
Dagens tema er “Hvordan løse Web Browser SSO vha SAML”.

På forespørsel fra Trygve har jeg blitt bedt om å holde et lite foredrag om SAML og gi en liten demo på hvordan det kan brukes for web single-sign on.
Jeg har ikke jobbet spesielt mye med SAML, men har sett litt på det I forbindelse med noen små-prosjekter på jobben.

Noen ord om single-sign on:  (tror jeg dropper å si noe av dette..)

Det sier seg vel egentlig selv hvorfor vi ønsker å ha single sign on for våre web applikasjoner. Jeg trenger vel egentlig ikke å gå nærmere inn på det nå.

Om noen skulle trenge noen argumenter kan jeg nevne:
1. brukeren trenger bare å huske ett brukernavn/passord
2. slipper å bruke tid på å skrive inn credentials mange ganger
3. mindre administrasjon

Det har ikke bare fordeler.. det finnes også noen ulemper:
1. sentraliseringen av autentisering gjør systemet sårbart; oppetid og redundans må være god
2. større konsekvenser dersom en bruker mister brukernavn og passord


Jeg er da Jan Asle og jobber til daglig i TANDBERG. For tiden er jeg utvikler i et prosjekt som driver r


Det finnes mange alternativer som kan brukes for å løse single-sign on problemet.

Man kan for eksempel implementere en NTLM-basert løsning vha jCIFS. JCIFS er et åpent kildekode prosjekt som tilbyr NTLMv1 autentisering. Det var også støttet i spring security, men ble fjernet i spring security 3.0. NTLMv1 er altfor usikkert og fra og med Vista og Windows Server 2008 er det skrudd av by-default og man anbefaler ntlmv2 og kerberos istedet.

NTLMv2 kan implementeres vha et kommersielt prosjekt som heter Jespa. Jespa bruker NETLOGON servicen på Active Directory for å validere bruker credentials. Du kan riktignok laste ned en gratis variant, men denne støtter ikke mer enn 25 brukere.

Kerberos er en annen mulighet. Her kan man for eksempel bruke Java's GSS-API og implementere hele greia selv. Det finnes også en extension til Spring Security 3.0 som tilbyr kerberos-støtte. Her må du også inn med SPNEGO for å transportere tickets I en web context.(støttet I java 1.6 I gss-api). Ulemper: konfigurasjon kan være noe dritt, api er vel ikke det mest elegante som finnes.

Andre alternativer som bør nevnes er feks CAS og OpenSSO. Ticket basert.

OpenID: er også et alternativ. Jeg har ikke sett noe særlig på OpenID, men den virker å være litt “enklere” enn SAML siden den sender meldinger som key-value par og ikke benytter seg av XML og XML Signering/kryptering.  Den skal visstnok også være enklere å deploye.

Men, hovedfokus akkurat nå er SAML


Hva med SAML? Kan det brukes for å realisere web sso?

SAML er en XML protokoll for å utveksle autentiserings- og autorisasjonsinformasjon mellom sikkerhetsdomener. Det er et produkt av OASIS Security Services Technical Committee og ble utviklet spesielt for å løse web browser single-sign on problemet. Den finnes både en 1.0, 1.1 og 2.0. 2.0 løser flere av problemene med 1.x og er å anbefale.

Det er støttet av flere store aktører som feks Google (Google App Engine), Microsoft (AD Federation Service).

Støtter ikke bare sso, men også federation. Det gjør det enklere å kommunisere autentiseringsinformasjon mellom enterpriser.

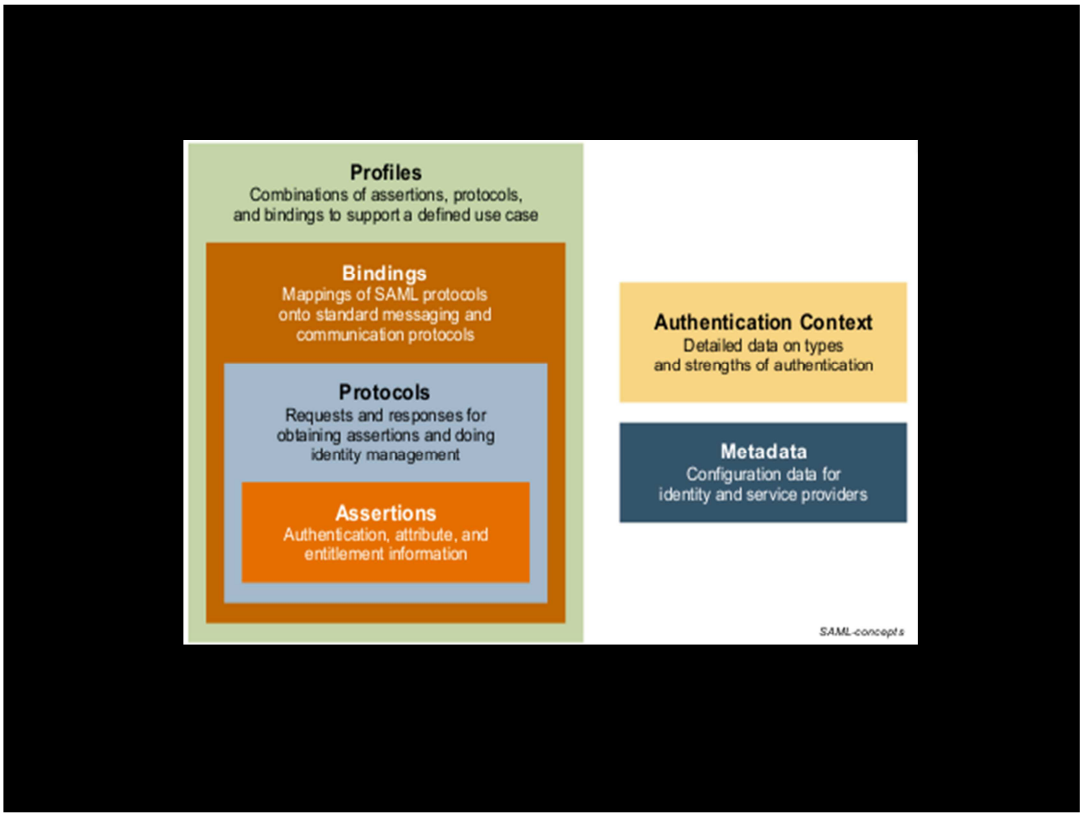
Hovedargumenter for å bruker SAML istedet for noen av de andre vi nevnte:

• Bruker ikke cookies. Cookies kan ikke sendes mellom DNS domener uten en eller annen proprietær løsning.

