Abstract: “This session will provide a technical overview of SAML, the XML-based Security Assertion Markup Language being standardized at OASIS. It will discuss how SAML enables Single Sign-On and other security scenarios, and provide details about the authentication, attribute, and authorization information that SAML can convey. It will also cover the protocol by which security information can be requested from SAML Authorities and the practical realities of how this information can be transported securely across domains.”

Audience: Technically knowledgeable IT people; doesn’t assume any special security or provisioning knowledge, but the audience should be at least a little familiar with directories, XML
Agenda

- The problem space
- SAML concepts
- Walking through scenarios
- Status of SAML and related standards efforts
- Your questions

Thanks to Prateek Mishra (Netegrity), RLBob Morgan (UWash/Internet2), and Darren Platt (RSA) for some material in this presentation
Agenda

- The problem space
  - Why invent SAML at all?
  - What are the use cases that drive SAML’s design?
- SAML concepts
- Walking through scenarios
- Status of SAML and related standards efforts
What problems does SAML try to solve?

- Standards are emerging for many facets of collaborative e-commerce, such as:
  - Business transactions (e.g., ebXML)
  - Software interactions (e.g., SOAP)
- But communicating security properties of these interactions isn’t well standardized
  - Low interoperability between PMI solutions
  - Tight coupling within components
- Web-based commerce shows the need for federation, standardization, and a more cohesive user experience
Use cases for sharing security information

- SAML developed three “use cases” to drive its requirements and design:
  1. Single sign-on (SSO)
  2. Distributed transaction
  3. Authorization service
- Each use case has one or more “scenarios” that provide a more detailed roadmap of interaction

The SAML “use cases” are reminiscent of aggregations of UML-style use cases, rather than individual ones, so the name isn’t quite accurate.

The “scenarios” are message flow diagrams that have quite a bit of detail in them.

Ultimately, these (developed as part of our requirements-gathering process) are being turned into designs for what SAML calls “profiles”.

5
Steps:
1. Web user authenticates to the source Web site.
2. Web user uses a secured resource at the destination Web site.

This use case has specific **push** and **pull** scenarios.

These companies have partnered in such a way that the Company B has agreed to accept Company A’s web users and to trust Company A’s authentications of those users. This implementation will require a linkage between the single sign-on systems of the two domains. SAML provides a standard way for these single sign-on solutions to communicate about the authentications of these end users in such a way as to facilitate this interaction.

An example of this type of scenario would be where a company is using an ASP (Application Service Provider) to provide its expense management functionality. It would be very desirable for the company’s employees to not have to authenticate...
#2: Distributed transaction

- Employees at SmithCo are allowed to order office supplies from OfficeBarn if they are authorized to spend enough

![Diagram of transaction process]

This transaction could be a financial transaction flowing between B2B gateways or a simple SOAP method invocation. Depending on the business relationship between the parties, different security information related to the transaction will need to be communicated between the parties of the transaction (or perhaps even by a third party). Examples of the types of information that will need to be passed are the identity of the user who initiated the transaction, the type(s) of authentication that were provided, attributes about the user, and authorization decisions. This security information will be used by the security systems of the organizations that are processing the transaction.

An example of this scenario is where a user (Sam) accesses his company (ACME)’s supply management application to order some computers. ACME’s supply management application, in turn, places an order with its computer supplier. Securely included in this order are SAML assertions which contain statements that
#3: Authorization service

- Employees at SmithCo order office supplies directly from OfficeBarn, which performs its own authorization

Steps:

1. User accesses a resource controlled by PEP.
2. PEP checks permission for user to access resource with PDP.

This could be used with programs too.

A third scenario where SAML will be useful is when an organization needs to request security information from a trusted third party.

