REGIONE BASILICATA

IMS Documento di integrazione v. 1.5

ALLEGATO C08

Regione Basilicata

BAS2009

Identity Management

Documento di integrazione

BAS2009-INTEGRAZIONE-IDMS-1.4

Responsabilità

	Nome	Firma
Redatto da:	Maurizio Argoneto	
Rivisto da:		
Approvato da:		

Registrazione modifiche al documento

Edizione	Data	Motivo
1.0	01/12/2009	
1.4	20/02/2010	Integrazione di un SP Shibboleth 2.0 con IDP

Indice

Indice delle figure

1 – Scopo

Il documento si pone come obiettivo quello di illustrare la soluzione adottata dalla Pubblica Amministrazione Regionale relativamente alla gestione delle identità digitali dei cittadini ed in particolare fornire una descrizione delle metodologie e degli strumenti d'integrazione delle applicazioni Web sviluppate in Java con l' IMS in analisi.

2 – Campo di applicazione

Il presente documento è una linea guida all'integrazione di tutti i software WebBased con il sistema di IM.

3 – Riferimenti

3.1 Documenti di Riferimento

Modello di gestione federata delle identità digitali (GFID)

3.2 Bibliografia

Modello di gestione federata delle identità digitali (GFID)

4 – Definizioni e acronimi

AA	Attribute Authority
DS – Discovery Service	Un servizio Shibboleth 2.0 che permette agli utenti di individuare gli Identity Provider che si desidera utilizzare per l'autenticazione
IdM/IM	Identity Management
IP/IdP	Identity Provider. A kind of service provider that creates, maintains, and manages identity information for principals and provides principal authentication to other service providers within a federation, such as with web browser profiles.
Enterprise Service Bus (ESB)	Infrastruttura software che fornisce servizi di supporto a architetture SOA complesse. Abilita e semplifica la sincronizzazione dei dati ai vari sistemi.
SASL	Simple Authentication and Security Layer
SP	Service Provider

SSL	Secure Socket Layer
TLS	Transport Layer Security
Trust	Trust is a quality of a relationship between two or more entities, in which an entity assumes that another entity in the relationship will behave in a fashion agreed beforehand, and in which the first entity is willing to act on this assumption.
WAYF - Where Are You From	Ul servizio Shibboleth 1.x che permette all'utente di scegliere/scoprire l'IdP da utilizare per l'autenticazione.

5 – Tassonomia dell'Identity Management

I sistemi (o i modelli) di IM possono essere classificati in base alla specifica 'filosofia' di gestione del *trust il nostro* modello di IM è costituito essenzialmente da un unico IdP che si occupa di identificare gli utenti per conto di una molteplicità di SP. Le informazioni costituenti l'identità digitale di un utente possono anche in questo caso essere distribuite tra i provider, ma l'identificatore a essa associato è unico e gestito dall'IdP. Come il precedente, anche questo modello permette il SSO.

5.1 Caratteristiche

L' Identity Management della Regione Basilicata mira a fornire:

- un sistema centralizzato di gestione delle identità;
- un sistema scalabile, sia in termini di numero di utenze che di numero delle applicazioni che possono agganciarsi a esso per la gestione delle utenze
- un sistema pienamente compatibile con gli standard più diffusi, SAML v.2.0, XACLM 2.0, SOAP 2.0, e sviluppato completamente in Java Enterprise Edition (Java EE o Java J2EE). La maggior parte delle specifiche, per quanto riguarda le caratteristiche dell' IMS in ambito federato, derivano da quelle emesse in seno al progetto ICAR
- un sistema che garantisca la massima sicurezza, mediante il ricorso a protocolli di comunicazione sicuri (https) e opportune politiche, ampiamente configurabili, di accesso e visibilità dei dati.

6 – Considerazioni di propedeuticità

Il sistema IMS certifica l'utente e lo abilita ad interfacciarsi ai servizi esposti dall'ente, per questo motivo si rende necessario garantire al massimo la sicurezza della fase di autenticazione che potrà avvenire tramite smart card con certificato d'identità valido (controllo con CA riconosciuta) e/o sistema di verifica di codici per l'accesso con username e password.

7 – Modello di funzionamento generale

L'immagine che segue presenta una visione complessiva del modello dell' IMS, con la descrizione delle interazioni delle principali componenti del sistema.

