ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Μεταπτυχιακή Εργασία

Μηχανισμοί Πιστοποίησης και Εξουσιοδότησης για Ροές Επιστημονικών Εργασιών

Χαράλαμπος Ι. Γκίκας

Ηράκλειο, Μάρτιος 2004

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Μηχανισμοί Πιστοποίησης και Εξουσιοδότησης για Ροές Επιστημονικών Εργασιών

Εργασία που υποβλήθηκε από τον

ΧΑΡΑΛΑΜΠΟ Ι. ΓΚΙΚΑ

ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Χαράλαμπος Ι. Γκίκας
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Εξεταστική Επιτροπή:

Βασίλης Χριστοφίδης, Επίκουρος Καθηγητής
Επόπτης

Δημήτρης Πλεξουσάκης, Αναπληρωτής Καθηγητής
Μέλος

Αικατερίνη Χούστη, Καθηγήτρια, Μέλος
Πανεπιστήμιο Θεσσαλίας

Δεκτή:

Δημήτρης Πλεξουσάκης, Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Μάρτιος 2004

Μηχανισμοί Πιστοποίησης και Εξουσιοδότησης για Ροές Επιστημονικών Εργασιών

Χαράλαμπος Ι. Γκίκας

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Ολοένα και περισσότεροι επιστημονικοί οργανισμοί χρησιμοποιούν για τις καθημερινές τους ερευνητικές δραστηριότητες επιστημονικούς υπολογισμούς. Αυτοί οι υπολογισμοί συνήθως εκτελούνται για μεγάλο χρονικό διάστημα και χρησιμοποιούν ένα σημαντικό ποσοστό υπολογιστικών πόρων. Για να βελτιωθεί η συνεργασία μεταξύ οργανισμών με ίδιους ή παρόμοιους τομείς, καθώς και να μοιραστεί το κόστος εκτέλεσης ενός επιστημονικού υπολογισμού, οι οργανισμοί βασίζονται στα κατανεμημένα πληροφοριακά συστήματα. Για παράδειγμα, μια υλοποίηση κατανεμημένων επιστημονικών υπολογισμών μπορεί να γίνει με την χρήση ενός κατάλληλου συστήματος διαχείρισης ροών εργασιών (WfMS).

Όμως είναι εμφανής η έλλειψη μηχανισμών πιστοποίησης και εξουσιοδότησης κατάλληλων για τις ανάγκες των επιστημονικών υπολογισμών. Τα περισσότερα μοντέλα εξουσιοδότησης ροών εργασιών διαχειρίζονται τον έλεγχο πρόσβασης σε επίπεδο συστήματος των διεργασιών και των αρχείων τους. Ακόμη προτάσεις για «καλές» άδειες δεν προσφέρονται από το σύστημα, με αποτέλεσμα οι επιστήμονες να μην γνωρίζουν εκ των προτέρων αν μπορούν να εκτελέσουν μια ροή επιστημονικών εργασιών και ούτε τι πράξεις πρέπει να κάνουν στην περίπτωση που δεν έχουν τις κατάλληλες άδειες. Τέλος δεν έχουν προβλεφθεί μηχανισμοί χρέωσης για επιστημονικούς υπολογισμούς.

Σε αυτή την εργασία προτείνουμε ένα καινούργιο μοντέλο για την εξουσιοδότηση ροών επιστημονικών εργασιών. Σε αυτό το μοντέλο, ορίζουμε ιεραρχίες χρηστών και ρόλων για να υποστηρίζουμε ένα ευέλικτο σύστημα διαχείρισης των ορισμών εξουσιοδότησης από κατανεμημένους οργανισμούς. Προτείνουμε επαρκείς αλγορίθμους για την επιβεβαίωση και προτάσεις αδειών και ρόλων κατά την επεξεργασία των εξουσιοδοτήσεων που έχουν οριστεί σε μία ροή εργασιών. Ορίζουμε ακόμη συναλλακτικές μονάδες που σε συνδυασμό με τις εξουσιοδοτήσεις παρέχουν την δυνατότητα χρέωσης. Τέλος περιγράφουμε την αρχιτεκτονική ενός συστήματος πιστοποίησης και εξουσιοδότησης που βασίζεται στο μοντέλο μας και μπορεί με ευκολία να ενοποιηθεί με συστήματα διαχείρισης ροών επιστημονικών εργασιών όπως το ARION.

Επόπτης
Βασίλης Χριστοφίδης
Επίκουρος Καθηγητής

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Authentication and Authorization Mechanisms for Scientific Workflows

Charalampos I. Gkikas

Master of Science Thesis

Department of Computer Science
University of Crete
Greece

Abstract

More and more scientific organizations are using in their everyday research activities scientific computations. These computations usually have a long duration and consume significant resources. In order to enhance scientific collaboration on the same or similar research topics, as well as to share the cost of expensive scientific computations, organizations rely on distributed information technology. For instance, the implementation of such distributed scientific computations can be achieved with the use of an appropriate workflow management system (WfMS).

However, authentication and authorization mechanisms tailored to the needs of scientific computations are still missing. Most of the authorization models for WfMSs deal with low level access of machine process and files. In addition, suggestions about "good" permissions are not provided by the system. As a result, a scientist is not aware in advance whether he/she can execute a scientific workflow neither of what actions has to undertake when the appropriate permissions are not granted by the system. Finally, charging mechanisms for scientific computations are not also foreseen.

In this thesis we propose a new authorization model for scientific workflows. In this model, we define a hierarchy of user and roles in order to support a flexible management of authorization primitives by distributed organizations. We propose adequate algorithms for verifying and suggesting permissions and roles during the compilation of the authorizations specified in a scientific workflow. We also define credits, which can be used in conjunctions with authorization in order to provide a charging functionality. Finally, we outline the architecture of an authentication and authorization system based in our model which can be easily integrated with a scientific WfMS such as ARION.

Advisor
Vassilis Christophides
Assistant Professor

Department of Computer Science
University of Crete
Greece

Αφιερωμένο στους γονείς μου που με έχουν στηρίξει στη ζωή μου μέχρι σήμερα.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα Μεταπτυχιακή Εργασία δεν θα είχε πραγματοποιηθεί χωρίς τις επισημάνσεις καθηγητών και ερευνητών των Τομέων των Πληροφοριακών και των Παράλληλων και Κατανεμημένων Συστημάτων.

Καταρχάς, θα ήθελα να ευχαριστήσω θερμά τον επόπτη καθηγητή κ. Χριστοφίδη Βασίλη για την ουσιαστική καθοδήγησή που μου πρόσφερε. Ακολούθως, επιθυμώ να ευχαριστήσω την καθηγήτρια του Πανεπιστημίου Θεσσαλίας κα. Χούστη Αικατερίνη, για το θέμα της μεταπτυχιακής εργασίας. Επίσης, θα ήθελα να ευχαριστήσω των κ. Πλεξουσάκη Δημήτρη, αναπληρωτή καθηγητή του Πανεπιστημίου Κρήτης (Π.Κ), ως μέλος της Εισηγητικής Επιτροπής, για τις πολύτιμες παρατηρήσεις τους, συντελώντας έτσι στην ολοκλήρωσή της ερευνητικής εργασίας.

Επιπροσθέτως, θα ήθελα να ευχαριστήσω: τον Κρητικό Κυριάκο, την Κατελανή Σταματίνα, τον Σαχτούρη Σταύρο, τον Ακριτίδη Περικλή, τον Γάσπαρη Στυλιανό και την Ζαφιράτου Αικατερίνη για την πολύτιμη βοήθεια που μου προσέφεραν με τις διορθώσεις τους στην διατύπωση και σύνταξη της παρούσας αναφοράς.

Ακόμη θα ήθελα να ευχαριστήσω τον Βεντούρα Νίκο, τον Παντελέρη Πασχάλη, τον Σιδηρουργό Λευτέρη, τον Αντώνη Σμαρδά, τον Σταύρου Θωμά για την παρέα που πρόσφεραν στις διάφορες εξωπανεπιστημιακές και μη δραστηριότητες που είχαμε και για την σωστή επένδυση του ελεύθερου χρόνου μου.

Τέλος, τους γονείς μου που χωρίς την έμπρακτη συμπαράσταση τους η Μεταπτυχιακή Εργασία αυτή δεν θα ήταν εφικτή.

Χαράλαμπος Γκίκας

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ-ΣΧΗΜΑΤΩΝ

1 Εισαγωγή

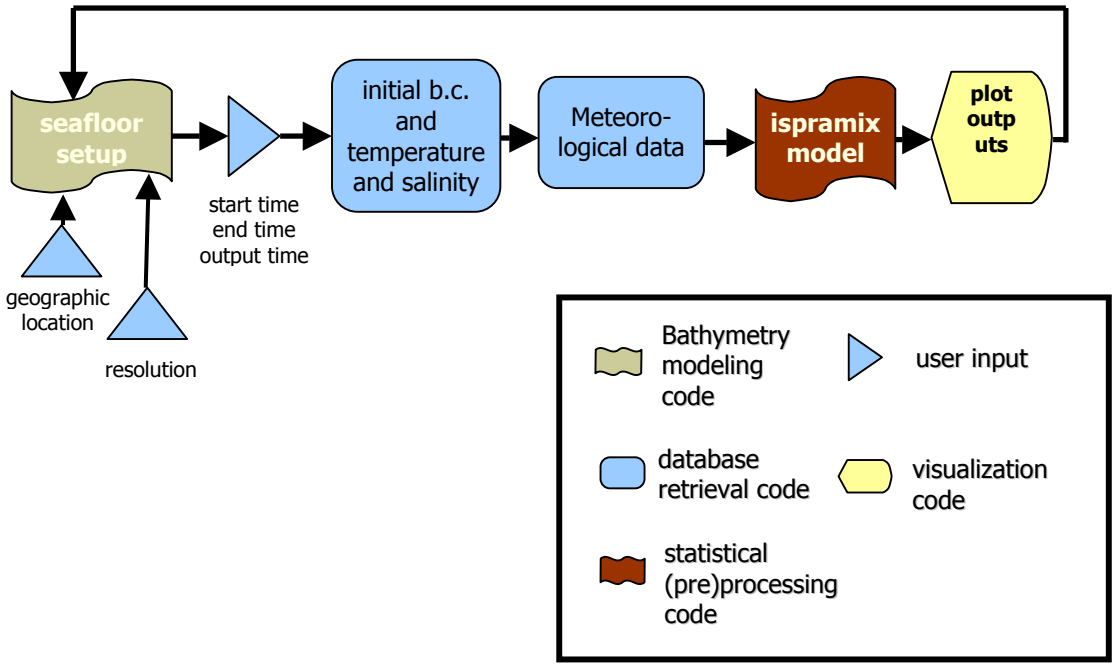
1.1 Πλαίσιο της Εργασίας

Η ραγδαία ανάπτυξη της κοινωνίας της γνώσης έχει κάνει απαραίτητη την αξιοποίηση των επιτευγμάτων της πληροφορικής σε διάφορες επιστημονικές περιοχές. Η χρήση πληροφοριακών εφαρμογών για την υποστήριξη επιστημονικών υπολογισμών είναι σήμερα ευρέως διαδεδομένη, καθώς η έρευνα σε αυτές τις περιοχές βασίζεται πλέον συστηματικά σε επιστημονικούς υπολογισμούς π.χ. για την πρόγνωση πολλών φυσικών φαινομένων μέσω προσομοιώσεων.

Παρόλη την ανάπτυξη της τεχνολογίας της πληροφορικής και των επικοινωνιών, πολλοί και σημαντικοί υπολογισμοί εξακολουθούν να είναι απαιτητικοί σε υπολογιστικούς πόρους ενώ οι επιστημονικές κοινότητες εστιάζουν σε ολοένα και πιο απαιτητικά προβλήματα. Σε αρκετές περιπτώσεις αυτό το μεγάλο υπολογιστικό κόστος είναι δυνατό να αντιμετωπισθεί με την χρήση παράλληλων μηχανών. Στην πράξη όμως μόνο ορισμένοι επιστημονικοί οργανισμοί διαθέτουν τέτοια υποδομή. Επιπλέον αναδεικνύεται επιτακτικά η ανάγκη μιας ολοένα και μεγαλύτερης συνεργασίας μεταξύ διαφορετικών επιστημονικών κοινοτήτων και οργανισμών. Για παράδειγμα, οι επιστημονικοί υπολογισμοί κάποιου οργανισμού σχετικά με ένα φυσικό φαινόμενο, μπορούν να γίνουν αρκετά πιο ακριβείς χρησιμοποιώντας τις μετρήσεις που διαθέτει κάποιος άλλος οργανισμός.

Η πλέον διαδεδομένη προσέγγιση βασίζεται στην συνεργασία των οργανισμών που ασχολούνται με το ίδιο ή παραπλήσιο επιστημονικό πεδίο. Συγκεκριμένα, ο συνολικός υπολογισμός μπορεί να κατατμηθεί και κατανεμηθεί κατάλληλα στους συνεργαζόμενους οργανισμούς. Ο κατάλληλος συνδυασμός των επιμέρους υπολογισμών επιφέρει τα επιθυμητά αποτελέσματα στην έρευνα καθώς και οικονομίες κλίμακας στην συλλογή και επεξεργασία των επιστημονικών δεδομένων.

Η υποστήριξη κατανεμημένων επιστημονικών υπολογισμών συνήθως πραγματοποιείται με τη χρήση κατάλληλων συστημάτων διαχείρισης και παρακολούθησης ροών εργασίας. Συγκεκριμένα, μια ροή επιστημονικών εργασιών αποτελείται από ένα σύνολο από διεργασίες επιστημονικών υπολογισμών, οι οποίες εκτελούνται σύμφωνα με τις απαιτήσεις ροής δεδομένων ενός συγκεκριμένου πειράματος, και μπορεί να περιλαμβάνει πολλά άτομα ή οργανισμούς.



Εικόνα 1-1 Παράδειγμα Ροής Επιστημονικών Εργασιών

Στην Εικόνα 1-1 παρουσιάζεται ένα παράδειγμα μιας ροής εργασίας που χρησιμοποιείται για την υποστήριξη των επιστημονικών υπολογισμών δύο οργανισμών: JRC και CNR. Το JRC (Joint Research Center) και το CNR (National Research Center) βρίσκονται στην "Ispra" της Ιταλίας και συνεργάζονται σε προβλήματα διαχείρισης ωκεανών. Η εν λόγω ροή επιστημονικών εργασιών χρησιμοποιείται για την προσομοίωση δυο ωκεανογραφικών μοντέλων. Δύο οργανισμοί απαιτούνται για την ολοκλήρωση της περιγραφόμενης ροής εργασίας. Ο λόγος είναι ότι η προσομοίωση που γίνεται στον οργανισμό CNR απαιτεί διάφορες μετρήσεις που παρέχονται μόνο από τον οργανισμό αυτό. Οι υπόλοιπες διεργασίες επιστημονικών υπολογισμών της εν λόγω ροής εργασίας γίνονται στον οργανισμό JRC. Το παράδειγμα αυτό θα χρησιμοποιεί σε αυτή την αναφορά σαν κεντρικό παράδειγμα για την αποσαφήνιση των εννοιών.

Ένα κατανεμημένο σύστημα διαχείρισης ροών επιστημονικών εργασιών προσφέρει την δυνατότητα δημιουργίας, εκτέλεσης, και επίβλεψης ροών εργασιών σε διαφορετικούς οργανισμούς. Ένα τέτοιο σύστημα είναι και το ARION [HLCPVPKSG02], το οποίο αναπτύχθηκε από το Ινστιτούτο Τεχνολογίας και Ερευνάς στα πλαίσια ενός ευρωπαϊκού έργου της κοινωνίας της πληροφορίας σχετικά με επιστημονικά δεδομένα που αφορούν το θαλάσσιο περιβάλλον. Ένα από τα κύρια πλεονεκτήματα του συστήματος ARION είναι ότι παρέχει την δυνατότητα περιγραφής των δεδομένων που πιθανόν να χρειάζονται ως είσοδος ή να

παράγονται ως έξοδος από μία διεργασία επιστημονικών υπολογισμών (π.χ. για την προσομοίωση θαλάσσιων φαινομένων).

Η αρχιτεκτονική του συστήματος ARION, χρησιμοποιεί σύγχρονα εργαλεία: (α) Στηρίζεται στο πακέτο εφαρμογών "RDF Suite", για να διαχειρίζεται αποδοτικά και εύκολα τα μετά-δεδομένα (metadata) που υπάρχουν. Τα μετά-δεδομένα περιγράφουν τα δεδομένα επιστημονικών προγραμμάτων (data sets), τα ίδια τα επιστημονικά προγράμματα καθώς και τις ροές εργασίας. Με την χρήση των εν λόγω εφαρμογών είναι δυνατή η εύρεση μιας ροής εργασίας από την ίδια, ή από τα αποτελέσματα που παράγει. (β) Ένα υποσύστημα από κινούμενους πράκτορες που μπορούν να μετακινηθούν σε οποιαδήποτε μηχανή μέσα στο κατανεμημένο σύστημα. Αυτό το υποσύστημα αποτελείται από ένα δίκτυο με αρκετούς υπολογιστές, οι οποίοι ανήκουν σε διάφορους οργανισμούς. Οι πράκτορες μπορούν να εκτελέσουν κατάλληλα τις διεργασίες που ανατίθενται καθώς επίσης και να μεταφέρουν τα δεδομένα στο επιθυμητό μέρος. (γ) Ένα σύστημα διαχείρισης των ροών επιστημονικών εργασιών που αναλαμβάνει αφενός να δημιουργήσει τα στιγμιότυπα των ροών εργασιών και αφετέρου να ορίσει τις κατάλληλες οδηγίες για το υποσύστημα κινούμενων πρακτόρων ώστε να εκτελεστούν κατάλληλα οι διεργασίες.

Για να θεωρηθεί το σύστημα ARION ολοκληρωμένο χρειάζεται επίσης έναν μηχανισμό πιστοποίησης και εξουσιοδότησης των χρηστών. Με αυτού του είδους τον μηχανισμό θα προσφερόταν ο κατάλληλος έλεγχος ώστε να βρεθούν οι ροές εργασίας που μπορεί ένας χρήστης να εκτελέσει καθώς και τα αντικείμενα στα οποία μπορεί να αποκτήσει πρόσβαση. Ακόμη, θα ήταν επιθυμητό ορισμένοι χρήστες να χρεώνονται κατάλληλα από το εν λόγω σύστημα για την εκτέλεση συγκεκριμένων διεργασιών.

Στην εργασία αυτή αναλύουμε διάφορα μοντέλα πιστοποίησης και εξουσιοδότησης, που έχουν προταθεί από διακεκριμένους ερευνητές. Τα μοντέλα αυτά είτε επικεντρώνονται στην σχέση χρήστη – αντικειμένου, είτε στο πως πρέπει να εξουσιοδοτούνται οι διεργασίες. Τα μοντέλα που στηρίζονται στην σχέση χρήστη – αντικειμένου δεν προσφέρουν μηχανισμούς που να κάνουν εφικτή την εξουσιοδότηση ροών εργασίας. Τα μοντέλα που επικεντρώνονται σε διεργασίες ή ροές εργασίας προσφέρουν μηχανισμούς που ορίζουν τον τρόπο με τον οποίο γίνονται οι εξουσιοδοτήσεις των διεργασιών, αλλά στηρίζονται στο γεγονός ότι κάποιο άτομο ή ρόλος «πρέπει» να εκτελέσει κάποια συγκεκριμένη διεργασία και

δεν υποστηρίζουν ότι ένας χρήστης ή ο ρόλος μπορεί να είναι απλά εξουσιοδοτημένος για την εκτέλεση μιας διεργασίας. Κανένας από τους δύο αυτούς τύπους μοντέλων δεν προσφέρει τεχνικές επιβεβαίωσης της δυνατότητας εκτέλεσης από έναν χρήστη μιας ολόκληρης ροής εργασίας. Τα μοντέλα αυτά προσπαθούν να κάνουν μια εκτίμηση για το πόσο χρόνο θα εκτελεστεί μια διεργασία, ώστε να κάνουν διαχείριση των διαφόρων αδειών του συστήματος. Όμως αυτή είναι μια εκτίμηση και πρέπει να υπάρχουν καλές εκτιμήσεις για να λειτουργήσουν σωστά. Οι εκτιμήσεις αυτές είναι σχετικά δύσκολες στις περιπτώσεις επιστημονικών υπολογισμών.

Πιστεύουμε ότι μπορεί να οριστεί ένα μοντέλο πιστοποίησης και εξουσιοδότησης για ροές εργασίας που να προσφέρει τους κατάλληλους μηχανισμούς για επιβεβαίωση της δυνατότητας εκτέλεσης της ροής εργασίας από έναν χρήστη. Ταυτόχρονα το μοντέλο αυτό θα μπορεί να εκφράσει τις ευκολίες και τα πλεονεκτήματα των μοντέλων εξουσιοδοτήσεων που μελετήθηκαν.

1.2 Συνεισφορά της Εργασίας

Στην εργασία αυτή προτείνουμε ένα τυπικό μοντέλο εξουσιοδότησης ροών εργασιών που βασίζεται σε δύο ιεραρχίες: την ιεραρχία χρηστών και την ιεραρχία ρόλων. Η ιεραρχία χρηστών μας ορίζει που πρέπει να καθορίζονται οι χρήστες, και μας προσφέρει μια συσχέτιση των οργανισμών μεταξύ τους. Η ιεραρχία ρόλων μας προσφέρει την καλύτερη περιγραφή των ρόλων που μπορεί να έχει κάποιος χρήστης σε έναν οργανισμό. Με το μοντέλο εξουσιοδότησης αποκτούμε τις εξής λειτουργικότητες: (α) Οι χρήστες να ανήκουν σε διαφορετικούς οργανισμούς. (β) Οι χρήστες να μπορούν να έχουν τους ρόλους που ανήκουν σε διάφορους οργανισμούς. (γ) Να δίνεται η δυνατότητα για εξουσιοδότηση διεργασιών επιστημονικών υπολογισμών. (δ) Να υποστηρίζεται η χρέωση της εκτέλεσης επιστημονικών διεργασιών χρησιμοποιώντας κατάλληλες συναλλακτικές μονάδες.

Στο μοντέλο αυτό, προτείνονται αλγόριθμοι που κάνουν μια εκτίμηση της δυνατότητας του χρήστη να εκτελέσει μια ροή εργασίας. Στην περίπτωση που κάποιος χρήστης δεν είναι ικανός να εκτελέσει μια ροή επιστημονικών εργασιών, αναφέρεται αλγόριθμος που προτείνει τους ρόλους που είναι ικανοί να εκτελέσουν τις διεργασίες επιστημονικών υπολογισμών που δεν μπορεί ο χρήστης. Τέλος η εργασία αυτή προτείνει μια αρχιτεκτονική πιστοποίησης και εξουσιοδότησης που μέρος της οποίας έχει εφαρμοστεί στο σύστημα ARION. Η αρχιτεκτονική αυτή,

εκτός από την διασφάλισης της ασφάλειας, ορίζει τον τρόπο με τον οποίο πρέπει να γίνει η διαχείριση των χρηστών, των ρόλων και των εξουσιοδοτήσεων.

Προτείνοντας αυτό το μοντέλο εξουσιοδότησης και την αρχιτεκτονική πιστοποίησης καταφέρνουμε να αποκτήσουμε μια ασφαλή αρχιτεκτονική για συστήματα διαχείρισης ροών επιστημονικών εργασιών που η εκτέλεση τους γίνεται σε πολλούς οργανισμούς. Οι εξουσιοδοτήσεις είναι αρκετά περιγραφικές και εύκολες στην διαχείριση τους. Οι αλγόριθμοι που δίνονται προσφέρουν σωστή επαλήθευση και επιβεβαίωση των ορισμών των εξουσιοδοτήσεων.

1.3 Οργάνωση της Αναφοράς

Η αναφορά της μεταπτυχιακής εργασίας οργανώνεται ως εξής:

- Στο κεφάλαιο 2 αναφέρουμε τις προηγούμενες εργασίες που έχουν γίνει σε θέματα πιστοποίησης και εξουσιοδότησης. Αναφέρουμε μερικές από τις βασικές τεχνικές πιστοποίησης μαζί με τα πλεονεκτήματα και μειονεκτήματα της κάθε τεχνικής. Μετά αναλύουμε τους μηχανισμούς εξουσιοδότησης που υπάρχουν χωρίζοντας τα σε τρεις κατηγορίες: (α) Εξουσιοδοτήσεις που ορίζονται από την σχέση χρήστη – αντικειμένου. (β) Εξουσιοδοτήσεις που ορίζονται κατά την διάρκεια εκτέλεσης μίας διεργασίας. (γ) Τρόποι Εξουσιοδοτήσεων για κατανεμημένα συστήματα.
- Στο κεφάλαιο 3 αναφέρουμε το μοντέλο εξουσιοδότησης που προτείνουμε για ροές επιστημονικών εργασιών. Δείχνουμε τυπικούς ορισμούς για τις ροές εργασίας και τις εξουσιοδοτήσεις. Παρουσιάζουμε αλγορίθμους επιβεβαίωσης, που ελέγχουν κατά πόσο ένας χρήστης μπορεί να εκτελέσει κάποια ροή εργασίας.
- Στο κεφάλαιο 4 παρουσιάζουμε μια σχεδίαση για ένα σύστημα πιστοποίησης και εξουσιοδότησης ροών επιστημονικών εργασιών σε ένα κατανεμημένο περιβάλλον εκτέλεσης. Ένα μέρος του συστήματος αυτού έχει ενοποιηθεί με τον υπάρχον σύστημα του ARION. Το συνολικό σύστημα έχει φυσικά την δυνατότητα χρήσης του από το σύστημα ARION.

- Στο κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα της εργασίας μας. Τέλος αναφερόμαστε σε μερικά θέματα που αποτελούν μελλοντικές κατευθύνσεις έρευνας.

2 Πιστοποίηση χρηστών και Εξουσιοδοτημένη Πρόσβαση για Αρχεία και Διαδικασίες

Η ανάγκη για πιστοποίηση καθώς και για εξουσιοδότηση εμφανίζεται όταν ένας χρήστης ή μια διεργασία επιθυμούν να αποκτήσουν άδειες πρόσβασης σε κάποιο πόρο ενός πληροφοριακού συστήματος. Έχουν προταθεί στην βιβλιογραφία ποικίλες μέθοδοι πιστοποίησης και εξουσιοδότησης. Στο πρώτο μέρος αυτού του κεφαλαίου θα παρουσιαστούν οι μηχανισμοί πιστοποίησης καθώς και τα διάφορα διαπιστευτήρια που χρειάζονται. Στο δεύτερο μέρος θα παρουσιαστούν μοντέλα εξουσιοδοτημένης πρόσβασης που στηρίζονται σε εκφραστικές σχέσεις μεταξύ προσώπων (χρηστών) και αντικείμενων. Στο τρίτο μέρος θα παρουσιαστούν μοντέλα δυναμικών εξουσιοδοτήσεων για διεργασίες που δημιουργούν εξουσιοδοτήσεις. Στο τέταρτο μέρος θα δούμε πώς μπορούμε να υποστηρίζουμε μηχανισμούς εξουσιοδότησης σε μεγάλα κατανεμημένα συστήματα. Και στο τελευταίο μέρος του κεφαλαίου θα παρουσιαστεί μια πρότυπη γλώσσα για τον ορισμό πολιτικών εξουσιοδοτημένης πρόσβασης που βασίζεται στο πρότυπο XML καθώς και κατανεμημένοι μηχανισμοί ελέγχου πρόσβασης.

Η δομή αυτού του κεφαλαίου έχει σκοπό την εξερεύνηση πάνω στα ήδη υπάρχοντα μοντέλα και τις εφαρμογές που προσφέρουν εξουσιοδότηση και έλεγχο πρόσβασης σε αρχεία ή διεργασίες. Ξεκινώντας από την ομάδα μοντέλων του Προαιρετικού Ελέγχου Πρόσβασης θα παρατηρήσουμε τα μειονεκτήματα τους και τους λόγους που δεν μπορούν να χρησιμοποιηθούν σε ένα μεγαλύτερο κατανεμημένο σύστημα. Από τα υπόλοιπα μοντέλα ελέγχου πρόσβασης που στηρίζονται στις εκφραστικές σχέσεις χρήστη – αντικειμένου θα προσπαθήσουμε να εξάγουμε χρήσιμες σχέσεις χρηστών μεταξύ αντικειμένων ή ακόμη σχέσεις χρηστών μεταξύ τους. Από τα μοντέλα εξουσιοδοτήσεων για διεργασίες θα παρουσιάσουμε τα προβλήματα και τις απαιτήσεις που έχουν οι διεργασίες ως προς τις εξουσιοδοτήσεις που χρειάζονται και ακόμη θα παρουσιάσουμε μερικές προτάσεις επίλυσης. Τέλος θα παρουσιαστούν διάφορα πρότυπα ορισμού εξουσιοδοτήσεων, πως περιγράφονται και τι υποστηρίζουν. Στον επίλογο αυτού του κεφαλαίου θα δούμε αν μπορούν να συνδυαστούν στοιχεία από τα μοντέλα αυτά,

για μια ολοκληρωμένη επίλυση, για την υλοποίηση ενός κατανεμημένου μηχανισμού εξουσιοδότησης για ροές επιστημονικών εργασιών.

2.1 Μηχανισμοί Πιστοποίησης

Ένας μηχανισμός πιστοποίησης μας επιτρέπει να πιστοποιήσουμε ότι μια οντότητα (π.χ. ένας χρήστης, ένα πρόγραμμα) είναι πράγματι αυτό που ισχυρίζεται ότι είναι. Οι μηχανισμοί ποικίλουν αρκετά, αλλά η βασική ιδέα είναι ότι: «Η πιστοποίηση στηρίζεται σε κάποια πληροφορία που την έχει ή την γνωρίζει η οντότητα που θέλει να πιστοποιηθεί». Οι διάφοροι μηχανισμοί προσπαθούν να ορίσουν αυτή την πληροφορία έτσι ώστε να είναι εύκολος ο χειρισμός της από την πιστοποιούμενη οντότητα ή να είναι αρκετά δύσκολη η υποκλοπή της.

2.1.1 Διαπιστευτήρια

Η πληροφορία που χρησιμοποιείται μαζί με το διακριτικό όνομα της οντότητας λέγεται «διαπιστευτήριο» (credential) και θα δούμε παρακάτω μερικές από της μορφές που παίρνει:

2.1.1.1Κωδικός Πρόσβασης

Το σύνθημα (password) είναι μια σειρά χαρακτήρων που την γνωρίζει η οντότητα που θέλει να πιστοποιηθεί, καθώς και ο μηχανισμός πιστοποίησης. Κάθε μία οντότητα έχει το δικό της κωδικό πρόσβασης για να πιστοποιηθεί από το σύστημα. Κάθε φορά που πρέπει να πιστοποιηθεί, παρουσιάζει μαζί με το όνομα της (διακριτικό όνομα της οντότητας ώστε να μπορεί να διακρίνεται από τις υπόλοιπες οντότητες) και τον κωδικό πρόσβασης της. Αν αυτά τα δύο συμπίπτουν με αυτά που γνωρίζει ο μηχανισμός πιστοποίησης, τότε πιστοποιείται η οντότητα αυτή, οπότε μπορεί να εξουσιοδοτηθεί ή όχι για την πρόσβαση της στους πόρους του συστήματος.

2.1.1.2Κρυπτογραφημένος Κωδικός Πρόσβασης

Σε αυτή την περίπτωση ο μηχανισμός πιστοποίησης δεν χρειάζεται κάποιον συγκεκριμένο κωδικό πρόσβασης από την οντότητα, αλλά το αποτέλεσμα μιας συνάρτησης που θα έχει ως παράμετρο τον κωδικό πρόσβασης της. Η συνάρτηση αυτή είναι μη αντιστρέψιμη. Συνήθως χρησιμοποιούνται συναρτήσεις διασποράς (hash functions). Μία τέτοια συνάρτηση είναι η crypt, που χρησιμοποιείται από τα συστήματα Unix. Η οντότητα εκτελεί τη γνωστή αυτή συνάρτηση περνώντας ως

παράμετρο τον κωδικό πρόσβασης και παρουσιάζει στον μηχανισμό πιστοποίησης το αποτέλεσμα της εκτέλεσης. Παρομοίως, ο μηχανισμός πιστοποίησης περνάει τον κωδικό πρόσβασης από την ίδια συνάρτηση και ελέγχει αν το αποτέλεσμα που έλαβε και αυτό που παρουσίασε η οντότητα ταιριάζουν. Αν ταιριάζουν, τότε η οντότητα πιστοποιείται και μπορεί να εξουσιοδοτηθεί ή όχι για την πρόσβαση της στο υπόλοιπο σύστημα.

2.1.1.3Δημόσια Κλειδιά και Προσωπικά Κλειδιά

Η ιδέα των κλειδιών στηρίζεται σε αλγορίθμους κρυπτογράφησης όπως είναι ο αλγόριθμος DSA [MOV96]. Η κάθε οντότητα έχει ένα σύνολο από αριθμούς (συνήθως δύο), που λέγονται προσωπικοί αριθμοί ή προσωπικά κλειδιά, τους οποίους χρησιμοποιεί για να εκτελέσει μια συγκεκριμένη πράξη (συνήθως στηρίζεται στον πολλαπλασιασμό). Παράγεται, έτσι, ένας αριθμός, που ονομάζεται δημόσιο κλειδί. Το σημαντικό είναι ότι δεν υπάρχει συνάρτηση ή εύκολος τρόπος υπολογισμού των προσωπικών κλειδιών αν κάποιος έχει μόνο το δημόσιο κλειδί. Το επόμενο κύριο χαρακτηριστικό των αλγορίθμων αυτών είναι η κρυπτογράφηση και η υπογραφή που μπορεί να πραγματοποιήσουν χρησιμοποιώντας τα κλειδιά αυτά. Τα κλειδιά αυτά είναι γνωστά και ως ψηφιακά πιστοποιητικά (digital certificates).

Η κρυπτογράφηση είναι η διαδικασία σύμφωνα με την οποία, γνωρίζοντας το δημόσιο κλειδί κάποιας οντότητας, μπορούμε να κρυπτογραφήσουμε τα δεδομένα μας χρησιμοποιώντας το κλειδί αυτό και ο μόνος που μπορεί να τα αποκρυπτογραφήσει είναι η οντότητα που έχει τα προσωπικά κλειδιά. Η υπογραφή είναι η πράξη που μπορεί να κάνει μια οντότητα που έχει κάποια προσωπικά κλειδιά πάνω σε κάποια δεδομένα. Όποιος έχει το δημόσιο κλειδί αυτού που το υπέγραψε μπορεί να ελέγξει τη γνησιότητα των υπογεγραμμένων δεδομένων.

Δεδομένων των μηχανισμών κρυπτογράφησης και υπογραφής, τα κλειδιά αυτά μπορεί να χρησιμοποιηθούν ως διαπιστευτήρια για ένα μηχανισμό πιστοποίησης. Ο μηχανισμός πιστοποίησης στέλνει ένα αντικείμενο στην οντότητα για να το υπογράψει, το οποίο είναι κρυπτογραφημένο με το δημόσιο κλειδί της. Η οντότητα το αποκρυπτογραφεί, το υπογράφει και τέλος το κρυπτογραφεί με το δημόσιο κλειδί του μηχανισμού πιστοποίησης. Ο μηχανισμός πιστοποίησης αποκρυπτογραφεί το μήνυμα και ελέγχει αν υπογράφτηκε από την οντότητα χρησιμοποιώντας το δημόσιο κλειδί της.

2.1.1.4Βιομετρικά Χαρακτηριστικά

Τα βιομετρικά χαρακτηριστικά ενός φυσικού προσώπου μπορεί να είναι το δακτυλικό αποτύπωμα του, η φωνή του, η ίριδα του ματιού του, κλπ. Οι μηχανισμοί πιστοποίησης που βασίζονται σε βιομετρικά χαρακτηριστικά προσφέρουν υψηλή ασφάλεια. Από την άλλη οι μηχανισμοί αυτοί είναι επιρρεπείς σε λάθη και υπάρχει πιθανότητα μια οντότητα σε λίγο διαφορετικές περιβαλλοντολογικές συνθήκες να μην μπορεί να πιστοποιηθεί. Τέλος, οι εν λόγω μηχανισμοί είναι δύσκολο να υλοποιηθούν για συστήματα που έχουν απομακρυσμένη πρόσβαση, γιατί οι χρήστες πρέπει να έχουν συγκεκριμένο υλικό (hardware) που να υποστηρίζει την μέτρηση αυτών των βιομετρικών τους χαρακτηριστικών.

2.1.2 Απομακρυσμένη Πιστοποίηση

Σε συστήματα όπου απομακρυσμένοι χρήστες θέλουν να πιστοποιηθούν, υπάρχει μία δυσκολία στον τρόπο που θα περάσουν τα διαπιστευτήρια πάνω από το δίκτυο ώστε να μην υποκλαπούν κατά την μεταφορά τους. Ακόμη, πρέπει να λάβουμε υπόψη μας ότι ο απομακρυσμένος χρήστης πρέπει να έχει τα μέσα για να χειριστεί τα διαπιστευτήρια. Τα βιομετρικά χαρακτηριστικά δεν μπορούν να χρησιμοποιηθούν για απομακρυσμένη πρόσβαση, όχι μόνο για τον λόγο που αναφέρθηκε στο προηγούμενο κεφάλαιο, αλλά γιατί υπάρχει και η πιθανότητα υποκλοπής της πληροφορίας τους κατά την μεταφορά τους.

Ένας από τους πλέον διαδεδομένους τρόπους αντιμετώπισης του προβλήματος αυτού είναι η χρήση προσωπικών και δημόσιων κλειδιών, προκειμένου να επιτευχθεί ένα ασφαλές κανάλι επικοινωνίας και να πιστοποιηθούν και οι δύο πλευρές. Αυτός είναι ο τρόπος που δουλεύει η πιστοποίηση σε πλέγματα υπολογισμού (computational grid). Όμως, όταν η εφαρμογή στηρίζεται πάνω στον παγκόσμιο ιστό (world wide web), όπου χρησιμοποιούνται συνήθης φυλλομετρητές ιστοσελίδων (web browsers) που δεν προσφέρουν τον μηχανισμό αμοιβαίας πιστοποίησης και τον δύο πλευρών, απαιτείται μια κρυπτογραφημένη σύνδεση ώστε να σταλθούν τα κρυπτογραφημένα ή μη διαπιστευτήρια μέσω αυτής.

2.2 Εξουσιοδότηση βασισμένη σε πρόσωπα και αντικείμενα

Μερικά από τα πιο διάσημα μοντέλα εξουσιοδοτημένης πρόσβασης ελέγχουν τις σχέσεις που υπάρχουν μεταξύ ενός χρήστη (subject) και ενός

αντικειμένου (object). Θα παρουσιαστούν στην συνέχεια οι δύο μεγάλες οικογένειες αυτών των μοντέλων: Προαιρετικός και Υποχρεωτικός Έλεγχος Πρόσβασης. Για κάθε οικογένεια θα παρουσιάσουμε τα κυριότερα μοντέλα της καθώς και τα πλεονεκτήματα και μειονεκτήματα τους.

2.2.1 Προαιρετικός Έλεγχος Πρόσβασης

Ο προαιρετικός έλεγχος πρόσβασης (Discretionary Access Control) προβλέπει ότι τα δικαιώματα πρόσβασης πάνω σε ένα αντικείμενο (π.χ. ένα αρχείο) ορίζονται από ένα χρήστη (ή ομάδα χρηστών) στον οποίο ανήκει το αντικείμενο. Ο έλεγχος είναι προαιρετικός με την έννοια του ότι ένας χρήστης μπορεί να αναθέσει, να ανακαλέσει, ή να σβήσει τις άδειες πρόσβασης από τα αντικείμενα που του ανήκουν.

Ο προαιρετικός έλεγχος πρόσβασης είναι πιο διαδεδομένος σήμερα και χρησιμοποιείται από τα περισσότερα λειτουργικά σύστημα για την ασφάλεια των δεδομένων των χρηστών. Ο χρήστης, ιδιοκτήτης κάποιου αντικειμένου, έχει το δικαίωμα να θέτει όποια δικαιώματα πρόσβασης θέλει, όπως αυτός επιθυμεί.

Το μεγάλο πλεονέκτημα του Προαιρετικού Ελέγχου Πρόσβασης είναι ότι είναι αρκετά ευέλικτος έτσι ώστε να μπορεί να χρησιμοποιηθεί από πολλά συστήματα. Ο διαχειριστής των συστημάτων, αν και του δίνεται η δυνατότητα να αλλάζει τα δικαιώματα πρόσβασης των αντικειμένων των χρηστών, δε χρειάζεται να ασχολείται με αυτά. Όμως, αυτό δεν είναι αρκετό για τα συστήματα μεγάλης ασφάλειας (high assurance), καθώς ο προαιρετικός έλεγχος πρόσβασης δεν μπορεί να υποστηρίξει τις αυξημένες απαιτήσεις ασφαλείας των εμπορικών εφαρμογών (π.χ. η εφαρμογή που χειρίζεται την διανομή συγκεκριμένων εγγράφων σε μια εταιρία). Τέλος, ο εν λόγω έλεγχος πρόσβασης υποφέρει από το πρόβλημα του δούρειου ίππου (Trojan horse problem) που σημαίνει ότι αν ένας κακοπροαίρετος χρήστης αποκτήσει ιδιοκτησία σε κάποιο αντικείμενο, μπορεί να μεταφέρει τα δικαιώματά του και σε άλλους κακοπροαίρετους χρήστες.

2.2.1.1Ομάδες Χρηστών

Πάρα πολλές εφαρμογές και συστήματα υπολογιστών που χρησιμοποιούνται στην αγορά σήμερα έχουν αναπτύξει μηχανισμούς ασφαλείας που στηρίζονται στον προαιρετικό έλεγχο πρόσβασης, όπως τα λειτουργικά συστήματα τύπου Unix [S00, MFCF02], Windows NT/2000/XP [MFCF02, MS01, SBDGHCGJ01], ή συστήματα

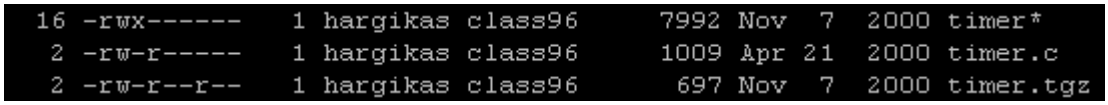
τύπου WebDAV [WEBDAV00], AFS [CSLD01] και άλλα. Αυτό συμβαίνει κυρίως γιατί είναι δύσκολο σε διαδεδομένα συστήματα να προσφέρονται κανόνες ασφαλείας για μεμονωμένους χρήστες, οργανισμούς, εταιρίες ιδρύματα και άλλα.

Συνήθως σε αυτά τα συστήματα ορίζονται κάποιες βασικές άδειες χειρισμού δεδομένων όπως ανάγνωση, εγγραφή και εκτέλεση. Τα συστήματα αυτά δίνουν την ελευθερία στους χρήστες και στον διαχειριστή τους να ορίζουν τις κατάλληλες άδειες που χρειάζονται.

Εντούτοις, έχει φανεί αρκετά δύσκολο, ένας χρήστης (ακόμη και ο διαχειριστής) να καταφέρει να δώσει άδειες (ή να τις αφαιρέσει) σε όλους του πιθανούς χρήστες των αντικειμένων του. Ως λύση αυτού του προβλήματος μερικά από τα συστήματα αυτά όρισαν τις ομάδες χρηστών (user group). Μια ομάδα χρηστών είναι ένα σύνολο από χρήστες όπου μπορούν να διεκδικήσουν τα δικαιώματα και τις άδειες που έχουν δοθεί στην ομάδα αυτή.