An example of this scenario would be if an employee of Company A has logged into the supplier’s website directly and has initiated an order for paper clips. The supplier’s system would need some more information to determine whether to allow this order to be processed. In this example, the supplier would use SAML to query Company A for a decision as to whether to allow the transaction, or perhaps some attributes about the employee (for example role or employee status) so that the supplier can decide using their own authorization policy.
What’s needed to accomplish all this

- A standard XML message format
  - It’s just data traveling on any wire
  - No particular API mandated
  - Lots of XML tools available
- A standard message exchange protocol
  - Clarity in orchestrating how you ask for and get the information you need
- Rules for how the messages ride “on” and “in” transport protocols
  - For better interoperability

With XML, you often see standards that are simply wire protocols; no API is mandated, and in some cases no binding to some transport mechanism (such as HTTP or SMTP or whatever) is provided. We felt that the latter is definitely needed so that proprietary mechanisms don’t creep in.
Agenda

- The problem space
- SAML concepts
  - SAML in a nutshell
  - SAML assertions
  - Producers and consumers of assertions
  - Message exchange protocol
  - Bindings and profiles
- Walking through scenarios
- Status of SAML and related standards efforts
SAML in a nutshell

- It’s an XML-based framework for exchanging security information
  - XML-encoded security “assertions”
  - XML-encoded request/response protocol
  - Rules on using assertions with standard transport and messaging frameworks
- It’s an emerging OASIS standard
  - Vendors and users are involved
  - Codifies current system outputs rather than inventing new technology
  - Today I’m presenting SAML as of 10 January 2002

Original SAML players: Netegrity and Securant (since acquired by RSA) contributed the initial specifications (S2ML, AuthXML). There were other contributions as well that were more peripheral to SAML’s core purpose.

Other major vendors/contributors: These include, for example, Baltimore, BEA, Entrust, Entegrity, HP, IBM/Tivoli, Oblix, Sun, VeriSign, Jamcracker, and Internet2/Shibboleth.

Areas of expertise: Distributed systems development, PKI, and XML.

What about Microsoft? They didn't participate in our early work, but they received some "encouragement" to take a look at SAML later. They have contributed design ideas, mostly about Kerberos support. A subcommittee has formed to pursue this more. The latest .NET/Passport story addresses "federated" functions, based on Kerberos. They have made no commitment to SAML, but the lines of communication have been opened.
Agenda

- The problem space
- **SAML concepts**
  - SAML in a nutshell
  - *SAML assertions* (syntax from core-25 draft)
  - Producers and consumers of assertions
  - Message exchange protocol
  - Bindings and profiles
- Walking through scenarios
- Status of SAML and related standards efforts
SAML assertions

- Assertions are declarations of fact, according to someone
- SAML assertions are compounds of one or more of three kinds of “statement” about “subject” (human or program):
  - Authentication
  - Attribute
  - Authorization decision
- You can extend SAML to make your own kinds of assertions and statements
- Assertions can be digitally signed

Typically a subject is human -- it’s somebody who is trying to accomplish some task on a computer and who needs to be checked out before they’re allowed to proceed. But a subject can equally well be another computer/program.

SAML has the XML Signature standard from W3C/IETF baked into it.
All statements in an assertion share common information

- Issuer ID and issuance timestamp
- Assertion ID
- Subject
  - Name plus the security domain
  - Optional subject confirmation, e.g. public key
- “Conditions” under which assertion is valid
  - SAML clients must reject assertions containing unsupported conditions
  - Special kind of condition: assertion validity period
- Additional “advice”
  - E.g., to explain how the assertion was made

If an assertion is a declaration of fact according to some **issuer**, then it’s really important to document who said something.

**Validity periods** become important because not all assertions will be good forever. It’s an important security consideration in SAML to make the validity period be as short as is practicable.

SAML **conditions** are a little bit like SOAP mustUnderstand. You can add your own conditions on top of SAML. Such a feature can be open to abuse and lock-in by proprietary systems. Ideally new conditions will be standardized or at least openly published.
Assertion structure
Example common information for an assertion

```xml
<saml:Assertion
    MajorVersion="1" MinorVersion="0"
    AssertionID="128.9.167.32.12345678"
    Issuer="Smith Corporation"
    IssueInstant="2001-12-03T10:02:00Z">
    <saml:Conditions
        NotBefore="2001-12-03T10:00:00Z"
        NotOnOrAfter="2001-12-03T10:05:00Z">
        <saml:AudienceRestrictionCondition>
            <saml:Audience>…URI…</saml:Audience>
        </saml:AudienceRestrictionCondition>
    </saml:Conditions>
    <saml:Advice>
        …a variety of elements can go here…
    </saml:Advice>
    …statements go here…
</saml:Assertion>
```

The `saml:` namespace corresponds to an XML Schema that just focuses on assertions.

