Plattformübergreifende Autorisierung im Web mit OAuth2

Heutzutage verwenden wir zahlreiche Webseiten, Programme und mobile Apps, um unser privates und berufliches Leben einfacher oder komfortabler zu gestalten. Angesichts dieser großen Vielfalt an Tools ist es inzwischen für viele Menschen normal geworden, die Inhalte einer Anwendungswebsite (zum Beispiel Instagram) auch auf einer anderen Website (beispielsweise Facebook) zu nutzen – also plattformübergreifend zu agieren. Da bei diesen Vorgängen aber eine Menge persönlicher Daten übertragen werden, stellt sich die Frage nach der Sicherheit der eigenen Privatsphäre. Das Autorisierungsprotokoll OAuth soll das Risiko für unbefugten Datenmissbrauch mindern.

Was ist OAuth?

OAuth, kurz für „Open Authorization“, ist ein offenes Standardprotokoll, das eine sichere API-Autorisierung ermöglicht. Der Programmierer-Fachbegriff API (kurz für „Application Programming Interface“) bezeichnet in diesem Zusammenhang eine Schnittstelle, die als Datenübermittler zwischen verschiedenen Anwendungen, Benutzeroberflächen oder Webseiten fungiert. Eine Autorisierung solcher von APIs durchgeführten Datenübermittlungen ist deshalb erforderlich, weil ohne sie das Risiko besteht, dass unbefugte Dritte die Daten abfangen und für ihre Zwecke missbrauchen.

Das heißt: Soll eine App zum Beispiel im Namen des Nutzers auf Facebook posten (also auf das API von Facebook zugreifen), muss sie eine Erlaubnis dieses Nutzers vorweisen können. Ebenso benötigt eine Anwendung die Vollmacht des Nutzers, um seine Profilinformationen aus einem anderen Dienst ziehen zu können. Mittels OAuth kann der Nutzer solch eine Vollmacht (Autorisierung) erteilen, ohne der autorisierten Anwendung selbst seinen Benutzernamen und sein Passwort mitteilen zu müssen – er behält also die volle Kontrolle über seine Daten.

Anbieter wie Google, Twitter und Facebook können OAuth im Umkehrschluss dazu nutzen, ihre Produkte und Dienstleistungen flexibler und zugleich sicherer zu gestalten, etwa mittels Single-Sign-On-Lösungen. Und auch Amazon und Microsoft zählen inzwischen zum Kreis großer Unternehmen, die OAuth als Standard zur Zugriffsdelegierung für ihre Dienstleistungen nutzen.

Aufgabe 1: Beschreiben Sie den wesentlichen Vorteil von OAuth, welcher auch gleichzeitig zu mehr Sicherheit führt.

OAuth ermöglicht sicheren Zugriff auf geschützte Ressourcen, indem es Token anstelle von sensiblen Anmeldeinformationen verwendet. Dies erhöht die Sicherheit, da Anwendungen nur autorisierten Zugriff erhalten, ohne direkten Zugang zu Benutzerpasswörtern zu haben.

Tafel:
1. Plattformübergreifende Autorisierung (ohne Passw. austauschen zu müssen)

OAuth vs. OAuth2

Das nicht rückwärtskompatible OAuth2 (auch „OAuth 2.0“ genannt) wurde im Oktober 2012 als vollständige Überarbeitung von OAuth veröffentlicht und hat selbiges inzwischen weitestgehend abgelöst. So unterstützt die Graph-API von Facebook nur noch das neue Protokoll als Autorisierungsstandard.

Prinzipiell ist die Aufgabe von OAuth2 dieselbe wie die seines Vorgängers: mittels API-Autorisierung dem Nutzer mehr Flexibilität bei gleichzeitig hoher Sicherheit zu ermöglichen. Jedoch wurden zahlreiche Schwächen des ursprünglichen Protokolls behoben, die das Coding und die Implementierung umso schwieriger machten, je komplexer die Systeme von Facebook, Twitter und anderen API-Betreibern im Laufe der Zeit wurden.

