UNIVERSITÀ DEGLI STUDI ROMA TRE
FACOLTÀ DI SCIENZE M.F.N.
CORSO DI LAUREA IN MATEMATICA

Tesi di Laurea Magistrale in Matematica

Modelli, algoritmi e strumenti per la sicurezza dei sistemi informativi complessi

Sintesi

Candidato
Laura Tiburzi

Relatore
Prof. Marco Liverani

ANNO ACCADEMICO 2009–2010
Maggio 2011

Classificazione AMS: 68A05, 68M01, 68R05

Parole chiave: Sistemi informativi, Sicurezza informatica, Autenticazione, Autorizzazione, RBAC

Modelli, algoritmi e strumenti per la sicurezza dei sistemi informativi complessi

Il sistema informativo di un'organizzazione è l'insieme dei dati, degli strumenti, dei modelli tecnici ed organizzativi e dei protocolli applicativi attraverso i quali viene utilizzata l'informatica all'interno dell'azienda con l'obiettivo di distribuire l'informazione a coloro ai quali è necessaria, garantendo che sia accessibile nelle forme e nei tempi opportuni. I sistemi informativi rappresentano oggi uno strumento essenziale per lo svolgimento delle attività in ogni tipo di organizzazione.

Nel mio lavoro, partendo da una descrizione delle diverse tipologie di architetture informatiche presenti in un sistema informativo, a partire dalle più semplici, come quella di un un computer connesso ad una rete locale, ma utilizzato come ausilio alla elaborazione di dati residenti su un'unitá di memoria "locale", o quella dei grandi sistemi centralizzati, fino a quelle più complesse e moderne, come le architetture dei sistemi distribuiti, dei *Web Information System* o le *Service Oriented Architecture* (SOA), ho affrontato il tema della sicurezza di tali sistemi.

Con il crescere della complessità di un sistema informativo aziendale e delle componenti informatiche che ne fanno parte, che spesso sono eterogenee, basate su linguaggi, prodotti e protocolli diversi e talvolta incompatibili, diventa di cruciale importanza rendere sicuro il sistema informativo stesso. Questo significa adottare delle politiche di sicurezza e gli strumenti informatici necessari per attuarle, tali da garantire che tutti e soli gli operatori autorizzati possano accedere alle informazioni di loro competenza, con le modalità richieste dal loro ruolo aziendale/istituzionale.

Rendere sicuro un sistema informatico, infatti, significa non solo proteggerlo da

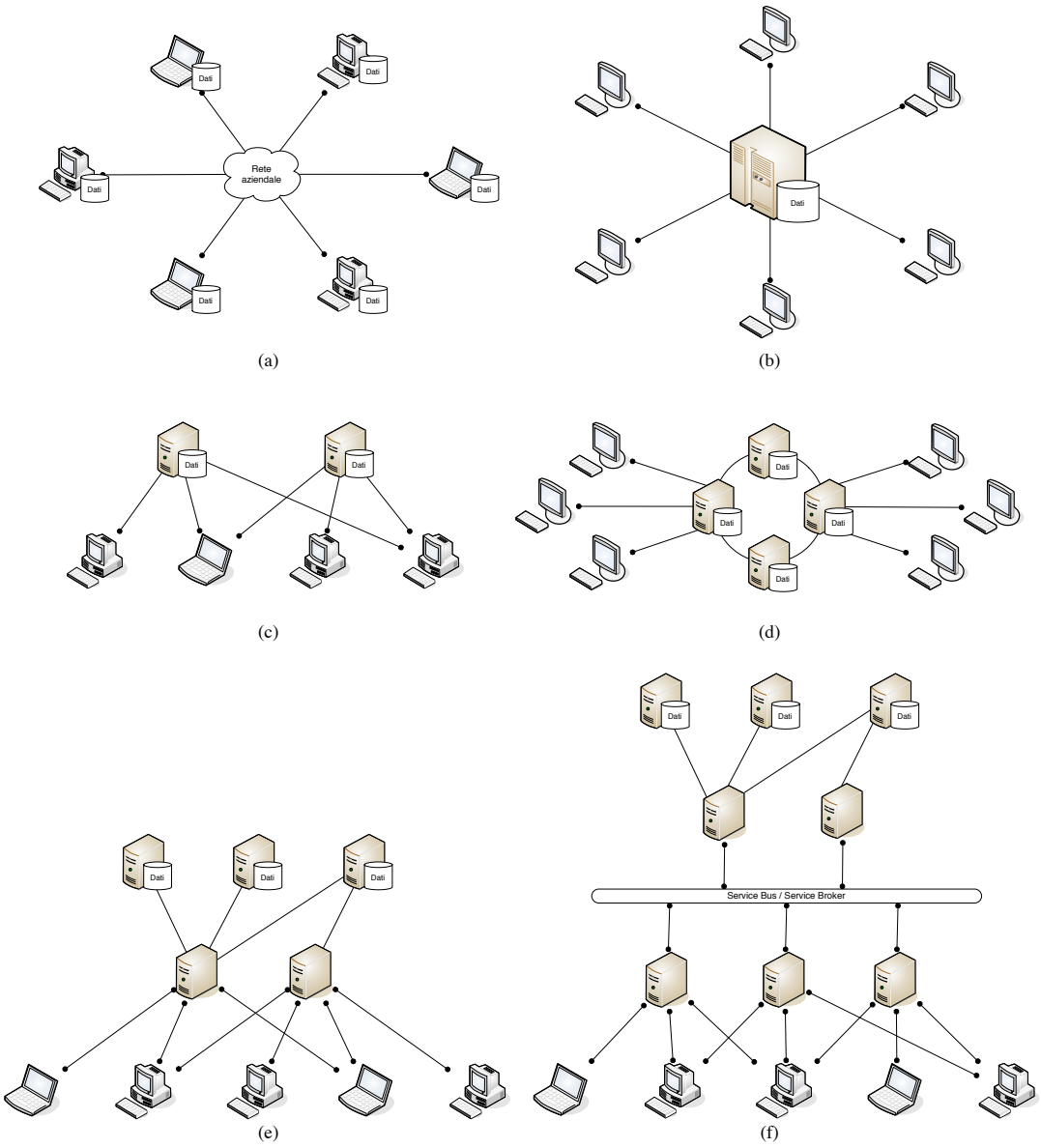

Figura 1: Schematizzazione dei diversi modelli architetturali che si possono ritrovare nella struttura di un sistema informativo complesso ed eterogeneo: (a) una rete locale di personal computer; (b) un grande sistema multiutente centralizzato; (c) un'architettura client/server; (d) un sistema distribuito; (e) un'infrastruttura web-based; (f) un'architettura orientata ai servizi (SOA)

malfunzionamenti interni o da virus informatici, ma anche definire dei processi che indirizzino la modalità d'uso del sistema stesso da parte degli utenti, in modo tale da garantire anche la riservatezza delle informazioni e la loro integrità.

Per tutelare il sistema informativo da minacce fisiche o logiche, che mirano ad alterare l'organizzazione logica del sistema con l'obiettivo di sottrarre i dati, effettuare accessi non autorizzati alle applicazioni, alterare la normale esecuzione dei programmi, sono state progettate strategie operative e tecnologie mirate alla difesa del sistema da attacchi provocati da sorgenti esterne; tuttavia spesso le minacce più dannose e pericolose sono quelle effettuate dall'interno del sistema informativo, da personale interno all'organizzazione proprietaria del sistema informatico, autorizzato ad accedere alle parti più vulnerabili del sistema. Mentre per contrastare le minacce esterne possono essere sufficienti contromisure di tipo tecnico, per quelle interne sono necessarie procedure organizzative specifiche. Per poter sviluppare tali procedure organizzative è necessario introdurre i concetti di *autenticazione* e *autorizzazione* di un utente, sulla base di credenziali assegnate personalmente ad ognuno, per consentire l'accesso alle applicazioni del sistema in base al profilo autorizzativo associato a ciascun utente e al ruolo che egli assume all'interno dell'organizzazione.

Nel mio lavoro, dunque, dopo aver introdotto le principali architetture dei sistemi informativi aziendali ed i criteri generali volti a garantire la sicurezza di tali sistemi, ho affrontato lo studio di una classe di strumenti software, i cosiddetti sistemi di *identity and access management*, disponibili da alcuni anni per contribuire al consolidamento della sicurezza dei sistemi informativi complessi ed eterogenei. In particolare con strumenti di questo genere si riesce a realizzare pienamente il criterio di controllo degli accessi basato sui ruoli (RBAC – *role based access control*), un approccio piuttosto moderno al tema del controllo degli accessi alle risorse informatiche di un sistema; su tale aspetto si è concentrata l'ultima parte del mio lavoro.

