OAuth und OpenID Connect - Erfahrungen und Konzepte

Roland H. Steinegger[1] · Nadina Hintz[2] · Martina Müller[2] · Philipp Schleier[2]
Fabian Zoller[2] · Sebastian Abeck[1]

Karlsruher Institut für Technologie[1]
Forschungsgruppe Cooperation & Management (Prof. Abeck)
Zirkel 2, 76137 Karlsruhe
steinegger — abeck@kit.edu

iC Consult [2]
iC Consult Gesellschaft für Systemintegration und Kommunikation mbH
Zettachring 8a, 70567 Stuttgart
nadina.hintz — martina.mueller — philipp.schleier —
fabian.zoller@ic-consult.de

Zusammenfassung

Das OAuth 2.0 Autorisierungsrahmenwerk (OAuth) und die darauf aufbauende OpenID Connect 1.0 Identitätsschicht (OpenID Connect) sind seit 2012 bzw. 2014 standardisiert und werden in aktuellen Web-Anwendungen, wie Facebook oder LinkedIn, rege genutzt. OAuth bietet als Rahmenwerk zwar gewisse Leitplanken für eine Umsetzung, allerdings ist hierdurch eine Einordnung des Standards in bestehende Konzepte nicht offensichtlich und auch der Einsatz in Unternehmen führt zu verschiedenen Problemen. Existierende Veröffentlichungen zu OAuth und OpenID Connect beschreiben zwar technische Umsetzungen und Sicherheitsprobleme der Standards, dabei werden aber konzeptionelle Probleme bei der Umsetzung kaum adressiert. Zudem ist die verwendete Teminologie der verschiedenen Quellen inkonsistent, bspw. werden im Zusammenhang mit OAuth einerseits das Konzept der Identitäts- und andererseits der Befugnisdelegation verwendet. Mit unserem Beitrag wollen wir auf Gründe für den Einsatz von OAuth in Unternehmen und dabei auftretende konzeptionelle Probleme eingehen und zudem, durch die Beschreibung von OAuth als Sicherheitsmusterkandidat, eine einheitliche Terminologie und Einordnung in bestehende Ansätze leisten. Als Ergebnis präsentieren wir Erfahrungen und Probleme im Praxiseinsatz, die durch die Beschreibung des Standards als Sicherheitsmuster aufgegriffen und konzeptionell eingeordnet werden. Durch den Vergleich der Praxisprobleme mit den abstrahierten, wiederkehrenden Lösungen und die Verwendung der Sicherheitsmuster in der SmartCampus Web-Anwendung evaluieren wir unseren Ansatz.

1 Einführung

Die Absicherung von Web-Anwendungen hat sich in den letzten Jahren durch aktuelle Technologietrends stark verändert. In Webanwendungen hat sich der Representational State Transfer (REST) Architekturstil gemeinsam mit der JavaScript Object Notation (JSON) als Nachrichtenformat durchgesetzt und den schwergewichtigen WS-* Ansatz für Webservices basierend auf

P. Schartner · N.N (Hrsg.) · D•A•CH Security 2016 · syssec (2016) 1-12.

SOAP [GHMM+03] und WSDL [CCMW01] zur Umsetzung von serviceorientierte Architekturen abgelöst. Aus dieser Veränderung heraus haben sich die Standards OAuth und OpenID Connect zur Absicherung von Application Programming Interfaces (API) und Autorisationsdelegation sowie zur Authentifizierung entwickelt.

Der Einzug von OAuth und OpenID Connect in Unternehmen aus Branchen, bei denen das Kerngeschäft nicht an das Internet gebunden ist, geht schleppend voran [WeWA15], obwohl einerseits die Attraktivität des Online-Angebots durch diese Technologien gesteigert werden und andererseits eine Auslagerung der Identitäten wirtschaftliche Vorteile bringen kann.

Grund für diese Beobachtung könnte ein fehlendes Verständnis der von den Standards angebotenen Funktionalität sein. Zudem ist die Steigerung der Anwendungskomplexität durch den Einsatz von OAuth und OpenID Connect aus den Standards nicht absehbar. Als Ressourcenanbieter ist bspw. der Bezug zu bestehenden Konzepten, wie der Autorisierung, unklar; wie soll eine rollenbasierte Zugriffskontrolle mit den *Scopes* von OAuth umgesetzt werden?

Der Standard selbst lässt diese Zusammenhänge offen und fokussiert nur die Schnittstellenbeschreibung sowie die Interaktionen. Aktuelle Literatur zu den Standards bespricht nur eine technische Umsetzung basierend auf Beispielprojekten [Bihi15, Siri14], ohne die Standards konzeptionell einzuordnen oder auf Probleme beim Einsatz einzugehen. In [Siri14] werden zwar Sicherheitsmuster passend zur API-Sicherheit vorgestellt, aber nicht in Bezug zu den Standards gesetzt. Beispielsweise wird die Notwendigkeit eines Speicherorts für Token, die die Basis der Zugriffskontrolle sind, nicht offen diskutiert.

Des Weiteren ist die Verwendung von Begriffen inkonsistent, bspw. wird OAuth sowohl als Standard zur Identitäts-, Autorisations- und Befugnisdelegation bezeichnet [Bihi15, Siri14, Hard12]. Auch wissenschaftliche Publikationen fokussieren technische Abläufe und Details, um bspw. Schwachstellen offenzulegen, und vernachlässigen die Konzepte hinter dem Standard [ChJR11, PSKP+11, SuBe12, YaMa13].

