ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΕΤΑΚΙΝΟΥΜΕΝΟΙ ΠΡΑΚΤΟΡΕΣ ΚΑΙ ΟΙ ΑΠΕΙΛΕΣ ΤΗΣ ΑΣΦΑΛΕΙΑΣ ΤΟΥΣ

Διπλωματική Εργασία

Δημήτρης Γεννηματάς

Επιβλέπων: Ασπασία Δασκαλοπούλου

Βόλος 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

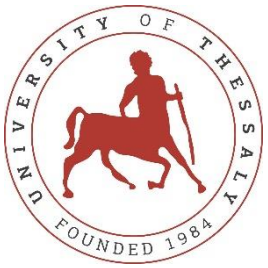
ΜΕΤΑΚΙΝΟΥΜΕΝΟΙ ΠΡΑΚΤΟΡΕΣ ΚΑΙ ΟΙ ΑΠΕΙΛΕΣ ΤΗΣ ΑΣΦΑΛΕΙΑΣ ΤΟΥΣ

Διπλωματική Εργασία

Δημήτρης Γεννηματάς

Επιβλέπων: Ασπασία Δασκαλοπούλου

Βόλος 2020



UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

MOBILE AGENTS AND THEIR SECURITY THREATS

Diploma Thesis

Dimitris Gennimatas

Supervisor: Aspasia Daskalopoulou

Volos 2020

Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο/Η Δηλών/ούσα

Δημήτρης Γεννηματάς
17/7/2020

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω καταρχάς την οικογένεια μου η οποία μου στάθηκε όλα αυτά τα χρόνια στην προσπάθεια μου να αποκτήσω το πτυχίο που ήθελα. Στη συνέχεια θα ήθελα να ευχαριστήσω το διδακτικό προσωπικό του τμήματος και ιδιαίτερα την Καθηγήτρια Ασπασία Δασκαλοπούλου η οποία με εμπιστεύτηκε για την εκπόνηση αυτής της διπλωματικής και μου παρείχε την αμέριστη υποστήριξή της καθ' όλη τη διάρκειά της.

ΠΡΟΛΟΓΟΣ

Η παρούσα εργασία εκπονήθηκε ως το τελευταίο βήμα για την απόκτηση διπλώματος και την ολοκλήρωση των σπουδών μου στο Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας στην πόλη του Βόλου υπό την επίβλεψη της Επίκουρου Καθηγήτριας Ασπασίας Δασκαλοπούλου.

ΠΕΡΙΛΗΨΗ

Η εφαρμογή και η υποστήριξη του μετακινούμενου κώδικα στα σύγχρονα κατανεμημένα συστήματα είναι ζωτικής σημασίας λειτουργικότητα. Τεχνολογίες όπως το μοντέλο πελάτη-εξυπηρετητή (client-server), κώδικα κατόπιν αιτήματος (code on demand) και απομακρυσμένου υπολογισμού (remote evaluation**) [1], [2]** αποτελούν αντικείμενο μελέτης εδώ και αρκετά χρόνια, γνωρίζοντας μια συνεχή εξέλιξη. Μια εναλλακτική εκδοχή είναι οι μετακινούμενοι πράκτορες (mobile agents), οντότητες κώδικα, οι οποίες μετακινούνται αυτόνομα μέσα σε ένα κατανεμημένο σύστημα, επιτελώντας με έξυπνο τρόπο τις λειτουργίες που τους ανατίθενται. Ο καίριας σημασίας ρόλος τους, τους καθιστά αντικείμενο επίθεσης από κακόβουλους χρήστες, οι οποίοι με συγκεκριμένους τρόπους επιθυμούν να εκμεταλλευτούν προς όφελός τους την πολλαπλή λειτουργικότητα τους. Η παρούσα διπλωματική στοχεύει να εξετάσει διεξοδικά τους κινδύνους των συστημάτων μετακινούμενων πρακτόρων, καθώς και τρόπους για την πρόληψη και την καταστολή τους. Αρχικά θα γίνει μια ανάλυση του τρόπου λειτουργίας των μετακινούμενων πρακτόρων, πώς υλοποιούνται οι διάφορες λειτουργικότητες τους σε JADE καθώς και η υλοποίηση ενός μετακινούμενου πράκτορα. Στη συνέχεια ακολουθεί μια θεωρητική ανάλυση των κινδύνων και των μεθόδων αντιμετώπισης και πρόληψης και τέλος εξετάζονται μέθοδοι για τη βελτιστοποίηση της ασφάλειας των υλοποιήσεων σε JADE με χρήση plugins.

Λέξεις Κλειδιά

Μετακινούμενοι Πράκτορες, Ασφάλεια, JADE, Κατανεμημένο Σύστημα

ABSTRACT

The implementation and support of mobile code functionality in modern distributed systems is vital. Technological concepts such as client-server model, code on demand and remote evaluation **[1]**, **[2]** have been the subject of studies for many years, resulting in an ongoing evolution. An alternative approach to the subject refers to the concept of mobile agents, software entities that have the ability to move autonomously inside a distributed system and accomplish intelligently the various tasks that they are being assigned with. Their greatly important role makes them a target for malicious users, who want to exploit the multiple functionalities they offer. The current thesis aims to examine in depth the dangers mobile agent systems undergo, as well as various methods to prevent and protect them from malicious attacks. Firstly, an overview analysis of the mobile agent architecture and functionality will be discussed, accompanied by how various functionalities are implemented in JADE and a mobile agent implementation. Furthermore, in the next sections a theoretical approach of the dangers and the countermeasures will be presented, followed by an analysis of ways that could enhance the security of the JADE framework by using plugins.

Keywords

Mobile Agents, Security, JADE, Distributed System

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Η έννοια του μετακινούμενου κώδικα γεννήθηκε περίπου το 1960 από την IBM έτσι ώστε να καλύψει την ανάγκη για ένα κεντρικοποιημένο σύστημα επεξεργασίας κατά το οποίο, ένας ισχυρός υπολογιστής (Mainframe) εκτελούσε όλες τις απαιτητικές διεργασίες των αιτούντων. Η ονομασία αυτής της μεθόδου ήταν "Remote Job Entry" η οποία θεωρείται ο πρόδρομος του μοντέλου πελάτη-εξυπηρετητή (client-server model) **[3]**

Οι μετακινούμενοι πράκτορες προκύπτουν από τον συνδυασμό δύο διαφορετικών πεδίων της πληροφορικής. Το πρώτο είναι το πεδίο της τεχνητής νοημοσύνης, το οποίο σύμφωνα με τους Russel & Norvig **[4],** όρισε την ιδέα του πράκτορα. Το δεύτερο, είναι το πεδίο των κατανεμημένων συστημάτων, το οποίο όρισε την έννοια του μετακινούμενου κώδικα.

Η ιδέα για τους μετακινούμενους πράκτορες γεννήθηκε κυρίως για να λύσει το πρόβλημα της υπερφόρτωσης του δικτύου, καθώς η κινητικότητά τους εξαλείφει την ανάγκη ύπαρξης ενεργής σύνδεσης ανάμεσα σε πελάτη και εξυπηρετητή, όπως για παράδειγμα μέσω του πρωτοκόλλου επιπέδου μεταφοράς TCP *(Εικόνα 1.1)*. Ο πελάτης εκκινεί τον μετακινούμενο πράκτορα, ο οποίος με τη σειρά του μετακινείται από κόμβο σε κόμβο του συστήματος χωρίς να υπάρχει ενεργή σύνδεση μεταξύ τους, εκτελώντας την διεργασία που του έχει ανατεθεί. Όταν τελειώσει, επιστρέφει πίσω στον πελάτη μεταβιβάζοντάς του το αποτέλεσμα και όλα αυτά με ελάχιστη χρήση του εύρους (bandwidth) του δικτύου **[5].** Οπότε η ασύγχρονη αυτή επικοινωνία είναι το μέλλον για λόγους εξοικονόμησης πόρων του δικτύου.

Η τάση αυτή προς την συγκεκριμένη τεχνολογία έχει κεντρίσει και το ενδιαφέρον κακόβουλών χρηστών, οι οποίοι προσπαθούν να εκμεταλλευτούν τις αδυναμίες και τα κενά ασφαλείας των συστημάτων πληροφορίας.

Σε αυτή τη διπλωματική θα εστιάσουμε στους κινδύνους που διατρέχουν τα συστήματα μετακινούμενων πρακτόρων από τέτοιες κακόβουλες επιθέσεις, καθώς και σε τρόπους πρόληψης και αντιμετώπισής τους.

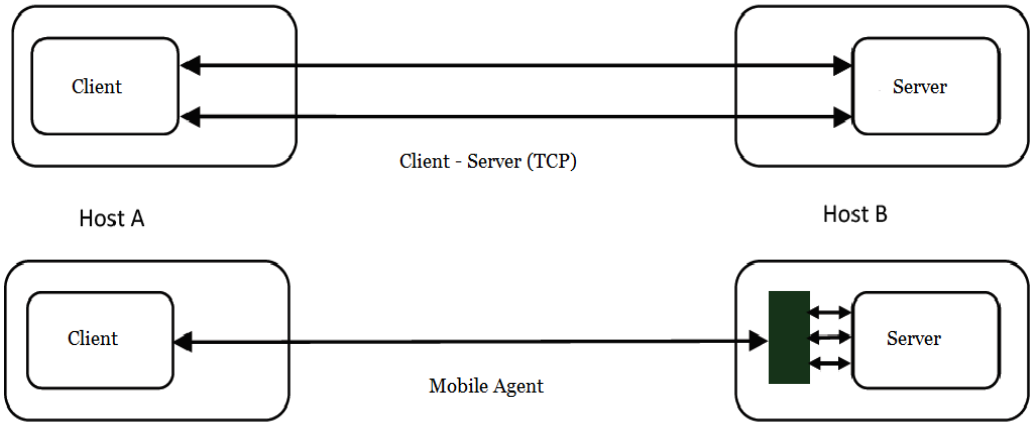
Συγκεκριμένα:

- Στο Κεφάλαιο 2 γίνεται μια εισαγωγή στον τρόπο λειτουργίας των συστημάτων μετακινούμενων πρακτόρων, καθώς επίσης και παρουσίαση των μεθόδων που προσδίδουν στους πράκτορες την ιδιότητα της φορητότητάς τους, χρησιμοποιώντας το JADE framework. Τέλος, κατασκευάζεται ένας μετακινούμενος πράκτορας συγκεκριμένου σκοπού.

- Στο Κεφάλαιο 3 γίνεται παρουσίαση των διαφόρων απειλών που ενδεχομένως μπορεί να αντιμετωπίσει ένα τέτοιο σύστημα.

- Στο Κεφάλαιο 4 μελετώνται διάφορες τεχνικές και μέθοδοι αντιμετώπισης των απειλών που αναφέρθηκαν στο κεφάλαιο 3.

- Τέλος, στο Κεφάλαιο 5 γίνεται μια προσπάθεια διεύρυνσης του πλαισίου ασφάλειας που διέπει μια πλατφόρμα μετακινούμενων πρακτόρων υλοποιημένη σε JADE, με χρήση διαφόρων security plugins.



Εικόνα 1.1: Σχηματική Αναπαράσταση Επικοινωνίας με Μοντέλο Client-Server και με Mobile Agents **[6]**

ΚΕΦΑΛΑΙΟ 2

ΜΟΝΤΕΡΝΑ ΣΥΣΤΗΜΑΤΑ ΜΕΤΑΚΙΝΟΥΜΕΝΩΝ ΠΡΑΚΤΟΡΩΝ

2.1 - Αρχές Λειτουργίας των Μετακινούμενων Πρακτόρων

2.1.1 – Ορισμός ενός Μετακινούμενου Πράκτορα.

Ως μετακινούμενος πράκτορας, ορίζεται ο πράκτορας εκείνος ο οποίος έχει όλα τα χαρακτηριστικά ενός πράκτορα (αυτονομία, δυναμική λήψη αποφάσεων, αλληλεπίδραση με άλλους πράκτορες) με επιπρόσθετο και βασικότερο χαρακτηριστικό την ικανότητα να μετακινείται από κόμβο σε κόμβο μέσα σε μια πλατφόρμα. Κατά τη μετακίνησή του, ένας μετακινούμενος πράκτορας φέρει μαζί του τους πόρους που απαιτούνται ώστε να φέρει εις πέρας την δουλειά που του έχει ανατεθεί από κάποιον πελάτη (client) της πλατφόρμας ακόμα και αν αυτός αποχωρήσει από το δίκτυο πριν την ολοκλήρωσή της.

Επίσης, μια διαφορετική προσέγγιση ορίζει τον μετακινούμενο πράκτορα ως «ένα πρόγραμμα λογισμικού το οποίο διαθέτει το χαρακτηριστικό της κινητικότητας. Δημιουργείται από κάποιον χρήστη σε ένα δίκτυο και μετακινείται στους κόμβους του για να μπορέσει να υλοποιήσει τις ενέργειες που του ζητήθηκαν εκ μέρους του χρήστη που τον δημιούργησε» **[7]**

Η ικανότητα του αυτή να μετακινείται επιτρέπει σε έναν μετακινούμενο πράκτορα να αλληλεπιδρά άμεσα με το αντικείμενο που επιθυμεί, έχοντας το πλεονέκτημα να βρίσκεται στο ίδιο φυσικό σύστημα με εκείνο, και όχι μέσω του δικτύου. **[8], [9]**

Κάθε μετακινούμενος πράκτορας χωρίζεται σε 3 υπο-μέρη **[10]**:

1. **Κώδικας (Code):** Ο πηγαίος κώδικας στον οποίο έχει υλοποιηθεί ο μετακινούμενος πράκτορας.

2. **Κατάσταση (State):** Αποθήκευση των εσωτερικών μεταβλητών και επαναφορά της λειτουργίας μετά τη μετακίνησή του σε επόμενο κόμβο από το σημείο στο οποίο είχε σταματήσει πριν τη μετακίνησή του.

3. **Χαρακτηριστικά (Attributes):** Περιλαμβάνει γενικότερες πληροφορίες σχετικά με τον μετακινούμενο πράκτορα.

    - Σε ποιον πελάτη ανήκει (Host Owner).
    - Ιστορικό μετακίνησης (Migration History).
    - Διάφορες λειτουργικότητες ασφάλειας που αναλύονται αργότερα.

2.1.2 – Αρχιτεκτονική ενός Συστήματος Μετακινούμενων Πρακτόρων.

1. ***Περιβάλλον (Environment):***

Το περιβάλλον με το οποίο αλληλεπιδρά ένας μετακινούμενος πράκτορας περιλαμβάνει τα εξής μέρη:

- **Εξυπηρετητές (Servers/Hosts):** Είναι οι κόμβοι αυτοί στους οποίους ένας μετακινούμενος πράκτορας μπορεί να μετακινηθεί για να επιτελέσει τους υπολογισμούς που επιθυμεί καθώς του προσφέρουν κατάλληλους πόρους γι' αυτό το σκοπό (Επεξεργαστική ισχύς, μνήμη).
- **Άλλοι Μετακινούμενοι Πράκτορες (Other Mobile Agents):** Κάθε μετακινούμενος πράκτορας μπορεί να αλληλεπιδράσει με άλλους, ανταλλάσοντας πληροφορίες και δεδομένα, το οποίο θα οδηγεί είτε σε αμοιβαίο όφελος, είτε ο μετακινούμενος πράκτορας που αιτείται δεδομένα από κάποιον άλλο τα χρησιμοποιεί μόνο για δικό του όφελος.
- **Διαδικτυακοί Σύνδεσμοι (Network Links):** Απαιτούνται για να επιτρέπουν τη μεταφορά των μετακινούμενων πρακτόρων από κόμβο σε κόμβο. Οι σύνδεσμοι αυτοί δεν είναι απαραίτητο να είναι εντελώς αξιόπιστοι, καθώς ένα βασικό πλεονέκτημα των μετακινούμενων πρακτόρων έναντι άλλων υλοποιήσεων (π.χ client- server**)** είναι ότι μπορούν να λειτουργήσουν χωρίς να υπάρχει κάποια απευθείας ενεργή σύνδεση, γεγονός που τους προσδίδει την ιδιότητα της ασύγχρονης επικοινωνίας (asynchronous

communication) και κάνει το σύστημα πραγματικά αποκεντρωμένο (truly decentralized).

2. Μνήμη (Memory):

Ένα πολύ βασικό χαρακτηριστικό ενός μετακινούμενου πράκτορα είναι η μονάδα αποθήκευσης (Μνήμη). Η χρησιμότητά της χωρίζεται σε δυο μέρη:

- **Αποθήκευση:** Η μνήμη είναι απαραίτητη για την αποθήκευση δεδομένων, διάφορων χαρακτηριστικών καθώς και της κατάστασης λειτουργίας που επιτελεί ο μετακινούμενος πράκτορας έτσι ώστε να μπορεί να συνεχίζει από εκεί που τη σταμάτησε.
- **Επιστροφή αποτελέσματος:** Εφόσον ο μετακινούμενος πράκτορας ολοκληρώσει την επεξεργασία της αίτησης του πελάτη που του την ανέθεσε, θα πρέπει να επιστρέψει το αποτέλεσμα σε αυτόν. Αυτό καθίσταται δυνατό μέσω της μνήμης του, καθώς αποθηκεύει το αποτέλεσμα μέχρι να επιστρέψει στον αιτούντα κόμβο/χρήστη για να του το παραδώσει.

