Ατομική Διπλωματική Εργασία

ΥΛΟΠΟΙΗΣΗ ΕΡΓΑΛΕΙΟΥ ΜΕΣΩ ΤΟΥ ΠΡΟΤΥΠΟΥ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΕ ΟΜΑΔΕΣ ΓΙΑ ΕΛΕΓΧΟ ΕΑΝ ΕΝΑ ΣΥΣΤΗΜΑ ΙΚΑΝΟΠΟΙΕΙ ΜΙΑ ΠΟΛΙΤΙΚΗ

Φωτεινή Νικολαΐδου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΥΠΙΚΕΣ ΜΕΘΟΔΟΙ ΙΔΙΩΤΙΚΌΤΗΤΑΣ

Φωτεινή Νικολαΐδου

Επιβλέπουσα Καθηγήτρια
Άννα Φιλίππου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2015

Ευχαριστίες

Η ολοκλήρωση της εργασίας αυτής θα ήταν αδύνατη χωρίς την πολύτιμη υποστήριξη της καθηγήτριας μου Κας Άννα Φιλίππου. Αξίζει λοιπόν να της αφιερώσω αυτή τη σελίδα για να την ευχαριστήσω.

Ως επιβλέπουσα καθηγήτρια μου έδωσε τα απαραίτητα εφόδια για να καταφέρω να ολοκληρώσω με επιτυχία αυτή την εργασία. Θα ήθελα να την ευχαριστήσω που πίστεψε σε μένα και με βοήθησε να αναπτύξω ικανότητες που πίστευα πως δεν έχω καθώς και για τον πολύτιμο χρόνο που διέθεσε για να μου δώσει σημαντικά στοιχεία για το θέμα μου, αλλά και για την προθυμία και την βοήθεια, που ποτέ δε δίστασε να μου δώσει κατα τη διάρκεια εκπόνησης της εργασίας μου.

Σε αυτό το σημείο θέλω να αναφερθώ σε άτομα τα οποία υπήρξαν σημαντικοί πόλοι στη ζωή μου. Θέλω αρχικά να ευχαριστήσω μια φίλη η οποία ήταν μεγάλο στήριγμα για μένα κατα τη διάρκεια των σπουδών μου. Έπειτα θα ήθελα να ευχαριστήσω τις φίλες και τους φίλους των φοιτητικών μου χρόνων για την αξέχαστη εμπειρία που μου προσέφεραν τα τέσσερα αυτά χρόνια.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στους γονείς μου Σταύρο και Άντρη, των οποίων η πίστη στις δυνατότητες μου αποτέλεσε αρωγός σε όλα τα όνειρα και τους στόχους μου για αυτό και τους αφιερώνω την παρούσα εργασία μου.

Ευχαριστώ Φωτεινή Νικολαΐδου

Περίληψη

Με την αλματώδη ανάπτυξη της τεχνολογίας, της Πληροφορικής και των επικοινωνιών έχει έντονα παρατηρηθεί η τάση της διαχείρισης, της συλλογής και της ανταλλαγής των προσωπικών πληροφοριών σε συστήματα οργανισμών αλλά και στο διαδίκτυο. Η ανάπτυξη των τεχνολογιών κατέστησε αντιληπτή την μετατροπή της κοινωνίας σε Κοινωνία της Πληροφορίας και το ρόλο που διαδραματίζει αυτή στην ενιαία αγορά ώστε να τεθεί πλέον το ζήτημα της προστασίας των προσωπικών δεδομένων. Επειδή όμως, τα νομοθετικά μέτρα δεν επαρκούν θα πρέπει να χρησιμοποιηθούν και οι κατάλληλες τεχνολογίες ενίσχυσης της ιδιωτικότητας ανάλογα πάντα με τα τεχνολογικά ζητήματα που προκύπτουν και τις απαιτήσεις του χρήστη.

Στην παρούσα διπλωματική εργασία μελετήθηκαν οι προσδοκίες και οι πρακτικές προστασίας της ιδιωτικότητας αλλά και πως μπορούν οι πολιτικές και τα συστήματα να διατυπωθούν μέσα από το πρότυπο των αλγεβρών διεργασιών, συγκεκριμένα το π-calculus με ομάδες, έτσι ώστε να ελεγχθεί εάν ένα σύστημα ικανοποιεί την πολιτική ιδιωτικότητας και δεν καταπατάται η ιδιωτικότητα.

Στόχος της εργασίας αυτής είναι η δημιουργία ενός εργαλείου το οποίο παίρνει ως είσοδο ένα σύστημα σε XML μορφή και μια πολιτική ιδιωτικότητας επίσης σε XML μορφή. Ακολούθως τα δύο αυτά αρχεία αναλύονται μέσω του Java Dom Parser ο οποίος παίρνει την XML μορφή και δημιουργεί ένα δέντρο. Αφού γίνει αυτό, το εργαλείο αποφασίζει εάν το σύστημα ικανοποιεί την πολιτική ιδιωτικότητας. Ως αποτέλεσμα, το εργαλείο απαντά ότι ένα σύστημα ικανοποιεί μια πολιτική ιδιωτικότητας εαν το σύστημα δεν κάνει περισσότερα από αυτά που επιτρέπει η πολιτική.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο

Η τεχνολογική εξέλιξη και η συνακόλουθη συνειδητοποίηση των κινδύνων και απαιτήσεων προστασίας του ατόμου αποτελεί το κυριότερο κίνητρο για την μελέτη των τυπικών μεθόδων ιδιωτικότητας.

Ζούμε στην εποχή της ηλεκτρονικής κοινωνικής δικτύωσης. Οι περισσότεροι απο εμάς χρησιμοποιούμε λίγο-πολύ κάποια ιστοσελίδα στο διαδίκτυο όπου επικοινωνούμε με φίλους και γνωστούς. Σύμφωνα με τον Kieron O'Hara, "Ζούμε σε μια εποχή που οι ιδιωτικές μας ζωές ανεβαίνουν στο διαδίκτυο, με τις λογικές προσδοκίες να αλλάζουν συνεχώς ". Δίνει το εξής παράδειγμα στο BBC: "Πριν μια δεκαετία, τραβούσαμε μια φωτογραφία σε πάρτυ και κυκλοφορούσε μεταξύ των φίλων. Τώρα μπορεί να κυκλοφορεί στο Ίντερνετ και να την βλέπουν άγνωστοι!".

Η σύγκλιση των τεχνολογιών πληροφορικής και επικοινωνιών, η αποκέντρωση της επεξεργασίας, η διείσδυση της επεξεργασίας και της δικτύωσης στο σύνολο σχεδόν της ανθρώπινης δραστηριότητας αλλάζουν ριζικά το περιβάλλον χρήσης της προσωπικής πληροφορίας, αλλά και τα ζητήματα που εγείρονται σε σχέση με την προστασία της. Με τη βοήθεια της τεχνολογίας η συγκέντρωση δεδομένων από ποικίλες πηγές γίνεται ταχύτερα, ο συνδυασμός και η αντιστοίχιση αυτών επιτυγχάνεται ευκολότερα και καθίσταται δυνατή η

ανασυγκρότηση νέων πληροφοριών. Η φύση της πληροφορίας αλλάζει καθώς αυτή μετασχηματίζεται σε συναλλακτικό αγαθό. Οι αλλαγές αυτές οδήγησαν στη δημιουργία μιας «αγοράς προσωπικών πληροφοριών» με επιπτώσεις στα δικαιώματα και τις ελευθερίες των ατόμων και κατ΄ επέκταση του κοινωνικού συνόλου.

Σε αυτό το πλαίσιο διαμορφώνεται το αίτημα για προστασία προσωπικών δεδομένων. Σε αντίθεση με την ιδιωτικότητα υπο τη στενή έννοια, η προστασία προσωπικών δεδομένων εγείρεται ως αίτημα αναπόσπαστα συνδεδεμένο με την τεχνολογική εξέλιξη, καθώς αξιολογείται πως οι υφιστάμενες ρυθμίσεις δεν προσφέρουν επαρκή προστατευτική ασπίδα έναντι των διαφαινόμενων κινδύνων.

1.2 Στόχος

Η εργασία αυτή παρουσιάζει ένα τυπικό πλαίσιο σχετικά με τις προσδοκίες και τις πρακτικές προστασίας της έννοιας της Ιδιωτικότητας. Ένας από τους στόχους είναι να επισημοποιηθεί η έννοια της Ιδιωτικότητας, έτσι ώστε οι κατευθυντήριες γραμμές της ιδιωτικής ζωής, οι πολιτικές και οι προσδοκίες να μπορούν να είναι ακριβείς, συγκρίσιμες καθώς και να εφαρμοστούν σε ένα σύστημα επεξεργασίας πληροφοριών και να μπορεί επίσης να διαπιστωθεί εαν ένα σύστημα ικανοποιεί μια πολιτική ιδιωτικότητας [1].

Η παρούσα εργασία ασχολείται με ένα επίσημο πλαίσιο για τη μελέτη της ιδιωτικής ζωής. Το πλαίσιό μας βασίζεται στον π-λογισμό με ομάδες [13] που συνοδεύεται από ένα σύστημα τύπου για τη σύλληψη των απαιτήσεων προστασίας της ιδιωτικής ζωής που σχετίζονται με τη συλλογή πληροφοριών, την επεξεργασία των πληροφοριών και τη διάδοση πληροφοριών. Το πλαίσιο ενσωματώνει μια γλώσσα προστασίας της ιδιωτικής ζωής. Θα δείξουμε ότι ένα σύστημα σέβεται μια πολιτική, εάν ο τύπος του συστήματος είναι συμβατός με την πολιτική.

Περνώντας απο την θεωρία στην πράξη κύριος στόχος είναι να ορίσουμε μια απλή γλώσσα προκειμένου να μπορέσουμε να μεταφράσουμε ένα σύστημα και μια πολιτική μέσα απο το πρότυπο του π-λογισμού με ομάδες [13] σε XML μορφή, βάσει μιας αυστηρής γραμματικής έτσι ώστε να μπορεί να γίνει αναλυθεί μέσω του Java dom Parser και τέλος να μπορέσουμε να συμπεράνουμε εαν ένα σύστημα ικανοποιεί μια πολιτική.

1.3 Μεθοδολογία

Λόγω του ότι η ΑΔΕ έχει θεωρητικό αλλα και πρακτικό χαρακτήρα, για την επίτευξη των στόχων της διπλωματικής εργασίας ακολουθήθηκε η πιο κάτω μεθοδολογία:

Όσον αφορά το θεωρητικό μέρος, αρχικά μελέτησα διάφορα επιστημονικά άρθρα σχετικά με το θέμα μου, έτσι ώστε να αντιληφθώ καλύτερα την έννοια της Ιδιωτικότητας αλλά και τη μεθοδολογία που έπρεπε να ακολουθήσω έτσι ώστε να γίνει η υλοποίηση του θεωρητικού μέρους. Στη συνέχεια ασχολήθηκα με την επίλυση διάφορων ασκήσεων σχετικά με τις άλγεβρες διεργασιών CCS και το p-calculus που αποτελεί επέκταση, προκειμένου να κατανοήσω καλύτερα κάποια σημεία έτσι ώστε να μπορώ με ευκολία να προχωρήσω στο πρακτικό μέρος.

Αφού μελέτησα εις βάθος τις άλγεβρες διεργασιών, μου δόθηκε ένα επιστημονικό άρθρο της επιβλέπουσας καθηγήτριας μου που ήταν το τελικό βήμα πρωτού προχωρήσω στο πρακτικό μέρος. Συγκεκριμένα, μέσα από το άρθρο αυτό κατανόησα ακόμη καλύτερα την έννοια του π-λογισμού με ομάδες, τις πολιτικές ιδιωτικότητας πως πρέπει να γράφονται για να είναι καλά ορισμένες, τους κανόνες ελέγχου τύπων του συστήματος αλλά και για το πότε ένα σύστημα ικανοποιεί ή όχι μια πολιτική ιδιωτικότητας.

Αφού ενημερώθηκα και κατανόησα αυτές τις έννοιες έπρεπε να εξοικοιωθώ με τον Java Dom Parser αλλά και με το να ορίσω αυστηρές γραμματικές για να μεταφράσω το σύστημα και την πολιτική σε XML μορφή και τέλος ακολούθησε η υλοποίηση.

1.4 Οργάνωση της εργασίας

Η παρούσα διπλωματική εργασία αποτελείται απο έξι κεφάλαια. Στο κεφάλαιο 2 γίνεται μια ιστορική αναδρομή όσον αφορά την έννοια της ιδιωτικότητας, για τις απαιτήσεις που υπάρχουν καθώς και τις διάφορες μορφές της ιδιωτικότητας που προέκυψαν μέσα από το πέρασμα των χρόνων, αλλά και για το ποιές προκλήσεις και κινδύνοι εμφανίζονται σύμφωνα με τον ρόλο που κατέχει η τεχνολογία λόγω της ραγδαίας ανάπτυξη της. Σε αυτό το κεφάλαιο επίσης γίνεται μια γενική περιγραφή σχετικά με τις τυπικές μεθόδους ιδιωτικότητας.

Στο Κεφάλαιο 3 παρουσιάζονται οι άλγεβρες διεργασιών και συγκεκριμένα CCS, π-λογισμός και π-λογισμός με ομάδες. Εν συνεχεία, παρουσιάζεται ο λογισμός της προηγούμενης εργασίας, οι κανόνες του τύπου συστήματος και των πολιτικών, πάνω στο οποίο βασίζεται και η ανάπτυξη του εργαλείου.

Στο Κεφάλαιο 4 παρουσιάζεται η ανάλυση απαιτήσεων και προδιαγραφών του λογισμικού που δημιουργήθηκε. Αναλυτικότερα, περιγράφεται γενικά το εργαλείο, ο σκοπός του και οι λειτουργίες που πρέπει να κάνει βάσει των απαιτήσεων και οι γενικοί περιορισμοί. Έπειτα, αναφέρονται οι ειδικές απαιτήσεις του εργαλείου, δηλαδή τι ζητά το εργαλείο από τον χρήστη και ως επακόλουθο προκύπτουν και τα χαρακτηριστικά του λογισμικού.

Στο κεφάλαιο 5 παρουσιάζεται η υλοποίηση και το περιβάλλον στο οποίο αναπτύχθηκε το λογισμικό, η XML μορφή που επιλέχθηκε για να μεταφράσουμε τις γραμματικές για αυτό και υπάρχουν οι οδηγοί για τη συγγραφή των δύο xml αρχείων που θα δίνει ο χρήστης ως είσοδο, καθώς και η γλώσσα προγραμματισμού που χρησιμοποιήθηκε. Επιπρόσθετα, παρουσιάζονται και οι ελέγχοι που πραγματοποιήθηκαν οι οποίοι πιστοποιούν την ορθότητα του λογισμικού.

Στο Κεφάλαιο 6 παρουσιάζονται τα συμπεράσματα, τα προβλήματα και οι δυσκολίες που προέκυψαν κατά την εκπόνηση της παρούσας εργασίας αλλά και μελλοντικές επεκτάσεις που θα μπορούσαν να γίνουν.

Τέλος, ως παράρτημα παρουσιάζεται ο κώδικας που αναπτύχθηκε για την υλοποίηση του εργαλείου.

Κεφάλαιο 2

Ιδιωτικότητα

2.1 Εισαγωγή στην Ιδιωτικότητα

Το ζήτημα της ιδιωτικότητας δεν είναι ζήτημα που απασχόλησε την κοινωνία για πρώτη φορά. Η ιδιωτικότητα είναι έννοια παλιά όσο η Εύα και σύγχρονη όσο το Google, ανάγεται στις απαρχές της ανθρώπινης ιστορίας και διατρέχει την εξέλιξη της [4].

Εντούτοις, δεν αποτελεί μια ανθρωπολογική σταθερά. Ως έννοια δεν είχε πάντα την ίδια βαρύτητα και το ίδιο περιεχόμενο. Σημαίνει κάτι διαφορετικό για όλους. Η απάντηση σήμερα στο ακριβώς εννούμε με τον όρο ιδιωτικότητα εξαρτάται προς ποιόν απευθύνεται η ερώτηση. Ο κάθε πολιτισμός αντιλαμβάνεται αλλιώς την έννοια της. Για παράδειγμα σε κάποιους πολιτισμούς θεωρείται αγένεια να ρωτήσουμε την ηλικία κάποιου ή το μισθό του [3]. Σε θεσμικό επίπεδο η ιδιωτικότητα παρουσιάζεται όχι μόνο σαν μια φυσική ανάγκη αλλά ταυτόχρονα και η κατάκτηση ενός προνομίου μιας κοινωνικής ομάδας. Συχνά διατυπώνεται και ως δικαίωμα της προσωπικότητας.

Με την πάροδο του χρόνου η τεχνολογική επανάσταση έχει εμπλουτίσει την «κλασσική» αντίληψη της ιδιωτικότητας με δικαιώματα όπως τον περιορισμό προσβασιμότητας, το δικαίωμα για ιδιωτική ζωή, το δικαίωμα στο απόρρητο. Παράλληλα γίνεται όλο και περισσότερο κατανοητό ότι ως αξίωση σεβασμού του απορρήτου από την μια θεωρείται αναγκαία, αλλά απο την άλλη ανεπαρκής για την προστασία στο άτομο [4]. Σύμφωνα μάλιστα με τον ορισμό που δίνει ο Westin, η έννοια της ιδιωτικότητας ορίζεται ως η αξίωση των ατόμων να προσδιορίζουν οι ίδιοι πότε, πως και σε ποια έκταση οι προσωπικές τους πληροφορίες θα αποκαλύπτονται [5].

Στη βιβλιογραφία, τα παλαιότερα ζητήματα ιδιωτικότητας αφορούν την εδαφική ακεραιότητα. Στα τέλη του 19ου αιώνα το "δικαίωμα να μένεις μόνος", όπως το αντιλήφθηκαν οι Brandeis και Warren, αντικατόπτριζε την ανάγκη για εδαφική ιδιωτικότητα (territorial privacy), δίνοντας νομική υπόσταση στο συλλογικό φαντασιακό του "my home is my castle".

Εν κατακλείδι, σύμφωνα με τον όρο της ηθικής αυτονομίας, σαν ευρωπαϊκή προσέγγιση, η ιδιωτικότητα συνδέεται με την ανθρώπινη αξιοπρέπεια και εκφράζεται ως αξίωση να μην καθίσταται το άτομο ως υποχείριο για την επίτευξη ενός σκοπού [4].

2.2 Μορφές Ιδιωτικότητας

Η ιδιωτικότητα παίρνει διαφορετική υπόσταση ανάλογα με την τρέχουσα κοινωνική συνθήκη. Θα μπορούσε κανείς να υποστηρίξει ότι εκείνο που διαφοροποιεί την έννοια της ιδιωτικότητας μέσα στον χρόνο είναι το μέσο. Μέσα από τα χρόνια η ανάγκη για ιδιωτικότητα καλύπτεται με διαφορετικά μέσα, έτσι λοιπόν με βάση αυτό αναδύονται και οι ποικίλες μορφές της.

Γύρω στο 1930 εμφανίζεται η τηλεφωνική υποκλοπή. Ο ανώτατος δικαστής Brandeis με βάσει τα συμφραζόμενα της εποχής εξηγά πως η εδαφική ακεραιότητα μεταβαίνει στις τηλεπικοινωνίες. Οι τηλεφωνικές υποκλοπές χρησιμοποιήθηκαν απο την αμερικάνικη κυβέρνηση ως αποδείξεις τη στιγμή που κάτι τέτοιο δεν υπήρχε στα συμβόλαια των τηλεφωνικών παροχών αλλά ούτε και στους νόμους της πολιτείας. Στη δίκη του Olmstead εναντίων των Ηνωμένων Πολιτειών υποστηρίχθηκε ότι η προστασία του Συντάγματος για την ιδιωτικότητα του οίκου ανάγεται στην ιδιωτικότητα των τηλεπικοινωνιών.

Ως αποτέλεσμα της απόφασης της ναζιστικής υπηρεσίας για ιατρικά πειράματα και υποχρεωτικό ευνουχισμό καταπατάται η σωματική ιδιωτικότητα (bodily privacy). Σήμερα, συζητήσεις για τη σωματική ιδιωτικότητα επανεμφανίζονται στην επικαιρότητα με τη συγκέντρωση γενετικού υλικού (DNA) πολιτών. Παράδειγμα μπορεί να αποτελέσει στη Μεγάλη Βρετανία, η εθνική βάση δεδομένων DNA και ο τρόπος με τον οποίο εξελίσσονται πλέον συλλήψεις και προσαγωγές.

Μεταξύ του 1960-1970 με την ανακάλυψη της αυτοματοποιημένης επεξεργασίας δεδομένων που αποτελεί αποτελεσματικό μέσο για την καταλογράφηση των πολιτών, προέκυψε και η

πληροφοριακή ιδιωτικότητα (informational privacy). Τα ευρωπαϊκά κράτη προστατεύουν την πληροφοριακή ιδιωτικότητα μέσω της νομοθεσίας ή της νομολογίας.

Πρέπει να αντιληφθούμε την προστασία της ιδιωτικότητας, εστιάζοντας στις ειδικές μορφές της διαταραχής και τις συγκεκριμένες δραστηριότητες που την διαταράσσουν παρά να ψάχνει κανείς για τον κοινό παρονομαστή που συνδέει όλα αυτά.

2.3 Ταξινόμηση Ιδιωτικότητας

Ο Daniel J. Solove, χρησιμοποίησε «ταξινομήσεις» προκειμένου να διευκολυνθεί η ανάλυση του σχετικά με τις μορφές ιδιωτικότητας. Ο Solove διαιρεί την ιδιωτική ζωή στις εξής 4 κατηγορίες οι οποίες όπως λέει, είναι συχνα επικαλυπτόμενες. Πιο κάτω παραθέτονται οι 4 ταξινομήσεις που όρισε ο Solove .

- **Συλλογή πληροφοριών:** Η συλλογή πληροφοριών μπορεί να οδηγήσει σε ρήξη βασιζόμενη στη διαδικασία με την οποία γίνεται η συλλογή πληροφοριών ακόμα και στην περίπτωση όπου δεν γίνει αποκάλυψη των πληροφοριών που συλλέχθηκαν. Υπάρχουν δύο τρόποι συλλογής πληροφοριών, είτε μέσω της επιτήρησης που γίνεται με την οπτική παρακολούθηση και την παρατήρηση ήχου , είτε μέσω της ανάκρισης όπου γίνονται ερωτήσεις προκειμένου να αποκτήσουν τις απαραίτητες πληροφορίες.

- **Επεξεργασία πληροφοριών:** Η επεξεργασία πληροφοριών περιλαμβάνει τη χρήση, την αποθήκευση και το χειρισμό των δεδομένων που συλλέχθηκαν (απο την πρώτη μορφή κατηγορίας). Υπάρχουν πέντε μορφές επεξεργασίας πληροφοριών. Η πρώτη είναι η συνάθροιση (aggregation) όπου γίνεται συνδυασμός μεταξύ διάφορων κομματιών δεδομένων και έτσι σχηματίζεται το "πορτρέτο" του ατόμου. Η δεύτερη μορφή είναι η ταυτοποίηση (identification) όπου συνδέει πληροφορίες για ένα άτομο. Σύμφωνα με τον Roger Clarke, η ταυτοποίηση-αναγνώριση είναι «η σύνδεση των δεδομένων με ένα συγκεκριμένο ανθρώπινο ον». Για παράδειγμα μας παρέχει τη δυνατότητα εξακρίβωσης της ταυτότητας ότι το πρόσωπο που έχει πρόσβαση σε αρχεία της είναι πράγματι ο ιδιοκτήτης του λογαριασμού αλλά και να ανακαλύψει το δράστη ενός εγκλήματος από τα ίχνη που άφησαν πίσω, όπως δακτυλικά αποτυπώματα και γενετικό υλικό. Η ταυτοποίηση είναι παρόμοια με τη συνάθροιση καθώς και οι δύο περιλαμβάνουν τον συνδυασμό των διαφορετικών πληροφοριών, η

μία είναι η ταυτότητα ενός ατόμου. Η τρίτη μορφή είναι η μη επαρκής παροχή ασφάλειας (insecurity) η οποία καθώς δημιουργεί κινδύνους που προκύπτουν από ανεπαρκή προστασία των δεδομένων. Η τέταρτη μορφή είναι η δευτεροβάθμια χρήση όπου τα δεδομένα που αφορούν κάποιο συγκεκριμένο άτομο χρησιμοποιήθηκαν για σκοπούς προς τους οποίους το άτομο δεν έδωσε την έγκριση του. Τέλος, η πέμπτη μορφή είναι ο αποκλεισμός που επίσης σχετίζεται με την μη επαρκή παροχή ασφάλειας λόγω της έλλειψης λογοδοσίας στα συστήματα καταγραφής των προσωπικών δεδομένων.

- **Διάδοση πληροφοριών:** Η διάδοση των πληροφοριών είναι μια από τις πιο επικίνδυνες μορφές της ιδιωτικότητας. Προκαλεί πολλά προβλήματα όπως είναι η αποκάλυψη των προσωπικών δεδομένων ή, η απειλή της διάδοσης πληροφοριών. Η ομάδα αυτή περιλαμβάνει την παραβίαση του απορρήτου με την αποκάλυψη των προσωπικών πληροφοριών χωρίς την εξουσιοδότηση του ατόμου, τη γνωστοποίηση, την έκθεση όταν αποκαλύπτοναι προσωπικά δεδομένα εκτίθοντας το άτομο, την αύξηση της προσβασιμότητας όπου δίνεται δικαίωμα στην πρόσβαση προσωπικών στοιχείων , τον εκβιασμό όπου απειλούν να αποκαλύψουν προσωπικά δεδομένα, την πίστωση και την παραμόρφωση.

- **Αποκάλυψη:** Η αποκάλυψη συμβαίνει στην περίπτωση που αποκαλύπτονται πληροφορίες που σχετίζονται με κάποιο άτομο σε άλλα άτομα. Η αποκάλυψη διαφέρει απο την παραβίαση του απορρήτου, επειδή η βλάβη στην αποκάλυψη περιλαμβάνει τη ζημιά στη φήμη που προκαλείται από τη διάδοση σε αντίθεση με το κακό με την παραβίαση του απορρήτου είναι η παραβίαση της εμπιστοσύνης.