• Automagisk interoperabilitet; de fleste produktene som løser SSO bruker proprietære protokoller. Dette krever at det samme SSO produktet må brukes mellom domener/enterpriser.

• Federation blir enklere. Utveksling av autentiseringsdata mellom organisasjoner blir enklere.

• hver idp kan ha sin egen autentiseringsmekanisme, kan bruke hva som helst: sertifikater, to-faktor, etc



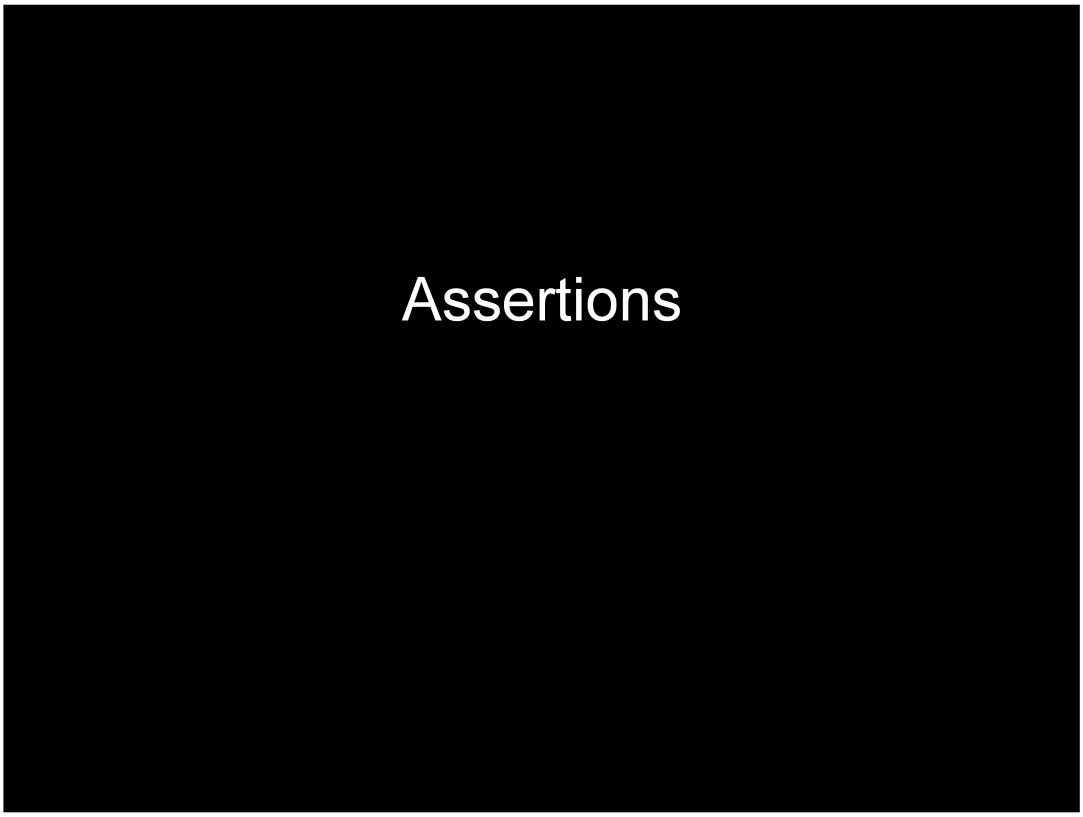

Denne figuren representerer de viktigste konseptene/komponentene I SAML.

En ting den ikke spesifikt nevner er IdP og SP. Dette er et viktig skille I SAML, identity provideren vil typisk være feks opensso eller AD, mens service provider vil være web appen din.

De vi ser på venstre siden skal vi gå igjennom I tur og orden på de neste slidene. På høyresiden finner vi to andre elementer vi trenger for å bygge og deploye et saml miljø.

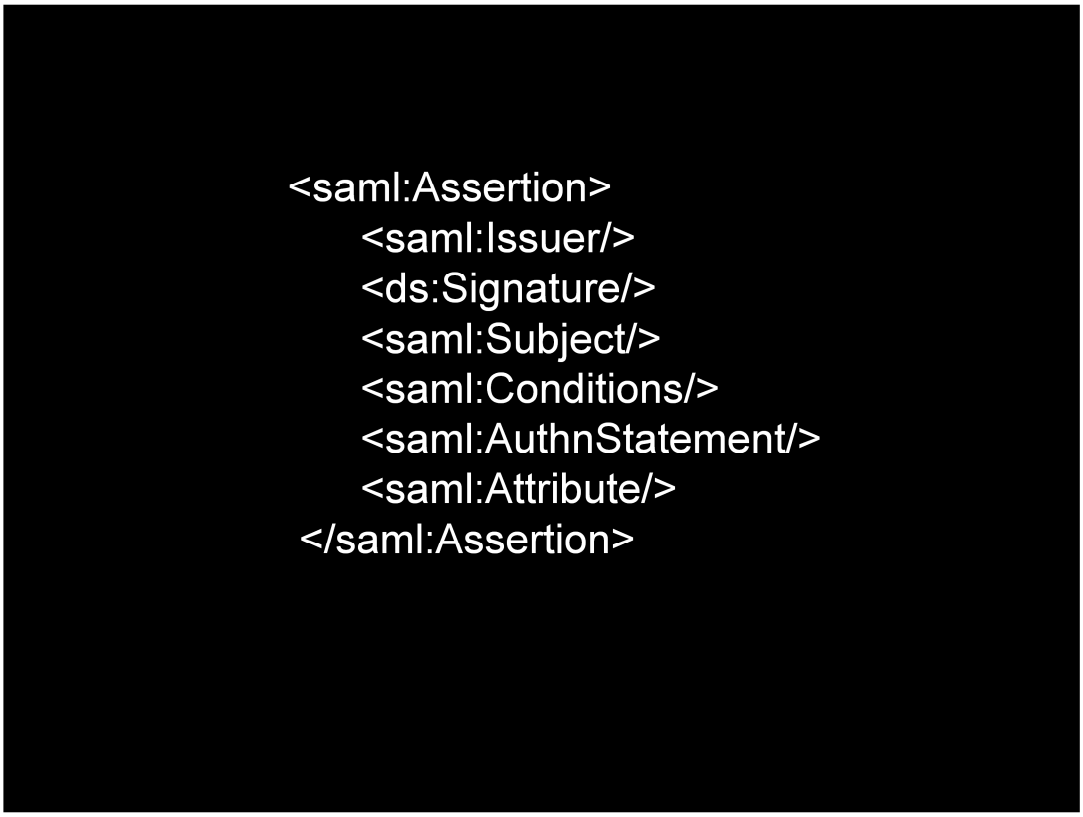
Metadata er brukes for å beskrive og dele konfigurasjonsdetaljer mellom saml entiteter som feks mellom identity provider og service provider. Det kan være alt fra hva slags identitets attributter den støtter til nøklene den bruker til signering og kryptering. Vi skal se nærmere på metadata I demonstrasjonen.