3. Συμπεριφορά και Αυτονομία (Behavior and Autonomy):

- **Κινητικότητα (Mobility):** Επιτρέπει στον μετακινούμενο πράκτορα να κινείται αυτόνομα μέσα στο δίκτυο από κόμβο σε κόμβο, διενεργώντας ανταλλαγή πληροφοριών, εκτέλεση εργασιών και επικοινωνώντας με άλλους πράκτορες ή κόμβους του συστήματος, εφόσον είναι αναγκαίο, για την περαίωση της εργασίας του.
- **Ανεξαρτησία (Independency):** Προσδίδει την απαραίτητη αυτονομία στον μετακινούμενο πράκτορα, δίνοντας του το δικαίωμα να ενεργεί με βάση τις δικές του αποφάσεις οι οποίες πολλές φορές πηγάζουν από τη δυναμική διαδικασία μάθησης (adaptive learning).
- **Επικοινωνία (Communication):** Η δυνατότητα που έχει ο μετακινούμενος πράκτορας να αποκτά πρόσβαση σε χρήσιμες γι' αυτόν πληροφορίες, μέσω ανταλλαγής μηνυμάτων.
- **Μάθηση (Learning):** Η ιδιότητα που κάνει έναν πράκτορα «ευφυή» είναι η ικανότητά του να λαμβάνει τις σωστές αποφάσεις για τη γρηγορότερη και αποτελεσματικότερη εκτέλεση της λειτουργίας που του έχει ανατεθεί. Σε αντίθεση με

το μοντέλο των στατικών προδιαγεγραμμένων ενεργειών, οι «ευφυείς» μετακινούμενοι πράκτορες δημιουργούν μια συνεχώς μεταβαλλόμενη βάση γνώσης η οποία βασίζεται σε προηγούμενες εμπειρίες και στην αποτελεσματικότητά τους. Η ιδιότητα τους αυτή τους προσδίδει την ικανότητα τροποποίησης της διαδρομής που θα ακολουθήσουν (dynamic itinerary mobile agents) σε αντίθεση με την άλλη κατηγορία μετακινούμενων πρακτόρων, οι οποίοι δεν μπορούν να λάβουν δυναμικά αποφάσεις για τη διαδρομή την οποία θα ακολουθήσουν μέσα στην πλατφόρμα και η πορεία τους είναι προκαθορισμένη από τον δημιουργό τους (static itinerary agents).

- **Επιμονή (Persistence):** Προσδίδει στους μετακινούμενους πράκτορες ένα από τα μεγαλύτερα πλεονεκτήματά τους σε σχέση με άλλες τεχνολογίες απομακρυσμένων υπολογισμών καθώς δεν απαιτείται συνεχής ενεργή σύνδεση μεταξύ του πελάτη και του δικτύου για την ολοκλήρωση της «συναλλαγής». Ο πράκτορας στέλνεται από τον πελάτη στο δίκτυο, διακόπτοντας εν συνεχεία την μεταξύ τους σύνδεση για αποφυγή άσκοπης υπερφόρτωσης του δικτύου. Όταν ο μετακινούμενος πράκτορας έχει ολοκληρώσει την εργασία που του είχε ζητηθεί, επιστρέφει στον κόμβο του πελάτη ανοίγοντας ξανά μια σύνδεση με τον πελάτη, επιστρέφοντας μαζί με το αποτέλεσμα.

2.1.3 – Γεγονότα & Κύκλος Ζωής ενός Μετακινούμενου Πράκτορα.

- **Δημιουργία (Creation):** Ένας καινούριος μετακινούμενος πράκτορας δημιουργείται και αρχικοποιείται η κατάστασή του.

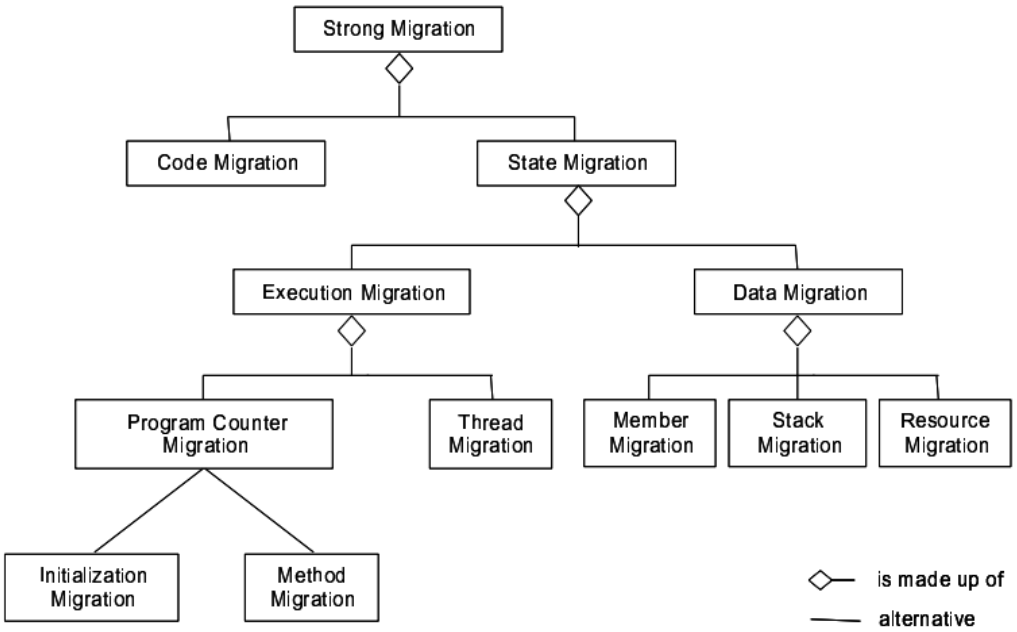
- **Μετακίνηση (Migration):** Η διαδικασία κατά την οποία ένας μετακινούμενος πράκτορας μεταφέρεται σε έναν άλλο κόμβο. Οι λόγοι που μπορεί να οδηγήσουν έναν μετακινούμενο πράκτορα σε αυτή την κίνηση ποικίλουν. Συνηθέστεροι λόγοι είναι οι εξής:
  - **i.** Ο τρέχων κόμβος εξυπηρετεί πολλές αιτήσεις και τοποθετεί τον μετακινούμενο πράκτορα σε μια ουρά αναμονής, οπότε η βέλτιστη λύση πολλές φορές είναι να μεταφερθεί σε κάποιον άλλο κόμβο της πλατφόρμας που μπορεί να τον εξυπηρετήσει αμεσότερα.
  - **ii.** Πολλές φορές, η εργασία που καλείται να επιτελέσει ένας μετακινούμενος πράκτορας, προσφέρεται μόνο από συγκεκριμένους κόμβους του συστήματος,

γεγονός που απαιτεί τη μετακίνησή του σε αυτό τον κόμβο για την ολοκλήρωσή της.

Η διαδικασία της μετακίνησης μπορεί να χωριστεί σε δύο τύπους:

a. ***Ισχυρή Μετακίνηση (Strong Migration)*:** Κατά την ισχυρή μετακίνηση, ο μετακινούμενος πράκτορας φέρει μαζί του και τον κώδικά του (code) και την κατάσταση εκτέλεσής του (state) *(Εικόνα 2.1)*.

b. ***Αδύναμη Μετακίνηση (Weak Migration):*** Ως αδύναμη μετακίνηση ορίζεται οποιαδήποτε μετακίνηση δεν είναι ισχυρή. Συνήθως ο όρος χρησιμοποιείται για μετακινήσεις στις οποίες ο μετακινούμενος πράκτορας φέρει μαζί του μόνο τον κώδικά του.



Εικόνα 2.1: Διάγραμμα μιας μετακίνησης τύπου Strong Migration

- **Κλωνοποίηση (Cloning):** Δημιουργείται ένα αντίγραφο ενός μετακινούμενου πράκτορα το οποίο κατέχει την τρέχουσα κατάσταση του πρωτότυπου.

- **Ενεργοποίηση/Απενεργοποίηση (Activation/Deactivation):**

  i. **Ενεργοποίηση:** Όταν ένας μετακινούμενος πράκτορας επανέρχεται στη ζωή μετά από μια παύση λειτουργίας του, επαναφέροντας και την κατάσταση που είχε πριν τη διακοπή της λειτουργίας του από την μόνιμη μνήμη.

  ii. **Απενεργοποίηση:** Είναι η αντίστροφη διαδικασία της ενεργοποίησης. Εδώ διακόπτεται η λειτουργία ενός μετακινούμενου πράκτορα και αποθηκεύεται σε μόνιμη μνήμη η κατάστασή του.

- **Ανάκληση (Retraction):** Όταν ο μετακινούμενος πράκτορας καλείται να επιστρέψει στον κόμβο στον οποίο ανήκει, μαζί με την κατάστασή του και τυχόν αποτελέσματα που έχει υπολογίσει μέχρι τότε.

- **Επικοινωνία (Communication):** Ο μετακινούμενος πράκτορας ειδοποιείται ότι κάποιος άλλος πράκτορας θέλει να επικοινωνήσει μαζί του, έτσι ώστε, εφόσον το επιθυμεί, να ανοίξει ένα δίαυλο επικοινωνίας.

- **Εξολόθρευση (Termination):** Όταν ο μετακινούμενος πράκτορας ολοκληρώσει τη διαδικασία που του έχει ανατεθεί τερματίζει τη λειτουργία του αφού πρώτα επιστρέψει τα αποτελέσματα στον χρήστη που του ανέθεσε τη δουλειά.

2.2 – Υλοποίηση Πλατφόρμας Μετακινούμενων Πρακτόρων σε JADE

Το JADE είναι ένα Framework της Java το οποίο παρέχει έτοιμες κλάσεις και μεθόδους, παρέχοντας με αυτό τον τρόπο έναν απλοποιημένο και ολοκληρωμένο τρόπο δημιουργίας και διαχείρισης συστημάτων πρακτόρων. Στη συνέχεια θα δούμε πώς μπορούν να υλοποιηθούν προγραμματιστικά οι βασικές λειτουργίες των μετακινούμενων πρακτόρων σε JADE. **[11], [12]**

2.2.1 – Βασικές Έννοιες:

- Η πλατφόρμα των πρακτόρων υλοποιημένη σε JADE μπορεί να κατανεμηθεί σε ανεξάρτητα φυσικά συστήματα (τα οποία μπορούν να τρέχουν και διαφορετικά λειτουργικά).

- Κάθε τρέχον στιγμιότυπο του runtime environment του JADE ονομάζεται **Container** καθώς μπορεί να περιέχει περισσότερους από έναν πράκτορες

- Το σύνολο των ενεργών containers ονομάζεται **Platform (Πλατφόρμα)**.

- Πάντα σε μια πλατφόρμα θα πρέπει να υπάρχει ένας ειδικός τύπος container ονόματι **Main-Container**, το οποίο λαμβάνει αυτή την ιδιότητα όντας συνήθως το πρώτο container της πλατφόρμας, και αναλαμβάνει ρόλο «συντονιστή» μεταξύ των containers και των πρακτόρων που ζουν στην πλατφόρμα. Στο main container, κατά τη δημιουργία του, προσάπτονται αυτόματα σε αυτό δύο πράκτορες ειδικού σκοπού. Ο **AMS** (Agent Management System), ο **DF** (Directory Facilitator) και ο **RMA**. Σκοπός του AMS είναι η διαχείριση βασικών λειτουργιών της πλατφόρμας όπως η ονοματοδότηση των πρακτόρων εστιάζοντας στην αποφυγή διπλότυπων και στη δημιουργία/καταστροφή πρακτόρων από διάφορα containers της πλατφόρμας. Ο DF χρησιμοποιείται ως ευρετήριο της πλατφόρμας κρατώντας yellow pages τα οποία μπορούν να προσφέρουν πληροφορίες σε όποιον πράκτορα τις ζητήσει όσον αφορά στις υπηρεσίες/πληροφορίες που μπορεί να του προσφέρει οποιοσδήποτε άλλος πράκτορας της πλατφόρμας. Τέλος, ο RMA αναλαμβάνει τη γραφική απεικόνιση της κατάστασης του container στο οποίο ανήκει.

2.2.2 – Ανάπτυξη λειτουργιών σε JADE: **[11]**

1. Δημιουργία Πράκτορα:

```
import jade.core.*;

public class mobileAgent extends Agent {

    private static final long serialVersionUID = 1L;

        //overrides constructor
        protected void setup(){
                System.out.println("Agent Created!ID:" + this.getAID().getName());
            }
}
```

Εικόνα 2.2: Κώδικας δημιουργίας πράκτορα σε JADE

Κάθε στιγμιότυπο ενός Agent ταυτοποιείται από αντικείμενο της κλάσης jade.core.AID();

- Ένα AID αποτελείται από ένα και μοναδικό όνομα και κάποιες διευθύνσεις.
- Μπορούμε να αποκτήσουμε πρόσβαση στο AID μέσω της μεθόδου Agent.getAID();
- Τα ονόματα που δίδονται στους πράκτορες είναι της μορφής <localname>@<platformname> και για κάθε πράκτορα εντός της πλατφόρμας θα πρέπει να είναι καθολικά μοναδικό. Αυτό συμβαίνει γιατί για να γίνει αναφορά σε έναν πράκτορα μέσα στην πλατφόρμα, αυτό συμβαίνει με χρήση του <localname> του. Τέλος, γνωρίζοντας το όνομα ενός πράκτορα μπορούμε να δούμε το AID που του έχει ανατεθεί ως εξής:

  AID id = newAID(localname, AID);

2. Εκχώρηση Ορισμάτων σε έναν Πράκτορα:

```
import jade.core.*;

public class mobileAgent extends Agent {

    private static final long serialVersionUID = 1L;

        //overrides constructor
        protected void setup(){
                System.out.println("Agent Created!ID:" + this.getAID().getName());
                Object [] args =this.getArguments();
                    if(args != null) {

                        System.out.println("The Argument are:");

                        for (int i = 0; i < args.length; ++i) {
                            System.out.println("-"+args[i]);
                            }
                    }
            }
}
```

Εικόνα 2.3: Κώδικας εκχώρησης ορισμάτων πράκτορα σε JADE

- Κατά την εκκίνηση ενός πράκτορα είναι δυνατόν να περαστούν σε αυτόν κάποια ορίσματα .

- Ο πράκτορας μπορεί να έχει πρόσβαση σε αυτά τα ορίσματα μέσω της μεθόδου getArguments() της κλάσης jade.core.Agent;

3. Τερματισμός Λειτουργίας ενός Πράκτορα

```
import jade.core.*;

public class mobileAgent extends Agent {

    private static final long serialVersionUID = 1L;

        //overrides constructor
        protected void setup(){
                System.out.println("Agent Created!ID:" + this.getAID().getName());
                Object [] args =this.getArguments();
                    if(args != null) {

                        System.out.println("The Argument are:");

                        for (int i = 0; i < args.length; ++i) {
                            System.out.println("-"+args[i]);
                            }
                    }

                    this.doDelete();
            }

        protected void takeDown() {
    //what to do when we delete agent (eg free resources)
            System.out.println("Agent Deleted");
        }
}
```

Εικόνα 2.4: Κώδικας τερματισμού λειτουργίας ενος πράκτορα σε JADE

- Μετά την κλήση της μεθόδου doDelete(); η μέθοδος takeDown(); προγραμματίζεται για εκτέλεση. Η takeDown() είναι υπεύθυνη για garbage collection, μηνύματα αποχαιρετισμού και άλλα. Υπάρχει η δυνατότητα να γίνει override της takeDown() για να προσθέσουμε κάποια λειτουργικότητα που επιθυμούμε.

4. Η κλάση Behaviour:

Η κλάση Behaviour ορίζει τις ενέργειες που εκτελεί ο μετακινούμενος πράκτορας. Για να δημιουργήσουμε Behaviours πρέπει να κάνουμε extend την κλάση jade.core.behaviours.Behaviour. Κάθε υποκλάση της κλάσης Behaviour πρέπει να υλοποιεί τις μεθόδους:

```
import jade.core.behaviours.Behaviour;

public void action() {
// ενέργειες που εκτελώνται από τη συγκεκριμένη behaviour
  }

public boolean done() {
  // καθορίζει το τέλος της behaviour και εκτελείται μετά τον τερματισμό
της action()
}
```

Κάθε πράκτορας μπορεί να τρέχει ταυτόχρονα πολλές behaviours. Το JADE ορίζει ότι κάθε πράκτορας τρέχει σε ένα thread και γι αυτό το λόγο οι behaviours δεν εκτελούνται σε πολλαπλά threads αλλά εναλλάσσονται στο ίδιο thread μέσω ενός behavior pool από το οποίο επιλέγονται προς εκτέλεση. Όταν η μέθοδος done() επιστρέψει TRUE, σημαίνει πως η συγκεκριμένη behaviour ολοκληρώθηκε και αφαιρείται από το pool των τρεχόντων behaviours.

Χωρίζονται σε τρεις τύπους:

- Behaviours μίας εκτέλεσης (One Shot Behaviours):
  Επεκτείνουν την κλάση jade.core.behaviours.Behavior.OneShotBehaviour
  Σε αυτές τις behaviours η μέθοδος done() επιστρέφει πάντα TRUE, οπότε η μέθοδος action καλείται μια μοναδική φορά *(Εικόνα 2.5)*.