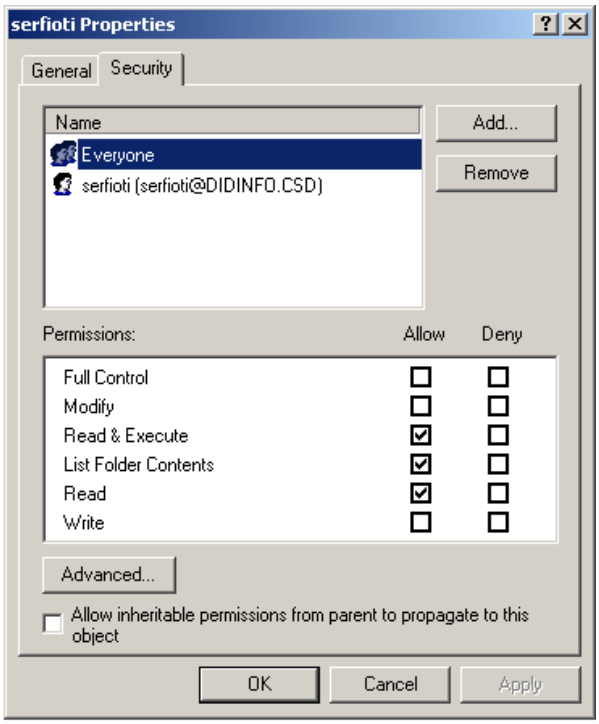
Σε συστήματα τύπου Unix, Mac OSX κλπ ένας χρήστης μπορεί να ανήκει σε πολλές διαφορετικές ομάδες. Στα αντικείμενα, όμως, (συνήθως αρχεία) μπορεί να οριστούν άδειες μόνο για τον ιδιοκτήτη, για μία ομάδα χρηστών και για τους υπόλοιπους χρήστες. Οι άδειες είναι εγγραφής, ανάγνωσης και εκτέλεσης του αντικειμένου αυτού. Ο ιδιοκτήτης του αντικειμένου μπορεί να αλλάξει την ομάδα χρηστών που έχει οριστεί για το αρχείο αρκεί να είναι και αυτός μέλος της. Οπότε, βλέπουμε ότι ένας χρήστης μπορεί να δηλώσει άδειες για τον εαυτό του, κάποια από την ομάδα χρηστών στην οποία ανήκει και για όλους τους υπόλοιπους χρήστες, όπως φαίνεται στην Εικόνα 2-1 που έχουν οριστεί άδειες για 3 αρχεία (timer, timer.c, timer.tgz). Στο αρχείο “timer” έχουν οριστεί άδειες ανάγνωσης, εγγραφής και εκτέλεσης μόνο για τον ιδιοκτήτη του αρχείου που είναι ο χρήστης “hargikas”. Για το αρχείο “timer.c” ο χρήστης “hargikas” έχει άδειες ανάγνωσης και εγγραφής αλλά και όλοι οι χρήστες που ανήκουν στην ομάδα “class96” μπορούν να το διαβάσουν. Τέλος για το αρχείο “timer.tgz” ορίζονται παρόμοιες άδειες για τον χρήστη “hargikas” και την ομάδα “class96” με το αρχείο “timer.c”, αλλά έχει οριστεί και η άδεια ανάγνωσης για όλους τους χρήστες του συστήματος. Επειδή αυτός ο τρόπος ελέγχου πρόσβασης μπορεί να είναι περιοριστικός για κάποιες εφαρμογές, δίνεται επίσης η δυνατότητα στον χρήστη να ορίσει ότι κάποιος μπορεί να εκτελέσει ένα αρχείο σαν να ήταν ο ιδιοκτήτης (setUID). Με αυτόν τον τρόπο επιτρέπει να εκχωρούνται άδειες που προηγουμένως δεν ήταν δυνατές. Για παράδειγμα, αυτός ο τελευταίος μηχανισμός χρησιμοποιείται συχνά από τους

διαχειριστές ενός μηχανήματος στην εκκίνηση ενός συστήματος για να εκτελέσουν μερικές εργασίες χρηστών.



Εικόνα 2-1 Άδειες σε Unix σύστημα

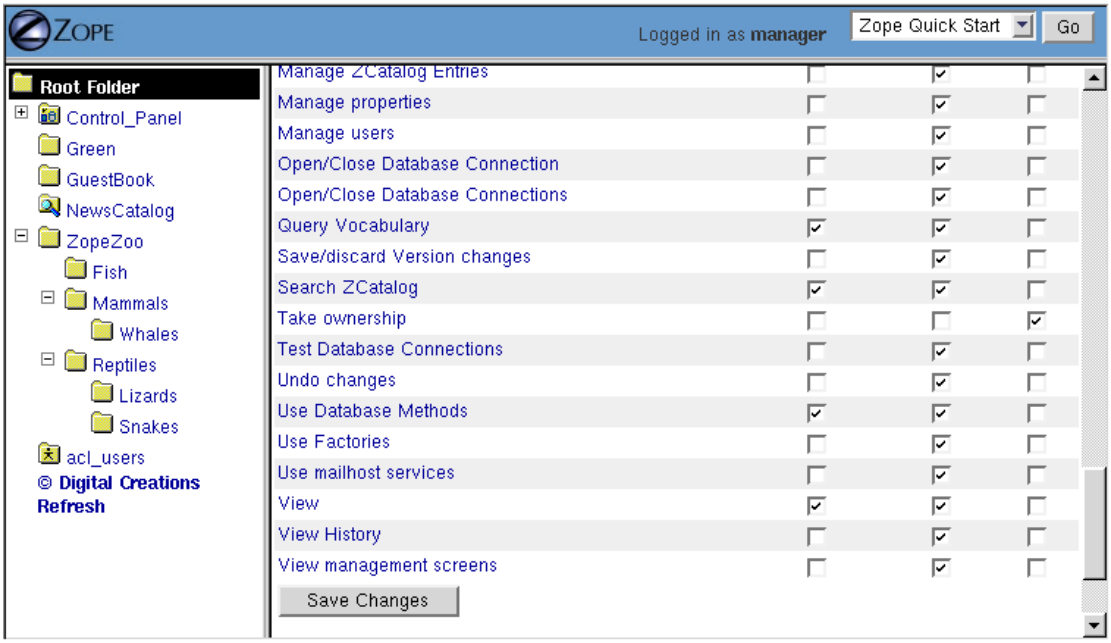
Σε συστήματα Windows υπάρχουν πάλι ομάδες χρηστών, αλλά ένα αντικείμενο μπορεί να έχει άδειες τόσο για πολλούς ξεχωριστούς χρήστες, όσο και για πολλές ομάδες χρηστών. Μάλιστα ο χρήστης μπορεί να θέσει άδειες για ομάδες χρηστών στις οποίες δεν ανήκει (σε αντίθεση με τα συστήματα Unix). Υπάρχει τέλος και μια ειδική ομάδα χρηστών (Everyone) που περιέχει όλους του χρήστες που χρησιμοποιούν το σύστημα. Οι άδειες που μπορούν να οριστούν για ένα αντικείμενο, σε αντίθεση με τα συστήματα Unix, περιλαμβάνουν και αρνήσεις αδειών. Οι αρνήσεις αδειών μπορούν να οδηγήσουν μερικές φορές σε καταστάσεις όπου το σύστημα να μην ξέρει αν θα πρέπει να αφήσει την πρόσβαση σε έναν χρήστη ή όχι. Για παράδειγμα έστω ότι ένας χρήστης ανήκει στις ομάδες χρηστών Α, Β και για ένα αντικείμενο υπάρχει η άδεια για ανάγνωση από την ομάδα Α και ακόμη υπάρχει η άρνηση της άδειας για ανάγνωση από την ομάδα Β. Για την επίλυση πιθανών συγκρούσεων στις άδειες έχουν οριστεί προτεραιότητες: Αν κάποια άδεια έχει οριστεί αναφορικά με έναν χρήστη τότε έχει την μέγιστη προτεραιότητα σε σχέση με τις υπόλοιπες άδειες που έχουν ανατεθεί σε ομάδες χρηστών, οπότε αυτή είναι και η άδεια που θα εφαρμοστεί. Αν δεν μπορεί να επιλυθεί το πρόβλημα με τον προηγούμενο κανόνα, τότε προτεραιότητα έχει η άρνηση της άδειας. Στην Εικόνα 2-2 φαίνονται οι άδειες που έχουν οριστεί για την ομάδα “Everyone”, οι οποίες αφορούν στην ανάγνωση και εκτέλεση αρχείων που βρίσκονται στον συγκεκριμένο κατάλογο, καθώς και στη δυνατότητα λήψης της λίστας των αρχείων που υπάρχουν στον κατάλογο αυτό.



Εικόνα 2-2 Άδειες σε ένα Windows σύστημα

Υπάρχουν και συστήματα ελέγχου πρόσβασης που δίνουν τις άδειες μόνο σε ομάδες χρηστών και όχι στους ίδιους του χρήστες. Ένα τέτοιο σύστημα είναι το ZOPE [ZOPE03] που ουσιαστικά μπορεί να στείλει σε έναν απομακρυσμένο χρήστη δυναμικά παραγόμενες σελίδες μορφής HTML, XML και άλλων προτύπων. Οι άδειες είναι πάρα πολλές (σε αντίθεση με τα δύο προηγούμενα συστήματα) γιατί κάθε αντικείμενο μπορεί να ορίσει τις δικές του άδειες, όπως φαίνεται και στην Εικόνα 2-3. Στην εικόνα[1] αυτή παρατηρούμε ότι για το βασικό αντικείμενο του Zope που είναι ο αρχικός κατάλογος (/) (το αντικείμενο που είναι στην ρίζα του δέντρου των αντικειμένων και περιέχει όλα τα αντικείμενα του Zope) ορίζονται πολλές άδειες. Σε αυτό το σύστημα κάθε αντικείμενο έχει ένα πίνακα πρόσβασης (access matrix) όπου ορίζονται σε αυτόν όλες οι πιθανές άδειες καθώς και όλες οι πιθανές ομάδες. Πάνω σε αυτόν τον πίνακα γίνονται οι αναθέσεις των αδειών στις ομάδες. Έτσι ένας χρήστης αποκτά πρόσβαση μόνο μέσω των ομάδων στις οποίες ανήκει. Επειδή η εσωτερική βάση δεδομένων του συστήματος αυτού είναι οντοκεντρική και υπάρχει μια δενδρική τοποθέτηση των αντικειμένων μέσα στην βάση αυτή, δίνεται η δυνατότητα οι άδειες ενός αντικειμένου να κληρονομηθούν από τους απογόνους τους. Με αυτόν τον τρόπο υπάρχει η δυνατότητα ένα ολόκληρο υπόδεντρο να μοιράζεται τις ίδιες άδειες πρόσβασης.

[1] Το γραφικό περιβάλλον που φαίνεται στην εικόνα είναι από το σύστημα διαχείρισης του Zope



Εικόνα 2-3 Έλεγχος Πρόσβασης στο αρχικό αντικείμενο του Zope

2.2.2 Υποχρεωτικός Έλεγχος Πρόσβασης

Ο αντίστροφος τρόπος για τον έλεγχο πρόσβασης είναι ο υποχρεωτικός έλεγχος πρόσβασης (Mandatory Access Control). Σε αυτό τον τρόπο, η πρόσβαση είναι τόσο βασισμένη σε κανόνες όσο και στην πληροφορία που συσχετίζει ένα αντικείμενο με τον χρήστη [BCKNRBCMOW94]. Στα περισσότερα μοντέλα που ανήκουν στον υποχρεωτικό έλεγχο πρόσβασης, ο ιδιοκτήτης κάποιου αντικείμενου δεν έχει κάποιες ειδικές άδειες πάνω στο αντικείμενο αυτό. Η πληροφορία που συσχετίζει ένα χρήστη και ένα αντικείμενο συνήθως λέγεται «επίπεδο ασφαλείας» ή «επίπεδο ευαισθησίας» και συνήθως υλοποιείται ως ετικέτα στους χρήστες και τα αντικείμενα.

Στον χρήστη δίνεται μια ετικέτα που αναπαριστά την άδεια ασφαλείας (security clearance) του χρήστη αυτού. Στο αντικείμενο δίνεται μια ετικέτα που αναπαριστά την ταξινόμηση ασφαλείας (security classification) του αντικειμένου αυτού. Οι ετικέτες των χρηστών και των αντικειμένων ανήκουν στο ίδιο σύνολο. Στο σύνολο αυτό υπάρχουν ετικέτες που μπορούν να συγκριθούν μεταξύ τους και άλλες που δεν μπορούν. Συνήθως αυτό που ισχύει είναι:

- Αν οι ετικέτες του χρήστη και του αντικειμένου δεν μπορούν να συγκριθούν, τότε απαγορεύεται η πρόσβαση σε αυτό το αντικείμενο.

- Αν οι ετικέτες του χρήστη και του αντικειμένου είναι συγκρίσιμες, τότε η πρόσβαση καθορίζεται από τους κανόνες που έχουν οριστεί για το μοντέλο αυτό.

Για παράδειγμα σε ένα στρατιωτικό οργανισμό μπορεί να έχουν οριστεί τα εξής επίπεδα ασφαλείας: Μη ευαίσθητο < Ευαίσθητο < Απόρρητο < Άκρως απόρρητο. Όπως φαίνεται, αν ένας χρήστης έχει άδεια ασφαλείας «Απόρρητη» (δηλαδή μπορεί να έχει πρόσβαση στα αντικείμενα που ανήκουν στο επίπεδο ασφαλείας Απόρρητο) τότε μπορεί να αποκτήσει πρόσβαση στα αντικείμενα που έχουν ετικέτες: «μη ευαίσθητο», «ευαίσθητο», «απόρρητο» αλλά όχι στα αντικείμενα που έχουν ετικέτα «άκρως απόρρητο». Αν κάποιο αντικείμενο έχει κάποια ετικέτα διαφορετική από τις: «μη ευαίσθητο», «ευαίσθητο», «απόρρητο», «άκρως απόρρητο», τότε ο χρήστης δεν μπορεί να έχει πρόσβαση στο αντικείμενο αυτό.

2.2.2.1 Το Μοντέλο Bell LaPadula

Το μοντέλο Bell LaPadula (BLM) [M85], που επίσης αποκαλείται το «πολύ – επίπεδο» μοντέλο, είχε προταθεί από τους Bell και LaPadula για να επιβάλει έλεγχο πρόσβασης σε αντικείμενα για στρατιωτικές και κυβερνητικές εφαρμογές. Σε αυτές τις εφαρμογές οι χρήστες και τα αντικείμενα είναι διαχωρισμένα σε διαφορετικά επίπεδα ασφαλείας. Ένας χρήστης μπορεί να αποκτήσει πρόσβαση μόνο σε ορισμένα αντικείμενα που βρίσκονται σε κάποια επίπεδα ασφαλείας που συσχετίζονται με το δικό του επίπεδο ασφαλείας. Για παράδειγμα, οι δυο επόμενες προτάσεις είναι κάποιες τυπικές προτάσεις πρόσβασης που μπορούν να διατυπωθούν μέσα από το μοντέλο αυτό:

- Μη ταξινομημένο προσωπικό δεν μπορεί να διαβάσει δεδομένα που βρίσκονται σε εμπιστευτικά επίπεδα.
- Άκρως απόρρητα δεδομένα δεν μπορούν να γραφτούν σε αρχεία που βρίσκονται σε μη εμπιστευτικά επίπεδα.

Ένα από τα σημαντικά στοιχεία του μοντέλου Bell LaPadula που το διαφοροποιεί από τα άλλα μοντέλα είναι ότι εστιάζει και στον έλεγχο της ροής της πληροφορίας. Αυτό το μοντέλο ασφαλείας αποτελείται από τα εξής στοιχεία:

- Ένα σύνολο από χρήστες (πρόσωπα), ένα σύνολο από αντικείμενα και έναν πίνακα ελέγχου πρόσβασης (access control matrix).

- Μερικά διατεταγμένα επίπεδα ασφαλείας. Κάθε χρήστης (πρόσωπο) έχει άδεια να διαχειρίζεται αντικείμενα κάποιου επιπέδου και κάθε αντικείμενο είναι ταξινομημένο έτσι ώστε να ανήκει σε κάποιο επίπεδο ασφαλείας. Ακόμη, ο κάθε χρήστης ορίζει ένα τρέχων επίπεδο εξουσιοδότησης το όποιο δεν ξεπερνά το επίπεδο εξουσιοδότησης που έχει οριστεί για τον χρήστη.

Το σύνολο από τα δικαιώματα πρόσβασης είναι το παρακάτω:

- **Read-Only**: Ο χρήστης μπορεί μόνο να διαβάσει το αντικείμενο.
- **Append**: Ο χρήστης μπορεί να προσαρτήσει πληροφορία στο αντικείμενο αλλά δεν μπορεί να το διαβάσει.
- **Execute**: Ο χρήστης μπορεί να εκτελέσει το αντικείμενο αλλά δεν μπορεί να το διαβάσει ή να το γράψει.
- **Read-Write**: Ο χρήστης έχει δικαίωμα να γράψει και να διαβάσει το αντικείμενο.

Χαρακτηριστικό έλεγχου: Το χαρακτηριστικό αυτό δίνεται στον χρήστη ο οποίος δημιουργεί ένα αντικείμενο. Λόγω αυτού, ο δημιουργός ενός αντικείμενου μπορεί να δώσει οποιοδήποτε (από τα παραπάνω) δικαίωμα πρόσβασης σε οποιονδήποτε χρήστη, εκτός βέβαια από το ίδιο χαρακτηριστικό ελέγχου. Ο χρήστης δημιουργός θα ονομάζεται και «χειριστής» του αντίστοιχου αντικειμένου.

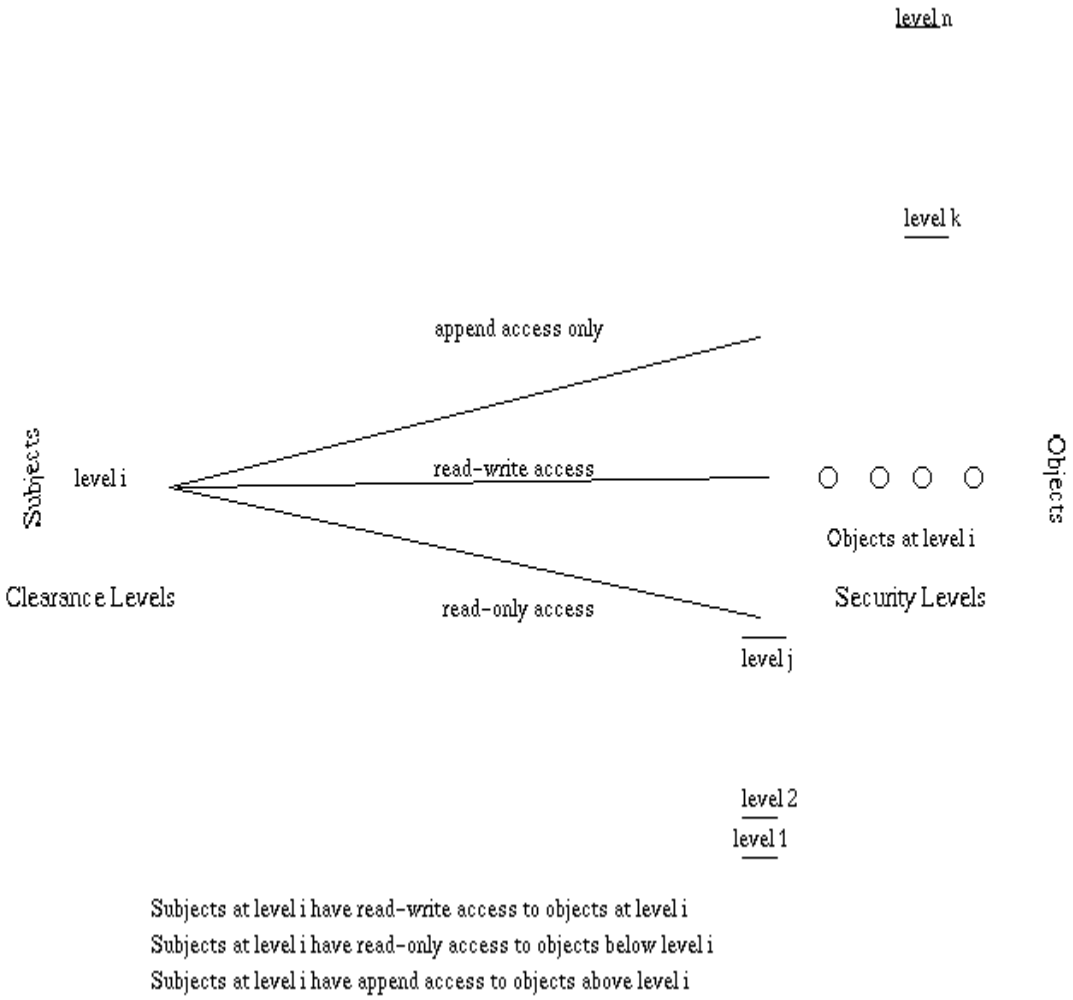
Οι παρακάτω είναι οι περιορισμοί που επιβάλλονται από το μοντέλο:

- **reading down**: Ένας χρήστης έχει άδεια για να διαβάσει τα αντικείμενα των οποίων το επίπεδο ασφαλείας είναι χαμηλότερο από το τρέχων επίπεδο αδείας του χρήστη. Αυτό εμποδίζει ένα χρήστη να αποκτήσει πρόσβαση σε πληροφορίες που βρίσκονται σε επίπεδα ασφαλείας υψηλότερα από το τρέχων επίπεδο αδείας του.
- **writing up**: Ένας χρήστης έχει την άδεια «Append» να γράφει και να μην διαβάζει σε αντικείμενα των οποίων το επίπεδο ασφαλείας είναι μεγαλύτερο από το τρέχων επίπεδο αδείας. Αυτό εμποδίζει ένα χρήστη να περάσει πληροφορία σε επίπεδα χαμηλότερα από το τωρινό επίπεδο.

Για παράδειγμα, έστω ότι υπάρχουν τα επόμενα επίπεδα ασφαλείας: Μη ευαίσθητο < Ευαίσθητο < Απόρρητο < Άκρως απόρρητο. Έστω ότι ένας χρήστης έχει επίπεδο άδειας «απόρρητο». Τότε μπορεί μόνο να προσαρτήσει πληροφορία

στα αντικείμενα που έχουν ετικέτα «άκρως απόρρητο», μπορεί να αναγνώσει και να γράψει τα αντικείμενα που βρίσκονται στο επίπεδο «απόρρητο», και τέλος μπορεί μόνο να διαβάσει τα αντικείμενα που έχουν ετικέτα: «ευαίσθητο» και «μη ευαίσθητο». Αυτός ο μηχανισμός φαίνεται στην Εικόνα 2-4.

Το μοντέλο Bell LaPadula συμπληρώνει το πίνακα πρόσβασης (access matrix) με τους παραπάνω περιορισμούς για να ελέγξει την πρόσβαση και την ροή δεδομένων. Για παράδειγμα, ένας χρήστης μπορεί να έχει την άδεια για ανάγνωση ενός αντικείμενου (read access) μέσω του πίνακα ελέγχου πρόσβασης, αλλά να μην μπορεί να ασκήσει αυτό το δικαίωμα αν το αντικείμενο είναι σε υψηλότερο επίπεδο ασφάλειας από το επίπεδο αδείας του χρήστη.



Εικόνα 2-4 Τρόπος Λειτουργίας Μοντέλου Bell Lapadula

Το Bell and LaPadula έχει μοντελοποιήσει την συμπεριφορά ενός συστήματος ασφαλείας ως μια μηχανή πεπερασμένων καταστάσεων (finite state

machine) και έχει ορίσει ένα σύνολο από μεταβάσεις όπου δεν παραβαίνουν την ασφάλεια του συστήματος. Οι παρακάτω πράξεις εγγυούνται ένα ασφαλές σύστημα:

- *get access*: Χρησιμοποιείται από ένα χρήστη για να ξεκινήσει την πρόσβαση σε ένα αντικείμενο. (read, append, execute...)
- *release access*: Χρησιμοποιείται από ένα χρήστη για να παραδώσει μια πρόσβαση που έχει αρχικοποιηθεί.
- *give access*: Ο χειριστής ενός αντικειμένου (δημιούργησε το αντικείμενο) μπορεί να δώσει μία συγκεκριμένη πρόσβαση (στο αντικείμενο αυτό) σε κάποιον χρήστη.
- *rescind access*: Ο χειριστής ενός αντικειμένου μπορεί να ανακαλέσει μια συγκεκριμένη πρόσβαση (στο αντικείμενο αυτό) από ένα χρήστη.
- *create object*: Αφήνει ένα χρήστη να δημιουργήσει ένα αντικείμενο.
- *delete object*: Αφήνει ένα χρήστη να σβήσει ένα αντικείμενο.
- *change security level*: Αφήνει ένα χρήστη να αλλάξει το τρέχων του επίπεδο άδειας (χαμηλότερο από το επίπεδο άδειας που του έχει ανατεθεί)

Όμως υπάρχουν κάποιες συνθήκες που πρέπει να ισχύουν για να μπορούν να εκτελεστούν οι παραπάνω πράξεις. Για παράδειγμα, ένας χρήστης πρέπει να εκχωρεί και ανακαλεί δικαιώματα σε ένα αντικείμενο αν και μόνο αν έχει τα χαρακτηριστικά ελέγχου του αντικειμένου αυτού.

Το Bell LaPadula είναι ένα απλό γραμμικό μοντέλο που ασκεί έλεγχο στην πρόσβαση και ροή δεδομένων μέσω των παραπάνω ιδιοτήτων και πράξεων. Βλέπουμε ακόμη ότι ενώ άρχισε ως ένα μοντέλο υποχρεωτικού ελέγχου πρόσβασης, η ύπαρξη ενός πίνακα πρόσβασης (access matrix) του προσδίδει κάποια στοιχεία από τα μοντέλα προαιρετικού ελέγχου πρόσβασης, τα οποία είναι αρκετά περιορισμένα έτσι ώστε να μην υπάρχουν τα προβλήματα του προαιρετικού μοντέλου πρόσβασης στο μοντέλο αυτό. Όμως ένα από τα κύρια μειονεκτήματα του είναι ότι τα επίπεδα ασφαλείας είναι στατικά. Οι ιδιότητες αυτού του μοντέλου μπορεί να γίνουν πολύ περιοριστικές σε περιπτώσεις που συγκεκριμένες λειτουργίες είναι εκτός του πλαισίου του συστήματος ασφαλείας.

2.2.2.2Έλεγχος Πρόσβασης Βασισμένος σε Ρόλους

Ως εξέλιξη των ομάδων χρηστών είναι ο έλεγχος πρόσβασης που βασίζεται σε ρόλους. Η πρώτη αναφορά έγινε το 1992 από τους Ferraiolo και Kuhn ως μια

λύση για τα πολύπλοκα προβλήματα ασφαλείας μεγάλων δικτύων και οργανισμών [FKC03]. Μέχρι σήμερα πάρα πολλές εταιρίες και οργανισμοί καθώς και καινούργιες εφαρμογές (π.χ. υπηρεσίες διαδικτυακών πυλών (portlets)) χρησιμοποιούν το μοντέλο ελέγχου πρόσβασης βασισμένο σε ρόλους (Role Based Access Control).

Η μεγάλη καινοτομία του μοντέλου ελέγχου πρόσβασης βασισμένο σε ρόλους σε σχέση με τις ομάδες χρηστών είναι ότι ένας χρήστης μπορεί να ανήκει σε πολλούς ρόλους (και ένας ρόλος μπορεί να έχει πολλούς χρήστες) ενώ ακόμη σε ένα ρόλο μπορεί να ανατεθούν αρκετές άδειες (permissions), κάτι που στις ομάδες χρηστών δεν ισχύει. Όποτε βλέπουμε ότι η πολιτική ελέγχου πρόσβασης είναι ενσωματωμένη με τις διάφορες συνιστώσες του μοντέλου αυτού όπως οι σχέσεις ρόλου – άδειας, χρήστη – ρόλου και ρόλου – ρόλου [SCFY96]. Σε αυτό το μοντέλο ένας ρόλος είναι υπεύθυνος για να εκτελέσει κάποια εργασία ή έχει τις απαραίτητες άδειες για να προσπελάσει κάποιο αντικείμενο και όχι οι χρήστες. Μάλιστα, η σχέση του χρήστη με τους ρόλους απεικονίζει την οργάνωση μιας εταιρίας ή ενός οργανισμού, αφού συνήθως και η οργάνωση και στελέχωση μιας εταιρίας ή ενός οργανισμού στηρίζεται συχνά στην ύπαρξη ρόλων. Με αυτόν τον τρόπο είναι αρκετά πιο εύκολη η διαχείριση ενός συστήματος μιας και ο διαχειριστής μπορεί εύκολα να αναθέσει ή να ανακαλέσει την συσχέτιση ενός χρήστη με κάποιους ρόλους. Βέβαια, κάτι τέτοιο δεν συμβαίνει και τόσο συχνά μέσα σε έναν οργανισμό (χρειάζεται να συμβεί μόνο όταν ένας εργαζόμενος αλλάζει τμήμα, ή παίρνει προαγωγή ή απολύεται). Ταυτόχρονα είναι το ίδιο εύκολη η προσθήκη ή η αφαίρεση αδειών από κάποιο ρόλο, πράγμα το οποίο και πάλι δεν είναι κάτι το σύνηθες να συμβεί (συμβαίνει μόνο όταν ένα καινούργιο έργο υπάρχει στον οργανισμό και χρειάζεται να ανατεθεί στους υπευθύνους και στους εργαζόμενους για την εκπόνησή του).

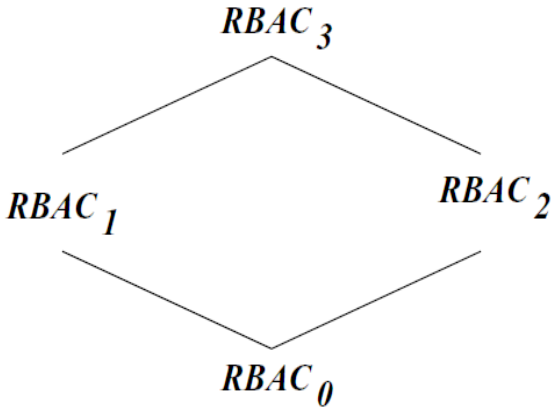
Ενώ το μοντέλο ελέγχου πρόσβασης βασισμένο σε ρόλους έχει ουδέτερη πολιτική (policy neutral), υποστηρίζει απευθείας τρεις αρκετά γνωστές αρχές ασφαλείας (security principles): των ελάχιστων προνομιών (least privilege), διαχωρισμό καθηκόντων (separation of duties) και αφαίρεση δεδομένων (data abstraction). Ελάχιστα προνόμια δίνονται στον χρήστη γιατί το μοντέλο αυτό είναι ρυθμισμένο να δίνει τα προνόμια που χρειάζονται στο ρόλο ώστε να εκτελέσει την εργασία που χρειάζεται. Διαχωρισμός καθηκόντων μπορεί να υπάρξει με το να χρησιμοποιηθούν αμοιβαίως αποκλειόμενοι ρόλοι σε μια ευαίσθητη εργασία. Αυτό

μπορεί να υλοποιηθεί αν οι αμοιβαίως αποκλειόμενοι ρόλοι δεν έχουν κανένα κοινό χρήστη και για την σωστή εκτέλεση της εργασίας χρειάζονται και οι δυο ρόλοι. Αφαίρεση δεδομένων υποστηρίζεται με την έννοια ότι σαν αφηρημένες άδειες μπορούν να δοθούν διαφορετικές άδειες που είναι πιο λειτουργικές για την εφαρμογή και όχι οι άδειες ανάγνωσης, εγγραφής και εκτέλεσης που προσφέρονται από τα λειτουργικά συστήματα. Όμως, αν και οι αρχές αυτές υποστηρίζονται από το μοντέλο, πρέπει να γίνει και η αντίστοιχη σχεδίαση. Σε περίπτωση που η σχεδίαση δεν είναι σωστή, ενδέχεται οι αρχές αυτές να παραβιάζονται στην τελική υλοποίηση.

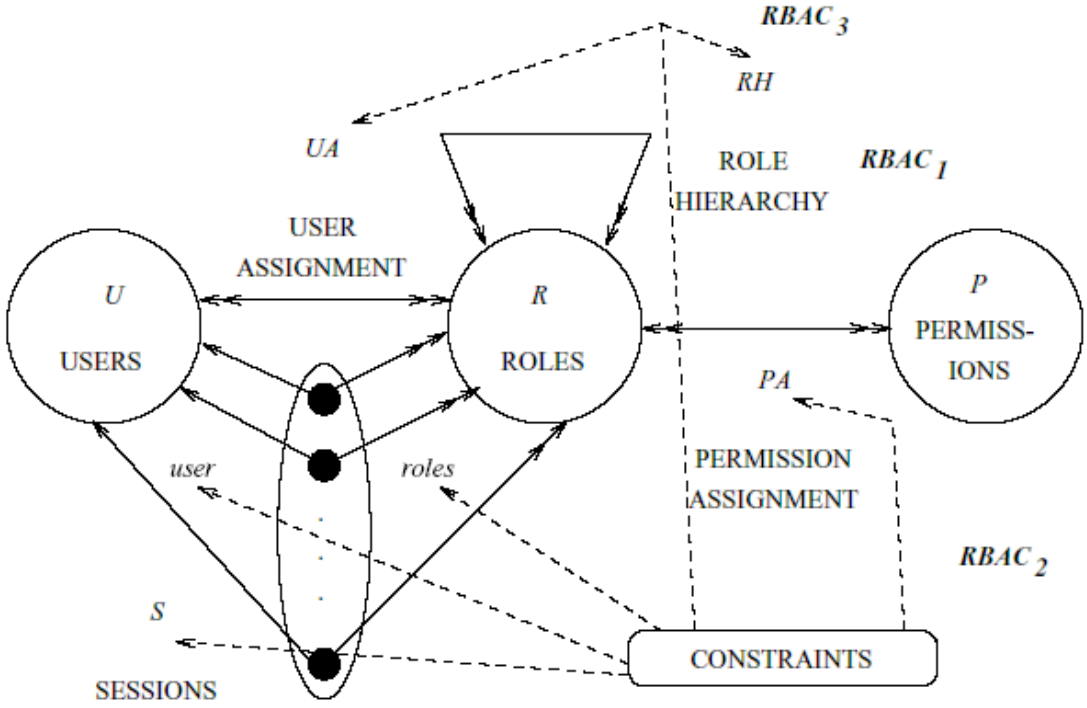
Για να καταλάβουμε τις πολλαπλές διαστάσεις του ελέγχου πρόσβασης βασισμένο σε ρόλους (RBAC) θα ορίσουμε τέσσερα θεμελιώδη μοντέλα. Η σχέση των μοντέλων φαίνεται στην Εικόνα 2-5 ενώ τα επιμέρους χαρακτηριστικά φαίνονται στην Εικόνα 2-6 . Το $RBAC_0$ είναι το βασικό μοντέλο πού πρέπει να υλοποιεί κάθε σύστημα που υποστηρίζει το μοντέλο RBAC. Τα $RBAC_1$ και $RBAC_2$ εμπεριέχουν και τα δυο το μοντέλο $RBAC_0$ αλλά προσθέτουν σ' αυτό επιπλέον στοιχεία. Το $RBAC_1$ θέτει την αρχή της ιεραρχίας των ρόλων, δηλαδή τη δυνατότητα των ρόλων να κληρονομούν άδειες από άλλους ρόλους. Το $RBAC_2$ εισάγει την έννοια των περιορισμών (constraints) στο πώς μπορεί να οριστεί κάποιο συγκεκριμένο RBAC μοντέλο. Για παράδειγμα, οι περιορισμοί μπορεί να βοηθήσουν στην υλοποίηση των αμοιβαίως αποκλειόμενων ρόλων. Έστω ότι οι δύο αμοιβαίως αποκλειόμενοι ρόλοι είναι οι «Χ» και «Υ», τότε ο περιορισμός μπορεί να γραφτεί ως εξής:

- cannot_belong(u_i, "X") ← belong(u_i, "Y")
- cannot_belong(u_i, "Y") ← belong(u_i, "X")

Στην δεξιά πλευρά του περιορισμού χρησιμοποιούμε λογικές συναρτήσεις που επιστρέφουν κατάλληλες τιμές αλήθειας. Στην περίπτωση μας είναι η συνάρτηση που ελέγχει αν ο χρήστης u_i ανήκει σε έναν ρόλο. Η αριστερή πλευρά του περιορισμού ορίζει ότι ο χρήστης αυτός δεν μπορεί να ανήκει σε κάποιο ρόλο. Τέλος το $RBAC_3$ μοντέλο είναι ο συνδυασμός των $RBAC_1$ και $RBAC_2$, και εμμέσως του $RBAC_0$ μοντέλου.



Εικόνα 2-5 Σχέση μεταξύ των μοντέλων RBAC



Εικόνα 2-6 Επιμέρους χαρακτηριστικά των μοντέλων RBAC

Ο ορισμός του $RBAC_0$ μοντέλου εμπεριέχει τα παρακάτω στοιχεία:

- U, R, P, and S (χρήστες, ρόλους, άδειες και συνεδρίες (sessions) αντίστοιχα),
- PA ⊆ P × R, πολλαπλή (many-to-many) σχέση ανάθεσης αδειών σε ρόλους,
- UA ⊆ U × R, πολλαπλή (many-to-many) σχέση ανάθεσης χρηστών σε ρόλους,

- user : S → U, μια συνάρτηση που αντιστοιχεί κάθε συνεδρία s_i στον συγκεκριμένο χρήστη *user(s_i)* (σταθερό για την διάρκεια της συνεδρίας), και
- roles : S → 2^R, μια συνάρτηση που αντιστοιχεί κάθε συνεδρία s_i σε ένα σύνολο από ρόλους *roles(s_i)* ⊆ {r | (*user(s_i)*, r) ∈ UA} (που μπορεί να αλλάζει με τον χρόνο) ενώ κάθε συνεδρία s_i έχει ένα σύνολο άδειων ∪ $_{r\in roles(si)}${p | (p,r) ∈ PA }

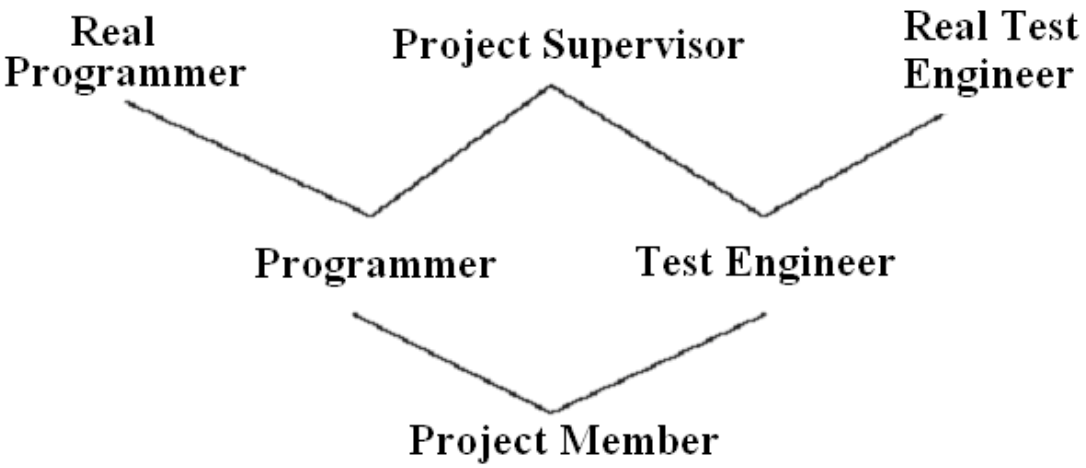
Στην απλή περίπτωση ένας χρήστης είναι ένας άνθρωπος, αλλά γενικότερα μπορεί να είναι ένα πρόγραμμα, ένας αυτόνομος πράκτορας, ένας υπολογιστής ακόμη και ένα δίκτυο από υπολογιστές. Η άδεια p_i είναι η αναπαράσταση μίας άδειας πρόσβασης σε κάποιο συγκεκριμένο αντικείμενο, που θα την αναγνωρίζει το σύστημα που χρησιμοποιεί το μοντέλο αυτό. Επίσης, μια συνεδρία s_i είναι η αντιστοίχηση του χρήστη με κάποιο σύνολο από ρόλους. Μια συνεδρία αρχικοποιείται όταν ο χρήστης ζητήσει ένα αντικείμενο και ορίσει και ένα υποσύνολο των ρόλων του. Τέλος, πολλές δημοσιεύσεις [FCK95, BCFG97] προτείνουν ένα επιπλέον χαρακτηριστικό ενός ρόλου: Το καθήκον (duty) του, δηλαδή το τι είναι υποχρεωμένος ο ρόλος να κάνει ώστε να μπορεί να λειτουργήσει φυσιολογικά κάποια συγκεκριμένη εργασία.

Στο $RBAC_1$ μοντέλο, εισάγεται η έννοια των ιεραρχιών. Οι ιεραρχίες ρόλων είναι μια φυσική μέθοδος δόμησης των ρόλων ώστε να αντανακλούν την οργάνωση ενός οργανισμού σε θέματα ευθύνης και εξουσίας. Τυπικά οι ιεραρχίες είναι μερικώς διατεταγμένες (partial ordering). Ο ορισμός για το $RBAC_1$ μοντέλο δίνεται ακολούθως:

- Τα σύνολα U, R, P, S, PA, UA και η συνάρτηση *user()* είναι ορισμένα όπως ακριβώς και στο $RBAC_0$ μοντέλο.
- RH ⊆ R × R, είναι η μερική διάταξη πάνω στο R και ονομάζεται ιεραρχία ρόλων ή σχέση κυριαρχίας ρόλων (role dominance relation), που επίσης μπορεί να αποδοθεί και ως ≥, και
- roles : S → 2^R, έχει τροποποιηθεί από το $RBAC_0$ μοντέλο ώστε *roles(s_i)* ⊆ {r | (∃ r' ≥ r)[(*user(s_i)*, r') ∈ UA]} (που μπορεί να

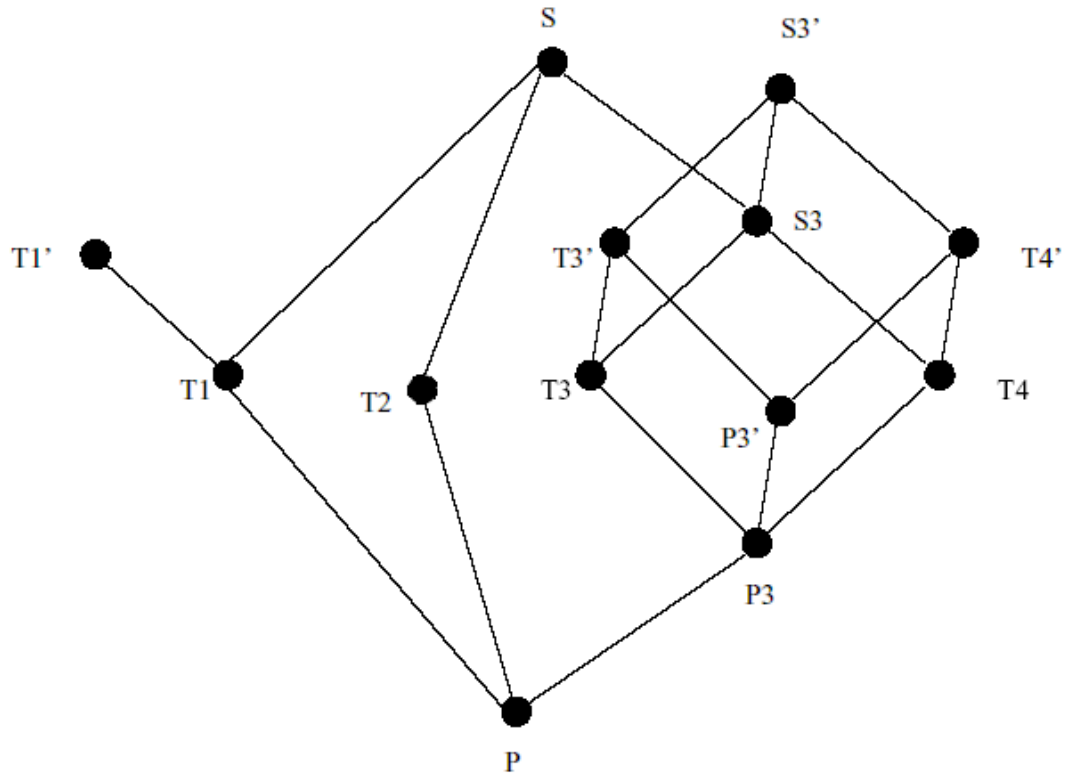
αλλάζει με τον χρόνο) και κάθε συνεδρία s_i έχει άδειες $\cup_{r \in roles(si)}\{p \mid (\exists\ r'' \leq r)\ [(p,r'') \in PA]\}$.

