Πίνακας Περιεχομένων

Εισαγωγή

Η Java είναι σήμερα το προιόν στο οποίο αναφέρονται σχεδόν καθημερινά, όλα τα έντυπα των Η/Υ. Οι δυνατότητες της, διαπλατφορμικότητα, αντικειμενοστρεφής και ευκολία στην χρήση και στην υλοποίηση την κάνουν ιδιαίτερα ελκυστική. Επειδή είναι προφανές ότι θα καταλάβει μεγάλο μερίδιο της αγοράς του Ιντερνέτ ένα σοβαρό ζήτημα που προκύπτει είναι η ασφάλεια που μπορεί να προσφέρει αυτό το προιόν σε ένα ιδιαίτερα ανασφαλές περιβάλλον. Εδώ ερευνούμε τα προβλήματα ασφάλειας που αφορούν την Java και ο τρόπος που αυτά αντιμετωπίζονται.

Introduction

Today Java is the hot product, in which computer magazines refers almost every issue. It's capabilities, cross-pltform, object oriented and the easy of use and implementation make it very attractive. Because it's apparent that in the near future it will control a big percentage of the Internet market, a very important security issue arises. How secure it can be in such an untrusted environment (Internet). Here we search the security problems of Java and the methods to handle them.

1. Τί είναι η Java;

1.1 Η πλατφόρμα ανάπτυξης εφαρμογών Java

Η Java είναι μια νέα επαναστατική πλατφόρμα εφαρμογών απ' την εταιρεία Sun Microsystems. Όπως και οι άλλες πλατφόρμες η Java δίνει την δυνατότητα για διαχείριση Ε/Ε, δικτύωσης, παραθύρων και γραφικών και άλλες διευκολύνσεις οι οποίες είναι απαραίτητες για την ανάπτυξη και την εκτέλεση σύγχρονων εφαρμογών. Η πλατφόρμα της Java έχει μια δυνατότητα την οποία δεν συναντούμε στις παραδοσιακές πλατφόρμες. Αυτή η δυνατότητα που ονομάζεται Write Once/Run Anywhere executables, δηλαδή Γράψε μια φορά/Εκτέλεσε παντού, επιτρέπει στα προγράμματα της Java τα οποία έχουν γραφτεί σε ένα συγκεκριμένο τύπο Η/Υ και λειτουργικού συστήματος να εκτελούνται ΧΩΡΙΣ ΤΗΝ ΠΑΡΑΜΙΚΡΗ ΜΕΤΑΒΟΛΗ σε σχεδόν οποιονδήποτε άλλο τύπο Η/Υ.

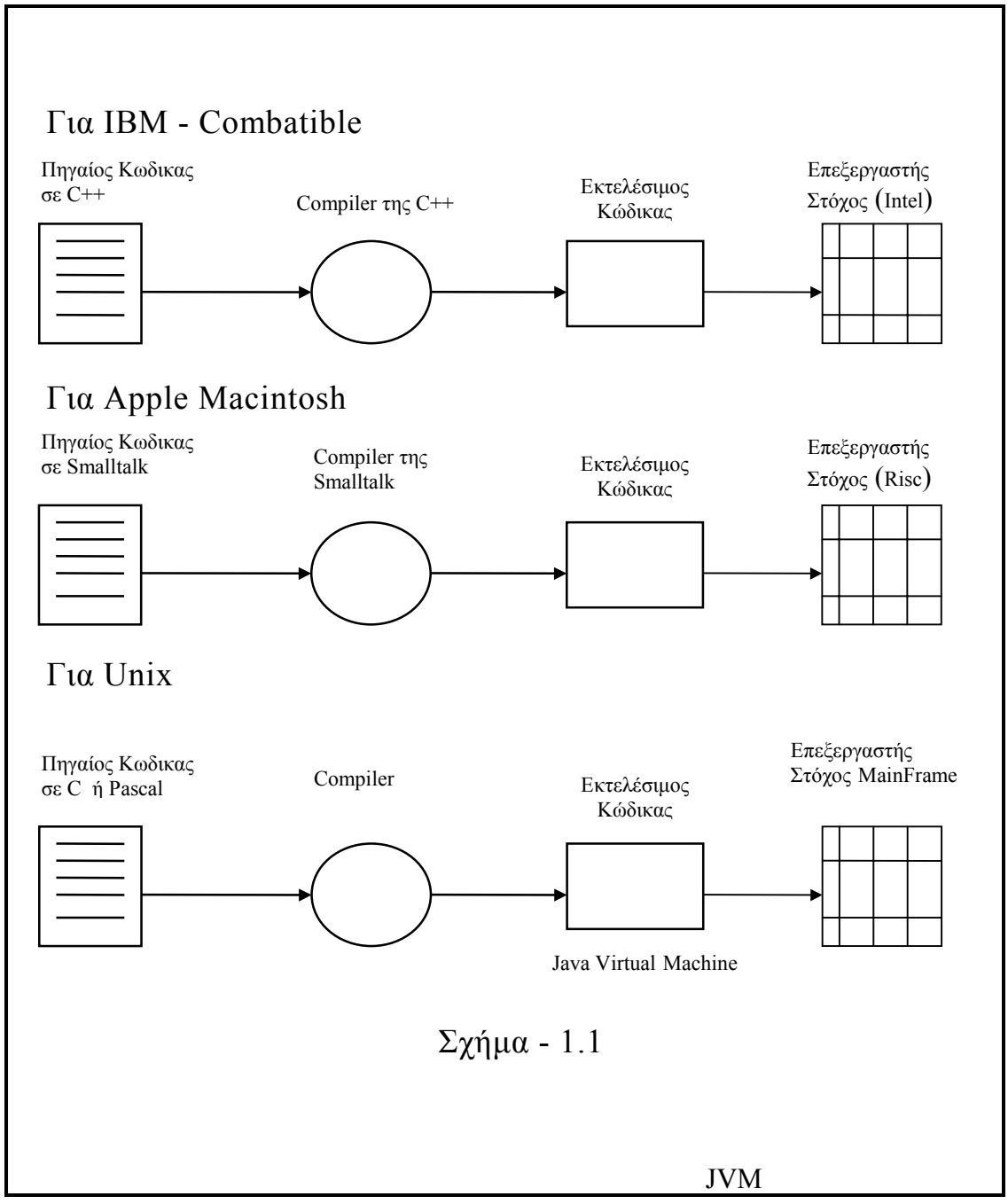
Όπως φαίνεται και στο Σχήμα-1.1 οι εφαρμογές που γράφονται για παραδοσιακά λειτουργικά συστήματα είναι στενά δεμένες στην συγκεκριμένη πλατφόρμα και δεν μπορούν με ευκολία να "μετακινηθούν" σε ένα άλλο τύπο Η/Υ ή λειτουργικού. Αυτό δεσμεύει τους προγραμματιστές σε συγκεκριμένη πλατφόρμα υλικού, λειτουργικού συστήματος. Για να μεταφερθεί η εφαρμογή σε άλλο

σύστημα απαιτείται ειδική διαδικασία (porting) που συχνά είναι ιδιαίτερα χρονοβόρα και έχει σαν συνέπεια να αφιερώνεται πολύς χρόνος και για την δημιουργία διαφόρων εκδόσεων αλλά και για την συντήρηση των εκδόσεων αυτών. Αυτός ο φόρτος διαχείρισης κάνει την ανάπτυξη εφαρμογών ειδικά για ετερογενή περιβάλλοντα δικτύου ασύμφορα.

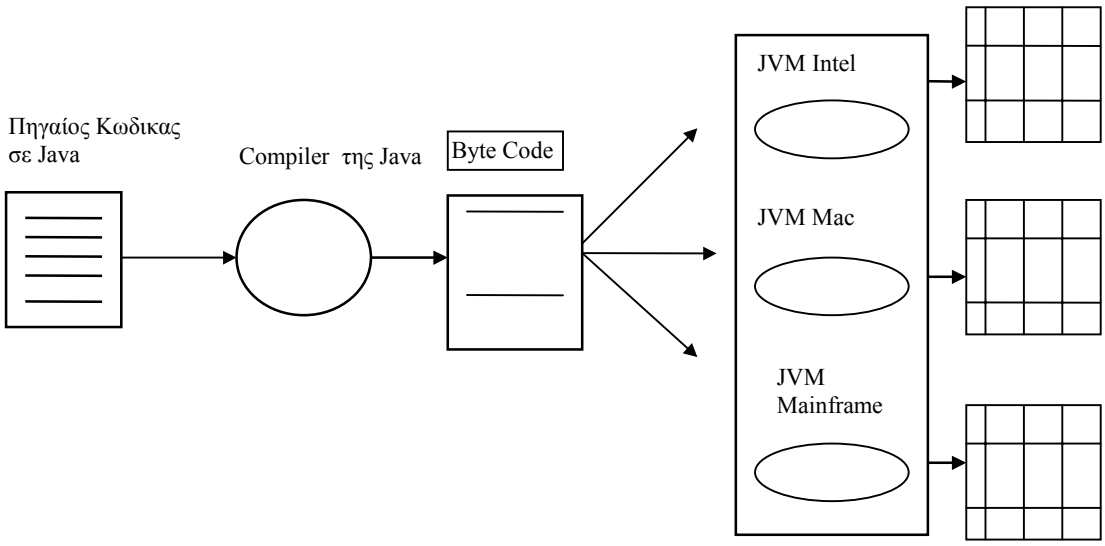
Με την δυνατότητα της η Java του Write Once/Run Anywhere δεν έχει τέτοια προβλήματα. Προγραμματιστές που δουλεύουν σε ένα μηχάνημα Sun Ultra το οποίο τρέχει το λειτουργικό σύστημα Solaris μπορούν να παράγουν εκτελέσιμο κώδικα που τρέχει σε PC, Macintosh και στους περισσότερους άλλους τύπους Η/Υ. Πως επιτυγχάνεται όμως αυτό το εντυπωσιακό αποτέλεσμα.

Όπως φαίνεται και στο Σχήμα-1.2 όλο το κόλπο είναι στην JVM (Java Virtual Machine, Εικονική μηχανή της Java) η οποία υλοποιείται στην κορυφή του λειτουργικού συστήματος του συγκεκριμένου μηχανήματος. Το πηγαίο πρόγραμμα της Java με την βοήθεια του compiler μετατρέπεται σε ψηφιοκώδικα (Bytecode) που ουσιαστικά είναι μια ψευτογλώσσα μηχανής αποτελούμενη από opcodes. Αυτός ο ψηφιοκώδικας είναι κοινός για όλους τους τύπους των μηχανημάτων. Στην συνέχεια ο ψηφιοκώδικας αυτός είναι η είσοδος της JVM όπου και παράγεται απ’ τον interpeter της JVM ο

τελικός εκτελέσιμος κώδικας που τρέχει πάνω στο συγκεκριμένο μηχάνημα με το συγκεκριμένο λειτουργικό. Άρα ένας πηγαίος κώδικας, ένας ψηφιοκώδικας, εικονικές μηχανές για κάθε πλατφόρμα, εκτελέσιμο για κάθε πλατφόρμα. Ήδη υπάρχουν εικονικές μηχανές για τις πιο δημοφιλείς πλατφόρμες και αναμένονται να δημιουργηθούν και για τις λιγότερο δημοφιλείς.