2.4 Ο ρόλος της ανάπτυξης της τεχνολογίας

Αν και οι τεχνολογικές εξελίξεις οδήγησαν σε μια ακόμα πιο άνετη ζωή για τους ανθρώπους, εντούτοις η αλματώδης ανάπτυξη των δικτύων, γνωστότερο των οποίων είναι το Διαδίκτυο, διαμορφώνουν ένα ριζικά διαφορετικό πλαίσιο παραγωγής και επεξεργασίας πληροφοριών. Ο κόσμος εξελίσσεται σε μία «δικτυωμένη κοινωνία» όπου τα προσωπικά δεδομένα συλλέγονται, εμπλουτίζονται, τροποποιούνται, ανταλλάσσονται και επαναχρησιμοποιούνται διαρκώς.

Οι εξελίξεις στην πληροφορική τεχνολογία έχουν ως επόμενο ότι, σε θεωρητικό επίπεδο, δεν υπάρχει πλέον όριο στην πληροφορία που μπορεί να καταχωρηθεί, στην ανάλυση που μπορεί να γίνει, στο χρονικό διάστημα για το οποίο μπορεί να τηρηθεί. Αντίστοιχα απεριόριστες μοιάζουν να είναι οι δυνατότητες εκμετάλλευσης: Διατυπώσεις όπως η «εξόρυξη δεδομένων» (data mining) δεν αποδίδουν μόνο περιγραφικά τεχνικές αλλά υποδηλώνουν έναν νέο «πυρετό του (πληροφοριακού) χρυσού». Σε αυτό συντελεί το αδιαμφισβήτητο γεγονός ότι όλο και περισσότερες δραστηριότητες, της πιο ποικίλης φύσης, οργανώνονται και διεκπεραιώνονται σε απευθείας σύνδεση. Ήδη ως προς την ποσοτική διάσταση πρέπει να παρατηρηθεί ότι η ηλεκτρονική επικοινωνία, αμφίδρομη ή μη, παράγει πολύ περισσότερα δεδομένα, δεδομένα που απεικονίζουν την κίνηση (πλοήγηση, επικοινωνία) του χρήστη μέσα στα δίκτυα. Τα όρια μεταξύ των εξωτερικών στοιχείων και του περιεχομένου μιας επικοινωνίας είναι πλέον δυσδιάκριτα ενώ σταδιακά γίνεται όλο και πιο σαφές ότι η επεξεργασία προσωπικών πληροφοριών και η επικοινωνία είναι διαδικασίες που αναπόφευκτα συμπλέκονται. Περαιτέρω, στα δίκτυα χάνει εν τέλει τη σημασία της η διάκριση σε συλλογή, καταχώριση και διαβίβαση των δεδομένων.

Η διαθεσιμότητα και διάδοση των νέων τεχνολογιών όχι μόνο αποκεντρώνει τις δυνατότητες επεξεργασίας αλλά καθιστά τον καθένα χρήστη τους και ταυτόχρονα πηγή παραγωγής και διάθεσης πληροφορίας. Στο περιβάλλον του Web 2.0 συζητείται και προωθείται η παροχή διαδραστικών υπηρεσιών, όπου όλο και περισσότερο εξαφανίζονται τα όρια μεταξύ παρόχων και χρηστών. Η παραγωγή και διάδοση της πληροφορίας δεν είναι πλέον θέμα και αντικείμενο μιας κατηγορίας ειδικών: το Web 2.0 (θα) επικυριαρχείται από το περιεχόμενο που παράγουν οι χρήστες (user generated content), από πληροφορίες που αυτοί διαμοιράζονται, αναζητούν και λαμβάνουν.

Ποτέ στην ιστορία δεν μπορούσε να αποτυπωθεί με τεχνικό τρόπο και σε τεχνικά μέσα όλη η ζωή ενός προσώπου , όπως αυτό συμβαίνει σήμερα. Η καταγραφή των δεδομένων κίνησης, η αξιοποίηση των δεδομένων που παράγονται και ανταλλάσσονται στα δίκτυα κοινωνικής δικτύωσης όπως το Facebook, ο πολλαπλασιασμός των κλειστών κυκλωμάτων τηλεόρασης, η αυξημένη χρήση βιομετρίας και τεχνολογιών RFID ως μέσου αποθήκευσης πληροφορίας και παρακολούθησης πραγμάτων και ανθρώπων επιτρέπουν τη διεισδυτική όσο και ολιστική διάγνωση και αποτύπωση ιδιαίτερων πτυχών της προσωπικότητας και καθημερινότητας ενός προσώπου αλλά και την ανασύνθεση αυτών μέσω των ψηφίδων πληροφορίας που συμπληρώνουν το puzzle της ζωής του.

2.5 Προκλήσεις και κινδύνοι

Παρόλο που οι διάχυτες υπολογιστικές τεχνολογίες οδήγησαν στην ανάπτυξη πολλών υπηρεσιών προσφέροντας στο ευρύ κοινό μια πιο «άνετη» ζωή καθώς και τη δυνατότητα αμεσότερης επικοινωνίας και ανταλλαγής πληροφοριών μέσω του Παγκόσμιου ιστού (Web) στο χώρο του ηλεκτρονικού επιχειρείν, εντούτοις η ανεξέλεγκτη καταχώρηση προσωπικών δεδομένων σε αρχεία υπηρεσιών (οργανισμών, εταιρειών) μπορεί να εγείρει ζητήματα σχετικά με την ιδιωτικότητα του ατόμου καθώς και προοπτικές συρρίκνωσης των ατομικών ελευθεριών και της αυτονομίας.

Στις μέρες μας οι πλείστοι οργανισμοί καθώς και τα ιδρύματα (π.χ τράπεζες, κέντρα υποστήριξης πελατών, ιδρύματα υγειονομικής περίθαλψης) έχουν αποθηκευμένες τις προσωπικές πληροφορίες των πελατών σε βάσεις δεδομένων, γεγονός που καθιστά αυτές τις πληροφορίες ευαίσθητες αφού τα στελέχη των οργανισμών έρχονται σε επαφή καθημερινά με τις πληροφορίες προκειμένου να διεκπεραιώσουν τις εργασίες που τους ανατίθονται. Η συνεχής αυτή αλληλεπίδραση τους με τα δεδομένα δημιουργεί πολλές δυσκολίες αλλά ταυτόχρονα και κινδύνους μια και είναι υπεύθυνοι να τα διαχειριστούν με ορθολογισμό και συμμόρφωση πηγαίνοντας κόντρα στην παραβίαση της ιδιωτικής ζωής.

Συγκεκριμένα το μεγαλύτερο πρόβλημα που αντιμετωπίζουν οι οργανισμοί και τα ιδρύματα που χρησιμοποιούν βάσεις δεδομένων είναι ο ορθός σχεδιασμός των δραστηριοτήτων τους έτσι ώστε να αποφεύγονται οι κίνδυνοι όπως η αποκάλυψη των προσωπικών δεδομένων και να εξυπηρετούνται οι πελάτες εύκολα και αποτελεσματικά. Ένα τρανταχτό παράδειγμα ιδρύματος που αντιμετωπίζει τέτοιες προκλήσεις είναι τα νοσοκομεία όπου οι προσωπικές πληροφορίες πρέπει να χρησιμοποιούνται αποκλειστικά για την ιατρική περίθαλψη των

ασθενών χωρίς να αποκαλύπτονται άσκοπα για άλλους λόγους εκτός πλαισίου της ιατρικής περίθαλψης τους. Η σοβαρότητα των κινδύνων κρίνεται μεγάλη, όπως φάνηκε τον Μάιο του 2006 με το περιστατικό της κλοπής των 26 εκατομμυρίων δολαρίων εγγραφών από βετεράνους που κλάπηκαν μετά απο τη διάρρηξη στο σπίτι ενός αναλυτή ηλεκτρονικών υπολογιστών και ακόμη οι Ομοσπονδιακοί πράκτορες προσπαθούν να τα ανακτήσουν [8].

Προκειμένου να αντιμετωπιστούν αυτοί οι κινδύνοι που παραβιάζουν την ιδιωτική ζωή αναπτύχθηκαν κάποιες πολιτικές προστασίας της ιδιωτικότητας. Κάποιες απο αυτές τις πολιτικές στις Ηνωμένες Πολιτείες όπως το HIPAA [9] για την υγειονομική περίθαλψη, το COPPA [4] για το διαδίκτυο και το GLBA [13,14] για τα χρηματοπιστωτικά ιδρύματα αποτέλεσαν ώθηση πολλών επιχειρήσεων έτσι ώστε να διορίσουν υπεύθυνους προστασίας προσωπικών δεδομένων οι οποίοι έχουν την ευθύνη για τον απόλυτο έλεγχο των δεδομένων. Τα ιδρύματα και οι οργανισμοί που εφαρμόζουν αυτές τις πολιτικές θα πρέπει να είναι ιδιαίτερα προσεκτικοί στον τρόπο με τον οποίο διαχειρίζονται τις πληροφορίες έτσι ώστε να υπάρχει μια ισορροπία μεταξύ των στόχων της ιδιωτικής ζωής και της επιχείρησης.

2.6 Τυπικές μέθοδοι ιδιωτικότητας

Στόχος των τυπικών μεθόδων ιδιωτικότητας είναι να βελτιώσουν τα συστήματα όσον αφορά την ποιότητα τους. Οι μεθόδοι αυτοί ορίζονται ως τυπικές μια και στηρίζονται κυρίως στη λογική, σε μαθηματικές θεωρίες και στα αυτόματα. Οι τυπικές μέθοδοι μπορούν και πρέπει να εφαρμόζονται στην ιδιωτική ζωή.

Όλα τα μηχανήματα που εντάσσονται στην κοινότητα των τυπικών μεθόδων μπορούν να μας βοηθήσουν να αποκτήσουμε μια πιο αυστηρή κατανόηση των δικαιωμάτων της ιδιωτικής ζωής, των απειλών και των παραβιάσεων. Μπορούμε να χρησιμοποιήσουμε τυπικά μοντέλα, από αυτόματα (μηχανές καταστάσεων) μέχρι άλγεβρες διεργασιών, για να περιγράψουμε τη συμπεριφορά ενός συστήματος αλλά και τις απειλές του περιβάλλοντος.

Επιπρόσθετα, μπορούμε να χρησιμοποιήσουμε τυπικές λογικές και επίσημες γλώσσες για να αναπαραστήσουμε διάφορες πτυχές της ιδιωτικής ζωής, να δηλώσουμε τις ιδιότητες των συστημάτων, να δηλώσουμε πολιτικές προστασίας προσωπικών δεδομένων, για τον λόγο σχετικά με το πότε ένα μοντέλο ικανοποιεί μια ιδιότητα ή την πολιτική, και τον εντοπισμό περιπτώσεων ασυμφωνίας μεταξύ των διαφόρων πολιτικών προστασίας της ιδιωτικής ζωής. Γενικά, οι τυπικές μέθοδοι αποτελούν τεχνικές για την ανάλυση και την περιγραφή

συστημάτων και παραδοσιακά μοντελοποιούμε ένα σύστημα, το περιβάλλον του και τις αλληλεπιδράσεις μεταξύ των δύο.

Κύρια χρησιμότητα των τυπικών μεθόδων είναι σε συστήματα όπου πολλές διεργασίες ενεργούν ταυτόχρονα, οπότε οδηγούμαστε στο συμπέρασμα του μη-ντετερμινισμού. Η αυτοματοποιημένη ανάλυση και τα εργαλεία μας επιτρέπουν να αναβαθμίσουμε την εφαρμογή αυτών των θεμελιακών μοντέλων και λογικών σε ρεαλιστικά συστήματα.

Κεφάλαιο 3

Τύπος συστήματος και πολιτικές ιδιωτικότητας στον π-λογισμό

3.1 Άλγεβρες διεργασιών

Μια άλγεβρα διεργασιών ορίζεται ως μια αυστηρά μαθηματική γλώσσα με καθορισμένη σημασιολογία όπου επιτρέπει την περιγραφή και την επαλήθευση ιδιοτήτων της ταυτόχρονης επικοινωνίας συστημάτων (παράλληλων) και ανήκει στην οικογένεια των τυπικών μεθόδων.

Η προσέγγιση που βασίζεται στις άλγεβρες διεργασιών υπήρξε πολύ επιτυχής όσον αφορά την παροχή τυπικής σημασιολογίας των παράλληλων συστημάτων και την απόδειξη ιδιοτήτων τους. Η δημιουργία των αλγεβρών διεργασιών προέκυψε απο τους δύο πρωτοπόρους ιδρυτές των αλγεβρών διεργασιών Tony Hoare και Robin Milner στους οποίους έχει δοθεί το βραβείο Turing [6].

Το βασικό στοιχείο μιας άλγεβρας διεργασιών είναι η σύνταξη της. Η σύνταξη μιας άλγεβρα διεργασιών είναι το σύνολο των κανόνων που καθορίζουν τους συνδυασμούς των συμβόλων που θεωρούνται σωστά δομημένα προγράμματα σε αυτή τη γλώσσα. Υπάρχουν πολλές προσεγγίσεις για την παροχή μιας αυστηρής μαθηματικής κατανόησης της σημασιολογίας

των συντακτικά σωστών όρων της διαδικασίας. Οι κυριότερες από αυτές είναι εκείνες που χρησιμοποιούνται επίσης για να περιγράψουν τη σημασιολογία των διαδοχικών συστημάτων δηλαδή η λειτουργική (operational), δηλωτική (denotational) και αλγεβρική (algebraic) σημασιολογία [6].

Για να μοντελοποιήσουμε ένα σύστημα μέσω μιας άλγεβρας διεργασιών αρκεί να ορίσουμε το όνομα των καναλιών για να επιτευχθεί η επικοινωνία και τους βασικούς τελεστές από τους οποίους προκύπτουν οι κανόνες. Οι άλγεβρες μπορούν να θεωρηθούν ως μοντέλα διεργασιών, δηλαδή ως οντότητες που δρουν και αλληλεπιδρούν συνεχώς με άλλες οντότητες και με το κοινό τους περιβάλλον. Οι οντότητες μπορεί να είναι οντότητες που ανήκουν στον πραγματικό κόσμο (ακόμα και άνθρωποι) ή μπορεί να είναι αντικείμενα, που ενσωματώνονται στο υλικό του υπολογιστή ή σε συστήματα λογισμικού [6]. Προκειμένου να γίνει αυτή η επικοινωνία χρειάζεται να ορίσουμε τις ενέργειες εισόδου και εξόδου μεταξύ των καναλιών. Με το $a(b)$ εκφράζουμε ενέργεια εισόδου όπου το το b λαμβάνεται μέσω του καναλιού a και το $\bar{a}(b)$ όπου το b στέλλεται μέσω του καναλιού a.

Υπάρχουν τρία είδη τελεστών. Ο τελεστής (+) ο οποίος επιλέγει μια απο τις δύο διεργασίες για να εκτελεστεί και μπορεί να γραφτεί π.χ ως A + B δηλαδή μπορεί να προχωρήσει στην εκτέλεση η διεργασία A ή, η διεργασία B. Ο δεύτερος τελεστής (|) ο οποίος εκφράζει την παράλληλη σύνθεση διεργασιών, δηλαδή οι διεργασίες μπορούν να συντονιστούν μεταξύ τους και να ανταλλάξουν πληροφορίες και μπορεί να γραφτεί ως A | B . Ο τελευταίος τελεστής (.) εκφράζει τη διαδοχικότητα. Έστω ότι έχουμε το α.B , τότε εκτελείται η ενέργεια α και ακολούθως συνεχίζει ως η διεργασία B.

3.1.1 Η Άλγεβρα διεργασιών CCS

Η άλγεβρα διεργασιών CCS [2] αναπτύχθηκε απο τον Robin Milner (Turing Award, 1991) κατα τα τέλη του 1970/ αρχές του 1980. Η βασική μέθοδος επικοινωνίας αυτής της άλγεβρας είναι ο δυαδικός συγχρονισμός. Οι διεργασίες της κτίζονται απο ένα σύνολο ατομικών ενεργειών με τη χρήση τελεστών σύνθεσης διεργασιών. Οι ενέργειες στη CCS μπορούν να είναι ενέργειες εισόδου, ενέργειες εξόδου ή εσωτερική ενέργεια [2].

Στη CCS, τα συστήματα αλληλεπιδρούν μεταξύ τους αλλά και με το περιβάλλον τους μέσω συγχρονισμών που συμβαίνουν σε κανάλια. Ο συγχρονισμός συμβαίνει ως εξής: όταν μια διεργασία είναι έτοιμη για είσοδο σε κάποιο κανάλι και την ίδια στιγμή κάποια άλλη

διεργασία είναι έτοιμη για έξοδο επίσης στο ίδιο κανάλι με την πρώτη, τότε πετυχγάνεται ο συγχρονισμός και οι διεργασίες προχωρούν με την εκτέλεση τους. Οι εξωτερικές ενέργειες που εκτελεί ένα σύστημα μπορούν να θεωρηθούν ως η διεπιφάνειά του[2].

Η άλγεβρα διεργασιών CCS ακολουθεί την πιο κάτω γραμματική:

E ::=	0	τερματισμός
|	$a.E$	διαδοχή
|	$E_1 + E_2$	επιλογή
|	$E_1 \mid E_2$	ταυτοχρονισμός
|	$E \setminus L$	περιορισμός
|	$E[f]$	μετονομασία
|	C	κλήση

Με βάση την πιο πάνω γραμματική μπορούμε να θεωρήσουμε τους τελεστές της CCS ως πράξεις για κτίσιμο και συνδυασμό κουτιών ως εξής[2]:

- 0 : Διεργασία που δεν ανταποκρίνεται (τερματισμός ή έξοδος)

- $a.E$: Διεργασία πρόθυμη να εκτελέσει την ενέργεια α και στη συνέχεια συμπεριφέρεται ως E.

- $E_1 + E_2$: Διεργασία η οποία προσφέρει την επιλογή ανάμεσα στις E_1 και E_2.

- $E_1 \mid E_2$: Διεργασία στην οποία τρέχουν παράλληλα οι E_1 και E_2. Οι υποδιεργασίς αυτές μπορούν να εκτελούν ανεξάρτητα την εργασία τους ή και να επικοινωνούν μεταξύ τους στα κανάλια που τις συνδέουν.

- $E \setminus L$: Διεργασία με τα κανάλια L δεσμευμένα για χρήση αποκλειστικά εντός της E.

- $E[f]$: Η διεργασία Ε αφού έχουν μετονομασθεί τα κανάλια της σύμφωνα με το f

3.1.2 Η Άλγεβρα διεργασιών π-λογισμός

Ο π-λογισμός [6] αποτελεί επέκταση της άλγεβρα διεργασιών CCS. Την επέκταση αυτή πρότειναν οι Robin Milner, Joachim Parrow και David Walker. Είναι ένα μαθηματικό μοντέλο διεργασιών όπου οι διασυνδέσεις τους μεταβάλλονται καθώς οι διεργασίες αλληλεπιδρούν. Το βασικό στοιχείο του π-λογισμού είναι η μεταφορά καναλιών επικοινωνίας μεταξύ διεργασιών. Αυτό κάνει τον λογισμό κατάλληλο για μοντελοποίηση κινητού υπολογισμού.

Συγκεκριμένα, στον π-λογισμό ένα κανάλι επικοινωνίας έχει διπλό ρόλο. Μπορεί να χρησιμοποιηθεί ως αντικείμενο και να σταλεί μέσω ενός άλλου καναλιού αλλά και να χρησιμοποιηθεί ως κανάλι για να σταλεί μέσω αυτού ένα άλλο αντικείμενο-κανάλι.
Πιο κάτω παρουσιάζεται η σύνταξη του :

Prefixes	a	::= $\bar{a}(x)$	Output
		$a(x)$	Input
		τ	Silent

Agents	P	::= 0	Nil
		$a.P$	Prefix
	|	$P + P$	Sum
	|	$P \mid P$	Parallel
	|	If $x = y$ then P	Match
	|	If $x \neq y$ then P	Mismatch
	|	$(\nu x)P$	Restriction
	|	A($y_1,...,y_n$)	Identifier

Definitions A($x_1,...,x_n$) = P (where $i \neq j \Rightarrow x_i \neq x_j$)

- Το $\bar{a}(x)$ αποτελεί ενέργεια εξόδου όπου το x στέλλεται μέσω του καναλιού α.

- Το $a(x)$ αποτελεί ενέργεια εισόδου όπου το x λαμβάνεται μέσω του καναλιού α.

- Η άδεια οντότητα 0 δεν μπορεί να κάνει οποιαδήποτε ενέργεια.

- Το $a.P$ αποτελεί τον κανόνα Prefix όπου εκτελείται το α και η διεργασία συνεχίζει ως P.

- Το άθροισμα $P + P$ υποδεικνύει μια οντότητα η οποία μπορεί να δράσει ως P ή P.

- Η παράλληλη σύνθεση $P \mid P$ υποδεικνύει το συνδυασμό της συμπεριφοράς του P και του P καθώς εκτελούνται παράλληλα. Μπορούν να λειτουργήσουν και ανεξάρτητα αλλά και να συντονιστούν μεταξύ τους και να ανταλλάξουν πληροφορίες.

- Στον κανόνα Match (If $x = y$ then P) η οντότητα θα συμπεριφερθεί σαν P εαν το $x = y$

- Στον κανόνα Mismatch (If $x \neq y$ then P) η οντότητα θα συμπεριφερθεί σαν P εαν το $x \neq y$

- Στον κανόνα Restriction $((\nu x)P)$ η οντότητα συμπεριφέρεται σαν P, όμως το x είναι τοπικό κάτι το οποίο σημαίνει ότι δεν μπορεί αμέσως να χρησιμοποιηθεί για την επικοινωνία του P και του περιβάλλοντος του.

- Στον κανόνα Identifier (A(y_1,..., y_n)) κάθε ένας Identifier έχει ένα ορισμό A(x_1,..., x_n) = P (where $i \neq j \Rightarrow x_i \neq x_j$) όπου κάθε x_i πρέπει να είναι διακριτό και η διαίθηση είναι ότι το A(y_1,..., y_n) συμπεριφέρεται σαν P όπου το x_i αντικαθιστά το y_i για κάθε i .

3.1.3 Η Άλγεβρα διεργασιών π-λογισμός με ομάδες

Ο π-λογισμός με ομάδες αποτελεί επέκταση του π-λογισμού και έχει επιπρόσθετα την έννοια της ομάδας, όπου μια ομάδα είναι ένας τύπος καναλιού. Στο [7], οι συγγραφείς έχουν δημιουργήσει μια στενή σχέση μεταξύ της ομάδας και της μυστικότητας δείχνοντας ότι όταν μια ομάδα γνωρίζει ένα μυστικό δεν δύναται αυτό το μυστικό να αποκαλυφθεί εκτός του αρχικού πεδίου εφαρμογής της ομάδας. Αυτό οφείλεται στο γεγονός ότι οι ομάδες δεν μπορούν να επικοινωνήσουν με διαδικασίες. Σε αυτό το σημείο δεν θα αναφερθούμε περαιτέρω στον π-λογισμό με ομάδες αφού θα αναλυθεί στο επόμενο υποκεφάλαιο η παραλλαγή αυτού με την οποία ασχοληθήκαμε σε αυτή την εργασία.

3.2 Ο λογισμός

Ο λογισμός που θα χρησιμοποιηθεί στην παρούσα εργασία βασίζεται στον π-λογισμό με ομάδες, με κάποιες όμως μικρές διαφοροποιήσεις. Εδώ έρχεται να προστεθεί η έννοια του συστήματος και της διεργασίας. Έστω ότι έχουμε τα G , G_1 τα οποία είναι ένα σύνολο απο ομάδες και N=a , b , x , y είναι ένα σύνολο απο ονόματα.

Με βάση αυτό, η σύνταξη του λογισμού κυμαίνεται σε δύο επίπεδα. Το πρώτο επίπεδο είναι το επίπεδο διεργασίας (process level) και ακολουθεί τη σύνταξη του π-λογισμού, ενώ το δεύτερο επίπεδο είναι το επίπεδο συστήματος (system level). Εδώ ακριβώς προστίθεται η έννοια των ομάδων και στα δύο επίπεδα, της διεργασίας και του συστήματος.

Πιο κάτω ακολουθούν οι ορισμοί των δύο επιπέδων: συστήματος και διεργασίας

P ::= x(y : T). P | $\bar{x}(z)$. P | (ν α : T) . P | $P_1|P_2$ | !P | 0

S ::= (ν G) P | (ν G) S | (ν α : T) S | $S_1|S_2$ | 0

Στο επίπεδο συστήματος έχουμε την δημιουργία ομάδας και σε διεργασίες (ν G) P και σε συστήματα (ν G) S όπου τα G δεσμεύονται στο P και στο S αντίστοιχα. Επιπλέον, όπως και στο π-calculus έχουμε τον κανόνα restriction (ν α : T) . P και (ν α : T) S όπου το α δεσμεύεται και πάλι στο στο P και στο S αντίστοιχα. Όσον αφορά τις ενέργειες εισόδου

έχουμε το x(y : T). P σε επίπεδο διεργασίας , όπου το y δεσμεύεται στο P. Και στα δύο επίπεδα ακολουθείται η παράλληλη σύνθεση όπως και στο π-calculus.

Για το σύνολο των ονομάτων που είναι ελεύθερα σε μια διεργασία P και ένα σύστημα S τα συμβολίζουμε με fn(P) και fn(S), ενώ για το σύνολο των ομάδων fg(P) και fg(S) αντίστοιχα.

Σε αυτό το σημείο χρειάζεται να ορίσουμε ένα σημασμένο σύστημα μετάβασης (labelled transition system) για το λογισμό προκειμένου να δώσουμε τη σημασιολογία του προτύπου.
Για να ορίσουμε ένα σημασμένο σύστημα μετάβασης πρώτα ορίζουμε ένα σύνολο απο σημάνσεις (labels):

L ::= τ | x(y) | $\bar{x}(y)$ | (ν y) $\bar{x}(y)$

Η σήμανση τ είναι εσωτερική ενέργεια , το x(y) είναι ενέργεια εισόδου και το $\bar{x}(y)$ είναι ενέργεια εξόδου.

Θα χρησιμοποιήσουμε τη μετασημειογραφία(metanotation) (F ::= P | S) για να ορίσουμε το σημασμένο σύστημα μεταβάσεων.

- $x(y{:}T).P \xrightarrow{x(z)} P\{z / y\}$ (In)

  Το x(y:T).P μπορεί να αλληλεπιδράσει με το x(z) και στη συνέχεια να συνεχίσει ως P αντικαθιστώντας το y με το z.

- $\bar{x}(z).P \xrightarrow{\bar{x}(z)} P$ (Out)

  Εκτελείται το $\bar{x}(z)$ και στη συνέχεια συνεχίζει ως P

- $\dfrac{F_1 \xrightarrow{l} F_{1'} bn(l) \cap fn(F_2) = 0}{F_1 \mid F_2 \xrightarrow{l} F_{1'} \mid F_2}$ (ParL)

  Στο ParL οι ενέργειες είναι σε παράλληλη σύνθεση και δεν υπάρχει σύγκρουση μεταξύ των δεσμευμένων (bn) ονομάτων της ενέργειας και των ελεύθερων ονομάτων της παράλληλης διεργασίας.