Μερικά παραδείγματα του $RBAC_1$ μοντέλου φαίνονται στις παρακάτω εικόνες. Ένα παράδειγμα ιεραρχίας των ρόλων RH φαίνεται στην Εικόνα 2-7. Ο ρόλος "Project Supervisor" κληρονομεί τις άδειες που έχουν οι ρόλοι "Test Engineer" και "Programmer", ενώ ο ρόλος "Real Test Engineer" κληρονομεί από τον "Test Engineer" μόνο και ο "Real Programmer" κληρονομεί από τον "Programmer" μόνο. Όλοι οι ρόλοι κληρονομούν τις άδειες του ρόλου "Project Member". Με αυτήν την ιεραρχία υλοποιούνται οι προσωπικές άδειες των ρόλων "Real Test Engineer" και "Real Programmer". Ειδικότερα, έστω ότι ο ρόλος "Programmer" έχει την άδεια να γράφει τον πηγαίο κώδικα ενός προγράμματος και ο ρόλος "Test Engineer" έχει την άδεια να τρέχει για να ελέγξει την ορθότητα αυτού του προγράμματος. Αν ένας χρήστης (e.g. "hargikas") ανήκει στον ρόλο "Project Supervisor", τότε μπορεί κατά την πρόσβαση του στο σύστημα να αποκτήσει μια συνεδρία που να τον συσχετίζει με έναν ή περισσότερους ρόλους από το σύνολο ["Project Supervisor", "Programmer", "Test Engineer", "Project Member"], αφού όλοι αυτοί οι ρόλοι κληρονομούνται από τον ρόλο "Project Supervisor". Με την κατάλληλη συνεδρία, ο χρήστης μπορεί να έχει άδεια εκτέλεσης ή άδεια εγγραφής ή και τα δύο πάνω στο εν λόγω πρόγραμμα. Έστω, όμως, ότι θέλουμε ο πραγματικός προγραμματιστής ("Programmer") ή ελεγκτής ("Test Engineer") του προγράμματος να έχει κάποιες άδειες που να μην κληρονομούνται από τον "Project Supervisor". Μπορούμε, τότε, να αναθέσουμε στους ρόλους "Programmer" και "Test Engineer" τις άδειες που θέλουμε να κληρονομήσει ο "Project Supervisor", και στους ρόλους "Real Programmer" και "Real Test Engineer" να αναθέσουμε τις επιπλέον άδειες.



Εικόνα 2-7 Παράδειγμα Ιεραρχίας Ρόλων

Στην Εικόνα 2-8 φαίνεται πως μπορεί να υπάρχει μια ολόκληρη ιεραρχία ρόλων που να έχει ιδιωτικές άδειες, οι οποίες κληρονομούνται μόνο στην τοπική ιεραρχία. Στη γενική ιεραρχία υπάρχουν τέσσερις ρόλοι T1, T2, T3 και T4 οι οποίοι όλοι κληρονομούν άδειες από τον κοινό ρόλο P. Ο ρόλος S κληρονομεί όλες τις άδειες που έχουν ανατεθεί στους ρόλους P, T1, T2, T3, T4, P3 και S3. Οι ρόλοι T3 και T4 ανήκουν σε μια άλλη υποκατηγορία που κληρονομεί από τον τοπικό κοινό του ρόλο P3, ενώ ο S3 κληρονομεί άδειες από τους ρόλους T3, T4, P3, P. Οι ρόλοι S3', T3', T4' και P3' μας δείχνουν πως μπορεί να υπάρχουν ιδιωτικές άδειες οι οποίες να κληρονομούνται σε ένα μικρό μέρος της ιεραρχίας. Οι τέσσερις τελευταίοι ρόλοι μπορούν να ορίσουν τις δικές τους άδειες, χωρίς να κληρονομηθούν από άλλους ρόλους.



Εικόνα 2-8 Παράδειγμα Ρόλων

Το $RBAC_2$ μοντέλο είναι ίδιο με το $RBAC_0$ μοντέλο, προσφέροντας επιπλέον τη δυνατότητα να καθορίζουμε ένα σύνολο από περιορισμούς που αφορούν στο ποιες από τις συσχετίσεις του $RBAC_0$ μοντέλου επιτρέπονται ή όχι. Με το μοντέλο αυτό μπορεί να οριστούν επιπλέον περιορισμοί, όπως αμοιβαίως αποκλειόμενοι ρόλοι και άλλα. Το $RBAC_3$ μοντέλο συνδυάζει τις ιεραρχίες του $RBAC_1$ μοντέλου μαζί με τους περιορισμούς του $RBAC_2$ μοντέλου. Θα μπορούσε να οριστεί η ιεραρχία ρόλων σαν περιορισμοί στο μοντέλο $RBAC_2$, αλλά το $RBAC_1$ μοντέλο θεωρείται αρκετά σημαντικό μοντέλο και από μόνο του.

Από ότι φάνηκε παραπάνω, το RBAC μοντέλο μπορεί να αναπαραστήσει εύκολα την δομή ενός οργανισμού καθώς και να ορίσει αφηρημένες άδειες σε συγκεκριμένους ρόλους. Ταυτόχρονα ο διαχειριστής του συστήματος έχει να διαχειριστεί μόνο τους ρόλους και τις άδειες, παρά όλους τους χρήστες ξεχωριστά. Φυσικά το μοντέλο ελέγχου πρόσβασης δεν είναι πανάκεια για όλα τα θέματα που αφορούν στον έλεγχο πρόσβασης. Χρειάζονται πιο εξεζητημένα μοντέλα ελέγχου πρόσβασης σε περίπτωση που υπάρχει σειρά από διεργασίες που πρέπει να ελεγχθούν και να εκτελεστούν.

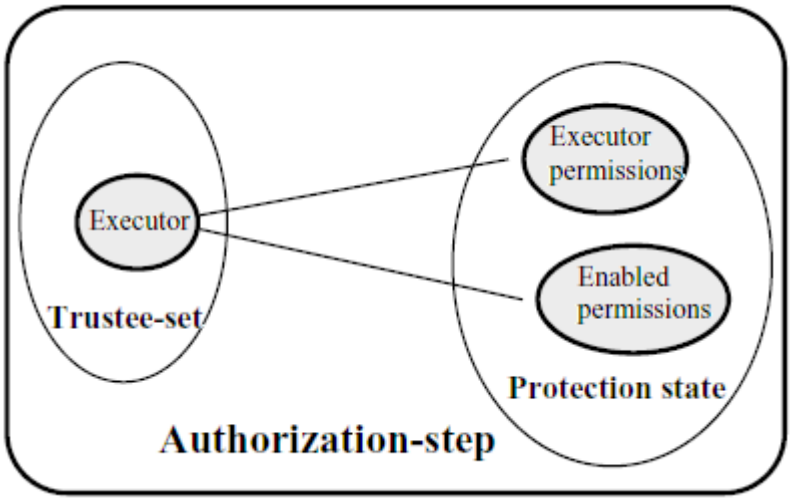
2.3 Εξουσιοδότηση Διεργασιών

Σε αυτήν την ενότητα θα παρουσιαστούν ορισμένα μοντέλα ελέγχου, πρόσβασης και ασφάλειας που ορίζονται για διεργασίες και ροές εργασιών. Η εκτέλεση των διεργασιών ή των ροών εργασίας μπορεί να είναι από μια εκτέλεση σε ένα μηχάνημα μέχρι την παράλληλη εκτέλεση, χρησιμοποιώντας πράκτορες, σε πολλαπλά μηχανήματα.

2.3.1 Ρύθμιση Εξουσιοδότησης Βασισμένη σε Διεργασίες

Μια εναλλακτική οπτική στο θέμα της εξουσιοδότησης από την κλασσική οπτική χρήστη – αντικειμένου, είναι ο έλεγχος πρόσβασης βασιζόμενος σε διαδικασίες. Αυτή η οπτική επιτρέπει τον ορισμό των ενεργών μοντέλων ασφαλείας που απαιτούνται από κατανεμημένα συστήματα βασισμένα σε πράκτορες (agents) ή σε ροές εργασίας (workflows) [TS97].

Με την κλασσική έννοια της εξουσιοδότησης (χρήστης – αντικείμενο) δεν γίνεται σωστή διαχείριση των αδειών αναφορικά με την διάρκεια εκτέλεσης μιας διεργασίας. Υπάρχει πιθανότητα οι άδειες να δοθούν αρκετά νωρίτερα ή αργότερα από την πραγματική στιγμή που απαιτείται, ή να είναι ακόμη ενεργές για πολύ ώρα μετά από την στιγμή που μια διεργασία ή ροή εργασιών έχει εκτελεστεί. Αυτό μάλιστα μπορεί να κάνει το σύστημα ευάλωτο σε επιθέσεις από κακοπροαίρετους χρήστες που επιθυμούν να αποκτήσουν παράνομα δικαιώματα πρόσβασης. Στα κλασσικά μοντέλα, ο διαχειριστής του συστήματος πρέπει να ελέγχει όλες τις άδειες που δίνονται σε μια συγκεκριμένη διεργασία και ταυτόχρονα να ελέγχει και τους χρονικούς περιορισμούς. Αυτός ο έλεγχος δεν είναι εφικτός αν ο διαχειριστής είναι άνθρωπος. Σε αυτές τις περιπτώσεις, ο διαχειριστής πρέπει να βρει τρόπους ελέγχου της κατάστασης εκτέλεσης κάθε διεργασίας και των αδειών που απαιτούνται.



Εικόνα 2-9 Βήμα Εξουσιοδότησης

Το πιο σημαντικό στοιχείο στο μοντέλο εξουσιοδότησης μέσω διεργασιών, είναι το βήμα εξουσιοδότησης (Εικόνα 2-9). Στο πραγματικό κόσμο, το βήμα αυτό μπορεί να συσχετιστεί με την ενέργεια απόκτησης μιας υπογραφής σε κάποιο αντικείμενο. Όταν αυτός που εκτελεί μια εργασία χρειάζεται κάποια σχετική άδεια εκτέλεσης, τότε ζητά από έναν έμπιστο (Trustee) να του υπογράψει την συγκεκριμένη άδεια για να μπορεί να την χρησιμοποιήσει. Ο κόμβος που θα αναλάβει να φέρει εις πέρας την αίτηση αυτή ονομάζεται έμπιστος κόμβος εκτέλεσης (Executor-Trustee) και οι άδειες που χρειάζεται για να φέρει εις πέρας μια τέτοια ενέργεια λέγονται άδειες έμπιστου κόμβου εκτέλεσης (Executor permissions). Τέλος, οι υπογεγραμμένες άδειες που παρέχει ο κόμβος αυτός ονομάζονται ενεργές άδειες (Enabled Permissions). Η ένωση των συνόλων “Executor Permissions” και “Enabled Permissions” ονομάζεται «κατάσταση ασφαλείας» “Protection-state”. Οι άδειες που έχουν δοθεί με τον παραπάνω τρόπο ισχύουν μόνο για ένα περιορισμένο χρονικό διάστημα.

Έλεγχος Πρόσβασης (χρήστη – αντικειμένου):

$$P \subseteq S \times O \times A$$

Έλεγχος Πρόσβασης (βασισμένη σε διεργασίες):

$$P \subseteq S \times O \times A \times U \times AS$$

Όπως φαίνεται και στην παραπάνω σχέση το μοντέλο αυτό διαφέρει από τα κλασικά μοντέλα χρήστη – αντικειμένου, γιατί οι άδειες εξαρτώνται και από άλλα

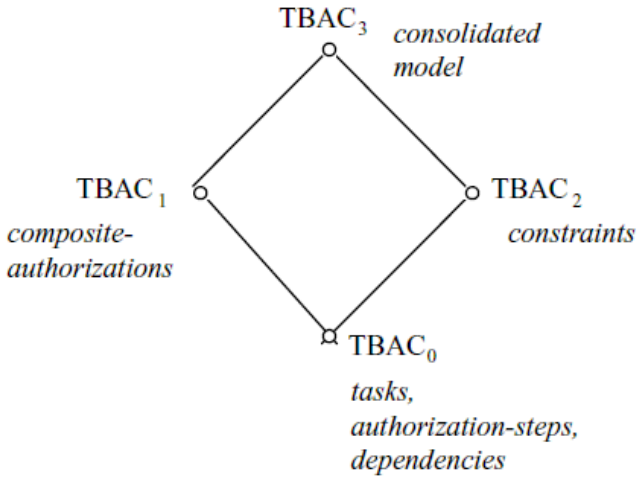
πεδία εκτός από τον χρήστη (S), το αντικείμενο (O) και το σύνολο από πράξεις (A) (actions). Επιπλέον, οι άδειες εξαρτώνται από δύο πεδία: U που είναι μετρητές χρήσης (usage) και εγκυρότητας (validity), καθώς και από τα βήματα εξουσιοδότησης (AS) (authorization step). Σύμφωνα με το μοντέλο αυτό, η άδεια που δίνεται μέσω ενός βήματος εξουσιοδότησης μπορεί να χρησιμοποιηθεί μόνο για περιορισμένο χρονικό διάστημα και για μερικές μόνο χρήσεις. Στη μαθηματική σχέση του ελέγχου πρόσβασης, ο περιορισμός αυτός περιγράφεται από το πεδίο U.

Σε μια σειρά από βήματα εξουσιοδοτήσεων που χρειάζεται για να ολοκληρωθεί μια απλή ή σύνθετη διεργασία, δεν έχουν κάποια ιδιαίτερη σχέση μεταξύ τους. Η κατάσταση ασφαλείας του κάθε βήματος εξουσιοδότησης είναι ξεχωριστή και πιθανώς διαφορετική από τα υπόλοιπα.

Για παράδειγμα, ας πάρουμε την περίπτωση της σύνθετης διαδικασίας έκδοσης μια επιταγής. Η διαδικασία αυτή αποτελείται από τρεις ποιο απλές: (α) την προετοιμασία της επιταγής, (β) την έγκριση της επιταγής και (γ) την έκδοση της επιταγής. Οι δύο πρώτες απλές διαδικασίες γίνονται από διαφορετικά άτομα (χρήστες). Ο χρήστης Α προετοιμάζει την επιταγή και μετά αναλαμβάνει ένας άλλος χρήστης, ο Β να την εγκρίνει. Στο τέλος μπορεί να εκδοθεί από τον χρήστη Α (ή κάποιον άλλο Ψ). Το σημαντικό στοιχείο είναι ότι κατά την διάρκεια της εκτέλεσης της διεργασίας, ο χρήστης Α προετοιμάζει την επιταγή και ζητάει από έναν έμπιστο κόμβο (την τράπεζα) άδεια για να γράψει πάνω στην επιταγή. Όταν εγκριθεί αυτή η αίτηση ο χρήστης μπορεί να την προετοιμάσει. Μετά το πέρας όμως της διεργασίας αυτής η τράπεζα ανακαλεί την άδεια για εγγραφή στην επιταγή και την δίνει στον χρήστη Β (μετά από την αίτηση του) για να την εγκρίνει. Όπως είναι φυσικό, μετά από το τέλος της δεύτερης διεργασίας, η άδεια εγγραφής ανακαλείται από τον χρήστη Β και δίδεται στο χρήστη που θα κάνει την έκδοση (που μπορεί να είναι και πάλι ο Α). Σε αυτό λοιπόν το μοντέλο εξουσιοδότησης αδειών βλέπουμε ότι οι άδειες δίνονται πάντα την κατάλληλη στιγμή (όταν γίνει η αίτηση) και δεν ισχύουν περισσότερο από ότι χρειάζεται μια διεργασία.

Έχουν προταθεί διάφορα μοντέλα εξουσιοδότησης βασισμένα σε διεργασίες τον οποίων η σχέση φαίνεται στην Εικόνα 2-10. Το βασικό μοντέλο $TBAC_0$ είναι αυτό που ορίζει τα βήματα εξουσιοδότησης, τις διεργασίες που τα χρειάζονται και το πως όλα αυτά συνδυάζονται: Πότε δίνονται οι άδειες, με τι χρονικά περιθώρια και τι γίνεται όταν μια διεργασία σταματήσει, σκοτωθεί κλπ, σε σχέση με τις άδειες που τις έχουν δοθεί. Το $TBAC_1$ μοντέλο ορίζει την αφαίρεση (abstraction) των

βημάτων εξουσιοδότησης για μεγάλες διεργασίες. Όταν οι $TBAC_0$ εξουσιοδοτήσεις είναι πολύ λεπτομερείς και δυσκολεύουν την διαχείριση των πληροφοριών, προτιμάται το $TBAC_1$ μοντέλο. Στο $TBAC_2$ μοντέλο ορίζονται διάφοροι περιορισμοί που μειώνουν τους πιθανούς συνδυασμούς κατά την διάρκεια ενός βήματος εξουσιοδότησης. Οι περιορισμοί αυτοί μπορεί να είναι στατικοί ή δυναμικοί ανάλογα με το αν επηρεάζουν την αρχικοποίηση ή την εκτέλεση ενός βήματος εξουσιοδότησης αντίστοιχα. Τα $TBAC_1$ και $TBAC_2$ μοντέλα κληρονομούν την συμπεριφορά το $TBAC_0$ μοντέλου. Το $TBAC_3$ μοντέλο είναι ο συνδυασμός των $TBAC_1$ και $TBAC_2$ μοντέλων.



Εικόνα 2-10 Σχέση μοντέλων βασισμένα σε διεργασίες

2.3.2 Ασφαλείς Ροές Εργασίας

Με τον όρο ασφαλής ροή εργασίας (Secure Workflow) υποδηλώνουμε ότι πρώτον η εκτέλεση μιας ροή εργασίας γίνεται με τέτοιο τρόπο ώστε μόνο τα κατάλληλα άτομα να μπορούν να εκτελέσουν συγκεκριμένες διεργασίες και δεύτερον ότι δεν πρέπει να δοθούν λιγότερες ή παραπάνω άδειες από αυτές που χρειάζονται για να εκτελεστεί σωστά η διεργασία αυτή. Στην συνέχεια θα παρουσιαστεί μια πολύ βασική σχεδίαση για την λύση αυτού του προβλήματος, καθώς και μερικές αρχιτεκτονικές υλοποίησης.

Οι παρακάτω ορισμοί συνοψίζουν τα βασικά χαρακτηριστικά ενός μοντέλου εξουσιοδότησης για ροές εργασίας (Workflow Authorization Model) WAM [AH96].

- Έστω S = {s1, s2, …} το σύνολο των χρηστών,
- O = {o1, o2, ...} το σύνολο των αντικειμένων,

- $\Gamma = \{\gamma 1, \gamma 2, ...\}$ το πεπερασμένο σύνολο των τύπων αντικειμένων
- Η συνάρτηση $Y : O \rightarrow \Gamma$, η οποία αν $Y(o_i) = \gamma_j$, τότε το αντικείμενο o_i είναι του τύπου γ_j.
- Έστω $PR = \{pr1, pr2, \ldots\}$ ένα πεπερασμένο σύνολο από άδειες

Βασικό στοιχείο του WAM είναι η περίοδος χρόνου (time set), ώστε οι άδειες να δίνονται για περιορισμένο χρόνο και όχι αόριστα.

- Το σύνολο του χρόνου $T = \{\tau \in R^2 \mid \tau \geq 0\}$ και
- ένα χρονικό διάστημα $\{[\tau_\iota, \tau_u] \in R \times R \mid \tau_\iota \leq \tau_u\}$ που αναπαριστά όλα τα σύνολα κλειστών διαστημάτων.

Ορίζουμε μια διεργασία w_i ως: $(OP_i, \Gamma_{IN}, \Gamma_{OUT}, [\tau_{Ii}, \tau_{Ui}])$ όπου:

- OP_i είναι ένα σύνολο από πράξεις που εκτελούνται στην w_i
- $\Gamma_{INi} \subseteq \Gamma$ είναι το σύνολο των τύπων αντικειμένων που επιτρέπονται για είσοδο
- $\Gamma_{OUTi} \subseteq \Gamma$ είναι το σύνολο των τύπων αντικειμένων που είναι πιθανά για έξοδο
- και $[\tau_{Ii}, \tau_{Ui}]$ η χρονική περίοδος μέσα στην οποία η διεργασία w_i πρέπει να εκτελεστεί

Ένα στιγμιότυπο διεργασίας (task-instance) $w\text{-}inst_i$ ορίζεται ως $(OPER_i, IN_i, OUT_i, [\tau_{Si}, \tau_{Fi}])$ όπου:

- $OPER_i$ είναι ένα σύνολο από πράξεις που εκτελούνται κατά την διάρκεια εκτέλεση της w_i,
- IN_i ένα σύνολο από αντικείμενα εισόδου για την w_i τέτοια ώστε $IN_i = \{\chi \in O \mid Y(\chi) \in \Gamma_{INi}\}$
- OUT_i ένα σύνολο από αντικείμενα εξόδου για την w_i τέτοια ώστε $OUT_i = \{\chi \in O \mid Y(\chi) \in \Gamma_{OUTi}\}$
- $[\tau_{Si}, \tau_{Fi}]$ μια χρονική περίοδος κατά την διάρκεια της οποίας η w_i εκτελούταν.

[2] Το R παρουσιάζει το σύνολο των πραγματικών αριθμών.

Η εξουσιοδότηση ορίζεται ως μια πλειάδα A = (s, o, pr, [τ_b, τ_e]) όπου στον χρήστη s δίνεται πρόσβαση στο αντικείμενο o με άδεια pr την χρονική στιγμή τ_b και ανακαλείται την χρονική στιγμή τ_e.

Δοσμένης μιας διεργασίας w_i ορίζουμε μια φόρμα εξουσιοδότησης (authorization template) AT(w_i), ως μία πλειάδα AT(w_i) = (s_i, (γ_i, -), pr_i) όπου:

- $s_i \in S$,
- (γ_i, -) είναι μια τρύπα αντικειμένου (object hole), όπου μπορεί να συμπληρωθεί από ένα αντικείμενο o_i του τύπου γ_i, και
- pr_i είναι η άδεια που δίδεται στον χρήστη s_i για το αντικείμενο o_i όταν η (γ_i, -) συμπληρωθεί από o_i.

Στο τέλος χρειαζόμαστε κανόνες παραγωγής εξουσιοδοτήσεων (Authorization Derivation Rule). Δοσμένης μίας φόρμας εξουσιοδότησης AT(w_i) = (s_i, (γ_i, -), pr_i) και μίας διεργασίας w_i = (OP_i, Γ_{IN}, Γ_{OUT}, [τ_{Ii}, τ_{Ui}]), μια εξουσιοδότηση Ai = (s_i, o_i, pr_i, [τ_{bi}, τ_{ei}]) παράγεται ως εξής:

- Κανόνας εφαρμογής: Έστω ότι το αντικείμενο $o_i \in \Gamma_{INi}$ στέλνεται στην διεργασία w_i την χρονική στιγμή $\tau_{\alpha i}$ για να ξεκινήσει w_i. Έστω η w_i ξεκινά την χρονική στιγμή τ_{si}.
  - Εάν $\tau_{\alpha i} \leq \tau_{ui}$ τότε $s_i \leftarrow s(AT)$, $pr_i \leftarrow pr(AT)$, $\tau_{ei} \leftarrow \tau_{ui}$ και
  - (εάν $\tau_{\alpha i} \leq \tau_{Ii}$ τότε $\tau_{bi} \leftarrow \tau_{Ii}$ αλλιώς $\tau_{bi} \leftarrow \tau_{\alpha i}$)
- Κανόνας ανάκλησης: Έστω η διεργασία w_i τερματίζει την χρονική στιγμή τ_{fi} την οποία το αντικειμένο o_i φεύγει από την w_i.
  - Εάν $\tau_{fi} \leq \tau_{ui}$ τότε $\tau_{ei} \leftarrow \tau_{fi}$

Με τον αυτό τρόπο έχει οριστεί ένα πλήρες μοντέλο εξουσιοδότησης και μπορούμε με τις φόρμες εξουσιοδότησης και τον κανόνα παραγωγής εξουσιοδοτήσεων να βγάλουμε τις κατάλληλες εξουσιοδοτήσεις ώστε να εκτελεστούν σωστά οι εφαρμογές. Μάλιστα με το ορισμό των χρονικών περιόδων, οι άδειες δίνονται στον χρήστη μόνο για την χρονική στιγμή που πραγματικά χρειάζονται.

Αν θεωρήσουμε το παράδειγμα με την έκδοση της επιταγής που είχε αναφερθεί και στην εξουσιοδότηση διεργασιών, μπορούμε να ορίσουμε τις επόμενες τρεις εργασίες σύμφωνα με το μοντέλο WAM:

- w1 = ({read request, prepare check}, {request, check}, {check}, [10,50])
- w2 = ({approve check}, {check}, {check}, [20,60])
- w3 = ({issue check}, {check}, {check}, [40,80])

Οι χρόνοι που ορίζονται είναι σε μια υποθετική μονάδα και μας δείχνουν από πότε και μέχρι πότε θα πρέπει να εκτελείται η κάθε εργασία. Έστω ότι έχουμε τρεις χρήστες (John, Mary, Ken) και ορίζουμε εδώ τις φόρμες εξουσιοδότησης:

- AT1(w1) = (John, (request, -), read)
- AT2(w1) = (John, (check, -), prepare)
- AT(w2) = (Mary, (check, -), approve)
- AT(w3) = (Ken, (check, -), issue)

Έστω ότι έρχονται οι αιτήσεις για επιταγές ως εξής:

- Αίτηση rq1 την χρονική στιγμή 40
- Αίτηση rq2 την χρονική στιγμή 55

Την χρονική στιγμή 40 η αίτηση rq1 φτάνει στην εργασία w1. Από την φόρμα εξουσιοδότησης AT1(w1) παίρνουμε την εξουσιοδότηση (John, rq1, read, [40,50]). Μετά την εκτέλεση της εργασίας w1 δημιουργείται το αντικείμενο ck023, οπότε από την AT2(w1) παίρνουμε την εξουσιοδότηση (John, ck023, prepare, [40, 50]). Έστω ότι η επιταγή είναι προετοιμασμένη την χρονική στιγμή 47, οπότε ανακαλούνται όλες οι άδειες που δόθηκαν στον χρήστη John και μπορεί να ειπωθεί ότι οι ακόλουθες εξουσιοδοτήσεις πραγματοποιήθηκαν: (John, rq1, read, [40,47]) και (John, ck023, prepare, [40, 47]). Το στιγμιότυπο διεργασίας που δημιουργείται τώρα είναι w-inst1 = ({read rq1, prepare ck023}, {rq1}, {ck023}, [40,47]).

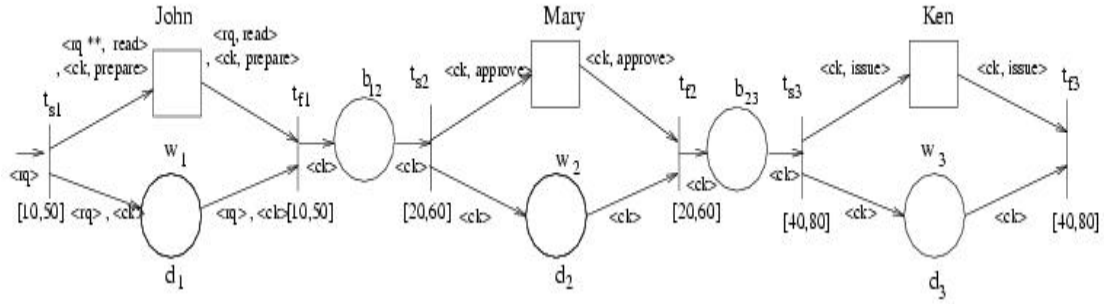
Έστω την χρονική στιγμή 47 η επιταγή ck023 στέλνεται στην διεργασία w2, οπότε από την φόρμα εξουσιοδότησης AT(w2) παράγεται η εξουσιοδότηση (Mary, ck023, approve, [47,60]). Έστω ότι η διεργασία τελειώνει την χρονική στιγμή 54. Τότε η εξουσιοδότηση αλλάζει σε (Mary, ck023, approve, [47,54]). Έπειτα το αντικείμενο ck023 στέλνεται στην διεργασία w3 και μέσω της φόρμας εξουσιοδότησης AT(w3) δημιουργείται η εξουσιοδότηση (Ken, ck023, issue, [54, 80]). Την χρονική στιγμή 60 τερματίζει και η διεργασία w3, οπότε η εξουσιοδότηση γίνεται (Ken, ck023, issue, [54, 60]).

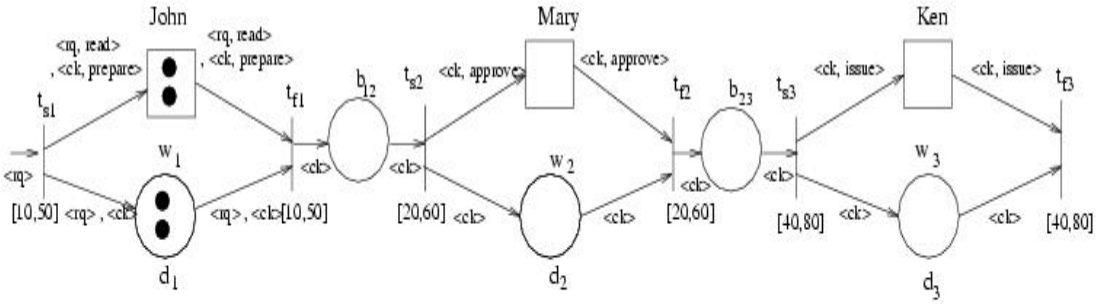
Όταν έρθει η επόμενη αίτηση rq2, την χρονική στιγμή 55, και επειδή η χρονική αυτή στιγμή ξεπερνάει τα χρονικά όρια εκτέλεσης της w1, δεν

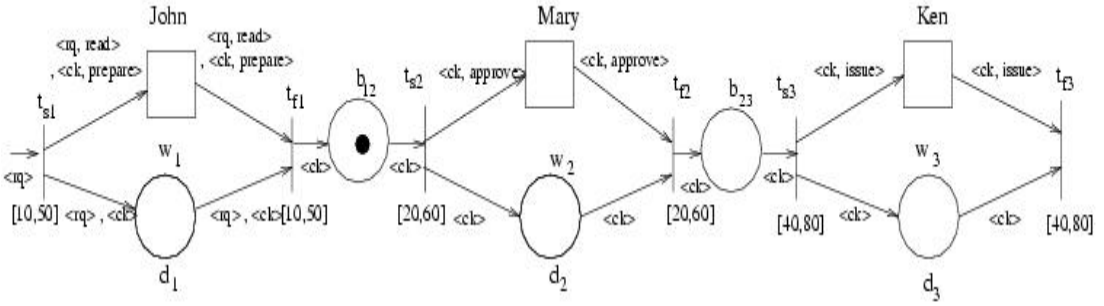
δημιουργείται καμία εξουσιοδότηση και έτσι δεν υπάρχει στιγμιότυπο διεργασίας και δεν εκτελούνται οι διεργασίες. Οι χρονικές στιγμές έχουν οριστεί ως απόλυτος χρόνος, αλλά χωρίς να αλλάξουν πολύ οι ορισμοί, μπορούν να μετατραπούν σε σχετικό χρόνο, αρκεί να υπάρχει κάποιο σημείο αναφοράς των χρονικών στιγμών.

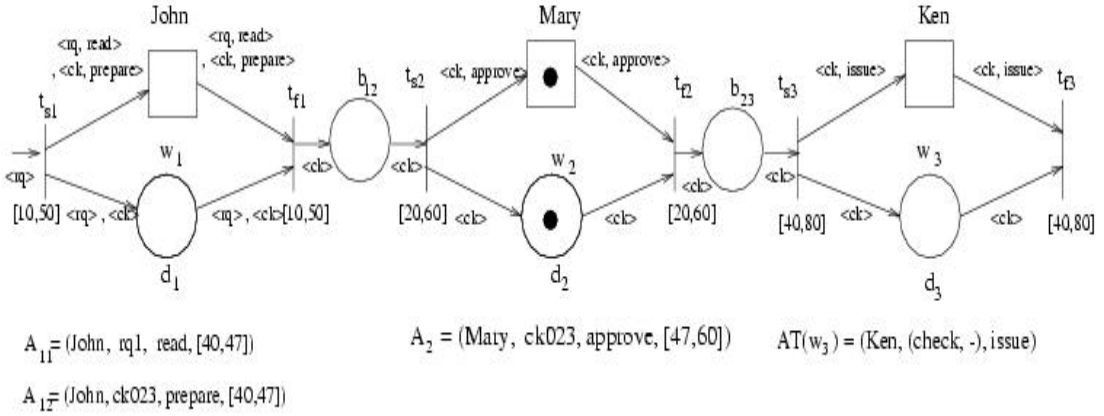
Είναι γνωστό ότι μπορεί να υλοποιηθεί μαθηματικά μία ροή εργασίας χρησιμοποιώντας ένα δίκτυο Petri (Petri-net). Ένα χρωματιστό δίκτυο Petri (colored Petri-net), δηλαδή όταν υπάρχουν πολλά κουπόνια (tokens) σε ένα κόμβο, μπορεί να προσομοιώσει μια εκτέλεση ροής εργασίας με εξουσιοδοτήσεις. Το προηγούμενο παράδειγμα φαίνεται στην Εικόνα 2-11, όπου γίνεται αντιληπτό πως πραγματοποιείται η εκτέλεση της ροής εργασίας μέσω χρωματιστών δικτύων Petri.

Ένα δίκτυο Petri αποτελείται από κόμβους – τοποθεσίες (places) και μεταβάσεις (transitions) που συνδέονται με έναν κατευθυνόμενο γράφο. Οι κόμβοι μπορεί να έχουν κάποιο συγκεκριμένο αριθμό από κουπόνια (tokens) διαφορετικού χρώματος. Οι ακμές του γράφου μπορούν να «μεταφέρουν» μόνο συγκεκριμένου χρώματος κουπόνια. Όταν μια μετάβαση είναι ενεργή σημαίνει ότι όλοι οι κόμβοι με τους οποίους συνδέεται ως είσοδος έχουν κουπόνια. Όταν μια μετάβαση πυροδοτηθεί μεταφέρει τα κουπόνια της εισόδου της στους κόμβους που συνδέεται ως έξοδος. Ο τρόπος συνδεσμολογίας του δικτύου Petri μπορεί να αναπαραστήσει μια ροή εργασίας. Σε μια τέτοια περίπτωση θεωρούμε τις μεταβάσεις του δικτύου Petri ως τις διεργασίες της ροής εργασίας, ενώ οι μεταβάσεις πυροδοτούνται όταν ολοκληρωθεί η συγκεκριμένη εργασία.









Εικόνα 2-11 Παράδειγμα εξουσιοδότησης για ροές εργασίας χρησιμοποιώντας δίκτυα Petri

Η παραπάνω εικόνα μας παρουσιάζει τέσσερις φάσεις ενός δικτύου Petri, το οποίο αναπαριστά την σειριακή εκτέλεση του παραδείγματος που αναφέρθηκε προηγουμένως. Κάθε κόμβος που σχεδιάστηκε ως τετράγωνο μπορεί να πάρει κουπόνια εξουσιοδοτήσεων και κάθε κόμβος που σχεδιάστηκε ως κύκλος παίρνει ως κουπόνια τις αναπαραστάσεις των αντικειμένων. Μια μετάβαση μπορεί να πυροδοτηθεί μόνο όταν όλες οι εξουσιοδοτήσεις και τα αντικείμενα μπορούν να χρησιμοποιηθούν και η διεργασία έχει τελειώσει. Έτσι, μπορούμε να δούμε γραφικά τις εξουσιοδοτήσεις που χρειάζονται να γίνουν για να εκτελεστεί μια ροή εργασίας.

Ένα από τα μειονεκτήματα αυτού του μοντέλου είναι ότι πρέπει να γνωρίζουμε πόσο πολύ θα διαρκέσει μια διεργασία. Αυτό μπορεί να είναι εφικτό όταν μιλάμε για διεργασίες που γίνονται από άτομα, ή αν η εκτέλεση της διεργασίας χρειάζεται σταθερό χρόνο. Αλλά υπάρχουν πολλές φορές διεργασίες που δεν γνωρίζουμε πόσο χρόνο παίρνουν.

2.3.3 Επεκτάσεις στο βασικό μοντέλο ασφαλών ροών εργασίας

Διάφορες επεκτάσεις έχουν προταθεί στο προηγούμενο μοντέλο για την υποστήριξη και άλλων μοντέλων εξουσιοδότησης μέσα στο WAM. Μια από τις πιο ενδιαφέρουσες επεκτάσεις είναι η προσθήκη ιεραρχικών ρόλων στην εκτέλεση ροών εργασίας μέσω του προηγούμενου μοντέλου WAM [BFA97, HK03]. Στην προσέγγιση αυτή, οι φόρμες εξουσιοδότησης παίρνουν ρόλους αντί για χρήστες στον ορισμό τους, όπως φάνηκε προηγουμένως. Στο προηγούμενο παράδειγμα οι φόρμες εξουσιοδότησης γίνονται:

- AT1(w1) = (clerk, (request, -), read)
- AT2(w1) = (clerk, (check, -), prepare)
- AT(w2) = (supervisor, (check, -), approve)
- AT(w3) = (clerk, (check, -), issue)

Αν όμως πρέπει οι διεργασίες w1 και w2 να εκτελεστούν από διαφορετικούς χρήστες, θα πρέπει οι ρόλοι “clerk” και “supervisor” να είναι αμοιβαίως αποκλειόμενοι ρόλοι. Στο παραπάνω παράδειγμα το διαχωρισμό των καθηκόντων (separation of duties) τον αναλαμβάνει το μοντέλο RBAC. Υπάρχουν, όμως, μερικές δημοσιεύσεις που υποστηρίζουν ότι ο διαχωρισμός των καθηκόντων πρέπει να γίνεται από περιορισμούς (constraints) που ορίζονται πάνω στο μοντέλο WAM. Για παράδειγμα ας θεωρήσουμε το ακόλουθο: Εάν ο χρήστης u_i ανήκει στον

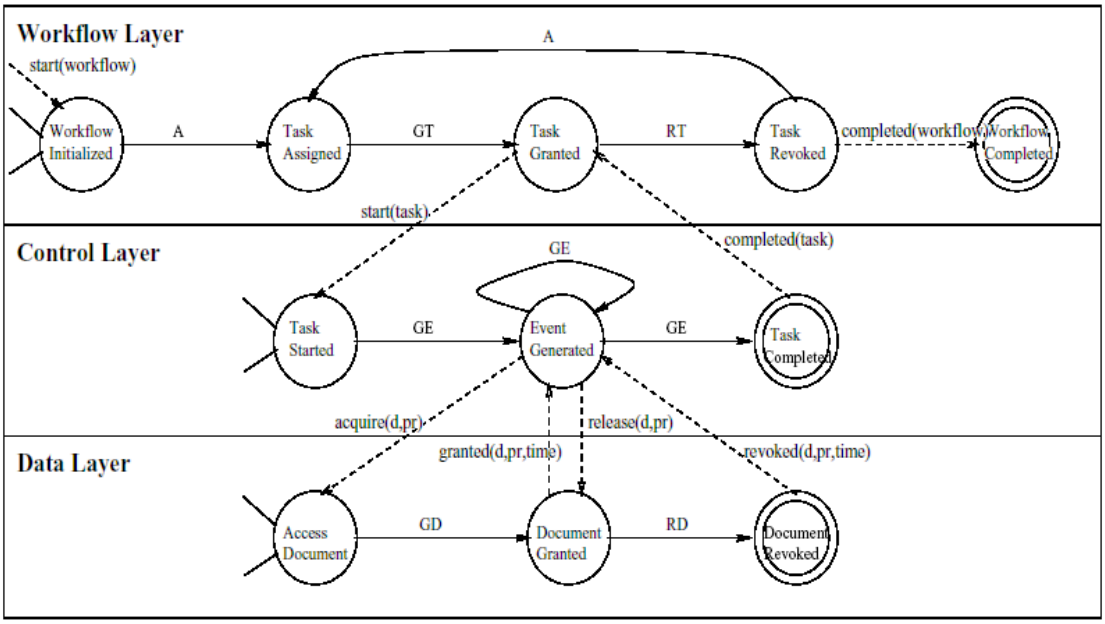
ρόλο “supervisor” και έχει εκτελέσει την διεργασία w1, τότε δεν μπορεί να εκτελέσει την διεργασία w2. Αυτό μπορεί να μεταφραστεί ως:

- $\text{cannot\_do}_u(u_i, w2) \leftarrow \text{belong}(u_i, \text{'supervisor'}), \text{execute}_u(u_i, w1, k)$

Στην δεξιά πλευρά των περιορισμών χρησιμοποιούμε λογικές συναρτήσεις που επιστρέφουν κατάλληλες τιμές αλήθειας. Για παράδειγμα, η συνάρτηση $\text{execute}_u(u_i, w1, k)$ είναι αληθής αν ο χρήστης u_i εκτέλεσε την διεργασία w1 όταν η w1 εκτελέστηκε για k φορά. Στην αριστερή πλευρά βλέπουμε τους περιορισμούς. Στην περίπτωση του παραδείγματος είναι η cannot_do_u $(u_i, w2)$ όπου απαγορεύει τον χρήστη να u_i εκτελέσει την w2 διεργασία.

2.3.4 Προτεινόμενες αρχιτεκτονικές για ασφαλείς ροές εργασίας

Έχουν προταθεί διάφορες αρχιτεκτονικές [GT02, HK03] για την βασική υλοποίηση του παραπάνω μοντέλου εξουσιοδότησης για ασφαλείς ροές εργασίας. Αυτό μας επιτρέπει να αποκτήσουμε σωστές άδειες πρόσβασης για κάποιον χρήστη που εκτελεί μια διεργασία μιας ροής εργασίας. Στην Εικόνα 2-12 βλέπουμε μια μηχανή πεπερασμένων καταστάσεων που μοντελοποιεί την συμπεριφορά του μοντέλου WAM. Η λειτουργία της αρχίζει με την αρχικοποίηση της ροής εργασίας και βρίσκεται στο επίπεδο «ροής εργασίας» (Workflow Layer). Κάθε φορά που μια διεργασία πρέπει να εκτελεστεί, ανατίθεται σε κάποιον χρήστη. Όταν ο χρήστης αναλαμβάνει να τρέξει τη διεργασία, τότε η μηχανή αλλάζει επίπεδο και πηγαίνει στο επίπεδο «ελέγχου» (Control Layer), όπου και παραμένει εκεί όσο εκτελείται η συγκεκριμένη διεργασία. Κάθε φορά που η διεργασία χρειάζεται κάποιο αντικείμενο για να αποκτήσει πρόσβαση, τότε δημιουργείται ένα γεγονός (event) και η μηχανή καταστάσεων αλλάζει επίπεδο και πηγαίνει στο επίπεδο «δεδομένων» (Data Layer), οπού και παραμένει όσο διαρκούν τα γεγονότα για κάποια αντικείμενα. Όπως φαίνεται και στο παρακάτω σχήμα βλέπουμε πως η διαδικασία αυτή μπορεί να επαναληφθεί για κάθε διεργασία ή αντικείμενο αντίστοιχα.



Εικόνα 2-12 Πολλαπλών επιπέδων μηχανή καταστάσεων

Ενώ με τον παραπάνω τρόπο μπορούμε να δούμε πως υλοποιείται η ανάθεση και η ανάκληση των αδειών για τα αντικείμενα κατά την διάρκεια εκτέλεσης μιας διεργασίας, υπάρχουν δημοσιεύσεις [GT02] που δείχνουν ότι ένας καλός τρόπος υλοποίησης των ρόλων μπορεί να γίνει με την χρήση αυτόνομων πρακτόρων (autonomous agents) που δρουν για λογαριασμό του χρήστη. Κάθε πράκτορας εκτελεί δύο λειτουργίες όπως διατυπώνεται στις παρακάτω περιπτώσεις: Η μία περίπτωση είναι όταν ο χρήστης ζητήσει να εκτελεστούν κάποιες διεργασίες, οπότε ο πράκτορας αναλαμβάνει την εκτέλεσή τους. Η άλλη περίπτωση είναι όταν ζητηθεί από το σύστημα να εκτελεστεί κάποια διεργασία από κάποιο συγκεκριμένο ρόλο, οπότε ο πράκτορας αναλαμβάνει να επικοινωνήσει (με διάφορους τρόπους) με τον χρήστη για την εκτέλεση της διεργασίας αυτής.

2.4 Κατανεμημένος Έλεγχος Πρόσβασης

Στην ενότητα αυτή θα παρουσιαστούν διάφορες μέθοδοι ελέγχου πρόσβασης σε κατανεμημένα συστήματα. Θα δούμε πως αυτός ο έλεγχος επιτυγχάνεται με την χρήση χαρακτηριστικών πιστοποίησης και πως μερικά σύστημα που υποστηρίζουν πλέγματα υπολογισμού λύνουν το πρόβλημα της πιστοποίησης και της εξουσιοδότησης. Τέλος, θα παρουσιαστεί το πως μπορούμε να ορίσουμε πολιτικές ελέγχου πρόσβασης για αντικείμενα χρησιμοποιώντας γλώσσες που βασίζονται στο πρότυπο XML.