Σχήμα - 1.1



Σχήμα -1.2

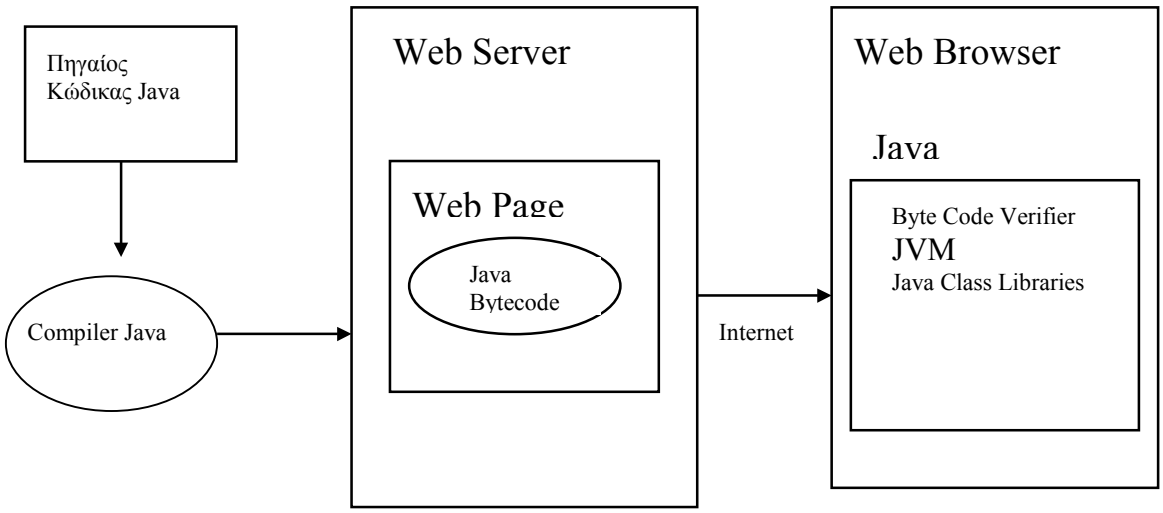
1.2 Τα applets της Java

Το applet είναι ένα μικρό εκτελέσιμο πρόγραμμα γραμμένο σε γλώσσα Java και ενσωματωμένο σε μια σελίδα υπερκειμένου. Η ανάγκη για τα applets που υλοποιούνται στην Java προέκυψε απ' το γεγονός ότι μέχρι τότε οι σελίδες υπερκειμένου ήταν ιδιαίτερα απλοικές και "επίπεδες" και δεν είχαν την δυνατότητα να εκμεταλλευτούν στο έπακρο τις δυνατότητες του Internet. Με τον ερχομό των applets μια σελίδα υπερκειμένου έχει την δυνατότητα να

περιέχει λογική, να είναι αλληλεπιδραστική και να αποτελεί το front end κομμάτι για μια επιχειρησιακή βάση δεδομένων. Επίσης το μοντέλο των applets της Java είναι αυτό του distributed computing όπου δεν έχουμε πια μεγάλα μονολιθικά προγράμματα πολλών Megabytes αλλά αντίθετα έναν πυρήνα που υλοποιείται με συστατικά (components) τα οποία είναι μικρά στο μέγεθος και διακινούνται εύκολα και γρήγορα στο δίκτυο. Ένα άλλο πλεονέκτημα αυτής της αρχιτεκτονικής είναι ότι έχουμε την δυνατότητα κλιμακούμενων (scalable) εφαρμογών αλλά και παράλληλης εργασίας για την δημιουργία συστατικών που λειτουργούν μαζί. Μια παρόμοια τεχνική που προωθείται απ' την Microsoft είναι τα συστατικά ActiveX τα οποία όμως δεν είναι διαπλατφορμικά.

Τα applets όπως είπαμε και πρίν είναι μικρά εκτελέσιμα προγράμματα γραμμένα σε Java τα οποία ενσωματώνονται σε μια σελίδα υπερκειμένου και εκτελούν μια συγκεκριμένη εργασία. Αυτή η εργασία ποικίλει απ' την εμφάνιση ενός σχεδίου το οποίο αλλάζει μορφή on-line μέχρι την διαχείριση βάσεων δεδομένων. Όταν φορτώνουμε στον browser μας μια σελίδα υπερκειμένου αυτόματα γίνεται download απ' τον αντίστοιχο server και το ή τα applets τα οποία περιέχονται σ' αυτήν την σελίδα και εκτελούνται τοπικά στον

Η/Υ μας. Τα παραπάνω φαίνονται παραστατικά στο σχήμα 1.3. Και μόνο αυτή η περιγραφή αρκεί για να δημιουργήσει και στον λιγότερο καχύποπτο αναγνώστη κάποιες απορίες.



Σχήμα 1.3

Από ποιον host προέρχεται αυτό το applet; Ποιός το έχει γράψει; Ποιός με βεβαιώνει ότι δεν έχει σφάλματα (bugs); Ποιός με βεβαιώνει ότι δεν θα κάνει μια εργασία στον Η/Υ μου την οποία εγώ δεν θέλω (αντιγραφή αρχείου, έναρξη σύνδεσης με κάποιον άλλο host, αποστολή κάποιου δικού μου αρχείου κάπου αλλού κ.λ.π.); Ποιός με βεβαιώνει ότι δεν κάνει κάποια ζημιά στον Η/Υ μου (διαγραφή αρχείου, φορμάρισμα του δίσκου μου, εξάντληση των πόρων του συστήματος μου); Όλες αυτές οι απορίες είναι όχι μόνο λογικές αλλά και επιβεβλημένες. Γιατί όσοι έχουν γράψει ακόμη και

ένα μικρό πρόγραμμα γνωρίζουν τι ζημιά επίτηδες ή άθελα μπορεί να δημιουργήσει αυτό το μικρό δημιουργημά.

2. Τι εννοούμε όταν λέμε Security;

2.1 Ορισμός της ασφάλειας

Ας δούμε όμως εμείς γενικά με τον όρο ασφάλεια τι πρέπει να συμπεριλαμβάνουμε και τι να αφήνουμε απ' έξω. Τι σημαίνει τελικά αυτή η περίφημη λέξη "ασφάλεια" τουλάχιστον στον χώρο των Η/Υ και των δικτύων. Σύμφωνα λοιπόν με έναν ορισμό, ασφάλεια είναι η πρακτική με την οποία άτομα και οργανισμοί προστατεύουν την φυσική και πνευματική τους ιδιοκτησία από κάθε μορφή επίθεσης και λεηλασίας. Μια καλή πολιτική ασφάλειας περιλαμβάνει έναν αριθμό από ιδιότητες. Εδώ θα συζητήσουμε αυτές τις ιδιότητες.

2.2 Ιδιότητες της ασφάλειας

- Η πρώτη ιδιότητα είναι αυτή της **πιστοποίησης (authentication).** Η πρώτη δουλειά οποιουδήποτε συστήματος ασφαλείας θα πρέπει να είναι η πιστοποίηση απομακρυσμένων χρηστών, συστημάτων και applets. Ειδικότερα ένα τέτοιο σύστημα θα πρέπει να μπορεί να βεβαιώσει ότι αυτός ή αυτό που βρίσκεται στην άλλη πλευρά του "σύρματος" είναι αυτός ή

αυτό που φανταζόμαστε και ισχυρίζεται ότι είναι. Ακόμη ειδικότερα πρέπει να είμαστε βέβαιοι ότι ο host με τον οποίο είμαστε συνδεδεμένοι είναι αυτός που νομίζουμε και το applet που κάναμε download είναι αυτό που περιμέναμε ότι είναι.

- Το δεύτερο χαρακτηριστικό που πρέπει να παρέχει ένα σύστημα ασφάλειας είναι η **εξουσιοδότηση (authorization).** Οι χρήστες αλλά πολύ περισσότερο οι διαχειριστές συστημάτων θα πρέπει να είναι σε θέση να καθορίζουν το επίπεδο εξουσιοδότησης και πρόσβασης που θα παρέχουν. Εδώ μεγάλη σημασία έχει η έννοια των επιπέδων, δηλαδή σε αξιόπιστες πηγές να δίνουμε μεγαλύτερη εξουσιοδότηση απ' ότι σε πηγές λιγότερο ή καθόλου αξιόπιστες.

- Το τρίτο χαρακτηριστικό είναι αυτό της **εμπιστευτικότητας (confidentialy).** Το χαρακτηριστικό αυτό έρχεται σε πολλές μορφές. Για παράδειγμα ο χρήστης θα πρέπει να είναι βέβαιος ότι τα δεδομένα που ανταλάσσει με έναν απομακρυσμένο host παραμένουν εμπιστευτικά. Ακόμη οι χρήστες ενδεχομένως να θέλουν να κρατηθεί μυστικό και το γεγονός ακόμη ότι συνδέθηκαν με τον συγκεκριμένο host. Εάν το παραπάνω γίνει με κάποιο τρόπο γνωστό αυτό θα αποκαλύψει πληροφορίες για σχέσεις με πελάτες και προμηθευτές και θα αποτελέσει όπλο

στα χέρια των ανταγωνιστών. Σε συντομία ένα ασφαλές σύστημα θα πρέπει να διασφαλίζει ότι όλες οι ιδιότητες μιας σύνδεσης, χρόνος, τόπος, διάρκεια, περιεχόμενο κ.λ.π. παραμένουν μυστικά.

- Με δεδομένα την πιστοποίηση και την εξουσιοδότηση ένα σύστημα μπορεί να εγγυηθεί ένα ικανοποιητικό επίπεδο **περιορισμού (containment).** Αυτό σημαίνει ότι με την δημιουργία του περιβάλλοντος εκτέλεσης και του "καλώς έχειν" των applets που εκτελούνται σε αυτό το περιβάλλον, μπορούμε να είμαστε βέβαιοι ότι όλα πηγαίνουν καλά και ελέγχουμε την κατάσταση βάζοντας τα applets σε ένα "κουτί" (applets in a box). Το σύστημα πια μπορεί να καθορίσει το μέγεθος, το σχήμα και τα όρια αυτού του "κουτιού" με βάση το επίπεδο της εξουσιοδότησης. Όπως θα δούμε και παρακάτω η Java υλοποιεί ένα μοντέλο ασφάλειας βασισμένο σε "κουτί" το sandbox.

- Το επόμενο χαρακτηριστικό είναι αυτό της **παρακολούθησης (auditing).** Κάθε σύστημα ασφαλείας που σέβεται τον εαυτό του θα πρέπει να έχει δυνατότητες παρακολούθησης και ταυτόχρονα καταγραφής. Έτσι αν σημειωθεί μια ανωμαλία ή ένα "ρήγμα" το σύστημα παρακολούθησης θα βοηθήσει τον διαχειριστή στην προσπάθεια του να απομονώσει και να

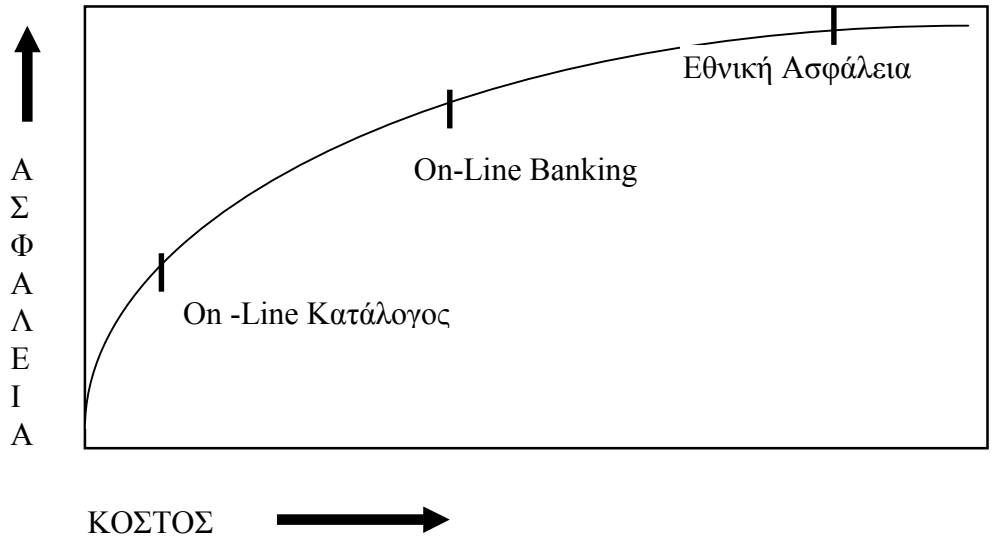
διορθώσει το πρόβλημα. Πιο πρακτικά, βοηθάει τους ανθρώπους που ασχολούνται να εκτιμήσουν το μέγεθος της ζημιάς που έχει γίνει και να επιληφθούν των κατάλληλων ενεργειών. Είναι σημαντικό βέβαια να σημειώσουμε ότι η καταγραφή δεν αποτρέπει μια επίθεση, μπορεί όμως να αφυπνήσει το προσωπικό, να καταγράψει το μέγεθος του "ρήγματος" και να βοηθήσει στην τελική διόρθωση.

