Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Scienze di Internet

Identity Provider Shibboleth per il servizio di federazione e Single Sign-On di Ateneo

Tesi di Laurea in
Architettura degli Elaboratori

Relatore:
Chiar.mo Prof.
VITTORIO GHINI

Presentata da:
GIACOMO FARNETI

Sessione III

Anno Accademico
2012/2013

PAROLE CHIAVE

Single Sign-On, Shibboleth, Identity Provider, Identity Federation, SAML

INDICE

INDICE DELLE FIGURE

GLOSSARIO

AAI (Authentication and Authorization Infrastructure): insieme delle component hardware e software che consentono l'implementazione dei meccanismi di autenticazione e autorizzazione degli utenti.

ADFS (Active Directory Federation Services): è l'implementazione Microsoft del server **IDP**, fornisce servizi di **Single Sign-On** ospitati in ambiente Windows Server 2008/2012. Si interfaccia ad Active Directory e supporta i protocolli **WS-Federation** e **SAML**.

Asserzioni: vedi **claim**.

Autenticazione: processo di identificazione di un utente registrato in precedenza.

Autenticazione Federata: il processo di accesso a risorse esterne o appartenenti alla propria home organization effettuato utilizzando la propria identità digitale.

Autorizzazione: processo di controllo dell'accesso a determinate risorse per un utente autenticato.

Claim: è un'affermazione che l'**IDP** rilascia ad un **SP** relativamente ad un utente. Il claim può essere una proprietà dell'utente, un'autorizzazione, o un'informazione generica. Alcuni esempi di claim sono il nome, l'indirizzo email o i ruoli di un utente.

Cookie: informazioni di tracciamento che i siti web memorizzano nella cache del browser per scambiare informazioni con altri siti web o per implementare meccanismi di navigazione stateful (in questo caso si parla di cookie di sessione).

COT (Circle Of Trust): insieme di organizzazioni che si accordano sull'autenticazione e sui livelli di autorizzazione per l'accesso alle risorse condivise.

Discovery Service: vedi **WAIF**.

EntityId: identificativo univoco di ogni **Identity Provider** e **Service Provider**.

Federazione: un insieme di organizzazioni che stabiliscono una rete di *trust* e collaborano in base a linee guida comuni.

Home Organization: un'organizzazione (università, fondazione, ente regionale ecc.) che partecipa ad una federazione di identità.

Identità Digitale: un insieme di informazioni attribuibili ad un utente, rilasciate e gestite dalla **Home Organization** dello stesso.

IDP (Identity Provider): è il servizio che si occupa di verificare l'identità dell'utente e di restituire all'applicazione web le informazioni minime necessarie all'identificazione e all'autenticazione dello stesso.

Interfederazione: una federazione di federazioni (tipicamente una federazione internazionale che raccoglie molte federazioni nazionali).

Metadati: documento elettronico (spesso in formato XML) che contiene dettagli e informazioni tecniche su **Identity Providers** e **Service Providers**.

Proxy: in generale, è una qualsiasi componente hardware/software che si interpone tra le comunicazioni di un client e di un server inoltrando i messaggi a livello di un determinato protocollo. Nell'ambito di ADFS, è il server che ospita l'interfaccia web di autenticazione e fa da ponte tra il browser dell'utente e il server STS.

Relying Party: terminologia Microsoft per il **Service Provider**.

SAML (Security Assertion Markup Language): è uno dei protocolli più diffusi per lo scambio di informazioni relative ad autenticazione e autorizzazione tra Identity Provider e Service Provider. Utilizza un formato basato su XML per lo scambio dei claim.

Shibboleth: middleware open source creato e manutenuto dal consorzio non-profit Internet2 per facilitare il processo di verifica dell'identità tra diversi domini

di sicurezza. Bassato sul protocollo **SAML**, implementa sia le componenti **IDP** che **SP**.

SP (Service Provider): è il servizio al quale l'utente vuole accedere, nel contesto del Web SSO si tratta tipicamente di applicazioni web.

SSO (Single Sign-On): nei sistemi di autenticazione e autorizzazione, è la proprietà che garantisce ad un utente l'accesso a più servizi (**SP**) effettuando il login una sola volta. La procedura di login e di rilascio dei **claim** è tipicamente affidata ad una componente software (**IDP**) che verifica l'identità dell'utente tramite username/password, smart card, token di autenticazione o meccanismi simili.

STS (Security Token Services): è la componente dell'IDP che si occupa di generare, firmare e verificare i token di autenticazione degli utenti.

WAIF (Where Are You From): servizio che consente ad un utente di effettuare la selezione dell'**IDP** della propria **Home Organization** in fase di autenticazione.

Web SSO (Web Single Sign-On): è il processo di Single Sign-On nell'ambito delle applicazioni web. Prevede almeno due componenti: l'applicazione web alla quale l'utente vuole accedere (**SP**) e l'interfaccia web per la verifica dell'identità dell'utente (**IDP**). I claim sono tipicamente trasmessi tra le due parti utilizzando **cookie** o chiamate **HTTP** (GET/POST).

WIF (Windows Identity Foundation): Framework Microsoft per l'implementazione di Service Provider basati sul linguaggio Microsoft .NET Framework.

WS-Federation: Protocollo sviluppato dal consorzio OASIS per lo scambio di **claim** relativi a autenticazione e autorizzazione degli utenti basato su XML.

INTRODUZIONE

Negli ultimi anni l'attenzione verso i sistemi di autenticazione e autorizzazione via web ha visto un forte incremento di attori e di tecnologie disponibili, diventando un aspetto sempre più importante sia per le aziende private che per le istituzioni pubbliche.

L'università di Bologna, da sempre attenta alle nuove tecnologie e all'innovazione, si è dotata nel 2010 di uno strumento per la gestione del Web Single Sign-On e dell'autenticazione federata: Microsoft Active Directory Federation Services (ADFS) 2.0.

Tale strumento è un Identity Provider (IDP), ovvero un servizio per la verifica dell'identità degli utenti dell'organizzazione tramite username e password oppure Smart Card in grado di sollevare le applicazioni web (anche esterne all'organizzazione) dall'onere di verificare direttamente le credenziali dell'utente delegando totalmente la responsabilità sul controllo dell'identità digitale all'IDP.

La soluzione Microsoft si è dimostrata generalmente semplice da configurare e da gestire come del resto l'implementazione del Web Single Sign-On.

Discorso a sé stante deve essere fatto invece per quanto riguarda l'integrazione con le principali federazioni di identità regionali e italiane (FedERa e IDEM) rivelatasi problematica a causa di una incompatibilità con il protocollo SAML 1.1, ancora utilizzato da alcuni dei servizi federati.

Per risolvere tale incompatibilità, in considerazione anche del fatto che sarebbe stato impossibile obbligare tutti i servizi al momento esposti da soggetti terzi a migrare alla versione aggiornata del protocollo SAML, il CeSIA – Area Sistemi Informativi e Applicazioni dell'Università di Bologna ha deciso di dotarsi di un Identity Provider Shibboleth, alternativa *open source* ad ADFS che presenta funzionalità equivalenti ed è in grado di gestire tutte le versioni del protocollo SAML (attualmente rilasciato fino alla versione 2.0).

Contributo personale

Il mio compito è stato quello di analizzare, installare, configurare e integrare con le federazioni IDEM e FedERa un'infrastruttura basata sull'IDP Shibboleth prima in test poi in produzione, con la collaborazione dei colleghi che in precedenza si erano occupati della gestione della soluzione Microsoft ADFS.

Il lavoro che ho svolto è stato suddiviso in quattro fasi:

- Analisi della situazione esistente
- Progettazione della soluzione
- Installazione e configurazione di un Identity Provider in ambiente di test
- Deploy dell'Identity Provider in ambiente di produzione

La struttura della tesi rispecchia le fasi appena elencate: nei primi capitoli affronto la storia dei sistemi di autenticazione e la loro evoluzione, che li ha portati dall'essere un semplice strumento per la verifica dell'identità di un utente all'interno di un dominio a mezzo efficace per la collaborazione e l'accesso a risorse condivise.

In seguito espongo l'analisi effettuata sull'infrastruttura di gestione dell'identità digitale adottata dall'Università di Bologna e dei problemi ad essa collegati.

Il capitolo 4 illustra il lavoro pratico che ho svolto in prima persona: la progettazione dell'architettura del servizio IDP, le prove che ho effettuato prima su una macchina virtuale poi su un server di test nella server farm del Ce.S.I.A., l'installazione e la configurazione del servizio IDP in ambiente di produzione e l'integrazione di quest'ultimo con le principali federazioni di identità italiane (IDEM e FedERa).

L'ultimo capitolo illustra le nuove tecnologie e le novità degli ultimi anni nell'ambito della gestione dell'identità, evidenziando i possibili sviluppi futuri del progetto di Single Sign-On dell'Università di Bologna.

1 L'IDENTITÀ DIGITALE

L'identità digitale non è che la declinazione tecnologica, applicata ai sistemi informatici, del concetto tradizionale di identità.

Nei sistemi classici l'identificazione personale avviene per mezzo di documenti rilasciati dagli uffici preposti –comuni, prefetture ecc.– che contengono informazioni riconducibili in modo univoco alla persona che ne è il proprietario, ad esempio il nome, il codice fiscale, la fotografia o l'indirizzo di residenza. Alcuni esempi di questi documenti possono essere la carta di identità, la tessera sanitaria o la patente di guida.

Il concetto di identità digitale, pur mantenendo in comune alcune caratteristiche con quello appena citato, prevede alcuni elementi aggiuntivi tipici dei sistemi informatici:

- **Identificativo univoco:** è l'informazione che identifica univocamente il soggetto dell'identità in un determinato contesto, ad esempio l'indirizzo email o l'ID legato all'anagrafica.
- **Credenziali:** informazioni che il soggetto esibisce per dimostrare la propria identità, ad esempio la coppia username/password oppure il certificato di autenticazione X509 contenuto nella propria Smart Card personale.
- **Attributi base:** un insieme di informazioni relative all'utente che servono per la propria identificazione in diversi sistemi di autenticazione, ad esempio nome, cognome ed indirizzo email.
- **Attributi contestuali:** informazioni aggiuntive relative all'utente che aiutano a descriverne l'identità ma che sono di interesse solo in un determinato contesto, ad esempio il ruolo che ricopre in un determinato settore o le impostazioni prescelte per determinate applicazioni.

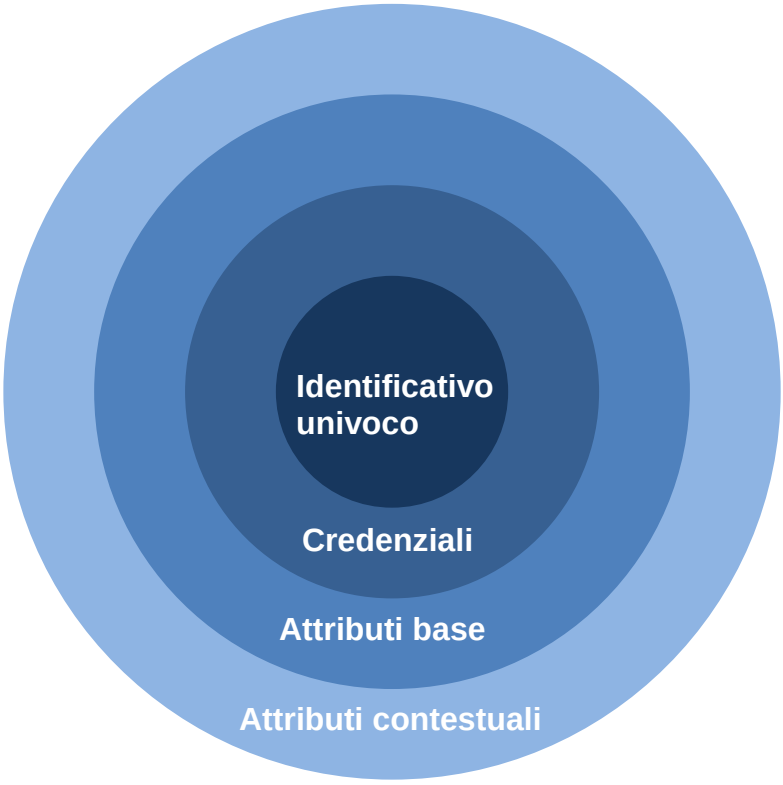


Figura 1: Componenti dell'identità digitale

L'infrastruttura –informatica e non– per la gestione dell'identità digitale viene definita Identity Management (IdM). Una delle definizioni più note di IdM è quella fornita da James Lewis, del Burton Group (Lewis, 2003):

L'Identity Management è l'insieme dei processi di business, e l'infrastruttura necessaria per la creazione, l'aggiornamento e l'utilizzo delle identità digitali.

Nel prossimo paragrafo verranno illustrati alcuni sistemi software per la gestione delle identità digitali.

1.1 I sistemi di gestione dell'identità

Gli strumenti informatici più diffusi per gestire gli account dei propri utenti (provisioning/deprovisioning, aggiornamento dei dati, autenticazione e autorizzazione) sono i Directory Service.

Un Directory Service è un servizio che persiste le informazioni relative a tutti gli utenti di un'organizzazione, e offre le seguenti funzionalità:

- Supporto per diversi tipi di oggetti: utenti, gruppi, unità organizzative ecc.
- Memorizzazione degli attributi degli utenti con diversi livelli di sicurezza e policy di accesso
- Gestione delle password e dei meccanismi per la loro modifica o reset
- *Schema* degli attributi e dei tipi di oggetti estensibile in base alle esigenze
- Blocco degli account e dell'accesso alle risorse
- Appartenenza ricorsiva a gruppi di sicurezza
- Integrazione con i principali sistemi operativi
- Registrazione dei log degli accessi
- aderenza agli standard di sicurezza e rispetto dei vincoli sulla privacy imposti per legge

Esistono numerose soluzioni informatiche che implementano servizi di directory: le più diffuse, solo per citarne alcune, sono Microsoft Active Directory (usata dall'Università di Bologna), OpenLDAP (suite di prodotti open source), Novell eDirectory (prima conosciuto come Novell Directory Services) e Oracle Internet Directory (OID).

I Directory Service semplificano la gestione del ciclo di vita delle identità digitali degli utenti, che si può riassumere nelle seguenti fasi:

- provisioning, ovvero la creazione dell'account, l'assegnazione di un identificativo univoco (chiamato in questo contesto *distinguished name*) e, opzionalmente, la creazione di una casella email
- role management, cioè l'assegnazione di ruoli all'utente tramite l'appartenenza a gruppi di sicurezza
- blacklisting, la disabilitazione dell'account per prevenire l'accesso a determinati servizi
- deprovisioning, ovvero l'eliminazione dell'account al terminare del rapporto tra l'utente e l'organizzazione

Spesso i Directory Service interagiscono con altri sistemi informativi quali database, HRMS (Human Resource Management System, ovvero "sistemi di gestione delle risorse umane") e Web Service.

1.2 I protocolli di autenticazione e autorizzazione

Per rendere possibile la verifica dell'identità digitale di un utente che accede ad un servizio i Directory Service espongono funzionalità di autenticazione e autorizzazione.

Con **autenticazione** si intende il processo di verifica dell'identità di un utente che è stato precedentemente registrato sul proprio Directory Service (fase di provisioning). L'autenticazione può avvenire tramite tre meccanismi (o una combinazione di essi):

- Something You Know (qualcosa che conosci), ad esempio la coppia username/password che l'utente ha memorizzato
- Something You Have (qualcosa che hai), come una Smart Card, un generatore OTP (One Time Password) o una chiave crittografica
- Something You Are (qualcosa che sei), ovvero i dati biometrici: impronta digitale, caratteristiche della retina, tono della voce ecc.

La scelta dei meccanismi più adatti per la propria organizzazione dipende dall'importanza dei dati ai quali si può accedere: per leggere le proprie email di solito è sufficiente fornire username e password, per entrare nel proprio conto bancario vengono spesso utilizzate combinazioni di username/password e chiavi OTP, mentre nel caso di accesso ad aree che richiedono alti livelli di sicurezza (ad esempio alcune zone del CERN) vengono usati i dati biometrici.

L'**autorizzazione** rappresenta invece l'insieme dei diritti di accesso alle risorse.

Nei Directory Service questi diritti vengono solitamente espressi tramite l'appartenenza a gruppi di sicurezza: quando questi gruppi identificano dei ruoli all'interno dell'organizzazione si parla di Role-Based Access Control (RBAC).

L'Attribute-Based Access Control (ABAC) è invece il processo di consentire l'accesso ad una risorsa in base alla presenza di determinati attributi dell'utente.

Gli accessi possono essere garantiti in base all'inclusione all'interno di liste di accesso (Whitelist) oppure negati tramite liste di esclusione (Blacklist).

Di seguito una lista dei più diffusi protocolli di autenticazione e autorizzazione.

1.2.1 *Kerberos*

Kerberos è un protocollo di autenticazione integrato nella maggior parte dei sistemi operativi (nei sistemi Microsoft è il protocollo di default da Windows 2000 in poi, è integrato in tutti i sistemi Unix-like compreso Mac OS X).

Sviluppato dall'MIT nella metà degli anni ottanta per la protezione delle comunicazioni di rete del progetto Athena, si è evoluto negli anni fino a raggiungere la versione 5, considerata come la versione "standard" di Kerberos grazie alle numerose funzionalità e ai miglioramenti alla sicurezza introdotti rispetto alle versioni precedenti.

Kerberos è un sistema di autenticazione distribuito che consente ad un processo (**client**), eseguito per conto di un determinato **utente**, di provare la sua identità ad un processo verificatore (**verifier**) grazie ad un **Authentication Server** in maniera sicura. Per farlo utilizza comunicazioni di rete che, se intercettate, non consentono ad un malintenzionato di impersonare l'utente (è quindi protetto contro i replay attack).

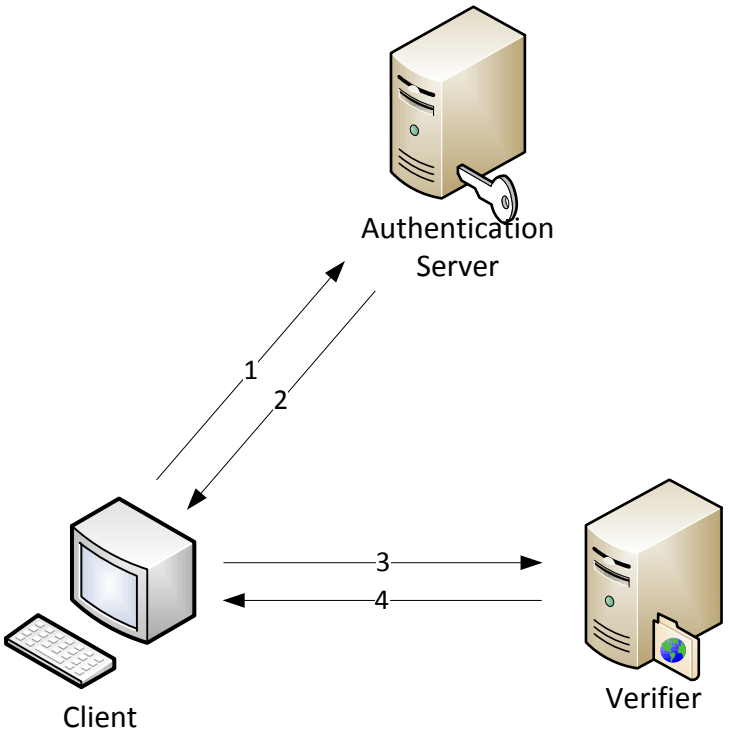


Figura 2 : Il protocollo di autenticazione Kerberos (schema semplificato)

Quando il client vuole accedere ad una risorsa e ha bisogno quindi di autenticarsi sul verifier, chiede all'Authentication Server (AS) un ticket Kerberos (*1-2*).Tale ticket è un certificato firmato dall'AS, che contiene la chiave di sessione random che client e verifier utilizzeranno per scambiarsi le informazioni (*3-4*). Essendo il ticket firmato dall'AS il client non può modificarlo, altrimenti il verifier si accorgerebbe dell'invalidità del ticket. Infine, per impedire attacchi di tipo replay (replay attack), il ticket contiene un timestamp che ne specifica il tempo di validità.

Nonostante Kerberos sia stato creato quasi 30 anni fa, resta uno dei più importanti e diffusi protocolli di autenticazione, grazie alla forte attenzione alla sicurezza e ad una implementazione open source che si è consolidata e migliorata nel tempo.