Durch Beispiele aus Kundenprojekten zeigen wir die vielfältigen Einsatzmöglichkeiten der Protokolle, beschreiben die Probleme beim Einsatz und gewonnene Erfahrungen. Zur konzeptionellen Einordnung und als Beschreibungsform der Sichten auf die Standards verwenden wir Sicherheitsmuster. Einerseits ermöglichen diese eine systematische Beschreibung der Lösung samt statischer und dynamischer Aspekte, andererseits können die neu entstehenden Sicherheitsmuster in eine bestehende Sicherheitsmusterlandschaft eingeordnet und so deren Beitrag herausgestellt und die Verwendung durch andere erleichtert werden. Durch diese Einordnung können zudem typische Muster aus dem IAM-Bereich zur Verbesserung von Qualitätseigenschaften genutzt werden, wie bspw. in [SSVA14] beschrieben.

2 Hintergrund

Der Einsatz von Identity Providern (IdP) wie Facebook, Twitter oder Xing steigt in der privaten Welt stetig. Aber auch im betrieblichen und industriellen Einsatz werden immer mehr IdPs genutzt, die auf OAuth bzw. OpenID Connect basieren. Der Unterschied zwischen OAuth und OpenID Connect sowie die historische Entwicklung sollen an dieser Stelle erläutert werden.

Sowohl im beruflichen als auch im privaten Umfeld haben IdPs eine hohe Anzahl von Benutzerprofilen, die mit Daten gefüllt sind. Ziel ist es, diese Benutzerdaten über verschiedene Anwendungen hinweg nutzen zu können, ohne dass diese Anwendungen die Zugangsdaten des Nutzers erhalten.

2.1 OAuth 2.0 Rahmenwerk

Das Autorisierungskonzept OAuth verfolgt den Grundgedanken der Autorisierung und Passwortweitergabe. Das Konzept ist auf einem hohen Abstraktionslevel mit dem Konzept des Valet-Parkens zu vergleichen. Beim Valet-Parken gibt es einen gesonderten Schlüssel, der nur eingeschränkte Berechtigungen hat, wie etwa eine beschränkte Geschwindigkeit und Kilometerlaufzeit, und dem einparkenden Valet (Fahrer) zur Verfügung gestellt wird.

OAuth ermöglicht im Vergleich den Zugriff auf geschützte Ressourcen, bzw. ermöglicht den bedingten Zugriff. Beim OAuth Konzept ist die Ressource das Auto, die Drittanwendung (engl. Client) ist der Valet und das sogenannte Access Token der Valet-Schlüssel. Das OAuth-Protokoll in der Version 1.0 ist 2007 entstanden [OAu07]. Diese Version hat mehrere Revisionen erlebt, bis schließlich 2012 OAuth 2.0 in Form eines Request for Recommendations (RFCs) vorgestellt wurde [Hard12]. Bei OAuth 2.0 handelt es sich allerdings um kein striktes Protokoll sondern um ein Framework. Wesentliche Unterscheide sind, dass OAuth 2.0 dem Entwickler viele Freiheiten lässt und Nachrichten nicht signiert sind. Im Gegenzug wird die Verwendung von TLS bei HTTP-Aufrufen vorausgesetzt.

Grundsätzlich sind im OAuth 2.0 Framework vier Parteien definiert:

1. *Eigentümer* (engl. Resource Owner, RO),
2. *Ressourcenserver* (engl. Resource Server, RS),
3. *Client* und
4. *Autorisierungsserver* (engl. Authorization Server, AzS).

Der RO ist häufig ein Endbenutzer und die Ressource sind dessen Daten. Der RS kümmert sich um die Aufbewahrung der Ressourcen und reguliert den Zugriff darauf. Der Client ist eine Dritt-Anwendung, die im Namen des RO zugreifen will. Der AzS ist dafür zuständig, den Zugriff des Clients zu gewährleisten. In der Praxis werden die Rollen von RS und AzS oft in einer Komponente abgebildet. Über den sogenannten *Scope* kann der Umfang der Autorisation definiert werden.

2.2 OpenID Connect 1.0

OAuth ist ein Standard zur *delegierten Autorisation*, nicht zur Authentisierung. Ein OAuth Access Token besagt, dass der Überbringer (engl. Bearer) vom RO die Erlaubnis bekommen hat, in dessen Namen für eine begrenzte Zeit auf bestimmte Ressourcen des Benutzers im Umfang des Scopes beim RS zuzugreifen. Dies darf nicht mit einer Identitäts-Delegation verwechselt werden, die einen vollen Zugriff im Namen des RO nach sich ziehen würde.

An dieser Stelle kommt OpenID Connect [SBJM+14] ins Spiel. Der Standard fußt auf OAuth und ermöglicht einem Client, auf einheitliche Weise die Identität des Benutzers zu verifizieren und Profil-Informationen abzurufen.

OpenID Connect kann als flexibles und leichtgewichtiges Pendant zur Security Assertion Markup Language (SAML) [CKPM05] gesehen werden, wobei die *ID Tokens* in OpenID Connect die Rolle einer SAML Assertion einnehmen. Damit sind ID Tokens in Single Sign-On (SSO) Web-Szenarien geeignet, um Benutzerinformationen zu einer Zielapplikation zu übertragen und dort eine Applikationssitzung zu generieren. Prinzipiell können sie auch als Identitätsnachweis zwischen Mobile Apps und REST APIs zum Einsatz kommen.