- Τέλος το χαρακτηριστικό της πιστοποίησης προσφέρει ένα ακόμη σημαντικό πλεονέκτημα αυτό της **μη-αποδοχής (non-repudiation).** Αυτό σημαίνει ότι τα μέρη που συμμετέχουν σε μια σύνδεση και στην συνέχεια σε μια δοσοληψία δεν μπορούν αργότερα να αρνηθούν την συμμετοχή τους,. Το χαρακτηριστικό αυτό μπορεί να αποδείξει ότι ήταν εκεί και συμμετείχαν. Σαν αποτέλεσμα αυτού του χαρακτηριστικού συνναλασόμενα μέρη μπορούν να συμμετέχουν σε "ηλεκτρονικές συμφωνίες" καθώς κάθε πλευρά μπορεί να πιστοποιηθεί και να αποδείξει και την παρουσία της άλλης πλευράς.

2.3 Τιμή και Ευχρηστία (Price and Usability)

Η απόφαση για το πόσο θα ξοδέψουμε για την ασφάλεια είναι ένα σημαντικό θέμα. Διαφορετικό κόστος έχει για την διασφάλιση της μια εταιρεία με καταλόγους αγορών on-line, διαφορετικό μια

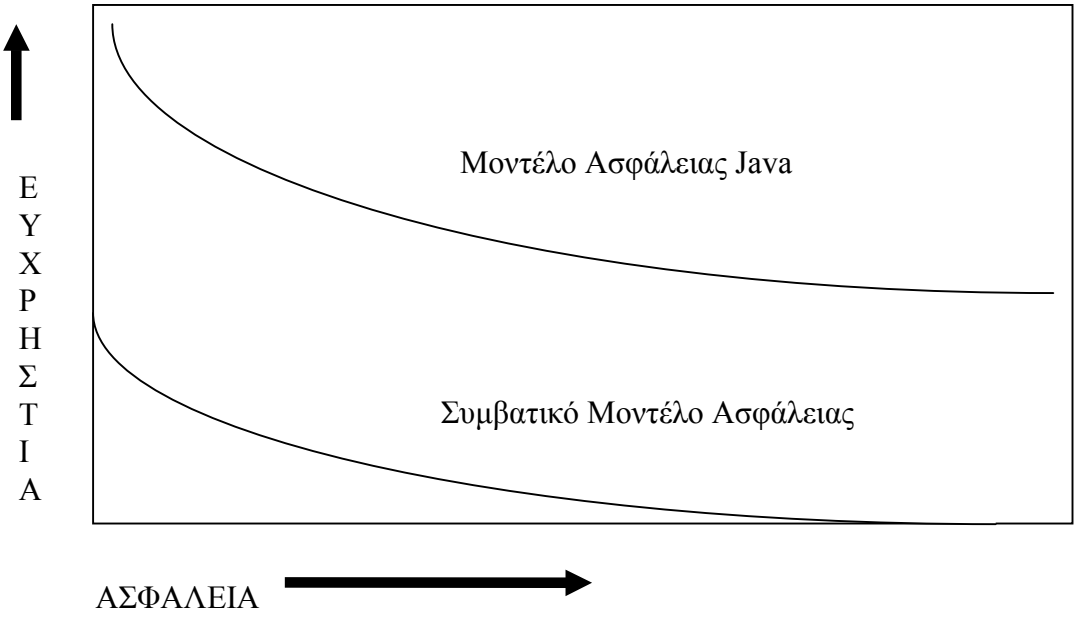
τράπεζα που κάνει δοσοληψίες και σίγουρα πολύ διαφορετικό ένα υπουργείο το οποίο μεταφέρει απόρρητα κρατικά μυστικά. Το Σχήμα 2.1 μας δείχνει την σχέση κόστους και ασφάλειας.



Σχήμα 2.1

Ένα άλλο σημαντικό γεγονός είναι ότι η ασφάλεια έρχεται σε αντίθεση με την ευχρηστία. Δεν θα πρέπει η ασφάλεια να δημιουργεί ιδιαίτερα προβλήματα ευχρηστίας στους χρήστες. Τα συστήματα θα πρέπει να είναι προσβάσιμα σ' αυτούς που τα χρησιμοποιούν. Ιδιαίτερα περιοριστικοί κανόνες μπορούν να μειώσουν σημαντικά την παραγωγικότητα των χρηστών. Σε ακραίες περιπτώσεις οι έμπειροι χρήστες θα παρακάμψουν την ασφάλεια για να κάνουν "καλύτερα την δουλειά τους". Η Java και εδώ υπερέχει διότι έχοντας αρκετά χαρακτηριστικά ασφαλείας ενσωματωμένα (built -in) είναι

αρκετά “διάφανη” προς τον χρήστη. Αντίθετα τα δυαδικά μοντέλα ασφάλεις όπως φαίνεται και στο Σχήμα 2.2 πρέπει να θυσιάσουν την ευχρηστία τους για να επιτύχουν έναν ικανοποιητικό βαθμό ασφάλειας.



Σχήμα 2.2

Αυτά βέβαια που αναφέραμε παραπάνω είναι καλά και αποτελεσματικά με δεδομένη την ασφάλεια **του ανθρώπινου παράγοντα**. Δηλαδή τα εργαλεία και οι τεχνικές μπορούν να προσφέρουν πολύ λίγο έως καθόλου στην προστασία ενός οργανισμού αν οι ίδιοι οι άνθρωποι του οργανισμού δεν τηρούν κάποιους κανόνες ασφάλειας. Ένας οργανισμός θα πρέπει να καθιστά υπεύθυνους και υπόλογους κάθε μέλος του προκειμένου να υπάρχουν πρακτικές ασφάλειας σε όλα τα επίπεδα. Σε αυτές τις

πρακτικές θα πρέπει να "εθιστούν" τα μέλη και να επαναπροσδιορίζονται με συνεχή εκπαίδευση. Το αποτέλεσμα μιας επίθεσης ή της απάθειας κάποιων είναι το ίδιο τραγικό.

3. Τι συγκεκριμένα είναι το Java Security;

3.1 Η ασφάλεια της γλώσσας προγραμματισμού Java.

Τα προβλήματα ασφάλειας χωρίζονται σε δύο μεγάλες κατηγορίες, αυτά που γίνονται κατά λάθος και αυτά που δημιουργούνται ηθελημένα και χαρακτηρίζονται ως "κακόβουλα". Ας ξεκινήσουμε απ' τα πρώτα και να πούμε ότι όσοι έχουν γράψει έστω και ένα μικρό πρόγραμμα γνωρίζουν τι θα πει πρόβλημα. Ένας βρόγχος που δεν τερματίζει ποτέ, μια συνθήκη που είναι πάντα σωστή ή πάντα λάθος, ένας τύπος δεδομένων ασύμβατος, μια μεταβλητή που της αποθηκεύεται λάθος τύπος τιμής ή τιμή εκτός ορίων, μια μη αρχικοποιημένη μεταβλητή είναι τα πιο συνηθισμένα λάθη που κάνουν αρχάριοι αλλά ακόμη και έμπειροι προγραμματιστές. Ένα τέτοιο λάθος κάνει συνήθως το πρόγραμμα μας να "κρεμάσει", τον Η/Υ να "παγώσει" και εμάς να καθόμαστε με αμηχανία μπροστά στην οθόνη μας. Φανταστείτε λοιπόν ότι το ίδιο πράγμα κάνει και μια κακογραμμένη εφαρμογή σε Java η οποία συντακτικά μοιάζει με την C++. Βεβαίως τέτοιου είδους σφάλματα γίνονται από την άγνοια

αυτού που τα γράφει και όχι από κακή πρόθεση, πάντως το πρόβλημα έχει δημιουργηθεί.

Ευτυχώς που η Java μοιάζει με την γλώσσα προγραμματισμού C++ αλλά διαφέρει στο ότι η η δεύτερη είναι πιο επιρρεπής στα σφάλματα. Η Java είναι κατασκευασμένη εκ' θεμελίων με τους κανόνες ασφαλείς υπ' όψη και έτσι αποφεύγει πολλές απ' τις κακοτοπιές στις οποίες θα μπορούσε να πέσει κάποιος προγραμματιστής που χρησιμοποιούσε την C++. Χαρακτηριστικά παραδείγματα είναι για την Java, ο έλεγχος ορίων πίνακα, η ισχυρή τυποποίηση των μεταβλητών και η αυτόματη απελευθέρωση της μνήμης που δεν χρησιμοποιείται. Έτσι αποφεύγονται καταστάσεις δημιουργίας εξαιρέσεων (exceptions) όταν π.χ. πάμε να καταχωρήσουμε τιμή στην εντέκατη θέση ενός πίνακα δέκα θέσεων, ή όταν πάμε να αποθηκεύσουμε μια πραγματική τιμή σε μια ακέραιη μεταβλητή ή όταν εξαντλήσουμε τα αποθέματα μνήμης του Η/Υ μας γιατί ξεχάσαμε να απελευθερώσουμε την μνήμη που καταλάμβαναν οι δείκτες αλλά το έκανε για μας η ίδια η γλώσσα Java με τον μηχανισμό περισυλλογής "σκουπιδιών" (garbage collector). Βέβαια καμία γλώσσα προγραμματισμού δεν είναι 100% ασφαλής απέναντι

σε σφάλματα, π.χ. ούτε η Java μπορεί να μας προστατεύσει από έναν βρόγχο που δεν τελειώνει ποτέ αλλά περιορίζει τις πιθανότητες.

3.2 Η ασφάλεια των applets της Java.

Η λογική με την οποία λειτουργεί το applet είναι απ' την φύση της επικίνδυνη. Ενώ επενδύονται τεράστια ποσά για την δημιουργία μηχανισμών ασφαλείας που θα προστατεύουν τα δίκτυα και τους Η/Υ τον καθένα ξεχωριστά, με τον ερχομό των applets και της Java ανατρέπονται όλα όσα ξέραμε. Οι μέθοδοι προστασίας πρέπει να αναθεωρηθούν και αν ειδωθούν μέσα από το πρίσμα των νέων τεχνολογιών. Για να το κάνουμε πιο κατανοητό οι κλασσικές μέθοδοι ασφάλειας επικέντρωνουν στην αποτροπή της εισβολής στο δίκτυο, στο να κρατάνε δηλαδή μακριά οποιονδήποτε εισβολέα. Εδώ επιτρέπουμε στον εισβολέα να μπεί θεωρώντας τον ακίνδυνο και στην συνέχεια εξετάσουμε την επικινδυνότητα του. Ενώ μέχρι τώρα κλειδώναμε την πόρτα μας και βάζαμε συναγερμούς και συστήματα ασφάλειας, τώρα ανοίγουμε την πόρτα μας διάπλατα σε ένα αθώο μικρό παιδάκι που μπορεί όμως στην πορεία να αποδειχθεί μανιακός δολοφόνος.

3.3 Τελικά αξίζει τον κόπο να χρησιμοποιήσουμε την Java.

