Authentication e authorization federate nelle cloud: Estensioni a Shibboleth per l'applicazione in contesti di cloud computing

Andrea Biancini[1] , Luca Prete[2] , Simon Vocella[2]

[1]INFN – Sezione di Milano Bicocca, [2]Consortium GARR

Abstract. Molti sistemi informativi fanno uso di tecnologie cloud che sfruttano un paradigma distribuito ed espongono interfacce non necessariamente *web-based*. Contemporaneamente si va diffondendo l'uso di tecnologie per l'autenticazione e l'autorizzazione federate, che invece sono tipicamente web-based. Sono stati fatti vari tentativi per adattare i meccanismi federati alla tecnologia cloud, tuttavia le soluzioni introdotte sono ancora complesse e difficilmente integrabili. L'articolo propone tre nuove estensioni di Shibboleth che permettono: l'integrazione di sistemi non web-based, basati su Java e Python; l'autenticazione di sistemi Linux basati su PAM e NSS; l'integrazione del protocollo per l'archiviazione dati nelle cloud Amazon S3. Dopo un breve confronto con le precedenti tecnologie, si discute in dettaglio l'architettura di tali moduli, quanto sia già possibile fare e gli sviluppi futuri.

1. Introduzione

Oggi gli ambienti informatici fanno sempre più uso di applicazioni distribuite in differenti domini applicativi. In questi ambienti l'autenticazione e l'autorizzazione devono offrire meccanismi di *Single Sign-On* (SSO)[1]. GARR promuove un approccio federato alla gestione delle identità degli utenti attraverso la Federazione IDEM [2], che sfrutta l'implementazione di SAML 2 [3] offerta da Shibboleth [4]. Gli approcci di *Authentication and Authorization Infrastructure* (AAI) ben si applicano agli emergenti paradigmi di cloud computing, termine con il quale si è soliti indicare l'insieme di modelli di accesso a risorse informatiche (capacità computazionale, storage, applicazioni, ecc.) on demand attraverso Internet. Grazie alla virtualizzazione, i sistemi cloud fanno della trasparenza la loro caratteristica fondamentale. Per trasparenza s'intende la capacità di astrarre dallo strato fisico, disaccoppiando la gestione delle risorse fisiche dal loro utilizzo. Una soluzione cloud per l'università e la ricerca dovrebbe porsi l'obiettivo di estendere i modelli di federazione esistenti, sfruttandone i benefici.

Nell'ambito *storage cloud*, sono nate in Italia esperienze relative alla realizzazione di un servizio di questo tipo [5]. Dall'esperienza progettuale sono emerse alcune limitazioni tecnologiche di IDEM per il supporto specifico alle soluzioni cloud:

1. Cloud, per definizione, è multiprotocollo, quindi non accessibile solo da *web browser*. In questo caso, l'approccio standard offerto da Shibboleth può essere limitante e poco flessibile per applicazioni *mobile* e *client*.
2. In ambito cloud sono emersi nuovi protocolli, diventati standard *de facto*. Alcuni di essi descrivono e disciplinano gli aspetti di autenticazione e autorizzazione. I meccanismi federati di Shibboleth devono offrire interfacce compatibili con questi nuovi standard.

Tali aspetti saranno discussi dettagliatamente nel seguito dell'articolo, che è organizzato come segue: la prima sezione contiene una breve panoramica dei lavori correlati ai temi di autenticazione e autorizzazione federata; la seconda presenta la soluzione realizzata; la terza sezione presenta alcuni dettagli sulla realizzazione tecnica del software; l'ultima sezione presenta le conclusioni e i possibili sviluppi futuri.

2. Lavori correlati

Per l'accesso a schemi d'identità federata da

Workshop GARR Calcolo e Storage Distribuito - Selected Papers. Roma 29-30 novembre 2012
ISBN: 978-88-905077-4-8 © 2013 Associazione Consortium GARR

parte di applicazioni non basate sul web, sono in corso di realizzazione diverse iniziative. In particolare nell'ambito del software Shibboleth, il progetto Moonshot [6] si prefigge di affrontare questo problema. Il progetto mira esplicitamente a sviluppare una singola tecnologia che sia in grado di estendere i benefici delle federazioni d'identità a servizi non basati sul web.