Autenticazione ed Autorizzazione

Rendere sicuro un sistema informativo significa innanzi tutto garantire che le proprietà di integrità, autenticità, non ripudio, autorizzazione degli accessi ai dati e riservatezza delle informazioni, siano sempre rispettate. Gli "attacchi" ad un sistema informativo, condotti con tecniche ben note, come lo *sniffing*, l'*address spoofing*, il *data spoofing*, i

denial-of-service (DOS) ed altre ancora, sono sempre finalizzati a compromettere una o più d'una delle proprietà di sicurezza sopra elencate. Per evitare tali attacchi si utilizzano, tra le altre, tecniche crittografiche oggi ben consolidate, come l'*hashing*, la *crittografia simmetrica*, la *crittografia asimmetrica*, l'autenticazione basata su certificati digitali.

L'accesso ad informazioni riconducibili a persone, come i dati anagrafici o dati altamente "sensibili" (appartenenza ad organizzazioni politiche, dati sanitari o giudiziari riferiti a persone fisiche, ecc.), deve essere regolamentato in modo da garantire la riservatezza e limitare la possibilità di acquisire e diffondere tali dati. In Italia nel giugno del 2003 è stato emanato il nuovo "Testo Unico sulla Privacy", rappresentato dal D.lgs 196/2003 [11] e dai suoi allegati: questa normativa, che costituisce un importantissimo riferimento ed un punto di partenza fondamentale per quanti si occupano di sicurezza delle informazioni, stabilisce, tra l'altro, i requisiti tecnici di base nell'ambito della sicurezza informatica, di cui devono essere dotati i sistemi informativi, nel caso in cui vengano utilizzati per l'elaborazione di dati personali.

L'autenticazione è un servizio di sicurezza che si realizza attraverso una serie di meccanismi di verifica dell'identità dell'utente che richiede di utilizzare un determinato servizio attraverso un sistema informatico. Il processo di autenticazione verifica l'identitá dell'utente basandosi su sue informazioni segrete o su meccanismi di verifica che forniscono un elevato grado di garanzia sulla veridicità dell'identità dichiarata dall'utente. L'autenticazione, in una comunicazione informatica tra due interlocutori, può essere effettuata verificando che la controparte possieda almeno uno dei seguenti requisiti "di base", ordinati in base al loro grado di "robustezza":

1. conoscenza esclusiva di una certa informazione, come una *password*;
2. possesso esclusivo di un oggetto, come ad esempio una carta magnetica;
3. possesso di una o più caratteristiche fisiche, misurabili con appositi sensori, che lo differenzino da tutti gli altri soggetti, come ad esempio l'impronta digitale o l'immagine della retina dell'occhio, in una persona fisica;

L'autenticazione si definisce "debole" se prevede l'utilizzo di un singolo requisito tra quelli sopra elencati, solitamente è il primo di essi. L'autenticazione è "forte" (*strong authentication*) quando è realizzata combinando almeno due dei precedenti criteri di base.

Questo aspetto, che certamente è basilare per l'utilizzo dei servizi più delicati, da solo non basta: oltre alla necessità di verificare con certezza che le parti coinvolte nel processo di "interazione informatica" siano veramente chi dicono di essere, è necessario verificare quali siano le operazioni che sono autorizzate a compiere sul sistema informatico a cui hanno avuto accesso, ovvero sorge il bisogno di occuparsi del problema dell'*autorizzazione* e del *controllo degli accessi* alle applicazioni e ai dati da esse gestiti.

In genere, quindi, a ciascun utente di una determinata applicazione informatica viene associato un "profilo autorizzativo", cioè lo spettro delle funzionalità che sono attivabili e fruibili, che consente di identificare "finestre funzionali" differenti per utenti con profili diversi; ciascun profilo abiliterà gli utenti associati ad eseguire determinate funzionalità, tra quelle rese disponibili dall'applicazione, e non altre. Tipicamente, dove non è presente un'infrastruttura che offre il servizio di autenticazione ed autorizzazione, il compito di gestire i profili autorizzativi per l'accesso alle funzionalità e ai dati è demandata alle singole applicazioni. In questo caso, per ciascuna applicazione deve essere definito un profilo di amministratore dell'applicazione stessa, che consenta ad alcuni utenti di gestire l'assegnazione e la revoca delle autorizzazioni agli altri utenti della medesima applicazione.

Con il termine di *role management* si identifica una metodologia adottata per agevolare la gestione delle autorizzazioni, basata sul concetto di *ruolo*. Il *ruolo* è una "costruzione semantica" attribuita ad un utente in base alla funzione di lavoro che esso svolge all'interno di un'organizzazione, ai permessi ad esso concessi, alla sua autorità, e alle responsabilità e qualificazioni lavorative. Il ruolo può rappresentare competenze per attività specifiche, deve definire l'individuo specifico al quale è attribuito, ma deve contemporaneamente esprimere il limite con cui esso può accedere alle risorse del sistema: è quindi costituito da un determinato insieme di utenti e dai permessi ad essi associati. La combinazione di utenti e permessi caratterizzanti un ruolo può cambiare nel tempo.

Una volta stabiliti i ruoli, è possibile creare regole di accesso alle applicazioni informatiche in vari modi. In questo ambito, la *role engineering* è la disciplina che si occupa della definizione dei ruoli basata sui bisogni effettivi degli utenti che operano sul sistema informativo di una determinata organizzazione.

RBAC, *Role Based Access Control* [10], è un metodo con cui viene effettuato il

controllo di accesso in base al *ruolo* attribuito all'utente che chiede di accedere ad un'applicazione. La politica di controllo degli accessi è rappresentata nelle componenti di RBAC tramite relazioni tra ruoli, tra permessi e ruoli, e tra utenti e ruoli. RBAC ha una politica neutrale, risulta infatti adattabile a diversi modelli applicativi, e supporta tre principi basilari di sicurezza: *privilegio minimo*, cioè i permessi assegnati ad un ruolo sono solo quelli strettamente necessari ad assolvere i compiti ad esso associati, la *separazione dei compiti* e l'*astrazione dei dati*. RBAC offre anche un indubbio vantaggio grazie alla possibilità di variare facilmente il controllo degli accessi per venire incontro alle necessità di cambiamento delle organizzazioni. Ad esempio, senza alcun impatto sulla logica applicativa, si possono introdurre nuovi ruoli o modificare quelli esistenti, modificando in questo modo le politiche di accesso alle diverse funzionalità del sistema informatico e ai dati.

Nel mio lavoro ho presentato in sintesi quattro modelli RBAC. In essi le principali entità considerate sono utenti, ruoli, permessi e sessioni: gli *utenti* presi in considerazione sono per semplicità delle persone; il *ruolo* è il nome di una funzione di lavoro, all'interno di un'organizzazione, che descrive l'autorità e la responsabilità conferita al membro del ruolo; il *permesso* è l'abilitazione a svolgere una determinata operazione su uno o più oggetti del sistema, dati e risorse, e può essere anche chiamato *autorizzazione*, *diritto d'accesso*, o *privilegio*; gli utenti infine stabiliscono *sessioni di lavoro* con il sistema informatico, durante le quali possono "impersonare" un sottoinsieme di ruoli tra quelli assegnati, durante una sessione, infatti, un utente può essere associato a ruoli multipli e i permessi ad esso disponibili sono il risultato dell'unione dei permessi provenienti da tutti i ruoli attivati. Ogni sessione è associata ad un singolo utente per tutta la durata della sessione stessa, d'altra parte, un utente può essere simultaneamente collegato al sistema mediante più sessioni di lavoro, dalla stessa postazione di lavoro.

$RBAC_0$ è il modello base nel quale sono presenti *utenti*, *ruoli* e *permessi*.