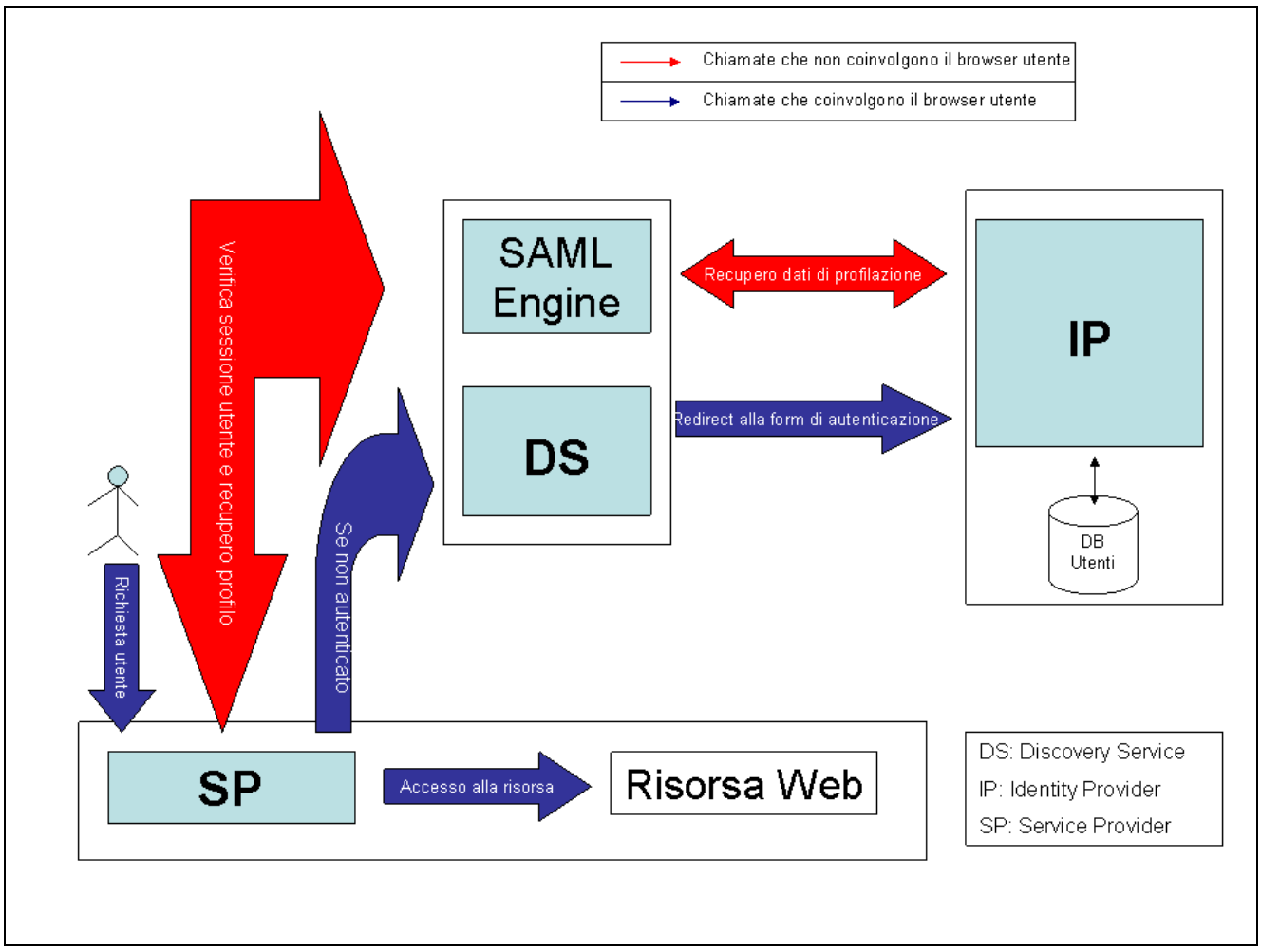


Figura 1 - Modello SSO

7.1 Service Provider 1

- Ogni richiesta utente a una risorsa viene intercettata dal Service Provider (SP);
- Il SP chiede all'Engine SAML informazioni sul profilo utente;
- L'Engine SAML verifica che la richiesta provenga da un SP valido;

7.2 Discovery Service

- Se l'utente non è già stato autenticato si chiede all'Engine SAML l'indirizzo del server che eroga il Discovery Service (DS);

- Il browser mostra all'utente l'elenco dei possibili Identity Provider (IP);
- L'utente viene rediretto alla pagina di login Single Sign On (SSO).

7.3 Identity Provider

- L'utente inserisce le proprie credenziali attraverso il browser (oppure si autentica tramite CIE, CNS o altro);
- Le credenziali vengono verificate da IP;
- Se l'autenticazione ha esito positivo l'Engine SAML attiva la sessione utente.

7.3.1 Attribute Authority

- I dati utente vengono estratti dal repository degli utenti attraverso il modulo Attibute Authority;
- Il risultato viene trasmesso all'Engine SAML.

7.4 Service Provider 2

- L'Engine SAML decodifica i dati del profilo utente e li invia al SP;
- L'Engine SAML notifica al SP che l'utente può accedere alla risorsa richiesta;
- Il browser dell'utente viene reindirizzato alla pagina web richiesta.