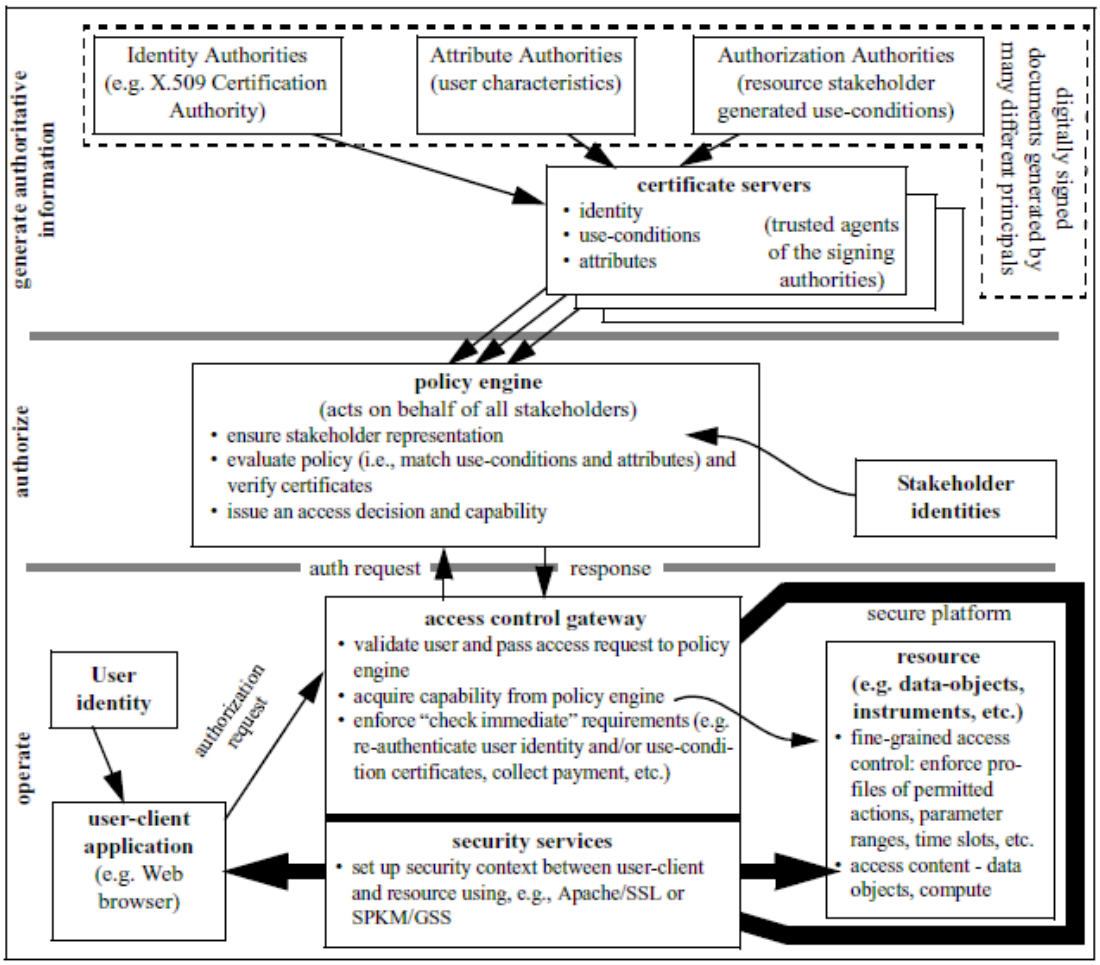
2.4.1 Κατανεμημένος Έλεγχος Πρόσβασης με Χαρακτηριστικά Πιστοποίησης

Σε αρκετές δημοσιεύσεις σχετικές με κατανεμημένους μηχανισμούς ελέγχου πρόσβασης έχει προταθεί η χρήση των χαρακτηριστικών πιστοποίησης (attribute certificates) [JMT98, TEM03]. Η βασική ιδέα αυτής της τεχνικής βασίζεται στο γεγονός ότι για να αποκτήσει ένας χρήστης πρόσβαση σε κάποιο αντικείμενο θα πρέπει να τηρεί ορισμένες προϋποθέσεις οι οποίες καθορίζονται από συγκεκριμένες οντότητες όπως φυσικά πρόσωπα ή οργανισμούς. Οι οντότητες αυτές ορίζουν κανόνες, και για να δοθεί η άδεια πρόσβασης σε ένα χρήστη, αυτός πρέπει να αποδείξει ότι τους τηρεί. Εάν υπάρχουν περισσότερες από μία οντότητες που θέτουν τέτοιους κανόνες, ο χρήστης θα πρέπει να ικανοποιεί όλους όσους έχουν οριστεί για να καταφέρει να αποκτήσει την άδεια πρόσβασης.

Για την υλοποίηση ενός τέτοιου συστήματος σημαντικό ρόλο παίζει η τεχνολογία των ψηφιακών πιστοποιητικών (digital certificates). Η τεχνολογία αυτή χρησιμοποιείται στην ανάπτυξη της εμπιστοσύνης (trust) και των ψηφιακών υπογραφών (digital signatures). Κάθε χρήστης έχει ένα ψηφιακό πιστοποιητικό που τον αντιπροσωπεύει. Οι οντότητες που θέτουν τους κανόνες πρέπει να εμπιστεύονται τον οργανισμό που έχει εκδώσει το πιστοποιητικό που αντιπροσωπεύει το χρήστη. Παρομοίως, η κάθε οντότητα που θέτει κανόνες που είναι απαραίτητοι για την πρόσβαση σε κάποιο αντικείμενο έχει ένα ψηφιακό πιστοποιητικό με το οποίο υπογράφει τους κανόνες. Τους κανόνες αυτούς θα τους ονομάσουμε όρους χρήσης (use-conditions). Όταν οι όροι χρήσης είναι υπογεγραμμένοι, τότε το σύστημα που τους ελέγχει γνωρίζει ότι ισχύουν και ότι δεν έχουν τροποποιηθεί. Τέλος υπάρχουν και οι οντότητες χαρακτηριστικών (attribute authorities) που πιστοποιούν αν πράγματι ένας χρήστης έχει κάποιες ιδιότητες ή χαρακτηριστικά. Η πιστοποίηση αυτή γίνεται με την δημιουργία ενός ψηφιακού πιστοποιητικού που είναι υπογεγραμμένο από τις οντότητες χαρακτηριστικών, καθώς και από τον χρήστη στον οποίο αυτά τα χαρακτηριστικά αναφέρονται.

Έχοντας όλα αυτά τα στοιχεία (identity certificates, attribute certificates, signed use-conditions) μπορεί να γίνει η πιστοποίηση ενός χρήστη και ο κατάλληλος έλεγχος πρόσβασης. Ο μηχανισμός που κάνει τον υπολογισμό και την επιβεβαίωση πάνω στους όρους χρήσης είναι η μηχανή πολιτικής ελέγχου πρόσβασης (policy engine). Η μηχανή πολιτικής ελέγχου πρόσβασης υπολογίζει

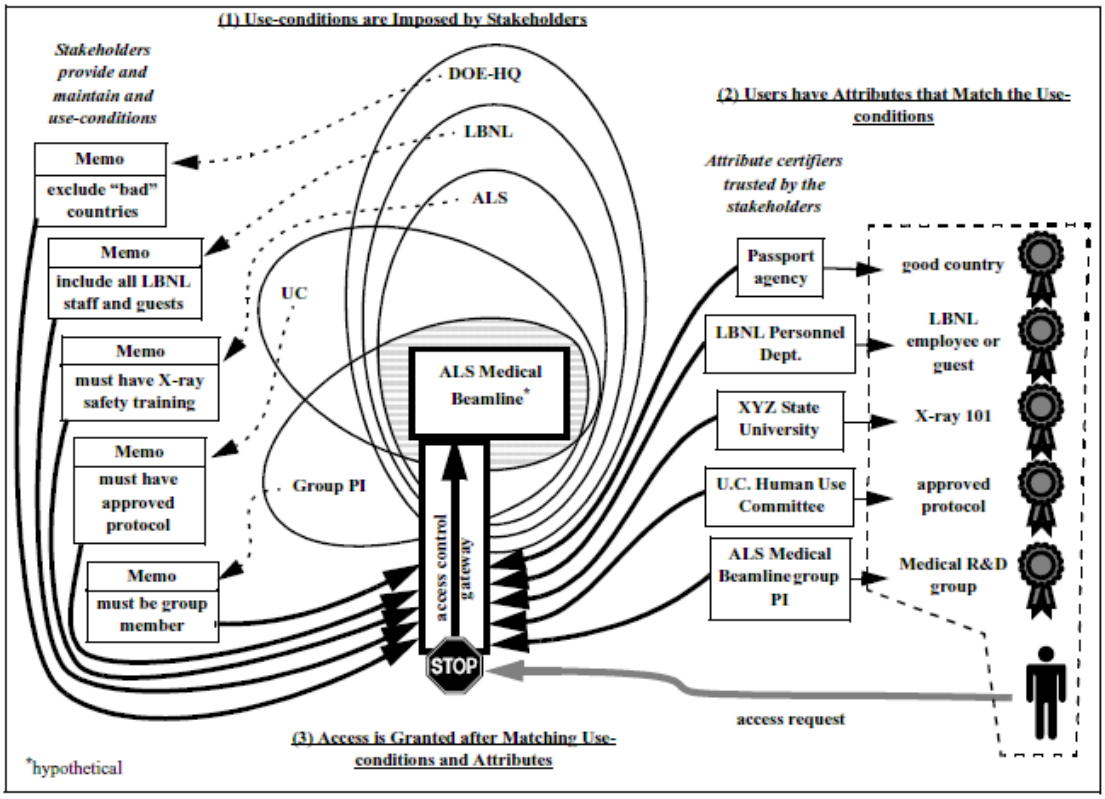
τους όρους χρήσης που πρέπει να χρησιμοποιηθούν για ένα αντικείμενο στο οποίο έχει γίνει μια αίτηση πρόσβασης και προσπαθεί να εφαρμόσει όλους τους όρους χρήσης ψάχνοντας εάν ο χρήστης έχει όλα τα κατάλληλα χαρακτηριστικά πιστοποίησης. Εάν έστω και ένας κανόνας αποτύχει τότε απαγορεύει την πρόσβαση στον χρήστη για το αντικείμενο αυτό.



Εικόνα 2-13 Αρχιτεκτονική Ελέγχου Πρόσβασης με Πιστοποιητικά Χαρακτηριστικών

Στο παραπάνω σχήμα παρουσιάζεται μια προτεινόμενη αρχιτεκτονική για την υλοποίηση του ελέγχου πρόσβασης μέσω χαρακτηριστικών πιστοποίησης. Σε αυτήν την αρχιτεκτονική ο έλεγχος πρόσβασης αρχίζει όταν ο χρήστης προσπαθεί να προσπελάσει ένα αντικείμενο. Ο χρήστης κάνει μια αίτηση πιστοποίησης από το σύστημα, η οποία παραλαμβάνεται από την πύλη ελέγχου πρόσβασης (access control gateway) στην οποία γίνεται ο έλεγχος των χαρακτηριστικών ασφαλείας του χρήστη (credentials). Αν ο έλεγχος είναι σωστός, τότε η πύλη έλεγχου πρόσβασης ζητάει να δοθεί πρόσβαση στο συγκεκριμένο αντικείμενο από την μηχανή πολιτικής ελέγχου πρόσβασης. Η μηχανή πολιτικής ελέγχει τους όρους χρήσης που ισχύουν

για το εν λόγω αντικείμενο και αν έχουν τεθεί από τις σωστές οντότητες μέσω των πιστοποιητικών τους. Όταν όλοι οι όροι χρήσης έχουν βρεθεί και έχει πιστοποιηθεί η γνησιότητά τους, το σύστημα προσπαθεί να δει αν όλοι οι όροι χρήσης επαληθεύονται για τον συγκεκριμένο χρήστη, αναζητώντας τα κατάλληλα πιστοποιητικά χαρακτηριστικών. Μόνο όταν όλοι οι όροι χρήσης ικανοποιούνται, η πύλη ελέγχου επιτρέπει την πρόσβαση στο αντικείμενο. Όλες οι αναζητήσεις και οι έλεγχοι για τα πιστοποιητικά (χαρακτηριστικών, όρους χρήσης, χρήστη κλπ) γίνονται μέσω των εξυπηρετών πιστοποιητικών (certificate servers). Εάν η μηχανή πολιτικής έχει επιστρέψει θετικό αποτέλεσμα για την πρόσβαση ενός χρήστη για ένα αντικείμενο, τότε η πύλη ελέγχου πρόσβασης αναλαμβάνει να δημιουργήσει ένα ασφαλές κανάλι επικοινωνίας με τον χρήστη ώστε να παραδώσει στην εφαρμογή του χρήστη το αντικείμενο που ζήτησε.



Εικόνα 2-14 Παράδειγμα χρήσης χαρακτηριστικα πιστοποίησης

Έστω ότι ένας χρήστης θέλει να αποκτήσει πρόσβαση σε ένα αντικείμενο που βρίσκεται κάτω από πολλές οντότητες ελέγχου όπως φαίνεται και στην Εικόνα 2-14. Έστω ότι για ένα αντικείμενο χρειάζονται να γίνουν οι παρακάτω έλεγχοι: από τα κεντρικά του υπουργείου ενεργείας θέλουν αυτοί που έχουν πρόσβαση να είναι μόνο πολίτες συμβεβλημένων χωρών, ένα εργαστήριο (LBNL) θέλει το προσωπικό και οι επισκέπτες του να έχουν πρόσβαση , η διεύθυνση του τομέα "Laser" θέλει να

έχουν πρόσβαση μόνο αυτοί που έχουν ειδική εκπαίδευση, ένα πανεπιστήμιο θέλει το πρωτόκολλο που θα ακολουθήσει ο χρήστης να είναι αποδεκτό και τέλος η εταιρία "Beamline" θέλει να είναι μέλος του ομίλου της. Όλα τα προηγούμενα είναι οι όροι χρήσης που έθεσαν κάποιες οντότητες ελέγχου. Ο χρήστης για να αποκτήσει πρόσβαση πρέπει να προσκομίσει όλα τα πιστοποιητικά πρόσβασης από τους παρακάτω φορείς για να αποδείξει ότι πληρεί όλους τους όρους χρήσης. Στην Εικόνα 2-14, στα αριστερά, φαίνονται οι οντότητες και οι όροι χρήσης που έχουν θέσει, ενώ στα δεξιά φαίνεται τα χαρακτηριστικά πιστοποίησης του χρήστη καθώς και από ποιες οντότητες είναι υπογεγραμμένα. Ο χρήστης πληρεί τους όρους χρήσης όποτε μπορεί να αποκτήσει πρόσβαση στο αντικείμενο.

Ο έλεγχος πρόσβασης με χαρακτηριστικά πιστοποίησης είναι ένα πολύ δυναμικό σύστημα, όπου δυναμικά αντικείμενα μπορούν να αποκτούν τους καταλλήλους όρους χρήσης και να τους τροποποιούν εύκολα. Τα δυο αυτά γεγονότα μπορεί να συμβαίνουν χωρίς να υπάρχει κεντρικός έλεγχος πάνω τους. Η ασφάλεια στηρίζεται στην εμπιστοσύνη σε αυτούς που ορίζουν τους όρους χρήσης και υλοποιείται με το μηχανισμό των ψηφιακών υπογραφών. Η προσέγγιση αυτή δεν μας δίνει πάντα τον κατάλληλο έλεγχο αλλά ούτε και προτάσεις στην περίπτωση που απορριφθεί μια αίτηση πρόσβασης για ένα αντικείμενο. Σε τελική ανάλυση, η εν λόγω προσέγγιση δεν προσφέρει τις απαραίτητες επεξηγήσεις σ' ένα χρήστη ώστε να καταφέρει να αποκτήσει πρόσβαση στο αντικείμενο που ζήτησε.

2.4.2 Ασφάλεια σε πλέγμα υπολογισμού

Ένα πλέγμα υπολογισμού (Computational Grid) αποτελείται από αρκετούς υπολογιστές συνδεδεμένους μεταξύ τους. Η απόσταση μπορεί να είναι μικρή ώστε να δημιουργείται με αυτό το τρόπο ένα παράλληλο σύστημα αλλά μπορεί επίσης να είναι μεγάλη ώστε να δημιουργείται ένα κατανεμημένο σύστημα, ανάλογα πάντα με το είδος των υπολογισμών που θέλουν διαφορετικοί οργανισμοί να υποστηρίξουν [RRAAGSCWGBH02, SC02].

Σε αυτό το πλαίσιο, το πρόβλημα της πιστοποίησης και εξουσιοδότησης είναι πολύ σημαντικό. Η λύση που δίνεται αφορά κυρίως στην ύπαρξη κοινού τρόπου πρόσβασης σε συγκεκριμένα αντικείμενα, για κάθε υπολογιστή που ανήκει στο πλέγμα, ακόμη και αν ο υπολογιστής ανήκει και σε διαφορετικό τοπικό δίκτυο από αυτό του χρήστη.

Ένα από τα πιο σημαντικά συστήματα ενδιάμεσου λογισμικού (middleware) για την δημιουργία και σχεδιασμό ενός πλέγματος υπολογισμού είναι το "Globus". Το εργαλείο "Globus Security Infrastructure" προσφέρει την ασφάλεια και τον έλεγχο πρόσβασης στο πλέγμα. Το σύστημα ασφαλείας του "Globus" είναι ένα από τα πιο κλασσικά συστήματα ασφαλείας που χρησιμοποιείται και από άλλα συστήματα για πλέγματα υπολογισμού.

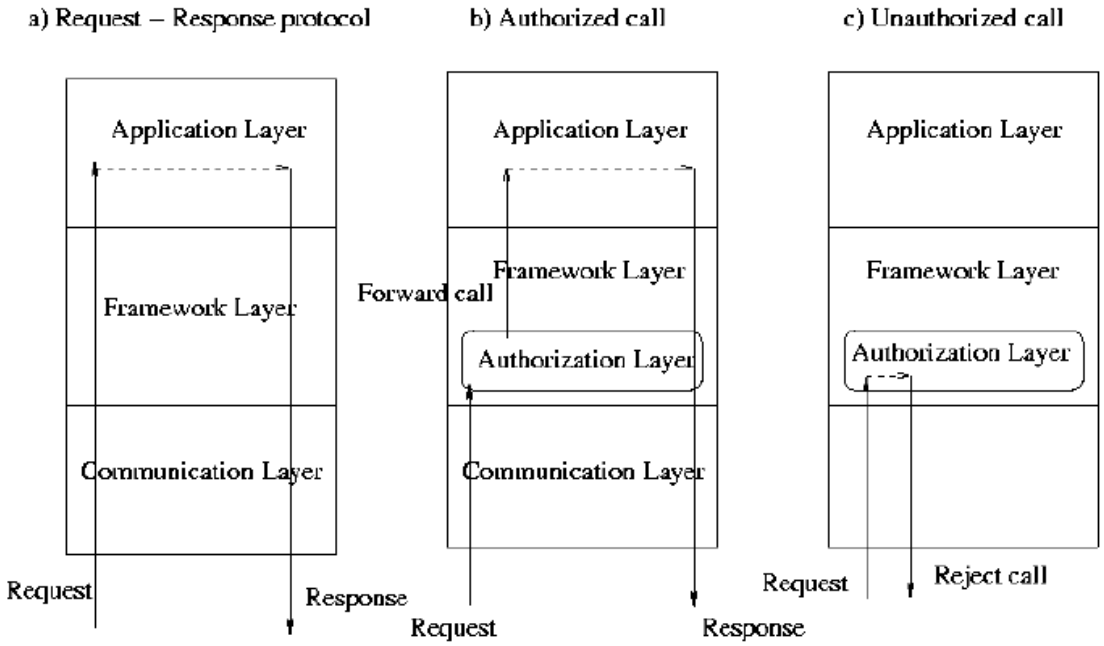
Το σύστημα αυτό βασίζεται στα ψηφιακά πιστοποιητικά. Στην αρχή ο χρήστης δημιουργεί ένα αντικείμενο που δρα ως πληρεξούσιο του χρήστη (Users' Proxy). Επειδή τα διαπιστευτήρια (credentials) του χρήστη στηρίζονται σε ψηφιακά πιστοποιητικά δημιουργούνται προσωρινά πιστοποιητικά για το "User Proxy" που ισχύουν για περιορισμένο χρονικό διάστημα και είναι τα διαπιστευτήρια του πληρεξούσιου ώστε να μπορεί να δρα για λογαριασμό του χρήστη.

Η πιστοποίηση (authentication) των στοιχείων του χρήστη γίνεται κατά την αρχικοποίηση του "User Proxy". Στην συνέχεια, ο "User Proxy" αναλαμβάνει να αποκτήσει πρόσβαση σε οποιοδήποτε μηχάνημα επιθυμεί ο χρήστης. Η εξουσιοδότηση (authorization) όμως του χρήστη στηρίζεται στο υπάρχον λειτουργικό σύστημα που υπάρχει στο μηχάνημα αυτό. Κάθε χρήστης του πλέγματος έχει ένα κοινό όνομα που με αυτό πιστοποιείται στο σύστημα. Σε κάθε μηχάνημα όμως υπάρχει μια συσχέτιση των χρηστών του πλέγματος με τους πραγματικούς χρήστες του λειτουργικού του συστήματος. Έτσι μετά την εξουσιοδότηση πρόσβασης για ένα μηχάνημα ο χρήστης παίρνει τις άδειες που προσφέρει το λειτουργικό σύστημα για τον τοπικό χρήστη με τον οποίο συσχετίστηκε.

Για κάθε αίτηση που θέλει να κάνει ο χρήστης σε κάποιο μηχάνημα (να ξεκινήσει μια διεργασία, να ελέγξει την κατάσταση μιας διεργασίας, να πάρει ένα αρχείο, να σβήσει ένα άλλο αρχείο, και άλλες ενέργειες) ο "User Proxy" αναλαμβάνει να συνδιαλλαγεί με το κατάλληλο μηχάνημα στο οποίο θα γίνουν οι ζητούμενες ενέργειες. Αν κάποια διεργασία θέλει με την σειρά της να επικοινωνήσει με κάποιο άλλο μηχάνημα, τότε μέσω του "User Proxy" η ενέργεια αυτή θα εκτελεστεί με τις κατάλληλες άδειες του χρήστη.

Έχουν γίνει κάποιες προσπάθειες για να προστεθούν κάποιοι πιο λεπτομερείς κανόνες εξουσιοδότησης αλλά πάντα στο επίπεδο της εφαρμογής και όχι στο επίπεδο του "Globus". Στην Εικόνα 2-15 βλέπουμε πως μπορούμε να προσθέσουμε κανόνες εξουσιοδότησης σε XML-RPC μεθόδους όπως η SOAP. Στην

εικόνα αυτή βλέπουμε δύο περιπτώσεις για το πώς μπορεί να γίνει η κλήση μιας συνάρτησης (SOAP) και πώς γίνεται αποδεκτή ή όχι. Όπως φάνηκε παραπάνω μπορούν να οριστούν αρκετές εξουσιοδοτήσεις στην εφαρμογή που εκτελείται στο "Globus". Έχει προταθεί ότι η εξουσιοδότηση πρέπει να ορίζεται με βάση την χρήση της υπολογιστικής ισχύος που κάνει ο χρήστης.



Εικόνα 2-15 Αρχιτεκτονική Υλοποίησης Εξουσιοδότησης στο Globus

Στην πρόταση αυτή, ορίζονται διάφοροι κανόνες που επιβάλλονται στο χρήστη που έχει πιστοποιηθεί με σκοπό να ελέγξουν την χρήση του πλέγματος υπολογισμού. Ένα παράδειγμα κανόνων φαίνεται στην Εικόνα 2-16 που ορίζονται για κάποιο χρήστη με το διακριτικό όνομα "Alice". Φαίνεται ότι υπάρχει ένας περιορισμός πάνω στο πόσες εργασίες μπορεί να εκτελέσει ή όχι (στο παράδειγμα βλέπουμε ότι μπορεί να εκτελέσει το πολύ 15 διεργασίες). Τέτοιοι μηχανισμοί μπορούν να δώσουν κάποια περαιτέρω ευελιξία. Ωστόσο δεν πρόκειται για μέθοδο που μπορεί να εφαρμοστεί γενικά και για κάθε εφαρμογή που εκτελείται σε ένα πλέγμα υπολογισμού.

```
<?xml version="1.0" encoding="utf-8" ?>
<policy_u458>
<initCrediti>1000</initCredit> <availCredit>1000</availCredit>
<maxJobs>15</maxJobs> <group>users</group>
</policy_u458>
```

Εικόνα 2-16 Παράδειγμα Usage Policy

2.4.3 Γλώσσα ελέγχου πρόσβασης

Σε αυτήν την ενότητα παρουσιάζεται μια γλώσσα ορισμού εξουσιοδοτήσεων σε αντικείμενα που βασίζονται στο πρότυπο XML. Αυτό που έχει ενδιαφέρον είναι η δυνατότητα να εξουσιοδοτηθούν διαφορετικοί χρήστες με διαφορετικά δικαιώματα πρόσβασης ώστε να μπορούν να επιτελούν ενέργειες (ανάγνωσης, εγγραφής) στα αντικείμενα.

Ο χρήστης ξεκινάει την διαδικασία ζητώντας την πραγματοποίηση κάποιας ενέργειας στο αντικείμενο (access request). Το σύστημα, σύμφωνα με την πολιτική ελέγχου πρόσβασης που έχει δηλωθεί, ελέγχει τις άδειες του χρήστη και επιτρέπει η αρνείται την συγκεκριμένη ενέργεια πρόσβασης στα αντικείμενα. Στην περίπτωση που την επιτρέπει, την εκτελεί και επιστρέφει τα αποτελέσματα στον χρήστη (π.χ. το έγγραφο που έχει ζητήσει ο χρήστης αν η ενέργεια είναι ανάγνωση).

Στην συνέχεια παρουσιάζουμε το συντακτικό και τα βασικά δικαιώματα πρόσβασης που ορίζει η γλώσσα XACL.

2.4.3.1Συντακτικό XACL

Τα πεδία που καθορίζουν τον κάθε κανόνα στο XACL είναι τέσσερα. Οι κανόνες πρόσβασης είναι εκφρασμένοι σε XML οπότε και προσδιορίζονται από ένα DTD. Παρακάτω αναλύεται το καθένα πεδίο.

subject: Αυτό το πεδίο χαρακτηρίζει τον χρήστη και παρέχει τις πληροφορίες για την αναγνώριση του από το σύστημα. Η σύνταξη του είναι η ακόλουθη:

```
<!ELEMENT   subject     (uid?, role*, group*)>
<!ELEMENT   uid         (#PCDATA)>
<!ELEMENT   role        (#PCDATA)>
<!ELEMENT   group       (#PCDATA)>
```

Όπως φαίνεται ο χρήστης καθορίζεται από το συμβολικό του όνομα (uid) τις ομάδες χρηστών στις οποίες ανήκει (group) και τους ρόλους που του έχουν ανατεθεί (role). Πλέον εδώ δεν υπάρχει η δυνατότητα να διαχωριστούν οι χρήστες ανάλογα με το μηχάνημα στο οποίο εργάζονται.

object: Το αντικείμενο ή το σύνολο αντικειμένων από το έγγραφο, στο οποίο καθορίζεται η άδεια πρόσβασης, το οποίο περιγράφεται από XPath εκφράσεις. Το συντακτικό του object είναι το ακόλουθο:

```
<!ELEMENT   object      EMPTY>
<!ATTLIST   object      href CDATA #REQUIRED>
```

action:

```
<!ELEMENT  action             (provisional_action*)>
<!ATTLIST  action name        (read|write|create|delete) #REQUIRED
                              permission     (grant|deny) #REQUIRED>
<!ELEMENT  provisional_action  (parameter*)>
<!ATTLIST  provisional_action  name CDATA #REQUIRED
                                timing (before|after) "after">
```

Οι πράξεις που υποστηρίζονται είναι τέσσερις: ανάγνωση, εγγραφή, δημιουργία και διαγραφή. Υπάρχει η δυνατότητα να δοθούν θετικές και αρνητικές άδειες για μεγαλύτερη εκφραστικότητα της γλώσσας. Ένα εντελώς νέο χαρακτηριστικό είναι η δυνατότητα ορισμού επιπλέον ενεργειών που θα πραγματοποιηθούν μαζί με την ενέργεια, πριν ή μετά από αυτήν. Παραδείγματος χάριν ένας χρήστης θα ήθελε να καταγραφόταν κάθε ενημέρωση που πραγματοποιείται στο έγγραφο. Έτσι μπορεί να δηλώσει ότι μαζί με την ενέργεια ενημέρωσης (*update*) θα πραγματοποιείται και μια ενέργεια καταγραφής (*log*) αυτόματα από το σύστημα. Το συντακτικό δίνει αυτή την δυνατότατα καθορίζοντας την ενέργεια που θα γίνει, ορίζοντας επιπλέον τις παραμέτρους που απαιτούνται και δηλώνοντας την χρονική στιγμή που θα γίνει η πράξη, πριν ή μετά την βασική ενέργεια. Παραδείγματος χάριν, μια ενέργεια *encrypt* στα δεδομένα που στέλνονται πρέπει να πραγματοποιηθεί προτού σταλούν τα δεδομένα δηλαδή πριν την βασική ενέργεια.

conditions: Εδώ καθορίζεται μια λογική συνθήκη η οποία πρέπει να είναι αληθής για να δοθεί η άδεια στον χρήστη. Λογικοί τελεστές and, or και not μπορούν να χρησιμοποιηθούν για να δημιουργηθούν πολύπλοκες συνθήκες. Η συνθήκη αποτελείται από συναρτήσεις οι οποίες παίρνουν τις επιθυμητές παραμέτρους και επιστρέφουν μια τιμή αληθείας.

```
<!ELEMENT   condition   (predicate| condition)*>
<!ATTLIST   condition   operation (and| or| not) #REQUIRED >
<!ELEMENT   predicate   (parameter*)>
<!ATTLIST   predicate   name CDATA #REQUIRED>
<!ELEMENT   parameter   ANY>
```

```
<!ATTLIST   parameter   name CDATA #IMPLIED>
<!ELEMENT   function    (parameter*)>
<!ATTLIST   function    name CDATA #REQUIRED>
```

Παράδειγμα: Κάποιος ζητά να επιστραφεί στον χρήστη η ενέργεια «είναι η σημερινή ημερομηνία πριν τις 23/4/2004;». Η δήλωση θα ήταν η ακόλουθη:

```
<condition>
 <predicate name="compareDate">
  <parameter>before</parameter>
  <parameter>Today</parameter>
  <parameter>23/4/2004</parameter>
 </predicate>
</condition>
```

2.4.3.2Πολιτική Εξουσιοδοτήσεων XACL

Κάθε τετράδα των παραπάνω πεδίων (subject, object, action, condition) καθορίζει έναν κανόνα πρόσβασης. Εκτός από αυτούς τους κανόνες πρόσβασης που αφορούν στους χρήστες, υπάρχουν και κανόνες πρόσβασης που αφορούν το έγγραφο. Η συνολική πολιτική για το έγγραφο προκύπτει συνδυάζοντας όλους τους παραπάνω κανόνες (ως προς τον χρήστη και ως προς το έγγραφο). Το συντακτικό του XACL για την πολιτική που ακολουθείται είναι το ακόλουθο:

```
<!ELEMENT   policy                  (property?, xacl*)>
<!ELEMENT   xacl                    (object+,rule+)>
<!ELEMENT   rule                    (acl)+>
<!ELEMENT   acl                     (subject*, action+, condition?)>
<!ELEMENT   property                (propagation?,
                                    conflict_resolution?, default?)>
<!ELEMENT   propagation             EMPTY>
<!ATTLIST   propagation             read   (no|up|down) "down"
                                    write  (no|up|down) "down"
                                    create (no|up|down) "no"
                                    delete (no|up|down) "up">
<!ELEMENT   conflict_resolution     EMPTY>
<!ATTLIST   conflict_resolution     read   (dtp|gtp|ntp) "dtp"
                                    write  (dtp|gtp|ntp) "dtp"
                                    create (dtp|gtp|ntp) "dtp"
                                    delete (dtp|gtp|ntp) "dtp">
<!ELEMENT   default                 EMPTY>
<!ATTLIST   default                 read    (grant|deny) "deny"
```

```
write    (grant|deny) "deny"
create   (grant|deny) "deny"
delete   (grant|deny) "deny">
```

Για κάθε πολιτική εξουσιοδότησης πρέπει να καθοριστούν τρία χαρακτηριστικά: (α) Ο τρόπος διάδοσης των κανόνων. Μπορεί να καθοριστούν ότι οι κανόνες που ισχύουν για έναν κόμβο θα ισχύουν και για τους απογόνους του ή για τους πρόγονους του. Υπάρχει επίσης και η δυνατότητα να καθοριστεί ότι ο κανόνας θα ισχύει μόνο για το αντικείμενο για το οποίο ορίστηκε. (β) Το αποτέλεσμα της πολιτικής όταν δυο ή περισσότεροι κανόνες έρχονται σε σύγκρουση. Οι επιλογές είναι να επιτραπεί ή όχι η ενέργεια ή να μην παρθεί καμία απόφαση. (γ) Σε αυτήν την τελευταία περίπτωση το αποτέλεσμα της πολιτικής για την σύγκρουση των κανόνων προέρχεται από την προκαθορισμένη πολιτική (default policy) που είναι και το τελευταίο χαρακτηριστικό της πολιτικής που καθορίζεται. Στην περίπτωση αυτή, αρχικά ελέγχεται αν έχει οριστεί κάποια προκαθορισμένη άδεια για το συγκεκριμένο αντικείμενο. Αν έχει οριστεί, επιβάλλεται αυτή. Αν όχι, επιβάλλεται η προκαθορισμένη άδεια του εγγράφου.

Παράδειγμα πολιτικής: Η παρακάτω πολιτική ορίζει ότι ο χρήστης με όνομα Alice μπορεί να διαβάσει τα αντικείμενα με σημαδούρα XML “contents” αλλά όχι και να τα γράφει:

```
<policy>
 <xacl>
 <object href = “/contents/” />
 <rule>
  <acl>
   <subject>
    <uid>Alice</uid>
   </subject>
   <action name=“read” permission=“grant” />
   <action name=“write” permission=“deny” />
  </acl>
 </rule>
</xacl>
</policy>
```

Αυτό το παράδειγμα παρουσιάζει μια απλή πολιτική που δεν προσδιορίζει κανένα από τα χαρακτηριστικά που αναφέρθηκαν για την πολιτική. Έτσι αυτά παίρνουν τις προκαθορισμένες τους τιμές όπως μπορούν να φανούν από το DTD.

2.4.3.3Καθορισμός πρόσβασης

Χρησιμοποιώντας την εκφραστική δύναμη της XACL, ο διαχειριστής των εγγράφων είναι σε θέση καθορίσει τις άδειες πρόσβασης που επιθυμεί. Εφόσον αυτό γίνει, το σύστημα είναι σε θέση να εξουσιοδοτεί τις ενέργειες στα έγγραφα που οι χρήστες ζητάνε κάθε φορά.

Η διαδικασία του καθορισμού πρόσβασης ξεκινάει με τον χρήστη να κάνει μια αίτηση πρόσβασης (access request) στο σύστημα. Μέσω αυτής ο χρήστης καθορίζει την ταυτότητα του, το αντικείμενο που τον ενδιαφέρει, την ενέργεια που θα εκτελεστεί καθώς και τις ενδεχόμενες παράμερους της. Το συντακτικό του access request περιγράφεται από το παρακάτω DTD:

```
<!ELEMENT   access_req        (object, subject, action)>
<!ATTLIST   access_req        type (query|execute) "query">
<!ELEMENT   subject           (uid?,role*)>
<!ELEMENT   uid               (#PCDATA)>
<!ELEMENT   role              (#PCDATA)>
<!ELEMENT   object            EMPTY>
<!ATTLIST   object            href CDATA #REQUIRED>
<!ELEMENT   action            (parameter*)>
<!ATTLIST   action            name CDATA #REQUIRED>
<!ELEMENT   parameter         ANY>
```

Το γνώρισμα type της κάθε αίτησης πρόσβασης προσδιορίζει τον τύπο της: αίτηση (query) ή εκτέλεση (execute). Στην πρώτη περίπτωση πρέπει να διαπιστωθεί αν ο χρήστης έχει απλώς την άδεια να πραγματοποιήσει την ενέργεια. Στην δεύτερη περίπτωση πρέπει να διαπιστωθεί αν ο χρήστης μπορεί να πραγματοποιήσει την ενέργεια και στην συνέχεια να γίνουν οι ενημερώσεις που απαιτούνται (αν η ενέργεια είναι ενημέρωσης (update)) ή να επιστραφεί το έγγραφο (αν η ενέργεια είναι ανάγνωσης (read)). Η εκτέλεση μιας αίτησης πρόσβασης (execute access request) πραγματοποιείται αφού το σύστημα έχει δεχτεί την αντίστοιχη αίτηση πρόσβασης (query access request) και έχει δώσει την άδεια πρόσβασης.

Για κάθε αίτηση πρόσβασης (access request) το σύστημα παράγει μία απόφαση πρόσβασης (access decision) που περιέχει τα αποτελέσματα για κάθε ενέργεια που έχει ζητηθεί από την αίτηση. Το συντακτικό είναι το ακόλουθο:

```
<!ELEMENT   decision_list          (object, action, decision*)>
<!ATTLIST   decision_list          type (query|execute) #REQUIRED>
<!ELEMENT   decision          (object, subject, action)>
<!ELEMENT   object            EMPTY>
```

```
<!ATTLIST   object              href CDATA #REQUIRED>
<!ELEMENT   subject             (uid?,role*)>
<!ELEMENT   uid                 (#PCDATA)>
<!ELEMENT   role                (#PCDATA)>
<!ELEMENT   action              (parameter*, provisional_action*)>
<!ATTLIST   action              name (read|write|create|delete)
                                #REQUIRED permission
                                (grant|deny) #REQUIRED>
<!ELEMENT   provisional_action      (parameter*)>
<!ATTLIST   provisional_action      name CDATA #REQUIRED>
<!ELEMENT   parameter           ANY>
```

Σύμφωνα με την απόφαση πρόσβασης, το σύστημα θα εκτελέσει τις ενέργειες που προσδιορίζονται στα πεδία αντικειμένου (object) και της ενέργειας (action).

Παράδειγμα: Ο χρήστης “Alice” ζητά να εκτελέσει την ενέργεια read στα αντικείμενα “contents”, οπότε σύμφωνα με την πολιτική που αναφέρθηκε στο προηγούμενο παράδειγμα, το σύστημα παράγει την ακόλουθη απόφαση πρόσβασης, η οποία είναι θετική, σύμφωνα με τη καθορισμένη πολιτική.

```
<access_req type=“query”>
  <object href=“/contents/ />
  <subject>
    <uid>Alice<uid>
  </subject>
  <action name=“read”/>
</access_req>
```

```
<decision_list
<object
    href=“/contents/list/entry[position()=
    ” />
<action
<decision
<object
    href=“/contents/list/entry[position()=
    ” />
<subject>
   <uid>Alice</ uid>
</subject>
<action name=“read” permission”grant”
</decision
</decision_list
```

2.4.3.4Αλγόριθμος καθορισμού πρόσβασης

Matching: Στο πρώτο βήμα ο αλγόριθμος παίρνει σαν είσοδο μία αίτηση πρόσβασης (access request) και παράγει μια απόφαση πρόσβασης (access decision) βασισμένη στην πολιτική εξουσιοδότησης που έχει οριστεί. Για κάθε ζευγάρι αντικειμένου (object), χρήστη (subject) που υπάρχει στην αίτηση, ο αλγόριθμος βρίσκει το αντίστοιχο ζευγάρι στον ορισμό της πολιτικής ελέγχοντας αν η συνθήκη

(condition) είναι αληθής. Στην περίπτωση που υπάρχει ένα ζευγάρι, εισάγεται στην απόφαση (decision) που παράγεται. Όταν βρεθούν δυο κανόνες της πολιτικής που έρχονται σε σύγκρουση μεταξύ τους, αποφασίζεται για το ποιος ισχύει ανάλογα με το τι έχει δηλωθεί στην πολιτική στο πεδίο επίλυσης σύγκρουσης (conflict_resolution).

Propagation: Γνωρίζοντας την απόφαση από το πρώτο βήμα, η απόφαση διαδίδεται στους απόγονους ή στους πρόγονους ανάλογα με το τι έχει δηλωθεί στην πολιτική στο πεδίο propagation. Στην περίπτωση που έχει δηλωθεί άρνηση της διάδοσης (no) η απόφαση δεν διαδίδεται καθόλου. Η διάδοση γίνεται καλώντας αναδρομικά το πρώτο βήμα του αλγορίθμου δημιουργώντας μια νέα αίτηση πρόσβασης (access request) κάθε φορά.

Apply Default policy: Στην περίπτωση που για κάποιο από τα δύο παραπάνω βήματα δεν έχει ληφθεί καμία απόφαση, αυτή παράγεται χρησιμοποιώντας την προκαθορισμένη πολιτική (default policy) που έχει οριστεί.

Select one decision: Αντίθετα, στην περίπτωση που από τα δύο πρώτα βήματα έχουν παραχθεί παραπάνω από μια αποφάσεις, οι οποίες έρχονται σε αντιπαράθεση, διαλέγεται μόνο μια χρησιμοποιώντας την επίλυση σύγκρουσης (conflict_resolution). Σε περίπτωση που δεν έχει οριστεί προκαθορισμένη πολιτική επιλέγεται αυθαίρετα μια απόφαση.

Request Execution: Έχοντας φτάσει σε αυτό το σημείο το σύστημα έχει δημιουργήσει μια απόφαση πρόσβασης (access decision) με τις ενέργειες που ζητήθηκαν από τον χρήστη, οι οποίες και θα εκτελεστούν, καθώς σύμφωνα με την πολιτική ο χρήστης έχει το δικαίωμα να τις εκτελέσει. Έτσι, για κάθε ενέργεια, ο χρήστης εκτελεί τυχόν προσωρινές ενέργειες (provisional actions) που έχουν δηλωθεί να εκτελεστούν πριν την ενέργεια, κατόπιν την ενέργεια και τέλος τις προσωρινές ενέργειες (provisional actions) που έχουν δηλωθεί μετά την ενέργεια (after).

2.4.3.5Κατανεμημένος Έλεγχος Πρόσβασης με XACL

Για να υλοποιηθεί το προηγούμενο μοντέλο για ένα κατανεμημένο σύστημα υπάρχουν δύο τρόποι: (α) Η απόφαση και η επιβολή της πολιτικής πρόσβασης να βρίσκονται στο ίδιο μέρος (είτε στο ίδιο μηχάνημα ή να εκτελούνται από την ίδια εφαρμογή). (β) Η απόφαση και η επιβολή της πολιτικής πρόσβασης να εκτελούνται ξεχωριστά και σε διαφορετικά μέρη.

Ο μηχανισμός που χρειάζεται για να εκφέρει μια απόφαση πολιτικής πρόσβασης χρειάζεται να έχει τις εξής συνιστώσες (components): (α) Μια αποθήκη (repository) που να περιέχει τις πολιτικές ελέγχου πρόσβασης, (β) τα δεδομένα που χρειάζονται για να αποφασιστεί η πολιτική πρόσβασης, (γ) τέλος, τη συνιστώσα που είναι υπεύθυνη για την διαχείριση των πολιτικών ελέγχου πρόσβασης. Για την επιβολή μιας πολιτικής πρόσβασης, ο μηχανισμός χρειάζεται την συνιστώσα που εκτελεί την πρόσβαση στο αντικείμενο αυτό.

Ο μηχανισμός που έχει σε διαφορετικό μέρος την επιβολή πολιτικής πρόσβασης από την απόφαση της, χρησιμοποιείται όταν τα δεδομένα στα οποία γίνεται ο έλεγχος πρόσβασης είναι πολλά. Με αυτό τον τρόπο βοηθάει στο να είναι πιο δομημένες οι πολιτικές ελέγχου πρόσβασης. Ακόμη είναι ευκολότερο να υπάρξουν βάσεις δεδομένων για την αποθήκευση και ανεύρεση των πολιτικών ελέγχου πρόσβασης. Όμως επειδή η επιβολή μιας πολιτικής πρόσβασης χρειάζεται την απόφαση που έχει παρθεί, αυτή πρέπει να σταλεί στην συνιστώσα ή εφαρμογή που εκτελεί την επιβολή της πολιτικής πρόσβασης. Και για λόγους ασφαλείας πρέπει η επικοινωνία αυτή να είναι ασφαλής (κρυπτογραφημένη αν μιλάμε για απομακρυσμένα μηχανήματα) και κυρίως να γνωρίζουμε ότι δεν έχει αλλάξει από την αποστολή της (χρησιμοποιείται η τεχνική των ψηφιακών υπογραφών).

Όταν όμως τα δεδομένα στα οποία θέλουμε να γίνει ο έλεγχος πρόσβασης είναι λίγα, είναι καλύτερο οι πολιτικές ελέγχου πρόσβασης να βρίσκονται στο ίδιο μέρος, πιθανότατα και στο ίδιο αντικείμενο.