- Κυκλικά επαναλαμβανόμενες Behaviours (Cyclic Behaviours):

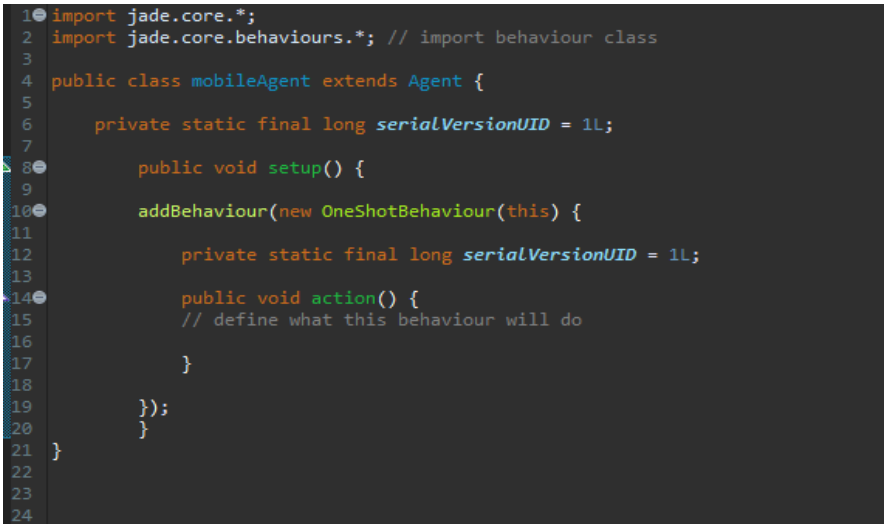
  Επεκτείνουν την κλάση jade.core.behaviours.Behavior.CyclicBehaviour.

  Σε αυτές τις behaviours η μέθοδος done() επιστρέφει πάντα FALSE οπότε η μέθοδος action καλείται.

- Σύνθετες Behaviours (Complex Behaviours):

  Περιέχουν μεταβλητές κατάστασης οι οποίες ρυθμίζουν την λειτουργία της μεθόδου action() ανάλογα με την τιμή τους.

  Η μέθοδος done() επιστρέφει TRUE μόνο όταν ικανοποιηθεί κάποια συνθήκη που έχουμε ορίσει εμείς.



```
import jade.core.*;
import jade.core.behaviours.*; // import behaviour class

public class mobileAgent extends Agent {

    private static final long serialVersionUID = 1L;

        public void setup() {

        addBehaviour(new OneShotBehaviour(this) {

            private static final long serialVersionUID = 1L;

            public void action() {
            // define what this behaviour will do

            }

        });
        }
}
```

Εικόνα 2.5: Παράδειγμα υλοποίησης μιας One Shot Behaviour

5. Η κλάση ACL Message:

Χρησιμοποιείται για την υλοποίηση της επικοινωνίας μέσω της ανταλλαγής μηνυμάτων μεταξύ των πρακτόρων.

- Αποστολή μηνύματος:

```
import jade.lang.acl.ACLMessage;

//Δημιουργία αντικειμένου ACLMessage κάνοντας χρήση του field INFORM για να ορίσουμε τον τύπο του μηνύματος
ACLMessage informMessage = new ACLMessage(ACLMessage.INFORM);
```

```
//Ανάκτηση AID με βάση το localname για να ξέρουμε το AID του πράκτορα που θέλουμε να του στείλουμε το μήνυμα
AID receiverAid = new AID(localname,AID.ISLOCALNAME);

//προσθήκη παραλήπτη με βάση το receiverAid
informMessage.addReceiver(receiverAid);

//προσθήκη περιεχομένου στο μήνυμα που θα στείλουμε
informMessage.setContent("Message body goes Here");
```

- Λήψη μηνύματος:

```
import jade.lang.acl.ACLMessage;

public void action() {
ACLMessage informMessage = thisAgent.receive();
    if (informMessage != null) {
    // Επεξεργασία του μηνύματος που έλαβε
    }
    else {
    block(); // αν το μήνυμα είναι NULL μπλόκαρε το
    }
          }
```

6. Κινητικότητα:

Στο JADE υποστηρίζεται η κινητικότητα του κώδικα ενός μετακινούμενου πράκτορα. Η μετακίνηση ενός πράκτορα μπορεί να ξεκινήσει είτε λόγω του ίδιου μέσω της μεθόδου doMove() είτε επειδή ο AMS έλαβε αίτηση από κάποιο container της πλατφόρμας η οποία ζητάει να μεταβεί ο μετακινούμενος πράκτορας σε κάποιο συγκεκριμένο container της πλατφόρμας.

Οι βασικές μέθοδοι που υλοποιούν τη λειτουργικότητα της μετακίνησης ενός πράκτορα είναι οι εξής:

- `void doMove(Location Destination);` // Η παράμετρος Location πρέπει να είναι αντικείμενο της κλάσης ContainerID

  .

- `void beforeMove();` // Το σύνολο των διεργασιών που πρέπει να εκτελεστούν λίγο πριν ο πράκτορας μετακινηθεί με κυριότερη όλων την μετάβαση της κατάστασης του πράκτορα από active σε transit, προκαλώντας διακοπή των τρεχουσών λειτουργιών που επιτελούσε και αποθήκευση της κατάστασής του για μετακίνηση.

- `void afterMove();`// Το σύνολο των διεργασιών που πρέπει να εκτελεστούν ακριβώς αμέσως μετά τη μετακίνηση του πράκτορα και πριν την ανάκτηση της κατάστασής του σε ενεργή (currentState = Active)

- `void doClone(Location Destination, String newName);` // Η παράμετρος Destination αναφέρεται στο target container στο οποίο θα κλωνοποιηθεί ο πράκτορας, ενώ η παράμετρος newname αναφέρεται στο νέο AID που λαμβάνει ο πράκτορας για αποφυγή διπλότυπων.

2.2.3 – Ανάπτυξη Εφαρμογής Μετακινούμενου Πράκτορα σε JADE.

Η ιδέα της υλοποίησης είναι η εξής:

Θα κατασκευάσω έναν μετακινούμενο πράκτορα στο Container-1, ο οποίος θα μετακινηθεί από το Container-1 στο Main-Container της πλατφόρμας, θα ελέγξει την διαθέσιμη μνήμη του JVM του κάθε κόμβου και στο τέλος θα μετακινηθεί στον κόμβο της πλατφόρμας με τη μεγαλύτερη διαθέσιμη μνήμη. Για τη διενέργεια του πειράματος χρησιμοποίησα δύο υπολογιστές συνδεδεμένους στο ίδιο δίκτυο, τους JIM-PC και DA-PC (τοπικά ονόματα των υπολογιστών μου).

Στη συνέχεια στο DA-PC ξεκίνησα τον Main-Container της Πλατφόρμας και στο JIM-PC δημιούργησα έναν πράκτορα, ο οποίος ξεκινάει στο Container-1 και θα επιτελέσει την εύρεση του συστήματος με τους περισσότερους διαθέσιμους πόρους εκ των δύο.

Η μετακίνηση με αυτόν τον τρόπο είναι εφαρμόσιμη μόνο σε Intra-Platform συστήματα, καθώς γίνεται χρήση του ContainerID και όχι του PlatformID.

Δηλαδή οι μετακινήσεις γίνονται μόνο από container σε container τα οποία υπάγονται στο Main-Container.

Οι κώδικες σε JADE είναι οι εξής: *(Εικόνα 2.6), (Εικόνα 2.9)*

```
1  import jade.core.*;
2
3  public class MainContainerCreate extends Agent {
4
5      private static final long serialVersionUID = 1L;
6
7      // Used to start Main-Container
8
9  }
10
11
```

Εικόνα 2.6: DA-PC (κώδικας που ξεκινάει το Main-Container της πλατφόρμας)

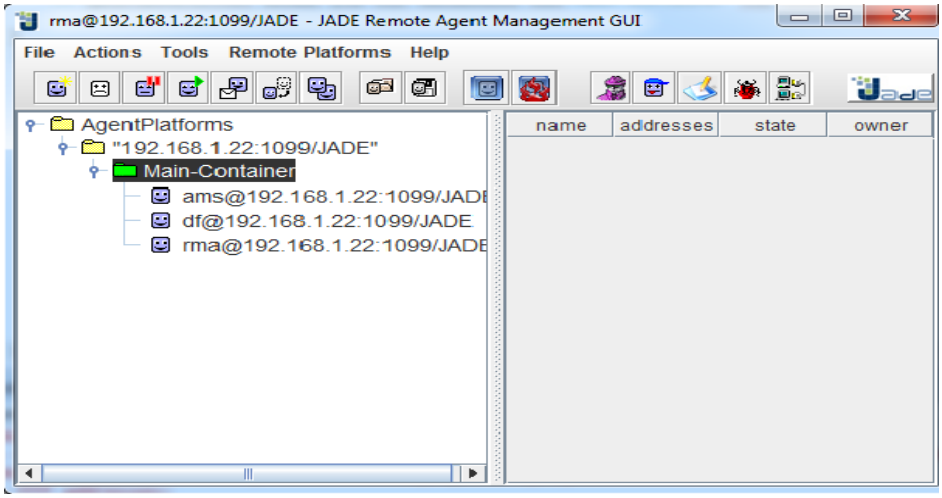
-gui: Εδώ δίνουμε μόνο την παράμετρο -gui ώστε να τρέξει και το RMA για να μπορούμε να έχουμε μια οπτική απεικόνιση του τι συμβαίνει στην πλατφόρμα μας.

Έτσι έχουμε κατασκευάσει το Main-Container της πλατφόρμας *(Εικόνα 2.7)*

```
INFO: MTP addresses:
http://DA-PC:7778/acc
Φεβ 06, 2020 11:12:36 M.M. jade.core.AgentContainerImpl joinPlatform
INFO: ----------------------------------------
Agent container Main-Container@192.168.1.22 is ready.
--------------------------------------------
```

Εικόνα 2.7: Επιτυχής εκκίνηση του Main-Container.

Το GUI του RMA της πλατφόρμας *(Εικόνα 2.8)*



Εικόνα 2.8: Γραφικη διεπαφή του Remote Agent Management της πλατφόρμας.

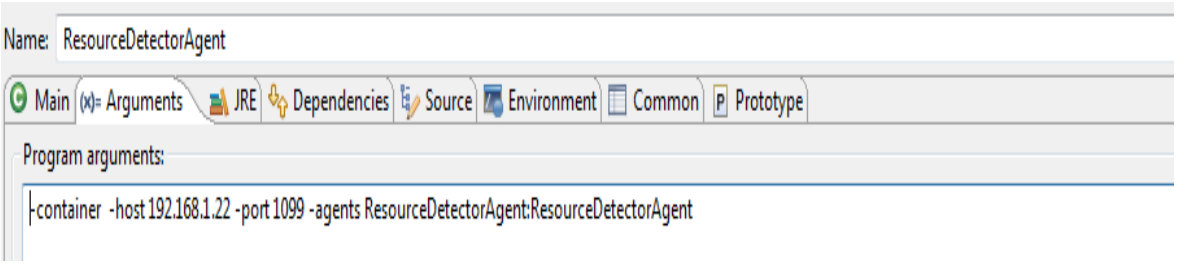
Η πλατφόρμα μας προς το παρόν αποτελείται από το Main-Container το οποίο τρέχει στο DA-PC και τους default πράκτορες AMS, DF και RMA.

Στη συνέχεια ακολουθεί ο κυρίως κώδικας υλοποίησης του μετακινούμενου πράκτορα σε JADE *(Εικόνα 2.9)*.

```java
/* Owner: Dimitris Gennimatas
 * University of Thessaly . E-CE Department .
 * 7/2/2020
 * JADE implementation of a Mobile Agent
 * who finds the PC with the most available JVM Memory
 * inside a two-container platform and finally migrates to the PC with the most available Memory
 */



import jade.core.*;
import java.lang.Runtime;
import jade.core.behaviours.*;

public class ResourceDetectorAgent extends Agent {


    int node = 0;
    String betterLocation;
    float mem1 = 0;
    float mem2 = 0;
    float max = 0;


    private static final long serialVersionUID = 1L;

public void setup() {


    addBehaviour(new CyclicBehaviour(this) {            //Create a new CyclicBehaviour

        private static final long serialVersionUID = 1L;

        public void action() {                          // implement action method inside addBevahiour

            switch(node) {                              // Create Finite State Machine

            case 0:
                String name = System.getenv("COMPUTERNAME");
                System.out.println("Free Memory in JVM of:" +name+ " is "+ Runtime.getRuntime().freeMemory() );
                float mem1 = Runtime.getRuntime().freeMemory();
                if(mem1>max) {
                    max = mem1;

                    betterLocation = "Container-1";
                }

                node++;
                myAgent.doMove(new ContainerID("Main-Container", null));
                break;


            case 1:
                String name2 = System.getenv("COMPUTERNAME");
                System.out.println("Free Memory in JVM of:" +name2+ " is "+ Runtime.getRuntime().freeMemory() );
                float mem2 = Runtime.getRuntime().freeMemory();

                if(mem2>max) {
                    max = mem2;

                    betterLocation =  "Main-Container";
                }

                node++;
                break;


            case 2:
                node++;
                myAgent.doMove(new ContainerID(betterLocation,null));
                break;

            case 3:
                System.out.println("Agent moved here" + "("+betterLocation+")"+ " because it has more free JVM Memory .");
                node++;
                break;


            default:
            myAgent.doDelete(); //delete agent when done to avoid loop of CylicBehaviour

            }

    }


});


}
}
```

Εικόνα 2.9: JIM-PC (κυρίως κώδικας υλοποίησης του πράκτορα / Δημιουργία του στο Container-1)

Για να τρέξουμε τον κώδικα από το Eclipse, πρέπει να εισάγουμε τα Arguments *(Εικόνα 2.10)*:



Εικόνα 2.10: Ορίσματα για σύνδεση στο Main-Container και δημιουργία πράκτορα ResourceDetectorAgent

Επεξήγηση Ορισμάτων:

-container: για τη δημιουργία του "Container-1" και όχι καινούργιας πλατφόρμας με καινούργιο "Main-Container". Σε περίπτωση που δεν εισάγουμε το όρισμα αυτό δημιουργείται καινούργιο Main-Container.

-host: Εδώ εισάγουμε τη διεύθυνση του Main-Container για να μπορέσει να συνδεθεί σε αυτό ο Container-1.

-port: Η πόρτα στην οποία ο "Main-Container" δέχεται συνδέσεις.

-agents: Εδώ δίνουμε ως παράμετρο το όνομα του πράκτορα που θα δημιουργηθεί και έχει τη μορφή *{Δικό_μας_όνομα:Όνομα_Κλάσης}*.

Μετά την εκτέλεση του παραπάνω κώδικα στο JIM-PC δημιουργούμε το Container-1 καθώς και τον πράκτορα με όνομα ResourceDetectorAgent ο οποίος δημιουργείται στο Container-1. Το Container-1 γνωρίζει που θα βρει το Main-Container στο οποίο θα συνδεθεί μέσω των παραμέτρων **-host** και **-port.** Στη συνέχεια εκτελείται ο κυρίως κώδικας, κατά τον οποίο:

- Ο Μετακινούμενος πράκτορας που δημιουργήσαμε στο JIM-PC["Container-1"] (δηλ. Ο ResourceDetectorAgent) λαμβάνει τη διαθέσιμη μνήμη του υπολογιστή μέσω του: `Runtime.getRuntime().freeMemory();` , το αποθηκεύει και το εκτυπώνει (η εκτύπωση

του ονόματος του PC όπου βρίσκεται εκείνη τη στιγμή ο πράκτορας γίνεται μέσω του `String name = System.getenv("COMPUTERNAME");`).

- Στη συνέχεια μετακινείται στο DA-PC[Main-Container] όπου με τον ίδιο τρόπο αποθηκεύει την ελεύθερη μνήμη αυτού του υπολογιστή.
- Τέλος συγκρίνει τα αποτελέσματα και πηγαίνει στον κόμβο με την περισσότερη ελεύθερη μνήμη επιστρέφοντας κατάλληλο μήνυμα όπου και διαγράφεται.

Ακολουθούν οι εικόνες εκτέλεσης:

JIM-PC["Container-1"]:

```
INFO: Service jade.core.event.Notification initialized
Φεβ 06, 2020 11:29:25 M.M. jade.core.AgentContainerImpl joinPlatform
INFO: ----------------------------------------
Agent container Container-1@192.168.1.20 is ready.
------------------------------------------------
Free Memory in JVM of:JIM-PC is 121634816
Agent moved here(Container-1) because it has more free JVM Memory .
```

Εικόνα 2.11: Εμφάνιση στο JIM-PC.

DA-PC["Main-Container"]:

```
Φεβ 06, 2020 11:29:28 M.M. jade.core.PlatformManagerImpl localAddNode
INFO: Adding node <Container-1> to the platform
Φεβ 06, 2020 11:29:28 M.M. jade.core.PlatformManagerImpl$1 nodeAdded
INFO: --- Node <Container-1> ALIVE ---
Free Memory in JVM of:DA-PC is 42295296
```

Εικόνα 2.12: Εμφάνιση στο DA-PC.