8 – SAML Web Single Sign On

In questa implementazione dell' IMS si è scelto di abbracciare lo standard SAML 2.0 per la gestione del Web Single Sign-On (SSO). Tre sono i *profili* (per usare la terminologia SAML) che sono stati attualmente implementati i seguenti:

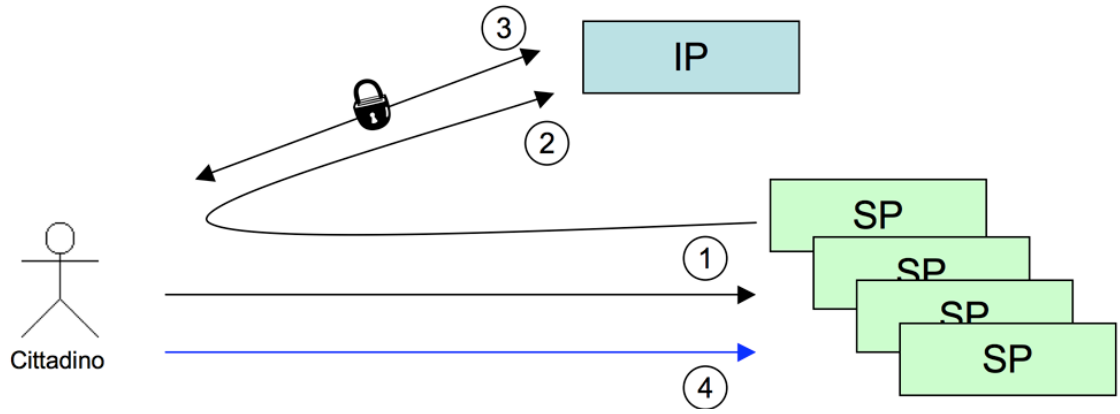
- Web Browser SSO Profile;
- Single Logout Profile;
- Assertion Query/Request Profile: che permette di interrogare l'IdMS per ottenere informazioni (attributi) su un utente.

E due sono i profili di Binding:

- Http Redirect Binding;
- Http Post Binding.

Nel linguaggio SAML si identificano l'*Identity Provider*, che certifica l'identità di un utente, e i *Service Providers*, ovvero i siti web a cui un utente vuole accedere per ottenere un servizio. Gli scenari di accesso sono di due tipi: *SP-initiated* e *IP-initiated*. Il primo caso, il più frequente, si ha quando un utente accede direttamente al SP per richiedere un servizio. Nel secondo caso invece l'utente

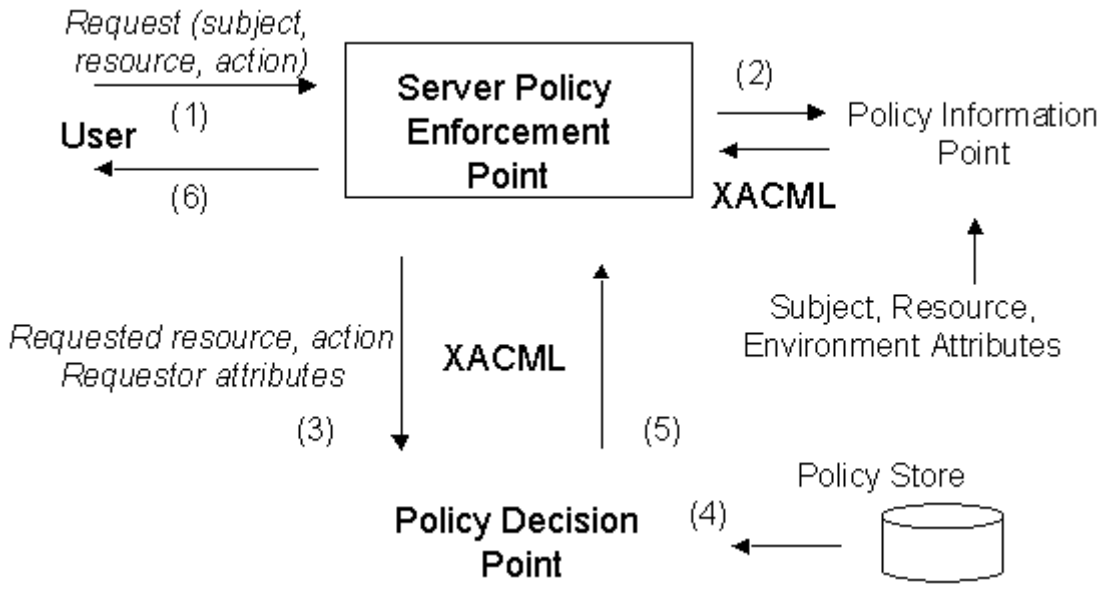
accede prima all'IP, si autentica, e da qui accede ad uno dei vari SP disponibili. Descriviamo nel dettaglio il primo caso.



