Paweł Rajba
pawel@cs.uni.wroc.pl
http://itcourses.eu/

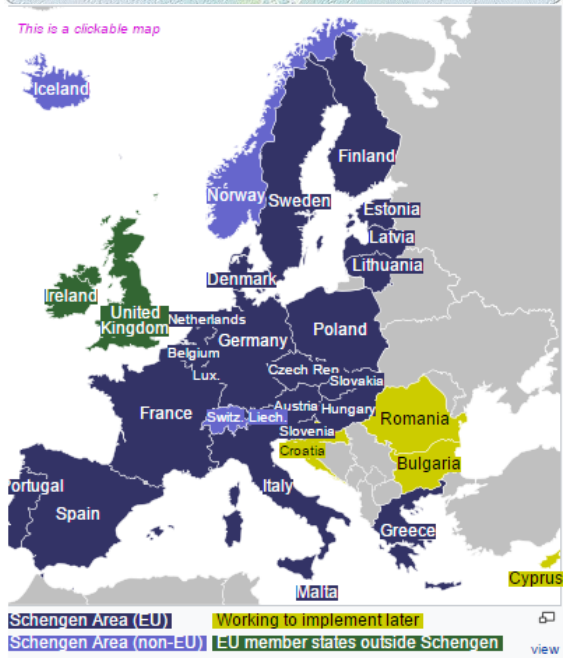
Application Security
Identity Federation

Agenda

- Introduction
- OpenID Connect
- SAML2
- WS-Trust
- WS-Federation

Introduction

- Federation example:
  - Agreement between countries
    - So one can visit another country
    - Level of trust defines the rules
      - Schengen Area vs. North Korea
  - Federated Identity: passport



Introduction

- A federated identity in information technology is the means of linking a person's electronic identity and attributes, stored across multiple distinct identity management systems.

Introduction

- Federation
  - provides a mechanism where one identity is shared in different applications/companies
  - Is based on trust
  - Usually executed by a token-based system
    - E.g. SAML2, Open ID Connect, WS-Trust, WS-Federation
- Federation vs. SSO
  - Federation allows SSO without storing password
    - Otherwise, client needs to authenticate in every app
  - SSO is a subset of Federation
- Central Authentication Service (CAS)
  - Protocol for central authN for web applications
  - Not the same as SSO
    - One can log into APP1 and need to log again into APP2 (but centrally)
  - No federation is possible

Introduction

- In other words:
  - Federation: identity shared between realms
    - For instance: using a username in both companies, the same person is identified (first name, last name, e-mail, birth date, etc.)
  - SSO: user authenticates once for a set of applications
    - Can be implemented in different ways, e.g. based on session ID, token, storing the password, etc.

Introduction

- A scenario
  - 1 user (U) with a browser (B)
  - 2 applications (APP1 and APP2) in different realms
- Federation without SSO:
  - U visits APP1 by the B and make an authN
  - A session is established between B and APP1
  - U visits APP2 by the B and make an authN again
    - Maybe with different password
  - A session is established between B and APP2
  - Where is the additional value?
    - Even if the authN is required twice, the account is available in APP1/2
- Federation with SSO
  - The same, but in step 3 authN is not required again
- What about SSO without Federation?
  - Credentials stored on the client
  - The same usernames are used in APP1 and APP2, but they are not federated

Introduction

- Basic terminology
  - IdP: Identity Provider
    - Authorization Server in Auth2
  - RP: Relying Party
    - Client in Auth2
  - STS: Security Token Service
    - Authorization Server in Auth2

Introduction

- Typical use-cases or challenges
  - Cross-domain
  - Web-based single sign-on
  - Cross-domain user account provisioning
  - Cross-domain entitlement management
  - Cross-domain user attribute exchange.

Introduction

- Some products supporting federation
  - Oracle Identity Federation
  - PingFederate Federation Server
  - Tivoli Federated Identity Manager (IBM)
  - AWS Identity and Access Management (IAM)
  - Identity Federation and Remote Access (F5)
  - CA Single Sign-On
  - Microsoft Azure Access Control Service
  - NetIQ Access Manager

Introduction

- Let's take a closer look on
  - Open ID Connect
  - SAML 2.0
- … and a quick look on
  - WS-Trust
  - WS-Federation

OpenID Connect

Authentication & OAuth2

- It is quite popular that OAuth2 is abused for authentication
- The most common scenario is as follows:
  - User authenticates on AS
  - Afterwards an application exchange code for access token
  - The assumption is that if the application is able to get data using access token, then it means that user properly authenticated on AS

Authentication & OAuth2

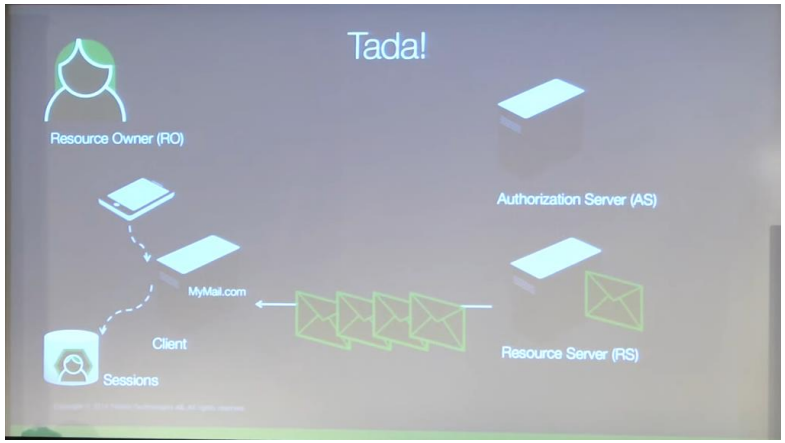
- Main problems
  - OAuth2 is an authorization framework, there is no flow related to authentication
    - Although authentication is a part of the OAuth2 flow
  - The focus is on the client application, not on a user
    - In other words, authorization is for the client application, not for the user
    - After getting an access token, user is no more involved

Authentication & OAuth2

- Main issue with applying OAuth2 for authN
  - The goal is to provide a token which allows to get specific information
    - As a result there is only an access_token
    - There is no information about the user
  - If another app gets the token, only can obtain the same data
    - If used for authentication, app can impersonate the user
    - There is no additional verification who is the proper receiver of the token

Authentication & OAuth2

- An example where OAuth2 is not enough
  - Application get e-mails
    - … but wants to not only show them, but also .e.g translate and store in the application
  - In this scenario we need identity, not only accesses
  - ID is never sent outside the application
  - And we don't built any auth services locally



https://www.youtube.com/watch?v=BdKmZ7mPNns

Authentication & OAuth2

- Very good considerations
  - OAuth 2.0 and Sign-In
    by Vittorio Bertocci
    - http://www.cloudidentity.com/blog/2013/01/02/oauth-2-0-and-sign-in-4/
  - The problem with OAuth for Authentication
    by John Bradley
    - http://www.thread-safe.com/2012/01/problem-with-oauth-for-authentication.html