- $\dfrac{F_2 \xrightarrow{l} F_{2'} bn(l) \cap fn(F_1) = 0}{F_1 \mid F_2 \xrightarrow{l} F_1 \mid F_{2'}}$ (ParR)

  Παρόμοια με το ParL αφού είναι συμμετρικά

- $\dfrac{F \xrightarrow{l} F' \; x \notin fn(l)}{(\nu x : T)F \xrightarrow{l} (\nu x : T)F'}$ (ResN)

  Οι ενέργειες βρίσκονται κάτω απο τον τελεστή restriction (ν) και το restricted όνομα θεωρείται δεσμευμένο κατα την ενέργεια και όχι ελεύθερο.

- $\dfrac{F \xrightarrow{\bar{x}(y)} F'}{(\nu y : T)F \xrightarrow{(\nu y)\bar{x}(y)} F'}$ (Scope)

  Εαν το y είναι δεσμευμένο τότε και το $\bar{x}(y)$ είναι restricted

- $\dfrac{F \xrightarrow{l} F'}{(\nu G)F \xrightarrow{l} (\nu G)F'}$ (ResG)

  Ο κανόνας restriction μπορεί επίσης να χρησιμοποιηθεί και για ομάδες ResG παρόμοια με τον κανόνα ResN

- $\dfrac{F \equiv \alpha F'' \; F'' \xrightarrow{l} F'}{F \xrightarrow{l} F'}$ (Alpha)

- $\dfrac{P \xrightarrow{l} P'}{!P \xrightarrow{l} P' | !P}$ (Repl)

- $\dfrac{F_1 \xrightarrow{l1} F_{1'} F_2 \xrightarrow{l2} F_{2'} l_1 = \overline{l_2}}{F_1 \mid F_2 \xrightarrow{\tau} (\nu bn(l_1) \cup bn(l_2))(F_{1'} \mid F_{2'})}$ (Com)

  Ο κανόνας (Com) σημαίνει ότι οι διεργασίες μπορούν να επικοινωνήσουν μεταξύ τους με διπλές ενέργειες χρησιμοποιώντας την εσωτερική ενέργεια τ.

3.3 Σύστημα τύπων

Στην ενότητα αυτή ορίζουμε ένα σύστημα τύπων για το λογισμό, που βασίζεται στο σύστημα τύπων του [7]. Στο σύστημα τύπων θα έχουμε ένα σύνολο ομάδων (όπως στο [7]) και ένα σύνολο βασικών τύπων D, που θα τους συμβολίζουμε με t_i και θα αναφέρονται στους βασικούς τύπους του λογισμού, στους οποίους πρέπει να επιβληθούν οι απαιτήσεις ιδιωτικότητας. Επομένως, ένα όνομα μπορεί να είναι της μορφής t ή G[T] όπου το G είναι το όνομα του group και το T είναι ο τύπος του. Με βάση αυτά ένα τύπος κατασκευάζεται απο το ακόλουθο BNF:

$$T ::= t \mid G[T]$$

3.3.1 Αποφάσεις τύπων

Θα ονομάσουμε Γ τη διεπιφάνεια βάσει της οποίας γίνεται ο έλεγχος τύπων. Μέσα στο Γ θα συμπεριλάβουμε τα ονόματα ,τις ομάδες και τον τύπο του καθενός όπως πιο κάτω:

$$\Gamma ::= \varnothing \mid \Gamma.\, x : T \mid \Gamma.\, G$$

Θα ορίσουμε 3 typing judgements:

- $\Gamma \vdash x : T$

  Η μεταβλητή x θα έχει τύπο T

- $\Gamma \vdash P : \Delta$

  Η διαδικασία P είναι ορθά ορισμένη και παράγει μια διεπιφάνεια δικαιωμάτων Δ και συγκεκριμένα το Γ περιέχει τους τύπους των ονομάτων του P ενώ το Δ περιέχει τα δικαιώματα των ονομάτων του P για κάθε βασικό τύπο.

- $\Gamma \vdash S : \Theta$

  Το Γ παράγει τη διεπιφάνεια Θ, το οποίο περιέχει τα μέλη των ομάδων όλων των στοιχείων του S, καθώς και τα δικαιώματα που ασκούνται από κάθε συνιστώσα.

Οι διεπιφάνειες Δ και Θ ορίζονται ως εξής:

Prp ⊂ Eper και το Eper = {read, write, access}∪{disclose Gλ | G ∈ P, λ ∈{1,2,. . .} ∪{*}} το οποίο αποτελεί το σύνολο των δικαιωμάτων.

$$\Delta \quad ::= \varnothing \mid t : prp \,.\, \Delta \quad \Theta ::= \varnothing \mid <G_1 \cdot ... \cdot G_n , \Delta> \cdot \Theta$$

Όπως φαίνεται πιο πάνω τα δικαιώματα του συστήματος είναι το read, write, access, disclose. Με το δικαίωμα read, μια οντότητα του συστήματος μπορεί να διαβάσει τα ευαίσθητα δεδομένα, ενώ με το write μπορεί να γράψει και να τα τροποποιήσει. Με το δικαίωμα access μια οντότητα του συστήματος έχει πρόσβαση στα δεδομένα ενώ με το disclose G λ, μια οντότητα έχει το δικαίωμα να αποκαλύψει κάτι σε μια άλλη οντότητα μια ή περισσότερες φορές, ακόμα και άπειρες φορές.

3.3.2 Τύπος συστήματος

Πιο κάτω παρουσιάζονται οι κανόνες του συστήματος τύπων:

Typing System Rules

$$\frac{fg(T) \subseteq \Gamma}{\Gamma \cdot x : T \mid\!- x \rhd T} \text{ (Name)} \qquad \Gamma \mid\!- 0 \rhd \varnothing \text{ (Nil)}$$

$$\frac{\Gamma \cdot y : T \mid\!- P \rhd \Delta \;\; \Gamma \mid\!- x \rhd G[T]}{\Gamma \mid\!- x(y : T).P \rhd \Delta \oplus \Delta \overset{r}{T}} \text{ (In)} \qquad \frac{\Gamma \mid\!- P \rhd \Delta \;\; \Gamma \mid\!- x \rhd G[T] \;\; \Gamma \mid\!- y \rhd T}{\Gamma \mid\!- x < y > .P \rhd \Delta \oplus \Delta \overset{w}{T}} \text{ (Out)}$$

$$\frac{\Gamma \cdot P1 \rhd \Delta_1 \;\; \Gamma \mid\!- P_2 \rhd \Delta_2}{\Gamma \mid\!- P_1 \mid P_2 \rhd \Delta_1 \oplus \Delta_2} \text{ (ParP)} \qquad \frac{\Gamma \cdot S_1 \rhd \Theta_1 \;\; \Gamma \mid\!- S_2 \rhd \Theta_2}{\Gamma \mid\!- S_1 \mid S_2 \rhd \Theta_1 \oplus \Theta_2} \text{ (ParS)}$$

$$\frac{\Gamma \cdot x : T \mid\!- P \rhd \Delta}{\Gamma \mid\!- (\nu x : T)P \rhd \Delta} \text{ (ResNP)} \qquad \frac{\Gamma \cdot x : T \mid\!- S \rhd \Theta}{\Gamma \mid\!- (\nu x : T)S \rhd \Theta} \text{ (ResNS)}$$

$$\frac{\Gamma \cdot G \mid\!- P \rhd \Delta}{\Gamma \mid\!- (\nu G)P \rhd\!< G : \Delta >} \text{ (ResGP)} \qquad \frac{\Gamma \cdot G \mid\!- S \rhd \{< G_i, \Delta_i >\} i \in I}{\Gamma \mid\!- (\nu G)S \rhd \{< G, G_i : \Delta_i >\} i \in I} \text{ (ResGS)}$$

$$\frac{\Gamma \mid -P \triangleright \Delta}{\Gamma \mid -!^{!} P \triangleright \Delta} \text{ (Rep)}$$

Notations and auxiliary functions:

$$\Delta \overset{r}{T} = \begin{cases} \text{t: read} & \text{if T=t} \\ \text{t: access} & \text{if T=G[t]} \\ \varnothing & \text{otherwise} \end{cases}$$

$$\Delta \overset{w}{T} = \begin{cases} \text{t : write} & \text{if T=t} \\ \text{t : disclose G1} & \text{if T=G[t]} \\ \varnothing & \text{otherwise} \end{cases}$$

Επιπρόσθετα θα ορίσουμε ένα τελεστή για να συνδυάσουμε τις Δ διεπιφάνειες, $\oplus$, ο οποίος θα συνδυάζει τα δικαιώματα απο κάθε Δ διεπιφάνεια για κάθε βασικό τύπο. Γενικά αυτός ο τελεστής αποτελεί την ένωση των δικαιωμάτων, εκτός στην περίπτωση του δικαιώματος disclose , όπου σε αυτή την περίπτωση εαν τα δύο δικαιώματα αναφέρονται στην ίδια ομάδα,τότε προσθέτουμε τους αριθμούς των δικαιωμάτων disclose.

$$\Delta_1 \oplus \Delta_2 = \{\text{t}: prp_1 \oplus prp_2 \mid \text{t}: prp_1 \in \Delta_1, \text{t}: prp_2 \Delta_2\}$$

Όπου $prp_1 \oplus prp_2 = \{\text{p} \in prp_1 \cup prp_2 \mid \text{p} \neq \text{disclose G } \lambda\} \cup \{\text{disclose G } (\lambda_1 + \lambda_2) \mid \text{disclose G } \lambda_1 \in prp_1, \text{disclose G } \lambda 2 \in prp_2\}$

Ο κανόνας (Name) χρησιμοποιείται για τον τύπο των ονομάτων. Σχετικά με αυτό τον κανόνα απαιτείται τα ονόματα των ομάδων να βρίσκονται στη διεπιφάνεια Γ.

Ο κανόνας (In) χρησιμοποιείται για ενέργειες εισόδου. Εάν το interface Γ επεκταθεί με τον τύπο y παράγει το Δ σαν interface του P, καταλήγουμε στο συμπέρασμα ότι η διεργασία x(y).

P παράγει ένα interface όπου ο τύπος του T θα επεκταθεί με το $\Delta\overset{r}{T}$ όπου (i) αν το T είναι βασικός τύπος t τότε το Δ επεκτείνεται με το t : read δεδομένου ότι η διαδικασία διαβάζει ένα αντικείμενο του τύπου T, (ii) αν T = G [t] τότε το Δ επεκτείνεται με το t:access, δεδομένου ότι η διαδικασία έχει αποκτήσει πρόσβαση σε μια σύνδεση για τον βασικό τύπο t (iii) το Δ παραμένει ανεπηρέαστο διαφορετικά.

Παρόμοια με το (In) γίνεται και με τον κανόνα (Out). Υποθέτοντας ότι το y είναι τύπου T και το x είναι τύπου G [T] και το Δ είναι το interface για το P, τότε, το $\bar{x}(y)$.P παράγει ένα περιβάλλον το οποίο αποτελεί επέκταση του Δ με δικαιώματα $\Delta\overset{w}{T}$ όπου (i) αν το T είναι βασικός τύπος t τότε το Δ επεκτείνεται με το {t : write}, (ii) αν T = G [t] τότε το Δ επεκτείνεται με το {disclose G 1} και (iii) το Δ παραμένει ανεπηρέαστο διαφορετικά.

Το (ParP) αναφέρεται στην παράλληλη σύνθεση διεργασιών, όπου αυτός ο κανόνας χρησιμοποιεί τον τελεστή $\oplus$ για να συνδυάσει τα interface των διεργασιών P1, P2. Παρόμοια γίνεται και με τον κανόνα (ParS) όσον αφορά τα S1 και S2.

Το (ResNP) λέει ότι εαν ο τύπος μιας διεργασίας σε μια διεπιφάνεια Γ έχει το x:T τότε η διεργασία με το x γίνεται περιορισμένη στη διεπιφάνεια Γ. Παρόμοια γίνεται και στο (ResNS).

Στο (ResGP) όταν μια διεργασία P παράγει το interface Δ , τότε το σύστημα (ν G) P παράγει το interface < G, Δ> και για το (ResGS) όταν το S παράγει ένα interface $\{<\tilde{G}i\,,\ \Delta i>\}\ i \in I$, τότε η διεργασία (ν G) S παράγει το interface

$\{< G\cdot\ \tilde{G}i\,,\ \Delta i>\}\ u \in I$.

Σχετικά με τον κανόνα (Repl) , εαν μια διεργασία παράγει ένα interface Δ τότε η P! παράγει ένα interface Δ! όπου το Δ! είναι το ίδιο με το Δ με τη διαφορά ότι εαν κάποιος τύπος γίνεται disclose λ > 1 φορές, τότε στο Δ! θα γίνεται disclose * , δηλαδή θα μπορεί να αποκαλυφθεί άπειρες φορές.

3.4 Πολιτικές

Σε αυτή την ενότητα θα ορίσουμε μια απλή γλώσσα για να μπορούμε να εκφράσουμε τις απαιτήσεις ιδιωτικότητας ενός συστήματος μέσω του λογισμού. Μέσα απο αυτή τη γλώσσα θα εκφράσουμε θετικούς και αρνητικούς κανόνες οι οποίοι αναμένεται να υπάρχουν σε ένα υφιστάμενο σύστημα. Με τον όρο «θετικοί κανόνες» εννοούμε τι επιτρέπεται να συμβεί στο σύστημα, σε αντίθεση με τον όρο «αρνητικοί κανόνες» όντας τι δεν επιτρέπεται να συμβεί στο σύστημα.

Αυτοί οι νόμοι εφαρμόζονται σε ευαίσθητα δεδομένα που μπορεί να έχει ένα σύστημα, όπως για παράδειγμα η τοποθεσία, τα στοιχεία ενός ασθενή ή το τηλέφωνο. Τα ευαίσθητα αυτά δεδομένα ονομάζονται βασικοί τύποι. Σε αυτή την εργασία η γλώσσα που χρησιμοποιούμε θα εκφράζει τα επιτρεπόμενα και μη επιτρεπόμενα δικαιώματα ενός συνόλου από ομάδες για κάποιο βασικό τύπο.

Τα δικαιώματα που μπορεί να έχει μια πολιτική είναι τα εξής: PPer={read, write, access, nondisclose}$\cup$ {disclose G λ| G $\in$ G , λ $\in$ {1,2, …} $\cup${*}}. Από τα πιο πάνω δικαιώματα τα read, write, access και disclose αποτελούν θετικούς κανόνες και συγκεκριμένα το read εκφράζει ότι η πληροφορία μπορεί να διαβαστεί, το write ότι μπορεί να γραφτεί, το access ότι η πληροφορία μπορεί να είναι προσβάσιμη και το disclose G λ ότι η πληροφορία μπορεί να αποκαλυφθεί σε μέλη της ομάδας λ φορές ή άπειρες φορές.

Μια πολιτική μπορεί να οριστεί με το ακόλουθο BNF και pop $\subseteq$ PPer:

P ::= t >> H | P;P H::= ε | G : pop[H_i] i $\in$ I

Μια πολιτική έχει τη μορφή $t_1 >> H_1$;…; $t_n >> H_n$ όπου τα ti αποτελούν τους βασικούς τύπους (base types) ενώ τα H_i αποτελούν τις ιεραρχίες των δικαιωμάτων (permissions hierarchies), δηλαδή τα δικαιώματα για κάθε βασικό τύπο (base type). Μια ιεραρχία δικαιωμάτων (permission hierarchy H) έχει τη μορφή G : pop [H_i ,..... H_m], όπου η εξωτερική ομάδα έχει τα επιτρεπόμενα δικαιώματα pop και επίσης τα δικαιώματα της ιεραρχίας. Συνεπώς, μέσα απο αυτό προκύπτει ότι μια οντότητα η οποία συμμετέχει σε μια ομάδα (group) G έχει τα δικαιώματα pop και εαν επίσης ανήκει και σε μια άλλα ομάδα G_I όπου H_i = G_I : pop i[…] τότε επίσης έχει και αυτά τα δικαιώματα pop.

Με βάση την ιεραρχία H ορίζουμε τους απογόνους μιας ομάδας G ως εξής:

$$\text{desc}(G,H) = \begin{cases} \varnothing & \text{if } H=\varepsilon \\ \cup i \in I \text{ desc}(G, H_i) & \text{if } H=G': \text{pop}[H_i]\ i \in I,\ G \neq G' \\ \cup i \in I \text{ groups}(G, H_i) & \text{if } H=G: \text{pop}[H_i]\ i \in I \end{cases}$$

$$\text{groups}(H) = \begin{cases} \varnothing & \text{if } H=\varepsilon \\ \cup i \in I \text{ groups}(H_i) & \text{if } H=G: \text{pop}[H_i]\ i \in \end{cases}$$

$$\text{perms}(G,H) \begin{cases} \text{perms}(G,H) = \varnothing & \text{if } H=\varepsilon \\ \cup i \in I \text{ perms}(G,Hi) & \text{if } H=G': \text{pop}[Hi]\ i \in I,\ G \neq G' \\ \text{pop} & \text{if } H=G: \text{pop}[Hi]\ i \in I \end{cases}$$

Figure 2 – Permission extraction relation

$$\mathbf{H} \Vdash \varnothing \triangleright \varnothing \qquad \frac{Hi \Vdash -\tilde{G} \triangleright pop \oplus i,\ foralli \in I}{G : pop[Hi] i \in I \Vdash -G \cup \tilde{G} \triangleright pop \oplus G(\otimes i \in Ipopi)}$$

Αρχίζουμε απο την ρίζα της ιεραρχίας και συνδυάζουμε τα δικαιώματα με τα υπάρχουσα δικαιώματα των υπο-ιεραρχιών της πολιτικής.

Για την ιεραρχία:

pop_1 (.) $Gpop_2$ = {p ∈ $pop_1 \cup pop_2$ | p ≠ disclose G λ, p ≠ nondisclose}

∪{disclose G ($\lambda_1 + \lambda_2$) | disclose G λ1 ∈ pop_1, disclose G $\lambda_2 \in pop_2$}

∪ {nondisclose G | nondisclose ∈ pop_1, nondisclose G' ∉ pop_2}

∪ {nondisclose G' | nondisclose G' ∈ pop_2}

Σχετικά με την ιεραρχία, συνδυάζουμε τα δικαιώματα της με τα δικαιώματα των υπο-ιεραρχιών της ως εξής:

Εαν κάποιο δικαίωμα δεν είναι disclose G λ αλλά ούτε και nondisclose τότε συνενώνονται τα δικαιώματα. Για παράδειγμα, σε περίπτωση που έχουμε στην μια υπο-ιεραρχία disclose G λ_1

και στην άλλη υπο-ιεραρχία disclose G λ_2 τότε το δικαίωμα που προκύπτει είναι disclose G ($\lambda_1 + \lambda_2$) δηλαδή προστίθονται τα λ_1 και λ_2 των υπο-ιεραρχιών για να βγεί το συνολικό άθροισμα στο οποίο γίνεται το disclose σε κάποιο μέλος της ομάδας. Εαν έχουμε το δικαίωμα nondisclose στην μια ιεραρχία και στην άλλη δεν έχουμε nondisclose G' τότε το δικαίωμα που προκύπτει είναι nondisclose G, ενώ εαν έχουμε nondisclose G' στην δεύτερη ιεραρχία τότε το δικαίωμα που προκύπτει είναι nondisclose G'.

Για τα αδέλφια :

$pop_1 \otimes pop_2 = \{p \in pop_1 \cup pop_2 \mid p \neq \text{disclose G } \lambda\}$

$\cup \{\text{disclose G } (\lambda_1 + \lambda_2) \mid \text{disclose G } \lambda_1 \in pop_1, \text{disclose G } \lambda_2 \in pop_2\}$

Σχετικά με τα αδέλφια εαν ο ένας αδελφός έχει το δικαίωμα disclose G λ_1 και ο άλλος αδελφός disclose G λ_2 τότε το δικαίωμα που προκύπτει είναι disclose G (λ1+ λ_2) δηλαδή προστίθονται τα λ_1 και λ_2 των αδελφιών για να βγεί το συνολικό άθροισμα στο οποίο γίνεται το disclose σε κάποιο μέλος της ομάδας. Πιο κάτω ακολουθεί παράδειγμα με βάσει τον κανόνα Figure 2 – Permission extraction relation.

Ως παράδειγμα ας θεωρήσουμε την πιο κάτω πολιτική η οποία προστατεύει τα ευαίσθητα δεδομένα των ασθενών σε ένα νοσοκομείο το οποίο περιλαμβάνει τα τμήματα για χειρουργείο (Surgery), καρδιολογία (Cardiology), ψυχοθεραπεία (Psychotherapy) και όπου η ομάδα CarSurgeon αναφέρεται στους γιατρούς που είναι μέλη και στην ομάδα χειρουργείο (Surgery) και στην καρδιολογία (Cardiology).

P = t >> H

H = Hospital: {nondisclose} [Cardiology:{read, access, write, disclose Cardiology *} [CarSurgeon],
Surgery: { read, access, write, disclose Surgery *} [CarSurgeon],
Psychotherapy:{nondisclose, read, access, write }]

Ένας γιατρός ο οποίος είναι μέλος και στα δύο τμήματα καρδιολογίας και χειρουργείου θα πρέπει να διατίθεται σε αυτές τις ομάδες Hospital · Cardiology · Surgery · CarSurgeon. Αυτό περιλαμβάνει δύο μονοπάτια της ιεραρχίας και το σύνολο των δικαιωμάτων που θα προκύψει θα πρέπει να είναι ένας συνδυασμός των δύο μονοπατιών. Με βάση τις συναρτήσεις που αναφέρθηκαν πιο πάνω για την ιεραρχία και τα αδέλφια τα δικαιώματα συνενώθηκαν ως εξής:

H ⊩ Hospital · Cardiology · Surgery · CarSurgeon {nondisclose Hospital, read, access, write, disclose Cardiology *, disclose Surgery*}

H ⊩ Hospital · Psychotherapy {nondisclose Psychotherapy, read, access, write}

3.4.1 Ορισμός 2 – ορθά διατυπωμένη πολιτική

Ορισμός 2. Μια πολιτική ορίζεται ως ορθά διατυπωμένη εαν ικανοποιεί τα παρακάτω :

1. for all i, j, 1≤ i, j ≤ n, if i≠j then $t_i \neq t_j$
2. for all i, 1≤ i ≤ n, if G ∈ groups (H_i) then G ∉ desc(G, H_i)
3. for all i, 1≤ i ≤ n, if G ∈ groups (H_i), nondisclose ∈ perms (G, H_i) and disclose G'λ ∈ perms (G'', H_i), for some G'' ∈ desc(G, H_i), then G' ∈ desc(G, H_i)

Με βάση τα πιο πάνω μια πολιτική είναι ορθά διατυπωμένη εαν δεν υπάρχουν βασικοί τύποι με το ίδιο όνομα δηλαδή για κάθε βασικό τύπο έχουμε μόνο μια ιεραρχία (1), εαν για κάθε ομάδα μιας ιεραρχίας δεν υπάρχει το ίδιο όνομα ομάδας στους απογόνους της δηλαδή δεν υπάρχει κύκλος (2) και εαν σε κάποιο επίπεδο μιας ιεραρχίας κάποια ομάδα G έχει το δικαίωμα nondisclose και κάποια ομάδα G'' που είναι απόγονος του G έχει το δικαίωμα disclose G' τότε το G' πρέπει να είναι απόγονος του G (3).

Έστω ότι έχουμε την πιο κάτω πολιτική .Ο κανόνας 1 ικανοποιείται αφού ο μόνος βασικός τύπος που έχουμε είναι το examination. Ο κανόνας 2 ικανοποιείται αφού για κάθε ομάδα δεν έχουμε κάποιο απόγονο με το ίδιο όνομα. Ο κανόνας 3 ικανοποιείται μια και η ομάδα Hospital που έχει σαν δικαίωμα το nondisclose και η ομάδα GP που αποτελεί απόγονο της έχει σαν δικαίωμα disclose Doctor * όπου η ομάδα Doctor είναι απόγονος της ομάδας Hospital. Συνεπώς, αφού ικανοποιούνται και οι 3 κανόνες η πολιτική θεωρείται ορθά διατυπωμένη.

P = examination >> H
H ⊩ Hospital: nondisclose [Patient: read; Doctor:{read, write} [Surgeon:{empty} GP:disclose Doctor *]]

Για να μην θεωρείται η πολιτική ορθά διατυπωμένη έστω ότι έχουμε την πιο κάτω πολιτική όπου η ομάδα Doctor έχει ως δικαίωμα το nondisclose. Η ομάδα GP που αποτελεί απόγονο

της ομάδας Doctor έχει σαν δικαίωμα disclose Patient *. Εδώ παραβιάζεται ο κανόνας 3 λόγω του ότι η ομάδα Patient δεν είναι απόγονος του Doctor, επομένως η πολιτική δεν είναι ορθα διατυπωμένη.

P = examination >> H

H ╟ Hospital: nondisclose [Patient: read; Doctor:{read, write, nondisclose} [Surgeon:{empty} GP:disclose Patient *]]

3.4.3 Ικανοποίηση πολιτικής από το σύστημα

Ορισμός 3. Ας υποθέσουμε ότι το pop είναι ένα σύνολο απο δικαιώματα πολιτικής ιδιωτικότητας, το prp ένα σύνολο απο δικαιώματα πρόσβασης και το P μια πολιτική ιδιωτικότητας. Λέμε ότι το pop $\geq_P$ prp, εαν ισχύουν τα ακόλουθα:

1. if read $\in$ prp τότε το read $\in$ pop,
2. if write $\in$ prp τότε το write $\in$ pop,
3. if access $\in$ prp τότε το access $\in$ pop,
4. if disclose G $\lambda \in$ prp τότε το disclose G $\lambda' \in$ pop όπου $\lambda' \geq \lambda$
5. if nondisclose G $\in$ pop τότε για όλα τα disclose G' $\lambda \in$ prp, G' $\in$ desc(G, P)

Ο ορισμός αυτός αποτυπώνει τις προϋποθέσεις υπό τις οποίες ένα σύνολο δικαιωμάτων που ασκούνται από ένα πρόγραμμα, prp, είναι συμβατό με ένα σύνολο δικαιωμάτων που επιτρέπεται από ένα σύνολο δικαιωμάτων πολιτικής, pop. Σύμφωνα με τον ορισμό, εάν ένα από τα δικαιώματα ανάγνωσης, εγγραφής και η πρόσβαση είναι prp στη συνέχεια τα ίδια δικαίωμα πρέπει να υπάρχουν, επίσης, στην pop. Επιπλέον, εάν τα δικαιώματα του προγράμματος prp επιτρέπουν τη δημοσιοποίηση των δεδομένων σε ένα κανάλι της ομάδας G για λ φορές, τότε η pop θα πρέπει να επιτρέπει τέτοια αποκάλυψη για τουλάχιστον λ φορές.

