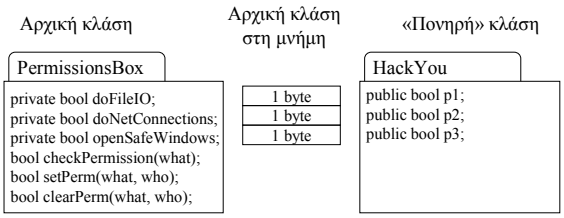
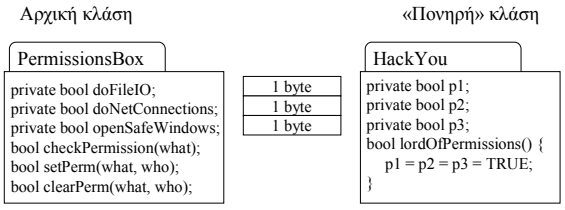
Ασφάλεια τύπων δεδομένων

- Δεν υπάρχουν επικίνδυνες μετατροπές δεδομένων
- Δεν καλούνται λάθος μέθοδοι



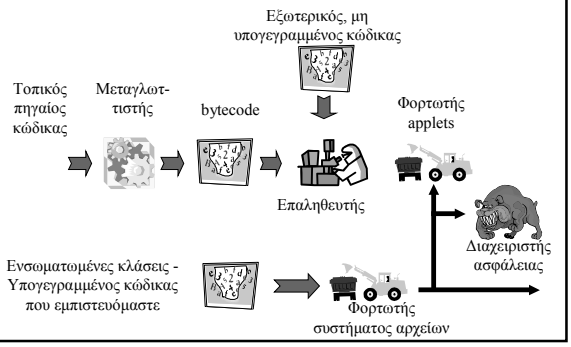
PermissionsBox a; ((HackYou)a).p1 = TRUE;

Ασφάλεια τύπων δεδομένων

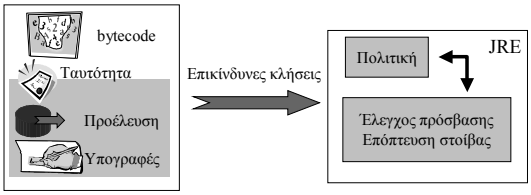


PermissionsBox a; ((HackYou)a).lordOfPermissions();

Κλάσεις ασφάλειας – Java 1.x



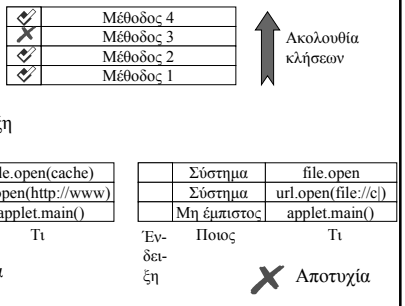
Κλάσεις ασφάλειας – Java 2



Έλεγχος πρόσβασης και επόπτευση στοίβας

- Χρειάζεται λεπτομερέστερο μοντέλο από το SandBox
  - Κώδικας βιβλιοθήκης καλεί κώδικα χρήστη
    - » *Όταν παραληφθούν δεδομένα από το δίκτυο κάλεσε την τάδε διαδικασία*
  - Διαδικασίες συστήματος μπορεί να έχουν διαφορετικά προνόμια, ανάλογα με το ποιος τις κάλεσε και πώς

Έλεγχος πρόσβασης και επόπτευση στοίβας



Έλεγχος πρόσβασης και επόπτευση στοίβας

- Η επόπτευση στοίβας χρησιμοποιεί ένα πιο λεπτομερές μοντέλο παραχώρησης-ανάκλησης δικαιωμάτων
  - checkPrivilege()
    - » Ελέγχει αν υπάρχουν τα δικαιώματα χρήσης
  - enablePrivilege()
    - » Δίνει δικαίωμα χρήσης και στους επόμενους
  - disablePrivilege()
    - » Αφαιρεί από τους επόμενους το δικαίωμα χρήσης
  - revertPrivilege()
    - » Αντιστρέφει το δικαίωμα