Source: http://nat.sakimura.org/2012/01/20/openid-connect-nutshell/

OpenID Connect

- The solution is the OpenID Connect
  - An authentication protocol built on top of OAuth2
    - We can consider OpenID Connect as a OAuth2 profile which defines a flow for authentication
  - Allows to get the information about the user
    - Adds ID Token where this information is stored
  - Emerging protocol, but has many implementations
    - Google is probably the best one
  - The main website: http://openid.net/connect/
  - A very good introduction
    - http://nat.sakimura.org/2012/01/20/openid-connect-nutshell/
- Let's see the presentation video
  - https://www.youtube.com/watch?v=Kb56GzQ2pSk
    - We will use the offline mode ☺

OpenID Connect Request

- To make a request the following information is required
  - Client ID
  - Client Secret
  - End-user authorization endpoint
  - Token endpoint
  - User info endpoint
- Additionally:
  - grant_type = token id_token
  - scope = openid profile email …

OpenID Connect Request

- GET
  - /authorize?grant_type=token%20id_token&
    scope=openid%20proflie&
    redirect_uri=https%3A%2F%2Fclient%2Eexample
    %2Ecom%2Fcb
    HTTP/1.1
  - Host: server.example.com

OpenID Connect Response

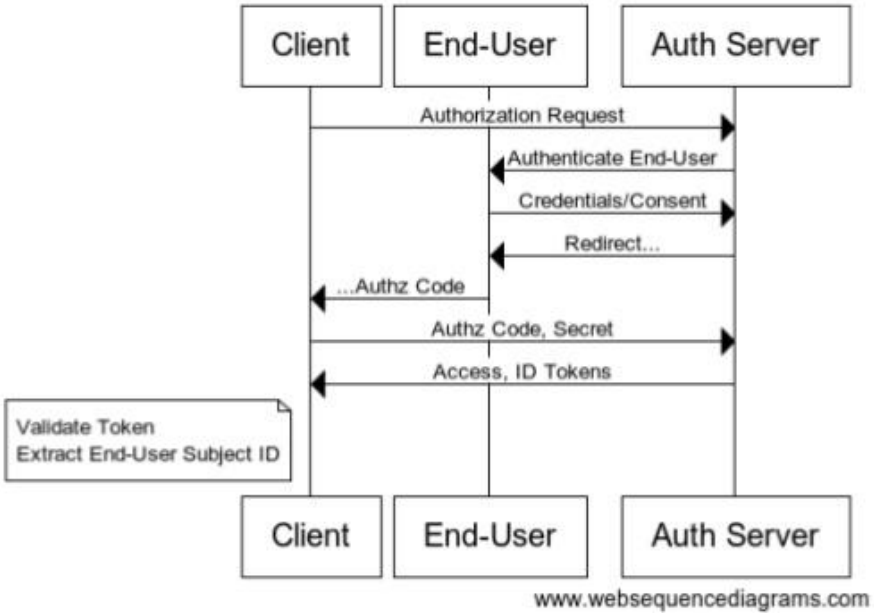
- Beside access_token included in OAuth2 response, one gets id_token with the following information
  - aud (audience)
    - The client_id that this id_token is intended for.
  - exp (expiration)
    - The time after which this token must not be accepted
  - sub (subject)
    - A locally unique and never reassigned identifier for the user (subject)
    - E.g. "24400320" or "AItOawmwtWwcTok51BayewNvutrJUqsvl6qs7A4".
  - iss (issuer)
    - A https: URI specifying the fully qualified host name of the issuer, which when paired with the user_id, creates a globally unique and never reassigned identifier.
    - E.g. "https://aol.com", "https://google.com", or "https://sakimura.org".
  - nonce - nonce value sent in the request.
- All these parameters are required

OpenID Connect Rules

- The following rules should be applied
  - An authorization server must only issue assertions about user identifiers within its domain
  - The client MUST verify that the aud matches its client_id and iss matches the domain (including sub-domain) of the issuer of the client_id
  - The authorization server is responsible for managing its own local namespace and enforcing that each user_id is locally unique and never reassigned
  - When the client stores the user identifier, it MUST store the tuple of the user_id and iss. The user_id MUST NOT be over 255 ASCII characters in length

OpenID Connect Profiles

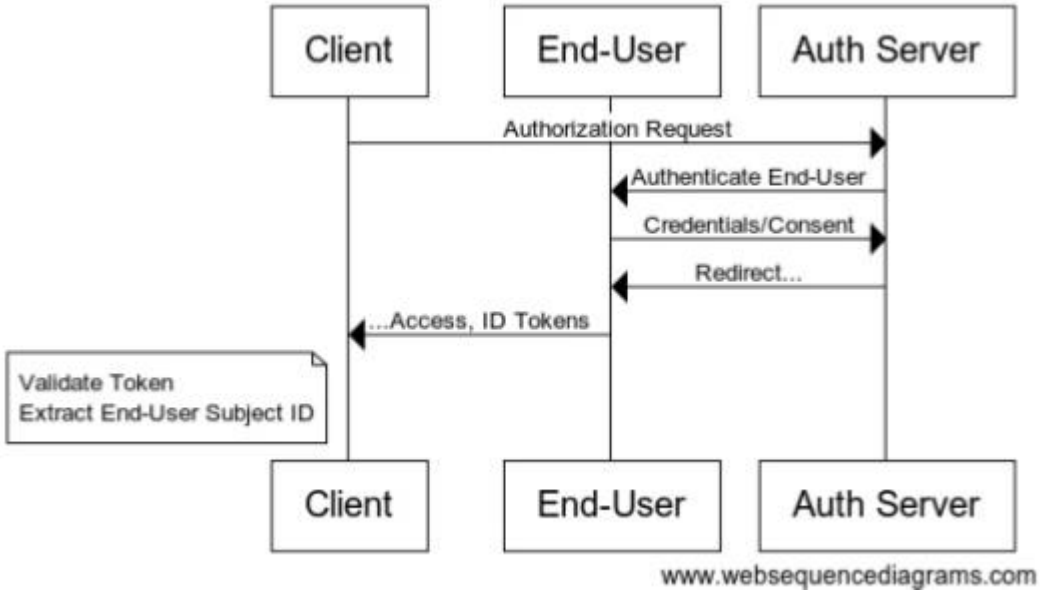
- Basic Client Profile
  - Based on OAuth2 code flow
  - Designed for a web-based relying parties
  - Subset of OpenId Connect Core specification
  - More: http://openid.net/specs/openid-connect-basic-1_0.html



Source: http://www.slideshare.net/metadaddy/openid-connect-an-overview

OpenID Connect Profiles

- Implicit Client Profile
  - Based on OAuth2 implicit flow
  - Designed for a web-based relying parties
  - Subset of OpenId Connect Core specification
  - More: http://openid.net/specs/openid-connect-implicit-1_0.html



Source: http://www.slideshare.net/metadaddy/openid-connect-an-overview

OpenID Connect Discovery and dynamic registration

- Discovery
  - Allows client app to
    - determine the identity of the End-User
      - Based on authentication performed in Authorization Server
    - obtain a basic profile a of End-User
  - Uses WebFinger (RFC7033)
  - More: https://openid.net/specs/openid-connect-discovery-1_0.html
- Registration
  - Allows client app to register on the server
  - More: http://openid.net/specs/openid-connect-registration-1_0.html