Εδώ προκύπτει το εξής ερώτημα: ωραία, αφού τα applets είναι τόσο επικίνδυνα γιατί δεν τα αφήνουμε στην άκρη μαζί και την Java

και να συνεχίσουμε να δουλεύουμε όπως δουλεύαμε και πριν με πράγματα γνωστά και ακίνδυνα. Η απάντηση είναι απλή: βεβαίως και μπορούμε να συνεχίσουμε να δουλεύουμε όπως ξέραμε. Απομονώνουμε το δίκτυο μας από το Internet συνεχίζουμε με τις κλασσικές μεθόδους και έχουμε το κεφάλι μας ήσυχο. Η Java είναι μια τεχνολογία όπως οι άλλες και σίγουρα θα υπάρχουν εναλλακτικές λύσεις.. Ας τις ακολουθήσουμε και δεν θα έχουμε πρόβλημα.

Δυστυχώς τα πράγματα δεν είναι ακριβώς έτσι. Αναφέρουμε μόνο το γεγονός ότι η παγκοσμιοποίηση της αγοράς και ο ανταγωνισμός που είναι καθημερινά και πιο έντονος αναγκάζει τις επιχειρήσεις να καινοτομούν και να βρίσκονται ή να προσπαθούν τουλάχιστον να βρίσκονται στην αιχμή της τεχνολογίας. Αιχμή της τεχνολογίας σήμερα είναι το Internet. Σε μερικά χρόνια όποιος δεν έχει βρεί το ρόλο του και την θέση του σ' αυτό το παγκόσμιο χωριό είναι καταδικασμένος σε μαρασμό και αποτυχία. Άρα θα πρέπει η κάθε επειχείρηση να εξερευνά και να ενημερώνεται για τις εξελίξεις σε αυτόν τον τεχνολογικό χώρο. Αιχμή του δόρατος στο Internet είναι οι εμπορικές, τραπεζικές και λοιπές συναλλαγές μέσα στο δίκτυο. Προκειμένου να επιτευχθούν τα παραπάνω θα πρέπει να υπάρχουν τα εργαλεία που θα δημιουργήσουν αυτές τις εφαρμογές. Τα εργαλεία

αυτά θα πρέπει να είναι εύχρηστα ,user-friendly αλλά το σπουδαιότερο θα πρέπει να παράγουν προγράμματα που θα εκτελούνται σε ΟΛΩΝ ΤΩΝ ΕΙΔΩΝ τα μηχανήματα που είναι συνδεδεμένα στο Internet, PC, Apple, Unix systems κ.λ.π.

Το μόνο εργαλείο που έχει τις παραπάνω προυποθέσεις σήμερα είναι η Java και αυτή η διαπλατφορμική της ιδιότητα, δηλαδή να εκτελείται σε οποιονδήποτε τύπο μηχανήματος την έκανε το de facto development standard στο Ιντερνέτ, τόσο διάσημη και απαραίτητη σε τόσο μικρό χρονικό διάστημα. Για να αποδείξουμε ότι κάπως έτσι είναι τα πράγματα οι δύο μεγάλες εταιρείες κατασκευής browsers η Microsoft και η Netscape έχουν κάνει τις τελευταίες εκδόσεις των browsers τους Java-enabled δηλαδή έχουν την δυνατότητα να χειριστούν applets.

4. Το μοντέλο ασφάλειας της Java;

4.1 Η έννοια του αμμοδοχείου (sandbox)

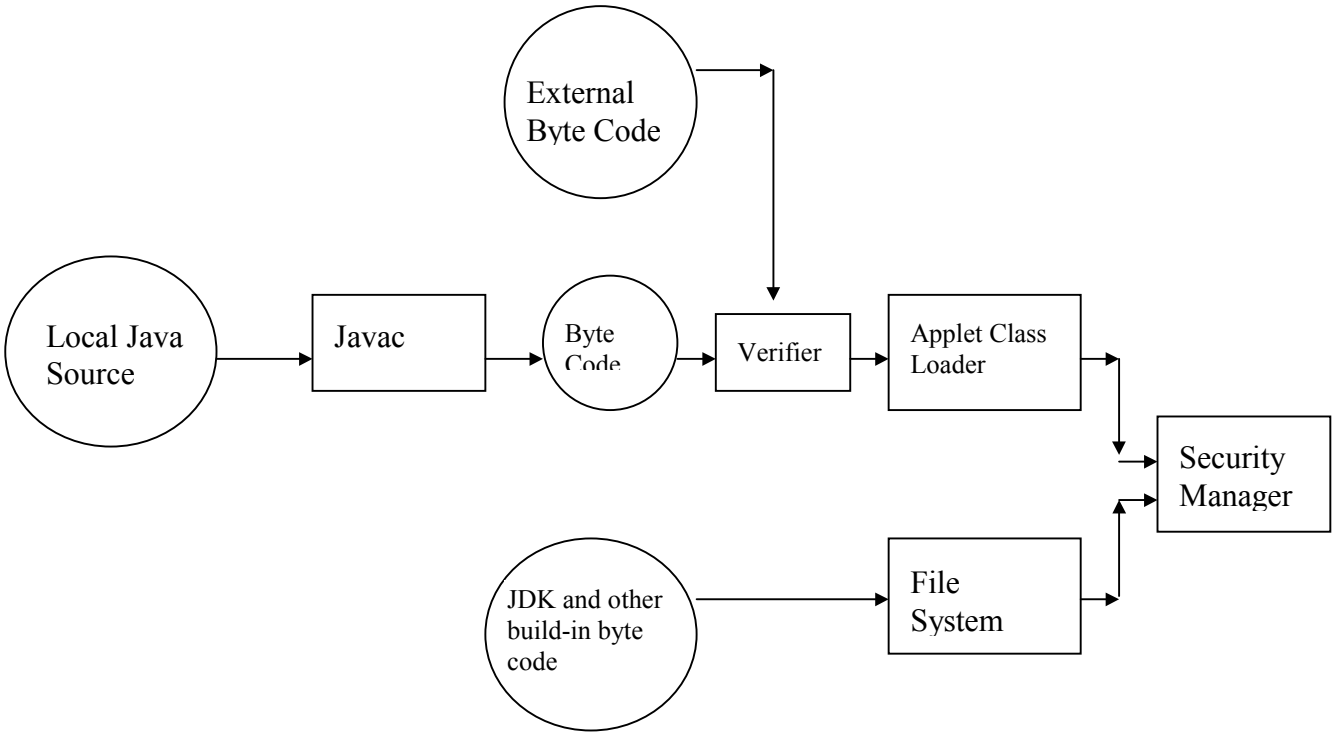
Επανερχόμαστε στα είδη προβλημάτων και εξετάζουμε την περίπτωση δημιουργίας επίτηδες ενός εχθρικού applet. Όταν εισάγει στο μηχάνημα μας έναν ιό, έναν δούρειο ίππο (trojan horse) ή όταν αυτό το applet αρχίζει να κάνει πράγματα που δεν θα έπρεπε να

κάνει. Τα πιο χαρακτηριστικά απ' αυτά είναι να διαβάσει αρχεία του Η/Υ μας, να διαγράψει αρχεία, να μεταφέρει αρχεία μας, να δημιουργήσει καινούργια σύνδεση με άλλο μηχάνημα εκτός απ' τον server απ' τον οποίο προήλθε. Πως αντιμετωπίζουμε εμείς αυτές τις καταστάσεις.

Ευτυχώς η ίδια η αρχιτεκτονική της Java προέβλεψε όλες αυτές τις περιπτώσεις ή τουλάχιστον τις περισσότερες απ' αυτές. Έτσι μέσα στις θεμελιώδεις αρχές της Java είναι ότι ΤΟ ΚΑΘΕ APPLET ΚΑΝΕΙ ΣΥΓΚΕΚΡΙΜΕΝΑ ΠΡΑΓΜΑΤΑ ΣΕ ΣΥΓΚΕΚΡΙΜΕΝΟΥΣ ΧΩΡΟΥΣ ΚΑΙ ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΚΑΝΕΙ ΚΑΤΙ ΠΟΥ ΔΕΝ ΘΑ ΕΠΡΕΠΕ ΠΟΤΕ ΝΑ ΚΑΝΕΙ. Υπάρχει εδώ η μεταφορική έννοια του αμμοδοχείου (sandbox) όπου το κάθε applet λειτουργεί μέσα σε αυτό το αμμοδοχείο και πουθενά αλλού. Η Java με αυτόν τον απλό τρόπο επιτρέπει το download και την εκτέλεση μη - ασφαλή κώδικα χωρίς να εκθέτει το σύστημα μας σε ανώφελα ρίσκα. Ένα applet μπορεί να προκαλέσει το "χάος" στο δικό του αμμοδοχείο αλλά θα αφήσει ανεπηρέαστα όλα τα άλλα αμμοδοχεία. Ακόμη και μέσα στο δικό του αμμοδοχείο, μπορούμε να καθορίσουμε τι μπορεί και τι δε μπορεί να κάνει το applet. Έτσι λοιπόν η Java με απλά λόγια επιτρέπει "ύποπτο" κώδικα να εκτελείται σε ένα ασφαλές περιβάλλον χωρίς το

παραμικρό πρόβλημα. Ας δούμε όμως με λεπτομέρεια πως ακριβώς δουλεύει το μοντέλο ασφάλειας της Java.

4.2 Τι περιλαμβάνει το μοντέλο ασφάλειας της JAVA



Σχήμα 4.1

Το μοντέλο ασφάλειας της Java διαφέρει ριζικά απ' τις παραδοσιακές προσεγγίσεις. Τα περισσότερα λειτουργικά συστήματα επιτρέπουν στις εφαρμογές να έχουν πλήρη σχεδόν πρόσβαση στους πόρους του συστήματος. Οι διαχειριστές θα πρέπει να βασίζονται στους χρήστες για να διασφαλίσουν ότι οι πόροι είναι αρκούντος

προστατευμένοι. Δεύτερον, είναι συνήθως υποχρέωση του χρήστη και κάνει έναν έλεγχο του προγράμματος (π.χ. για ύπαρξη ιών) πριν την εκτέλεση. Αυτή η τακτική έχει δύο προφανή μειονεκτήματα, επαφίεται στην καλή διάθεση του χρήστη για να κάνει τον έλεγχο και ακόμη βασίζεται στην ακρίβεια του προγράμματος ανίχνευσης ιών. Όπως θα δούμε η Java λαμβάνει μια πιο δυναμική προσέγγιση. Εδώ συνδυάζει ένα σύνολο συστατικών που συνεργάζονται μεταξύ τους, και ποικίλουν από διαχειριστές ασφάλειας οι οποίοι εκτελούνται σαν μέρος της εφαρμογής, έως μετρήσεις ασφάλειας ενσωματωμένες μέσα στην Java. Για να δούμε λοιπόν αυτά τα συστατικά με μεγαλύτερη λεπτομέρεια.