Andererseits muss beim Zugriff auf eine REST-API aufgrund der Zustandslosigkeit keine Sitzung erzeugt werden, daher ist keine Authentifizierung notwendig, sondern die Autorisierung der Anfrage per OAuth Access Token ist vollkommen ausreichend. Der Einsatz von OpenID Connect ID Tokens ist höchstens dann sinnvoll, wenn nachgelagerte Services Identitätsinformationen benötigen.

3 Praxiseinsatz

In diesem Kapitel wird auf den Einsatz von OAuth und OpenID Connect in aktuellen Projekten eingegangen. Dies soll einerseits den vielfältigen Einsatz der Standards und andererseits die Probleme, die beim Einsatz auftreten, aufzeigen. Einige der Probleme treten auf Grund der Offenheit des Standards auf. Auf diesen Punkt wird in einem Resümee in der Evaluation eingegangen.

3.1 Praxiseinsatz in Deutschland

Der folgende Abschnitt behandelt ein Projekt, welches innerhalb eines international tätigen Unternehmens mit mehr als 200.000 Mitarbeitern im süddeutschen Raum realisiert wird, und dessen verschiedene Blickwinkel auf einen OpenID Connect Service. Der OpenID Connect Service wird innerhalb des Unternehmens Projekten zur Verfügung gestellt, die sich im Identitäts- und Zugriffsmanagement (IAM)-Bereich für Konsumenten bewegen und Applikationen bereit stellen, die der Endkunde direkt verwendet.

3.1.1 Warum eine Umstellung?

Vor OpenID Connect befand sich das System in einem klassischen Modus: eine SSO Sitzung wurde mittels einer zentralen, monolithischen Software und dazu gehörigen *Intercepting Webagents* realisiert. Diese Session war über alle teilnehmenden Applikationen im SSO-Verbund dieselbe. Das kann schwerwiegende Folgen haben, wenn zum Beispiel eine fehlerhafte Applikation oder Komponente zwischen Client und Applikation unfreiwillig die Sitzung preisgibt. Somit wäre es für Dritte möglich im Namen eines Anderen mit einer Applikation zu interagieren.

Einen Unterschied bietet hier OpenID Connect. Es gibt zwar eine SSO-Sitzung, jedoch ist diese hostbasiert (die Sitzung wird auf den AzS ausgestellt) und nicht mehr domänenbasiert. Die Client-Applikationen bekommen jetzt nur noch Informationen über die Sitzung und nicht mehr die Sitzung selbst. Sie sind jetzt selbst verantwortlich eine Client-Applikationssitzung zu erstellen, zu verwalten und zuletzt wieder zu löschen. Diese Sitzung hat keine Gültigkeit bei anderen Applikationen im SSO-Verbund. Ist eine Sitzung Dritten zugänglich, kann der Angreifer nur mit einer Applikation interagieren und nicht mehr mit allen.

Ein weiterer Grund zur Umstellung auf OpenID Connect ist die Offenlegung und standardisierte Weise, wie das Protokoll und die verschiedenen Flüsse funktionieren. Die zunehmende Popularität der Smartphones und deren nativen Applikationen sind ein weiterer Grund des Wechsels. Der antiquierte und verknöcherte Monolith will sich nicht so richtig in die moderne Zeit einpassen und ist nicht ohne Abstriche und Kompromisse dazu fähig APIs zu schützen.

3.1.2 Kundenakzeptanz

Durch die Standardisierung und gute Dokumentation ist die Entwicklung neuer Applikationen, sowie die Migration alter Applikationen, schnell möglich. Gleich mehrere große Projekte haben

zeitnah nach Bekanntgabe der Einführung im Unternehmen eine Umstellung angestrebt.

Noch in der Testphase des OpenID Connect Identity Providers haben erste Projekte angefangen diesen zu adoptieren. Dies war anfangs noch etwas holprig, da sich im eingesetzten Produkt des AzS noch kleine Fehler befanden. Im Laufe der Monate haben sich diese Probleme aufgelöst zu einem stabilen Betrieb.

Stand heute befinden sich zwei große Online-Geschäfte im produktiven Betrieb. Mehrere kleinere Projekte werden dieses Jahr noch zusätzlich Live gehen.

3.1.3 Herausforderungen und Vorteile im Praxiseinsatz

Für Softwareentwickler bietet sich der Vorteil, dass keine zentrale Infrastruktur bereit gestellt werden muss. Es wird lediglich eine *Client ID* samt *Secret*, eine/mehrere *Redirect URL(s)* und eine/mehrere *Logout URL(s)* benötigt, die dem AzS und dem Entwickler bekannt sind.

Durch das fehlen eines zentralen Systems, welches die Client Applikationen vor Angriffen schützt, ist bei OpenID Connect die Client Applikation mehr auf sich selbst gestellt. Zum Beispiel sollte die Client Applikation gegen Cross-Site Scripting (XSS) und SQL Injections abgesichert und die Prüfung des Zugriffs auf geschützten Ressourcen muss gewissenhaft implementiert werden. Des Weiteren ist die Client Applikation selbst für die Sitzungsverwaltung verantwortlich und muss sich um Timeouts kümmern.