OpenID Connect Playground

- A very good open source provider and a set of samples
  - https://identityserver.github.io/Documentation/
- Getting started videos
  - Introduction into the topic
    - https://vimeo.com/113604459
  - Provider introduction
    - http://vimeo.com/91397084
  - Walkthrough samples
    - http://vimeo.com/91405115

SAML2

Introduction

- Security Assertion Markup Language
- XML based protocol
- OASIS standard
  - SAML 1.0: 2002
  - SAML 1.1: 2003
  - SAML 2.0: 2005
- Flexible and extensible protocol

Definitions

- **Entity (or system entity)**: An active element of a computer/network system
- **Principal**: An entity whose identity can be authenticated
- **Subject**: A principal in the context of a security domain

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Definitions

- **Identity**: The essence of an entity, often described by one's characteristics, traits, and preferences
  - **Anonymity**: Having an identity that is unknown or concealed
- **Identifier**: A data object that uniquely refers to a particular entity
  - **Pseudonym**: A privacy-preserving identifier
- **Federated identity**: Existence of an agreement between providers on a set of identifiers and/or attributes to use to refer to a principal
  - **Account linkage**: Relating a principal's accounts at two different providers so that they can communicate about the principal

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

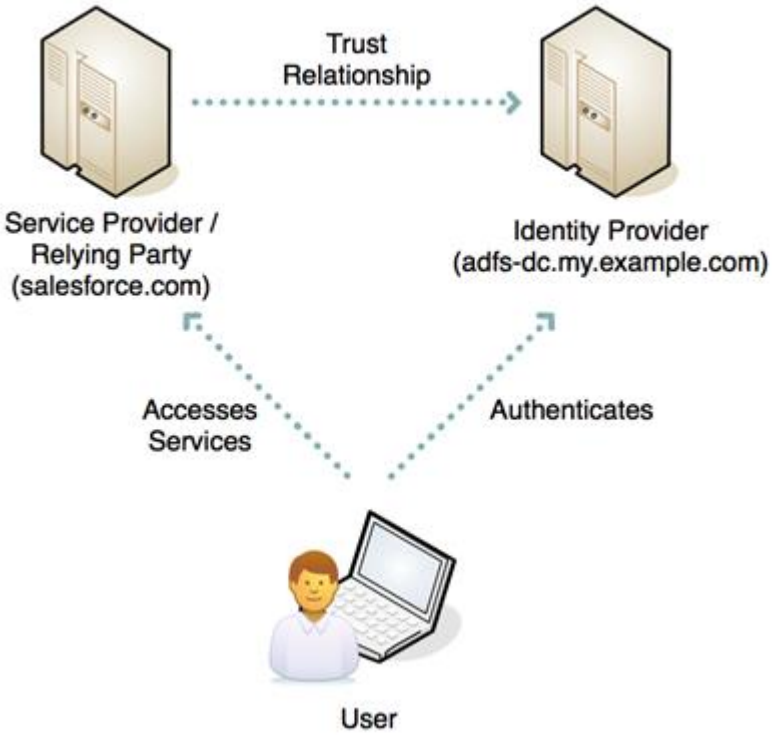
Definitions

- **Asserting party (SAML authority)**: An entity that produces SAML assertions
  - **Identity provider**: An entity that creates, maintains, and manages identity information for principals and provides principal authentication to other service providers
- **Relying party**: An entity that decides to take an action based on information from another system entity
  - **Service provider**: An entity that provides services to principals or other entities

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Roles & relationship

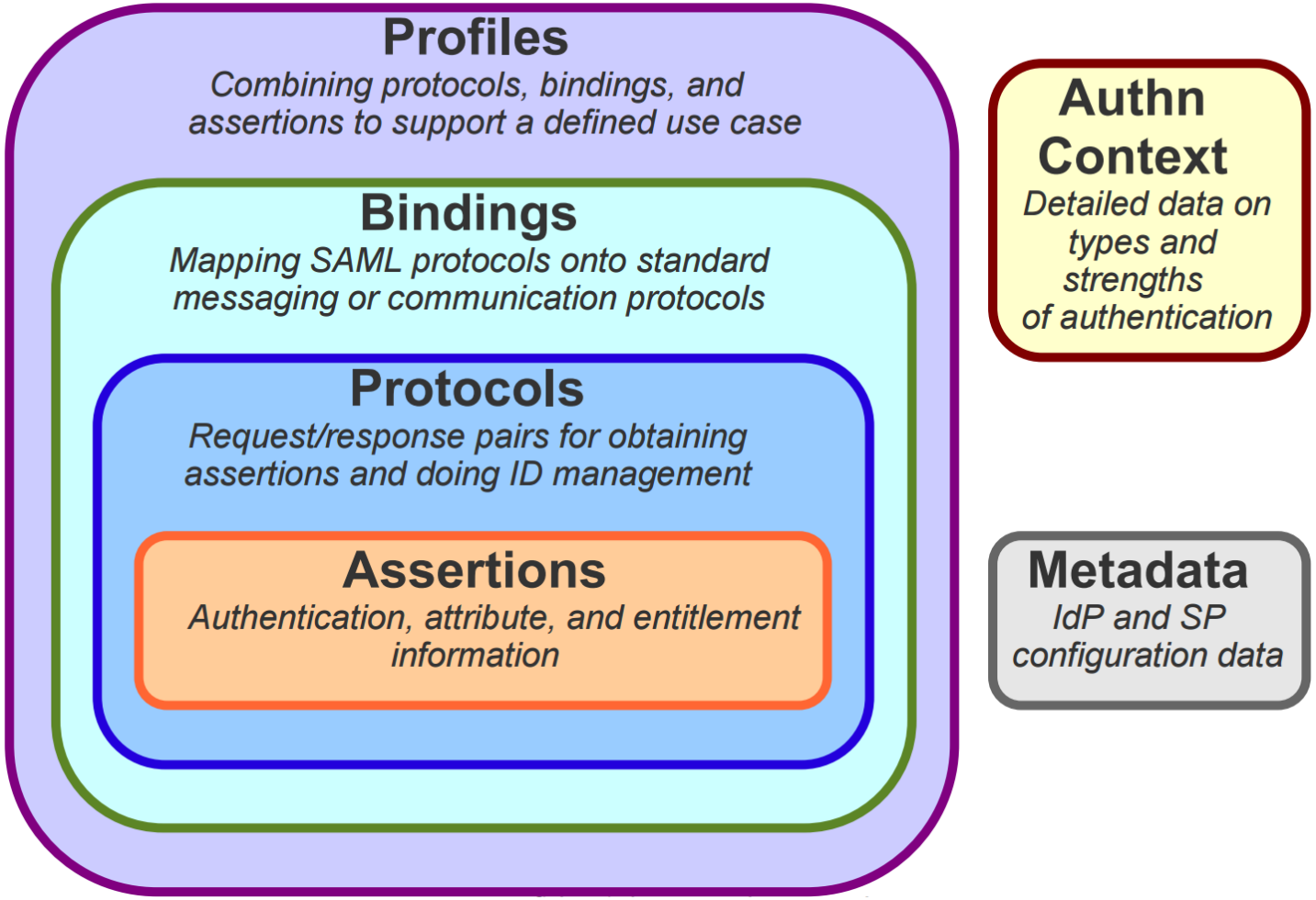
- User
  - Subject, principal
- Identity Provider
  - Asserting party
- Service Provider
  - Relying party