So far, we have decided that **unique identifiers** like AssertionID are essentially opaque to SAML; it’s up to you to ensure that they’re unique in the domain you care about.
Authentication statement

- An issuing authority asserts that subject S was authenticated by means M at time T
- Targeted towards SSO uses
- **Caution:** Actually checking or revoking of credentials is not in scope for SAML!
- It merely lets you link back to acts of authentication that took place previously

It was a hard decision not to put **credential passing** in scope for SAML 1.0 (for example, S2ML does it), but we probably would never finish because the whole problem is hard. We are examining **credential pass-through** to increase SAML’s utility in this area marginally.

A frequently asked question is: How secure is it to have this **disconnect between the actual authentication and the sending of an authentication assertion**? It is up to the system who needs the information to decide what authentication assertion issuers it chooses to trust.

The means of **ensuring that the entity making a request and the entity referred to by an assertion are one and the same** is dependent on the environment and protocols being used. The general mechanism provided is the SubjectConfirmation element, which is intended carry data appropriate to the environment (such as a public key).
Authentication statement structure
Example assertion with authentication statement

```xml
<saml:Assertion …>
  <saml:AuthenticationStatement
    AuthenticationMethod="password"
    AuthenticationInstant="2001-12-03T10:02:00Z">
    <saml:Subject>
      <saml:NameIdentifier
        SecurityDomain="smithco.com"
        Name="joeuser"/>
      <saml:ConfirmationMethod>
        http://...core-25/sender-vouches
      </saml:ConfirmationMethod>
    </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>
```

The `saml:` namespace corresponds to an XML Schema that just focuses on assertions.

So far, we have decided that **unique identifiers** like AssertionID are essentially opaque to SAML; it’s up to you to ensure that they’re unique in the domain you care about.
Attribute statement

- An issuing authority asserts that subject S is associated with attributes A, B, ... with values “a”, “b”, “c”...
- Useful for distributed transactions and authorization services
- Typically this would be gotten from an LDAP repository
  - “john.doe” in “example.com”
  - is associated with attribute “Department”
  - with value “Human Resources”

Attributes of a user (such as their role and spending limit) might be passed in a distributed transaction or as part of the “outsourcing” of an authorization decision.

These assertions will be cryptographically bound to the transaction in a way that is defined as part of the SAML specification.

Attributes are the universal solvent of security information. Authentication and authorization information could be passed as attributes, but SAML chose to optimize for those common cases. Your own deployment could certainly choose to use attribute statements only.
Attribute statement structure
Example assertion with attribute statement

```
<saml:Assertion ...>
  <saml:AttributeStatement>
    <saml:Subject>...</saml:Subject>
    <saml:Attribute
      AttributeName="PaidStatus"
      AttributeNamespace="http://smithco.com">
      <saml:AttributeValue>
        PaidUp
      </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute
      AttributeName="CreditLimit"
      AttributeNamespace="http://smithco.com">
      <saml:AttributeValue>
        <my:amount currency="USD">500.00</my:amount>
      </saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

SAML requires attribute names to be qualified as being in a “namespace” (not an XML namespace), so that you can indicate the universe in which the attributes of interest were defined. This makes SAML neatly extensible in this area.
In the authorization service example, a user is accessing an ASP-hosed application that her company is using. She navigates to a secure area of the application which requires special permissions. The ASP application can then use the SAML protocol to ask her company’s security system whether to allow her to access that secure area.
Authorization decision statement structure
Example assertion with authorization decision statement