Definizione 1 *Il modello $RBAC_0$ è composto dalle seguenti componenti:*

- *gli insiemi che possiamo denominare $USERS$, utenti, $PERMS$, permessi, $ROLES$, ruoli, e S, sessioni;*
- *$PA \subseteq PERMS \times ROLES$ è una relazione tra permessi e ruoli, del tipo "molti-a-molti" (PA, permessi d'accesso);*

- *$UA \subseteq USERS \times ROLES$ è una relazione tra utenti e ruoli, del tipo "molti-a-molti" (UA, user access);*

- *$user : S \rightarrow USERS$ è una funzione che associa ad ogni singola sessione $s_i \in S$ un singolo utente $user(s_i) \in USERS$, ed è un valore costante durante il tempo di vita della sessione;*

- *$roles : S \rightarrow 2^{ROLES}$, una funzione che manda ogni singola sessione $s_i \in S$ in un insieme di ruoli $roles(s_i) \subseteq \{r \in ROLES \mid (user(s_i), r) \in UA\}$.*

$RBAC_1$, un'evoluzione del modello precedente, in cui viene introdotto il concetto di *gerarchia di ruoli*, una relazione di ordine parziale che permette di semplificare l'attribuzione di permessi a ruoli, in base all'autorità ed alla responsabilità associata a tali ruoli: in questo modello si distinguono infatti i *senior role*, che possiedono tutti i permessi relativi agli *junior role* ad essi collegati nel diagramma delle gerarchie, oltre ad ulteriori privilegi dovuti alla loro maggiore "autorità".

Definizione 2 *Il modello $RBAC_1$ è fondato sulle seguenti componenti:*

- *gli insiemi USERS, ROLES, PERMS, S, PA e UA;*

- *$RH \subseteq ROLES \times ROLES$ è un ordine parziale su ROLES, detto* gerarchia di ruoli *o* relazione di ruolo dominante*, indicata con "$\geq$";*

- *$roles : S \rightarrow 2^{ROLES}$, diversa dall'analoga funzione definita nel modello $RBAC_0$ perché può cambiare nel tempo ed è tale che:*

$$roles(s_i) \subseteq \{r \in ROLES \mid \exists\, r' \in ROLES, r' \geq r, \text{ tale che } (user(s_i), r') \in UA\};$$

inoltre la sessione s_i ha i seguenti permessi:

$$\bigcup_{r \in roles(s_i)} \{p \in PERMS \mid \exists\, r'' \in ROLES, r'' \leq r, \text{ tale che } (p, r'') \in PA\}.$$

Nel modello $RBAC_2$ si aggiungono dei *vincoli* che riguardano le relazioni tra utenti, permessi, ruoli e sessioni, come ad esempio i vincoli sulle condizioni di implementazione, sulla mutua disgiunzione dei ruoli organizzativi, sulla mutua esclusività dei permessi, sul divieto di combinazione di ruoli e sulla massima cardinalità.

Definizione 3 *$RBAC_2$ è un modello che differisce da $RBAC_0$ solo per la presenza di vincoli, in base ai quali si determina se accettare o meno le componenti di $RBAC_0$.*

Infine è stato definito il modello $RBAC_3$, che rappresenta uno stadio terminale nell'evoluzione dei modelli RBAC e che include tutti i modelli precedenti.

Sistemi di Identity Management e Access Management

Quando un sistema informativo è composto da un gran numero di applicazioni, ognuna avente il suo sistema di controllo degli accessi, gestire e controllare la sicurezza generale del sistema risulta complicato perché la situazione che si presenta è eterogenea e "polverizzata" su numerosi ambienti diversi, ognuno dei quali è dotato di uno specifico strumento di gestione. Inoltre, poiché il compito di supervisionare e gestire la sicurezza di ogni specifica componente del sistema è affidato ad un responsabile con ambiti di visibilità e autorità limitati, in un sistema informativo come quello sopra descritto si avranno un numero di responsabili di sicurezza pari al numero di sotto-sistemi presenti, una quantità che può quindi essere molto elevata, che non contribuirà di certo a costituire un quadro complessivo facilmente gestibile, in quanto totalmente decentralizzato. L'eccessiva autonomia di ogni applicazione oltre a provocare, come abbiamo accennato, difficoltà di gestione della sicurezza complessiva del sistema informativo, crea ulteriori problemi di tipo tecnico.

Come soluzione ai problemi sopra esposti, si propongono due infrastrutture trasversali al sistema informativo: un sistema di *Identity Management* ed un sistema di *Access Management*.

Il sistema di *Identity Management*, IDM, è un'infrastruttura trasversale all'organizzazione aziendale che permette di gestire centralmente le *identità virtuali* degli utenti abilitati ad accedere al sistema informativo, e di controllare centralmente la sicurezza del sistema informativo nel suo complesso. Nel sistema informativo, IDM viene collocato in uno strato intermedio tra quello delle fonti di informazione fidate, che gli forniscono informazioni attendibili sugli utenti del sistema informativo, e quello dei sistemi di *access management* che gestiscono il controllo degli accessi alle applicazioni dello stesso (vedi Figura 2).

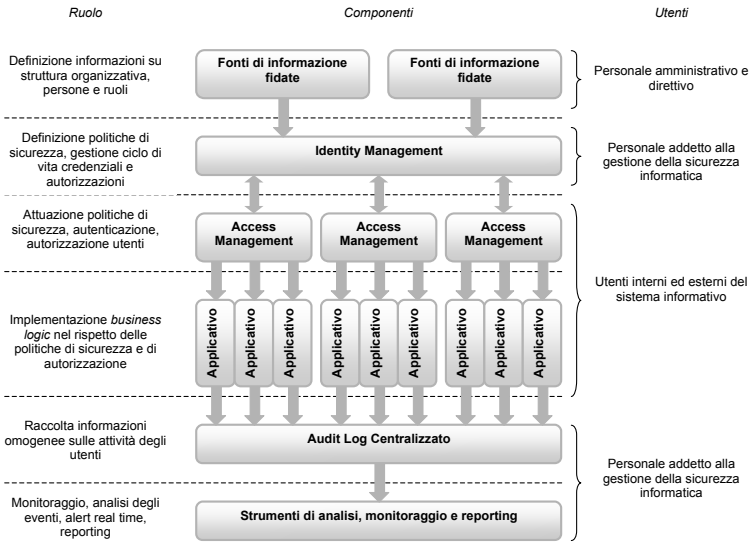

Figura 2: Stratificazione delle diverse componenti nell'architettura complessiva del sistema

Si definisce *risorsa* di IDM, o *piattaforma target*, ogni sistema informatico integrato con IDM. Ad ogni utente del sistema informativo, è associata una *virtual identity* sul sistema informatico IDM, nella quale vengono raccolte tutte le informazioni relative all'utente che permettono di determinare univocamente l'identità della persona, e tutte le credenziali che l'utente utilizza per accedere alle risorse del sistema informativo.

Durante la normale attività di un sistema informativo, l'archivio delle *virtual identity* viene continuamente aggiornato con l'aggiunta o con la modifica di informazioni, ed utilizzato per trasmettere queste informazioni ai sistemi informatici di autenticazione ed autorizzazione presenti nelle piattaforme target. Le operazioni effettuate sul/dal database delle *virtual identity* sono la *sincronizzazione*, tramite cui il database delle *virtual identity* viene alimentato automaticamente da un aggiornamento continuo delle informazioni anagrafiche ed organizzative, la *riconciliazione*, con cui cui le piattaforme target trasmettono al sistema IDM le informazioni relative agli account di accesso e agli attributi identificativi ed autorizzativi degli utenti memorizzate nei database delle risorse, e infine lo *user provisioning*, ovvero l'operazione in cui il sistema IDM trasferisce ad ogni piattaforma target con cui è connesso le credenziali e le informazioni identificative di ciascun utente (vedi Figura 3).

IDM in questo modo gestisce e controlla i sistemi di gestione degli utenti; crea un

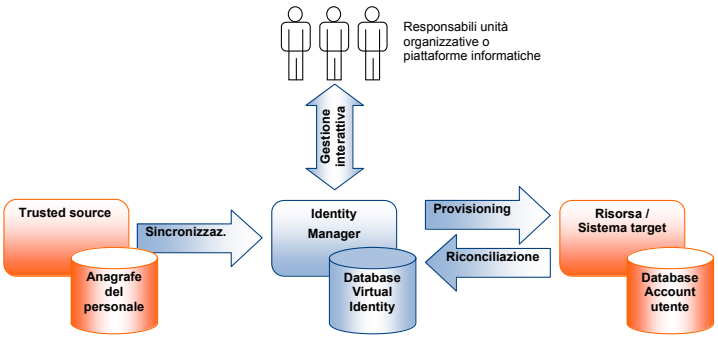

Figura 3: Interazione del sistema Identity Manager con l'esterno

quadro unitario per quanto riguarda gli account e le autorizzazioni; gestisce l'intero ciclo di vita delle credenziali per tutte le applicazioni collegate al sistema IDM; recepisce tempestivamente le variazioni dell'assetto organizzativo e, se necessario, cambia di conseguenza le credenziali e le autorizzazioni; stabilisce politiche di sicurezza unitarie e le applica a tutte le componenti del sistema informativo monitorando così la sicurezza globale.