Main features

- Identity Federation
- SSO / Single Sign-Out
- Securing Web Services
- Attribute Services

SAML Concepts



https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

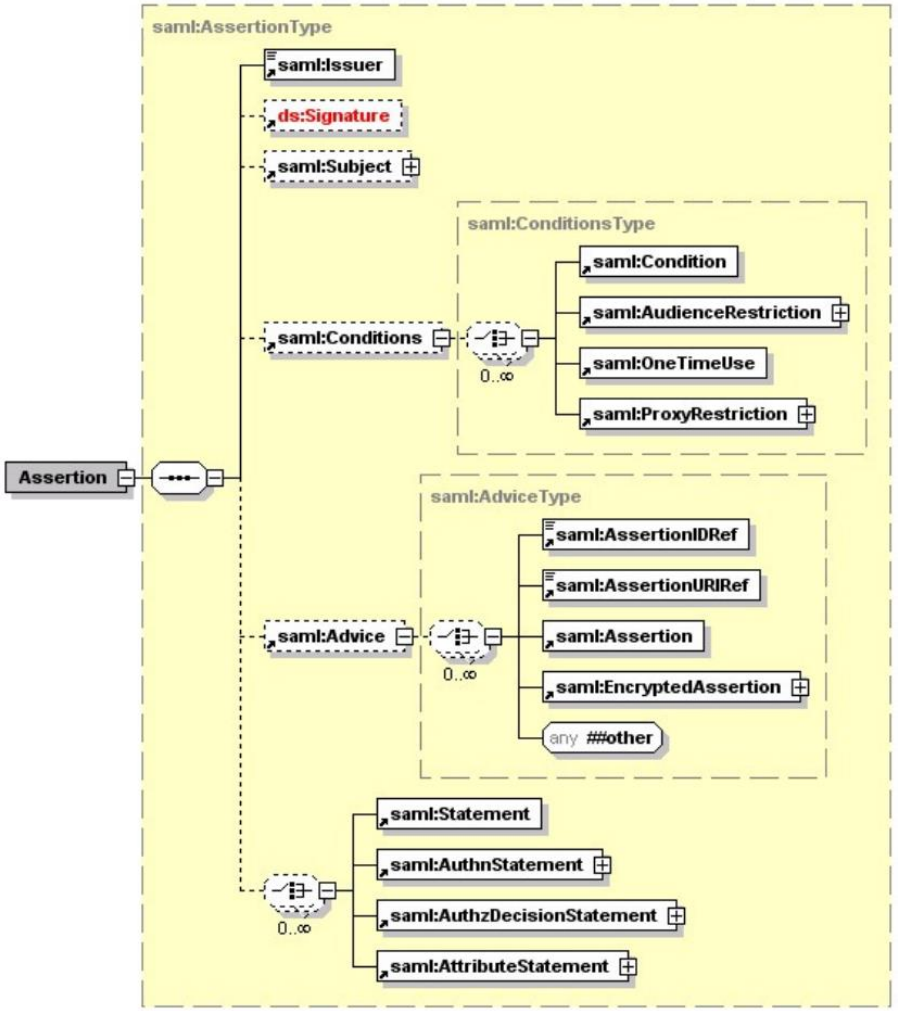
Assertions

- An assertion is a declaration of fact, according to someone
- SAML assertions contain one or more statements about a subject:
  - Authentication statement
    - Joe authenticated with a password at 9:00am
  - Attribute statement (which itself can contain multiple attributes):
    - Joe is a manager with a $500 spending limit

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Assertions

- Structure



https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Assertions

- Example

```
<saml:Assertion
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   ID="b07b804c-7c29-ea16-7300-4f3d6f7928ac"
   Version="2.0"
   IssueInstant="2004-12-05T09:22:05">
   <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
   <ds:Signature
     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
   <saml:Subject>
     <saml:NameID
       Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
       3f7b3dcf-1674-4ecd-92c8-1544f346baf8
     </saml:NameID>
     <saml:SubjectConfirmation
       Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
       <saml:SubjectConfirmationData
         InResponseTo="aaf23196-1773-2113-474a-fe114412ab72"
         Recipient="https://sp.example.com/SAML2/SSO/POST"
         NotOnOrAfter="2004-12-05T09:27:05"/>
     </saml:SubjectConfirmation>
   </saml:Subject>
```

```
   <saml:Conditions
     NotBefore="2004-12-05T09:17:05"
     NotOnOrAfter="2004-12-05T09:27:05">
     <saml:AudienceRestriction>
       <saml:Audience>https://sp.example.com/SAML2</saml:Audience>
     </saml:AudienceRestriction>
   </saml:Conditions>
   <saml:AuthnStatement
     AuthnInstant="2004-12-05T09:22:00"
     SessionIndex="b07b804c-7c29-ea16-7300-4f3d6f7928ac">
     <saml:AuthnContext>
       <saml:AuthnContextClassRef>
         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
       </saml:AuthnContextClassRef>
     </saml:AuthnContext>
   </saml:AuthnStatement>
   <saml:AttributeStatement>
     <saml:Attribute
       xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
       x500:Encoding="LDAP"
       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
       FriendlyName="eduPersonAffiliation">
       <saml:AttributeValue
         xsi:type="xs:string">member</saml:AttributeValue>
       <saml:AttributeValue
         xsi:type="xs:string">staff</saml:AttributeValue>
     </saml:Attribute>
   </saml:AttributeStatement>
</saml:Assertion>
```

https://en.wikipedia.org/wiki/SAML_2.0

Assertions

- Example explained
  - In words, the assertion encodes the following information:
    - The assertion "b07b804c-7c29-ea16-7300-4f3d6f7928ac" was issued at time "2004-12-05T09:22:05Z" by identity provider (https://idp.example.org/SAML2) regarding subject (3f7b3dcf-1674-4ecd-92c8-1544f346baf8) exclusively for service provider (https://sp.example.com/SAML2).
  - The authentication statement, in particular, asserts the following:
    - The principal identified in the <saml:Subject> element was authenticated at time "2004-12-05T09:22:00" by means of a password sent over a protected channel.
  - Likewise the attribute statement asserts that:
    - The principal identified in the <saml:Subject> element is a staff member at this institution.

https://en.wikipedia.org/wiki/SAML_2.0

Artifacts

- A SAML message is transmitted from one entity to another either by value or by reference.
  - A **reference** to a SAML message is called an **artifact**.
- The receiver of an artifact resolves the reference by sending a request directly to the issuer of the artifact
- Sending references may have sources in:
  - Technical constraints, e.g. limited length of URL
  - Security reasons, e.g. to not expose secret data to a browser