Τέλος, αν το δικαιώμα της πολιτικής απαγορεύσει τη γνωστοποίηση των προσωπικών δεδομένων εκτός της ομάδας G, τότε κάθε λειτουργία αποκάλυψης που ασκείται στο πλαίσιο του προγράμματος, όπως η αποκάλυψη στο G' λ φορές , πρέπει να αποκαλύπτει μόνο στις ομάδες G' οι οποίες βρίσκονται κάτω από την ομάδα G στην ιεραρχία της ομάδας του προγράμματος που είναι και το G' πρέπει να είναι απόγονος του G. Μπορούμε τώρα να καθορίσει πότε μια διεπιφάνεια Θ η οποία ικανοποιεί την πολιτική ιδιωτικότητας P ως εξής:

Ορισμός 4. Λαμβάνοντας υπόψη ένα policy P = $t_i >> H_1 ;\ldots; t_n >> H_n$; και μια διεπιφάνεια Θ, ορίζουμε ότι Θ ικανοποιεί την P, αν για όλα τα $<\tilde{G}$, t : prp . Δ> ∈ Θ, ισχύει ότι t = t_i για κάποιο 1≤i≤ n και υπάρχει pop H_i ||- $\tilde{G}$ pop και pop ≥ P prp.

Σύμφωνα με τον ορισμό μια διεπιφάνεια Θ ικανοποιεί μια πολιτική P αν τους δοθεί οποιοδήποτε σύνολο δικαιωμάτων του προγράμματος $<\tilde{G}$, t : prp . Δ> ∈ Θ, το σύνολο των δικαιωμάτων, pop, που αποδίδεται από την πολιτική P σε ομάδες G για το βασικό τύπο t είναι τέτοια ώστε prp ≤ P prp.

Κεφάλαιο 4

Ανάλυση απαιτήσεων και προδιαγραφών λογισμικού

4.1 Εισαγωγή

Η ανάλυση απαιτήσεων αποτελεί τη συστηματική προσέγγιση για τον καθορισμό των λειτουργιών και των περιορισμών που πρέπει να εκτελεί ένα έργο της Πληροφορικής. Σε αυτό το σημείο θα καθοριστεί τι πρέπει να κάνει το σύστημα και όχι τον τρόπο με τον οποίο θα το κάνει.

Η ανάλυση απαιτήσεων, για οποιοδήποτε έργο Πληροφορικής, προσφέρει στο σύνολο των εμπλεκομένων (προγραμματιστών, αναλυτών, διευθυντών, πελατών, χρηστών) σημαντική ευκαιρία για καλύτερη οργάνωση, προγραμματισμό και διαχείριση του έργου.

Επιπλέον, η ανάλυση απαιτήσεων συμπιέζει σημαντικά το κόστος υλοποίησης του έργου, καθώς εξασφαλίζει ότι η ολοκλήρωσή του θα γίνει βάσει χρονοδιαγράμματος, πλαισίου και συγκεκριμένων προδιαγραφών. Αυτό έχει μεγάλη αξία, γιατί όχι σπάνια σε έργα για τα οποία δεν έχει προηγηθεί ανάλυση απαιτήσεων παρατηρούνται φαινόμενα όπως:

- Το πρόγραμμα αποδεικνύεται ότι έχει αρκετά ελαττώματα (bugs), τα οποία πρέπει να διορθωθούν με "μπαλώματα" και τη συγγραφή καινούργιου κώδικα.
- Η εφαρμογή που δημιουργήθηκε αποκλειστικά για ένα συγκεκριμένο πελάτη

αποδεικνύεται ελλειπής στην πράξη και χρήζει βελτίωσης.

- Ο πελάτης συνειδητοποίησε την τελευταία στιγμή ότι θέλει το ηλεκτρονικό του κατάστημα να περιέχει και "κάτι ακόμα".

Ασφαλώς, σε αυτά τα τρία παραδείγματα θα μπορούσαν να προστεθούν και πολλά άλλα. Αυτό στο οποίο οφείλουμε να εστιάσουμε είναι ότι η προσθήκη, η βελτίωση, η "επισκευή" σε ένα έργο πληροφορικής αποτελούν ιδιαίτερα κοστοβόρα και χρονοβόρα υπόθεση. Δεν είναι λίγες οι φορές που η προσθήκη κάποιας νέας εφαρμογής απαιτεί την εκ θεμελίων αναδόμηση και ανασυγκρότηση του προγράμματος, γεγονός που ισοδυναμεί με πολλές εργατοώρες και ασφαλώς κόστος. Για το ζήτημα αυτό διάφορες έρευνες έχουν επισημάνει ότι κάθε εργατοώρα που επενδύεται στην ανάλυση απαιτήσεων εξοικονομεί δεκάδες εργατοώρες για μετέπειτα εργασίες βελτίωσης, αναπροσαρμογής. Ένα ρητό που προφέρεται συχνά από τα χείλη των προγραμματιστών είναι ότι καλύτερα να ξαναγράψεις κώδικα δέκα φορές, παρά να διορθώσεις μία.

4.2 Σκοπός

Η παρούσα ενότητα σχετίζεται με την προδιαγραφή απαιτήσεων για το λογισμικό που αναπτύχθηκε σχετικά με την ιδιωτικότητα. Σκοπός του λογισμικού είναι η ενημέρωση για το εαν μια πολιτική ιδιωτικότητας ικανοποιείται απο ένα σύστημα και αντίστροφα, εαν ένα σύστημα ικανοποιά μια πολιτική ιδιωτικότητας. Το εργαλείο απευθύνεται σε όλα τα άτομα, κάθε ηλικίας και φύλου.

Το λογισμικό θα μπορεί να ενημερώνει το χρήστη σχετικά με το εαν ένα σύστημα με τύπους ικανοποιά μια πολιτική. Επίσης, σύμφωνα με την πολιτική ιδιωτικότητας και το σύστημα, θα μπορεί να εμφανίζει αρχικά ποιά δικαιώματα έχει το σύστημα και στη συνέχεια να ειδοποιεί σχετικά με την πολιτική εαν είναι ορθά διατυπωμένη δηλαδή εαν ακολουθεί τους κανόνες που αναφέρθηκαν σε προηγούμενο υποκεφάλαιο. Εαν η πολιτική είναι ορθά διατυπωμένη τότε θα παρουσιάζει και τα δικαιώματα της πολιτικής, εαν όχι τότε θα ειδοποιά ότι η πολιτική δεν είναι ορθά διατυπωμένη και θα αναφέρει ποιός κανόνας της πολιτικής παραβιάστηκε. Επίσης, αφού γίνει η εξαγωγή των δικαιωμάτων της πολιτικής και του συστήματος, στο τέλος θα εμφανίζει τη σύγκριση των δικαιωμάτων των δύο και θα εξάγεται το αποτέλεσμα σχετικά με το εαν το σύστημα ικανοποιά την πολιτική ή όχι.

Το λογισμικό δεν θα μπορεί να προχωρήσει στην διεξαγωγή του αποτελέσματος εαν η πολιτική δεν είναι ορθά διατυπωμένη και εαν ο έλεγχος τύπου του συστήματος δεν ακολουθεί τους κανόνες του συστήματος. Το λογισμικό θα είναι αποκλειστικά για

προσωπική χρήση και δεν θα δημοσιοποιεί οποιαδήποτε στοιχεία (xml αρχεία) που θα εισάγει ο χρήστης, ούτε θα απαιτεί αναγνώριση συγκεκριμένου χρήστη (αυθεντικοποίηση).

4.3 Γενική περιγραφή

Το εργαλείο που δημιουργήθηκε έχει ως σκοπό όπως αναφέρθηκε να απαντά στο ερώτημα εαν ένα σύστημα ικανοποιά μια πολιτική ιδιωτικότητας. Συγκεκριμένα, το εργαλείο αυτό θα λαμβάνει ως είσοδο δύο αρχεία xml , ένα το οποίο θα είναι το σύστημα και το δεύτερο θα είναι η πολιτική ιδιωτικότητας.

Αφού διεξαχθούν τα δικαιώματα του συστήματος και της πολιτικής, σε αυτό το σημείο γίνεται σύγκριση των δύο για να απαντηθεί εαν το σύστημα ικανοποιά την πολιτική. Το εργαλείο απαντά θετικά εαν όλα τα δικαιώματα του συστήματος υπάρχουν και σαν δικαιώματα της πολιτικής, δηλαδή το σύστημα δεν πρέπει να κάνει περισσότερα απο αυτά που κάνει και η πολιτική.

4.4 Λειτουργίες

Με βάση την περιγραφή του εργαλείου το εργαλείο πρέπει να ικανοποιεί τις τρείς πιο κάτω λειτουργίες οι οποίες διακρίνονται ως εξής:

- **Λειτουργία 1: parse xml αρχείου συστήματος και διεξαγωγή των δικαιωμάτων του**

Η 1η λειτουργία που θα κάνει το εργαλείο θα είναι ως εξής: Το εργαλείο θα παίρνει το xml αρχείο για το σύστημα που θα δώσει ο χρήστης ως είσοδο. Θα δημιουργείται το dom tree του συστήματος αφού θα αναλυθεί μέσω του Java Dom Parser και ακολούθως με βάση το tree, το εργαλείο θα εμφανίζει τα δικαιώματα του συστήματος με βάση τον τύπο του συστήματος.

- **Λειτουργία 2: parse xml αρχείου πολιτικής ιδιωτικότητας και έλεγχος εαν είναι ορθά διατυπωμένη.**

Η 2η λειτουργία που θα κάνει το εργαλείο θα είναι ως εξής: Το εργαλείο θα παίρνει το xml αρχείο για την πολιτική ιδιωτικότητας που θα δώσει ο χρήστης ως είσοδο. Θα δημιουργείται

το dom tree της πολιτικής μέσω του Java Dom Parser και ακολούθως με βάση το tree, θα αποφασίζει εαν η πολιτική ιδιωτικότητας είναι ορθά διατυπωμένη. Συγκεκριμένα, για να απαντήσει το εργαλείο ότι η πολιτική είναι ορθά διατυπωμένη θα πρέπει να ικανοποιούνται οι 3 κανόνες για το εαν μια πολιτική είναι ορθά διατυπωμένη που έχουν αναφερθεί σε προηγούμενο υποκεφάλαιο (3.4.1 Ορθά διατυπωμένη πολιτική)

- **Λειτουργία 3: Διεξαγωγή δικαιωμάτων της πολιτική ιδιωτικότητας**

Η 3η λειτουργία που θα κάνει το εργαλείο θα είναι ως εξής: Αφού κριθεί ότι η πολιτική ιδιωτικότητας είναι well-formed θα γίνεται διεξαγωγή των δικαιωμάτων της πολιτικής με βάση το 3.4.1 Figure 2-Permission Extraction Relation .

4.5 Γενικοί Περιορισμοί

- Το λογισμικό θα υλοποιηθεί στη γλώσσα προγραμματισμού Java μέσω του εργαλείου ολοκληρωμένης ανάπτυξης Eclipse IDE

- Ο χρήστης θα έχει προκαθορισμένες επιλογές, έτσι ώστε να ελαχιστοποιηθούν οι έλεγχοι εγκυρότητας των εισαγόμενων δεδομένων.

- Υπάρχει ελάχιστη συχνότητα εμφάνισης αστοχιών (σφαλμάτων), αφού οι εισόδοι του χρήστη είναι προκαθορισμένες λόγω του οδηγού συγγραφής για το xml αρχείο της πολιτκή και του συστήματος που θα παίρνει το πρόγραμμα ως είσοδος.

- Η σημαντικότερη λειτουργία του λογισμικού είναι το αποτέλεσμα της απόφασης για το σχετικά με το εαν ένα σύστημα ικανοποιά μια πολιτική καθώς και τα ενδιάμεσα βήματα που παρουσιάζονται στο χρήστη όπου φαίνονται ξεχωριστά τα δικαιώματα του συστήματος και της πολιτικής.

4.6 Ειδικές απαιτήσεις

4.6.1 Εξωτερικές απαιτήσεις διεπαφής

Για τη λειτουργία του εργαλείου, ο χρήστης απαιτείται να εισάγει τα ακόλουθα δεδομένα, τα οποία θα συνδυάζονται μεταξύ τους για καθορισμό των δικαιωμάτων του καθενός αλλά και για τη σύγκριση τους προκειμένου να δούμε εαν το σύστημα ικανοποιά την πολιτική. Θα υπάρχουν οι ανάλογες έξοδοι ανάλογα με τις εισόδους του χρήστη. Ο χρήστης θα μπορεί αναπάσα στιγμή να μεταβάλλει τα περιεχόμενα της πολιτικής ή του συστήματος, μέσω τροποποίησης του xml αρχείου.

- Ο χρήστης θα δημιουργά ένα xml αρχείο για την πολιτική ακολουθώντας την γραμματική της πολιτικής, το οποίο θα παίρνει ως είσοδο το πρόγραμμα.
- Ο χρήστης θα δημιουργά ένα xml αρχείο για το σύστημα, ακολουθώντας τους κανόνες type check για το σύστημα που αναφέρθηκαν σε προηγούμενο κεφάλαιο.

4.6.2 Χαρακτηριστικά λογισμικού

Αξιοπιστία

Για να διασφαλιστεί η αξιοπιστία του εργαλείου εφαρμόζονται είσοδοι περιορισμένης επιλογής (για αποφυγή λανθασμένου τύπου εισόδου) όπως φαίνεται και από τους οδηγούς για τη συγγραφή των αρχείων εισόδου που θα παίρνει το εργαλείο. Γι’ αυτό οι περιπτώσεις σφαλμάτων απαλείφονται με την προυπόθεση ότι ο χρήστης ακολούθησε σωστά τον οδηγό. Στις περιπτώσεις κατά τις οποίες ο χρήστης εισάγει δεδομένα για τα οποία δεν είναι έγκυρα με βάση τους οδηγούς, θα γίνονται οι κατάλληλοι έλεγχοι για επιτρεπόμενες τιμές (τύπος και εύρος τιμών), όπως επίσης σε περίπτωση που ο χρήστης δεν συμπληρώσει κάποιο πεδίο εισόδου θα εμφανίζεται στην οθόνη σχετικό μήνυμα σφάλματος λόγω της δομής του λογισμικού, με επακόλουθο τον τερματισμό της λειτουργία του όλου προγράμματος. Επίσης, σε περίπτωση τερματισμού κάποιας συγκεκριμένης λειτουργίας (έλεγχος εαν η πολιτική είναι ορθά διατυπωμένη), θα εμφανίζεται μήνυμα το οποίο θα ενημερώνει για πιο λόγο το πρόγραμμα τερμάτισε.

Διαθεσιμότητα

Το εργαλείο υλοποιείται αποκλειστικά στο περιβάλλον της Eclipse και δεν επικοινωνεί με βάση δεδομένων ή κάτι άλλο γι’ αυτό η περίπτωση να μην είναι διαθέσιμο είναι μηδαμινή.

Ασφάλεια

Η εγκυρότητα των δεδομένων εξασφαλίζεται με τους ελέγχους εισόδων των περιεχομένων των αρχείων xml αφού γίνουν parse μέσω του Java Dom Parser.

Με την ατομική χρήση του εργαλείου, το εργαλείο δεν θα απαιτεί οποιανδήποτε ταυτοποίηση χρήστη (login).

Λόγω του ότι δεν υπάρχει χρήση και συνδεσιμότητα του εργαλείου στο Διαδίκτυο, δεν δύναται να υπάρξει περίπτωση κακόβουλης πρόσβασης .

Συντηρησιμότητα

Η δημιουργία του εργαλείου θα είναι βασισμένη σε αντικειμενοστρεφή προγραμματισμό άρα αυτό εξασφαλίζει τη χαμηλή πολυπλοκότητα του. Ο κώδικας θα είναι αρκετά σχολιασμένος ώστε να είναι εύκολα κατανοητός από μετ' έπειτα συντηρητές. Το εργαλείο θα δημιουργηθεί με στόχο την ευχρηστία και την απλότητα, ούτως ώστε οι μελλοντικές αλλαγές να μην είναι αναγκαίες.

Φορητότητα

Το εργαλείο είναι μεταφέρσιμο αφού είναι υλοποιημένο με την γλώσσα προγραμματισμού Java η οποία λειτουργεί σε κάθε είδους πλατοφόρμα και λειτουργικό.

Επεκτασιμότητα

Η σημαντικότερη απαίτηση ήταν το εργαλείο να είναι εύχρηστο και επεκτάσιμο, στοιχείο το οποίο έχει.

Κεφάλαιο 5

Υλοποίηση και Έλεγχος

5.1 Εισαγωγή

Κατά της φάση της υλοποίησης γίνεται η μετατροπή των απαιτήσεων και των λειτουργιών σε προγραμματιστικό επίπεδο. Σε αυτό το σημείο δίνεται έμφαση στους ελέγχους που πρέπει να γίνουν έτσι ώστε να επικυρωθεί το λογισμικό και να πιστοποιηθεί ότι πληρεί τις απαιτήσεις.

Σε αυτή την ενότητα παρουσιάζεται το εργαλείο στο οποίο αναπτύχθηκε το λογισμικό καθώς και η γλώσσα προγραμματισμού που χρησιμοποιήθηκε. Επιπρόσθετα, παρουσιάζονται οι λειτουργίες που υλοποιήθηκαν καθώς και ο τρόπος που έγιναν και τέλος οι ελέγχοι που πραγματοποιήθηκαν οι οποίοι δείχνουν την ορθότητα του λογισμικού.

5.2 Εργαλεία

Ένα μεγάλο πλήθος εργαλείων έχει κατασκευαστεί για την διευκόλυνση των προγραμματιστών στη φάση της κωδικοποίησης. Τα πρωταρχικά εργαλεία στη φάση αυτή είναι οι γλώσσες προγραμματισμού και γύρω απο αυτές περιστρέφεται ένας μεγάλος αριθμός επιμέρους εργαλείων. Η κατάλληλη επιλογή εργαλείου αποτελεί καθοριστικό παράγοντα στην επίτευξη των στόχων του λογισμικού.

Τα εργαλεία που έχω επιλέξει για την υλοποίηση του λογισμικού θεωρώ ότι παρέχει την ευελιξία για συντήρηση, επεκτασιμότητα και γενικά μετατροπές που μπορεί να προκύψουν είτε για να γίνουν αλλαγές, είτε για μελλοντικές εργασίες.

5.2.1 Eclipse

Στον προγραμματισμό ηλεκτρονικών υπολογιστών, η Eclipse αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE). Περιέχει μια βάση του χώρου εργασίας και ένα επεκτάσιμο σύστημα plug-in για την προσαρμογή του περιβάλλοντος. Γράφει κυρίως στη γλώσσα προγραμματισμού Java. Ωστόσο η Eclipse μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών. Με τη βοήθεια των διαφόρων plug-ins, η Eclipse μπορεί επίσης να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών σε άλλες γλώσσες προγραμματισμού όπως: Ada, ABAP, C, C ++, COBOL, Fortran, Haskell, JavaScript, Lasso, Lua, Φυσικό, Perl, PHP, Prolog, Python, R, Ruby (συμπεριλαμβανομένων των Ruby on Rails πλαίσιο), Scala, Clojure, Groovy, Scheme, και Erlang.

Μπορεί επίσης να χρησιμοποιηθεί για την ανάπτυξη πακέτων για το λογισμικό Mathematica. Τα περιβάλλοντα ανάπτυξης περιλαμβάνουν τα εργαλεία ανάπτυξης Eclipse Java (JDT) για Java και τη Scala, το Eclipse CDT για την C / C ++ και Eclipse PDT για την PHP, μεταξύ άλλων.

Η αρχική βάση κώδικα προήλθε από την IBM VisualAge. Η Eclipse software development kit (SDK), η οποία περιλαμβάνει τα εργαλεία ανάπτυξης της Java, προορίζεται για προγραμματιστές Java. Οι χρήστες μπορούν να επεκτείνουν τις ικανότητές τους με την εγκατάσταση plug-ins για την πλατφόρμα Eclipse, όπως είναι η ανάπτυξη εργαλείων για άλλες γλώσσες προγραμματισμού, και μπορεί να γράψει και να συμβάλει με τις δικές τους μονάδες plug-in.

Εκδίδεται σύμφωνα με τους όρους της Eclipse Public License, Eclipse SDK είναι ελεύθερο και ανοικτού κώδικα λογισμικό (αν και είναι ασυμβίβαστη με την GNU General Public License). Ήταν ένα από τα πρώτα IDEs που τρέχει κάτω από GNU Classpath.

5.2.2 Οδηγός για τη συγγραφή xml αρχείου για το σύστημα

Αρχικά ο χρήστης θα πρέπει να γράψει σε ένα σημαντήρα που θα πρέπει να ονομάζεται <g_attributes> τα κανάλια επικοινωνίας, όπως για παράδειγμα : <g_attributes>fee:ETP[Fee],send:ETP[ETP[Fee]]</g_attributes> . Ακολούθως τον επόμενο σημαντήρα πρέπει να τον ονομάσει <base_type> και μέσα να γράψει τους βασικούς τύπους διαχωρίζοντας τους με space όπως για παράδειγμα <base_type> Loc Fee </base_type>. Τέλος, θα πρέπει να μεταφράσει τους κανόνες του π-λογισμού με ομάδες σε μορφή xml αρχείου. Για την μετάφραση αυτή ο χρήστης θα πρέπει να ακολουθήσει ακριβώς τους πιο κάτω κανόνες.

```
P + Q: <sumP>+
          <process>P</process >
          < process >Q</ process >
        </sumP>
```

Εκτελείται η διεργασία P ή Q. Ο αρχικός σημαντήρας <sumP> δηλώνει το όνομα του κανόνα, ενώ τα παιδιά του που είναι οι σημαντήρες <process> δηλώνουν τις 2 υποψήφιες διεργασίες που πρόκειται να εκτελεστεί μια απο τις δυο.

```
P | Q: <parP>
          < process>P</ process>
          < process>Q</ process>
       </parP>
```

Εκτελείται η διεργασία P παράλληλα με την Q. Ο αρχικός σημαντήρας <parP> δηλώνει το όνομα του κανόνα, ενώ τα παιδιά του που είναι οι σημαντήρες <proccess> δηλώνουν τις 2 διεργασίες που πρόκειται να εκτελεστούν παράλληλα.

$x(y:T)P$: <in>
<subj>x</subj>
<obj>y</obj>
<type>T</type>
< process>P</ process>
</in>

Ο αρχικός σημαντήρας <in> δηλώνει το όνομα του κανόνα. Το <subj> δηλώνει το κανάλι το οποίο θα λάβει το αντικείμενο-κανάλι (<obj>) και ο σημαντήρας <type> δηλώνει τον τύπο του αντικείμενου-καναλιού. Αφού εκτελεστεί ο κανόνας έχουμε τον σημαντήρα <proccess> ο οποίος δηλώνει ότι μπορεί να ακολουθήσει στη συνέχεια οποιοσδήποτε άλλος κανόνας απο τους υπάρχοντες που σχετίζονται με διεργασίες.

$\bar{x}(z).P$: <out>
<subj>x</subj>
<obj>z</obj>
< process >P</ process >
</out>

Ο αρχικός σημαντήρας <out> δηλώνει το όνομα του κανόνα. Το <subj> δηλώνει το κανάλι το οποίο μέσω αυτού θα σταλεί ένα άλλο αντικείμενο (<obj>). Αφού εκτελεστεί ο κανόνας έχουμε τον σημαντήρα <proccess> ο οποίος δηλώνει ότι μπορεί να ακολουθήσει στη συνέχεια οποιοσδήποτε άλλος κανόνας από τους υπάρχοντες που σχετίζονται με διεργασίες.

$(va:T)P$: <resNP>
<name>a</name>
<type>T</type>
< process >P</ process >
</ resNP >

Ο αρχικός σημαντήρας <resNP> δηλώνει το όνομα του κανόνα. Το <name> δηλώνει το όνομα του δεσμευμένου καναλιού ενώ το <type> τον τύπο του. Αφού εκτελεστεί ο κανόνας έχουμε τον σημαντήρα <proccess> ο οποίος δηλώνει ότι μπορεί να ακολουθήσει στη συνέχεια οποιοσδήποτε άλλος κανόνας από τους υπάρχοντες που σχετίζονται με διεργασίες.

```
!P: <repl>!
      < process >P</ process >
    </repl>
```

Ο αρχικός σημαντήρας <repl> δηλώνει το όνομα του κανόνα. Αφού εκτελεστεί ο κανόνας έχουμε τον σημαντήρα <proccess> ο οποίος δηλώνει ότι μπορεί να ακολουθήσει στη συνέχεια οποιοσδήποτε άλλος κανόνας από τους υπάρχοντες που σχετίζονται με διεργασίες.

```
0:<Nil>0</Nil>
```

Ο αρχικός σημαντήρας <Nil> δηλώνει το όνομα του κανόνα. Εδώ δενέχουμε τον σημαντήρα <proccess> ο οποίος δηλώνει ότι μπορεί να ακολουθήσει στη συνέχεια οποιοσδήποτε άλλος κανόνας από τους υπάρχοντες όπως είχαμε και στους προηγούμενους κανόνες αφού με το Nil έχουμε τερματισμό.

```
(νG)P :  <resGP>
           <group>G</group>
           < process >P</ process>
         </resGP>
```

Ο αρχικός σημαντήρας <resGP> δηλώνει το όνομα του κανόνα. Το <group> δηλώνει το όνομα της ομάδας. Αφού εκτελεστεί ο κανόνας έχουμε τον σημαντήρα <proccess> ο οποίος δηλώνει ότι μπορεί να ακολουθήσει στη συνέχεια οποιοσδήποτε άλλος κανόνας από τους υπάρχοντες που σχετίζονται με διεργασίες.

```
(νG)S :  <resGS>
           <group>G</group>
           < system >S</ system>
         </resGS>
```

Ο αρχικός σημαντήρας <resGS> δηλώνει το όνομα του κανόνα. Το <group> δηλώνει το όνομα της ομάδας. Αφού εκτελεστεί ο κανόνας έχουμε τον σημαντήρα <system> ο οποίος δηλώνει ότι μπορεί να ακολουθήσει στη συνέχεια οποιοσδήποτε άλλος κανόνας από τους υπάρχοντες που σχετίζονται με συστήματα.

```
(va:T)S : <resNS >
            <name>a</name>
            <type>T</type>
            < system >S</ system>
          </resNS >
```

Ο αρχικός σημαντήρας <resNS> δηλώνει το όνομα του κανόνα. Το <name> δηλώνει το όνομα του δεσμευμένου καναλιού ενώ το <type> τον τύπο του. Αφού εκτελεστεί ο κανόνας έχουμε τον σημαντήρα <system> ο οποίος δηλώνει ότι μπορεί να ακολουθήσει στη συνέχεια οποιοσδήποτε άλλος κανόνας από τους υπάρχοντες που σχετίζονται με συστήματα.

```
S1|S2: <parS>
            <system>S1</ system>
            < system>S2</ system>
          </parS>
```

Παράλληλη εκτέλεση των συστημάτων S1 και S2. Ο αρχικός σημαντήρας <parS> δηλώνει το όνομα του κανόνα, ενώ τα παιδιά του που είναι οι σημαντήρες <system> δηλώνουν τα 2 συστήματα που πρόκειται να εκτελεστούν παράλληλα.