2.5 Συμπεράσματα

Το πρόβλημα που θα μας απασχολήσει στο επόμενο κεφάλαιο είναι το πρόβλημα της εξουσιοδότησης σε ροές επιστημονικών εργασιών. Συνήθως, οι εν λόγω ροές είναι μια συλλογή από προγράμματα επιστημονικών υπολογισμών που εκτελούνται με κάποιο συνδυασμένο τρόπο έτσι ώστε να μπορούν να χρησιμοποιήσουν ως είσοδο τα αποτελέσματα κάποιου προηγούμενου προγράμματος αν υπάρχει. Τα επιστημονικά προγράμματα που εκτελούνται συνήθως είναι χρονοβόρα και απαιτούν αρκετή υπολογιστική ισχύ, οπότε η πρόσβαση για την εκτέλεση τέτοιων προγραμμάτων αποτελεί ένα σημαντικό παράγοντα. Συνήθως μια ροή επιστημονικών εργασιών εκτελείται από ένα χρήστη που έχει τις κατάλληλες άδειες, και σε μερικές μόνο περιπτώσεις χρειάζεται την εκτέλεση κάποιων διεργασιών από κάποιον άλλο χρήστη (όπως γίνεται σε

προσομοιώσεις επιστημονικών εργαστηρίων). Σε τέτοιες ροές είναι σπάνιο να υπάρχει μια διεργασία που να εκτελείται από άνθρωπο και όχι από κάποιο υπολογιστικό σύστημα.

Μας ενδιαφέρουν ακόμη οι ροές εργασιών που εκτελούνται σε περισσότερους από έναν οργανισμό. Έτσι είναι αναγκαία η κατανεμημένη εκτέλεσης της ροής εργασίας καθώς και του ελέγχου πρόσβασης. Οι χρήστες, οι διεργασίες και τα αντικείμενα είναι πιθανό να μην βρίσκονται στον ίδιο οργανισμό οπότε ο ορισμός της σχέσης τους να μην είναι κάτι το προφανές.

Τα μοντέλα που παρατηρήσαμε δεν μας δίνουν μια λύση στο πρόβλημα μας γιατί δεν είχαν σχεδιαστεί για μία τέτοια εφαρμογή. Το μοντέλο RBAC ενώ δίνει αρκετές ιδιότητες για τον εύκολο ορισμό εξουσιοδοτήσεων, δεν είναι προφανές πως μπορεί να εφαρμοστεί σε παραπάνω από ένα οργανισμό. Τα χαρακτηριστικά πιστοποίησης μπορούν να λύσουν κάποιο μέρος του προβλήματος, αλλά όπως αναφέρθηκε δεν υπάρχει τρόπος να καταλάβουμε για ποιόν λόγο δεν δόθηκε πρόσβαση σε περίπτωση αποτυχίας.

Το μοντέλο WAM μπορεί να χρησιμοποιηθεί σε ένα κατανεμημένο σύστημα, όμως δεν μας δίνει ένα τρόπο να ελέγξουμε αν ένας χρήστης μπορεί να εκτελέσει μια ολόκληρη ροή εργασίας ή τι χρειάζεται για την εκτέλεση της. Το τελευταίο μάλιστα είναι και ο λόγος που τα προηγούμενα μοντέλα δεν μπορούν να εφαρμοστούν στο πρόβλημα της εξουσιοδότησης επιστημονικών εφαρμογών. Τα μοντέλα που παρουσιάστηκαν έχουν σχεδιαστεί για επιχειρησιακές διεργασίες που η πιθανότητα να εκτελεστεί μια ροή εργασίας από αρκετούς χρήστες είναι αρκετά μεγάλη.

Τέλος, το μοντέλο εξουσιοδότησης βασισμένο σε διεργασίες και το μοντέλο WAM μπορούν να μας βοηθήσουν στο να χειριστούμε τις άδειες που μας προσφέρει ένα λειτουργικό σύστημα ή κάποια υλοποίηση όπως το “Globus”. Οπότε, συνδυάζοντας τα επιμέρους χαρακτηριστικά και τις δυνατότητες κάθε μοντέλου θα παρουσιάσουμε στο επόμενο κεφάλαιο μια λύση για το πρόβλημα της εξουσιοδότησης σε επιστημονικές ροές επιστημονικών εργασιών.

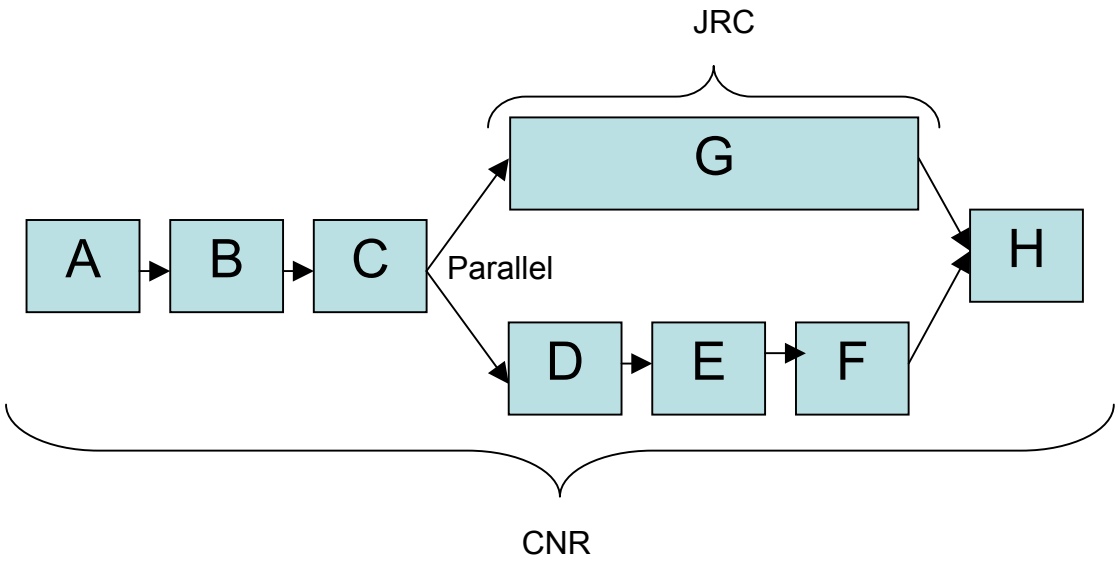
3 Μηχανισμός Εξουσιοδότησης για Ροές Επιστημονικών Εργασιών

Στο κεφάλαιο αυτό θα παρουσιάσουμε το μοντέλο που προτείνεται για την υποστήριξη εξουσιοδοτημένης πρόσβασης σε ροές επιστημονικών εργασιών, οι οποίες μπορούν να εκτελούνται σε διαφορετικούς οργανισμούς. Η εξουσιοδοτημένη πρόσβαση σε αυτό το πλαίσιο είναι ένα ενδιαφέρον πρόβλημα γιατί ένα μέρος από τις διεργασίες, ή κάποια από τα δεδομένα που χρειάζονται οι διεργασίες μιας ροής βρίσκονται σε οργανισμούς διαφορετικούς των υπολοίπων διεργασιών και η πιστοποίηση των χρηστών καθώς και η εξουσιοδότηση πρέπει να γίνεται αυτόνομα σε κάθε οργανισμό. Τα κύρια προβλήματα που θα μας απασχολήσουν είναι πρώτον πως μπορούμε να συμπεράνουμε αν ένας χρήστης επιτρέπεται να εκτελέσει μια ροή εργασίας και δεύτερον, στην περίπτωση που δεν μπορεί, πως θα καταφέρουμε να προτείνουμε τι χρειάζεται ο χρήστης για να την εκτελέσει.

3.1 Παράδειγμα Ροής Επιστημονικών Εργασιών

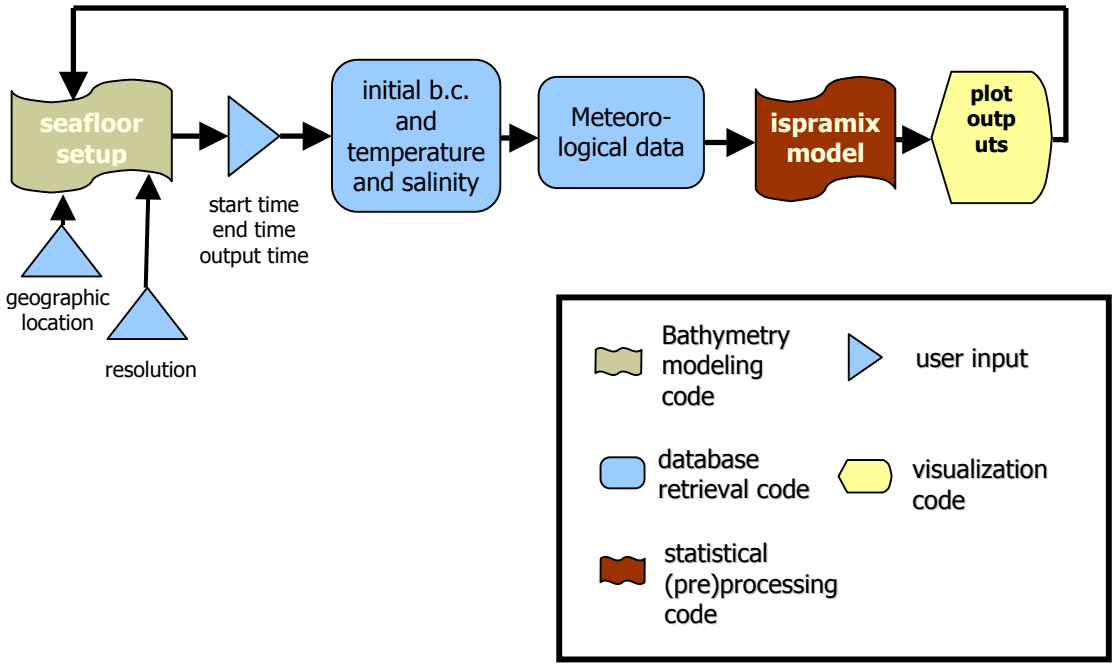
Θα προσπαθήσουμε να δείξουμε ένα παράδειγμα χρήσης ενός μηχανισμού διαχείρισης επιστημονικών διεργασιών. Για να κατανοήσουμε τις απαιτήσεις ενός τέτοιου μηχανισμού θα παρουσιάσουμε ένα παράδειγμα εκτέλεσης μιας επιστημονικής ροής εργασίας μέσα από το κατανεμημένο σύστημα ARION. Μία από τις ροές εργασίας που υπάρχει στο σύστημα ARION είναι αυτή μεταξύ των δύο οργανισμών JRC και CNR. Στην ροή εργασίας εκτελούνται εφτά από τις οχτώ διεργασίες στον οργανισμό CNR και μόνο μία στον οργανισμό JRC. Η γραφική αναπαράσταση της ροής αυτής φαίνεται στην Εικόνα 3-1. Παρατηρούμε ακόμη ότι η διεργασία "G" εκτελείται παράλληλα με τις διεργασίες "D", "E", "F".

Τα δεδομένα που παράγονται από κάποια διεργασία και χρησιμοποιούνται από κάποια άλλη ορίζουν διάφορες εξαρτήσεις (dependencies) μεταξύ αυτών τον διεργασιών. Οι εξαρτήσεις ορίζουν την σειρά με την οποία πρέπει να εκτελεστούν οι διεργασίες επιστημονικών διεργασιών. Πολλές φορές, στις επιστημονικές διεργασίες ο τρόπος που συνδυάζονται οι διεργασίες είναι με βάση τις εξαρτήσεις αυτές.



Εικόνα 3-1 Γράφημα Ροής Εργασίας JRC/CNR

Ο σκοπός αυτής της ροής εργασίας είναι να προσομοιώσει μέσω ωκεανογραφικών μοντέλων το ύψος της θάλασσας, την θερμοκρασία, την αλμυρότητα και τα ρεύματα της θάλασσας. Ο χρήστης μπορεί να ορίσει πια περιοχή που τον ενδιαφέρει, τον χρόνο που δίνεται σαν η αρχή της προσομοίωσης και τα χρονικά διαστήματα όπου θέλει να πάρει τα αποτελέσματα του. Οι βιολόγοι χρειάζονται πολλές φορές τέτοια δεδομένα για να υπολογίσουν πως τα θρεπτικά συστατικά που έρχονται από τα ποτάμια ή δημιουργούνται από τις ακτές, μετακινούνται στην θάλασσα μετά από κάποιο χρόνο. Έτσι θα μπορούν να προβλέψουν τι αλλαγές θα υπάρχουν στο θαλάσσιο οικοσύστημα σε κάποια χρονική στιγμή. Στο σύστημα ARION, αυτή η ροή εργασίας έχει εξειδικευτεί για περιοχές της Αδριατικής θάλασσας, επειδή υπάρχουν τα κατάλληλα δεδομένα γι' αυτή την θάλασσα. Τέλος υπάρχουν δύο οργανισμοί συμβούλων που ενδιαφέρονται για τα αποτελέσματα αυτής της ροής εργασίας γιατί θέλουν να εκφέρουν απόψεις και συμβουλές σε ζητήματα περιβαλλοντικής πολιτικής. Οι μετασχηματισμοί που γίνονται στα δεδομένα για το μοντέλο αυτό φαίνονται στην Εικόνα 3-2. Τα δύο μοντέλα που εκτελούνται είναι το μοντέλο που υπολογίζει τον βυθό της Αδριατικής "seafloor model" και το μοντέλο που προσομοιώνει τον ωκεανό και την ατμόσφαιρα "ispramix model". Μπορούμε να δούμε ποια δεδομένα ορίζει ο χρήστης ως είσοδο και ποια δεδομένα υπάρχουν στο σύστημα.



Εικόνα 3-2 Μετασχηματισμός Δεδομένων για το Ωκεανογραφικό μοντέλο

Στην ροή εργασίας της Εικόνα 3-1 υπάρχουν οι εξής διεργασίες:

A: Input Applet (CNR). Σε αυτή τη διεργασία ο χρήστης ορίζει τις παραμέτρους που θέλει, ώστε οι επόμενες διεργασίες να εκτελεστούν μόνο για την περιοχή που έχει ορίσει ο ίδιος και όχι για όλα τα δεδομένα που υπάρχουν στο σύστημα.

B: Merge Dataset (CNR). Έχοντας τις επιλογές του χρήστη καθώς και ποια βαθυμετρικά δεδομένα θα χρησιμοποιηθούν, στον υπολογισμό αυτό ενοποιούνται τα δεδομένα αυτά και παράγουν τα δεδομένα που χρειάζονται οι υπόλοιπες εφαρμογές. Τα βαθυμετρικά δεδομένα προέρχονται από μετρήσεις που έχουν γίνει σε κάποια σημεία της θάλασσας ενώ τα υπόλοιπα παράγονται μετά από υπολογισμό.

C: TIN Interpolation (CNR). Στο σημείο αυτό παράγονται τα βαθυμετρικά δεδομένα για τα σημεία εκείνα που υπάρχουν δεδομένα αλμυρότητας, ανέμων, κυμάτων και ρευμάτων. Πιθανότατα, τα βαθυμετρικά δεδομένα να είναι ορισμένα σε διαφορετικά σημεία από τα σημεία εκείνα που υπάρχουν οι κατάλληλες μετρήσεις αλμυρότητας, ανέμων, κυμάτων και ρευμάτων.

D: Gridded Bathymetry to metadata & Metadata Loader (CNR). Εδώ παράγονται τα κατάλληλα μετά-δεδομένα για κάθε σημείο των

βαθυμετρικών δεδομένων. Έτσι μπορεί να βρεθούν ευκολότερα σε μια μελλοντική εκτέλεση της ροής.

E: Scale latitude and longitude (CNR). Εδώ κλιμακώνονται τα δεδομένα σε μια άλλη κλίμακα ώστε να είναι εύκολη η γραφική τους αναπαράσταση.

F: Gridded Bathymetry to VRML (CNR). Δημιουργείται από τα προηγούμενα δεδομένα η τρισδιάστατη αναπαράσταση του βυθού μέσω της πρότυπης γλώσσας περιγραφής τρισδιάστατων αντικειμένων VRML.

G: Ispramix Adriatic set-up (JRC). Στο σημείο αυτό γίνεται η προσομοίωση με τα στοιχεία αλμυρότητας, ρευμάτων, και των υπολοίπων στοιχείων για μια συγκεκριμένη χρονική στιγμή. Επειδή τα δεδομένα αυτά παράγονται και ελέγχονται από τον οργανισμό JRC, είναι ευαίσθητα δεδομένα και ο υπολογισμός πρέπει να γίνει αναγκαστικά στον οργανισμό αυτό. Επί πλέον το μοντέλο Ispramix έχει αναπτυχθεί τα τελευταία 20 χρόνια από τον οργανισμό JRC και δεν είναι μετακινήσιμο.

H: Merge Results (CNR). Και τέλος συνδυάζονται τα δυο αποτελέσματα για να δημιουργηθεί μια τρισδιάστατη αναπαράσταση της προσομοίωσης.

Στην συνέχεια θα παρουσιάσουμε κατά πόσο οι μηχανισμοί εξουσιοδότησης, που παρουσιάστηκαν στο προηγούμενο κεφάλαιο, μπορούν να χρησιμοποιηθούν στην ροή εργασίας του παραδείγματος μας. Είναι προφανές ότι δεν μπορούν να χρησιμοποιηθούν μοντέλα που ανήκουν στον Προαιρετικό Έλεγχο Πρόσβασης γιατί είναι αδύνατον ο δημιουργός (διαχειριστής) της ροής εργασίας να ορίσει για κάθε επιστημονική διεργασία όλους τους πιθανούς χρήστες που μπορούν να την εκτελέσουν. Τα μοντέλα που ανήκουν στον Υποχρεωτικό Έλεγχο Πρόσβασης μας προσφέρουν καλύτερες λύσεις αλλά όχι ακριβώς αυτές που απαιτούνται. Μοντέλα σαν το μοντέλο Bell & Lapadula δεν μας δίνουν κατάλληλο έλεγχο πρόσβασης γιατί η ιεραρχία των χρηστών είναι αρκετά απλή και μπορούμε να δούμε ότι η διεργασία που έχει την πιο αυστηρή ετικέτα είναι αυτή που θα καθορίσει και την πρόσβαση για όλη την ροή εργασίας. Μάλιστα τα θετικά στοιχεία του μοντέλου Bell & Lapadula, που αφορούν τη ροή της πληροφορίας, δεν μπορούν να χρησιμοποιηθούν αποδοτικά στην περίπτωση μας, γιατί σε μια ροή εργασίας η ροή της πληροφορίας επιβάλλεται από την ροή εργασίας. Το μοντέλο του ελέγχου πρόσβασης βασισμένο σε ρόλους μας προσφέρει μια πολύ καλή μέθοδο για την ιεράρχηση των πιθανών

ρόλων των χρηστών, αλλά απαιτεί την δημιουργία πολλών αδειών ανά διεργασία. Επιπλέον, το παραπάνω προσδιορίζει τις άδειες πάνω στην συσχέτιση ρόλων – αδειών, κάτι το οποίο γίνεται δύσχρηστο αν πρέπει να ορίσουμε τις άδειες αυτές για κάθε διεργασία σε οποιαδήποτε ροή εργασίας στο σύστημα. Ταυτόχρονα το μοντέλο αυτό δεν μας προσφέρει την κατάλληλη οργάνωση χρηστών για να προτείνει λύσεις σε συστήματα πολλών οργανισμών.

Από την άλλη πλευρά, το μοντέλο εξουσιοδότησης, βασισμένο σε διεργασίες, μας προσφέρει μια λύση για να ρυθμίζουμε εξουσιοδοτήσεις βασισμένες στα λειτουργικά συστήματα των μηχανών που υλοποιούν τις επιστημονικές διεργασίες. Μέσω του μοντέλου αυτού μπορούμε να ρυθμίσουμε τις άδειες για τα αρχεία που περιέχουν τα δεδομένα ή τα αποτελέσματα της εκτέλεσης των επιστημονικών διεργασιών. Όμως δεν μας προσφέρει ένα τρόπο να εξουσιοδοτήσουμε αυτή καθ' αυτή την εκτέλεση της διεργασίας. Το βασικό μοντέλο WAM καθώς και οι εξελίξεις του μας προσφέρουν την δυνατότητα να ορίσουμε ποιοι χρήστες «πρέπει» να εκτελέσουν τις διεργασίες για να ολοκληρωθεί η ροή εργασίας. Δυστυχώς, ο τρόπος αυτός δεν μας παρέχει την δυνατότητα να ορίσουμε στις διεργασίες κατάλληλες άδειες για τους χρήστες. Το μόνο που μπορούμε να ορίσουμε είναι υποχρεώσεις. Έτσι το μοντέλο WAM δεν επιτρέπει ούτε τον έλεγχο εγκυρότητας (validation) των εξουσιοδοτήσεων που ορίζονται, ούτε μπορεί να κάνει υποδείξεις στην περίπτωση που ένας χρήστης δεν μπορεί να εκτελέσει μια ροή εργασίας.

Τα σημαντικά αντικείμενα σε μια ροή επιστημονικών εργασιών είναι τα δεδομένα και η ροή που έχουν μέσα σε αυτή. Ο βασικός λόγος για την εκτέλεση κάποιων ροών επιστημονικών εργασιών είναι η μετατροπή των δεδομένων. Σε μια ροή επιστημονικών εργασιών υπάρχουν διάφορα δεδομένα που χρησιμοποιούνται από τις διεργασίες. Τα δεδομένα αυτά είναι δύο τύπων: (α) Στατικά δεδομένα, δηλαδή υπάρχουν κάπου μέσα στο σύστημα διαχείρισης επιστημονικών ροών εργασίας (αρχεία, βάσεις δεδομένων, κλπ). (β) Δυναμικά δεδομένα, δηλαδή δημιουργούνται κατά την εκτέλεση διαφόρων επιστημονικών διεργασιών. Το ερώτημα που θα μας απασχολήσει είναι αν τα δεδομένα χρειάζεται να έχουν και αυτά τις δικές τους εξουσιοδοτήσεις.

Θα δείξουμε ότι οι εξουσιοδοτήσεις για στατικά ή δυναμικά δεδομένα κατά την διάρκεια της εκτέλεσης μια ροής επιστημονικών εργασιών πρέπει να είναι ίδιες με αυτές των διεργασιών που τις χρησιμοποιούν και τις παράγουν. Τα στατικά

δεδομένα που υπάρχουν στο σύστημα, είναι τα δεδομένα για τα οποία θα μπορούσαμε να ορίσουμε εξουσιοδοτήσεις. Οπότε, αυτά τα δεδομένα, αν είναι να έχουν εξουσιοδοτήσεις, θα πρέπει να είναι ίδιες με τις εφαρμογές που τα χρησιμοποιούν. Τα δυναμικά δεδομένα, που παράγονται από την εκτέλεση κάποιας εφαρμογής, δεν μπορούμε να ορίσουμε εξαρχής εξουσιοδοτήσεις. Έτσι όταν ένας ρόλος εκτελεί μια διεργασία και αυτή παράγει κάποια δεδομένα τότε σίγουρα πρέπει να έχει και τις άδειες για να αποκτήσει πρόσβαση σε αυτά. Επειδή τα δεδομένα μπορεί να μην υπάρχουν από την αρχή είναι δύσκολο να ορίσουμε γι' αυτά εξουσιοδοτήσεις και για το λόγο αυτό θα ορίσουμε εξουσιοδοτήσεις μόνο για τις διεργασίες.

Τα δεδομένα δεν χρειάζονται να έχουν εξουσιοδοτήσεις ακόμη και στην περίπτωση που έχουν στατικές χρεώσεις γι' αυτά. Οι χρεώσεις αυτές μπορούν να θεωρηθούν στις διεργασίες που τις καλούν. Στην αναφορά αυτή δεν θα ασχοληθούμε με δεδομένα που χρεώνονται διαφορετικά με τον χρόνο.

Αυτό που χρειάζεται να υποστηριχθεί από ένα μοντέλο για ροές επιστημονικών εργασιών είναι: (α) Εύκολη διαχείριση των χρηστών, γνωρίζοντας ότι μπορεί να ανήκουν σε διαφορετικούς οργανισμούς (β) Διαφορετικές ιεραρχίες χρηστών ανά οργανισμό του οποίου οι διεργασίες συμμετέχουν σε μία ροή, δηλαδή κάθε οργανισμός θα πρέπει να έχει την δική του όψη στην σχέση χρηστών – ρόλων (γ) Δυνατότητα ορισμού πολλαπλών εξουσιοδοτήσεων για μία διεργασία (δ) Έγκυρη επιλογή του ρόλου που αναλαμβάνει την εκτέλεση της διεργασίας, ώστε να χρησιμοποιούνται οι κατάλληλες άδειες (ε) Δυνατότητα χρέωσης για την εκτέλεση κάποιας διεργασίας (στ) Υποστήριξη κατάλληλων προτάσεων όταν ένας χρήστης δεν μπορεί να εκτελέσει μία ή περισσότερες διεργασίες.

3.2 Τυπικό Μοντέλο μιας Ροής Επιστημονικής Εργασίας

Στην ενότητα αυτή θα παρουσιάσουμε έναν τυπικό ορισμό μιας επιστημονικής ροής εργασίας, τον οποίο θα χρησιμοποιήσουμε στην συνέχεια για την θεμελίωση του μηχανισμού εξουσιοδότησης που προτείνουμε.

Η ροές επιστημονικών εργασιών θεωρούνται ότι είναι δομημένες ροές εργασίας (structured workflows), που σημαίνει ότι κάθε κόμβος συντονισμού parallel, choice, while συσχετίζεται με ακριβώς ένα κόμβο συντονισμού synchronize, merge, do αντίστοιχα. Μια καλά δομημένη (well structured) ροή

εργασίας αποτελείται από m κόμβους που εκτελούνται σειριακά T_1, T_2, ..., T_m. Ο κάθε κόμβος T_i μπορεί να είναι είτε κάποια διεργασία (Activity) που δεν μπορεί να αναλυθεί περισσότερο, είτε κάποια σύνθεση διεργασιών (Sub Process) που αποτελείται από n_i παράλληλες ή σειριακές ή υπό συνθήκη διεργασίες.

Έστω G = <N, H> ένα δέντρο. N: είναι ένα πεπερασμένο σύνολο από κόμβους, και H: οι σχέσεις πατέρα - απογόνου όπου $H \subseteq N \times N$ που ορίζουν την δομή της ροής εργασίας.

- $\forall$ n $\in$ N, Node_Type: n $\rightarrow$ {Control Flow, Task}. Κάθε κόμβος του γράφου ανήκει σε ένα από δύο τύπους. Οι τύποι αυτοί είναι: (α) "Task" όταν θέλουμε να ορίσουμε ότι περιγράφουμε μια διεργασία. (β) "Control Flow" όταν περιγράφουμε τα στοιχεία που ορίζουν την ροή της εργασίας.
- $N = C \cup T$, $C \cap T = \emptyset$ όπου C: το σύνολο των κόμβων που είναι του τύπου "Control Flow", T: το σύνολο των κόμβων που είναι του τύπου "Task". Έτσι έχουμε ορίσει ότι δεν υπάρχουν κόμβοι που ανήκουν και στους δύο τύπους.
- $\forall$ n $\in$ C, Control_Flow_Type: n $\rightarrow$ {Sequence, Parallel, Choice, While}. Κάθε τέτοιος κόμβος έχει ως απογόνους του κόμβους τύπου "Task". Κάθε κόμβος του τύπου "Control Flow", μπορεί να είναι ένα από τους τύπους συντονισμού (coordination):

  (α) "Sequence" όταν οι κόμβοι που ορίζονται σε αυτόν εκτελούνται σειριακά. Στο κόμβο αυτό ορίζεται η σχέση ροής $F \subseteq N \times N$ όπου επιβάλλεται η σειρά εκτέλεσης των απογόνων.

  (β) "Parallel" όταν οι κόμβοι που ορίζονται σε αυτόν εκτελούνται ταυτόχρονα.

  (γ) "Choice" όταν εκτελούνται ένας ή περισσότεροι κόμβοι οι οποίοι ορίζονται σε αυτόν.

  (δ) "While" είναι ο κόμβος που ορίζει έναν κόμβο που μπορεί να εκτελεστεί μία ή περισσότερες φορές.
- Η συνολική σχέση ροής $F \subseteq N \times N$ ορίζεται από το σύνολο των ροών που έχουν οριστεί στους κόμβους τύπου "Sequence", και αντικαθιστώντας τους κόμβους τύπου "Control Flow", με τους

κόμβους που περιέχουν. Αναλυτικότερα στην περίπτωση του "Sequence":

- Αν (ni, nj) $\in$ F και Control_Flow_Type(nj) = "Sequence", τότε αντικαθιστούμε το nj με τον αρχικό του κόμβο που ορίστηκε στην ροή σχέση του.
- Αν (nj, ni) $\in$ F και Control_Flow_Type(nj) = "Sequence", τότε αντικαθιστούμε το nj με τον τελικό του κόμβο που ορίστηκε στην ροή σχέση του.

- Παρατηρούμε ότι η συνολική σχέση ροής F είναι η κατά βάθος διάσχιση του δέντρου (depth first traversal), και σημειώνοντας μόνο τα φύλλα.
- $\forall$ n $\in$ T, Task_Type: T $\rightarrow$ {Activity, Sub Process}. Όπου "Activity" αναπαριστά την εκτέλεση μίας διεργασίας, ενώ "Sub Process" αναπαριστά την εκτέλεση μια σύνθεσης διεργασιών ή μια άλλη ροή εργασίας. Ο κόμβος όπου είναι "Sub Process" μπορεί να αναλυθεί περισσότερο, ορίζοντας ένα κόμβο τύπου "Control Flow".
- Έστω P ένα κατευθυνόμενο μονοπάτι στο γράφο G, τέτοιο ώστε P = {n1,n2, …, nk} , (ni, ni+1) $\in$ F για i = 1, 2, …, k-1. Τα πιθανά μονοπάτια που υπάρχουν σε μία ροή εργασίας ορίζονται ως ακολούθως ($\mathbb{N}$: το σύνολο φυσικών αριθμών):

(α) $\forall$ n $\in$ N, I:N $\rightarrow$ $\mathbb{N}$, I(n): Είναι μια συνάρτηση που μας δείχνει τον αριθμό των εισερχόμενων ροών για τον κόμβο n.

(β) $\forall$ n $\in$ N, O:N $\rightarrow$ $\mathbb{N}$, O(n): Είναι μια συνάρτηση που μας δείχνει τον αριθμό των εξερχόμενων ροών για τον κόμβο n.

- Μια ροή εργασίας είναι ένας κατευθυνόμενος μη κυκλικός γράφος (DAG) W = <N, H, F> τέτοιος ώστε:
  - $\forall$ n $\in$ N, $\exists$ P, τέτοιο ώστε P = {n0, …n, …, nf}. (n0 είναι ο αρχικό κόμβος της ροής και nf ο τελικός κόμβος) Με την προηγούμενη συνθήκη διαβεβαιώνουμε ότι κάθε κόμβος ανήκει σε ένα τουλάχιστον μονοπάτι από την αρχή του γράφου μέχρι το τέλος του.

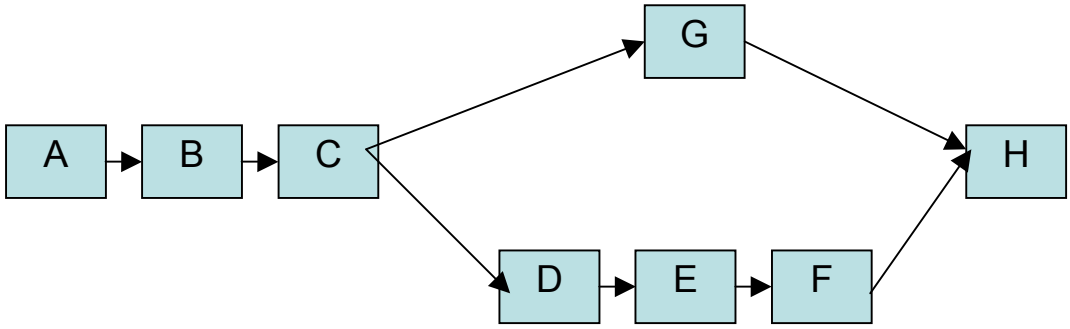
- ∀ n ∈ T, I(n) + O(n) > 1 τέτοιο ώστε n != n0 και n !=nf. Κάθε κόμβος τύπου "Task" έχει τουλάχιστον μία εισερχόμενη ροή και μια εξερχόμενη ροή. Οι μόνοι κόμβοι που δεν έχουν εισερχόμενες και εξερχόμενες ροές είναι οι αρχικοί και τελικοί κόμβοι.

Παρατηρούμε λοιπόν ότι μία ροή εργασίας μπορεί να αναλυθεί έτσι ώστε, η αναπαράσταση της σε δένδρο να αποτελείται από τους κόμβους τύπου "Task" για φύλλα και τους υπόλοιπους κόμβους που είναι τύπου "Control Flow".

Χρησιμοποιώντας τον μαθηματικό ορισμό που δόθηκε παραπάνω, μπορούμε να ορίσουμε την ροή εργασίας του παραδείγματος ως εξής:

- C = {n0, n1, n2}
- Control_Flow_Type(n0) = Sequence, Control_Flow_Type(n1) = Parallel, Control_Flow_Type(nf) = Sequence.
- T = {A, B, C, D, E, F, G, H}
- ∀ n ∈ T, Task_Type(n) = Activity
- F = {(A, B), (B, C), (C, D), (C, G), (D, E), (E, F), (F, H), (G, H)}

Μια γραφική αναπαράσταση των ροών φαίνεται στην Εικόνα 3-3:

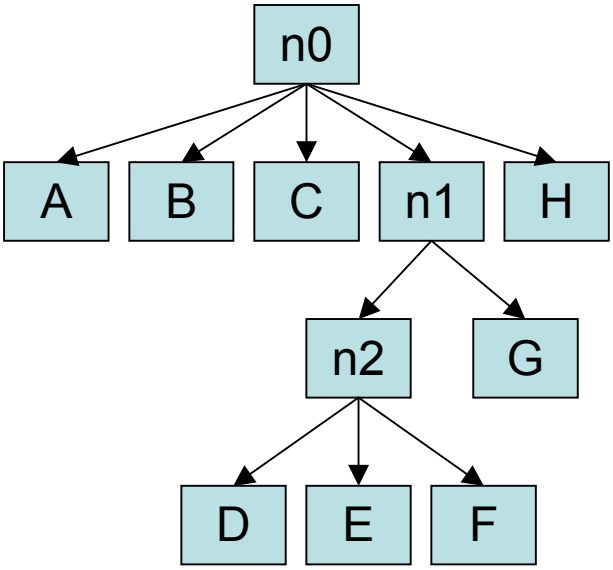


Εικόνα 3-3 Γραφική αναπαράσταση της ροής εργασίας

Βλέπουμε ότι η ροή εργασίας που παρουσιάζουμε στο παράδειγμα αυτό είναι μια καλά δομημένη ροή εργασίας γιατί μπορούμε με αφαίρεση να δημιουργήσουμε μια σειριακή ροή δεδομένων. Παρακάτω παρουσιάζουμε την γενική δομή της ροής εργασία και αναλύουμε κάθε σύνθετη διεργασία. Στην Εικόνα 3-4 φαίνεται γραφικά η δομή της ροής εργασίας.

```
Workflow:
      sequence:
            A, B, C, n1, H
      end
```

```
n1: Sub Process
      parallel:
            G, n2
      end
n2: Sub Process
      sequence:
            D, E, F
      end
A, B, C, D, E, F, G, H: Task
```



Εικόνα 3-4 Δομή Ροής Εργασίας

3.3 Ιεραρχία Χρηστών

Σε πολλά σύγχρονα λειτουργικά συστήματα, οι χρήστες ορίζονται σε ένα μέρος του συστήματος και έχουν καθολική ισχύ, ανεξάρτητα από την μηχανή στην οποία ο χρήστης είναι συνδεδεμένος. Για παράδειγμα σε συστήματα τύπου "UNIX" οι χρήστες ορίζονται στο αρχείο "/etc/passwd" μαζί με τα διακριτικά τους στοιχεία. Άλλα συστήματα, όπως το Zope, ορίζουν τους χρήστες σε σχέση με διάφορους καταλόγους του συστήματος αρχείων. Με τον τρόπο αυτό ο χρήστης έχει δικαίωμα πρόσβασης μόνο στα αντικείμενα και τους καταλόγους που είναι απόγονοι του καταλόγου στον οποίον έχει αρχικά οριστεί.

Επειδή προσπαθούμε να παρέχουμε εξουσιοδοτημένη πρόσβαση σε χρήστες που ανήκουν σε διαφορετικούς οργανισμούς, είναι προφανές ότι δεν μπορούμε να οργανώσουμε τους χρήστες με έναν καθολικό τρόπο. Μια χρήσιμη οργάνωση των χρηστών είναι όταν αναπαριστούν την δομή των διαφόρων οργανισμών που υπάρχουν στο κατανεμημένο σύστημα μας. Ο τρόπος που θα παρουσιάσουμε

παρακάτω στηρίζεται στο πρότυπο LDAP (Lightweight Directory Access Protocol) το οποίο ορίζει έναν κατανεμημένο κατάλογο χρηστών, ο οποίος είναι δομημένος σύμφωνα με τον οργανισμό στον οποίο ανήκουν οι χρήστες. Το πρότυπο αυτό ορίζει εκτός από τους χρήστες και διάφορα αντικείμενα ώστε να μπορούν να αποθηκευτούν στον κατάλογο εικόνες, αρχεία και άλλα. Η πιο συνηθισμένη χρήση του προτύπου αυτού είναι η αποθήκευση προφίλ χρηστών για διάφορες εφαρμογές.

Ο ορισμός που παρουσιάζουμε στην συνέχεια είναι ένα υποσύνολο του προτύπου LDAP:

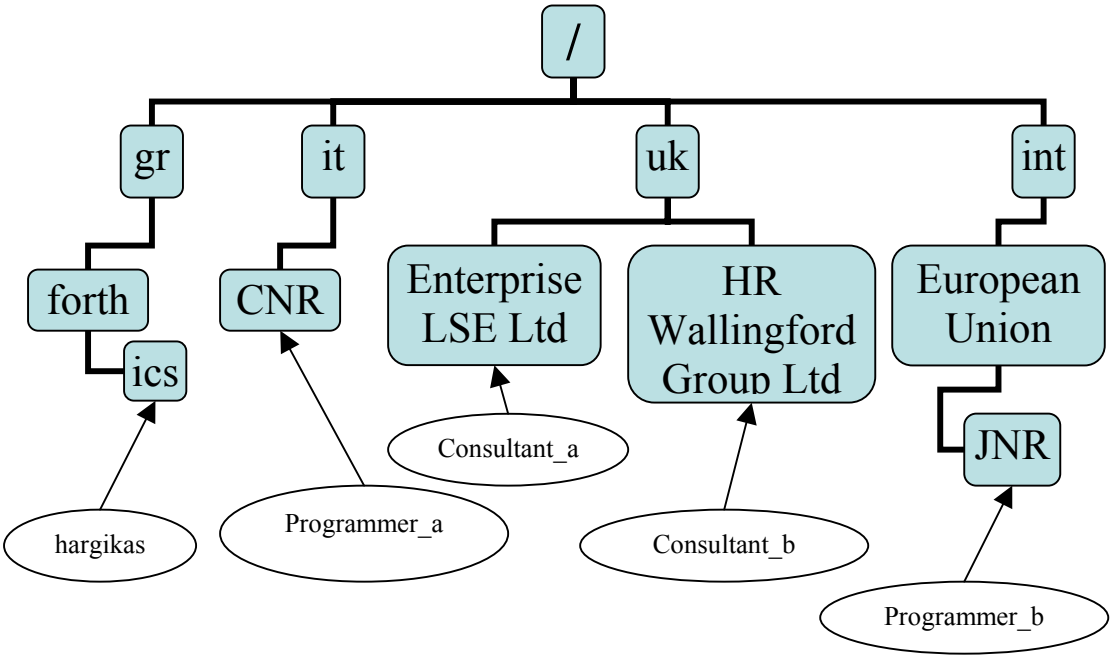
- Έστω U το σύνολο των χρηστών
- Έστω OU το σύνολο των οργανισμών ή τμημάτων των οργανισμών.
- $OUH \subseteq OU \times OU$ είναι η δενδρική ιεραρχία των οργανισμών ή τμημάτων τους. Η OUH είναι μια μερική διάταξη πάνω στους οργανισμούς ή τμήματά τους.
- $OUA \subseteq OU \times U$ είναι η ανάθεση των χρηστών σε αντικείμενα οργανισμών ή τμημάτων τους. Ένας χρήστης μπορεί να ανατεθεί σε ένα και μόνο οργανισμό ή τμήμα του ενώ σε ένα οργανισμό ή τμήμα του μπορεί να έχουν ανατεθεί πολλοί χρήστες.

Μια διαφορετική οπτική γωνία για να κατανοήσουμε την ιεραρχία που ορίστηκε προηγουμένως είναι να την θεωρήσουμε σαν ένα δέντρο όπου οι κόμβοι είναι οργανισμοί ή τμήματά τους και τα φύλλα είναι χρήστες. Οι χρήστες μπορεί να έχουν το διακριτικό όνομα τους και τα διαπιστευτήρια τους (είτε αυτά είναι κρυπτογραφημένοι κωδικοί πρόσβασης είτε τα δημόσια κλειδιά από τα ψηφιακά χαρακτηριστικά του χρήστη).

Ο έλεγχος και η διαχείριση αυτής της ιεραρχίας είναι κατανεμημένη. Οι διαχειριστές ανήκουν σε κάποιο οργανισμό ή τμήμα του, και είναι εξουσιοδοτημένοι για την διαχείριση όλων των χρηστών που ανήκουν στο ίδιο οργανισμό ή τμήμα. Είναι επίσης υπεύθυνοι για την διαχείριση όλων των οργανισμών ή τμημάτων τους που είναι απόγονοι του οργανισμού στον οποίο ανήκει ο διαχειριστής, αν τα αντικείμενα αυτά δεν έχουν το δικό τους διαχειριστή. Δηλαδή κάθε οργανισμός ή χρήστης είναι υπό διαχείριση από τον διαχειριστή που ανήκει στον κοντινότερο πρόγονο του στην ιεραρχία.

Όπως θα δούμε και στην επόμενη ενότητα, που περιγράφουμε μια προτεινόμενη αρχιτεκτονική για το σύστημα εξουσιοδότησης επιστημονικών ροών

εργασίας, η αποθήκευση του δέντρου γίνεται σε διαφορετικούς υπολογιστές ανάλογα με τον οργανισμό.



Εικόνα 3-5 Γραφική αναπαράσταση Ιεραρχίας Χρηστών

Στην Εικόνα 3-5, βλέπουμε ένα παράδειγμα ιεραρχίας πέντε οργανισμών. Ένας από τους οργανισμούς είναι το ITE (FORTH) που έχει σαν τμήμα το ινστιτούτο πληροφορικής, το οποίο υλοποίησε το σύστημα ARION. Υπάρχουν ακόμα δύο οργανισμοί το CNR και το JNR, οι οποίοι ορίζουν την ροή εργασίας που παρουσιάστηκε στο προηγούμενο κεφάλαιο, καθώς και δύο οργανισμούς συμβούλων όπου χρειάζονται να εκτελούν την ροή εργασίας που αναφέραμε προηγουμένως για να μπορούν να εκφέρουν διάφορες απόψεις και συμβουλές σε ζητήματα περιβαλλοντικής πολιτικής.

Σε άσπρους κύκλους στην Εικόνα 3-5, βλέπουμε τους διάφορους χρήστες του συστήματος και σε ποιους ρόλους ανήκουν. Για παράδειγμα, ο χρήστης "hargikas" ανήκει στο Ινστιτούτο Πληροφορικής του ITE και έχει το διακριτικό όνομα `"user=hargikas, ou=ics, ou=forth, ou=gr"`. Γενικότερα το διακριτικό όνομα ενός χρήστη περιλαμβάνει το όνομα του (hargikas) καθώς και τα ονόματα όλων των οργανισμών στην ιεραρχία στην οποία ανήκει (ics, forth, gr).

3.4 Ιεραρχία Ρόλων

Η ιεραρχία των χρηστών από μόνη της δεν είναι αρκετά περιγραφική ώστε να μπορούμε να ορίσουμε εξουσιοδοτήσεις για τις διεργασίες μιας ροής επιστημονικών εργασιών. Αν στηριζόμασταν στην προηγούμενη ιεραρχία χρηστών, θα μπορούσαμε να προσφέρουμε εξουσιοδοτημένη πρόσβαση μόνο σε χρήστες ορισμένων οργανισμών ανεξάρτητα από τον ρόλο που έχουν σ' ένα οργανισμό για την εκτέλεση ή όχι των διεργασιών μιας ροής εργασίας.

Αντιθέτως, έχοντας μια ιεραρχία ρόλων πετυχαίνουμε να δούμε έναν χρήστη τόσο από πλευράς οργανισμού, αλλά και από πλευράς του ρόλου που έχει στην εσωτερική οργάνωση του οργανισμού.