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Artifacts

- So, artifacts are a small, fixed-size, structured data object pointing to a typically larger, variably sized SAML protocol message
- Designed to be embedded in URLs and conveyed in HTTP messages
- Allows for "pulling" SAML messages rather than having to push them
- SAML defines one artifact format but you can roll your own

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Protocols

- Assertion query and request
  - Query for existing assertion based on simple reference, subject-matching, or statement type, e.g. by <AssertionIDRequest>
- Authentication request (the most important one)
  - SP requests a fresh authn assertion that adheres to various requirements (specified by means of Authentication Context)
- Artifact resolution ("meta-protocol")
  - Dereferences an artifact to get a protocol message
- Name identifier management
  - IdPs and SPs inform each other of changes to their mutual understanding of what a principal's name is
- Name identifier mapping
  - Privacy-preserving way for two SPs to refer to the same principal, e.g. by obtaining encrypted ID <saml:EncryptedID>
- Single logout
  - Signals to all SPs using the same session to drop the session

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Bindings

- SOAP
  - Basic way for IdPs and SPs to send SAML protocol messages
- Reverse SOAP (PAOS)
  - Multi-stage SOAP/HTTP exchange that allows an HTTP client to send an HTTP request containing a SOAP response
- HTTP redirect
  - Method to send SAML messages by means of HTTP 302
- HTTP POST
  - Method to send SAML messages in base64-encoded HTML form control
- HTTP artifact
  - Way to transport an artifact using HTTP in two ways: URL query string and HTML form control
- URI
  - How to retrieve a SAML message by resolving a URI

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Profiles

- Web browser SSO
  - SSO using standard browsers to multiple SPs: profiles Authn Request protocol and HTTP Redirect, POST, and artifact bindings
- Enhanced client and proxy (ECP)
  - SSO using ECPs: profiles Authn Request protocol and SOAP and PAOS bindings
- IdP discovery
  - One way for SPs to learn the IdPs used by a principal
- Single logout
- Name identifier management
  - Profiles the NIM protocol with SOAP, HTTP redirect, HTTP POST, and HTTP artifact bindings
- Artifact resolution
- Assertion query/request

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

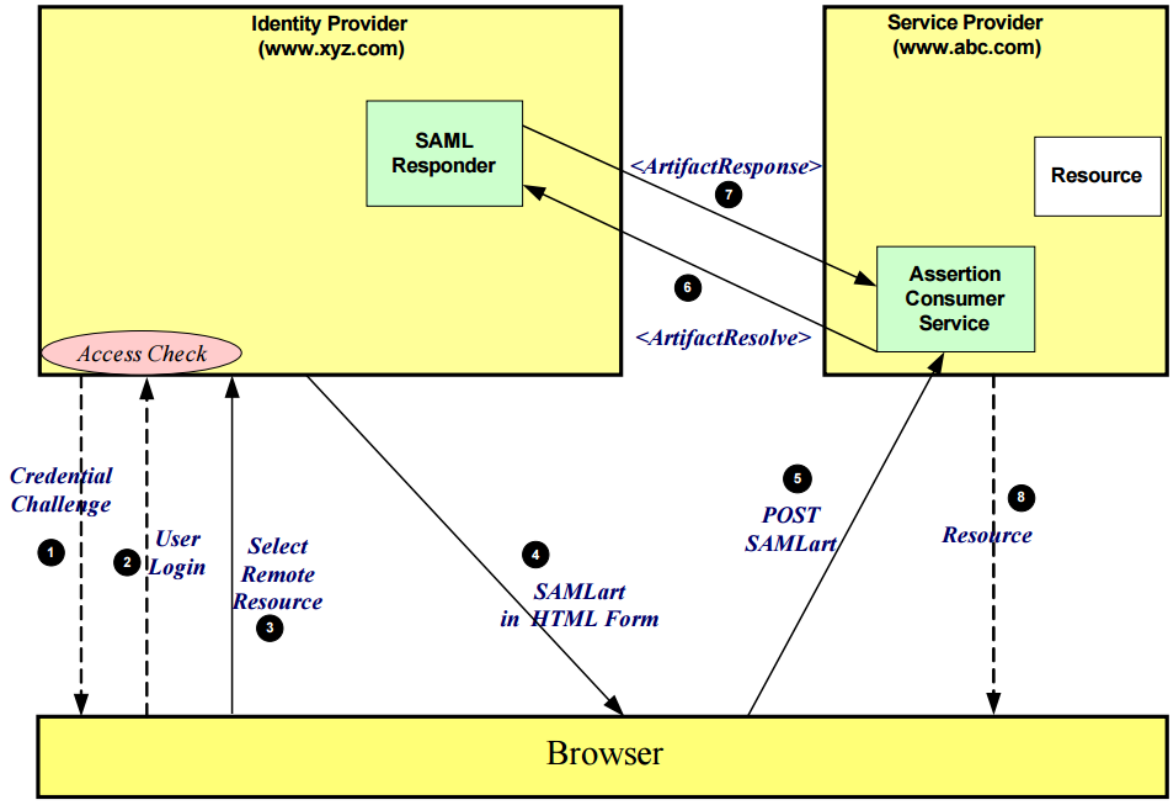
Profiles

- Within profiles, different flows and binding choices are possible
  - E.g., in the web browser SSO profile:
    - Authn request from SP to IdP can use any of HTTP redirect or HTTP POST or HTTP artifact
    - IdP response to SP can use either HTTP POST or HTTP artifact
  - E.g., in the ECP SSO profile using the PAOS binding, two flows are possible:
    - ECP to SP, SP to ECP to IdP
    - IdP to ECP to SP, SP to ECP

