Was bin ich

SAML 2.0, ein Tutorium – Teil 1: Theorie

Michael Kain, Guido Keller

Die Security Assertion Markup Language 2.0 (SAML) ermöglicht den Austausch von Authentifizierungs- und Autorisierungsinformationen über Organisationsgrenzen hinweg. Trotz der umfangreichen OASIS-Spezifikation ist ein schneller Einstieg in die SAML 2.0 möglich. Dieses zweiteilige Tutorium beginnt mit einer allgemeinen, theoretischen Einführung und endet im zweiten Teil mit einem anschaulichen Beispiel.

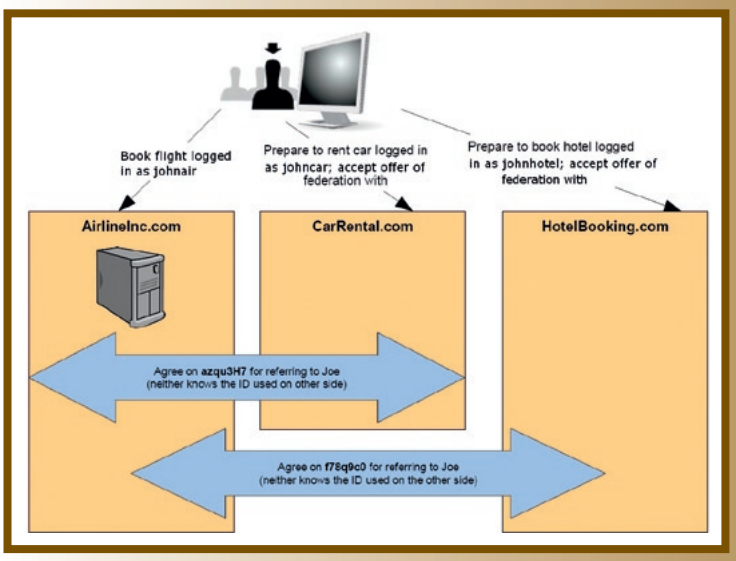

Abb. 1: Identity Federation

Motivation

Die *Security Assertion Markup Language* 2.0 (SAML) ist eine auf XML basierende Auszeichnungssprache für den Austausch von Authentifizierungs- und Autorisierungsinformationen über Organisationsgrenzen hinweg. Dies ist beispielsweise wichtig bei firmenübergreifenden Projekten, die im Bereich des Wertschöpfungsketten-Managements oder der Auslagerung von Diensten angesiedelt sind. Unter dem Dach der *Organization for the Advancement of Structured Information Standards* (OASIS) definiert die SAML ein „XML-basiertes Framework zur Beschreibung und zum Austausch von Sicherheitsinformationen zwischen Online-Geschäftspartnern". Dieses kann nahtlos in eine bestehende Webservices-Architektur integriert werden.

Wie diese sehr allgemeine Definition schon vermuten lässt und wie es für viele Standards heute üblich ist, enthält die SAML 2.0 eine Vielzahl von Dokumenten und Seiten. Das *Security Services Technical Commitee* (SSTC) hat die SAML 2.0 spezifiziert und ist Teil der OASIS. Wenn man sich auf der Homepage des SSTC unter [OasisSSTC] umsieht und die komplette SAML 2.0-Spezifikation herunterlädt, dann erhält man ungefähr 350 Seiten in Form von .pdf-Dateien sowie einige zusätzliche XML-Schema-Definitionen.

Viele Interessierte könnte diese schiere Masse an Seiten erst einmal abschrecken. Trotz und gerade wegen all dieser Dokumente ist ein Einstieg in die SAML 2.0 aber mehr als lohnenswert. Alle elementaren Begriffe und Komponenten sind ausreichend dokumentiert und erklärt. Außerdem ist es allein mit Hilfe des *Web Browser SSO Profile* möglich, eine Single Sign-On(SSO)-Lösung zwischen ansonsten völlig getrennten Unternehmensportalen zu realisieren – und dies alles auf Basis eines bekannten Standards.

