Understanding Digital Identity Management

Phillip J. Windley, Ph.D.
phil@windley.com
www.windley.com

In his book, "The Age of Access," Jeremy Rifkin argues that economic shifts over the last several decades have given rise to a regime where anonymous transactions are nearly impossible. In a service-based economy, digital identity matters; I have to know who you are in order to sell you access to my service. Since these services are increasingly delivered over digital networks, businesses need reliable, secure, and private means for creating, storing, transferring, and using digital identities. Understanding how your organization will manage and use digital identity is a crucial part of any business strategy.

There are a number of metaphysical and post-modern discussions talking place concerning what a digital identity is and how that relates to the person, but that's a different paper. Instead, this paper will focus on defining the language of digital identity, relating digital identity to identity scenarios in the physical world, describing the primary protocols for creating, exchanging, and using digital identity, and discussing how you can develop an identity management strategy in your business.

The Language of Digital Identity

In the world of digital identity, a **subject** is a person, group, corporation, software program or other entity making a request to access a **resource**. A resource might be a web page, a piece of data in a database, or even a transaction on a credit card. In order to gain access to the resource, the subject lays claim to an **identity**.

In this context, **identities** are collections of data that represent **attributes**, **preferences**, and **traits**. Attributes are things like medical history, past purchasing behavior, bank balance, credit rating, dress size, age, and so on. Preferences represent desires such as preferred seating on an airline, brand of hot dog liked, use of one encryption standard over another, currency used, and so on. Traits are features of the subject that are inherent. Examples include blue eyes for a person, or how and where a company was incorporated.

For a subject to use an identity to justify accessing a resource, they must present **credentials**. Credentials are the proof that a subject has the right to assert a particular identity. Credentials are a way of transferring **trust** between entities. When credentials are presented to a **security authority** or **policy enforcement point (PEP)**, the authority **authenticates** the credentials. Authentication can be done using a simple username and password, an X.509 certificate, or biometrics.

The level of authentication required is usually proportional to the **risk** that attends access to the resource.

Once the credentials are believed to be authentic, the authority retrieves the **security policy** for the resource or passes them to a separate **policy decision point (PDP)**. The PDP uses the policy and asserted identity to determine the **entitlements** and **permissions** associated with that resource for the asserted identity. Entitlements are the services and resources to which an identity is entitled. Examples include a credit limits, disk space or bandwidth allocations, and so on. Permissions are the actions that the subject is allowed to perform with respect to the resource, such as withdrawing funds, completing a purchase, and updating a record. When the PDP transfers this information back to the PEP, it does so in an **authorization decision assertion (ADA).**

Information **security** protects information from unauthorized access, destruction, or alteration. Thus security is an identity and authorization issue. **Privacy** is the protection of the attributes, preferences, and traits associated with an identity from being disseminated beyond the subject's needs in any particular transaction. In a circular manner, privacy is built upon a foundation of good information security and that is dependent upon good identity management. This relationship is shown in Figure 1.

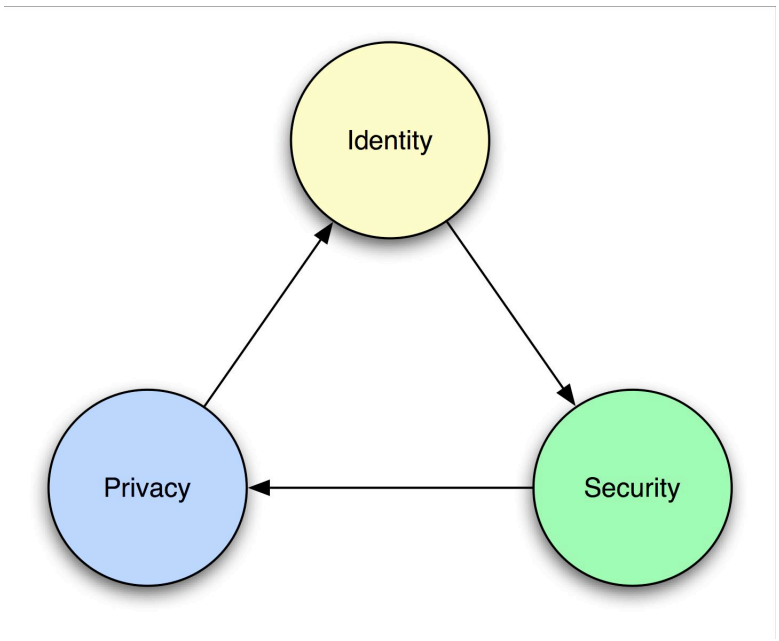

Figure 1: The Identity, Security, Privacy Triangle

Identity Scenarios in the Physical World

The world of digital identity is full of concepts that at first glance appear foreign because of the need to give precise definitions to terms in order to discuss them and specify system behavior using them. In reality, most of these concepts are

perfectly understandable given our everyday experience in personal and commercial transactions in the physical world.

For example, a driver's license is a credential that asserts that a person has certain attributes and traits. The license contains authorization to perform certain tasks, specifically to drive a car. When a person (i.e. the subject) wants to buy beer (i.e. perform an action on a resource), the clerk (i.e. security authority) examines the license to see if it looks real (i.e. determines the validity of the credential) and uses the picture (i.e. embedded biometric device) to see if the person presenting the license is the same person who owns it (i.e. authenticates the credential). Once certain that the license is authentic, the clerk, reads the birth date (i.e. attribute) from the license and determines whether the person is over 21 (i.e. consults a security policy determined by the state and makes a policy decision about permissions associated with the identity).