<u>Παράδειγμα</u>

Έστω ότι έχουμε το σύστημα ETP(Electronic Pricing Authority) ως παράδειγμα και τους δύο βασικούς τύπους Loc και Fee και Tl = ETP[Loc], Tr = Car[Tl], Tpa = ETP[Tl], Tx = ETP[Tl] and Tsc = ETP[Tx].

O = ! read(loc : Tl).topa <loc>.0

| ! spotcheck(s : Tx): read(ls : Tl). s<ls>.0

L = !(ν newl : Tl) read<newl>.0

A = ! topa(z : Tl).z(l : Loc).0

|! send<fee>.0

| !(ν x : Tx) spotcheck<x>.x(y : Tl).y(ls : Loc).0

Με βάση τους πιο πάνω κανόνες για τη συγγραφή xml κάθε κομμάτι του συστήματος μεταφράζεται ως εξής:

- O = ! read(loc : Tl).topa <loc>.0

  | ! spotcheck(s : Tx): read(ls : Tl). s<ls>.0

  <group>OBE</group>

```
<parP>
  <process>
    <repl>!
      <process>
        <in>
          <subj>read</subj>
          <obj>loc</obj>
          <type>ETP[Loc]</type>
           <process>
             <out>
             <subj>topa</subj>
             <obj>loc</obj>
             <process>
               <nil>0</nil>
             </process> </out></process></in> </process> </repl></process>
           <process>
          <repl>!
          <process>
          <in>
            <subj>spotcheck</subj>
           <obj>s</obj>
          <type>ETP[ETP[Loc]]</type>
          <process>
           <in>
             <subj>read</subj>
             <obj>ls</obj>
             <type>ETP[Loc]</type>
             <process>
               <out>
              <subj>s</subj>
              <obj>ls</obj>
              <process>
             <nil>0</nil>
           </process> </out> </process> </in> </process> </in> </process> </repl> </process>
</parP> </resGP> </process>
```

- L = !(ν newl : Tl) read<newl>.0

```
<repl>!
  <process>
    <resNP>
      <name>newl</name>
      <type>ETP[Loc]</type>
      <proccess>
        <out>
          <subj>read</subj>
          <obj>newl</obj>
          <process>
            <nil>0</nil>
          </process>
         </out>
        </process>
       </resNP>
      </process>
     </repl>
```

5.2.3 Οδηγός για τη συγγραφή xml αρχείου για την πολιτική

P ::= t >>H | P;P H::= ε | G : pop [H_i] i ∈ I

Ο χρήστης θα πρέπει να μεταφράσει την πιο πάνω γραμματική της Πολιτικής ως εξής:

t = <base_type> H </base_type>

H ::= ε | G : pop [H_i] i ∈ I = empty| G

ε = empty

G = <gph>

<group> όνομα ομάδας . pop</group>

<gph>

pop = <perm>p</perm>

p = <read></read>.p | <write></write> . p | <nondisclose></nondisclose> . p | <disclose>Gλ</disclose>.p | ε | h | 0

h = <hierarchy> H <hierarchy>

Έστω ότι έχουμε την πιο κάτω πολιτική:

Loc>>ETP:nondisclose[

Car:[

OBE:{access,disclose ETP2}

GPS:{disclose Car*}

SC:{access,read}]

PA:{access,read}]

Με βάση τον πιο πάνω οδηγό η πολιτική θα μεταφραστεί ως εξής:

```
<base_type>Loc
  <gph>
   <group>ETP</group>
   <perm>
    <nondisclose></nondisclose>
   </perm>
   <hierarchy>
    <gph>
     <group>Car</group>
     <perm></perm>
     <hierarchy>
      <gph>
```

```
          <group>OBE</group>
          <perm>
            <access></access>
            <disclose>ETP*</disclose>
          </perm>
          <hierarchy>empty</hierarchy>
        </gph>
        <gph>
          <group>GPS</group>
          <perm>
            <disclose>ETP*</disclose>
          </perm>
          <hierarchy>empty</hierarchy>
        </gph>
      </hierarchy>
    </gph>
    <gph>
     <group>PA</group>
     <perm>
       <access></access>
       <read></read>
     </perm>
     <hierarchy>empty</hierarchy>
   </gph>
  </hierarchy>
 </gph>
</base_type>
```

5.3 Γλώσσα προγραμματισμού Java

Η Java είναι μια γενικού σκοπού γλώσσα προγραμματισμού υπολογιστών που είναι αντικειμενοστρεφείς και χρησιμοποιά κλάσεις. Σκοπός της είναι να παρέχει στους προγραμματιστές εφαρμογών το "write once, run anywhere» που σημαίνει ότι ο μεταγλωττισμένος κώδικας Java μπορεί να τρέξει σε όλες τις πλατφόρμες που υποστηρίζουν Java, χωρίς την ανάγκη για επαναμεταγλώττιση. Οι εφαρμογές Java συνήθως συγκεντρώνονται σε bytecode και μπορεί να τρέξει σε οποιαδήποτε εικονική μηχανή (JVM) ανεξάρτητα από την αρχιτεκτονική του υπολογιστή. Από το 2015, η Java είναι μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού σε χρήση, ιδίως για web εφαρμογές πελάτη-εξυπηρετητή. Η Java αρχικά αναπτύχθηκε από τον James Gosling στο Sun Microsystems (η οποία έκτοτε έχει αποκτηθεί από την Oracle Corporation) και κυκλοφόρησε το 1995 ως βασική συνιστώσα της πλατφόρμας Java της Sun Microsystems. Η γλώσσα αντλεί μεγάλο μέρος της σύνταξης του από το C και C ++, αλλά έχει λιγότερες εγκαταστάσεις χαμηλού επιπέδου από ό, τι οποιοδήποτε από αυτά.

Η αρχική εφαρμογή αναφοράς των μεταγλωττιστών της Java, των εικονικών μηχανών, και βιβλιοθηκών αρχικά κυκλοφόρησε από τη Sun κάτω από ιδιόκτητες άδειες. Από τον Μάιο του 2007, η Sun ανάκτησε την αξιοποίηση των τεχνολογιών της Java υπό την άδεια GNU General Public License. Άλλοι έχουν επίσης αναπτύξει εναλλακτικές εφαρμογές των τεχνολογιών αυτών όπως ο μεταγλωττιστής GNU για την Java (bytecode compiler), το GNU Classpath και IcedTea-Web (plugin πρόγραμμα περιήγησης για τα applets).

5.4 Υλοποίηση λειτουργιών εργαλείου

Σκοπός του κεφαλαίου αυτού είναι η παρουσίαση των εργασιών κατά τη συγγραφή του πηγαίου κώδικα του λογισμικού. Αφού επιλέχθηκε το εργαλείο και η γλώσσα προγραμματισμού για τη δημιουργία λογισμικού έγινε η υλοποίηση. Η γλώσσα προγραμματισμού που επιλέχθηκε για την υλοποίηση του λογισμικου είναι η Java την οποία υποστηρίζει το IDE Eclipse.

Σχετικά και με τις 3 λειτουργίες συμβαίνει το εξής: Για την πρώτη λειτουργία γίνεται η εξαγωγή των δικαιωμάτων του συστήματος, για την δεύτερη λειτουργία γίνεται έλεγχος για την πολιτική έτσι ώστε να αποφασιστεί εαν είναι ορθά διατυπωμένη. Εαν η πολιτική δεν είναι διατυπωμένη τότε το πρόγραμμα τερματίζει αναφέροντας το πρόβλημα. Για την τρίτη λειτουργία γίνεται διεξαγωγή των δικαιωμάτων της πολιτικής ιδιωτικότητας

5.4.1 Υλοποίηση πρώτης λειτουργίας

Η πρώτη λειτουργία που υλοποιεί το εργαλείο είναι η εξής: Ο Java Dom Parser με βάση το xml αρχείο του συστήματος παράγει το DOM δέντρο από το οποίο γίνεται και διεξαγωγή των δικαιωμάτων του συστήματος. Πιο κάτω φαίνεται ο ψευδοκώδικας για κάποια βασικά σημεία όπου βάση αυτού δημιουργούνται τα δικαιώματα του συστήματος.

```
/*if the tag name is <in> we get the type and we check if is base type we
 * add the permission READ,otherwise we add the permission ACCESS*/
  if(nodename==<in>){
     get the children of node <in> /*gets the children of tag <in>*/
     for (int i = 0 i < children  i++){

      /*if one of <in> children it's <type> then we get its text value*/
        if(if(children==“type”){
           if (<type> text value is a base type){
                add permission “read”
           }
           //the type is not a base type but is something like G[ti]
           else{
               add permission “access”
           }
```

```
        }
    }
```

Εαν συναντήσουμε το σημαντήρα <in> τότε χρησιμοποιούμε τον κανόνα in όπου ελέγχουμε τον τύπο του καναλιού και εαν είναι βασικός τύπος (base type), τότε προσθέτουμε το δικαίωμα read για το σύστημα, εαν όμως ο τύπος είναι κάτι του τύπου G[ti] τότε προσθέτουμε σαν δικαίωμα το access.

```
/*if the tag name is <out> we get the obj and we check if is base type we
 * add the permission WRITE,otherwise we add the permission DISCLOSE G 1*/
     if(nodename==<out>){
        get the children of node <out> /*gets the children of tag <in>*/
        for (int i = 0 i < children i++){

        /*if one of <in> children it's <type> then we get its text value*/
        if(if(children==“obj”){
           if (<obj> text value is a base type){
                add permission “write”
           }
           //the type is not a base type but is something like G[ti]
           else{
               add permission “disclose G 1”
           }}}
```

Εαν συναντήσουμε το σημαντήρα <out> τότε χρησιμοποιούμε τον κανόνα out όπου ελέγχουμε τον τύπο του καναλιού και εαν είναι βασικός τύπος (base type), τότε προσθέτουμε το δικαίωμα write για το σύστημα, εαν όμως ο τύπος είναι κάτι σαν G[ti] τότε προσθέτουμε σαν δικαίωμα το disclose G 1.

```
/*if the tag name is <repl> we just check if we have disclose G in the
values
     * of the hashmap and we put the * instead of the number */
     if(nodename==<repl>){
        if we already have the permission disclose G number
          then we do replace number with “*”
}
```

Εαν συναντήσουμε το σημαντήρα <repl> τότε χρησιμοποιούμε τον κανόνα repl και εαν έχουμε ήδη το δικαίωμα disclose G 3 για παράδειγμα τότε αντικαθιστούμε τον αριθμό 3 με το * , δηλαδή μπορεί να γίνει αποκάλψη άπειρες φορές.

5.4.2 Υλοποίηση δεύτερης λειτουργίας

Η δεύτερη λειτουργία που υλοποιεί το εργαλείο είναι η εξής: Ο Java Dom Parser με βάση το xml αρχείο της πολιτικής ιδιωτικότητας παράγει το DOM δέντρο από το οποίο προκύπτουν τα δικαιώματα της πολιτικής. Πιο κάτω φαίνεται ο ψευδοκώδικας για κάποια βασικά σημεία από όπου ελέγχουν εαν η πολιτική είναι ορθά διατυπωμένη.

Ψευδοκώδικας για τον Ορισμό 2, 1

```
/*check  DEFINITION 2,RULE 1 for all i, j, 1<=i, j<= n, if i!=j then ti !=
t j, children of the root must be different base types*/

 for(int k=0; k<children; k++){
    for(int m=0; m<children; m++){
         if(k!=m){/*dont compare node with its self*/
         /*if the two text values are the same means that policy violated*/
      if(nodesk.value== nodesm.value){
           print("\n DEFINITION 2,1 VIOLATED.NOT IN WELL FORMED");
           Program exits
      }
     } } }
```

Πιο πάνω γίνεται ο έλεγχος εαν υπάρχουν ίδια ονόματα βασικών τύπων σε μια πολιτική ιδιωτικότητας, εαν υπάρχουν τότε η πολιτική δεν είναι ορθά διατυπωμένη και το πρόγραμμα τερματίζει.

Ψευδοκώδικας για το Ορισμο 2, 2

```
/*for all i,1<=i<n, if G is in groups(Hi) then G must not int desc(G,Hi)*/
private static void Definiton2_rule2(Node node,Document doc){

 children = node.getChildNodes();/*get nodes children*/
 for (i=0 to children i++) {
    /*if node's name is hierarchy and has children*/
 if(child.name==group && child.parent.LastChild.getChildNodes()!=null){
      find text value of node using xpath
      find the descendants of node hierarchy
      for(int x=0; to nodes){
      /*if one of the descedants contains the node name group*/
      if(nodes.item(x).getNodeName().contains("group")){

/*if one group has the same name with its descedants group then policy is
not in well formed*/
if(nodes.value==nodes2.value){
   print("DEFINITION 2,2 VIOLATED.THE POLICY IS NOT IN WELL-FORMED");
   program exits
}}}}
```

Πιο πάνω γίνεται ο έλεγχος με βάση τον Ορισμό 2, 2 που αναφέρθηκε στο υποκεφάλαιο 3.4.1 Ορθά διατυπωμένη πολιτική. Διασχίζοντας το Dom δέντρο παίρνουμε τα παιδιά κάθε κόμβου. Ακολούθως ελέγχουμε εαν κάποιο απο τα παιδία του έχει το σημαντήρα <group> και εαν το τελευταίο παιδί του πατέρα του group δηλαδή το hierarchy να μην είναι null και βρίσκουμε τα descedants του hierarchy όπου ελέγχουμε εαν το όνομα κάποιου descedant είναι το ίδιο με το group, εαν ναι τότε η πολιτική παραβιάζεται και το πρόγραμμα τερματίζει

αφού με βάση τον κανόνα δεν επιτρέπεται οι απογόνοι κάποιας ομάδας να έχει το ίδιο όνομα με την ομάδα.

5.4.3 Υλοποίηση τρίτης λειτουργίας

Η τρίτη λειτουργία που θα κάνει το εργαλείο θα είναι ως εξής: Αφού κριθεί ότι η πολιτική ιδιωτικότητας είναι ορθά διατυπωμένη απο την λειτουργία 2 θα γίνεται διεξαγωγή των δικαιωμάτων της πολιτικής με βάση το Figure 2-Permission Extraction Relation. Πιο κάτω παρουσιάζονται σε ψευδοκώδικα βασικά σημεία απο τη συνάρτηση.

```
public static ArrayList<String>  permission_extraction_relation(){
   for(int h=0; h<permnode_children; h++){
      //if the permission is "access" and there isnt already in the
arraylist then we add it

if(permnode_children.name=="access" && !permission contains (“access”)){
   //add permission to the arraylist with the permissions
   permissions.add(“access”);
}
//if the permission is "read" and there isnt already in the arraylist then
we add it
if(permnode_children.name=="read" && !permission contains (“read”)){
   //add permission to the arraylist with the permissions
   permissions.add(“read”);
}

if(permnode_children.name=="write" && !permission contains (“write”)){
   //add permission to the arraylist with the permissions
   permissions.add(“write”);
}

if(permnode_children.name=="nondisclose”){
   for (int v = 0; to permissions; v++) {

if(permissions.get(v).contains("nondisclose")){
   removing the first occurrence of item permissions.get(v)
   remove the nondisclose is on the permissions arraylist
   permissions.remove(permissions.get(v));
}}
//add new permission using xpath
permissions.add(per);

}
```

Πιο πάνω φαίνεται η λογική σχετικά με το πως γίνεται η διεξαγωγή των δικαιωμάτων μιας πολιτικης.

- Εαν το δικαιώμα που συναντάμε είναι το access και δεν υπάρχει ήδη στο arraylist με τα δικαιώματα το προσθέτουμε.
- Εαν το δικαιώμα που συναντάμε είναι το read και δεν υπάρχει ήδη στο arraylist με τα δικαιώματα το προσθέτουμε.

- Εαν το δικαιώμα που συναντάμε είναι το write και δεν υπάρχει ήδη στο arraylist με τα δικαιώματα το προσθέτουμε.
- Εαν το δικαίωμα είναι nondisclose προς μια ομάδα, όπως για παράδειγμα nondisclose ETP το οποίο έστω ότι υπάρχει ήδη στη λίστα μας και ξανασυναντήσουμε το δικαίωμα nondisclose σε μια ομάδα που βρίσκεται σε πιο κάτω επίπεδο, τότε αφαιρούμε το nondisclose που ήδη υπήρχε και βάζουμε αυτό που βρίσκεται στο πιο κάτω επίπεδο της ιεραρχίας.

5.5 Έλεγχος λειτουργιών εργαλείου

Οι διαδικασίες επαλήθευσης και επικύρωσης έχουν στόχο να δείξουν ότι το σύστημα είναι σύμφωνο με τις απαιτήσεις και προδιαγραφές του και ότι ικανοποιεί τις προσδοκίες του χρήστη. Περιλαμβάνουν τον έλεγχο και την επισκόπηση διαδικασιών, καθώς και δοκιμές του συστήματος.

Η δοκιμή ενός συστήματος συνεπάγεται την εκτέλεσή του με στιγμιότυπα δεδομένων ελέγχου τα οποία εξάγονται από τις προδιαγραφές των πραγματικών δεδομένων που πρόκειται να επεξεργαστεί το σύστημα.
Αφού ολοκλήρωσα την υλοποίηση ακολούθησε έλεγχος των λειτουργιών με διάφορα σενάρια έτσι ώστε να εξακριβωθεί η ορθότητα του λογισμικού και η πληρότητα ως προς τις απαιτήσεις. Οι ελέγχοι ακολουθούν δύο στάδια δοκιμών: τις δοκιμές συστατικών στοιχείων (ή υπομονάδων) όπου κάθε συστατικό στοιχείο δοκιμάζεται ανεξάρτητα πρωτούν ενωθούν σε ένα και τις δοκιμές συστήματος όπου γίνεται η δοκιμή του εργαλείου στο σύνολο του. Τα σενάρια που χρησιμοποιήθηκαν είναι τα εξής:

- Διεξαγωγή δικαιωμάτων συστήματος και έλεγχος εαν το σύστημα ικανοποιεί την πολιτική. Υπάρχουν δύο περιπτώσεις όπου στην πρώτη η πολιτική ιδιωτικότητας ικανοποιείται απο το σύστημα, ενώ στη δεύτερη περίπτωση η πολιτική ιδιωτικότητας δεν ικανοποιείται από το σύστημα.

- Ελέγχοι Ορισμού 2, 1 όπου ελέγχεται εαν οι βασικοί τύποι που υπάρχουν είναι διαφορετικοί.

- Ελέγχοι Ορισμού 2, 2 όπου ελέγχεται για κάθε ομάδα εαν υπάρχει στους απογόνους

της ομάδα με το ίδιο όνομα.

- Ελέγχοι Ορισμού 2, 3 εαν σε κάποιο επίπεδο μιας ιεραρχίας κάποια ομάδα G έχει το δικαίωμα nondisclose και κάποια ομάδα G'' που είναι απόγονος του G έχει το δικαίωμα disclose G' τότε το G' πρέπει να είναι απόγονος του G.

- Έλεγχος για τη διεξαγωγή δικαιώματος nondisclose.

- Έλεγχος για τη διεξαγωγή δικαιώματος disclose G.

- Έλεγχος για τη διεξαγωγή δικαιώματος access.

- Έλεγχος για τη διεξαγωγή δικαιώματος read.

5.5.1 Έλεγχος 1ης και 3ης λειτουργίας

- Για τον έλεγχο εξαγωγής δικαιωμάτων του συστήματος χρησιμοποιήσα ως πολιτική το centralized_policy.xml και σύστημα το system_parse.xml. Στην πρώτη περίπτωση το σύστημα ικανοποιεί την πολιτική όπως φαίνεται πιο κάτω αφού σε κάθε σύγκριση το σύστημα έχει τόσα δικαιώματα που είναι που είναι ίδια με την πολιτική ιδιωτικότητας, ή εχει λιγότερα δικαιώματα.

------System permissions are------:
Θ= < ETP. PA, Loc[read, access].Fee[disclose ETP *].>
< ETP. Car. OBE, Loc[disclose ETP *, access].>
< ETP. Car. GPS, Loc[disclose ETP *]>

THE POLICY IS IN WELL-FORMED

------Policy permissions are------:
ETP. PA[nondisclose ETP, access, read]
ETP. Car. OBE[nondisclose ETP, access, disclose ETP *]
ETP. Car. GPS[nondisclose ETP, disclose ETP *]

[nondisclose ETP, access, read] >= [, read, access]To sistima ikanopoia tin politiki

[nondisclose ETP, access, disclose ETP *] >= [, disclose ETP *, access]To sistima ikanopoia tin politiki

[nondisclose ETP, disclose ETP *] >= [, disclose ETP *]To sistima ikanopoia tin politiki

- Στην δεύτερη περίπτωση χρησιμοποιήθηκαν τα αρχεία centralized_policy_error.xml και σύστημα το system_parse.xml. Η πολιτική δεν ικανοποιείται αφού δεν επιτρέπεται το σύστημα να κάνει περισσότερα απο αυτά που κάνει η πολιτική. Όπως φαίνεται πιο κάτω το σύστημα δεν ικανοποιεί την πολιτική ιδιωτικότητας αφού στην πρώτη σύγκριση το σύστημα έχει ως δικαίωμα το read ενώ η πολιτική ιδιωτικότητας δεν έχει τέτοιο δικαίωμα. Επίσης, στη δεύτερη σύγκριση πάλι υπάρχει παραβίαση της πολιτικής αφού στο σύστημα παρουσιάζεται το δικαίωμα disclose ETP *, ενώ πάλι η πολιτική ιδιωτικότητας δεν έχει τέτοιο δικαίωμα.

  THE POLICY IS IN WELL-FORMED

  ------Policy permissions are------:
  ETP. PA[nondisclose ETP, access]
  ETP. Car. OBE[nondisclose ETP, access]
  ETP. Car. GPS[nondisclose ETP, disclose ETP *]

  [nondisclose ETP, access] < [, read, access]To sistima den ikanopoia tin politiki

  [nondisclose ETP, access] < [, disclose ETP *, access]To sistima den ikanopoia tin politiki

  [nondisclose ETP, disclose ETP *] >= [, disclose ETP *]To sistima ikanopoia tin politiki

5.5.2 Έλεγχος 2ης λειτουργίας

Ελέγχοι για τον Ορισμό 2, 1

- Στην περίπτωση αυτή στο test1_2.1_wellformed.xml έχω βάλει σαν βασικούς τύπους (base type) το Loc και το Fee και επειδή είναι διαφορετικά τα ονόματα τους η πολιτική είναι ορθά διατυπωμένη.

- Στην περίπτωση αυτή στο test2_2.1_wellformed.xml έχω βάλει σαν βασικούς τύπους (base type) το Loc και το loc και επειδή είναι ίδια τα ονόματα τους η πολιτική είναι δεν είναι ορθά διατυπωμένη και παραβιάζεται ο κανόνας 1 του Ορισμού 2 και το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

  DEFINITION 2,1 VIOLATED.THE POLICY IS NOT IN WELL FORMED
  Problem with base types [Loc] and [loc] because they have the same name

- Στην περίπτωση αυτή στο test3_2.1_wellformed.xml έχω βάλει σαν βασικούς τύπους (base type) το to Loc Fee,MOB,fee και παραβιάζεται ο κανόνας 1 του definition 2 λόγω του Fee και του fee και το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

  DEFINITION 2,1 VIOLATED.THE POLICY IS NOT IN WELL FORMED
  Problem with base types [Fee] and [fee] because they have the same name
  .