La soluzione proposta da Moonshot è tanto completa quanto articolata. Essa si avvale di diverse tecnologie: Kerberos (per l'autenticazione in ambiete distribuito), le librerie *Generic Security Services* (GSS) e un server *Remote Authentication Dial In User Service* (RADIUS). Per quanto la soluzione sia valida e affidabile, essa si basa su un'architettura piuttosto complessa e richiede notevoli cambiamenti sulle macchine client, sul *Service Provider* (SP) e l'*Identity Provider* (IdP). Ciò rischia di rendere l'approccio di difficile adozione nelle federazioni esistenti.

La necessità di sfruttare federazioni d'identità Shibboleth in applicazioni non basate su web è particolarmente sentita nei servizi di Grid Computing [7], ispirati dall'articolo originario di Foster [8]. Questi ambienti hanno introdotto soluzioni specifiche ai problemi di sicurezza che differiscono da quelli proposti dal sistema SSO di Shibboleth. A causa della complessità del progetto Moonshot, l'integrazione dei due schemi di sicurezza è stata realizzata con soluzioni differenti: ad esempio quelle proposte da Wang [9] e Jensen [10] permettono agli utenti di spostarsi in modo trasparente tra applicazioni Shibboleth e ambienti Grid utilizzando le stesse credenziali. Queste soluzioni riescono a facilitare l'esperienza utente e a fornire risultati efficaci. Tuttavia si deve rilevare come loro non siano in grado di fornire soluzioni alle esigenze di AAI per servizi non basati sul web.

A oggi, la letteratura offre scarse informazioni sullo sviluppo di meccanismi d'integrazione di AAI cloud in federazioni già esistenti. Il Cloud computing si è sviluppato come paradigma autonomo, implementato attraverso soluzioni verticali molto specifiche, quindi scarsamente integrabili con il pregresso tecnologico. Tra le iniziative sviluppate intorno a Shibboleth per implementare meccanismi di autenticazione diversi e non basati solamente su *username* e *password*, nessuna ha preso ancora direttamente in considerazione i nuovi protocolli sviluppati in ambito Cloud.

3. Soluzioni proposte

Per risolvere i problemi descritti sono state realizzate, in ambito cloud federato per il mondo dell'università e della ricerca, tre estensioni di Shibboleth:

1. Autenticazione e autorizzazione per applicazioni non web-based (Java e Python).

Permette di sfruttare i meccanismi di autenticazione tipici delle realtà federate anche in applicazioni non basate sull'uso di un browser internet, quali ad esempio applicazioni desktop tradizionali. La soluzione garantisce il SSO, accedendo a risorse web da applicazioni che ne integrino le API. I meccanismi di autenticazione Shibboleth sono stati implementati in una libreria che comunica con il SP e l'IdP, utilizzando HTTPS e *Basic Authentication*.

La libreria è stata sviluppata sia per Java, come modulo JAAS [11], sia per Python, a oggi i linguaggi principali per realizzare applicazioni utente ad alto livello.

2. Autenticazione e autorizzazione di sistemi Linux (tramite PAM e NSS).

Questa estensione sfrutta un meccanismo simile a quello descritto nel punto precedente. Anch'essa è basata su HTTPS e *Basic authentication* e permette l'autenticazione di utenti in sistemi Linux attraverso Shibboleth. Grazie ai moduli sviluppati, in fase di login, SP e IdP sono contattati per autenticare l'utente e ottenere i dati necessari alle logiche di autorizzazione. In tal modo il sistema e le applicazioni riconoscono gli utenti della federazione trasparentemente e senza nessuna ulterior modifica.

Inoltre, anche applicativi come SSH, NFS o Apache, ereditano la possibilità di autenticare gli utenti attraverso Shibboleth. Ciò accade poiché molte applicazioni Linux delegano i compiti di autenticazione al sistema operativo attra-