Now, suppose the person pays with a credit card. The credit card (a separate identity credential) is presented to the clerk. The clerk just saw the driver's license and so can establish the validity of this credential based on the first. The clerk, acting as the policy enforcement point, runs the card through the POS terminal which transmits identity attributes from the card (the credit card number and expiration date) along with the resource to be accessed (credit in the amount necessary to buy the beer) to the bank which acts as the policy decision point and determine whether or not the subject is entitled to credit in the necessary amount. The clerk receives the credit authorization (authorization decision assertion) and completes the transaction.

The Business Context of Identity

In addition to being able to identify customers so that they can sell them services, businesses have an increasing need to identify employees, systems, resources, and services in a systematic way to create business agility and ensure the security of business assets.

In the past people have thought of security as an edge game. Given a firewall and access control to the network, a business can be made reasonably secure. However, the economic shifts spoken of above have driven the need to integrate systems, not only internally, but with trading partners and customers as well. This has been fueled by XML and the creation of standards for exchanging data and the increasing trend to decentralized computing embodied in Web services. This trend has a huge ramification for business security: we can no longer treat the edges of the network as a secure perimeter.

When integration is driven by business needs, security policies need to talk about documents, data, actions, people, and corporations instead of machines and networks. This security model is infinitely more complex than the old "secure perimeter" model. But even if you define your policy, how do you ensure that it is properly implemented across dozens or even hundreds of systems and at the same time control access to resources as granular as fields of a database or paragraphs of a document? The answer lies in digital identity management.

Identity Framework

Digital identity management is built on a set of concepts that provide a framework for addressing the identity in a business context instead of an IT security context:

- Integrity and non-repudiation
- Confidentiality
- Authentication and authorization
- Identity provisioning
- Representing and managing authorization policy

The most significant challenge in each of these areas is interoperability since the goal is to build an enterprise wide identity system, even one that federates outside the enterprise. To that end, there are a number of standards bodies working to build a common foundation in these areas. The following sections briefly describe these problem domains and the standards being developed to address them.

Integrity and Non-Repudiation: XML Signatures

XML signatures can ensure message integrity and authenticate all or part of an XML document. The standard doesn't define new signature methods, but rather specifies how existing digital signature technology can be used inside XML documents. Message integrity ensures that the data has not been changed through some error during transport or as the result of a malicious attack. Digital signatures also prevent a signer from repudiating a message.

Like any standard, the XML Signature standard has options for numerous contingencies that make it seem complicated, but at its heart its quite simple. An XML signature is contained in a <Signature/> element and consists of three main parts:

1. The <SignedInfo/> element contains a reference to the data that has been signed along with information about how it was made canonical, what signature method was used, relevant digest information, and any transformations that were applied.
2. The <SignatureValue/> element contains the actual signature.
3. The <KeyInfo/> element contains the key information necessary to validate the signature.

Using the XML signature standard, parts of a document can be signed and multiple entities can sign the same or different parts of a document.

The following is an example of an XML signature shows each part:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo Id="fizz">
  <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <SignatureMethod
```

```
    Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
  <Reference URI="http://www.fizzco.com/news/2003/07/27.html">
   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
   <DigestValue>f7dhfj38sdjdf89fjskfhweol=</DigestValue>
  </Reference>
 </SignedInfo>
 <SignatureValue>KLUR~HF=</SignatureValue>
 <KeyInfo>
  <X509Data>
   <X509SubjectName>
     CN=Phil Windley,O=The Windley Group, LLC,ST=Utah, C=US
   </X509SubjectName>
   <X509Certificate>
     MIID5jCCA0+gA...lVN
   </X509Certificate>
  </X509Data>
 </KeyInfo>
</Signature>
```

This XML signature is signing the document located at the URL shown in the <Reference/> element.

Confidentiality: XML Encryption

Like any other communication on the Internet, any XML document can be encrypted in its entirety and sent across the wire to be decrypted at the other end. Alternately, the channel could be encrypted, using SSL for example, ensuring that any message transferred is protected.

Frequently, however, parts of XML message need to be sent in the clear. For example, all or part of the body of a SOAP message may need protection while the headers are sent in the clear so that routing and other information can be seen by intermediaries. The XML Encryption standard is designed to provide these benefits.

Any encrypted data in an XML document is identified using the <EncryptedData/> element. The <EncryptedData/> element consists of two parts:

1. An optional <EncryptionMethod/> element that gives <KeyInfo/> information. The <KeyInfo/> element is the same as the one defined for the XML signature specification.
2. A <CipherData/> element that contains the raw encrypted data contained in a <CipherValue/> element or a <CipherReference/> element that contains a URI that points to the raw encrypted data.

The following is an example of an <EncryptedData/> element:

```
<PaymentInfo xmlns='http://paymentco.org/payment_info.ver1.3b'>
  <Name>John Smith<Name/>
  <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
      xmlns='http://www.w3.org/2001/04/xmlenc#'>
   <CipherData><CipherValue>sh748fj14…</CipherValue></CipherData>
  </EncryptedData>
```