Τέλος να σημειωθεί ότι η σειρά εξέλιξης της εκτέλεσης είναι:

i. `Free Memory in JVM of JIM-PC is 121634816` (Δημιουργία πράκτορα και Έλεγχος μνήμης του PC που τρέχει το Container-1).
ii. Μετακίνηση στο Main-Container.
iii. Free Memory in JVM of DA-PC is 42295296 (Ελεγχος μνήμης του PC που τρέχει το Main-Container)
iv. Σύγκριση.

v. Μετακίνηση στο Container που τρέχει στο PC με την περισσότερη διαθέσιμη μνήμη (Στη δική μου εκτέλεση περισσότερη διαθέσιμη μνήμη έχει το JIM-PC που τρέχει το Container-1) γι αυτό και εμφανίζεται:

```
Agent moved here(Container-1) because it has more JVM Memory.
```

ΚΕΦΑΛΑΙΟ 3

ΚΙΝΔΥΝΟΙ ΑΣΦΑΛΕΙΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΚΙΝΟΥΜΕΝΩΝ ΠΡΑΚΤΟΡΩΝ

3.1 – Οι Τύποι των Επιθέσεων

Προτού γίνει ο διαχωρισμός και η κατηγοριοποίηση των επιθέσεων, ας ορίσουμε τον όρο «επίθεση» αναφερόμενοι στα συστήματα πληροφορίας.

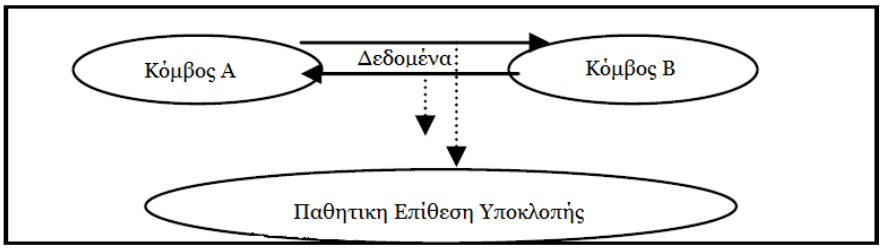
«Με τον όρο επίθεση αναφερόμαστε σε ένα σύνολο ενεργειών και συμπεριφορών, οι οποίες λαμβάνουν χώρα εντός του συστήματος, και αποτελούν απειλή για την απρόσκοπτη και ορθή λειτουργία του». **[13], [14]**

Στην κορυφή της πυραμίδας, οι επιθέσεις χωρίζονται σε 2 υποκατηγορίες. Στις ενεργητικές και παθητικές επιθέσεις, οι οποίες επεξηγούνται παρακάτω.

3.1.1 – Ενεργητικές/Παθητικές Επιθέσεις (Active/Passive Attacks)

Οι **ενεργητικές επιθέσεις** αποτελούν την ισχυρότερη απειλή εκ των δύο, καθώς ο τρόπος δράσης τους περιλαμβάνει όχι μόνο υποκλοπή πληροφοριών αλλά και τροποποίηση του τρόπου λειτουργίας του συστήματος. Κάποια παραδείγματα τέτοιων επιθέσεων είναι η μεταβολή (alternation), η επίθεση άρνησης υπηρεσίας (denial of service) κ.α.

Οι **παθητικές επιθέσεις**, παρόλο που δεν αποτελούν άμεσο κίνδυνο για τη λειτουργία του συστήματος, εμπεριέχουν μεγάλο κίνδυνο διαρροής ευαίσθητων πληροφοριών σε κακόβουλους χρήστες. Κάποια παραδείγματα τέτοιων επιθέσεων είναι η υποκλοπή (eavesdropping), η παράνομη αντιγραφή (illegal copying) κ.α. **[15].**



Εικόνα 3.1: Σχηματική Αναπαράσταση Παθητικής Επίθεσης Υποκλοπής

3.2 – Κατηγοριοποίηση των απειλών

3.2.1 – Επιθέσεις Μετακινούμενων Πρακτόρων στην Πλατφόρμα (Mobile Agent to Platform Attacks).

Οι επιθέσεις αυτές αναφέρονται στις απειλές που ενδέχεται να αντιμετωπίσουν οι κόμβοι του συστήματος όταν κακόβουλοι πράκτορες μετακινούνται ανενόχλητοι μέσα στο σύστημα, έχοντας τη δυνατότητα να επηρεάσουν τη λειτουργία τους. Ένας κακόβουλος χρήστης, εφόσον πάρει υπό τον έλεγχό του έναν κόμβο του συστήματος, μπορεί να εξαπλώσει την πρόσβασή του σε ολόκληρη την πλατφόρμα μέσω των μετακινούμενων πρακτόρων, «μολύνοντας» τους κόμβους έναν προς έναν. Λόγω αυτού, γίνεται αντιληπτό πως σε ένα τέτοιο σύστημα είναι ζωτικής σημασίας να υπάρχει επαρκής ασφάλεια από εξωτερικές απειλές μέσω τειχών προστασίας (Firewalls) ή και άλλων μεθόδων, καθώς αρκεί η παραβίαση ενός κόμβου για να γίνει παραβίαση ολόκληρης της πλατφόρμας.

Με λίγα λόγια και χρησιμοποιώντας ένα συμβολικό παράδειγμα, μπορούμε να πούμε ότι, όταν ένας κόμβος μολύνεται, οι μετακινούμενοι πράκτορες μπορούν να γίνουν οι μεταφορείς της μόλυνσης σε ολόκληρη την πλατφόρμα **[14],[16],[17],[18],[19].**

i. **<u>Επίθεση Μεταμφίεσης (Masquerading Attack):</u>**

Σε αυτή την επίθεση ο κακόβουλος πράκτορας μπορεί να αλλάξει την ταυτότητά του (π.χ. αλλάζοντας το ID attribute του), έτσι ώστε να αποκτήσει πρόσβαση σε πόρους στους οποίους προηγουμένως με την παλιά του ταυτότητα δεν είχε.

Επίσης, αυτή η επίθεση μπορεί να γίνει για να συγκαλύψει μια παράνομη δραστηριότητα που διατέλεσε εντός του συστήματος, ενοχοποιώντας έναν άλλο μετακινούμενο πράκτορα αντιγράφοντας αρχικά το ID του και μετά ξαναλλάζοντας το.

ii. <u>Μη Εξουσιοδοτημένη Πρόσβαση (Unauthorized Access):</u>

Ένας κακόβουλος πράκτορας μπορεί να επιχειρήσει να αποκτήσει πρόσβαση σε πόρους και υπηρεσίες της πλατφόρμας για τα οποία δεν έχει επαρκή αδειοδότηση.

Το βασικό πρόβλημα/ερώτημα είναι το πώς θα μπορεί ένας κόμβος του συστήματος να ταυτοποιεί τον κάθε μετακινούμενο πράκτορα που θα φτάνει σε αυτόν, ειδικά όταν έχει περάσει από πολλούς κόμβους μέχρι να φτάσει σε αυτόν, γεγονός το οποίο αυξάνει την πιθανότητα τροποποίησης του πράκτορα κατα τη διάρκεια της πορείας του.

iii. <u>Αποκήρυξη (Repudiation):</u>

Όταν ένας κόμβος της πλατφόρμας αρνείται να προβεί σε μια ανταλλαγή πληροφοριών ή εξυπηρέτησης του μετακινούμενου πράκτορα, όταν εκείνος του το ζητήσει, έχουμε το φαινόμενο της επίθεσης αποκήρυξης.

Μια παραλλαγή αυτής της επίθεσης είναι μετά την πραγματοποίηση της διαδικασίας ανταλλαγής ή εξυπηρέτησης, ο κόμβος να αρνείται ότι έχει «συναναστραφεί» με τον μετακινούμενο πράκτορα με τον οποίο όμως όντως συναναστράφηκε.

iv. <u>Επίθεση Ενόχλησης (Annoyance Attack):</u>

Σ’ αυτή την επίθεση, ο κακόβουλος πράκτορας «ενοχλεί» τον κόμβο τον οποίο επισκέπτεται. Παραδείγματα τέτοιας επίθεσης είναι επαναλαμβανόμενο άνοιγμα παραθύρων στον υπολογιστή και η αναπαραγωγή διάφορων ήχων.

Παρόλο που αυτή η επίθεση δεν αποτελεί ιδιαίτερο κίνδυνο για την πλατφόρμα καλό θα είναι να αντιμετωπίζεται.

v. <u>Άρνηση Υπηρεσίας (Denial of Service – DoS):</u>

Οι επιθέσεις DoS είναι από τις πιο διαδεδομένες στα συστήματα πληροφορίας. Στη περίπτωση των συστημάτων μετακινούμενων πρακτόρων, ένας κακόβουλος πράκτορας

μπορεί να καταναλώσει ένα πολύ μεγάλο ποσοστό των πόρων των κόμβων του συστήματος (επεξεργαστική ισχύς, εύρος δικτύου, μνήμη) τους οποίους δε χρειάζεται, με σκοπό να μην τους επιτρέπει να εξυπηρετούν άλλες σημαντικές αιτήσεις.

3.2.2 – Επιθέσεις Πλατφόρμας στους Μετακινούμενους Πράκτορες (Platform to Mobile Agent Attacks)

Οι επιθέσεις αυτές αναφέρονται στις απειλές που ενδέχεται να αντιμετωπίσουν οι μετακινούμενοι πράκτορες όταν επισκέπτονται κακόβουλους κόμβους του συστήματος, οι οποίοι στοχεύουν να αλλοιώσουν τη λειτουργία τους και να εξαπλώσουν τη μόλυνση σε όλη την πλατφόρμα μέσω αυτών ή να υποκλέψουν δεδομένα από τη μνήμη τους.

i. <u>Επίθεση Μεταμφίεσης (Masquerading Attack):</u>

Σε αυτή την περίπτωση, ένας κόμβος του συστήματος μπορεί να αποκρύψει την πραγματική του ταυτότητα αποκτώντας μια άλλη ενός έμπιστου κόμβου, ξεγελώντας με αυτό τον τρόπο τους μετακινούμενους πράκτορες οι οποίοι έχουν στην λίστα με τους κόμβους τους οποίους εμπιστεύονται την ταυτότητα του μεταμφιεσμένου κόμβου, ο οποίος πλέον μπορεί με την έλευσή τους, να αποσπάσει σημαντικές πληροφορίες ή ακόμα και να τροποποιήσει τον κώδικά τους.

ii. <u>Λανθασμένη/Τροποποιημένη Εκτέλεση Κώδικα (Incorrect Code Execution):</u>

Ο κακόβουλος κόμβος μεταβάλλει τον τρόπο με τον οποίο εκτελείται ο κώδικας της διεργασίας του μετακινούμενου πράκτορα, χωρίς όμως να αλλάζει κανένα μέρος του κώδικα και της κατάστασης εκτέλεσής του. Αυτό το επιτυγχάνει μέσω της επιστροφής ψευδών και λάθος αποτελεσμάτων κατά τη διάρκεια των υπολογισμών, γεγονός το οποίο μπορεί να δημιουργήσει μια χαοτική κατάσταση στο σύστημα.

iii. <u>Άρνηση Υπηρεσίας (Denial of Service – DoS):</u>

Κατά την άφιξή του σε έναν κόμβο, ο μετακινούμενος πράκτορας απαιτεί την ταχύτερη δυνατόν διεκπεραίωση της εργασίας που φέρει μαζί του. Κάποιες φορές όμως ο κόμβος

αρνείται την άμεση ή ακόμα και την εξ' ολοκλήρου περαίωση της διεργασίας που του έχει ζητηθεί, γεγονός που μπορεί να βάλει τον μετακινούμενο πράκτορα σε μια κατάσταση αδιεξόδου (deadlock).

iv. <u>Μεταβολή (Alternation):</u>

Στόχος αυτής της επίθεσης είναι είτε η τροποποίηση του πηγαίου κώδικα του μετακινούμενου πράκτορα (προσθήκη, αφαίρεση, αλλαγή), είτε της κατάστασης εκτέλεσής του, ή των δεδομένων του, από τον κακόβουλο κόμβο.

Με αυτό τον τρόπο ο κακόβουλος κόμβος μπορεί να «εμφυτεύσει» στον κώδικα του μετακινούμενου πράκτορα διάφορους ιούς (trojans, spyware, malware) ο οποίος γίνεται άθελά του φορέας τους.

Οποιαδήποτε από τις παραπάνω ενέργειες μπορεί να προκαλέσει μη ορθή λειτουργία και ταυτόχρονα πιθανή απώλεια ευαίσθητων πληροφοριών στον κόμβο που δημιούργησε τον συγκεκριμένο μετακινούμενο πράκτορα, αλλά και σε οποιονδήποτε κόμβο του συστήματος κατά την επίσκεψή του σε αυτόν.

Σύμφωνα με τα παραπάνω, μπορούμε να συμπεράνουμε ότι αποτελεί μια από τις σημαντικότερες απειλές ενός συστήματος επειδή εύκολα μπορεί να προκαλέσει την κατάρρευσή του.

3.2.3 – Επιθέσεις Μεταξύ Μετακινούμενων Πρακτόρων (Mobile Agent to Mobile Agent Attacks)

Όπως είχαμε αναφέρει και στην ενότητα του τρόπου λειτουργίας τους, οι μετακινούμενοι πράκτορες μπορούν να επικοινωνούν μεταξύ τους για ποικίλους λόγους. Αυτή τους η ιδιότητα μπορεί να χρησιμοποιηθεί για επιτέλεση κακόβουλων πράξεων, εφόσον κάποιος καταφέρει να αποκτήσει πρόσβαση έστω και σε έναν, σε ολόκληρο το σύστημα.

i. Επίθεση Μεταμφίεσης (Masquerading Attack):

Όπως είδαμε και στις άλλες δυο κατηγορίες επιθέσεων, οι επιθέσεις τύπου μεταμφίεσης αφορούν την απόκρυψη ταυτότητας του κακόβουλου χρήστη και την «μεταμφίεσή» του σε κάποιον άλλο χρήστη (κόμβος/πράκτορας).

Εδώ ο κακόβουλος χρήστης ο οποίος έχει υπό την κατοχή του έναν μετακινούμενο πράκτορα του συστήματος, αλλάζει την τρέχουσα και πραγματική του ταυτότητα με ενός άλλου πράκτορα του συστήματος. Η επιλογή του «θύματος» συνήθως βασίζεται στον στόχο που θέλει να πετύχει.

Σε αυτή την περίπτωση που μιλάμε μόνο για επίθεση μεταξύ πρακτόρων, ο βασικός στόχος του κακόβουλου χρήστη είναι να υποκλέψει πληροφορίες από έναν άλλο πράκτορα μέσω της λειτουργικότητας της επικοινωνίας μεταξύ δυο πρακτόρων, αλλά δεν έχει το κατάλληλο επίπεδο εμπιστοσύνης από τον άλλο πράκτορα.

Έτσι, αρχικά κλέβει την ταυτότητα ενός αξιόπιστου πράκτορα του συστήματος (ο οποίος μπορεί να ανταλλάξει ευαίσθητες πληροφορίες με τους υπολοίπους) και στη συνέχεια «αλιεύει» τις πληροφορίες που θέλει, ενώ ο άλλος πράκτορας νομίζει ότι έχει επικοινωνήσει με τον άλλον έμπιστο πράκτορα.

ii. Μη Εξουσιοδοτημένη Πρόσβαση(Unauthorized Access):

Όταν ένας κακόβουλος πράκτορας αποκτάει πρόσβαση σε κάποιες δημόσιες μεθόδους (public methods) ενός άλλου μη-κακόβουλου πράκτορα του συστήματος, με σκοπό να τροποποιήσει τον κώδικά του έτσι ώστε να αλλοιώσει την λειτουργία του και να τον μετατρέψει σε κακόβουλο.

iii. Άρνηση Υπηρεσίας (Denial of Service – DoS):

Υπάρχει η δυνατότητα η επίθεση άρνησης υπηρεσίας να συμβεί μεταξύ δυο πρακτόρων ή μεταξύ ενός πράκτορα και μιας ομάδας πρακτόρων του συστήματος όπου και έχουμε Κατανεμημένη Επίθεση Άρνησης Υπηρεσίας (Distributed Denial of Service Attack - DDoS). Σε κάθε περίπτωση, ο κακόβουλος πράκτορας στέλνοντας πολλά μηνύματα σε μικρό χρονικό διάστημα, γεμίζει το message buffer του θύματος με άχρηστες πληροφορίες, έχοντας ως αποτέλεσμα να μην μπορεί να ανταποκριθεί άμεσα ή ακόμα και καθόλου σε άλλες μη-κακόβουλες και σημαντικές ανταλλαγές μηνυμάτων.

