موجود بحث کنند. آنها پس از بررسی عملکرد OpenID موجود و همچنین سایر شیوه‌های صنعت به این نتیجه رسیدند که هیچ استاندارد بازی برای تفویض دسترسی API وجود ندارد. گفتگو آنها برای چند ماه به‌صورت آنلاین ادامه یافت.

در آوریل ۲۰۰۷ (فروردین سال ۱۳۸۶) یک گروه مجازی با تعداد اندکی از برنامه‌نویس‌ها برای نوشتن یک پروپوزال در مورد یک پروتکل باز در خصوص تفویض دسترسی ایجاد شد.

در آن زمان مشخص شد که این مشکل منحصر به OpenID نبوده است. زمانی که دویت کلینتون از شرکت گوگل متوجه این موضوع شد علاقه خود را برای حمایت از این پروژه، حتی به‌عنوان یکی از سهام‌داران ابراز کرد. در ژوئیه ۲۰۰۷ (تیرماه ۱۳۸۶) تیم پیش‌نویس سند اولیه را تهیه کرد و عضویت در گروه برای علاقه‌مندان به مشارکت باز شد. در ۳ اکتبر ۲۰۰۷ (۱۱ مهر ۱۳۸۶) پیش‌نویس نهایی OAuth Core 1.0 منتشر شد.

در هفتاد و سومین جلسه کارگروه مهندسی اینترنت در مینیاپولیس در نوامبر ۲۰۰۸ (آبان ۱۳۸۷) یک کارگروه برای بحث در مورد استاندارد کردن پروتکل OAuth و ثبت آن در استانداردهای IETF شکل گرفت. این رویداد با مشارکت خوبی همراه شد و حمایت گسترده‌ای از OAuth صورت گرفت.

پروتکل OAuth 1.0 به‌عنوان یک استاندارد رسمی با کد RFC5849، در آوریل ۲۰۱۰ (فروردین ۱۳۸۹) منتشر شد و سرانجام تمام برنامه‌های کاربردی سوم‌شخص توییتر از تاریخ ۳۱ آگوست ۲۰۱۰ (۹ شهریور ۱۳۸۹) ملزم به استفاده از OAuth شدند.

دو سال پس از آن در اکتبر سال ۲۰۱۲ میلادی (مهر ۱۳۹۱) چارچوب OAuth 2.0 با درنظرگرفتن موارد استفاده بیشتر و الزامات توسعه‌پذیری با بررسی نظرات جامعه گسترده‌ای از اعضای IETF تحت عنوان RFC 6749 منتشر شد.

نقش‌ها

وقتی به هتلی مراجعه می‌کنیم، ابتدا نزد مسئول پذیرش می‌رویم و کارت شناسی خود را تحویل می‌دهیم. در مقابل مسئول پذیرش هتل کارت کلیدی در اختیار ما می‌گذارد که با واردکردن آن در کارت‌خوان ورودی می‌توانیم وارد اتاق خود شویم. این مشابه تبادلی است که در پروتکل OAuth انجام می‌شود. برای کارت کلید مهم نیست که ما چه کسی هستیم. قرار نیست کارت کلید شناسه کاربری منحصربه‌فرد یا نام ما را بداند. صرفاً دردست‌داشتن کارت کلید کافی است تا در اتاق باز شود و این امر هم با تصمیم و به اختیار میز پذیرش صورت می‌گیرد. اینجا، مسئول پذیرش مانند سرور مجوز عمل می‌کند و کارت کلید در نقش توکن دسترسی API یا سرور منبع است. این مثال نشان می‌دهد که یک API و یک سیستم با پروتکل OAuth می‌تواند بدون نیاز به دانستن هویت کاربر کار کند. البته ممکن است

متناسب با برخی کاربری‌ها لازم باشد که API از هویت کاربر مطلع شود. در این شرایط، OAuth کمکی نمی‌کند، بلکه برای احراز هویت کاربر به امکانی فراتر از OAuth و ابزارهای مانند OpenID connect نیاز خواهیم داشت.

OAuth در ابتدا برای برنامه شخص ثالث ایجاد شده بود. فیس‌بوک می‌خواست از طریق Google به مخاطبین شما دسترسی پیدا کند، اما به‌مرور همه چیز بالغ شد. حالا چارچوب OAuth برای برنامه‌های شخص اول نیز راه‌حلی بسیار خوب ارائه می‌دهد. هنگامی که به gmail.com مراجعه می‌کنید و روی ورود کلیک می‌کنید، Gmail از شما رمز عبور نمی‌خواهد، بلکه برای واردشدن (Login) شما را به حساب‌های Google's OAuth server مانند Google.com ارجاع می‌دهد.

این روش ورود به یک برنامه چند مزیت عمده دارد. برخی حساب‌های گوگل رمز عبور ندارند، چراکه راه‌های دیگری برای ایجاد یک حساب گوگل وجود دارد. یکی از این راه‌ها تفویض اختیار به یک ارائه‌دهنده هویت دیگر است. در این موقعیت، ابتدا آدرس ایمیل خود را وارد می‌کنید و سپس تعیین می‌کنید که چگونه وارد خواهید شد. اگر بخواهید تأیید دومرحله‌ای را انجام دهید، در ادامه گوگل رمز عبورتان را می‌خواهد و از شما در مورد دوفاکتوری‌شدن روند تأیید می‌پرسد. در واقع، با هر روشی که وارد گوگل شوید، سرورهای مجوز هویت کاربر را احراز می‌کنند و بعد از تأیید به Gmail باز می‌گردید و حالا Gmail به شما اجازه ورود می‌دهد. زیرا یک توکن دسترسی ایجاد شده و الان موجود است و این تنها چیزی است که Gmail برای صدور مجوز ورود به آن نیاز دارد.

تعاریف اولیه

در متن استاندارد OAuth مفاهیم و تعاریف به‌صورت دقیق و فنی تعریف شده‌اند. بااین‌حال نیاز بود تا برخی از تعاریف اولیه جدا از تعریف استانداری که در متن کتاب دارند به‌صورت جداگانه و با مثال‌های بیشتری توضیح داده شوند.

منبع حفاظت شده

اگر بخواهیم تعاریف OAuth را بررس می‌کنیم مهم‌ترین بخش همین منبع حفاظت‌شده است. منبع حفاظت‌شده اطلاعاتی است که کاربر از قبل با یک سرور ابری

به اشتراک گذاشته است. این منابع و دسترسی به آنها برای کاربران دارای اهمیت فراوانی هستند. منابع تنها محدود به عکس‌ها، فهرست تماس‌ها یا ایمیل‌های کاربر نیست. آنها می‌توانند شامل اطلاعات حساب بانکی، ده گردش آخر حساب یا اطلاعات مالی باشد. این اطلاعات توسط خود کاربر یا یک نرم‌افزار تولید شده است. محل نگهداری این اطلاعات از نظر کاربر امن است. برای مثال سرورهای گوگل وظیفه نگهداری اطلاعات تماس و عکس مخاطبان شما را دارند. شما از قبل این اطلاعات با یک سرور امن در اشتراک گذاشته‌اید.

مالکِ منبع یا کاربر

همان‌طور که پیش‌تر گفتیم مالکِ منبع یا کاربر، مالکِ منبع حفاظت شده است. مالکِ منبع همیشه یک کاربر یا انسان نیست. در برخی موارد یک شخصیت حقوقی می‌تواند مالک یک منبع باشد. برای مثال شهرداری یک شهر اطلاعات مربوط به یک قطعه زمین را در سامانه خود درج می‌کند و کاربری می‌تواند به این اطلاعات دسترسی پیدا کند. در این مثال مالکِ منبع یک سامانه تحت‌وب است. در مواردی که یک شخص مالکِ منبعی است به وی کاربر نهایی یا به‌صورت خلاصه کاربر می‌گوییم.

اعتبارنامه

اعتبارنامه چیزی است که نشان می‌دهد یک مالکِ منبع، صاحب حقیقی یک منبع حفاظت‌شده است. برای مثال گذرواژه شما در گوگل نشان‌دهنده مالکیت شما در یک حساب کاربری گوگل است. هرکسی که اعتبارنامه یک حساب گوگل را در اختیار داشته باشد می‌تواند به‌صورت مستقیم به منبع حفاظت‌شده دسترسی پیدا کند. در فرآیندهای OAuth اعتبارنامه کاربر با یک توکن تبادل می‌شود و این توکن امکانات امنیتی بیشتری نسبت به اعتبارنامه در اختیار کاربر قرار می‌دهد.

سرور منبع

سرور منبع محل نگهداری منابع حفاظت‌شده است. این سرور طوری تنظیم شده است که درخواست‌های دسترسی به منابع را می‌پذیرد آنها را بررسی می‌کند و در صورت معتبر بودن دسترسی منابع را در اختیار درخواست‌دهنده قرار می‌دهد.