- Al passo 1 l'utente accede al SP. Il SP si accorge (secondo una sua logica personale, indipendente da SAML) che l'utente non si è autenticato.
- (passo 2) Genera allora una ridirezione HTTP (HTTP Status 302 o 303) al servizio di login dell'IP secondo le specifiche SAML. In particolare sarà specificata la richiesta di autenticazione (AuthnRequest) e verrà indicata la URL di ritorno (attributo RelayState).
- (passo 3) L'utente si autentica all'IP secondo varie logiche (ad esempio invio login/password su sessione HTTPS). L'IP, riconosciuto l'utente, produce (specifiche SAML) una pagina che contiene un form HTTP con associata un'azione POST verso il SP.
- (passo 4) L'utente accede al SP, che verifica la Response SAML ricevuta via POST e fa accedere l'utente.

9 – Accreditamento e autorizzazione all'accesso ai servizi

XACML eXtensible Access Control Markup Language è un linguaggio di Policy, utilizzato per descrivere i requisiti generali del controllo degli accessi a risorse distribuite (xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"). Un linguaggio per gestire gli accessi a risorse, che permette di sapere quando una data azione su di una risorsa può essere compiuta o meno e di interpretarne un eventuale risultato. Ecco un esempio di funzionamento del Policy Engine di Ibasho:



9.1 PEP

E' quell'entità di sistema che effettua il controllo sugli accessi, facendo richieste di decisione e facendo rispettare le decisioni di autorizzazione. Livello logico che protegge la risorsa richiesta (posta su file system distribuito o web server che sia)

9.2 PIP

E' l'entità di sistema che ha la funzione di archivio dei valori dei vari attributi di risorsa, azione o ambiente. Esso fornisce i valori degli attributi al context handler.

9.3 PDP

E' l'entità di sistema che valuta le policy applicabili e produce la decisione di autorizzazione per l'esecuzione dell'azione sulla risorsa richiesta. Quando un utente cerca di accedere ad una risorsa, il PEP ne definisce gli attributi ed assegna al PDP il compito di decidere se autorizzare o meno la richiesta. La decisione è presa in base alla descrizione degli attributi dell'utente.

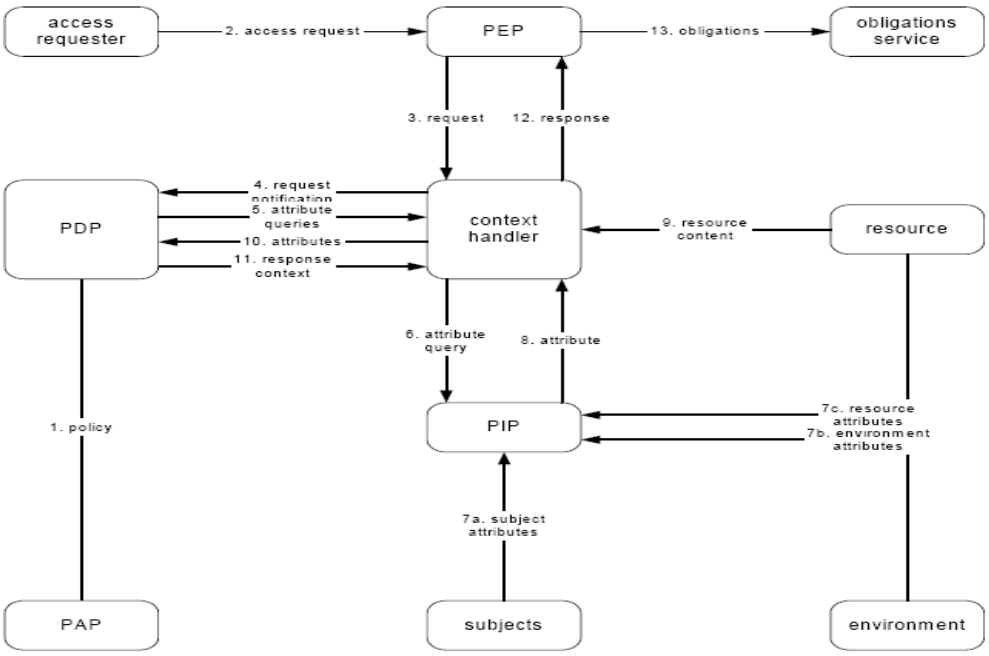
9.4 Context Handler

E' l'entità di sistema che converte la richiesta dal suo formato nativo al formato canonico XACML e viceversa e che permette la comunicazione tra tutte le altre componenti del sistema.