1.2.2 LDAP

Tutti i Directory Service moderni supportano il protocollo LDAP (**Lightweight Directory Access Protocol**), uno standard basato su TCP/IP per effettuare operazioni di ricerca e aggiornamento sui servizi di directory.

Un insieme di server (i Domain Controller) contengono i dati dei quali si compone il Directory Sevice, divisi in una struttura ad albero (i cui rami vengono denominati **domini**); tali dati vengono replicati costantemente tra i vari server per aumentarne l'affidabilità e le performance tramite meccanismi di ridondanza geografica.

Un client può collegarsi ad un Domain Controller (**bind**), ricercare oggetti tramite dei filtri (**search**), richiedere le proprietà di un oggetto (**lookup**) oppure modificarne gli attributi (**update**).

I Domain Controller cooperano come se fossero un'unica entità: se il server al quale è collegato un client non contiene le informazioni richieste (ad esempio si ricerca l'oggetto relativo ad uno studente su un Domain Controller appartenente al dominio del personale), tale server può richiedere le informazioni al Domain Controller appropriato per restituirle al client oppure indicare al client l'indirizzo del Domain Controller da contattare.

Il protocollo LDAP è stato pensato per garantire operazioni di ricerca veloci ed efficienti; per questo motivo si integra alla perfezione con i Directory Service che, a differenza dei database, contengono dati descrittivi e basati su attributi fortemente legati ad un determinato oggetto, ottimizzati per le operazioni di ricerca.

L'autenticazione e l'autorizzazione in LDAP sono gestite tramite l'operazione di **bind**, che consente modificare l'**authorization state** della connessione tra il client e il server.

Inizialmente, quando una connessione è stabilita, l'authorization state è impostato su anonymous; in questo stato la connessione non è sicura e di default la maggior parte delle implementazioni di Directory Service non consentono di eseguire ricerche sui dati.

Un'operazione di bind consente di autenticarsi tramite una coppia username/password inviata in chiaro nel messaggio (per questo motivo tutte le richieste LDAP dovrebbero essere inviate tramite una connessione sicura); il bind può andare a buon fine (impostando quindi l'authorization state al livello di sicurezza delle credenziali dell'utente) oppure fornire indicazioni sul problema riscontrato (ad esempio indicando all'utente che la sua password è scaduta).

Se le credenziali dell'utente che si sta autenticando non sono presenti sul server al quale è collegato, e quest'ultimo supporta la **pass-through authentication**, può inoltrare il processo di autenticazione al server più adatto; in questo modo è possibile aumentare la sicurezza dell'intero sistema memorizzando le credenziali degli utenti in server specifici protetti da regole del firewall più stringenti.

1.2.3 Radius

Il protocollo Remote Authentication Dial In User Service (**RADIUS**) è lo standard de facto per la gestione di autenticazione e autorizzazione nell'ambito delle connessioni di rete; la stragrande maggioranza dei dispositivi di rete attualmente in commercio supporta nativamente questo protocollo.

Tra i vantaggi principali di RADIUS troviamo:

- L’attenzione alla sicurezza: implementa meccanismi che ostacolano l’autenticazione di un malintenzionato in grado di ascoltare le comunicazioni di rete;
- La semplicità di implementazione, che consente di utilizzare RADIUS anche su sistemi embedded come router e switch;
- L’integrazione con il protocollo LDAP.

Il flusso di autenticazione client-server è piuttosto semplice:

1. il client invia un pacchetto di tipo Access-Request contenente la coppia username/password cifrata utilizzando la chiave condivisa tra il client e l’access point al momento del collegamento alla rete;
2. il server verifica tramite bind LDAP la validità di username e password;
3. se la password è valida il server invia un pacchetto di tipo Access-Accept, altrimenti invia un Access-Reject;
4. se il server necessita di informazioni ulteriori (ad esempio per autorizzare il client all’accesso ad una particolare risorsa) invia un pacchetto di tipo Access-Challenge; in questo caso il client mostra all’utente un prompt con la richiesta contenuta nel corpo del pacchetto, e invia la risposta inserita dall’utente al server, che può decidere di consentire o negare l’accesso alla risorsa in base alla risposta inserita.

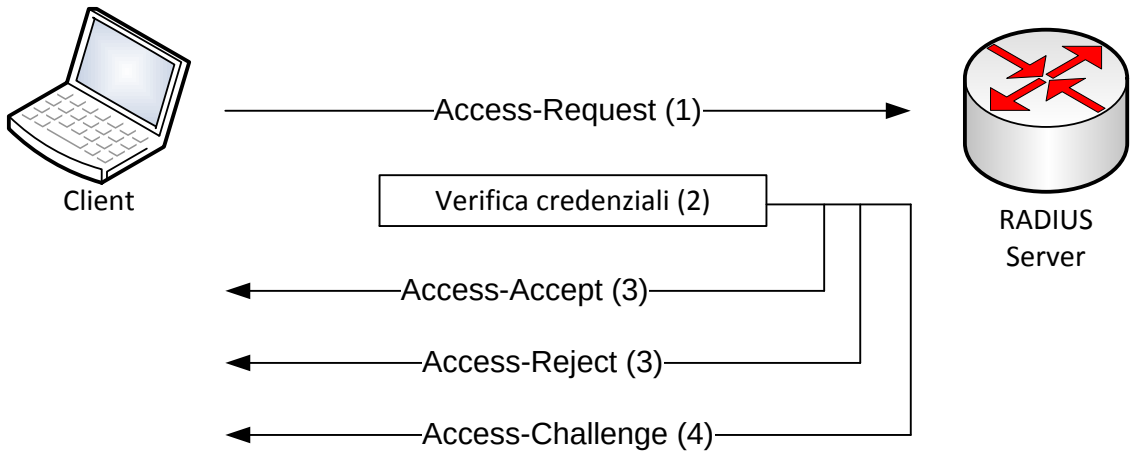


Figura 3: Lo schema di autenticazione di RADIUS

1.3 Gestione delle identità in applicazioni web

Nell'ambito delle applicazioni web le metodologie per autenticare e autorizzare gli utenti sono molteplici. Dalle soluzioni più naive come credenziali memorizzate in chiaro su database a quelle più complete e sicure come il Single Sign-On, i programmatori hanno sempre avuto un ampio spettro di scelta riguardo alle possibili tecnologie da utilizzare.

Quando si parla di credenziali scambiate tra un browser e un web server ci sono due requisiti minimi indispensabili per garantire un livello base di sicurezza ai propri utenti:

- L'utilizzo di un canale criptato (HTTPS) per tutte le comunicazioni tra il browser e il web server
- La verifica dell'identità del server tramite un certificato valido rilasciato da una **Certification Authority** (CA) fidata

La mancanza anche solo di uno di questi due elementi facilita tutta una serie di attacchi da parte di malintenzionati:

- **Eavesdropping (o Sniffing)**, il piu semplice degli attacchi: consiste nell'intercettare i messaggi di una comunicazione privata per utilizzarli a scopi malevoli;
- **Replay Attack**, ovvero la possibilità di intercettare un messaggio (del quale si può anche non conoscere il contenuto) e di inviarlo in un secondo momento al server per ripetere un'azione o un comando e ottenere lo stesso risultato del pacchetto originario. Molti web server sono stateless (non persistono cambiamenti di stato tra due richieste successive), caratteristica che li rende estremamente vulnerabili a questo tipo di attacchi;
- **Man In The Middle (MITM)**, che consente ad un malintenzionato di intercettare le chiavi di cifratura scambiate su un canale sicuro e di sostituirle con le proprie, ottenendo così la capacità di cifrare e decifrare tutti i messaggi scambiati tra client e server.

Quelli appena elencati sono solo alcuni tra tutti i possibili tipi di attacchi praticabili su una rete non sicura; sono stati citati solo i più diffusi in quanto un elenco esaustivo non è nello scopo di questa tesi.

1.3.1 Basic Authentication

Basic Authentication è sicuramente il metodo più semplice per autenticare un browser durante l'accesso ad una risorsa.

L'autenticazione avviene tramite l'utilizzo di un header HTTP che contiene al suo interno username e password dell'utente, richiesti di solito tramite una finestra di login integrata nel browser.

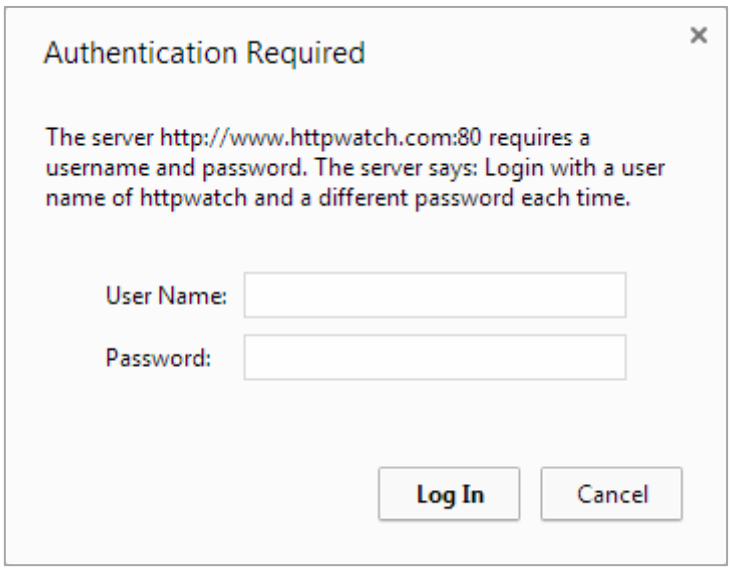


Figura 4: Maschera di login per la Basic Authentication

Il server può verificare le credenziali dell'utente invocando un web service, eseguendo un bind LDAP o una query al database.

Il prezzo da pagare per la facilità di implementazione e di utilizzo è un basso livello di sicurezza.

Basic Authentication è sensibile ai seguenti attacchi:

- Eavesdropping, se le credenziali sono inviate tramite HTTP e non HTTPS

- Replay Attack: anche nel caso di utilizzo di HTTPS, un pacchetto inviato due volte rimane valido perchè le credenziali cifrate all'interno continuano ad essere valide; l'unico modo per proteggersi da questo tipo di attacchi è implementare meccanismi di timestamp nel codice lato server
- Man In The Middle: essendo le credenziali inviate in chiaro, tramite un attacco di questo tipo è possibile avere accesso a username/password dell'utente per riutilizzarli in un secondo momento

Basic Authentication è quindi un protocollo con un basso livello di sicurezza e alta vulnerabilità, a meno di non implementare dei forti meccanismi di contrattacco nel codice lato server.

1.3.2 Utilizzo di Web Service

Un metodo semplice ed efficace di implementare meccanismi di autenticazione e autorizzazione in ambito web è tramite un web service ad hoc per la verifica delle credenziali, il caricamento delle proprietà di un utente e il controllo sull'appartenenza ai gruppi di sicurezza.

In questo scenario l'utente, tramite il suo browser, si collega ad una applicazione web che tramite una form di login richiede username e password:

- l'utente si collega all'applicazione web ed inserisce le sue credenziali nella form di login;
- l'applicazione web verifica le credenziali invocando il web service di autenticazione;
- l'applicazione web ha l'onere di gestire tutte i possibili esiti e le interazioni con l'utente (password valida, password errata, password scaduta, ecc.).

Il primo ed evidente problema di questo approccio è la difficoltà di dover implementare, all'interno di tutte le applicazioni web che invocano il web service, le logiche per la **gestione di tutti i possibili scenari** che si possono presentare; basti pensare ad esempio alla gestione delle policy di sicurezza sulle password: se l'organizzazione decide di modificare uno di questi requisiti

(la lunghezza minima della password, la complessità ecc.) bisognerà contattare i programmatori di tutte le applicazioni web per modificare l'interfaccia di cambio password di ogni singola applicazione. È evidente che, a lungo andare, una soluzione del genere non è gestibile.

Il secondo problema riguarda la sicurezza: mentre è semplice obbligare tutte le web application a comunicare con il web service di autenticazione su di un canale sicuro, non è possibile controllare che l'altra metà della comunicazione (tra il browser e la web application) sia in **HTTPS**; purtroppo è sufficiente un singolo canale non protetto per annullare la sicurezza di tutta la comunicazione.

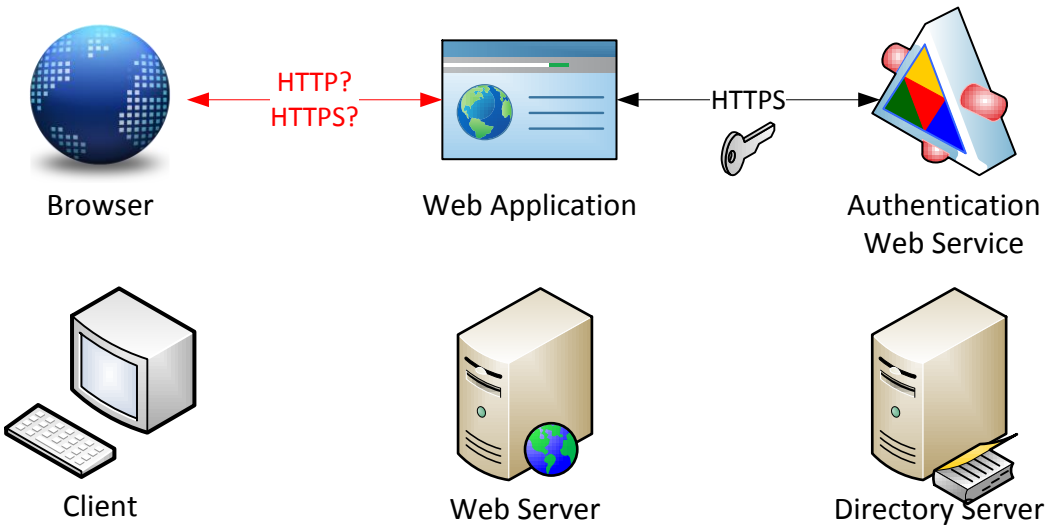


Figura 5: problemi di sicurezza tra Browser e Web Application

Infine se la Web Application è implementata e gestita da un soggetto esterno all'organizzazione si aggiunge un problema di **fiducia**, dal momento che dall'applicazione passano tutte le coppie username/password degli utenti che vi accedono; le password potrebbero anche essere salvate ad ogni login su di un database locale, e i gestori dell'applicazione non avrebbero alcuno strumento a disposizione per accorgersene.

1.3.3 Login tramite Smart Card

Le Smart Card sono uno strumento sempre più diffuso ai fini dell'autenticazione, grazie alla sicurezza garantita dal loro **chip crittografico** e alla semplicità di utilizzo per l'utente finale.

L'autenticazione ad applicazioni web tramite Smart Card avviene tipicamente tramite applet Java, l'unica tecnologia oggi in grado di interagire con i chip crittografici delle carte su diversi sistemi operativi.

In questo scenario l'utente deve:

- inserire la sua Smart Card in un lettore collegato al PC;
- collegarsi all'applicazione web, che tramite l'applet Java accederà al certificato di autenticazione presente nella carta e genererà un **challenge** in base alla chiave pubblica del certificato;
- inserire il PIN; l'applet Java utilizza il PIN per accedere alle funzionalità crittografiche della Smart Card che utilizzano la chiave privata del certificato per cifrare il challange;
- l'applicazione web verifica la stringa firmata dall'applet Java tramite la chiave pubblica associata all'utente e pubblicata dalla Certification Authority;
- un metodo alternativo prevede l'utilizzo del certificato di sottoscrizione anzichè quello di autenticazione;
- se il PIN è verificato, l'applicazione identifica l'utente tramite il codice fiscale (o un altro identificativo univoco dell'utente) presente nel certificato.

Il login tramite Smart Card, nonostante fornisca un alto livello di sicurezza, presenta alcuni problemi pratici:

- L'utente può accedere all'applicazione solo se ha la Smart Card con sé;
- Il computer che utilizza per accedere all'applicazione deve essere dotato di un lettore Smart Card, oppure l'utente deve portare sempre con sé un lettore Smart Card USB;

- Come tecnologia, sia le Smart Card che i certificati che contengono a bordo non sono una scelta economica.

Per questi motivi le Smart Card vengono oggi utilizzate principalmente in contesti che richiedono un alto livello di sicurezza e requisiti come la non repudiabilità e l'integrità dei dati.

2 IL SINGLE SIGN-ON

Gli scenari e le tecnologie presentati finora per la gestione dell'autenticazione nell'ambito delle applicazioni web presentano notevoli svantaggi:

- Non è sempre possibile garantire la sicurezza delle credenziali dell'utente: attacchi di diverso tipo possono consentire ad un malintenzionato l'accesso alle credenziali o al token di autenticazione dell'utente
- L'integrazione tra applicazioni afferenti a domini di sicurezza diversi non è facile, ogni dominio ha una sua gestione dell'identità isolata e indipendente dalle altre
- L'utente ha credenziali diverse per ogni dominio di sicurezza al quale deve accedere

Gli ultimi due punti causano un ulteriore abbassamento del livello di sicurezza delle credenziali: quando un utente deve registrare diversi account tende o ad utilizzare sempre le stesse password, a selezionare delle password molto semplici e facili da ricordare o a scriverle su fogli e file di testo. Questi comportamenti semplificano gli attacchi alle password: se un malintenzionato entrasse in possesso della password relativa ad un account gli sarebbe facile accedere anche ad altri servizi per i quali l'utente ha registrato la stessa password, oppure se l'utente avesse selezionato una password molto semplice questa sarebbe più vulnerabile ad attacchi di tipo brute-force basati su dizionario.

La soluzione al problema della proliferazione delle credenziali è il **Single Sign-On**: questo termine, che in origine veniva utilizzato solo per indicare una particolare proprietà dei sistemi di autenticazione, ovvero la capacità di effettuare il login una sola volta per accedere a diverse risorse o applicativi, oggi ha assunto il significato esteso di sistema che consente ad un utente di accedere ad applicazioni o risorse appartenenti a diversi domini di sicurezza utilizzando le credenziali afferenti a uno di questi domini.

Prima dell'introduzione del Single Sign-On un utente doveva ricordarsi un numero di credenziali proporzionali al numero di organizzazioni (scuole, aziende, enti pubblici) di cui faceva parte per accedere ai diversi servizi offerti.

Inoltre la gestione delle credenziali (verifica validità password, controllo scadenza password, cambio password, aggiornamento dei recapiti, ecc) doveva essere implementata all'interno di ogni singola applicazione/risorsa, moltiplicando il lavoro necessario ad aggiornare ogni singola procedura per il numero totale delle risorse.

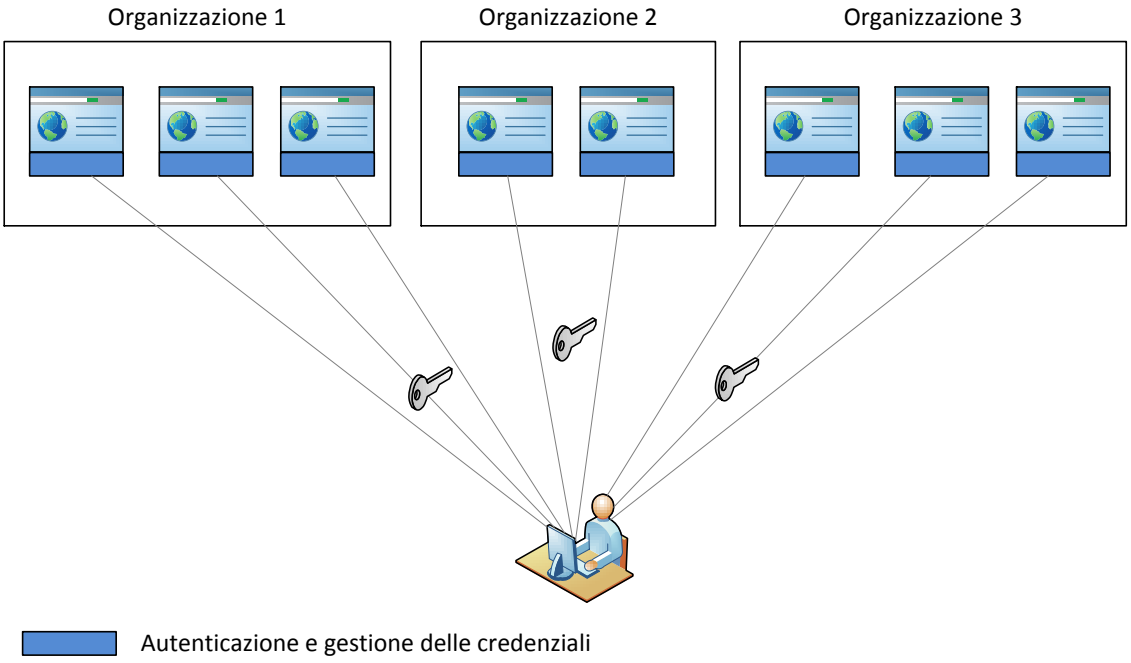


Figura 6: La profilerazione delle credenziali in diversi domini di sicurezza

Il Single Sign-On permette la rimozione dello strato di gestione delle credenziali da ogni singola applicazione per avere un unico punto centralizzato per ogni organizzazione: l'Identity Provider.