3.1.4 Notwendige Soft- und Hardware-Komponenten

Für den Betrieb des OpenID Connect Services wurde eine Appliance gewählt, die sowohl ein Proxy ist und zusätzlich eine Implementierung eines AzS bereitstellt. Des Weiteren sind die notwendigen OpenID Connect Endpunkte wie z.B. UserInfo oder Token Endpoint über die Appliance erreichbar. Ein vorhandenes Verzeichnis mit Identitäten von Endkunden dient dem OpenID Connect Service als IdP.

3.2 Praxiseinsatz in Österreich

Im Folgenden wird die geplante Anwendung von OpenID Connect bei einem mittelständischen, österreichischen Unternehmen beschrieben. Das Projekt befindet sich derzeit in der Konzeptionsphase. Derzeit wird die SAML zur Unterstützung einer Föderation genutzt, dabei werden mehrere Service Provider und ein IdP genutzt.

3.2.1 Geplante Erweiterung

Im Umfang der Planung soll OpenID Connect und OAuth als zusätzliche Komponente zu und zur SAML verwendet werden. SAML soll auch weiterhin ein grundlegendes Element bleiben. Durch die Ausgestaltung von OAuth als Rahmenwerk, wird dieses Zusammenspiel erleichtert.

Der RFC 7522 beschreibt eine mögliche Kombination von SAML und OAuth, wobei SAML-Aussagen innerhalb der OAuth-Nachrichten ausgetauscht werden [CaMJ15]. Auf diese Weise kann auf der bewährten Umsetzung basierend auf SAML aufgebaut und trotzdem die Vorteile der neueren Standards genutzt werden.

3.2.2 Ziele der Einführung

Die angestrebte Weiterentwicklung ist einerseits technisch motiviert und soll die Erweiterbarkeit fördern. Sollten entsprechende Anforderungen vom Access Management oder von den Nut-

zern entstehen, so möchten der Kunde eine bereits bestehende Infrastruktur anbieten können.

Die Verbindung der drei Standards, SAML, OpenId Connect und OAuth, zur Verwendung mit Mirco Services und in Kombination mit Social Media soll andererseits die Authentifizierung und Autorisierung für Mitarbeiter und Kunden zu einer nahtlosen Erfahrung machen.

3.2.3 Ausblick

In Zukunft sollen die derzeit noch bestehenden Grenzen zwischen internen und externen Benutzern aufgelöst werden. Das aktuelle Hauptaugenmerk liegt auf einer technischen Umsetzung der Integrationsanforderungen im Unternehmen. Aspekte wie benutzergesteuerte Freigaben von Unternehmensdiensten durch externe AzS sind noch Zukunftsthemen.

4 Einführung in Sicherheitsmuster und verwandte Arbeiten

Als Grundlage für eine Beschreibung und Einordnung des Rahmenwerks in die Welt der Sicherheitsmuster wird in diesem Abschnitt ein Überblick der Sicherheitsmusterlandschaft rund um die von OAuth und OpenID Connect behandelten Themen gegeben.

4.1 Einführung in Sicherheitsmuster

Sicherheitsmuster sind ein konkreter Fall von Mustern und beschreiben die Lösung wiederkehrender Sicherheitsprobleme. Derartige Muster dienen der frühzeitigen Abwendung von Sicherheitsbedrohungen und sind nicht dazu gedacht, auftretende Schwachstellen in einem System zu reparieren.

Die Beschreibung eines Sicherheitsmusters gliedert sich grob in drei Bereiche. Zunächst illustriert der Kontext anhand eines Szenarios, wo das Muster auftritt. Die Beschreibung des Problems verdeutlicht, welche konkreten Bedrohungen im beschriebenen Kontext auftreten und durch das Muster gelöst werden sollen. In der Lösung wird anschließend für jede Bedrohung eine Gegenmaßnahme angeführt. [Schu03]

Da ein Muster gegebenenfalls nur Teile des Problems löst oder gar zu neuen Problemen führt, sollten laut [Schu03] die Beziehungen zu anderen Mustern aufgezeigt werden. Ergänzend werden durch sogenannte Zwänge (engl. Forces) offene Probleme eines Musters angegeben [Fern13]. Diagramme werden üblicherweise bei der Beschreibung von Mustern zur Verdeutlichung von Abläufen und Strukturen verwendet, wobei [Fern13] Diagramme, die Struktur und Dynamik der Lösung (bspw. Klassen- und Sequenzdiagramme) beschreiben, als festen Bestandteil des Lösungsabschnittes sieht.

4.2 Verwandte Sicherheitsmuster

Die Sicherheitsmusterlandschaft zur Zugriffskontrolle rund um OAuth kann grob in drei Bereiche eingeteilt werden: Beschreibungform der Autorisationsregeln, ihre Administration und die Durchsetzung.

Die *Discretionary Access Control* beschreibt, dass die Autorisation je Benutzer bestimmt werden kann [SaSa94]. Vergleichbar mit der Vergabe der Autorisation bei OAuth je Client. Die Autorisation muss bei OAuth allerdings nicht gespeichert werden, sofern der Zugriffs-Token alle notwendigen Informationen enthält und vom RS analysiert werden kann. Somit ist die Beschreibungsform der Autorisation in OAuth offen gehalten.