Le piattaforme di *Web Access Management* (AM) sono infrastrutture trasversali di servizio, implementate su sistemi informativi web based, ma anche su infrastrutture di tipo SOA, che effettuano le operazioni di autenticazione ed autorizzazione degli utenti per conto delle applicazioni di un sistema informativo. I sistemi AM servono a consolidare la sicurezza informatica delle applicazioni web ed il modo in cui operano è indipendente dal tipo di applicazione (o dal linguaggio di programmazione con cui è scritta) che sfrutta il servizio. I principali metodi di autenticazione adottati dai sistemi AM sono l'*autenticazione password based*, in cui bastano username e password per autenticare l'utente; l'*autenticazione forte con certificati digitali X.509*, in cui l'utente viene identificato tramite il certificato digitale memorizzato nel repository locale del web browser (*key store*) o su una smart-card o token USB, sbloccati mediante un PIN locale non trasmesso in rete; l'autenticazione integrata con il dominio Microsoft Active Directory, secondo cui una volta effettuato il login su questo dominio non è necessario che l'utente si autentichi nuovamente sulle applicazioni legate al dominio perché i dati della sua identificazione vengono trasmessi automaticamente dal *browser* dell'utente al sistema AM in modo "trasparente". Un sistema AM può anche effettuare servizi di autenticazione ed autorizzazione in base al ruolo attribuito all'utente, consentendo

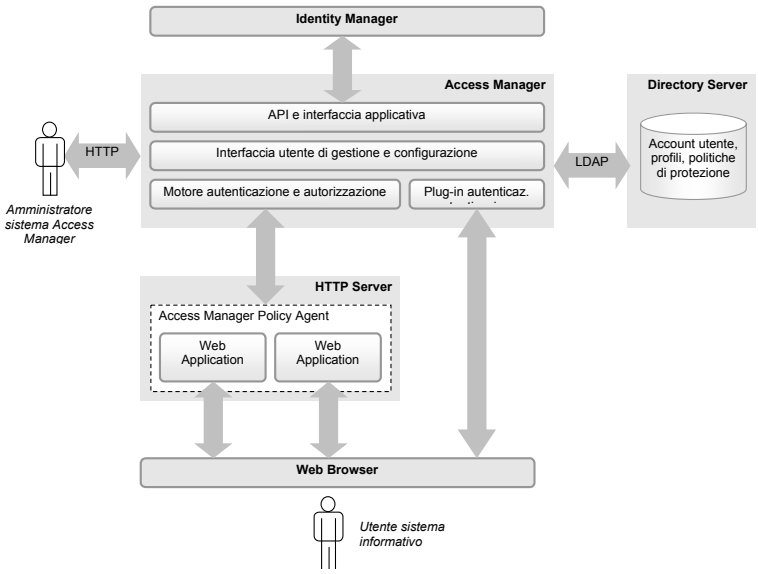

Figura 4: Componenti di un sistema di *access management*

di definire meccanismi di *single sign-on* che permettono di far effettuare all'utente un solo login per poter poi lavorare su varie applicazioni che fanno parte di uno stesso "*realm* di autenticazione" nell'ambito della medesima sessione di lavoro.

Un sistema di *Access Management* è formato da tre componenti principali: l'*Access Manager*, lo *user store* e l'*agent di autenticazione* (vedi Figura 4).

La prima delle componenti, l'*Access Manager*, è composta da interfacce API, utilizzate da IDM per effettuare lo *user provisioning* su AM; da interfacce *web based*, dalle quali gli amministratori incaricati della gestione e della configurazione del sistema effettuano il loro compito; da numerosi *plug-in*, grazie ai quali si implementano i diversi meccanismi di autenticazione, ed infine da un motore che esegue effettivamente le operazioni di autorizzazione ed autenticazione applicando le policy di sicurezza definite dagli amministratori nella configurazione del sistema. La seconda componente, lo *user store*, è un *directory server* nel quale vengono memorizzate le credenziali degli utenti che accedono all'applicazione, le informazioni sulla profilazione autorizzativa di tali utenti, la configurazione del sistema, e le politiche di protezione delle risorse. La terza componente, l'*agent di autenticazione*, è un modulo software installato sui web server delle applicazioni da proteggere che ha la funzione di rilevare, valutare e stabilire se la richiesta di accesso alla web application del server è legittima o meno.

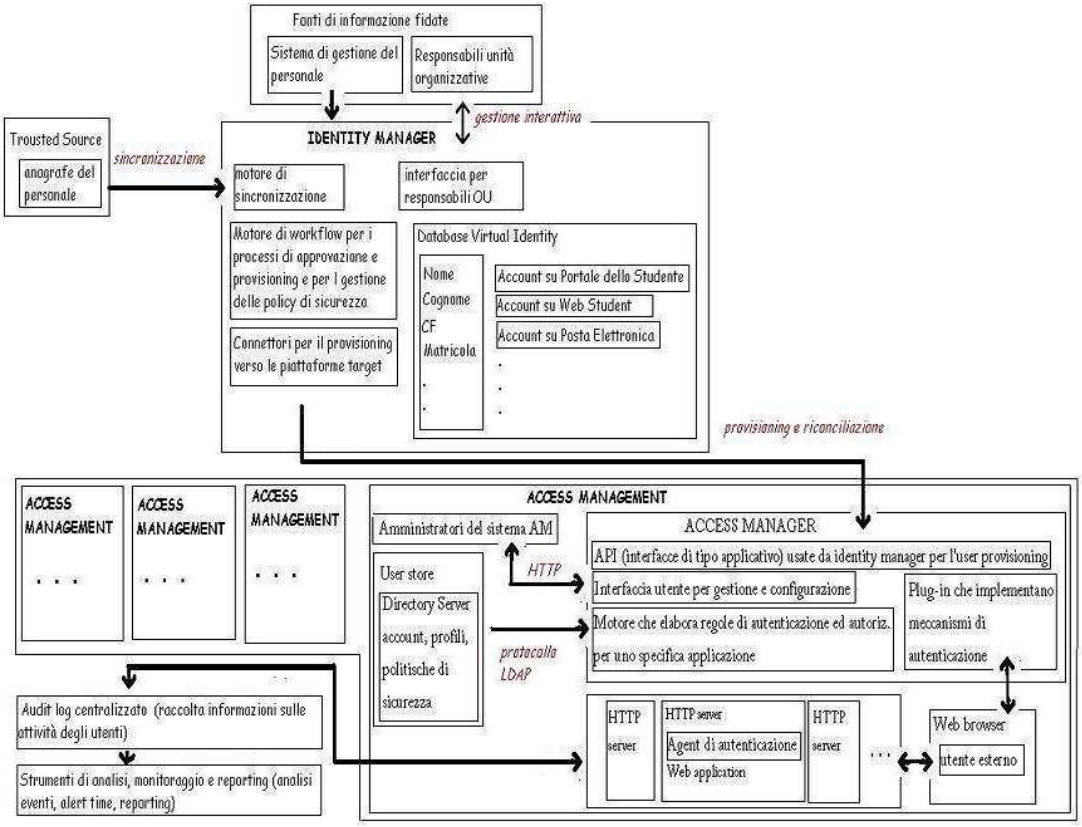

Figura 5: Architettura di un sistema informativo in cui sono presenti i sistemi di Identity e Access Management

Spesso il *web agent* è realizzato come un "filtro" del web server che, quando un utente effettua dal suo *web browser* una richiesta in formato HTTP al *web server* dell'applicazione, "intercetta" la richiesta (la filtra), la valuta e stabilisce se è legittima in base alle policy di sicurezza definite sull'*access manager* per quella specifica risorsa. In caso affermativo, l'*agent* ordina al *web server* dell'applicazione di inviare all'utente la risorsa desiderata, altrimenti, se la richiesta non è legittima, o fa comparire sulla pagina del *web browser* dell'utente un messaggio di errore, o redirige l'utente ad una pagina di autenticazione per tentare un nuovo login.