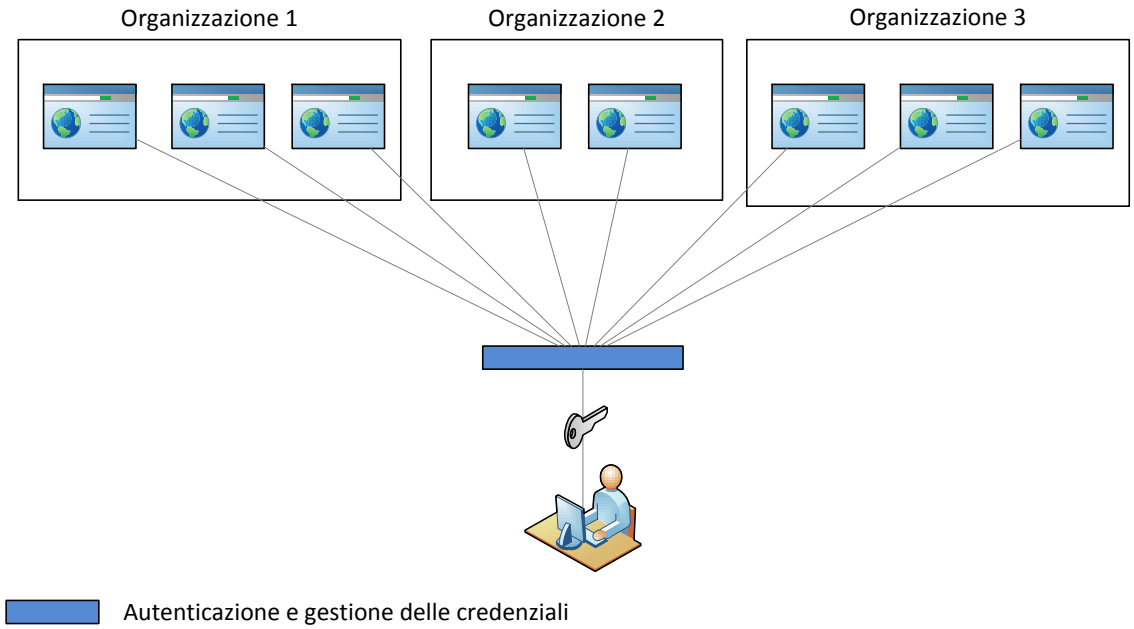


Figura 7: La semplificazione introdotta dal Single Sign-On

Al momento dell'accesso ad una applicazione l'utente seleziona da un elenco l'Identity Provider della sua organizzazione, il suo browser viene rediretto alla pagina di login dell'IDP selezionato e, dopo aver inserito le credenziali, si ritrova autenticato all'applicazione.

Una delle caratteristiche del Single Sign-On è che se in un secondo momento l'utente accederà ad un'altra applicazione (della stessa organizzazione o di un'organizzazione appartenent al *Circle Of Trust*) si ritroverà autenticato senza dover inserire le credenziali una seconda volta.

2.1 Flusso di navigazione

La navigazione dell'utente finale parte dal servizio al quale vuole accedere: il servizio utilizzato come esempio per i prossimi screenshot è il FileSender del GARR.

L'utente apre quindi il browser alla pagina https://filesender.garr.it:

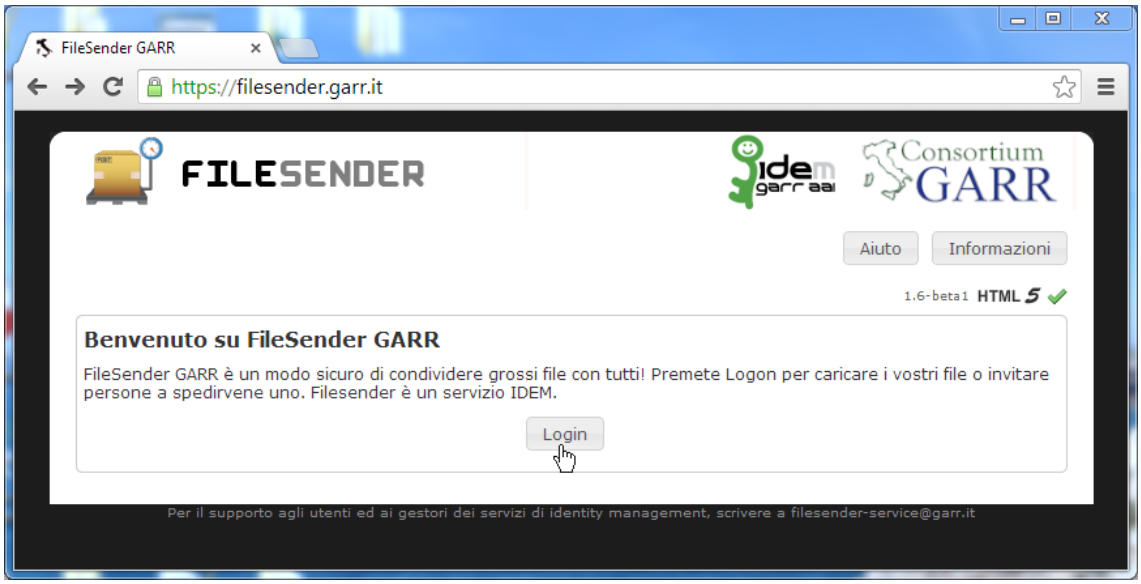


Figura 8: FileSender, pagina introduttiva

Il click sul pulsante Login scatena il processo di autenticazione, visto che l'utente sta accedendo ad una risorsa protetta.

L'applicazione web mostra quindi all'utente la pagina di WAYF (Where Are You From), che gli consente di selezionare la propria Home Organization.

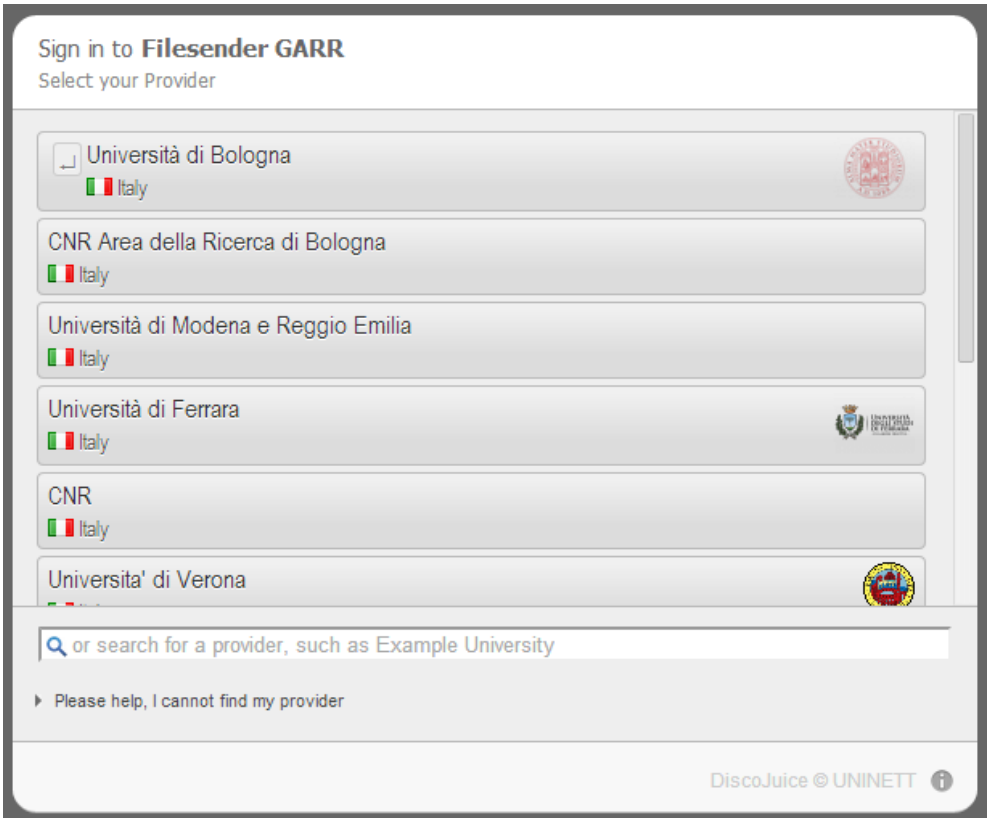


Figura 9: Where Are You From (WAYF) per la selezione dell'Identity Provider

Dopo aver selezionato l'Identity Provider adatto (in questo caso "Università di Bologna") il browser dell'utente viene rediretto alla pagina di login dell'IDP.

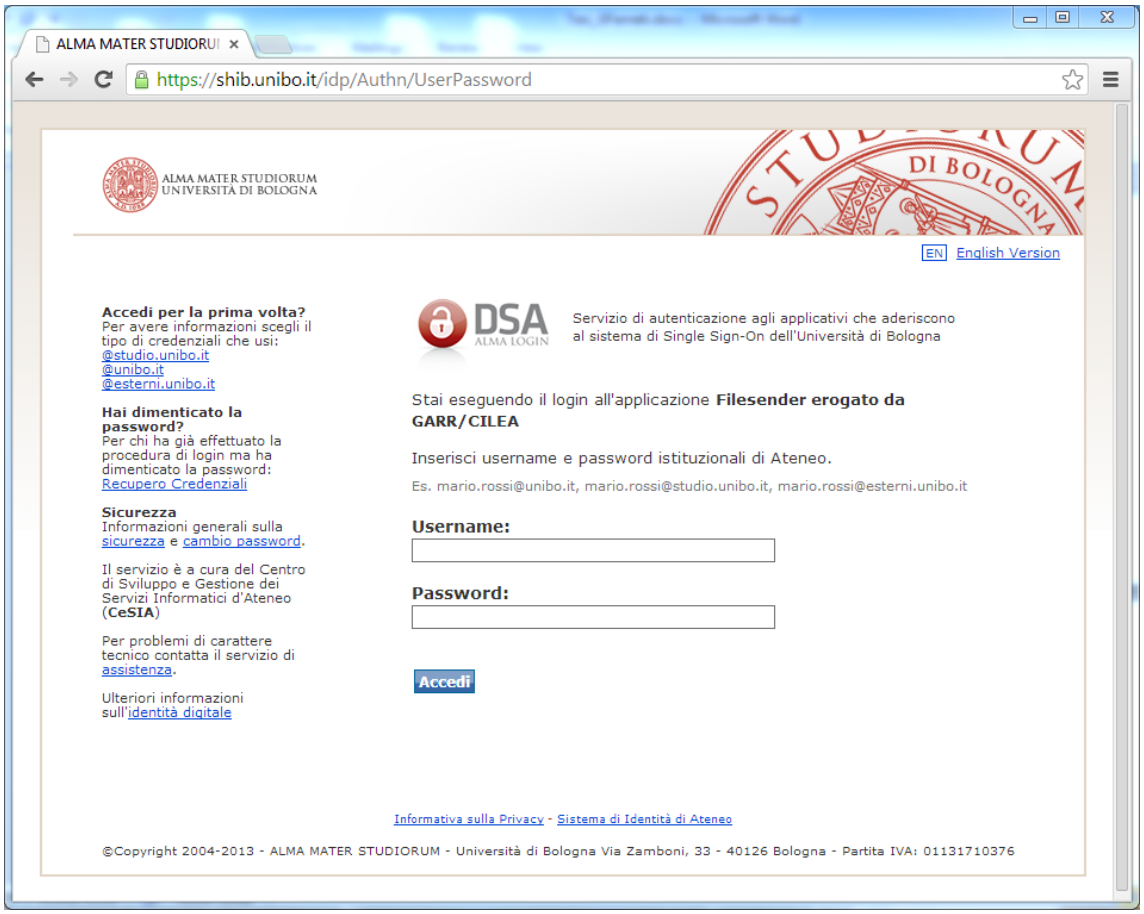


Figura 10: La pagina di login dell'IDP

L'IDP gestisce la verifica delle credenziali, la gestione della password ed eventualmente i recapiti dell'utente (numero di cellulare, indirizzo e-mail personale, ecc).

Dopo che l'utente ha inserito le proprie credenziali l'IDP genera un token di autenticazione che contiene i dati sull'utente necessari al corretto funzionamento del servizio; il set di dati richiesto può variare da servizio a servizio, ma alcuni dati di base (nome, cognome, indirizzo email) vengono sempre restituiti.

Il token viene passato dall'IDP al servizio in maniera del tutto trasparente all'utente, tipicamente tramite una chiamata POST a un endpoint concordato in precedenza. Dopo questo processo l'utente si ritrova autenticato al servizio.

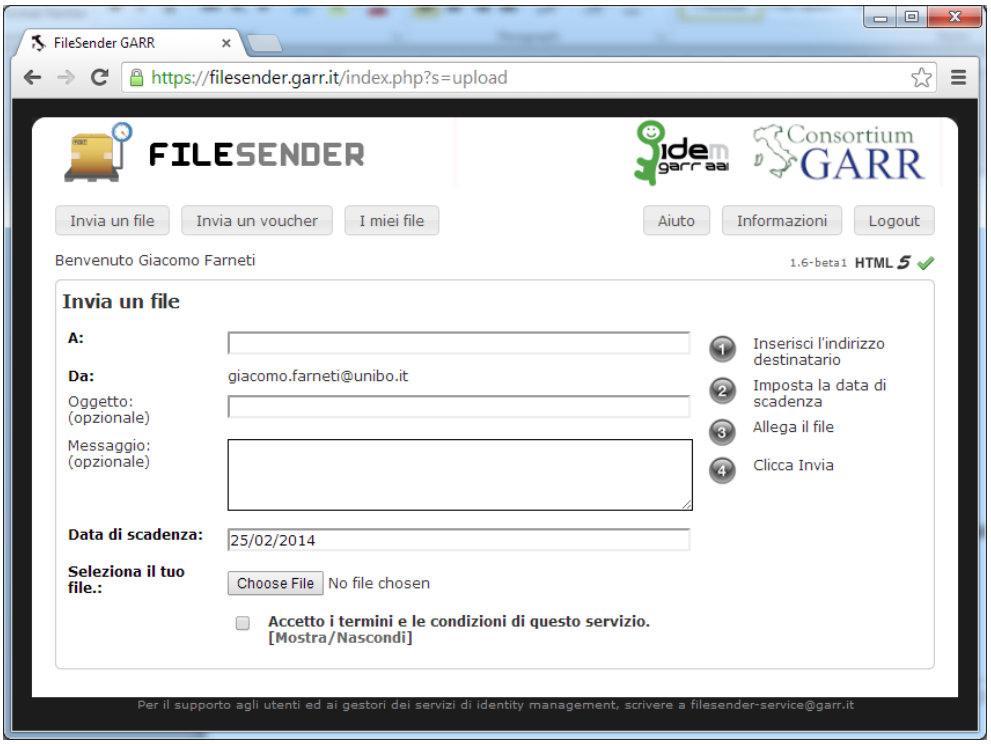


Figura 11: L'utente è autenticato al servizio

Alcune federazioni supportano anche il Single Sign-Off (o Single Logout), una funzionalità per eseguire un login centralizzato che disconnette da tutte le applicazioni alle quali si è collegati.

Figura 12: Il Single Sign-Off dell'IDP dell'Università di Bologna

2.2 Circle Of Trust (COT)

Lo schema illustrato precedentemente prevede che le tre organizzazioni instaurino un rapporto di fiducia reciproco: ognuna di esse si deve fidare dei dati inviati dagli IDP delle altre organizzazioni.

Questo rapporto viene formalizzato tramite accordi di fiducia (tipicamente contratti firmati dai partecipanti) e accordi tecnici (protocolli utilizzati, *schema* dei dati scambiati, metadati relativi a servizi e IDP).

Un raggruppamento di COT è detto **Federazione**: una federazione contiene al suo interno un grande numero di organizzazioni partecipanti che possono essere raggruppate in base al COT di appartenenza: ad esempio la federazione dell'Emilia Romagna (FedERa), gestita dalla società Lepida, contiene al suo interno diversi COT in base alla password policy delle organizzazioni partecipanti (basato sulla complessità e la frequenza di aggiornamento delle password degli utenti) e in base al tipo di riconoscimento dell'utente in fase di provisioning (self-provisioning, registrazione verificata, riconoscimento de visu).

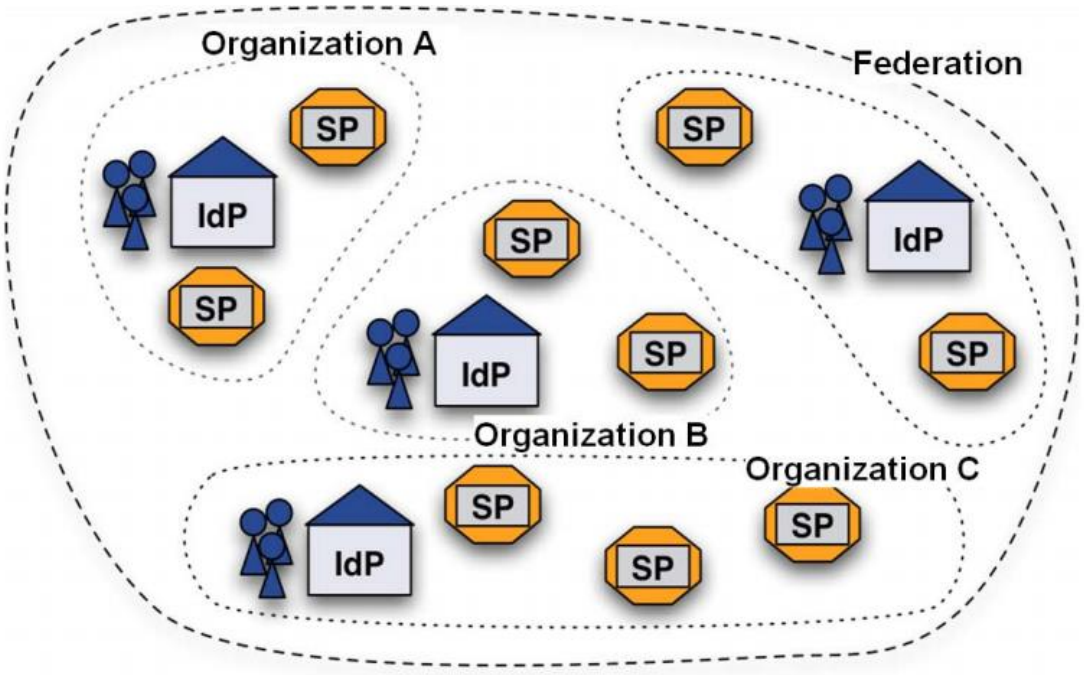


Figura 13: Rappresentazione schematica di una federazione

Organizzazioni appartenenti alla stessa federazione ma a COT diversi non sono quindi legate da un rapporto di trust e gli utenti di tali organizzazioni non potranno accedere ai servizi di organizzazioni diverse da quella di appartenenza, se non registrandosi ad esse e accedendo con un nuovo set di credenziali.

2.3 Where Are You From (WAIF)

Il Where Are You From (WAIF, vedi Figura 9) è il meccanismo che permette all'utente di selezionare l'IDP della propria organizzazione: all'interno di un COT ci sono infatti diversi IDP (uno per ogni organizzazione partecipante), tutti equivalenti e utilizzabili dall'utente.

Il meccanismo di WAYF è gestito dalla federazione, ed è presente solo se l'utente sta cercando di accedere ad un servizio che è reso disponibile all'interno di una federazione. Se il servizio al quale si vuole accedere non è inserito all'interno di una federazione l'utente finale inserirà le credenziali direttamente nell'IDP dell'organizzazione senza prima passare da un WAYF; questo perchè il servizio è accessibile solo tramite l'account dell'organizzazione di appartenenza e non avrebbe quindi senso dover selezionare un IDP.