4.2.1. Class Loader

Πρώτο συστατικό είναι ο class loader. Όπως έχουμε αναφέρει τα applets εκτελούνται σαν μέρος μιας ιστοσελίδας και μάλιστα χωρίς ο χρήστης να αντιλαμβάνεται οτιδήποτε (διαφανώς). Για να φορτωθεί το applet, ο Browser καλεί πρώτα και υποχρεωτικά τον class loader. Εδώ αρχίζει η πρώτη γραμμή άμυνας του μοντέλου. Ο class loader καθορίζει πως και πότε ένα applet μπορεί να φορτώσει μια κλάσση δηλαδή κώδικα που πρόκειται να εκτελεστεί. Οι βασικές λειτουργίες του είναι οι ακόλουθες, φέρνει τον κώδικα του applet από το απομακρυσμένο μηχάνημα, ορίζει μια ιεραρχία namespace (δηλαδή

χώρων ξεχωριστών για κάθε είδος κλάσης, που από εδώ και πέρα θα ονομάζονται περιοχές). Μια από τις πιο βασικές λειτουργίες του είναι να διασφαλίσει ότι τα εκτελούμενα applets δεν αντικαθιστούν συστατικά του συστήματος. Πιο συγκεκριμένα αποτρέπει τα applets απ' το να δημιουργήσουν τους δικούς τους class loaders. Επίσης αποτρέπει τα applets απ' το να καλούν μεθόδους οι οποίες είναι μέρος του class loader. Το εκτελέσιμο (run-time) περιβάλλον την Java, επιτρέπει σε πολλαπλούς class loaders να συνυπάρχουν ταυτόχρονα ο καθένας όμως να βρίσκεται στην δικιά του περιοχή. Το περιβάλλον της Java επιτρέπει να ομαδοποιούνται οι κλάσεις με βάση την προέλευση τους, αλλού οι τοπικές, αλλού οι απομακρυσμένες από έμπιστα μηχανήματα, αλλού οι ύποπτες. Θέτωντας τέτοιους περιορισμούς στις περιοχές ο class loader αποτρέπει "ύποπτα" applets απ' το να αποκτήσουν πρόσβαση σε πόρους του συστήματος. Έτσι αυτές οι περιοχές αποτελούν την πρώτη γραμμή άμυνας που παρέχει το αμμοδοχείο.

Όπως έχουν σχεδιαστεί οι class loaders φορτώνουν τα applets και τις αντίστοιχες κλάσεις. Όταν εκτελείται το applet ο Browser καλεί το applet class loader το οποίο φαίρνει το applet. Φυσιολογικά τα applets δεν μπορούν να εγκαταστήσουν νέους class loaders έτσι ο αρχικός class loader ελέγχει το περιβάλλον. Αυτός ο class loader

δημιουργεί τις ξεχωριστές περιοχές μια για κάθε applet. Έτσι τα applets έχουν πρόσβαση μόνο στις δικές τους κλάσεις και σε κλάσεις που ανήκουν στις τυποποιημένες βιβλιοθήκες της Java. Έτσι δεν έχουν πρόσβαση σε κλάσεις που ανήκους σε άλλα applets. Αυτό έχει δύο πλεονεκτήματα, οι ξεχωριστές περιοχές κάνουν δύσκολο για τα applets να μοιραστούν τους πόρους που καταλαμβάνουν και έτσι να κάνουν μια οργανωμένη επίθεση και επίσης οι προγραμματιστές των applets δεν χρειάζεται αν ασχολούνται με όμοια ονόματα, μιας και κάθε όνομα αρκεί να είναι μοναδικό μέσα σε ένα και μόνο applet.

Τα προβλήματα που μπορεί να προκύψουν είναι δύο, στο περιβάλλον του Browser συνήθως τον class loader τον παρέχει ο κατασκευαστής του Browser. Παρ' όλο που οι συγκεκριμένοι class loaders βασίζονται σε πίνακες οι οποίοι δίνονται απ' την μητρική εταιρεία την Sun πολλές φορές η υλοποίηση τους δεν είναι το ίδιο ασφαλής. Επίσης οι εφαρμογές της Java έχουν το δικαίωμα να δημιουργήσουν τους δικούς τους class loaders. Θα πρέπει να προσέξουμε όταν σχεδιάζουμε class loaders να ακολουθούμε όλες τις αρχές που έχουν καθοριστεί απ' το JDK (Java Development Kit). Αν γίνει αυτό οι δυσκολίες θα είναι ελάχιστες.

4.2.2 ByteCode Verifier

Το δεύτερο συστατικό στοιχείο που προσδίδει ασφάλεια στο αμμοδοχείο είναι ο Bytecode Verifier. Έχουμε πει ότι ο πηγαίος κώδικας της Java μετατρέπεται με τον compiler σε έναν διαπλατφορμικό ψηφιοκώδικα (Byte Code). Πριν ο Class Loader επιτρέψει στο applet να εκτελεστεί, ο κώδικας του πρέπει να ελεγχθεί απ' τον Bytecode Verifier. Ο ελεγκτής αυτός μπορεί να διαπιστώσει ότι ο κώδικας του applet ο οποίος μπορεί να μην έχει δημιουργηθεί από έναν Java compiler ακολουθεί όλους τους κανόνες της γλώσσας. Ο ελεγκτής εκτελεί διάφορους ελέγχους. Σε ένα βασικό επίπεδο βεβαιώνει ότι ο κώδικας είναι σύμφωνος με τους κανόνες και τις προδιαγραφές της γλώσσας. Σε ένα πιο σύνθετο επίπεδο ο ελεγκτής εφαρμόζει ένα ενσωματωμένο θεώρημα σε σχέση με τον κώδικα. Με το παραπάνω διασφαλίζεται ότι το applet ανάμεσα στα άλλα δεν πλαστογραφεί δείκτες, δεν καταστρατηγεί τους κανόνες πρόσβασης ή δεν αποκτά πρόσβαση σε αντικείμενα μέσα από παράνομα αντίγραφα. Ο Bytecode Verifier σε συνδυασμό με χαρακτηριστικά ασφάλειας που είναι ενσωματωμένα μέσα στην γλώσσα μπορεί να διασφαλίσει ότι:

- ο κώδικας που έχει υποστεί compile είναι φορμαρισμένος σωστά

- τα εσωτερικά stacks (χώροι της μνήμης που αποθηκεύονται τιμές μεταβλητών) δεν θα υπερχειλίσουν ούτε θα υποχειλίσουν. Σε τέτοιες καταστάσεις οι κακόβουλοι hackers βρίσκουν πρόσφορο έδαφος για να δράσουν.
- Δεν υπάρχουν "παράνομες " μετατροπές δεδομένων (για παράδειγμα ο verifier δεν θα επιτρέψει σε ακεραίους να λειτουργήσουν σαν δείκτες). Αυτό μας διασφαλίζει ότι δεν θα έχουν δυνατότητα κάποιες μεταβλητές να έχουν πρόσβαση σε απαγορευμένες περιοχές της μνήμης.
- Οι εντολές Byte Code θα έχουν παραμέτρους σωστά "τυποποιημένες".

Όλες οι προσβάσεις σε μέλη κλάσσεων είναι νομότυπες. Αυτό σημαίνει ότι τα ιδιωτικά δεδομένα ενός αντικειμένου παραμένουν ιδιωτικά.

4.2.3 Security Manager

Το τρίτο και πιο σημαντικό συστατικό του μοντέλου ασφάλειας της Java είναι ο Security Manager. Η δουλειά του είναι να κάνει ελέγχους κατά τον χρόνο εκτέλεσης για όλες αυτές τις μεθόδους που θα χαρακτηρίζαμε επικίνδυνες, μεθόδους δηλαδή που αιτούνται Ε/Ε αρχείων, πρόσβαση στο δίκτυο ή αυτές που θέλουν να ορίσουν έναν

νέο class loader. Σε όλες αυτές τις περιπτώσεις ο Security Manager μπορεί να ασκήσει βέτο σε οποιαδήποτε αίτηση. Για παράδειγμα εαν ένα applet καλέσει μια μέθοδο “ανάγνωσης”, η JVM συμβουλεύται τον Security Manager. Εαν το applet είναι έμπειστο (trusted) τότε η αίτηση θα επιτραπεί, εαν όχι η αίτηση θα απορριφθεί. Κάποια απ’ τα καθήκοντα του Security Manager περιλαμβάνουν:

- Διαχείριση των διαδικασιών socket.
- Προστασία της πρόσβασης σε προστατευμένους πόρους όπως αρχεία, προσωπικά δεδομένα.
- Έλεγχος της δημιουργίας και της πρόσβασης σε προγράμματα και διαδικασίες λειτουργικού συστήματος.
- Αποτροπή της εγκατάστασης νέων Class Loader.
- Διατήρηση της ακεραιότητας των threads.
- Έλεγχος της πρόσβασης σε πακέτα (ομάδες κλάσσεων) της Java.

4.3 Type Safety

Η Java είναι σχεδιασμένη ώστε να υποχρεώνει το type safety. Αυτό σημαίνει ότι τα προγράμματα δεν μπορούν να έχουν πρόσβαση στην μνήμη με μη επιτρεπτούς τρόπους. Ας το εξηγήσουμε με ένα παράδειγμα για να γίνει πιο κατανοητό. Υποθέτουμε ότι έχουμε ένα αντικείμενο που ονομάζεται ημερολόγιο. Αυτό έχει ιδιότητες όπως

Ημερομηνία, Ραντεβού, Ξυπνητήρι κ.λ.π. Η ιδιότητα Ξυπνητήρι προφανώς θα είναι λογικού τύπου και θα έχει δύο τιμές, αληθής και ψευδής. Ας υποθέσουμε ακόμη ότι υπάρχει και ένα άλλο αντικείμενο που για απλότητα το ονομάζουμε applet και όπου κατά διαβολική σύμπτωση η πρώτη του ιδιότητα είναι λογικού τύπου και αφορά την δυνατότητα πρόσβασης στα αρχεία. Αν δεν υπήρχε ο μηχανισμός του type safety τότε ένα πρόγραμμα θα μπορούσε να εφαρμόσει την ιδιότητα Ξυπνητήρι στο αντικείμενο applet με αποτέλεσμα να έχει την δυνατότητα να μεταβάλει όχι προφανώς την ιδιότητα Ξυπνητήρι αλλά αυτήν που αφορά την πρόσβαση στα αρχεία και κάνοντας την αληθή κάποιος να έχει απεριόριστη πρόσβαση στο σύστημα αρχείων. Αντιλαμβανόμαστε όλοι την ζημιά που θα μπορούσε να γίνει.

Πως αντιμετωπίζει αυτό το πρόβλημα η Java. Κάθε αντικείμενο της Java αποθηκεύεται σε μια συγκεκριμένη περιοχή της μνήμης. Η Java τοποθετεί μια ετικέτα σε κάθε αντικείμενο που δείχνει από ποια κλάση προέρχεται. Προκειμένου να θέσουμε την type safety ενώ εκτελείται το πρόγραμμα και πριν γίνει οποιοδήποτε ενέργεια ελέγχουμε αυτές τις ετικέτες. Αυτό ονομάζεται δυναμικός έλεγχος. Το μειονέκτημα του είναι ότι είναι σχετικά αργός. Γι’ αυτό και περισσότερο χρησιμοποιείται ο στατικός έλεγχος που ουσιαστικά γίνεται πριν εκτελεστεί το πρόγραμμα.

4.4 Επιπλέον Στοιχεία Ασφάλειας

4.4.1 Ασφάλεια των εφαρμογών Java.

Αρχίζουμε με την ασφάλεια των εφαρμογών της Java. Όπως έχουμε ξαναπεί τα προγράμματα της Java είναι δύο ειδών, ολοκληρωμένες εφαρμογές ή applets. Οι εφαρμογές της αναπτύσονται, αγοράζονται και εγκαθίστανται ακριβώς με τον ίδιο τρόπο που θα γινόταν και σε οποιαδήποτε κλασσική εφαρμογή. Επειδή αυτές οι εφαρμογές εκτελούνται τοπικά, συνήθως δεν επιδέχονται τους λεπτομερείς ελέγχους ασφαλείας οι οποίοι εφαρμόζονται στα applets. Σαν συνέπεια αυτού εκτελούνται σαν "ασφαλής" εφαρμογές όπου η ασφάλεια επιτυγχάνεται με τον φυσικό έλεγχο του τοπικού περιβάλλοντος. Μια λεπτομέρεια, οι χρήστες θα πρέπει πριν εγκαταστήσουν οποιοδήποτε πρόγραμμα στον Η/Υ τους να κάνουν έλεγχο για ιούς, πολύ περισσότερο μάλιστα όταν αυτό το πρόγραμμα προέρχεται από το δίκτυο.