Ein schneller Einstieg in die SAML 2.0 ist möglich und soll mit diesem Tutorium erleichtert werden. Es beginnt daher in Teil 1 mit einer allgemeinen, theoretischen Einführung und führt diese in Teil 2 mit einem sehr plastischen Beispiel fort: einem Domain-übergreifenden, Browser-basierten Single Sign-On-Beispiel auf Basis der SAML 2.0.

An dieser Stelle sei der Leser darauf verwiesen, dass das SSTC ebenfalls zwei zusätzliche Übersichtsdokumente zum Download anbietet, die für einen ersten Einstieg sehr hilfreich sind: den *Executive Overview* und den *Technical Overview*. Dieses Tutorium soll eine erste Brücke zwischen all diesen Materialien des SSTC schlagen.

Das große Bild

Die SAML 2.0 kann hauptsächlich für zwei Anwendungsfälle eingesetzt werden: zum einen für *Web Single Sign-On* und darüber hinaus für *Identity Federation*. In beiden Fällen tritt eine Geschäftsentität A als Identity Provider (*IdP*) für eine andere Geschäftsentität B, den sogenannten Service Provider (*SP*), auf. A bietet somit B Informationen, die die Identität der Subjekte von A betreffen und B dabei unterstützen seine Dienste anzubieten. Welche Art von Informationen über die Subjekte jedoch von A an B übermittelt wird, ist jeweils – meist vertraglich – zwischen den Geschäftsentitäten zu vereinbaren und kann deshalb sehr flexibel in SAML 2.0 abgebildet werden.

Im Kontext der SAML bedeutet dies, dass der Identity Provider in der Rolle der Asserting Party (*AP*) auftritt und der Service Provider in der Rolle der Relying Party (*RP*). Die Asserting Party, die auch als SAML Authority bezeichnet wird, bescheinigt der Relying Party die Identität eines Subjektes. Dies geschieht durch die Übermittlung einer Zusicherung (*SAML Assertion*) von der AP zur RP, wobei vorausgesetzt wird, dass die RP der Zusicherung der AP vertraut.

Das nachfolgende Beispiel, das in Abbildung 1 dargestellt ist, veranschaulicht ein mögliches Szenario für Identity Federation und ist ursprünglich Teil des Technical Overview, wurde aber für diesen Artikel deutlich modifiziert. Alle drei in Abbildung 1 dargestellten Portale – AirlineInc.com, CarRental.com und HotelBooking.com – werden von unterschiedlichen Firmen betrieben und verwalten juristisch getrennte Kundendaten. Nun aber haben sich die drei Firmen vertraglich geeinigt, lose miteinander zu kooperieren, um das gemeinsame Geschäft zu steigern. Das Ziel soll es sein, dass ein Benutzer von AirlineInc.com sowohl zu CarRental.com als auch zu HotelBooking.com im Web navigieren kann, ohne sich erneut authentifizieren zu müssen. Die Benutzerfreundlichkeit der dabei beteiligten Portale soll damit deutlich verbessert werden.

Dies wird technologisch mit Hilfe von SAML 2.0 und Identity Federation realisiert – in Form von Account Linking auf Basis von persistenten Pseudonymen. Da-

Abb. 2: SAML 2.0-Komponenten

Statements	Kurzbeschreibungen
Authentication `<AuthnStatement>`	Definiert, dass und wie eine Authentifizierung stattgefunden hat.
Attribute `<AttributeStatement>`	Enthält eine beliebige Anzahl von Attributen, in denen die Eigenschaften eines Subjektes festgelegt werden. Hier kann z. B. hinterlegt werden, ob ein Benutzer Privat- oder Geschäftskunde ist.
Authorization Decision `<AuthzDecisionStatement>`	Definiert, welche Rechte ein Subjekt hat. Typischerweise sind dies Rechte, mit denen der SP entscheiden kann, welche Aktionen ein Benutzer in seinem System ausführen kann. Hier kann z. B. das Recht hinterlegt werden, Waren einkaufen zu dürfen.

Tabelle 1: SAML 2.0-Statement-Typen

bei tritt in diesem Szenario die AirlineInc als Asserting Party (*AP*) und Identity Provider (*IdP*) auf, wohingegen CarRental und HotelBooking beide jeweils Relying Party (*RP*) und Service Provider (*SP*) sind.