Role mining problem

RBAC (*role based access control*) è un modello assai efficace per la gestione del controllo degli accessi. In questo contesto ciò che risulta complesso è l'identificazione di un insieme di ruoli facilmente gestibile, completo ed idoneo a rappresentare tutti gli utenti che vogliono accedere ad un'applicazione. Esistono vari algoritmi per generare questo tipo di insiemi di ruoli, detti algoritmi di *role mining*.

Nel primo articolo di Colantonio, Di Pietro e Ocello [5] preso in esame, viene affrontato il il problema del *role mining* con l'obiettivo di minimizzare il costo di amministrazione di un contesto RBAC, proponendo un algoritmo, denominato RBAM (*role-based association-rule mining*), che genera un insieme di ruoli tale da rendere minima una *funzione costo*, che rappresenta lo sforzo necessario per gestire il contesto RBAC.

Il problema può essere posto come un problema di ottimizzazione combinatoria: dato l'insieme finito di relazioni tra utenti e permessi, UP, l'obiettivo del *role engineering* è quello di determinare gli insiemi di ruoli, *ROLES*, di relazioni tra permessi e ruoli, *PA*, di relazioni tra utenti e ruoli, *UA* e di gerarchie tra ruoli, *RH* che ricoprono tutte le possibili combinazioni dei permessi posseduti dagli utenti con il costo di amministrazione più basso possibile.

La soluzione prodotta da un algoritmo di *role mining* sarà un insieme di ruoli avente le seguenti proprietà:

1. *Completezza*: l'insieme è completo se, per ogni utente, è sempre possibile definire un sottoinsieme di ruoli appropriato tale che l'unione dei permessi as-

sociati a quei ruoli corrisponda esattamente all'insieme dei permessi posseduti dall'utente.

2. *Costo minimo*: un insieme di ruoli è a costo minimo, ovvero è ottimale, se è completo e minimizza il costo di amministrazione, espresso tramite una funzione di costo $f : |ROLES| \times |PA| \times |UA| \times |RH| \times attrs \rightarrow \mathbb{R}$. L'insieme "*attrs*" rappresenta altri attributi di business.

Grazie a definizioni di funzioni e strumenti matematici necessari all'analisi, viene studiato il comportamento di una funzione costo scelta in modo tale da raggiungere il suo valore minimo se calcolata nella soluzione ottimale cercata. Una funzione di costo ragionevole potrebbe essere la seguente:

$$f = \alpha|UA| + \beta|PA| + \gamma|ROLES| + \delta \sum_{r \in ROLES} c(r) \tag{1}$$

dove $\alpha, \beta, \gamma, \delta \geq 0$ e la funzione $c : ROLES \rightarrow \mathbb{R}$ esprime ulteriori valori di costo, relativi ad informazioni di business.

Possiamo introdurre il concetto di *supporto* di un ruolo $r \in ROLES$, definendolo come la percentuale di utenti che possiedono tutti i permessi assegnati al ruolo r:

$$support(r) = \frac{|auth\_users(r)|}{|USERS|}$$

Tali utenti possiedono quei permessi o perché sono stati assegnati direttamente al ruolo r, o perché sono stati assegnati ad un ruolo senior di r. Pertanto con la notazione

$$actual\_support(r) = \frac{|ass\_users(r)|}{|USERS|}$$

possiamo indicare la percentuale di utenti che assegnati direttamente al ruolo r.

Dato l'insieme di ruoli formato da tutte le possibili combinazioni di permessi, l'obiettivo del lavoro è capire quali combinazioni cancellare per dal reticolo per poter ottenere convergenza verso l'insieme di ruoli ottimale.

Descriviamo in sintesi la strategia adottata dall'algoritmo RBAM. Inizialmente, basandosi sul reticolo dei permessi *PERMS*, si genera un insieme di ruoli candidato (R_1). Quando si elimina un ruolo dall'insieme dei ruoli candidato, la funzione costo f, definita in (1), subisce una fluttuazione, per cui ad essa va aggiunta la seguente quantità:

$$-\alpha u_r + \alpha \mu_r u_r - \beta p_r - \gamma - \delta c(r) \tag{2}$$

in cui $u_r = |ass\_users(r)|$ è il numero di utenti assegnati al ruolo r e non ai suoi figli, per cui $-\alpha u_r$ indica il numero di relazioni tra utenti e ruoli cancellate in UA; $+\alpha\mu_r u_r$ indica che il ruolo r è stato sostituito con μ_r ruoli diversi, in modo che ogni utente che aveva una relazione con il ruolo r non possieda nuove relazioni con i μ_r ruoli diversi; $p_r = |auth\_perms(r)|$ è il numero di permessi autorizzati da r quindi, $-\beta p_r$ indica che in PA sono state rimosse p_r relazioni tra ruoli e permessi; $-\gamma$ indica un ruolo rimosso da $ROLES$ ed infine $-\delta c(r)$ indica che non è più necessario il costo relativo ad altre informazioni di business. Se la cancellazione del ruolo r ha provocato una diminuzione della funzione costo, allora la quantità aggiunta deve essere negativa. Si pone quindi:

$$-\alpha u_r + \alpha\mu_r u_r - \beta p_r - \gamma - \delta c(r) \leq 0$$

dalla quale si otterrà la seguente disequazione

$$(\mu_r - 1)u_r \leq \sigma p_r + \tau + vc(r) \tag{3}$$

dove $\sigma = \beta/\alpha, \tau = \gamma/\alpha$ e $v = \delta/\alpha$ sono parametri che dipendono dal contesto su cui si sta lavorando. La disequazione (3) indica che il numero di utenti assegnati ad un ruolo, moltiplicato per il numero di ruoli necessari per poter ottenere il ruolo diminuito di un'unità, deve essere minore del numero di permessi autorizzati dal ruolo in esame aggiunto al costo del ruolo rimosso e a quello relativo ad informazioni esterne. Da questa, procedendo con dei passaggi, si ottiene una condizione che attesta l'effettiva diminuzione della funzione costo in seguito alla cancellazione del ruolo:

$$(\mu_r - 1) \cdot actual\_support(r) \leq \bar{\sigma} p_r + \bar{\tau} + \bar{\nu} c(r) \tag{4}$$

L'Algoritmo 1 riporta, in sintesi, i passi dell'approccio adottato da RBAM per effettuare l'operazione di *role mining* a partire da un contesto in cui i ruoli non sono definiti, ma è preassegnato un insieme di permessi agli utenti del sistema informatico.

Il vincolo di minimo supporto usato nel passo 5 è un caso particolare dell'equazione (4); applicando tale osservazione, si ottiene la seguente *condizione di pruning*:

$$support(r) > \bar{\sigma} k + \bar{\tau} + \bar{v} c(r). \tag{5}$$

Inoltre la correttezza del *prune step* (passo 5 dell'algoritmo) viene verificata tramite il seguente teorema:

Algoritmo 1 RBAM

1: analisi dei permessi assegnati agli utenti e creazione dell'insieme R_1, con i ruoli corrispondenti a ciascun permesso (un permesso per ogni ruolo)
2: $k = 1$
3: **fintanto che** $R_k \neq \emptyset$ **ripeti**
4: $k = k + 1$
5: genera l'insieme di ruoli R_k ottenuto mediante la combinazione a coppie di tutti i ruoli di R_{k-1} con il vincolo che i permessi con un supporto inferiore ad una soglia predefinita non vengono presi in considerazione ("potatura dei ruoli")
6: definisci la gerarchia di ruoli H_k tra i ruoli di R_k e quelli di R_{k-1}, definendo come ruoli senior i ruoli di R_k che contengano completamente i permessi dei ruoli di R_{k-1} (ruoli junior)
7: **fine-ciclo**
8: restituisci $ROLES = \cup_k R_k$ e $RH = \cup_k H_k$

Teorema 1 *Dati* $r_1, r_2 \in ROLES$ *tali che* $r_1 \succeq r_2$ *e* $c(r_1) \geq c(r_2)$*, se un ruolo non soddisfa la disequazione* (5)*, allora non potrà essere generato nessuno dei suoi ruoli figli.*