```
</PaymentInfo>
```

In this example, the cipher value field presumably would hold the credit card number and expiration data if we unencrypted it.

Authentication and Authorization Assertions: SAML

SAML, of the Security Assertion Mark-up Language, is a security credential standard. SAML provides standardized ways to use XML to represent security credentials and a protocol for requesting and receiving credential data from a SAML authority service. When combined with the WS-Security specification, SAML can be used to transport credential data in a SOAP message.

In practice, a client makes a request about a subject to a SAML authority and the authority returns assertions about the identity of the subject in a particular security domain. For example, the subject might be a person identified by their email address in a DNS domain. In the most common scenario, the requests and responses are transported over HTTP in a SOAP envelope.

SAML Authorities comes in three types: authentication authorities, attribute authorities, and policy decision points (PDP). These three types authorities return three distinct types of assertions:

1. **SAML Authentication Assertion**—when a SAML Authentication Authority performs an action and, as a consequence, makes a determination about a particular subject's credentials, the result is returned as a SAML Authentication Assertion. An authentication authority assets that subject S was authenticated by means M at time T. For example, subject Alice in company example.com was authenticated by means of her password at time 2003-05-06T13:20:00-05:00.
2. **SAML Attribute Assertions**—once an authentication assertion has been returned, a SAML Attribute Authority may be asked for the attributes associated with the subject. These are returned as a SAML Attribute Assertion. An attribute authority asserts that subject S is associated with attributes A, B, etc. with values a, b, etc. For example, subject Bob is associated with attribute Department with value Engineering.
3. **SAML Authorization Assertions**—the permissions associated with an authenticated subject with respect to a specific resource are returned by the PDP as a SAML Authorization Assertion. A PDP asserts that subject S has (or has not) been granted permissions for action A on resource R given evidence E. For example, subject http://A.com/services/foo is granted permission to read the file at http://B.com/bar as evidenced by a collection of other assertions.

In practice, a single authority can produce all three types of assertions, or authorities may produce a subset. Authorities can be both producers of assertions as well as consumers of assertions from other authorities (clients).

Assertions contain the following common elements:

- Issuer ID and issuance timestamp
- Assertion ID
- Subject
    - Name and security domain
    - Subject's authentication data (optional)
- Advice (optional additional information provided by the issuing authority)
- Conditions under which the assertion is valid
    - Assertion validity period (e.g. NotBefore and NotOnOrAfter)
- Audience restrictions
- Target restrictions (intended URLs for the assertion)
- Application specific conditions

The following example code shows the XML for a SAML attribute request:

```
<samlp:Request ...>
  <samlp:AttributeQuery>

    <saml:Subject>
      <saml:NameIdentifier
        SecurityDomain="A.com"
        Name="cn=Bob"/>
    </saml:Subject>
    <saml:AttributeDesignator
      AttributeName="Department"
      AttributeNamespace="A.com">
    </saml:AttributeDesignator>
  </samlp:AttributeQuery>
</samlp:Request>
```

In this request, the client is asking which department Bob@A.com is associated with.

The following XML is an example of a SAML authentication assertion:

```
<samlp:Response
  MajorVersion="1" MinorVersion="0"
  RequestID="128.14.234.20.90123456"
  InResponseTo="123.45.678.90.12345678"
  StatusCode="Success">

  <saml:Assertion
    MajorVersion="1" MinorVersion="0"
    AssertionID="123.45.678.90.12345678"
    Issuer="Example Company, Inc."
    IssueInstant="2003-01-14T10:00:23Z">

    <saml:Conditions
      NotBefore="2003-01-14T10:00:30Z"
      NotAfter="2003-01-14T10:15:00Z" />