ΚΕΦΑΛΑΙΟ 4

ΑΝΤΙΜΕΤΩΠΙΣΗ ΑΠΕΙΛΩΝ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΚΙΝΟΥΜΕΝΩΝ ΠΡΑΚΤΟΡΩΝ

4.1 – Θεμελιώδεις Απαιτήσεις Ασφάλειας των Συστημάτων Πρακτόρων

Για να μπορεί ένα σύστημα πληροφορίας να είναι όσο το δυνατόν ασφαλέστερο από διαφόρων ειδών επιθέσεις , είναι υψίστης σημασίας η ύπαρξη μηχανισμών για την διασφάλιση των θεμελιωδών αναγκών ασφάλειας. Οι θεμελιώδεις αυτές απαιτήσεις είναι οι εξής**: [13],[14]**

i. Έλεγχος Ταυτότητας – Αυθεντικοποίηση (Authentication – Authorization):

Αυθεντικοποίηση είναι η διαδικασία κατά την οποία μια οντότητα ενός συστήματος (άνθρωπος, υπολογιστής, πράκτορας), ελέγχει την ταυτότητα μιας άλλης οντότητας με την οποία πρόκειται να αλληλεπιδράσει, έτσι ώστε να βεβαιωθεί ότι πρόκειται όντως γι' αυτή την οποία δηλώνει ότι είναι. Μόλις ολοκληρωθεί η αυθεντικοποίηση και είναι επιτυχής, τότε επιτρέπεται η αμφίδρομη ανταλλαγή πόρων και πληροφοριών.

ii. Ιδιωτικότητα – Εμπιστευτικότητα (Privacy – Confidentiality):

Τα ευαίσθητα δεδομένα ενός συστήματος πρέπει να είναι προσβάσιμα μόνο από αναγνωρισμένους και επιβεβαιωμένους διαχειριστές και οντότητες. Σε ένα σύστημα πρακτόρων, τα ευαίσθητα δεδομένα είναι συνήθως οι πόροι και οι υπηρεσίες που προσφέρει το σύστημα, ο πηγαίος κώδικας και τα αποθηκευμένα στην τοπική τους μνήμη δεδομένα των πρακτόρων.

iii. Διαθεσιμότητα (Availability):

Ο όρος διαθεσιμότητα αναφέρεται στην ορθολογική χρήση των πόρων και των υπηρεσιών ενός συστήματος, έτσι ώστε να μπορεί να εξυπηρετεί δίκαια όλες τις αιτήσεις.

Γι’ αυτόν το λόγο πρέπει κάθε πράκτορας και κάθε κόμβος του συστήματος ανά πάσα στιγμή να είναι σε ετοιμότητα να διεκπεραιώσουν τη δουλειά που θα τους ανατεθεί όσο πιο άμεσα γίνεται, αποφεύγοντας φαινόμενα deadlock και starvation.

iv. Ακεραιότητα (Integrity):

Η αρχή αυτή αναφέρεται στη βαρυσήμαντη απαίτηση ενός συστήματος πληροφορίας για τη διατήρηση της ακεραιότητας των δεδομένων του, αποφεύγοντας ανεπιθύμητες τροποποιήσεις τους. Είναι μια από τις βασικότερες αρχές ασφαλείας η οποία επιτρέπει την αμοιβαία εμπιστοσύνη μεταξύ των οντοτήτων του συστήματος καθώς διασφαλίζει πως ό,τι δεδομένα λαμβάνει μια οντότητα από κάποια άλλη είναι αληθή και δε θα προσβάλλουν την ομαλή λειτουργία του συστήματος.

v. Λογοδοσία - Μη Αποκήρυξη (Accountability – Non Repudiation):

Καθετί που συμβαίνει στο σύστημα αποθηκεύεται σε μια μόνιμη μνήμη, έτσι ώστε κάποιος διαχειριστής να μπορεί να ελέγξει το ιστορικό των συμβάντων. Συνήθως τα συμβάντα αυτά αφορούν την ανταλλαγή δεδομένων και τη χρήση πόρων. Κάθε οντότητα που συμμετέχει σε ένα τέτοιο συμβάν θα πρέπει οποιαδήποτε στιγμή να είναι σε θέση να αποδείξει ότι συμμετείχε σε αυτό, καθώς και να μην αρνηθεί τη συμμετοχή του. Έτσι, για παράδειγμα, όταν ένας πράκτορας κάνει χρήση των πόρων ενός εξυπηρετητή για να εκτελέσει μια διεργασία θα πρέπει και οι δυο πλευρές να αναγνωρίσουν οποιαδήποτε στιγμή στο μέλλον, ότι συμμετείχαν σε αυτό το συμβάν καθώς και να μπορούν να αποδείξουν τον ακριβή τρόπο αλληλεπίδρασής τους, παραθέτοντας τα δεδομένα και τους πόρους που χρησιμοποιήθηκαν καθώς και το τελικό αποτέλεσμα της αλληλεπίδρασης.

vi. Έλεγχος Πρόσβασης (Access Control):

Η ύπαρξη ελέγχου πρόσβασης σε ένα σύστημα έχει ως στόχο να αποτρέψει διάφορες οντότητες να έχουν πρόσβαση σε πληροφορίες και πόρους τους οποίους δε δικαιούνται. Κάθε έμπιστη οντότητα έχει τον δικό της κωδικό τον οποίο της έχει αναθέσει ο διαχειριστής του συστήματος με τον οποίο μπορεί να έχει πρόσβαση σε λειτουργίες του συστήματος στις οποίες δεν έχουν πρόσβαση οι οντότητες που δεν έχει ορίσει ο διαχειριστής. Έτσι γίνεται

ένας διαχωρισμός των προνομίων και της δικαιοδοσίας μεταξύ των οντοτήτων του συστήματος.

4.2 – Αντιμετώπιση Απειλών των Συστημάτων Πρακτόρων

Οι μέθοδοι που περιγράφονται στη συνέχεια, όπως αναλύθηκαν και στο κεφάλαιο 3, αφορούν την αντιμετώπιση των επιθέσεων τύπου:

- Των μετακινούμενων πρακτόρων στην πλατφόρμα.
- Της πλατφόρμας στους μετακινούμενους πράκτορες.

Στο σημείο αυτό, είναι θεμιτό να ορίσουμε τι εννοούμε με τον όρο «αντιμετώπιση». Με τον όρο αντιμετώπιση, εννοούμε είτε τις τεχνικές πρόληψης που θα χρησιμοποιήσουμε για να αποφύγουμε εξ' ολοκλήρου την παραβίαση του συστήματος μας, είτε τις τεχνικές εντοπισμού και εκκαθάρισης της απειλής εφόσον αυτή έχει ήδη συμβεί.

Βέβαια, οι μηχανισμοί εντοπισμού των απειλών δεν περιλαμβάνουν και την εκκαθάρισή τους και σε αρκετές περιπτώσεις είναι πολύ δύσκολο να συμβεί αυτό. Επομένως, η πρόληψη εναντίον των απειλών είναι πιο επιθυμητή από τον ύστερο εντοπισμό και την αβέβαια αποτελεσματική εκκαθάρισή τους. Στην ανάλυση που ακολουθεί, γίνεται κατηγοριοποίηση των μηχανισμών που θα συζητηθούν σε πρόληψης και εντοπισμού.

4.2.1 – Αντιμετώπιση Επιθέσεων Πλατφόρμας στους Μετακινούμενους Πράκτορες (Platform to Mobile Agent Attacks Countermeasures)

1. **<u>Μηχανισμοί Εντοπισμού:</u>**

i. ***Παρακολούθηση Εκτέλεσης (Execution Monitoring)*:** Η ιδέα αυτή δημοσιεύτηκε για πρώτη φορά το 1994 και επανεξετάστηκε τη δεκαετία του '00 **[16].** Η ιδέα του μηχανισμού βασίζεται στην παρακολούθηση των καταγραφών των συμβάντων (logs) της πλατφόρμας από τους διαχειριστές του συστήματος κατά τις αλληλεπιδράσεις που συμβαίνουν. Σε περίπτωση που ένας κακόβουλος κόμβος τροποποιήσει τον πηγαίο κώδικα ή και τα δεδομένα κάποιου πράκτορα, θα μπορούν να εντοπιστούν οι τροποποιήσεις που έγιναν και το πότε έγιναν. Κάθε οντότητα του συστήματος έχει ένα

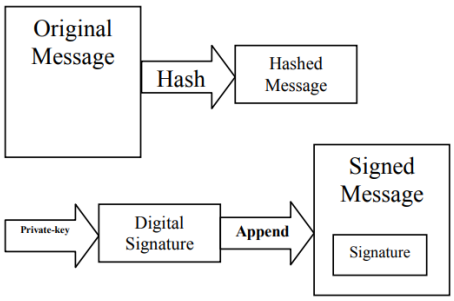
δημόσιο και ένα ιδιωτικό κλειδί (public and private key) τα οποία χρησιμοποιούνται ως ψηφιακή υπογραφή (digital signature) για την αναγνώριση των συμμετεχόντων των αλληλεπιδράσεων. Η πιο εκσυγχρονισμένη εκδοχή **[16], [20]**, περιλαμβάνει έναν authentication server ο οποίος ουσιαστικά αναλαμβάνει τη διαδικασία ελέγχου του πράκτορα. Κάθε φορά που ένας πράκτορας επισκέπτεται ένα νέο κόμβο-εξυπηρετητή, πριν επιτελέσει οποιαδήποτε ενέργεια σε αυτόν, στέλνει ένα αντίγραφο του στον authentication server. Όταν τελειώσει η διαδικασία εκτέλεσής του στον συγκεκριμένο κόμβο-εξυπηρετητή, το log με τις ενέργειες που πραγματοποιήθηκαν στέλνεται στον authentication server από τον κόμβο-εξυπηρετητή. Τότε ο authentication server εξομοιώνει την εκτέλεση των ενεργειών που έγιναν βασιζόμενος στο log που του έστειλε ο κόμβος-εξυπηρετητής σ' ενα αντίγραφο του πράκτορα. Τέλος, συγκρίνει τις τελικές καταστάσεις του πράκτορα και του αντιγράφου του και αν αυτές ταυτίζονται δεν υπάρχει υποψία αλλοίωσης, αν δεν ταυτίζονται τότε είναι πιθανό να έχει υπάρξει αλλοίωση του πράκτορα. Ο authentication server κάνει αυτή τη διαδικασία για κάθε πράκτορα του συστήματος κάθε φορά που ένας από αυτούς μετακινείται σε έναν νεο κόμβο και έως ότου επιστρέψει στον κόμβο που τον δημιούργησε.

Η τεχνική αυτή, παρόλο που είναι αρκετά βελτιωμένη σε σχέση με την πιο πρώιμη έκδοσή της, όπου ο διαχειριστής του συστήματος πρέπει να ελέγχει τους πράκτορες του συστήματος έναν προς έναν ελέγχοντας τα logs, η οποία είναι μια χρονοβόρα και δύσκολη διαδικασία, έχει αρκετές ατέλειες. Αρχικά, επιβαρύνει αρκετά το σύστημα καθώς διεξάγει πολλούς ελέγχους, για κάθε πράκτορα και για κάθε μετακίνησή του, χωρίς πολλές φορές να χρειάζεται ενώ θα ήταν πιο ορθολογικό να βάσιζε τους ελέγχους του σε κάποια στοιχεία έτσι ώστε να έκανε λιγότερους ελέγχους σε πιο ύποπτα περιστατικά. Τέλος, σε μεγάλα συστήματα είναι λογικό να υπάρχουν περισσότεροι από ένας authentication servers. Αυτό έχει ως αποτέλεσμα, και εφόσον η επιλογή του authentication server σε κάθε περίπτωση γίνεται από τον κόμβο-εξυπηρετητή και όχι από τον πράκτορα, να υπάρξουν σενάρια παραβίασης του authentication server από τον κακόβουλο κόμβο, έτσι ώστε να μην αναγνωρίσει την αλλοίωση που θα προκαλέσει στον πράκτορα που θα τον επισκεφτεί.

ii. ***Συνεργαζόμενοι Πράκτορες (Cooperative Agents):*** Η βασική ιδέα **[21]** του μηχανισμού αυτού είναι η ανάθεση των πολύ σημαντικών ενεργειών που έχει να επιτελέσει ένας πράκτορας και σε έναν co-op agent. Δηλαδή, οι δυο πράκτορες έχουν κοινά δεδομένα

στη μνήμη τους, επικοινωνούν με δικό τους κωδικοποιημένο τρόπο και καλούνται να επιτελέσουν τις ίδιες ακριβώς διεργασίες, με την προϋπόθεση όμως ότι δεν πρέπει να επισκέπτονται ποτέ τον ίδιο κόμβο. Κάθε φορά που ο πράκτορας μετακινείται από κόμβο σε κόμβο χρησιμοποιεί ένα κρυπτογραφημένο κανάλι για να στείλει πληροφορίες σχετικά με τη διαδρομή που θα ακολουθήσει στον πράκτορα-συνεργάτη του. Όταν υπάρξει υποψία ότι συμβαίνει κάτι κακόβουλο, όπως για παράδειγμα ο κόμβος να στείλει τον πράκτορα κάπου άλλου από το αναμενόμενο, ο πράκτορας συνεργάτης γνωρίζοντας την πορεία που πρέπει να ακολουθήσει, τον επαναφέρει στη επιθυμητή κατεύθυνση. Η λογική πίσω από αυτή την ιδέα είναι πως μέσω των αλληλοεπιβεβαιώσεων και των μειωμένων πιθανοτήτων να βρεθούν και οι δυο πράκτορες σε κακόβουλους κόμβους ενισχύεται η σωστή εκτέλεση της κοινής τους διεργασίας. Τέτοιοι πράκτορες χρησιμοποιούνται συνήθως σε συστήματα δημοπρασιών και πληρωμών.

iii. ***Ψηφιακή Υπογραφή (Digital Signature):*** Η ψηφιακή υπογραφή είναι ένα θεμελιακό στοιχείο της κρυπτογραφίας και χρησιμοποιείται για την τήρηση βασικών απαιτήσεων ασφαλείας ενός συστήματος **[22]**, όπως τις ορίσαμε προηγουμένως. Μια γραφική απεικόνιση της διαδικασίας παρουσιάζεται στην *(Εικόνα 4.1)*. Με αυτή τη μέθοδο, η πλατφόρμα - ιδιοκτήτης ενός πράκτορα δημιουργεί μέσω συναρτήσεων ένα κρυπτογραφημένο hash sum του πηγαίου κώδικα ή/και των δεδομένων του πράκτορα. Το hash sum αυτό μετά προσκολλάται στον πράκτορα, πιθανώς ως attribute, έτσι ώστε οποιοσδήποτε κόμβος της πλατφόρμας να μπορεί να βεβαιώσει ότι ο κώδικας και τα δεδομένα του συγκεκριμένου πράκτορα είναι αναλλοίωτα, καθώς και να εντοπίσει την ταυτότητα του κόμβου που τον δημιούργησε.



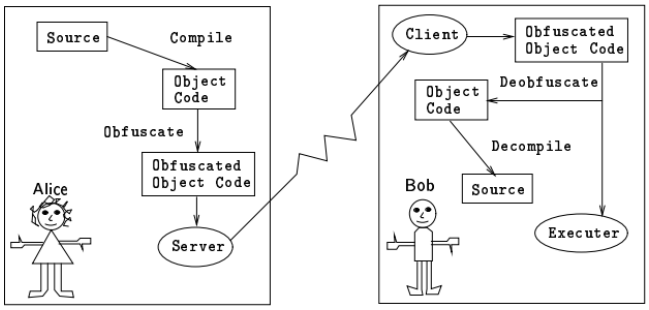
Εικόνα 4.1: Διαδικασία Ψηφιακής Υπογραφής

iv. ***Μερική Ενθυλάκωση Αποτελέσματος (Partial Result Encapsulation):*** Η μέθοδος αυτή χρησιμοποιείται για την επιβεβαίωση της ορθότητας των υπολογισμών που επιτελεί ο πράκτορας σε κάθε κόμβο της πλατφόρμας που επισκέπτεται. Σύμφωνα με τον Jansen **[23]** υπάρχουν τρεις τρόποι μερικής ενθυλάκωσης αποτελέσματος: Χρησιμοποιώντας τον ίδιο τον πράκτορα, τον κόμβο της πλατφόρμας που επισκέπτεται και τέλος έναν ανεξάρτητο έμπιστο κόμβο ο οποίος αναλαμβάνει την ενθυλάκωση του αποτελέσματος χρησιμοποιώντας ψηφιακές υπογραφές και χρονοσφραγίδες.

2. <u>Μηχανισμοί Πρόληψης:</u>

i. ***Συσκότιση Κώδικα (Code Obfuscation):*** Η συσκότιση είναι μια διαδικασία κατά την οποία ο πηγαίος κώδικας μιας εφαρμογής ανακατατάσσεται, μπερδεύεται και τροποποιείται στοχευμένα και οικειοθελώς έτσι ώστε να είναι δυσνόητος προς τον κόσμο αλλά επιτελεί ακριβώς τις ίδιες λειτουργίες με τον αρχικό κώδικα. Η λειτουργία του obfuscation, de-obfuscation παρουσιάζεται στην *(Εικόνα 4.2)*. Σύμφωνα με τους **[24]**, η συσκότιση κώδικα πραγματοποιείται με δυο τρόπους. Ο πρώτος είναι η μεταμόρφωση σχεδίου (layout transformation) ο οποίος στοχεύει στην οπτική αλλαγή του κώδικα (όπως για παράδειγμα η αλλαγή ονόματος μεταβλητών) αφήνοντας ανέγγιχτο σημασιολογικά και συντακτικά τον κώδικα. Γίνεται προφανές ότι είναι σχετικά ανίσχυρη μέθοδος συσκότισης και μπορεί εύκολα να παραβιασθεί. Ο δεύτερος τρόπος συσκότισης (control/data transformation) επιτελεί πιο ουσιαστικό έργο καθώς στοχεύει στην εις βάθος τροποποίηση του κώδικα, αλλάζοντας τον τρόπο με τον οποίο υπολογίζονται οι τιμές των μεταβλητών, αλλάζει τις δομές δεδομένων που μπορεί να χρησιμοποιηθούν, προσθέτει κομμάτια κώδικα τα οποία ουσιαστικά έχουν στόχο να μπερδέψουν και δεν χρησιμοποιούνται κάπου (dead code) και κάποιες φορές τροποποιούν εντελώς τον αλγόριθμο λειτουργίας του προγράμματος. Για Java υπάρχουν αρκετοί obfuscators οι οποίοι μπορούν να βοηθήσουν στην ενίσχυση της ασφάλειας μιας πλατφόρμας μετακινούμενων πρακτόρων σε JADE, τροποποιώντας τον κώδικα των πρακτόρων ώστε να μην μπορούν να παραβιαστούν. Ένας αξιόπιστος και δωρεάν είναι ο JODE ο οποίος μπορεί να τροποποιήσει τον κώδικα κυρίως με τον τρόπο αλλαγής ονομάτων μεθόδων, κλάσεων και μεταβλητών, της προσθήκης (και αφαίρεσης) κομματιών dead code, καθώς και να αλλάξει την κατανομή των μεταβλητών στη στοίβα **[25]** . Τέλος, η συσκότιση κώδικα μπορεί να καθυστερήσει αλλά όχι να αποτρέψει

πλήρως τις επιθέσεις σε έναν πράκτορα καθώς ένας κακόβουλος χρήστης θα μπορέσει κάποια στιγμή, η οποία εξαρτάται από την υπολογιστική ισχύ που διαθέτει, να κάνει reverse engineer τον τροποποιημένο κώδικα στον αυθεντικό **[26]**.