Ein Subjekt, in diesem Fall der menschliche Benutzer einer Web-Anwendung mit dem Namen Joe, hat drei unterschiedliche Accounts bei allen drei Web-Portalen. Diese hat Joe sich bereits vor der Identity Federation, also vor der Kooperationsvereinbarung der drei Firmen, selbst angelegt und unabhängig voneinander verwendet. Seine drei Account-Namen lauten dementsprechend: johnair, johncar und johnhotel. Für alle drei Accounts verwalten die zugehörigen Portale voneinander getrennte Kunden- und Rechnungsinformationen. Daran soll sich auch nichts ändern. Dies bedeutet beispielsweise, dass in diesem Szenario der Identity Federation dem Kunden weiterhin seine Account-bezogenen Leistungen getrennt in Rechnung gestellt und abgerechnet werden.

Joe ist bei *AirlineInc.com* als *johnair* eingeloggt und hat einen Flug gebucht. Im Anschluss daran klickt er auf einen Link im AirlineInc-Portal. Dieser führt ihn zu *CarRental.com*, um einen Leihwagen zu buchen. In diesem Moment erzeugt *AirlineInc.com* im Hintergrund für Joe das Pseudonym *azqu3H7* und überträgt dieses in einer signierten SAML Assertion, einer definierten XML-Struktur, an *CarRental.com*. Diese Übertragung läuft in diesem Fall über Joes Browser, aber völlig transparent für Joe. Die SAML Assertion ist in einem Web-Formular enthalten, das mittels OnLoad-Submit direkt weiter zu *CarRental.com* geschickt wird. Der Benutzer muss dieses nicht selbst versenden. Das CarRental-Portal erkennt die signierte SAML Assertion und liest das darin enthaltene Pseudonym aus. Es fragt Joe nach dessen Einloggen, ob er einer *Identity Federation* zwischen beiden Portalen zustimmen würde. Wird dieser durch Joe zugestimmt, verknüpft *CarRental.com* das Pseudonym *azqu3H7* mit dem Account *johncar* und persistiert dieses. Über einen anderen, Browser-unabhängigen Dienst im Hintergrund informiert *CarRental.com* anschließend *AirlineInc.com* über die Zustimmung des Account Linking für *azqu3H7*. *AirlineInc.com* persistiert daraufhin ebenfalls das Pseudonym und verknüpft dieses mit *johnair*. Nach seiner erfolgten Zustimmung muss sich Joe bei oben beschriebenem Szenario nicht mehr erneut bei *CarRental.com* einloggen, da das Account Linking abgeschlossen ist.

Der gleiche Prozess des Account Linking kann von *AirlineInc.com* zu *HotelBooking.com* wiederholt werden, aber eben mit einem anderen Pseudonym, wie beispielsweise *f78q9c0*. Kein Unternehmen hat bis zu diesem Zeitpunkt einen Account-Namen, noch irgendwelche anderen vertraulichen Kundendaten herausgegeben.

Wie in so einem Szenario die *Pseudonyme* aufgebaut sind, kann – was SAML 2.0 betrifft – frei von den beteiligten Unternehmen definiert werden. Das Gleiche gilt für die Realisierung des Hintergrunddienstes zum Austausch der Account Linking-Zustimmungen. Dieser könnte beispielsweise mit Hilfe einer geschützten Webservices-Kommunikation realisiert werden.

Assertions

Der SAML 2.0-Standard besteht aus den aufeinander aufbauenden Komponenten, die in Abbildung 2 dargestellt sind. Im Folgenden werden diese einzelnen Komponenten und ihr Zusammenspiel näher beschrieben.

Eine Assertion bestätigt die Authentifizierung eines Subjektes (*Subject*), die einem SP (*Audience*) ein IdP (*Issuer*) garantiert, und auf welche Art und Weise dieses authentifiziert wurde. Außerdem können Eigenschaften über ein Subjekt in einer Assertion abgelegt werden, die dem IdP bekannt sind und die dieser dem SP mitteilen will. Eine Assertion kann bis zu drei verschiedene Statement-Typen enthalten, die in Tabelle 1 aufgeführt sind.