کلاینت

کلاینت برنامه‌ای است که می‌خواهد با دریافت اجازه از مالکِ منبع به اطلاعات حفاظت شده وی دسترسی پیدا کند. کلاینت عموماً یک برنامه کاربردی است که یا نمی‌تواند یا نمی‌خواهد اطلاعات کاربر را به‌صورت مستقیم در اختیار بگیرد. برای مثال اینستاگرام می‌خواهد این اجازه را داشته باشد که عکسی که بارگذاری می‌کنید را به‌جای شما در فیس‌بوک هم منتشر کند. در اینجا کلاینت شما اینستاگرام است که می‌خواهد برای شما کاری (بارگذاری عکس روی فیس‌بوک به‌صورت هم‌زمان با اینستاگرام) انجام دهد. فیس‌بوک سرور منبع است که این بار می‌خواهد به‌صورت غیرمستقیم اطلاعاتی را برای شما ذخیره کند و در حساب کاربری شما نمایش دهد.

تفاوت OAuth با مدل‌های سنتی احراز هویت این است که نقش کلاینت را با مالکِ منبع یکی نمی‌داند. درست است که یک نرم‌افزار مانند اینستاگرام در کنترل شماست و با درخواست خود شما می‌تواند تصاویر شما را روی نمایه فیس‌بوک شما منتشر کند با این حال شما به‌عنوان مالکِ منبع از کلاینت خود جدا هستید و طبعاً میزان دسترسی کلاینت با شما برابری نمی‌کند.

دامنه دسترسی

در مثال اینستاگرام دیدیم که کاربر دسترسی ارسال پست در فیس‌بوک را به اینستاگرام کاربر داد. اما سؤال اینجاست که آیا اینستاگرام می‌تواند نام شما را در فیس‌بوک تغییر دهد؟ یا برای مثال می‌تواند کسی را به‌جای شما دنبال کند؟ برای پاسخ به این سؤال باید با مفهوم دامنه در OAuth آشنا شویم. دامنه این قابلیت را ایجاد می‌کند که نه‌تنها خود دسترسی مدیریت شده بلکه سطح دسترسی هم مدیریت شود. برای مثال اینستاگرام فقط بتواند پست‌های شما را در فیس‌بوک بیفزاید. اما این دامنه چگونه مدیریت می‌شود؟ سرور صدور مجوز می‌تواند این کار را به‌صورت کلی انجام دهد. برای مثال گوگل به‌هیچ‌عنوان نمی‌پذیرد کلاینتی بتواند گذرواژه شما را تغییر دهد و این دامنه دسترسی به کل از سمت سامانه‌های گوگل برای کلاینت‌ها لغو شده است.

مسیر دیگر این است که به‌تناسب میزان امنیت کلاینت می‌تواند سطح خاصی از

دسترسی‌ها را برای یکی از کلاینت‌ها به طور مجزا اعمال کند مثلاً سرورهای یک بانک تصمیم می‌گیرند به یک کلاینت دسترسی برداشت وجه از حساب‌های دارندگان حساب خود را بدهد و به یک کلاینت دیگر این دامنه دسترسی تعلق نگیرد. همچنین کاربر هم می‌تواند در زمان اعطای دسترسی از میان دامنه‌های ارائه‌شده از سوی سرور صدور مجوز انتخاب کند که کدام یکی از سطوح دسترسی را به کلاینت خود اعطا کند. برای مثال از میان دامنه‌های نوشتن روی تقویم کاربر، تغییر تصویر کاربری و دسترسی به عکس‌ها فقط دامنه نوشتن روی تقویم کاربر را به کلاینت خود که برای مثال یک سامانه رزرو بلیت آنلاین است، اعطا می‌کند.

سرور صدور مجوز

سرور صدور مجوز وظیفه بررسی صحت اطلاعات اعتبارنامه کاربر و تبدیل آن به یک توکن دسترسی را برعهده دارد. توکن دسترسی رشته‌ای است که سه امر مهم را کنترل می‌کند. یکی دسترسی کلی کلاینت به منابع حفاظت‌شده، یکی دامنه دسترسی و یکی دیگر مدت‌زمان این دسترسی. سرور صدور مجوز می‌تواند فرآیند تبادل اعتبارنامه اصلی کاربر به یک مجوز دسترسی کنترل شده یا همان توکن را با روش‌های مختلفی انجام دهد.

فرآیند پروتکل

باتوجه‌به میزان نیاز کلاینت‌ها، نوع منابع حفاظت‌شده، ملاحظات فنی و غیره روش‌های مختلفی برای تبادل اعتبارنامه کاربر با توکن دسترسی در استاندارد پیش‌بینی شده است. با این حال ضمن تعریف چند نمونه از پیش تعیین‌شده روشی توضیح داده شده است که توسعه‌دهندگان بر مبنای آن می‌توانند پروتکل موردنظر خود را بر مبنای استاندارد توسعه دهند. هرکدام از فرآیندها ویژگی‌های مخصوص به خود را دارند و در شرایط خاصی که امکان پیاده‌سازی بعضی از فرآیندها وجود ندارد و باتوجه‌به محدودیت‌ها می‌توان مدل‌های دیگر را جایگزین کرد.

حق امتیاز مجوز

در فرآیند تبادل اعتبارنامه با توکن دسترسی، سرور صدور مجوز یک حق امتیاز برای

کلاینت ایجاد می‌کند که کلاینت با برگرداندن این حق امتیاز به سرور صدور مجوز می‌تواند آن را با توکن دسترسی تعویض کند. این حق امتیاز پلی بین اعتبارنامه اصلی مالکِ منبع با توکن دسترسی است و به‌منظور ویژگی‌های امنیتی استاندارد باتوجه‌به مسائل فنی به کار می‌رود.

نماینده کاربر

نماینده کاربر عموماً یک صفحه مرورگر است که کارهای دسترسی از سمت کلاینت به سرور صدور مجوز را برای کاربر انجام می‌دهد. برای مثال زمانی که کاربر می‌خواهد یک دسترسی برای کلاینتی روی تقویم گوگل خود ایجاد کند این اتفاقات در یک صفحه مرورگر رخ می‌دهد.

این صفحه مرورگر همان نماینده کاربر است.

OAuth چگونه کار می‌کند؟

برای سردرآوردن از نحوه عملکرد OAuth باید روش کار آن را از نقطه‌نظر برنامه و همچنین API ببینیم. از نقطه‌نظر برنامه کاربردی هدف اصلی دریافت یک توکن دسترسی و دریافت کلیدی برای دسترسی به یک حساب شخصی است. همچنین، برای برنامه کاربردی مهم است که این کلید بر چه مبنایی است و در چه اپلیکیشنی بنا شده و در کجا عمل می‌کند؛ بنابراین، جریان‌های مختلف OAuth وجود دارد که هرکدام تعیین می‌کند که چگونه توکن دسترسی دریافت می‌شود.

فرآیندهای OAuth

همان‌طور که در بخش قبل گفتیم باتوجه‌به مسائل فنی در هر پروژه یک جریان پروتکل برای اجرای استاندارد OAuth انتخاب می‌شود. با اینکه توضیح دلایل انتخاب و برتری هرکدام از فرآیندها نسبت به یکدیگر خارج از حوصله فصل صفر است و در خود استاندارد به آن اشاره شده ولی در اینجا به‌اجمال فرآیندهای اصلی OAuth را توضیح می‌دهیم.

OAuth 2.0 چهار فرآیند اصلی تعریف می‌کند:

- کد مجوز؛
- مجوز ضمنی؛
- اعتبارنامه گذرواژه مالکِ منبع؛
- اعتبارنامه کلاینت.

کد مجوز

این نوع حق امتیاز مناسب‌ترین فرآیند برای برنامه‌های وب است. پس از اینکه مالکِ منبع اجازه دسترسی به اطلاعات خود را صادر کرد، برنامه یک کد مجوز دریافت می‌کند. این کد برای گرفتن توکن دسترسی توسط کلاینت مبادله می‌شود. این تغییر سرور به سرور انجام می‌شود و کلاینت نیاز دارد که از قبل در سرور صدور مجوز ثبت شده باشد. این نوع حق امتیاز امکان دسترسی طولانی‌مدت به یک API را فراهم می‌کند.

جریان کد مجوز برای زمانی مناسب است که دسترسی با مدت‌زمان زیاد مدنظر است و نیز کلاینت یک برنامه وب است.

مجوز ضمنی

حق امتیاز ضمنی ساده‌ترین نوع فرآیند است و برای کلاینت‌هایی که برنامه‌های کاربردی وب هستند و در یک مرورگر اجرا می‌شوند، مناسب است. در این فرآیند مالکِ منبع اجازه دسترسی به برنامه را صادر می‌کند و یک توکن دسترسی جدید ساخته و دریافت می‌شود. برنامه توکن دسترسی را استخراج می‌کند و سپس می‌توان با آن توکن درخواست‌های API خود را انجام دهد. این نوع حق امتیاز نیاز به‌واسطه ندارد ولی نمی‌توان از آن برای ارتباط طولانی‌مدت استفاده کرد.