9.5 Data Flow Model

Situazione di base: qualcuno vuole effettuare un'azione su di una risorsa. Questo il flusso delle operazioni:

- Il PAP scrive policy singole o set di policy e le rende disponibili al PDP. Questi set di oggetti rappresentano le politiche per uno specifico target;
- Chi richiede l'accesso alla risorsa, effettua una richiesta al PEP;
- Il PEP manda la richiesta al Context handler aggiungendo attributi per la risorsa, l'azione e il sistema;
- Il context handler, prima richiede gli attributi dal PIP, poi costruisce una richiesta XACML e la manda al PDP;
- Il PDP valuta le politiche allegate alla richiesta;
- Infine ritorna la risposta (con inclusa la decisione di autorizzazione) al context handler;
- Il context handler traduce la risposta e la ritorna al PEP;
- Il PEP fa rispettare gli obblighi dati dalla decizione di autorizzazione;
- Se l'accesso è permesso, quindi, il PEP autorizza il richiedente ad accedere alla risorsa, altrimenti gli nega l'accesso;



10 – Ibasho: framework IMS della Regione Basilicata

La nostra architettura è un'implementazione innovativa di un sistema IMS basato su Shibboleth 2.0 che permette di avere un maggior livello di granularità nella gestione di un sistema AAI:

- È interamente sviluppato in Java;
- È interamente Web Oriented;
- Rispetta gli standard SOAP, SAML 2.0, XACML 2.0;
- Ha un gestione degli attributi che rispetta lo standard eduPerson;

10.1 Componenti di Ibasho

L'infrastruttura di autorizzazione e autenticazione di Ibasho prevede i seguenti moduli

- Ibasho Identity Provider:
  - Attribute Authority;
  - Single Sign On;
- Ibasho Filter Location: WAYF
- Ibasho Service Provider:
  - SAML Engine;
  - Guard: Resource Filter;

10.2 Ibasho Profile

Ibasho implementa principalmente il Web Browser SSO Profile (Profile for the OASIS Security Assertion Markup Language v2.0) ed in particolare è stato implementato sulla base dei seguenti scenari (use case) previsti da Security assertion markup Language v2.0 Technical Overview:

- SP-Initiated SSO: Redirect/POST Bindings
- SP-Initiated Single LogOut with Multiple SPs

10.3 Ibasho MyPage

Ibasho MyPage è un'applicazione protetta che gestisce l'accesso ai servizi da parte dell'utente che ha effettuato l'accesso al sistema tramite IMS. Attraverso

questa applicazione l'utente potrà trovare una lista di tutte le applicazioni web gestite tramite IMS alle quali può accedere direttamente in modalità di SSO, secondo le specifiche SAML 2.0 prima descritte.

11 – Integrazione dei sistemi con Ibasho

L'integrazione dei sistemi web con il sistema di autenticazione e autorizzazione, adottato dalla Regione Basilicata, ha un impatto minimo con la struttura dell'applicazione stessa. Ci sono alcune caratteristiche di base che devono essere rispettate affinchè l'applicazione possa integrarsi all'IMS:

- Le applicazioni devono essere sviluppate con tecnologia Java come definito dalle specifiche tecniche e standard dell'ufficio SIRS;
- Devono poter essere raggiungibili tramite nome di dominio (es. http://nomeapplicazione.dominio);

Per l'integrazione devono essere eseguiti i seguenti step:

- Registrazione del servizio tramite console di amministrazione di Ibasho:
  - Registrazione del SP(Service Provider);
  - Definizione di un file che descriva le politiche di autorizzazione all'accesso della risorsa attraverso linguaggio descrittivo XACML 2.0.
- Inclusione del Filtro di Ibasho nell'applicazione web Java:
  - Inclusione di un JAR nelle librerie dell'applicazione;
  - Modifica del web.xml con le specifiche del filtro;
  - Sviluppo di una Classe Java che implementi l'interfaccia del LogIn di Ibasho al fine di autenticare l'utente nella base dati locale al servizio.

11.1 Registrazione del servizio

11.1.1 Registrazione Service Provider

Dopo aver configurato correttamente la proprio applicazione protetta è possibile proseguire con la registrazione di quest'ultima sull'Idp e sull'Engine. Questa operazione può essere effettuata solamente dall'utente amministratore del sistema di identità. (Rif. maurizio.argoneto@supporto.regione.basilicata.it)

11.1.2 Registrazione file di Policy

La policy deve essere generata come nell'esempio sotto descritto. Per trovare i codici aggiornati degli uffici e delle aree dei dipendenti è necessario interrogare il servizio web di cui forniamo le specifiche.

DocumentazioneTecnica Web service Sigru (nello zip allegato)

Una volta creato la policy sarà possibile verificarne la corretta implementazione utilizzando un eseguibile di seguito fornito che simula un PDP base per l'incrocio tra la policy di request e policy del servizio.