4.4.2 Ασφάλεια των applets

Ακολουθεί η ασφάλεια των applets. Στο παραδοσιακό δικτυακό περιβάλλον η προστασία είχε αποτέλεσμα με τον περιορισμό της πρόσβασης σε δεδομένα, λογισμικό και στο δίκτυο. Είναι αλήθεια ότι αν επισκεφτούμε μια σελίδα στο Ιντερνέτ είναι πιθανό να κατεβάσουμε μη-ασφαλή κώδικα στον Η/Υ μας. Ακόμη χειρότερα

εκτός και αν έχουμε θέσει κάποιους περιορισμούς ο κώδικας αυτός μπορεί να εκτελεστεί με απρόβλεπτο και επικίνδυνο τρόπο στον Η/Υ μας. Έτσι λοιπόν είναι πολύ πιθανό αυτός ο κώδικας να είναι το αντίστοιχο του "δούρειου ίππου" (trojan horse, κώδικας δηλαδή που εξωτερικά φαίνεται αθώος αλλά μπορεί να προκαλέσει μεγάλη ζημιά). Η Java περιορίζει αυτήν την πιθανότητα θέτοντας αυστηρούς περιοριστικούς όρους στην συμπεριφορά των applets, και πιο συγκεκριμένα:

Μη-ασφαλή applets δεν μπορούν να διαβάσουν και να γράψουν αρχεία στον τοπικό σκληρό δίσκο. Αυτό σαφώς εκμηδενίζει την δυνατότητα στα applets να προκαλέσουν κάποια σημαντική βλάβη στον Η/Υ μας

Όλα τα παράθυρα που δημιουργούνται απ' τα applets παίρνουν μια ετικέτα που το δείχνει αυτό. Έτσι οι χρήστες γνωρίζουν ότι ενδεχομένως εισάγουν δεδομένα σε τέτοιου είδους οθόνες.

Τα μη-ασφαλή applets δεν μπορούν να δημιουργήσουν συνδέσεις με άλλα μηχανήματα. Αυτό αποτρέπει την κακόβουλη συμπεριφορά πίσω από ένα firewall.

Παρ' όλα αυτά οι παραπάνω περιορισμοί είναι οι εξ' ορισμού. Έχουμε δει παραπάνω ότι υπάρχει η δυνατότητα να δεχθούμε κάποια applets απ' το δίκτυο ως ασφαλή. Έτσι αν εμείς το θελήσουμε αυτά

τα applets μπορούν να τύχουν της ίδιας μεταχείρισης με τα αντίστοιχα επιβεβαιωμένα-ασφαλή που βρίσκονται στον τοπικό μας σκληρό δίσκο. Μας δίνεται η δυνατότητα απ' την ίδια την Java δηλαδή να καθορίσουμε εμείς την πολιτική ασφάλειας που μας ταιριάζει. Για παράδειγμα μπορούμε να ορίσουμε ότι τα applets που προέρχονται απ' το site Α μπορούν να διαβάζουν αρχεία, ενώ αυτά που προέρχονται απ' το site Β μπορούν και να διαβάζουν και να γράφουν αρχεία. Αυτή η ρυθμιζόμενη ασφάλεια δεν εφαρμόζεται μόνο σε αρχεία αλλά σε συσκευές, σε θύρες και σε συνδέσεις.

4.4.3 Η ασφάλεια στον Browser.

Τέλος θα αναφερθούμε και στην ασφάλεια του Browser. Γνωρίζουμε ότι οι δύο πιο γνωστοί Browsers της Netscape και της Microsoft είναι Java-enabled. Είναι επίσης γνωστό ότι οι δύο αυτοί Browsers, έφαρμόζουν αυστηρούς πανομοιότυπους κανόνες ασφάλειας. Οι κανόνες αυτοί είναι:

- τα applets δεν μπορούν να διαβάσουν και να γράψουν αρχεία
- τα applets μπορούν να εγκαταστήσουν συνδέσεις μόνο με τον host απ' τον οποίο προήλθαν.

- τα applets έχουν πρόσβαση σε περιορισμένο αριθμό ιδιοτήτων του συστήματος.
- αν φορτωθεί ένα τοπικό applet και το αρχείο κλάσης του δεν βρίσκεται στο συγκεκριμένο ασφαλές μονοπάτι στο οποίο θα έπρεπε, τότε το θεωρείται μη-ασφαλές applet.

5. Κάποια συγκεκριμένα προβλήματα

Για να γίνουμε πιο συγκεκριμένοι και για να κατανοήσουμε καλύτερα τα προβλήματα ασφάλειας της Java, θα αναφέρουμε εδώ δύο χαρακτηριστικά παραδείγματα τα οποία καταδεικνύουν το πρόβλημα. Έχουν αναφερθεί διάφορες περιπτώσεις παραβιάσεων ασφάλειας με την βοήθεια της Java, οι οποίες ευτυχώς κατά κανόνα δημιουργήθηκαν σε κάποιο πανεπιστημιακό εργαστήριο με σκοπό την έρευνα και τίποτα περισσότερο. Τα προβλήματα αυτά αναφέρθηκαν και διορθώθηκαν σε επόμενες εκδόσεις των Broswer.

5.1. Το πρόβλημα με το DNS

Κάθε Η/Υ που είναι συνδεδεμένος στο Ιντερνέτ έχει έναν αριθμό IP που προσδιορίζει αυτόν τον Η/Υ μοναδικά. Αυτός ο αριθμός τεμαχίζεται σε τέσσερις αριθμούς από 0-255 οι οποίοι χωρίζονται μεταξύ τους με τελείες. Ένα παράδειγμα είναι ο αριθμός 195.20.10.2

κ.λ.π. Επειδή για τον χρήστη η απομνημόνευση τέτοιων αριθμών είναι δύσκολη, κάθε τέτοιος αριθμός έχει αντιστοιχιστεί σε ένα όνομα που σαφώς είναι πολύ πιο κατανοητό, π.χ. www.station1.uom.gr, www.terminal2.microsoft.com κ.λ.π. Αυτός ο μηχανισμός ονοματοδοσίας λειτουργεί με την βοήθεια του DNS (Domain Name Server). Το πρόβλημα ξεκινάει επειδή υπάρχει η δυνατότητα ένα μηχάνημα με ένα όνομα DNS να έχει δύο ή και περισσότερες διευθύνσεις IP αλλά και το αντίθετο, δύο διαφορετικά ονόματα DNS να έχουν το ίδιο IP.

Σύμφωνα με τους κανόνες των applets, ένα applet δεν μπορεί να συνδεθεί με άλλο μηχάνημα στο δίκτυο εκτός απ' αυτό απ' το οποίο προήλθε. Με ποιό μηχανισμό διασφαλίζεται αυτό. Όταν ένα applet αιτήται μια σύνδεση με κάποιον μηχάνημα το DNS μεταφράζει πρώτα την διεύθυνση του Web Server απ' τον οποίο προήλθε σε μια λίστα IP διευθύνσεων. Το ίδιο κάνει και με το όνομα αυτού του μηχανήματος με το οποίο πρόκειται να συνδεθεί. Έτσι υπάρχουν δύο λίστες διευθύνσεων IP τις οποίες συγκρίνει και αν υπάρχει μια κοινή διεύθυνση, επιτρέπει την σύνδεση, εάν δεν υπάρχει που σημαίνει ότι το applet προσπαθεί να συνδεθεί με τρίτο μηχάνημα και όχι σ' αυτό απ' το οποίο προήλθε τότε ο μηχανισμός ασφαλείας αρνείται να κάνει την σύνδεση.

Πως μπορεί τώρα να ενεργήσει κάποιος κακόβουλος. Απ΄ την μια μεριά είναι ο κακός που "στήνει" ένας Web Server με το όνομα www.κακόβουλος.com με διεύθυνση IP 66.66.66.66. Μεσα σ' αυτόν υπάρχει ένα μηχάνημα το www.H/Y1.κακόβουλος.com. Απ' την άλλη πλευρά υπάρχει ο www.περίεργος.com με διεύθυνση IP 1.1.1.1 και ο www.αθώος.com με διεύθυνση IP 0.0.0.0. Το σκηνικό είναι έτοιμο. Ο περίεργος επισκέπτεται το domain του κακόβουλου απ' το οποίο "κατεβαίνει" ένα applet. Στην συνέχεια αυτό το applet αιτήται μια σύνδεση με το μηχάνημα H/Y1 του κακόβουλου. Το DNS επειδή έχει την λάθος δυνατότητα να επιστρέψει περισσότερες από μια διευθύνσεις IP μπορεί να επιστρέψει τις διευθύνσεις 0.0.0.0 και 66.66.66.66. Ο μηχανισμός ασφαλείας της Java επειδή βρίσκει την διεύθυνση 66.66.66.66 να ταιριάζει νομίζει ότι η σύνδεση θα γίνει με τον Web Server απ' τον οποίο προήλθε το applet. Τελικά όμως εξαιτίας των διπλών διευθύνσεων το μηχάνημα συνδέεται στην διεύθυνση 0.0.0.0 με το μηχάνημα του αθώου ο οποίος στο τέλος την πληρώνει.

Τελικά η λύση στο πρόβλημα βρέθηκε με τον περιορισμό του DNS στο να δίνει μια μόνο διεύθυνση η οποία αποθηκεύεται και είναι αυτή του Web Server απ' την οποία προήλθε το applet και καμία άλλη.

5.2 Το πρόβλημα με τις καθέτους.

Το δεύτερο πρόβλημα προκύπτει απ'το γεγονός ότι η Java θεωρεί ως ασφαλή κώδικα Java αυτόν που προέρχεται απ' τον τοπικό σκληρό δίσκο και ύποπτο αυτόν που προέρχεται απ' το δίκτυο. Άρα αν κάποιος μπορέσει να τοποθετήσει ένα κακόβουλο κομμάτι κώδικα στον τοπικό σκληρό δίσκο τότε η Java θα το χειριστεί σαν ασφαλή κώδικα. Αφού ο κώδικας αυτός τοποθετηθεί στον σκληρό δίσκο θα πρέπει να τοποθετηθεί και στο κατάλληλο κατάλογο για να μπορέσει η ίδια η Java να το καλέσει, μιας και ο κώδικας της Java φορτώνεται καλώντας το ένα κομμάτι το άλλο.

Ας πάρουμε όμως τα πράγματα με την σειρά. Το να τοποθετήσουμε ένα κομμάτι κώδικα μέσα στον τοπικό σκληρό δίσκο κάποιου δεν είναι ιδιαίτερα δύσκολο. Πολλά μηχανήματα έχουν δημόσιο FTP κατάλογο όπου μπορούμε να βάλουμε ότι θέλουμε, επίσης πολλά υποψήφια θύματα χρησιμοποιούν ένα δημόσιο μηχάνημα και ο επιτιθέμενος παίρνωντας λογαριασμό σε αυτό το μηχάνημα μπορεί να βάλει αρχεία όπου θέλει. Επίσης υπάρχει και ένας τρόπος χρησιμοποιώντας τον μηχανισμό cache του Browser, αλλά δεν θα αναφερθώ εδώ με λεπτομέρειες.