Il WAYF può essere gestito direttamente dalla federazione oppure configurato direttamente all'interno del servizio appartenente alla federazione (come nel caso di FileSender, che integra nell'applicazione la selezione dell'IDP); in questo secondo caso esistono diverse soluzioni open-source da integrare nel servizio, la più famosa delle quali è DiscoJuice (http://discojuice.org/).

2.4 Le federazioni di identità

Le federazioni di identità risolvono il problema degli **Identity Silos**: grandi repository di identità digitali appartenenti a organizzazioni diverse che contengono credenziali e dati relativi agli utenti delle varie organizzazioni.

I problemi principali degli Identity Silos sono:

- la ridondanza del dato: i dati dell'utente sono replicati nei diversi sistemi

- l’aggiornamento del dato: l’utente non si ricorda di aggiornare i suoi dati su tutti i sistemi di identità che utilizza
- la proliferazione delle credenziali: l’utente deve memorizzare una grande quantità di username e password, finendo per utilizzare sempre le stesse e abbassando il livello di sicurezza

L’obiettivo delle federazioni di identità è quello di abbattere le barriere presenti tra i diversi “silos”, mettendoli in comunicazione e instaurando un rapporto di *trust* tra i partecipanti.

Le federazioni di identità semplificano la vita sia agli utenti finali che ai system administrators delle organizzazioni partecipanti, eliminando processi ridondati (ad esempio il riconoscimento *de visu* degli utenti) e rendendo il processo di login omogeneo e familiare per l’utente finale.

Le federazioni sono solitamente organismi pubblici gestiti a livello nazionale, e possono anche essere a loro volta raggruppate in federazioni più grandi dette **interfederazioni**.

2.4.1 Le federazioni italiane

L’Università di Bologna partecipa a due federazioni molto importanti nell’ambito degli enti pubblici e di ricerca italiani: la federazione **IDEM** del GARR e la federazione **FedERa** di Lepida.

IDEM è la Federazione Italiana delle università e degli enti di ricerca per l’autenticazione e l’autorizzazione, gestita dal GARR, e presenta le seguenti caratteristiche:

- Facilita la condivisione di servizi all’interno della comunità scientifica e accademica
- Crea un ambiente di identità verificate a supporto delle attività di ricerca e didattica
- Concentra le energie sulle risorse da rendere disponibili e non sulla gestione dell’identità, che viene delegata alla singola organizzazione

Ad oggi (Febbraio 2014) la federazione IDEM conta 74 Identity Provider che consentono l'accesso a 90 risorse. Le organizzazioni afferenti alla comunità GARR sono 45, a cui si aggiungono 21 partner internazionali.

I servizi che possono essere acceduti tramite la Federazione IDEM sono sia nazionali (accesso a reti WiFi delle organizzazioni partecipanti, applicazioni web di CNR, INFN e CINECA, e-learning, librerie digitali e wiki, il già citato FileSender) sia internazionali (librerie digitali e risorse elettroniche quali Elsevier Science Direct, IEEE Xplore, SpringerLink e molte altre).

Lo standard utilizzato dalla federazione è SAML, e la maggior parte dei partecipanti (sia come gestori dell'identità sia come servizi) utilizza il framework open-source Shibboleth.

FedERa (Federazione degli Enti dell'Emilia-Romagna per l'Autenticazione) è invece la federazione, gestita da Lepida S.P.A., che opera all'interno della regione Emilia-Romagna per mettere in comunicazione i diversi soggetti pubblici –università, comuni, province, enti– di questa regione.

FedERa ha caratteristiche simili a quelle di IDEM ma è più orientata al mondo della pubblica amministrazione locale; alcuni esempi di servizi offerti sono:

- Il portale dei servizi online del comune di Bologna
- L'interscambio dei dati digitali dei cittadini della provincia di Rimini
- L'osservatorio prezzi della Regione Emilia Romagna.

Ad oggi FedERa conta 314 Identity Provider (di cui 290 sono comuni, per molti dei quali il sistema di identità è integrato e gestito direttamente dal sistema FedERa) e 112 servizi (di cui 11 sono servizi di accesso a reti WiFi).

Una delle caratteristiche di FedERa è quella di dividere Identity Provider e Service Provider in cinque diversi Circle Of Trust in base al metodo di riconoscimento dell'utente e alle password policy:

- Livello di Affidabilità **Basso**: self-provisioning dell'utente; nessuna password policy
- Livello di Affidabilità **Medio**: utente riconosciuto tramite verifica del numero di telefono; nessuna password policy

- Livello di Affidabilità **Alto**: utente riconosciuto *de visu* oppure autenticato tramite Smart Card; nessuna password policy
- Livello di Affidabilità Alto con Password Policy "**Dati Personali**": utente riconosciuto *de visu* oppure autenticato tramite Smart Card; richiesta una password complessa con scadenza a 6 mesi
- Livello di Affidabilità Alto con Password Policy "**Dati Sensibili**": utente riconosciuto *de visu* oppure autenticato tramite Smart Card; richiesta una password complessa con scadenza a 3 mesi

3 DESCRIZIONE DEL PROBLEMA

Il sistema di gestione delle identità dell'Università di Bologna include fin dal 2010 un Identity Provider Microsoft **Active Directory Federation Services** (ADFS), che si integra perfettamente nell'infrastruttura di IdM dell'Università (fortemente incentrata sulla tecnologia Microsoft Active Directory).

Tale Identity Provider è stato utilizzato per implementare il **Web Single Sign-On** di molti servizi dell'Ateneo come l'accesso alla posta degli studenti, il portale web e la intranet, la gestione del controllo accessi o il software di emissione di badge e Smart Card ai docenti.

La maggior parte dei servizi integrati era sviluppata in ambiente Microsoft .NET Framework e utilizzava la specifica WS-Federation per lo scambio dei token e dei claim di autenticazione e autorizzazione, specifica che ADFS supporta in maniera nativa fin dalla prima versione; non ci sono quindi state grosse difficoltà nella configurazione e nell'integrazione della maggior parte dei servizi dell'Ateneo.

Il problema si è presentato invece al momento della partecipazione dell'Ateneo alle **federazioni IDEM e FedERa**, entrambe basate sul protocollo SAML (nelle versioni 1.1 e 2.0) con il quale si sono fin da subito evidenziate grosse difficoltà tecniche che richiedevano numerosi workaround per funzionare; la creazione e la gestione delle configurazioni dei diversi servizi erano quindi processi complicati e proni ad errori.

Il ritardo nell'ingresso dell'Università di Bologna nella federazione IDEM portava con sé un ulteriore problema, quello dell'accesso alle risorse elettroniche.

Le risorse elettroniche sono banche dati e librerie online messe a disposizione degli utenti dell'Ateneo, spesso pubblicate da editori internazionali di grande rilievo.

Prima dell'introduzione del Single Sign-On e delle federazioni l'unico modo che gli editori avevano a disposizione per autorizzare gli utenti al download dei file era la verifica dell'indirizzo IP chiamante: ogni università doveva quindi indicare

all’editore tutti i range di indirizzi IP in suo possesso e gli utenti potevano scaricare i file solo dalla rete universitaria.

Per rendere possibile l’accesso alle banche dati anche dall’esterno della rete universitaria, il CIB (Centro Inter-Bibliotecario), dal 2011 Area Sistemi Dipartimentali e Documentali, ha istituito il servizio **Proxy Risorse** per l’accesso alla rete d’Ateneo tramite un proxy autenticato con username e password, a disposizione di studenti e docenti.

Il proxy tuttavia presentava (e presenta tuttora) alcuni problemi di gestione:

- necessita di un aggiornamento costante dei range di indirizzi IP abilitati con tutti gli editori;
- richiede l’installazione di un software ad-hoc oppure di un plugin del browser
- la configurazione del software ad-hoc o del plugin deve essere aggiornata da tutti gli utilizzatori in seguito a una modifica della configurazione (ad esempio l’hostname del proxy)
- non funziona se la connessione ad internet passa già attraverso un proxy (ad esempio all’interno di uno studentato)
- non lo si può utilizzare da smartphone e tablet
- presenta problemi di rendicontazione in quanto non è possibile tenere traccia in modo preciso di tutte le risorse elettroniche alle quali gli utenti accedono

I problemi appena elencati non si presenterebbero se per l’accesso alle risorse elettroniche gli utenti potessero usare l’IDP dell’Università integrato all’interno di una federazione, in quanto l’unico requisito per l’utilizzo è un browser (oggi presente in tutti i sistemi operativi per smartphone e tablet oltre che su quelli per personal computer).

Nei paragrafi successivi verrà illustrato lo stato del sistema di gestione di identità dell’Università di Bologna prima del mio lavoro di installazione di un server IDP basato sulla tecnologia Shibboleth, volto a risolvere i problemi appena elencati.

3.1 Il sistema di identità dell'Università di Bologna

3.1.1 Architettura di Active Directory

Il sistema di IdM dell'Ateneo si appoggia su un'infrastruttura Microsoft Active Directory e Microsoft SQL Server definita DSA (**Directory Service di Ateneo**).

Il DSA accentra le informazioni presenti nei diversi sistemi di gestione anagrafiche dell'Ateneo (database della formazione, del personale, database applicativi, ecc.) in un'unica base dati distribuita allo scopo di fornire agli utenti le credenziali istituzionali univoche utilizzate per l'autenticazione ai servizi di Ateneo e a quelli federati.

Il DSA contiene sia le informazioni personali di un utente (nome e cognome, username e password, e-mail, matricola, ecc.) sia le autorizzazioni espresse tramite gruppi di sicurezza.

I motivi per cui è stato deciso di utilizzare un Directory Service (in particolare Active Directory) sono molteplici:

- è nativamente organizzato in utenti, gruppi, risorse informatiche (computer, stampanti)
- è sicuro: le comunicazioni fra i server sono protette, nemmeno gli amministratori possono accedere ai dati criptati (password)
- è robusto e ridondato: può funzionare anche in presenza di interruzione della connettività su parte della rete
- è progettato per essere robusto: le operazioni di sincronizzazione tra i server sono automatiche
- è efficiente, ottimizzato per la lettura: il tempo impiegato per la verifica delle credenziali è ridotto al minimo
- ha una struttura standardizzata, ma estendibile, che la rende facilmente integrabile dalle applicazioni (es: Exchange, Oracle)

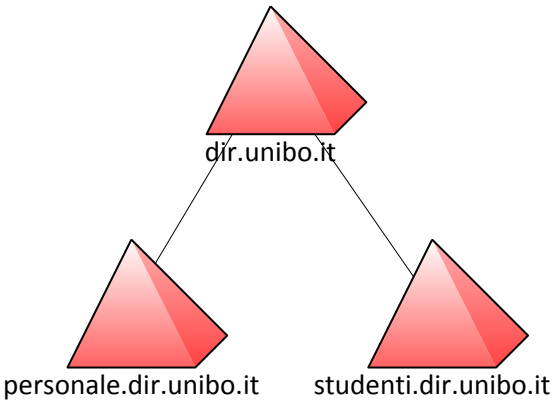


Figura 14: La struttura logica del Directory Service di Ateneo

L'immagine precedente mostra la struttura del **Directory Service di Ateneo** (DSA): i due alberi del personale e degli studenti sono rami (domini) figli del nodo radice **dir.unibo.it**, e contengono rispettivamente tutti gli oggetti –account, gruppi e unità organizzative– relativi ai dipendenti (account docenti e personale TA, uffici e dipartimenti) e agli studenti (account studenti, corsi di laurea e facoltà).

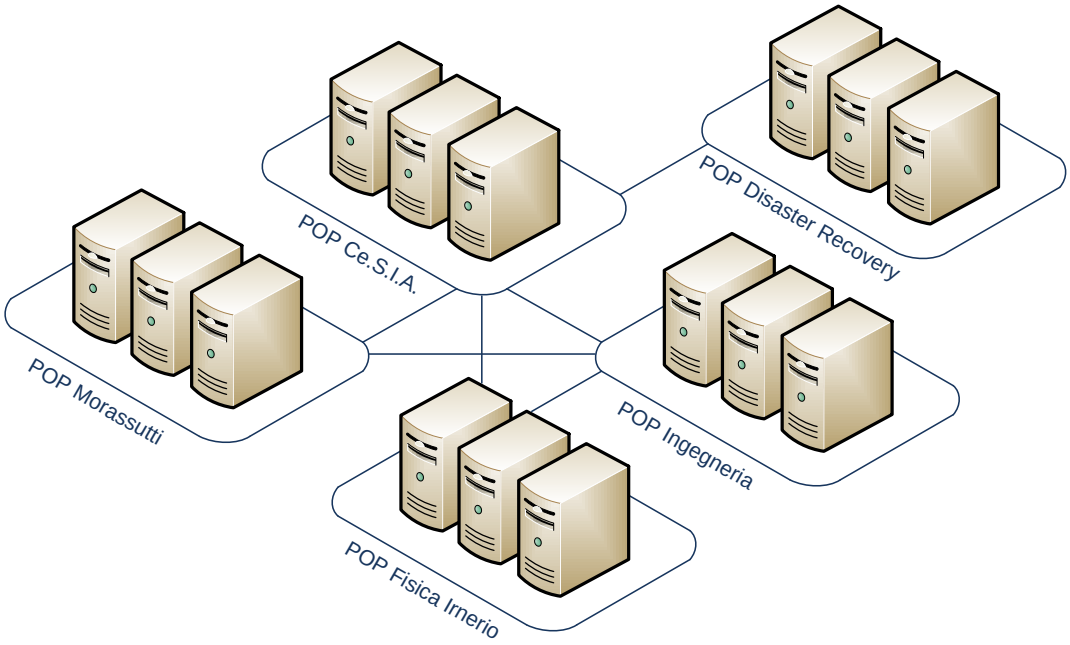


Figura 15: La struttura fisica del DSA

La struttura fisica del DSA è invece illustrata nell'immagine precedente, nella quale si può notare la ridondanza dei Domain Controller in POP (**Point Of Precence**) fisicamente distanti tra loro in grado di fornire continuità di servizio nel caso di problemi su alcuni Domain Controller o di mancato accesso di rete verso alcuni server.

La struttura qui illustrata rappresenta quindi un modo efficiente e funzionale di gestire i dati degli utenti. Tuttavia, prima dell'integrazione di un servizio di Single Sign-On, c'erano diversi problemi nell'accesso ai dati da parte delle applicazioni web, che verranno illustrati nei due paragrafi seguenti.

3.1.2 Accesso via LDAP

Prima del Single Sign-On alcune applicazioni web accedevano direttamente via LDAP ai Domain Controller del Directory Service di Ateneo. I motivi per questa scelta implementativa erano principalmente tre:

- l'accesso via LDAP consente di verificare le credenziali dell'utente (username/password) tramite un processo chiamato **bind**, semplice da implementare;
- tramite LDAP è possibile cercare utenti e accedere ai loro attributi pubblici tramite delle semplici query (tuttavia alcuni attributi, come il codice fiscale o la domanda/risposta segreta per il reset della password, sono protetti e accessibili solo con un account autorizzato);
- tutti i Domain Controller che fanno parte del DSA sono aperti all'interno della rete AlmaNet, quindi le applicazioni possono accedervi senza richiedere autorizzazioni particolari al Ce.S.I.A., l'Area Sistemi Informativi e Applicazioni.

Per quanto l'accesso diretto via LDAP comportasse facilità di implementazione e immediatezza nell'utilizzo, presentava dei seri problemi a livello pratico e gestionale:

- le applicazioni avevano l'onere di gestire tutte le casistiche possibili relative alla situazione della password dell'utente: password scaduta, password errata, e così via. Non sempre gli sviluppatori prevedevano

tutti i casi e mostravano all'utente un errore generico di credenziali errate, senza rimandarlo alle pagine di cambio/reset password (quelle integrate nel portale DSA all'indirizzo https://www.dsa.unibo.it);

- le applicazioni spesso fornivano una loro **interpretazione errata** degli attributi letti tramite LDAP; ad esempio non considerando le differenze tra il campo che memorizza la matricola (employeeId) dell'utente nei due domini degli studenti e del personale, oppure utilizzando come identificativo univoco dell'utente il campo sAMAccountName (che in realtà è univoco solo all'interno del dominio, non a livello dell'intera foresta dir.unibo.it);
- se il Ce.S.I.A. avesse modificato lo *schema* degli attributi dell'utente, cambiando il nome o il significato di un attributo, non sarebbe stato semplice informare tutti gli sviluppatori che stavano utilizzando tale attributo (soprattutto se l'applicazione era stata creata senza informarne il Ce.S.I.A.);
- gli sviluppatori implementavano le applicazioni senza informarne il Ce.S.I.A., che in questo modo non poteva garantire né avere il controllo sulla sicurezza delle credenziali dell'utente. Gli applicativi potevano memorizzare una copia delle credenziali dell'utente in chiaro in un database locale, oppure gestirle tramite un canale non sicuro (HTTP anzichè HTTPS);

La preoccupazione principale sull'utilizzo di LDAP è quindi relativo alla sicurezza delle credenziali dell'utente, sicurezza che nella gestione di un sistema di identità è fondamentale (l'accesso alle credenziali di un utente garantisce l'accesso alla sua posta elettronica e a tutte le risorse –anche quelle di importanza critica– alle quali ha accesso): si pensi ad esempio al caso di un ricercatore che utilizza le proprie credenziali per accedere tramite web a un repository di dati genetici o alle cartelle cliniche di alcuni pazienti.

L'accesso via LDAP è ancora oggi possibile, anche perchè a livello tecnico tutti i computer registrati nel dominio dir.unibo.it devono poter eseguire il bind e fare delle query; tuttavia il Ce.S.I.A. porta continuamente avanti tutti i controlli possibili e organizza degli incontri con tutti i tecnici dell'Ateneo (gli *open days*)

per sensibilizzarli al problema e per illustrare i vantaggi apportati dal Single Sign-On.

3.1.3 Web Service DsaAuthentication

Prima del sistema di Single Sign-On, l'alternativa all'accesso diretto ad LDAP per la procedura di autenticazione e autorizzazione messa a disposizione dal Ce.S.I.A. per le applicazioni web dell'Università di Bologna consisteva nel Web Service **DsaAuthentication**, nato nel 2004.

Tale Web Service pubblicava i metodi necessari ad autenticare gli utenti dell'Università di Bologna (nei domini personale e studenti, ovvero per le credenziali @unibo.it e @studio.unibo.it), restituiva informazioni utili alla profilazione degli stessi (ad esempio il codice della sede di servizio o, nel caso degli studenti, il corso di laurea) e l'appartenenza a gruppi per permettere l'autorizzazione all'accesso alle risorse.

Nonostante la facilità di integrazione e l'interoperabilità garantita dal protocollo SOAP, l'utilizzo del Web Service condivideva con l'accesso diretto ad LDAP alcuni evidenti limiti:

- Privacy: tutte le informazioni messe a disposizione dal Web Service erano accessibili allo sviluppatore, senza limitazioni, il quale a volte le utilizzava per fare reverse engineering del sistema, rischiando di ricavarne deduzioni errate; oltre alle informazioni presenti in Active Directory, il web service DsaAuthentication forniva l'accesso a dati presenti sul database DSA, accentratore dei database della formazione e del personale;
- Sicurezza: pur consentendo l'accesso al Web Service solo e unicamente via HTTPS, era difficile verificare e/o impedire che le applicazioni sviluppate esternamente fossero o meno accessibili in HTTP; inoltre anche con il web service DsaAuthentication non si poteva escludere il rischio che venisse effettuato un password logging;
- Service Level Agreement diversificati relativamente alla qualità dei dati: come detto nel primo punto il Web Service forniva l'accesso a dati

eterogenei e per alcuni dei quali il Directory Service dell'Ateneo non era autoritativo. Per alcune informazioni i tempi di aggiornamento erano diversi, anche se spesso lo sviluppatore non ne aveva la consapevolezza. Anche la qualità del dato (origine del dato, logiche di provisioning, etc) poteva essere piuttosto variabile.

L'utilizzo del Web Service DsaAuthentication consentiva tuttavia al Ce.S.I.A. di tenere traccia di tutte le applicazioni integrate, di fornire agli sviluppatori un'interfaccia certificata e aggiornata e informazioni affidabili sull'utente e sulle sue credenziali (ad esempio il dettaglio sullo stato della password dell'utente: scaduta, errata, ecc.); forniva inoltre un canale aperto e diretto di comunicazione tra il Ce.S.I.A. e gli sviluppatori, per fornire all'utente finale un sistema sicuro e affidabile.