PolicyEngine.zip

Esempio di Policy di Request generata dall' IMS: ***request.xml***

Esempio di Policy generata da un'applicazione secondo lo standard XACML 2.0: ***Policy.xml***

Per quanto concerne la descrizione degli uffici, delle aree e dei dipartimenti è necessario fare riferimento al servizio web descritto nel seguente documento il quale potrà fornire i codici corretti ed aggiornati di tutto l'ente. Di seguito riportiamo l'elenco degli URN degli attributi da utilizzare nella Policy per quanto riguarda la categoria del Dipendente:

urn:oasis:names:tc:xacml:2.0:ibasho:attribute:codarea
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:codufficio
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:codservizio
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:codpresidio
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:data_assunzione
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:data_cessazione
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:codlivello
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:codqualifica
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:codtitolostudio
urn:oasis:names:tc:xacml:2.0:ibasho:attribute:codposizione_organizzativa

11.2 Accreditamento ai servizi e controllo delle Policy

Le strategie di accreditamento ai servizi prevedono due scenari differenti e possono essere riassunti in questo modo: accreditamento diretto e accreditamento indiretto.

11.2.1 Accreditamento diretto

Nel caso dell'accreditamento diretto una volta che l'utente accede e che ha superato il controllo del PDP, basato sulla validità della Policy di richiesta e sulla base della policy definita dal servizio, ha a tutti i diritti per accedere a tale risorsa e come tale deve poter essere anche registrato in modo trasparente. Per esempio se un cittadino ha intenzione di accedere alla Comunity del sito di

Basilicatanet e vuole partecipare alle attivtà e ai servizi esposti su tale sito, dato che tale applicazione prevede una policy di accesso a tutti i cittadini, non dovrà fare richiesta esplicita di registrazione ma la sua richiesta di autorizzazione è implicita nel controllo delle policy stesse. In tale circostanza l'applicativo che viene raggiunto dal cittadino, e che quindi ha superato il controllo di sicurezza, hai due scenari complementari:

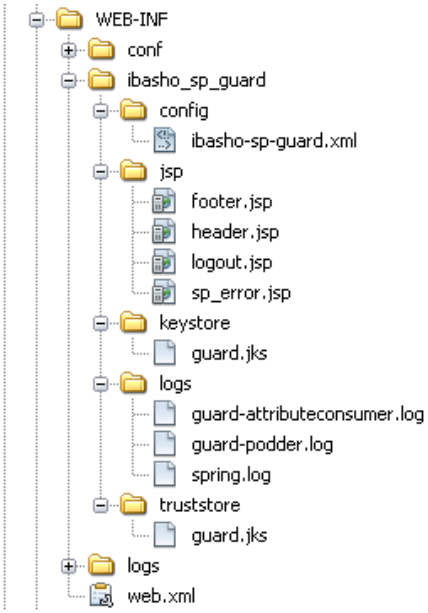
- Il cittadino non esiste quindi lo creo sul db dell'applicativo, lo autentico;
- Il cittadino già esiste e lo autentico al sistema;

11.2.2 Accreditamento indiretto

Nel caso dell'accreditamento indiretto una volta che l'utente accede e si presenta come nel caso di sopra, e nelle stesse condizioni di permesso dell'accesso,l'applicativo può decidere di trattare l'accesso alla sua applicazione nel modo più "custom" possibile. Potrebbe infatti decidere di creare automaticamente un utente come "pending" nella propria applicazione, o chiedere all'utente di integrare alcuni dati di specifiche e mirata utilità. Questo perché ci possono essere dei casi in cui è necessario attribuire dei ruoli, localmente all'applicazione, al cittadino e/o al dipendente che fa richiesta di una certa applicazione e questo presupporrebbe una "mediazione" nella validazione di un utente che non è possibile definire in modo automatico e attraverso processi deterministici.

11.3 Inclusione del Filtro di Ibasho nell'applicazione web

Struttura dell'integrazione delle applicazioni web:



ibasho-sp-guard/config

Contiene il file (ibasho-sp-guard.xml) di configurazione per l'applicazione protetta nel quale vengono indicate le informazioni sul Guard (ID, HostName e gestione Cookie), sull'Engine (AuthConsumerURL, WAYFLocationService, LogoutServiceURL) e sul certificato da utilizzare per connessioni HTTPS. (Prestare molta attenzione alla configurazione di questo file)

ibasho-sp-guard/jsp

Contiene le pagine jsp utilizzate dall'IbashoFilter per eventuali errori e informazioni sulla procedura di Logout. (Modificabili per adattarle allo style della propria applicazione)

ibasho-sp-guard/keystore e ibasho-sp-guard/truststore

Contengono il truststore e il keystore utilizzati per le comunicazioni su HTTPS.

ibasho-sp-guard/logs

Contiene eventuali file di log indicati nella configurazione di Log4j.