Autentiseringskonteksten kan for eksempel brukes av SP til å kreve at brukeren autentiseres vha en gitt autentiseringsmekanisme. Feks kreve at det brukes to-faktor autentisering. Jeg har ikke tenkt å gå mer inn på dette temaet.




Assertions brukes for å bekrefte informasjon om en bruker. Typisk vil en IdP legge til informasjon der den går god for at brukeren har blitt autentisert, har visse attributter eller er autorisert til å gjennomføre en gitt operasjon.

Disse er gjerne signert og/eller kryptert av utsteder slik at mottaker kan forsikre seg om at informasjonen den inneholder er til å stole på.




Her har vi en oversikt på hva en slik assertion kan inneholde.

**Issuer** angir hvem som har utstedt/laget denne xml snutten. Typisk identiteten til Identity Provideren.

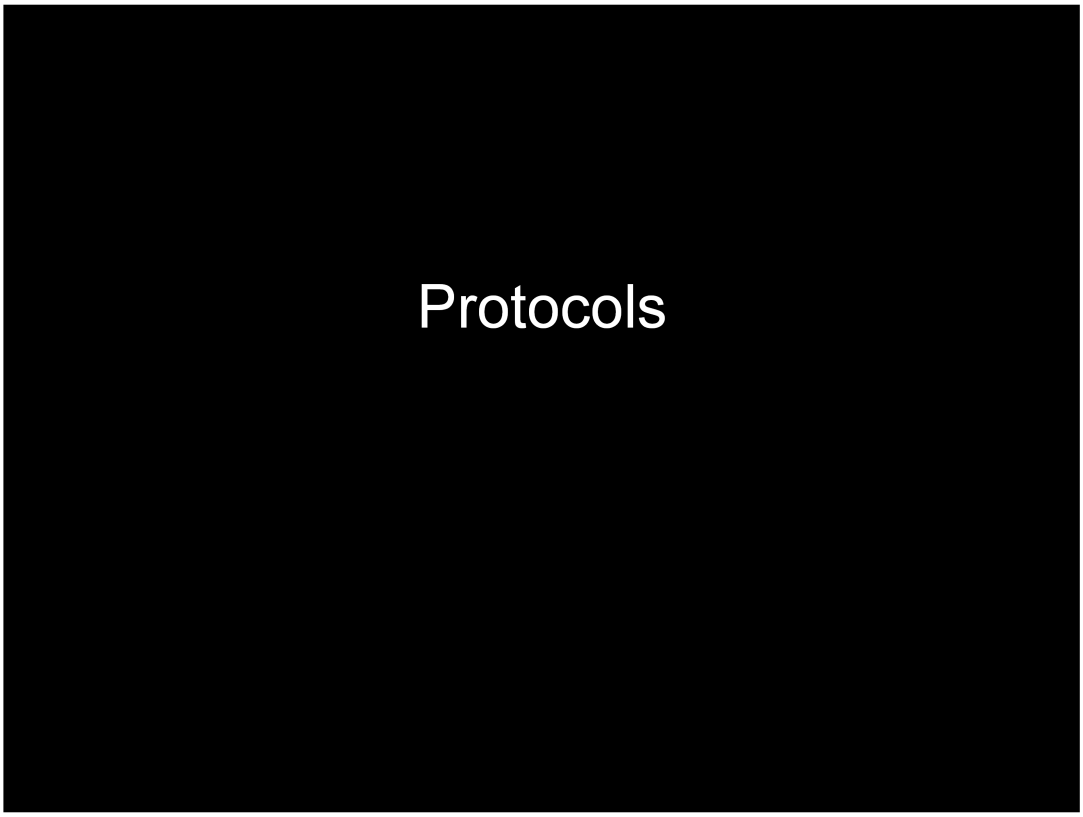
**Signature**: xml signatur av dataene

**Subject**: identifiserer brukeren. Kan være email, brukernavn eller lignende. Kan også være et pseudonym dersom identiteten vil holdes hemmelig

**Conditions**: angir når denne assertionen er gyldig. Feks før/etter en gitt klokkeslett, eller mellom klokkeslett

**AuthnStatement**: angir hvilken autentiseringsmekanisme som ble brukt for å autentisere brukeren

**Attribute**: ekstra informasjon om en bruker. Feks rolleinformasjon.




SAML definerer en rekke request/response meldinger som brukes I kommunikasjonen mellom  identitetsprovidereren og serviceprovideren.