Der *Reference Monitor* beschreibt in der abstraktesten Form die Durchsetzung von Autorisationsregeln [SFBHB+06]. Anfragen werden von dem Reference Monitor abgefangen, die Autorisation des authentifizierten Benutzers geprüft und der Zugriff gewährt bzw. abgelehnt. Die *Policy-Based Access Control* spezialisiert das abstrakte Muster und beschreibt eine Aufteilung in eine abfangende Komponente, den *Policy Enforcement Point*, und eine Komponente, die über den Zugriff entscheidet, den *Policy Decision Point*. Auf OAuth übertragen kann setzt der RS einen Policy Enforcement Point um und der AzS einen Policy Decision Point.

Die Administration der Autorisationsregeln ist nicht spezielle durch ein Sicherheitsmuster dargestellt, da dies keinen direkten Bezug zum Sicherheitsniveau des Systems hat. Allerdings wird im zuvor vorgestellten Muster auch auf die Notwendigkeit eines *Policy Administration Point* eingegangen. Somit wird indirekt auf die Administration der Regeln verwiesen.

OpenID Connect kann in die Sicherheitsmusterlandschaft der Muster zur Authentifizierung eingeordnet werden, wobei es primär den Ablauf zur Identifizierung von Benutzern beschreibt, eine Identitätsschicht wie es der Standard nennt. Die Authentifizierung ergibt sich aus dem Vertrauen der Relying Party in den OpenID Provider. Das passende Sicherheitsmuster hierfür ist der *Circle of Trust* [Fern13]. Dies ist derzeit auch der Knackpunkt, warum eine Authentifizierung mit OpenID Connect meist nur als schwacher Authentifizierungsmechanismus angesehen werden kann.

Der *Authenticator* [Fern13] beschreibt abstrakt die Authentifizierung eines Benutzers, von der Authentifizierungsanfrage, über die Durchführung eines Authentifizierungsprotokolls bis zur Bestätigung der Identität mit einem *Proof Of Identity*. In verteilten Systemen spezialisiert sich dieses Muster zum *Token Authenticator* [FBYWY14], wobei als Proof Of Identity ein (Identitäts-) Token zwischen den Parteien ausgetauscht wird.

5 Ergebnisse

Durch seinen Charakter als Rahmenwerk ist OAuth offen spezifiziert und bietet diverse Stellen zur Erweiterung und Veränderung. Bei der Entwicklung des Standards wurden mehrere Ziele verfolgt, die eine Einordnung der angebotenen Funktionalität erschweren. Dennoch lassen sich einige wiederkehrende Sichten identifizieren, die je nach Anwendungsfall von den verschiedenen beteiligten Parteien eingenommen werden.

Als Nachfolger des OAuth 1.0 Protokolls liegt ein Schwerpunkt bei dem OAuth 2.0 Rahmenwerk auf der Delegation von Rechten eines Benutzers an Dienste [Bihi15, Siri14]. Dies wird durch drei der vier Autorisierungsabläufe (Grant Types) ermöglicht, “Authorization Code”, “Implicit” sowie “Resource Owner Credentials”.

Jedoch soll durch OAuth auch die Autorisierung ohne Delegation ermöglicht werden. Hierfür wird üblicherweise der durch den “Client Credentials Grant” beschriebene Ablauf zur Autorisierung genutzt. Dieser sieht keine benutzerbestimmte Autorisierung vor, sondern der Authorization Server kann dabei selbst über die Autorisierung entscheiden. Grundlage für diese Entscheidung können eine rollenbasierte Zugriffskontrolle, Zugriffsmatrizen oder andere Autorisierungsregeln sein.

Neben dieser Einordnung basierend auf der Autorisierung, kann die Sicht eines konkreten Akteurs auf OAuth 2.0 unterschiedliche Einblicke ermöglichen. Das Rahmenwerk umfasst die vier Akteurgruppen aus 2.1, die jeweils eine eigene Sicht einnehmen. Jedoch sind für die Betrachtung einer möglichen Umsetzung nur die Sichten des AzS, der die Autorisierung von Zugriffen

erlaubt, des RS, der Daten speichert und anbieten möchte, sowie des Clients, der diese konsumieren möchte, relevant. Der RO hat auf die Umsetzung keinen Einfluss.

Bei der konkreten Umsetzung des Rahmenwerks ergeben sich für diese Akteurgruppen Probleme, auf die im Standard nicht eingegangen wird. Es ist beispielsweise nicht offensichtlich, dass eine Komponente zur Verwaltung von Tokens sowohl beim AzS als auch beim RS benötigt wird. Zudem wird nicht genauer auf die Interaktion zwischen AzS und RS eingegangen.

5.1 Sichten auf die Standards

Über die bereits genannten Sichten lassen sich eine Reihe von Sicherheitsmustern ableiten, wobei eine Sicht über mehrere Muster abgedeckt werden kann. In diesem Fall existiert üblicherweise ein allgemeines Muster, das durch Änderungen an einzelnen Bestandteilen zu Varianten abgewandelt wird. Abbildung 1 zeigt die von uns identifizierten Sicherheitsmuster (gestrichelt) und deren Beziehung sowohl untereinander als auch mit bestehenden Sicherheitsmustern.