Importare nel proprio progetto le librerie necessarie:

IbashoFilter.jar, Beans.jar, Common.jar, bcprov-jdk14-136.jar, commons-logging-1.1.1.jar, jsr173_1.0_api-2.3.0.jar, log4j-1.2.14.jar, spring-2.5.0.jar, spring-webmvc-2.5.0.jar, xalan-2.7.0.jar, xbean-2.3.0.jar, xercesImpl-2.9.0.jar, xml-apis-1.0.b2.jar, xml-apis-2.9.0.jar, xmlsec-1.3.0.jar.

Configurazione web.xml

Altro file fondamentale per il corretto funzionamento del sistema IMS è il **web.xml**, nel quale vanno inserite le seguenti direttive:

```
<filter>
    <filter-name>Ibasho Resource Guard</filter-name>
    <filter-class>it.regione.basilicata.ibasho.sp.guard.Guard</filter-class>
    <init-param>
        <param-name>configFile</param-name>
        <param-value>/WEB-INF/ibasho_sp_guard/config/ibasho-sp-guard.xml</param-value>
    </init-param>
    <init-param>
        <param-name>loginModuleClass</param-name>
        <!— QUESTA E' LA CLASSE CHE DEVE IMPLEMENTARE it.regione.basilicata.ibasho.sp.guard.IbashoInitializeImpl --!>
    </init-param>
        <param-value>miopackage.miaclasseLoginIbasho</param-value>
    <init-param>
        <param-name>homePage</param-name>
        <param-value>home.jsp</param-value>
    </init-param>
</filter>
<filter-mapping>
     <filter-name>Ibasho Resource Guard</filter-name>
     <url-pattern>/ibasho/protected/protected.jsp</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>Ibasho Resource Guard</filter-name>
    <url-pattern>*.guanxiGuardlogout</url-pattern>
</filter-mapping>
<!--
    Ibasho Guard Session Verifier Service
-->
<servlet>
```

```
	<servlet-name>SessionVerifier</servlet-name>
	<servlet-
class>it.regione.basilicata.ibasho.sp.guard.SessionVerifier</servlet-class>
	<load-on-startup>2</load-on-startup>
</servlet>
<!--
	Ibasho Guard Attribute Consumer Service
-->
<servlet>
	<servlet-name>IbashoGuardAttributeConsumerService</servlet-
name>
	<servlet-class>
it.regione.basilicata.ibasho.sp.guard.AttributeConsumer</servlet-class>
	<load-on-startup>3</load-on-startup>
</servlet>
<!--
	Guard Podder
-->
<servlet>
	<servlet-name>Podder</servlet-name>
	<servlet-class> it.regione.basilicata.ibasho.sp.guard.Podder</servlet-
class>
	 <load-on-startup>4</load-on-startup>
</servlet>
<!--
	Guard Logout - servlet invocata durante la procedura di Logout
-->
<servlet>
	<servlet-name>Logout</servlet-name>
	<servlet-class> it.regione.basilicata.ibasho.sp.guard.Logout</servlet-
class>
	<load-on-startup>5</load-on-startup>
</servlet>
<!--
	Pagina jsp protetta - utilizzata come default
-->
```

```
<servlet>
	<servlet-name>org.apache.jsp.ibasho.protected_.protected_jsp</servlet-name>
	<servlet-class>org.apache.jsp.ibasho.protected_.protected_jsp</servlet-class>
</servlet>
<!--
	Servlet's Mapping
-->
<servlet-mapping>
	<servlet-name>org.apache.jsp.ibasho.protected_.protected_jsp</servlet-name>
	<url-pattern>/ibasho/protected/protected.jsp</url-pattern>
</servlet-mapping>
 <!--
	Servlet per il controllo dell'identità' di Ibasho. (Controllo della sessione, degli attributi e della richiesta di Logout)
-->
<servlet-mapping>
	<servlet-name>SessionVerifier</servlet-name>
	<url-pattern>*.sessionVerifier</url-pattern>
</servlet-mapping>
<servlet-mapping>
	<servlet-name>IbashoGuardAttributeConsumerService</servlet-name>
	<url-pattern>*.guanxiGuardACS</url-pattern>
</servlet-mapping>
<servlet-mapping>
	<servlet-name>Podder</servlet-name>
	<url-pattern>*.guanxiGuardPodder</url-pattern>
</servlet-mapping>
<servlet-mapping>
	<servlet-name>Logout</servlet-name>
	<url-pattern>*.GuardLogout</url-pattern>
</servlet-mapping>
```

Di seguito il file ZIP del filtro da utilizzare per l'integrazione delle applicazioni

IbashoFilter1.1.zip

11.3.1 Lettura dei dati che provengono dall'autenticazione

11.3.2 Importazione dei certificati SSL

Importazione dei certificati SSL sui server di Deploy delle applicazioni integrate nell'IMS

- keytool -import -alias ibasho -file public.crt -keystore %JAVA_HOME%/jre/lib/security/cacerts [password "changeit"]

Il file del certificato pubblico VeriSign da importare è il seguente:

public.crt

11.4 Integrazione di applicazioni non-Java

Concludiamo questo documento con la descrizione di un componente che si è reso necessario realizzare nello sviluppo del sistema di Single Sign On.