    <saml:AuthenticationStatement
          AuthenticationMethod="Password"
```

```
          AuthenticationInstant="2003-01-14T10:00:20Z">

      <saml:Subject>
        <saml:NameIdentifier
          SecurityDomain="A.com"
          Name="cn=Alice" />
      </saml:Subject>
    </saml:AuthenticationStatement>
  </saml:Assertion>
</samlp:Response>
```

In this example response, the Authentication authority is asserting that Alice@A.com was authenticated on January 14, 2003 at 10:00:23 and this authentication is valid between 10:00:30 and 10:15:00 on the same day.

SAML Use Cases

There are four primary SAML use cases, or profiles: two for Web browsers and two that use SOAP. The first, called the "pull profile" has SAML artifacts (in essence, tokens) passed from one site to another using a URL query string. The site making the assertion (the source site) creates a link to the destination site containing the artifact in the URL and when the user clicks on it, the destination site receives the artifact. The artifact is a key that the second site can then use to pull the actual assertion from the source site. This profile is illustrated in

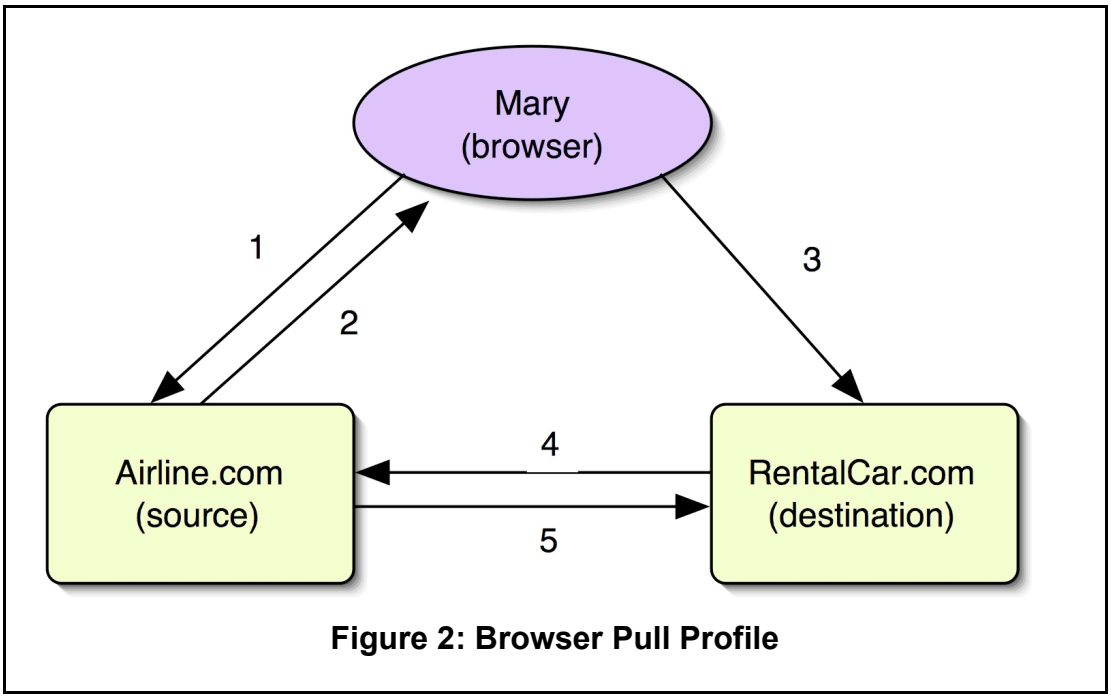

Figure 2: Browser Pull Profile

Figure 2.

In the example shown in Figure 2, Mary, visits Airline.com (1) and purchases airline tickets. As a result she has logged into Airline.com and been

authenticated. The site recommends that Mary rent a car from RentalCar.com during her trip and provides a link with an embedded SAML artifact (2). When Mary clicks on the link the artifact is transferred to RentalCar.com (3). RentalCar.com makes a SAML request (4) using the artifact and receives (5) an authentication assertion in return, allowing Mary to use the services on RentalCar.com without signing onto RentalCar.com. Later in the session, when Mary decides to rent a car, RentalCar.com may make an attribute request to Airline.com, getting Mary's credit card and address information to complete the purchase (not shown).

The second Web browser profile is called a "push" profile. In it, the source site creates a Web form containing the complete assertion. When the user submits the form, it is POSTed to the destination site and the assertion is pushed using the HTML form mechanism. In this scenario, the assertion is signed by the

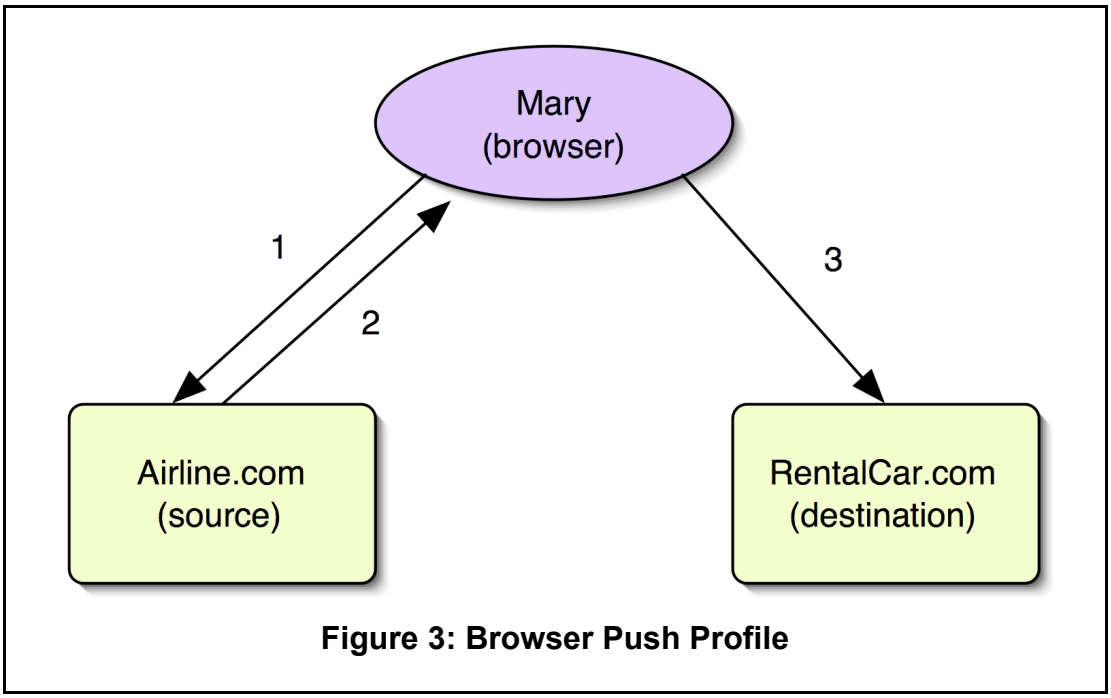

Figure 3: Browser Push Profile

source site so that the destination site can be assured of its authenticity. This profile is illustrated in Figure 3.

In the push profile, Mary visits Airline.com (1) just as she did in the pull profile. At some point in the session, Airline.com returns a page (2) containing an HTML form containing the complete, digitally-signed identity assertion. Mary submits the form which makes a POST request to RentalCar.com (3). RentalCar.com checks the digital signature and, being satisfied with the credentials, processes Mary's request. As in the pull profile, RentalCar.com may make other SAML requests of Airline.com to obtain Mary's pertinent attributes.

