Esprimere in FACPL politiche di controllo degli accessi basate sul comportamento passato


| | | |
|---|---|---|
| **Candidato:** | Federico Schipani | federico.schipani@stud.unifi.it |
| **Relatore:** | Rosario Pugliese | rosario.pugliese@unifi.it |
| **Correlatore:** | Andrea Margheri | andrea.margheri@unifi.it |


## Riassunto


Usage Control è un nuovo approccio sul controllo degli accessi introdotto recentemente da Ravi Sandhu e Jaehong Park, due ricercatori dell'università di San Antonio in Texas. Partendo dall'analisi di vari sistemi di *Access Control*, una breve disamina su Usage Control e dall'introduzione di due casi di studio, questa tesi si è posta l'obiettivo di estendere Formal Access Control Policy Language (FACPL) in modo tale che potesse prendere decisioni basate sul comportamento passato. FACPL è un linguaggio basato su eXtensible Access Control Markup Language (XACML) ed è supportato da una libreria scritta in Java. Inoltre FACPL è supportato anche da un Plugin per Eclipse basato su Xtext che permette di facilitare la creazione e la valutazione di policy, che fa uso di Xtend per tradurre autonomamente codice FACPL in codice Java.

Il processo di estensione di FACPL è stato svolto insieme al mio collega *Filippo Mameli* ed è fondamentalmente diviso in due parti. Nella prima parte del lavoro è stato necessario innanzitutto pensare a quali componenti mancassero a FACPL per prendere decisioni di Usage Control. Le componenti mancanti erano due: uno stato del sistema ed un insieme di funzioni che possano modificare, o comparare, i valori contenuti al suo interno. Una volta individuate è stato necessario inserirle all'interno del processo di valutazione di FACPL. Lo stato è definito come un insieme di attributi e valori usati durante la valutazione di una policy, quindi la valutazione è stata estesa per permettere al Policy Decision Point (PDP) di andare a cercare i valori di cui necessita anche all'interno dello stato. Le funzioni di modifica dello stato introdotte servono per soddisfare una caratteristica peculiare di quest'ultimo, ovvero la sua mutabilità nel tempo dovuta al susseguirsi di richieste. Queste nuove funzioni sono state introdotte durante il processo di enforcement del Policy Enforcement Point (PEP), sotto forma di *Obligation Status*. Per inserire all'interno del linguaggio queste nuove componenti è stata estesa la grammatica e la semantica di FACPL, mostrata anche con i casi di studio introdotti all'inizio dell'elaborato.

La seconda parte del lavoro ha riguardato l'estensione della libreria Java su cui si basa FACPL, così da creare un supporto alle nuove strutture analizzate in precedenza. In un primo momento è stato dato un supporto allo stato ed ai suoi attributi, creando due semplici classi che li modellassero. Poi sono state implementate le funzioni di modifica dello stato, creando una gerarchia del tutto analoga alle funzioni di comparazione già esistenti. Successivamente è stato modificato il processo del PEP in modo tale che riesca ad eseguire questo nuovo tipo di funzioni. Dopo la creazione dello stato sono state estese anche le politiche, dove sono state modificate le funzioni di comparazione e create le Obligation Status. Infine è stato necessario estendere il plugin, così da permettergli di riconoscere i nuovi costrutti e di conseguenza offrire funzioni tipiche di un IDE.

La tesi del mio collega parla di come è stato possibile implementare in FACPL, grazie anche alle estensioni di cui parla questa tesi, un monitor a runtime per il supporto al controllo continuativo degli accessi.