https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Profiles

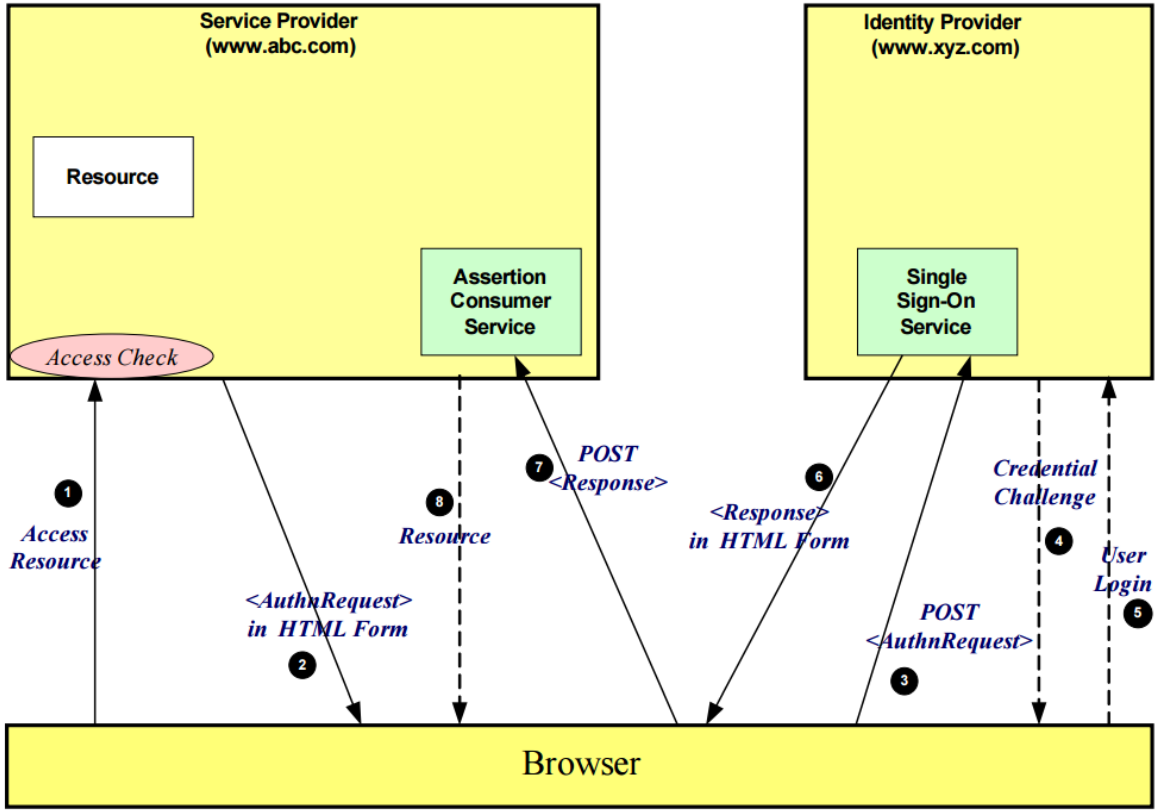
- Example 1: Browser/artifact flow, IdP-initiated



https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Profiles

- Example 2: Browser/POST flow, SP-initiated



https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf

Profiles

- More details one can see at Wikipedia
  - https://en.wikipedia.org/wiki/SAML_2.0#SAML_2.0_Profiles

Authentication context classes

- Internet Protocol
- Internet Protocol Password
- Kerberos
- Mobile One Factor Unregistered
- Mobile Two Factor Unregistered
- Mobile One Factor Contract
- Mobile Two Factor Contract
- Password
- Password Protected Transport
- Previous Session
- Public Key – X.509
- Public Key – PGP
- Public Key – SPKI
- Public Key – XML Signature
- Smartcard
- Smartcard PKI
- Software PKI
- Telephony
- Nomadic Telephony
- Personalized Telephony
- Authenticated Telephony
- Secure Remote Password
- SSL/TLS Cert-Based Client Authn
- Time Sync Token
- Unspecified

SAML 2.0 Metadata

- Provide information about entities in the flow
  - Identity Provider Metadata
    - SSO Service Metadata
  - Service Provider Metadata
    - Assertion Consumer Service Metadata
- The information allows to
  - Check correctness of service and identity providers
    - e.g. there is no phishing on the line
  - Validate the asserations based on the public keys
  - Find endpoint to resove artifacts

References

- SAML V2.0 Basics
  - https://www.oasis-open.org/committees/download.php/12958/SAMLV2.0-basics.pdf
- Wikipedia
  - https://en.wikipedia.org/wiki/SAML_2.0
- SAML 2.0 Core
  - https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
- SAML 2.0 Bindings
  - https://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
- Profiles for the OASIS SAML V2.0
  - https://svn.softwareboersen.dk/sosi-gw/tags/v1.0.1/vendor/doc/saml-profiles-2.0-os.pdf
- Profiles exaplained
  - https://help.scorpionsoft.com/hc/en-us/articles/218317597-SAML-2-0-Profiles-explained-Building-your-own-SAML-integrations
- ECP Profile
  - https://indico.egi.eu/indico/event/1019/session/46/contribution/262/material/slides/0.pdf

WS-Trust

WS-Trust

- Actors & scenario example:
  - A wine web service (W-WS) with a policy
    - Policy says that a SAML token is required with
      - Age
      - Department Of Driving License (DODL)
  - A DODL web service (D-WS) with a policy
  - A user (U) who wants wine
- Every actor has a certificate with a private key

Based on: http://channel9.msdn.com/Shows/Going+Deep/Vittorio-Bertocci-WS-Trust-Under-the-Hood

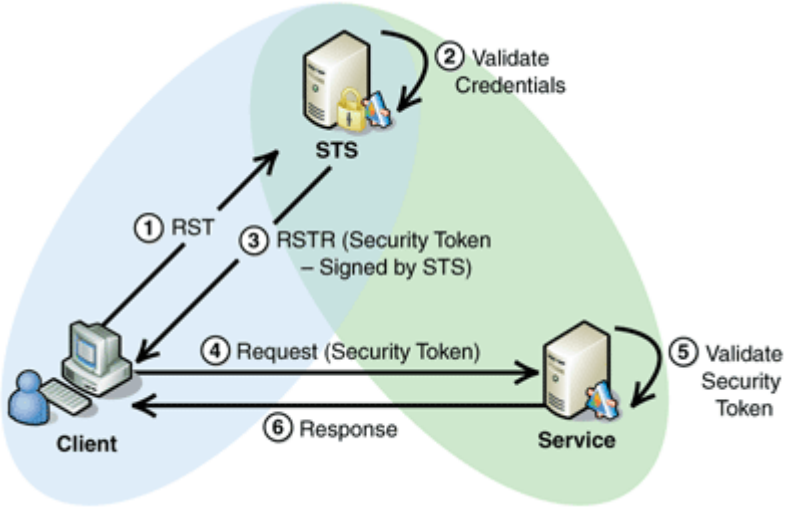
WS-Trust

- Dedicated to SOAP Web Services
- Based on
  - WS-Security
    - message authenticity, integrity, confidentiality
  - WS-SecurityPolicy
    - description of the security requirements of services via assertions about the security mechanisms of the services (i.e. algorithms and types of tokens that the service accepts).
- WS-Trust adds
  - Security Token Service
  - Protocol for requesting/issuing security tokens used by WS-Security and described by WS-SecurityPolicy

http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-FederationSpec05282007.pdf?S_TACT=105AGX04&S_CMP=LP

WS-Trust

- The flow (simplified)
  - U gets metadata from W-WS
  - U asks D-WS for a security token which fulfill policy
  - U authenticates and gets the security token
  - U uses the security token and buy a wine in W-WS



WS-Trust

- Terminology
  - D-WS we usually call Security Token Service (STS)
    - Or Identity Provider (IP)
  - W-WS we usually call Relying Party (RP)
  - U we usually call client

WS-Trust References

- A very good video
  - http://channel9.msdn.com/Shows/Going+Deep/Vittorio-Bertocci-WS-Trust-Under-the-Hood
- Some introductions
  - http://fusesource.com/docs/esb/4.4.1/cxf_security/WsTrust-Intro.html
  - http://msdn.microsoft.com/en-us/library/bb498017.aspx
  - http://msdn.microsoft.com/en-us/library/ff650503.aspx
  - http://documentation.progress.com/output/Iona/artix/5.5/security_guide_java/WsTrust-SSO-Example.html
- How to create a STS
  - http://msdn.microsoft.com/en-us/magazine/dd347547.aspx