<u>Ελέγχοι για τον Ορισμό 2, 2</u>

- Στην περίπτωση αυτή στο test4_2.2_wellformed.xml έχω βάλει ξανά το group obe κάτω απο την ιεραρχία του ήδη υπάρχοντος OBE γιατί επίσης αγνοώ το case των γραμμάτων και σωστά παραβιάζεται ο κανόνας 2 αφού δεν επιτρέπεται να έχω στους απογόνους κάποιας ομάδας το ίδιο όνομα με αυτή . Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

  DEFINITION 2,2 VIOLATED.THE POLICY IS NOT IN WELL FORMED BECAUSE OF DESCEDANT [obe]

- Στην περίπτωση αυτή στο test5_2.2_wellformed.xml έχω βάλει ξανά το group ETP κάτω απο την ιεραρχία του αρχικού ETP και σωστά παραβιάζεται ο κανόνας αφού δεν επιτρέπεται να έχω στους απογόνους κάποιας ομάδας το ίδιο όνομα με αυτή. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

DEFINITION 2,2 VIOLATED.THE POLICY IS NOT IN WELL FORMED BECAUSE OF DESCEDANT [ETP]

<u>Ελέγχοι για τον Ορισμό 2, 3</u>

- Στην περίπτωση αυτή στο test6_2.3_wellformed.xml έχω nondisclose στο ETP και στο Car και το policy γίνεται violated στον κανόνα 3 αφού η ομάδα Car έχει στους απογόνους της το OBE:{access,disclose ETP2} και αφού αυτό έχει το δικαίωμα disclose ETP2 το ETP θα έπρεπε να ήταν απόγονος του Car για να μην υπήρχε πρόβλημα . Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

DEFINITION 2,3 VIOLATED.THE POLICY IS NOT IN WELL FORMED

- Στην περίπτωση αυτή στο test7_2.3_wellformed.xml έχω nondisclose στο ETP και στο PA και το policy είναι well formed αφού το PA δεν έχει στα descendants ομάδα που να έχει το δικαίωμα disclose G για να το επηρεάσει. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

THE POLICY IS IN WELL FORMED

- Στην περίπτωση αυτή στο test8_2.3_wellformed.xml έχω nondisclose στο Car και στο OBE έχω disclose home και το home είναι στους απογόνους του car αφού το home είναι παιδί του SC οπότε σωστά δεν παραβιάζεται η πολιτική. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

THE POLICY IS IN WELL FORMED

- Στην περίπτωση αυτή στο test9_2.3_wellformed.xml έχω non disclose μόνο στο car και στα descendants του car έχω disclose car και αφού το car υπάρχει ήδη σωστά το policy δεν παραβιάζεται. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

  THE POLICY IS IN WELL FORMED

Έλεγχος για το permission nondisclose

- Στην περίπτωση αυτή στο test1_permission_extraction.xml έχω nondisclose στο ETP και στο PA και σωστά εμφανίζει σαν permission το nondisclose PA αντί nondisclose ETP αφού το PA βρίσκεται σε πιο κάτω επίπεδο και πάντα όταν έχω nondisclose παίρνω εκείνο το group που βρίσκεται στο πιο κάτω επίπεδο. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

  THE POLICY IS IN WELL FORMED
  PERMISSION EXTRACTED ARE:[nondisclose ETP, disclose Car*]

- Στην περίπτωση αυτή στο test2_permission_extraction.xml έχω nondisclose μόνο στο ETP και το permission είναι nondisclose ETP. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

  THE POLICY IS IN WELL FORMED
  PERMISSION EXTRACTED ARE:[nondisclose ETP, disclose Car*]

- Στην περίπτωση αυτή στο test3_permission_extraction.xml έχω σε όλα τα groups το permission non disclose και πρέπει να επιστρέφει σωστά σαν permission nondisclose OBE,GPS,SC αφού αυτά είναι τα group που βρίσκονται στο κατώτατο επίπεδο και είναι siblings.(κάνω HIDE το WELL FORMED FUNCTION για να μην υπάρχει πρόβλημα) . Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

PERMISSION EXTRACTED ARE:[access, read, disclose ETP 2, nondisclose OBE, disclose Car*, nondisclose GPS, nondisclose SC]

Έλεγχος για το permission disclose G

- Στην περίπτωση αυτή στο test4_permission_extraction.xml έχω disclose ETP2 και disclose ETP* και σωστά το permission είναι disclose ETP*. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση

  THE POLICY IS IN WELL FORMED
  PERMISSION EXTRACTED ARE:[nondisclose ETP, access, read, disclose ETP*]

- Στην περίπτωση αυτή στο test5_permission_extraction.xml έχω disclose ETP2,disclose ETP9 και disclose ETP6 και σωστά το permission είναι disclose17. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση

  THE POLICY IS IN WELL FORMED
  PERMISSION EXTRACTED ARE:[nondisclose ETP, access, read, disclose ETP17]

- Στην περίπτωση αυτή στο test6_permission_extraction.xml έχω disclose ETP2,disclose ETP* και disclose SC8 και σωστά το permission είναι disclose ETP*,disclose SC8. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση:

  THE POLICY IS IN WELL FORMED
  PERMISSION EXTRACTED ARE:[nondisclose ETP, access, read, disclose ETP*]

Έλεγχος για το permission access και read

- Στην περίπτωση αυτή στο test4_permission_extraction.xml έχω disclose ETP2 και disclose ETP*,acces στο group OBE, access και read στο SC και στο PA και σωστά

το permission είναι nondisclose ETP,disclose ETP*, access, read. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση

THE POLICY IS IN WELL FORMED
PERMISSION EXTRACTED ARE:[nondisclose ETP, access, read, disclose ETP*]

- Στην περίπτωση αυτή στο test5_permission_extraction.xml έχω disclose ETP2,disclose ETP9 ,disclose ETP6 acces στο group OBE, access και read στο SC και στο PA και σωστά τa permission είναι nondisclose ETP, access, read, disclose ETP 17. Το εργαλείο παρουσιάζει την πιο κάτω ειδοποίηση

  THE POLICY IS IN WELL FORMED
  PERMISSION EXTRACTED ARE:[nondisclose ETP, access, read, disclose ETP17]

Κεφάλαιο 6

Συμπεράσματα

6.1 Γενικά συμπεράσματα

Στόχος της εργασίας αυτής ήταν να υλοποιηθεί ένα εργαλείο με σκοπό να βοηθήσει τους χρήστες (οι οποίοι έχουν βασικές γνώσεις για τη δημιουργία συστημάτων και πολιτικών με την χρήση του π-λογισμού με ομάδες) να ελέγχουν εαν μια πολιτική είναι well formed, να εξάγουν τα δικαιώματα της πολιτικής, να ελέγχουν τον τύπο του συστήματος καθώς και να εξάγουν τα δικαιώματα του συστήματος και έπειτα να ελέγχεται εαν ένα σύστημα ικανοποιά την πολιτική που αυτό αποτελεί και τον βασικό στόχο.

Οι στόχοι που τέθηκαν στην αρχή ανάπτυξης του εργαλείου αυτού έχουν επιτευχθεί καθώς υλοποιήθηκαν και οι 3 απαιτούμενες λειτουργίες. Το εργαλείο αυτό θεωρείται εύχρηστο μια και υπάρχουν οδηγοί για την δημιουργία των xml αρχείων της πολιτικής και του συστήματος και το μόνο που πρέπει να κάνει ο χρήστης είναι να εισάξει στο πρόγραμμα τα ονόματα των 2 αρχείων. Αφού γίνει αυτό, το εργαλείο κάνει parse το xml αρχείο του συστήματος χρησιμοποιώντας τον Java Dom Parser και εξάγει τα δικαιώματα του συστήματος όπου αυτό αποτελεί και την πρώτη λειτουργία του συστήματος.

Εν συνεχεία, γίνεται parse το xml αρχείο της πολιτικής χρησιμοποιώντας και πάλι τον Java Dom Parser και ελέγχεται εαν η πολιτική είναι well formed. Εαν η πολιτική είναι well formed το εργαλείο συνεχίζει εκτελώντας την επόμενη λειτουργία, εαν όχι τότε εμφανίζει στον χρήστη ποιος κανόνας της πολιτικής παραβιάστηκε καθοδηγώντας τον που βρίσκεται το σφάλμα. Αφού υπολογιστούν και τα δικαιώματα της πολιτικής, συγκρίνονται στο τέλος τα δικαιώματα της πολιτικής και του συστήματος για να συμπερανθεί εαν το σύστημα ικανοποιεί την πολιτική.

Παρατηρούμε ότι το εργαλείο έχει τη δυνατότητα να πάρει ως είσοδο ένα σύστημα και μια πολιτική ιδιωτικότητας, οπότε προκύπτει ότι για ένα σύστημα μπορούν να δοκιμαστούν διάφορες πολιτικές ιδιωτικότητας για να ελεγχθεί η ικανοποιησιμότητα.

Τέλος, το εργαλείο έχει δημιουργηθεί με τέτοιο τρόπο έτσι ώστε σε περίπτωση καταχώρησης λανθασμένων δεδομένων στα xml αρχεία, το πρόγραμμα να τερματίζει αναφέροντας στον χρήστη το σφάλμα που προέκυψε γι'αυτό και θεωρείται φιλικό, με απαραίτητη όμως την επανεκκίνηση του προγράμματος.

6.2 Προβήματα υλοποίησης – αντιμετώπιση

Η μέθοδος η οποία χρησιμοποιήθηκε για την ανάπτυξη του εργαλείου είναι βασισμένη σε μια μέθοδο ανάπτυξης λογισμικού, η οποία χρησιμοποιείται σχεδόν σε όλα τα έργα ανάπτυξης λογισμικού και ονομάζεται «Κύκλος ζωής ανάπτυξης λογισμικού».

Κατά το στάδιο της υλοποίησης ο αρχικός προβληματισμός ήταν να επιλέξουμε το περιβάλλον ανάπτυξης για το λογισμικό καθώς και τη γλώσσα προγραμματισμού. Σύμφωνα με τις απαιτήσεις, θέλαμε μια γλώσσα προγραμματισμού η οποία να υποστήριζε έτοιμο parser , για αυτό μετά απο αναζήτηση, κρίθηκε κατάλληλη η Java, η οποία έχει τον Java Dom Parser και η Eclipse ως το περιβάλλον ανάπτυξης, η οποία υποστηρίζει την γλώσσα αυτή.

Αφού επιλέχθηκε το περιβάλλον ανάπτυξης και η γλώσσα προγραμματισμού, αρχικά έγιναν parse τα δύο αρχεία, προκειμένου να δημιουργηθούν τα dom trees και να μπορούμε να τα επεξεργαστούμε. Το πρόβλημα εδώ ήταν ότι σε περίπτωση που για σημαντήρες με βάση τη γραμματική είχαν το ίδιο όνομα αλλά διαφορετικό περιεχόμενο, όταν ήθελα να πάρω το περιεχόμενο για να το επεξεργαστώ τότε μου έβγαζε τα περιεχόμενα για όλους τους σημαντήρες που είχαν το ίδιο όνομα. Για την αντιμετώπιση του προβλήματος αυτού αποφάσισα να δώσω κάτι σαν id στους σημαντήρες που είχαν το ίδιο όνομα, ετσι ώστε τώρα να ήταν μοναδικοί και να προσφερόταν η δυνατότητα για το κάθε ένα να έπερνα το περιεχόμενο του συγκεκριμένου σημαντήρα ξεχωριστά.

Ακολούθως, λύνοντας αυτό το πρόβλημα προχώρησα στην διεξαγωγή των δικαιωμάτων του συστήματος και στη συνέχεια διεξαγωγή των δικαιωμάτων της πολιτικής με βάση τις ομάδες

που επέλεγε το σύστημα. Τέλος, για τη σύγκριση των δικαιωμάτων των δύο έπρεπε να σκεφτώ πως θα κρατάω κάπου τα αποτελέσματα του συστήματος και της πολιτικής, έτσι ώστε να μπορώ να τα συγκρίνω. Για αυτό το σκοπό χρησιμοποίησα κατάλληλες δομές, οι οποίες μου έδωσαν την ευελιξία να κάνω αποδοτικά και εύκολα τις συγκρίσεις αυτές.

6.3 Μελλοντικές εργασίες

Η έρευνα για ανάπτυξη εργαλείων τα οποία βοηθούν τους χρήστες να ελέγξουν εαν κάποιο σύστημα ικανοποιεί διάφορες πολιτικές, δεν ολοκληρώνεται με την ολοκλήρωση της παρούσας διπλωματικής εργασίας.

Αναμφίβολα, υπάρχουν διάφορες κατευθύνσεις οι οποίες θα μπορούσαν να ακολουθηθούν έχοντας ως βάση την παρούσα εργασία. Μια απο αυτές τις κατευθύνσεις θα ήταν να γίνει αύξηση της λειτουργικότητας του παρόντος εργαλείου. Θα ήταν χρήσιμο ως προς τον χρήστη να παρουσιαζόταν γραφικά η πολιτική που απεικονίζει το xml αρχείο αλλά και το σύστημα, έτσι ώστε αυτό να τον διευκολύνει να διορθώσει το σύστημα ή την πολιτική ανάλογα στην περίπτωση κάποιου προβλήματος. Μια επιπλέον αύξηση της λειτουργικότητας θα ήταν να παρουσιαζόταν στον χρήστη σε ποιο σημείο υπάρχει πρόβλημα στην περίπτωση που το σύστημα δεν ικανοποιεί την πολιτική έτσι ώστε να μπορεί να το αλλάξει χωρίς να χρειάζεται να ψάχνει ο ίδιος λύση στο πρόβλημα.

Ακόμη, θα μπορούσε να δημιουργηθεί μια γραφική διεπαφή για να γίνει περισσότερο εύχρηστο το εργαλείο έτσι ώστε να δίνεται η δυνατότητα στο χρήστη να εισάγει τα δεδομένα του με μεγαλύτερη ευκολία. Συγκεκριμένα, θα μπορούσαν να υπάρχουν διάφορα κουμπιά έτσι ώστε να μπορεί ο ίδιος ο χρήστης να καθορίσει εαν απλά θέλει να ελέγξει την πολιτική ξεχωριστά, η το σύστημα ξεχωριστά ή και τα δύο μαζί.

Επιπρόσθετα, κάτι άλλο που θα μπορούσε να γίνει είναι να προσφέρεται και μέσω διαδικτύου το εργαλείο έτσι ώστε να μπορεί ο κάθε χρήστης που χρησιμοποιεί το διαδίκτυο να έχει πρόσβαση σε αυτό το εργαλείο οποιαδήποτε στιγμή το χρειαστεί.

Όσον αφορά τη θεωρητική προσέγγιση θα μπορούσε να εισαχθούν αυτοί οι μεθόδοι σε πραγματική γλώσσα η οποία να έχει συγκεκριμένη σύνταξη με βάση τις γραμματικές που ορίσαμε

Βιβλιογραφία

[1] A. Barth, A. Datta, J. C. Mitchell, H.Nissenbaum, "Privacy and Contextual Integrity: Framework and Applications," August 2008.

[2] Α.Φιλίππου, "Άλγεβρες Διεργασιών", Σημειώσεις μαθήματος ΕΠΛ 664-Ανάλυση και Επαλήθευση Συστημάτων, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου, 2014.

[3] M. C. Carl Tschantz, J. M.Wing. "Formal Methods for Privacy," CMU-CS-09-154, August 2009.

[4] Federal Trade Commission. How to comply with the children's online privacy protection Rule, Available: http://www.ftc.gov/bcp/conline/pubs/buspubs/coppa.htm,1999.

[5] G. Laurie, Genetic Privacy – A challenge to Medico-Legal Norms, Cambridge 2002

[6] J.Parrow* , "An Introduction to the Π-Calculus".

[7] L. Cardelli, G. Ghelli, and A. D. Gordon, Secrecy and group creation, Information and Computation, 2005.

[8] "CNN. FBI Seeks Stolen Personal Data on 26 Million Vets,". Available:http://www.cnn.com/2006/US/05/22/vets.data/.

[9] Oce for Civil Rights. Summary of the HIPAA privacy rule. US Department of Health & Human Services, 2003.

[10] "Breach of Privacy and Confidentiality under Information Technology Act,2000,". Available: http://www.legalserviceindia.com/article/l288-Breach-of-privacy-&-Confidentiality-.html.)

[11] Κ. Λαμπρινουδάκης, Λ. Μήτρου, Σ. Γκρίτζαλης, Σ. Κάτσικας «Προστασία» της Ιδιωτικότητας & Τεχνολογίες Πληροφορικής και επικοινωνιών-Τεχνικά και νομικά θέματα, εκδ. Παπασωτηρίου, Αθήνα 2010

[12] D. Kouzapas, A. Philippou, Type checking privacy policies in the π-calculus,

[13] EPIC. The Gramm-Leach-Bliley Act, Available: http://epic.org/privacy/glba/.

[14] Federal Trade Commission. In brief: the financial privacy requirements of the Gramm-Leach-Bliley Act, Available: http://www.ftc.gov/bcp/conline/pubs/buspubs/glbshort.htm, 2002.