Abgesehen von einer gänzlich veränderten Terminologie ist die wichtigste Neuerung von OAuth2, dass es im Gegensatz zu seinem Vorgänger keine kryptografischen Signaturen mehr für jede Maschine-zu-Maschine-Kommunikation im Protokollablauf verlangt. Das erleichtert die Anwendung und Erweiterung des Protokolls ungemein. Es bedeutet aber auch, dass das neue Protokoll technisch gesehen weniger sicher ist – ein Fakt, der OAuth2 einiges an Kritik einbrachte.

Open Authorization 2.0 wurde außerdem um stärker differenzierte Genehmigungsprozesse (Grant Types) ergänzt und die Performance des Protokolls verbessert. Das haben die Entwickler dadurch erreicht, dass OAuth2 nicht mehr bei jedem Kommunikationsschritt nach den Zugangsdaten des Nutzers (Resource Owner) fragt, sondern nur bei der erstmaligen Autorisierung der jeweiligen Anwendung (Client). Eine weitere nennenswerte Neuerung sind Access-Token mit kürzerer Gültigkeit, die es einem Dienst (Resource Server) erleichtern, erteilte Autorisierungen wieder zurückzunehmen. Zudem kann der Nutzer unter OAuth2 selbst entscheiden, welche Berechtigungen (scope) er einer Anwendung zugesteht.

Aufgabe 2: Nennen Sie einen Grund, warum die technische Implementierung von OAuth2 im Vergleich zu OAuth leichter geworden ist.

OAuth2 ist technisch einfacher als OAuth1, da es eine klare Trennung von Authentifizierung und Autorisierung ermöglicht, was zu einer modularen Implementierung führt und die Anpassung an verschiedene Szenarien erleichtert.

Tafel:

In OAuth2 keine kryptographischen Signaturen mehr eingesetzt
(=> Probleme bei Sicherstellung der Integrität

+ leichtere Implementierung
- weniger Sicherheit

OpenID und SAML als ähnliche Protokolle zu OAuth

OpenID (kurz für „Open Identification“) ist ein offenes Protokoll: Wenn ein Nutzer sich einen OpenID-Account erstellt, kann er sich mit diesem via Token bei anderen Diensten und Anwendungen anmelden, die ebenfalls OpenID unterstützen. Ein gutes Beispiel hierfür ist der Button „Mit Google anmelden“, den man inzwischen auf vielen Webseiten findet und der ein Single-Sign-on-Verfahren über den Google-Account des Nutzers zulässt.

Somit ist OpenID im Gegensatz zu OAuth strenggenommen kein Autorisierungs-, sondern ein Authentifizierungsstandard. Allerdings sind die beiden Protokolle seit 2014 eng miteinander verknüpft: OAuth 2.0 ist nämlich die Grundlage, auf der die neue Version von OpenID, genannt OpenID Connect (OIDC), aufbaut. Durch Authentifizierung wird bestätigt, dass Benutzer die sind, die sie zu sein vorgeben, während diese Benutzer per Autorisierung die Erlaubnis erhalten, auf Ressourcen zuzugreifen.

Die offene und XML-basierte Security Assertion Markup Language (kurz: SAML) verbindet gewissermaßen die Eigenschaften der beiden vorangegangenen Konzepte. Bei ihr handelt es sich nämlich sowohl um einen Standard für die Authentifizierung als auch für die Autorisierung. SAML ähnelt in seiner Funktionsweise OpenID und kommt ebenfalls bei Single-Sign-on-Verfahren zum Einsatz. Single Sign-On ist ein Verfahren der einmaligen Authentifizierung, nach der ohne erneute Eingabe der Login-Daten auf weitere Anwendungen oder Dienste zugegriffen werden kann. SSO authentifiziert den Benutzer automatisch für die Anmeldung bei nachgelagerten Anwendungen.