Εικόνα 4.2: Διαδικασία Συσκότισης Κώδικα

ii. ***Υπολογισμός με Κρυπτογραφημένες Συναρτήσεις (Computation with Encrypted Functions)*:** Ο υπολογισμός με κρυπτογραφημένες συναρτήσεις αποσκοπεί στην παροχή ενός ασφαλούς μηχανισμού μέσω του οποίου οι πράκτορες μπορούν να εκτελέσουν κρίσιμες εργασίες σε μη ασφαλή περιβάλλοντα, διατηρώντας παράλληλα την ακεραιότητα και την εμπιστευτικότητα των υπολογισμών τους **[21] .** Η ιδέα του μηχανισμού αυτού βασίζεται στο γεγονός ότι μπορούμε μέσω κάποιας συνάρτησης *f* να κρυπτογραφήσουμε τον κώδικα του μετακινούμενου πράκτορα και οποιοσδήποτε κόμβος του συστήματος να έχει τη δυνατότητα να εκτελέσει τον κρυπτογραφημένο κώδικα χωρίς να χρειάζεται πρώτα να τον αποκρυπτογραφήσει.
Το πρόβλημα αυτό έχει παρουσιαστεί από τους Sander & Tschudin με την ακόλουθη μορφή **[27]**.

- *Η Alice έχει έναν αλγόριθμο για τον υπολογισμό μιας συνάρτησης f.*
- *Ο Bob έχει ένα input x και είναι πρόθυμος να υπολογίσει το f(x) γι' αυτήν, αλλά η Alice δε θέλει ο Bob να γνωρίζει κάτι ουσιαστικό για τον τρόπο υλοποίησης της f.*
- *Τέλος, ο Bob δεν θα πρέπει να χρειάζεται να αλληλεπιδράσει με την Alice κατά τη διάρκεια υπολογισμού του f(x).*

Ειδικότερα, ο αλγόριθμος υλοποίησης της παραπάνω διαδικασίας, όπως φαίνεται και στην *(Εικόνα 4.3)* είναι ο εξής:

i. Η Alice κωδικοποιεί την συνάρτηση *f*.

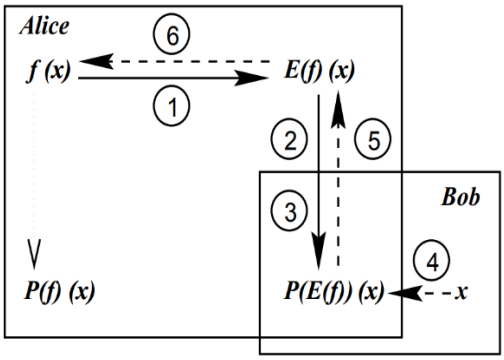
ii. Η Alice δημιουργεί ένα πρόγραμμα *P(E(f))* το οποίο υλοποιεί τη συνάρτηση *E(f)*.

iii. Η Alice στέλνει το *P(E(f))* στον Bob.

iv. Ο Bob υπολογίζει το αποτέλεσμα της συνάρτησης για κάποια τιμή του x, [*P(E(f)), input = x*].

v. Ο Bob στέλνει την τιμή που υπολόγισε P(E(*f*))(x) στην Alice.

vi. Η Alice αποκωδικοποιεί το *P(E(f))(x)* και λαμβάνει την τιμή της συνάρτησης *f(x)* για το input x του Bob μέσω της διαδικασίας



Εικόνα 4.3: Αλγόριθμος Υλοποίησης Υπολογισμού με Κρυπτογραφημένη Συνάρτηση

iii. ***Παραγωγή Περιβαλλοντικού Κλειδιού (Environmental Key Generation)*:** Η παραγωγή περιβαλλοντικού κλειδιού επιτρέπει σε έναν μετακινούμενο πράκτορα να εκτελέσει ένα σαφώς ορισμένο σύνολο ενεργειών, αν και μόνο αν, μια περιβαλλοντική συνθήκη είναι, ή γίνει, αληθής. Οι πράκτορες οι οποίοι κρυπτογραφούνται με τέτοια κλειδιά συνήθως παραμένουν αδαείς ως προς την λειτουργικότητά τους έως ότου η περιβαλλοντική συνθήκη γίνει αληθής **[28].** Όταν ένας πράκτορας φτάσει σε έναν κόμβο της πλατφόρμας, ψάχνει να βρει συγκεκριμένες πληροφορίες/τιμές. Αν τις βρει, τότε μπορεί να παράξει ένα κλειδί. Αυτό το κλειδί «αποκωδικοποιεί» τον πηγαίο κώδικα και τα δεδομένα που φέρει ο πράκτορας τα οποία πλέον είναι ορατά. Αυτή η μέθοδος είναι κυρίως χρήσιμη για επιθέσεις τύπου μεταμφίεσης (masquerading) καθώς κάποιος κακόβουλος κόμβος της πλατφόρμας θα πρέπει να έχει υπό μοναδική κατοχή τις συνθήκες για να μπορεί να παράξει ένα κλειδί για ένα συγκεκριμένο πράκτορα και εν τέλει να αποκτήσει πρόσβαση στον κώδικα και τα δεδομένα του, γεγονός που είναι πολύ

δύσκολο. Επίσης, ανάλογα με τις πληροφορίες/τιμές που διαθέτει ο κάθε κόμβος, ο πράκτορας μπορεί να αποφασίσει το ποσοστό δικαιοδοσίας και πρόσβασης που θα δώσει σε αυτόν τον κόμβο.

Έστω ***N*** ακέραιος αριθμός που αντιστοιχεί σε μια περιβαλλοντική παρατήρηση, ***H*** μια μονόδρομη συνάρτηση κατακερματισμού (hash function) και ***M*** hash της συνάρτησης που απαιτείται για να γίνει αληθής η περιβαλλοντική συνθήκη. Ο μετακινούμενος πράκτορας δεν γνωρίζει την τιμή του ***N*** αλλά μόνο την τιμή του ***H(N)***.

Η παραπάνω διαδικασία μπορεί να περιγραφεί από την συνθήκη:

if (***H(N) == M) {***

K = N

}

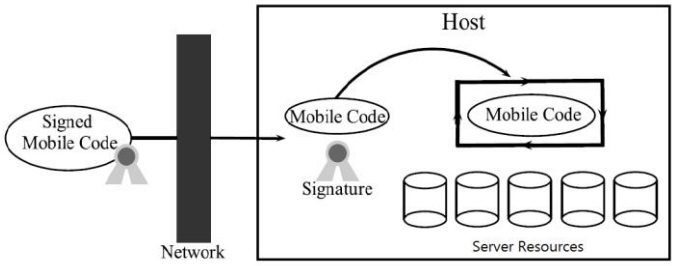
Το ***K*** λαμβάνει την τιμή του περιβαλλοντικού κλειδιού το οποίο θα χρησιμοποιηθεί για την αποκωδικοποίηση των δεδομένων και του κώδικα του μετακινούμενου πράκτορα στον συγκεκριμένο κόμβο της πλατφόρμας.

4.2.2 – Αντιμετώπιση Επιθέσεων Μετακινούμενων πρακτόρων στην πλατφόρμα (Mobile Agent to Platform Attacks Countermeasures)

1. **<u>Μηχανισμοί Εντοπισμού:</u>**

i. ***Υπογραφή Κώδικα (Code Signing):***. Κατά τη δημιουργία του από έναν κόμβο της πλατφόρμας ο μετακινούμενος πράκτορας υπογράφεται έτσι ώστε να μπορεί να ταυτοποιηθεί από όλους τους κόμβους της πλατφόρμας κατά τις μετακινήσεις του. Συγκεκριμένα, φέροντας την υπογραφή του κόμβου δημιουργού του, επιτρέπει στους κόμβους του συστήματος τους οποίους επισκέπτεται να ελέγξουν αν ο κώδικας του ή τα δεδομένα του έχουν τροποποιηθεί από τη στιγμή της δημιουργίας του. Με αυτό τον τρόπο επιτυγχάνεται μια αποτελεσματική διαδικασία αυθεντικοποίησης της ακεραιότητας του κώδικα και των δεδομένων του. Οι κυρίως τρόποι με τους οποίους επιτελείται η διαδικασία υπογραφής κώδικα είναι μέσω ψηφιακών υπογραφών και μονόδρομων συναρτήσεων κατακερματισμού. Παράδειγμα υλοποίησης υπογραφής κώδικα με μονόδρομες συναρτήσεις κατακερματισμού αποτελεί το Microsoft Authenticode, το οποίο χρησιμοποιείται στο framework ActiveX.

Το κυριότερο μειονέκτημα της υπογραφής κώδικα είναι ότι ο κόμβος ο οποίος δημιουργεί τον πράκτορα θα πρέπει να είναι έμπιστος, έτσι ώστε να θεωρηθεί η υπογραφή του έγκυρη. Αυτό ελλοχεύει πολλούς κινδύνους καθώς αν κάποιος κακόβουλος χρήστης αποκτήσει πρόσβαση σε κάποιον αξιόπιστο κόμβο της πλατφόρμας, θα μπορεί να δημιουργεί πράκτορες με κακόβουλο κώδικα οπότε η υπογραφή τους δεν θα σημαίνει πως είναι έμπιστοι και μη κακόβουλοι.



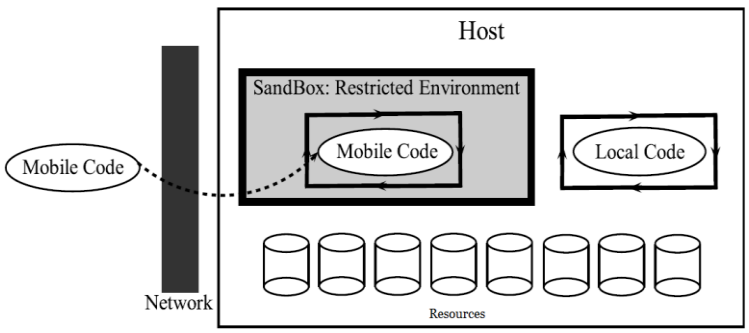
Εικόνα 4.4: Διαδικασία Υπογραφής Κώδικα

ii. ***Ιστορικό διαδρομών (Path History):*** Ο στόχος αυτού του μηχανισμού είναι να καταγράφει τις διαδρομές ενός μετακινούμενου πράκτορα μέσα στο σύστημα. Αυτό είναι βασικό, γιατί βάσει πιθανοτήτων, όσο περισσότερους κόμβους επισκέπτεται ένας πράκτορας τόσο περισσότερες πιθανότητες έχει να βρεθεί σε κάποιον κακόβουλο ο οποίος θα αλλοιώσει τη λειτουργία του. Προκειμένου να δημιουργηθεί ένα ιστορικό διαδρομών για κάθε μετακινούμενο πράκτορα, ο εκάστοτε κόμβος υπογράφει ότι ο πράκτορας πέρασε από αυτόν, καθώς και τον επόμενο κόμβο που πρέπει να επισκεφτεί κανονικά (αν ακολουθεί προκαθορισμένη διαδρομή). Σύμφωνα με αυτές τις πληροφορίες, ο κάθε κόμβος θα μπορεί να αποφασίσει αν θα εκτελέσει τον κώδικα του πράκτορα ή όχι. Το πρόβλημα με αυτό τον μηχανισμό είναι ότι το ιστορικό διαδρομών μετά από πολλές μετακινήσεις μεγαλώνει υπερβολικά, γεγονός που δεν το κάνει αποτελεσματικό από άποψης διαχείρισης αποθηκευτικού χώρου της μνήμης του πράκτορα.

2. Μηχανισμοί Πρόληψης:

i. ***Απομόνωση Εκτέλεσης (Sandboxing):*** Τα sandboxes είναι ένα ραγδαία αυξανόμενο και σημαντικό δομικό υλικό για την αύξηση ασφαλείας των συστημάτων λογισμικού **[29].** Με τον ορό sandboxing εννοούμε την απομόνωση της εκτέλεσης ενός προγράμματος με στόχο να μην μπορεί να έχει απευθείας πρόσβαση σε κρίσιμους πόρους ενός συστήματος. Ουσιαστικά, με το sandboxing δημιουργούμε ένα ενδιάμεσο στρώμα ασφαλείας ανάμεσα σε μια εφαρμογή και στο περιβάλλον εκτέλεσής της. Αυτό είναι επιθυμητό σε περιπτώσεις όπου κάποια κακόβουλη εφαρμογή προσπαθεί να αποκτήσει πρόσβαση σε πόρους του συστήματος για τους οποίους δεν έχει εξουσιοδότηση. Με αυτόν τον τρόπο, εκτελώντας δηλαδή ύποπτες εφαρμογές μέσα σε ένα sandbox, αποτρέπουμε τη διάδοση του κακόβουλου λογισμικού στο σύστημά μας.

Σ' ένα σύστημα μετακινούμενων πρακτόρων ο μηχανισμός αυτός χρησιμοποιείται για την προστασία των κόμβων από κακόβουλους μετακινούμενους πράκτορες. Κάθε κόμβος δεν εκτελεί απευθείας τον κώδικα του εκάστοτε πράκτορα, αλλά μέσα σε ένα sandbox. Με αυτό τον τρόπο απομονώνονται πιθανόν κακόβουλες ενέργειες του πράκτορα από το περιβάλλον εκτέλεσης του κόμβου (λειτουργικό σύστημα), όπως πρόσβαση σε αρχεία, εκτέλεση διεργασιών και άνοιγμα πορτών (ports) και sockets. Ακολουθεί μια σχηματική αναπαράσταση του sandboxing στην *(Εικόνα 4.5)*



Εικόνα 4.5: Αναπαράσταση Sandboxing

ΚΕΦΑΛΑΙΟ 5

ΕΝΙΣΧΥΣΗ ΑΣΦΑΛΕΙΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΚΙΝΟΥΜΕΝΩΝ ΠΡΑΚΤΟΡΩΝ

5.1 – Πρόσθετο Ασφάλειας JADE (JADE-S v.2)

5.1.1 – Εισαγωγή στο JADE-S v.2

Όπως είδαμε, η υλοποίηση λειτουργιών ασφαλείας σε ένα σύστημα μετακινούμενων πρακτόρων είναι ύψιστης σημασίας έτσι ώστε να μπορεί να χρησιμοποιηθεί σε πρακτικές εφαρμογές. Ειδικότερα σε ένα ευρύ περιβάλλον όπως το internet, η επικοινωνία μπορεί να παραβιαστεί και η ταυτότητα των πρακτόρων να αλλοιωθεί.

Μια λύση έρχεται να δώσει το JADE-S v.2, ο βελτιωμένος απόγονος του αρχικού JADE-S v.1. Ουσιαστικά, αυτό που προσφέρει το JADE-S v.2 είναι κάποια επιπρόσθετα χαρακτηριστικά ασφάλειας έτσι ώστε να μπορούν τα συστήματα πρακτόρων να βρουν εφαρμογή σε πραγματικές συνθήκες λειτουργίας. Βασίζεται στο μοντέλο ασφάλειας της java (java security model) και προσφέρει τα εξής χαρακτηριστικά:

- **JAAS (Java Authentication-Authorization Service):** Επιτρέπει την πρόσβαση για την τέλεση διαφόρων ενεργειών σε ένα σύνολο κλάσεων, βιβλιοθηκών και αντικειμένων (owner username, password).

- **JCE (Java Cryptography Extension):** Περιλαμβάνει ένα σύνολο μεθόδων που χρησιμοποιούνται για τη δημιουργία κλειδιών κρυπτογράφησης,

- **JSSE (Java Secure Socket Extension):** Επιτρέπει τη δημιουργία ασφαλών διαύλων μεταφοράς δεδομένων πάνω σε ένα δίκτυο όπως για παράδειγμα το SSL (Secure Sockets Layer).