Για τον ορισμό μιας ιεραρχίας ρόλων θα χρησιμοποιήσουμε ένα μέρος από το μοντέλο RBAC, που αναφέρθηκε στο προηγούμενο κεφάλαιο. Δεν μπορούμε όμως να χρησιμοποιήσουμε αυτούσιο αυτό το μοντέλο γιατί εκτός από την συσχέτιση των χρηστών με τους ρόλους, το RBAC συσχετίζει και τους ρόλους και με τις άδειες πρόσβασης. Κάτι τέτοιο δεν λαμβάνει υπόψη τις εξουσιοδοτήσεις που πρέπει να εκχωρούνται σε μια διεργασία κατά την εκτέλεση μιας ροής εργασίας. Επειδή μάλιστα υπάρχουν πολλές ροές εργασίας που μπορούν να εκτελεστούν από τους ίδιους ή διαφορετικούς ρόλους χρειαζόμαστε ένα εκφραστικότερο μοντέλο εξουσιοδοτημένης πρόσβασης.

Για το μοντέλο μας, παραθέτουμε τους εξής ορισμούς:

- Έστω R το σύνολο των ρόλων
- $RH \subseteq R \times R$, Η ιεραρχία των ρόλων RH είναι η μερική διάταξη πάνω στους ρόλους R. Η ιεραρχία καλείται επίσης ως σχέση κυριαρχίας (domination) ρόλων και συμβολίζεται ως $\geq$. Δηλαδή όταν ένας ρόλος κυριαρχεί πάνω σε κάποιους ρόλους, τότε μπορεί να δρα σαν αυτούς.

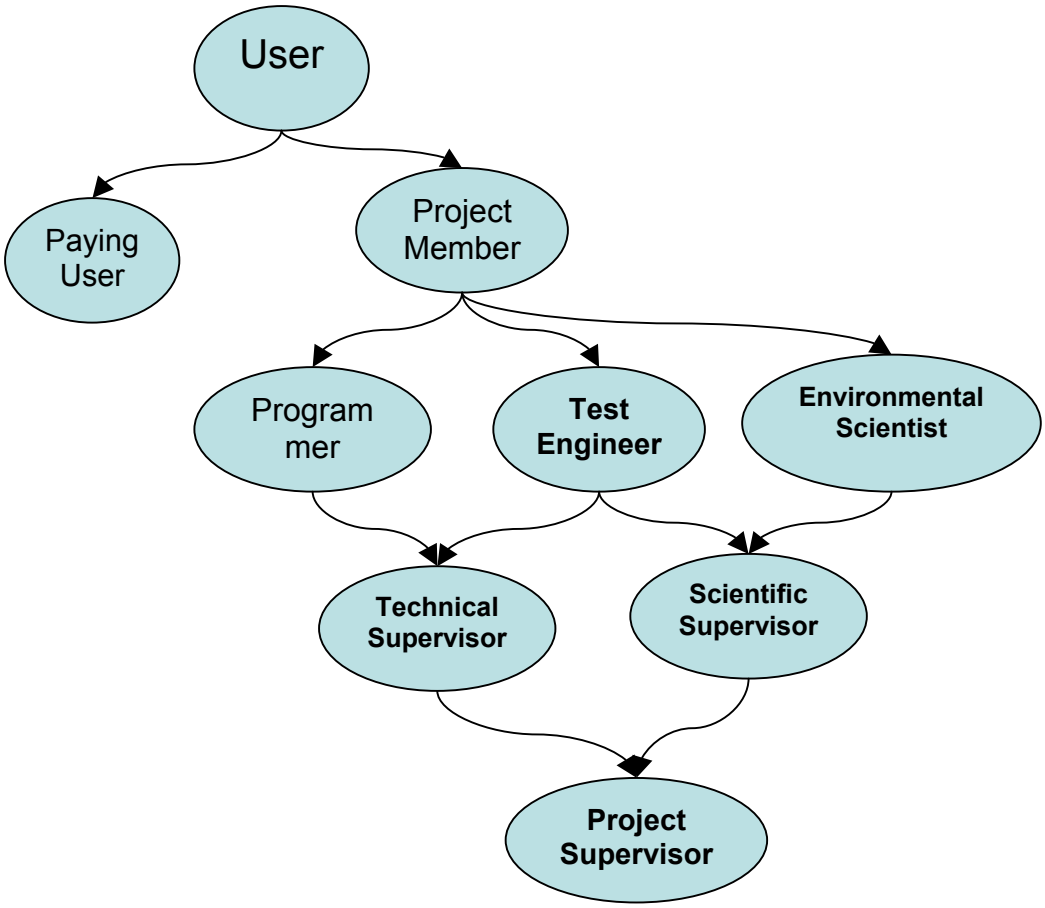
Οι ρόλοι συσχετίζονται με τους χρήστες ως εξής:

- Έστω U το σύνολο των χρηστών
- Έστω R το σύνολο των ρόλων
- $UA \subseteq U \times R$, είναι σχέση ανάθεσης χρήστη ρόλου. Ένας χρήστης μπορεί να συσχετιστεί με πολλούς ρόλους, όπως και ένας ρόλος να συσχετιστεί με πολλούς χρήστες.

- roles: $U \rightarrow 2^R$, είναι μια συνάρτηση που αντιστοιχεί έναν χρήστη $u_i \in U$ με κάποιους από τους ρόλους που του έχουν ανατεθεί. Δηλαδή $roles(u_i) \subseteq \{[(u_i, r) \in UA]\}$
- extended_ roles: $U \rightarrow 2^R$, είναι μια συνάρτηση που αντιστοιχεί έναν χρήστη $u_i \in U$ με κάποιους ρόλους που είτε του έχουν ανατεθεί άμεσα είτε κληρονομούνται από την ιεραρχία των ρόλων. Δηλαδή $extended\_roles(u_i) \subseteq \{r'|(\exists r' \geq r)[(u_i, r') \in UA]\}$

Με τους παραπάνω ορισμούς έχουμε ορίσει την συσχέτιση χρήστη με ρόλους, καθώς ορίσαμε και δύο συναρτήσεις αντιστοίχησης όπου μπορούμε να δούμε ποιους ρόλους έχει ο χρήστης αυτός.

Η ιεραρχία των ρόλων είναι κοινή για όλο το σύστημα σε αντίθεση με την συσχέτιση των χρηστών η οποία δεν είναι καθολική. Για κάθε οργανισμό που έχει οριστεί στην ιεραρχία χρηστών, ορίζονται οι συσχετίσεις των χρηστών με τους ρόλους. Έτσι ένας χρήστης μπορεί να ανήκει σε διαφορετικούς ρόλους τόσο στα πλαίσια του ίδιου οργανισμού όσο και σε διαφορετικούς οργανισμούς.



Εικόνα 3-6 Ιεραρχία Ρόλων

Στην Εικόνα 3-6 βλέπουμε ένα παράδειγμα ιεραρχίας ρόλων. Ο πιο βασικός ρόλος είναι ο "User" και όλοι οι υπόλοιποι ρόλοι μπορούν να υπαχθούν σε αυτόν. Όσο «κατεβαίνουμε» στο γράφημα, βλέπουμε τους ρόλους που έχουν όλο και μεγαλύτερη «κυριαρχία». Με τον όρο «κυριαρχία» εννοούμε ότι ένας ρόλος «Α» «κυριαρχεί» πάνω σε κάποιον άλλο ρόλο «Β», τότε ο ρόλος «Α» μπορεί να δράσει σαν το ρόλο «Β». Βλέπουμε ότι υπάρχει ο ρόλος "Paying User", όπου σε αυτό τον ρόλο συσχετίζονται οι χρήστες που διατίθενται να πληρώσουν για τις υπηρεσίες που χρησιμοποιούν. Ορίζονται ακόμη ρόλοι προγραμματιστών για τους χρήστες που υλοποιούν κάποιο συγκεκριμένο πρόγραμμα (πιθανότατα μια διεργασία της ροής εργασίας). Ο ρόλος "Test Engineer" αναπαριστά τους χρήστες που είναι υπεύθυνοι για τον έλεγχο ορθότητας κάποιων διεργασιών. "Environmental Scientist" είναι ο ρόλος που συσχετίζεται με τους επιστήμονες που χρειάζονται την εκτέλεση μιας ροής εργασίας για επιστημονικούς λόγους. Έπειτα, βλέπουμε πως οι ρόλοι «επίβλεψης» κληρονομούν τους κατάλληλους ρόλους ώστε να μπορούν να εκτελέσουν επιτυχώς την διεργασία που εποπτεύουν.

Στην συνέχεια παρουσιάζουμε την συσχέτιση ορισμένων χρηστών στις ιεραρχίες των οργανισμών "ou=CNR, ou=it" και "ou=JNR, ou=European Union, ou=int":

- ou=CNR, ou=it:
  - Programmer = {"user=**Programmer_a**, ou=CNR, ou=it"}
  - Test Engineer = {"user=**Programmer_b**, ou=JNR, ou=European Union, ou=int", "user=**hargikas**, ou=ics, ou=forth, ou=gr"}
  - Paying User = {"user=**hargikas**, ou=ics, ou=forth, ou=gr"}
  - Environmental Scientist = {"user=**Consultant_b**, ou=HR Wallingford Group Ltd ,ou=uk"}
  - Scientific Supervisor = {"user=**Consultant_a**, ou=Enterprise LSE Ltd, ou=uk"}
- ou=JNR, ou=European Union, ou=int:
  - Programmer = {"user=**Programmer_b**, ou=JNR, ou=European Union, ou=int"}
  - Test Engineer = {"user=**Programmer_a**, ou=CNR, ou=it", "user=**hargikas**, ou=ics, ou=forth, ou=gr"}
  - Paying User = {"user=**Consultant_b**, ou=HR Wallingford Group Ltd, ou=uk"}
  - Scientific Supervisor = {"user=**Consultant_a**, ou=Enterprise LSE Ltd, ou=uk"}

Με το παραπάνω παράδειγμα, βλέπουμε ότι ο χρήστης "user=**Programmer_a**, ou=CNR, ou=it" έχει τον ρόλο του προγραμματιστή για τον οργανισμό CNR, ενώ για τον οργανισμό JNR αναλαμβάνει το ρόλο του ελεγκτή. Ενώ ο χρήστης "user=**hargikas**, ou=ics, ou=forth, ou=gr" έχει τον ρόλο του ελεγκτή και στις δύο περιπτώσεις.

3.5 Εξουσιοδότηση Διεργασιών

Έχοντας ορίσει την ιεραρχία χρηστών καθώς και την ιεραρχία των ρόλων, αυτό που απομένει είναι ο ορισμός της εξουσιοδότησης μιας διεργασίας που ανήκει

σε μια ροή εργασίας. Αλλά πρώτα θα πρέπει να εισάγουμε τις έννοιες των αδειών και των συναλλακτικών μονάδων (credits).

3.5.1 Άδειες Διεργασιών

Στις διεργασίες δεν μπορούμε να ορίσουμε πολλά είδη αδειών όπως στην περίπτωση των αντικειμένων αρχείων, βάσεων δεδομένων κλπ. Η κύρια άδεια που μπορούμε να εκχωρήσουμε σε κάποια διεργασία είναι αυτή της εκτέλεσης.

Πιο συγκεκριμένα ορίζουμε δύο είδη αδειών:

- `Execute`. Με την άδεια αυτή ένας χρήστης εξουσιοδοτείται να εκτελέσει μια διεργασία. Η εκτέλεση της διεργασίας θα γίνει στην τοποθεσία όπου διατίθεται μια υπηρεσία επιστημονικού υπολογισμού.
- `Exclusive Execute`. Με την άδεια αυτή ένας χρήστης εξουσιοδοτείται να εκτελέσει μια διεργασία αλλά η διεργασία του θα είναι η μόνη που εκτελείται στην τοποθεσία εκτέλεσης. Υπάρχουν τρεις περιπτώσεις που πρέπει να λάβουμε υπόψη:
  - Αν υπάρχουν άλλες διεργασίες (που εκτελούνταν με την άδεια "`Execute`") θα σταματήσουν προσωρινά και θα ξεκινήσουν από το σημείο που σταμάτησαν όταν τερματίσει η διεργασία που είχε άδεια "`Exclusive Execute`".
  - Αν υπάρχει κάποια διεργασία που εκτελείται με την άδεια "`Exclusive Execute`", τότε η διεργασία θα εκκινήσει μόλις τελειώσει αυτή που ήδη εκτελείται.
  - Αν υπάρχουν πολλές διεργασίες που θέλουν να εκτελεστούν με άδεια "`Exclusive Execute`", τότε αυτές εισάγονται σε μια ουρά αναμονής (FIFO: First In – First Out) και εκτελούνται με την σειρά, κάθε μία τη στιγμή που η προηγούμενη της σειράς έχει τερματιστεί.

Για τον σωστό χειρισμό των διεργασιών που εκτελούνται χρειαζόμαστε εργαλεία χαμηλού επιπέδου χειρισμού όπως π.χ. να τίθενται κάποιες διεργασίες ως διεργασίες υπόβαθρου (background processes). Τέτοια καθήκοντα μπορεί να τα αναλαμβάνει η πλατφόρμα πρακτόρων (agent platform) του ARION που εκτελεί τις διεργασίες στις διάφορες τοποθεσίες.

3.5.2 Συναλλακτικές Μονάδες

Ως συναλλακτικές μονάδες (credits) θεωρούμε είτε χρήματα είτε κάποιους υπολογιστικούς πόρους (resources). Αν οι συναλλακτικές μονάδες αντιπροσωπεύουν κάποιο χρηματικό ποσό τότε η εκτέλεση μιας διεργασίας προϋποθέτει τον έλεγχο αν ο χρήστης διαθέτει το κατάλληλο ποσό για την εκτέλεση της. Μετά την εκτέλεση της, αυτό το πόσο θα αφαιρεθεί από το αρχικό που είχε στην διάθεση του ο εν λόγω χρήστης. Αν όμως οι μονάδες υποδηλώνουν κάποιο υπολογιστικό πόρο, για παράδειγμα την μέγιστη μνήμη που δικαιούται ο χρήστης να χρησιμοποιήσει ή πόσο χρόνο χρησιμοποιεί το σύστημα, τότε για κάθε διεργασία χρειαζόμαστε να ελέγχουμε αν το πόσο που χρειάζεται η διεργασία είναι μικρότερο ή ίσο με το ποσό που διαθέτει ο χρήστης. Η διαφορά με τις συναλλακτικές μονάδες που αντιπροσωπεύουν κάποιο χρηματικό ποσό είναι ότι για τις συναλλακτικές μονάδες που αντιπροσωπεύουν υπολογιστικούς πόρους δεν αφαιρείται κάποιο ποσό από αυτό που διαθέτει ο χρήστης μετά την εκτέλεση της διεργασίας.

Ορίζουμε την συνάρτηση:

- $\forall u_i \in U$, credits: $U \rightarrow \mathbb{N}$, credits(u_i): Η συνάρτηση αυτή μας επιστρέφει το πόσο των συναλλακτικών μονάδων που έχει ο χρήστης.

3.5.3 Ορισμός Εξουσιοδότησης

Για κάθε διεργασία που υπάρχει σε μία ροή εργασίας μπορούμε να ορίσουμε την εξής πλειάδα:

- AC(n, r, p, c) όπου $n \in T$ (T είναι το σύνολο κόμβων της ροής εργασίας που είναι τύπου “Task”), $r \in R$ (R είναι το σύνολο των ρόλων), $p \in$ {Execute, Exclusive Execute} και c είναι το ποσό των συναλλακτικών μονάδων.

Η σημασία της παραπάνω πλειάδας είναι ότι για τον n κόμβο της ροής εργασίας, ο ρόλος r θα αποκτήσει την άδεια p εφόσον ο χρήστης έχει τουλάχιστον c συναλλακτικές μονάδες.

Για να μπορεί να εκτελέσει ένας χρήστης u μια συγκεκριμένη διεργασία θα πρέπει να ισχύει η ακόλουθη συνθήκη:

- r ∈extended_roles(u) ∧c ≤ credits(u), δηλαδή πρέπει ο χρήστης να μπορεί να αποκτήσει τον ρόλο που περιγράφεται στην πλειάδα πρόσβασης και να έχει τουλάχιστον c συναλλακτικές μονάδες.

Επιπλέον χρειαζόμαστε μια βοηθητική συνάρτηση η οποία μας επιστρέφει όλες τις πλειάδες AC που αφορούν ένα συγκεκριμένο κόμβο μιας ροής εργασίας.

- ACs(n) = {AC(n,r,p,c) ∈ACLs}, όπου ACLs είναι το σύνολο όλων των πλειάδων που έχουν οριστεί για αυτήν την ροή εργασίας.

Ένας χρήστης μπορεί να αποκτήσει το δικαίωμα εκτέλεσης (αποκλειστικής ή όχι) μιας διεργασίας για την οποία έχει οριστεί ένα σύνολο από πλειάδες ACs μόνο άν ισχύει η παρακάτω συνθήκη:

- ∃ AC(n,r,p,c) ∈ ACs(n) τέτοιο ώστε r ∩ extended_roles(u) != Ø ∧ c ≤ credits(u)

Χρησιμοποιώντας τους παραπάνω ορισμούς μπορούμε να θέσουμε εξουσιοδοτήσεις για τους κόμβους της ροής επιστημονικών εργασιών του παραδείγματος μας. Οι συναλλακτικές μονάδες που χρησιμοποιούνται στο παρακάτω παράδειγμα αναπαριστούν χρηματικά ποσά.

- ACs(A) = {AC(A, "User", Execute, 0)}
- ACs(B) = {AC(B, "Project Member", Execute, 0), AC(B, "Environmental Scientist", Exclusive Execute, 10), AC(B, "Paying User", Exclusive Execute, 20)}
- ACs(C) = {AC(C, "User", Execute, 0) ,AC(C, "Paying User", Exclusive Execute, 10)}
- ACs(D) = {AC(D, "User", Execute, 0) ,AC(D, "Paying User", Exclusive Execute, 10)}
- ACs(E) = {AC(E, "Programmer", Execute, 0), AC(E, "Test Engineer", Execute, 10), AC(E, "Scientific Supervisor", Exclusive Execute, 10), AC(E, "Paying User", Exclusive Execute, 20)}
- ACs(F) = { AC(F, "Programmer", Execute, 0), AC(F, "Scientific Supervisor", Exclusive Execute, 10), AC(F, "Paying User", Exclusive Execute, 20)}
- ACs(G) = {AC(G, "Programmer", Execute, 0), AC(G, "Test Engineer", Execute, 10), AC(G, "Paying User",

Exclusive Execute, 50), AC(G, "Environmental Scientist", Exclusive Execute, 20)}

- ACs(H) = {AC(H, "User", Execute, 0)}

Έχοντας ορίσει τις πιθανές εξουσιοδοτήσεις μπορούμε να ελέγξουμε αν ένας χρήστης μπορεί να εκτελέσει και με ποια συγκεκριμένη άδεια μια συγκεκριμένη διεργασία. Για παράδειγμα, ας θεωρήσουμε ότι ο χρήστης "ou=hargikas, ou=ics, ou=forth, ou=gr" θέλει να αποκτήσει πρόσβαση στην διεργασία "G". Επειδή η διεργασία "G" ορίσθηκε και εκτελείται στον οργανισμό "ou=JNR, ou=European Union, ou=int", γνωρίζουμε ότι ο ρόλος του χρήστη αυτού είναι "Test Engineer". Άρα η μοναδική εξουσιοδότηση που μπορεί να εφαρμοστεί είναι η AC(G, "Test Engineer", Execute, 10).

Με παρόμοιο τρόπο μπορούμε να εξάγουμε ότι για τον χρήστη "user=**Consultant_a**, ou=Enterprise LSE Ltd, ou=uk", που θέλει να αποκτήσει πρόσβαση στην διεργασία "B", υπάρχουν δύο εξουσιοδοτήσεις που μπορεί να επιλέξει: AC(B, "Project Member", Execute, 0) και AC(B, "Environmental Scientist", Exclusive Execute, 10). Αυτό συμβαίνει διότι ο ρόλος "Environmental Scientist" κυριαρχεί του ρόλου "Project Member" σύμφωνα με την ιεραρχία ρόλων που παρουσιάσαμε στην Εικόνα 3-6. Δηλαδή ο χρήστης μπορεί να διαλέξει αν θέλει να τρέξει την διεργασία ως επιστήμονας περιβάλλοντος και να πάρει τα αποτελέσματα του γρηγορότερα ή να εκτελέσει την διεργασία ως μέλος του προγράμματος ARION.

Για να διαλέξει ο μηχανισμός την κατάλληλη εξουσιοδότηση για τον χρήστη, όταν υπάρχουν μία ή περισσότερες εξουσιοδοτήσεις, πρέπει να ορίσουμε ορισμένες πολιτικές εκτέλεσης. Δύο πολιτικές που θα υλοποιηθούν στην επόμενη ενότητα είναι: (α) η πολιτική της φθηνότερης εκτέλεσης (β) η πολιτική καλύτερης άδειας. Στην πρώτη περίπτωση διαλέγουμε από το σύνολο των πιθανών εξουσιοδοτήσεων που μπορεί ο χρήστης να χρησιμοποιήσει αυτήν που χρειάζεται τις λιγότερες συναλλακτικές μονάδες. Αν υπάρχουν δυο ή περισσότερες εξουσιοδοτήσεις που χρειάζονται το ίδιο ποσό συναλλακτικών μονάδων για να εκτελεστούν διαλέγουμε αυτήν που δίνει την καλύτερη άδεια. Στην δεύτερη περίπτωση διαλέγουμε από το σύνολο των εξουσιοδοτήσεων, την άδεια που είναι καλύτερη για το χρήστη με την έννοια η εκτέλεση του να έχει μεγαλύτερη προτεραιότητα. Για παράδειγμα η άδεια "Exclusive Execute" είναι

προτιμότερη από την "Execute". Οπότε στην περίπτωση που επιλεχθούν δυο ή περισσότερες εξουσιοδοτήσεις με την άδεια "Exclusive Execute", διαλέγουμε την εξουσιοδότηση που χρειάζεται τις λιγότερες συναλλακτικές μονάδες.

Τέλος μπορούμε να παρατηρήσουμε ότι ο χρήστης "user=**Consultant_b,** ou=HR Wallingford Group Ltd, ou=uk" δεν μπορεί να πάρει καμία εξουσιοδότηση για την διεργασία "F". Σε αυτή την περίπτωση πρέπει να του προταθεί από το σύστημα κάποιος ρόλος που θα μπορεί να εκτελέσει την εν λόγω διεργασία. Εάν επιθυμεί ο χρήστης μπορεί να αναθέσει την εκτέλεση της διεργασίας σε έναν άλλο χρήστη που ανήκει στον προτεινόμενο ρόλο. Οι αλγόριθμοι για το πως μπορεί να γίνει μια τέτοια πρόταση παρουσιάζονται στο επόμενο κεφάλαιο.

3.6 Αλγόριθμοι Ελέγχου και Προτεινομένων Ρόλων

Μια από τις λειτουργικότητες που ένας μηχανισμός εξουσιοδότησης πρέπει να προσφέρει είναι ο έλεγχος του κατά πόσο ένας χρήστης μπορεί να εκτελέσει μια ολόκληρη ροή εργασιών.

Για να μπορεί ένας χρήστης να εκτελέσει ένα συγκεκριμένο μονοπάτι διεργασιών σε μια ροή εργασίας, θα πρέπει να έχει τις απαραίτητες εξουσιοδοτήσεις για να εκτελέσει όλες τις διεργασίες στο μονοπάτι. Δηλαδή:

- Έστω P ένα κατευθυνόμενο μονοπάτι στο γράφο G, τέτοιο ώστε P = {n1,n2, …, nk} , (ni, ni+1) ∈ F για i = 1, 2, …, k-1
- ∀ n ∈ P: ∃ AC(n,r,p,c) ∈ ACs(n) τέτοιο ώστε r ∈ extended_roles(u) != Ø ∧ c ≤ credits(u)

Αξίζει να σημειώσουμε όμως ότι σε μία ροή εργασίας δεν είναι όλα τα μονοπάτια διεργασιών υποχρεωτικά για την εκτέλεση της. Αυτό συμβαίνει όταν η ροή εργασίας χρησιμοποιεί κόμβους ελέγχου ροής (control flow) με επιλογές που βασίζονται σε τιμές. Σε αυτή την περίπτωση, ο χρήστης μπορεί να καταφέρει να εκτελέσει σωστά την ροή εργασίας ενώ μερικές εξουσιοδοτήσεις διεργασιών σε μονοπάτια με κόμβους επιλογής να μην του έδιναν άδειες εκτέλεσης. Ο χρήστης μπορεί να εκτελέσει μόνο το μονοπάτι που είχε τις κατάλληλες εξουσιοδοτήσεις. Όταν μια διεργασία έχει προαιρετικά μονοπάτια, είναι σχετικά δύσκολο να εκφέρουμε άποψη για τον αν ο χρήστης μπορεί να εκτελέσει την ροή εργασίας ή όχι. Η επιλογή των μονοπατιών που θα εκτελεστούν γίνεται δυναμικά κατά την εκτέλεση της ροής εργασίας και δεν μπορούμε να γνωρίζουμε από πριν ποια

μονοπάτια θα εκτελεστούν. Αν έχουμε στατιστικά στοιχεία από προηγούμενες εκτελέσεις της ροής, μπορούμε να υποθέσουμε ποια από τα μονοπάτια με κόμβους επιλογής θα εκτελεστούν.

Για τον λόγο αυτό, οι αλγόριθμοι που θα παρουσιάσουμε εδώ είναι ικανοί να μας επιστρέφουν τα εξής αποτελέσματα: (α) "True" όταν ο αλγόριθμος γνωρίζει ότι είναι εξουσιοδοτημένος για όλα τα πιθανά μονοπάτια, (β) "False" όταν ο αλγόριθμος γνωρίζει ότι σίγουρα δεν είναι δυνατή η εκτέλεση του αλγορίθμου και (γ) "Maybe" όταν δεν γνωρίζει αν μπορεί να το εκτελέσει ή όχι. Στους αλγορίθμους που παρουσιάζουμε θα θεωρήσουμε ότι μια ροή εργασίας παριστάνεται ως μια λίστα από λίστες κόμβων. Όταν ο κόμβος είναι του τύπου "Task", τότε περιέχει τις εξουσιοδοτήσεις του κόμβου αυτού. Όταν ο κόμβος είναι του τύπου "Control Flow", τότε περιέχει τις λίστες των μονοπατιών που ορίζει. Η αναπαράσταση του παραδείγματος σε αυτή την μορφή θα είναι:

- `[Task(ACs(A)), Task(ACs(B)), Task(ACs(C)), Parallel([ [ Task(ACs(D)), Task(ACs(E)), Task(ACs(F))], [Task(ACs(G))] ]), Task(ACs(H))]`

Οπότε ο γενικός αλγόριθμος είναι ο εξής:

```
def authorization(wf, user, ac_path, suggestions):
   result = true
   for i in wf:
          if nodeType(i) is "Control Flow":
                 #Καλείται η κατάλληλη συνάρτηση ανάλογα με τον τύπο του κόμβου που
                 #βρισκόμαστε.
                 if controlFlowType(i) is "Parallel":
                        temp_result    =    authorization_Parallel(i,    user,    ac_path,
suggestions)
                 if controlFlowType(i) is "Choice":
                        temp_result = authorization_Choice(i, user, ac_path, suggestions)
                 if controlFlowType(i) is "While":
                        temp_result = authorization_while(i, user, ac_path, suggestions)
                 #Έλεγχος της τιμής που επιστράφηκε από το κάλεσμα της συνάρτησης και
                 #ανάθεση της στην μεταβλητή επιστροφής. Αν είναι false η μεταβλητή
                 #επιστροφής και η τιμή που επιστράφηκε είναι maybe ή true δεν γίνεται
                 #η ανάθεση. Αν η μεταβλητή επιστροφής είναι maybe τότε δεν γίνεται η
                 #ανάθεση αν η τιμή που επιστράφηκε είναι true.
                 if result == true:
                        result = temp_result:
                 if result == Maybe and temp_result != true:
                        result = temp_result:
          if nodeType(i) is "Task":
                 applicable_ac = []
                 #Προσθέτουμε στον πίνακα applicable_ac, όλες τις εξουσιοδοτήσεις που
                 #μπορούν να εφαρμοστούν στον χρήστη.
                 for ac in ACs(i):
                        if (ac['roles'] in extended_roles(user))
                            and (ac['credits'] <= credits(user)):
                               applicable_ac.append(ac)
                 #Αν ο πίνακας με τις πιθανές εξουσιοδοτήσεις έχει μέγεθος 0 (άδειος)
                 #ο χρήστης δεν μπορεί να εκτελέσει την διεργασία, και πρέπει να βάλουμε
                 #τις εξουσιοδοτήσεις αυτές στον πίνακα των προτάσεων
                 if len(applicable_ac) == 0:
                        suggestions.append(ac)
                        result = false
                 #Αν ο πίνακας με τις πιθανές εξουσιοδοτήσεις έχει μέγεθος 1, τότε
                 #επιλέγεται αυτή ως η εξουσιοδότηση του χρήστη
                 if len(applicable_ac) == 1:
                        ac_path.append(applicable_ac[0])
                 #Αν ο πίνακας με τις πιθανές εξουσιοδοτήσεις έχει μέγεθος παραπάνω από
                 #1, καλείται η συνάρτηση πολιτικής για να επιλυθεί το ζήτημα.
                 if len(applicable_ac) > 1:
                        choose_ac(applicable_ac, ac_path)
   if (result != false):
          suggestions_roles = minimize_suggestions(suggestions)
   return result
```

Αλγόριθμος 3-1 Έλεγχος Εξουσιοδότησης

Παρουσιάζουμε εδώ την συνάρτηση που εκτελείται πάνω στην ροή εργασίας (wf) και παίρνει σαν ορίσματα τον χρήστη (user), και δύο πίνακες που στον πρώτο επιστρέφονται οι επιλεγμένες εξουσιοδοτήσεις ενώ στον δεύτερο οι ρόλοι που χρειάζεται ο χρήστης, για να τους αναθέσει τις διεργασίες που δεν μπορεί να εκτελέσει. Η συνάρτηση επιστρέφει "True" αν ο χρήστης μπορεί να εκτελέσει την ροή, "False" αν δεν μπορεί να την εκτελέσει και "Maybe" αν δεν μπορεί να εκφέρει άποψη.

Η λογική του αλγορίθμου είναι απλή: (α) αν ο κόμβος είναι τύπου "Task", τότε ελέγχει την ύπαρξη εξουσιοδότησης που μπορεί να εφαρμοστεί στους ρόλους του χρήστη καθώς επίσης ελέγχει αν ο χρήστης έχει και τις κατάλληλες συναλλακτικές μονάδες. Αν υπάρχει μόνο μία εξουσιοδότηση, τότε αυτή επιλέγεται και τοποθετείται στον πίνακα με τις επιλεγμένες εξουσιοδοτήσεις. Αν υπάρχουν

περισσότερες από μια, τότε καλείται η συνάρτηση επιλογής πολιτικής που θα αναλύσουμε στον Αλγόριθμος 3-5. Στην περίπτωση που δεν υπάρχει εξουσιοδότηση τότε οι ρόλοι των εξουσιοδοτήσεων γράφονται στον πίνακα των προτεινόμενων λύσεων. Η επιλογή του κατάλληλου ρόλου θα αναλυθεί στον Αλγόριθμος 3-6.

(β) Αν ο κόμβος είναι του τύπου "Control Flow", τότε έχουμε τις ακόλουθες περιπτώσεις. Σε περίπτωση που είναι τύπου "Parallel" καλείται η συνάρτηση authorization_Parallel, αν είναι τύπου "Choice" καλείται η συνάρτηση authorization_choice και τέλος αν είναι του τύπου "While", τότε καλείται η συνάρτηση authorization_while.

```
def authorization_Parallel(control_flow, user, ac_path, suggestions):
	result = true
	for routes in control_flow.getSubFlows():
		temp_result = authorization(routes, user, ac_path, suggestions)
		#Έλεγχος της τιμής που επιστράφηκε από το κάλεσμα της συνάρτησης και
		#ανάθεση της στην μεταβλητή επιστροφής. Αν είναι false η μεταβλητή
		#επιστροφής και η τιμή που επιστράφηκε είναι maybe ή true δεν γίνεται
		#η ανάθεση. Αν η μεταβλητή επιστροφής είναι maybe τότε δεν γίνεται η
		#ανάθεση αν η τιμή που επιστράφηκε είναι true.
		if result == true:
			result = temp_result:
		if result == Maybe and temp_result != true:
			result = temp_result:
	return result
```

Αλγόριθμος 3-2 Έλεγχος Εξουσιοδότησης σε περίπτωση κόμβου τύπου Parallel

Ο Αλγόριθμος 3-2 ελέγχει τις ροές που ορίζονται μετά από έναν κόμβο τύπου "Parallel" χρησιμοποιώντας αναδρομικά τον Αλγόριθμος 3-1. Πιο συγκεκριμένα αν μια ή περισσότερες επιστρέψουν "false", η συνάρτηση επιστρέφει "false", αν μία ή περισσότερες επιστρέψουν "Maybe" και καμία άλλη δεν έχει επιστρέψει "false", επιστρέφει "Maybe". Τέλος αν όλες οι κλήσεις επιτρέψουν "true", τότε και η συνάρτηση επιστρέφει "true".

```
def authorization_choice(control_flow, user, ac_path, suggestions):
	result = true
	routes_true  = 0
	routes_false = 0
	routes_no = len(control_flow.getSubFlows())
	for routes in control_flow.getSubFlows():
		temp_result = authorization(routes, user, ac_path, suggestions)
		if temp_result == false:
			routes_false = routes_false + 1
		elif temp_result == true:
			routes_true = routes_true + 1
	#Έλεγχος για το αν όλες οι διαδρομές είναι true ή false.
	if routes_true == routes_no:
		result = true
	elif routes_false == routes_no:
		result = false
	else:
		result = Maybe
	return result
```

Αλγόριθμος 3-3 Έλεγχος Εξουσιοδότησης σε περίπτωση κόμβου τύπου Choice

Ο Αλγόριθμος 3-3 ελέγχει τις ροές που ορίζονται μετά από έναν κόμβο "Choice" χρησιμοποιώντας αναδρομικά τον Αλγόριθμος 3-1. Για κάθε μια ροή

εκτελείται η συνάρτηση authorization που είναι ορισμένη. Εάν όλες οι εκτελέσεις επιστρέψουν “true”, τότε και η συνάρτηση επιστρέφει “true”. Εάν όλες οι εκτελέσεις επιστρέψουν “false”, τότε και η συνάρτηση επιστρέφει “false”. Σε κάθε άλλη περίπτωση επιστρέφει “Maybe”. Αυτό γίνεται επειδή δεν είναι δυνατόν να γνωρίζουμε από πριν αν θα εκτελεστούν ή όχι τα μονοπάτια που ορίζονται μετά από ένα κόμβο τύπου “Choice”. Η συνάρτηση αυτή είναι που επιστρέφει τον τύπο “Maybe”. Όλες οι υπόλοιπες απλά τον προωθούν στις απαντήσεις τους.

```
def authorization_while(control_flow, user, ac_path, suggestions):
	result = true
	for routes in control_flow.getSubFlow():
		#Αυτό εκτελείται μόνο μια φόρα, επειδή επιστρέφεται μια λίστα με
		#αντικείμενο.
		temp_result = authorization(routes, user, ac_path, suggestions)
	if global.credits_type == ‘Money’:
		if result == true:
			result = maybe
	return result
```

Αλγόριθμος 3-4 Έλεγχος Εξουσιοδότησης σε περίπτωση κόμβου τύπου While

Αυτή η συνάρτηση καλείται όταν βρεθεί ένας κόμβος τύπου “While”. Ο Αλγόριθμος 3-4 ελέγχει την ροή που ορίζει ο κόμβος “While” και καλεί αναδρομικά την συνάρτηση authorization όπου και επιστρέφει το αποτέλεσμα της. Μόνο στην περίπτωση που οι συναλλακτικές μονάδες που χρησιμοποιούνται αντιπροσωπεύουν χρηματικά ποσά τότε στην περίπτωση που το αναδρομικό κάλεσμα της authorization επέστρεψε “true”, η συνάρτηση authorization_while γυρνάει ως αποτέλεσμα “maybe”. Αυτό συμβαίνει διότι δεν γνωρίζουμε πόσες φορές θα γίνει η εκτέλεση τον διεργασιών που ορίζεται μέσα στον κόμβο τύπου “While”.

```
def choose_ac(applicable_ac, ac_path):
   selected_ac = applicable_ac[0]
   #Περίπτωση Πολιτικής Ελαχίστων Συναλλακτικών Μονάδων
   if policy == 'Min Credits':
          min_credits = 0
          for cur_ac in applicable_ac:
                 if (cur_ac['credits'] < min_credits):
                        selected_ac = cur_ac
                        min_credits = cur_ac['credits']
                 elif (cur_ac['credits'] == min_credits):
                        if selected_ac['Permission'] == cur_ac['Permission']:
                               if cur_ac['Role'].isAncestor(selected_ac['Permission']):
                                      selected_ac = cur_ac
                                      min_credits = cur_ac['credits']
                        elif selected_ac['Permission'] != cur_ac['Permission']:
                               if cur_ac['Permission'] == 'Exclusive Execute':
                                      selected_ac = cur_ac
                                      min_credits = cur_ac['credits']
   #Περίπτωση Πολιτικής Καλύτερης Αδείας
   elif policy == 'Max Priority':
          for cur_ac in applicable_ac:
                 if selected_ac['Permission'] == cur_ac['Permission']:
                        if cur_ac['Role'].isAncestor(selected_ac['Permission']):
                               selected_ac = cur_ac
                 elif selected_ac['Permission'] != cur_ac['Permission']:
                        if cur_ac['Permission'] == 'Exclusive Execute':
                               selected_ac = cur_ac
   ac_path.append(selected_ac)
   return
```

Αλγόριθμος 3-5 Επιλογή Εξουσιοδότησης βάση πολιτικής

Ο παραπάνω αλγόριθμος χρησιμοποιείται για την πολιτική που έχουμε ορίσει, όταν υπάρχουν για τον χρήστη δύο ή περισσότερες πιθανές εξουσιοδοτήσεις σε κάποια διεργασία. Η πολιτική που θα χρησιμοποιηθεί έχει επιβληθεί από το σύστημα. Ο παραπάνω αλγόριθμος βρίσκει τις κατάλληλες εξουσιοδοτήσεις σύμφωνα με την πολιτική που έχει οριστεί. Σε περίπτωση που βρει δυο ή περισσότερες εξουσιοδοτήσεις που είναι ισοδύναμες με την πολιτική που έχει οριστεί (δηλαδή στην περίπτωση των ελάχιστων συναλλακτικών μονάδων υπάρχουν δυο ή περισσότερες εξουσιοδοτήσεις με τις ίδιες συναλλακτικές άδειες, και στην περίπτωση της καλύτερης αδείας υπάρχουν δυο ή περισσότερες εξουσιοδοτήσεις με τις ίδιες άδειες), επιλέγει μεταξύ αυτών των εξουσιοδοτήσεων αυτή που επιλέγεται από την αντίστροφη πολιτική. Για παράδειγμα που επιλεχθούν δύο εξουσιοδοτήσεις με την πολιτική των ελαχίστων συναλλακτικών μονάδων, τότε επιλέγεται από αυτές τις δυο αυτή με την καλύτερη άδεια. Αν και πάλι είναι περισσότερες από μία επιλεγμένες εξουσιοδοτήσεις, τότε επιλέγεται αυτή που χρησιμοποιεί τον ρόλο με την μεγαλύτερη «κυριαρχία».

```
def minimize_suggestions(suggestions):
        role_tree = getARoleHierarchyCopy()
        ac_joined = 0
        ac_joined_role =  role_tree.getRoot()
        #Σημείωση τον κόμβων της ιεραρχίας ρόλων με τα ονόματα των διεργασιών που
        #τους χρειάζονται.
        for cur_node in range(len(suggestions)):
                for cur_ac in suggestions[cur_node]:
                        role_tree.find(cur['role']).appendLabel(cur_node)
                        ac = role_tree.find(cur['role']).getLabelSize()
                        if ac > ac_joined:
                                ac_joined = ac
                                ac_joined_role = cur['role']
                        for sub_role in getDescentants(cur['role']):
                                role_tree.find(sub_role).appendLabel(cur_node)
                                ac = role_tree.find(sub_role).getLabelSize()
                                if ac > ac_joined:
                                        ac_joined = ac
                                        ac_joined_role = sub_role
        labels_nodes = createHashTableFromLabels(role_tree)
        finding = true
        ac_joined_labels = ac_joined_role.getLabel()
        solution_roles = [ac_joined_role]
        #Εύρεση των ρόλων που χρησιμοποιούνται περισσότερο.
        while finding:
                for label in findRemainingLabels(len(suggiestions), ac_joined_labels):
                        if labels_nodes.has_key(label):
                                ac_joined_labels = ac_joined_labels + label
                                result_role = labels_nodes[label][0]
                                for role in labels_nodes[label]:
                                        if label.isAncestor(result_role):
                                                result_role = role
                                solution_roles.append(result_role)
                                if (len_label(ac_joined_labels) == len(suggiestions)):
                                        finding= false
                                break
        return solution_roles
```

Αλγόριθμος 3-6 Δημιουργείας προτάσεων για τις διεργασίες που δεν μπορεί να εκτελέσει ο χρήστης

Ο παραπάνω αλγόριθμος καλείται στην περίπτωση που η συνάρτηση authorization, έχει επιστρέψει ως αποτέλεσμα “Maybe” ή “false”, όπου και υπάρχει μια λίστα από σύνολα ρόλων που μπορούν να εκτελέσουν τις διεργασίες που ο χρήστης δεν μπορεί. Η συνάρτηση αυτή προσπαθεί να μειώσει τις επιλογές των ρόλων, ώστε η πρόταση που θα γίνει στον χρήστη να περιέχει τους δυνατόν λιγότερους ρόλους. Με άλλα λόγια προσπαθεί να βρει τους πιο κοντινούς (με βάση την ιεραρχία) ρόλους που μπορούν να εκτελέσουν όλες τις διεργασίες που δεν μπορεί να εκτελέσει ο χρήστης.

Ο αλγόριθμος λειτουργεί ως εξής:

1. Για κάθε διεργασία που δεν μπορεί να εκτελέσει ο χρήστης και για κάθε ρόλο που μπορεί να εκτελέσει την διεργασία, σημείωσε τον ρόλο αυτό καθώς και όλους τους ρόλους που τον κληρονομούν με το διακριτικό όνομα της διεργασίας που μπορεί να εκτελέσει.
2. Εφόσον έχει γίνει η διαδικασία (1) για όλες τις διεργασίες που δεν μπορεί να εκτελέσει ο χρήστης, προσπαθούμε να βρούμε τον ρόλο που έχει τις περισσότερες ετικέτες από διεργασίες. Για κάθε

διεργασία που δεν μπορεί να εκτελέσει ο επιλεγμένος ρόλος, αναζητούμε τον ρόλο (αρχικά) ή τους ρόλους που μπορούν. Αν τα καταφέρουμε επιστρέφουμε στον χρήστη το σύνολο των ρόλων που επιλέξαμε. Αν όχι επαναλαμβάνουμε την διαδικασία (2) για τον αμέσως επόμενο ρόλο που έχει τις αμέσως περισσότερες ετικέτες από διεργασίες.

Αν εκτελέσουμε τους αλγορίθμους που περιγράψαμε τότε θα έχουμε τα εξής αποτελέσματα.

Για τον χρήστη "user=Programmer_a, ou=CNR, ou=it":

- AC(A, "User", Execute, 0)
- AC(B, "Project Member" Execute, 0)
- AC(C, "User", Execute, 0)
- AC(D, "User", Execute, 0)
- AC(E, "Programmer", Execute, 0)
- AC(F, "Programmer", Execute, 0)
- AC(G, "Test Engineer", Execute, 10)
- AC(H, "User", Execute, 0)

Έτσι παρατηρούμε ότι ο συγκεκριμένος χρήστης μπορεί να εκτελέσει όλη την ροή εργασίας, αφού σε κάθε διεργασία βρίσκουμε μόνο μια εξουσιοδότηση που μπορεί να εφαρμοστεί σ' αυτόν. Η μόνη διεργασία για την οποία είναι απαραίτητες περισσότερες από 10 συναλλακτικές μονάδες είναι η "G". Η συνάρτηση authorization θα επιστρέψει "true".