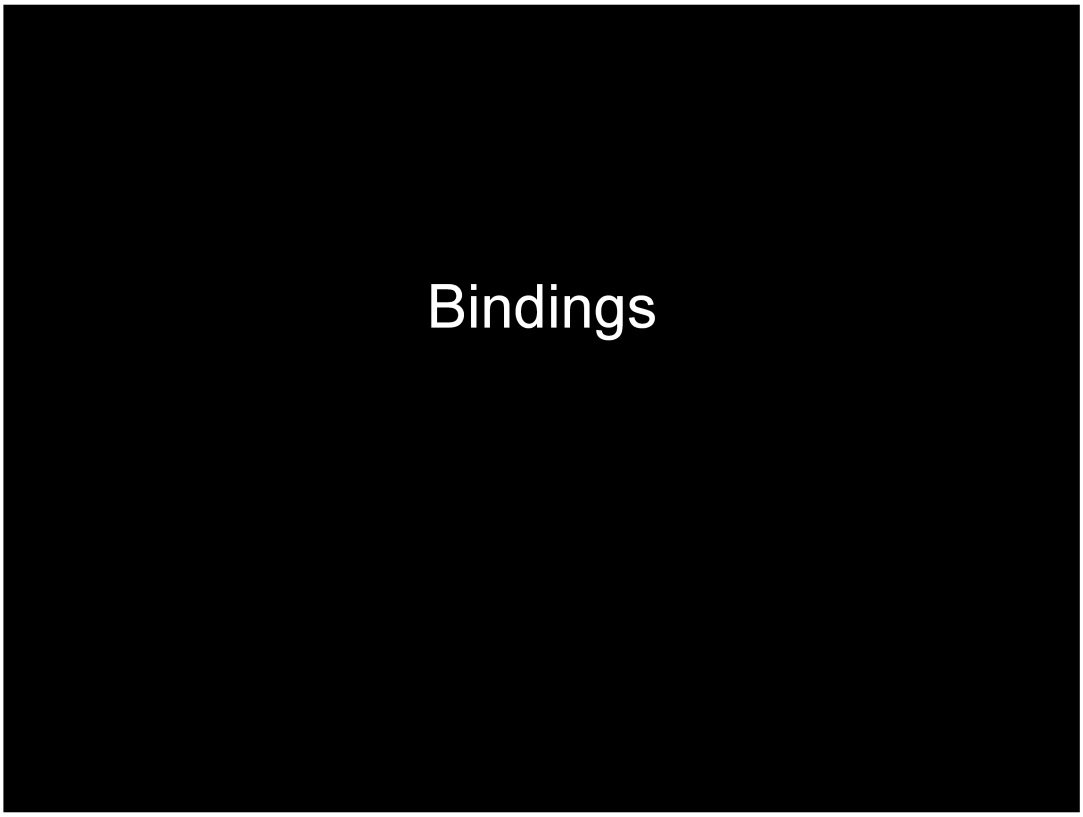
Disse kan brukes til alt fra å be om autentiseringsinformasjon om en bruker til å endre identiteten til en bruker, i.e. endre assosiasjoner mellom identitet og bruker.

Authentication Request Protocol, Single Logout Protocol, Assertion Query and Request Protocol, Artifact Resolution Protocol, Name Identifier Management Protocol


Her har vi et eksempel på en Autentiseringsrequest.

Issuer vil da typsik være identiteten til service provideren som ber om autentiseringsinformasjon om en bruker. NameIDPolicy angir hvilken identifikator service provider ønsker, det være seg email, brukernavn eller feks x.509 sertifikat.

AuthnRequest vil typisk også inneholde en del attributter som feks når requesten ble generert, hvor og hvordan SP vil at responsen skal returneres/leveres osv.




En “binding” definerer hvordan saml protokollmeldinger kan transporteres over underliggende protokoller. Det kan feks være hvordan du kan du kan bruke HTTP Redirect til å transportere meldingene.

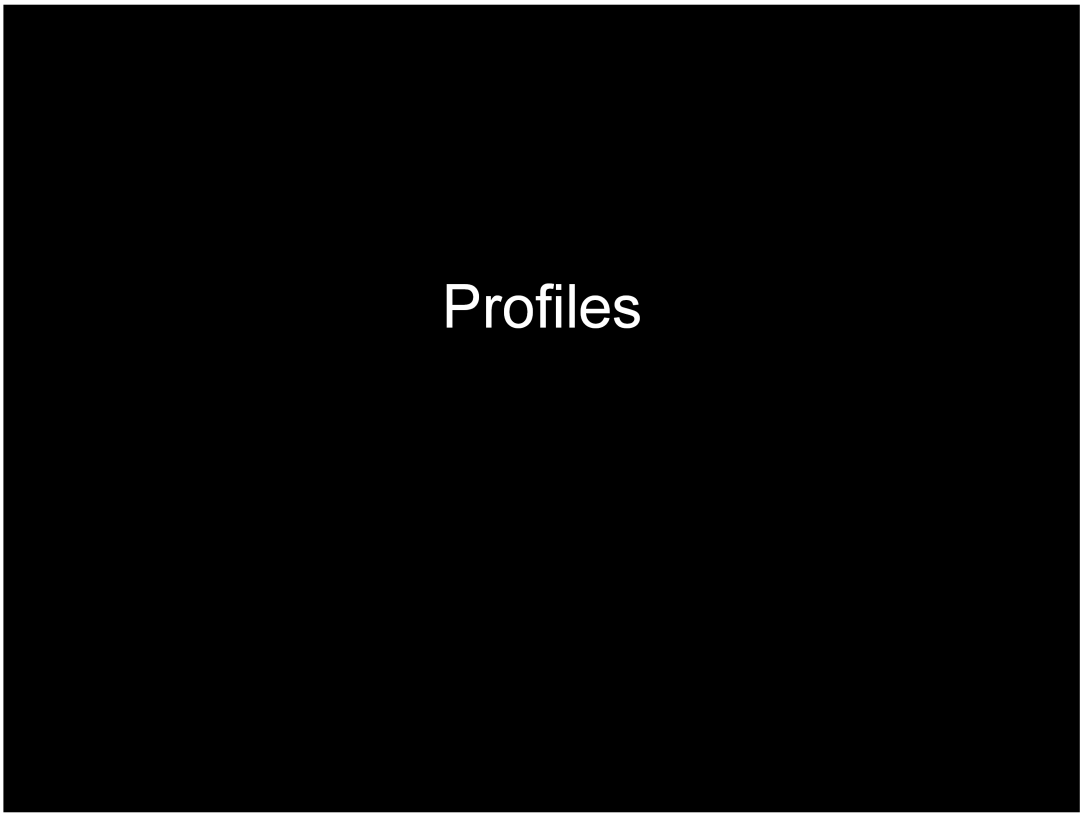
Vanlige måter å sende saml meldinger er vha http redirect, http post og soap.

```
<form method="post“
        action="http://sp.taa.com/SAML2/POST">
    <input type="hidden" name="SAMLResponse"
                value="response" />
        ...
    <input type="submit" value="Submit" />
 </form>


window.onload = function () { document.forms[0].submit(); }
```

For web browser sso brukes ofte HTTP Post bindingen.