Als Beispiel für die Statements aus Tabelle 1 soll die Assertion in Listing 1 betrachtet werden. Die Assertion wurde erzeugt vom IdP *AirlineInc.com* (Element `<Issuer>`) zum Zeitpunkt t1 (Attribut `IssueInstant`). Authentifiziert wurde der Benutzer mit der E-Mail-Adresse *azqu3H7@airlineinc.com* (Element `<Subject>`). Der Gültigkeitszeitraum der Assertion ist im Element `<Conditions>` abgelegt. Diese Authentifizierung ist gültig vom Zeitpunkt t2 bis zum Zeitpunkt t3 (Attribute `NotBefore` und `NotOnOrAfter`) und für den SP *CarRental.com* bestimmt (Element `<Audience>`).

Im Element `<AuthnStatement>` ist im Attribut `AuthnInstant` ein Zeitstempel t4 gesetzt, der besagt, wann sich der Benutzer authentifiziert hat. Im Element `<AuthnContext>` und dem Sub-Element `<AuthnContextClassRef>` ist die Art, wie der Benutzer sich authentifiziert hat, abgelegt. In diesem Fall hat er sich mit seinem Passwort authentifiziert.

Im Element `<AttributeStatement>` ist ein Sub-Element `<Attribute>` mit Namen `„CreditLimit“` abgelegt. Hier können beliebig viele Sub-Elemente vom Typ `<Attribute>` abgelegt werden. Der Wert des Attributes ist 500. Der SP interpretiert dies so, dass dem Benutzer ein Kreditlimit von 500 Euro eingeräumt wird. Die Interpre-

```
<Assertion Version="2.0"
    IssueInstant="2007-01-22T12:00:00.00+01:00" ID="11111">
  <Issuer>AirlineInc.com</Issuer>

  <Subject>
    <NameID Format=
      "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
    >azqu3H7@airlineinc.com</NameID>
  </Subject>

  <Conditions
    NotOnOrAfter="2007-01-22T12:05:00.00+01:00"
    NotBefore="2007-01-22T11:31:30.048+01:00">
    <AudienceRestriction>
      <Audience>CarRental.com</Audience>
    </AudienceRestriction>
  </Conditions>

  <AuthnStatement AuthnInstant="2007-01-22T12:00:00.00+01:00">
    <AuthnContext>
      <AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:Password
      </AuthnContextClassRef>
    </AuthnContext>
  </AuthnStatement>

  <AttributeStatement>
    <Attribute NameFormat="urn:x-aol:attribute"
               Name="CreditLimit">
      <AttributeValue xsi:type="ns8:number"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ns8="http://www.w3.org/2001/XMLSchema"
      >500</AttributeValue>
    </Attribute>
  </AttributeStatement>

  <AuthzDecisionStatement
    Resource="http://www.airlineinc.com/viewbill.html"
    Decision="Permit">
  </AuthzDecisionStatement>

</Assertion>
```

Listing 1: SAML 2.0 Assertion

```
...
<samlp:Response
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  Version="2.0"
...
>
  <saml:Issuer>http://www.AirlineInc.com</saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value=
      "urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>

  <saml:Assertion>
  ...SAML assertion content...
  </saml:Assertion>
</samlp:Response>
...
```

Listing 2: Authentication Request-Protokoll – Response

tation und die Anzahl der Attribute sind nicht durch SAML 2.0 vorgegeben. Dies muss von SP und IdP untereinander festgelegt werden.

Im Element `<AuthzDecisionStatement>` ist ein Recht hinterlegt. In diesem Fall bekommt der Benutzer das Recht eingeräumt, die Seite `http://www.airlineinc.com/viewbill.html` aufzurufen.