```xml
<saml:Assertion ...>
  <saml:AuthorizationStatement
    Decision="Permit"
    Resource="http://jonesco.com/rpt_12345.htm">
    <saml:Subject>...</saml:Subject>
    <saml:Actions
      ActionNamespace="http://...core-25/rwec">
      <saml:Action>Read</saml:Action>
    </saml:Actions>
  </saml:AuthorizationStatement>
</saml:Assertion>
```
Agenda

- The problem space
- **SAML concepts**
  - SAML in a nutshell
  - SAML assertions
  - Producers and consumers of assertions
  - Message exchange protocol
  - Bindings and profiles
- Walking through scenarios
- Status of SAML and related standards efforts
1. We’ve already shown examples of XML documents containing SAML assertions. The stuff in the gray box is standardized by the SAML specifications. But something has to produce them, and something has to consume them. The model I’ll flesh out for you here was developed to help the committee understand the basic inputs and outputs.

2. Assertions are produced (issued) by **authorities**. An authentication authority produces authentication assertions, an attribute authority produces attribute assertions, and a policy decision point or PDP (in other words, an authorization decision authority) produces authorization decision assertions.

3. But these authorities don’t live in a vacuum. They may each get various kinds of data from the “outside”, and this data may reside in a separate or integrated policy store. For example, a PDP might have to look up a policy statement about Joe User and check the current date to know whether Joe can look at the report he wants.

4. And here’s the rest of the picture. This just shows that credential checking happens outside the scope of SAML (this was a big point of contention) and that it all starts with some kind of system entity (a human or program) that wants to accomplish something. Note that the **policy enforcement point (PEP)** is just a special name for the particular kind of requester that wants the things that a PDP produces.
This model is conceptual only

- In practice, multiple kinds of authorities may reside in a single software system
  - SAML allows, but doesn’t require, total federation of these jobs
- Also, the arrows may not reflect information flow in real life
  - The order of assertion types is insignificant
  - Information can be pulled or pushed
  - Not all assertions are always produced
  - Not all potential consumers (clients) are shown
Agenda

- The problem space
- SAML concepts
  - SAML in a nutshell
  - SAML assertions
  - Producers and consumers of assertions
  - Message exchange protocol (syntax from core-25 draft)
    - Bindings and profiles
- Walking through scenarios
- Status of SAML and related standards efforts
1. We’re already familiar with assertions. They’re in the gray box here because SAML specifies them.

2. But they have to be delivered by authorities somehow; this is done in an XML document called a SAML response.

3. Of course, a response is produce in response to something. That something is an XML document called a SAML request.

4. And finally, these don’t live in a vacuum. We introduced the “authority” terminology a moment ago; a generic way of viewing issuers/producers/authorities is as asserting parties, and a generic way of viewing the systems that need, request, and use the assertions is as relying parties.
Assertions are normally provided in a SAML response

- Existing tightly coupled environments may need to use their own protocol
  - They can use assertions without the rest of the structure
- The full benefit of SAML will be realized where parties with no direct knowledge of each other can interact
  - Via a third-party introduction

Some scenarios for using SAML have, as part of their flow, raw assertions attached to other data. There’s nothing wrong with this, but if the SAML request-response protocol could be used for basic querying of authorities and isn’t, you might very well be missing an opportunity to interoperate with other systems.
Requests can take several forms

- You can query for specific kinds of assertion/statement
  - Authentication query
  - Attribute query
  - Authorization decision query

- You can ask for an assertion with a particular ID
  - By providing an ID reference
  - By providing a SAML “artifact”

I will cover artifacts later.
Authentication query

- “Please provide the authentication information for this subject, if you have any”
- It is assumed that the requester and responder have a trust relationship
  - They are talking about the same subject
  - The response with the assertion is a “letter of introduction” for the subject
Authentication query structure
Example request with authentication query

```xml
<samlp:Request
    MajorVersion="1" MinorVersion="0"
    RequestID="128.14.234.20.12345678">
    <samlp:AuthenticationQuery>
        <saml:Subject>
            <saml:NameIdentifier
                SecurityDomain="smithco.com"
                Name="joeuser" />
        </saml:Subject>
    </samlp:AuthenticationQuery>
</samlp:Request>
```
Attribute query

- “Please provide information on the listed attributes for this subject”
- If you don’t list any attributes, you’re asking for all available ones
- If the requester is denied access to some of the attributes, only the allowed attributes would be returned
  - (This situation is indicated in the status code of the response)
Attribute query structure
Example request with attribute query