این فرآیند زمانی مناسب است که به مدت‌زمان دسترسی موقتی نیاز است و کاربر به طور منظم در ارائه‌دهنده API لاگین می‌کند و کلاینت در مرورگر اجرا می‌شود، مرورگر به‌شدت قابل‌اعتماد است و نگرانی کمی از نشت توکن دسترسی به کاربران یا برنامه‌های غیرقابل‌اعتماد وجود دارد.

اعتبارنامه گذرواژه مالکِ منبع

در این فرآیند نام کاربری و گذرواژه مالکِ منبع با یک توکن دسترسی مبادله می‌شود. این فرآیند فقط برای کلاینت‌های کاملاً قابل‌اعتماد استفاده می‌شود. برای مثال شرکت گوگل به یکی از سرویس‌های خود دسترسی استفاده از بخشی از اطلاعات کاربر را می‌دهد و به‌جای ذخیره کردن نام کاربری و گذرواژه خود کاربر آن را با یک توکن دسترسی مبادله می‌کند. در این حال با اینکه کلاینت می‌تواند گذرواژه و نام کاربری را مشاهده کند اقدام به ذخیره‌سازی آن نمی‌کند و از طرفی کاربر بدون تغییر گذرواژه می‌تواند دسترسی برنامه را لغو کند، همچنین دامنه دسترسی قابل مدیریت است.

در چه شرایطی می‌توان از اعتبارنامه گذرواژه مالکِ منبع استفاده کرد:

ازآنجایی‌که رمز عبور مالکِ منبع در معرض دسترسی کلاینت قرار می‌گیرد این روش فقط برای برنامه‌های «رسمی» شخص اول که توسط ارائه‌دهنده API منتشرشده‌اند، توصیه می‌شود. مثلاً جیمیل می‌خواهد از تقویم گوگل دسترسی بگیرد.

اعتبارنامه کلاینت

زمانی که کلاینت بخواهد به یکی از منابع تحت مالکیت خود به‌صورت امن دستیابی پیدا کند این فرآیند پیشنهاد می‌شود. این نوع حق امتیاز برای برنامه‌هایی که نیاز به دسترسی به APIها از طرف خودشان دارند، مناسب است (مانند سرویس‌های ذخیره‌سازی یا پایگاه‌های داده). اما نکته مهم این است که نتیجه نهایی همه این جریان‌ها همیشه یکسان است. در همه موارد یک توکن دسترسی وجود دارد و تا آنجا که به برنامه مربوط می‌شود، توکن دسترسی درست مانند کارت کلید هتل فقط حاوی اطلاعات لازم جهت دسترسی است. در استفاده از کلید هتل چه از نوع RFID یا NFC و چه کلید فیزیکی محتوای رمز درج شده بر کلید برای ما مهم نیست و فقط می‌خواهیم با استفاده از کلید در اتاق هتل را باز کنیم و به اتاق دسترسی پیدا کنیم. در پروتکل OAuth پنج نقش وجود دارد که هر یک در متن اصلی استاندارد اصطلاح ویژه‌ای دارد. مثلاً استاندارد کاربر[1] را صاحب منبع[2] می‌نامد.

1. The User
2. Resource Owner

معادل‌های رایج و آشناتر این نقش‌ها در استاندارد عبارت‌اند از:

۱. صاحب منبع به‌جای کاربر؛

۲. عامل کاربر[1] به‌جای دستگاه[2]؛

۳. کلاینت[3] به‌جای برنامه کاربردی[4]؛

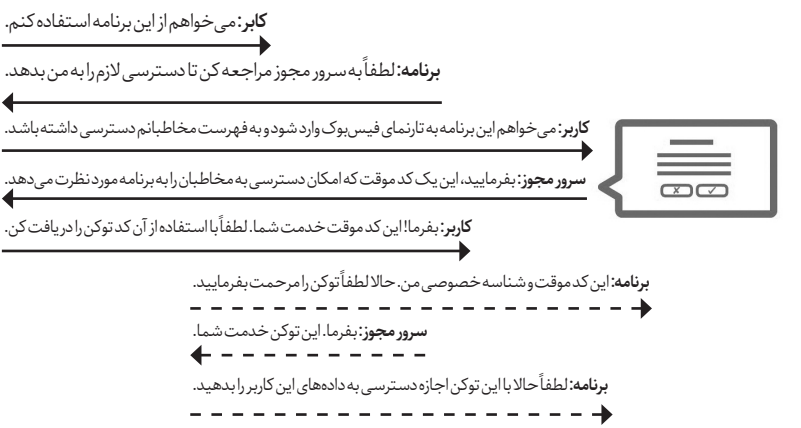
۴. سرور مجوز[5] یا سازنده توکن[6] به‌جای سرور OAuth؛

۵. سرور منبع[7] به‌جای API.

در ادامه برای نشان‌دادن ارتباط میان این نقش‌ها به معرفی دیگر مفاهیم استاندارد می‌پردازیم:

جریان کد مجوز:

از میان جریان‌های موجود، با جریان کد مجوز شروع می‌کنیم که ساده‌ترین جریان و درعین‌حال پایه جریان‌های OAuth است.



شکل ۴. جریان‌های چارچوب مجوز OAuth برای دسترسی برنامه شخص ثالث

1. User Agent
2. Device
3. Client
4. The Application
5. Authorization Server
6. The Token Factory
7. Resource Server

در شـکل ۴ می‌بینیم که ابتدا کاربـر در مرورگر خود از وب‌سـایتی بازدید می‌کنـد و روی دکمه ورود کلیک می‌کند. حالا درخواست کاربر برای ورود ارسال شده است. برنامه پاسخ می‌دهد: «شما می‌توانید وارد شوید، اما من نمی‌دانم چگونه این کار را انجام دهم. می‌توانید به صفحه دیگری بروید و از آنجا وارد شوید.» اینجا کاربر به سمت سرور OAuth هدایت می‌شود. چیزهایی نظیر تأیید صحت شناسه مشتری (Client ID) در آن برنامه، محدوده و شرح آنچه (مثلاً فهرست مخاطبین) قرار است به آن دسترسی داده شود، در این تغییر مسـیر وجود دارد. سـرور OAuth از کاربر می‌خواهد در صفحه تعیین‌شـده وارد شود. در این مرحله سـرور OAuth کد موقتی (کد مجوز) تحویل می‌دهد. ایـن کد، یک‌بار مصرف و برای مدت‌زمان محدودی معتبر است. مرورگر مجدد به صفحه ورود به وب‌سایت اصلی برمی‌گردد. وب‌سایت که ارائه‌دهنده خدمات برنامه خاص است، کد موقت را تحویل می‌گیرد و توکن دسترسـی را دریافت می‌کند. بـه این منظور، در پشـت‌صحنه برنامه کـد موقت و شناسه مشتری را به سرور OAuth تحویل می‌دهد و درخواست صحت‌سنجی و دریافت توکن دسترسی می‌کند. سرور OAuth توکن دسترسی را به برنامه تحت وب ارائه می‌کند؛ بنابراین، حالا یک توکن دسترسی وجود دارد و می‌تواند درخواست‌های API را انجام دهد.

همان‌طور که در شکل ۴ نشان‌داده‌شده است، در جریان کد مجوز، اطلاعات به دو روش ارسال می‌شود که با بهره‌گیری از هر دو فرآیند امن‌ترین جریان را برای به‌دست‌آوردن توکن دسترسی فراهم می‌کند. جریان‌هایی را که در پنج پیکان نخست انتقال پیدا می‌کنند، «کانال جلویی[1]» شامل جریان‌هایی است که بین مرورگر و سرورها (پنج پیکان نخست در شکل ۴) انتقال پیدا می‌کنند و کاربر به آن دسترسی دارد. قسمت بعدی این جریان بین سرور مجوز و کلاینت اتفـاق می‌افتد و کاربـر به آن دسترسـی ندارد. این جریان (سـه پیـکان بعدی در شکل ۴) را «کانال پشتی[2]» می‌خوانیم.

کانال پشـتی چند ویژگی واقعاً مهـم دارد که کانال جلویی فاقد آنهاسـت. با اسـتفاده از کانال جلویی به معنـای واقعی کلمه از نوار آدرس برای ارسـال داده اسـتفاده می‌کنیم. همچنین کانال پشتی یک کانال ارتباطی امن برای ارتباط کلاینت به سرور فراهم می‌کند.