WS-Federation

WS-Federation

- Federation
  - A collection of domains with a trust
  - Allows interactions between users, applications and other players
- Main Goal of WS-Federation
  - Simplify the development of federated services (FS) through cross-realm communication and management of Federation Services
  - Re-using the WS-Trust STS model and protocol.
  - Single Sign-On inside trust boundaries

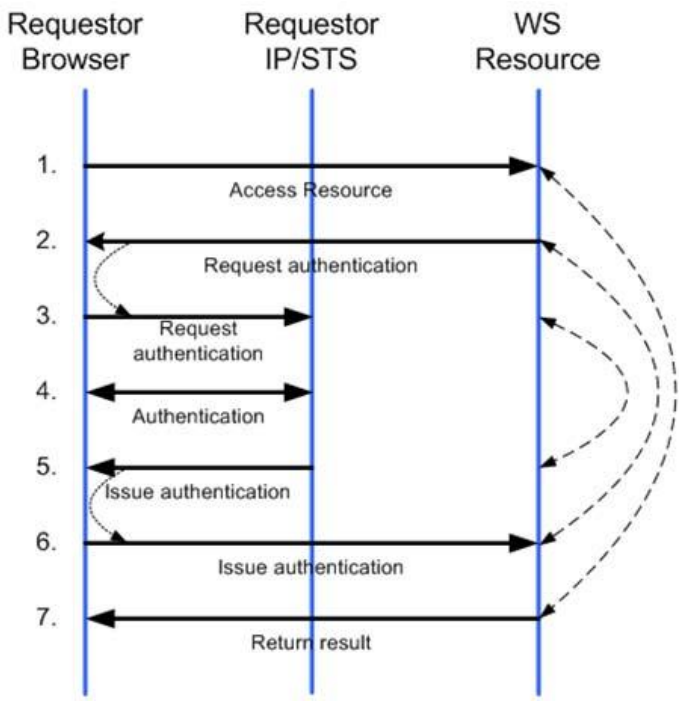
Based on:
http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf
http://www.cs.virginia.edu/~acw/security/doc/Tutorials/WS-Federation.ppt

WS-Federation

- WS-Trust makes possible to have a basic federation between IdP and RP
- WS-Federation
  - Adds Federation Metadata to simplify the setup of federated trust relationship between parties
  - Adds Single Sign On & Single Sign Off
  - Adds profiles for classic web applications
  - Adds mechanism for better discovery
  - Adds services for attributes and pseudonyms
  - Adds claims transformation

WS-Federation Profiles

- Active Requestor Profile
  - Focus on SOAP Web Services
- Passive Requestor Profile
  - Dedicated for browser client
  - Based on URLs
  - Uses redirections to send messages

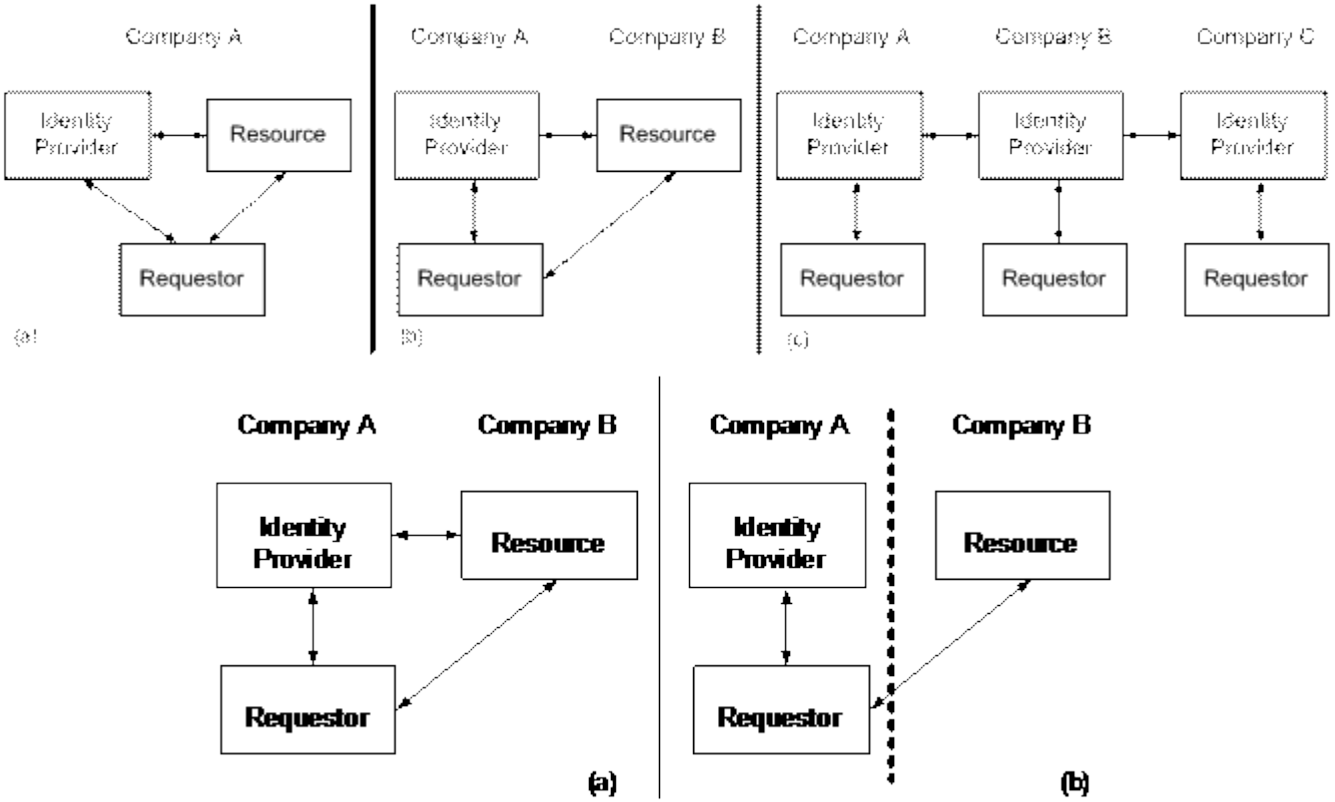


WS-Federation

- Architecture of federation should be able to
  - Model business requirements
  - Leverage existing infrastructure
- Main trust topologies
  - Direct trust
    - Exchange
    - Validation
  - Indirect trust
  - Delegation

