SPID GEL (Gateway Enti Locali)

Guida per l'integrazione a SPID/CIE tramite GEL

STORICO REVISIONI

Data	Revisione	Descrizione	Verificato	Approvato
05/03/2018	01	Prima stesura	DG 5010 Staff 92	05/03/2018
09/04/2018	02	Aggiunte istruzioni su creazione certificato self-signed	DG 5010 Staff 92	09/04/2018
20/04/2018	03	Modificata AuthnContextClassRef nella componente Shibboleth	DG 5010 Staff 92	20/04/2018
23/04/2018	04	Logo e intestazione Regione Campania	DG 5010 Staff 92	23/04/2018
15/07/2021	05	Variazione competenza uffici regionali Aggiunta descrizione processo di adesione ed esempi di integrazione di applicazioni	UOD 60 11 02	15/07/2021
19/10/2021	06	Aggiunta segnalazione su configurazioni in ambienti PHP	UOD 60 11 02	19/10/2021
03/03/2022	07	Aggiornamento titolo documento e correzione riferimenti	UOD 60 11 02	03/03/2022
05/04/2022	08	Integrazione informazioni in ambito PHP per limitazioni headers contenenti "_".	UOD 60 11 02	05/04/2022

DIFFUSIONE E RISERVATEZZA DEI CONTENUTI

leggi vigenti. Regione Campania non si assume alcuna responsabilità per l'uso del materiale contenuto nel presente documento.

INDICE DEGLI ARGOMENTI

1. Introduzione

1.1 Scopo e campo di applicazione

Questo documento ha lo scopo di introdurre il componente denominato Gateway Enti Locali (GEL) messo a disposizione da Regione Campania e di definirne le modalità di integrazione da parte di applicazioni che hanno la necessità di abilitare l'autenticazione mediante SPID.

Tale componente regola la fase di autenticazione degli utenti (cittadini) che, utilizzando un web browser, richiedono i servizi online - offerti dalla Regione Campania stessa o da Enti Locali della regione Campania - autenticandosi tramite il Sistema Pubblico per l'Identità Digitale (SPID).

Nel seguito, dopo una breve presentazione dell'architettura del sistema, verranno forniti i dettagli sulle interfacce fornite e richieste da parte di GEL nei confronti dei fornitori di servizi esterni con i quali avverrà lo scambio di informazioni, al fine di gestire il processo di autenticazione degli utenti.

1.2 Riferimenti

Nel presente documento si fa riferimento alla seguente documentazione:

[1] OASIS – "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0" - OASIS Standard, 15March 2015

[2] OASIS – "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0" - OASIS Standard, 15March 2015

[3] Documentazione ufficiale SPID: https://www.spid.gov.it/ e https://www.agid.gov.it/it/piattaforme/spid

1.3 Acronimi e Definizioni

Nel corso del documento sono introdotti e utilizzati i seguenti acronimi:

	Definizione
AgID	Agenzia per l'Italia Digitale
CA	Certification Authority
CAD	Codice dell'Amministrazione Digitale
CN	Common Name
CNS	Carta Nazionale dei Servizi
CRS	Carta Regionale dei Servizi
EELL	Enti Locali
GEL	Gateway Enti Locali
HTTP	Hyper Text Transfer Protocol
IdP	Identity Provider
IdPC	Identity Provider del Cittadino
OTP	One Time Password
PEO	Posta Elettronica Ordinaria
RC	Regione Campania
SAML	Security Assertion Markup Language
Shibboleth	OpenSource Shibboleth-ServiceProvider (https://shibboleth.net/)
SP	Service Provider
SPID	Sistema Pubblico per l'Identità Digitale
SSL	Secure Socket Layer
TLS	Transport Layer Security
TS-CNS	Tessera Sanitaria – Carta Nazionale dei Servizi
XML	Extensible Markup Language

2. Introduzione al servizio GEL

Al fine di supportare l'adesione a SPID degli applicativi web destinati ai cittadini ed erogati dalla Regione Campania e dagli Enti Locali della regione Campania, è stato reso disponibile il servizio denominato GEL (Gateway Enti Locali), preso a riuso dalla Regione Lombardia, che è messo a disposizione gratuitamente in modalità SaaS (Software as a Service).

Il servizio GEL è progettato in architettura "multi-tenant" ovvero in modo che sia possibile creare istanze separate per ogni singolo Ente Locale.

Ogni Ente Locale[1] che intenda avvalersi del servizio GEL sarà quindi autonomo nella possibilità di configurare la propria istanza avvalendosi comunque delle componenti di base comune a tutte le istanze e potendo contare sull'impegno di Regione Campania a adeguare il servizio GEL ad ogni modifica delle regole tecniche emanate da AgID.

La soluzione adottata è descritta sinteticamente di seguito:

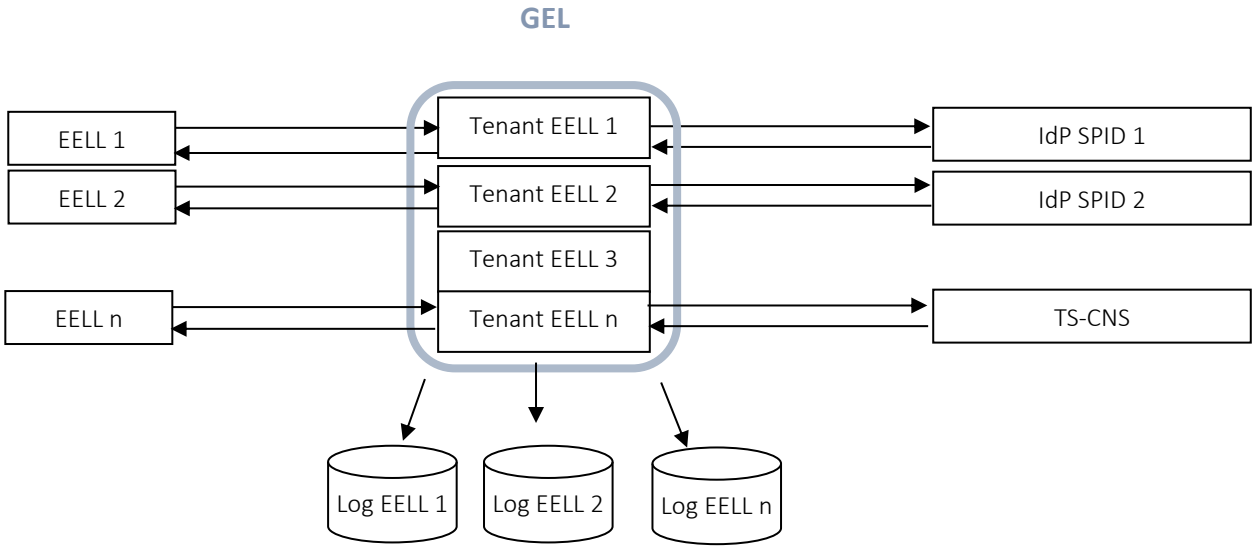

Figura 1 Architettura di alto livello sistema GEL

Il servizio GEL metterà a disposizione degli enti aderenti una console di gestione con la quale potranno essere svolte tutte le attività di configurazione necessarie. Tramite la stessa console ogni ente potrà prelevare il proprio LOG, previsto dal regolamento attuativo di SPID, per conservarlo attraverso i propri processi di conservazione[2].

Per facilitare le attività di integrazione esiste un ambiente di test, agganciato agli IdP SPID di test e fruibile su Internet.

[1] Per gli uffici regionali, come meglio descritto nel seguito, verrà creato un tenant dedicato di tipo "non Ente Locale", gestito direttamente dalla UOD 60 11 02 - Servizi ed ecosistemi digitali previa sottoscrizione di una apposita liberatoria

[2] Per gli uffici regionali la console sarà gestita direttamente dalla UOD 60 11 02.

3. Dettagli tecnici

Questa sezione descrive le operazioni lato SP (applicativo da integrare con SPID) necessarie per la fruizione degli IdP SPID attraverso il servizio GEL.

L'elemento cardine dell'integrazione è l'avvenuta installazione e configurazione presso il SP di una istanza del prodotto **Shibboleth** (https://wiki.shibboleth.net/confluence/display/SP3/Home)[3]. Si noti che la suite Shibboleth è composta da diversi moduli: per questa specifica integrazione, è necessario e sufficiente il solo modulo "service provider".

Per eventuali approfondimenti su caratteristiche e configurazioni base di questo prodotto, è possibile consultare il capitolo 6 e successivi.

3.1 Kit di integrazione al servizio GEL

Per semplificare il processo di integrazione Regione Campania fornisce un "**kit di integrazione**" composto da:

- ✓ il presente documento ;
- ✓ un file (*idp_metadata.xml*) da copiare sulla istanza di Shibboleth SP, in accordo alle istruzioni incluse nella sezione 3.4; questo file regola l'ingaggio con l'ambiente GEL di Produzione;
- ✓ un file (*idp_metadata_test.xml*) da copiare sulla istanza di Shibboleth SP, in accordo alle istruzioni incluse nella sezione 3.4; questo file regola l'ingaggio con l'ambiente GEL di Test;
- ✓ un file (*attribute_map.xml*) da copiare sulla istanza di Shibboleth SP, in accordo alle istruzioni incluse nella sezione 3.5; questo file regola il rilascio degli attributi utente;
- ✓ una utenza SPID di test afferente ad un IdP accreditato da AgID;
- ✓ un tool per la decifratura del file di log e per la firma dei metadati del SP.

3.2 Chiavi crittografiche

Ogni SP deve avere a disposizione[4], una coppia di chiavi crittografiche personalizzate, che sono utilizzate per molteplici finalità:

- garantiscono la firma delle richieste di autenticazione verso GEL ;
- nella loro componente pubblica, sono utilizzate per la cifratura dei log di pertinenza del SP;
- sono utilizzate per la firma del metadata specifico del SP, che va indirizzato ad AgID nell'ambito del processo di abilitazione dell'Ente.

[3] E' possibile utilizzare anche applicazioni diverse dalla componente Shibboleth SP, purchè conformi allo standard SAML e che implementano le funzionalità previste per un SP SAML. Tuttavia, il presente documento fa riferimento all'utilizzo solo di tale componente, ampiamente testato per l'integrazione con GEL. Pertanto, a meno di esigenze particolari, si consiglia l'utilizzo di tale modulo ai fini dell'integrazione con GEL.Tale modulo è di tipo Open source, con licenza Apache 2.0, e disponibile gratuitamente.

[4] Per gli uffici regionali la generazione e la messa a disposizione avviene a cura d UOD 60 11 02.

In linea generale un certificato SSL dovrebbe essere ottenuto da una "Certification Authority" riconosciuta ed abilitata, è possibile però creare in autonomia ed in maniera del tutto gratuita tale certificato, che si dice in tal caso "self-signed".
Per creare una coppia chiave privata/certificato SSL esistono diversi strumenti, uno dei più diffusi è il tool "OpenSSL[5]", standard de-facto per tale scopo sui sistemi Linux e non solo.
Un esempio di comando per generare tale coppia di file potrebbe essere il seguente:

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout ssl.key -out ssl.crt
```

In particolare si notino i parametri:

- **-nodes**: salva la chiave privata senza proteggerla con una passphrase. Nonostante non sia considerata una pratica corretta è molto diffusa dal momento che i servizi che usano la chiave con passphrase non possono essere riavviati automaticamente senza immetterla manualmente.
- **-days**: è la durata di validità in giorni del certificato.
- **rsa:xxxx**: specifica la lunghezza in bit (e dunque la sicurezza) della chiave privata. Nell'esempio viene creata una chiave a 2048 bit.

L'applicazione richiede l'immissione di alcuni valori:

```
Country Name (2 letter code) [XX]:IT
State or Province Name (full name) []:Italia
Locality Name (eg, city) [Default City]:Napoli
Organization Name (eg, company) [Default Company Ltd]:Ente Locale X
Organizational Unit Name (eg, section) []:Ente locale X
Common Name (eg, your name or your server's hostname) []:gel-entelocalex
Email Address []:
```

La compilazione della maggior parte dei parametri richiesti è a scelta dell'utente, è molto importante però fare attenzione al valore immesso per il campo "**Common Name**" poiché sarà referenziato nei file di configurazione di shibboleth.
Può essere utile conservare la coppia chiave/certificato in un file in formato PKCS12, il comando messo a disposizione da OpenSSL è il seguente:

```
openssl pkcs12 -export -in ssl.crt -inkey ssl.key -out ssl.p12 -name "gel-entelocalex"
```

Il comando prende come parametri di input il file chiave privata, il file certificato, una etichetta come alias ed il nome del file di tipo PKCS12 da creare. Verrà richiesta una password per poter accedere successivamente all'archivio.

Si noti che in ambiente di produzione la generazione delle chiavi crittografiche deve rispettare le specifiche tecniche dettate da AGID relative all'avviso n.29 v3[6]

[5] https://www.openssl.org/

[6] https://www.agid.gov.it/it/piattaforme/spid/avvisi-spid Per gli EE.LL. è possibile utilizzare il tool messo a disposizione Developers Italia al seguente indirizzo: https://github.com/italia/spid-compliant-certificates

3.3 Set di dati richiesti dal SP

Il SP dovrà scegliere quali dati di identità dell'utente intende richiedere all'IdP SPID che governa il processo di autenticazione. Si noti che questi dati saranno un subset rispetto a quelli rilasciati dal servizio GEL ed illustrati nella sezione 3.5. Le possibili combinazioni di dati sono aggregate in "set" autoconsistenti, tra cui il SP dovrà scegliere. Nella sezione 3.5 sono anche evidenziati i campi *obbligatoriamente* restituiti dagli IdP e quelli *opzionalmente* restituiti.
I "set" ad oggi previsti sono i seguenti:

- Set 0: nome, cognome, codice fiscale, email, codice identificativo SPID
- Set 1: nome, cognome, codice fiscale, email, codice identificativo SPID, sesso, data di nascita, luogo di nascita
- Set 2: nome, cognome, codice fiscale, email, codice identificativo SPID, sesso, data di nascita, luogo di nascita, provincia di nascita
- Set 3: nome, cognome, codice fiscale, email, codice identificativo SPID, sesso, data di nascita, luogo di nascita, provincia di nascita, telefono mobile
- Set 4: nome, cognome, codice fiscale, codice identificativo SPID
- Set 5: nome, cognome, codice fiscale, codice identificativo SPID, ragioneSociale, sedeLegale, partitaIVA

Si ricorda quanto previsto dal "Regolamento recante le modalità attuative per la realizzazione dello SPID":

> ***Art. 27 (Uso degli attributi SPID)***
> *I fornitori di servizi, per verificare le policy di sicurezza relativi all'accesso ai servizi da essi erogati potrebbero avere necessità di informazioni relative ad attributi riferibili ai soggetti richiedenti. Tali policy dovranno essere concepite in modo da richiedere per la verifica il* ***set minimo di attributi pertinenti e non eccedenti*** *le necessità effettive del servizio offerto e mantenuti per il tempo strettamente necessario alla verifica stessa, come previsto dall'articolo 11 del decreto legislativo n. 196 del 2003.*

Si consiglia pertanto di scegliere il Set 4, salvo motivate necessità del servizio.

L'indicazione del set scelto andrà inserita nella richiesta di autenticazione creata dalla componente Shibboleth (vedi sezione successiva).

3.4 Configurazione della componente Shibboleth

L'istanza di Shibboleth utilizzata dal SP va corredata da alcune informazioni peculiari che consentono l'integrazione al servizio GEL.

Il file *shibboleth2.xml* deve contenere le personalizzazioni descritte nel seguito. Alcuni tag vanno valorizzati in accordo alla struttura dell'applicazione, in questi casi è presente una nota a piè pagina esplicativa. Gli altri valori vanno inseriti senza modificarli. Si noti in particolare che Shibboleth viene istruito per lavorare in **SAML2**.

Si noti che, come previsto dall'infrastruttura di autenticazione SAML2, entrambe le controparti (SP e IdP) devono essere dotate di un "metadata", ed ogni entità deve conoscere il "metadata" della controparte.

La configurazione della componente Shibboleth illustrata in questo paragrafo consente, tra le altre cose, al SP di installare sui propri sistemi il "metadata" dell'IdP (il servizio GEL).

L'operazione speculare, ovvero l'installazione sull'IdP del "metadata" del SP, che è *specifico di ogni EELL*, avviene con procedure automatizzate e mediate dalla console di gestione menzionata nella sezione 2.

```
...
<RequestMapper type="Native">
  <RequestMap applicationId="default">
    <Host name="HOST_NAME" [7] scheme="https" port="443">
      <Path name="URL_PROTETTO" [8] applicationId="ID_APP" [9] authType="shibboleth"
                                requireSession="true" exportAssertion="true"/>
    </Host>
  </RequestMap>
</RequestMapper>
...
<ApplicationOverride id="ID_APP[10]" policyId="default" entityID="ENTITYID" [11]
REMOTE_USER="cf"
                                signing="true" encryption="false">
  <Sessions lifetime="28800" timeout="3600" checkAddress="false"
            handlerURL="URL_PROTETTO/Shibboleth.sso[12]" handlerSSL="true"
            exportLocation="/GetAssertion" exportACL="127.0.0.1" idpHistory="false"
            idpHistoryDays="7" cookieProps="; path=/; secure; HttpOnly">
    <SessionInitiator type="SAML2" Location="/Login"
                          entityID="https://spidgateway.regione.campania.it/idpcgel" >
      <samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="idpcgel"
        Version="2.0" IssueInstant="2012-01-01T00:00:00Z"
```

7 Personalizzare con hostname del servizio
8 Personalizzare con la URL (o sezione del sito) da proteggere con autenticazione
9 Identificativo dell'applicazione (tipicamente una stringa autoesplicativa)
10 Vedi nota 9; si noti che la URL dell'AssertionConsumer del singolo SP è prodotta automaticamente da Shibboleth ed inserita nella AuthRequest
11 Rappresenta il codice identificativo del SP. Il formato più comune è del tipo "http://spidgateway.regione.campania.it/gelmetadata/**COD_ISTAT**". Personalizzare la URL indicando il codice ISTAT dell'Ente Locale.
12 Path relativo, a partire dalla context root dell'applicazione, dove è situata la risorsa da proteggere (es. /my_site/protected/Shibboleth.sso)