Aufgabe 3: Beschreiben Sie den Unterschied zwischen Authentifizierung und Autorisierung. Nennen Sie außerdem für beide Verfahren ein Beispiel.

Authentifizierung: Überprüfen der Identität des Benutzers (=> Passwort/ Username - Abfrage)

Beispiel: OpenID

Authorisierung: Festlegen von Rechten (innerhalb eines Systems)

Beispiel: OAuth2.0

Aufgabe 4: Nennen Sie einen Vorteil und einen Nachteil von Single-Sign-On.

Vorteil: Benutzerfreundlichkeit durch einmalige Anmeldung für verschiedene Dienste. (Zeitersparnis)

Nachteil: Erhöhte Sicherheitsanfälligkeit bei Kompromittierung des Single-Sign-On-Kontos (Sicherheitsrisiko), Technischer Implementierungsaufwand

Die Rollen bei OAuth2

Oauth2 definiert klare Rollen, um den Zugriff auf geschützte Daten zu regeln. Innerhalb des Oauth2-Frameworks gibt es verschiedene zentrale Rollen.

Die Rolle des Resource Owners oder Nutzers bzw. User ist entscheidend. Dies repräsentiert eine Entität, die einem sogenannten Client Zugriff auf seine geschützten Daten, auch als Ressourcen bezeichnet, gewährt. Der Resource Owner ist somit die Quelle der autorisierten Informationen. Der Resource Server (Dienst) fungiert als der Ort, an dem die geschützten Daten des Resource Owners gespeichert sind. Er ist dafür verantwortlich, den Zugriff auf diese Daten gemäß den erteilten Berechtigungen zu verwalten. Die Rolle des Clients (Dritter oder Third Party) bezieht sich auf Desktop-, Web- oder Mobile-Anwendungen, die den Wunsch haben, auf die geschützten Daten des Resource Owners zuzugreifen. Der Client spielt eine zentrale Rolle bei der Anfrage und Verwaltung von Zugriffsberechtigungen. Der Authorization Server ist von entscheidender Bedeutung, da er den Resource Owner authentifiziert und daraufhin zeitlich begrenzte Zugriffstoken (Access-Token) ausstellt, die den definierten Anwendungsbereich, auch als „scope“ bezeichnet, abdecken. Es ist erwähnenswert, dass Authorization Server und Resource Server oft in der Praxis zusammen betrieben werden und dann als Oauth-Server bezeichnet werden.

Aufgabe 5: Nennen Sie die beteiligten Rollen in einem OAuth2-Autorisierungsprozess und beschreiben sie deren Aufgaben kurz und stichpunktartig.

Rolle	Beschreibung bzw. Aufgabe
Resource Owners (Nutzer/User)	Die Entität, die einem Client Zugriff auf geschützte Daten (Ressourcen) gewährt. Der Resource Owner ist die Quelle der autorisierten Informationen.
Resource Servers (Ort d. geschützten Daten)	Der Ort, an dem die geschützten Daten des Resource Owners gespeichert sind. Der Resource Server ist verantwortlich für die Verwaltung des Zugriffs auf diese Daten gemäß den erteilten Berechtigungen.
Clients (Dritter, Thrid Party, App, Anwendung)	Desktop-, Web- oder Mobile-Anwendungen, die darauf abzielen, auf die geschützten Daten des Resource Owners zuzugreifen. Der Client spielt eine zentrale Rolle bei der Anfrage und Verwaltung von Zugriffsberechtigungen.
Authorization Server	Authentifiziert den Resource Owner und stellt zeitlich begrenzte Zugriffstoken (Access-Token) aus, die den definierten Anwendungsbereich (scope) abdecken. Der Authorization Server und der Resource Server werden oft zusammen betrieben und als OAuth-Server bezeichnet.

Ressource Server
Authorization Server } OAuth – Server

OAuth2-Autorisierungsablauf

Aufgabe 6: Lesen Sie die Informationen zum Autorisierungsablauf und stellen Sie diesen Ablauf anschließend grafisch mit einer Skizze da. Nutzen Sie dazu das vorgegebene Grundgerüst.