3.2 Il Single Sign-On di Ateneo

Con l'introduzione del Single Sign-On di Ateneo tramite l'installazione e la configurazione di una *farm* di server basati sulla tecnologia Microsoft Active Directory Federation Services (ADFS), il Ce.S.I.A. ha finalmente potuto offrire agli sviluppatori il metodo più sicuro, robusto e di facile integrazione per la verifica dell'autenticazione e dell'autorizzazione degli utenti.

3.2.1 Tecnologie utilizzate

Microsoft ADFS, inizialmente chiamato Geneva Framework, è la soluzione Microsoft per la gestione del Single Sign-On e della federazione di identità per le organizzazioni.

Il Ce.S.I.A. ha deciso di adottare questa soluzione sia per la facile integrazione con l'ecosistema software presente al Ce.S.I.A. –fortemente incentrato su soluzioni Microsoft– sia per via dell'alta affidabilità e della scalabilità garantite da Microsoft ADFS.

Il server Identity Provider (IDP) di Microsoft ADFS consente di integrare sia servizi che utilizzano il framework Microsoft **Windows Identity Foundation**

(WIF) sia quelli che sono sviluppati in ambito open-source e che utilizzano di conseguenza il framework **Shibboleth Service Provider**.

La soluzione implementata funziona molto bene quando integrata con servizi Microsoft (basati su WIF) e con i servizi Shibboleth più recenti (basati sul protocollo SAML 2.0); purtroppo però moltissimi fornitori di servizi, tra cui il gateway della federazione FedERa e la maggior parte dei Service Provider della federazione IDEM, utilizzano ancora la vecchia versione di Shibboleth (basata su protocollo **SAML 1.0 / SAML 1.1**) che presenta notevoli **difficoltà di integrazione** con il server Microsoft ADFS.

3.2.2 Architettura

I server che compongono la farm che rende possibile il servizio di Identity Provider sono divisi nei due ambienti di test e di produzione.

Ogni farm prevede due ruoli:

- ✓ i **Security Token Service** (STS) si occupano di:
  - verificare le credenziali dell'utente e le sue autorizzazioni;
  - emettere e firmare i token di autenticazione e autorizzazione;
  - verificare i token emessi in precedenza (in modo che gli accessi successivi non richiedano nuovamente l'inserimento delle credenziali);
  - verificare i messaggi firmati dai Service Providers in base ai certificati pubblicati nei metadati degli stessi.
- ✓ i **Proxy** si occupano di:
  - fornire l'interfaccia web di inserimento credenziali all'utente finale;
  - gestire i certificati SSL della web application (https://idptest.unibo.it in test, https://idp.unibo.it in produzione);
  - inoltrare le credenziali dell'utente agli STS;
  - inoltrare i token al browser dell'utente (sotto forma di cookie o tramite chiamate di tipo POST agli endpoint dei Service Provider).

STS e Proxy appartengono tipicamente a due reti diverse, richiedendo diversi livelli di sicurezza; essendo i proxy pubblicati su internet e raggiungibili

dall'esterno vengono di solito inseriti nella DMZ. Gli STS, al contrario, appartengono sempre alla rete locale protetta da firewall e non raggiungibile direttamente dai client perchè richiedono un livello maggiore di sicurezza (effettuando le verifiche delle credenziali dell'utente e occupandosi di gestire i certificati di encryption dei token).

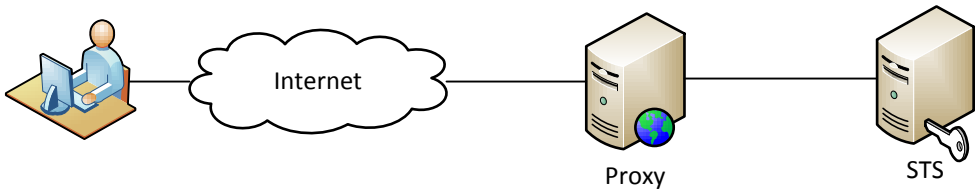


Figura 16: Architettura del servizio IDP di test

In test si è ritenuto sufficiente prevedere un solo server STS e un solo Proxy, mentre in produzione –per via del grande carico di lavoro dovuto ai login di centinaia di migliaia di studenti– sono stati installati due Proxy e due STS, entrambi dietro dei balancer che si occupano di smistare le richieste e, di conseguenza, il carico di lavoro.

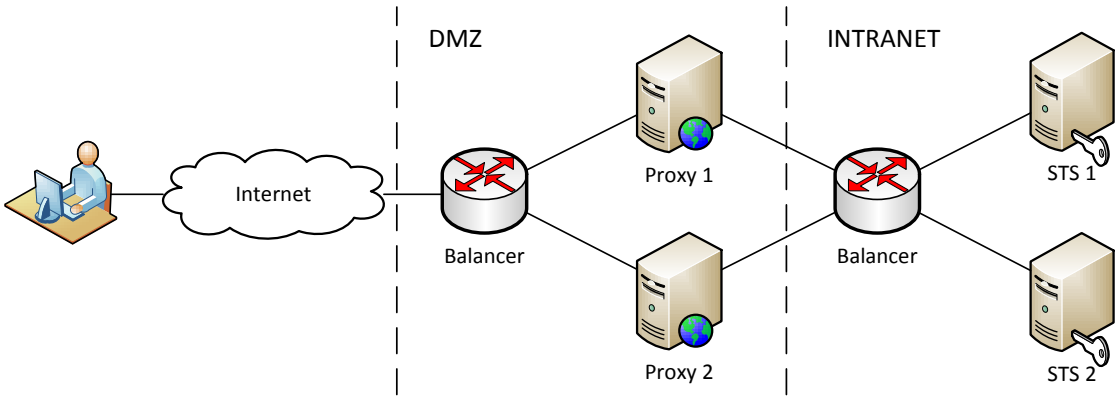


Figura 17: Architettura del servizio IDP di produzione

Come già accennato in precedenza l'Identity Provider è in grado di interoperare con servizi implementati tramite due framework diversi da un punto di vista tecnico ma simili come concetto: Shibboleth (per il mondo open source) e

Windows Identity Foundation (ambiente Microsoft), come mostrato nell'immagine seguente.

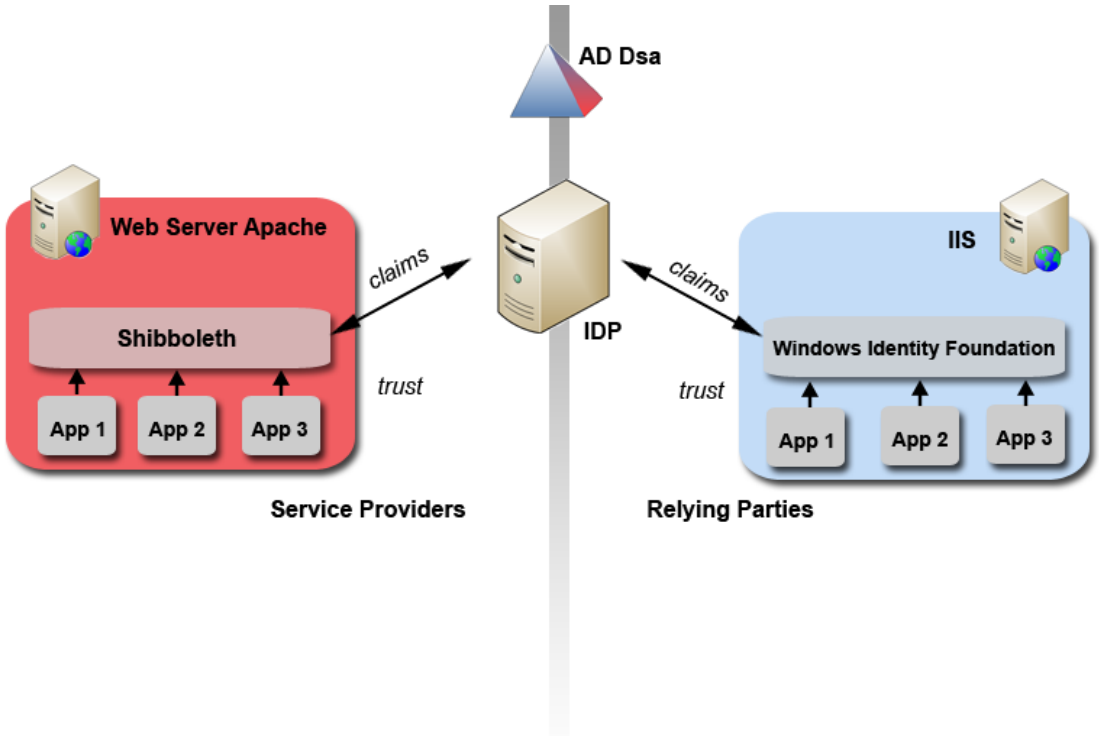


Figura 18: Architettura delle interazioni tra IdP unibo e Service Providers

L'Active Directory DSA è la sorgente dati utilizzata per autenticare, autorizzare e ottenere informazioni (*claims*) su tutti gli utenti dell'Università di Bologna.

I servizi vengono chiamati in modo differente nei due ambienti: **Relying Parties** nella terminologia Microsoft, **Service Providers** in quella Shibboleth.

3.3 Integrazione con le federazioni di identità

L'installazione del server IDP non doveva offrire solo servizi di Single Sign-On alle applicazioni web interne ed esterne, ma doveva anche essere integrato con le federazioni FedERa e IDEM per consentire l'accesso ai loro servizi tramite le credenziali dell'Ateneo.

Le prime prove purtroppo hanno mostrato che l'integrazione con entrambe le federazioni era difficile da realizzare tramite ADFS per via del suo forte orientamento al protocollo WS-Federation (utilizzato solo in ambiente Microsoft)

e a causa del linguaggio utilizzato per configurare i claim da restituire ai servizi delle federazioni: il Microsoft Claim Rule Language.

ADFS fornisce un'interfaccia a wizard semplice e intuitiva per l'emissione di semplici claim con attributi standard, che consente di selezionare gli attributi da caricare da Active Directory e il tipo di claim in uscita.

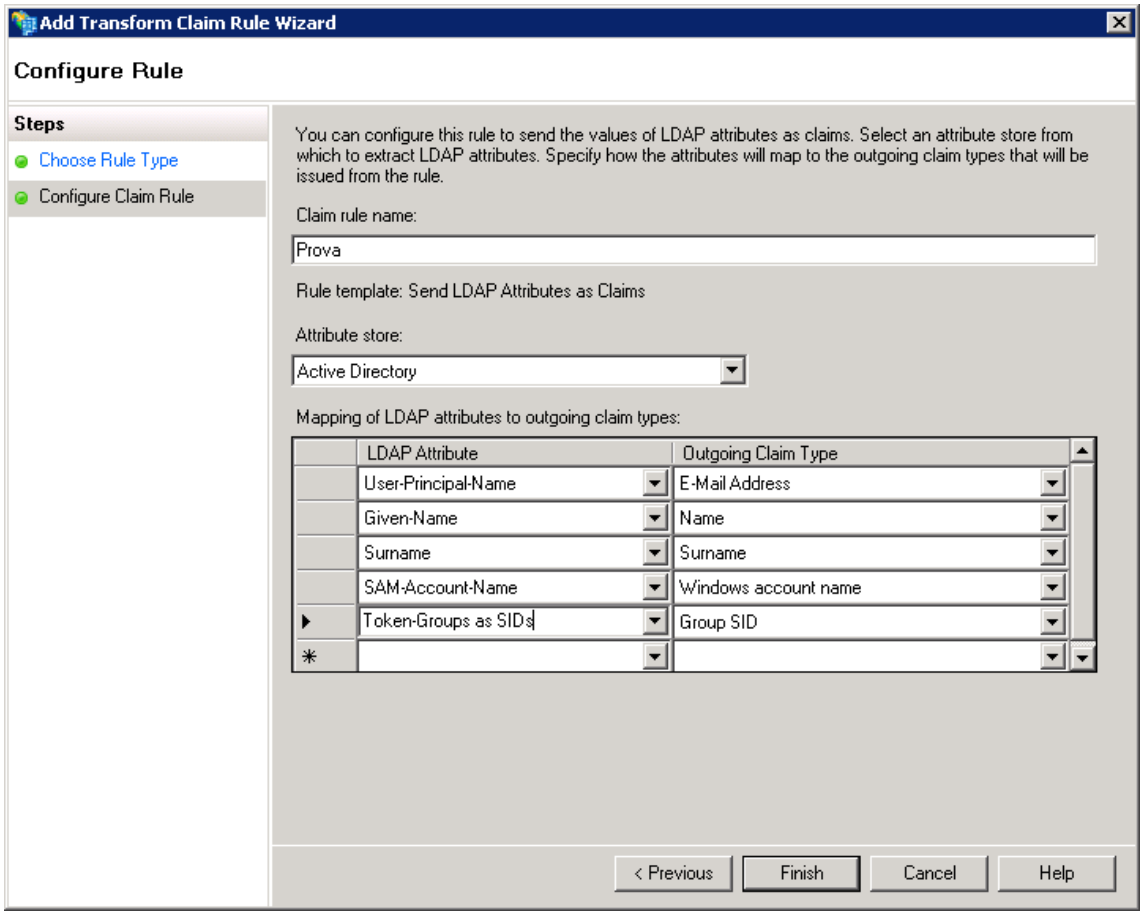


Figura 19: configurazione dei claim tramite ADFS

Per rispettare la compatibilità con il protocollo SAML e con il framework Shibboleth è invece indispensabile creare due regole custom (scritte nel linguaggio Microsoft Claim Rule Language) per ogni claim.

La prima regola carica la proprietà dell'utente da Active Directory (nell'esempio che segue la proprietà è *userPrincipalName*) ed emette un claim del tipo indicato (*http://schemas.xmlsoap.org/ws/2005/05/identity/claims/userPrincipalName)*.

```
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname"] => add(store = "Active Directory", types = ("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/userPrincipalName"), query = ";userPrincipalName;{0}", param = c.Value);
```

La seconda regola serve invece per aggiungere un attributo che è indispensabile per la compatibilità con Shibboleth, ma che ADFS normalmente non aggiunge; tale regola intercetta il claim emesso in precedenza e aggiunge l'attributo *http://schemas.xmlsoap.org/ws/2005/05/identity/claimproperties/attributename*.

```
c:[Type == "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/userPrincipalName"]
 => issue(Type = "urn:mace:dir:attribute-def:eduPersonPrincipalName", Value = c.Value, Properties["http://schemas.xmlsoap.org/ws/2005/05/identity/claimproperties/attributename"] = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri");
```

La gestione di tali regole è onerosa e complicata anche per il fatto che la documentazione sul Claim Rule Language è quasi assente, e che non tutti i claim sono semplici proprietà dell'utente (si pensi ad esempio all'appartenenza a gruppi) e richiedono workaround complicati per funzionare.

4 CONTRIBUTO PERSONALE

I problemi di integrazione tra il server IDP ADFS e le federazioni IDEM e FedERa hanno portato dopo approfondite analisi (alle quali ho contribuito assieme ai colleghi del gruppo di lavoro sul Single Sign-On) alla decisione di installare e configurare un server dedicato ad ospitare un servizio Shibboleth IdP, lavoro che ho personalmente svolto assieme al collega Cristian Mezzetti e che è oggetto di questa tesi.

La motivazione è stata supportata anche dal fatto che la maggior parte degli attori nella community del mondo della ricerca (in particolare quella universitaria) utilizza sistemi di autenticazione basati su Shibboleth, facilmente interoperabili tra loro.

Il mio lavoro è stato suddiviso in diverse fasi: inizialmente ho analizzato lo stato del pre-esistente sistema di Single Sign-On di Ateneo, basato sulla tecnologia Microsoft ADFS, individuando i punti di forza e le problematiche riscontrate più di frequente, illustrate nel capitolo precedente (3. Descrizione del problema).

In seguito ho installato e configurato il servizio Shibboleth IDP su di una macchina virtuale con sistema operativo Ubuntu Server 10.04, emulata sul mio computer tramite il software VMware Workstation fornito dal Ce.S.I.A.

Dopo le prove nell'ambiente virtualizzato ho installato e configurato l'IDP su di un Ubuntu Server 10.04.4 LTS, che ho collegato al Directory Service di test e integrato con gli ambienti di test delle federazioni FedERa (Lepida) e IDEM (GARR), oltre a configurare alcuni servizi di test scritti in PHP ed ospitati su un server web Apache.

Infine ho installato e configurato il servizio Identity Provider Shibboleth in ambiente di produzione su Ubuntu Server 12.04.2 LTS, e l'ho integrato con gli ambienti di produzione delle due federazioni FedERa e IDEM.

I dettagli operativi sull'installazione dei vari ambienti sono illustrati nel capitolo successivo (4.3. Descrizione dell'implementazione).

4.1 Identificazione dell'obiettivo

Lo scopo principale del progetto di configurazione e installazione dell'IDP Shibboleth è quello di consentire l'accesso alle risorse federate per tutti gli utenti dell'Ateneo tramite le loro credenziali istituzionali, semplificare il processo di integrazione di Service Provider basati sul framework Shibboleth e migliorare la *user experience* di quegli utenti che accedono alle risorse elettroniche degli editori convenzionati da reti esterne ad Almanet.

Il progetto prevede la creazione di diversi *deliverables* sia in **test** che in **produzione**:

- ✓ File di configurazione dello Shibboleth IDP per la connessione al Directory Service;
- ✓ Metadati dell'IDP per l'integrazione con i Service Provider;
- ✓ Servizio Shibboleth IDP installato, configurato e funzionante;
- ✓ Un Service Provider integrato con l'IDP;
- ✓ Servizi federati (IDEM e FedERa) integrati con l'IDP;
- ✓ Report sull'accesso alle risorse elettroniche per statistiche e rendicontazione.

In particolare, gli ambienti di test e produzione dovranno essere speculari; gli unici parametri differenti saranno relativi alla connessione ad Active Directory.

Una volta completato il progetto, gli utenti saranno in grado di accedere ai servizi federati utilizzando le proprie credenziali inserite nell'interfaccia web dell'IDP Shibboleth. Per l'accesso alle risorse elettroniche dalle reti esterne ad Almanet non dovranno più installare e configurare un client per il tunnelling della connessione verso il proxy del CIB, ma potranno utilizzare il proprio browser da qualsiasi dispositivo.

Obiettivo ulteriore è il miglioramento dei log di accesso alle risorse elettroniche, al fine di generare statistiche e rendicontazioni più accurate. In particolare, l'IDP Shibboleth permetterà di generare report aggregati degli accessi, per verificare quali siano le risorse più accedute e da parte di quali corsi di laurea / facoltà.

4.2 Progettazione della soluzione

Per la progettazione della soluzione abbiamo utilizzato le linee guida fornite dal consorzio Internet2 (http://www.internet2.edu/) e dalla fondazione svizzera SWITCH (http://www.switch.ch/), i due principali enti promotori del progetto.

La prima decisione da prendere è stata relativa all'architettura degli ambienti di test e produzione, in particolare riguardo al tema del *load balancing*.

4.2.1 Stateless vs Stateful

In un ambiente *load balanced* ci sono più server che ricoprono lo stesso ruolo, e un nodo esterno (il bilanciatore) decide a quale server inviare le richieste in arrivo. Il load balancing consente di migliorare le performance del sistema e abbassare i tempi di risposta, ma necessita di un requisito spesso complicato da soddisfare: i server devono essere *stateless*.

La proprietà stateless di un server fa sì che questo possa gestire ogni chiamata in modo indipendente dalle altre, a differenza di un server stateful che può correlare chiamate successive mantenendo attiva una sessione di comunicazione con il client. In un processo come quello del Single Sign-On è indispensabile mantenere attiva una sessione di autenticazione: il Single Sign-On è un processo stateful.

In una server farm con più nodi in load balancing l'unico modo per implementare un servizio stateful è utilizzare un **database esterno** per gestire lo stato condiviso tra i vari server, implementando di fatto uno storage unico per la persistenza delle sessioni. La presenza del database tuttavia rappresenta un *point of failure* aggiuntivo, oltre a richiedere un carico di lavoro notevole per la gestione e l'aggiornamento dello stesso.