Protocols	Kurzbeschreibungen
Assertion Query and Request	Eine RP fordert bei einer AP entweder eine existierende Assertion mit Hilfe einer Assertion ID an oder bittet um eine oder mehrere Assertions – mit Hilfe von vier Abfragetypen.
Authentication Request	Ein Subjekt möchte bei einer RP authentifiziert werden. Die RP stellt dazu eine Anfrage an eine AP. Diese bestätigt der RP die Authentifizierung des anfragenden Subjektes.
Artifact Resolution	Eine RP stellt eine Anfrage an eine AP. Diese dient dazu, Referenzen auf SAML-Nachrichten aufzulösen, die beim HTTP Artifact Binding übertragen werden. Die AP gibt die SAML-Nachrichten zurück, die zu den empfangenen Referenzen (Artifacts) passen.
Name Identifier Management	Der Name Identifier für ein Subjekt, wie z. B. dessen Benutzername, hat sich geändert. Mit Hilfe dieses Protokolls können solche Änderungen zwischen AP und RP ausgetauscht werden.
Name Identifier Mapping	Mit Hilfe dieses Protokolls kann eine AP aufgerufen werden, die für zwei RPs gültig ist. Diese mappt daraufhin die unterschiedlichen Name Identifier zwischen den beiden RPs.
Single Logout	Dieses Protokoll ermöglicht einen (beinahe-)simultanen Logout von Subjekten mit aktiven Sessions bei mehreren Partys.

Tabelle 2: SAML 2.0-Protokolle

Protocols

Der SAML 2.0-Standard spezifiziert sechs Request-Response-Protokolle. Alle Nachrichten, die in diesen Protokollen übertragen werden, sind dementsprechend von den gemeinsamen Eltern-Elementtypen `RequestAbstractType` oder `StatusResponseType` abgeleitet. Im nachfolgenden Bindings-Kapitel ist beschrieben, mit Hilfe welcher Transportschicht diese Nachrichten zwischen AP und RP transferiert werden. Tabelle 2 enthält eine Auflistung aller Protokolle sowie deren Kurzbeschreibungen.

An dieser Stelle wird einzig das Authentication Request-Protokoll näher betrachtet, da es einen Baustein im *Web Browser SSO Profile* darstellt, das später im Kapitel Profiles genauer beschrieben wird. Wenn ein SP eine Authentifizierungsanfrage an einen IdP stellt (*SP-initiated*), schickt er einen `<AuthnRequest>` an einen IdP, den dieser mit einem `<Response>` beantworten muss. Die im `<Response>` enthaltene Assertion muss ein `<AuthnStatement>` enthalten, kann aber optional `<AttributeStatement>`- bzw. `<AuthzDecisionStatement>`-Elemente enthalten. Für den einfacheren Fall, dass sich ein Subjekt bereits beim IdP authentifiziert hat (*IdP-initiated*) und anschließend beim SP authentifiziert werden soll, ist es ausreichend, wenn an den SP nur ein `<Response>` mit der notwendigen Assertion übertragen wird. Listing 2 stellt exemplarisch einen SAML 2.0 `<Response>` dar.

Bindings

Bindings definieren den Transport von SAML-Nachrichten über Standard-Protokolle. Als Transportschicht gibt es zur Zeit HTTP und SOAP. Tabelle 3 enthält die Bindings, die in SAML 2.0 definiert sind.

Als Beispiel soll hier das *HTTP Post Binding* herausgegriffen werden. Dieses Binding nutzt den Browser des Benutzers zum Transport der SAML-Nachrichten (s. Abb. 3). Es ist ebenfalls Teil des Web Browser SSO Profile. Der Benutzer ruft eine Webseite des Identity Provider *www.xyz.com* auf:

1: Die Webseite zeigt eine Login-Maske und bittet den Benutzer sich einzuloggen.
2: Der Benutzer loggt sich mit Benutzername und Passwort ein.
3: Nach einem erfolgreichen Login sieht der Benutzer ein Angebot an Ressourcen. Diese Ressourcen werden nicht vom IdP gehostet. Der Identity Provider bietet lediglich die Auswahl an. Der Benutzer wählt eine Ressource aus.
4: Der Single Sign-On-Service des IdP erzeugt eine SAML `<Response>` mit einer darin enthaltenen SAML `<Assertion>`. Die `<Response>` wird digital signiert. Die `<Response>` und die digitale Signatur werden nach dem Base64-Verfahren kodiert als hidden-fields in einem Onload-Submit-HTML-Formular abgelegt. Dieses Formular wird an den Browser des Benutzers ausgeliefert.
5: Nachdem das Formular an den Browser ausgeliefert wurde, wird es sofort an die Adresse „www.abc.com“ des SP gepostet. Der Benutzer bekommt das Formular nicht zu sehen.
6: Der Asssertion Consumer Service des SP prüft zunächst die digitale Signatur gegen die SAML `<Response>`. Danach wird die SAML `<Assertion>` ausgewertet.
7: Ist die Autorisierung gültig, bekommt der Benutzer Zugriff auf die angeforderte Ressource. Anderenfalls würde der Benutzer eine Login-Maske des SP zu sehen bekommen oder an den IdP zurückgeleitet werden.