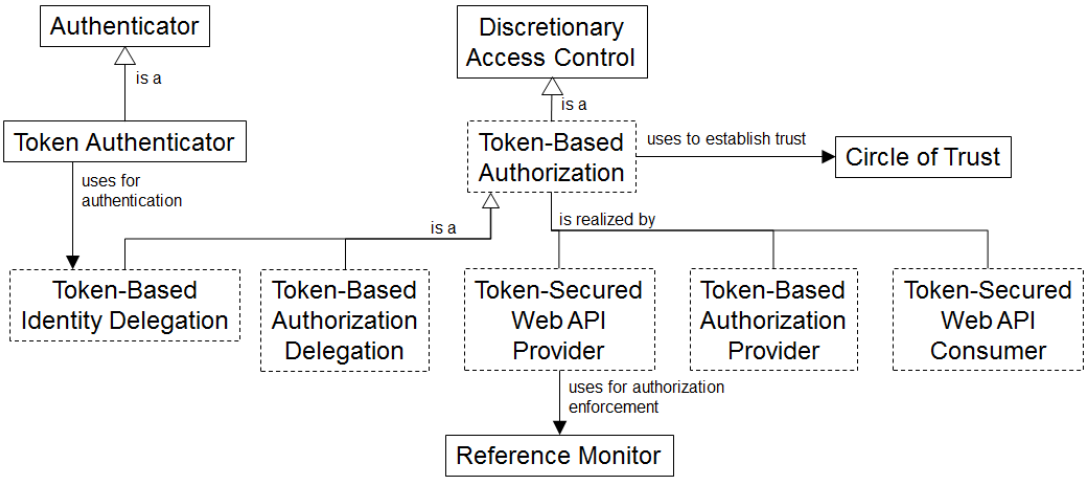

Abb. 1: Beziehung der Sicherheitsmuster zu OAuth 2.0. Gestrichelt sind die von uns ergänzten Muster.

Die *Token-Based Authorization* löst das abstrakte Problem der Autorisierung mit Tokens. Der Problemraum kann dabei in drei realisierende Sicherheitsmuster aufgeteilt werden. Der Betreiber eines RS wird beschrieben durch das Muster *Token-Secured Web API Provider*. Unter Vorlage eines gültigen Tokens gewährt er Zugriff zur API. Das Ziel des RO besteht in der Wahrung der Vertrauenswürdigkeit und Integrität der durch die API bereitgestellten Daten. Die Vergabe und Überprüfung von Tokens zur Autorisierung beschreibt das Sicherheitsmuster *Token-Based Authorization Provider*. Dieser Authorization Provider, also der Betreiber eines AzS, kann je nach verwendetem OAuth 2.0 Grant Type durch eine unterschiedliche Ausprägung beschrieben werden.

Die Sicht des Clients ist im Sicherheitsmuster *Token-Secured Web API Consumer* beschrieben. Die Entwicklersicht des Clients lässt sich dabei in zwei Varianten einteilen. Die Erste ist die eines Entwicklers einer klassischen Web-Anwendung. Die Anwendung verwaltet die Access Tokens verschiedener Benutzer und muss entsprechend über deren Zugehörigkeit zu Benutzern sowie gespeicherten Daten Buch führen. Die zweite Variante ist relevant für Entwickler einer mobilen Anwendung, oftmals als App bezeichnet, die nur Informationen des Gerätebesitzers

verwaltet und nur in dessen Namen auf APIs zugreift. Wie auch beim *Token-Based Authorization Provider* können die entwicklerbezogenen Muster darüber hinaus in die verschiedenen Grant Types aufgeteilt werden.

5.2 Beispielmuster: Token-Based Authorization Provider

In diesem Abschnitt wird beispielhaft ein Auszug des Sicherheitsmusterkandidaten Token-Based Authorization Provider vorgestellt. Als zentraler Vertrauensanker beschreibt der **Token-Based Authorization Provider** die Abläufe zur Autorisierung des Zugriffs auf Ressourcen. Der Provider beschreibt die Mittlung zwischen Client und Anbieter (RS) einer API, die den Zugriff auf Ressourcen bereitstellt.

Kontext: Web-APIs, die Ressourcen von Benutzern unternehmensfremden Services bereitstellen. Die API soll verteilt per Token geschützt werden. Ein Vertrauensverhältnis zwischen den Parteien besteht.

Problem: Von einer Web-API bereitgestellte Ressourcen eines Benutzers sollen vor unbefugtem Zugriff geschützt werden. Andere Services sollen auf die Ressourcen zugreifen können. Die Informationen zur Authentifizierung des Benutzers sollen nicht an den zugreifenden Service weitergegeben werden. Folgenden Zwängen müssen durch die Lösung angegangen werden:

- **Vertraulichkeit und Integrität:** Nur befugte Personen sollen Zugriff auf die API und somit die Ressourcen erhalten.

- **Besitzerbestimmt:** Der RO soll die Kontrolle über seine Daten haben.

- **Performanz:** Die Vergabe und Verifizierung von Token soll performant sein und auch bei großen Nutzerzahlen skalieren.

- **Sicherheit:** Token müssen sicher abgelegt und übertragen werden und nur für Befugte zugänglich sein.

- **Vertrauen:** Zwischen Client, RS und RO muss ein Vertrauensverhältnis bestehen.

Lösungsskizze: Die Strukturbeschreibung anhand des Klassendiagramms (Abbildung 2) verdeutlicht den Aufbau eines AzS und zugehöriger Bestandteile. Zusätzlich zu den im Standard beschriebenen Komponenten ist jeweils ein Speicherort für Clients, Benutzer und Tokens ergänzt. Der AzS delegiert eine Authentifizierung von Client und Benutzer an den Authenticator, um dem Client anschließend einen Token samt autorisiertem Scope zuzuweisen. Ein AzS kann durch beliebig viele RS genutzt werden.