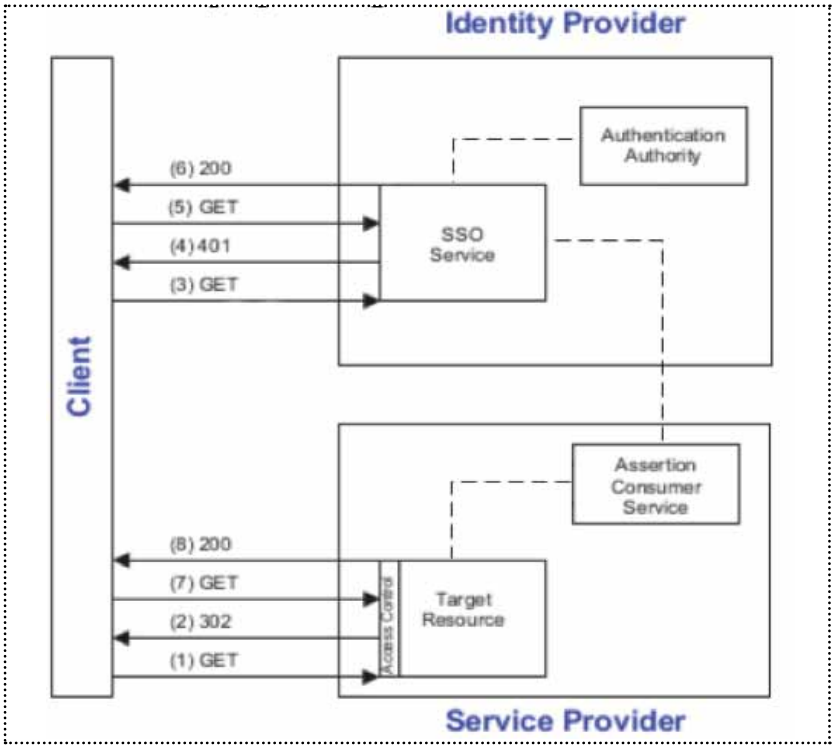

Fig 1 - Procedura di login di Shibboleth

verso i moduli PAM, aggiunti ad hoc nell'implementazione dell'estensione.

Per la realizzazione dell'estensione sono stati implementati degli specifici moduli per PAM e NSS, i meccanismi standard di user authentication e naming service di Linux

3. Autenticazione e autorizzazione tramite lo schema di sicurezza del protocollo S3.

Questa estensione permette di effettuare l'autenticazione Shibboleth di utenti che utilizzino il protocollo S3 [12] per la gestione dei dati. Il protocollo S3 è stato sviluppato da Amazon per il proprio servizio di storage cloud ed è oggi uno standard de facto, sia in ambito commerciale sia Open Source.

L'autenticazione di S3 non avviene tramite l'uso di username e password, ma attraverso la condivisione di un segreto tra l'utente e l'applicazione server. A ogni richiesta, il client cripta una stringa con una chiave segreta condivisa con il server, che non è mai trasmessa o distribuita. Il server, eseguendo la stessa operazione, confronta i risultati della cifratura autenticando l'utente in caso di corrispondenza. Per integrare tale modello di autenticazione in Shibboleth è stato sviluppato un nuovo modulo che permetta di verificare le credenziali utente utilizzando il segreto condiviso nel modo normato dal protocollo.

3.1 Architettura

La soluzione proposta sfrutta i meccanismi di autenticazione e autorizzazione di Shibboleth [13], permettendo di eseguire un login all'IdP usando il meccanismo Basic HTTP [14]. Il diagramma delle interazioni è quello mostrato in Figura 1.

Tuttavia sono necessarie le seguenti precisazioni:

1. La richiesta originale è indirizzata a una pagina web sul SP che si trova dietro autenticazione Shibboleth. La pagina produce un elenco di righe chiave=valore contenente le informazioni degli attributi da inserire nella sessione utente.
2. Il server web sul SP risponde con un redirect ad una pagina dell'IdP.
3. Il client segue il reindirizzamento e apre la pagina di login dell'IdP.
4. La richiesta ottiene una risposta HTTP 401 ("autorizzazione richiesta") e richiede l'autenticazione tramite HTTP Basic.
5. Il client esegue la stessa richiesta, specificando nell'header HTTP il nome utente e la password, verificati dall'IdP.
6. L'IdP interagisce conil SP per creare una sessione valida contenente gli attributi della sessione utente.
7. Il client apre l'URL originale, fornendo il cookie ottenuto dall'IdP dopo l'autenticazione. Ottiene quindi la pagina con le righe chiave=valore e inizializza la sessione utente sul client.

Il meccanismo simula ciò che accade all'interno del browser durante un'autenticazione Shibbo-

leth via web. Tutte le interazioni HTTP sono tuttavia automatiche e sono "nascoste" all'utente.

4. Implementazioni

Le implementazioni delle soluzioni descritte sono raggruppabili in tree aree: le implementazioni Java e Python, l'implementazione PAM e NSS e l'implementazione per l'autenticazione S3. Nel corso del capitolo sono presentati i dettagli relativi ai tre moduli realizzati.