Νέοι μηχανισμοί ασφάλειας στη Java 2

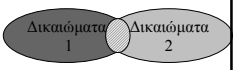
- Ταυτότητα → (προέλευση, υπογραφή)
  - Ουσιαστικά μας πληροφορεί ποιος «εγγυάται» για τον κώδικα
- Προνόμια
  - » java.io.FilePermission → πρόσβαση σε αρχεία
  - » java.net.SocketPermission → πρόσβαση στο δίκτυο
  - » java.lang.PropertyPermission → ιδιότητες Java
  - » java.lang.RuntimePermission → πόροι περιβάλλοντος εκτέλεσης
  - » java.security.NetPermission → διαδικασία πιστοποίησης
  - » java.awt.AWTPermission → πόροι γραφικών, π.χ. παράθυρα
  - Δημιουργία προνομίων μέσω κλήσεων
    - » p = new FilePermission("/applets/tmp/scratch", "read");
    - » = new SocketPermission("www.di.uoa.gr:0-1023", "connect")

Νέοι μηχανισμοί ασφάλειας στη Java 2

- Προνόμια (συνέχεια)
  - Σχέση συνεπαγωγής p1 ⇒ p2
    - » FilePermission("/tmp/*", "read") ⇒ FilePermission("/tmp/scratch", "read")
- Πολιτική
  - Χορήγηση *προνομίων* σε συγκεκριμένες *ταυτότητες*

```
grant CodeBase "http://www.di.uoa.gr/java", SignedBy "*" {
   permission java.io.FilePermission "read,write", "/tmp/java/*";
   permission java.net.SocketPermission "connect", "*.uoa.gr";
}
```

- Προνόμια και πολλαπλές υπογραφές*: προσθετική πολιτική*



Δομικές επεκτάσεις στη Java για ασφάλεια

- Η επόπτευση στοίβας είναι αποτελεσματική, αν θυμόμαστε να ανακαλούμε τα προνόμια

```
<normal code>
try {
   AccessController.beginPrivileged();
   <επικίνδυνος κώδικας>
} finally {
   AccessController.endPrivileged();
}
<more normal code>
```

Δομικές επεκτάσεις στη Java για ασφάλεια

- Ένθετες κλάσεις

```
somemethod() {
   final String lib = "awt";
   <normal code>
   AccessController.doPrivileged(new PrivilegedAction() {
      public Object run() {
            <επικίνδυνος κώδικας που χρησιμοποιεί το lib>
            return null;
      }
   });
   <more normal code>
}
```

- Η υποχρέωση δήλωσης των μεταβλητών ως *final*, μολονότι ενοχλητική, είναι σημαντική

Ασφάλεια στη Java: Οι εχθροί

- Κακόβουλες εφαρμογές
  - λίγο έως πολύ ενοχλητικές
  - κυκλοφορούν ελεύθερα στο διαδίκτυο
- Εφαρμογές για επίθεση
  - λίγο έως πολύ καταστροφικές
  - μέχρι τώρα έχουν περιορισθεί στα εργαστήρια

Java - Κακόβουλες Εφαρμογές

- Τι κάνει μια κακόβουλη εφαρμογή
  - Άρνηση εξυπηρέτησης
  - Παραβίαση της ιδιωτικότητας
  - Καθαρή ενόχληση

Java - Ενοχλητικές Εφαρμογές

- Ενόχληση

```
public class NoisyApplet extends java.applet.Applet implements Runnable {
    public void init() {bark = getAudioClip(getCodeBase(), "bark.au"); }
    public void start() {
      if (noisethread == null) {
            noisethread = new Thread(this); noisethread.start();}}
    public void stop() {
      // if (bark != null) bark.Stop();
      if (noiseThread != NULL) {noisethread.stop(); noisethread = null; }}
    }
Ή
public void run() {
    try { //ο κανονικός κώδικας}
     catch(ThreadDeath td){System.out.println("I'm Invincible!");}
// Το ζόμπι ξαναγυρνά!
     finally{Thread.reborn = new Thread(, "new"); reborn.start();}
}
```