Παρομοίως για τον χρήστη "user=hargikas, ou=ics ,ou=forth, ou=gr" θα έχουμε δυο διαφορετικά σενάρια ανάλογα με το ποια πολιτική εκτέλεσης έχει διαλέξει:

Λιγότερων συναλλακτικών μονάδων:

- AC(A, "User", Execute, 0)
- AC(B, "Project Member" Execute, 0)
- AC(C, "User", Execute, 0)
- AC(D, "User", Execute, 0)
- AC(E, "Test Engineer", Execute, 10)
- AC(F, "Paying User", Exclusive Execute, 20)
- AC(G, "Test Engineer", Execute, 10)

- AC(H, "User", Execute, 0)

Μεγαλύτερη προτεραιότητα κατά την εκτέλεση:

- AC(A, "User", Execute, 0)
- AC(B, "Paying User" Exclusive Execute, 20)
- AC(C, "Paying User", Exclusive Execute, 10)
- AC(D, "Paying User", Exclusive Execute, 10)
- AC(E, "Paying User", Exclusive Execute, 20)
- AC(F, "Paying User", Exclusive Execute, 20)
- AC(G, "Paying User", Exclusive Execute, 50)
- AC(H, "User", Execute, 0)

Παρατηρούμε ότι ο χρήστης "user=Programmer_b, ou=JNR, ou=European Union, ou=int" δεν μπορεί να εκτελέσει την διεργασία "F", οπότε θα του προταθούν κάποιοι ρόλοι που μπορούν. Αυτοί οι ρόλοι είναι οι "Programmer", "Paying User", "Scientific Supervisor" και για την εκτέλεση θα χρειαστούν 0, 20, 10 συναλλακτικές μονάδες αντίστοιχα. Ο λόγος που στο παράδειγμα μας δεν μειώνουμε τις προτάσεις είναι ότι μόνο μια διεργασία δεν κατάφερε να εκτελεστεί, οπότε η επιλογή της κατάλληλης εξουσιοδότησης θα γίνει από την πολιτική εκτέλεσης που έχει επιλέξει ο χρήστης.

Παρουσιάζει ενδιαφέρον όμως ο "user=**Consultant_b**, ou=HR Wallingford Group Ltd, ou=uk" χρήστης, ο οποίος δεν μπορεί να εκτελέσει τις διεργασίες "E" και "F". Οπότε ανάλογα με την πολιτική υπάρχουν και οι κατάλληλες προτάσεις. Οι προτάσεις γενικά είναι:

Ρόλος	Συναλλακτικές Μονάδες για την διεργασία F	Συναλλακτικές Μονάδες για την διεργασία F
"Programmer"	0	0
"Scientific Supervisor"	10	10
"Paying User"	20	20

Οπότε με την πολιτική των λιγότερων συναλλακτικών μονάδων κατασκευάζεται πρόταση για τον ρόλο "Programmer", ενώ για την πολιτική της μεγαλύτερης προτεραιότητας κατά την εκτέλεση προτείνεται ο ρόλος "Scientific Supervisor".

Η εκτέλεση αυτών των συναρτήσεων γίνεται με κατανεμημένο τρόπο. Το πρώτο μηχάνημα που ασχολείται με την εκτέλεση αυτών τον αλγορίθμων είναι το

μηχάνημα από το οποίο βρέθηκε η ροή εργασίας και ο χρήστης κάνει μια αίτηση εκτέλεσης. Το μηχάνημα αυτό που είναι υπεύθυνο για την εκκίνηση της εκτέλεσης της ροής εργασίας, αναλαμβάνει να κάνει και τον έλεγχο εξουσιοδότησης. Οπότε για κάθε διεργασία που βρίσκει μέσα στην ροή εργασίας καλεί την συνάρτηση που περιγράφεται από τον Αλγόριθμος 3-7 στο μηχάνημα που είναι υπεύθυνο για τις εξουσιοδοτήσεις του οργανισμού που θα εκτελεστεί η διεργασία. Έτσι γνωρίζει όλες τις συσχετίσεις χρήστη – ρόλου καθώς και τις εξουσιοδοτήσεις που έχουν οριστεί. Περισσότερα για την κατανεμημένη λειτουργία θα περιγραφτούν στο επόμενο κεφάλαιο.

```
def remote_check_task(user, i):
  applicable_ac = []
  suggestions = []
  #Προσθέτουμε στον πίνακα applicable_ac, όλες τις εξουσιοδοτήσεις που
  #μπορούν να εφαρμοστούν στον χρήστη.
  for ac in ACs(i):
  if (ac['roles'] in extended_roles(user)) and (ac['credits'] <= credits(user)):
         applicable_ac.append(ac)
  #Αν ο πίνακας με τις πιθανές εξουσιοδοτήσεις έχει μέγεθος 0 (άδειος)
  #ο χρήστης δεν μπορεί να εκτελέσει την διεργασία, και πρέπει να βάλουμε
  #τις εξουσιοδοτήσεις αυτές στον πίνακα τον προτάσεων
  if len(applicable_ac) == 0:
         suggestions.append(ac)
         result = false
  #Αν ο πίνακας με τις πιθανές εξουσιοδοτήσεις έχει μέγεθος 1, τότε
  #επιλέγεται αυτή ως η εξουσιοδότηση του χρήστη
  if len(applicable_ac) == 1:
         ac_path.append(applicable_ac[0])
  #Αν ο πίνακας με τις πιθανές εξουσιοδοτήσεις έχει μέγεθος παραπάνω από
  #1, καλείται η συνάρτηση πολιτικής για να επιλυθεί το ζήτημα.
  if len(applicable_ac) > 1:
         choose_ac(applicable_ac, ac_path)
  return (applicable_ac, suggestions)
```

Αλγόριθμος 3-7 Ελέγχου Πρόσβασης μίας διεργασίας

3.7 Περιγραφή των ροών εργασίας και των εξουσιοδοτήσεων

Η ροές επιστημονικών εργασιών επειδή είναι καλά δομημένες (well structured) μπορούν να περιγραφούν με διάφορους τρόπους. Ένας πρότυπος τρόπος περιγραφής τους είναι μέσω της γλώσσας XRL (eXchangeable Routing Language) που επιτρέπει την αναπαράσταση των στιγμιότυπων ροών εργασιών που μπορούν να εκτελούνται σε διάφορους οργανισμούς. Το σύστημα ARION στηρίχτηκε σε μια γλώσσα περιγραφής ροών εργασιών (XASC: XML based active service composition) που περιγράφεται στο παράρτημα Α και είναι παραπλήσια της XRL. Η γλώσσα αυτή ορίζει τις διεργασίες ως οντότητες μέσα στην περιγραφή της και ταυτόχρονα ορίζει τον τρόπο που έχουν συνδυαστεί οι διεργασίες αυτές. Στο παράρτημα Β, φαίνεται πως το παράδειγμα που περιγράψαμε στα προηγούμενα κεφάλαια μπορεί να οριστεί μέσω της γλώσσας αυτής.

Θα παρουσιάσουμε μια γλώσσα εξουσιοδοτήσεων βασισμένη στην XACL, που θα είναι ικανή να ορίσει εξουσιοδοτήσεις για ροές επιστημονικών εργασιών. Κάθε οργανισμός που διαθέτει διεργασίες επιστημονικών υπολογισμών ορίζει τις δικές του εξουσιοδοτήσεις για τις διεργασίες αυτές. Επειδή η XACL ορίζει τέσσερις βασικές πράξεις (ανάγνωση, εγγραφή, δημιουργία και διαγραφή) στο πεδίο "action" για τις εξουσιοδοτήσεις αντικειμένου, θα θεωρήσουμε δύο ακόμη βασικές άδειες που μπορούν να χρησιμοποιηθούν για ροές επιστημονικών εργασιών. Οι άδειες αυτές είναι η "exclusive" και "execute". Τέλος τα πεδία: συνθήκης (condition), χρήστη (subject), αντικειμένου (object) είναι τα πεδία που χρειαζόμαστε από την XACL. Στο πεδίο "subject" ορίζουμε το ρόλο στον οποίο απευθύνεται η εξουσιοδότηση. Στο πεδίο "object" ορίζουμε σε ποία διεργασία απευθύνεται η συγκεκριμένη εξουσιοδότηση. Το πεδίο συνθήκης το χρειαζόμαστε για να ορίζουμε στο πεδίο αυτό τις συνθήκες για τις συναλλακτικές μονάδες. Με αυτό τον τρόπο κρατάμε όλη την λειτουργικότητα της XACL, αν χρειαστούν εξουσιοδοτήσεις στο αντικείμενο της ροής επιστημονικών εργασιών, και έχοντας προσθέσει το επόμενο πεδίο, δίνουμε την δυνατότητα για εξουσιοδοτήσεις εκτέλεσης.

action:

```
<!ELEMENT  action                    (provisional_action*)>
<!ATTLIST  action name               (read|write|create|delete|execute|exclusive)
#REQUIRED permission     (grant|deny)>
<!ELEMENT  provisional_action  (parameter*)>
<!ATTLIST  provisional_action  name CDATA #REQUIRED timing (before|after) "after">
```

Για παράδειγμα ας πάρουμε την διεργασία "G" που εκτελείται στον οργανισμό JNR. Έχει τις εξής εξουσιοδοτήσεις:

- ACs(G) = {AC(G, "Programmer", Execute, 0), AC(G, "Test Engineer", Execute, 10), AC(G, "Paying User", Exclusive Execute, 50), AC(G, "Environmental Scientist", Exclusive Execute, 20)}

Οπότε αυτή μπορεί να οριστεί ως εξής σε XACL:

```
<xacl>
    <object href="document(http://www.arion-dl.com/senarios/JRC-
CNR)/workflow/task[@id = 'task4'] "/>
    <!-- AC(G, "Programmer", Execute, 0) -->
    <rule id="rule1">
      <acl>
        <subject>
          <role>Programmer</role>
```

```
    </subject>
    <condition>
     <predicate name=”compare”>
           <parameter> greater_or_equal </parameter>
           <parameter> UserCredits </parameter>
           <parameter> 0 </parameter>
     </predicate>
    </condition>
     <action name="execute" />
  </acl>
</rule>
<!-- AC(G, “Test Engineer”, Execute, 10) -->
<rule id=”rule2”>
  <acl>
    <subject>
      <role>Test Engineer</role>
    </subject>
    <condition>
     <predicate name=”compare”>
           <parameter> greater_or_equal </parameter>
           <parameter> UserCredits </parameter>
           <parameter> 10 </parameter>
     </predicate>
    </condition>
     <action name="execute" />
  </acl>
</rule>
<!-- AC(G, “Paying User”, Exclusive Execute, 50) -->
<rule id=”rule3”>
  <acl>
    <subject>
      <role>Paying User</role>
    </subject>
    <condition>
     <predicate name=”compare”>
           <parameter> greater_or_equal </parameter>
           <parameter> UserCredits </parameter>
           <parameter> 50 </parameter>
     </predicate>
    </condition>
     <action name="exclusive" />
  </acl>
</rule>
<!-- AC(G, “Enviromental Scientist”, Exclusive Execute, 20) -->
<rule id=”rule4”>
  <acl>
    <subject>
      <role>Enviromental Scientist</role>
    </subject>
    <condition>
     <predicate name=”compare”>
```

```
            <parameter> greater_or_equal </parameter>
            <parameter> UserCredits </parameter>
            <parameter> 20 </parameter>
         </predicate>
        </condition>
         <action name="exclusive" />
      </acl>
    </rule>
</xacl>
```

3.8 Συμπεράσματα

Έχοντας ορίσει το μοντέλο εξουσιοδότησης για ροές επιστημονικών εργασιών μπορούμε να παρατηρήσουμε ότι έχουν ικανοποιηθεί οι απαιτήσεις που θέσαμε για τις ροές επιστημονικών εργασιών:

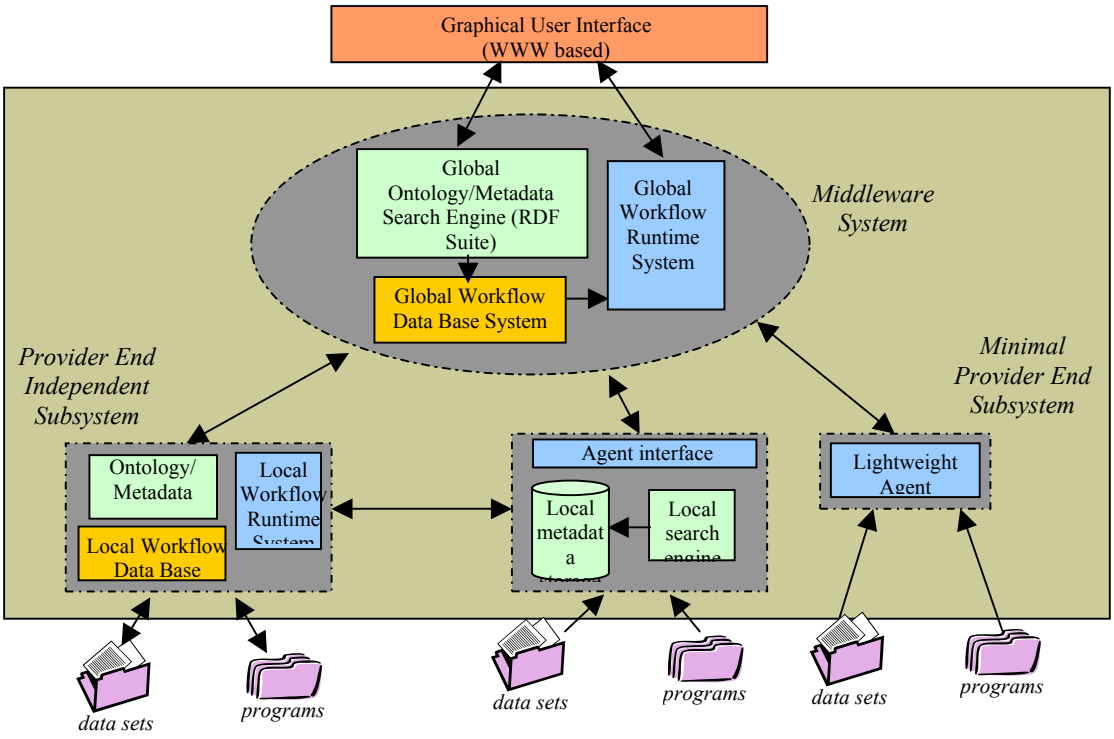
- Η ιεραρχία χρηστών, διευκολύνει τον ορισμό των χρηστών καθώς και την διαχείριση τους. Ο διαχειριστής ενός οργανισμού είναι υπεύθυνος για την διαχείριση των χρηστών που ανήκουν στον ίδιο οργανισμό με αυτόν.
- Έχοντας ορίσει διαφορετικές συσχετίσεις ρόλων – χρηστών ανά οργανισμό, ο κάθε οργανισμός μπορεί να καθορίσει τους ρόλους των χρηστών σύμφωνα με τα δικές του απαιτήσεις.
- Σε κάθε διεργασία επιστημονικού υπολογισμού μπορούν να οριστούν πολλαπλές εξουσιοδοτήσεις για διαφορετικούς ρόλους. Ο τρόπος αυτός αυξάνει την περιγραφική δυνατότητα των εξουσιοδοτήσεων.
- Ορίζονται συναλλακτικές μονάδες όπου μπορούν να χρησιμοποιηθούν για διάφορους τύπους χρεώσεων.
- Δόθηκαν αλγόριθμοι για την επιλογή του κατάλληλου ρόλου, σύμφωνα με την πολιτική που έχει οριστεί, στην περίπτωση που υπάρχουν περισσότερες από μία εξουσιοδοτήσεις.
- Ορίσθηκε αλγόριθμος για την δημιουργία κατάλληλων προτάσεων στην περίπτωση που κάποιος χρήστης δεν μπορεί να εκτελέσει κάποια διεργασία, δηλαδή δεν υπάρχουν κατάλληλες εξουσιοδοτήσεις για τον χρήστη αυτό.

4 Αρχιτεκτονική Έλεγχου Πρόσβασης και Εξουσιοδότησης

Στο κεφάλαιο αυτό θα παρουσιάσουμε την αρχιτεκτονική ενός συστήματος διαχείρισης και εκτέλεσης ροών επιστημονικών εργασιών. Επειδή το αρχικό κίνητρο για την σχεδίαση του μηχανισμού πιστοποίησης και εξουσιοδότησης ήταν το σύστημα ARION [HLCPVPKSG02], η προτεινόμενη αρχιτεκτονική στηρίζεται στην αρχιτεκτονική του συστήματος αυτού. Ακόμη θα παρουσιάσουμε διάφορους μηχανισμούς πιστοποίησης, δηλαδή πως το σύστημα γνωρίζει ότι ο χρήστης είναι πράγματι αυτός που ισχυρίζεται ότι είναι. Το τελευταίο είναι υλοποιημένο στο σύστημα ARION. Τέλος θα δούμε έναν κατανεμημένο μηχανισμό εξουσιοδότησης, δηλαδή πως οι αλγόριθμοι που παρουσιάσαμε στο προηγούμενο κεφάλαιο μπορούν να εκτελεστούν κατανεμημένα. Ο κατανεμημένος μηχανισμός εξουσιοδότησης δεν έχει υλοποιηθεί στην έκταση που αναφερόμαστε στο κεφάλαιο αυτό, αλλά είναι σχεδιασμένος για την εύκολη ενοποίηση του με το σύστημα ARION.

4.1 Συνολική Αρχιτεκτονική Συστήματος Διαχείρισης Ροών επιστημονικών εργασιών

Σε ένα σύστημα διαχείρισης ροών επιστημονικών ροών δεδομένων χρειαζόμαστε για την υλοποίηση του αρκετές συνιστώσες (components). Στην ενότητα αυτή θα αναλύσουμε περιληπτικά μερικές από αυτές τις συνιστώσες. Μερικές από αυτές θα δούμε ότι μας επηρεάζουν και για τις αρχιτεκτονικές πιστοποίησης και εξουσιοδότησης που προτείνουμε στις επόμενες ενότητες.



Εικόνα 4-1 Συνολική Αρχιτεκτονική Συστήματος Διαχείρισης Ροών επιστημονικών εργασιών

Στην Εικόνα 4-1 βλέπουμε πως αλληλεπιδρούν τα διάφορα υποσυστήματα του ARION. Ακόμη μπορούμε να δούμε από ποιες συνιστώσες αποτελείται το κάθε υποσύστημα. Η πρώτη συνιστώσα με την οποία αλληλεπιδρά ένας χρήστης είναι η «Γραφική Διεπαφή Χρήσης» (Graphical User Interface). Αυτή η συνιστώσα αποτελείται από κάποιες δυναμικές ιστοσελίδες που παρουσιάζουν με γραφικό τρόπο τις λειτουργίες του συστήματος στον χρήστη. Σε αυτό το σημείο ο χρήστης μπορεί να διαλέξει ποια δεδομένα θέλει να αποκτήσει ή ποιες ροές θέλει να εκτελέσει. Η γραφική διασύνδεση αυτή εκτελείται σε μερικά μόνο μηχανήματα (δεν χρειάζεται να είναι σε όλα τα μηχανήματα που ανήκουν στο σύστημα ARION). Η συνιστώσα αυτή επικοινωνεί με το υποσύστημα «ενδιάμεσου λογισμικού» (Middleware System), το οποίο απαρτίζεται από τις παρακάτω συνιστώσες:

(α) Οι ορισμοί των μετα-δεδομένων και οι οντολογίες που τα περιγράφουν. Η συνιστώσα αυτή είναι υπεύθυνη για την σωστή περιγραφή των διαφόρων επιστημονικών δεδομένων που είτε υπάρχουν στο σύστημα, είτε παράγονται δυναμικά από προγράμματα. Μέσω της συνιστώσας αυτής μας δίνεται η δυνατότητα για εύρεση των δεδομένων που χρειάζεται ο χρήστης, και αν δεν υπάρχουν να βρεθεί κάποια ροή επιστημονικής εργασίας που να τα παράγει.

(β) Η βάση δεδομένων για τις ροές επιστημονικών εργασιών. Η συνιστώσα αυτή είναι το μέρος της αποθήκευσης των ροών επιστημονικών εργασιών, ώστε να μπορεί να γίνεται η κατάλληλη αναζήτηση πάνω στα δεδομένα που παράγουν.

(γ) Το υποσύστημα εκτέλεσης των ροών επιστημονικών εργασιών. Στο σύστημα αυτό δημιουργείται ένα στιγμιότυπο μιας ροής δεδομένων, και δημιουργούνται ταυτόχρονα και οι οδηγίες για την εκτέλεση της μέσω πρακτόρων από το σύστημα που θα περιγραφεί παρακάτω.

Το τελευταίο υποσύστημα είναι αυτό των κινούμενων πρακτόρων. Το σύστημα αυτό προσφέρει την δυνατότητα εκτέλεσης μια διεργασίας επιστημονικών υπολογισμών σε ένα απομακρυσμένο μηχάνημα, της μεταφοράς των δεδομένων που χρειάζονται η διεργασία σε μία καθορισμένη τοποθεσία, της αποθήκευσης στατιστικών για την εκτέλεση των διεργασιών και τέλος την δυνατότητα τερματισμού ή αναμονής μερικών διεργασιών.

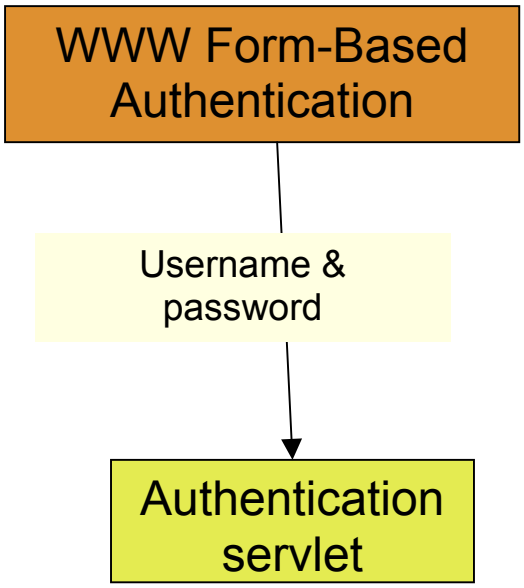
Το υποσύστημα ενδιάμεσου λογισμικού και το σύστημα διαχείρισης κινούμενων πρακτόρων μπορεί να είναι εγκατεστημένα πολλές φορές σε διαφορετικούς οργανισμούς. Υπάρχουν περιπτώσεις οργανισμών που έχουν και τα δυο αυτά υποσυστήματα και προσφέρουν λειτουργία αναζήτησης δεδομένων καθώς επίσης και εκτέλεσης ροών εργασιών, αλλά υπάρχουν επίσης περιπτώσεις οργανισμών που έχουν μόνο το σύστημα πρακτόρων για την διαχείριση των προγραμμάτων που προσφέρονται.

Για να μπορέσουμε να σχεδιάσουμε την αρχιτεκτονική ελέγχου πιστοποίησης και εξουσιοδότησης για ένα τέτοιο μηχανισμό, θα ασχοληθούμε με τις συνιστώσες: (α) Γραφική Διεπαφή Χρήσης (β) Σύστημα κινούμενων πρακτόρων.

4.2 Προτεινόμενες Αρχιτεκτονικές για Πιστοποίηση

Επειδή ο χρήστης αρχικά βλέπει μόνο τις δυναμικά παραγόμενες “web” σελίδες, η πιστοποίηση του μπορεί να γίνει μόνο σε αυτό το σημείο. Δηλαδή θα υπάρχει μια δυναμικά παραγόμενη σελίδα που θα ελέγχει τα διαπιστευτήρια του χρήστη και ανάλογα αν αυτά τον πιστοποιούν θα του επιτρέπεται η πρόσβαση στο υπόλοιπο σύστημα. Ο χρήστης χρησιμοποιεί για την πιστοποίηση του ένα απλό φυλλομετρητή διαδικτύου (web browser). Τα διαπιστευτήρια είναι ένας κωδικός πρόσβασης που γνωρίζει ο χρήστης και το εισάγει στο σύστημα όταν του ζητηθεί. Η μεταφορά του κωδικού πρόσβασης γίνεται πάνω από μία ασφαλή σύνδεση. Σε μια

ασφαλή σύνδεση τα δεδομένα που μεταφέρονται κρυπτογραφούνται από τον αποστολέα και τα αποκρυπτογραφεί ο παραλήπτης. Με τον τρόπο αυτό είναι δύσκολο να «υποκλαπούν» από κάποιους κακόβουλους χρήστες. Χρησιμοποιείται τεχνολογία δημοσίων / προσωπικών κλειδιών για την υλοποίηση της κρυπτογράφησης [MOV96].

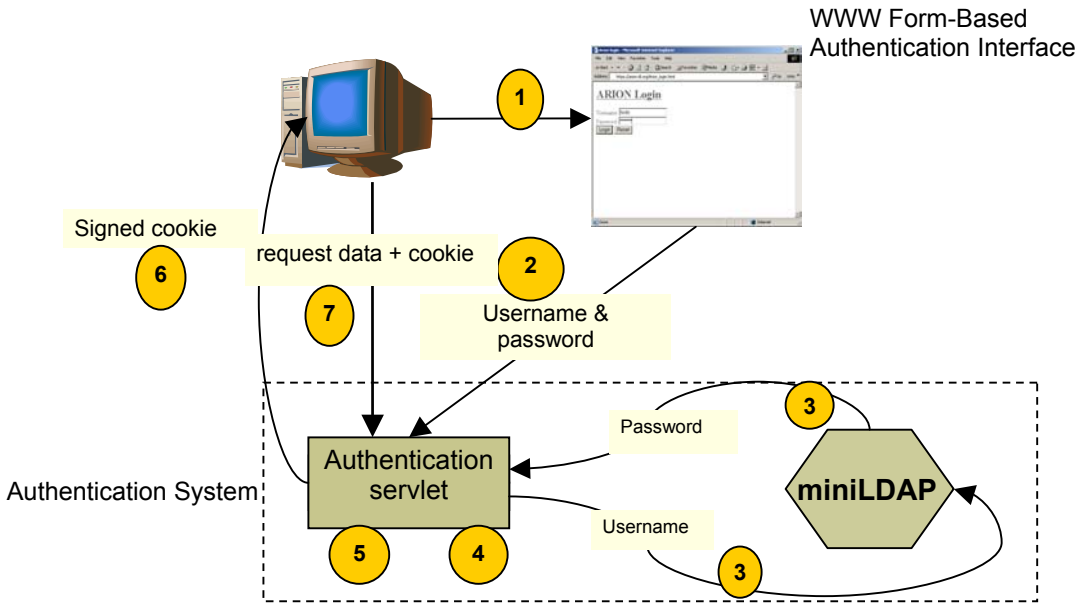


Εικόνα 4-2 Μεταφορά των διεπιστευτηρίων του χρήστη

Εφόσον τα διαπιστευτήρια μεταφερθούν στον εξυπηρετητή δυναμικών σελίδων (όπως φαίνεται και στην Εικόνα 4-2) γίνεται και ο έλεγχος πιστοποίησης. Για να γίνει ο έλεγχος πιστοποίησης πρέπει να βρεθούν τα διαπιστευτήρια του χρήστη που έχουν αποθηκευτεί στο σύστημα. Τα διαπιστευτήρια είναι αποθηκευμένα μαζί με το αντικείμενο που αναπαριστά τον χρήστη στην ιεραρχία χρηστών της εφαρμογής “miniLDAP” (όπως είχε αναφερθεί στο προηγούμενο κεφάλαιο η ιεραρχία χρηστών είναι ένα υποσύνολο από την ιεραρχία που ορίζεται από το LDAP).

Όταν η εφαρμογή επιστρέψει το αντικείμενο που αναπαριστά τον χρήστη, μαζί με τα διαπιστευτήρια του, τότε ο χρήστης ανακατευθύνεται στις υπόλοιπες δυναμικές σελίδες από τις οποίες μπορεί να χειριστεί το σύστημα ARION. Ταυτόχρονα γίνεται και η αίτηση, στον φυλλομετρητή του χρήστη, για την δημιουργία ενός αντικείμενου τύπου “cookie” το οποίο περιέχει ένα κλειδί με τον διακριτικό αριθμό της συνεδρίας (session). Το αντικείμενο συνεδρίας περιγράφει ποιος χρήστης είναι αυτός που έχει πιστοποιηθεί, από ποια διεύθυνση δικτύου (IP) και για πόσο χρονικό διάστημα θα είναι πιστοποιημένος αν δεν κάνει κάποια εργασία (idle). Το κλειδί που έχει σταλεί στον χρήστη είναι υπογεγραμμένο με το

ψηφιακό πιστοποιητικό της εφαρμογής που δημιουργεί τις ψηφιακές ιστοσελίδες. Όπως θα δούμε στην επόμενη ενότητα, το ίδιο πιστοποιητικό χρησιμοποιείται και στην εφαρμογή "miniLDAP". Για κάθε δυναμική σελίδα που θα επισκεφτεί στην συνέχεια ο χρήστης θα του ζητηθεί να παρουσιάσει το αντικείμενο τύπου "cookie" που του δόθηκε πριν. Έτσι σε κάθε σελίδα γίνεται ο έλεγχος για το αν ο χρήστης είναι πιστοποιημένος.



Εικόνα 4-3 Βήματα Πιστοποίησης

Όπως μπορούμε να δούμε και στην Εικόνα 4-3, τα βήματα που χρειάζονται για να πραγματοποιηθεί η πιστοποίηση είναι:

1. Ο χρήστης εισάγει τα στοιχεία του (όνομα χρήστη και τον κωδικό πρόσβασης) σε φόρμα HTML.
2. Τα στοιχεία αυτά περνούν στην επόμενη δυναμική ιστοσελίδα πάνω από ασφαλή σύνδεση (https)
3. Η δυναμική σελίδα αυτή ζητάει από την εφαρμογή "miniLDAP", τα στοιχεία του χρήστη που είναι αποθηκευμένα στο σύστημα (βλέπε αναλυτική περιγραφή στην επόμενη ενότητα).
4. Η δυναμική σελίδα ελέγχει τα στοιχεία που δόθηκαν με αυτά που είναι καταχωρημένα στο σύστημα.

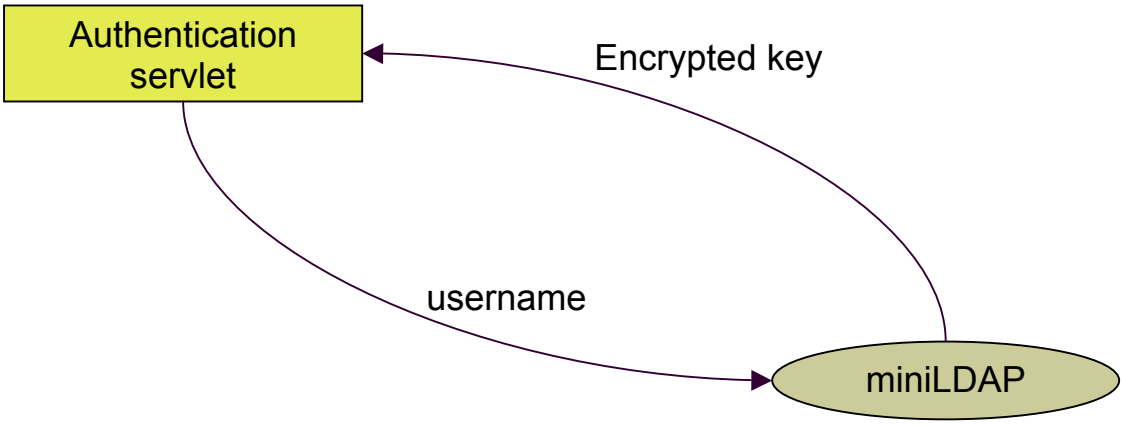
5. Δημιουργείται ένα αντικείμενο τύπου “cookie”, το οποίο είναι υπογεγραμμένο. Το εν λόγω αντικείμενο περιέχει ένα κλειδί για τα δεδομένα της πιστοποίησης του.
6. Το κλειδί αυτό στέλνεται στον φυλλομετρητή του χρήστη.
7. Για κάθε επόμενη δυναμική σελίδα που ζητάει ο χρήστης, πρέπει να παρουσιάζει και το αντικείμενο τύπου “cookie” που του δόθηκε.

Όπως είχαμε αναφέρει και στην προηγούμενη ενότητα, η αρχιτεκτονική της πιστοποίησης είναι ουσιαστικά υλοποιημένη σε συνδυασμό με την συνιστώσα του ARION Γραφική Διεπαφή Χρήσης.

4.2.1 Εφαρμογή miniLDAP

Μια από τις βασικές λειτουργίες του miniLDAP είναι να επιστρέφει διάφορες πληροφορίες που έχουν οριστεί για έναν χρήστη. Οι άμεσες πληροφορίες που υπάρχουν για τον χρήστη είναι:

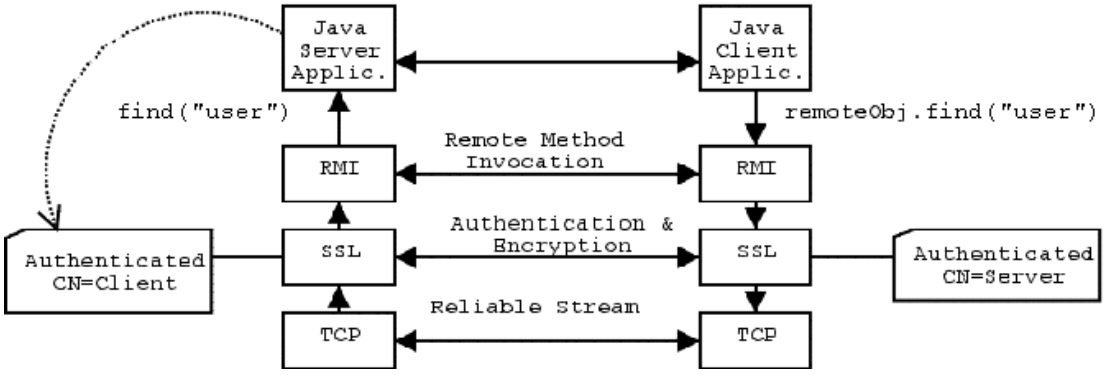
- Το διακριτικό όνομα (id)
- Ο κωδικός πρόσβασης
- Το πραγματικό ονοματεπώνυμο του χρήστη
- Η ηλεκτρονική του διεύθυνση (e-mail)
- Και μια περιγραφή (description), στην οποία τοποθετεί ο διαχειριστής του συστήματος οποιαδήποτε βοηθητική πληροφορία.



Εικόνα 4-4 Επικοινωνία με "miniLDAP"

Η δυναμική “web” σελίδα η οποία έχει πάρει τα διαπιστευτήρια του χρήστη ρωτά την εφαρμογή “miniLDAP” σε σχέση με τον κωδικό πρόσβασης του χρήστη. Κάθε “miniLDAP” εφαρμογή έχει μόνο ένα μέρος από την συνολική ιεραρχία των χρηστών. Πιο συγκεκριμένα, κάθε “miniLDAP” έχει το υπό-δέντρο της ιεραρχίας

χρηστών που ορίζεται στον οργανισμό στον οποίο ανήκει. Στην περίπτωση που ο χρήστης, για τον οποίο ζητείται πληροφορία, ανήκει σε άλλο οργανισμό από αυτό του "miniLDAP", τότε είναι απαραίτητη η επικοινωνία με το κατάλληλο "miniLDAP" του αντίστοιχου οργανισμού. Η επικοινωνία αυτή βασίζεται σε απομακρυσμένες κλήσεις συναρτήσεων (RMI) στην γλώσσα προγραμματισμού "Java". Λόγω αυξημένων απαιτήσεων ασφάλειας χρειάστηκε να υλοποιηθεί η ασφαλής απομακρυσμένη κλήση συναρτήσεων (secure-RMI). Ο τρόπος λειτουργίας της φαίνεται στην Εικόνα 4-5. Ουσιαστικά δημιουργείται μια κρυπτογραφημένη σύνδεση στην οποία περνάνε οι πληροφορίες που χρειάζονται για να γίνει μια απομακρυσμένη κλήση. Ταυτόχρονα για να επιτευχθεί η σύνδεση πρέπει να υπάρχει αμοιβαία εμπιστοσύνη ανάμεσα στα δύο μέρη της σύνδεσης. Οπότε κάθε εφαρμογή που θέλει να επικοινωνήσει μέσω ασφαλούς απομακρυσμένης κλήσης, πρέπει να διαθέτει ένα ψηφιακό πιστοποιητικό έτσι ώστε: (α) να μπορεί, αυτός που θέλει να επικοινωνήσει μαζί της, να κρυπτογραφήσει τα δεδομένα. (β) να εμπιστευτεί όποια εφαρμογή συνδεθεί μαζί της. Η εμπιστοσύνη επιτυγχάνεται χρησιμοποιώντας ψηφιακές υπογραφές: Η εφαρμογή που θέλει να επικοινωνήσει πρέπει να έχει ένα πιστοποιητικό υπογεγραμμένο από κάποια οντότητα που εμπιστεύεται η εφαρμογή στην οποία συνδέεται.

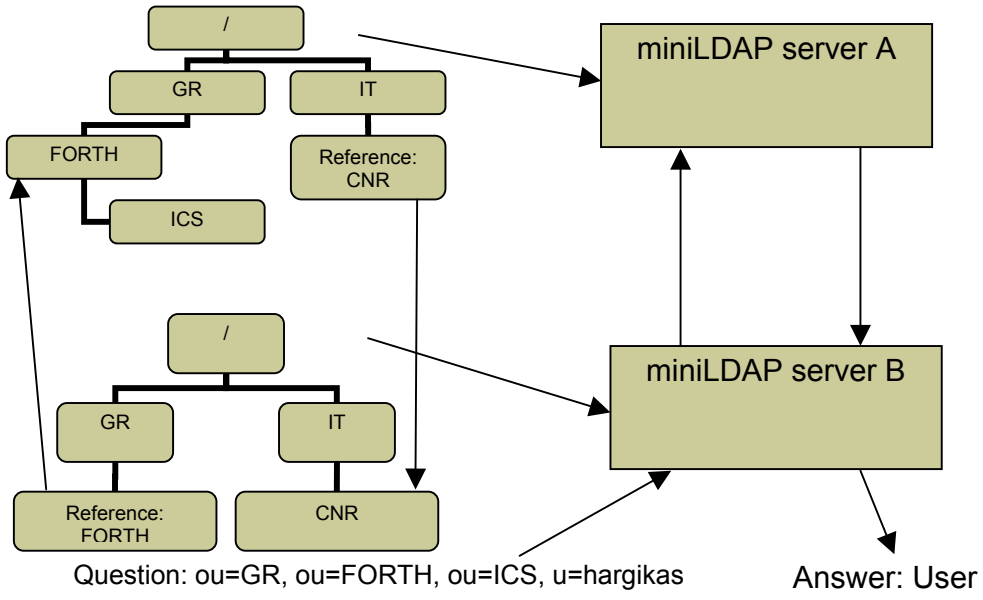


Εικόνα 4-5 Υλοποίση Secure-RMI

Στην συνέχεια θα παρουσιάσουμε πως μια εφαρμογή "miniLDAP" μπορεί να μάθει ποιόν πρέπει να ρωτήσει για να πάρει απάντηση στην αναζήτηση ενός χρήστη.

Κάθε miniLDAP, περιέχει ορισμένα αντικείμενα «ανακατεύθυνσης» στην ιεραρχία των χρηστών. Τα αντικείμενα αυτά «δείχνουν» σε ποία τοποθεσία βρίσκεται η εφαρμογή "miniLDAP", που έχει ορίσει το συγκεκριμένο αυτό υπό-δέντρο της ιεραρχίας. Επομένως, όταν ζητείται από μία εφαρμογή η αναζήτηση ενός

χρήστη, τότε η εφαρμογή διασχίζει την ιεραρχία χρηστών και αν στην πορεία συναντήσει κάποιο αντικείμενο ανακατεύθυνσης τότε μεταφέρει την ερώτηση στην κατάλληλη εφαρμογή “miniLDAP”. Στην Εικόνα 4-6 βλέπουμε μια ερώτηση για τον χρήστη “`u=hargikas, ou=ics, ou=forth, ou=gr`” που γίνεται στην εφαρμογή “miniLDAP”, η οποία βρίσκεται στον οργανισμό CNR. Η εφαρμογή αυτή, στην προσπάθεια εύρεσης του χρήστη, βρίσκει ένα αντικείμενο ανακατεύθυνσης για τον οργανισμό “forth” (ITE) και κάνει την ίδια ερώτηση στην αντίστοιχη εφαρμογή “miniLDAP” του ITE. Επειδή τα δεδομένα που μεταφέρονται δεν είναι αρκετά μεγάλα, μετά από μία ερώτηση η εφαρμογή θα κρατήσει την απάντηση για κάποιο μικρό χρονικό διάστημα στην μνήμη της, ώστε να μπορεί να την απαντήσει απ’ ευθείας αν της ξαναγίνει η ίδια ερώτηση.



Εικόνα 4-6 Ανακατευθύνσεις Ερωτήσεων στην εφαρμογή "miniLDAP"

Με αυτόν τον τρόπο βλέπουμε πως υλοποιείται μια ασφαλής πιστοποίηση, ακόμη και αν τα διαπιστευτήρια βρίσκονται σε κάποια απομακρυσμένη μηχανή. Έτσι κάθε χρήστης μπορεί να αποκτήσει πρόσβαση στο σύστημα από οποιαδήποτε εγκατάσταση του συστήματος ARION.

4.3 Αρχιτεκτονική για Πολιτικές Εξουσιοδότησης

Σε αυτήν την ενότητα θα παρουσιάσουμε μια κατανεμημένη αρχιτεκτονική εκτέλεσης των αλγορίθμων που παρουσιάστηκαν στην ενότητα 3.6 του προηγουμένου κεφαλαίου. Μέσα σε αυτό το πλαίσιο χρειαζόμαστε να

αποφασίσουμε για τον χώρο αποθήκευσης των στοιχείων που αφορούν την εξουσιοδότηση ενός χρήστη δηλαδή για: (α) τις εξουσιοδοτήσεις - διεργασίες, (β) τις συσχετίσεις ρόλων – χρηστών. Ταυτόχρονα, θα δείξουμε ποιες συνιστώσες αναλαμβάνουν τον έλεγχο πρόσβασης και τι γίνεται στην περίπτωση που ένας χρήστης δεν μπορεί να εκτελέσει κάποια διεργασία.

4.3.1 Συσχετίσεις Ρόλων – Χρηστών

Στην προηγούμενη ενότητα περιγράψαμε πως μπορούμε να αποθηκεύσουμε μια ιεραρχία χρηστών και πως μπορεί να γίνει η αναζήτηση κάποιου χρήστη. Μια καλή θεώρηση είναι ότι κάθε οργανισμός που εκτελεί την εφαρμογή "miniLDAP", ουσιαστικά μπορεί να ορίσει στο ίδιο σημείο (στην εφαρμογή) και τις συσχετίσεις των χρηστών με τους ρόλους. Έτσι η ερώτηση σε ποιους ρόλους ανήκει κάποιος χρήστης μπορεί επίσης να υλοποιηθεί από την εφαρμογή "miniLDAP".

Επειδή το πλήθος των χρηστών μπορεί να είναι αρκετά μεγάλο, ο διαχειριστής που ορίζει ρόλους μπορεί να μην είναι σε θέση να ορίσει τις συσχετίσεις των ρόλων με όλους τους χρήστες. Για ευκολία μπορούμε να θεωρήσουμε την ακόλουθη λειτουργικότητα της εφαρμογής "miniLDAP". Όταν οι συσχετίσεις κάποιου χρήστη δεν υπάρχουν σε κάποια εφαρμογή, τότε αυτή ρωτάει τις συσχετίσεις που βρίσκονται στον οργανισμό που είναι άμεσος πρόγονος. Αν δεν βρεθεί ούτε στον άμεσο πρόγονο τότε εκτελείται αναδρομικά μέχρι να φτάσουμε στην ρίζα. Στην ρίζα (που δεν είναι κάποιος πραγματικός οργανισμός) μπορούμε να θεωρήσουμε ότι κάθε χρήστης ανήκει στον βασικότερο ρόλο. Στο παράδειγμα που είχαμε αναπτύξει στο προηγούμενο κεφάλαιο, στη ρίζα των οργανισμών ("/") θα είχαμε ορίσει όλους τους χρήστες να ανήκουν στο ρόλο "Users". Με τον μηχανισμό αυτό αυξάνουμε την πολυπλοκότητα για την εύρεση του ρόλου κάποιου χρήστη, αλλά διευκολύνουμε την οργάνωση των συσχετίσεων. Στην χείριστη περίπτωση, θα χρειαστούν τόσες ερωτήσεις όσες και το βάθος, στην ιεραρχία χρηστών, που βρίσκεται ο οργανισμός.

4.3.2 Εξουσιοδοτήσεις

Στο προηγούμενο κεφάλαιο είδαμε ένα τρόπο περιγραφής των εξουσιοδοτήσεων που είναι βασισμένος στο πρότυπο XML. Οι περιγραφές αυτές πρέπει να βρίσκονται στον ίδιο οργανισμό με τις διεργασίες τις οποίες εξουσιοδοτούν.