Quindi, se non è verificata la condizione di *pruning*, il ruolo preso in esame viene scartato e di conseguenza non verranno generati suoi figli. Questa strategia non non sempre conduce alla soluzione ottima perché quando si genera un ruolo al passo $k+1$ tramite fusioni di ruoli del livello k, si potrebbe avere una cancellazione dei ruoli di livello k e una diminuzione degli utenti assegnati ai parenti del ruolo cancellato, che potrebbe aumentare il valore della funzione costo. C'è da dire quindi che la (5) non tiene conto della cancellazione di ruoli dovuta alla generazione di altri ruoli nei passi successivi. Per $k = 1$, la condizione fa sì che al primo step vengano scartati tutti i permessi aventi un supporto troppo basso e questo comporta che l'insieme dei ruoli candidato risulti incompleto. Per aggirare il problema, per ogni permesso scartato a causa della condizione (5), vengono creati ruoli che avranno, quindi, un numero di utenti basso, ma accettabile. Inoltre, quando al passo $k+1$ vengono generati i figli di r, il numero di utenti assegnato ad r ed ai suoi figli, u_r, diminuisce al livello $k+1$, per cui è necessario controllare nuovamente la validità della condizione (4) per ogni ruolo del livello k.In conclusione, possiamo dire che l'algoritmo RBAM offre un'approssi-

mazione ragionevole della soluzione, la possibilità di scegliere se utilizzare o meno il modello RBAC gerarchico. Per migliorare l'algoritmo si potrebbe fornire una funzione costo più raffinata e tale da ottenere risultati migliori, o si potrebbe cercare di capire quali siano i valori migliori dei parametri $\sigma, \tau, \nu, c(r)$.

Il secondo articolo di *role mining problem* che ho preso in esame è [6]. Viene di nuovo considerato l'insieme di tutti i permessi attribuibili agli utenti del sistema e creato un ruolo per ogni loro possibile combinazione. L'obiettivo é, anche in questo caso, eliminare *ruoli ridondanti*, riducendo la cardinalità dell'insieme di ruoli candidati e diminuendo, automaticamente, la complessità di gestione ed amministrazione di RBAC.

Ricordiamo che un *insieme parzialmente ordinato*, *poset*, è una coppia $\langle S, \succeq \rangle$, in cui S e $\succeq$ sono rispettivamente un insieme ed una relazione binaria che indicano, per alcune coppie di elementi dell'insieme, quale elemento precede l'altro.

Un *reticolo* (*lattice*), $\langle L, \succeq, \curlyvee, \curlywedge \rangle$, è un poset, $\langle L, \succeq \rangle$, nel quale ogni coppia di elementi $x, y \in L$ ha un unico *join*(l'estremo superiore, detto anche *lub*), denotato con $x \curlyvee y$, e un *meet* (un più grande limite inferiore, detto anche *glb*),denotato con $x \curlywedge y$ dentro L. Se $\langle L, \succeq, \curlyvee, \curlywedge \rangle$ è un reticolo, allora $\langle \Lambda, \succeq, \curlyvee, \curlywedge \rangle : \Lambda \subseteq L$ è un *sottoreticolo* se e solo se $\forall x, y \in \Lambda : x \curlyvee y \in \Lambda \wedge x \curlywedge y \in \Lambda$.

L'attribuzione $S = PERMS$, rende possibile la costruzione di un *modello RBAC basato sui ruoli derivabili da un insieme di permessi dato*. Sostituiamo l'operatore di inclusione $\supseteq$ con l'operatore $\succeq$, che rappresenta un ordine parziale sui ruoli, l'operatore $\curlyvee$ con $\cup$ e l'operatore $\curlywedge$ con $\cap$.

Nel reticolo risultante $\langle 2^{PERMS}, \succeq, \curlyvee, \curlywedge \rangle$ ogni combinazione di permessi rappresenta un ruolo dell'insieme *ROLES*, rappresenta le relazioni corrispondenti a tale permesso in *PA*, rappresenta le inclusioni dei permessi in *RH* che coinvolgono il ruolo e rappresenta infine le relazioni tra gli utenti e queste combinazioni di permessi in *UA*. *RH* è la riduzione transitiva del grafo associato al reticolo. Inoltre, se un utente è assegnato al ruolo r, allora *UA* conterrà le relazioni tra r, i suoi figli e gli utenti ad esso assegnati. Sapendo che la funzione $ass\_perms : ROLES \rightarrow 2^{PERMS}$ identifica i permessi assegnati ai ruoli, notiamo che questo reticolo è dotato di alcune importanti proprietà:

Lemma 1 *Rimuovendo un ruolo r dall'insieme ROLES, e le sue relazioni corrispondenti in PA, UA, RH tali che* $ass\_perms(r) \neq \cap_{r' \in ROLES} ass\_perms(r')$ *e* $ass\_perms(r) \neq$

$\cup_{r' \in ROLES} ass\_perms(r')$*, l'insieme ROLES risultante è ancora un reticolo.*

Per cui i ruoli ai quali sono assegnate combinazioni di permessi inutilizzati non rappresentano ruoli candidati significanti. Questi ruoli hanno un supporto uguale a 0 e possono essere eliminati dall'insieme *ROLES*, tranne per il join ed il meet che sono richiesti per preservare le proprietà di reticolo. Rimuovendo questi ruoli il reticolo risultante soddisfa la seguente proprietà:

Lemma 2 *Il padre diretto del ruolo* $r \in ROLES$ *differisce da* r *tramite un singolo permesso, che è* $\forall r, s \in ROLES : s \gg r \Rightarrow degree(r) = degree(s) + 1$.

Con le affermazioni sostenute finora si è quindi constatato che è possibili eliminare ruoli con supporto uguale a zero. Vediamo ora che è inoltre possibile eliminare ruoli equivalenti ad altri ruoli.

Definizione 4 (Ruoli ed insiemi di ruoli equivalenti) *Dato un ruolo* r *e un insieme di ruoli* $\{r_1, \ldots, r_n\}$ *essi sono equivalenti, e scriveremo* $r \equiv \{r_1, \ldots, r_n\}$*, se*

$$auth\_users(r) = \bigcup_{i=1}^{n} auth\_users(r_i)$$

Ovviamente quindi $r_1, r_2 \in ROLES$ *si dicono* equivalenti*, e scriveremo* $r_1 \equiv r_2$*, se* auth_users(r_1)= auth_users(r_2).

Quando un ruolo ha più padri equivalenti, la combinazione dei permessi ad esso assegnati rappresenta nuovamente un ruolo equivalente, che non aggiunge ulteriori informazioni rispetto a quelle fornite da tale ruolo equivalente, per cui uno dei due ruoli considerati può essere cancellato. Ampliando il discorso dei ruoli equivalenti possiamo arrivare a stabilire che nel reticolo di ruoli considerato possono esserci diverse repliche di sottoreticoli equivalenti, che possono quindi essere rimosse tranne i ruoli equivalenti massimi di ogni sottoreticolo.

L'algoritmo RBAPRIORI offre una procedura per rimuovere i sottoreticoli equivalenti tranne i ruoli di grado massimo. Viene presentato in [6] come estensione di APRIORI, un algoritmo che genera un reticolo parziale eliminando i permessi aventi un supporto minore di una soglia prestabilita $s_{\min}$, che esegue questi passi:

- Generazione di R_1, formato dai ruoli candidati di grado 1 e con un supporto maggiore di $s_{\min}$.

- Quando $k \geq 2$, l'insieme $R_k := \{$ruoli calcolati al passo $k\}$ è generato unendo tutte le possibili coppie di ruoli in R_{k-1} (*join step*). Per non generare ruoli con lo stesso insieme di permessi, vengono considerate solo le coppie di ruoli aventi la maggior parte dei permessi diversi. Il *prune step* consiste nell'eliminazione di tutte le combinazioni di permessi con supporto minore di $s_{\min}$. Vengono create le relazioni gerarchiche H_k, in relazione con i ruoli in R_k, i cui permessi assegnati sono un superset di permessi dei ruoli in R_{k-1}.

- L'algoritmo termina quando $R_k = \emptyset$, dando come risultato l'insieme $ROLES = \cup_i R_i$ e $RH = \cup_i H_i$.