1. Front Channel
2. Back Channel

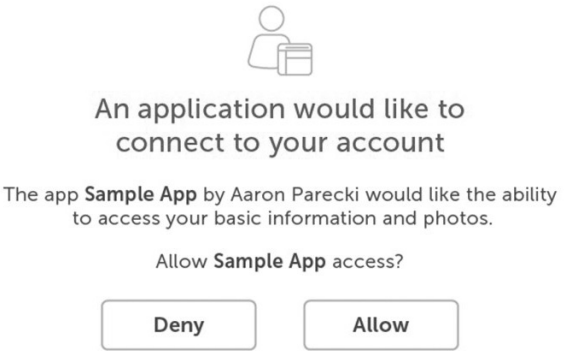
دلیل امنیت بالای این کانال این است که عملاً کاربر به آن دسترسی ندارد. همچنین، هنگام ارسال کد از کاربر به کلاینت برای دریافت توکن دسترسی از سرور در کانال جلویی این کد دزدیده شود، سارق با استفاده از کد سرقت‌شده نمی‌تواند به جایی دست پیدا کند. چون به همراه کد کلاینت باید شناسه کلاینت و شناسه خصوصی نیز برای سرور مجوز ارسال شود. اگر کد به همراه شناسه کلاینت دیگری ارسال شود، «سرور مجوز» بلافاصله متوجه این موضوع می‌شود و از سرقت توکن دسترسی جلوگیری می‌کند. مانند اینکه پیامی را با دست تحویل دهید. وقتی خودتان یک پیام را تحویل بدهید، می‌توانید ببینید که طرف مقابل آن را می‌گیرد و می‌پذیرد. طرف مقابل هم می‌تواند تأیید کند که شما چه کسی هستید و همه چیز درست پیش می‌رود. بنابراین، مزیت مهم استفاده از کانال پشتی این است که وقتی کلاینت از طریق اتصال HTTP با سرور ارتباط برقرار می‌کند، این موضوع تأیید و سپس اتصال رمزگذاری می‌شود؛ یعنی نمی‌توان آن را دست‌کاری کرد و پاسخ بازگشتی بخشی از همان اتصال است؛ بنابراین، کلاینت می‌تواند به آن اعتماد کند.

ولی انتقال داده ازطریق کانال جلویی مانند پرتاب‌کردن داده به آن‌سوی دیوار است. امیدواریم کسی داده را از طرف دیگر بگیرد و هیچ‌کدام از طرفین نمی‌توانند روی دیوار را ببینند. در واقع، سرور OAuth سعی می‌کند این کد مجوز را به برنامه برساند و فقط آن را روی دیوار می‌اندازد و نمی‌تواند تشخیص دهد که آیا برنامه‌ها واقعاً آن را دریافت کرده‌اند یا فردی ناشناس در این میان آن را دزدیده است. این مورد بسیار مهمی در مورداستفاده از کانال جلویی در OAuth است که هیچ‌یک از طرفین نمی‌توانند واقعاً مطمئن باشند که عملیات موفقیت‌آمیز بوده است. ممکن است بگویید پس چرا از کانال جلویی استفاده می‌کنیم، درحالی‌که می‌دانیم این شیوه ناامن است؟ واقعیت این است که هر دو کانال‌ها به نوبه خود مفیدند. کانال جلو هم چند مزیت واقعاً مهم دارد که شامل نحوه تعامل با کاربر و احراز هویت چندعاملی که ما را مطمئن می‌کند که کاربر آنجا بوده و مجوز انجام آن کار را داده است و همچنین گیرنده این کار را انجام نداده است.

مثالی را گام‌به‌گام مرور کنیم. با برنامه‌ای شروع می‌کنیم که لینک لازم برای لاگین را می‌سازد. بخش اول آدرس سرور مجوز است. سپس یک‌رشته کوئری وجود دارد که محدوده درخواست دسترسی را مشخص می‌کند و بدین شرح است

ساختار لینک:	https://authorization-server.com/auth
نشان‌دهنده نوعی پاسخ است که برنامه دریافت خواهد کرد، در این مثال «کد مجوز».	response_type=code:
نشان‌دهنده شناسه کلاینتی است که پس از ایجاد برنامه دریافت می‌کنیم.	client_id=CLIENT_ID
نشانگر نشانی‌ای است که کاربر پس از صدور مجوز به آن برمی‌گردد، مثلاً: htpps://example-app.com/callback	redirect_uri=REDIRECT_URI
نشان‌دهنده آن بخشی از اطلاعات موجود در حساب کاربر است که برنامه می‌خواهد به آن دسترسی داشته باشد، در این مثال «فهرست مخاطبین».	Scope=contacts
پارامتر State شامل یک رشته تصادفی است که در برنامه تولید می‌شود و ما بعداً آنها صحت‌سنجی می‌کنیم	State:1234zyx

کاربر بر روی دکمه ورود کلیک می‌کند و در مرحله بعد دسترسی برنامه به اطلاعات خود را تأیید می‌کند.



شکل ۶. تأیید دسترسی به حساب کاربری از طریق برنامه شخص ثالث

سپس با کد مجوز به برنامه هدایت می‌شود. توجه کنید که همه اینها در بخش کانال جلویی اتفاق می‌افتد. حالا درخواست به برنامه با یک کد مجوز تغییر مسیر می‌دهد. در

این حالت، در واقع برنامه کد مجوز را از بالای دیوار دریافت کرده است و برنامه می‌بایست مطمئن شود که کد مجوز دریافتی از یک سرور OAuth واقعی تولید و دریافت شده است. این به برنامه اطمینان می‌دهد که کد دریافتی حاصل یک حمله تبادل اطلاعات نیست. در اینجا، برنامه از طریق کانال پشتی کد مجوز دریافتی را با یک توکن دسترسی مبادله می‌کند. در این راستا برنامه درخواستی خطاب به توکن در سرور OAuth ارسال می‌کند که ساختار آن بدین صورت است:

ساختار آدرس:	https://api.authorization-server.com/token
نشان می‌دهد که این درخواست شامل یک کد مجوز است.	Grant_type=authorization_code
شامل کد مجوز در رشته کوئری است.	Code=CODE_FROM_QUERY
این نشانی باید با نشانی مندرج در درخواست اصلی انطباق داشته باشد.	Redirect_uri=REDIRECT_URI
نشانگر شناسه کلاینتی است که در ابتدا برنامه دریافت کرده است.	Client_id=CLIENT_ID
باتوجه‌به اینکه این درخواست از سمت سرور ارائه شده شامل رمز کلاینت است.	Client_secret=CLIENT_SECRET

درخواست صحت‌سنجی کد مجوز موقت دریافتی از طریق کانال پشتی، برای سرور OAuth ارسال کند. این کار با استفاده از بررسی صحت پارامتر State انجام می‌شود.

بنابراین، برنامه یک درخواست را به سرور منبع ارسال می‌کند، درخواست به سرور OAuth برمی‌گردد و می‌گوید: «این کدی است که از کاربر دریافت کرده‌ام» و شناسه کلاینت و شناسه خصوصی کلاینت[1] و URI تغییر مسیر را ارسال می‌کند. این روشی است که سرور OAuth حلقه را به‌نوعی می‌بندد و اطمینان می‌دهد که کد مجوز به سرقت نرفته است. چراکه اگر کد دزدیده می‌شد، مهاجم سارق نمی‌توانست به شناسه خصوصی کلاینت دست‌یافته باشد. این تصویری کلی از جریان‌های OAuth در کلانیت‌هایی محرمانه بود که می‌توانند کد مجوز را محرمانه نگه دارند.

1. Client secret

```
POST https://api.authorization-server.com/token
Content-type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=AUTH_CODE_HERE&
redirect_uri=REDIRECT_URI&
client_id=CLIENT_ID&
client_secret=CLIENT_SECRET
```

تا اینجا از دیدگاه در مورد OAuth صحبت کردیم در فصول بعدی چارچوب استاندارد OAuth را از نظر می‌گذرانیم.

آنچه در این کتاب می‌خوانید:

فصل اول: در بخش اول با تاریخچه شکل‌گیری و کلیات تعاریف و فرآیندهای OAuth آشنا می‌شویم. پس از آن انواع حق امتیازها را بررسی می‌کنیم. با انواع توکن آشنا خواهیم شد و درباره امنیت لایه انتقال بحث خواهیم کرد. همچنین درباره نحوه نگارش کتاب و ضرورت‌های اجرایی اطلاعاتی کسب خواهیم کرد.

فصل دوم: در فرآیندهای OAuth نیاز است تا کلاینت در سرور صدور مجوز ثبت شود. در فصل دو به جزئیات این ثبت شدن می‌پردازیم. انواع کلاینت‌های تعریف شده در OAuth را خواهیم شناخت و نحوه احراز هویت کلاینت را مرور خواهیم کرد.

فصل سوم: فصل سوم مختص بررسی نقاط پایانی است. اینکه بعد از انجام هر فرآیند کار در کجا به اتمام می‌رسد. نقاط پایانی صدور مجوز بررسی می‌شود، نوع پاسخ‌ها و تغییر مسیرها شرح داده می‌شود و در خصوص امنیت و محرمانگی بحث می‌شود. بعد از آن در خصوص مبحث نقاط پایانی توکن‌ها و دامنه دسترسی نیز شرح و بسط داده می‌شود.