Profiles

Die SAML 2.0 definiert insgesamt dreizehn Profile. Diese sind in Abbildung 4 zusammenfassend dargestellt. Zehn Profile lassen sich einer der zwei Kategorien zuordnen: den *Attribute Profiles* oder den *SSO Profiles*. Im Gegensatz zu anderen Standards ist der Begriff Profil in SAML 2.0 aber nicht eindeutig.

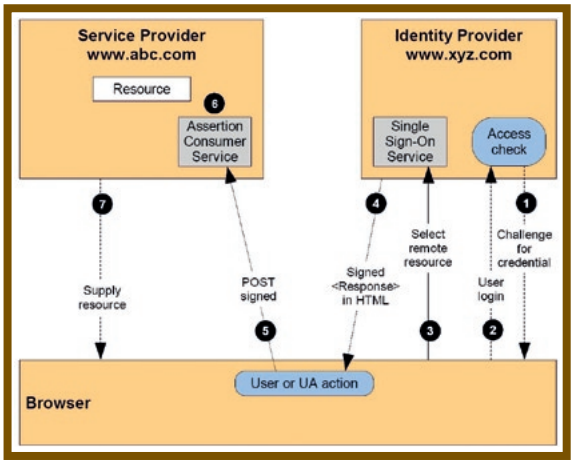

Abb. 3: SAML 2.0 HTTP Post Binding

Bindings	Kurzbeschreibungen
SAML SOAP	Transport über Webservices mit SOAP 1.1
Reverse SOAP (PAOS)	Mehrstufiger Austausch von SAML-Nachrichten über SOAP 1.1
HTTP Redirect	SAML-Nachrichten werden über HTTP Redirect transportiert.
HTTP POST	SAML-Nachrichten werden base64-encoded in einem HTML-Formular abgelegt und über Onload-Submit an die RP gepostet.
HTTP Artifact	Es wird nur die Referenz (call-by-reference) auf eine SAML-Nachricht (Request oder Response) transportiert, nicht die Nachricht selbst. Dies findet Verwendung vor allem bei Smart-Clients, die keine vollständigen SAML-Nachrichten übertragen können. Die Referenz kann in einem URL-Parameter oder in einem HTML-Formular abgelegt werden, sodass die RP unabhängig vom Smart-Client bei der AP – mit Hilfe der Referenz und des SAML SOAP Bindings – die erwartete SAML-Nachricht abholen kann.
SAML URI	Es wird nur eine URI auf eine SAML Assertion übertragen. Dieses Binding ist unabhängig von SAML-Nachrichten (Request oder Response), es dient allein dazu, dass bei Anfragen an diese URI in dem Element `<saml:AssertionIDRef>` eine SAML Assertion zurückgegeben wird.

Tabelle 3: SAML 2.0 Bindings

Zum einen versteht man darunter – wie in anderen Standards – eine Kombination beziehungsweise Einschränkung der dem Standard zugrundeliegenden Komponenten für einen bestimmten *Anwendungsfall*. Die fünf SSO Profiles und die drei einzelnen Profile sind ein Beispiel hierfür. Sie kombinieren *Assertions*, *Protocols* und *Bindings*, um beispielsweise eine Browser-basierte Single Sign-On-Lösung auf Basis von SAML 2.0 zu ermöglichen.