Οι ασφαλείς κλάσσεις της Java βρίσκονται σε συγκεκριμένο κατάλογο του τοπικού σκληρού δίσκου. Για παράδειγμα είναι στον

κατάλογο \programs\netscape\java\applets. Ότι μπορεί να μπεί εκεί μέσα απ' το παράθυρο έχει την δυνατότητα να κάνει την ζημιά. Τα ονόματα των κλάσσεων που καλούνται έχουν την μορφή java.applets.class1, java.applets.class2 κ.λ.π. όπου προσέξτε όμως όταν γίνεται η κλήση η Java μετατρέπει τις τελείες σε καθέτους "\" και ψάχνει τις κλάσσεις στο java\applets\class1 κ.λ.π. Επειδή η διαδρομή ξεκινάει ΧΩΡΙΣ κάθετο η θέση της κλάσης είναι σχετική και βρίσκεται πάντοτε εκεί που καθορίζει το σύστημα με την βοήθεια του Browser, στην συγκεκριμένη περίπτωση μέσα στον κατάλογο \programs\netscape. Μια συγκεκριμένη όμως έκδοση του Netscape είχε το πρόβλημα ότι ΤΟΠΟΘΕΤΟΥΣΕ ΚΑΘΕΤΟ ΚΑΙ ΣΤΗΝ ΑΡΧΗ. Έτσι στο παράδειγμα μας ζητούσε την κλάσση ένα στον κατάλογο \java\applets\class1, θεωρώντας ότι ο κατάλογος java είναι κάτω απ' τον κεντρικό (root). Με αυτό τον τρόπο έχοντας πρόσβαση στον κεντρικό κατάλογο θα μπορούσε να φορτωθεί ΟΠΟΙΟΔΗΠΟΤΕ ΑΡΧΕΙΟ ΟΠΟΥ ΚΑΙ ΑΝ ΒΡΙΣΚΟΤΑΝ, για παράδειγμα στο \windows\command. Φυσικά το πρόβλημα διορθώθηκε με το να μην μπαίνει κάθετος στην αρχή της κλάσσης.

6. Κριτική πάνω στο μοντέλο ασφάλειας της Java

Φυσικά όταν ένα καινούργιο μοντέλο ασφάλειας εμφανίζεται στο προσκήνιο αυτό είναι σίγουρο ότι θα δεχτεί κριτική. Και όσο παρουσιάζονται σφάλματα τόσο η κριτική αυτή γίνεται πιο έντονη και ο σκεπτικισμός μεγαλώνει. Αυτό που πρέπει να ξέρουμε και να παραδεχθούμε είναι ορισμένες αρχές πάνω στις οποίες θα προχωρήσουμε. Η πρώτη είναι ότι δεν υπάρχει σύστημα εκατό τις εκατό ασφαλές. Η δεύτερη είναι ότι δεν υπάρχει σύστημα φτιαγμένο από ανθρώπινο χέρι που να μην έχει σφάλματα. Επίσης όσοι ασχολούνται με την ασφάλεια ξέρουν τους γνωστούς συμβιβασμούς, ασφάλεια-ευχρηστία και ασφάλεια-κόστος, όπου στην πρώτη περίπτωση είναι αντιστρόφως ανάλογα και στην δεύτερη ανάλογα. Έχοντας όλα αυτά υπ' όψη και παραδεχόμενοι ότι ούτε η Java είναι πανάκια μπορούμε να παρακολουθήσουμε την σκέψη κάποιων ανθρώπων που ξέρουν από πρώτο χέρι την ασφάλεια.

Είναι γεγονός ότι το επίπεδο ασφάλειας που παρέχει η Java κάποιους θα τους ικανοποιήσει και κάποιους όχι. Είναι θέμα αυτού που αγοράζει την ασφάλεια να αποφασίσει αν αυτό που του προσφέρει η Java σ' αυτόν τον τομέα του κάνει ή όχι. Και το βασικότερο είναι να ξεπεραστεί ο "μύθος" περί απρόσβλητου από επιθέσεις προιόντος και να μπούν τα πράγματα στην πραγματική τους διάσταση.

Υπάρχει μια διχογνωμία σχετικά με το κατά πόσο μια λύση ασφάλειας που βασίζεται στο λογισμικό είναι καλύτερη από μια αντίστοιχη που βασίζεται στο υλικό.

Αν τελικά καθιερωθεί το μοντέλο του αμμοδοχείου, όπου τα applets περιορίζονται σε ένα συγκεκριμένο χώρο κάνοντας συγκεκριμένα πράγματα ίσως αυτό να αποτελεί και την καταδίκη τους. Το ζητούμενο είναι να μπορούν τα applets να δράσουν ελεύθερα προκειμένου να κάνουν κάτι πραγματικά χρήσιμο και αποδοτικό. Αν σήμερα κάνουν κάποια απλοική εργασία, αύριο θα μπορούν να εκτελούν ένα πρόγραμμα εντοπισμού ιών, να εγκαθιστούν μια εφαρμογή, να κάνουν ένα τυπικό έλεγχο στο σύστημα μας για σφάλματα, με την προυπόθεση όμως ότι θα είναι ασφαλή.

Μια άλλη άποψη είναι ότι η προσβολή του συστήματος ασφάλειας της Java είναι ολοκληρωτική. Έτσι και κάποιο applet ξεφύγει απ' το αμμοδοχείο του δεν υπάρχει άλλη γραμμή άμυνας και το σύστημα είναι στην διάθεση του εισβολέα. Ακόμη μια καλή επίθεση μπορεί να επιτευχθεί αν βασίζεται στο στοιχείο του αιφνιδιασμού. Δηλαδή αν ο εισβολέας προβλέψει κάτι που δεν έχει προβλέψει ο σχεδιαστής. Γενικά είναι ενδιαφέρουσα η ιστορία όπου οι πιο σημαντικές διεισδύσεις σε συστήματα δεν έγιναν με την εκμεταλλεύση κάποιου

σφάλματος αλλά αντίθετα κάποιου σχεδιαστικού χαρακτηριστικού του οποίου την κακή χρήση δεν μπορούσε ούτε ο ίδιος σχεδιαστής να φανταστεί.

Ας τελειώσουμε με κάτι θετικό και αισιόδοξο. Στο παρελθόν πολλά δίκτυα και συστήματα προσπαθούσαν να διατηρήσουν την ασφάλεια τους κρύβοντας την εσωτερική τους δομή και τις πολιτικές του συστήματος. Αυτή η πρακτική υπέθετε ότι αν το σύστημα παρουσιαζόταν σαν "μαύρο κουτί" κανένας δεν θα έκανε την προσπάθεια να ανακαλύψει τις κρυμμένες του αδυναμίες. Η πράξη έδειξε το άτοπο αυτής της υπόθεσης και απέδειξε ότι τελικά το "μαύρο κουτί" δεν ήταν τόσο μαύρο τελικά. Αυτό είναι ακόμη πιο πιθανό για επιτυχημένα εμπορικά προιόντα όπου πολλοί ανθρωποι γνωρίσουν την εσωτερική δομή του αλλά και όπου η ανταμοιβή απ' το "σπάσιμο" του συστήματος είναι σαφώς επικερδής.

Η εταιρεία Sun ακολούθησε τελείως διαφορετική πορεία και δημοσιοποίησε όλες τις λεπτομέρεις του μοντέλου ασφάλειας της Java. Αυτό περιελάμβανε τις προδιαγραφές της γλώσσας, το αμμοδοχείο και την πλήρη πηγαία υλοποίηση. Αυτή η προσέγγιση, ασφάλεια μέσα απ' την δημοσιοποίηση, είχε την πρόθεση να ενθαρύνει τους ανθρώπους του ερευνητικού τομέα της ασφάλειας να εξετάσουν το μοντέλο της Java και να αναφέρουν οποίο σφάλμα

βρήκαν. Τα σφάλματα αυτά θα μπορούσαν να διορθωθούν πριν δημιουργήσουν σοβαρά προβλήματα στο Ιντερνέτ. Επίσης αυτή η προσέγγιση δίνει την ευκαιρία σε κάθε οργανισμό να μελετήσει το μοντέλο ασφάλειας με λεπτομέρεια και να κάνει μια εκτίμηση ορθολογική των πιθανών ρίσκων σε σχέση με τα ωφέλη απ' την χρήση της Java.

7. Το μέλλον της Java Security

7.1 Ψηφιακές υπογραφές και αρχεία JAR.

Υπάρχει ένα είδος επίθεσης στο οποίο όλα τα δίκτυα είναι ευαίσθητα. Ονομάζεται "ο άνθρωπος στο μέσον" ή ο ενδιάμεσος. Λειτουργεί δε ως εξής, ένας πελάτης κάνει μια αίτηση σε έναν server. Χωρίς να αντιληφθούν οτιδήποτε ούτε ο πελάτης ούτε ο server, μια κακόβουλη εφαρμογή παρακολουθεί την αίτηση και περιμένει την απάντηση του server. Όταν αυτό γίνει, η εφαρμογή αυτή υποκλέπτει την απάντηση και την αντικαθιστά με μια δική της και ο πελάτης νομίζει φυσικά ότι αυτή είναι και η σωστή. Φανταστείται την ζημιά που θα γίνει εάν αυτό που περιμένει να λάβει ο πελάτης είναι ένα applet το οποίο πια κάνει ότι η κακόβουλη εφαρμογή του έχει υποδείξει να κάνει.

Η λύση σε ένα τέτοιο πρόβλημα είναι τα applets με υπογραφή. Όπως ακριβώς όταν αγοράζουμε ένα προιόν λογισμικού στο κουτί του και ξέρουμε τον κατασκευαστή του, έτσι ακριβώς και τα applets θα φέρουν μια υπογραφή, όχι φυσική αλλά ψηφιακή η οποία θα βεβαιώνει το "καλώς έχειν". Για να γίνει αυτό πρακτικά θα πρέπει ο κατασκευαστής του applet να τοποθετίσει τον κώδικα της Java και οποιαδήποτε σχετικά αρχεία σε ένα αρχείο JAR (Java Archive). Ακολούθως ο κατασκευαστής υπογράφει αυτό το αρχείο με ψηφιακή

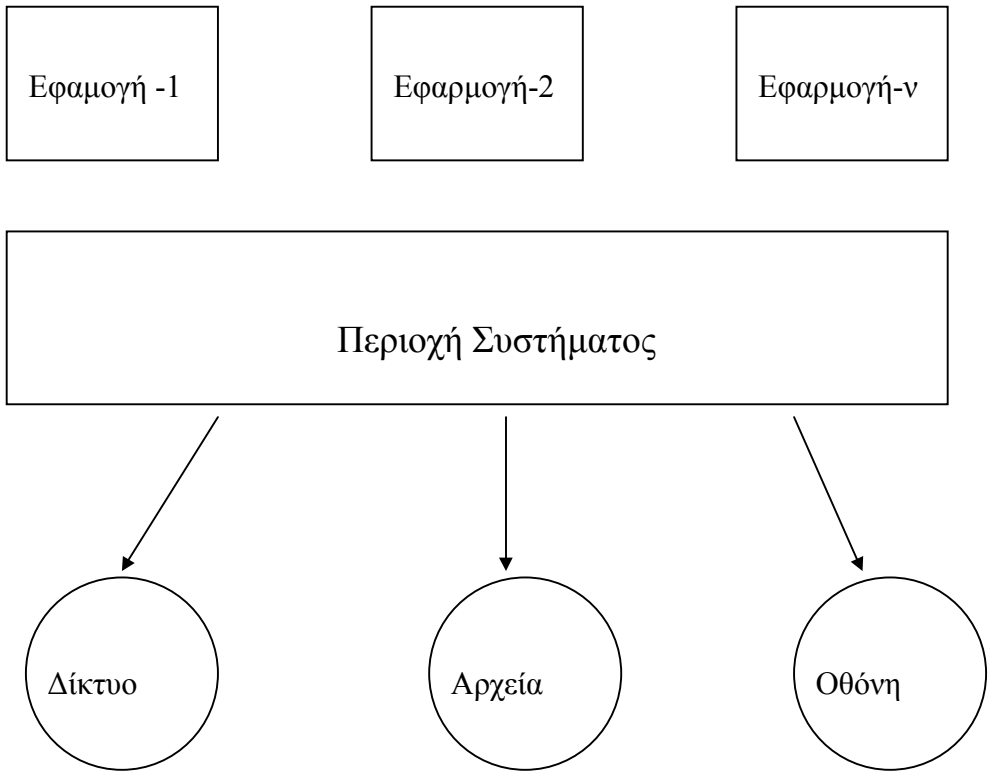
υπογραφή. Τέλος ο πελάτης ελέγχει την ταυτότητα του κατασκευαστή απ' την ψηφιακή υπογραφή.