Παράρτημα Α-κώδικας για την κλάση της πολιτικής ιδιωτικότητας

Στο παράρτημα αυτό παρουσιάζεται ο κώδικας όσον αφορά την πολιτική ιδιωτικότητας.

```
package diplomatiki;


import java.io.File;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import java.util.ArrayList;


public class politiki extends sistima{
	static int answer=0, count_groups=0, count_gph=0, count_perms=0, count_hierarchies=0, count_disclose=0;

  /*Returns the children and the parent of a specific node*/
  public static NodeList Get_children_ofa_node(NodeList elemNodeList){
    NodeList childrenList=null;
        for(int j=0; j<elemNodeList.getLength(); j++) {
          childrenList =elemNodeList.item(j).getChildNodes();
```

```
        }
    return childrenList;
  }


  /*Checks if the policy is well-formed,Definition 2*/
  public static void well_formed_policy (Document doc) throws XPathExpressionException{
     NodeList childrenList=null;
          /*Separate the parsed XML elements*/
          NodeList elemNodeList = doc.getElementsByTagName(doc.getDocumentElement().getNodeName());
          childrenList=Get_children_ofa_node(elemNodeList);/*returns the children of a specific node*/

     /*check  DEFINITION 2,RULE 1 for all i, j, 1<=i, j<= n, if i!=j then ti != t j,
           *children of the root must be different base types*/
           for(int k=0; k<childrenList.getLength(); k++){
             for(int m=0; m<childrenList.getLength(); m++){
                    if(childrenList.item(k).getNodeName()!="#text"){/*if the node name is not null*/
                      if(k!=m){/*dont compare node with its self*/

                        XPath xpathk = XPathFactory.newInstance().newXPath();
                        XPath xpathm = XPathFactory.newInstance().newXPath();
                              /*find the text value for each node*/
                                  XPathExpression expressionk = xpathk.compile("//"+childrenList.item(k).getNodeName()+"/text()[1]");
                                  NodeList nodesk = (NodeList) expressionk.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                                  XPathExpression expressionm=xpathm.compile("//"+childrenList.item(m).getNodeName()+"/text()[1]");
                                  NodeList nodesm = (NodeList) expressionm.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                              /*modify node value to dissappear space*/

        nodesk.item(k).setNodeValue(nodesk.item(k).getNodeValue().replaceAll("\\s+",""));
```

```
nodesm.item(m).setNodeValue(nodesm.item(m).getNodeValue().replaceAll("\\s+",""));

                        /*if the two text values are the same means that policy violated*/

if(nodesk.item(k).getNodeValue().compareToIgnoreCase(nodesm.item(m).getNodeValue())==0){
                        System.out.println("\n DEFINITION 2,1 VIOLATED.THE POLICY IS NOT IN WELL FORMED");
                    System.exit(0);
                        }
                       }
                     }
                   }
    }

        /*check  DEFINITION 2,RULE 2  and DEFINITION2,RULE 3*/
                for(int j=0; j<childrenList.getLength(); j++){
                  Definiton2_rule2(childrenList.item(j),doc);
                  Definiton2_rule3(childrenList.item(j),doc);
             }
                System.out.println("\nTHE POLICY IS IN WELL-FORMED");

        }


        /*modify tags to be unique*/
        private static void change_tags(Node node,Document doc) throws XPathExpressionException {
            int i;
                Node child;
                NodeList children;

                children = node.getChildNodes();
            for(int p=0; p<children.getLength(); p++){
              if(children.item(p).getNodeName()!="#text"){/*if nodename is not null*/
```

```
if(children.item(p).getNodeName().compareToIgnoreCase("group")==0){
          if (children.item(p) instanceof Element) {
                    String toTag_temp = String.valueOf(count_groups);
          String toTag=children.item(p).getNodeName().concat(toTag_temp);
                    Element elem = (Element)children.item(p);
          doc.renameNode(elem, elem.getNamespaceURI(), toTag);
          count_groups++;
      }

}
if(children.item(p).getNodeName().compareToIgnoreCase("gph")==0){
          if (children.item(p) instanceof Element) {
                    String toTag_temp = String.valueOf(count_gph);
          String toTag=children.item(p).getNodeName().concat(toTag_temp);
                    Element elem = (Element)children.item(p);
          doc.renameNode(elem, elem.getNamespaceURI(), toTag);
         count_gph++;
      }

}
if(children.item(p).getNodeName().compareToIgnoreCase("perm")==0){
          if (children.item(p) instanceof Element) {
                    String toTag_temp = String.valueOf(count_perms);
          String toTag=children.item(p).getNodeName().concat(toTag_temp);
                    Element elem = (Element)children.item(p);
          doc.renameNode(elem, elem.getNamespaceURI(), toTag);
         count_perms++;
      }

}
if(children.item(p).getNodeName().compareToIgnoreCase("hierarchy")==0){
          if (children.item(p) instanceof Element) {
                    String toTag_temp = String.valueOf(count_hierarchies);
          String toTag=children.item(p).getNodeName().concat(toTag_temp);
                    Element elem = (Element)children.item(p);
          doc.renameNode(elem, elem.getNamespaceURI(), toTag);
```

```
                    count_hierarchies++;
                }

            }
            if(children.item(p).getNodeName().compareToIgnoreCase("disclose")==0){
                    if (children.item(p) instanceof Element) {
                            String toTag_temp = String.valueOf(count_disclose);
                    String toTag=children.item(p).getNodeName().concat(toTag_temp);
                            Element elem = (Element)children.item(p);
                     doc.renameNode(elem, elem.getNamespaceURI(), toTag);
                    count_disclose++;
                }

            }

        }
        }
            for (i=0;i<children.getLength();i++) {
                    child = children.item(i);
              if (child.getNodeType()==1) change_tags((Element)child,doc);
                    }

    }

    public static void TreeTraverse(Node node,Document doc) throws
XPathExpressionException {
                int i;
                    Node child;
                    NodeList children;

                    children = node.getChildNodes();
                    System.out.println("Element " + node.getNodeName() + " has "
+children.getLength() + " children which are:->>>");
                    /*print children of a specific node*/
                for(int p=0; p<children.getLength(); p++){
                  if(children.item(p).getNodeName()!="#text"){
```

```
                    XPath xpath = XPathFactory.newInstance().newXPath();
                                //vrisko to text value gia kathe node
                                XPathExpression expression =
xpath.compile("//"+children.item(p).getNodeName()+"/text()[1]");
                                NodeList nodes = (NodeList)
expression.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);

System.out.print("<"+children.item(p).getNodeName()+">");
                                //System.out.println("length="+nodes.getLength());
                                for(int x=0; x<nodes.getLength(); x++){
                                        //System.out.println("x="+x);

//System.out.println("AAAAAA=="+children.item(p).getNodeName().compareToIgnoreCas
e("proccess"));
                                                //
if(children.item(p).getNodeName().compareToIgnoreCase("proccess")!=0)
                                        System.out.println("attribute=
"+nodes.item(x).getNodeValue());
                                }

                        }
                        }
                        System.out.print("\n");
                            for (i=0;i<children.getLength();i++) {
                                child = children.item(i);
                              if (child.getNodeType()==1) TreeTraverse((Element)child,doc);
                                }

                }

        /*for all i,1<=i<n, if G is in groups(Hi) then G must not int desc(G,Hi)*/
        private static void Definiton2_rule2(Node node,Document doc) throws
XPathExpressionException {
          int i;
                Node child;
                NodeList children;
```

```
children = node.getChildNodes();/*get nodes children*/
for (i=0;i<children.getLength();i++) {
  child = children.item(i);/*get child node*/
      if(child.getNodeName()!="#text" ){/*if child node name is not null*/
        /*if node's name is hierarchy and has children*/
          if(child.getNodeName().contains("group") && child.getParentNode().getLastChild().getChildNodes()!=null){
                XPath xpath1 = XPathFactory.newInstance().newXPath();
                /*find text value of the node*/
              XPathExpression expression1 = xpath1.compile("//"+child.getNodeName()+"/text()[1]");
                NodeList nodes1 = (NodeList) expression1.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                XPath xpath = XPathFactory.newInstance().newXPath();
                /*find hierarchy descedants*/
              XPathExpression expression = xpath.compile("//"+child.getParentNode().getLastChild().getNodeName()+"/descendant::*");
                NodeList nodes = (NodeList) expression.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                for(int x=0; x<nodes.getLength(); x++){
                    /*if one of the descedants contains the node name group*/

        if(nodes.item(x).getNodeName().contains("group")){
                    XPath xpath2 = XPathFactory.newInstance().newXPath();
                    /*then we find its text value to check if its the same with the current group text value*/
                    XPathExpression expression2 = xpath2.compile("//"+nodes.item(x).getNodeName()+"/text()[1]");
```

```
                                                    NodeList nodes2 =
(NodeList) expression2.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);

                                                    /*if one group has the same
name with its descedants group then policy is not in well formed*/

        if(nodes1.item(0).getNodeValue().compareToIgnoreCase(nodes2.item(0).getNodeVal
ue())==0){

System.out.println("DEFINITION 2,2 VIOLATED.THE POLICY IS NOT IN WELL-
FORMED");
                                                        System.exit(0);
                                                            }
                                                        }

                                                    }
                }
                                    }
                                if                          (child.getNodeType()==1)
Definiton2_rule2((Element)child,doc);/*recursive call*/

                                    }

                }

        /*Definition2,rule 3for all i,1<=i<=n, if G is in groups(Hi),nondisclose is in
perms(G,Hi)
         *  and disclose G' λ is in perms(G",Hi) for some G" is in desc(G,Hi) then G' must in
desc(G,Hi)*/
        private static void  Definiton2_rule3(Node node,Document doc) throws
XPathExpressionException {
          int i;
              Node child;
              NodeList children,children_perm;
              boolean part1,part2=false;
```

```
children = node.getChildNodes();/*get children of the node*/

for (i=0;i<children.getLength();i++) {
  child = children.item(i);/*gets children*/
      if(child.getNodeName()!="#text" ){/*if nodename is not null*/
        /*if we found node with the permissions and its children its not null*/
          if(child.getNodeName().contains("perm") && child.getChildNodes()!=null){
            children_perm=child.getChildNodes();/*get the permissions of node perm*/
            for(int u=0; u<children_perm.getLength(); u++){
              /*if one of the permission is nondisclose*/

if(children_perm.item(u).getNodeName().compareToIgnoreCase("nondisclose")==0){
                XPath xpath0 = XPathFactory.newInstance().newXPath();
                /*we find the text value of the group with this permission*/
                XPathExpression expression0 = xpath0.compile("//"+child.getParentNode().getFirstChild().getNodeName()+"/text()[1]");
                  NodeList nodes01 = (NodeList) expression0.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                  XPath xpath1 = XPathFactory.newInstance().newXPath();
                  /*we also find the descedants of the group that has the permission nondisclose*/
                XPathExpression expression1 = xpath1.compile("//"+child.getParentNode().getLastChild().getNodeName()+"/descendant::*");
                NodeList nodes1 = (NodeList) expression1.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                for(int g=0; g<nodes1.getLength(); g++){
                  /*if one of the descedants contains disclose*/
```

```
if(nodes1.item(g).getNodeName().contains("disclose") &&
nodes1.item(g).getNodeName()!="nondisclose"){
                                        XPath xpath3 =
XPathFactory.newInstance().newXPath();
                                        /*we get the textvalue of the group that
contains disclose*/
                                            XPathExpression expression3 =
xpath3.compile("//"+nodes1.item(g).getNodeName()+"/text()[1]");
                                            NodeList nodes3 = (NodeList)
expression3.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                                            /*if disclose group text value there is
on descedants or in the group that has
                                             * the permission nondisclose then
policy is in well formed otherwise is not
                                             * in well formed*/
                                        for(int p=0; p<nodes1.getLength();
p++){
                                         /*if one of the descedants contains the
name group*/

if(nodes1.item(p).getNodeName().contains("group")){
                                                XPath xpath4 =
XPathFactory.newInstance().newXPath();
                                                /*found text value of the
node*/
                                                    XPathExpression
expression4 = xpath4.compile("//"+nodes1.item(p).getNodeName()+"/text()[1]");
                                                    NodeList nodes4 =
(NodeList) expression4.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);

                                                    /*part1 has the answer
if group disclose name has the first name with gph*/

part1=nodes3.item(0).getNodeValue().contains(nodes01.item(0).getNodeValue());
```

```
                                                    /*part2 contains the
answer if group disclose has the same name with
                                                     * one of its
descedants*/

        part2=nodes3.item(0).getNodeValue().contains(nodes4.item(0).getNodeValue());
                                                   /*if group that has perm
disclose there is again in one of the descedants group
                                                    * or from the group that
has perm nondisclose then policy is in well formed*/
                                                    if(part1==true ||
part2==true){
                                                      answer=1;
                                                      break;
                                                    }
                                                    else{
                                                      answer=0;
                                                    }
                                                }
                                            }
                                          if(answer==0){
                                            System.out.println("DEFINITION 2,3
VIOLATED.THE POLICY IS NOT IN WELL-FORMED");
                                                    System.exit(0);
                                          }

                                        }
                                      }
                                    }
                                  }
                                }
                              }//end if

            if (child.getNodeType()==1) Definiton2_rule3((Element)child,doc);
    }
  }
```

```
        /*extracts the groups from the tree to an arraylist,to check if a set of groups
         *contains in the arraylist or no
         *Returns the arraylist contains the current groups of the tree*/
        public static ArrayList<String> find_current_groups_inthetree(Node node,Document doc,ArrayList<String> current_groups) throws XPathExpressionException{
                int i;
                Node child;
                NodeList children;

      children = node.getChildNodes();
                for(int p=0; p<children.getLength(); p++){
                  if(children.item(p).getNodeName()!="#text"){/*if children node name is not null*/
                    XPath xpath = XPathFactory.newInstance().newXPath();
                    /*find the text value for each node*/
                            XPathExpression expression = xpath.compile("//"+children.item(p).getNodeName()+"/text()[1]");
                            NodeList nodes = (NodeList) expression.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                            for(int x=0; x<nodes.getLength(); x++){
                                /*add the text value of node that has the tag group to "current groups" arraylist*/
                                if(children.item(p).getNodeName().contains("group")){

if(!current_groups.contains(nodes.item(x).getNodeValue())){

current_groups.add(nodes.item(x).getNodeValue());
                                        }
                                }
                        }
                            }
                }
                for (i=0;i<children.getLength();i++) {
                  child = children.item(i);
                        if (child.getNodeType()==1)
```

```
                    find_current_groups_inthetree((Element)child,doc,current_groups);

                }
                return current_groups;
    }


        /*check if the groups of the arraylist set_of_groups are also in the tree*/
        public static void check_if_groupset_isok(ArrayList<String> set_of_groups,ArrayList<String> current_groups){
                boolean found=false;/*flag to check if a group of the set contained also in the tree*/
                for (int i = 0; i < set_of_groups.size(); i++) {/*for every group in the given set*/
                    for(int j = 0; j < current_groups.size(); j++){/*we check if the group of the set there is on the tree*/
                        if(set_of_groups.get(i).equalsIgnoreCase(current_groups.get(j))){
                            found=true;
                        }
                    }
                    if(found==false){/*means that the specific group of the set is not contained in the tree*/
                        System.out.println("Group "+set_of_groups.get(i)+" is not in the tree.ERROR!!!");
                        System.exit(0);/*so,program exits*/
                    }
                    found=false;/*initialize again found to check the next group of the set*/
                }
        }

        //figure 2, permission extraction relation
        //input: a set of groups and a base type,the node to be start the traversal that must be the base type
        //and the document element
        //extracts the permission relation of a given set of groups and a base types
```

```
public static ArrayList<String> permission_extraction_relation(Node node,Document doc,ArrayList<String> set_of_groups, ArrayList<String> permissions) throws XPathExpressionException{
		int i;
			Node child;
			NodeList children,c,c1,perm_children,hier_children,permnode_children;
			int count=0,flag=0;
			String per;

			//to check the first time that we dont have hierarchy the group0
		if(node.getNodeName().compareToIgnoreCase("base_type")==0){
			//System.out.println("NAMEEEEE="+node.getNodeName());
			c=node.getChildNodes();//get the children of the base type
			for(int u=0; u<c.getLength(); u++){
				 if(c.item(u).getNodeName()!="#text"){//if the text value is not #text

//System.out.println("CCCCCCC="+c.item(u).getNodeName());
					//System.out.println(c.item(u).getNodeName()+" first child=="+c.item(u).getFirstChild().getNodeName());
					XPath xpath1 = XPathFactory.newInstance().newXPath();
					//find the text value of the node we want(the first group of the sub-tree we are)
					XPathExpression expression1 = xpath1.compile("//"+c.item(u).getFirstChild().getNodeName()+"/text()[1]");
					NodeList nodes1 = (NodeList) expression1.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);

//System.out.println("TEXTTTTTTTTT=="+nodes1.item(0).getNodeValue());
					//check if the text value of the group is a group of the given set
					for (int j = 0; j < set_of_groups.size(); j++) {
					    //System.out.println("Item: " + set_of_groups.get(j));
					    //if the group contained in the set of groups

if(nodes1.item(0).getNodeValue().compareToIgnoreCase(set_of_groups.get(j))==0){
```

```
                                        //System.out.println("YESSS"+"
g1=="+nodes1.item(0).getNodeValue()+"  g2=="+set_of_groups.get(j));
                                        c1=c.item(u).getChildNodes();//get the children of
gph0
                                        for(int r=0; r<c1.getLength(); r++){

if(c1.item(r).getNodeName().contains("perm")){//find perm node to get the permissions of
the specific group

//System.out.println("ppp==="+c1.item(r).getNodeName());

perm_children=c1.item(r).getChildNodes();//get the children of node perm
                                                for(int k=0;
k<perm_children.getLength(); k++){//check the permissions of perm node

//System.out.println("permission=="+perm_children.item(k).getNodeName());

if(perm_children.item(k).getNodeName().compareToIgnoreCase("nondisclose")==0){
                                                        String
perm=perm_children.item(k).getNodeName()+" "+nodes1.item(0).getNodeValue();

        //System.out.println("pppp=="+perm);
                                                        permissions.add(perm);//add
permission to the arraylist with the permissions

        //System.out.println("permissions=="+permissions+"\n\n");
                                                }

if(perm_children.item(k).getNodeName().compareToIgnoreCase("read")==0){

//System.out.println("rrr="+perm_children.item(k).getNodeName());

permissions.add(perm_children.item(k).getNodeName());//add permission to the arraylist
with the permissions
                                                }
```

```
if(perm_children.item(k).getNodeName().compareToIgnoreCase("write")==0){

//System.out.println("www="+perm_children.item(k).getNodeName());

permissions.add(perm_children.item(k).getNodeName());//add permission to the arraylist with the permissions
                                                    }

if(perm_children.item(k).getNodeName().compareToIgnoreCase("access")==0){

//System.out.println("aaa="+perm_children.item(k).getNodeName());

permissions.add(perm_children.item(k).getNodeName());//add permission to the arraylist with the permissions
                                                    }

if(perm_children.item(k).getNodeName().contains("disclose") && perm_children.item(k).getNodeName()!="nondisclose"){
                                                        XPath xpath2 = XPathFactory.newInstance().newXPath();
                                                        //find the text value of the node we want(the first group of the sub-tree we are)
                                                        XPathExpression expression2 = xpath2.compile("//"+perm_children.item(k).getNodeName()+"/text()[1]");
                                                        NodeList nodes2 = (NodeList) expression2.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                                                        String[] part = nodes2.item(0).getNodeValue().split("(?<=\\D)((?=\\*)|(?=\\d))");
                                                    //System.out.println("split=="+part[0] + "-" + part[1]);

//System.out.println("valueeee="+nodes2.item(0).getNodeValue());
                                                        String perm1="disclose "+part[0]+" "+part[1];
```

```
//System.out.println("permmm1=="+perm1);
                                                    permissions.add(perm1);//add
permission to the arraylist with the permissions
                                                }

                                            }

                                        }
                                    }
                                    break;//we dont need to check the whole set of groups
                                }
                            }
                        }
                    }


            }

                        children = node.getChildNodes();

                        /*print children of a specific node*/
                    for(int p=0; p<children.getLength(); p++){
                      if(children.item(p).getNodeName()!="#text"){
                        //if node name contains hierarchy
                        if(children.item(p).getNodeName().contains("hierarchy")){
                                hier_children=children.item(p).getChildNodes();//get the
children of hierarchy
                                for(int w=0; w<hier_children.getLength(); w++){
                                  if(hier_children.item(w).getNodeName()!="#text"){
                                    XPath xpath3 =
XPathFactory.newInstance().newXPath();
                                        //find the text value of the node we want(the
first group of the sub-tree we are)
```

```
XPathExpression expression3 = xpath3.compile("//"+hier_children.item(w).getFirstChild().getNodeName()+"/text()[1]");
NodeList nodes3 = (NodeList) expression3.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
// System.out.println("hhh child text value=="+nodes3.item(0).getNodeValue());
for(int t=0; t<set_of_groups.size(); t++){
//if the group of the set is the specific group in the tree

if(nodes3.item(0).getNodeValue().compareToIgnoreCase(set_of_groups.get(t))==0){
//then we get the perm node children in order to check what permissions we have

permnode_children=hier_children.item(w).getFirstChild().getNextSibling().getChildNodes();
for(int h=0; h<permnode_children.getLength(); h++){
//if the permission is "access" and there isnt already in the arraylist then we add it

if(permnode_children.item(h).getNodeName().compareToIgnoreCase("access")==0
&&
permissions.contains("access")==false){

permissions.add(permnode_children.item(h).getNodeName());//add permission to the arraylist with the permissions
}
//if the permission is "read" and there isnt already in the arraylist then we add it

if(permnode_children.item(h).getNodeName().compareToIgnoreCase("read")==0
&&
permissions.contains("read")==false){

permissions.add(permnode_children.item(h).getNodeName());//add permission to the arraylist with the permissions
}
```

```
                                        //if the permission is "write" and there isnt already in the arraylist then we add it

if(permnode_children.item(h).getNodeName().compareToIgnoreCase("write")==0
                                                                    &&
permissions.contains("write")==false){

permissions.add(permnode_children.item(h).getNodeName());//add permission to the arraylist with the permissions
                                                                }

if(permnode_children.item(h).getNodeName().compareToIgnoreCase("nondisclose")==0){
                                                            if(count==0){//we are in the first group of the current hierarchy
                                                                for (int v = 0; v < permissions.size(); v++) {

if(permissions.get(v).contains("nondisclose")){
                                                        // removing the first occurrence of item permissions.get(v)

permissions.remove(permissions.get(v));//remove the nondisclose is on the permissions arraylist

                                                                }

                                                            }
                                                                XPath xpath4 = XPathFactory.newInstance().newXPath();
                                                                //find the text value of the node we want(the first group of the sub-tree we are)
                                                                XPathExpression expression4 = xpath4.compile("//"+hier_children.item(w).getFirstChild().getNodeName()+"/text()[1]");
                                                                NodeList nodes4 = (NodeList) expression4.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
```

```
                                        //make the string nondisclose
group so we can add it into the permissions arraylist

per=permnode_children.item(h).getNodeName()+" "+nodes4.item(0).getNodeValue();
                                            permissions.add(per);
                                count=1;
                                        }
                                        //the group of the hierarchy is a
sibling so we just also add the nondisclose group
                                        else{
                                            XPath xpath5 =
XPathFactory.newInstance().newXPath();
                                            //find the text value of
the node we want(the first group of the sub-tree we are)
                                            XPathExpression
expression5 =
xpath5.compile("//"+hier_children.item(w).getFirstChild().getNodeName()+"/text()[1]");
                                            NodeList nodes5 =
(NodeList) expression5.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);

per=permnode_children.item(h).getNodeName()+" "+nodes5.item(0).getNodeValue();
                                            permissions.add(per);
                                        }


                                        }

if(permnode_children.item(h).getNodeName().contains("disclose")&&
permnode_children.item(h).getNodeName()!="nondisclose"){
                                        XPath xpath6 =
XPathFactory.newInstance().newXPath();
                                            //find the text value of
the node we want(the first group of the sub-tree we are)
                                            XPathExpression
expression6 = xpath6.compile("//"+permnode_children.item(h).getNodeName()+"/text()[1]");
```

```
NodeList nodes6 = (NodeList) expression6.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
//split the group and the number,eg if we have ETP2 then we split into ETP and 2 separately
//split the group based on the char and then based on delimiter "*" or a number from 0-9
String[] parts = nodes6.item(0).getNodeValue().split("(?<=\\D)((?=\\*)|(?=\\d))");
for (int v = 0; v < permissions.size(); v++) {
//check if disclose group is already added in permissions arraylist

if(permissions.get(v).contains("disclose "+parts[0]) && permissions.get(v).contains("nondisclose")==false){
flag=1;
String splited[]=permissions.get(v).split("[\\s]+");
//we do the calculation only if the disclose on the tree does not contain *
//or if the permission in the arraylist also does not contain *,otherwise we dont
//need to do the calculation because * is bigger than any nymber

if(splited[2].compareTo("*")!=0 && parts[1].compareTo("*")!=0){

permissions.remove(permissions.get(v));//remove it to calculate the new one disclose group
int x1 = Integer.parseInt(splited[2]);
int x2 = Integer.parseInt(parts[1]);
int sum=x1+x2;

permissions.add("disclose "+parts[0]+" "+sum);
```

```
                                                                    }
                                                                    else{
                                                                        //if the
new group we will add has * then we just add it
                                                                        //and we
dont need to do any calculations

if(parts[1].compareTo("*")==0){

permissions.remove(permissions.get(v));//remove it to calculate the new one disclose group

permissions.add("disclose "+parts[0]+" "+parts[1]);
                                                                        }
                                                                    }
                                                                    break;//need
break
                                                                }
                                                            }
                                                        //if its the first we just add the
disclose group and the number of possible disclosures
                                                            if(flag==0){
                                                            String permm="disclose
"+parts[0]+" "+parts[1];

permissions.add(permm);
                                                        }
                                                        flag=0;

                                                    }
                                                }
                                            }
                                        }
                                    }

                                }
```

```
                    }
                    count=0;
                  }
                }

                    for (i=0;i<children.getLength();i++) {
                        child = children.item(i);
                      if (child.getNodeType()==1)
permission_extraction_relation((Element)child,doc,set_of_groups, permissions);
                        }
                    return permissions;
        }

        //compare system and policy permissions
        public static String compare_permissions( ArrayList<String> system,
ArrayList<String> policy){
              //  System.out.println("POLICY=="+policy+"    SYSTEM==="+system);
               int flag=0;
               String result=" ";
               String fot[]=new String[3];
                for(int i=0; i<system.size(); i++){
                    //System.out.println("systemmmmmm=="+system.get(i));
                        if(system.get(i).matches(".*[a-z].*")){//ean den einai empty to
string
                            //System.out.println("IN");
                            //System.out.println("policy=="+policy+"
system="+system.get(i));

      //System.out.println("ddd="+policy.contains(system.get(i)) );
                            //system.get(i)
                              fot=system.get(i).split("[\\s]");
                             /* for(int t=0; t<fot.length; t++){
                                  System.out.println("fottttt="+fot[t]);
                              }*/

                              if(fot.length==2){
```

```
                                        system.set(i, fot[fot.length-1]);
                                    }
                                if (policy.contains(system.get(i)) ){
                                        if(system.get(i).contains("disclose")          &&
policy.contains("nondisclose") ){

                                        }

                                        else{
                                                //System.out.println("AAAAA");
                                         flag=1;
                                         //System.out.println("\n\n"+policy+"          >=
"+system);
                                        }
                                }
                                else{
                                        flag=0;
                                        //System.out.println("BBBB");
                                        //System.out.println("\n\n"+policy+"             <
"+system);
                                        break;
                                }
                        }
                }
                if(flag==0){
                        result="\n\n"+policy+"  <  "+system;
                        result+="To sistima den ikanopoia tin politiki";
                            // System.out.println("\n\n"+policy+"  <  "+system);
                            // System.out.println("To sistima den ikanopoia tin politiki");

                    }
                else{
                         result="\n\n"+policy+"  >=  "+system;
                         result+="To sistima ikanopoia tin politiki";
                        //System.out.println("\n\n"+policy+"  >=  "+system);
                        //System.out.println("To sistima ikanopoia tin politiki");
```

```
                }
                return result;
                //System.exit(0);
        }

                public static void policy_parse( ){
                    try {
                     sistima[] set=     system_parse ();
                            File file1 = new File("centralized_policy.xml");/*get the privacy policy xml file*/
                            DocumentBuilderFactory dbf=DocumentBuilderFactory.newInstance();
                         DocumentBuilder dBuilder1 = dbf.newDocumentBuilder();/* Set up the parser*/
                                    Document doc_privacy_policy = dBuilder1.parse(file1);/*parse the xml file*/

doc_privacy_policy.getDocumentElement().normalize();/*Normalize the XML Structure*/

                           /*IGNORING WHITESPACES for doc_privacy_policy*/
                            XPathFactory xpathFactory = XPathFactory.newInstance();
                            /*XPath to find empty text nodes.*/
                            XPathExpression xpathExp = xpathFactory.newXPath().compile("//text()[normalize-space(.) = "]");
                            NodeList emptyTextNodes = (NodeList)
                            xpathExp.evaluate(doc_privacy_policy, XPathConstants.NODESET);
                            /*Remove each empty text node from document*/
                            for (int i = 0; i < emptyTextNodes.getLength(); i++) {
                                Node emptyTextNode = emptyTextNodes.item(i);

emptyTextNode.getParentNode().removeChild(emptyTextNode);
                                }
```

```
change_tags(doc_privacy_policy.getDocumentElement(),doc_privacy_policy);//change the tags
                        //System.out.println("\n\nTRAVERSEEEEEEEEEEE");

//TreeTraverse(doc_privacy_policy.getDocumentElement(),doc_privacy_policy);
                        well_formed_policy (doc_privacy_policy);/*check if the set of policies are well formed*/


                        ArrayList<String> set_of_groups = new ArrayList<String>();//set of groups given as argument
                        ArrayList<String> current_groups = new ArrayList<String>();//groups are in the tree
                        ArrayList<String> permissions=new ArrayList<String>();//permissions extracted
                        String result=" ";
                       // set_of_groups.add("ETP");
                        //set_of_groups.add("OBE");
                        //set_of_groups.add("car");
                        //set_of_groups.add("PA");
                        //set_of_groups.add("home");
                        //set_of_groups.add("GPS");
                        //set_of_groups.add("SC");
                       System.out.println("\n ------Policy permissions are------:");
                       for(int f=0; f<set.length; f++){
                          if(!set[f].groups.isEmpty()){
                                        set_of_groups.addAll(set[f].groups);

current_groups=find_current_groups_inthetree(doc_privacy_policy.getDocumentElement(),doc_privacy_policy,current_groups);

check_if_groupset_isok(set_of_groups,current_groups);

permissions=permission_extraction_relation(doc_privacy_policy.getDocumentElement().getFirstChild(),doc_privacy_policy,set_of_groups,permissions);
```

```
                                        // Display the contents of the array list
                                        for(int x=0; x<set_of_groups.size(); x++){
                                                System.out.print(set_of_groups.get(x));
                                               if(set_of_groups.size()-1!=x){
                                               System.out.print(". ");
                                        }
                                        }
                                        System.out.println(permissions);


                                    }
                                //System.out.println("*****\n\nsystem
perms==="+set[f].com+"policy  perms=="+permissions+"\n\n");
                                if(!set[f].com.isEmpty()){
                                      result+=compare_permissions(set[f].com,permissions);
                                }

                                set_of_groups.clear();
                                current_groups.clear();
                                permissions.clear();

                              }
                               System.out.println(result);

//final_permission_extraction(doc_privacy_policy.getDocumentElement(),doc_privacy_polic
y,set_of_groups,permissions,current_groups);

                        } catch (Exception e) {
                                System.out.println(e.getMessage());
                            }
                    }

                /*Main*/
                      public static void main(String[] args) {
                              policy_parse();
                      }
```