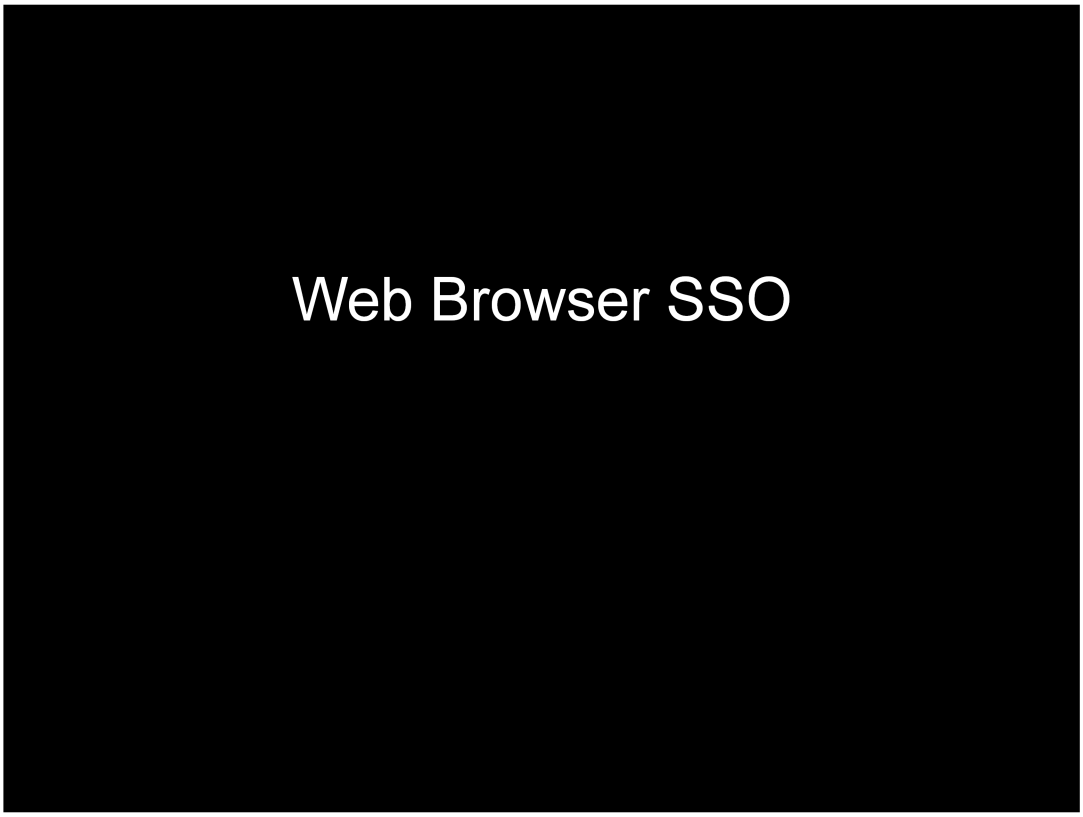
IdP inkluderer da saml responsen I et html form som igjen sendes tilbake til browseren I http responsen. HTML siden inneholder også gjerne et script som automagisk genererer en POST for å sende selve saml responsen til service provideren.




En SAML profil definerer hvordan de ulike komponentene kan kombineres for å løse en eller annen us

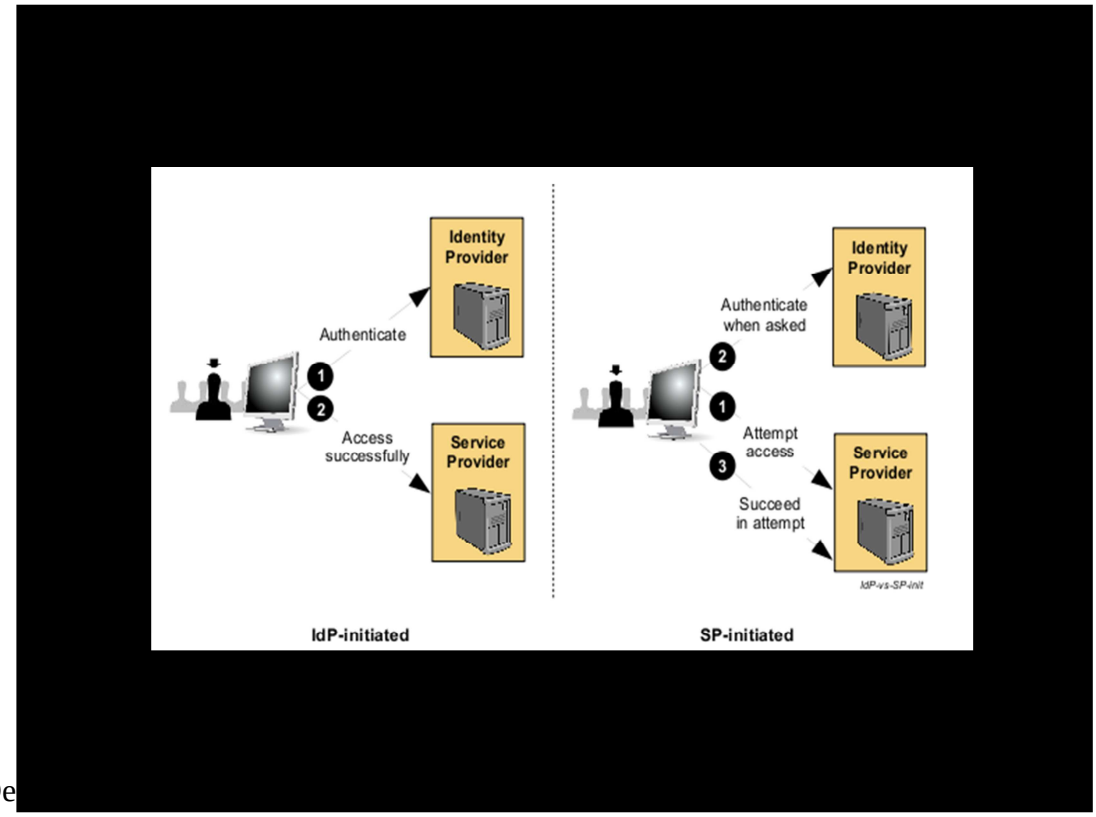

Web Browser SSO and Single Logout er to eksempler på hvordan de ulike komponentene har blitt sam

Jeg har ikke tenkt å gå nærmere inn på Single Logout. Det finnes også en del andre profiler som feks n




Denne profilen beskriver da hvordan man kan bruke SAML meldinger og bindinger for å løse web sing

Den tilbyr en rekke forskjellige muligheter for å løse denne problemstillingen.
I hovedsak dreier det seg mest om hvor SSO prosessen startes; om det er på Identity Provideren eller på
Det finnes også en rekke valg på hvordan du vil at meldingene skal transporteres mellom IdP og SP.