4.1 Java e Phyton

Java, nella versione enterprise, è divenuta una piattaforma molto conosciuta e utilizzata per realizzare differenti applicazioni. Da Java 1.4 è stato introdotto un insieme di servizi per l'implementazione di meccanismi AAI: J*ava Authentication and Authorization Service* (JAAS, [11]). JAAS può essere utilizzato per due scopi:

- Autenticare gli utenti per determinare in modo affidabile e sicuro chi sta eseguendo il codice Java, indipendentemente dal fatto che il codice sia eseguito come applicazione *client*, *applet* o *servlet*;
- Autorizzare gli utenti e verificare che abbiano opportuni diritti di accesso.

Per la sua natura modulare, JAAS può essere esteso per integrare meccanismi di autenticazione differenti. Grazie al modulo realizzato è possibile integrare l'autenticazione Shibboleth e la gestione degli attributi nelle applicazioni Java anche non web-based. Il modulo JAAS implementato segue lo schema descritto nella sezione 3.1 per la validazione delle credenziali utente. Internamente, esso implementa uno ShibbolethPrincipal che contiene una mappa associativa degli attribute Shibboleth dell'utente e che, quindi, rappresenta la sessione di login dell'utente stesso.

Come ulteriore esempio, è stata sviluppata una libreria per l'integrazione del meccanismo di AAI descritto con applicazioni Python. Python è un linguaggio di programmazione ad alto livello utilizzato per lo sviluppo di applicazioni client. Differentemente da Java, Python non offre un insieme di API per gestire l'autenticazione e l'autorizzazione. Quindi, il modulo che realizza l'integrazione di Shibboleth è stato implementato da zero, senza implementare nessun *framework* di sicurezza preesistente. Tale modulo è contenuto nel *package shibauth* ed espone un metodo login che restituisce il nome di un utente autenticato e un dizionario contenente gli attributi dell'utente per la sessione Shibbolet corrente. La realizzazione di un'applicazione che sfrutti il modulo è quindi semplice e immediata.

4.2 Moduli PAM e NSS

I meccanismi standard utilizzati dai moderni sistemi Linux per autenticare gli utenti sono PAM e NSS. Le due librerie hanno diversi scopi e funzionalità. *Pluggable Authentication Modules* (PAM) è il cuore dei meccanismi di autenticazione e permette ai programmi di autenticare trasparentemente gli utenti, come descritto nella Linux-PAM *System Administrator's Guide* [15]. La libreria è stata estesa con l'implementazione di un modulo specifico al fine di garantire il SSO su macchine Linux. Il modulo realizzato autentica utilizzando HTTP Basic secondo lo schema generale descritto nella sezione 3.1. Esso utilizza le librerie *libcurl [*16] per gestire i dialoghi HTTP e i *cookie* al fine di effettuare l'autenticazione HTTP Basic. Un modulo PAM è uno *shared object* con un'interfaccia pubblica specificata negli header di PAM, collegata ai programmi utente per compiere le operazioni di autenticazione, accounting e per recuperare i valori da inserire nella sessione utente. Il modulo PAM sviluppato può essere utilizzato nelle catene PAM presenti sui sistemi Linux. Per esempio, modificando il file */etc/pam.d/login* e aggiungendo il modulo Shibbolet realizzato, è possibile permettere il login alla macchina Linux a utenti definiti su un IdP federato.

Per consentire a un sistema Linux di utilizzare directory esterne per gli utenti e i gruppi, è necessario implementare un altro modulo per la libreria N*ame Service Switch* (NSS), [17]. Questa libreria permette di definire i servizi per l'accesso a database contenenti diverse informazioni. Per quanto riguarda le attività descritte nell'articolo, i database rilevanti sono: passwd, con-

tenente gli utenti e i *group*, database con tutti i gruppi. Il servizio realizzato per integrare NSS con Shibboleth è basato sull'adozione di una *servlet*, distribuita sull'IdP. La servlet si connette al database di autorizzazione sottostante (di solito un database LDAP) e recupera le informazioni sugli utenti e i gruppi riconosciuti dall'IdP.

Grazie ai nuovi moduli, Linux riconosce gli utenti Shibboleth nello stesso modo di quelli locali. Un utente può accedere alla macchina Linux con le proprie credenziali Shibboleth, avere diritto di accesso ai file, utilizzarne le ACL e così via. Dopo il login, l'utente trova nel suo ambiente tutti i valori della sessione Shibboleth: gli attributi scaricati, infatti, sono inseriti nella sessione e possono essere utilizzati per eseguire SSO con altre applicazioni non web-based.