```
AttributeConsumingServiceIndex="4" [13]>
        <samlp:NameIDPolicy AllowCreate="true"
                                Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:transient"/>
        <samlp:RequestedAuthnContext Comparison="exact"
                                    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
          <saml:AuthnContextClassRef xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
               https://www.spid.gov.it/SpidL1 oppure
               https://www.spid.gov.it/SpidL2 [14]
          </saml:AuthnContextClassRef>
        </samlp:RequestedAuthnContext>
        <samlp:Scoping ProxyCount="1"></samlp:Scoping>
      </samlp:AuthnRequest>
    </SessionInitiator>
  </Sessions>
 <AttributeExtractor type="XML" validate="true" reloadChanges="false" path="
PATH_ATTRS_MAP[15]"/>
  <CredentialResolver type="File">
    <Key password="PASSWORD_P12" format="PKCS12">[16]
      <Path>PATH_P12</Path>
      <Name>GEL-regione.campania.it</Name>
    </Key>
    <Certificate password="PASSWORD_P12" format="PKCS12">
      <Path>PATH_P12</Path>
    </Certificate>
  </CredentialResolver>
  <MetadataProvider type="Chaining">
    <MetadataProvider type="XML" file="PATH_METADATA_GEL"/>[17]
  </MetadataProvider>
</ApplicationOverride>
...
```

13 Indice che referenzia la sezione del metadata della componente "GEL" dove è contenuto il set di valori di interesse dell'applicazione – si veda paragrafo 3.3

14 Specificare il livello di autenticazione SPID richiesto (può essere L1 o L2 - utilizzare esclusivamente una delle due stringhe indicate)

15 Path del file attributes-map.xml

16 Il SP deve possedere una coppia di chiavi, necessarie per la firma delle richieste di autenticazione. In questa sezione andrà configurato il path assoluto del file (.p12) che le contiene, corredato della relativa password.

17 Qui si fa riferimento al metadata della componente GEL ed il file è presente nel kit di integrazione fornito. Questo file, denominato "idp_metadata.xml" (per l'ambiente di produzione), va referenziato in path assoluto. Utilizzare invece il file "idp_metadata_test.xml" per l'ingaggio dell'ambiente di test

3.5 Asserzione di identità rilasciata al SP

L'asserzione di identità rilasciata dalla componente GEL viene "mappata" in *header http* attraverso una operazione di configurazione della componente Shibboleth. Si sottolinea che alcune valorizzazioni dei tag sono peculiari nel caso in cui l'asserzione sia nativamente emessa da un IdP SPID:

Campo asserzione IdPC-GEL	Campo asserzione SPID	Note
Dati obbligatori e tipicamente valorizzati		
Nome	Name	
Cognome	familyName	
codice Fiscale	fiscalNumber	Privato del preambolo "TINIT:"
Sesso	Gender	
data Nascita	dateOfBirth	
luogo Nascita	placeOfBirth	Previa transcodifica Belfiore
provincia Nascita	countryOfBirth	
stato Nascita	-	calcolato da codiceFiscale
tipo Autenticazione	*derivato da AuthnContext, solo se status=Success*	Può valere uno dei seguenti: IDPC_AUTHENTICATION_SMARTCARD, IDPC_SPID_L2, IDPC_SPID_L3
identificativo Utente	spidCode	
nome Utente	spidCode	duplicato per compatibilità
livello Verifica	*derivato da AuthnContext, solo se status=Success*	Inserito per compatibilità con integrazioni già in essere. Può valere uno dei seguenti: 50 (se L1), 70 (se L2), 90 (se L3)
Dati tipicamente opzionali		
emailAddress	email	
Cellular	mobilePhone	
ragione Sociale	companyName	
sede Legale	registeredOffice	
partitaIVA	ivaCode	Si noti che gli IdP SPID possono gestire identità di persone giuridiche. I SP interessati devono quindi prevedere questi casi e predisporsi per gestirle
docIdentita	idCard	
scadDocIdentita	expirationDate	
domicilio Fisico	address	
domicilioDigitale	digitalAddress	
CNS_CARTA_REALE	-	Sempre NON_DISPONIBILE
CNS_ISSUER	-	Sempre NON_DISPONIBILE
CNS_SUBJECT	-	Sempre NON_DISPONIBILE
origineDatiUtente	-	Sempre SPID
statoValidazioneProfiloUtente	-	Sempre NON_DISPONIBILE
idComuneRegistrazione	-	Sempre 03
ora_ultimo_login	-	Disponibile a partire da IdPC v10.1.00 - indica l'ora (in formato hh:mm) della precedente autenticazione utente - dato opzionale, assente se l'utente si collega per la prima volta ad IdPC
data_ultimo_login	-	Disponibile a partire da IdPC v10.1.00 - indica la data (in formato dd-mm-yyyy)

		della precedente autenticazione utente - dato opzionale, assente se l'utente si collega per la prima volta ad IdPC
tipoAutenticazione_ultimo_login	-	Disponibile a partire da IdPC v10.1.00 - indica la modalità di autenticazione scelta dall'utente nel corso della precedente autenticazione; può valere una delle seguenti: IDPC_AUTHENTICATION_SMARTCARD, IDPC_AUTHENTICATION_OTP_SISS, IDPC_SPID_L2, IDPC_SPID_L3
idp_ultimo_login	-	Disponibile a partire da IdPC v10.1.00 - nel caso di ultima autenticazione avvenuta tramite SPID, specifica l'IdP utilizzato dall'utente
indirizzo_IP_precedente	-	Disponibile a partire da IdPC v10.1.00 - IP address della postazione utilizzata dall'utente per l'ultima autenticazione
indirizzo_IP_corrente	-	Disponibile a partire da IdPC v10.1.00 - IP address della postazione utilizzata dall'utente per l'autenticazione corrente

Al fine di facilitare la configurazione delle istanze di Shibboleth, all'interno del Kit di integrazione è fornito ai SP il file *attribute-map.xml* che consente la copia dei dati di asserzione nei http header.

4. Il processo di adesione a GEL

Di seguito viene illustrato il processo di adesione a GEL, differenziato a seconda che il richiedente sia un ufficio della regione stessa o un Ente Locale (in quest'ultimo caso, gli aspetti amministrativi saranno trattati in un documento separato).

4.1 Processo di adesione per Enti Locali

Il processo di adesione è descritto in Figura 2:

1. *Richiesta avvio integrazione*: il richiedente invia una richiesta al US11 di utilizzo della piattaforma GEL in ambiente di collaudo ed esercizio, fornendo le seguenti informazioni:
    a. Denominazione EELL
    b. Codice ISTAT
    c. Codice Fiscale del Responsabile
    d. Cognome Nome del Responsabile
    e. Email del Responsabile
    f. Telefono mobile del Responsabile

2. US11, a valle del processo autorizzativo qui non descritto, autorizza la creazione del Tenant e procede alla creazione dello stesso sulla piattaforma GEL, inviando la documentazione disponibile, il kit di integrazione e le credenziali di Responsabile assegnate all'Ente
3. L'Ente Locale procede in autonomia alle configurazioni di propria competenza:
    a. Assegnazioni altri responsabili per il proprio tenant
    b. Produzione/acquisizione chiavi crittografiche
    c. Caricamento certificato e produzione metadata sulla console GEL (creazione metadata, download, firma e upload metadata firmato con le chiavi crittografiche in proprio possesso)
    d. comunica ad AgID i metadata per la verifica e la propagazione agli IdP
    e. configura Shibboleth con parametri della propria istanza GEL
    f. configura i servizi applicativi per il recupero delle informazioni utente
    g. effettua i test di integrazione (con l'eventuale supporto di US 11 in caso di problemi)

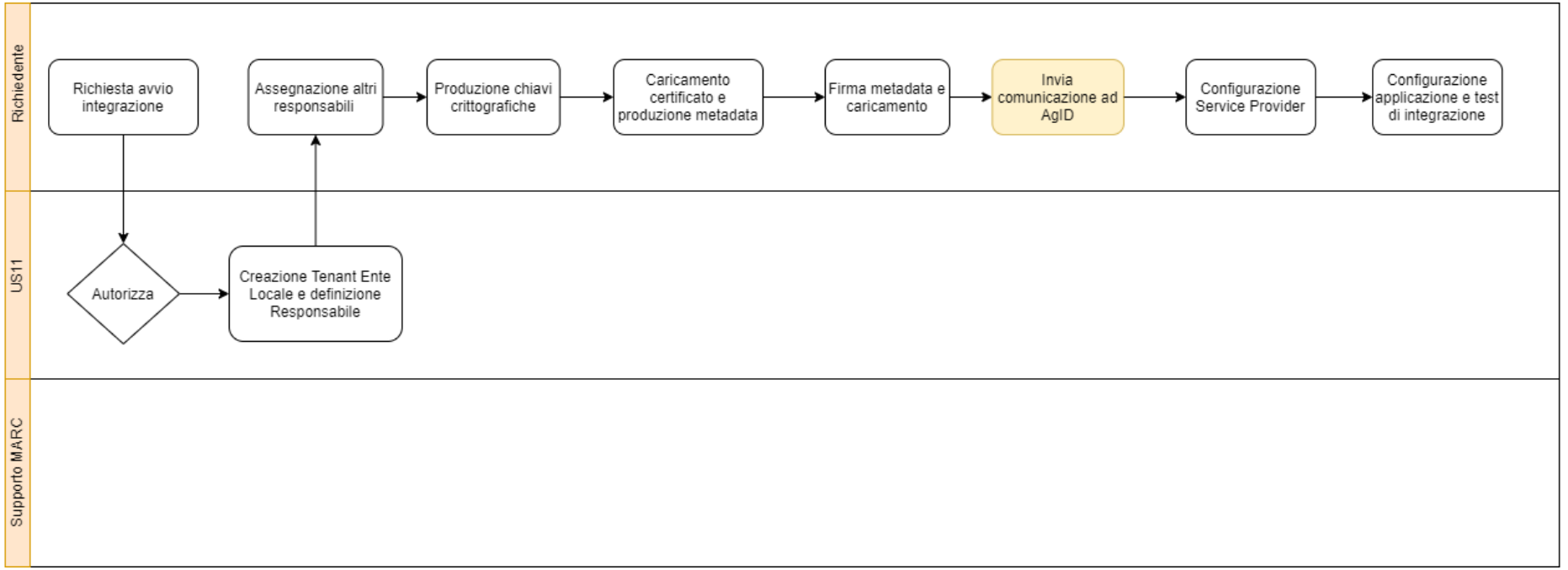

Figura 2 - Processo di adesione per EELL

4.2 Processo di adesione per altri attori (non Enti Locali)

Nel caso in cui strutture della Regione Campania abbiano la necessitò di integrarsi con GEL, il processo da seguire è descritto in Figura 3:

1. *Richiesta avvio integrazione*: il richiedente invia una richiesta al US11 di utilizzo della piattaforma GEL in ambiente di collaudo ed esercizio, fornendo le seguenti informazioni:
    a. Denominazione Applicativo
    b. Riferimento tecnico per l'integrazione
    c. Specifica se l'applicativo è ospitato presso il CRED
2. US11, a valle del processo autorizzativo qui non descritto, autorizza la creazione del Tenant e procede alla creazione dello stesso sulla piattaforma GEL, inviando la documentazione disponibile ed il kit di integrazione al riferimento indicato al punto precedente
3. US 11 produce le chiavi crittografiche necessarie e, nel caso in cui l'applicativo non sia ospitato presso il CRED e, per il solo ambiente di produzione, comunica l'Hash del certificato di produzione (certificato da configurare successivamente a carico del richiedente nello Shibboleth SP installato e configurato a cura del richiedente, come descritto in figura) da indicare nella liberatoria necessaria all'utilizzo del sistema
4. US 11 carica il certificato nel tenant creato per il richiedente e si occupa anche della produzione del metadata firmato digitalmente. Successivamente:
    a. Se l'applicativo è ospitato presso il CRED: US 11 si occupa delle attività di configurazione dello Shibboleth SP installato presso il CRED
    b. Se l'applicativo non è ospitato presso il CRED:
        i. US 11 trasmette le chiavi crittografiche al richiedente (a valle della sottoscrizione della liberatoria all'utilizzo di tali chiavi crittogarfiche)
        ii. Il richiedente configura il proprio Shibboleth SP con le chiavi ricevute
4. Il richiedente effettua i test di integrazione (con l'eventuale supporto di US 11 in caso di problemi)

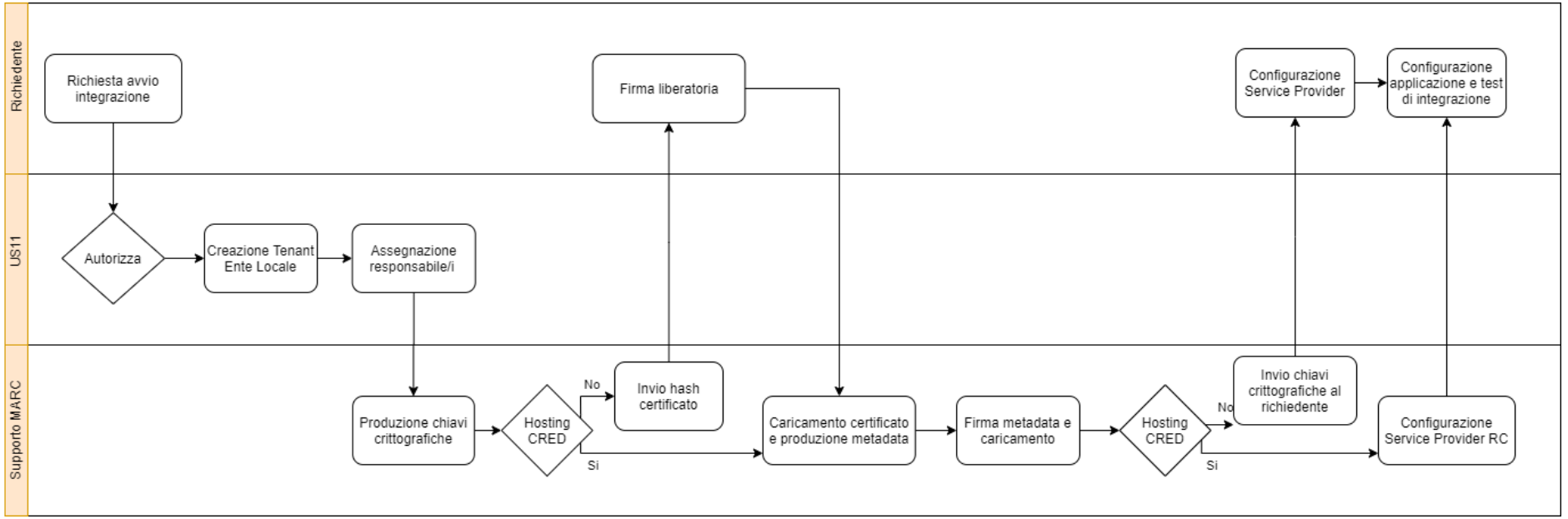

Figura 3 - Processo di adesione per altri attori

5. Istanza e procedure di test del GEL

5.1 Piattaforme di test

Al fine di permettere l'esecuzione di test in fase di sviluppo e pre-esercizio, è messa a disposizione degli uffici regionali e degli EELL della Regione Campania una istanza di test di GEL già configurata allo scopo.

L'istanza di test, funzionalmente equivalente, permette di testare l'integrazione di uno o più servizi con le istanze di test degli idP SPID che le hanno messe a disposizione.

È importante notare che, trattandosi di circuiti di test e non di circuiti reali, non è possibile utilizzare identità di cittadini reali, bensì nei circuiti di test è possibile unicamente utilizzare identità SPID di test fornite dagli IdP SPID.

Regione Campania è in possesso di un certo numero di "credenziali SPID di test" che può distribuire agli Enti. La soluzione adottata è descritta sinteticamente di seguito:

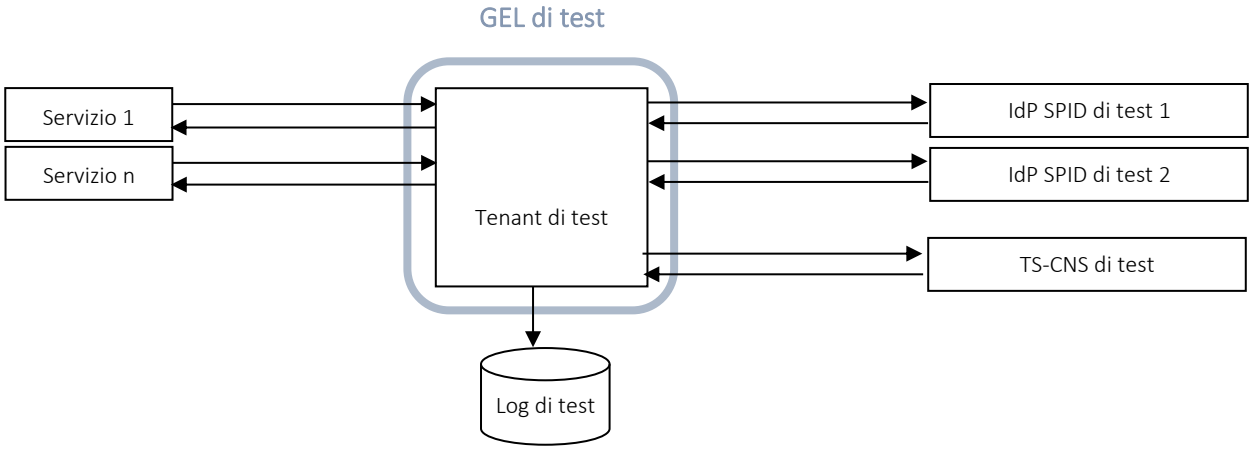

Figura 4 Architettura ambiente GEL di test

L'istanza di test del GEL è resa disponibile su Internet ad una URL che verrà comunicata agli Enti interessati.

6. Architettura Shibboleth SP

L'infrastruttura di gestione degli accessi a Service Provider realizzata mediante l'applicativo Shibboleth SP presenta un'architettura costituita da due sotto-sistemi principali, dei quali il primo è un web server di tipo Apache HTTP Server in versione 2.2.12 o successiva (che funge da reverse-proxy) e il secondo è un "demone" Shibboleth SP, anch'esso in versione 3.1 o successiva[18]. Tali sotto-sistemi comunicano tra loro attraverso uno specifico modulo di Apache denominato "**mod_shib**", in grado di intercettare l'accesso ad una o più risorse (URL) protette e attivare i componenti del demone Shibboleth SP per iniziare il processo di autenticazione. Nella figura successiva è illustrata l'architettura complessiva.

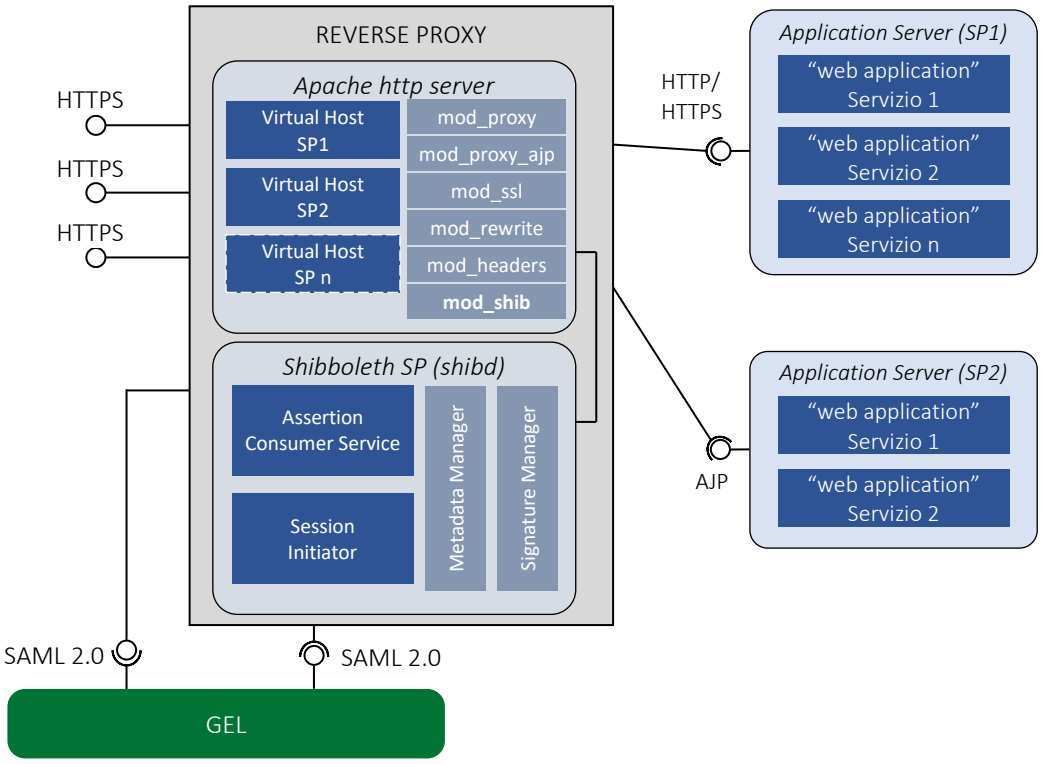

Figura 5 Architettura del modulo Shibboleth SP e interazioni con GEL

Il Reverse Proxy indicato in figura si caratterizza come entità a sé stante che può essere dispiegata indipendentemente dagli applicativi da "proteggere" e dai sistemi di autenticazione che si troverà ad utilizzare. In questo modo il Reverse Proxy può essere dispiegato in un contesto di DMZ in modo da essere accessibile da parte dell'utenza che deve accedere agli applicativi. Questi ultimi possono invece trovarsi in una rete interna, protetta. Attraverso questa architettura, infatti, gli applicativi che sarebbero normalmente accessibili direttamente perché espongono un'interfaccia di tipo HTTP o

[18] Il server Shibboleth SP è in grado di integrarsi anche con il server web Microsoft Internet Information Services (IIS). La trattazione tuttavia farà riferimento al caso dell'integrazione con Apache HTTP Server.

quelli accessibili su protocollo AJP mediante l'aggiunta di un semplice web server Apache con utilizzo del modulo "mod_proxy_ajp" vengono schermati dalla presenza del web server Apache dispiegato presso il Reverse Proxy, al quale pertanto è importante che giungano tutte le richieste di servizio.

L'accesso diretto agli applicativi retrostanti deve quindi essere impedito e gli indirizzi originariamente utilizzati per l'accesso devono essere rimappati in modo che corrispondano ad endpoint configurati presso il web server Apache del Reverse Proxy, secondo le modalità che saranno illustrate nel seguito.

Il server Shibboleth SP rappresentato in figura è in grado di operare utilizzando il protocollo SAML 2.0, adattandosi così ad interagire con Identity Provider quali **il sistema GEL di Regione Campania**.

Nel seguito del capitolo saranno descritti in maggiore dettaglio i sottosistemi relativi al web server Apache e al server Shibboleth SP.

6.1 Apache HTTP Server

Allo scopo di gestire l'accesso ai vari applicativi che si intende proteggere tramite Reverse Proxy, il web server Apache HTTP Server deve essere dotato di alcuni moduli aggiuntivi, le cui funzioni sono illustrate di seguito:

- ***mod_proxy***: consente di realizzare la funzione di reverse proxy esponendo gli endpoint URL necessari per la fruizione dei vari servizi applicativi e rimappandoli in modo che siano indirizzati agli specifici Application Server che ospitano i vari Service Provider;
- ***mod_proxy_ajp***: completa le funzionalità del modulo mod_proxy, consentendo di inoltrare le richieste pervenute ad Apache su protocollo HTTP o HTTPS utilizzando il protocollo AJP verso gli Application Server di destinazione;
- ***mod_shib***: consente al web server di proteggere le risorse (URL) predefiniti ed attivare i componenti del demone Shibboleth SP per attivare il processo di autenticazione;
- ***mod_ssl***: utilizzato per mettere a disposizione dei client degli end-point HTTPS;
- ***mod_rewrite***: utilizzato per effettuare degli adattamenti sui messaggi SAML generati dal demone Shibboleth quando è necessario interagire con Identity Provider conformi alle specifiche SAML.

Il web server inoltre gestisce l'accesso agli applicativi in modo isolato, l'uno rispetto all'altro, mediante l'impostazione di altrettanti Virtual Host. In tale modo è possibile delimitare e rendere indipendente la configurazione dedicata a gestire gli accessi verso un erogatore di servizi da quelle destinate ai rimanenti ed offrire ad esempio l'accesso in HTTP per uno e in HTTPS per un altro. In questo modo è altresì possibile mantenere gli hostname e le porte originali definiti per l'accesso diretto ai vari applicativi, prima della loro integrazione con il Reverse Proxy.

6.2 Shibboleth SP

Il demone Shibboleth SP, realizzato dal daemon "shibd" si compone dei seguenti elementi:

- **Session Initiator** – questo componente agisce come un filtro di autenticazione: ha infatti il ruolo di intercettare le richieste dirette al servizio applicativo e verificare il contesto di autenticazione per la sessione utente. Nel caso non esista un'autenticazione pregressa riutilizzabile, viene generata una nuova richiesta destinata ad un Identity Provider o altra infrastruttura di autenticazione. Per produrre tale richiesta, il componente accede alla propria configurazione dove sono elencate informazioni quali le modalità di autenticazione richieste e il protocollo SAML da utilizzare.

- **Assertion Consumer Service** – questo componente ha il compito di ricevere i messaggi SAML Response prodotti dall'infrastruttura di autenticazione (GEL) contenenti un'asserzione con uno statement di autenticazione. Assertion Consumer Service estrae i dati utente dal messaggio di Response e li mette a disposizione del servizio applicativo. Al pari del componente Session Initiator, anche Assertion Consumer Service si avvale di servizi di supporto realizzati dai componenti "Metadata Manager" e "Signature Manager". Il primo è utilizzato per accedere ai metadati del soggetto, interno all'infrastruttura di autenticazione, che ha inviato il messaggio SAML Response, allo scopo di verificare la sua firma digitale. Il secondo è acceduto durante il processo di verifica vera e propria della firma. Infine, il componente Assertion Consumer Service è in grado di estrarre i dati relativi all'utente autenticato e metterli a disposizione dell'erogatore in un formato neutro, in particolare attraverso un set di header HTTP appositamente definiti nella configurazione. Le chiavi, con le quali l'Assertion Consumer Service verifica la firma dell'asserzione prodotta da GEL sono censite all'interno del metadata specificato nel tag <MetadataProvider> del file *shibboleth2.xml*. **Questo file va modificato in modo da censire i certificati digitali utilizzati dal sistema di autenticazione di Regione Campania durante l'operazione di firma delle asserzioni. Si consulti a tal proposito la sezione predisposta allo scopo nel documento.**

- **Signature Manager** – È un componente che ha lo scopo di fornire le funzionalità atte a firmare digitalmente i messaggi in standard SAML prodotti dai vari soggetti interoperanti, prima del loro invio. Consente inoltre di verificare la firma dei messaggi SAML a valle della loro ricezione. Ciascun componente produttore di messaggi SAML, pertanto, deve essere dotato di una chiave privata per apporre le firme e di una corrispondente chiave pubblica contenuta in un certificato per consentire agli altri componenti di verificare la firma apposta. Le chiavi pubbliche sono reperibili direttamente consultando i metadati di ciascun soggetto.

- **Metadata Manager** – questo componente ha lo scopo di fornire le funzionalità di accesso ai servizi di pubblicazione delle strutture metadati esposte dai vari componenti dell'infrastruttura di autenticazione. È in grado di interpretare i metadati, distinguendo quelli propri delle varie tipologie di soggetti (es. Identity Provider, Service Provider ecc.) e di estrarre le informazioni utili agli scenari d'interazione. Tale componente realizza anche il servizio di pubblicazione dei metadati contattabile remotamente via HTTP da qualunque altro soggetto.

6.3 Interfacciamento verso soggetti esterni

Come illustrato nella figura precedente, il sistema Reverse Proxy presenta interfacce verso vari tipi di

soggetti esterni e in particolare verso i seguenti:

- client dei servizi applicativi che si vogliono proteggere tramite autenticazione (utenti che intendo accedere ai servizi)
- application server che ospitano i servizi applicativi veri e propri
- infrastrutture di autenticazione da attivare durante l'accesso ai servizi (GEL)

Nel seguito vengono illustrate brevemente tali interfacce, per ciascuno dei soggetti elencati. I dettagli della configurazione dei vari sotto-sistemi (Apache HTTP Server e Shibboleth SP) al fine di realizzare tali interfacce saranno forniti nel capitolo successivo.

6.3.1 Interfacce verso i fruitori di servizi applicativi

Nei confronti dei fruitori, o client, di servizi applicativi il Reverse Proxy può offrire interfacce di tipo HTTP o HTTPS, secondo le necessità dello specifico servizio che si intende proteggere mediante autenticazione. Tali interfacce sono realizzate da Apache HTTP Server, e in particolare dai vari Virtual Host configurati al suo interno.

Il generico client deve essere messo in condizioni di indirizzare il Virtual Host opportuno, in funzione del servizio richiesto, ad esempio registrando il nome di dominio relativo al Service Provider e facendolo puntare al Reverse Proxy. Si rimanda di nuovo al capitolo successivo per maggiori dettagli al riguardo.

6.3.2 Interfacce verso gli erogatori di servizi applicativi

Il Reverse Proxy inoltra le richieste, giunte dai vari client dai quali viene acceduto, verso gli opportuni erogatori dei servizi applicativi, dopo che il processo di autenticazione dell'utente richiedente si è concluso positivamente. La comunicazione tra Reverse Proxy e Service Provider può avvenire attraverso i protocolli standard HTTP e HTTPS, qualora l'application server retrostante esponga direttamente un endpoint di tale tipologia.

Tale situazione è tipica di applicativi preesistenti all'integrazione con il Reverse Proxy e che intendono mantenere la modalità di accesso originaria anche dopo essersi integrati. In alternativa, il Reverse Proxy è in grado di interagire con un Service Provider anche utilizzando il protocollo AJP 1.3, mediante il quale il Service Provider non è direttamente contattabile da client quali normali browser web.

Esempi di Application Server in grado di esporre endpoint AJP sono Apache Tomcat e Jboss. Dalla parte del Reverse Proxy, il Reverse Proxy utilizza Apache HTTP Server con i moduli "mod_proxy" e "mod_proxy_ajp" per supportare rispettivamente i protocolli HTTP/HTTPS e AJP per interagire con i Service Provider a valle.

6.3.3 Interfacce verso le infrastrutture di autenticazione e Identity Provider

Il Reverse Proxy è in grado di interfacciarsi con infrastrutture di autenticazione conformi alle specifiche

SAML. Per tale interfacciamento, il Reverse Proxy utilizza il server Shibboleth SP. In particolare per l'invio delle richieste di autenticazione viene coinvolto il componente "Session Initiator", mentre le risposte di autenticazione vengono indirizzate al componente "Assertion Consumer Service".

7. Configurazione del Reverse Proxy

In questo capitolo vengono descritte le fasi di configurazione da svolgere sui componenti del Reverse Proxy al fine di integrare con esso un nuovo applicativo e proteggerne quindi l'accesso mediante autenticazione. Le attività di configurazione che interessano i diversi tipi di Service Provider supportati saranno invece oggetto del prossimo capitolo.

7.1 Configurazione di Apache HTTP Server

Come presentato nel capitolo precedente, presso l'istanza Apache HTTP Server dispiegata come Reverse Proxy sono configurati vari Virtual Host, ciascuno dei quali dedicato all'integrazione di uno specifico applicativo posto in posizione logica retrostante al Reverse Proxy.

7.1.1 Attività di configurazione generali

Indipendentemente dai vari Virtual Host che dovranno essere creati, Apache HTTP Server necessita di alcune direttive base di configurazione atte a creare le condizioni operative comuni per attivare correttamente il demone Shibboleth SP per qualsiasi applicativo che verrà integrato. Tali direttive di configurazione vengono pertanto impostate una volta sola in fase di configurazione del Reverse Proxy e valgono per tutti gli applicativi che saranno integrati successivamente. In particolare, tali direttive sono le seguenti:

- all'interno del file di configurazione generale di Apache *httpd.conf* impostare il valore della direttiva *ServerName* con il nome dell'host del Reverse Proxy, ad esempio:

```
ServerName SERVICEPROVIDER.DOMAIN.COM
```

- caricare il modulo *mod_headers* per abilitare l'utilizzo degli header HTTP. Shibboleth SP infatti crea degli header HTTP ad-hoc per inviare al servizio applicativo il valore degli attributi del profilo utente ricevuti durante il processo di autenticazione SAML:

```
LoadModule headers_module modules/mod_headers.so
```

- abilitare la direttiva UseCanonicalName - se impostata ad off, il funzionamento dell'intera architettura è basato esclusivamente sull'utilizzo di IP address fisici e non su URL contenenti nomi logici; se impostata ad on, viene consentito l'utilizzo di nomi logici (valorizzazione raccomandata):

```
UseCanonicalName on
```

- caricare il modulo mod_shib per abilitare la connessione con il server Shibboleth SP:

```
LoadModule mod_shib <PATH_TO_SHIBBOLETH_SP>/lib/shibboleth/mod_shib_22.so
```

il percorso su file system del modulo mod_shib dipende dall'installazione del server Shibboleth SP fatta. Ad esempio, su sistemi Unix tale percorso potrebbe essere "/usr/lib/shibboleth/mod_shib_22.so";

- definire le seguenti regole di URL rewriting, valide per tutti i Service Provider che si dovranno integrare:

```
RewriteEngine On
RewriteOptions Inherit

RewriteCond%{QUERY_STRING}
idpUrl=([^?]+)(\?)([^?]+)&shire=([^&]+)&(.*)target=(.*)&providerId=.*
RewriteRule (.*) %1%2%3&TARGET=%4?target=%6 [R,NE,L]