De

1. IdP initiated og
2. SP initiated

Den mest vanlige er SP-initiated hvor det hele starter med at brukeren går inn på en eller annen tjeneste

Denne er den som er mest vanlig.

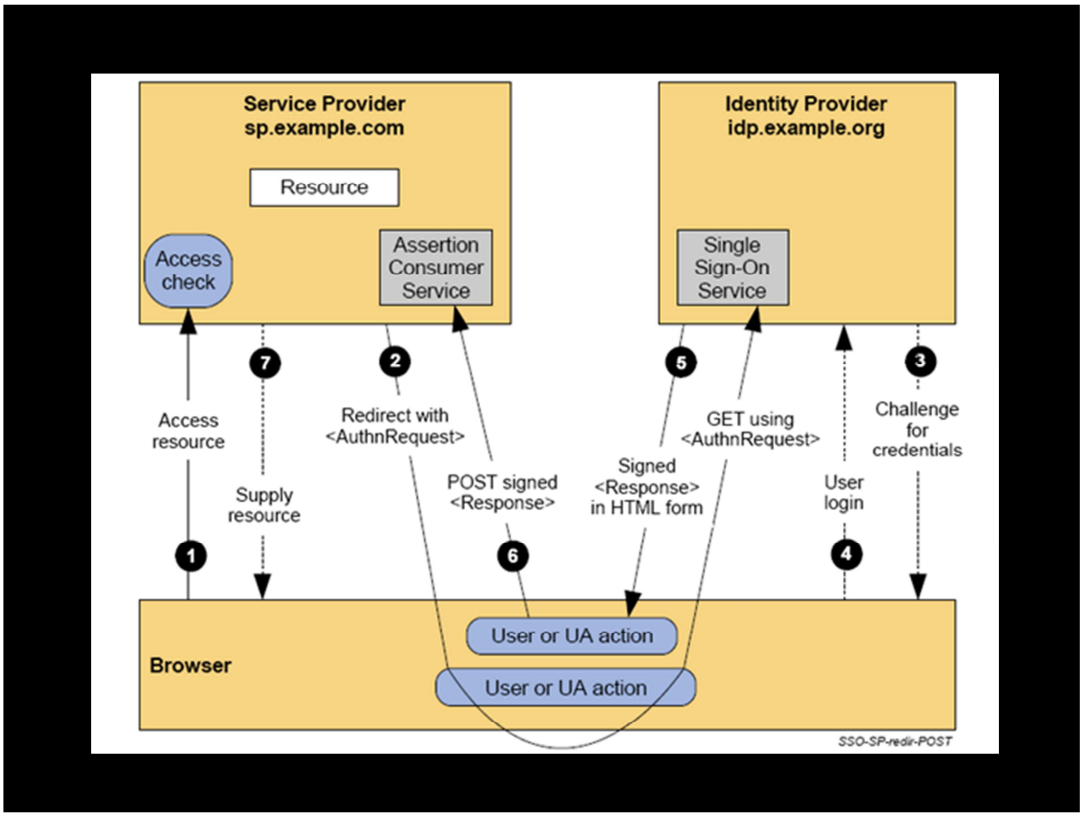
I det andre tilfelle er feks brukeren allerede inn på sidene til en IdP hvor han/hun klikker på en link til e

I dette tilfelle kreves det at IdP-en er konfigurert med linker til de ulike SP-ene den samarbeider med.

For SP tilfellet finnes det igjen to valgmuligheter for hvordan meldingene sendes til SP.
Alt 1:
Redirect fra SP til IdP med AuthnRequesten
POST for responsen fra IdP til SP.

Det er denne som brukes i demoen etterpå.

Alt 2:
POST for å sende AuthnRequest til IdP
Artifact Binding for å sende responsen til SP (SP får da en token som kan benyttes til å spørre etter en a




Her kan vi se meldingsflyten for en SP inititiert SSO prosess hvor Redirect brukes for å sende AuthnRe

1. Brukeren går inn på sidene til SP
2. SP krever autentisering.
3. SP lager så en autentiseringsrequest AuthnRequest som den sender til browseren vha av en HTTP r
4. Browseren kjører så en GET på det den fant I Location headeren.
5. Dersom brukeren er innlogget fra før av vil IdP bare generere en Assertion direkte, ellers vil den sp
6. Etter suksessfull innlogging vil så IdP sende tilbake responsen I et html form for igjen sørger for at
7. Er assertion godkjent (signaturen er gyldig osv) sendees browseren til den opprinnelige siden.

Dette er så også si det samme som vi vil se i demoen.


I tillegg til SSO vil SAML også kunne tilby andre tjenester som feks:

Federation:
I dette tilfelle må vi da selvfølgelig ha federerte identiteter. Må være en link mellom identiteten hos ent
Kan feks gjøres out-of-band vha synkroniserig av brukerdatabaser
Eller vha:

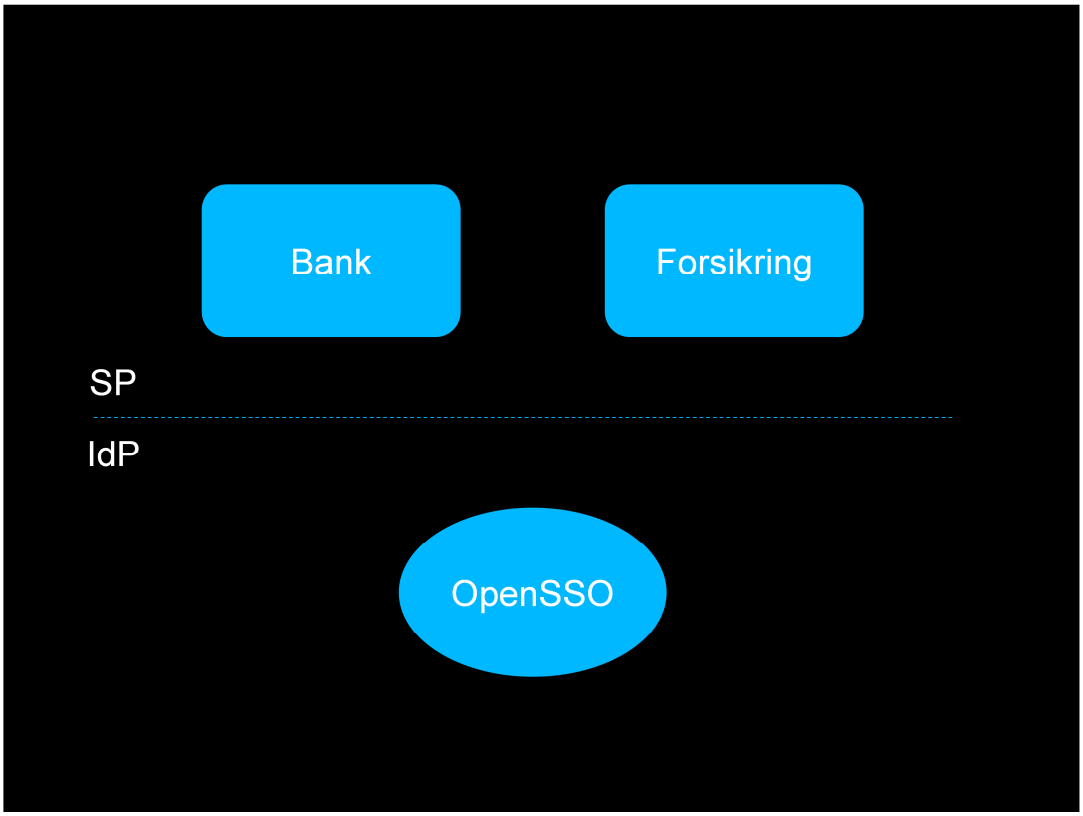
Pseudonymer; transiente (gyldige for en web sso session only) eller persistente

En bruker vil da identifiserer som en “token”. Det vil dynamisk bli laget en mapping mellom kontoene

Persistente: krever at du må ha en brukerdatabase hos SP
Transisente: da slipper du å vedlikeholde en database hos SP. Dette blir da en fullstendig anonym tjene

Web Service Security
• SAML assertions har også blitt tatt I bruk i andre frameworks som feks Web Service Security. Denne

DEMO




To service providere: Bank og Forsikring

1 IdP: OpenSSO

SP-ene er to enkle servlets.

Bank har ett beskyttet område: “Nettbank”. Fra nettbanken kan du klikke deg inn på “Mine Sider” hos forsikringsselskapet.

Forsikring har også ett: “Mine Sider”.

Spring Security benyttes for å sikre sidene.

IdP er en default konfigurasjon av OpenSSO som kjører på tomcat. Det har blitt oppretter konfig for å muliggjøre SAML2.0.

Rekkefølge på demo:

1. Vise raskt oppbyggingen av Bank og Forsikring
2. Vise bank med form authentication vha Spring Sec. (localhost:8090)
3. Logger inn med brukernavn og passord: jak/jak
4. Går så inn på Forsikring -> Mine Sider for å vise form autentisering her også. (localhost:9090)
5. Clear history i FF
6. Gå på nytt inn I banken -> Nettbank
7. Klikk deg videre til “Mine Sider”
8. Vis at du må skrive inn passord på nytt.

9. Nå enabler vi SAML på banken. Forklare hva de ulike bønnene I appContext gjør for noe


DONE

## Spring Security SAML Extension

SAMLEntryPoint
- bruker webSSOProfile til å generere og sende AuthnRequest til IdP

WebSSOProfile
- lager og sender AuthnRequest

SAMLProcessingFilter
- tar imot Assertion, ber authenticationManager om å verifisere den

SAMLAuthenticationProvider
- ber webSSOProfileConsumer om å validere Assertion, returnerer gyldig token