4.3 Integrazione con il protocollo S3

Amazon Simple Storage Service (S3) è un servizio cloud che permette l'archiviazione dei dati e files sul web. Esso rientra nell'insieme di servizi detti *Amazon Web Services* (la cloud realizzata da Amazon) e si presenta come uno standard *de facto*. Il protocollo S3 basa la comunicazione tra client e server su chiamate REST e, oltre agli aspetti di trasferimento e accesso ai file, definisce anche quelli relativi all'autenticazione. Per la parte di sicurezza, S3 sfrutta un meccanismo basato sulla condivisione di una chiave segreta (la cosiddetta *secret key*). A ogni chiamata S3, il client non scambia direttamente il segreto condiviso, ma crea un *token* ottenuto criptando un messaggio (la *canonical string*) attraverso un algoritmo noto e usando la *secret key* come chiave. La stringa criptata è inviata al server che è in grado di eseguire le stesse operazioni (utilizzando la stessa *secret key* condivisa con il client), quindi di autenticare l'utente.

Il meccanismo descritto non si presta a essere facilmente integrato con Shibboleth, che utilizza maschere di login che richiedono all'utente di inserire i propri nome utente e password. Per ovviare a questo problema è stato realizzato un nuovo modulo *ad hoc*, che permette a Shibboleth di verificare l'identità degli utenti secondo quanto stabilito nel protocollo S3. L'IdP è in grado di generare la *secret key* partendo da diversi attributi LDAP dell'utente (tra cui l'*hash* criptato e la sua password) e di comunicarlo all'utente stesso tramite l'invio di una mail. A questo punto il client opera in modo trasparente, senza nessuna modifica al suo comportamento standard, criptando la canonical string con la *secret key*. Il modulo realizzato sull'IdP, quindi, implementa un nuovo *Login Handler* specifico, che è in grado di processare le richieste in arrivo per autenticarle. Ad ogni richiesta da parte del client questo modulo genera nuovamente la secret key, cripta la canonical string e confronta il risultato con quanto inviato dal client. Se le due stringhe coincidono, allora l'utente è in possesso della secret key corretta e quindi può essere autenticato. In tal modo, l'implementazione realizzata è totalmente trasparente per i client e gli utenti possono continuare ad usare i tradizionali client S3. Tuttavia l'autenticazione avviene a questo punto ad opera di un IdP Shibboleth opportunamente esteso e quindi preservando l'architettura e gli schemi di identità federati in essere.

5. Conclusioni e sviluppi futuri

L'articolo ha descritto come siano state realizzate delle estensioni ai meccanismi di autenticazione e autorizzazione degli utenti per meglio adattare le federazioni d'identità all'erogazione di servizi cloud. In particolare le estensioni realizzate hanno mirato a garantire l'accesso alle risorse cloud, sia tramite il web, sia attraverso applicativi client non basati sul web. Un'ulteriore estensione è stata realizzata per permettere l'integrazione di schemi di autenticazione standard in ambito cloud all'interno di federazioni d'identità esistene. Come esempio per questo secondo punto, è stata descritta l'implementazione di un modulo S3 per l'autenticazione degli utenti all'interno di Shibboleth tramite il protocollo definito da Amazon.

Utilizzando le estensioni descritte è possibile implementare servizi cloud che sfruttino ed estendano tutti i meccanismi e le regole prescrit-

te da IDEM, la federazione di identità adottata dal mondo accademico e della ricerca in Italia. Il lavoro svolto rappresenta un primo passo verso meccanismi più complessi di estensione delle federazioni d'identità ai modelli e alle tecnologie cloud. Sebbene il lavoro descritto risolva già le esigenze specifiche di alcuni progetti in ambito di erogazione di servizi cloud, sono indubbiamente necessarie ulteriori indagini ed attività per raggiungere un livello di servizio migliore, in particolare:

- Supporto a WAYF. Le estensioni proposte non supportano il protocollo WAYF, per contattare l'IdP dell'ente di affiliazione dell'utente a seguito della verifica della sua identità. Occorre una nuova estensione per effettuare il discovery dei provider d'identità, utilizzando i profili descritti da SAML 2.
- Gestione multidominio per l'autenticazione in sistemi Linux. Qualora l'estensione per l'autenticazione utente su macchine Linux dovesse essere utilizzata in contesti federati di grandi dimensioni, sarebbe necessario risolvere un problema di mappatura degli utenti. I sistemi Linux associano a ogni utente uno *user id* numerico univoco. Gli id utente utilizzati dagli IdP dovrebbero essere associabili biunivocamente agli identificativi utenti dei sistemi.
- *Accounting*. Occorre capire come coniugare gli strumenti di monitoring dei singoli servizi cloud con quanto offerto dalle federazioni d'identità, ad esempio come gestire in modo federato l'uso di risorse virtuali fino all'esaurimento delle disponibilità assegnate a un utente.
- Performance. Relativamente al modulo per l'integrazione del protocollo S3, in ambiente di produzione è necessario introdurre meccanismi di *caching* sicuri sul SP. In tal modo non sarà necessario, una volta autenticato il client, dover contattare nuovamente l'IdP per una nuova richiesta di autenticazione.

Riferimenti bibliografici

[1] Shaer, C. 1995. Single sign-on. Network Security 1995, 8, 11–15

[2] https://www.idem.garr.it/

[3] Cantor, S., Kemp, J., Philpott, R., and Maler, E. 2005. Assertions and protocols for the OASIS security assertion markup language (SAML) v2.0. Tech. rep. 03.

[4] Morgan, R. L., Cantor, S., Carmody, S., Hoehn, W., and Klingenstein, K. 2004. Federated security: The shibboleth approach. EDUCAUSE Quarterly 27, 4, 12–17.

[5] Valli, C., Biancini, A., Reale, M., Farina, F., Vocella, S., Galeazzi, F. (2012). GARR Cloud Storage GARRBox. Proceedings of TICAL 2012.

[6] Howlett, J. and Hartman, S. (2005). Project Moonshot. Tech. rep. 07

[7] Kesselman, C. and Foster, I. 1998. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers

[8] Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. 1998. A security architecture for computational grids. In Proceedings of the 5th ACM conference on Computer and communications security. CCS '98. ACM, 83–92

[9] Wang, X. D., Jones, M., Jensen, J., Rirchards, A., Wallom, D., Ma , T., Frank, R., Spence, D., Young, S., Devereux, C., and Geddes, N. 2009. Shibboleth access for resources on the national grid service (sarongs). In Proceedings of the 2009 Fifth International Conference on Information Assurance and Security - Volume 02. IAS '09. IEEE Computer Society, 338–341

[10] Jensen, J., Wallom, D., Spence, D., Tang, K., Meredith, D., and Trenthefen, A. 2007. Shibgrid, a shibboleth based access method for the national grid service. In UK e-Science All Hands Meeting

[11] http://java.sun.com/products/jaas

[12] http://aws.amazon.com/s3

[13] Erdos, M. and Cantor, S. 2005. The Shibboleth architecture. http://shibboleth.internet2.edu

[14] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and Ste-

wart, L. 1999. Http authentication: Basic and digest access authentication. RFC 2617.

[15] Morgan, A. G. and Kukuk, T. 1996. The Linux-pam system administrators' guide. http://kernel.org

[16] Stenberg, D. 1996. curl and libcurl. http://curl.haxx.se

[17] http://www.gnu.org

Andrea Biancini
andrea.biancini@mib.infn.it

Ha sviluppato la carriera all'interno di società dei settori Finance e Information Technology ove si è occupato di: gestione progetti IT, pianificazione e governo su temi di controllo e gestione (budget/costi), gestione del portafoglio progetti.

Da sempre interessato alla dimensione umana, ha organizzato eventi formativi e corsi. Sta concludendo un secondo corso di laurea in psicologia che mi sta permettendo di sviluppare ulteriori competenze.

Luca Prete
luca.prete@garr.it

Svolge da diversi anni attività di consulenza in campo informatico, con particolare attenzione ai sistemi e alle reti. Interessato al networking, alle tecnologie cloud e quelle di virtualizzazione. Da poco più di un anno collabora con GARR occupandosi di Software Defined Networking.

Simon Vocella
simon.vocella@garr.it

lavora da diversi anni come software developer. Interessato a nuove tecnologie emergenti, in particolare a sistemi distribuiti e tecnologie cloud. Ha collaborato con GARR lavorando nei progetti europei FEDERICA e NOVI e nel progetto cloud storage GARRbox.