Zum anderen aber versteht man in SAML 2.0 unter einem Profil einen definierten Satz von Regeln zur *Abbildung von Attributen* aus anderen Systemen

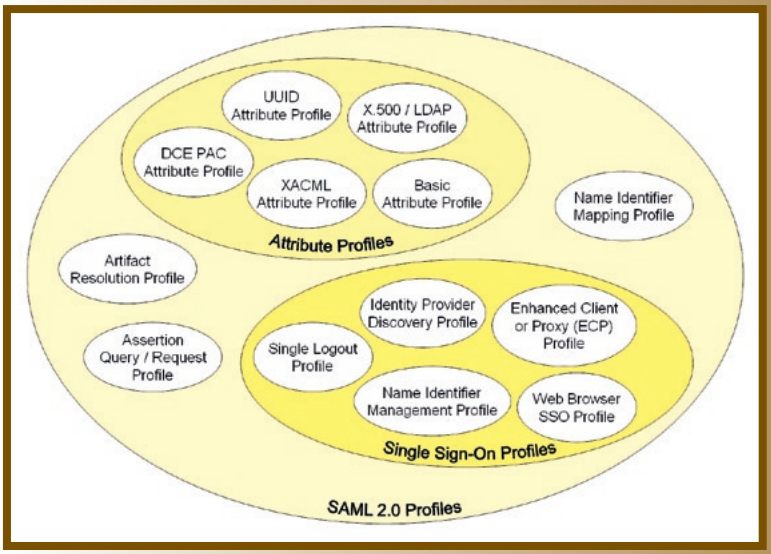

Abb. 4: SAML 2.0-Profile

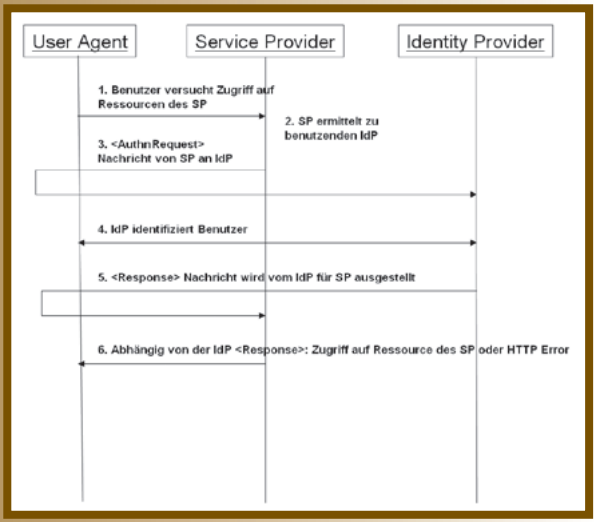

Abb. 5: SAML 2.0 Web Browser SSO Profile

```
<saml:Assertion>
...
  <saml:Attribute
    xmlns:xacmlprof=
      "urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"
    xacmlprof:DataType="http://www.w3.org/2001/XMLSchema#string"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    Name="urn:oid:2.5.4.42" FriendlyName="givenName">

    <saml:AttributeValue
      xsi:type="xs:string"
    >By-Tor</saml:AttributeValue>
  </saml:Attribute>
  ...
</saml:Assertion>
```

Listing 3: XACML Attribute Profile Statement in Assertion

nach SAML. Dies bedeutet, dass innerhalb der Attribute Profiles Regeln definiert sind, wie beispielsweise Attribute aus einem LDAP-System in einer SAML *Assertion* als Attribute abgebildet werden können. Dies ist spezifiziert im *X.500/LDAP Attribute Profile*. Die Protocols und Bindings spielen also in den Attribute Profiles keine Rolle.

Am anschaulichsten verstehen kann man die zwei unterschiedlichen Profiltypen innerhalb von SAML 2.0 anhand zweier repräsentativer Beispiele. So werden im Folgenden das *Web Browser SSO Profile* und das *XACML Attribute Profile* näher erläutert.