«Το JADE-S v.2 δομεί την πλατφόρμα των μετακινούμενων πρακτόρων ως ένα περιβάλλον πολλών χρηστών του οποίου τα δομικά στοιχεία (πράκτορες, containers) ανήκουν σε χρήστες οι οποίοι έχουν αυθεντικοποιηθεί (μέσω username, password) και εξουσιοδοτηθεί από τον διαχειριστή του συστήματος να επιτελούν κρίσιμες ενέργειες» **[30]**.

Βασικό στοιχείο της κάθε πλατφόρμας αποτελεί το αρχείο αδειών (permissions file) το οποίο περιέχει καταγραφές των ενεργειών που δικαιούται να εκτελέσει κάποιος χρήστης του συστήματος.

5.1.2 – Χαρακτηριστικά Ασφάλειας του JADE-S v.2

Τα χαρακτηριστικά ασφάλειας που προσφέρει το JADE-S v.2 σε ένα σύστημα μετακινούμενων πρακτόρων είναι τα ακόλουθα:

- **Αυθεντικοποίηση (Authentication):** Για να μπορεί ένας χρήστης να έχει πρόσβαση σε κάποια οντότητα της πλατφόρμας θα πρέπει πρώτα να εισάγει κάποιον έγκυρο συνδυασμό (username – password). Βέβαια, για να προστατέψουμε την πλατφόρμα μας, θα πρέπει οι τα username και τα password να είναι ανθεκτικά σε επιθέσεις τύπου Brute Force, οπότε θα πρέπει να επιλέγεται σχετικά μακροσκελής κωδικός, να περιλαμβάνει διάφορα σύμβολα (π.χ. #%$&%(^) και η φόρμα login να έχει εκθετικά αυξανόμενα timeout για επανειλημμένες αποτυχημένες προσπάθειες εισόδου. Επιπλέον, θα μπορούσε να προστεθεί με χρήση βιομετρικών επιβεβαιώσεων ένα επιπρόσθετο επίπεδο ασφάλειας κάνοντας το σύστημα two-factor authenticated (2FA). Εφόσον κάποιος χρήστης περάσει επιτυχώς τη διαδικασία εισόδου, μπορεί να έχει πρόσβαση σε AMS, DF, RMA, containers και μετακινούμενους πράκτορες.

- **Πολιτική (Policy):** Η κάθε πλατφόρμα διέπεται από κανόνες λειτουργίας οι οποίοι έχουν οριστεί από τον διαχειριστή. Αυτοί οι κανόνες αφορούν τη δικαιοδοσία που έχουν οι χρήστες, οι πράκτορες και τα containers της πλατφόρμας ως προς την εκτέλεση διαφόρων ενεργειών.

Το αρχείο policy ακολουθεί τη σύνταξη του JAVA/JAAS και ορίζει τα πεδία **[31]**:

i. ***Platform Permission*:** Δικαιοδοσία για δημιουργία/διαγραφή ενός Main Container.
ii. ***Container Permission*:** Δικαιοδοσία για δημιουργία/διαγραφή ενός Agent Container
iii. ***Agent Permission*:** Δικαιοδοσία για δημιουργία/διαγραφή πρακτόρων σε ένα Container.
iv. ***AMS Permission*:** Δικαιοδοσία για εγγραφή/απεγγραφή ενός πράκτορα από το AMS.
v. ***Message Permission*:** Δικαιοδοσία για αποστολή μηνυμάτων σε πράκτορες.

- **Εξουσιοδότηση (Authorization):** Το JADE-S v.2 χρησιμοποιεί την αφαίρεση 'principal' για να δηλώσει οποιοδήποτε στοιχείο μιας πλατφόρμας που είναι είτε λογαριασμός χρήστη, είτε ένας πράκτορας, είτε ένα container. Ένας 'principal' λοιπόν, πρέπει να εξουσιοδοτηθεί από το Java Security Manager, το οποίο λαμβάνει την απόφαση για το αν θα επιτρέψει την ενέργεια στον 'principal' με βάση την πολιτική της πλατφόρμας (policy).

- **Πιστοποιητικό - Αρχή Πιστοποίησης (Certificate – Certification Authority):** Η αρχή πιστοποίησης αναλαμβάνει τον ρόλο δημιουργίας πιστοποιητικών για όλη την πλατφόρμα, χρησιμοποιώντας ένα συνδυασμό public-private key (PKI infrastructure).

- **Αντιπροσωπεία (Delegation):** Σε μια πλατφόρμα, δίνεται η δυνατότητα να μπορούν οι πράκτορες εκτός από το προσωπικό τους πιστοποιητικό που τους δίνεται από την αρχή πιστοποίησης, να ανταλλάσσουν πιστοποιητικά μεταξύ τους με πεπερασμένο χρονικό διάστημα ισχύος τους.

- **Ασφαλής Επικοινωνία (Secure Communication):** Η επικοινωνία μεταξύ πρακτόρων οι οποίοι είναι σε διαφορετικούς κόμβους μπορεί να γίνεται μέσω του πρωτοκόλλου SSL. Το SSL είναι ένα Network Layer πρωτόκολλο το οποίο συμβάλλει στην ασφαλή επικοινωνία μεταξύ δυο οντοτήτων πάνω από ένα δίκτυο. Με αυτό τον τρόπο αποφεύγονται υποκλοπές πακέτων (packet sniffing) **[32]**. Οι κύριες ενέργειες του πρωτοκόλλου είναι οι εξής:
  a. Αυθεντικοποίηση του server από τον client

b. Αυθεντικοποίηση του client απο τον server

c. Δημιουργία ασφαλούς κρυπτογραφημένου διαύλου επικοινωνίας μεταξύ των δύο αυτών πλευρών

Οι αλγόριθμοι κρυπτογράφησης που υποστηρίζονται από το πρωτόκολλο είναι: DES - Data Encryption Standard, DSA - Digital Signature Algorithm, KEA - Key Exchange Algorithm, MD5 - Message Digest, RC2/RC4, RSA κ.α.

Το JADE-S λοιπόν, χρησιμοποιεί το παραπάνω java security model για να προσφέρει στο πολυπρακτορικό σύστημα τα βασικά χαρακτηριστικά ασφάλειας **[33]**:

- Αυθεντικοποίηση (Authentication) των χρηστών.
- Εξουσιοδότηση (Authorization) των ενεργειών που πραγματοποιούνται από τους principal.
- Υπογραφή μηνυμάτων (Message signature) για τη διασφάλιση ακεραιότητας των δεδομένων που στέλνονται.
- Απόκρυψη (Encryption) του περιεχομένου των μηνυμάτων για την αποφυγή υποκλοπών.

Εκτός από την υπηρεσία αυθεντικοποίησης η οποία αποτελεί τη βάση του JADE-S και εκτελείται κατά την εκκίνησή του, οι υπόλοιπες υπηρεσίες μπορούν να εκκινηθούν ανεξάρτητα μεταξύ τους και κατ' επιλογήν.

5.1.3 – Αρχιτεκτονική λειτουργίας του JADE-S v.2

Σε μια πλατφόρμα υλοποιημένη σε JADE (όπως και αυτή της υλοποίησης στο κεφάλαιο 2), οι containers είναι κατανεμημένοι σε κόμβους της πλατφόρμας. Για να μπορέσουμε να έχουμε ένα ικανοποιητικό επίπεδο ασφάλειας σε μια τέτοια πλατφόρμα, η οποία μπορεί να αριθμεί πολλούς κόμβους και διαδικτυακούς συνδέσμους, το JADE-S ορίζει την ιδέα της πλατφόρμας πολλαπλών χρηστών (multiuser platform). Κάθε container και κάθε πράκτορας της πλατφόρμας ανήκει σε αυθεντικοποιημένους χρήστες οι οποίοι, προτού να μπορούν να πραγματοποιήσουν κάποια ενέργεια εντός της πλατφόρμας (π.χ. δημιουργία/διαγραφή πράκτορα), πρέπει να λάβουν εξουσιοδότηση από τον/τους διαχειριστές της πλατφόρμας.

Οι διαχειριστές είναι αυτοί που παραχωρούν εξουσιοδότηση στους χρήστες για να μπορούν να επιτελούν συγκεκριμένες ενέργειες.

Το Main-Container, το οποίο δημιουργείται από τον διαχειριστή της πλατφόρμας, κατέχει τοπικά ή σε έναν remote server ένα αρχείο καταλόγου με στοιχεία σύνδεσης σε αυτό. Επίσης κατέχει ένα καθολικό αρχείο πολιτικής (global policy file) το οποίο ορίζει τους κανόνες λειτουργίας όλης της πλατφόρμας. Τα υπόλοιπα containers της πλατφόρμας περιέχουν ένα τοπικό αρχείο πολιτικής (local policy file) που ορίζει τους κανόνες λειτουργίας του container.

Τέλος, κάθε πράκτορας της πλατφόρμας φέρει ένα ζεύγος public – private key και ένα πιστοποιητικό της ταυτότητάς του. Αυτό περιέχει το όνομα του ιδιοκτήτη του, το όνομα που φέρει ως principal και υπογράφεται από τον πράκτορα AMS.

5.1.4 – Προσέγγιση των απαιτήσεων ασφαλείας με JADE-S

1. **Αυθεντικοποίηση (Authentication):** Είναι η βάση του JADE-S. Όπως αναλύσαμε και παραπάνω, κάθε οντότητα της πλατφόρμας ανήκει σε έναν αυθεντικοποιημένο χρήστη. Βασίζεται στο JAAS API και αποτελείται από δύο στοιχεία.
   i. ***Callback Handler*:** Είναι το γνωστό και ευρέως διαδεδομένο πλαίσιο, στο οποίο οι χρήστες εισάγουν τα στοιχεία σύνδεσής τους (username, password).
   ii. ***Login Module:*** Ελέγχει τα στοιχεία που δοθήκαν, αν είναι αληθή σύμφωνα με το αρχείο καταλόγου στοιχείων σύνδεσης.

   Αρχικά, ο διαχειριστής της πλατφόρμας δημιουργεί το Main-Container. Στη συνέχεια κάθε χρήστης που επιθυμεί να αποκτήσει πρόσβαση σε ένα από τα υπόλοιπα Containers, εισάγει τα στοιχεία μέσω του callback handler τοπικά στο Container αυτό, το οποίο στη συνέχεια τα στέλνει στο Main-Container. Το Main-Container με τη σειρά του χρησιμοποιώντας το login module ελέγχει αν τα στοιχεία είναι έγκυρα. Αν η διαδικασία είναι επιτυχής, ο χρήστης έχει πρόσβαση στην πλατφόρμα για επιτέλεση σαφώς ορισμένων ενεργειών.
2. **Εξουσιοδότηση (Authorization):** Οι σαφώς ορισμένες αυτές ενέργειες που δικαιούνται να πραγματοποιήσουν οι principals, ορίζονται από ένα σύνολο κανόνων (Access Control List). Οι κανόνες αυτοί είναι αποθηκευμένοι στο αρχείο policy όπως είδαμε και

παραπάνω. Η εξουσιοδότηση που δίνεται σε χρήστες, για διάφορες ενέργειες εντός της πλατφόρμας, αυτόματα παραχωρείται και στους πράκτορές του.

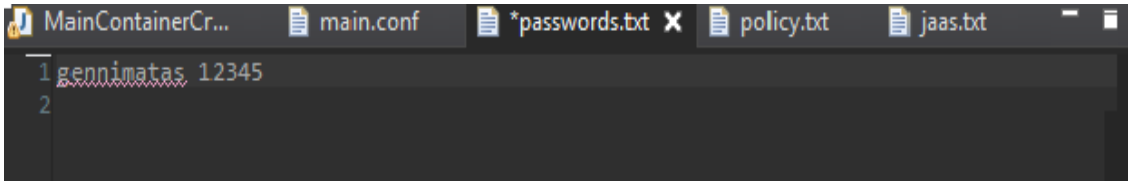
3. **Υπογραφή/Απόκρυψη (Signature/Encryption):** Οι παρακάτω απαιτήσεις ασφαλείας, όπως είδαμε, αφορούν την επικοινωνία μεταξύ των οντοτήτων μιας πλατφόρμας. Κατά την αποστολή ενός ACL μηνύματος, η υπογραφή ενός μηνύματος εγγυάται την ακεραιότητα των δεδομένων και την μη αποκήρυξη της ανταλλαγής του μηνύματος. Η απόκρυψη από την άλλη εγγυάται την ιδιωτικότητα των δεδομένων. Στο JADE-S v.2 η υπογραφή και η απόκρυψη υλοποιούνται με χρήση ενός ζεύγους public – private κλειδιού που φέρει ο κάθε πράκτορας. Οι πράκτορες μπορούν να υπογράψουν ή να αποκρύψουν το περιεχόμενο ενός μηνύματος κάνοντας χρήση μεθόδων της κλάσης *SecurityHelper*, setUseSignature() και setUseEncryption() αντίστοιχα. Σε περίπτωση που ο παραλήπτης πράκτορας αδυνατεί να επιβεβαιώσει (verify) ή να φανερώσει (decrypt) ένα μήνυμα, τότε το απορρίπτει.

5.1.5 – Υλοποίηση αυθεντικοποίησης και εξουσιοδότησης σε JADE

Όπως είδαμε η διαδικασία της αυθεντικοποίησης ενός χρήστη είναι η σημαντικότερη προσθήκη ασφάλειας του JADE-S. Στη συνέχεια θα εξετάσουμε πώς μπορούμε να κατασκευάσουμε ένα Main-Container για την πλατφόρμα ορίζοντας τα αρχεία passwords.txt, jaas.conf και main.conf.

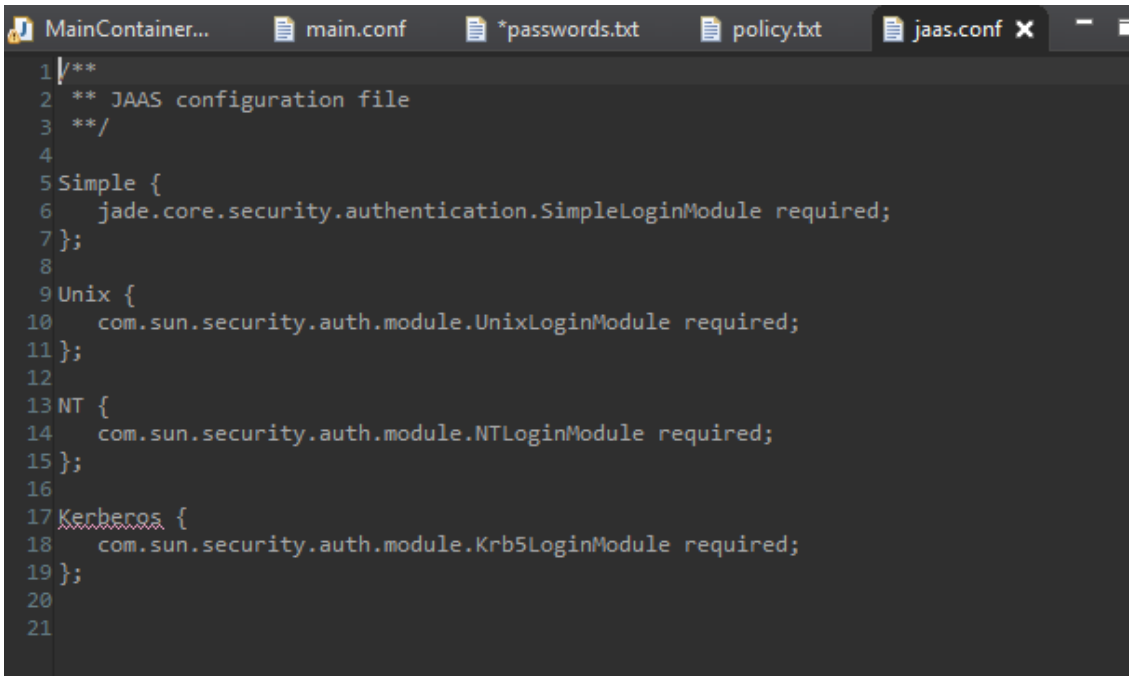
Για να μπορέσουμε να χρησιμοποιήσουμε το JADE-S πρέπει να κατεβάσουμε από τη σελίδα του JADE το security plugin και να εισάγουμε στο build path το jadeSecurity.jar

i. Αρχικά στο αρχείο passwords.txt εισάγουμε τους χρήστες με τους κωδικούς πρόσβασής τους *(Εικόνα 5.1)*.