فصل چهارم: در فصل چهارم به دریافت مجوز می‌پردازیم. در ابتدا با حق امتیاز کد مجوز آشنا می‌شویم، گام‌های آن را بررسی می‌کنیم و نمونه کدهایی در زمینه درخواست و پاسخ‌ها بررسی می‌کنیم، سپس نحوه دریافت توکن دسترسی را مرور خواهیم کرد. این امر برای سایر حق امتیازها مانند مجوز ضمنی نیز طی می‌شود.

فصل پنجم: صدور انواع توکن دسترسی موضوع اصلی فصل پنجم است. ابتدا به بررسی

پاسخ‌های موفق می‌پردازیم و سپس خطاها را بررسی می‌کنیم.

فصل ششم: فصل ششم مانند فصل قبل به توکن‌ها می‌پردازیم با این اختلاف که در این فصل توکن تازه‌سازی را بررسی می‌کنیم.

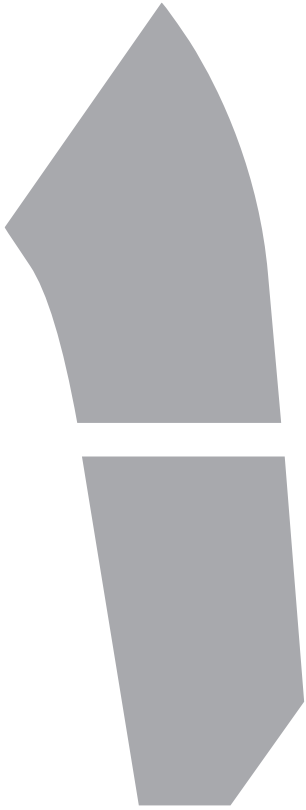
فصل هفتم: نحوه دسترسی به منابع محافظت‌شده را در فصل هفتم بررسی می‌کنیم.

فصل هشتم: همان‌طور که پیش‌تر گفته بودیم استاندارد روش‌هایی برای گسترش پروتکل‌ها پیشنهاد می‌دهد. در فصل هشت به این امر می‌پردازیم که برای تعریف موارد اختصاصی خودمان مانند نوع توکن جدید، نقاط نهایی بیشتر و کدهای خطای اضافی باید چه مواردی را مدنظر قرار دهیم.

فصل نهم: فصل نهم مربوط به ملاحظات برنامه‌های بومی است. برنامه‌هایی که بر روی دستگاه‌های مالکِ منبع نصب و اجرا می‌شوند. این برنامه‌ها باید ملاحظات امنیتی ویژه‌ای را مدنظر داشته باشند.

فصل دهم: اما این فصل در خصوص امنیت بیشتر است. طی بررسی‌های انجام‌شده در سال‌های گذشته حفره‌های امنیتی در OAuth کشف شد و برای حل آنها تغییراتی در استاندارد لحاظ شد و همچنین ملاحظات امنیتی آن به‌روز شدند. هرکدام از بخش‌ها مانند توکن‌های دسترسی، کدهای مجوز و غیره نیاز به اعمال تنظیمات یا محدودیت‌هایی دارند تا امنیت سامانه‌ها تأمین شود فصل دهم به این امر می‌پردازد.

فصل یازدهم: اگر بخواهیم OAuth را توسعه دهیم نیاز به تعریف انواع جدیدی از توکن‌های یا پارامترهای اضافی است. برای اینکه از چارچوب استاندارد خارج نشویم نیاز است که نقشه راهی پیشروی خود داشته باشیم. فصل یازدهم به این نکاتی در خصوص توسعه OAuth می‌پردازد.



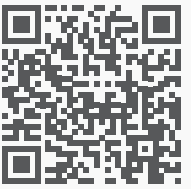
فصل اول

چارچوب صدور مجوز OAuth2.0

چارچوب صدور مجوز OAuth 2.0 برنامه‌های شخص‌ثالث[1] را قادر می‌سازد به یک سرویس HTTP دسترسی محدود شده داشته باشند. این دسترسی از طرف مالک منبع[2] و با تنظیم یک فرآیند تأیید بین مالک منبع و سرویس HTTP، یا اعطای اجازه به یک برنامه شخص سوم از طرف وی فراهم می‌شود. پروتکل OAuth 1.0 تشریح شده در RFC 5849، جایگزین و منسوخ می‌کند.

برای دسترسی به استاندارد OAuth 1.0 RFC 5849 کد را اسکن کنید.

این یک سند internet standard track یا مسیر استاندارد اینترنت[3] است. این سند محصول کارگروه مهندسی اینترنت (IETF[4]) است و همچنین مورد اتفاق‌نظر مجمع IETF قرار گرفته‌است. همچین گروه راهبری مهندسی اینترنت (IESG[5]) این سند را بازبینی و تأیید کرده‌است. جهت کسب اطلاعات بیشتر بخش ۲ استاندارد RFC 5741 را مطالعه نمایید.



برای دسترسی به استاندارد RFC 5741 کد را اسکن کنید.

اطلاعات مربوط به وضعیت فعلی این سند، فهرست اشتباهات[6] و نحوه ارائه بازخورد را

1 Third Party Application
2 Resource server
۳ فرآیند فعلی IETF دارای دو نوع RFC است: اسناد مسیر استاندارد و سایر RFCها (به عنوان مثال، اطلاعاتی و تجربی). مفاد مسیر ردیابی استانداردها واضح است و در یک استاندارد رسمی اینترنتی به اوج خود می رسد.
4 Internet Engineering Task Force
5 Internet Engineering Steering Group
۶ لازم به ذکر است که برخی از اشتباهات گزارش شده توسط IETF تأیید و در غلط‌نامه تصحیح شده است. بااین‌حال این اشتباهات هنوز در متن استاندارد ثبت نشده‌اند. ما در این سند، متن تصحیحات انجام شده را لحاظ کرده‌ایم.

می‌توانید از RFC 6749 دریافت کنید.

برای دسترسی به آدرس RFC 6749 کد را اسکن کنید.

در مدل‌های سنتی احراز هویت[1] کلاینت-سروری[2]، کلاینت[3] با استفاده از اعتبارنامه مالک منبع[4] و انجام احراز هویت توسط سرور، دسترسی به یک منبع دارای محدودیت (منبع حفاظت‌شده) را درخواست می‌کند. به‌منظور فراهم ساختن دسترسی برنامه‌های شخص‌ثالث به منابع دارای محدودیت، مالک منبع اعتبارنامه خود را با شخص‌ثالث به اشتراک می‌گذارد. این روش، سبب بروز مشکل و محدودیت‌هایی می‌شود:

- برنامه‌های شخص‌ثالث برای استفاده‌های بعدی از منبع ملزم به ذخیره مجوز مالک منبع هستند و معمولاً گذرواژه به‌صورت متن رمزگذاری‌نشده ذخیره می‌شود.
- با وجود ضعف‌های امنیتی ذاتی گذرواژه‌ها، سرورها ملزم به پشتیبانی از احراز هویت توسط گذرواژه‌ها هستند.
- برنامه‌های شخص‌ثالث دسترسی بیش‌ازحد وسیعی به منابع حفاظت‌شده مالک منبع می‌یابند و مالکان منابع، امکان محدودسازی مدت‌زمان یا دامنه دسترسی به منابع را ندارند.
- مالکان منابع نمی‌توانند بدون سلب دسترسی همه اشخاص، دسترسی یکی از اشخاص ثالث را سلب کنند و باید این کار را با تغییر گذرواژه خود انجام دهند.
- به خطر افتادن هرکدام از برنامه‌های شخص‌ثالث باعث به خطر افتادن گذرواژه

1 Authentication
2 Client-Server
3 Client
4 Resource owner

کاربر نهایی[1] و همه داده‌های حفاظت‌شده توسط آن گذرواژه می‌شود.

OAuth با معرفی یک لایه صدور مجوز و تفکیک نقش کلاینت از نقش مالک منبع، این مشکلات را حل‌وفصل می‌کند. در OAuth، کلاینت درخواست دسترسی به منابع در مالکیت مالک منبع و میزبانی‌شده در سرور منبع را دارد و مجموعه‌ای متفاوت از اعتبارنامه نسبت به اعتبارنامه‌ی شخص مالک منبع صادر می‌شود.

به جای استفاده از اعتبارنامه مالک منبع برای دستیابی به منابع محافظت‌شده، کلاینت یک توکن دسترسی[2] در اختیار می‌گیرد. توکن دسترسی، رشته‌ای[3] است که دسترسی به یک دامنه خاص با طول عمر[4] مشخص و سایر مشخصه‌های دسترسی را فراهم می‌کند. توکن‌های دسترسی برای کلاینت‌های شخص سوم با تأیید مالک منبع در سرور صدور مجوز صادر می‌شوند. کلاینت از توکن دسترسی به منظور دسترسی به منابع حفاظت شده سرور منبع استفاده می‌کند. برای مثال کاربر نهایی (مالک منبع) سایت چاپ عکس (کلاینت)، می‌تواند بدون به اشتراک گذاشتن نام‌کاربری و گذرواژه پروفایل خود، دسترسی مشاهده عکس های حفاظت شده خود در یک سرور اشتراکی (سرور منبع) را به سایت دهد. در این روش کاربر با احراز هویت سرور سایت، مجوز اختصاصی سرویس چاپ عکس را صادر می‌کند. این سند برای استفاده با پروتکل HTTP تحت عنوان RFC 2616 طراحی شده است. استفاده از OAuth برای هر پروتکلی غیر از HTTP خارج از اهداف این سند است.