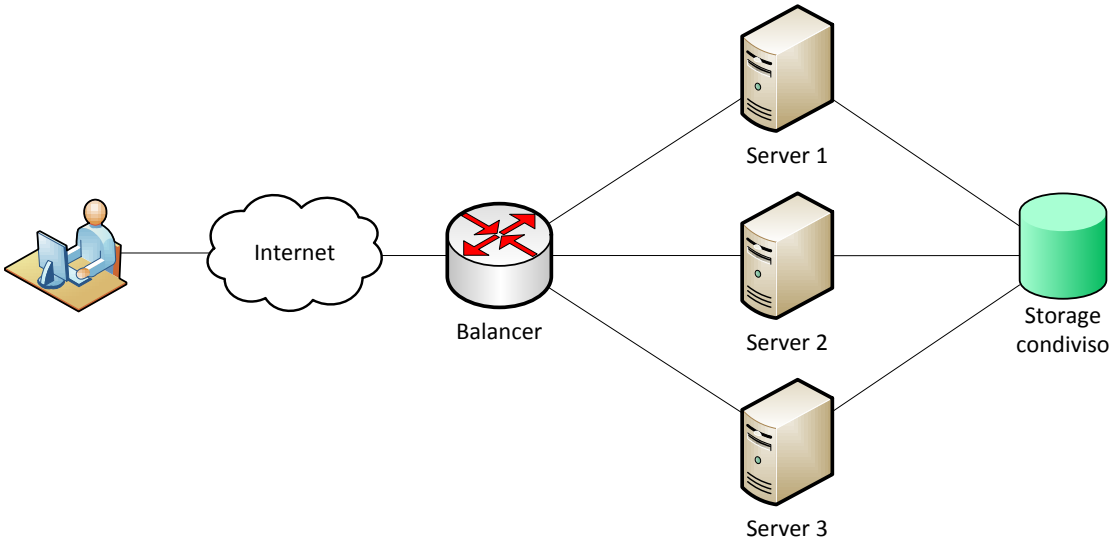


Figura 20: Esempio di load balancing con tre server e uno storage condiviso

Si è quindi deciso per i motivi appena illustrati di ospitare il servizio IDP su un solo server anzichè in una farm; la decisione è stata supportata da altre due considerazioni:

- è stato valutato che il carico di lavoro del server non sarebbe stato tale da giustificare la necessità di utilizzare una farm di server in load balancing;
- mentre l'IDP Microsoft ADFS supporta nativamente sia il load balancing che la divisione dei ruoli (proxy/sts), Shibboleth richiede soluzioni ad-hoc per entrambi (tramite l'utilizzo di software come Terracotta per il load-balancing, MySQL per il database e Apache per i web server nel ruolo di proxy).

4.2.2 Architettura

L'architettura scelta è quella di ospitare il servizio su un solo server per ogni ambiente (uno in test, uno in produzione) senza appoggiarsi ad uno storage esterno. Le comunicazioni http/https saranno gestite da un server web Apache, il servizio IDP Shibboleth verrà eseguito in Apache Tomcat e il servizio stesso sarà compilato in una servlet.

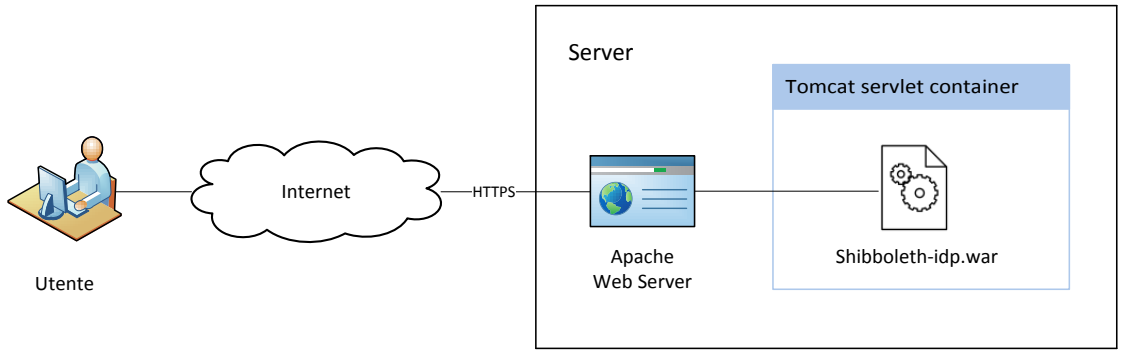


Figura 21: Architettura dei componenti del server IDP

Il server IDP di test (shibtest.unibo.it) è stato integrato con il gateway di test di **FedERa** (federatest.lepida.it), e dopo aver passato tutte le verifiche necessari la configurazione è stata copiata sul server IDP di produzione (shib.unibo.it) aggiornando i parametri di collegamento al gateway di produzione di FedERa (federa.lepida.it).

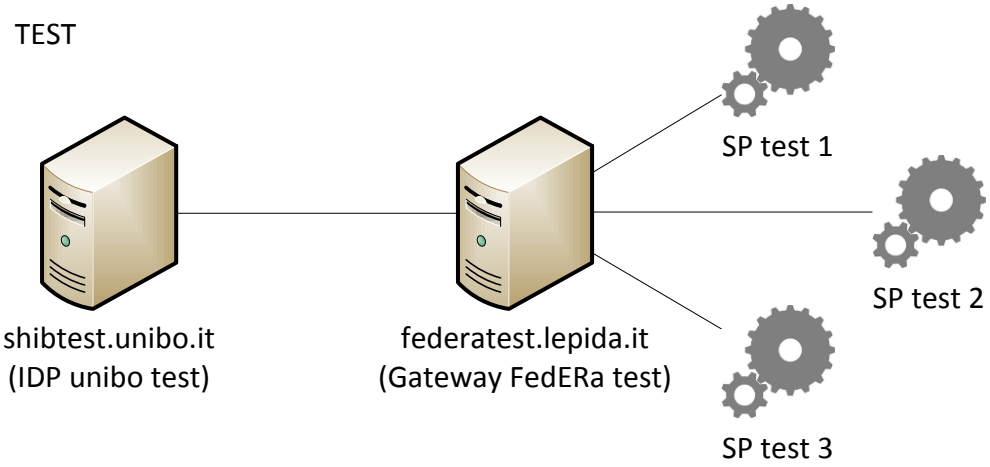


Figura 22: Integrazione con FedERa in test

Il gateway di FedERa è quindi un "ponte" tra gli IDP e gli SP partecipanti, creando quindi due relazioni (IDP ↔ Gateway FedERa e Gateway FedERa ↔ Service Providers) che FedERa gestisce sia a livello tecnico che a livello di trust.

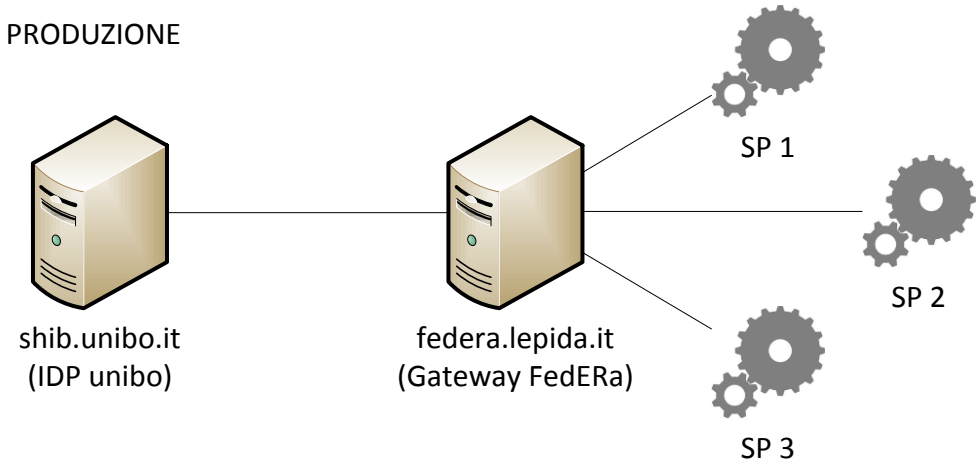


Figura 23: Integrazione con FedERa in produzione

Per l'integrazione con **IDEM** si è dovuto seguire un percorso diverso, in quanto questa federazione non possiede un gateway di comunicazione tra gli IDP e gli SP; fornisce invece un Service Provider di test (sp24-test.garr.it) da integrare direttamente con il proprio IDP, e dopo aver verificato il corretto funzionamento del login comunica a tutti i Service Provider partecipanti alla federazione i metadati dell'IDP, chiedendo di integrarli nei loro servizi; contestualmente chiede agli amministratori dell'IDP di configurare tutti i Service Provider che devono essere integrati. Questo doppio legame tra gli IDP e gli SP crea di fatto una rete di trust gestita, a livello organizzativo, da IDEM.

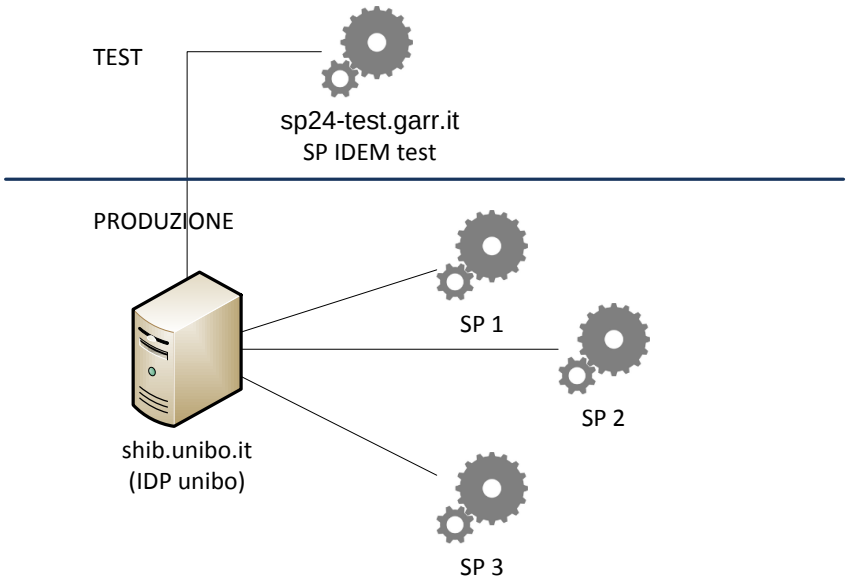


Figura 24: Integrazione con IDEM

Quindi, mentre con FedERa è sufficiente impostare nella configurazione dell'IDP un solo servizio (il Gateway FedERa, delegando la configurazione di nuovi SP a FedERa), con IDEM è necessario aggiungere e configurare manualmente i nuovi Service Provider che entrano nella federazione.

4.2.3 Ambiente software

Shibboleth è un prodotto open source scritto in Java, facilmente integrabile in ambienti GNU/Linux. Si è quindi deciso di utilizzare macchine virtuali basate su **Ubuntu Server**, il sistema operativo scelto per l'ambiente di virtualizzazione del Ce.S.I.A. al tempo dell'avvio del progetto.

L'ambiente di test è stato installato su un Ubuntu Server 10.04, mentre l'ambiente di produzione su un Ubuntu Server 12.04 (per l'ambiente di test è stato utilizzato un server creato mesi prima per effettuare delle prove di configurazione dei Service Provider Shibboleth). La differenza di versione tra gli ambienti di test e produzione non ha comportato nessun problema in quanto i pacchetti software disponibili sono praticamente gli stessi.

Il web server scelto è **Apache** nella versione 2.2, installato tramite il gestore di pacchetti apt-get.

Come ambiente di hosting delle servlet Java abbiamo deciso di utilizzare **Apache Tomcat 7** (versione 7.0.26), installata e aggiornata manualmente per avere maggiore controllo su differenze e incompatibilità tra le varie versioni.

La compilazione di Shibboleth a partire dai sorgenti richiede il Java Development Kit, disponibile nell'ambiente Ubuntu sia nella versione gestita da Oracle sia in quella open source gestita dalla community; abbiamo optato per questa seconda opzione e abbiamo installato la **OpenJDK 1.7.0**.

Infine, al momento dell'installazione dell'ambiente di test abbiamo utilizzato come IDP lo **Shibboleth Identity Provider 2.3.8**, la versione più recente disponibile in quel momento.

4.3 Descrizione dell'implementazione

Come anticipato nei capitoli precedenti, per testare le procedure di installazione e configurazione del server Shibboleth IdP prima di effettuare il deploy in produzione ho utilizzato due diversi ambienti:

- Una macchina virtuale con sistema operativo Ubuntu Server 12.04, emulata sul mio computer tramite il software VMware Workstation fornito dal Ce.S.I.A.
- Una macchina virtuale ospitata nella server farm del Ce.S.I.A. con sistema operativo Ubuntu Server 10.04

Sul mio computer ho configurato una macchina virtuale con lo stesso sistema operativo che sarebbe poi stato quello dell'ambiente di produzione, per effettuare delle prove realistiche e in linea con le caratteristiche del server che avrei utilizzato in seguito.

Poter disporre di una macchina virtuale sul mio computer è stato molto comodo e mi ha permesso di ottenere dei risultati in tempi brevi avendo totale libertà decisionale e di azione sul sistema operativo e sui pacchetti installati.

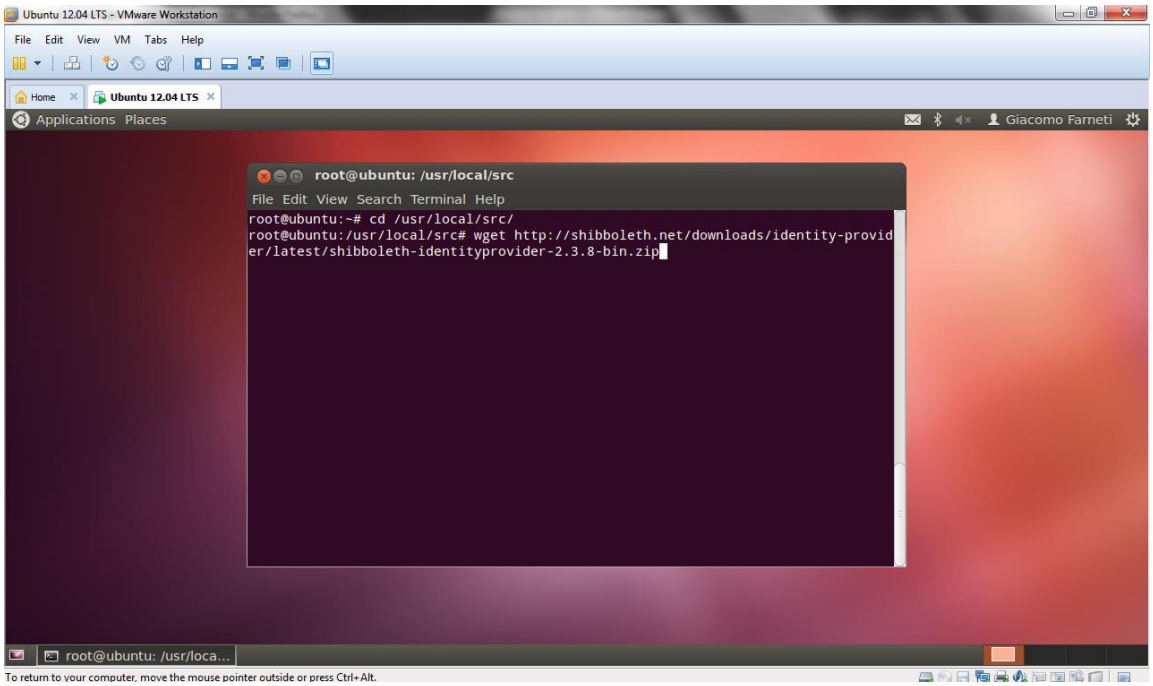


Figura 25: La macchina virtuale utilizzata sul mio computer per i test

Una volta completata la fase di test sulla mia macchina virtuale è stato deciso di utilizzare, per l'installazione dell'IDP Shibboleth di test, un server preesistente nella server farm del Ce.S.I.A. che era stato installato e configurato per ospitare alcuni servizi di prova basati sul framework Shibboleth Service Provider.

Avere accesso a un server ospitato in server farm è stato indispensabile per familiarizzare con questo ambiente, in particolare il meccanismo di login e gestione remota tramite SSH e SCP, la gestione delle credenziali (tutte le macchine sono joinate al Directory Service di Ateneo) e la procedura di richiesta di modifiche sui DNS e sui Firewall; fino a quel momento avevo infatti lavorato sempre e solo su macchine Windows Server 2003 e 2008.

Infine, per l'ambiente di produzione, abbiamo richiesto ai colleghi dell'ufficio sistemi di creare un nuovo server nell'ambiente di virtualizzazione che abbiamo utilizzato per ospitare il servizio Shibboleth IDP di produzione.

L'installazione dell'IdP Shibboleth si divide in tre fasi principali:

- Installazione e configurazione dell'ambiente e dei prerequisiti
- Installazione e configurazione del servizio Shibboleth IdP
- Personalizzazione della configurazione dell'IdP per la propria organizzazione

Tali fasi sono le stesse per tutti gli ambienti (macchina virtuale, test, produzione) e verranno illustrate nei paragrafi successivi.

4.4 Configurazione dell'ambiente e dei prerequisiti

Alcune operazioni della prima fase sono comuni a quasi tutte le installazioni di server Linux. La prima operazione da eseguire è l'aggiornamento della lista dei pacchetti software disponibili e l'upgrade di quelli esistenti:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Per il corretto funzionamento dell'IdP Shibboleth è necessario installare alcuni pacchetti software come prerequisiti, elencati di seguito.

4.4.1 *build-essential*

In ambienti Linux è comune scaricare, compilare e installare manualmente i pacchetti software a partire dai sorgenti: per farlo è necessario installare i pacchetti di base per la compilazione di software (make, gcc, ecc) raggruppati nel meta-pacchetto *build-essential*.

```
sudo apt-get install build-essential
```

Requisito fondamentale per la compilazione è la presenza degli header giusti in base alla versione del kernel utilizzato:

```
sudo apt-get install linux-headers-$(uname -r)
```

4.4.2 *OpenSSL*

```
apt-get install openssl
```

La suite di strumenti OpenSSL viene utilizzata per la gestione dei certificati e per le operazioni collegate alla gestione delle chiavi pubbliche/private:

- Creazione di certificati *self-signed*
- Verifica della validità dei certificati (sia certificati singoli che *chain* di certificati)
- Generazione di stringhe random (utilizzate ad esempio come salt del connettore ComputedId che verrà illustrato più avanti)

Altri strumenti utili alla verifica dei certificati e delle firme dei pacchetti sono:

- gnupg (GNU Privacy Guard): implementazione dello standard OpenPGP, per la firma e la verifica dei messaggi tramite chiavi pubbliche/private;
- gpgv: strumento per la verifica delle firme dei file;
- ca-certificates: installa nel *certificate store* i certificati delle CA principali su cui eseguire le verifiche.

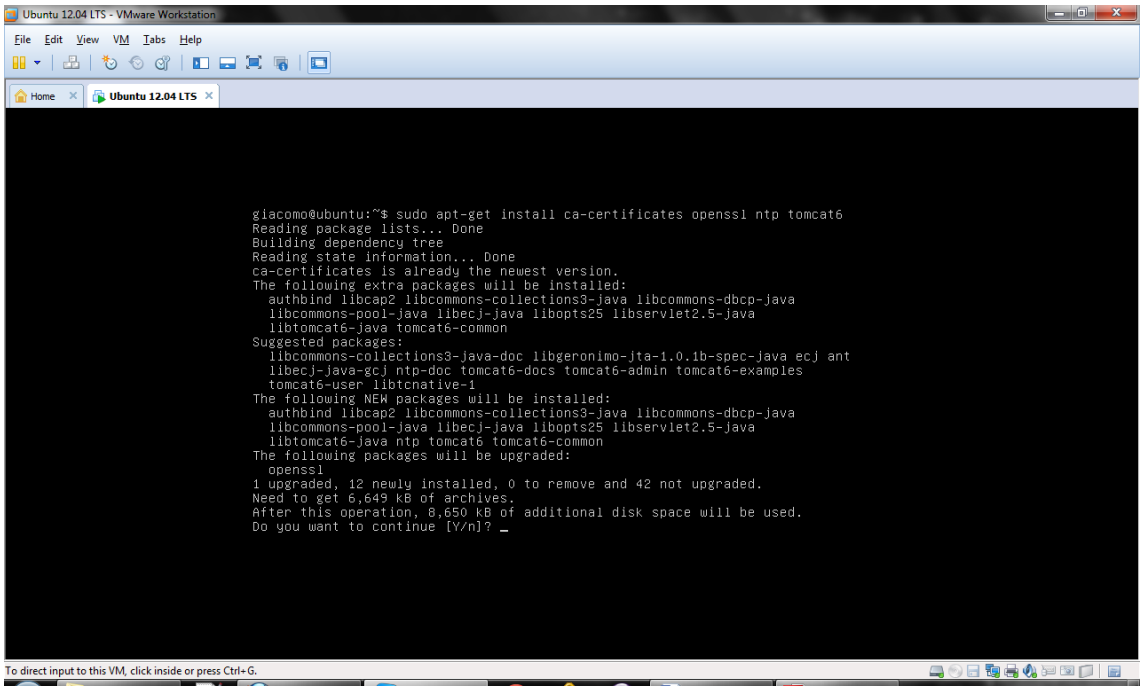


Figura 26: Installazione dei pacchetti nella macchina virtuale

4.4.3 NTP

```
apt-get install ntp
```

NTP (Network Time Protocol) è un protocollo per la sincronizzazione degli orologi utilizzato per mantenere costantemente aggiornata l'ora del server su cui viene installato.