If RentalCar.com does make attribute or authorization requests to Airline.com about Mary, these requests will be transferred from RentalCar.com to Airline.com. In this case, the requests and responses would be packaged and

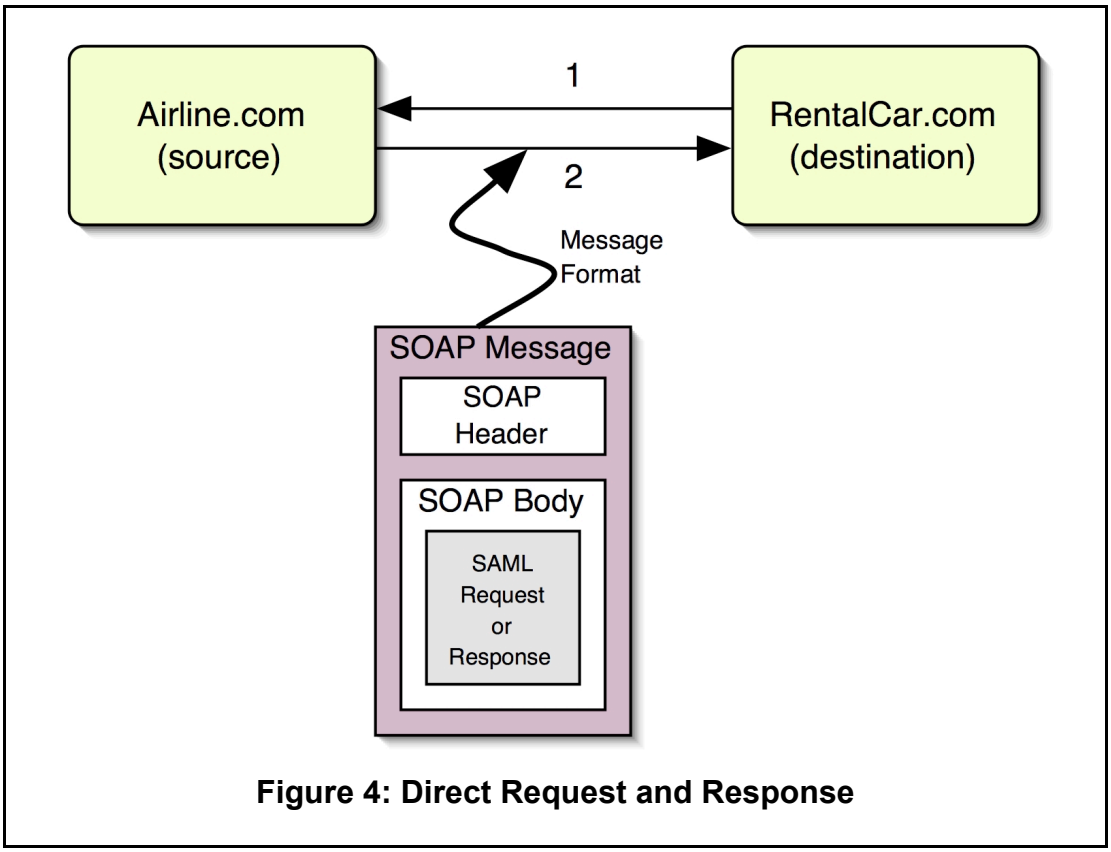

Figure 4: Direct Request and Response

sent across the wire in the bodies of SOAP messages as shown in Figure 4. In this illustration, RentalCar.com makes a request (1) and Airline.com responds

The Web service, or SOAP, profile passes the SAML assertion about the SOAP body in the SOAP header. In this case, the assertion in the header is a SAML artifact or digitally-signed assertion, just as in the pull and push profiles, that pertains to the action being taken in the body. The Web services profile is illustrated in Figure 5.

The primary benefit of SAML is that the security context for a subject travels with the subject (in the form of the assertion) so that a new context doesn't need to be created at each associated site or service the subject accesses. This reduces the need to store identification, authentication, and authorization information at each site or service and, as a consequence, this information does not need to be synchronized and duplicated at multiple sites and can be kept more securely. As a result,

- Single sign-on can be implemented,
- Attributed-based authorization can be support, and
- Multiple service providers can be federated using the same language and protocols.

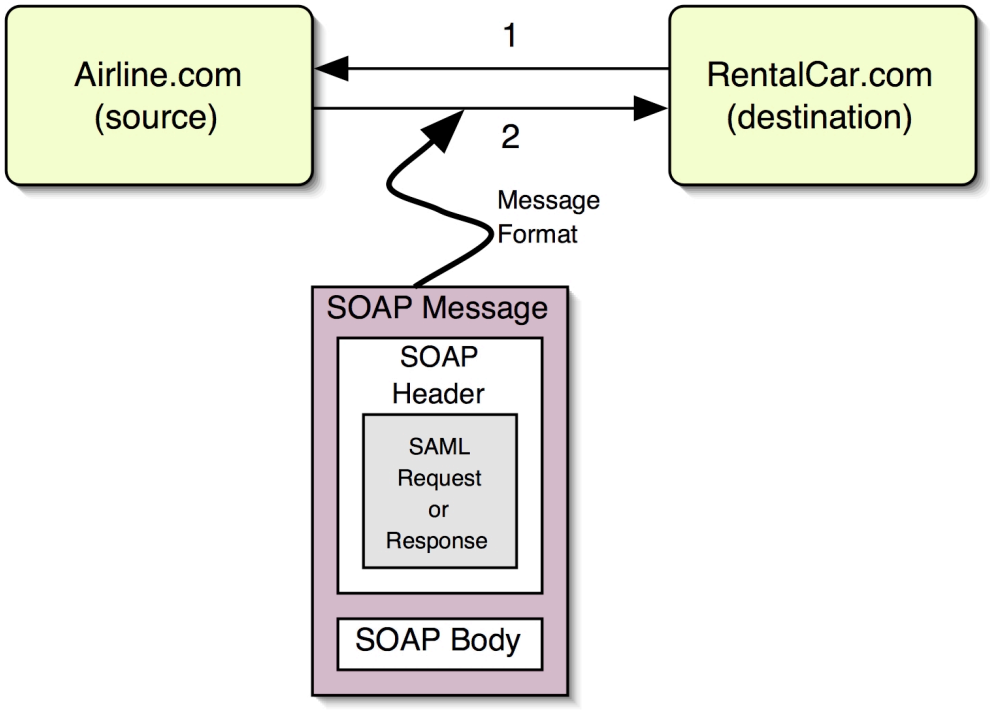

Figure 5: Web Services Profile

Identity Provisioning: SPML

SAML addresses the problem of how to exchange identity information between systems. That begs the question: how do these various systems set up the accounts and services that are being accessed? The process of preparing an IT system to provide services is called "provisioning." SPML provides a standard XML format for exchanging provisioning requests and responses.

SPML is defined in terms of three primary roles:

- A **Requesting Authority**, or RA, is the entity making the provisioning request.
- The **Provisioning Service Provider**, or PSP, is a SPML enabled software service that responds to SPML requests from the RA.
- The **Provisioning Service Target**, or PST, is the entity that actually carries out the provisioning work. At times the PSP and PST may be the same software agent. The crucial difference is that while the PSP is

required to understand SPML, the PST is not. So the PSP may be a front end for other non-SPML compliant software services.

There are several important concepts that flow from these definitions:

- SPML is a request/response protocol, with the RA making a request, in SPML, and the PSP returning a response or appropriate error code, in SPML.
- There must be a pre-existing trust relationship between the RA and the PSP. This trust relationship could be represented using SAML and transported using the Web services model shown above with the trust relationship being carried by SAML in the SOAP header and the SPML request and response being transported as the payload of the SOAP body.