SSO Profiles: Web Browser SSO Profile

Dieses Profil wird verwendet, wenn ein Benutzer im Web auf eine Ressource zugreifen will, die von einem Service Provider angeboten wird (s. Abb. 5, *Schritt 1*). Für die Authentifizierung ist ein Identity Provider notwendig. Entweder wird der Benutzer direkt vom IdP zum SP weitergeleitet (*Schritte 5 + 6, IdP-initiated*), nachdem er sich beim IdP authentifiziert hat, oder der SP schickt eine Authentifizierungsanfrage an den IdP (*Schritt 3, SP-initiated*), die dieser beantwortet (*Schritt 4*). Nach erfolgreicher Authentifizierung beim IdP gelangt der Benutzer wieder zum SP und erhält Zugriff auf die gewünschte Ressource (*Schritte 5 + 6*).

Dieses Profil ist an die Bindings HTTP Redirect, HTTP Post und HTTP Artifact gebunden, die auf Basis des Protokolls Authentication Request ausgetauscht werden. Beim SAML-spezifischen HTTP Artifact Binding wird an den User Agent vom IdP nur eine Referenz auf eine SAML-Nachricht übergeben, anstatt einer SAML-Nachricht selbst. Diese Referenz kann der SP anschließend dazu verwenden, sich die referenzierte SAML-Nachricht selbst beim IdP abzuholen.

Attribute Profiles: XACML Attribute Profile

Mit Hilfe dieses Profils werden XACML-basierte Informationen innerhalb von SAML 2.0 Assertions übertragen. Die XACML (eXtensible Access Control Markup Language) ist ein XML-basierter Standard, mit dessen Hilfe Policy-basierte Zugriffsentscheidungen auf Ressourcen definiert und evaluiert werden können. Der SP muss in der Lage sein, die in der Assertion übertragenen XACML-Attribute zu verarbeiten und in sein bestehendes XACML-System zu integrieren. Als Beispiel hierfür dient Listing 3.

So ist es in SAML 2.0 beispielsweise möglich, dass das Web Browser SSO Profile in Kombination mit dem XACML Attribute Profile eingesetzt werden kann.

Fazit

Alles in allem handelt es sich bei der SAML 2.0 um einen recht umfangreichen Standard. Dieser ist aber strukturiert und ausreichend dokumentiert und nach einer überschaubaren Einarbeitungszeit kann man sich schnell darin zurechtfinden. Hat man die ersten Hürden erfolgreich genommen, ist man anfangs vor allem von der großen Vielseitigkeit dieses Standards überrascht. Er ist sehr flexibel, individuell anpassbar und lässt sich nahtlos in eine bestehende Architektur aus Browser-basierten Diensten bzw. SOAP-Webservices integrieren. Das Konzept der Identity Federation dürfte für eine Vielzahl von kommerziellen Web-Anwendungen interessant sein und glänzt im Besonderen durch seine lose Kopplung der beteiligten Systeme.

Als Kritik am Standard kann festgehalten werden, dass der Begriff „Profile“ unglücklicherweise mehrdeutig gebraucht wird, was zu unnötiger Verwirrung führen kann. Des Weiteren sind leider alle unter [OasisSSTC] angebotenen Materialien sehr theoretisch und bieten einem Leser, der eher an einer technischen Realisierung bzw. an einem technischen Anschauungsbeispiel interessiert ist, keinerlei Anhaltspunkte und Beispiele. Diese Lücke soll aber in Teil 2 dieses Tutoriums gefüllt werden, sodass man sich anhand des kommenden praktischen Beispieles gut alleine in SAML 2.0 orientieren kann. Teil 1 hat dazu die theoretischen Grundlagen geschaffen.

Welche Werkzeuge sich im Umfeld von SAML 2.0 einsetzen lassen und angeboten werden, wird in diesem Tutorium bewusst nicht berücksichtigt. Dies würde den Rahmen bei Weitem sprengen, ist aber aus Sicht der Autoren auch nicht notwendig.

Links

[OasisSSTC] OASIS Security Services Technical Commitee,
`http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`

Michael Kain ist Consultant im Bereich Enterprise Application Development bei der Unternehmensberatung Resco in Hamburg. Er beschäftigt sich mit der Konzeption und Entwicklung von Web-Anwendungen.
E-Mail: Michael.Kain@resco.de

Guido Keller ist freiberuflicher Consultant im Bereich Enterprise Application Development. Sein Schwerpunkt liegt in der Konzeption und Entwicklung von Web-Anwendungen.
E-Mail: Keller.Guido@resco.de