5.3 Einsatz der Sicherheitsmusterkandidaten in der SmartCampus-Webanwendung

Der SmartCampus ist eine serviceorientierte Webanwendung, die Gästen, Partnern, Mitarbeitern und Studierenden smarte Services zur Unterstützung der Aktivitäten auf dem Campus anbieten soll. Zu den Services gehören eine Lern- bzw. Arbeitsplatzsuche, die Suche und Navigation zu Points of Interest und Informationsdienste für Studierende mit Behinderung. Er wird entwickelt in der Forschungsgruppe Cooperation & Management am Karlsruher Institut für Technologie. Authentifizierung und Zugriffskontrolle sind für den Erfolg der Anwendung, die viele sensible Benutzerdaten erfasst, ein wichtiger Faktor.

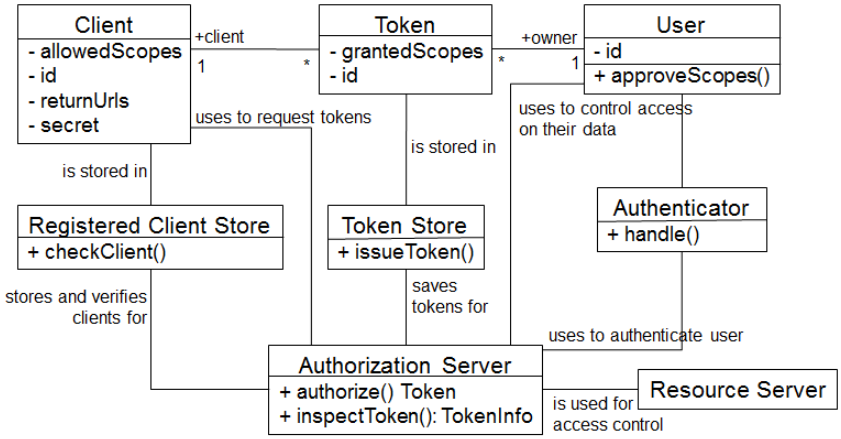

Abb. 2: Struktur des Sicherheitsmusterkandidaten Token-Based Authorization Provider

Sowohl OpenID Connect als auch OAuth kommen beim SmartCampus zum Einsatz. OpenID Connect wird bei der Authentifizierung der Benutzer über Servicegrenzen hinweg eingesetzt. Hierfür ist ein eigener Service Provider aufgesetzt, der verschiedene Authentifizierungsmethoden kapselt. Über ihn wird sowohl die Authentifizierung über das zentrale, SAML-basierte Shibboleth-System als auch die Anmeldung von Dingen aus dem Internet of Things, wie beispielsweise Räumen, ermöglicht. Nach erfolgreicher Authentifizierung erhält der Anmeldende einen ID Token (im JSON Web Token (JWT) Format) gemäß OpenID Connect Spezifikation.

OAuth wird innerhalb der Anwendung zur Absicherung der Service-APIs genutzt. Die RS bieten eine an REST angelehnte (kein HATEOS-Prinzip) API an. Da es sich um eine interne API handelt, wird der Client Credential Grant Type verwendet, bei dem sich der Client, in unserem Fall die Webanwendung im Browser, beim AzS die Erlaubnis zum Zugriff auf die API des RS einholt. Zur Authentifizierung schickt der Client den ID Token mit seinen OpenID Connect Anmeldedaten an den AzS. Die Überprüfung der Autorisierung ist simpel gehalten. Einzige Scopes sind *User* und *Administrator*. Zudem sind die Ressourcen in *zentralverwaltete* (Räume etc.) und *benutzerverwaltete* (Freunde etc.) eingeteilt. *User* können *zentralverwaltete* Ressourcen lesen und durchsuchen und haben Vollzugriff auf *benutzerverwaltete* Ressourcen, die dem Benutzer gehören. Der Administrator hat Vollzugriff auf *zentralverwaltete* Ressourcen und darf ebenfalls nur die eigenen *benutzerverwalteten* Ressourcen bearbeiten.

Die API des AzS zeigt Tabelle 1. Sie ist aus Abbildung 2 abgeleitet und nach den API-Richtlinien für RESTful Web Services [GGSS+15] entworfen.

Tab. 1: REST API des SmartCampus Authorization Servers

Pfad	HTTP-Methode	Beschreibung
/authorizations	GET	Liste der Token des Benutzers
/authorizations	POST	OAuth-Autorisierungsanfrage
/authorizations/{accessToken}	GET	Zur Verifikation des Token
/authorizations/{accessToken}	PUT	Erzeugt eine Fehlermeldung
/authorizations/{accessToken}	DELETE	Invalidiert den Token beim AzS

5.4 Einschränkungen

Die eingeführten Sicherheitsmuster sind als Kandidaten zu verstehen. Diese Arbeit soll als Grundlage für die Abstraktion der Standards hin zur Beschreibung als wiederkehrende Lösungen im Sicherheitsbereich dienen und so in der Entwurfsphase unterstützen. Unser Ziel ist es, die Konzepte hinter verbreiteten Standards auf ein abstraktes Niveau zu bringen, so dass sie in frühen Phasen der Softwareentwicklung mit einbezogen werden können.

6 Zusammenfassung und Ausblick

Die Standards OAuth und OpenID Connect sind heute zu wichtigen Bausteinen des Internets geworden - sie erlauben die Integration und Interoperation von unzähligen Web- und mobilen Diensten. Beide Spezifikationen folgen dem Mantra "Keep simple things simple, make complex things possible" und bieten dem Entwickler damit nahezu unbegrenzte Möglichkeiten.