- While the PST is not required to understand SPML, if it does, then the PSP is functioning as an RA making a request of another PSP.

There are two types of identifiers defined in SPML:

- A Provisioning Service Target Data identifier, or PSTD-ID, is the name given to the identifiers used by each PST. These are unique within the PST.
- A Provisioning Service Object Identifier, or PSO-ID, is a unique identifier that represents a collection of individual PSTD-ID's. The RA might choose this or the PSP might supply it.

These identifiers are the unique tokens used to identify the resource or subject.

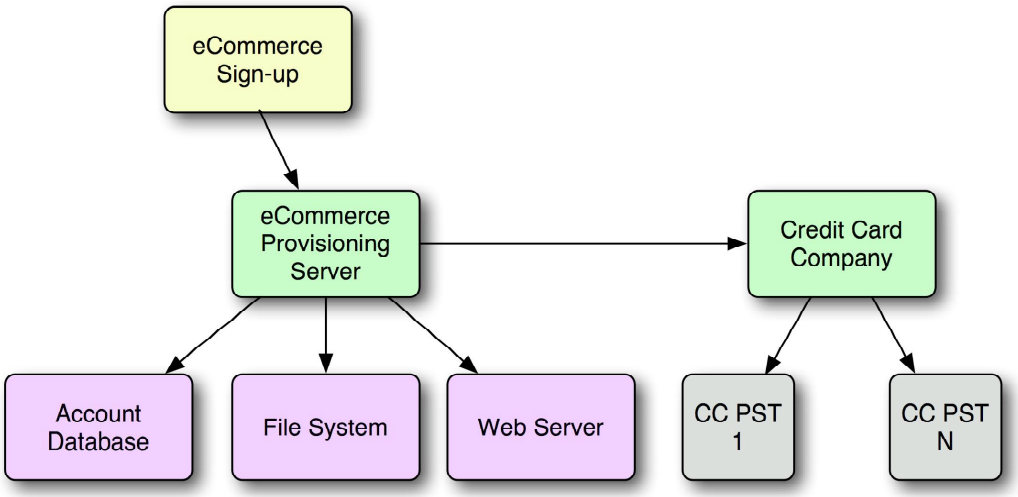

Figure 6: eCommerce Provisioning Example

Figure 6 diagrams an example of the provisioning activities related to creating a new account in an eCommerce system. The eCommerce sign-up process, acting as an RA, sends a SPML request to the eCommerce Provisioning server, acting as the PSP, asking that a new eCommerce account be created. There are four different activities that need to be completed for the provisioning action to be complete:

1. The eCommerce Provisioning Server uses JDBC to create an appropriate record in the account database.
2. The eCommerce Provisioning Server creates a directory on the file system for storing files related to the new account.
3. The eCommerce Provisioning Server requests that the LDAP server that functions as the Web server's authentication server create an account using JNDI.
4. The eCommerce Provisioning Server, acting as an RA, sends a SPML request to the credit card company's PSP requesting that a merchant account be provisioned. The credit card company's PSP, in turn, makes some number of requests of other PSTs or PSPs to service the request.

In the case of the initial request and the request from the eCommerce Provisioning Server to the Credit Card Company PSP, SPML requests are sent and SAML is used to verify the identity of these entities to each other and establish authorization to make the request. In the case of activities 1, 2, and 3, the authorization to make the requests is established in whatever way the database, file system, and LDAP server have traditionally used. None of these servers are SPML aware in this example.

SPML requests take one of several forms:

- An <addRequest/> element is used to request the creation of an account.
- A <modifyRequest/> element is used to request that an account be updated.
- A <deleteRequest/> element is used to request that an account be deleted.
- A <searchRequest/> element is used to query the PSP about accounts and their properties.

These requests can have operational attributes attached and can be done in batch as well as singleton mode.

SPML achieves flexibility by allowing PSPs to define provisioning schema. An RA can request a provisioning schema from the PSP using a <schemaRequest/> element and use the response to this request to build add, modify, delete, and search requests.

As an example, of what a SPML request response pair looks like, consider the following example of an <addRequest/> that creates an email account for a user:

```
<addRequest>
```

```
  <attributes>
    <attr name="objectclass">
      <value>urn:oasis:names:tc:SPML:interop:interopUser</value>
    </attr>
    <attr name="cn">
      <value>Jane Doe</value>
    </attr>
    <attr name="mail">
       <value>jdoe@acme.com</value>
    </attr>
    <attr name="description">
      <value>Jane Doe interopUser Subscription</value>
    </attr>
    <attr name="memberLevel">
      <value>1</value>
    </attr>
    <attr name="company">
      <value>Jane Doe Supply Co</value>
    </attr>
    <attr name="registrationTime">
      <value>17-Nov-2002 12:00 </value>
    </attr>
  </attributes>
</addRequest>
```

The PSP might respond to this request as follows:

```
<addResponse result = "urn:oasis:names:tc:SPML:1:0#success">
  <identifier type= "urn:oasis:names:tc:SPML:1:0#EMailAddress">
    <id>Jane.Doe@acme.com</id>
  </identifier>
  <attributes>
    <attr name="mailBoxLimit">
      <value>50MB</value>
    </attr>
  </attributes>
</addResponse>
```

Other examples of SPML requests and responses can be found in the OASIS Core document on SPML referenced in the Resources section of this paper.

Representing and Managing Authorization Policies: XACML

One of the things that plagues large enterprises is implementing an access control policy on a consistent basis across the entire organization. The problem is that access control policies are written in English and then implemented on dozens or even hundreds of different systems individually. Changing the access control policy requires redoing all of this work. The job is seldom finished or done correctly.

eXtensible Access Control Mark-up Language, or XACML, attempts to solve this problem by providing a XML-based language for storing and sharing access control policies. XACML is a language of the Policy Decision Point, or PDP. If each PDP in the organization is XACML enabled, a single policy can be created and understood by each PDP without constant configuration and reconfiguration.

SAML provides a format for exchanging identity and access information, XACML provides a format for describing what to do with that information. There is some overlap between SAML and XACML: both provide a request/response language. The two standards are complimentary, however, using the same definitions for subjects and actions. But XACML goes beyond SAML in creating a rule-based language for expressing access control policies.

Because XACML is at its core a rule-based language, it is much more complicated than the other standards that we've discussed which are used for representing data. Policy rules can be created which allow fine-grained control over access. These rules can be based on items such as:

- Attributes of the user making the request (i.e. only people who work in accounting are allowed to access this document).
- The action to be taken (i.e. reading a document vs. updating a document).
- The time of day (i.e. users can only log in between 9am and 5pm, or during their shift).
- The authentication mechanism (i.e. only allow users to log in who present a digital signature as their authentication mechanism)
- The protocol used for access (i.e. allow HTTPS access, but not HTTP access for this resource).
- These can be combined using Boolean connectives to create complex policies (i.e. only allow users from accounting to update this document over HTTPS but allow users from other departments to read it anytime over HTTP).

XACML also has a concept called a "combining algorithm" which allows multiple, perhaps conflicting, policies to be applied to a resource and some action taken as a result. For example, the combining algorithm might simply state "deny if any policy denies" or it might be more complicated.

The following XACML request (from Sun's XACML paper—see Resources) states that an unstated subject wants to log into a server called "SampleServer." The request has three primary parts: it identifies the subject making the request, names the resource, and describes the action.

```
<Request
      xmlns="urn:oasis:names:tc:xacml:1.0:context"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
        cs-xacml-schema-context-01.xsd">
  <Subject/>
  <Resource>
    <Attribute
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
               DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>SampleServer</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
```

```
    <Attribute AttributeId="ServerAction"
               DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>login</AttributeValue>
    </Attribute>
  </Action>
</Request>
```

The policy statement that is applied to this request is much more complicated but its not hard to figure out what it does. The <Target/> element indicates to what subjects, resources, and actions this policy applies. Rules have simple actions: permit or deny. These actions are called the “effect” of the rule. In addition, the <Rule/> element contains two parts, a <Target/> element that is matched against the current security request and a <Condition/> element that defines a Boolean policy that is only true between the hours of 9am and 5pm.

Reading through this example will give you an idea of what XACML policies look like. These policies are human readable, but for the most part, they would be created using a access control policy tool. The top-level element is the <Policy/> element:

```
<Policy
      xmlns="urn:oasis:names:tc:xacml:1.0:policy"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
        cs-xacml-schema-policy-01.xsd"
      PolicyId="SamplePolicy"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:first-applicable">
```

The <Policy/> element contains a <Target/> element and zero or more <Rule/> elements. The <Target/> element defines the class of subjects, resources, and actions to which the policy is applied . For example, the following <Target/> element applies to any subject, and any action, on a resource identified by the string “SampleServer”

```
<Target>
 <Subjects>
   <AnySubject/>
 </Subjects>
 <Resources>
  <Resource>
   <ResourceMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string">
      SampleServer
     </AttributeValue>
     <ResourceAttributeDesignator
      DataType=”http://www.w3.org/2001/XMLSchema#string”
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
   </ResourceMatch>
  </Resource>
 </Resources>
 <Actions>
  <AnyAction/>
```

```
 </Actions>
</Target>
```

Notice that operators and types are long an unwieldy in the style of XML-based languages. This is a reflection of the truth that XACML is not designed to be written by humans, but rather by programs. After the <Target/> element, the <Policy/> contains <Rules/>. The following two rules require that subjects login. The default action, taken by the second rule is to deny access (its effect is to "Deny"). The first rule requires the login and further stipulates that it must be between the hours of 9am and 5pm. Its effect, given as an attribute to the <Rule/> element, is to "Permit." If the first rule matches, then the second, default rule, is never seen.