RewriteCond %{QUERY_STRING} idpUrl=([^?]+)&shire=([^&]+)&(.*)target=(.*)&providerId=.*
RewriteRule (.*) %1?TARGET=%2?target=%4 [R,NE,L]

RewriteLog "rewrite.log"
RewriteLogLevel 3
```

Inoltre, nel Web Server Apache si intendono già caricati i moduli *mod_proxy, mod_proxy_ajp, mod_rewrite* e *mod_ssl* che sono normalmente presenti in molte distribuzioni standard di Apache. Se così non fosse è possibile abilitarli inserendo le apposite direttive "LoadModule" come quelle illustrate in precedenza.

7.1.2 Integrazione di un nuovo applicativo nell'Apache HTTP Server

Dopo aver effettuato gli interventi descritti in precedenza oppure dopo aver verificato che tale configurazione è già attiva, è possibile procedere con le attività mirate ad integrare un nuovo applicativo.

Ciò comporta pertanto come prima cosa la creazione di un nuovo Virtual Host presso il web server. A questo scopo è possibile aggiungere alla configurazione un frammento, posizionabile in un file separato rispetto al file di configurazione principale (httpd.conf), che deve tuttavia trovarsi nella cartella di nome "conf.d". Le modalità secondo cui è possibile creare un nuovo Virtual Host dipendono dal fatto che tale Virtual Host sarà acceduto su protocollo HTTP oppure HTTPS; tali alternative sono analizzate nel seguito.

7.1.2.1 Virtual Host su protocollo HTTP

Nel caso di utilizzo del protocollo HTTP è possibile utilizzare la modalità "name-based" per i Virtual Host in Apache, secondo cui all'host fisico del Reverse Proxy vengono assegnati più hostname alternativi, tanti quanti sono quelli ai quali devono essere acceduti i vari appliocativi protetti. È quindi possibile mantenere per il Reverse Proxy un unico indirizzo IP, al quale far corrispondere, su DNS, i vari hostname. Per abilitare tale modalità è necessario che nel file di configurazione principale di Apache HTTP Server (*httpd.conf*) sia riportata la seguente direttiva:

```
NameVirtualHost       *:80
```

Assumendo che venga utilizzata la porta HTTP default (80), in questa modalità il frammento di definizione del Virtual Host è il seguente:

```
<VirtualHost *:80>
        ServerName SERVICEPROVIDER.DOMAIN.COM
        ErrorLog   logs/SERVICEPROVIDER.DOMAIN.COM_error.log
        CustomLog  logs/SERVICEPROVIDER.DOMAIN.COM_access.log
   LogLevel warn
   ProxyPass /WEBAPP PROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INT_PROTECTED_PATH
   ProxyPassReverse /WEBAPP
PROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INT_PROTECTED_PATH
   ProxyPass /WEBAPP/PROTECTED_PATH/Shibboleth.sso !

   <Location /WEBAPP/PROTECTED_PATH>
           AuthType shibboleth
           ShibRequireSession On
           ShibUseHeaders On
           require shibboleth
   </Location>
   <IfModule mod_alias.c>
        <Location /shibboleth>
               Allow from all
        </Location>
        Alias /shibboleth/main.css SHIBBOLETH_INSTALL_PATH/doc/shibboleth/main.css
        Alias /shibboleth/logo.jpg SHIBBOLETH_INSTALL_PATH/doc/shibboleth/logo.jpg
   </IfModule>

   RewriteEngine On
   RewriteOptions Inherit
   RewriteCond %{REQUEST_METHOD} GET
   RewriteCond %{QUERY_STRING} target=([^\?]+)(\?*)(.*)
   RewriteRule /Shibboleth.sso/SAML/POST %1?%3 [R,L]
   RewriteLog " rewrite.log"
   RewriteLogLevel 3
</VirtualHost>
```

In esso occorre operare le seguenti sostituzioni:

- sostituire la stringa "**SERVICEPROVIDER.DOMAIN.COM**" con l'hostname del Service Provider come esposto verso i fruitori esterni;
- sostituire la stringa "**WEBAPP**" con il nome dell'applicazione presso l'Application Server di destinazione. Nel caso l'applicativo risponda sul path radice sarà sufficiente specificare semplicemente il percorso minimo ("/");
- sostituire la stringa "PROTO" con il protocollo utilizzato dall'Application Server che ospita il Service Provider da proteggere. I protocolli supportati sono "http", "https" e "ajp";
- sostituire la stringa "INTERNAL.SERVICEPROVIDER.DOMAIN.COM" con l'hostname dell'Application Server che ospita il Service Provider da proteggere;
- sostituire la stringa "PORT" con il numero di porta dell'endpoint presso l'Application Server che ospita il Service Provider da proteggere. Naturalmente, è necessario esplicitare il numero di porta soltanto nel caso in cui esso differisca da quello di default per il protocollo;
- sostituire la stringa "INT_PROTECTED_PATH" con il percorso dell'applicazione del Service Provider da proteggere. Come nel caso della stringa "WEBAPP", anche in questo caso se sull'Application Server viene utilizzato il percorso radice è sufficiente indicare unicamente la stringa "/";
- sostituire la stringa "PROTECTED_PATH" con il percorso, presso l'applicativo oggetto dell'integrazione, delle risorse che devono essere protette tramite autenticazione; questo consente di evitare di proteggere qualunque percorso presso l'applicativo, e di specificare solo quelli realmente necessari; qualora gli insiemi di risorse da proteggere fossero più di uno, sarà possibile includere ulteriori elementi <Location> simili a quello riportato, uno per ciascun percorso;
- sostituire la stringa "SHIBBOLETH_INSTALL_PATH" con il percorso assoluto su file system della cartella ove è stato installato il server Shibboleth. Tale percorso è utilizzato per localizzare il logo e il CSS di default dell'installazione Shibboleth. Tali elementi sono visualizzati nelle pagine di errore prodotte dal Reverse Proxy ed è possibile sostituirli con altri personalizzati per ciascun Virtual Host.

Si osserva che entrambe le stringhe **WEBAPP** e **PROTECTED_PATH** sopra riportate sono le stesse che è necessario riportare anche nella configurazione del demone Shibboleth SP come sarà illustrato nella sez. 7.2.1.

7.1.2.2 Virtual Host su protocollo HTTPS

Considerato che Service Provider deve essere esposto tramite Reverse Proxy su protocollo HTTPS, è necessario disporre della chiave privata e del relativo **certificato** con chiave pubblica, **rilasciato da una Certification Authority riconosciuta come attendibile**, che saranno usati per stabilire la connessione SSL/TLS tra il fruitore del servizio (l'utente che deve accedere alle applicazioni) e il Virtual Host presso il Reverse Proxy.

Utilizzando il protocollo HTTPS i dati di asserzione rilasciati dall'Identity Provider che transitano verso il browser utente viaggiano su canale cifrato.

Utilizzando il protocollo HTTPS, Apache HTTP Server non può conoscere l'hostname – e quindi il Virtual Host – richiesto dal client fino a che l'handshake SSL non si è completato, e non può completare tale handshake finché non ha presentato il certificato X.509 corrispondente al corretto Virtual Host. Per tale motivo risulta normalmente impossibile per i Virtual Host HTTPS utilizzare la modalità name-based. Esistono tuttavia almeno due modi per mantenere tale modalità, come descritto nel seguito.

Modalità IP-based

Utilizzando la modalità IP-based per definire i Virtual Host, comporta la necessità di assegnare un nuovo indirizzo IP pubblico al Reverse Proxy, a cui far corrispondere mediante DNS l'hostname del Service Provider che deve essere integrato. Il Reverse Proxy si troverebbe così a disporre di tanti indirizzi IP contattabili dai fruitori, quanti sono i Service Provider integrati con esso. In questa modalità il frammento riportato nella sez. 7.1.2.1 deve essere modificato come segue:

```
<VirtualHost IP_SERVICE_PROVIDER:443>
        ServerName SERVICEPROVIDER.DOMAIN.COM
        ErrorLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_error_log
        CustomLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_access_log common

LogLevel warn
SSLEngine on
SSLProtocol all -SSLv2
SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
SSLCertificateFile SERVICEPROVIDER_SSL_CERTIFICATE_FILE_PATH
SSLCertificateKeyFile SERVICEPROVIDER_SSL_CERTIFICATE_KEY_PATH
SSLVerifyDepth 10
SSLOptions +ExportCertData +StdEnvVars

ProxyPass        /WEBAPP
PROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH
ProxyPassReverse /WEBAPP
PROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH
ProxyPass /WEBAPP/PROTECTED_PATH/Shibboleth.sso !

<Location /WEBAPP/PROTECTED_PATH>
        AuthType shibboleth
        ShibRequireSession On
        ShibUseHeaders On
        require shibboleth
```

```
</Location>

<IfModule mod_alias.c>
        <Location /shibboleth>
                Allow from all
        </Location>
        Alias /shibboleth/main.css SHIBBOLETH_INSTALL_PATH/doc/shibboleth/main.css
        Alias /shibboleth/logo.jpg SHIBBOLETH_INSTALL_PATH/doc/shibboleth/logo.jpg
</IfModule>

RewriteEngine On
RewriteOptions Inherit

RewriteCond %{REQUEST_METHOD} GET
RewriteCond %{QUERY_STRING} target=([^\?]+)(\?*)(.*)
RewriteRule /Shibboleth.sso/SAML/POST %1?%3 [R,L]
RewriteLog " rewrite.log"
RewriteLogLevel 3

</VirtualHost>
```

Per il significato dei parametri che devono essere sostituiti in tale frammento si rimanda alla sez. 7.1.2.1. Rispetto al frammento già descritto in precedenza, si evidenziano unicamente i seguenti parametri aggiuntivi:

- Sostituire la stringa "**IP_SERVICE_PROVIDER**" con l'indirizzo IP pubblico assegnato al Reverse Proxy e dedicato alle richieste destinate all'applicativo oggetto dell'integrazione; tale indirizzo IP dovrà essere associato, mediante DNS, all'hostname specificato dalla stringa "**SERVICEPROVIDER.DOMAIN.COM**";
- sostituire la stringa "**SERVICEPROVIDER_SSL_CERTIFICATE_FILE_PATH**" con il percorso, relativo alla cartella contenente i file di configurazione di Apache HTTP Server, del file contenente il certificato SSL che deve essere utilizzato per l'endpoint HTTPS del Virtual Host; il certificato deve essere codificato in formato PEM (Base64);
- sostituire la stringa "**SERVICEPROVIDER_SSL_CERTIFICATE_KEY_PATH**" con il percorso, relativo alla cartella contenente i file di configurazione di Apache HTTP Server, del file contenente la chiave privata che deve essere utilizzata per l'endpoint HTTPS del Virtual Host; la chiave deve essere codificata in formato PEM (Base64).

Modalità name-based con condivisione del dominio

Nel caso in cui i vari Virtual Host HTTPS che si devono creare condividono lo stesso dominio (es. *sp1.domain.com*, *sp2.domain.com*, ecc.), è possibile adottare la modalità *name-based* per i Virtual Host, in modo non dissimile da quanto già presentato nella sezione relativa. Per ovviare al problema presentato sopra, relativamente all'impossibilità per Apache HTTP Server di determinare certificato SSL da utilizzare, corrispondente al Virtual Host corretto, è necessario far uso di certificati di tipo wildcard (nell'esempio, certificati con *CN=*.domain.com*) così da adattarsi a tutti i Virtual Host dello stesso dominio. In questa modalità, pertanto, il frammento di configurazione di Apache interessato

viene modificato come segue:

```
NameVirtualHost *:443
<VirtualHost *:443>
        ServerName SERVICEPROVIDER.DOMAIN.COM
        ErrorLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_error_log
        CustomLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_access_log common

LogLevel warn
SSLEngine on
SSLProtocol all -SSLv2
SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
SSLCertificateFile SERVICEPROVIDER_WILDCARD_SSL_CERTIFICATE_FILE_PATH
SSLCertificateKeyFile SERVICEPROVIDER_WILDCARD_SSL_CERTIFICATE_KEY_PATH
SSLVerifyDepth 10
SSLOptions +ExportCertData +StdEnvVars

ProxyPass        /WEBAPP
PROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH
ProxyPassReverse /WEBAPP
PROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH

ProxyPass /WEBAPP/PROTECTED_PATH/Shibboleth.sso !