برای دسترسی به استاندارد RFC 2616 کد را اسکن کنید.

پروتکل OAuth1.0 براساس نظرات تعداد کمی از خبرگان این حوزه تدوین شده

1 End-user
2 Access tokens
3 String
4 Lifetime

بود. در حالی که OAuth 2.0 بر اساس تجربه حاصل از تدوین OAuth 1.0 و نظرات جمع‌آوری‌شده از IETF است. پروتکل OAuth 2.0 با OAuth 1.0 سازگار نیست. بااین‌حال ممکن است این دو نسخه هم‌زمان در شبکه وجود داشته باشند و امکان پیاده‌سازی این استانداردها در یک شبکه به‌صورت هم‌زمان میسر باشد.

برای دسترسی به استاندارد OAuth 1.0 RFC 5849 کد را اسکن کنید.

نقش‌ها

OAuth چهار نقش تعریف می‌کند:

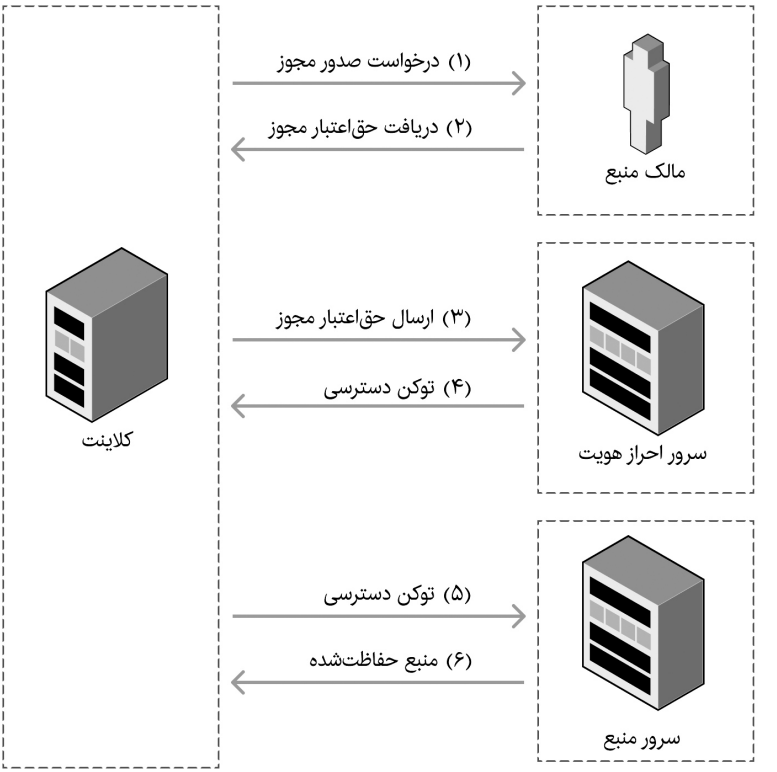
مالک منبع	موجودیتی که اجازه اعطای دسترسی به اطلاعات منبع حفاظت‌شده را می‌دهد. وقتی مالک منبع یک شخص باشد، از وی به‌عنوان کاربر نهایی یاد می‌شود.
سرور منبع	سروری که منابع حفاظت‌شده را میزبانی می‌کند، قادر به پذیرش و پاسخگویی به درخواست‌های منابع حفاظت‌شده با استفاده از توکن‌های دسترسی است.
کلاینت	برنامه‌ای که از طرف مالک منبع و با مجوز وی، دسترسی به اطلاعات منبع حفاظت‌شده را درخواست می‌کند. اصطلاح «کلاینت» به هیچ مدل پیاده‌سازی خاصی اشاره نمی‌کند. (مثلاً این‌که آیا برنامه بر روی سرور، کامپیوتر شخصی یا سایر دستگاه‌ها اجرا می‌شود.)
سرور صدور مجوز	توکن‌های دسترسی را پس از احراز هویت موفق مالک منبع و دریافت مجوز آن، برای کلاینت صادر می‌کند.

تعامل بین سرور صدور مجوز و سرور منبع خارج از این مبحث این سند است. سرور صدور مجوز ممکن است همان سرور منبع یا موجودیتی جداگانه باشد. یک سرور صدور

مجوز یکتا می‌تواند توکن‌های دسترسی‌ای صادر کند که توسط چندین سرور منبع پذیرفته شوند.

فرآیندپروتکل

خلاصه فرآیند OAuth 2.0 که در شـکل ۱ نشان داده شـده، تعامل بین چهار نقش را توصیف می‌کند و شامل مراحل زیر است:



شکل شماره ۱- خلاصه فرآیند OAuth

- کلاینت از مالک منبع درخواست مجوز می‌کند. درخواست صدور مجوز می‌تواند

مستقیماً به مالک منبع ارسال شود (همان‌طور که در شکل هم نشان داده شده است) یا ترجیحاً به‌صورت غیرمستقیم از طریق سرور صدور مجوز به‌عنوان واسطه ارسال شود.

- کلاینت حق‌امتیاز مجوز[1] را دریافت می‌کند که نشانگر اعتبار مجوز مالک منبع است. همچنین نشان می دهد که چه حق امتیازی استفاده شده است؛ چه یکی از چهار نوع حق امتیاز معرفی شده در این سند یا نوعی دیگر. نوع حق‌امتیاز مجوز به روش مورداستفاده کلاینت برای درخواست مجوز و انواع تحت پشتیبانی سرور صدور مجوز بستگی دارد.
- کلاینت با احرازهویت توسط سرور صدور مجوز و ارائه حق‌امتیاز مجوز، توکن دسترسی را درخواست می‌کند.
- سرور صدور مجوز، کلاینت را احراز هویت و اعتبار حق‌امتیاز مجوز را بررسی می‌کند و در صورت معتبر بودن، یک توکن دسترسی صادر می‌کند.
- کلاینت، منبع حفاظت‌شده[2] را از سرور منبع درخواست می‌کند و با ارائه توکن دسترسی، احراز هویت می‌شود.
- سرور منبع، توکن دسترسی را اعتبارسنجی می‌کند و در صورت معتبر بودن آن، درخواست می‌دهد.

روش بهتر جهت دریافت حق‌امتیاز مجوز از مالک منبع، برای کلاینت این است که (نشان‌داده‌شده در مراحل (۱) و (۲))، از سرور صدور مجوز به‌عنوان واسطه استفاده کند که شرح آن در شکل ۳ در فصل چهارم نشان داده شده است.

حق‌امتیازمجوز

حق‌امتیاز مجوز اعتبارنامه‌ای است که نشانگر مجوز مالک منبع (برای دسترسی به منابع حفاظت‌شده وی) است که توسط کلاینت برای دریافت توکن دسترسی استفاده می‌شود. در این سند، چهار نوع مجوز آمده است، همچنین سازوکاری برای توسعه انواع

1 Authorization grant
2 Protected resource

دیگر آن ارائه می‌شود. این چهار مجوز عبارت‌اند از:

- کد مجوز[1]؛
- مجوز ضمنی[2]؛
- اعتبارنامه گذرواژه مالک منبع[3]؛
- اعتبارنامه کلاینت.

کد مجوز

کد مجوز با استفاده از سرور صدور مجوز به‌عنوان واسطه بین کلاینت و مالک منبع به دست می‌آید. کلاینت به‌جای درخواست مستقیم مجوز از مالک منبع، وی را به سرور صدور مجوز هدایت می‌کند (از طریق نماینده کاربر[4] که در استاندارد RFC 2616 تعریف‌شده)، که در برگشت مالک منبع را به همراه کد مجوز به کلاینت هدایت می‌کند.

برای دسترسی به استاندارد RFC 2616 کد را اسکن کنید.

قبل از هدایت مالک منبع (همراه کد مجوز) به کلاینت، سرور صدور مجوز، مالک منبع را احراز هویت و مجوز را دریافت می‌کند. ازآنجاکه مالک منبع تنها توسط سرور صدور مجوز احراز هویت می‌شود، اعتبارنامه مالک منبع هرگز با کلاینت به اشتراک گذاشته نمی‌شود. کد مجوز چند مزیت امنیتی مهم دارد، مانند امکان احراز هویت کلاینت، ارسال مستقیم توکن دسترسی به کلاینت بدون ارسال آن از طریق نماینده کاربر مالک منبع جلوگیری از افشای آن برای دیگران، از جمله مالک منبع.