```xml
<samlp:Request ... >
  <samlp:AttributeQuery>
    <saml:Subject>
      <saml:NameIdentifier
        SecurityDomain="smithco.com"
        Name="joeuser" />
    </saml:Subject>
    <saml:AttributeDesignator
      AttributeName="PaidStatus"
      AttributeNamespace="http://smithco.com"/>
  </samlp:AttributeQuery>
</samlp:Request>
```
Authorization decision query

• “Is this subject allowed to access the specified resource in the specified manner, given this evidence?”
• This is a yes-or-no question
  – The answer is not allowed to be “no, but they’re allowed to access these other resources”
  – Or “yes, and they’re also allowed to perform these other actions”
Authorization decision query structure
Example authorization decision query

```xml
<samlp:Request ...>
  <samlp:AuthorizationQuery
    Resource="http://jonesco.com/rpt_12345.htm">
    <saml:Subject>
      <saml:NameIdentifier
        SecurityDomain="smithco.com"
        Name="joeuser" />
    </saml:Subject>
    <saml:Actions
      ActionNamespace="http://...core-25/rwedc">
      <saml:Action>Read</saml:Action>
    </saml:Actions>
    <saml:Evidence>
      <saml:Assertion>...</saml:Assertion>
    </saml:Evidence>
  </samlp:AuthorizationQuery>
</samlp:Request>
```

Just as for attributes, the potential actions are governed in a way that allows SAML not to be dependent on a “closed set” of them.
Responses just contain a set of assertions

- Or one or more assertions can be returned with status information
- If something went wrong, no assertions are returned, just status
  - Status information can have a complex structure
- Currently the status codes are:
  - Success
  - VersionMismatch
  - Receiver
  - Sender
- Responses are expected to be signed
Response structure
Responses aren’t very exciting to look at. They just contain assertions and some response metadata. I haven’t covered the boxcarring of assertions here.
Agenda

• The problem space
• SAML concepts
  – SAML in a nutshell
  – SAML assertions
  – Producers and consumers of assertions
  – Message exchange protocol
  – Bindings and profiles
• Walking through scenarios
• Status of SAML and related standards efforts
Profiles are sort of like choreography templates. SAML is very flexible; profiles give you a way to standardize on particular flows of SAML-related data in order to secure some other traffic.

The SAML committee is standardizing some of these, but we have every intention of seeing more created by others. We hope that these will be standardized by some body, or at least openly specified.
This just amounts to using SOAP to query a SAML authority and get back a response.
By contrast, the SOAP profile

On the other hand, this is a canonical way to provide assertions about some resource that is *in the body* of the same SOAP document that the assertions are in.

Note that this profile is in some flux.
Web browser profiles

• These profiles assume:
  – A standard commercial browser and HTTP(S)
  – User has authenticated to a local source site
  – Assertion’s subject refers implicitly to the user

• When a user tries to access a target site:
  – A tiny authentication assertion reference travels with the request so the real assertion can be dereferenced
  – Or the real assertion gets POSTed

The web browser profile is really a couple of different sub-profiles. They mostly deal with SSO. The trick is to convey an authentication assertion safely. You can simply POST it, or you can send a little reference to it in a small string such as a URL and then get asked for the real thing.
Future bindings and profiles

- The SAML committee will accept and register proposed new bindings and profiles
- Eventually we may standardize these
- Open publishing of these will at least help interoperability in the meantime
Agenda

- The problem space
- SAML concepts
- Walking through scenarios
  - SSO pull using browser/artifact profile
  - Back office transaction using SOAP binding and SOAP profile
- Status of SAML and related standards efforts
Here’s an example of the full-blown web browser profile at work. Jane User, who logs in to SmithCo.com, wants a research report residing at JonesCo. She clicks on the appropriate link, and then a bunch of stuff happens that is transparent to her but is fairly complex.

1. She logs in to SmithCo.
2. She clicks on what looks like the link to the research report.
3. She gets redirected to a way-station and a little “artifact” gets created that references her authentication assertion.
4. The way-station sends her on her way, with the artifact in tow.