<Location /WEBAPP/PROTECTED_PATH>
        AuthType shibboleth
        ShibRequireSession On
        ShibUseHeaders On
        require shibboleth
</Location>

<IfModule mod_alias.c>
       <Location /shibboleth>
              Allow from all
       </Location>
       Alias /shibboleth/main.css SHIBBOLETH_INSTALL_PATH/doc/shibboleth/main.css
       Alias /shibboleth/logo.jpg SHIBBOLETH_INSTALL_PATH/doc/shibboleth/logo.jpg
</IfModule>

RewriteEngine On
RewriteOptions Inherit

RewriteCond %{REQUEST_METHOD} GET
RewriteCond %{QUERY_STRING} target=([^\?]+)(\?*)(.*)
RewriteRule /Shibboleth.sso/SAML/POST %1?%3 [R,L]
RewriteLog " rewrite.log"
RewriteLogLevel 3

</VirtualHost>
```

Si noti la differenza rispetto al caso precedente, caratterizzata dal fatto che non è necessario esplicitare l'indirizzo IP del server. Per gestire l'handshake SSL iniziale, infatti, Apache utilizza il primo Virtual Host elencato, il quale specifica il certificato wildcard. Successivamente Apache è in grado di

selezionare il Virtual Host corretto in base al nome del server specificato nella richiesta HTTP. Essendo la configurazione pressoché identica al caso precedente, non vengono qui dettagliati nuovamente tutti i parametri di configurazione sui quali è possibile intervenire.

Modalità name-based con Server Name Indication (SNI)

In alternativa alle due modalità presentate sopra, ne esiste una terza che fa uso della modalità SNI (Server Name Indication) definita dal documento RFC3546, che costituisce un'estensione al protocollo TLS per lo scambio dei nomi dei server in fase di setup delle connessioni sicure su protocollo HTTP. Si osserva che tale modalità è ormai ampiamente supportata dalla totalità dei web server esistenti. Nel caso di Apache HTTP Server, è necessario utilizzare una versione 2.2.12 o successiva, compilata con un supporto della libreria OpenSSL 0.9.8f o successiva, nella quale siano state abilitate le estensioni TLS (abilitate per default a partire dalla versione 0.9.8j).

Utilizzando una versione idonea di Apache HTTP Server è possibile definire i Virtual Host per connessioni HTTPS in modalità name-based in modo del tutto simile a quanto descritto per il caso dei Virtual Host per connessioni HTTP. In particolare, il frammento di configurazione di Apache in questo caso è il seguente:

```
NameVirtualHost *:443

<VirtualHost *:443>
        ServerName SERVICEPROVIDER.DOMAIN.COM
        ErrorLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_error_log
        CustomLog logs/SERVICEPROVIDER.DOMAIN.COM_ssl_access_log common

LogLevel warn

SSLProxyEngine on
SSLEngine on
SSLProtocol +all -SSLv2 -SSLv3
SSLCipherSuite RC4-SHA:AES128-SHA:HIGH:MEDIUM:!aNULL:!MD5G
SSLHonorCipherOrder on
SSLCertificateFile SERVICEPROVIDER_SSL_CERTIFICATE_FILE_PATH
SSLCertificateKeyFile SERVICEPROVIDER_SSL_CERTIFICATE_KEY_PATH
SSLVerifyDepth 10
SSLOptions +ExportCertData +StdEnvVars

ProxyPass        /WEBAPP
PROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH
ProxyPassReverse /WEBAPP
PROTO://INTERNAL.SERVICEPROVIDER.DOMAIN.COM:PORT/INTERNAL_PROTECTED_PATH

ProxyPass /WEBAPP/PROTECTED_PATH/Shibboleth.sso !

<Location /WEBAPP/PROTECTED_PATH>
        AuthType shibboleth
        ShibRequireSession On
        ShibUseHeaders On
        require shibboleth
</Location>
```

```
<IfModule mod_alias.c>
        <Location /shibboleth>
                Allow from all
        </Location>
        Alias /shibboleth/main.css SHIBBOLETH_INSTALL_PATH/doc/shibboleth/main.css
        Alias /shibboleth/logo.jpg SHIBBOLETH_INSTALL_PATH/doc/shibboleth/logo.jpg
</IfModule>

RewriteEngine On
RewriteOptions Inherit

RewriteCond %{REQUEST_METHOD} GET
RewriteCond %{QUERY_STRING} target=([^\?]+)(\?*)(.*)
RewriteRule /Shibboleth.sso/SAML/POST %1?%3 [R,L]
RewriteLog " rewrite.log"
RewriteLogLevel 3

</VirtualHost>
```

Si noti che i valori "consigliati" da utilizzare per `SSLProtocol` e `SSLCipherSuite` cambiano continuamente nel tempo in base alle vulnerabilità che vengono scoperte nel tempo sui protocolli SSL/TLS ad alle nuove versioni di questi ultimi rilasciati nel tempo. È compito dell'Ente utilizzare i valori più opportuni ed adeguarli nel tempo.

7.2 Configurazione del demone Shibboleth SP

Il diagramma architetturale presentato sopra presenta la struttura del componente Shibboleth SP del quale sono evidenziati i componenti principali che sono oggetto di configurazione. In particolare, l'infrastruttura Shibboleth SP consiste di due componenti principali:

- un componente aggiuntivo per il Web Server (un modulo nel caso di Web Server Apache e un filtro ISAPI nel caso di Web Server IIS)
- un *demone* denominato *shibd* che si occupa della gestione delle sessioni di autenticazione

Il primo di tali componenti (*mod_*shib) è stato già descritto nella sez. 6.1 in merito alla configurazione di Apache HTTP Server. In questa sezione verrà invece dettagliata la configurazione del secondo componente, che costituisce un processo separato del sistema operativo nel quale opera, di nome *shibd*.

Un'installazione tipica del server Shibboleth SP crea alcune cartelle nel file system come descritto di seguito:

- *bin*: contiene alcune utility da linea di comando utilizzate dal software per firmare i messaggi SAML prodotti e verificare la firma dei messaggi ricevuti, per effettuare interrogazioni sui metadati, ecc.
- *doc*: contiene le licenze del software e le note di rilascio dello stesso
- *etc/shibboleth*: contiene i file di configurazione del software
- *lib*: contiene le librerie da cui dipende il software e, nella directory figlia *shibboleth*, i componenti per i web server
- *sbin*: contiene l'eseguibile del demone *shibd*
- *share/xml*: contiene gli schemi XML dei messaggi SAML e dei file di configurazione del software
- *var*: conterrà i file di log (nella directory figlia *log*) e i file utilizzati durante l'esecuzione del software (nella directory figlia *run*), ad esempio le copie locali dei metadati delle entità interagenti

Le cartelle indicate sopra sono raccolte in un'unica cartella contenitore nel caso di installazione fatta su sistema operativo Microsoft Windows. Nel caso di sistemi Unix/Linux, tali cartelle sono già presenti nell'alberatura standard della maggior parte delle distribuzioni.

La configurazione del software Shibboleth SP si compone di tre step, ai quali corrispondono altrettanti file di configurazione in formato XML presenti nella cartella "*/etc/shibboleth*":

- configurazione dei Virtual Host corrispondenti ai vari Service Provider integrati, delle modalità di autenticazione e dei metadati (file *shibboleth2.xml*);
- configurazione del mapping degli attributi del profilo degli utenti autenticati (file *attribute-map.xml*);

- configurazione delle policy di filtraggio degli attributi del profilo degli utenti autenticati (file *attribute-policy.xml*).

Nelle prossime sezioni verranno dettagliati questi tre step, descrivendo la struttura dei relativi file di configurazione.

7.2.1 File di configurazione shibboleth2.xml

Il file *shibboleth2.xml* rappresenta il nodo centrale della configurazione del Service Provider Shibboleth: con esso è possibile configurare la comunicazione tra il modulo del web server (*mod_shib*) ed il *demone shibd;* in esso, inoltre, sono definiti i singoli servizi applicativi che si desidera proteggere tramite autenticazione SAML, specificando per ciascuno gli endpoint SAML che si vuole esporre all'esterno (i quali rappresenteranno l'interfaccia verso le infrastrutture di autenticazione).

Si rammenta che, in linea generale, l'unica operazione da compiere lato SP è applicare le **personalizzazioni** descritte nella sezione 3.4, lasciando il restante contenuto del file inalterato rispetto a quanto installato dal prodotto in fase di setup. Il presente paragrafo ha carattere informativo qualora l'Ente intendesse sfruttare alcune personalizzazioni avanzate o sfruttare alcune potenzialità insite nel prodotto.

7.2.1.1 Configurazione generale

Il frammento del file inerente la configurazione della comunicazione tra il modulo *mod_shib* e il demone *shibd* è il seguente:

```
<OutOfProcess logger="shibd.logger">
        <!--  <Extensions> ...</Extensions>  -->
</OutOfProcess>
<InProcess logger="native.logger">
        <!--  <Extensions> ...</Extensions>  -->
</InProcess>
<TCPListener address="127.0.0.1"port="1600"acl="127.0.0.1"/>
```

Gli elementi *<OutOfProcess>*e *<InProcess>* configurano rispettivamente il comportamento del demone *shibd* e del modulo *mod_shib* o del filtro ISAPI (a seconda del web server che si sta utilizzando) attraverso l'uso di sotto-elementi *<Extensions>* e *<ISAPI>*, mentre gli attributi *logger* permettono di specificare i file di configurazione dell'infrastruttura di logging per questi componenti.

In caso di utilizzo del web server IIS è necessario configurare l'elemento *<ISAPI>* specificando uno o più sotto-elementi *<Site>*, che si riferiscono alle istanze dei siti gestiti da IIS. Si rimanda alla sezione apposita e alla documentazione Shibboleth, per maggiori dettagli relativi all'uso del web server Microsoft IIS.

Ai fini delle attività di integrazione oggetto di questo documento, non è necessario specificare nessun elemento *<Extensions>*. L'elemento *<TCPListener>* permette di specificare l'indirizzo e la porta a cui è connesso il demone *shibd* (queste informazioni sono state specificate in fase di installazione e non

necessitano di riconfigurazione).

7.2.1.2 Definizione dei Virtual Host

La definizione di un nuovo Service Provider viene compiuta andando ad operare in due sezioni distinte del file di configurazione *shibboleth.xml*.

Innanzitutto è necessario, tramite l'elemento *<RequestMapper>*, specificare il percorso dei singoli servizi che si desidera configurare, associando ad ognuno di essi un identificativo univoco. Il frammento interessato è mostrato di seguito:

```
<RequestMapper type="Native">
   <RequestMap applicationId="default">
      <Host name="SERVICEPROVIDER.DOMAIN.COM" scheme="https" port="443">
         <Path name="WEBAPP1/PROTECTED_PATH1" authType="shibboleth" requireSession="true"
                                                                    exportAssertion="true"/>
         <Path name="WEBAPP2/PROTECTED_PATH2" applicationId="SERVICE2_ID"
authType="shibboleth"
                                              requireSession="true" exportAssertion="true"/>
      </Host>
   </RequestMap>
</RequestMapper>
```

Il frammento sopra riportato definisce un Virtual Host di tipo HTTPS per il Service Provider con hostname **SERVIECPROVIDER.DOMAIN.COM**, in ascolto sulla porta standard (443). È possibile variare tali tre parametri per adattare la configurazione a qualunque scenario particolare. Si noti che in questo caso non è necessario fare distinzione tra Virtual Host per connessioni HTTP e Virtual Host per connessioni HTTPS in quanto il front-end è costituito dal server web Apache che riceve le richieste ed è in grado di attivare il Virtual Host corretto, il quale rimanderà a Shibboleth SP individuando già la parte di configurazione interessata.

Come per la configurazione di Apache, è necessario indicare i percorsi protetti da autenticazione per i quali Shibboleth SP può comportarsi in modo diverso relativamente all'attivazione della fase di autenticazione. Nell'esempio, mediante gli elementi *<Path>* definiti all'interno dell'elemento *<Host>*, è possibile definire altrettante risorse (URL) da proteggere mediante autenticazione, presso quel Service Provider.

Dopo aver definito le risorse protette, i dettagli sulle modalità con le quali deve essere attivata la procedura di autenticazione per l'accesso a ciascuna di esse vengono fornite in sezioni successive del file di configurazione. In particolare, se per una direttiva *<Path>* non viene specificato l'attributo "*applicationId*", i dettagli aggiuntivi della configurazione saranno riportati in una sezione denominata *<ApplicationDefaults>*. È possibile definire, per ciascuna risorsa protetta dei parametri personalizzati mediante l'indicazione di un attributo *"applicationId"* sull'elemento <Path>, fornendo come valore il nome di una sezione del file di configurazione, denominata *<ApplicationOverride>,* come elemento annidato entro *<ApplicationDefaults>*. Laddove in quest'ultima non siano specificati alcuni dati, verranno utilizzati i valori definiti in *<ApplicationDefaults>*.

Si segnala che è possibile raffinare la risoluzione con cui vengono individuate le risorse protette, includendo uno o più elementi <Path> all'interno di altri elementi <Path> e specificando per essi valori diversi rispetto agli elementi contenitori.

7.2.1.3 Configurazione delle interfacce SAML

Una volta associati degli identificatori ad ogni servizio, è possibile configurare il comportamento di Shibboleth a fronte delle richieste di risorse protette da autenticazione. Il server Shibboleth attiverà per ciascun nuovo accesso ad una risorsa nell'ambito di una sessione utente, un'interazione con l'infrastruttura di autenticazione o Identity Provider in accordo alla specifica SAML supportata da tale infrastruttura.

Per regolare tale comportamento, al file shibboleth2.xml occorre aggiungere una sezione <ApplicationOverride> per ciascuno dei Service Provider oggetto dell'integrazione, di solito in corrispondenza 1:1 con i Virtual Host come definiti in precedenza. Ogni sezione <ApplicationOverride> deve essere riportata all'interno della sezione <ApplicationDefaults>, dopo tutte le altre definizioni, prima della chiusura di tale sezione.

Ogni sezione <ApplicationOverride> ha la seguente struttura:

```
<ApplicationOverride id="applicationID" entityID="SP_ENTITY_ID"
  REMOTE_USER="saml_attribute_codiceFiscale"
homeURL="HTTPS://SERVICEPROVIDER.DOMAIN.COM/HOMEURL"
  signing="true" encryption="false">
  <Sessions lifetime="28800"timeout="3600"checkAddress="false"
handlerURL="/WEBAPP/PROTECTED_PATH/Shibboleth.sso" handlerSSL="false"
exportLocation="/GetAssertion" exportACL="127.0.0.1" idpHistory="false" idpHistoryDays="7">
      <SSO entityID="https://idpcrl.crs.campania.it/scauth">
         SAML1
      </SSO>
  </Sessions>
  <MetadataProvider type="XML" file="[PATH_TO_IDPC_METADATA]/IdPCRL-metadata.xml"/>
  <AttributeExtractor type="XML" validate="true" path="attribute-map-ipdcrl.xml"/>
</ApplicationOverride>
```

In esso occorre operare le seguenti sostituzioni:

- sostituire la stringa "**SP_ENTITY_ID**" con l'identificatore che si vuole associare allo specifico Service Provider; tale identificatore sarà riportato anche nei relativi metadati SAML come entityID;
- sostituire l'URL "**HTTPS://SERVICEPROVIDER.DOMAIN.COM/HOMEURL**" con l'URL della risorsa presso il Service Provider a cui indirizzare l'utente al termine del processo di autenticazione nel caso in cui non sia specificato nessun altro URL valido;
- sostituire la stringa **"/WEBAPP/PROTECTED_PATH**" con il percorso, presso il Service Provider, della risorsa che si intende proteggere**.**

7.2.1.4 Configurazione dell'Identity Provider

In linea generale, è possibile configurare il SP Shibboleth in modo che veicoli le proprie "authentication

request" verso un qualsiasi Identity Provider che supporti il protocollo SAML 2.0.

Nell'ambito specifico trattato dal presente documento, l'Identity Provider di riferimento è naturalmente quello di Regione Campania (IdPC), ed in particolare il servizio GEL - Gateway Enti Locali.

È necessario configurare Shibboleth in modo che:

1. consenta l'ingaggio dell'opportuno Identity Provider a fronte del tentativo di accesso ad una risorsa protetta;
2. sappia verificare le asserzioni di identità prodotte da IdPC. Tale operazione si compie censendo i **certificati** (e le relative **root**) cui corrispondono le chiavi crittografiche utilizzate da IdPC.

Entrambi i punti sono stati trattati nella sezione 3.4: in particolare, la trust chain di certificazione è contenuta nel file *idp_metadata.xml*, che viene reso disponibile da REGIONE CAMPANIA nell'ambito di un "Kit di integrazione" al servizio GEL.

7.2.2 Mappatura degli attributi utente

Il server Shibboleth SP è in grado di ricevere i messaggi SAML prodotti dall'infrastruttura di autenticazione o Identity Provider utilizzato e di estrarre da essi gli attributi relativi agli utenti autenticati. Tali attributi vengono quindi inseriti in altrettanti header HTTP, per consentire al Service Provider di accedervi ed utilizzarli.

Per definire le corrispondenze tra gli attributi presenti nei messaggi SAML e gli header HTTP prodotti, è necessario intervenire sul file "*/etc/shibboleth/attribute-map.xml*", oggetto di questa sezione. In tale file sono definite una serie di regole di mappatura come la seguente:

```
<Attribute name="NOME_ATTRIBUTO_SAML" id="NOME_HEADER_HTTP"
nameFormat="SAML_ATTR_NAME_FORMAT">
        <AttributeDecoder xsi:type="StringAttributeDecoder"/>
</Attribute>
```

Anche questo file (*attribute-map.xml*) viene fornito nel "Kit di integrazione" distribuito da REGIONE CAMPANIA in modo che risulti agevole il mapping dei principali attributi provenienti dal sistema di autenticazione.

Gli attributi configurabili sono:

- **name**: indica il nome dell'attributo così come specificato nell'asserzione SAML ricevuta
- **id**: indica il nome che si vuole dare all'header HTTP che conterrà il valore di questo attributo
- **nameFormat**: indica il formato dell'attributo, così come specificato dallo standard SAML 2.0

L'elemento *<AttributeDecoder>* specifica come deve essere interpretato il valore dell'attributo e in generale non deve essere modificato.

A titolo d'esempio si riporta una mappatura dell'attributo relativo al codice fiscale:

```
<Attribute name="codiceFiscale" id="saml_attribute_codicefiscale"
                                nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <AttributeDecoder xsi:type="StringAttributeDecoder"/>
</Attribute>
```

7.2.3 Filtraggio degli attributi utente restituiti

A fronte delle definizioni degli attributi mappati come descritto nella sezione apposita, è possibile definire quali tra essi il server Shibboleth utilizzerà effettivamente per popolare gli header HTTP. Di seguito è mostrata una semplice regola, presente nel file *attribute-policy.xml*, che abilita la comunicazione al Service Provider di un attributo, specificando il nome dell'header HTTP corrispondente, indipendentemente dai valori che questo assume:

```
<afp:AttributeFilterPolicy id="releaseToAnyone">
        <afp:PolicyRequirementRule xsi:type="ANY"/>
        <afp:AttributeRule attributeID="NOME_HEADER_HTTP">
                <afp:PermitValueRule xsi:type="ANY"/>
        </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

Per maggiori informazioni sulle regole che possono essere definite, si rimanda alla documentazione in linea del SP Shibboleth.

7.2.4 Casi particolari

7.2.4.1 Configurazione di più applicazioni in un Virtual Host

È possibile proteggere più applicazioni con la stessa istanza di Shibboleth. Per compiere tale operazione, occorre agire sulla stanza <*RequestMapper*> defininendo le varie applicazioni da proteggere ed associando ad ognuna un <*applicationId*> che va poi popolato nella relativa stanza.

Nell'esempio che segue, vengono definiti due applicationId (di cui uno deve sempre essere valorizzato a *default*) per due distinti path. Si noti che il Virtual Host è il medesimo (nell'esempio vale *erogatore.crs.Campania.it*):

```
<RequestMapper type="Native">
    <RequestMap applicationId="default">
    <Host name="erogatore.crs.Campania.it" authType="shibboleth" requireSession="true"
exportAssertion="true" scheme="https" port="443">
        <Path name="/protected1" applicationId="erogatore1" authType="shibboleth"
requireSession="true" exportAssertion="true"/>
        <Path name="/erogatore-main/protected2" applicationId="erogatore2" authType="shibboleth"
requireSession="true" exportAssertion="true"/>
     </Host>
    </RequestMap>
</RequestMapper>
```

7.2.4.2 Configurazione di più applicazioni in più Virtual Host

È altresì possibile configurare più applicazioni in differenti Virtual Host. L'accortezza da attuare è replicare la stanza *Host*, come nell'esempio che segue:

```
 <RequestMapper type="Native">
    <RequestMap applicationId="default">
<Host name="erogatore.crs.Campania.it" authType="shibboleth" requireSession="true"
exportAssertion="true" scheme="https" port="443">
            <Path name="/erogatore-servizio/protected" authType="shibboleth" requireSession="true"
exportAssertion="true"/>
      </Host>
      <Host name="faddev.crs.Campania.it" authType="shibboleth" requireSession="true"
exportAssertion="true" scheme="https" port="443">
            <Path name="/protected" authType="shibboleth" requireSession="true"
exportAssertion="true"/>
      </Host>
    </RequestMap>
</RequestMapper>
```

7.2.4.3 Dispiegamento di Shibboleth dietro un bilanciatore o reverse proxy

Se l'istanza di Shibboleth viene dispiegata "dietro" un bilanciatore o reverse proxy che si occupa di chiudere il canale SSL con il browser, vi sono due possibilità di configurazione:

1. si istanzia il listener SSL in Shibboleth (come già descritto nel presente documento) in modo che siano rispettate le regole di "ingaggio" di IdPC che prevedono l'indirizzo della componente *AssertionConsumer* esposto in *https* e non in *http*, **oppure**
2. si effettua una modifica al file di configurazione del Virtual Host di Apache in modo da "istruirlo" del fatto che il dominio che protegge è esposto con protocollo https. Il protocollo SSL viene così "recepito" da IdPC durante l'ingaggio al sistema di autenticazione (parametro "target" della URL di redirezione). Di seguito un esempio di configurazione:

```
Listen dcss.crs.Campania.it:80

<VirtualHost dcss.crs.Campania.it:80>

ServerName https://dcss.crs.Campania.it
ServerAdmin root@fe-dcsanita.cgi.crs.Campania.it
ErrorLog logs/error_log
TransferLog logs/access_log
LogLevel warn
...
```

7.2.5 Logout da una applicazione

A fronte della volontà di un utente di effettuare il logout da una certa applicazione protetta da una istanza di Shibboleth, l'applicazione stessa dovrà implementare la chiamata alla seguente URL:

http://*applicazione*/protected/Shibboleth.sso/Logout

Tale URL è altresì ricavabile ispezionando il parametro HTTP_SHIB_ASSERTION_01 restituito nell'http-header, e sostituendo le sezioni successive a *Shibboleth.sso* con la sola keyword *Logout*.

L'effetto di tale redirezione del browser sarà la cancellazione del cookie utente gestito da Shibboleth SP. Come conseguenza, un successivo tentativo di accesso ad una risorsa protetta dell'*applicazione* provocherà un nuovo ingaggio di GEL per la negoziazione di una nuova autenticazione.

Le applicazioni integrate a Shibboleth sono tenute ad effettuare quanto descritto, in caso contrario l'unica modalità con cui gli utenti potranno effettuare il logout sarà chiudere il browser.

Si noti che ogni applicazione protetta da Shibboleth SP ha un cookie di sessione "dedicato", quindi la rimozione di un cookie non interferisce con le altre eventuali sessioni utente attive.

Si segnala che la pagina di "local logout" messa a disposizione da Shibboleth è personalizzabile graficamente, in modo da poterla adeguare al *look-and-feel* della web application che la invoca. Tale possibilità è effettivamente sfruttabile nel solo caso in cui una istanza di Shibboleth protegga una sola applicazione, in quanto la pagina di "local logout" è unica per tutte le applicazioni protette.

Si noti infine che la URL sopra indicata può essere completata con il parametro return per indicare a quale indirizzo portare il browser utente dopo l'avvenuto logout, ad esempio:

http://*applicazione*/protected/Shibboleth.sso/Logout?return=http://www.regione.campania.it

Si noti che, per motivi di sicurezza, l'indirizzo specificato dal parametro "return" deve riferirsi allo stesso hostname della *applicazione*.

8. Esempi applicativi di integrazione cn Shibboleth SP

Di seguito si illustrano le due applicazioni web di esempio fornite nel kit di integrazione, che mostrano come con i rispettivi linguaggi di programmazione si è seguita l'idea chiave alla base dell'integrazione, ovvero il passaggio di parametri attraverso la '**request header**'

8.1 Java

L'applicazione consiste in pratica in un filtro servlet che viene eseguito ad ogni richiesta effettuata sula parte protetta dell'applicazione, che si occupa di prendere i valori messi dal SP in request header, istanziare un oggetto di tipo "***User***" contenente i dati dell'utente correntemente loggato ed inserirlo nel contesto "session" a disposizione di tutte le altre componenti dell'applicazione, come la pagina JSP "***index.jsp***" che si trova proprio nella cartella "***protected***"; il compito di questa risorsa e semplicemente stampare a video i dati dell'oggetto presente nella sessione utente.
Per installare l'applicazione scompattare il file "***demo-sso-java.zip***" e deployare il file "***demo-sso.war***" in un application server o in un servlet container come Apache Tomcat.