È molto importante che il server che ospita il servizio Shibboleth IDP abbia l'orologio sempre sincronizzato con un server centrale, in quanto deve gestire gli intervalli di validità dei token, i meccanismi contro i replay attack basati sugli orari di richieste successive, e la validazione dei timestamp delle richieste che riceve dai Service Provider: se il *clock-skew* –la differenza tra gli orari di due server– tra IDP e servizio è troppo alto l'IDP può decidere di respingere la richiesta di accesso.

4.4.4 cURL

```
apt-get install curl
```

L’applicazione da riga di comando cURL può essere utilizzata per il download dei metadati dei Service Provider o di altri file da siti web; un’alternativa valida è wget, che fornisce funzionalità simili.

4.4.5 Apache

```
apt-get install apache2
```

Il web server Apache può essere installato tramite il pacchetto apache2, e configurato tramite il file /etc/apache2/apache2.conf.

Le prestazioni del web server possono essere migliorate tramite l’installazione del pacchetto aggiuntivo apache2-mpm-worker: il **Multi-Processing Module** implementa infatti un server multi-processo e multi-thread per una gestione ottimizzata delle richieste:

```
apt-get install apache2-mpm-worker
```

Il forwarding delle richieste da Apache a Tomcat richiede l’abilitazione di alcuni moduli:

```
a2enmod proxy

a2enmod proxy_http

a2enmod proxy_ajp

a2enmod rewrite
```

Per la gestione dell’endpoint HTTPS esiste invece il modulo SSL:

```
a2enmod ssl
```

Infine, una diagnostica di base dello stato del server è disponibile grazie al modulo *status*:

```
a2enmod status
```

4.4.6 Java

Una delle differenze tra Ubuntu Server 10.04 e 12.04 è la possibilità, nel primo, di installare l'implementazione Oracle del pacchetto Java Development Kit (`sun-java6-jdk2`), mentre nel secondo tale versione è stata rimossa e di conseguenza è necessario installare l'OpenJDK (`openjdk-6-jdk`), ovvero l'implementazione open source del Java Development Kit.

In alternativa è possibile installare manualmente l'Oracle JDK scaricando il file .tar.gz dal sito http://www.oracle.com/technetwork/java/javase/downloads/index.html

Le versioni del JDK appena elencate offrono le stesse funzionalità, quindi la scelta del pacchetto più appropriato dipende dal metodo di installazione preferito.

4.4.7 Variabili di ambiente

Modificando il file /etc/profile è possibile impostare le variabili di ambiente persistenti, inserendole nel formato:

```
export NOME=VALORE
```

Gli script per l'installazione di Shibboleth richiedono che le seguenti variabili di ambiente siano valorizzate:

`JAVA_HOME` (percorso del folder che contiene i binari di Java)

`TOMCAT_HOME`, `CATALINA_HOME`, `CATALINA_OUT` (i primi due devono puntare al folder di installazione di Tomcat, il terzo al percorso del file di log)

Infine, per configurare i parametri di inizializzazione della Java Virtual Machine in modo da velocizzarne l'avvio, ottimizzare l'utilizzo delle risorse e migliorarne le prestazioni per l'esecuzione dell'IdP Shibboleth su Tomcat, è possibile utilizzare la variabile JAVA_OPTS:

```
JAVA_OPTS="-Djava.awt.headless=true -Xmx512M
-XX:MaxPermSize=128m"
```

La prima opzione attiva la modalità *headless* di Java: un ambiente di esecuzione ottimizzato che non richiede dispositivi grafici o di input (display, tastiera, mouse) ideale per l'esecuzione di un web server.

La seconda opzione aumenta la quantità massima di memoria allocata nel *pool* dedicato all'esecuzione del web server dal valore di default, 64 MB, a uno più adatto alle caratteristiche del server, 512 MB.

Infine, la terza opzione aumenta la dimensione massima del permanent space, lo spazio che la JVM alloca per memorizzare classi, metodi e altri oggetti che non vengono deallocati durante l'esecuzione del codice (il default è 64 MB).

4.5 Installazione e configurazione del servizio Shibboleth IdP

Per installare l'IDP Shibboleth è necessario scaricare l'ultima versione del pacchetto dal sito ufficiale:

http://shibboleth.net/downloads/identity-provider/latest/

Al momento dell'esecuzione dei test la versione più recente era la 2.3.8, scaricabile dall'indirizzo:

http://shibboleth.net/downloads/identity-provider/latest/shibboleth-identityprovider-2.3.8-bin.zip

La prima operazione da fare è l'estrazione del pacchetto in un folder ad-hoc (nei nostri ambienti abbiamo scelto il percorso /opt/shibboleth-identityprovider-2.3.8).

Il secondo step è l'*endorse* delle librerie Xerces e Xalan: tali librerie, compilate in file jar e incluse nel pacchetto dell'IDP Shibboleth, devono essere copiate nel folder *endorsed* di Tomcat.

Questo passaggio serve per consentire al server web di effettuare il parsing dei documenti XML.

```
mkdir $CATALINA_HOME/endorsed
```

```
cp /opt/shibboleth-identityprovider-2.3.8/endorsed/*.jar
$CATALINA_HOME/endorsed
```

Per il corretto caricamento delle librerie appena aggiunte bisogna aggiungere una nuova variabile di ambiente in /etc/profile (il percorso dipenderà dall'installazione di Tomcat):

```
JAVA_ENDORSED_DIRS=/usr/share/tomcat7/endorsed
```

È quindi possibile procedure all'installazione vera e propria tramite l'installer, che consiste in uno script bash:

```
cd /opt/shibboleth-identityprovider-2.3.8/

./install.sh
```

Nel wizard di installazione inserire come directory di destinazione il percorso in cui si vuole installare l'applicazione web dell'IDP: nel nostro caso abbiamo selezionato `/opt/shibboleth-idp`.

Una volta terminata l'installazione, aggiungere al file /etc/profile la variabile di ambiente relativa al folder dell'IDP indicando il percorso scelto:

```
IDP_HOME=/opt/shibboleth-idp
```

Infine, modificare il file di configurazione di Tomcat (`/etc/tomcat7/Catalina/localhost/idp.xml`) per il caricamento dei file relativi all'IDP Shibboleth e l'esecuzione della relativa web application:

```
<Context
    docBase="/opt/shibboleth-idp/war/idp.war"
    privileged="true"
    antiResourceLocking="false"
    antiJARLocking="false"
    unpackWAR="false"
    swallowOutput="true"
    cookies="false" />
```

4.5.1 Personalizzazione della configurazione dell'IDP

Dopo aver completato l'installazione dell'IDP è necessario personalizzarne la configurazione in base ai parametri di collegamento al proprio Directory Service e ai Service Provider che si vogliono integrare.

I file di configurazione si trovano all'interno della directory /opt/shibboleth-idp/conf. Di seguito la lista dei file che è stato necessario personalizzare per l'IDP Unibo e una breve indicazione del loro utilizzo.

logging.xml

Consente di impostare i livelli di logging dei moduli che compongono il servizio. I livelli possibili per ogni modulo sono (dal livello più basso al più alto):

OFF, ERROR, WARN, INFO, DEBUG, TRACE, ALL

I moduli primari da personalizzare e i livelli impostati sono:

edu.internet2.middleware.shibboleth (servizio Shibboleth – INFO)

org.opensaml (modulo relative all'implementazione del protocollo OpenSAML – WARN)

edu.vt.middleware.ldap (connessione al Directory Service tramite VTLDAP – WARN)

Nella fase di test iniziale e in fase di troubleshooting di problemi alla procedura di login può essere utile impostare livelli di logging più alti (TRACE o ALL) per i moduli che sono all'origine del problema.

login.config

contiene i parametri per l'autenticazione degli utenti sul Directory Service:

ldapUrl: nome e porta del server LDAP

baseDn: percorso dell'*Organizational Unit* di livello superiore da cui iniziare la ricerca degli utenti

userFilter: filtro LDAP per la ricerca dell'utente in base allo username inserito nella pagina di login

bindDn: username dell'utente utilizzato per effettuare il bind su LDAP

bindCredential: password dell'utente utilizzato per effettuare il bind su LDAP

relying-party.xml

Configurazioni dei servizi (o *relying parties*) ai quali gli utenti dell'organizzazione possono accedere. Contiene anche i parametri relativi ai metadati dell'IdP.

Per ogni servizio configurato è necessario indicare:

id: l'id univoco pubblicato nei metadati del servizio;

provider: l'id dell'IDP da utilizzare per quel servizio (ovvero l'id scelto per il proprio IDP).

Altri parametri di configurazione consentono di impostare il meccanismo di autenticazione previsto, i riferimenti ai certificati di encryption per garantire la sicurezza delle comunicazioni, la durata della sessione di autenticazione, ecc.

Per l'integrazione con le federazioni FedERa e IDEM, questo file va modificato in base alle indicazioni presenti sui rispettivi siti web: FedERa fornisce un file di metadati pubblicato all'indirizzo https://federa.lepida.it/gw/metadata con i dati del gateway, mentre IDEM ne fornisce due versioni (base: https://www.idem.garr.it/docs/conf/idem-metadata.xml e firmato: https://www.idem.garr.it/docs/conf/signed-metadata.xml) che contengono all'interno i dati di tutti i servizi appartenenti alla federazione, funzionando quindi da aggregatore; configurare i metadati di tutti i singoli servizi aderenti a IDEM sarebbe un lavoro troppo oneroso.

attribute-resolver.xml

Uno dei file più complessi e più importanti, consente di configurare le sorgenti da cui caricare gli attributi degli utenti (i cosiddetti *Data Connectors*) e di impostare le regole di generazione di tutti gli attributi previsti dal proprio IdP.

Nell'IdP Unibo abbiamo configurato tre Data Connectors:

LDAPDirectory: attributi caricati dal Directory Service di Ateneo (Active Directory) tramite query LDAP.

ComputedId: utilizzato per l'emissione di un id univoco opaco generato a partire dallo username dell'utente, dall'id dell'IDP e dall'id del servizio acceduto. La stringa è calcolata tramite un MD5 che prende in input i dati appena elencati separati dal carattere ';' e una stringa di *SALT*.

La stringa di SALT può essere generata tramite OpenSSL utilizzando il seguente comando:

```
openssl rand -base64 36 2>/dev/null
```

Ogni volta che un utente accede allo stesso servizio viene generato lo stesso ComputedId: in questo modo è possibile profilare gli utenti in modo anonimo e memorizzare impostazioni personalizzate all'interno del servizio pur non conoscendo i dati dell'utente.

Static: contiene le definizioni degli attributi statici, immutabili per tutti gli utenti.

Gli attributi generati possono essere di quattro tipi:

ad:Simple: attributi caricati tramite LDAP, è sufficiente indicare il nome dell'attributo sul Directory Service e il riferimento al data connector di tipo LDAPDirectory.

ad:Script: attributi generati dinamicamente tramite uno script inserito direttamente nel file di configurazione. Il linguaggio di scripting utilizzato è ECMAScript.

ad:SAML2NameID: utilizzato per l'attributo generato dal Data Connector ComputedId

ad:TransientId: identifica in modo univoco e opaco la sessione utente, è una stringa che cambia ogni volta che l'utente effettua il primo login all'IDP, e consente di gestire la durata della sessione centralizzata di login.

attribute-filter.xml

Contiene il mapping tra gli attributi configurati nel file attribute-resolver.xml e i service providers.

Ogni service provider è rappresentato da un nodo di tipo *afp:AttributeFilterPolicy* e identificato univocamente tramite l'attributo *value* del sottonodo *afp:PolicyRequirementRule*, che contiene l'id univoco del servizio pubblicato nei metadati dello stesso.

Un elenco di sottonodi di tipo *afp:AttributeRule* consentono di indicare quali attributi vanno rilasciati al servizio.

Ci sono poi tre file di configurazione che solitamente non vanno modificati:

handler.xml: contiene le definizioni di tutti gli endpoint web del servizio IDP: quello per la pubblicazione dei metadati, per la diagnostica dello stato del servizio (normalmente questo handler viene autorizzato solo per le chiamate che arrivano dal server locale –127.0.0.1– per motivi di sicurezza), l'handler per l'attribute query SAML 1, e quello più utilizzato: lo username/password handler che mostra all'utente l'interfaccia web in cui inserire le sue credenziali.

internal.xml: contiene i parametri per la configurazione interna del servizio relativi al sistema di loggin, la gestione della memoria, della cache, dei thread e dei parser.

service.xml: contiene la configurazione dei servizi necessari al corretto funzionamento dell'IDP: collegamento ai Data Connector, gestione dei Service Provider, degli handler, ecc.

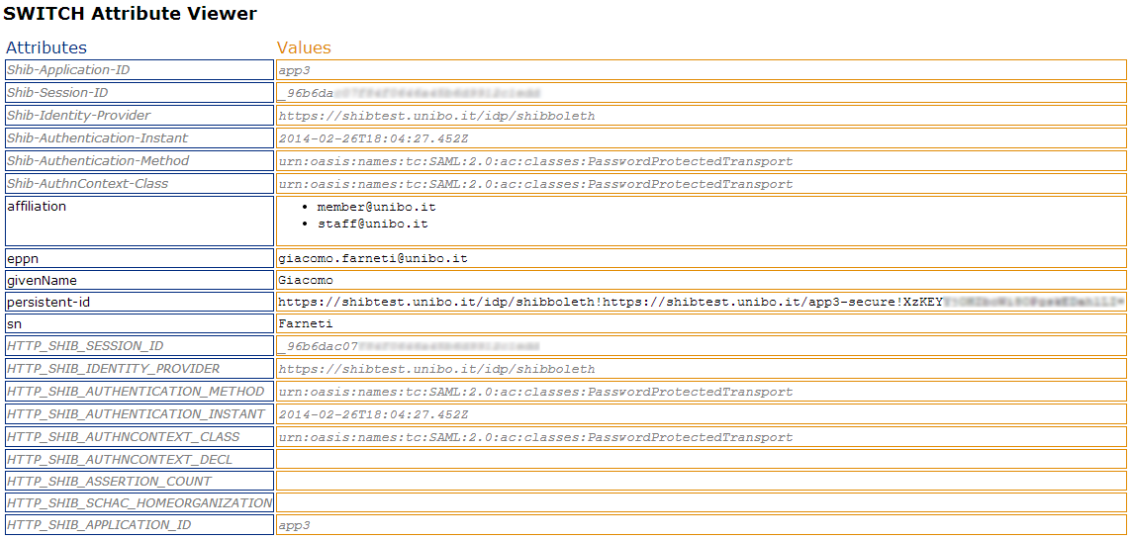
4.6 Prove sperimentali

Una volta terminata l'installazione e la configurazione del server IDP è iniziata una fase di test approfonditi, per simulare e prevedere il più possibile le problematiche che si sarebbero verificate in produzione.

4.6.1 Service Provider di test

Il server su cui abbiamo installato il servizio Shibboleth IDP di test (shibtest.unibo.it) ospita anche alcuni Service Provider di prova che utilizzano il framework Shibboleth Service Provider integrato nel web server Apache.

Tali applicazioni sono istanze dello **SWITCH Attribute Viewer**, un semplice script PHP –i cui sorgenti sono pubblici– che mostra tutti i claim ricevuti dal server IDP in una tabella che li elenca come coppie chiave/valore.



SWITCH Attribute Viewer

Attributes	Values
Shib-Application-ID	app3
Shib-Session-ID	_96b6da
Shib-Identity-Provider	https://shibtest.unibo.it/idp/shibboleth
Shib-Authentication-Instant	2014-02-26T18:04:27.452Z
Shib-Authentication-Method	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
Shib-AuthnContext-Class	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
affiliation	• member@unibo.it • staff@unibo.it
eppn	giacomo.farneti@unibo.it
givenName	Giacomo
persistent-id	https://shibtest.unibo.it/idp/shibboleth!https://shibtest.unibo.it/app3-secure!XzKEY
sn	Farneti
HTTP_SHIB_SESSION_ID	_96b6dac07
HTTP_SHIB_IDENTITY_PROVIDER	https://shibtest.unibo.it/idp/shibboleth
HTTP_SHIB_AUTHENTICATION_METHOD	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
HTTP_SHIB_AUTHENTICATION_INSTANT	2014-02-26T18:04:27.452Z
HTTP_SHIB_AUTHNCONTEXT_CLASS	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
HTTP_SHIB_AUTHNCONTEXT_DECL	
HTTP_SHIB_ASSERTION_COUNT	
HTTP_SHIB_SCHAC_HOMEORGANIZATION	
HTTP_SHIB_APPLICATION_ID	app3

Figura 27: Il Service Provider per la visualizzazione dei claim

L'utilizzo dei service provider di test è stato molto utile per eseguire le prove di integrazione con l'IDP, di aggiornamento dei metadati e di gestione dei claim restituiti.

4.6.2 Sistema di logging

Durante la fase di test ho spesso portato a "ALL" i livelli di logging dei diversi sottosistemi di cui è composto Shibboleth, in particolare quelli relativi al servizio IDP vero e proprio (edu.internet2.middleware.shibboleth) e quelli del sottosistema di collegamento ad LDAP (edu.vt.middleware.ldap).

I log vengono salvati nel folder /opt/shibboleth-idp/logs, divisi nei seguenti file:

- idp-access.log: traccia tutte le richieste ricevute dall'IDP, indipendentemente dalla risposta o dall'esito dell'operazione. I dati registrati sono data e ora della richiesta, l'indirizzo IP del client, nome e porta del server e URL della richiesta;
- idp-audit.log: contiene un record per ogni risposta restituita dall'IDP al Service Provider. Registra data e ora della risposta, gli ID di IDP e Service Provider, gli ID univoci della richiesta e della risposta, l'ID univoco dell'utente, il metodo di autenticazione e gli attributi utente resituiti;

- idp-process.log: il log più utile durante le operazioni di troubleshooting, contiene messaggi in formato *human-readable* di tutte le operazioni fatte dall'IDP, inclusi warning ed errori (completi di messaggio di eccezione e stack trace).

Il file idp-process.log, in particolare, è indispensabile per avere i dettagli relativi agli errori delle operazioni di bind e query LDAP, e al processo di generazione e emissione degli attributi utente.

4.6.3 Test delle query LDAP

Prima di configurare i due file che si collegano al Directory Service di Ateneo tramite LDAP, è fondamentale eseguire delle query di prova per verificare il funzionamento dei vari parametri.

Per le prove ho utilizzato la suite di tool **vt-ldap** (ora evoluta nel progetto **ldaptive**: http://www.ldaptive.org/), che mette a disposizione sia una libreria Java sia degli strumenti a riga di comando: per le prove ho utilizzato questi ultimi.

Una volta scaricato ed estratto il pacchetto vt-ldap-3.3.7-dist.tar.gz è possibile utilizzare gli eseguibili inclusi nel folder /bin:

- ldapauth: consente di testare il processo di autenticazione degli utenti, effettuando il bind sul Directory Service;
- ldapsearch: consente di effettuare una query LDAP sul Directory Service.

Il primo eseguibile può essere utilizzato per verificare il corretto funzionamento del bind e per provare i parametri che verranno poi configurati nel file login.config, il file che contiene la configurazione per il bind sul Directory Service: i nomi dei parametri dell'eseguibile e quelli del file di configurazione sono gli stessi.