Das OAuth2-Protokoll durchläuft einen präzisen Ablauf, um die Autorisierung und den sicheren Zugriff auf geschützte Daten zu gewährleisten.

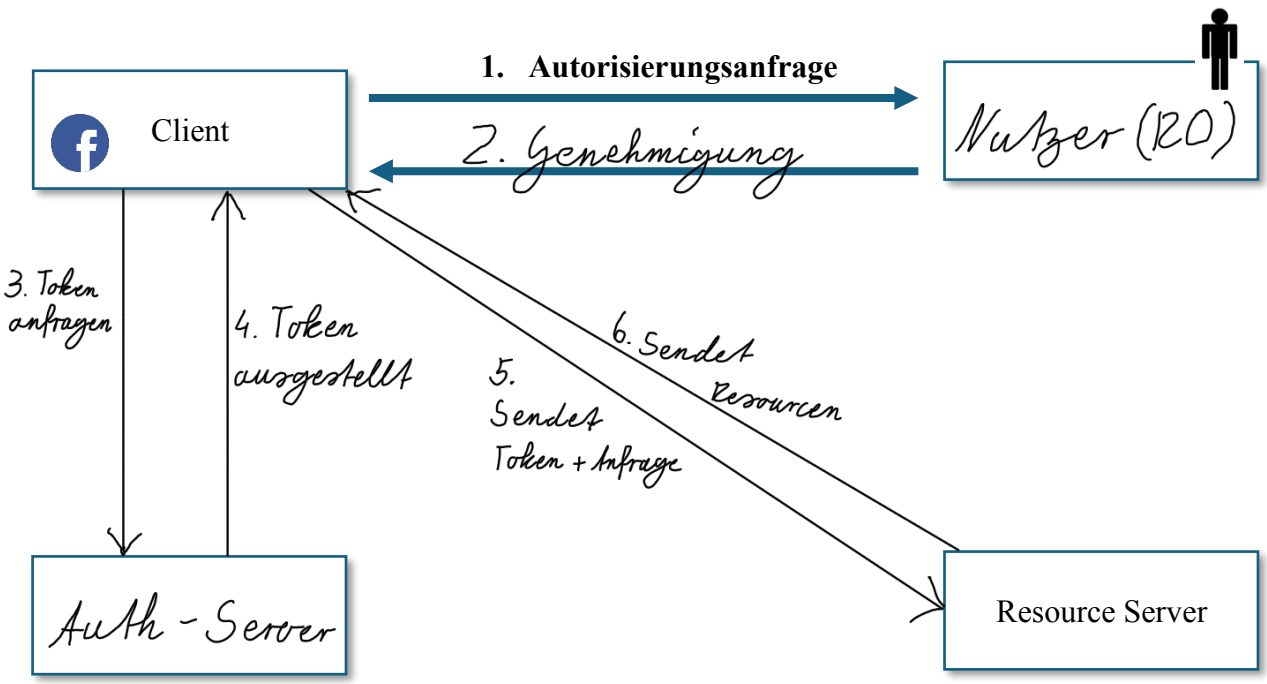
Zunächst fordert der Client, sei es direkt oder über den Authorization Server, eine Autorisierung beim Resource Owner an. Dieser Prozess beinhaltet die Aushandlung und Genehmigung der Zugriffsberechtigungen.

Nachdem der Resource Owner die Autorisierung erteilt hat, beantragt der Client mithilfe dieser Autorisierungsgenehmigung einen Access-Token beim Authorization Server. Hier erfolgt die Authentifizierung des Clients durch den Authorization Server, der daraufhin einen zeitlich begrenzten Access-Token ausstellt.

Der Client verwendet nun den erhaltenen Access-Token, um beim Resource Server die relevanten geschützten Daten des Resource Owners anzufordern. Dieser Zugriff erfolgt unter Verwendung des gültigen Tokens, was die Sicherheit des gesamten Prozesses gewährleistet.