Come già accennato esistono delle casistiche particolari in cui il componente Guard sviluppato in Java per il progetto Ibasho non si può inserire all'interno della web application che offre i servizi che devono essere integrati nel sistema SSO. Queste problematiche si possono riassumere con i seguenti casi:

- Incompatibilità delle librerie con l'ambiente di distribuzione: si sono verificati diversi casi in cui il server contiene una versione dell'ambiente Java troppo datato o le librerie adottate da Ibasho entrano in conflitto con quelle del server.

- Incompatibilità dell'ambiente di sviluppo: in questo caso non è questione di librerie Java ma di tecnologie e linguaggi di programmazione diversificati come ad esempio applicazioni scritte in .NET oppure in php.

- Impossibilità di effettuare le chiamate interne tra i server: in questo filone ricadono le situazioni che vedono i server posti su reti diverse tra le quali si interpongo dei firewall o altre strutture che impediscono le comunicazioni tra le chiamate interne delle componenti Guanxi.

- Impossibilità di modifica alle applicazioni preesistenti: questo avviene solitamente quando si chiede di modificare le web application sviluppate da altre aziende che non intendono modificare le librerie all'interno dei loro prodotti. In questo caso si cerca di offrire un componente software più leggero per l'integrazione con il sistema di single Sign On che non preveda l'utilizzo di librerie aggiuntive al di fuori di quelle J2EE standard.

11.4.1 Integrazione con sicurezza DEBOLE (WRAPPER)

L'idea di fondo della soluzione adottata è la creazione di una applicazione dedicata al Tunneling delle richieste di autenticazione. Questo significa che l'applicazione che definiremo client demanderà il processo di autenticazione utente ad una diversa applicazione che definiremo tunneling attraverso una apposita richiesta http. Questa applicazione tunneling ritornerà quindi l'elenco dei dati di profilazione utente alla applicazione client.

Questa soluzione è orientata ad un'integrazione delle applicazioni tramite una sorta di wrapper che effettuerà una redirect, dopo l'autenticazione con l'IMS, all'applicazione da proteggere attraverso l'invocazione di una FORM POST in HTTPS. In questo scenario è fondamentale definire delle politiche di sicurezza aggiuntive a quelle offerte dal framework di SingleSignOn, come un filtro sugli indirizzi IP "certificati/attendibili" dai quali ricevere connessioni etc.

La segretezza del canale di comunicazione che si instaura tra l'applicazione web del servizio e l'applicazione di tunneling viene garantita dall'utilizzo del Secure Sockets Layer attraverso chiamate con protocollo https.

11.4.2 Integrazione con sicurezza FORTE (SP di Shibboleth 2.0)

Questo è lo scenario più comune e tendenzialmente quello che nel medio lungo periodo sarà quello più utilizzato. È infatti possibile integrare con il sistema di autenticazione un qualunque Service Provider sviluppato con Shibboleth 2.0.

Questa soluzione permette di integrare qualsiasi Web server che gira su qualsiasi piattaforma e/o sistema operativo e permette quindi di integrare anche applicazioni sviluppate con tecnologie molto differenti (PHP, .NET etc).

Il nostro Idp è in grado quindi di rispondere a tutte le chiamate e le interrogazioni fatte tramite asserzioni SAML 2.0 su protocollo HTTPS sia in configurazione HTTP Redirect e http Post Binding. Le istruzioni ed il software per l'installazione di un service provider così descritto sono reperibili al seguente indirizzo: https://spaces.internet2.edu/display/SHIB2/Installation

Di seguito vengono forniti i metadati per la configurazione dei vostri ServiceProvider già configurati per il funzionamento con l'IMS.

Unica personalizzazione consiste nella definizione e configurazione degli attributi generati dall'IDP che desiderate siano visibili (in Session) nella vostra web application e il setting della variabile EntityID che sarà quella che vi verrà fornita in seguito alla registrazione del SP presso l' Idp della Regione Basilicata (Ved. Paragrafo).

Per la configurazione del vostro SP si rimanda alla documentazione ufficiale di Shibboleth 2.0 https://spaces.internet2.edu/display/SHIB2/Home

Shibboleth-sp-config.xml