5. The destination site sees the artifact and asks for the real assertion.
6. Having gotten it, the site decides whether to let Jane in.
More on the SSO pull scenario

• “Access inter-site transfer URL” step:
  – User is at: http://smithco.com
  – Clicks on a link that looks like it will take her to http://jonesco.com
  – It really takes her to inter-site transfer URL: https://source.com/intersite?dest=jonesco.com

• “Redirect with artifact” step:
  – Reference to user’s authentication assertion is generated as a SAML “artifact” (8-byte base64 string)
  – User is redirected to assertion consumer URL, with artifact and target attached: https://jonesco.com?SAMLart=<artifact>

Let’s look at some mockups of URLs.
Agenda

- The problem space
- SAML concepts
- Walking through scenarios
  - SSO pull using web browser profile
  - Distributed transaction using SOAP binding and SOAP profile
- Status of SAML and related standards efforts
This scenario has a different function and different players, but the flow is actually basically the same.

This time Joe User of SmithCo is trying to order some supplies from OfficeBarn. Joe knows about OfficeBarn, but OfficeBarn isn’t all that familiar with Joe.

The key here is that both Joe/SmithCo and OfficeBarn have an authentication/attribute authority that they’re both willing to trust.

1. Joe logs in.
2. He asks for and gets the assertions that say he’s logged in (authentication) and that he’s got a corporate spending limit of $500 (attribute).
3. These get attached to the purchase order he sends to OfficeBarn.
4. OfficeBarn looks everything over.
5. Then it acts appropriately.
More on the distributed transaction scenario

- An example of attaching SAML assertions to other traffic
- Asymmetrical relationship is assumed
  - Seller is already known to buyer, but buyer is not known to seller, a common situation
  - E.g., server-side certificates might be used to authenticate seller
- If it were symmetrical, additional SAML steps would happen on the right side too
  - This would likely be a different scenario

The typical business case is asymmetrical in that it involves (fewer) well-known sellers and many buyers, not all of whom have a direct relationship to the seller.

The symmetric case is a different business model. I send my RFQ (Request for Quote) with attached assertions to an aggregator who sends it out to many different business entities. The business entities respond with Quotations and include attached assertions to qualify and validate their quotations. I evaluate the returned Quotations and Assertions and choose a vendor.
Agenda

- The problem space
- SAML concepts
- Walking through scenarios
- Status of SAML and related standards efforts
SAML status

- Work started on 9 January 2001
  - From a base of S2ML and AuthXML
- Five near-final Committee Working Drafts were published 10 January 2002
  - [www.oasis-open.org/committees/security/](http://www.oasis-open.org/committees/security/)
- Implementations are starting to appear
  - JSAML Toolkit from Netegrity
    - [www.netegrity.com](http://www.netegrity.com)
- We plan to request OASIS Standard status on 1 March 2002
  - Please send your comments in!

There are also two other useful documents:

- Requirements and use cases
- Domain model
Important efforts related to SAML

• IETF/W3C XML Signature
  – Built into SAML for digitally signing assertions
  – www.w3.org/Signature/

• W3C XML Encryption and Canonicalization
  – Not quite ready yet, but encryption will be important
    for managing security and privacy risks
  – www.w3.org/Encryption/2001/

• XKMS and its relatives (now at W3C)
  – An XML-based mechanism for doing PKI
  – SAML traffic might be secured by XKMS-based PKI, by
    other PKI, or by other means entirely
  – www.w3.org/TR/xkms/
More efforts related to security and identity

- **OASIS XACML**
  - XML-based (and SAML-based) access control/policy language
  - www.oasis-open.org/committees/xacml/
- **OASIS Provisioning**
  - XML-based framework for user, resource, and service provisioning
  - www.oasis-open.org/committees/provision/
- **Liberty Alliance**
  - Identity solution for SSO of consumers and businesses
  - www.projectliberty.org
- **Internet2**
  - Higher-education effort to develop advanced network applications and technologies; parts will be SAML-compliant
  - http://www.internet2.edu/
Agenda

- The problem space
- SAML concepts
- Walking through scenarios
- Status of SAML and related standards efforts
- Questions?
Thank you

Eve Maler
eve.maler@sun.com