L'algoritmo RBAPRIORI è ottenuto da APRIORI richiamando alla fine di ogni passo k la procedura REMOVEEQUIVALENTSUBLATTICES (vedi l'Algoritmo 2). Una caratteristica di RBAPRIORI è che rigetta un solo sottoreticolo alla volta. L'applicazione dell'algoritmo a dati reali, rispetto a quella ottenuta tramite APRIORI, ha prodotto una riduzione del numero di ruoli prodotti e della complessità.

Algoritmo 2 REMOVEEQUIVALENTSUBLATTICES

$W = \emptyset, M_i = \emptyset$
per ogni $\rho \in \{h.junior \mid h \in H_k : h.confidence = 1\}$ **ripeti**
  $E = \{h.senior \mid h \in H_k : h.junior = \rho \wedge h.confidence = 1\}$
  $S = \{h.senior \mid h \in H_k : h.junior = \rho \wedge h.confidence < 1\}$
  $P = (\cup_{r \in E} ass\_perms(r))/ass\_perms(\rho)$
  $W = W \cup E$
  per ogni $\sigma \in S \cup \{\rho\}$ **ripeti**
    $\sigma.degree = \sigma.degree + |P|$
    $PA = PA \cup (P \times \{\sigma\})$
    $M_i = M_i \cup \{\sigma\}$
  fine-ciclo
fine-ciclo
$R_k = R_k/W$
$PA = \{< p, r > \in PA \mid r \notin W\}$
$UA = \{< u, r > \in UA \mid r \notin W\}$
$H_k = \{h \in H_k \mid h.senior \notin W\}$

Separation of duty in un contesto RBAC

La *separation of duty* (SoD), talvolta indicata come *conflitto di interessi* o *mutua esclusione*, rappresenta probabilmente la più importante policy di sicurezza del sistema perché identifica tutte le operazioni che non dovrebbero essere concesse ad ogni singolo utente di un sistema informativo. Nell'articolo [7] viene proposta una nuova struttura per amministrare i vincoli della separation of duty che permette di definire in modo semplice i vincoli di SoD e di ridurre contemporaneamente la complessità dei sistemi RBAC ed il costo di amministrazione necessario per gestirli.

Nel nuovo modello SoD viene introdotto il concetto di *attività di business*. Un'*attività* è un insieme di permessi necessari per definire compiti particolari, ottenuta tramite la decomposizione di processi di business. I concetti di *attività* e di *ruolo* sembrano somigliarsi perché entrambi costituiti da insiemi di permessi, ma differiscono nel significato, nella cardinalità e nell'astrazione ad essi associata. Il confronto tra il numero di ruoli ed il numero di attività generalmente presenti in una compagnia si può trarre dal fatto che, se in essa compaiono migliaia di ruoli, sono presenti poche centinaia di attività. Spesso infatti si ha la necessità di introdurre ruoli distinti mentre l'insieme di attività associato è lo stesso. Le attività, infine, rappresentano un concetto più astratto dei ruoli perché sono indipendenti dai permessi e la definizione di vincoli tra esse può essere effettuata anche solo dallo staff di business, che non è a conoscenza del controllo dagli accessi, mentre per definire i vincoli fra ruoli è spesso necessaria la collaborazione di staff di business e staff IT.

Sia $ACTVT$ l'insieme di tutte le attività ottenute come decomposizione di processi di business. Per ogni coppia $a_c, a_p \in ACTVT$, la relazione $a_c \succeq a_p$ indica l'esistenza di un cammino di relazione gerarchica che va da a_c ad a_p. Con $GRPS \subseteq 2^{PERMS}$ viene indicato l'insieme di raggruppamenti di permessi associati ad un'attività. In questo modello viene introdotto il concetto di *permession grouping* tramite il quale i permessi, invece di essere assegnati direttamente alle attività, vengono prima raggruppati in uno o più sottoinsiemi. In questo modo un utente che voglia eseguire un'attività deve possedere tutti i permessi del più piccolo raggruppamento associato a tale attività.I concetti di *attività* e *gruppo* possono essere visti come delle specificazioni del concetto di *ruolo*, si ha quindi che $ACTVT \subseteq ROLES$ e $GRPS \subseteq ROLES$.

Una prima descrizione dei conflitti tra le attività viene basata sull'idea che i conflit-

ti tra i permessi siano scaturiti da conflitti tra attività, visto che le attività sono insiemi di permessi. In questo senso si può effettuare una rassegna delle principali tipologie di conflitti: *permessi conflittuali ed illegali*, i permessi sono tra loro in conflitto se consentono l'esecuzione di attività in conflitto; *ruoli conflittuali ed illegali*, i ruoli sono in conflitto quando l'unione dei permessi ad essi assegnati consente l'esecuzione di attività conflittuali; *utenti conflittuali ed illegali*, gli utenti sono in conflitto quando l'unione dei permessi assegnata ai propri ruoli permette l'esecuzione di attività conflittuali. Quando $|P| = 1$, il permesso p si dice *illegale* perché consente da solo l'esecuzione di attività conflittuali; quando $|R| = 1$, il ruolo $r \in R$ è *illegale*, ovvero contiene permessi conflittuali o illegali ed infine quando $|U| = 1$, l'utente $u \in U$ si dice *illegale*, perché è assegnato a ruoli conflittuali o illegali.

La seconda classificazione dei conflitti si basa invece sulla classe di vincoli *Object-Based SoD*, secondo cui un utente non può eseguire attività conflittuali nello stesso oggetto. Per supportare questo tipo di vincolo in RBAC, in [7] viene introdotto il concetto di *dominio SoD*. Per esempio, un dominio SoD è un insieme di dati in cui un singolo utente non dovrebbe completare le attività in conflitto.

Un permesso RBAC è rappresentato da una coppia $\langle o, m \rangle$, dove o indica l'oggetto e m il modo in cui è possibile accedere all'oggetto. Quindi, dato un permesso, è possibile determinare in quale dominio opera tale permesso. Tutte le definizioni precedenti che rappresentano i conflitti tra permessi, ruoli ed utenti e che definiscono il caso in cui essi sono illegali, possono essere estese, includendo il concetto di dominio SoD.

L'introduzione dei domini SoD permette di introdurre nei sistemi RBAC il vincolo che lega utenti ed oggetti, ovvero i primi non possono eseguire attività conflittuali all'interno dello stesso oggetto, ma potrebbe comportare una complessità troppo grande da essere gestita. Si è visto che, definendo un dominio come l'insieme di tutti i dati accessibili da un'applicazione o da un insieme di applicazioni, vengono divisi in maniera opportuna i permessi conflittuali e viene limitata la complessità.

Consideriamo l'esempio riportato nella Figura 6. In esso sono presenti: un insieme di ruoli $ROLES = \{r_1, .., r_9\}$ posizionati su vari livelli in modo da rispecchiare le loro diverse autorità e collegati tramite relazioni gerarchiche; i permessi dell'insieme $PERMS = \{p_1, .., p_9\}$, che tramite le frecce orientate verso l'alto vanno a definire ruoli (ad esempio i permessi p_2, p_4 definiscono il ruolo il ruolo r_2), tramite quelle orientate verso il basso costituiscono raggruppamenti di permessi (ad esempio i permessi