1 Authorization code
2 Implicit
3 Resource owner password credentials
4 User agent

مجوز ضمنی

مجوز ضمنی، فرآیند ساده «کد مجوز»، برای آن دسته از کلاینت‌هایی است که در مرورگر با استفاده از یک‌زبان اسکریپت‌نویسی مانند جاوا اسکریپت نوشته شده‌اند. در فرآیند مجوز ضمنی، به‌جای صدور کد مجوز برای کلاینت، وی مستقیماً یک توکن دسترسی (باتوجه‌به مجوز مالک منبع) دریافت می‌کند. نوع مجوز، ضمنی است، زیرا هیچ اعتبارنامه واسطی (مانند کد مجوز) صادر نمی‌شود (که بعداً برای دریافت توکن دسترسی استفاده شود).

هنگام صدور یک توکن دسترسی در فرآیند حق‌امتیاز مجوز ضمنی، سرور صدور مجوز کلاینت را احراز هویت نمی‌کند. در برخی موارد هویت کلاینت را می‌توان از طریق URI[1] تغییر مسیر استفاده شده برای تحویل توکن دسترسی به کلاینت تأیید کرد. توکن دسترسی ممکن است برای مالک منبع یا سایر برنامه‌های دارای دسترسی به نماینده کاربر مالک منبع افشا شود.

مجوزهای ضمنی پاسخگویی و کارایی برخی کلاینت‌ها را بهبود می‌بخشند (مانند کلاینتی که به‌عنوان یک برنامه تحت مرورگر پیاده‌سازی شده)، زیرا تعداد مراحل رفت‌وبرگشت لازم برای دریافت توکن دسترسی را کاهش می‌دهد. بااین‌حال، این راحتی باید در برابر پیامدهای امنیتی استفاده از مجوزهای غیرمستقیم، مانند موارد تشریح شده در فصل ده (بخش توکن دسترسی و بخش سواستفاده از توکن دسترسی برای جعل هویت مالک منبع در مجوز ضمنی)، سنجیده شود؛ مخصوصاً زمانی که نوع حق‌امتیاز کد مجوز در دسترس باشد.

اعتبارنامه گذرواژه مالک منبع

اعتبارنامه گذرواژه مالک منبع (به طور مثال نام‌کاربری و گذرواژه) می‌تواند مستقیماً به‌عنوان حق‌امتیاز برای دریافت توکن دسترسی استفاده شود. این اعتبارنامه فقط درصورتی که بین مالک منبع و کلاینت اعتماد بالایی وجود داشته باشد (مثلاً کلاینت بخشی از سیستم‌عامل دستگاه یا یک برنامه بسیار معتبر است) یا زمانی که سایر

1 Uniform Resource Identifier

انواع حق‌امتیاز مجوز در دسترس نباشند (مانند کد مجوز) باید استفاده شود. هرچند این نوع مجوز مستلزم دسترسی مستقیم کلاینت به اعتبارنامه مالک منبع است، اعتبارنامه مالک منبع برای یک درخواست استفاده می‌شود و با یک توکن دسترسی مبادله می‌شود. این نوع حق‌امتیاز می‌تواند با تبادل اعتبارنامه با یک توکن دسترسی با عمر طولانی یا توکن تازه‌سازی[1]، نیاز کلاینت به ذخیره اعتبارنامه مالک منبع برای استفاده آتی را برطرف سازد.

اعتبارنامه کلاینت

زمانی که دامنه مجوز محدود به منابع حفاظت‌شده تحت کنترل کلاینت یا منابع حفاظت‌شده‌ای باشد که قبلاً با سرور صدور مجوز هماهنگ شده، می‌توان از اعتبارنامه کلاینت (یا سایر اشکال احراز هویت کلاینت) به‌عنوان حق‌امتیاز مجوز استفاده کرد. اعتبارنامه کلاینت معمولاً زمانی به‌عنوان حق‌امتیاز مجوز استفاده می‌شود که کلاینت از جانب خود عمل می‌کند (کلاینت نیز مالک منبع است) یا بر اساس احراز هویتی که قبلاً با سرور صدور مجوز هماهنگ شده است دسترسی به منابع حفاظت‌شده درخواست می‌کند.

توکن دسترسی

توکن‌های دسترسی اعتبارنامه‌هایی هستند که برای دسترسی به منابع حفاظت‌شده استفاده می‌شوند. توکن دسترسی رشته‌ای است که نشانگر مجوز صادرشده برای کلاینت است. این رشته معمولاً برای کلاینت مبهم است. توکن‌ها محدوده[2] و مدت‌زمان دسترسی خاصی دارند که توسط مالک منبع اعطا شده و توسط سرور منبع و سرور صدور مجوز اجرایی می‌شود.

توکن ممکن است نشانگر یک شناسه باشد که برای بازیابی اطلاعات مجوز استفاده می‌شود یا ممکن است اطلاعات مجوز را به شیوه ای قابل تأیید در خود داشته باشد (یعنی یک‌رشته توکن شامل مقداری داده و یک امضا). برای استفاده

1 Refresh Token
2 Scope

کلاینت از توکن، ممکن است اعتبارنامه احراز هویت دیگری نیاز باشد که خارج از این مبحث است.

توکن دسترسی یک لایه انتزاعی را فراهم می‌کند ساختارهای مختلف احراز هویت (مانند نام‌کاربری و گذرواژه) را با یک توکن واحد که توسط سرور منبع قابل‌درک است جایگزین می‌کند. این مفهوم امکان صدور توکن‌های دسترسی محدودتر از حق‌امتیاز مجوز مورداستفاده برای به‌دست‌آوردن همان توکن‌ها را امکان‌پذیر می‌کند و همچنین نیاز سرور منبع به درک طیف گسترده‌ای از روش‌های احراز هویت را برطرف می‌کند.

توکن‌های دسترسی می‌توانند بر اساس الزامات امنیتی سرور منبع دارای قالب، ساختار و روش‌های بکارگیری (مثلاً ویژگی‌های رمزنگاری) متفاوتی باشند. خصوصیات توکن دسترسی و روش‌های مورداستفاده برای دسترسی به منابع حفاظت‌شده خارج از این مبحث است و در توضیحات بخش پیوست مانند RFC 6750 تعریف شده است.

برای دسترسی به استاندارد RFC 6750 کد را اسکن کنید.

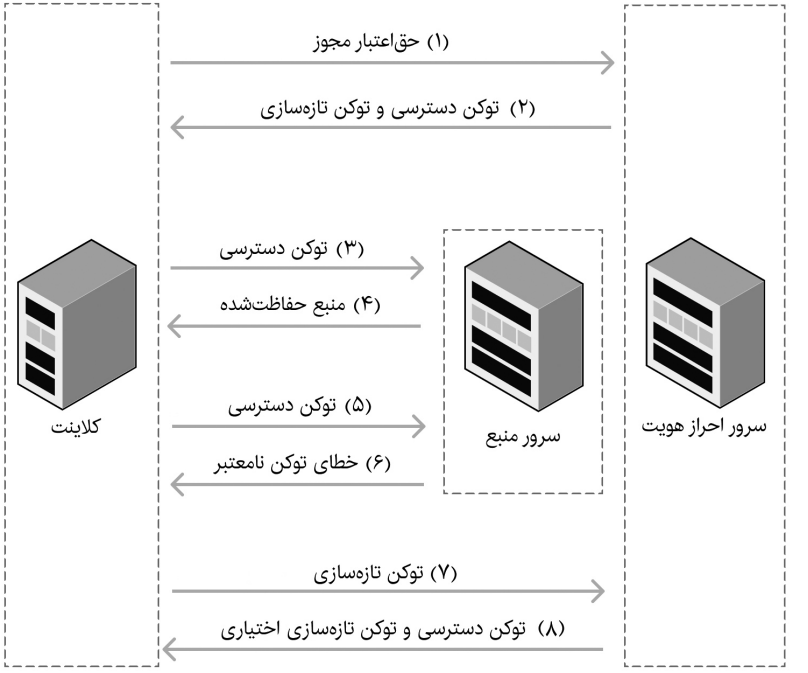
توکن تازه‌سازی

توکن‌های تازه‌سازی اعتبارنامه‌هایی هستند که برای به‌دست‌آوردن توکن‌های دسترسی استفاده می‌شوند. توکن‌های تازه‌سازی توسط سرور صدور مجوز برای کلاینت صادر می‌شوند و برای دریافت توکن دسترسی جدید زمانی که توکن دسترسی فعلی نامعتبر یا منقضی شده است یا برای دریافت توکن‌های دسترسی اضافی با دامنه یکسان یا محدودتر استفاده می‌شود (توکن‌های دسترسی ممکن است طول عمر کوتاه‌تر و دسترسی‌های کمتری نسبت به مجوز اعطاشده توسط مالک منبع داشته باشند).

صدور توکن تازه‌سازی به صلاحدید سرور صدور مجوز اختیاری است. اگر سرور صدور مجوز یک توکن تازه‌سازی صادر کند، این کار هنگام صدور توکن دسترسی (یعنی مرحله (۴) در شکل ۱) انجام می‌شود.