Η επιβολή των εξουσιοδοτήσεων γίνεται από την συνιστώσα εκτέλεσης ροών εργασίας (Workflow Runtime System), η οποία είναι υπεύθυνη και για την έκδοση οδηγιών για την εκτέλεση της ροής επιστημονικών εργασιών στην συνιστώσα την κινούμενων πρακτόρων. Κάθε φορά που κάποιος χρήστης έχει ορίσει ποιος ρόλος θα εκτελέσει την διεργασία, η συνιστώσα εκτέλεσης ροών εργασίας βρίσκει ποια εξουσιοδότηση είναι αυτή που επιβάλλετε στον ρόλο αυτό και μπορεί πλέον να ορίσει πως θα εκτελεστεί η διεργασία. Μετά την επιτυχή εκτέλεση της διεργασίας, αν οι συναλλακτικές μονάδες αντιπροσωπεύουν χρήματα, η προηγούμενη συνιστώσα είναι υπεύθυνη για την σωστή διαχείριση των συναλλακτικών μονάδων του χρήστη. Όμως οι ορισμοί των εξουσιοδοτήσεων μπορεί να βρίσκονται σε διαφορετικό μέρος (π.χ. μια βάση δεδομένων) μέσα στον ίδιο οργανισμό. Έτσι ο κάθε διαχειριστής μπορεί να έχει μια πλήρη άποψη για τις εξουσιοδοτήσεις που έχει ορίσει αλλά δεν είναι απαραίτητο να έχει την πλήρη άποψη για τις εξουσιοδοτήσεις που έχουν οριστεί σε άλλους οργανισμούς. Οι ορισμοί των εξουσιοδοτήσεων είναι σε διαφορετικό μέρος από εκεί που γίνεται ο έλεγχος τους.

4.3.3 Εκτέλεση αλγορίθμων σε ένα κατανεμημένο σύστημα

Αφού ο οργανισμός που προσφέρει κάποια διεργασία στο σύστημα έχει την (α) συσχέτιση χρηστών – ρόλων, και (β) τους ορισμούς των εξουσιοδοτήσεων της διεργασίας επιστημονικών υπολογισμών, μπορεί να εκτελέσει τον Αλγόριθμος 3-7 (με παραμέτρους τον χρήστη και την διεργασία). Η συνιστώσα (component) που εκτελεί τον αλγόριθμο αυτό ονομάζεται, «Απόφαση Πολιτικής Πρόσβασης» (Access Policy Decision) και επιστρέφει είτε τις εξουσιοδοτήσεις που μπορεί να επιβληθούν σε ένα χρήστη είτε τους ρόλους που πρέπει να εκτελέσουν την εφαρμογή εκ μέρους του (αν φυσικά δεν υπάρχει εξουσιοδότηση που να εφαρμόζεται για τους ρόλους που έχουν ήδη οριστεί).

Όταν ο χρήστης συνδέεται με κάποια δυναμική σελίδα του συστήματος (π.χ. τη συνιστώσα «Γραφική Διασύνδεση Χρήσης»), και μέσω αυτής βρίσκει κάποια ροή επιστημονικών εργασιών να εκτελέσει, τότε πρέπει σε αυτό το σημείο να γνωρίζει αν μπορεί να εκτελέσει την ροή η όχι. Ο οργανισμός στον οποίο έχει συνδεθεί ο χρήστης για να βρει την ροή εργασίας που πρόκειται να εκτελέσει, ξεκινά και τον έλεγχο εξουσιοδότησης. Έχοντας όλη την περιγραφή της συγκεκριμένης ροής εργασίας, γνωρίζει σε ποιους οργανισμούς πρόκειται να

εκτελεστούν οι διάφορες διεργασίες επιστημονικών υπολογισμών. Διασχίζει, λοιπόν, όλα τα μονοπάτια της ροής (βλέπε Αλγόριθμος 3-1) και για κάθε διεργασία που συναντά συνδέεται με την συνιστώσα «Απόφαση Πολιτικής Πρόσβασης» και ζητάει να εκτελέσει τον Αλγόριθμος 3-7 για τον χρήστη και για την διεργασία που συνάντησε. Η διαδικασία αυτή επαναλαμβάνεται για κάθε διεργασία ώστε να αποφασιστεί αν ο χρήστης μπορεί να εκτελέσει ή όχι την ροή εργασίας.

4.3.4 Προτάσεις Ρόλων

Εάν ο χρήστης δεν είναι ικανός να εκτελέσει κάποιες από τις διεργασίες μιας ροής, θα έχει σύμφωνα με τον Αλγόριθμος 3-6, κατάλληλες προτάσεις για ποιοι ρόλοι μπορούν να εκτελέσουν τις διεργασίες που αυτός δεν μπορεί. Πρέπει να επισημάνουμε ότι ο προηγούμενος αλγόριθμος εκτελείται ξεχωριστά για κάθε οργανισμό που συμμετέχει στην εκτέλεση της ροής εργασίας και τελικά επιστρέφεται στον χρήστη μια πρόταση η οποία προτείνει τον καλύτερο ρόλο (ή ρόλους) ανά οργανισμό για την εκτέλεση των διεργασιών.

Την στιγμή που έχουν προταθεί οι ρόλοι που είναι ικανοί για την εκτέλεση των διεργασιών, υπάρχουν δύο περιπτώσεις για να καταφέρει ο χρήστης να εκτελέσει την ροή εργασίας: (α) Ανατίθεται οι ρόλοι που έχουν προταθεί στον χρήστη, ώστε να καταφέρει να εκτελέσει επιτυχώς την ροή εργασίας. (β) Βρίσκεται κάποιος άλλος χρήστης που ανήκει στους προτεινόμενους ρόλους και αναλαμβάνει να εκτελέσει τις διεργασίες στην θέση του.

Για την δεύτερη περίπτωση, την στιγμή που ο χρήστης επιλέξει τους ρόλους που επιθυμεί, για να εκτελέσουν τις διεργασίες του, το υποσύστημα των κινούμενων πρακτόρων αναλαμβάνει να βρει τους χρήστες που θα τις εκτελέσουν. Ένας πράκτορας, που αργότερα θα εκτελέσει την διεργασία, θα πρέπει να επικοινωνήσει με έναν χρήστη που ανήκει στους επιλεχθέντες από το χρήστη ρόλους. Αν ο χρήστης δεν δεχτεί την συγκεκριμένη εκτέλεση, τότε θα επιλεχτεί κάποιος επόμενος χρήστης. Αυτή η επιλογή απορρέει από την συσχέτιση ρόλου – χρήστη αλλά και από τους αλγόριθμους ισοκατανομής φόρτου εργασίας για τους χρήστες (π.χ. Round – robin). Οι αλγόριθμοι αυτοί δεν θα αναλυθούν σε αυτήν την εργασία.

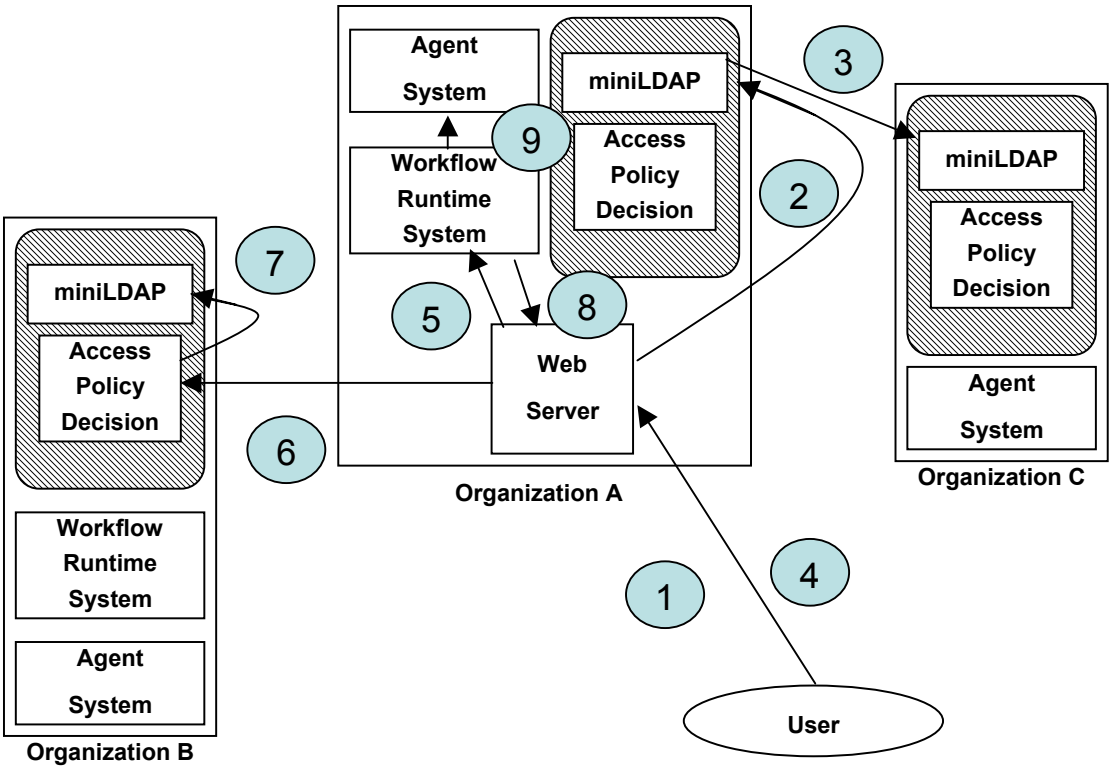
4.4 Αρχιτεκτονική Εκτέλεσης Εξουσιοδοτημένων Διεργασιών

Γνωρίζοντας ποιες εξουσιοδοτήσεις θα χρησιμοποιηθούν για την εκτέλεση των διεργασιών, η συνιστώσα εκτέλεσης ροών εργασίας αναλαμβάνει να δώσει τις κατάλληλες οδηγίες στο σύστημα κινούμενων πρακτόρων. Υπάρχουν περιπτώσεις, όμως, που τα δεδομένα που θα χρησιμοποιηθούν είναι ευαίσθητα. Για να μην δημιουργηθεί πρόβλημα αλλοίωσης από έναν κακόβουλο χρήστη (π.χ. εκτελώντας κάποιο δικό του πρόγραμμα το οποίο καταστρέφει κάποια δεδομένα) οι άδειες του λειτουργικού συστήματος για την πρόσβαση αυτών των δεδομένων είναι χρονικά περιορισμένες.

Οπότε, όταν δίνονται οι οδηγίες εκτέλεσης διεργασίας στο σύστημα κινούμενων πρακτόρων, δίνονται και οι κατάλληλες εντολές στο λειτουργικό σύστημα για την εκχώρηση των κατάλληλων αδειών πρόσβασης στα δεδομένα που χρειάζεται η διεργασία. Την χρονική στιγμή που τερματίζεται η εκτέλεση της διεργασίας, οι άδειες αυτές ανακαλούνται. Με αυτόν τον τρόπο μειώνεται ο αριθμός των δεδομένων που θα μπορούσε ένας κακόβουλος χρήστης να καταστρέψει ή να παραποιήσει.

4.5 Συνολική Αρχιτεκτονική

Στην Εικόνα 4-7 μπορούμε να δούμε την αρχιτεκτονική του συστήματος εξουσιοδότησης που προτείνουμε καθώς και τις επιμέρους συνιστώσες οι οποίες μπορούν να εγκατασταθούν σε διάφορους οργανισμούς.



Εικόνα 4-7 Παράδειγμα Αρχιτεκτονικής

Αν ένας χρήστης θέλει να χρησιμοποιήσει το σύστημα, αναγκαστικά θα συνδεθεί με τον οργανισμό «Α», γιατί αυτός παρέχει μέσω του εξυπηρετητή δυναμικών σελίδων (web server) την πρόσβαση στο σύστημα. Τα βήματα που γίνονται για μια επιτυχή εκτέλεση είναι τα εξής:

1. Ο χρήστης κάνει μια αίτηση πιστοποίησης στον εξυπηρετητή δυναμικών σελίδων.
2. Ο εξυπηρετητής δυναμικών σελίδων κάνει μια ερώτηση στην εφαρμογή "miniLDAP", που βρίσκεται στον οργανισμό αυτό.
3. Αν ο χρήστης ανήκει στον οργανισμό αυτό, επιστρέφει αμέσως τον κωδικό πρόσβασης στον εξυπηρετητή δυναμικών σελίδων. Αν, όμως, ανήκει σε κάποιο άλλο οργανισμό, η εφαρμογή "miniLDAP" μεταφέρει την ερώτηση στην αντίστοιχη εφαρμογή του απομακρυσμένου οργανισμού.
4. Εφόσον ο χρήστης έχει πιστοποιηθεί μπορεί να επισκεφτεί τις υπόλοιπες δυναμικές σελίδες του συστήματος. Από αυτές τις σελίδες κάνει αίτηση για την εκτέλεση μιας ροής επιστημονικών εργασιών.
5. Η αίτηση αυτή παραλαμβάνεται από την συνιστώσα εκτέλεσης ροής εργασιών, και βρίσκεί ποιες διεργασίες χρειάζεται να εκτελεστούν.

6. Για κάθε μία διεργασία, η συνιστώσα εκτέλεσης ροής εργασιών επικοινωνεί με την συνιστώσα της «απόφασης πολιτικής πρόσβασης» (Access Policy Decision) που βρίσκεται στον ίδιο οργανισμό στον οποίο θα εκτελεστεί η διεργασία.
7. Αυτή η συνιστώσα καλεί την εφαρμογή “miniLDAP” που βρίσκεται στον ίδιο οργανισμό για να βρει τους ρόλους στους οποίους ο χρήστης ανήκει. Από τους ρόλους βρίσκει τις εξουσιοδοτήσεις που απορρέουν και τις επιστρέφει.
8. Η συνιστώσα εκτέλεσης ροής εργασίας, αφού έχει ρωτήσει για την εξουσιοδότηση κάθε διεργασίας, επιστρέφει την τελική απόφαση στον χρήστη. Όταν αυτός κάνει την αίτηση για εκτέλεση, δημιουργούνται οι οδηγίες εκτέλεσης.
9. Οι οδηγίες εκτέλεσης στέλνονται στο σύστημα «κινούμενων πρακτόρων», το οποίο αναλαμβάνει να διαχειριστεί την διεργασία καθώς και τις άδειες του λειτουργικού συστήματος.

5 Συμπεράσματα και Μελλοντική Εργασία

5.1 Συμπεράσματα

Στην εργασία αυτή μελετήσαμε διάφορα μοντέλα εξουσιοδότησης και πιστοποίησης. Κατηγοριοποιήσαμε τα μοντέλα εξουσιοδοτήσεων σε τρεις ομάδες: (α) τις εξουσιοδοτήσεις χρηστών – αντικειμένου, (β) τις εξουσιοδοτήσεις διεργασιών και ροών εργασίας, (γ) τις κατανεμημένες εξουσιοδοτήσεις. Παρατηρήσαμε ότι σε κάθε ομάδα δεν υπάρχει κάποια εξουσιοδότηση που να μπορεί να υποστηρίξει πλήρως όλες τις ανάγκες που χρειάζεται μια ροή επιστημονικών εργασιών.

Στην συνέχεια της εργασίας ορίσαμε το μοντέλο εξουσιοδότησης για ροές επιστημονικών εργασιών, το οποίο βασίζεται στην ιεραρχία χρηστών καθώς και στην ιεραρχία των ρόλων. Η ιεραρχία των χρηστών προσφέρει την ευκολία ορισμού και διαχείρισης των χρηστών. Η ιεραρχία των ρόλων προσφέρει την δυνατότητα της συσχέτισης ενός χρήστη με κάποιον ρόλο ανά οργανισμό. Με τον τρόπο αυτό, μπορεί να οριστούν οι ρόλοι που έχει κάποιος χρήστης σε διάφορους οργανισμούς. Παρουσιάστηκαν όλοι οι αναγκαίοι αλγόριθμοι που χρειάζονται για τον ορθό έλεγχο πιστοποίησης. Τέλος, παρουσιάστηκε ένας αλγόριθμος για να προτείνει τους ελάχιστους ρόλους που χρειάζονται για να ολοκληρωθεί η εκτέλεση μια ροής εργασίας, αν ο χρήστης δεν μπορεί να εκτελέσει κάποιες διεργασίες σε αυτήν.

Στην εργασία αυτή προτείναμε μια αρχιτεκτονική πιστοποίησης και εξουσιοδότησης, όπου διασφαλίζεται η ασφαλής μεταφορά των αντικειμένων πιστοποίησης (διαπιστευτήρια), καθώς επίσης και η ορθή πιστοποίηση από οποιοδήποτε μηχάνημα υπάρχει στο σύστημα. Η αρχιτεκτονική δίνει έναν κατανεμημένο τρόπο εκτέλεσης των αλγορίθμων πιστοποίησης.

Με τον τρόπο αυτό καταφέραμε να δημιουργήσουμε ένα ολοκληρωμένο μηχανισμό πιστοποίησης και εξουσιοδότησης για ροές εργασιών. Μάλιστα, η πρόταση μας είναι προσανατολισμένη για ροές επιστημονικών εργασιών που χρειάζονται για την εκτέλεση τους περισσότερους από έναν οργανισμούς. Οπότε προτείνουμε μια κλιμακούμενη λύση για το πρόβλημα εξουσιοδότησης σε συστήματα όπως είναι το ARION καθώς επίσης και το GMES (Global Monitoring for Environment and Security) στο οποίο συμμετέχουν περισσότεροι οργανισμοί.

Επίσης δόθηκε και η δυνατότητα χρέωσης κατά την εκτέλεση μιας διεργασίας, με την χρήση κατάλληλων συναλλακτικών μονάδων.

5.2 Μελλοντική Εργασία

Από πλευράς υλοποίησης είναι σημαντικό να υλοποιηθεί μια γραφική εφαρμογή για την εύκολη διαχείριση των χρηστών, ρολών και των εξουσιοδοτήσεων για το μοντέλο που προτείναμε. Στην υλοποίηση μας όλα γίνονται από την γραμμή εντολών, αλλά θα είναι πολύ πιο χρηστικό αν υπάρχει μια γραφική εφαρμογή.

Στην εργασία αυτή θεωρήσαμε ότι ορίζουμε ένα τύπο συναλλακτικών μονάδων, και όλες οι εξουσιοδοτήσεις γίνονται με βάση αυτό τον τύπο. Στην περίπτωση, όμως, που θέλουμε να γίνεται η χρέωση με βάση κάποιου χρηματικού ποσού αλλά και κάποιου υπολογιστικού πόρου, πρέπει να ορίσουμε στις εξουσιοδοτήσεις πολλές συναλλακτικές μονάδες. Ο ορισμός περισσότερων του ενός τύπου συναλλακτικών μονάδων στον ορισμό των εξουσιοδοτήσεων είναι προφανής. Αντί για ένα ποσό συναλλακτικών μονάδων, ορίζουμε μία πλειάδα από ποσά για κάθε ένα τύπο, οπότε ο χρήστης θα πρέπει να έχει το απαιτούμενο ποσό από κάθε συναλλακτική μονάδα. Για παράδειγμα AC(Task, Permission, ($credit_1$, $credit_2$, …, $credit_n$)). Ενδιαφέρον έχει η μελέτη των διαφόρων πολιτικών εξουσιοδότησης, στην περίπτωση που εφαρμόζονται στον χρήστη περισσότερες από μία εξουσιοδοτήσεις.

Το μοντέλο περιπλέκεται ακόμη περισσότερο όταν τα ίδια τα δεδομένα έχουν χρέωση κλιμακούμενη με το χρόνο. Αυτό είναι δυνατό να συμβεί στις περιπτώσεις που τα δεδομένα είναι αποτέλεσμα πρόγνωσης. Για να μπορέσει το μοντέλο να κάνει μια σωστή εκτίμηση της χρέωσης του χρήστη, πρέπει να έχει μια εκτίμηση για την χρέωση των δεδομένων. Μία πρώτη ανάλυση στο θέμα αυτό έχει γίνει από την εργασία [EP01], αλλά πρέπει να εξελιχτεί ώστε να υποστηριχτεί η χρέωση των δεδομένων αυτών.

Ενδιαφέρον παρουσιάζεται η μελέτη σχέσεων ισοδυναμίας μεταξύ ρόλων διαφόρων οργανισμών. Για παράδειγμα, ο ρόλος “Programmer” στον οργανισμό “A”, να ισοδυναμεί με τον ρόλο “Test Engineer” στον οργανισμό “B”. Σε αυτό το πλαίσιο θα πρέπει πρώτα να μελετηθεί το κόστος σχέσεων ισοδυναμίας, στην αναζήτηση των ρόλων κάποιου χρήστη. Αν οριστούν τέτοιες σχέσεις, γίνεται ακόμη πιο εύκολη η ανάθεση ρόλων στους χρήστες, καθώς και η υποστήριξη ενός μεγαλύτερου συνόλου οργανισμών.

Τέλος το μοντέλο που προτείναμε, δεν θεωρεί κάποιο ρόλο «υπεύθυνο» για την εκτέλεση κάποιας διεργασίας. Αυτό γίνεται μόνο στην περίπτωση που κάποιος χρήστης δεν μπορεί να εκτελέσει ο ίδιος μια διεργασία αλλά το σύστημα αναζητά έναν χρήστη που θα μπορέσει να εκτελέσει την διεργασία για λογαριασμό του. Αν οριστούν σε αυτό το μοντέλο και υποχρεώσεις τότε, θα μπορεί να υποστηρίζει γενικότερες μορφές ροών εργασιών (π.χ. επιχειρησιακές), λαμβάνοντας υπόψη τις εργασίες για εξουσιοδότηση ροών εργασιών που αναφέρθηκαν στο κεφάλαιο 2.

Παράρτημα Α. Γλώσσα περιγραφής των ροών εργασίας XASC

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="flowchart_root">
	<xs:complexType>
		<xs:sequence>
			<xs:element ref="input_params" minOccurs="0" maxOccurs="1"/>
			<xs:element ref="output_params" minOccurs="0" maxOccurs="1"/>
			<xs:element ref="int_vars" minOccurs="0" maxOccurs="1"/>
			<xs:group ref="f_element" minOccurs="0" maxOccurs="1"/>
		</xs:sequence>
		<xs:attributeGroup ref="FlowchartRootAttributes"/>
	</xs:complexType>
</xs:element>
<xs:group name="f_element">
	<xs:choice>
		<xs:element ref="task"/>
		<xs:element ref="sequence"/>
		<xs:element ref="choice"/>
		<xs:element ref="switch"/>
		<xs:element ref="parallel"/>
		<xs:element ref="hparallel"/>
		<xs:element ref="while_do"/>
	</xs:choice>
</xs:group>
<xs:element name="task">
	<xs:complexType>
		<xs:sequence>
			<xs:element ref="task_input_params" minOccurs="0"
maxOccurs="1"/>
			<xs:element ref="task_return_params" minOccurs="0"
maxOccurs="1"/>
		</xs:sequence>
		<xs:attributeGroup ref="TaskAttributes"/>
	</xs:complexType>
</xs:element>
<xs:element name="sequence" type="flowchart_elements"/>
<xs:element name="choice">
	<xs:complexType>
		<xs:sequence>
			<xs:element ref="task_input_params" minOccurs="0"
maxOccurs="1"/>
			<xs:element ref="true" minOccurs="0" maxOccurs="1"/>
			<xs:element ref="false" minOccurs="0" maxOccurs="1"/>
		</xs:sequence>
		<xs:attribute name="condition" type="xs:string" use="required"/>
	</xs:complexType>
</xs:element>
<xs:element name="true" type="flowchart_element"/>
<xs:element name="false" type="flowchart_element"/>
<xs:element name="switch">
	<xs:complexType>
		<xs:sequence>
			<xs:element ref="task_input_params" minOccurs="0"
maxOccurs="1"/>
			<xs:element ref="case" minOccurs="2" maxOccurs="unbounded"/>
		</xs:sequence>
		<xs:attribute name="condition" type="xs:string" use="required"/>
	</xs:complexType>
</xs:element>
<xs:element name="case">
	<xs:complexType>
		<xs:sequence>
			<xs:element ref="flowchart_element"/>
		</xs:sequence>
		<xs:attribute name="value" type="xs:string" use="required"/>
	</xs:complexType>
</xs:element>
<xs:complexType name="flowchart_element">
```

```
        <xs:group ref="f_element"/>
</xs:complexType>
<xs:complexType name="flowchart_elements">
        <xs:sequence>
                <xs:group ref="f_element" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="parallel" type="flowchart_elements"/>
<xs:element name="hparallel">
        <xs:complexType>
                <xs:element ref="task_input_params" minOccurs="1" maxOccurs="1"/>
                <xs:element ref="flowchart_element"/>
        </xs:complexType>
</xs:element>
<xs:element name="while_do">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="task_input_params"/>
                        <xs:group ref="f_element"/>
                </xs:sequence>
                <xs:attribute name="condition" type="xs:string" use="required"/>
        </xs:complexType>
</xs:element>
<xs:element name="task_input_params">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="in_param" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="task_return_params">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="return_param" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="input_params">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="param" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="output_params">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="param" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="int_vars">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="param" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>

<!--XSD Complex Text-Only Elements from w3schools tutorial-->
<xs:element name="param">
        <xs:complexType>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="value_string" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_int" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_float" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_boolean" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_variable" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_set" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:choice>
```

```
                <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
</xs:element>
<xs:element name="in_param">
        <xs:complexType>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="value_string" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_int" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_float" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_boolean" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_variable" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_set" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:choice>
                <xs:attribute name="name" type="xs:string" use="optional"/>
        </xs:complexType>
</xs:element>
<xs:element name="return_param">
        <xs:complexType>
                <xs:choice minOccurs="1" maxOccurs="1">
                        <xs:element ref="value_string"/>
                        <xs:element ref="value_int"/>
                        <xs:element ref="value_float"/>
                        <xs:element ref="value_boolean"/>
                        <xs:element ref="value_variable"/>
                        <xs:element ref="value_set"/>
                </xs:choice>
                <xs:attribute name="name" type="xs:string" use="optional"/>
        </xs:complexType>
</xs:element>
<xs:element name="value_string" type="xs:string"/>
<xs:element name="value_int" type="xs:int"/>
<xs:element name="value_float" type="xs:float"/>
<xs:element namr="value_boolean" type="xs:boolean"/>
<xs:element name="value_variable" type="xs:string"/>
<xs:element name="value_set">
        <xs:complexType>
                <xs:choice>
                        <xs:element ref="value_string" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_int" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_float" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_boolean" minOcurs="0"
maxOccurs="unbounded"/>
                        <xs:element ref="value_variable" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:choice>
        </xs:complexType>
</xs:element>
<!-- definition of attributes -->
<xs:attribute name="name" type="xs:string" id="ID"/>
<xs:attributeGroup name="FlowchartRootAttributes">
        <xs:attribute ref="name"/>
        <xs:attribute name="created_by" type="xs:string" use="optional"/>
        <xs:attribute name="date" type="xs:date" use="optional"/>
</xs:attributeGroup>
<xs:attributeGroup name="TaskAttributes">
<xs:attribute name="name" type="xs:string" use="required"/>
<xs:attribute name="type" use="required">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="external"/>
                        <xs:enumeration value="internal"/>
                        <xs:enumeration value="internal_function"/>
                        <xs:enumeration value="repository"/>
                        <xs:enumeration value="event"/>
                        <xs:enumeration value="wait"/>
                </xs:restriction>
        </xs:simpleType>
```

```
</xs:attribute>
<xs:attribute name="action" type="xs:string" use="required"/>
<xs:attribute name="role" type="xs:string" use="optional"/>
<xs:attribute name="doc_read" type="xs:string" use="optional"/>
<xs:attribute name="doc_update" type="xs:string" use="optional"/>
<xs:attribute name="doc_create" type="xs:string" use="optional"/>
<xs:attribute name="result" type="xs:string" use="optional"/>
<xs:attribute name="status" use="optional">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="ready"/>
                        <xs:enumeration value="running"/>
                        <xs:enumeration value="finished"/>
                        <xs:enumeration value="not_ready"/>
                        <xs:enumeration value="aborted"/>
                </xs:restriction>
        </xs:simpleType>
</xs:attribute>
<xs:attribute name="start_time" type="xs:time" use="optional"/>
<xs:attribute name="end_time" type="xs:time" use="optional"/>
<xs:attribute name="notify" type="xs:string" use="optional"/>
<xs:attribute name="mode" use="required">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="request-response"/>
                        <xs:enumeration value="notification"/>
                </xs:restriction>
        </xs:simpleType>
</xs:attribute>
<xs:attribute name="for" type="xs:int" use="optional"/>
<xs:attribute name="until" type="xs:dateTime" use="optional"/>
</xs:attributeGroup>
</xs:schema>
```

Παράρτημα Β. Περιγραφή της Ροής επιστημονικών εργασιών JRC/CNR

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow PUBLIC "SYSTEM" "xrl.dtd">

<workflow ID="WFC_84094305" name="CNR-JRC Workflow" created_by="" date="20/02/2003"
last_modified="12/06/2003" description="for workshop deadline">
  <sequence>
    <task ID="task0">
      <task_input_params/>
      <program ID="program0" name="Applet" hostname="139.91.189.101" URI="applet code"
sync="N" heavy="N" mobile="Y" simulation="N" visualization="Y" statistical="N"
validation="N" database_retrieval="N" output_filter="N" transformation="N"/>
      <task_return_params>
        <data-set ID="data-set13" name="Applet Output" hostname="139.91.189.101"
URI="/home/arion/java/applet/objects/jrc_scenario.txt" ontology_name="ARION shared
ontology" ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-
ontology.rdfs" classname="Bathymetry_data" type="produced-storable"/>
      </task_return_params>
    </task>
    <task ID="task1">
      <task_input_params>
        <data-set ID="data-set0" name="Bathymetry Dataset Addresses"
hostname="mira.ima.ge.cnr.it" URI="/home/arion/BATHYMETRY_DATASET/Datasets.txt"
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Bathymetry_data" type="specific"/>
      </task_input_params>
      <program ID="program1" name="MergeDataset" hostname="mira.ima.ge.cnr.it"
URI="/home/arion/TOOLS/MergeDataset" sync="Y" heavy="N" mobile="Y" simulation="N"
visualization="N" statistical="N" validation="N" database_retrieval="N"
output_filter="N" transformation="Y"/>
      <task_return_params>
        <data-set ID="data-set1" name="Merged Datasets" hostname="mira.ima.ge.cnr.it"
URI="/home/arion/BATHYMETRY_DATASET/Merged_Datasets.pnt" ontology_name="ARION shared
ontology" ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-
ontology.rdfs" classname="Bathymetry_data" type="produced-storable"/>
      </task_return_params>
    </task>
    <task ID="task2">
      <task_input_params>
        <data-set ID="data-set2" name="Merged Dataset" hostname="mira.ima.ge.cnr.it"
URI="/home/arion/BATHYMETRY_DATASET/Merged_Datasets.pnt" ontology_name="OCEAN_FEATURE"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Bathymetry_data" type="specific" ref="data-set1"/>
        <data-set ID="data-set4" name="Adriatic Grid Mask"
URI="/home/arion/BATHYMETRY_DATASET/Adriatic_Grid_1.pnt" hostname="mira.ima.ge.cnr.it"
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Bathymetry_data" type="specific"/>
        <parameter name="InputFile" description="" type="string" default_value=""
syntax="" optional="false" type_of_value_constraint="free" value_constraint=""/>
      </task_input_params>
      <program ID="program2" name="TIN Interpolation" hostname="mira.ima.ge.cnr.it"
URI="/home/arion/TOOLS/TIN_Interpolation" sync="Y" heavy="N" mobile="Y" simulation="N"
visualization="N" statistical="N" validation="N" database_retrieval="N"
output_filter="N" transformation="Y"/>
      <task_return_params>
        <data-set ID="data-set3" name="Gridded Bathymetry"
hostname="mira.ima.ge.cnr.it"
URI="/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry.pnt"
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Bathymetry_Grid_Mask" type="produced"/>
      </task_return_params>
    </task>
    <parallel>
       <sequence>
           <task ID="task3-1">
               <task_input_params>
```

```
                    <data-set ID="data-set3-1" name="gridded bathymetry"
URI="/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry.pnt" ref="data-set3"
hostname="mira.ima.ge.cnr.it" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Bathymetry_Grid_Mask" type="general"/>
                </task_input_params>
                <program ID="program3-1" name="Gridded Bathymetry to metadata"
hostname="mira.ima.ge.cnr.it" URI="/home/arion/TOOLS/Gridded_Bathymetry_to_Metadata
/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry.pnt
/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry.meta" sync="Y" heavy="N"
mobile="Y" simulation="N" visualization="N" statistical="N" validation="N"
database_retrieval="N" output_filter="Y" transformation="N"/>
                <task_return_params>
                  <data-set ID="data-set3-2" name="Gridded Bathymetry Metadata"
hostname="mira.ima.ge.cnr.it"
URI="/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry.meta"
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="" type="produced"/>
                </task_return_params>
            </task>
            <task ID="task3-2">
                <task_input_params>
                    <data-set ID="data-set3-3" name="Gridded Bathymetry Metadata"
URI="/tmp/Adriatic_Gridded_Bathymetry.meta" ref="data-set3-2"
hostname="139.91.189.101" ontology_name="" ontology_uri="" classname=""
type="general"/>
                </task_input_params>
                <program ID="program3-2" name="Metadata Loader"
hostname="139.91.189.101" URI="/usr/java/bin/java -classpath /home/arion/metadata-
loader/DBLoader.jar gr.forth.ics.rssdb.rssdbgui.CNR_Metadata_Loader
/tmp/Adriatic_Gridded_Bathymetry.meta" sync="Y" heavy="N" mobile="Y" simulation="N"
visualization="N" statistical="N" validation="N" database_retrieval="N"
output_filter="Y" transformation="N"/>
                <task_return_params>
                </task_return_params>
            </task>
            <task ID="task3-3">
                <task_input_params/>
                <program ID="program3-3" name="Scale latitude and longitude"
hostname="mira.ima.ge.cnr.it" URI="/home/arion/TOOLS/ScaleXY
/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry.pnt
/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry_Scaled.pnt 1000" sync="Y"
heavy="N" mobile="Y" simulation="N" visualization="N" statistical="N" validation="N"
database_retrieval="N" output_filter="Y" transformation="N"/>
                <task_return_params>
                  <data-set ID="data-set21" name="Scaled Gridded Bathymetry"
hostname="mira.ima.ge.cnr.it"
URI="/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry_Scaled.pnt"
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Bathymetry_Grid_Mask" type="produced"/>
                </task_return_params>
            </task>
            <task ID="task3-4">
                <task_input_params>
                  <data-set ID="data-set22" name="name0" hostname="mira.ima.ge.cnr.it"
URI="/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry_Scaled.pnt"
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Bathymetry_Grid_Mask" type="general" ref="data-set21"/>
                </task_input_params>
                <program ID="program3-4" name="Gridded Bathymetry to VRML"
hostname="mira.ima.ge.cnr.it" URI="/home/arion/TOOLS/xyz2VRMLTIN
/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry_Scaled.pnt
/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry_Scaled.wrl" sync="Y"
heavy="N" mobile="Y" simulation="N" visualization="N" statistical="N" validation="N"
database_retrieval="N" output_filter="Y" transformation="N"/>
                <task_return_params>
                  <data-set ID="data-set23" name="VRML Gridded Bathymetry"
hostname="mira.ima.ge.cnr.it"
URI="/home/arion/BATHYMETRY_DATASET/Adriatic_Gridded_Bathymetry_Scaled.wrl"
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Wind_data" type="produced"/>
                </task_return_params>
            </task>
```

```
        </sequence>

    <task ID="task4">
      <task_input_params>
        <data-set ID="data-set14" name="User_Input"
URI="/local0/arion/model/user_input.dat" ref="data-set13" hostname="ispramix.jrc.it"
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Bathymetry_data" type="specific"/>
        <data-set ID="data-set6" name="Bathymetry data" hostname="ispramix.jrc.it"
URI="/local0/arion/model/input/Adriatic_Gridded_Bathymetry.pnt" ontology_name="ARION
shared ontology" ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-
ontology.rdfs" classname="Bathymetry_data" type="general" ref="data-set3"/>
        <data-set ID="data-set8" name="Salinity_data" hostname="ispramix.jrc.it"
URI="/local0/arion/model/input/insal.dat" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="OCEAN_FEATURE" type="specific"/>
        <data-set ID="data-set9" name="Temperature_data" hostname="ispramix.jrc.it"
URI="/local0/arion/model/input/intem.dat" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="OCEAN_FEATURE" type="general"/>
      </task_input_params>
      <program ID="program3" name="Ispramix Adriatic set-up"
hostname="ispramix.jrc.it" URI="sh /local0/arion/model/adria_model" sync="Y" heavy="N"
mobile="N" simulation="N" visualization="N" statistical="N" validation="N"
database_retrieval="N" output_filter="N" transformation="N"/>
      <task_return_params>
        <data-set ID="data-set7" name="Sea level output" hostname="ispramix.jrc.it"
URI="/local0/arion/model/output/adriatic.cdf" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Sea_level_data" type="produced-storable"/>
        <data-set ID="data-set10" name="Temperature output" hostname="" URI=""
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Temperature_data" type="produced-storable"/>
        <data-set ID="data-set11" name="Salinity output" hostname="ispramix.jrc.it"
URI="/local0/arion/model/output/adriatic.cdf" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Salinity_data" type="produced-storable"/>
        <data-set ID="data-set12" name="Currents output" hostname="ispramix.jrc.it"
URI="/local0/arion/model/output/adriatic.cdf" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Current_data" type="produced-storable"/>
      </task_return_params>
    </task>
  </parallel>
    <task ID="task5">
      <task_input_params>
        <data-set ID="data-set7" name="Sea level output" hostname="ispramix.jrc.it"
URI="/local0/arion/model/output/adriatic.cdf" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Sea_level_data" type="produced-storable"/>
        <data-set ID="data-set10" name="Temperature output" hostname="" URI=""
ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Temperature_data" type="produced-storable"/>
        <data-set ID="data-set11" name="Salinity output" hostname="ispramix.jrc.it"
URI="/local0/arion/model/output/adriatic.cdf" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Salinity_data" type="produced-storable"/>
        <data-set ID="data-set12" name="Currents output" hostname="ispramix.jrc.it"
URI="/local0/arion/model/output/adriatic.cdf" ontology_name="ARION shared ontology"
ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-ontology.rdfs"
classname="Current_data" type="produced-storable"/>
      </task_input_params>
      <program ID="program5" name="Merge Results" hostname="139.91.189.101" sync="N"
heavy="N" mobile="Y" simulation="N" visualization="Y" statistical="N" validation="N"
database_retrieval="N" output_filter="N" transformation="N"/>
      <task_return_params>
        <data-set ID="data-set13" name="Merged Results" hostname="139.91.189.101"
URI="/home/arion/java/applet/objects/jrc_scenario.txt" ontology_name="ARION shared
ontology" ontology_uri="http://dlforum.external.forth.gr:8080/ARION-shared-
ontology.rdfs" classname="Bathymetry_data" type="produced-storable"/>
      </task_return_params>

  </sequence>
</workflow>
```

Βιβλιογραφία

[AH96] "*An Authorization Model for Workflows*" Vijayalakshmi Atluri, Wei-Kuang Huang, Proceedings of the Fourth European Symposium on Research in Computer Security, Rome, Italy, September pages 25--27, 1996. http://citeseer.nj.nec.com/atluri96authorization.html

[BCFGK97] "*Role Based Access Control for the World Wide Web*" John F. Barkley, Anthony V. Cincotta, David F. Farraiolo, Serban Gavrilla, D. Richard Kuhn. National Institute of Standards and Technology Gaithersburg 1997 http://hissa.nist.gov/rbac/rbacweb/paper.ps

[BCKNRBCMOW94] "*Security in Open Systems*" Robert Bagwill, Lisa Carnahan, Richard Kuhn, Anastase Nakassis, Michael Ransom, John Barkley, Shu-jen Chang, Paul Markovitz, Karen Olsen, John Wack. NIST Special Publication 800-7 July 1994 http://csrc.nist.gov/publications/nistpubs/800-7/main.html

[BFA97] "*A Flexible Model Supporting the Specification and Enforcement of Role-based Authorizations in Workflow Management Systems*" Elisa Bertino, Elena Ferrari, Vijayalakshmi Atluri, Technical Report, Department of Computer Science, University of Milano, January 1997. http://citeseer.nj.nec.com/bertino97flexible.html

[CSLD01] "*Access Control and AFS Groups*" Computer Systems Lab Documentation. 2001 http://www.cs.wisc.edu/csl/doc/

[EP01] *"Managing Time in Workflow Systems"* Johann Eder Euthimios Panagos. WFMC 2001

[FCK95] "*Role-Based Access Control (RBAC): Features and Motivations*" David F. Ferraiolo, Janet A. Cugini, D. Richard Kuhn. 11th Annual Computer Security Applications Proceedings 1995 http://hissa.ncsl.nist.gov/rbac/newpaper/rbac.html

[GT02] *"Auto – A secure Web workflow system using autonomous objects"* Ehud Gudes, Aharon Tubman, Data & Knowledge Engineering Volume 43, Issue 1 (October 2002) Pages: 1 – 27 Year of Publication: 2002

[HK03] *"A Secure Workflow Model on a Multi-layered State Machine"* Patrick C. K. Hung, Kamalakar Karlapalem. Conferences in Research and Practice in Information Technology Series archive Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21 Adelaide, Australia Pages: 33 - 41 Year of Publication: 2003

[HLCPVPKSG02] *"A Service infrastructure for e-Science: the case of the ARION system"* C. Houstis, S. Lalis, V.Christophides, D. Plexousakis, E. Vavalis, M. Pitikakis, K. Kritikos, A. Smardas, Ch. Gikas. 14th International Conference on Advanced Information Systems Engineering (CAiSE 2002), E-Services and the Semantic Web workshop (WES2002), Toronto, Canada, May 27-28, 2002, Springer, Lecture Notes in Computer Science LNCS 2512, pp. 175-187
http://arion-dl.org

[JMT98] "*Authorization and Attribute Certificates for Widely Distributed Access Control*" William Johnston, Srilekha Mudumbai, Mary Thompson. Proceedings of the IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises - WETICE '98.
http://citeseer.nj.nec.com/johnston98authorization.html

[M85] "*A Comment on the 'Basic Security Theorem' of Bell and LaPadula*" John McLean Naval Research Laboratory, Information Processing Letters, vol. 20, no. 2, Feb. 1985.
http://citeseer.nj.nec.com/mclean84comment.html

[MFCF02] "*File Access Control in UNIX.*" MFCF Consultant. 2002
http://www.math.uwaterloo.ca/mfcf/documentation/document-protection/unix-access-control.html

[MOV96] *"Handbook of Applied Cryptography"* Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone CRC Press ISBN: 0-8493-8523-7 October 1996
http://www.cacr.math.uwaterloo.ca/hac/about/chap2.pdf

[MS01] "*User-Level Access Control.*" Microsoft. August 24, 2001
http://www.microsoft.com/windowsxp/pro/evaluation/overviews/accesscntrl.asp

[RRAAGSCWGBH02] *"An Authorization Framework for a Grid Based Component Architecture"* Lavanya Ramakrishnan, Helen Rehn, Jay Alameda, Rachana Ananthakrishnan, Madhusudhan Govindaraju, Aleksander Slominski, Kay Connelly, Von Welch, Dennis Gannon, Randall Bramley, Shawn Hampton. Proceedings of 3rd International Workshop on Grid Computing, Springer Press, pp. 169-180, Baltimore, Maryland, November 18, 2002.

[S00] "*Access Control in UNIX and Windows NT.*" Fred B. Schneider. 2000
http://www.cs.cornell.edu/Courses/cs513/2000sp/L07.html

[SBDGHCGJ01] "*Improving the granularity of access control in Windows NT*." (Michael M. Swift, Peter Brundrett, Cliff Van Dyke, Praerit Garg, Anne Hopkins, Shannon Chan, Mario Goertzel, Gregory Jensenworth) Microsoft Corp., Redmond, WA. ACM Workshop on Role Based Access Control Proceedings of the sixth ACM symposium on Access control models and technologies Chantilly, Virginia, United States Pages: 87 – 96 Year of Publication: 2001

[SC02] *"XML-Based Policy Engine Framework for Usage Policy Management in Grids"* Babu Sundaram, Barbara M. Chapman, GRID 2002, LNCS 2536, pp. 194–198, 2002.

[SCFY96] "*Role-Based Access Control Models*" Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, Charles E. Youman. IEEE Computer 29(2): 38-47, IEEE Press, 1996. http://citeseer.nj.nec.com/sandhu96rolebased.html

[TEM03] "*Certificate-based Authorization Policy in a PKI Environment*" M.Thompson, A. Essiari, S. Mudumbai. Submitted to a special issue of ACM Transactions on Infomation and System Security, Nov 2003 http://citeseer.nj.nec.com/546062.html

[TS97] "*Task-based Authorizations Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management*" R. K. Thomas and R. S. Sandhu, 11th IFIP Working Conference on Database Security, August 1997. http://citeseer.nj.nec.com/thomas97taskbased.html

[WEBDAV00] "*WebDAV Access Control Protocol*" webdav.org 2000 http://www.webdav.org/acl/

[ZOPE03] "*Access Control in Zope*" Zope Book (zope.org) http://zope.org/Documentation/Books/ZopeBook/2_6Edition/Security.stx