WebSSOProfileConsumer
- validerer innkommende Assertions

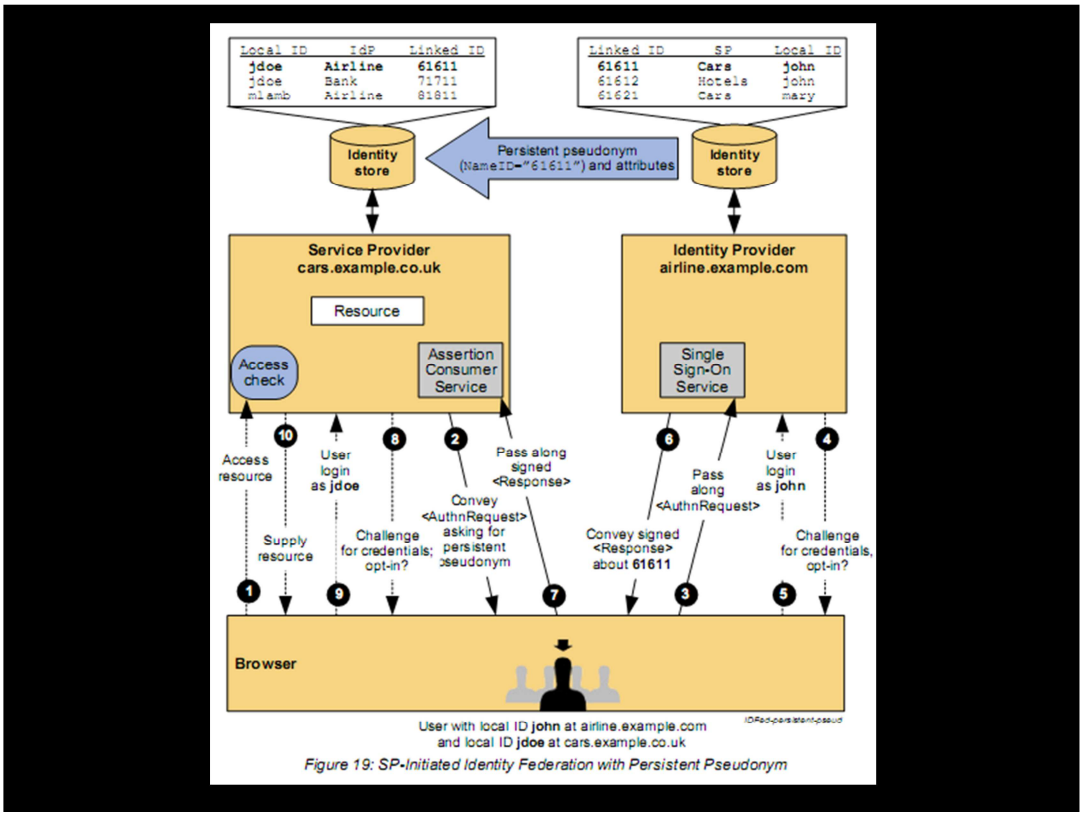



*Figure 19: SP-Initiated Identity Federation with Persistent Pseudonym*

Federation: Persistent Pseudonym

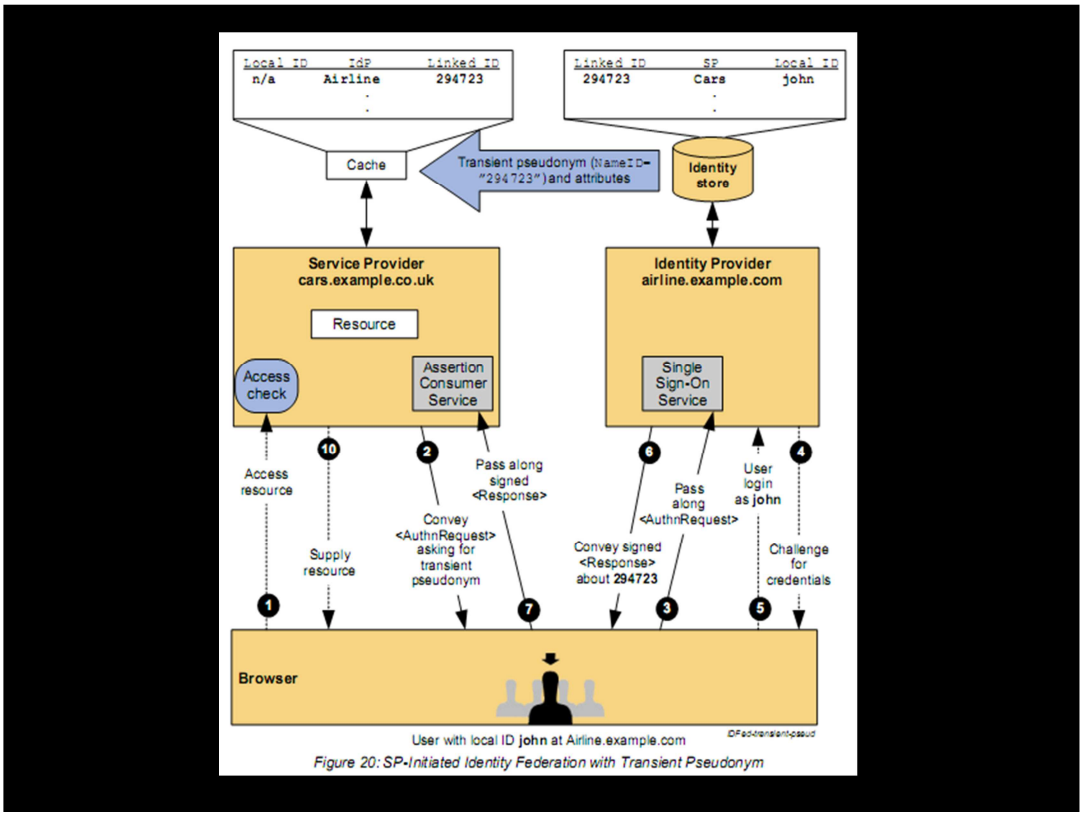



*Figure 20: SP-Initiated Identity Federation with Transient Pseudonym*

Federation: Transient pseudonym

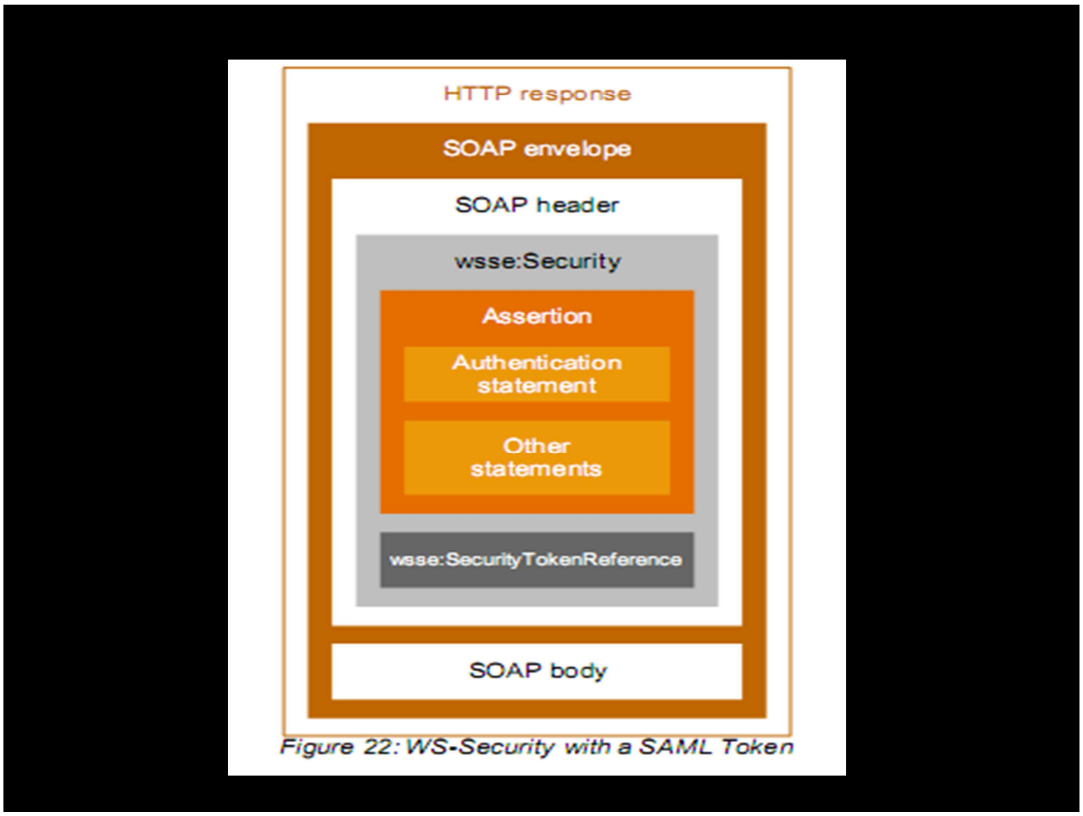


*Figure 22: WS-Security with a SAML Token*

WS-Security