Per effettuare il bind servono l'URL e il Distinguished Name della radice del Directory Service, le credenziali di un utente che ha i diritti per effettuare il bind e la query LDAP per trovare l'utente a partire dal suo username.

ldapauth	**login.config**	**utilizzo**
ldapUrl	ldapUrl	URL del Directory Service da utilizzare
baseDn	baseDn	Distinguished Name del nodo da cui partire per effettuare la ricerca dell'utente (nodo radice)
userFilter	userFilter	Filtro che rappresenta la query LDAP per cercare l'utente (il placeholder {0} verrà sostituito dal nome utente nella query finale)
-searchScope SUBTREE	subtreeSearch="true"	Parametro per cercare l'utente ricorsivamente in tutti i nodi figli del nodo radice
bindDn	bindDn	Username di un utente che ha i diritti per effettuare il bind sul Directory Service
bindCredential	bindCredential	Password di un utente che ha i diritti per effettuare il bind sul Directory Service

Tabella 1: Parametri per il bind sul Directory Service

Esempi di valori per questi parametri verranno illustrati nel paragrafo successivo.

Dopo aver verificato che la password inserita dall'utente sia valida, è possibile utilizzare **ldapsearch** per caricare dal Directory Service gli attributi utili alla sua profilazione.

I comandi ldapauth e ldapsearch condividono la maggior parte dei parametri di collegamento al Directory Service, ad eccezione di quello per la ricerca dell'utente: mentre nel primo caso si utilizza il parametro userFilter (che prevede l'uso di un placeholder per indicare lo username dell'utente), nel secondo caso si possono eseguire query generiche tramite il parametro query.

4.6.4 Collegamento ad Active Directory

Come illustrato al paragrafo “3.1 Il sistema di identità dell’Università di Bologna”, il Directory Service di Ateneo è diviso in tre domini:

- dir.unibo.it: è il nodo radice dell’intera directory
- personale.dir.unibo.it: contiene tutti gli account del personale (@unibo.it)
- studenti.dir.unibo.it: contiene tutti gli account degli studenti(@studio.unibo.it)

La differenza principale tra il nodo dir.unibo.it e i due nodi personale e studenti è la tipologia dei server: dir.unibo.it contiene i **Global Catalog**, mentre gli altri due domini contengono i **Domain Controller**.

Il Global Catalog è un tipo particolare di Domain Controller che contiene una replica di tutti gli oggetti (account, gruppi, organizational unit) della foresta, a differenza dei Domain Controller che contengono solo gli oggetti del proprio dominio. Tuttavia gli oggetti presenti nel Global Catalog contengono un sottoinsieme delle proprietà dell’utente, definito **Partial Attribute Set**: le proprietà che non appartengono al Partial Attribute Set non sono replicate sui Global Catalog.

Il servizio IDP deve verificare sia le credenziali degli studenti che quelle del personale, quindi entrambe le operazioni effettuate dall’IDP (bind e query ldap) devono poter accedere agli account di entrambi i domini.

La documentazione online di Shibboleth contiene una pagina ad-hoc relativa alle caratteristiche di Active Directory e agli accorgimenti che bisogna seguire quando ci si collega a questo tipo di Directory Service (Beall, 2013), in particolare riguardo al collegamento ai Global Catalog:

“Se gli account sono divisi in domini diversi (es. CN=Staff,DC=example,DC=edu and CN=Faculty,DC=example,DC=edu) possono essere utilizzare le porte del Global Catalog:

3268 per LDAP di base o LDAPS con StartTLS;

3269 for LDAPS.”

Le prime prove sono quindi state effettuate impostando l'URL LDAP "ldap://dir.unibo.it:3268" sia nel file login.config (per il bind) sia nell'attribute-resolver.xml (per le query).

Le prime prove hanno dimostrato che il collegamento al Global Catalog non presentava particolari problemi, ma l'integrazione con le federazioni ha fatto emergere alcuni aspetti che non avevamo preso in considerazione e che verranno illustrati nel paragrafo seguente.

4.6.5 Integrazione con le federazioni

L'integrazione con le federazioni in ambiente di test ci ha consentito di provare le procedure (tecniche e organizzative) per l'ingresso nei Circle Of Trust e ci ha permesso di provare la procedura di login con i servizi di test di IDEM e FedERa.

La prima fase dell'ingresso in una federazione è la gestione organizzativa del processo: i responsabili e i referenti tecnici delle due parti firmano accordi che sanciscono la relazione di fiducia tra il fornitore del servizio e il gestore dell'identità. In questa fase ci sono stati scambi di contratti tra i responsabili delle federazioni e quelli unibo e numerose email tra i tecnici di IDEM e FedERa e il gruppo di lavoro sul Single Sign-On del Ce.S.I.A.

Il passo seguente è lo scambio dei metadati: l'Identity Provider scarica quelli del servizio di gateway (come nel caso di FedERa) o dell'aggregatore di metadati (IDEM) e i servizi configurano i metadati dell'IDP tra quelli *trusted*. Ho seguito questa fase in prima persona, inviando i metadati ai referenti tecnici delle federazioni e impostando e mantenendo aggiornata la configurazione dell'IDP unibo con i metadati di tutti i servizi federati.

Successivamente ci si accorda sugli attributi minimi richiesti per garantire l'accesso ad ogni singolo servizio: la lista dei dettagli dei servizi IDEM è disponibile alla pagina https://www.idem.garr.it/servizi/sp, mentre FedERa pubblica l'elenco completo di tutti i possibili attributi previsti dai servizi appartenenti alla federazione (http://federazione.lepida.it/documentazione/documentazione-tecnica/attributi) e

in seguito comunica il set di attributi specifici richiesti dai singoli servizi; in questa fase ho modificato i file di configurazione attribute-resolver.xml e attribute-filter.xml per la generazione e l'emissione degli attributi corretti, secondo il formato richiesto dalle specifiche tecniche.

Mentre i servizi di IDEM necessitano quasi sempre degli stessi attributi (id univoco opaco oppure nome, cognome, email) i servizi FedERa richiedono alcuni dati personali in più. In particolare il comune di Bologna, ad esempio, richiede:

- codice fiscale
- data di nascita
- nazione di nascita
- provincia di nascita
- luogo di nascita

Questi dati sono memorizzati nel Directory Service di Ateneo in tre attributi utente distinti:

- unibo-NationalPIN per il codice fiscale;
- unibo-Birthday per la data di nascita;
- unibo-Birthplace per nazione, provincia e luogo di nascita.

I primi test hanno subito evidenziato un problema: il servizio Shibboleth non riusciva a caricare questi tre attributi (mentre non c'erano problemi con gli altri). Dopo aver verificato che l'utente di servizio utilizzato per caricare gli attributi dal Directory Service avesse sufficienti diritti, è emerso che gli attributi in questione non facevano parte del **Partial Attribute Set**.

Come illustrato al paragrafo precedente, sia il bind che le query venivano effettuati verso i Global Catalog, che hanno il vantaggio di contenere tutti gli account utente del dominio (sia studenti che dipendenti) ma contengono al loro interno solo un subset di attributi, denominato Partial Attribute Set (PAS): i tre attributi unibo-NationalPIN, unibo-Birthday e unibo-Birthplace, per questioni di performance, non erano stati inclusi nel PAS.

Il passaggio successivo è quindi stato quello di effettuare sia il bind che le query verso i Domain Controller, includendo parametri di configurazione doppi sia nel file login.config che nell'attribute-resolver.aspx. In questo modo è possibile caricare tutti gli attributi degli utenti, con lo svantaggio di rendere i file di configurazione più verbosi e meno facili da mantenere aggiornati e, nel caso peggiore, di raddoppiare i temi di risposta (in quanto bind e query vengono effettuate due volte in caso l'utente non venga trovato nel dominio che compare prima nei file di configurazione).

Per ottimizzare almeno una delle due operazioni ho configurato l'IDP in modo che effettuasse il bind verso il Global Catalog (sufficiente per la fase di autenticazione) e le query verso i Domain Controller: in questo modo il processo di autenticazione impiega la metà del tempo rispetto alla soluzione precedente.

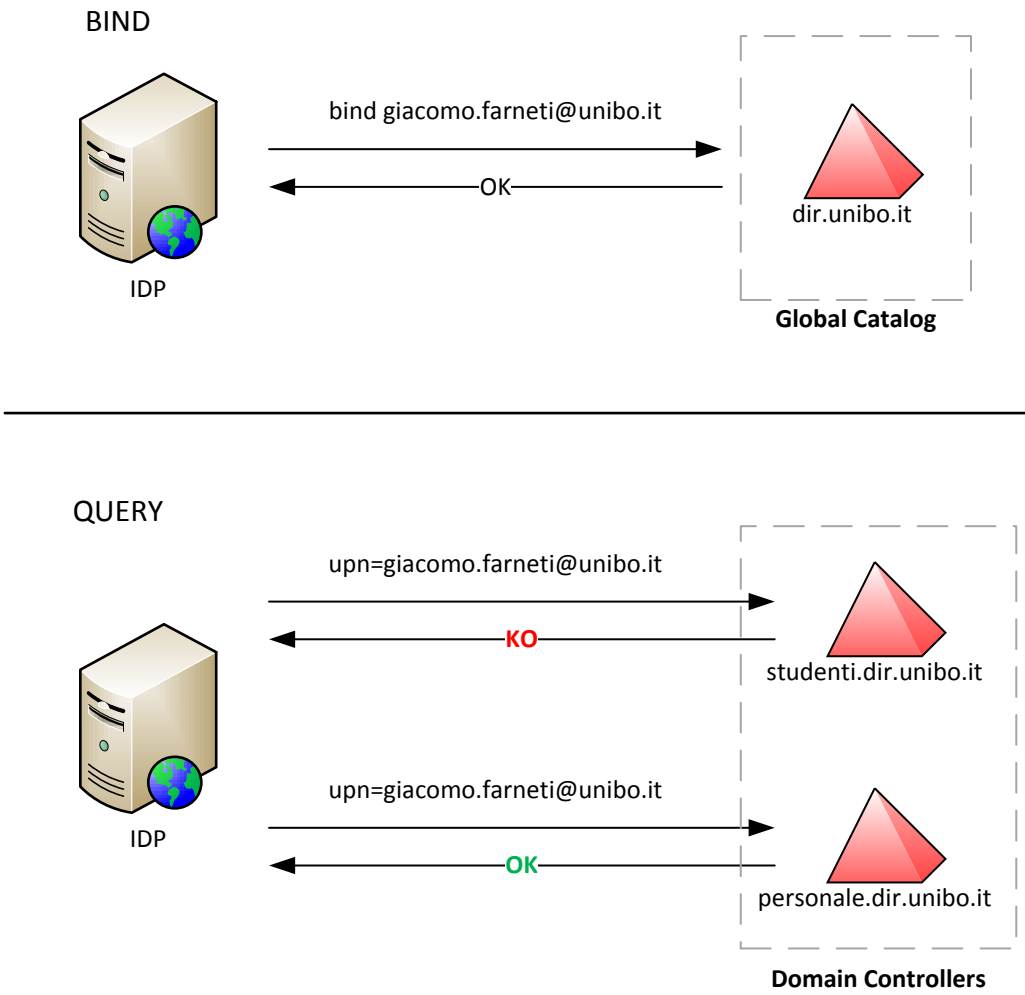


Figura 28: La soluzione adottata per Bind e Query LDAP

La configurazione potrà ulteriormente essere ottimizzata in futuro, includendo nel Partial Attribute Set anche gli attributi mancanti (eliminando quindi la necessità di effettuare le query su due domini distinti in favore di un'unica query verso il Global Catalog).

4.6.6 Passaggio in produzione

La procedura di installazione del servizio IDP in produzione non ha presentato problemi, in quanto tutte le procedure erano state già provate e documentate nell'ambiente di test. L'unica differenza sostanziale è stata la il **certificato SSL**: mentre in test abbiamo utilizzato un certificato autofirmato (shibtest.unibo.it), in produzione abbiamo richiesto ai colleghi dei sistemi un certificato valido per l'hostname shib.unibo.it.

Nel caso della federazione IDEM non è stato possibile utilizzare l'ambiente di test per effettuare le prove, in quanto i metadati dei servizi pubblicati sono relativi solo all'ambiente di produzione. Per IDEM quindi la configurazione è stata fatta direttamente sul server di produzione, ne è stato verificato il funzionamento con un Service Provider fornito dal GARR, poi sono stati integrati tutti i servizi uno ad uno.

Nel caso di FedERa è stato invece sufficiente adattare i file di configurazione ai parametri di produzione: metadati del gateway FedERa, parametri di collegamento al Directory Service di produzione, generazione ed emissione degli attributi utente.

Appena effettuato il passaggio in produzione, tuttavia, si è presentato un problema: alcuni utenti non riuscivano ad eseguire il login, mentre altri non avevano problemi ad accedere. I log hanno evidenziato un problema di collegamento a due Domain Controller (quelli presenti presso l'azienda ospedaliera, che ha una stretta collaborazione con l'università); ho verificato che risultava impossibile raggiungerli dal server di produzione, né tramite PING né tramite le porte LDAP/Global Catalog (rispettivamente 389 e 3268). Ho subito richiesto un intervento ai colleghi dell'ufficio reti che hanno aperto le porte necessarie sul firewall ed il problema è stato risolto.

Sull'ambiente di produzione ci siamo presto resi conto che non era possibile modificare direttamente i file di configurazione senza averli prima provati in test, ma non avendo a disposizione un ambiente di test per IDEM questo non era possibile. Abbiamo quindi trovato una soluzione creando l'**ambiente di sandbox**: un'installazione separata dell'IDP sul server di test con la configurazione di produzione, creata solo allo scopo di provare le modifiche prima di portarle in produzione. I test sono stati effettuati tramite uno script incluso nel pacchetto dell'IDP chiamato **Attribute Authority, Command Line Interface** (AACLI) che consente da riga di comando di testare l'accesso di un utente ad un determinato servizio.

Per testare il login del mio utente al gateway FedERa, ad esempio, è possibile utilizzare il comando:

```
./aacli.sh --configDir /opt/shibboleth-idp-prod-sandbox/conf/

--principal giacomo.farneti@unibo.it

--requester https://federa.lepida.it/gw/metadata
```

dove *configDir* è il percorso della cartella con i file di configurazione (questo esempio utilizza l'ambiente di sandbox), *principal* è il nome utente e *requester* è l'Entity ID del servizio; lo script stampa nella console il messaggio SAML con tutti gli attributi generati.

5 CONCLUSIONE E SVILUPPI FUTURI

L'Università di Bologna genera ogni anno decide di migliaia di credenziali per i nuovi iscritti, gestisce un totale di circa 300.000 account (tra studenti attivi e laureati) e migliaia di gruppi di autorizzazione. L'Identity Management è quindi uno dei servizi più importanti tra quelli gestiti dal C.e.S.I.A., è il motore che garantisce il corretto funzionamento di tutti i sistemi a collatere.

Il servizio di Identity Provider Shibboleth è stato un elemento indispensabile per estendere l'utilizzo di questa enorme mole di credenziali oltre i confini dell'ambito universitario, per favorire la collaborazione con il mondo della ricerca, della pubblica amministrazione e dei servizi ai cittadini.

L'obiettivo del progetto è stato raggiunto: abbiamo fornito uno strumento agli studenti e ai dipendenti dell'Università di Bologna che consente di fare login cone le proprie credenziali unibo a tutta una serie di servizi esterni senza doversi accreditare o registrare presso i fornitori di tali servizi. Ora è possibile, ad esempio, accedere ai servizi online del comune di Bologna, a Sebina On Line e alle librerie digitali più importanti (ScienceDirect, SpringerLink) con le proprie credenziali istituzionali in pochi semplici passaggi.

Il Single Sign-On fornisce il vantaggio ulteriore della condivisione della sessione di login tra più servizi web: se si è già autenticati tramite l'Identity Provider unibo l'accesso a ulteriori servizi non richiederà nuovamente l'inserimento delle proprie credenziali, riconoscendo in automatico l'utente e migliorando la sua esperienza di navigazione all'interno dell'ecosistema integrato con l'IDP unibo.

Il Single Sign-On federato ha come unico requisito l'utilizzo di un browser, un ulteriore vantaggio rispetto ai sistemi di autenticazione tradizionali che possono richiedere l'installazione di software aggiuntivi: l'accesso alle risorse elettroniche, ad esempio, prima del Single Sign-On si basava sull'analisi del range di appartenenza dell'indirizzo IP del computer dell'utente e richiedeva quindi o l'accesso dalla rete universitaria (AlmaNet) o l'installazione e la configurazione di un proxy sul proprio computer.

Il progetto di Single Sign-On non finisce con il lavoro svolto in questa tesi: in futuro saranno possibili integrazioni con altri sistemi, estensioni e miglioramenti al processo di login e aggiornamento dei sistemi per l'utilizzo delle ultime tecnologie introdotte nel campo dell'Identity Management.

Un argomento che è in corso di studio, ad esempio, è l'integrazione con gli smartphone: è attualmente attivo un tirocinio per studiare le possibilità offerte dal protocollo OAuth, lo standard de facto per il Single Sign-On nel mondo mobile, e per analizzare i diversi metodi di attivazione di tale protocollo sugli Identity Provider dell'Università.

In futuro si potrà valutare anche il supporto al login tramite Smart Card (ora in possesso di tutti i docenti dell'università), oppure l'interoperabilità tra il server IDP Shibboleth e quello Microsoft ADFS, che oggi sono indipendenti l'uno dall'altro e richiedono due login separati in caso si voglia accedere a servizi integrati nei diversi sistemi: le possibilità di espansione e di evoluzione del sistema sono molteplici, e potranno essere analizzati direttamente dal gruppo di lavoro sul Single Sign-On oppure con l'attivazione di tirocini e lavori di tesi.

Per concludere vorrei evidenziare come il lavoro svolto mi abbia portato ad approfondire molti aspetti tecnici che finora conoscevo solo superficialmente (ad esempio l'amministrazione di un server Linux o il funzionamento a basso livello di un sistema Active Directory) e a collaborare con un gruppo di persone competenti e disponibili che hanno reso possibile un progetto importante e complesso come quello del Single Sign-On.

RINGRAZIAMENTI

Questo lavoro di tesi è dedicato alla memoria di mia madre, una persona alla quale devo moltissimo, la prima a convincermi a continuare gli studi per iscrivermi alla laurea magistrale.

Un grazie di cuore a Eda per avermi sostenuto e per essere stata sempre presente, per avermi sopportato con tanta, tanta pazienza e per avermi spinto a non mollare.

Un ringraziamento speciale va a Benedetta per tutte le volte che è riuscita a rendere divertenti gli esami (spesso), per tutte le volte che mi ha fatto studiare quando non ne avevo voglia (spessissimo) e per aver reso unico questo viaggio.

BIBLIOGRAFIA

The LDAP Directory Service Model. (2012). Retrieved 2014, from MSDN: http://msdn.microsoft.com/en-us/library/windows/desktop/aa367023(v=vs.85).aspx

Baldoni, R. (2012). Federated Identity Management systems in e–government: the case of Italy. *Electronic Government, an International Journal, Volume 9*(Number 1/2012), 64-84.

Beall, E. (2013). *LDAP Server Issues.* Retrieved 2014, from Shibboleth Documentation: https://wiki.shibboleth.net/confluence/display/SHIB2/LdapServerIssues

Chong, F. (2004). Identity and Access Management. *The Architecture Journal*.

Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., et al. (1999, June). *HTTP Authentication: Basic and Digest Access Authentication*. Retrieved from http://tools.ietf.org/html/rfc2617

Hill, J. (2001). *An Analysis of the RADIUS Authentication Protocol*. Retrieved 2014, from http://www.untruth.org/~josh/security/radius/radius-auth.html

Howlett, J., Nordh, V., & Singer, W. (2010). *Deliverable DS3. 3.1: eduGAIN service definition and policy Initial Draft.* GÉANT.

Lewis, J. (2003). *Enterprise Identity Management: It's About the Business.* The Burton Group Directory and Security Strategies Report.

Recordon, D., & Reed, D. (2006). OpenID 2.0: A Platform for User-centric Identity Management. In *Proceedings of the Second ACM Workshop on Digital Identity Management* (pp. 11--16). ACM.

Zanmarchi, S. (2010, November 9). *Introduzione alle Infrastrutture di Autenticazione e Autorizzazione.* Retrieved 2014, from Servizio IDEM AAI: https://www.idem.garr.it/documenti/doc_details/170-introduzione-alle-infrastrutture-di-autenticazione-e-autorizzazione