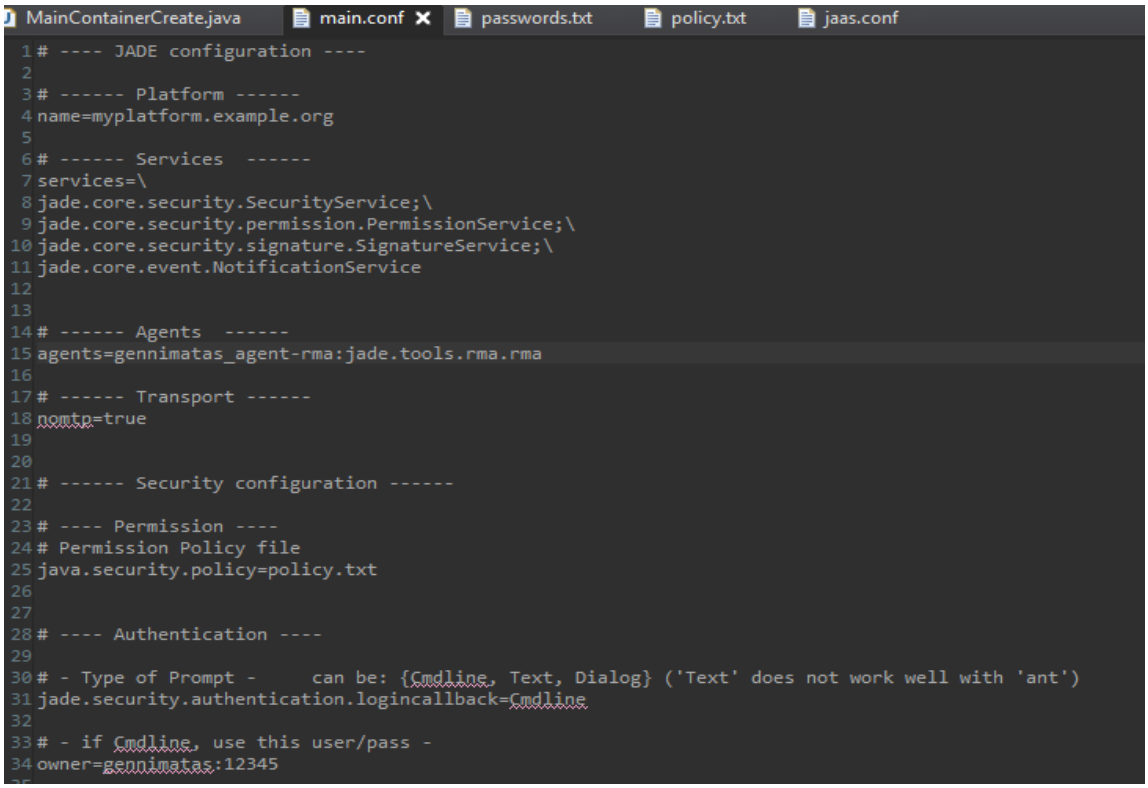
Εικόνα 5.1: Αρχείο paswords.txt

ii. Στο jaas.conf επιλέγουμε τους τρόπους αυθεντικοποίησης *(Εικόνα 5.2).*



Εικόνα 5.2: Επιλογή τρόπων αυθεντικοποίησης

iii. Ορίζουμε το main.conf στο οποίο εισάγουμε τα ορίσματα (arguments) της εκτέλεσης *(Εικόνα 5.3).*

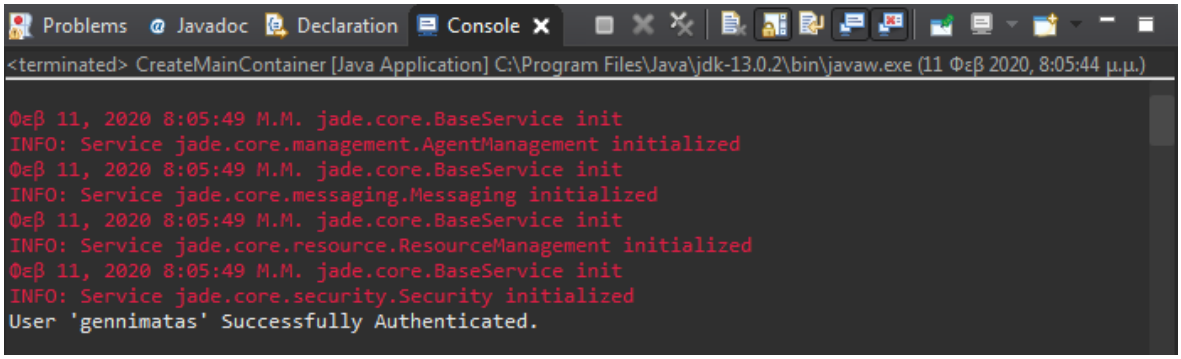


Εικόνα 5.3: Ορισμός του main.conf

iv. Ξεκινάμε το container ως χρήστης με το όνομα χρήστη "gennimatas" χρησιμοποιώντας το main.conf *(Εικόνα 5.4)*.

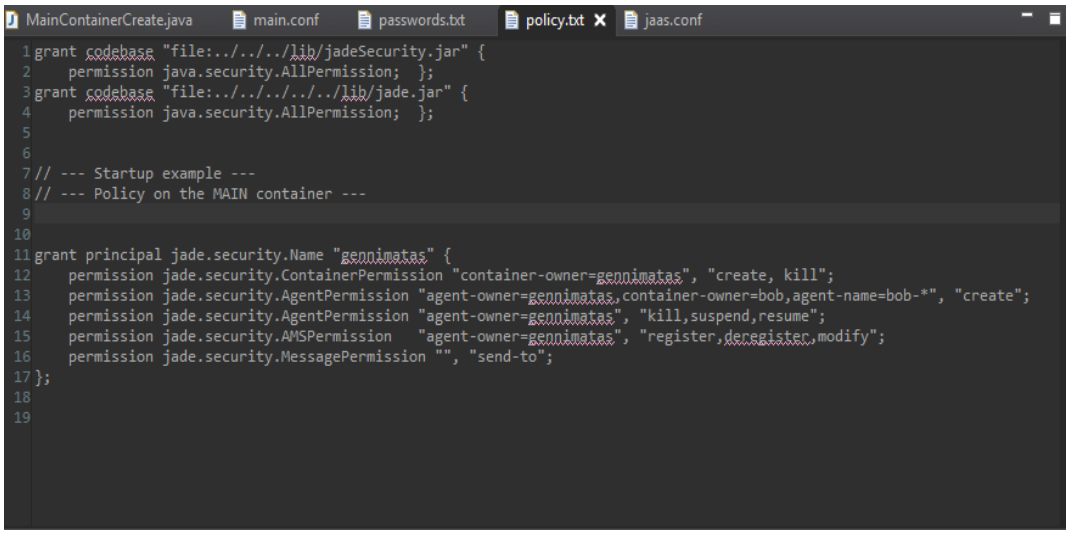
Εικόνα 5.4: Εκκίνηση Container.

v. Παρατηρούμε ότι η αυθεντικοποίηση ήταν επιτυχής *(Εικόνα 5.5)*



Εικόνα 5.5: Αυθεντικοποίηση επιτυχής

vi. Έχοντας αυθεντικοποιήσει πλέον τον χρήστη "gennimatas" μπορούμε να ελέγξουμε τη δικαιοδοσία του στο Main-Container μέσω του policy.txt *(Εικόνα 5.6)*



Εικόνα 5.6: Δικαιοδοσία του αυθεντικοποιημένου χρήστη

5.2 – Πρόσθετο Ασφάλειας IMTPoverSSL

Εκτός από το JADE-S, ένα άλλο πρότυπο ασφάλειας είναι το IMTPoverSSL. Το πρότυπο αυτό μας παρέχει έναν μηχανισμό ο οποίος εξασφαλίζει την ακεραιότητα, την ιδιωτικότητα και την αμοιβαία αυθεντικοποιημένη επικοινωνία μεταξύ των containers της πλατφόρμας, μέσω του πρωτοκόλλου ανταλλαγής μηνυμάτων IMTP (Internal Message Transport Protocol) πάνω σε ασφαλή σύνδεση SSL. Η φιλοσοφία διαφέρει από αυτή του JADE-S v.2 καθώς η επικοινωνία γίνεται μεταξύ των containers και όχι μεταξύ των πρακτόρων.

Κάθε container έχει ένα πιστοποιητικό και έναν τοπικό κατάλογο με τα containers τα οποία εμπιστεύεται (trusted containers list). Έτσι, κάθε φορά που ένα container επιθυμεί να επικοινωνήσει με ένα άλλο, αποστέλλει το πιστοποιητικό του και αν αυτό υπάρχει στη λίστα του παραλήπτη με τα έμπιστα containers, τότε δέχεται τη σύνδεση η οποία κρυπτογραφείται μέσω του πρωτοκόλλου SSL.

5.3 – Διαφορές JADE-S v.2 και IMTPoverSSL

1. IMTPoverSSL χρησιμοποιείται για την ασφαλή επικοινωνία μεταξύ των containers και δεν παρέχει μηχανισμό αυθεντικοποίησης των χρηστών όπως το JADE-S v.2, γεγονός που σε πολλά πραγματικά συστήματα το καθιστά υποδεέστερο, από την άποψη ότι η αυθεντικοποίηση των χρηστών είναι υψίστης σημασίας.

2. Η εφαρμογή της ασφάλειας μέσω του IMTPoverSSL γίνεται σε επίπεδο containers μόνο και όχι σε επίπεδο πρακτόρων. Αυτό αποτρέπει τους πράκτορες να ορίζουν το επίπεδο ασφάλειας που επιθυμούν.

3. Το γεγονός ότι, κάθε container πρέπει να έχει αποθηκευμένα όλα τα υπόλοιπα πιστοποιητικά από τα container που εμπιστεύεται, δημιουργεί περιορισμούς ως προς την κλιμακωσιμότητα (scalability). Σε πλατφόρμες με πολύ μεγάλο αριθμό από containers είναι μη αποδοτικό να ξοδεύεται τόσο μεγάλη ποσότητα μνήμης μόνο γι αυτόν το λόγο. Αυτό βέβαια, θα μπορούσε να επιλυθεί με την προσθήκη ενός server που θα κρατάει μια λίστα για το κάθε container, η οποία θα αναφέρει τα έμπιστα προς αυτό containers.

Γίνεται σαφές λοιπόν, πως το JADE-S v.2 παρέχει ένα πιο ολοκληρωμένο πλαίσιο προστασίας μιας πλατφόρμας πρακτόρων αλλά η επιλογή ενός εκ' των δύο βασίζεται κυρίως στις συγκεκριμένες ανάγκες ασφάλειας της κάθε περίπτωσης.

ΚΕΦΑΛΑΙΟ 6

ΕΠΙΛΟΓΟΣ

Τα κατανεμημένα συστήματα μετακινούμενων πρακτόρων είναι μια τεχνολογία που παρέχει τη δυνατότητα να έχουμε πραγματικά αποκεντρωμένα συστήματα, γεγονός που τα καθιστά έναν τομέα της πληροφορικής με τεράστιες δυνατότητες εξέλιξης. Ειδικότερα, ζούμε σε μια εποχή κατά την οποία το πεδίο της τεχνητής νοημοσύνης γνωρίζει άνθιση. Αυτή η εξέλιξη θα μπορούσε να χρησιμοποιηθεί και σε αυτή την περίπτωση, βελτιώνοντας τη δυνατότητα εξέλιξης των μετακινούμενων πρακτόρων δυναμικής πορείας (dynamic itinerary) κάνοντάς τους περισσότερο «ευφυείς».

Επίσης, γίνεται αντιληπτό πως οι κίνδυνοι που διατρέχουν τέτοια συστήματα είναι πολυάριθμοι και σε πολλές περιπτώσεις δύσκολα αντιμετωπίσιμοι. Με τα κατάλληλα πλαίσια ασφαλείας όμως μπορούμε να παρέχουμε ένα αρκετά ικανοποιητικό επίπεδο ασφάλειας με περιθώρια περεταίρω βελτίωσης.

Η συμβολή αυτής της διπλωματικής έγκειται στην ολοκληρωμένη παρουσίαση των συστημάτων των μετακινούμενων πρακτόρων, όσον αφορά την υλοποίησή τους σε κώδικα και τη θεωρητική ανάλυση των κινδύνων της ασφαλείας τους.

Τέλος, θα ήθελα να αναφερθώ στη δυσκολία της συγγραφής αυτής της διπλωματικής η οποία έγινε φανερή κυρίως κατά τη διάρκεια υλοποίησης του project σε jade, καθώς δεν υπάρχει αρκετό ευκολονόητο υλικό στο οποίο μπορεί να βασιστεί κάποιος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] B. M. Amro, "Mobile Agent Systems, Recent Security Threats and Counter Measures," *IJCSI International Journal of Computer Science Issues,* vol. 11, no. 2, pp. 145-151, 2014.

[2] F. Alfonso, P. G. P και G. Vigna, «Understanding Code Mobility,» *IEEE Transactions on Software Engineering* , τόμ. 24, αρ. 5 , pp. 352 - 354, 1998.

[3] M. Rouse, "searchdatacenter.techtarget.com," May 2007. [Online]. Available: https://searchdatacenter.techtarget.com/definition/Remote-Job-Entry. [Accessed 8 2 2020].

[4] S. Russel και P. Norvig, Artificial Intelligence: A Modern Approach, Upper Saddle River, New Jersey : Prentice Hall, 2009.

[5] M. Mittal και B. Tharun, «MOBILE AGENT,» *International Journal of Engineering Research & Technology (IJERT),* τόμ. 1, αρ. 8, pp. 1- 6, 2012.

[6] K. M. Μελισσίδης, «Κινητοί Πράκτορες με JADE».

[7] C. G. Harrison, D. M. Chess και A. Kershenbaum, «Mobile Agents: Are they a good idea ?,» IBM Research Division , 1995.

[8] D. B. Lange, «Mobile objects and mobile agents: The future of distributed,» *In Proceedings of the European conference on,* p. 1–12, 1998.

[9] D. B. Lange και M. Oshima, «Seven Good Reasons for Mobile Agents.,» *Communications of the ACM,* τόμ. 3, αρ. 42, pp. 88-89, 1999.

[10] S. Padney, «slideshare.net,» 17 12 2014. [Ηλεκτρονικό]. Available: https://www.slideshare.net/SantoshPandey29/mobile-agents-42789511. [Πρόσβαση 9 2 2020].

[11] F. Bellifemine, G. Caire και D. Greenwood, developing multi-agent systems with JADE, West Sussex: John Wiley & Sons Ltd, 2007.

[12] «jade.tilab.com,» Telecom Italia Lab, February 2000. [Ηλεκτρονικό]. Available: https://jade.tilab.com/. [Πρόσβαση 9 February 2020].

[13] H. Idrissi, «Contributions to the security of mobile agent systems,» Faculty of Sciences, Mohammed-V University in Rabat, Rabat, 2016.

[14] M. Mahmoodi και M. M. Varnamkhasti, «A Secure Communication in Mobile Agent System,» *International Journal of Engineering Trends and Technology (IJETT),* τόμ. 6, αρ. 4, pp. 186-187, 2013.

[15] P. Dadhich, K. Dutta και M. C. Govil, «Security Issues in Mobile Agents,» *International Journal of Computer Applications,* τόμ. 11, αρ. 4, pp. 1-5, 2010.

[16] M. Alfalayeh και L. Brankovic, «An Overview of Security Issues and Techniques in Mobile Agents,» University of Newcastle, Newcastle, Australia, 2005.

[17] E. C. Vigil, «Security Issues in Mobile,» Indian Institute of Technology, Mumbai.

[18] V. Sharma και P. Ahuja, «A Review on Mobile Agent Security,» *International Journal of Recent Technology and Engineering (IJRTE),* τόμ. 1, αρ. 2, pp. 84- 85 , 2012.

[19] A. W. Jansen, «Countermeasures for Mobile Agent Security,» National Institute of Standards and Technology, Gaithersburg.

[20] H. K. Tan, L. Moreau, D. Cruickshank και D. De Roure, «Certificates for Mobile Code Security,» σε *17th ACM Symposium on Applied Computing*, Marrakech, 2002.

[21] V. Roth, «Mutual protection of cooperating agents,» σε *Secure Internet Programming: Security issues for Mobile and Distributed Objects*, Verlag, Springer, 1999.

[22] A. J. Menezes, V. Oorschot και V. S. A, Handbook of Applied Cryptography, Florida : CRC Press , 2001.

[23] W. A. Jansen, «Countermeasures for mobile agent security,» *Computer Communications,* τόμ. 23, αρ. 17, pp. 1667-1676, 200.

[24] C. Collberg, C. Thomborson και D. Low, «Technical Report #148,» The University of Auckland, Auckland.

[25] «java-source.net,» Oracle, [Ηλεκτρονικό]. Available: https://java-source.net/open-source/obfuscators/jode. [Πρόσβαση 10 February 2020].

[26] L. D'Anna, B. Matt, A. Reisse, T. Van Vleck, S. Schwab και P. Leblanc, «Self-Protecting Mobile Agents Obfuscation Report,» Network Associates Laboratorie, 2003.

[27] T. Sander και C. Tshudin, «Protecting mobile Agents Against Malicious Hosts,» σε *International Computer Science Institute,* Berkeley, 1998.

[28] M. P. Omer, «Environmental Key Generation,» σε *A Security Framework for Mobile Agent Systems*, Boca Raton, Universal Publishers , 2006, p. 26.

[29] M. Maas, A. Sales, B. Chung και J. Sunshine, «A systematic analysis of the science of sandboxing,» *PeerJ Comput. Sci. 2:e43,* τόμ. 2, αρ. 3, pp. 1-36, 2016.

[30] R. Abassi, Artificial Intelligence and Security Challenges in Emerging Networks, Hersey: IGI Global, 2019.

[31] V. Snasel, E. El-Qawasmeh και J. Platos, Digital Information Processing and Communications, Part II, Ostrava: Springer , 2011.

[32] «techterms,» [Ηλεκτρονικό]. Available: https://techterms.com/definition/ssl. [Πρόσβαση 7 february 2020].

[33] X. Villa, A. Schuster και A. Riera, «Security for a Multi-Agent System based on JADE,» *computers & security* , αρ. 2 6, pp. 392-400, 2007.

[34] R. Ssekibuule, «Mobile Agent Security against Malicious Platforms,» *Cybernetics and Systems*, pp. 1-15, 2010.