توکن تازه‌سازی رشته‌ای است که نشانگر مجوز اعطا شده توسط مالک منبع به کلاینت است. این رشته معمولاً برای کلاینت مبهم است. این توکن، نشانگر شناسه مورداستفاده برای بازیابی اطلاعات مجوز است. برخلاف توکن‌های دسترسی، توکن تازه‌سازی تنها مختص استفاده در سرورهای مجوز هستند و هرگز به سرورهای منبع ارسال نمی‌شوند.

فرآیند نشان‌داده‌شده در شکل ۲ شامل مراحل زیر است:



شکل شماره ۲ - تازه‌سازی یک توکن دسترسی منقضی‌شده با استفاده از توکن تازه‌سازی

- کلاینت با احراز هویت توسط سرور صدور مجوز و ارائه حق‌امتیاز مجوز، درخواست توکن دسترسی می‌کند.
- سرور صدور مجوز کلاینت را احراز هویت می‌کند و حق‌امتیاز مجوز را اعتبارسنجی نموده و در صورت معتبر بودن، یک توکن دسترسی و یک توکن تازه‌سازی صادر می‌کند.
- کلاینت با ارائه توکن دسترسی درخواست منبع حفاظت‌شده را به سرور منبع می‌دهد.
- سرور منبع توکن دسترسی را اعتبارسنجی کرده و در صورت معتبر بودن، این درخواست را می‌پذیرد.
- مراحل (۳) و (۴) تا زمان انقضای دسترسی توکن تکرار می‌شود. اگر کلاینت بفهمد که توکن دسترسی منقضی شده، به مرحله (۷) می‌رود؛ در غیر این صورت، درخواست دیگری به منبع حفاظت‌شده می‌دهد.
- اگر توکن دسترسی نامعتبر باشد، سرور منبع خطای توکن نامعتبر می‌دهد.
- کلاینت با احراز هویت توسط سرور صدور مجوز و ارائه توکن تازه‌سازی، یک توکن دسترسی جدید درخواست می‌کند. شرایط احراز هویت کلاینت مبتنی بر نوع کلاینت و خط‌مشی‌های سرور صدور مجوز است.
- سرور صدور مجوز کلاینت را احراز هویت می‌کند و توکن تازه‌سازی را اعتبارسنجی می‌کند و در صورت معتبر بودن، یک توکن دسترسی جدید (و به‌صورت اختیاری یک توکن تازه‌سازی) صادر می‌کند.

همان‌طور که در فصل ۷ نیز توضیح داده خواهد شد، مراحل (۳)، (۴)، (۵) و (۶) خارج از این مبحث است.

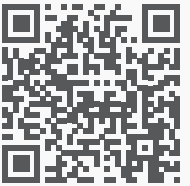
نسخه TLS

هر زمان که امنیت لایه انتقال (TLS[1]) در این استاندارد استفاده می‌شود، نسخه (یا نسخه‌ها) مناسب TLS باتوجه‌به گستردگی توسعه و آسیب‌پذیری‌های امنیتی شناخته‌شده در طول زمان تغییر می‌کند. در زمان نگارش این متن، TLS نسخه ۱.۲، RFC 5246 جدیدترین نسخه است اما کمتر رواج یافته است و شاید به‌راحتی برای

1 Transport Layer Security

پیاده‌سازی در دسترس نباشد. TLS 1.0 متداول‌ترین نسخه است و بیشترین قابلیت تعامل را ارائه می‌دهد.

برای دسترسی به استاندارد RFC 5246 کد را اسکن کنید.



برای دسترسی به استاندارد RFC 2246 کد را اسکن کنید.

تغییر مسیرهای HTTP

در این کتاب از تغییر مسیرهای HTTP استفاده می‌شود که در آن کلاینت یا سرور صدور مجوز نماینده کاربر مالک منبع را به مقصد دیگری هدایت می‌کند. هرچند مثال‌های موجود در این استاندارد استفاده از کد وضعیت HTTP 302 را نشان می‌دهند، اما استفاده از هر روش دیگری که توسط نماینده کاربر برای تغییر مسیر به کار می‌برد مجاز است و به بخشی از پیاده‌سازی محسوب می‌شود.

قابلیت‌تعامل

OAuth 2.0 چارچوبی را برای مجوزدهی با شاخصه‌های امنیتی شفاف فراهم می‌کند. بااین‌حال، این سند به‌عنوان یک چارچوب غنی و بسیار قابل توسعه با تعداد زیادی مولفه انتخابی، به احتمال زیاد طیف وسیعی از پیاده‌سازی‌های غیر قابل

تعامل را ایجـاد می‌کند. به‌علاوه تعـدادی از مولفه‌های الزامی در این سـند به صورت جزئی یا کامل تعریف‌نشـده‌اند (مثلاً ثبت کلاینت، قابلیت‌های سـرور صدور مجوز، کشف نقطه پایانی).

بـدون ایـن مولفه‌هـا، کلاینت بـه منظـور تعامـل بـا سـرورها باید بـه طور دسـتی و اختصاصی باتوجه‌به سرور صدور مجوز و سرور منبع پیکربندی شوند. این چارچوب با ایـن فکر نوشـته شـده اسـت کـه در آینده بـا توجـه بـه فعالیت‌هـای آتـی دورنمای پروفایل‌ها و افزونه‌ها را تعریف کند تا امکان دستیابی به قابلیت تعامل در سطح وب ایجاد شود.

قراردادهای نمادگذاری

کلیدواژه‌های[1] «باید»، « نباید»، «ضروری»، «بهتر است که»، «پیشنهادی»، «ممکن است» و «اختیاری» در این سـند باید به‌نحوی‌که در RFC 2119 شرح داده شده، تفسیر شوند.

برای دسترسی به استاندارد RFC 2119 کد را اسکن کنید.

در این سند از نمادنویسـی ABNF حاشـیه RFC 5234 استفاده‌شـده است. علاوه بر ایـن، قانـون URI-reference از «شناسـه منبـع یکنواخـت (URI) نحـو عمومی» RFC 3986 برداشت شده‌است. برخی از اصطلاحات مرتبط با امنیت را باید باتوجه‌به معنایی درک کرد که در RFC 4949 تعریف‌شـده اسـت. این اصطلاحات عبارت‌اند از «حمله»، «احراز هویت»، «صدور مجوز»، «گواهـی»، «محرمانگی»، «اعتبارنامه»، «رمزنگاری»، «هویت»، «علامت»، «امضا»، «اعتماد»، «اعتبارسنجی» و «تأیید».

۱ در ترجمه هر کجا که از این کلمات با معانی خاص استاندارد RFC ۲۱۱۹ استفاده شده، واژگان به‌صورت برجسته (BOLD) نوشته شده است. همچنین معانی دقیق این واژگان در انتهای کتاب آمده است.

هیچ ندیده‌ای هنوز

انتشارات راه‌پرداخت

برای سفارش اینترنتی این کتاب به وبسایت انتشارات راه‌پرداخت مراجعه کنید
way2pay.shop

استاندارد OAuth از شناسایی شکافی در میان خواسته‌های مشتریان و خواسته‌های برنامه‌ها متولد شـد. ماجرا به نوامبر ۲۰۰۶ برمی‌گردد؛ یعنی زمانی که بیلن کوک در حین کار بر روی اجرای Open ID به دنبال راهی بود تا به کاربر اجازه بدهد بدون نیاز به ثبت‌نام، با تکیه بر اطلاعاتی که قبلاً در سایت‌هایی مثل گوگل ثبت کرده است، بتواند در وب‌سایت مورد نظرش حسـاب کاربری بسـازد. برای ایـن کار لازم بود کـه کاربر با استفاده از برنامه‌ای در ارتباط با Open ID و API احراز هویت شود. اما کوک و همکارانش در این مسـیر متوجه شـدند که هیچ استاندارد بازی برای تفویض دسترسی وجود ندارد. بعد از این، بازار چاره‌جویی‌ها داغ شد تا اینکه سـرانجام گروهی مجازی از برنامه‌نویسان پروتکلی باز در خصوص تفویض دسترسـی نوشتند و در سـال ۲۰۰۷ پیش‌نویس نهایی آن را منتشـر کردند. به دنبال آن، کارگروهی بـرای بحث در مورد اسـتانداردکردن پروتکل OAuth و ثبت آن در استانداردهای IETF شکل گرفت و در نهایت، در سال ۲۰۱۰ پروتکل OAuth 1.0 به عنوان یک استاندارد رسمی با کد RFC 5849 منتشر شد و تمام برنامه‌های کاربردی سوم‌شخص توییتر ملزم به استفاده از آن شدند.

چارچوب OAuth 2.0 با درنظرگرفتن الزامات توسـعه‌پذیری، طبق آرای بخش گسترده‌ای از اعضای IETF و با عنوان RFC 6749 منتشر شده اسـت. کتابی که در دسـت دارید، برگردان نسخه اصلاح‌شده این استاندارد است.

ISBN 978-622-7702-47-7
انتشارات راه‌پرداخت
ناشر فناوری و نوآوری
way2pay.press
۸۴ هزار تومان