WS-Federation

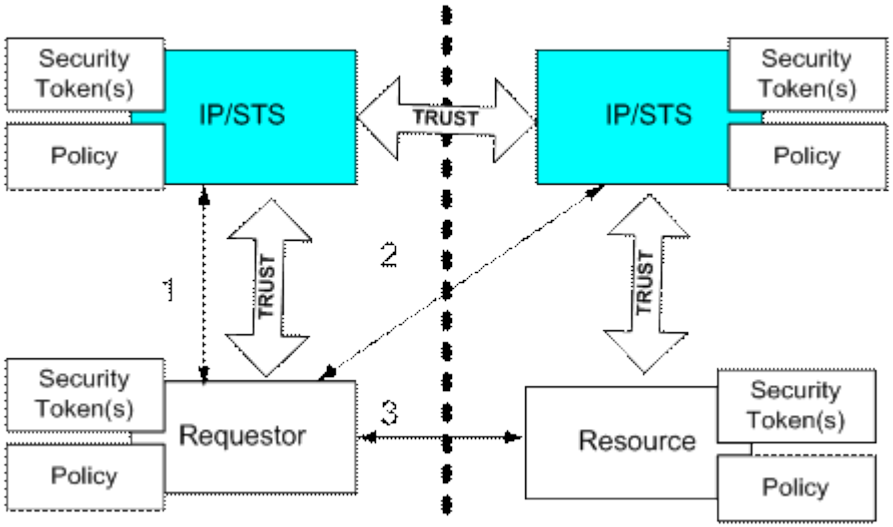
- Supports different scenarios



(a) Direct connection (b) Firewall in between, trust by using certificates

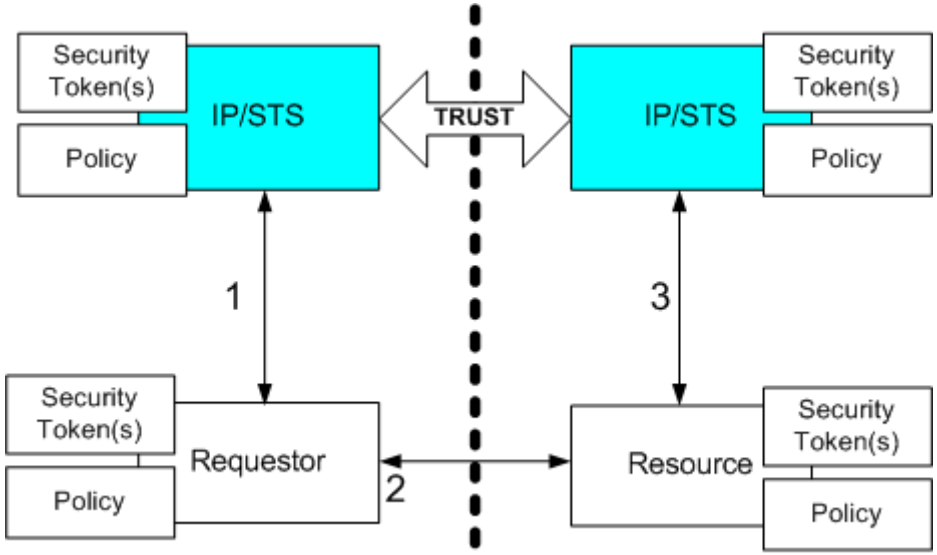
WS-Federation Trust Topologies

- Direct trust with token exchange



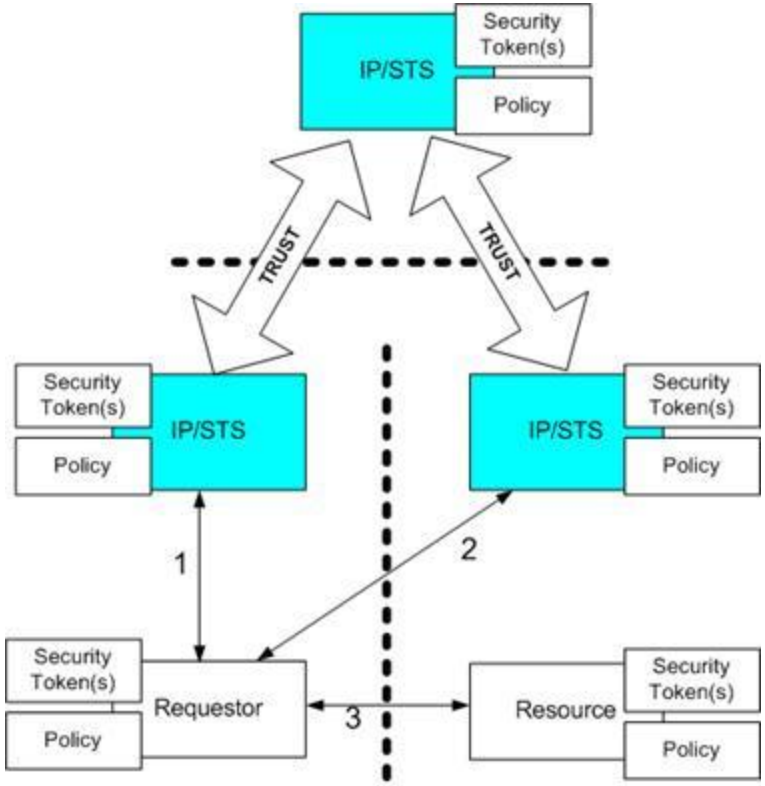
WS-Federation Trust Topologies

- Direct trust with token validation



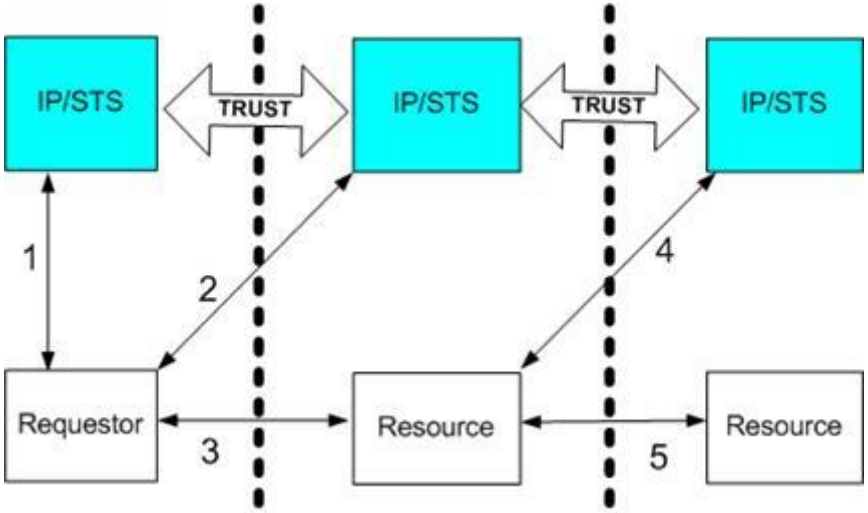
WS-Federation Trust Topologies

- Indirect trust



WS-Federation Trust Topologies

- Delegation



WS-Federation References

- Documentation
  - Web Services Federation Language Version 1.2
    http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf
- Tutorials & presentation
  - Understanding WS-Federation
    http://msdn.microsoft.com/en-us/library/bb498017.aspx
  - Claims-Based Architectures
    http://msdn.microsoft.com/en-us/library/ff359108.aspx
  - WS-Federation presentation
    http://www.cs.virginia.edu/~acw/security/doc/Tutorials/WS-Federation.ppt

Summary

- In this presentation we've covered
  - Open ID Connect, SAML2, WS-Trust, WS-Fed
- The main goals in those protocols
  - Authenticate
  - Express statements about the subject
  - Support federation
  - Support different scenarios
    - In many cases the same ones