/*end main*/

}

Παράρτημα Β-κώδικας για την κλάση του συστήματος

Στο παράρτημα αυτό παρουσιάζεται ο κώδικας για την κλάση του συστήματος

```
package diplomatiki;
import java.io.File;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import java.text.DateFormat.Field;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Collections;
import java.util.List;
```

```
import java.util.ListIterator;
import java.util.Map;
import java.util.Iterator;
import java.util.Set;
import java.util.Stack;
import java.lang.Object;

public class sistima {
   static int group=0, name=0, type=0, repl=0, subj=0, obj=0,
nil=0,proccess=0,parP=0,resGP=0;
   // Create new HashMap with key String and value String.
         static HashMap<String, String> hash = new HashMap<String, String>();
   ArrayList<String> Perms = new ArrayList<String>();
   static ArrayList<String> basetypes = new ArrayList<String>();
   static HashMap<String, List<String>> compare= new HashMap<String, List<String>>();
   ArrayList<String> com = new ArrayList<String>();
   ArrayList<String> groups = new ArrayList<String>();
   String base_type=null;
   static int pos=0;
   static int poss=0;
   static int count=0;

   public static sistima[] initialization(){
      sistima [] final_permissions = new sistima[10];
      for(int i = 0; i < final_permissions.length; i++) {
         final_permissions[i] =new sistima();
         final_permissions[i].base_type=null;
         for(int j=0; j<compare.size(); j++){
                  final_permissions[i].compare.put(null, null);
         }

      }
      return final_permissions;
   }
```

```
public static sistima []  system_parse () throws ParserConfigurationException, SAXException, IOException, XPathExpressionException{
    File file = new File("system_parse.xml");/*get the system xml file*/
    DocumentBuilder dBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();/* Set up the parser*/
    Document doc_system = dBuilder.parse(file);/*parse the xml file*/
    doc_system.getDocumentElement().normalize();/*Normalize the XML Structure; It's just too important !!*/

    unique_tag(doc_system.getDocumentElement(),doc_system);/*make tag uniques*/
    sistima [] final_permissions = new sistima[10];//domi 1
    sistima [] parP_permissions = new sistima[10];
    sistima [] policy_setofgroups = new sistima[10];
    sistima[] thita_permissions=new sistima[10];
    final_permissions=initialization();
    parP_permissions=initialization();
    policy_setofgroups=initialization();
    thita_permissions=initialization();

PostOrderTraversal(doc_system.getDocumentElement(),doc_system,final_permissions,parP_permissions);
    System.out.println("\n------System permissions are------:");
    System.out.print("Θ= ");
    Stack s= new Stack();
    ArrayList<String> nodename = new ArrayList<String>();
    ArrayList<String> nodepermission = new ArrayList<String>();
    ArrayList<String> set = new ArrayList<String>();


diasxisi(doc_system.getDocumentElement(),doc_system,s,nodename,nodepermission,set,policy_setofgroups,thita_permissions);

    List<String> list = new ArrayList<String>(s);
```

```
for (String y : list){
        String[] split = y.split("[:]");
        for(int j=0; j<split.length; j++){

                if(j==0){
                        nodename.add(split[j]);
                }
                else{
                        nodepermission.add(split[j]);
                }

        }


}

List<String> p=new ArrayList <String>();
        thita_permissions[poss].groups.addAll(nodename);//kanw add px to ETP,PA

        String splited2[]=new String[8];
        /******/ String bpolitiki="Loc";
        //System.out.println("NODE===="+nodepermission);
        String[] part = nodepermission.toString().split("(?<=\\D)(?=\\])");
         for(int r=0; r<part.length; r++){
                //System.out.println("QQQQ="+part[r]);
                if(part[r].contains(bpolitiki)){//we get the permissions that has the base type of policy
                        //System.out.println("AAAAA="+part[r]);
                         String strp = part[r].replace("[", "$").replace("]", ".");
         splited2=strp.split("[$.,]");
        //System.out.println("PART=="+Arrays.toString(splited2));
                }
         }

         for(int i=0; i<splited2.length; i++){
```

```
                        if(!basetypes.contains(splited2[i]) && splited2[i]!=" "){//ean den einai
base type
                                //System.out.println("CCCC="+splited2[i]);
                                thita_permissions[poss].com.add(splited2[i]);
                        }
                    }
                    poss++;
                    policy_setofgroups[pos].Perms.addAll(nodename);
                System.out.print("< ");
                for(int k=0; k<nodename.size(); k++){
                        System.out.print(nodename.get(k));
                        if(nodename.size()-1!=k){
                                System.out.print(". ");
                        }
                }
                System.out.print(", ");
                for(int k=0; k<nodepermission.size(); k++){
            System.out.print(nodepermission.get(k));
            if(nodepermission.size()-1!=k){
                                System.out.print(". ");
                        }
    }
            System.out.println(">");
            return thita_permissions;
}


    /*modify tags to be unique*/
    private static void unique_tag(Node node,Document doc) throws
XPathExpressionException {
        int i = 0;
        Node child = null;
        NodeList children = null;

        children = node.getChildNodes();
        /*print children of a specific node*/
```

```
for(int p=0; p<children.getLength(); p++){
  if(children.item(p).getNodeName()!="#text"){

    if(children.item(p).getNodeName().compareToIgnoreCase("group")==0){
      if (children.item(p) instanceof Element) {
        String toTag_temp = String.valueOf(group);
        String toTag=children.item(p).getNodeName().concat(toTag_temp);
         Element elem = (Element)children.item(p);
        doc.renameNode(elem, elem.getNamespaceURI(), toTag);
        group++;
      }
    }
    if(children.item(p).getNodeName().compareToIgnoreCase("name")==0){
      if (children.item(p) instanceof Element) {
        String toTag_temp = String.valueOf(name);
        String toTag=children.item(p).getNodeName().concat(toTag_temp);
         Element elem = (Element)children.item(p);
         doc.renameNode(elem, elem.getNamespaceURI(), toTag);
         name++;
      }
    }
    if(children.item(p).getNodeName().compareToIgnoreCase("type")==0){
      if (children.item(p) instanceof Element) {
        String toTag_temp = String.valueOf(type);
        String toTag=children.item(p).getNodeName().concat(toTag_temp);
         Element elem = (Element)children.item(p);
         doc.renameNode(elem, elem.getNamespaceURI(), toTag);
         type++;
      }
    }
    if(children.item(p).getNodeName().compareToIgnoreCase("repl")==0){
      if (children.item(p) instanceof Element) {
        String toTag_temp = String.valueOf(repl);
        String toTag=children.item(p).getNodeName().concat(toTag_temp);
         Element elem = (Element)children.item(p);
         doc.renameNode(elem, elem.getNamespaceURI(), toTag);
```

```
            repl++;
        }
    }
    if(children.item(p).getNodeName().compareToIgnoreCase("subj")==0){
        if (children.item(p) instanceof Element) {
            String toTag_temp = String.valueOf(subj);
            String toTag=children.item(p).getNodeName().concat(toTag_temp);
             Element elem = (Element)children.item(p);
            doc.renameNode(elem, elem.getNamespaceURI(), toTag);
            subj++;
         }
    }
    if(children.item(p).getNodeName().compareToIgnoreCase("obj")==0){
        if (children.item(p) instanceof Element) {
            String toTag_temp = String.valueOf(obj);
            String toTag=children.item(p).getNodeName().concat(toTag_temp);
             Element elem = (Element)children.item(p);
            doc.renameNode(elem, elem.getNamespaceURI(), toTag);
            obj++;
        }
    }
    if(children.item(p).getNodeName().compareToIgnoreCase("nil")==0){
        if (children.item(p) instanceof Element) {
            String toTag_temp = String.valueOf(nil);
            String toTag=children.item(p).getNodeName().concat(toTag_temp);
             Element elem = (Element)children.item(p);
             doc.renameNode(elem, elem.getNamespaceURI(), toTag);
             nil++;
        }
    }
    if(children.item(p).getNodeName().compareToIgnoreCase("proccess")==0){
        if (children.item(p) instanceof Element) {
            String toTag_temp = String.valueOf(proccess);
            String toTag=children.item(p).getNodeName().concat(toTag_temp);
             Element elem = (Element)children.item(p);
             doc.renameNode(elem, elem.getNamespaceURI(), toTag);
```

```
                proccess++;
            }
          }
          if(children.item(p).getNodeName().compareToIgnoreCase("parP")==0){
            if (children.item(p) instanceof Element) {
                String toTag_temp = String.valueOf(parP);
                String toTag=children.item(p).getNodeName().concat(toTag_temp);
                 Element elem = (Element)children.item(p);
                 doc.renameNode(elem, elem.getNamespaceURI(), toTag);
                parP++;
            }
          }
          if(children.item(p).getNodeName().compareToIgnoreCase("resGP")==0){
            if (children.item(p) instanceof Element) {
                String toTag_temp = String.valueOf(resGP);
                String toTag=children.item(p).getNodeName().concat(toTag_temp);
                 Element elem = (Element)children.item(p);
                 doc.renameNode(elem, elem.getNamespaceURI(), toTag);
                 resGP++;
            }
          }

     }
    }
     for (i=0;i<children.getLength();i++) {
          child = children.item(i);
          if (child.getNodeType()==1) unique_tag((Element)child,doc);
     }
}



  public static boolean check_if_isempty( sistima [] final_permissions){
        boolean answer=false;
        for(int x=0; x<final_permissions.length; x++){
                 if(final_permissions[x].base_type!=null){
```

```
                        answer=true;
                }
        }
        return answer;
    }

    //adiazo ti domi me ta base type kai ta permissions
    public static void make_empty(sistima [] final_permissions){
        for(int x=0; x<final_permissions.length; x++){
                final_permissions[x].base_type=null;
                final_permissions[x].Perms.clear();
        }
    }

    public static void diasxisi(Node node,Document doc,Stack s,ArrayList<String> nodename,ArrayList<String> nodepermission,ArrayList<String> set, sistima [] policy_setofgroups,sistima [] thita_permissions) throws XPathExpressionException{

        if (node == null || node.getNodeName() == null)
          return;

        // Do PostOrder on all children
        diasxisi(node.getFirstChild(),doc,s,nodename,nodepermission,set, policy_setofgroups,thita_permissions);

        // Now that all children were traversed, process the current node:
        if(node.getNodeName()!="#text"){

            XPath xpath = XPathFactory.newInstance().newXPath();
            XPathExpression expression = xpath.compile("//"+node.getNodeName()+"/text()[1]");
            NodeList nodes = (NodeList) expression.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
            if(node.getNodeName().contains("group")){
                    s.push(nodes.item(0).getNodeValue());
                NodeList children= node.getParentNode().getChildNodes();
```

```
for(int x=0; x<children.getLength(); x++){
    if(children.item(x).getNodeName().contains("parP")){
        List<String> list = new ArrayList<String>(s);

          for (String y : list){
              String[] split = y.split("[:]");
              for(int j=0; j<split.length; j++){
                  if(j==0){
                      nodename.add(split[j]);
                  }
                  else{
                      nodepermission.add(split[j]);
                  }
              }
          }
          List<String> p=new ArrayList <String>();
          //replace special characters like [ and ]

          //System.out.println("PART=="+Arrays.toString(part));
      //String str = nodepermission.toString().replace("[", "$").replace("]", ".");
          //String splited1[]=str.split("[$.,]");
          thita_permissions[poss].groups.addAll(nodename);//kanw add px to ETP,PA
         String splited2[]=new String[8];
          /******/ String bpolitiki="Loc";
          //System.out.println("NODE===="+nodepermission);
         String[] part = nodepermission.toString().split("(?<=\\D)(?=\\])");
          for(int r=0; r<part.length; r++){
              //System.out.println("QQQQ="+part[r]);
              if(part[r].contains(bpolitiki)){//we get the permissions that has the base type of policy
                  //System.out.println("AAAAA="+part[r]);
                  String strp = part[r].replace("[", "$").replace("]", ".");
                splited2=strp.split("[$.,]");
                //System.out.println("PART=="+Arrays.toString(splited2));
              }
```

```
                }

                for(int i=0; i<splited2.length; i++){
                    if(!basetypes.contains(splited2[i]) && splited2[i]!="  "){//ean
den einai base type
                        //System.out.println("CCCC="+splited2[i]);
                        thita_permissions[poss].com.add(splited2[i]);
                    }
                }
                /*******/
               //System.out.println("FFFFFFFFFF=="+Arrays.toString(splited1));
              /*  for(int i=0; i<splited1.length; i++){
                    if(!basetypes.contains(splited1[i])){//ean den einai base type
                        thita_permissions[poss].com.add(splited1[i]);
                    }
                }*/
               poss++;
                policy_setofgroups[pos].Perms.addAll(nodename);
                pos++;
                 System.out.print("< ");
                 for(int k=0; k<nodename.size(); k++){
                      System.out.print(nodename.get(k));
                      if(nodename.size()-1!=k){
                          System.out.print(". ");
                      }
                 }
                System.out.print(", ");
                for(int k=0; k<nodepermission.size(); k++){
                      System.out.print(nodepermission.get(k));
                      if(nodepermission.size()-1!=k){
                          System.out.print(". ");
                      }
                 }
              System.out.println(">");
                 nodename.clear();
                 nodepermission.clear();
```

```
                        s.pop();

                    }
                }
            }

        }
        // Do PostOrder on following siblings

diasxisi(node.getNextSibling(),doc,s,nodename,nodepermission,set,policy_setofgroups,thita_
permissions);
    }

    public static void PostOrderTraversal(Node node,Document doc,sistima[]
final_permissions,sistima[] parP_permissions) throws XPathExpressionException{
        boolean
found=false,found_access=false,found_write=false,found_disclose=false,found_basetype=fal
se;
        NodeList parent_children=null;
        String name=null, obj=null,var="";
        int cont;
        if (node == null || node.getNodeName() == null)
          return;

        // Do PostOrder on all children
        PostOrderTraversal(node.getFirstChild(),doc,final_permissions,parP_permissions);
        // Now that all children were traversed, process the current node:
        if(node.getNodeName()!="#text"){
            XPath xpath = XPathFactory.newInstance().newXPath();
            XPathExpression                    expression                    =
xpath.compile("//"+node.getNodeName()+"/text()[1]");
            NodeList nodes = (NodeList) expression.evaluate(doc.getDocumentElement(),
XPathConstants.NODESET);
            //System.out.println("name=="+node.getNodeName()+"
attribute=="+nodes.item(0).getNodeValue());
            //add the g attributes to hashmap
```

```
        if(node.getNodeName().contains("g_attributes")){
                XPath xpathy = XPathFactory.newInstance().newXPath();
            XPathExpression expressiony =
xpathy.compile("//"+node.getNodeName()+"/text()[1]");
            NodeList nodesy = (NodeList) expressiony.evaluate(doc.getDocumentElement(),
XPathConstants.NODESET);
            String split[]=nodesy.item(0).getNodeValue().split("[:,]");
            for(int x=0; x<split.length; x=x+2){
                hash.put(split[x],split[x+1]);//add the chanel and the type to the hashmap
            }

        }

        if(node.getNodeName().contains("base_type")){
                XPath xpathy = XPathFactory.newInstance().newXPath();
            XPathExpression expressiony =
xpathy.compile("//"+node.getNodeName()+"/text()[1]");
            NodeList nodesy = (NodeList) expressiony.evaluate(doc.getDocumentElement(),
XPathConstants.NODESET);
            String split[]=nodesy.item(0).getNodeValue().split("[\\s]");
            for(int y=0; y<split.length; y++){
                basetypes.add(split[y]);
            }
        }

        /*action to get the type of channels and construct Î“*/
        if(node.getNodeName().contains("name")){
            name=nodes.item(0).getNodeValue();
            /*get the children of parent nod to find the tag type so we can know the
             * type of name*/
            parent_children=node.getParentNode().getChildNodes();
            for (int j = 0; j < parent_children.getLength(); j++){
               if(parent_children.item(j).getNodeName().contains("type")){
                  XPath xpathy = XPathFactory.newInstance().newXPath();
                  XPathExpression expressiony =
xpathy.compile("//"+parent_children.item(j).getNodeName()+"/text()[1]");
```

```
                    NodeList nodesy = (NodeList) expressiony.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                    hash.put(name,nodesy.item(0).getNodeValue());//add the chanel and the type to the hashmap
                }
            }
        }
        /*if the nodename is obj then we want to get the type of obj to construct Î“ */
        if(node.getNodeName().contains("obj")){
            obj=nodes.item(0).getNodeValue();
            /*get the children of parent node of <obj> to find the tag <type> so we can know the
             * type of <obj>*/
            parent_children=node.getParentNode().getChildNodes();
            for (int j = 0; j < parent_children.getLength(); j++){
                if(parent_children.item(j).getNodeName().contains("type")){
                    XPath xpathx = XPathFactory.newInstance().newXPath();
                    XPathExpression expressionx = xpathx.compile("//"+parent_children.item(j).getNodeName()+"/text()[1]");
                    NodeList nodesx = (NodeList) expressionx.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                    //System.out.println("OBJ=="+obj+" TYPE==="+nodesx.item(0).getNodeValue());
                    hash.put(obj,nodesx.item(0).getNodeValue());
                }
            }

        }

        /*if the tag name is <in> we get the type and we check if is base type we
         * add the permission READ,otherwise we add the permission ACCESS*/
        if(node.getNodeName().contains("in")){
            NodeList children=node.getChildNodes();
            /*gets the children of tag <in>*/
            for (int i = 0; i < children.getLength(); i++){
                /*if one of <in> children it's <type> then we get its text value*/
```

```
            if(children.item(i).getNodeName().contains("type")){

                XPath xpath1 = XPathFactory.newInstance().newXPath();
                XPathExpression expression1 = xpath1.compile("//"+children.item(i).getNodeName()+"/text()[1]");
                NodeList nodes1 = (NodeList) expression1.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);

                /*if the type does not contain "[" mean that is a base type*/
                if(nodes1.item(0).getNodeValue().contains("[")==false){

                    cont=0;

                   //diatrexo ti domi gia na do an iparxei sto array to base type
                   for(int k=0; k<final_permissions.length; k++){

                     //ean to base type periexetai sti domi
                     if(final_permissions[k].base_type!=null){

if(final_permissions[k].base_type.equals(nodes1.item(0).getNodeValue())){
                            //ean iparxei to base type sti domi tote pao sti thesi t pinaka p einai to base type
                            //kai apla elegxo ean sto arraylist tou iparxei to read
                            //ean to read den iparxei tote to prostheto kai vgeno apo to loop
                            if(!final_permissions[k].Perms.contains("read")){
                              final_permissions[k].Perms.add("read");
                              found=true;
                              cont=k;
                              break;
                            }
                        }
                     }

                }

                //ean to base type den iparxei tote to prostheto kai prostheto kai sto arraylist
```

```
                //to read
                if(found==false && !final_permissions[cont].Perms.contains("read")){
                    final_permissions[cont].base_type=nodes1.item(0).getNodeValue();
                    final_permissions[cont].Perms.add("read");
                }
                //for(int k=0; k<final_permissions.length; k++){
                 //                   System.out.println("------"+final_permissions[k].base_type+"
perms=="+final_permissions[k].Perms);
                 //}

            }
            //the type is not a base type but is something like G[ti]
             else{
                    cont=0;
                //put the nodevalue in a string
                String str=nodes1.item(0).getNodeValue();
                //replace special characters like [ and ]
                str = str.replace("[", " ").replace("]", " ");
                String splited[]=str.split("[\\s]+");//split the string with delimiter //s
                //if splited length is even number mean that we have something like G[t]
                if(splited.length % 2==0){
                     //diatrexo ti domi gia na do an iparxei sto array to base type
                   for(int k=0; k<final_permissions.length; k++){
                     //ean to base type iparxei idi sti domi
                     if(final_permissions[k].base_type!=null){
                            if(final_permissions[k].base_type.equals(splited[splited.length-
1])){
                                //ean iparxei to base type sti domi tote pao sti thesi t
pinaka p einai to base type
                            //kai apla elegxo ean sto arraylist tou iparxei to access
                            //ean to access den iparxei tote to prostheto kai vgeno apo to loop
                            if(!final_permissions[k].Perms.contains("access")){
                              final_permissions[k].Perms.add("access");
                              found_access=true;
                              cont=k;
                              break;
```

```
                                }
                                    }
                                }
                               }//end for
                              //ean to base type den iparxei tote to prostheto kai prostheto kai sto arraylist
                               //to access
                               if(found_access==false                                    && !final_permissions[cont].Perms.contains("access")){
                                    final_permissions[cont].base_type=splited[splited.length-1];//add base type
                                    final_permissions[cont].Perms.add("access");//add access
                               }
                               // for(int k=0; k<final_permissions.length; k++){
                                //          System.out.println("!!!!!!!!"+final_permissions[k].base_type+" perms=="+final_permissions[k].Perms);
                                //}
                            }
                          }//end else
                        }

                      }
                    }//end if ,<in>


                    /*if the tag name is <out> we get the obj and we check if is base type we
                     * add the permission WRITE,otherwise we add the permission DISCLOSE G 1*/
                    if(node.getNodeName().contains("out")){

                          String splitt2[];
                          NodeList children=node.getChildNodes();
                       //gets the children of tag <out>
                       for (int i = 0; i < children.getLength(); i++){
                         //if one of <out> children it's <obj> then we get its text value
                         if(children.item(i).getNodeName().contains("obj")){
                            XPath xpath1 = XPathFactory.newInstance().newXPath();
```

```
XPathExpression expression1 = xpath1.compile("//"+children.item(i).getNodeName()+"/text()[1]");
NodeList nodes1 = (NodeList) expression1.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
//elegxo to type tou obj apo to hashmap opou kratw ta types
// Get values based on key
var= hash.get(nodes1.item(0).getNodeValue());

//var is null means that we have base type
if(var==null){
    cont=0;
    //diatrexo ti domi gia na do an iparxei sto array to base type
  for(int k=0; k<final_permissions.length; k++){

   if(final_permissions[k].base_type!=null){
    //ean to base type iparxei idi sti domi

  if(final_permissions[k].base_type.equals(nodes1.item(0).getNodeValue())){
        //ean iparxei to base type sti domi tote pao sti thesi t pinaka p einai to base type
      //kai apla elegxo ean sto arraylist tou iparxei to write
      //ean to write den iparxei tote to prostheto kai vgeno apo to loop
      if(!final_permissions[k].Perms.contains("write")){
        final_permissions[k].Perms.add("write");
        found_write=true;
        cont=k;
        break;
      }
    }
   }
  }//end for
 //ean to base type den iparxei tote to prostheto kai prostheto kai sto arraylist
  //to write
  if(found_write==false && !final_permissions[cont].Perms.contains("write")){
```

```
final_permissions[cont].base_type=nodes1.item(0).getNodeValue();//add base type
                        final_permissions[cont].Perms.add("write");//add write
                    }
                    //for(int k=0; k<final_permissions.length; k++){
                      //    System.out.println("@@@@@@"+final_permissions[k].base_type+"
perms=="+final_permissions[k].Perms);
                     //}

                }
                //we dont have a base type but something like G[t]
                else{
                        cont=0;
                   //put the nodevalue in a string
                   String str=var;
                   //replace special characters like [ and ]
                   str = str.replace("[", " ").replace("]", " ");
                   String splited[]=str.split("[\\s]+");//split the string with delimiter //s
                   //if splited length is even number mean that we have something like G[t]
                   if(splited.length % 2==0){
                        //diatrexo ti domi gia na do an iparxei sto array to base type
                      for(int k=0; k<final_permissions.length; k++){
                          if(final_permissions[k].base_type!=null                    ||
final_permissions[k].base_type==""){
                                    //ean to base type iparxei idi sti domi
                               if(final_permissions[k].base_type.equals(splited[splited.length-
1])){
                                    found_basetype=true;
                                     //pao kai vrisko ti thesi t arraylist p exo to permission
disclose G arithmos
                                    for (int r = 0; r < final_permissions[k].Perms.size();
r++) {

splitt2=final_permissions[k].Perms.get(r).split("[\\s]+");
                                    //ean t prwto stoixio t split einai disclose kai to deutero
einai to G TOU G[t]
```

```
                                        //diladi iparxei to permission disclose

                                        if(splitt2[0].contains("disclose")                    &&
splitt2[1].compareTo(splited[splited.length-2])==0){
                                                found_disclose=true;
                                                if(splitt2[2].compareTo("*")!=0){
                                                        int
number=Integer.parseInt(splitt2[2])+1;

        final_permissions[k].Perms.set(r,splitt2[0]+" "+splitt2[1]+" "+number);
                                                 break;
                                                }

                                        }

                                        }
                                        //diladi iparxei to base type alla den iparxei to disclose
sta permissions
                                        if(found_disclose==false  ){
                                                final_permissions[k].Perms.add("disclose
"+splited[splited.length-2]+" 1");
                                        }

                                    }
                                }
                        }//end for
                        //to base type den iparxei epomenws oute kai to disclose iparxei
                        if(found_basetype==false ){
                              for(int c=0; c<final_permissions.length; c++){
                                   if(final_permissions[c].base_type==null){
                                        final_permissions[c].base_type=splited[splited.length-
1];//add base type
                                   final_permissions[c].Perms.add("disclose
"+splited[splited.length-2]+" 1");//add disclose g1
                                     break;
                                   }
```

```
                        }
                    }
                    //for(int k=0; k<final_permissions.length; k++){
                      //      System.out.println("*******"+final_permissions[k].base_type+"
perms=="+final_permissions[k].Perms);
                     //}
                  }
                }
              }
            }

        /*if the tag name is <repl> we just check if we have disclose G in the values
         * of the hashmap and we put the * instead of the number */
        if(node.getNodeName().contains("repl")){
                //diasxizo gia kathe basetype p exo to arraylist t gia na do an exo mesa
disclose
                for(int k=0; k<final_permissions.length; k++){
                        for(int e=0; e<final_permissions[k].Perms.size(); e++){
                                String
splitt[]=final_permissions[k].Perms.get(e).split("[\\s]+");
                        if(splitt[0].contains("disclose")){
                                final_permissions[k].Perms.set(e,  splitt[0]+"  "+splitt[1]+"
"+"*");
                        }
                        }
            }
         }

        /*if the tag name is <proccess>  */
        if(node.getNodeName().contains("proccess")){
                boolean answer= check_if_isempty(final_permissions);
```

```
//elegxo ean o pateras tou proccess einai to parP kai episis i domi 1 na min einai adeia
if(node.getParentNode().getNodeName().contains("parP") && answer==true ){
boolean exist=false;
for(int k=0; k<final_permissions.length; k++){
for(int t=0; t<parP_permissions.length; t++){
//ean to base type iparxei idi sti domi 2 meta tha kitakso g ta permissions
if(final_permissions[k].base_type!=null ){
if(parP_permissions[k].base_type!=null){

if(final_permissions[k].base_type.compareTo(parP_permissions[t].base_type)==0 ){
//ean to permissions tis domis 1 den iparxei sto permissions tis domis 2 tote ta prostheto
for(int s=0; s<final_permissions[k].Perms.size(); s++){
//ean i domi 2 den periexei to permission tis domis 1 tote to prostheto

if(!parP_permissions[t].Perms.contains(final_permissions[k].Perms.get(s))){

parP_permissions[t].Perms.add(final_permissions[k].Perms.get(s));

}
}
exist=true;
break;//afou vrika to base type p iparxei sti domi 2 vazo ta analoga permissions
//k vgeno apo to loop
}
}
//den iparxei to base type sti domi 2 ara prosthero kai t base type kai ta permissions
if(exist==false && final_permissions[k].base_type!=null){
for(int g=0; g<parP_permissions.length; g++){
if(parP_permissions[g].base_type==null){
```

```
        parP_permissions[g].base_type=final_permissions[k].base_type;
                                        parP_permissions[g].Perms     =     (ArrayList<String>)
final_permissions[k].Perms.clone();
                                           break;
                                        }
                                  }
                               break;
                            }
                            }



                         }


                                  }
                                  //adiazo tin domi 1 otan tin kanw copy stin domi 2
                                   make_empty(final_permissions);//adeiazo ti domi
                              // for(int w=0; w<final_permissions.length; w++){
                          //                                                System.out.println("########
base_type=="+final_permissions[w].base_type+"    perms=="+final_permissions[w].Perms);
                           //}
                                  //for(int r=0; r<parP_permissions.length; r++){
                                        //
System.out.println("$$$$$$$"+parP_permissions[r].base_type+"
perms=="+parP_permissions[r].Perms);
                                  //}


                        }
                        //ean o pateras tou <proccess> einai to resGP kai i domi 1 den einai adeia
                        if(node.getParentNode().getNodeName().contains("resGP")                    &&
answer==true ){
                              //pare ta paidia tou resgp
                              NodeList resgp_children= node.getParentNode().getChildNodes();
                              for(int r=0; r<resgp_children.getLength(); r++){
                                    //vres to paidi t resgp p onomazetai group
```

```
                                if(resgp_children.item(r).getNodeName().contains("group")){
                                        XPath xpatha =
XPathFactory.newInstance().newXPath();
                    XPathExpression expressiona =
xpatha.compile("//"+resgp_children.item(r).getNodeName()+"/text()[1]");
                    NodeList nodesa = (NodeList)
expressiona.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                     String value="";
                    for(int h=0; h<final_permissions.length; h++){

//System.out.println("~~~~~~~~"+final_permissions[h].base_type+"
perms=="+final_permissions[h].Perms);
                                if(final_permissions[h].base_type!=null){

value=value+final_permissions[h].base_type+final_permissions[h].Perms;
                                }
                    }

resgp_children.item(r).setTextContent(nodesa.item(0).getNodeValue()+":"+value);
                     XPath xpathx = XPathFactory.newInstance().newXPath();
                XPathExpression expressionx =
xpathx.compile("//"+resgp_children.item(r).getNodeName()+"/text()[1]");
                NodeList nodesx = (NodeList)
expressionx.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
                                //adeiazo ti domi 1
                                make_empty(final_permissions);
                    //for(int w=0; w<final_permissions.length; w++){
                //                                System.out.println("########
base_type=="+final_permissions[w].base_type+"    perms=="+final_permissions[w].Perms);
                     //}
                                }
                     }
                }

        }
```

```
        //ean to nodename  einai <parP> kai i domi 2 na min einai adeia
        if(node.getNodeName().contains("parP")          &&          check_if_isempty(
parP_permissions)==true){
                String value="";
                for(int h=0; h<parP_permissions.length; h++){

//System.out.println("~~~~~~~~"+parP_permissions[h].base_type+"
perms=="+parP_permissions[h].Perms);
                        if(parP_permissions[h].base_type!=null){

value=value+parP_permissions[h].base_type+parP_permissions[h].Perms+".";
                        }
                }
                node.setTextContent(value);
                XPath xpathx = XPathFactory.newInstance().newXPath();
            XPathExpression                         expressionx                         =
xpathx.compile("//"+node.getNodeName()+"/text()[1]");
            NodeList nodesx = (NodeList) expressionx.evaluate(doc.getDocumentElement(),
XPathConstants.NODESET);
                        //adeiazo ti domi 2
                        make_empty(parP_permissions);
                // for(int w=0; w<parP_permissions.length; w++){
              //                                                  System.out.println("########
base_type=="+parP_permissions[w].base_type+"    perms=="+parP_permissions[w].Perms);
                // }
            }

        //en to nodename einai<resGP>
        if(node.getNodeName().contains("resGP")){
                //pare ta paidia tou resGP
                NodeList children=node.getChildNodes();
                for(int x=0; x<children.getLength(); x++){

                        if(children.item(x).getNodeName().contains("parP")){
                                XPath xpathx = XPathFactory.newInstance().newXPath();
```

```
XPathExpression expressionx = xpathx.compile("//"+children.item(x).getNodeName()+"/text()[1]");
NodeList nodesx = (NodeList) expressionx.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
for(int j=0; j<children.getLength(); j++){
if(children.item(j).getNodeName().contains("group")){
XPath xpatha = XPathFactory.newInstance().newXPath();
XPathExpression expressiona = xpatha.compile("//"+children.item(j).getNodeName()+"/text()[1]");
NodeList nodesa = (NodeList) expressiona.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);

children.item(j).setTextContent(nodesa.item(0).getNodeValue()+":"+nodesx.item(0).getNodeValue());
XPath xpathe = XPathFactory.newInstance().newXPath();
XPathExpression expressione = xpathe.compile("//"+children.item(j).getNodeName()+"/text()[1]");
NodeList nodese = (NodeList) expressione.evaluate(doc.getDocumentElement(), XPathConstants.NODESET);
}
}
}
}

}
// Do PostOrder on following siblings
PostOrderTraversal(node.getNextSibling(),doc,final_permissions,parP_permissions);
}

/*public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException, XPathExpressionException{
system_parse();
//policy_parse();}*/}
```