In dieser Arbeit haben wir an aktuellen Projekten den möglichen Einsatz von OAuth 2.0 und OpenID Connect in Unternehmen gezeigt. An Hand der Beispiele wurden Probleme und Schwächen, die auf Grund der Beschreibung als Rahmenwerk entstehen aufgezeigt. Als Lösung für diese Probleme auf konzeptioneller Ebene haben wir einen Ansatz zur Beschreibung der Standards als Sicherheitsmuster vorgestellt und konkret den Sicherheitsmusterkandidaten Token-basierter Autorisierer eingeführt.

Softwarearchitekten und -Entwickler können diese Musterkandidaten benutzen, bspw. beim Entwurf der Architektur, da die benötigten Komponenten offensichtlich sind und die zusätzliche Komplexität abgeschätzt werden kann, als Grundlage für eine einheitliche Terminologie im Team und um bestehende Systeme bei einem Technologiewechsel besser einordnen zu können.

Aufbauend auf unserem Beitrag können auf verschiedenen Ebenen Forschungsarbeiten anschließen. Die hier vorgestellten Sicherheitsmusterkandidaten bieten eine Grundlage für die finale Beschreibung der Standards als Sicherheitsmuster. Neben dieser konzeptionellen Ergänzungen haben die aufgezeigten Praxisbeispiele auch Fragen der Sicherheit auf Implementierungsebene aufgeworfen. Auch diese sind größtenteils aus den Standards nicht abzusehen und geeignete Wege zur Beschreibung könnten die Arbeit von Softwarearchitekten und -Entwicklern unterstützen.

Literatur

[Bihi15] C. Bihis: Mastering OAuth 2.0. Packt Publ. (2015).

[CaMJ15] B. Campbell, C. Mortimore, M. Jones: Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants. Rfc, Internet Engineering Task Force (IETF) (2015).

[CCMW01] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana: Web services description language (WSDL) 1.1. Tech. Rep., W3C recommendation (2001).

[ChJR11] S. Chari, C. S. Jutla, A. Roy: Universally Composable Security Analysis of OAuth v2. 0. In: *IACR Cryptology ePrint Archive*, 2011 (2011), 526.

[CKPM05] S. Cantor, J. Kemp, R. Philpott, E. Maler: Assertions and protocols for the oasis security assertion markup language. Standard, Organization for the Advancement of Structured Information Standards (OASIS) (2005).

[FBYWY14] E. Fernandez-Buglioni, N. Yoshioka, H. Washizaki, J. Yoder: Abstract security patterns for requirements specification and analysis of secure systems. In: *CIBSE 2014: Proceedings of the 17th Ibero-American Conference Software Engineering* (2014), 437–450.

[Fern13] E. Fernandez-Buglioni: Security patterns in practice: designing secure architectures using software patterns. John Wiley & Sons (2013).

[GGSS+15] P. Giessler, M. Gebhart, D. Sarancin, R. H. Steinegger, S. Abeck: Best Practices for the Design of RESTful Web Services. *In: The Tenth International Conference on Software Engineering Advances (ICSEA 2015), Barcelona, Spain* (2015).

[GHMM+03] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, Y. Lafon: SOAP Version 1.2. Tech. Rep., W3C recommendation (2003).

[Hard12] D. Hardt: The OAuth 2.0 Authorization Framework. RFC, Internet Engineering Task Force (IETF) (2012).

[OAu07] OAuth 1.0 Core. Standard, OAuth Core Workgroup (2007), .

[PSKP+11] S. Pai, Y. Sharma, S. Kumar, R. M. Pai, S. Singh: Formal verification of oauth 2.0 using alloy framework. *In: Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, IEEE (2011), 655–659.

[SaSa94] R. S. Sandhu, P. Samarati: Access control: principle and practice. In: *IEEE Communications Magazine*, 32, 9 (1994), 40–48.

[SBJM+14] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, C. Mortimore: OpenID Connect Core 1.0. Standard, The OpenID Foundation (2014).

[Schu03] M. Schumacher: Security engineering with patterns: origins, theoretical models, and new applications, Bd. 2754. Springer (2003).

[SFBHB+06] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, P. Sommerlad: Security Patterns - Integrating Security and Systems Engineering. Wiley Series in Software Design Patterns (2006).

[Siri14] P. Siriwardena: Advanced API Security: Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE. Apress (2014).

[SSVA14] R. Steinegger, J. Schäfer, M. Vogler, S. Abeck: Attack Surface Reduction for Web Services based on Authorization Patterns. In: *The Eighth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2014)* (2014), 194–201.

[SuBe12] S.-T. Sun, K. Beznosov: The devil is in the (implementation) details: an empirical analysis of oauth sso systems. *In: Proceedings of the 2012 ACM conference on Computer and communications security*, ACM (2012), 378–390.

[WeWA15] P. Weierich, D. Weich, S. Abeck: Identitäts- und Zugangsmanagement für Kundenportale - Eine Bestandsaufnahme. *In: Digital Enterprise Computing (DEC 2015)*, Lecture Notes in Informatics (LNI) - Proceedings, Gesellschaft für Informatik (2015), 111–115.

[YaMa13] F. Yang, S. Manoharan: A security analysis of the OAuth protocol. *In: Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on* (2013), 271–276.