```
<Rule RuleId="LoginRule" Effect="Permit">
 <Target>
  <Subjects>
   <AnySubject/>
  </Subjects>
  <Resources>
   <AnyResource/>
  </Resources>
  <Actions>
   <Action>
    <ActionMatch
       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">
      Login
     </AttributeValue>
     <ActionAttributeDesignator
       DataType="http://www.w3.org/2001/XMLSchema#string"
       AttributeId="ServerAction"/>
     </ActionMatch>
    </Action>
   </Actions>
  </Target>

  <!-- Only allow logins from 9am to 5pm -->
  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
   <Apply
    FunctionId=
    "urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
   <Apply
    FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-
only">
    <EnvironmentAttributeDesignator
       DataType="http://www.w3.org/2001/XMLSchema#time"
       AttributeId=
         "urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
   </Apply>
   <AttributeValue
     DataType="http://www.w3.org/2001/XMLSchema#time">
    09:00:00
   </AttributeValue>
```

```
  </Apply>
  <Apply
   FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-
equal">
   <Apply
    FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-
only">
    <EnvironmentAttributeDesignator
      DataType="http://www.w3.org/2001/XMLSchema#time"
      AttributeId=
        "urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
   </Apply>
   <AttributeValue
     DataType="http://www.w3.org/2001/XMLSchema#time">
    17:00:00
   </AttributeValue>
  </Apply>
 </Condition>
</Rule>

<!-- A final, "fall-through" Rule that always Denies -->
<Rule RuleId="FinalRule" Effect="Deny"/>
```

The response generated is simple and contains a decision (“Permit” in this case) and a status.

```
<Response>
  <Result>
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
</Response>
```

Creating a Digital Identity Strategy

None of these protocols will do you any good without a good digital identity strategy. A digital identity strategy is a long-term plan that models how identity information will be used by your business, taking into account the key stakeholders in identity: your partners, customers, and employees. There are several important steps:

- creating an enterprise information architecture (EIA) to determine the business context for your strategy,
- identify the standards that will be used by the enterprise as part of the EIA,
- developing a authentication and authorization policy consistent with the EIA,
- planning and implementing enterprise directory services and other infrastructure necessary to support your policies, and
- publishing a privacy policy based on the authentication and authorization policy along with relevant laws and stakeholder expectations.

Enterprises who implement an identity management strategy stand to reap significant benefits. Among these are a consistent and systematic approach to customers, improved security for corporate applications and information, lower user administration costs, and better compliance with internal and external policies.

Conclusion

Digital identity has the potential to be an asset to your business and one of your most crucial pieces of infrastructure. Having good identity systems enables other strategic relationships with stakeholders. Building a workable digital identity strategy is considerable work, but if you neglect it, rather than being an asset, issues of identity will be a constant source of worry and a roadblock to other strategic initiatives.

References, Resources and Further Reading

The following referenced can be used to gather more information on the indicated topics.

XML Signature and Encryption

Ed Simon ,Paul Madsen ,Carlisle Adams , An Introduction to XML Digital Signatures, O'Reilly xml.com, August 8, 2001. (http://www.xml.com/pub/a/2001/08/08/xmldsig.html)

Murdoch Mactaggart , An Introduction to XML Encryption and XML Signature, Ibm Developer Works, September 1, 2001. (http://www-106.ibm.com/developerworks/xml/library/s-xmlsec.html/index.html)

Rich Salz, Understanding XML Digital Signature, Microsoft Developer Network, July, 2003. (http://msdn.microsoft.com/webservices/understanding/xmlfundamentals/default.aspx?pull=/library/en-us/dnwebsrv/html/underxmldigsig.asp)

SAML

Jon Byous, Single Sign-On Simplicity with SAML (http://java.sun.com/features/2002/05/single-signon.html)

Jorgen Thelin, OReilly Emerging Technology Conference talk (http://www.capescience.com/articles/identity/ws-identity-2003-02.pdf)

Marc Chanliau, Security Assertion Markup Language (SAML) (http://www.simc-inc.org/archive0002/February02/Speakers/SAML-SIMC-short/index.htm)

SPML

Oasis Provisioning Services Technical Committee (http://www.oasis-open.org/committees/spml)

Gavenraj Sodhi, SPML and its Importance (http://www.omg.org/interop/presentations/2002/Gavenraj_Sodhi.pdf)

OpenSMPL, Open source software toolkit for SPML (http://www.openspml.org/index.html)

XACML

Oasis Extensible Access Control Mark-up Language Technical Committee (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)

XACML: A New Standard Protects Content in Enterprise Data Exchange, Sun Developer Web Site (http://developer.java.sun.com/developer/technicalArticles/Security/xacml/xacml.html)

Sun's XACML Implementation (http://sunxacml.sourceforge.net/)

About the Author

Phillip J. Windley is a nationally recognized expert in using information technology (IT) to add value to the business. Dr. Windley regularly consults with businesses on this topic. He is particularly interested in the areas of interoperability, web services, XML, and digital identity. Dr. Windley is a frequent author and speaker on these topics and authors a free, daily web-based newsletter at www.windley.com. His web-site contains numerous white papers in these areas and others.

Dr. Windley served from 2001-2002 as the Chief Information Officer (CIO) for the State of Utah serving on the Governor's Cabinet and as a member of his Senior Staff. In this capacity he was responsible for effective use of all IT resources in the state and advised the Governor on technology issues. During his tenure, the State of Utah was repeatedly recognized by many national groups for its excellence in the areas of IT and eGovernment.

Prior to his appointment as CIO, Dr. Windley served as Vice President for Product Development and Operations at Excite@Home, managing a large, interdisciplinary team of product managers, engineers, and technicians developing and operating large scale Internet and e-commerce products. Prior to joining Excite@Home, Dr. Windley served for two years as Chief Technology Officer (CTO) of iMALL, Inc. an early leader in electronic commerce. Dr. Windley has been a professor of Computer Science at Brigham Young University and the University of Idaho. At BYU he founded and directed the Laboratory for Applied Logic. Windley received his PhD in Computer Science from the University of California, Davis in 1990. Prior to doing graduate studies, Windley worked for 4 years as a nuclear metallurgist and a member of the technical staff at the Department of Energy's Division of Naval Reactors.

Copyright Information