7.2 Java Protected Domains (Προστατευμένες Περιοχές)

Ένα ουσιώδες στοιχείο στην εξέλιξη της Java Security είναι οι προστατευμένες περιοχές. Αυτές είναι συστατικά στα οποία έχουν πρόσβαση συγκεκριμένα αντικείμενα που ονομάζονται αρχές. Αρχή είναι οποιαδήποτε οντότητα σε έναν Η/Υ που είναι άξια εμπιστοσύνης και όπου μπορεί να δοθεί εξουσιοδότηση. Ένα παράδειγμα μιας τέτοιας αρχής είναι το αμμοδοχείο.

Εξ' ορισμού οι προστατευμένες περιοχές είναι εντελώς διαχωρισμένες. Η τυχόν αλληλεπίδραση τους μπορεί να γίνεται είτε με την βοήθεια ασφαλή κώδικα συστήματος, ή όταν σαφώς επιτρέπεται και απ' τις δύο προστατευόμενες περιοχές. Αυτό το μοναδικό χαρακτηρηστικό μπορεί να επεκτείνει το αμμοδοχείο στο σύστημα αρχείων, μια δυνατότητα ισχυρή αλλά και ευέλικτη. Για παράδειγμα δύο διαφορετικά applets μπορούν να γράψουν στην ίδια περιοχή του συστήματος αρχείων αλλά δεν μπορούν να αλληλεπιδράσουν μεταξύ τους. Αυτό θα μπορούσε να ρυθμίζεται απ' τον χρήστη ή να είναι προρυθμισμένο. Αυτός ο τύπος της διευκόλυνσης επεκτείνει τον έλεγχο της Java επιτρέποντας πολλαπλά και μοναδικά δικαιώματα για ξεχωριστές εφαρμογές.

Οι προστατευμένες περιοχές ανήκουν σε δύο κλάσεις, συστήματος και εφαρμογών. Σχεδιαστικά όλοι οι πόροι του συστήματος (αρχεία, συνδέσεις στο δίκτυο, οθόνη κ.λ.π) είναι προσβάσιμες μέσα από περιοχές του συστήματος. Αυτό φαίνεται στο σχήμα 7.1



Σχήμα 7.1

Θεωρητικά κάθε JVM χρειάζεται μόνο μια περιοχή συστήματος. Για να επιτύχουμε όμως μεγαλύτερη ασφάλεια η JVM μπορεί να εκκινήσει πολλές περιοχές συστήματος απ' τις οποίες κάθε μια θα είναι υπεύθυνη για μια ομάδα πόρων του συστήματος. Αυτό από μόνο του περιορίζει δραστικά τα προβλήματα που θα προκύψουν από προγραμματιστικά σφάλματα και αδυναμίες ασφάλειας. Έτσι τα

προβλήματα θα περιοριστούν σε μια περιοχή χωρίς παραπέρα συνέπειες. Απ' την άλλη, όλες οι προστατευμένες περιοχές είτε συστήματος είτε εφαρμογών μπορούν να θέσουν επιπλέον περιορισμούς στους εσωτερικούς τους πόρους.

7.3 Άλλες προβλέψεις ασφάλειας.

Όλες οι πολιτικές ασφάλειας που υπήρχαν μέχρι τώρα είναι διαθέσιμες μόνο σε προγραμματιστικό επίπεδο. Έτσι προκειμένου να επωφεληθούμε απ' αυτές θα έπρεπε να αγοράσουμε ειδικό κώδικα ή να τον παράγουμε εμείς, δηλαδή να επεκτείνουμε τις κλάσεις ασφαλείας. Μελλοντικές εκδόσεις της Java θα σχεδιαστούν για να επιτρέψουν τέτοιες πολιτικές να είναι πιο εύκολα εφαρμόσιμες απ' τους χρήστες μέσα από έτοιμους πίνακες.

Προκειμένου να αντιμετωπιστούν μη-ασφαλή μέσα μεταφοράς (το Ιντερνέτ) η ίδια η Sun δημιούργησε ένα πρωτόκολλο το SKIP (Simple Management Internet Protocol) που επιτρέπει στα μέρη που συνομιλούν να συμφωνήσουν σε ένα μοντέλο κρυπτογράφησης το οποίο εγγυάται απόρρητες επικοινωνίες. Το Java Encryption Extension είναι ένα πρόσθετο module που παρέχει έναν εύκολο τρόπο στους προγραμματιστές και στους διαχειριστές συστήματος να εφαρμόσουν εξελιγμένες τεχνικές κρυπτογράφησης στα προγράμματα Java.

Μια σημαντική πτυχή στην ασφάλεια συστημάτων είναι η "παρακολούθηση" (auditing) του συστήματος. Αν ο διαχειριστής ασφάλειας παρατηρήσει ή υποπτευθεί μια επίθεση η παρακολούθηση παρέχει ένα ιστορικό της επίθεσης που βοηθάει να καταλάβουμε τι πραγματικά συνέβη. Αυτή την στιγμή η Java δεν έχει τυποποιήσει αυτήν την διαδικασία παρακολούθησης. Γίνονται όμως προσπάθειες για να τυποποιηθούν κάποια τέτοια χαρακτηριστικά.

Ένα απ' τα προβλήματα που μπορεί να παρουσιαστεί στους χρήστες όταν χρησιμοποιούν τις προστατευμένες περιοχές είναι η πιστοποίηση των μηχανών. Αυτό σημαίνει πως ο κώδικας που εκτελείται σε μια περιοχή που βρίσκεται σε ένα απομακρυσμένο μηχάνημα μπορεί να αποκτήσει πρόσβαση σε τοπικό μηχάνημα. Το τοπικό μηχάνημα θα δεί τον απομακρυσμένο κώδικα σαν μη-ασφαλή και έτσι θα του αρνηθεί την πρόσβαση στους τοπικούς πόρους. Για να ανταπεξέλθει σ' αυτόν τον περιορισμό, μια προστατευμένη περιοχή μπορεί να αποδόσει "ταυτότητα" (ID) που επιτρέπει στο τοπικό μηχάνημα να πιστοποιήσει το απομακρυσμένο μηχάνημα και να επιτρέψει στον κώδικα του να εκτελεστεί.

Αυτή την στιγμή η χρήση πιστοποίησης απαιτεί ξεχωριστή κωδικοποίηση η οποία χρειάζεται πολύ χρόνο για να υλοποιηθεί. Για την απλοποίηση αυτού του προβλήματος, η Java θα προσφέρει

διεπαφές προκειμένου να υπάρχει ομοιόμορφη πρόσβαση σε κώδικα πιστοποίησης τρίτων κατασκευαστών, όπως το X.509v3 και θα παρέχει on-line server Αρχής Πιστοποίησης (Certificate Authority).

7.4 JavaBeans, Servlets και Java Chips

Τέλος οι ίδιες οι τεχνολογικές εξελίξεις στην Java οι οποίες δεν έχουν άμμεσο στόχο την ασφάλεια προσφέρουν επιπλέον μηχανισμούς ασφάλειας. Τα JavaBeans είναι συστατικά γραμμένα σε Java τα οποία μπορούν να λειτουργήσουν ως middleware σε μια αρχιτεκτονική τριών επιπέδων (three-tier) που είναι η εξέλιξη του client-server computing. Αυτά παρέχουν την ίδια ασφάλεια με τα applets αλλά στην εξέλιξη τους ενσωματώνουν οτιδήποτε καινούργιο σε θέματα ασφαλείας βγαίνει. Τα servlets είναι κομμάτια κώδικα σαν τα applets τα οποία εκτελούνται όμως στον server και όχι στον client. Κατά συνέπεια δεν επηρεάζουν σε τίποτα το τοπικό μηχάνημα, είναι διαφανή στον χρήστη και παρέχουν την μέγιστη ασφάλεια. Τέλος η Sun έχει προγραμματίσει την κατασκευή ειδικών επεξεργαστών που θα εκτελούν τον κώδικα της Java σε native-mode χωρίς την παρουσία JVM. Αυτό ενώ γίνεται με στόχο την βελτίωση της ταχύτητας εκτέλεσης των προγραμμάτων θα επιτρέψει και την δημιουργία hardware μηχανισμών ασφαλείας οι οποίοι κατά κανόνα είναι πιο ανθεκτικοί στις επιθέσεις απ’ ότι το software.

ΑΛΦΑΒΗΤΙΚΟ ΛΕΞΙΚΟ ΟΡΩΝ

Applet Viewer: είναι λογισμικό που μας επιτρέπει την εκτέλεση ενός applet χωρίς να υπάρχει ένας java-enabled Browser.

Applet: Μικρά εκτελέσιμα προγράμματα που γράφονται στην γλώσσα προγραμματισμού Java και ενσωματώνονται σε μια σελίδα υπερκειμένου.

ByteCode: μια γλώσσα ψευδομηχανής (όχι πραγματικού τύπου επεξεργαστών) χαμηλού επιπέδου.

Class: είναι ένας γενικευμένος πίνακας για μια ομάδα αντικειμένων (objects) με παρόμοια χαρακτηριστικά

Garbage Collector: το χαρακτηριστικό εκείνο της Java που απελευθερώνει κομμάτια μνήμης που δεν χρησιμοποιούνται πλέον.

HotJava: είναι ο Web Browser της εταιρείας Sun και ήταν ο πρώτος που μπορούσε και εκτελούσε applets.

Instance: είναι μια συγκεκριμένη διακριτή αναπαράσταση μιας class. Τα instances και τα objects είναι το ίδιο πράγμα.

JavaBean: τυποποιημένα συστατικά (components) της εταιρείας JavaSoft, που υποστηρίζονται ταυτόχρονα απ' τις εταιρείες Sun, Netscape και IBM.

JavaScript: είναι ένα εμπορικό προιόν της Netscape μια γλώσσα scripting που μοιάζει με την Java.

JavaSoft: η εταιρεία που δημιουργήθηκε απ' την Sun ειδικά για την προώθηση και αντιμετώπιση και πώληση του προιόντος που λέγεται Java.

JIT Compiler: Just in Time Compiler, Προιόν της Microsoft που κάνει την εκτέλεση προγραμμάτων της Java πιο γρήγορη.

OAK: είναι το αρχικό όνομα του project της Java.

Opcode: εντολές της εικονικής μηχανής (virtual machine).

Run-time Interpreter: ένας διερμηνευτής (interpreter) ο οποίος μεταφράζει ByteCode σε φυσική γλώσσα μηχανής.

Servlet: το αντίστοιχο του applet μόνο που αυτό δεν εκτελείται στο τοπικό μηχάνημα αλλά στον server στον οποίο είναι εγκατεστημένο.

Βιβλιογραφία

Βιβλία

1. Java Security, Hostile Applets, Holes, and Antidotes, Gary McGraw and Edward W. Felten, Willey 1997

2. Learn Java in 21 Days, Lemay and Perkings, Sams Publishing, 1997

3. Java Network Security , Robert MacGregor , Prentice Hall 1998

Περιοδικά

Περιοδικό Byte

Ιαν. 1996, “Wired on the Web” σελ. 77

Απρ. 1996, “A Hot cup of Java” σελ. 129

Ιουν. 1996, “How Java can pay the rent” σελ. 40

Ιαν. 1997, “Java Security and Type Safety” σελ. 63

Ιαν. 1997, “Today the Web, tomorrow the Word” σελ. 77

Μάιος. 1997, “Avoiding Hostile Applets” σελ. 89

Ιουν. 1997, “Java Servlets” σελ.115

Αυγ. 1997, “Persistent Java”, σελ 108

Σεπτ. 1997, “ActiveX Demystified”, σελ 56

Ιαν. 1998, “Making Java Development Jsafe”, σε 117

Web Sites

http://www.javasoft.com

http://www.cert.org

http://www.cs.princeton.edu/sip