Der Resource Server authentifiziert den Client anhand des Access-Tokens und stellt die angeforderten Daten bereit. Durch diesen klaren und strukturierten Ablauf ermöglicht OAuth2 einen effizienten Austausch von Autorisierungsdaten und gewährleistet gleichzeitig die Sicherheit und Integrität des gesamten Systems.

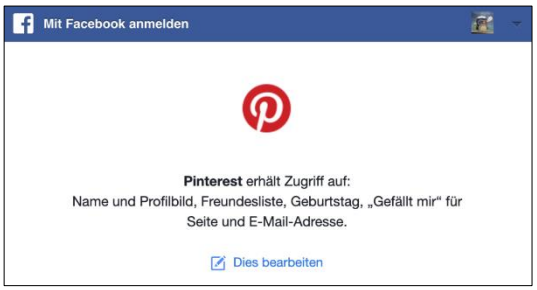
Grafische Darstellung:



Konkretes Beispiel für den OAuth2-Protokollablauf

Konkrete Beispiele für den OAuth2-Protokollablauf liefern die sozialen Netzwerke **Pinterest** und **Facebook**. So bietet Pinterest die Option, Kontakte aus Facebook-Freundeslisten zu importieren. Dafür benötigt Pinterest Zugriff auf den jeweiligen Account und die dort hinterlegten Informationen. Aus Gründen der Datensicherheit wäre es jedoch nicht ratsam, den Benutzernamen und das Passwort für Facebook an Pinterest weiterzugeben – schließlich hätte Pinterest dann jederzeit uneingeschränkten Zugang zu allen Daten und Funktionen des Facebook-Accounts. Damit Pinterest trotzdem auf die benötigten Facebook-Daten zugreifen kann, kommt OAuth2 zum Einsatz:

1. Zunächst loggt man sich in seinen Pinterest-Account ein und navigiert im Benutzerprofil zu den Einstellungen.
2. In der Menüleiste „Soziale Netzwerke“ schiebt man nun den Regler neben „Facebook“ auf „Ja“.
3. Pinterest bittet nun darum, sich bei Facebook einzuloggen und die Pinterest-App zu bestätigen. Diese Handlung gilt als Autorisierungsgenehmigung.
4. Pinterest fordert einen Access-Token beim Authorization Server von Facebook an und nutzt diesen anschließend, um auf die geschützten Daten auf dem Resourcer Server zuzugreifen.



5. Die importierten Facebook-Freunde werden nun auch im Pinterest-Account angezeigt.

Sicherheit und Kritik von OAuth2

Dass auch ein für den Schutz persönlicher Daten konzipiertes System wie OAuth nicht hundertprozentig perfekt sein kann, zeigte sich bereits im April 2009, als im Authentifizierungsablauf des Protokolls eine **Sicherheitslücke** entdeckt wurde. Wie bei vielen anderen solcher Systeme ist auch Phishing ein ständiges Risiko: So wurden zwischen April und Mai 2017 eine Million Gmail-User Opfer einer **OAuth-basierten Phishing-Attacke**. In einer betrügerischen E-Mail waren sie darum gebeten worden, ihre Autorisierung über ein gefälschtes Interface zu erteilen, um einer angeblichen Anwendung namens „Google Apps“ Zugriff auf ihre Account-Daten zu ermöglichen.

Aufgabe 7: Verschaffen Sie sich mit folgendem Artikel (siehe Link) einen Überblick über einen bekannten Angriff auf OAuth2-Logins. Warum war dieser Angriff möglich?

ZDNet / Sicherheit / Authentifizierung

Sicherheitslücke in OAuth 2.0 macht mehr als eine Milliarde Apps angreifbar

https://www.zdnet.de/88282389/sicherheitsluecke-in-oauth-2-0-macht-mehr-als-eine-milliarde-apps-angreifbar/

Quelle: https://www.ionos.de/digitalguide/server/sicherheit/was-ist-oauth/