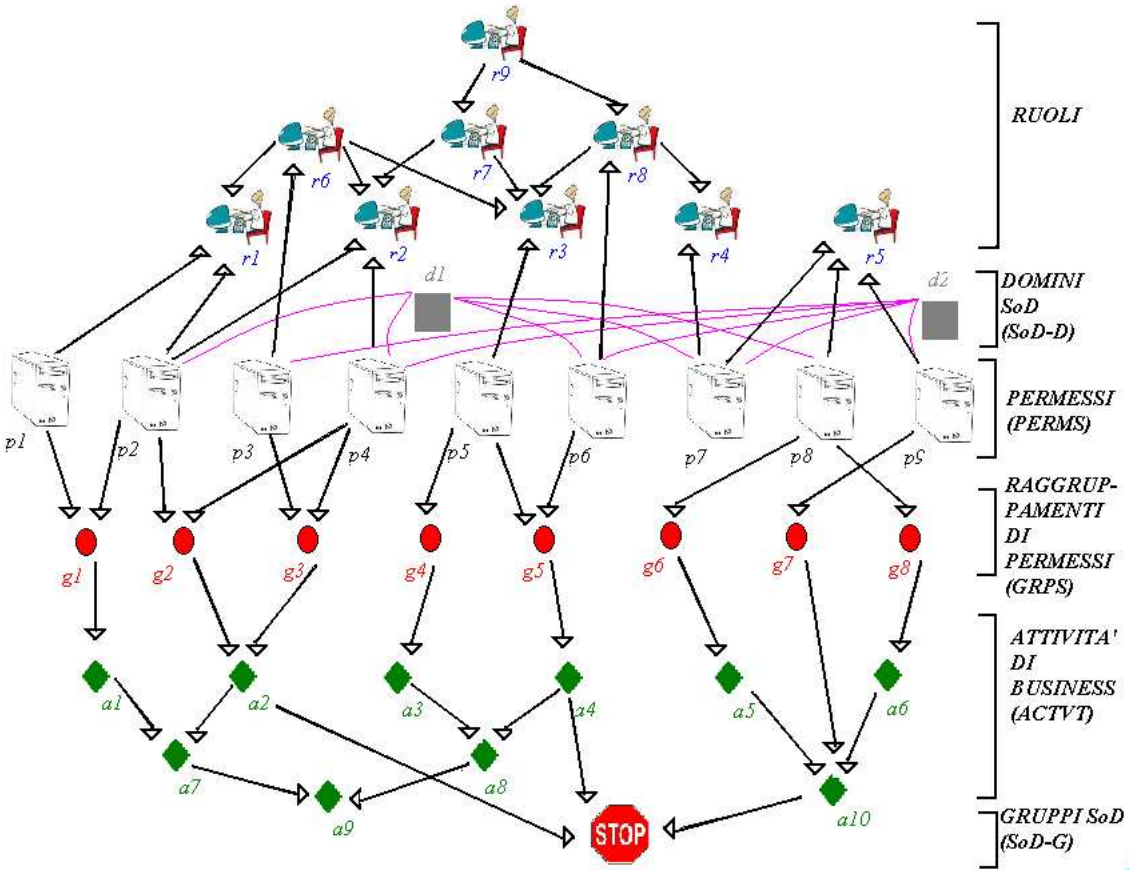

Figura 6: Rappresentazione grafica del modello SoD

p_3, p_4 costituiscono il gruppo g_3), ed infine tramite le linee di colore rosa si uniscono per definire i domini (ad esempio il dominio $d_1 = \{p_2, p_4, p_6, p_7, p_8\}$); l'insieme dei domini $SoD - D = \{d_1, d_2\}$; i gruppi dell'insieme $GRPS = \{g_1, .., g_8\}$, che sono collegati superiormente ai permessi ed inferiormente alle attività, in modo da definire quali gruppi possono eseguire determinate attività (ad esempio l'attività a_2 può essere eseguita solo dai ruoli aventi permessi presenti nei gruppi g_2, g_3); l'insieme delle attività $ACTVT = \{a_1, .., a_{10}\}$, che sono distribuite su vari livelli perché sono anch'esse legate da relazioni gerarchiche, ed infine l'insieme dei vincoli $SoD - G = \{\langle A, n\rangle\}$, che nell'esempio contiene un unico elemento: il vincolo $\{\langle\{a_2, a_4, a_{10}\}, 3\rangle\}$ che stabilisce che le attività a_2, a_4, a_{10} non possono essere eseguite tutte e tre contemporaneamente.

Un'attività può essere svolta solo quando un utente possiede tutti i permessi di un gruppo associato all'attività, ad esempio: a_4 può essere svolta solo se l'utente possiede i permessi p_6 e p_7 relativi al gruppo g_5; a_2 può essere svolta se l'utente possiede $\{p_2, p_4\} \in g_2$ o se possiede $\{p_3, p_4\} \in g_3$ mentre non può essere svolta se possiede p_2 e p_3 perché non appartengono entrambi ad gruppo necessario per svolgere a_2, infatti $actvt - grps(a_2) = \{g_2, g_3\} = \{\{p_2, p_4\}, \{p_3, p_4\}\}$. Per quanto riguarda i domini: un permesso può appartenere ai domini d_1 e d_2 ed operare in entrambi, come ad esempio p_6; può appartenere solo ad uno dei due, $p_2 \in d_1$ e $p_2 \notin d_2$, ed operare solo in quello, o può non appartenere ad alcun dominio, come $p_1 \notin d_1, d_2$.

Il vincolo $\{\langle\{a_2, a_4, a_{10}\}, 3\rangle\}$ in $SoD - G$ provoca conflitti tra utenti, ruoli e permessi, infatti: i *permessi* $\{p_2, p_4, p_6, p_7, p_8\}$, $\{p_3, p_4, p_6, p_7, p_9\}$, $\{p_2, p_4, p_6, p_7, p_8, p_9\}$, $\{p_3, p_4, p_6, p_7, p_8, p_9\}$ e $\{p_2, p_3, p_4, p_6, p_7, p_8, p_9\}$ sono permessi in conflitto mentre non esiste nessun permesso che da solo permetta l'esecuzione di a_2, a_4 e a_{10}, detto *illegale*, mentre i *ruoli in conflitto* sono $\{r_1, r_2, r_5, r_8\}$, $\{r_6, r_8, r_5\}$, $\{r_5, r_9\}$, $\{r_2, r_5, r_8\}$.

Con l'introduzione dei domini si ha che due ruoli sono in conflitto nel dominio se l'insieme dei permessi ad essi associato contiene permessi che non appartengono al dominio. Ad esempio, r_2, r_5, r_8 sono in conflitto perché $p_1, p_4, p_6, p_7, p_8 \in d_1$ ma $p_9 \notin d_1$.

Se considerassimo ora il seguente vincolo $\langle\{a_2, a_4, a_{10}\}, 2\rangle$ si avrebbe che due delle tre attività dell'insieme considerato non potrebbero essere svolte contemporaneamente. Per questo il permesso p_9 risulta *illegale* perché da solo consente di eseguire le attività conflittuali a_2, a_4.

Se considerassimo invece il vincolo $\langle\{a_5, a_6\}, 2\rangle$, p_8 sarebbe *illegale*.

Il modello illustrato in [7] è stato applicato dagli autori ad una grande azienda privata e da questa applicazione è stato riscontrato che, tramite la definizione dei conflitti SoD basati sul concetto di attività, il numero di relazioni da gestire diminuisce e con esso anche il costo di amministrazione; inoltre è possibile spartire il compito di identificazione delle relazioni tra le imprese e gli utenti, alleggerendo così il carico di lavoro dello staff IT e di business.

Bibliografia

[1] E. Ansuini, A. Lioy, M. Mecella, E. Melis, M. Mezzalama, G. Santucci, C. Simonelli, *Sistemi informativi – Sistemi distribuiti*, Franco Angeli Editore , Volume 5, 2001.

[2] D. Ardagna, M.G. Fugini, B. Pernici, P. Plebani, *Sistemi informativi – Sistemi informativi basati su web*, Franco Angeli Editore, Volume 6, 2006.

[3] P. Atzeni, C. Batini, F. Casati, B. Pernici, L. Saladini, *Sistemi informativi – Modelli e progettazione*, Volume 2, Franco Angeli Editore, 2001.

[4] F. Cantoni, B. Fitzgerald, N.L. Russo, E. Stolterman, *Lo sviluppo dei sistemi informativi*, Franco Angeli Editore, 2004.

[5] A. Colantonio, R. Di Pietro, A. Ocello, *A Cost-Driven Approach to Role Engineering*, Proceedings of the 2008 ACM symposium on Applied computing, ACM, New York, 2008.

[6] A. Colantonio, R. Di Pietro, A. Ocello, *Leveraging Lattices to Improve Role Mining*, Proceedings of 23rd International Information Security Conference IFIP International Federation for Information Processing, 2008, Volume 278/2008.

[7] A. Colantonio, R. Di Pietro, A. Ocello, *An activity-based model for separation of duty*, preprint, October 2008.

[8] N. Papatheodoulou, N. Sklavos, *Architecture and System Design of Authentication, Authorization, and Accounting Services*, Proceedings of The IEEE Region 8, EUROCON 2009, International Conference (IEEE EUROCON'09), San Pietroburgo, 2009.

[9] A. Marzona, M. Pighin, *Sistemi informativi aziendali – Struttura e applicazioni*, Pearson Education Italia, 2005.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, *Role-Based Access Control Models*, Computer, Volume 29 (2), IEEE, 1996.

[11] *Codice in materia di protezione dei dati personali*, Testo Unico sulla Privacy, d.lgs. 196/2003, allegato B.