Ipotizzando di avere un SP configurato opportunamente con dominio acme.it, inserendo nel browser l'indirizzo https://acme.it/demo-sso/ il server mostrerà l'home page:

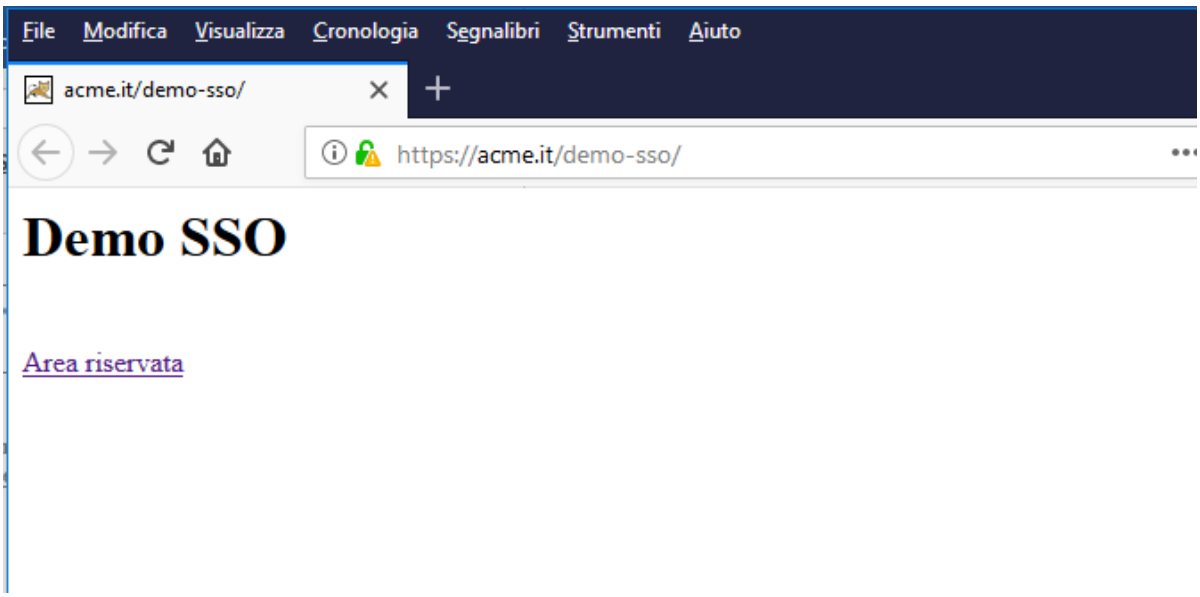

Figura 6 Integrazione con applicazione di test Java, Home Page

Cliccando poi sul link "Area riservata" si effettua una richiesta di una risorsa protetta, infatti il link punta a https://acme.it/demo-sso/protected/index.jsp, dove, se l'SP è configurato correttamente, si attiva Shibboleth che redirige verso GEL per l'immissione delle credenziali.

Una volta immesse le credenziali si viene rediretti all'applicazione che, come già descritto, stampa i dati relativi all'utente collegato:

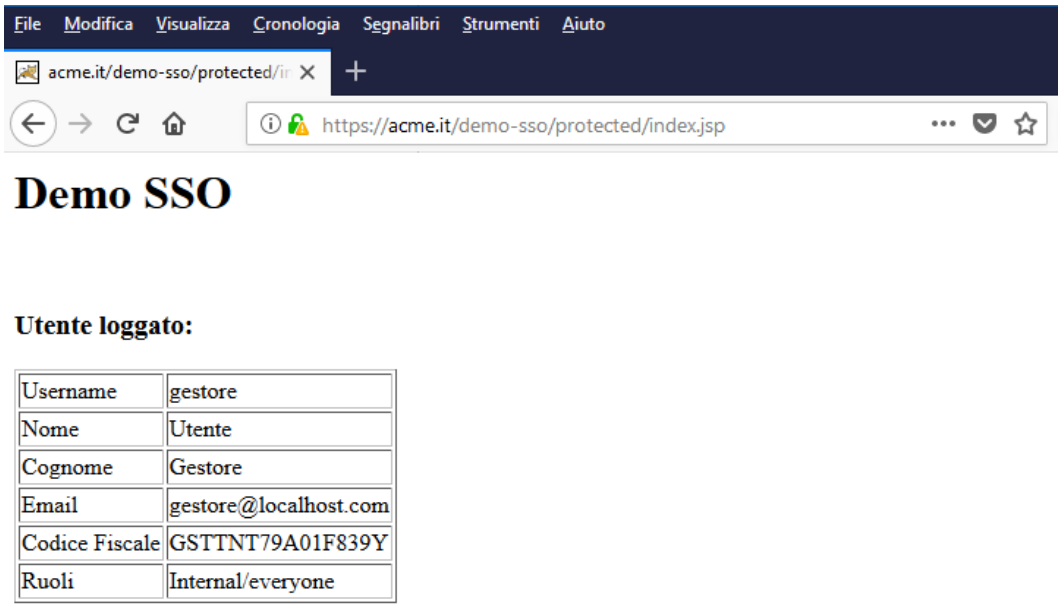

Figura 7 Integrazione con applicazione di test Java, Informazioni utente autenticato

Vediamo più in dettaglio le componenti principali. Innanzitutto il descrittore "web.xml" contiene la definizione del componente filtro:

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>

	<display-name>Demo SSO</display-name>

	<filter>
		<filter-name>LoginFilter</filter-name>
		<display-name>LoginFilter</display-name>
		<filter-class>it.brandid.demo.sso.LoginFilter</filter-class>
	</filter>
	<filter-mapping>
		<filter-name>LoginFilter</filter-name>
		<url-pattern>/protected/*</url-pattern>
	</filter-mapping>

</web-app>
```

Il codice di tale componente, la cui logica è molto semplice, è riportato di seguito:

```
package it.brandid.demo.sso;
```

```
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;

public class LoginFilter implements Filter {

   @Override
   public void init(FilterConfig config) throws ServletException {
   }

   @Override
   public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
                                                        throws IOException,
ServletException {
        HttpServletRequest request = (HttpServletRequest)req;

        User user = new User();
        user.setUserName(request.getHeader("saml_subject_username"));
        user.setFirstName(request.getHeader("saml_attribute_nome"));
        user.setLastName(request.getHeader("saml_attribute_cognome"));
        user.setEmail(request.getHeader("saml_attribute_emailaddress"));
        user.setCodiceFiscale(request.getHeader("saml_attribute_codicefiscale"));
        user.setRoles(request.getHeader("saml_attribute_role"));

        request.getSession().setAttribute("loggedUser", user);
        chain.doFilter(req, res);
   }

   @Override
   public void destroy() {
   }
}
```

Come si vede il filtro non fa altro che leggere i dati di interesse con le istruzioni del tipo "`request.getHeader(<NOME_ATTRIBUTO>)`", istanziare con tali dati un oggetto di tipo "`User`" per poi metterlo nella sessione utente corrente con il comando "`request.getSession().setAttribute("loggedUser", user);`".
La pagina JSP index.jsp mostrata di seguito recupera l'oggetto "loggeduser" dalla sessione e ne stampa gli attributi:

```
<%@page import="java.util.StringTokenizer"%>
<%@page import="it.brandid.demo.sso.User"%>

<h1>Demo SSO</h1>

<br/>

<% User loggedUser = (User)session.getAttribute("loggedUser"); %>

<h3>Utente loggato:</h3>
<table border=1>
	<tr>
		<td>Username</td>
		<td><%= loggedUser.getUserName() %></td>
	</tr>
	<tr>
		<td>Nome</td>
		<td><%= loggedUser.getFirstName() %></td>
	</tr>
	<tr>
		<td>Cognome</td>
		<td><%= loggedUser.getLastName() %></td>
	</tr>
	<tr>
		<td>Email</td>
		<td><%= loggedUser.getEmail() %></td>
	</tr>
	<tr>
		<td>Codice Fiscale</td>
		<td><%= loggedUser.getCodiceFiscale() %></td>
	</tr>
	<tr>
		<td>Ruoli</td>
		<td>
		<% String roles = loggedUser.getRoles(); %>
		<% if (roles!=null) { %>
		   <% StringTokenizer rolesTokenizer = new StringTokenizer(roles, ","); %>
		        <% while (rolesTokenizer.hasMoreTokens()) { %>
		                <%=rolesTokenizer.nextToken() %> <br/>
		        <% } %>
		   <% } %>
		</td>
	</tr>
</table>
```

8.2 PHP

Anche con il linguaggio php l'idea chiave è la stessa dell'esempio precedente. L'applicazione consiste in un file "***demo-sso-php.zip***" da decomprimere nella document root folder di un web server già configurato per l'utilizzo con PHP.

L'applicazione è in totale composta da 3 script, il più semplice dei quali è "***user.class.php***" che rappresenta il contenitore dei dati dell'utente loggato

```
<?php
class User
{
	public $userName='';
	public $firstName='';
	public $lastName='';
	public $email='';
	public $fiscalCode='';
	public $roles='';

	function __construct(){}
}
?>
```

Lo script "***index.php***" della cartella principale contiene semplicemente il link alla risorsa protetta

```
<!DOCTYPE html>
<html>
	<head>
		<title>Demo SSO</title>
	</head>
	<body>
<h1>Demo SSO</h1>

<br/>

<a href="protected/index.php">Area riservata</a>
	</body>
</html>
```

Lo script più interessante è invece quello che coinvolge la richiesta di risorsa protetta "***/protected/index.php***":

```
<?php
session_start();
include_once("../user.class.php");

if(!isset($_SESSION['loggedUser'])){
	$user = new User();
	$user->userName=$_SERVER['saml_subject_username'];
	$user->firstName=$_SERVER['saml_attribute_nome'];
```

```
        $user->lastName=$_SERVER['saml_attribute_cognome'];
        $user->email=$_SERVER['saml_attribute_emailaddress'];
        $user->fiscalCode=$_SERVER['saml_attribute_codicefiscale'];
        $user->roles=$_SERVER['saml_attribute_role'];
        $_SESSION['loggedUser'] = serialize($user);
}
$userData = unserialize($_SESSION['loggedUser']);
?>
<!DOCTYPE html>
<html>
        <head>
                <title>Demo SSO</title>
        </head>
        <body>
                <h3>Utente loggato:</h3>
                <table border="1">
                        <tr>
                           <td>Username</td>
                           <td><?php echo $userData->userName;?></td>
                        </tr>
                        <tr>
                           <td>Nome</td>
                           <td><?php echo $userData->firstName;?></td>
                        </tr>
                        <tr>
                           <td>Cognome</td>
                           <td><?php echo $userData->lastName;?></td>
                        </tr>
                        <tr>
                           <td>Email</td>
                           <td><?php echo $userData->email;?></td>
                        </tr>
                        <tr>
                           <td>Codice Fiscale</td>
                           <td><?php echo $userData->fiscalCode;?></td>
                        </tr>
                        <tr>
                           <td>Ruoli</td>
                           <td>
                              <?php
                               $ruoli = explode(",", $userData->roles);
                               for($i=0;$i<count($ruoli);$i++) {
                                  print $ruoli[$i]."<br>";
                               }
                              ?>
                           </td>
                        </tr>
                </table>
        </body>
</html>
```

Lo script, in maniera del tutto analoga all'esempio Java, recupera le informazioni inserite in request header dal modulo shibboleth del web server attraverso la variabile globale PHP **$_SERVER** ed inserite in un oggetto in sessione se non esiste in precedenza.
Lo script in fase di render visualizza i valori in una tabella, recuperandoli dall'oggetto nella sessione corrente attraverso la variabile PHP **$_SESSION**
Si fa presente che è opportuno verificare il comportamento del proprio ambiente rispetto all'utilizzo

degli header che contengono il carattere "_"*.[19]

8.2.1 Gestione degli header contenenti il carattere "_"

Nel caso in cui venga utilizzato come webserver il software NGINX, l'abilitazione degli header può essere effettuata con la seguente direttiva:

underscores_in_headers on;

Nel caso in cui invece venga utilizzato come webserver il software Apache HTTP Server (2.4.x), è possibile invece – abilitando il modulo mod_headers – indicare le seguenti direttive (è necessario, in questo caso, inserire le direttive per ciascun header rievuto da GEL):

SetEnvIfNoCase ^saml.attribute.username$ ^(.)$ saml_attribute_username =$1*
RequestHeader set saml_attribute_username %{saml_attribute_username}e env=saml_attribute_username

SetEnvIfNoCase ^saml.attribute.nome$ ^(.)$ saml_attribute_nome=$1*
RequestHeader set saml_attribute_nome %{saml_attribute_nome}e env=saml_attribute_nome

SetEnvIfNoCase ^saml.attribute.cognome$ ^(.)$ saml_attribute_cognome=$1*
RequestHeader set saml_attribute_cognome %{saml_attribute_cognome}e env=saml_attribute_cognome

SetEnvIfNoCase ^saml.attribute.emailaddress$ ^(.)$ saml_attribute_emailaddress=$1*
RequestHeader set saml_attribute_emailaddress %{saml_attribute_emailaddress}e
env=saml_attribute_emailaddress

SetEnvIfNoCase ^saml.attribute.codicefiscale$ ^(.)$ saml_attribute_codicefiscale=$1*
RequestHeader set saml_attribute_codicefiscale %{saml_attribute_codicefiscale}e
env=saml_attribute_codicefiscale

SetEnvIfNoCase ^saml.attribute.role$ ^(.)$ saml_attribute_role=$1*
RequestHeader set saml_attribute_role %{saml_attribute_role}e env=saml_attribute_role

N.B. Questa sezione è inserita a solo scopo indicativo per descrivere quali operazioni possono essere effettuate per abilitare l'utilizzo degli header rilasciati da GEL/Shibboleth/SP nel caso in cui non vengano ricevuti dall'applicativo a causa delle limitazioni imposta dai webserver nella ricezione di request header contenenti il carattere "_". Restano in capo a chi gestisce il sistema assicurare che non vengano introdotte problematiche di sicurezza

[19] Problemi che potrebbero inficiare il corretto passaggio degli attributi all'applicazione:

- Potrebbe essere abilitato sul web server dell'applicazione PHP il blocco degli header i cui nomi contengono il carattere "_". È una impostazione di sicurezza.
- Altra impostazione riguarda il prefisso che Apache/PHP aggiunge agli header, negli esempi del kit fornito si fa riferimento a nomi di attributi come ad esempio: "saml_attribute_emailaddress", con il prefisso tale attributo diventerebbe "HTTP_saml_attribute_emailaddress".