Java - Άρνηση εξυπηρέτησης

1. Δημιουργούμε ένα νήμα ελέγχου με μέγιστη προτεραιότητα
2. Επανορίζουμε τη μέθοδο stop να μην κάνει τίποτε
3. Κάνουμε κάτι απλό και ανώδυνο στην αρχή, μετά παύουμε την εκτέλεση για λίγο
4. Μετά την παύση, εκτελούμε κάποιο βρόχο που απλά καταναλώνει υπολογιστική ισχύ

- Μπορεί επίσης να καταναλώνει μνήμη
- Με επαρκώς κακό προγραμματισμό, ένα «κανονικό» πρόγραμμα μπορεί να έχει τα ίδια αποτελέσματα

Java – Παράθυρα χωρίς προειδοποίηση

```
try {
    littleWindow = new bigFrame("SomeName");
    littleWindow.resize(1000000, 1000000);
    littleWindow.move(-1000, -1000);
    littleWindow.show();
} catch (OutOfMemoryError o) { repaint();}
class bigFrame extends Frame { // constructor method
Label 1;
bigFrame(String title) {
    super(title); setLayout(new GridLayout(1, 1));
    Canvas whiteCanvas = new Canvas();
    whiteCanvas.setBackground(Color.white); add(whiteCanvas);
} }
```

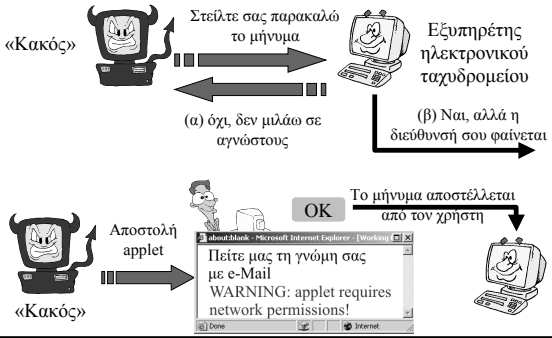
- Εντός βρόχου, ιδεώδες για «πάγωμα» του υπολογιστή
- Απόκρυψη της προειδοποίησης πρώτου παραθύρου → στο δεύτερο παράθυρο που θα ζητά το συνθηματικό, ο χρήστης δεν θα έχει προειδοποιηθεί

Java – Αξιοποίηση της ΚΜΕ
(για κάτι που μάλλον δεν θα θέλαμε)

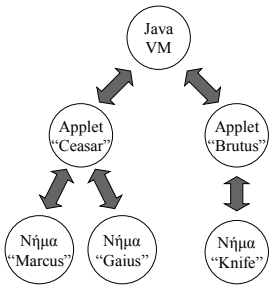
- Ο «κακός» δημιουργεί ένα νήμα ελέγχου χαμηλής προτεραιότητας που κάνει υπολογισμούς και στέλνει τα αποτελέσματα στον ιδιοκτήτη του

```
public class cycleStealer extends java.applet.Applet
implements Runnable {
    public void start() {
    if (myWorkThread == null) {
          myWorkThread = new Thread(myWork, “);
          myWorkThread.start();}}
    public void myWork() {
          while(TRUE) {
          work = getChunkFromMaster(); results =  doChunkOfWork();
          sendToMaster(results);
}}}
```

Java – Αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου



Java – Ο θάνατος του νήματος




```
Applet Brutus {
      findApplet(Ceasar);
      for (victim in Ceasar.threads) {
            knife(victim);
      }
}

private static void Knife (Thread t) {
      if ((t.name != "Brutus") &&
            (t.name != "Knife")) t.stop();
}
```

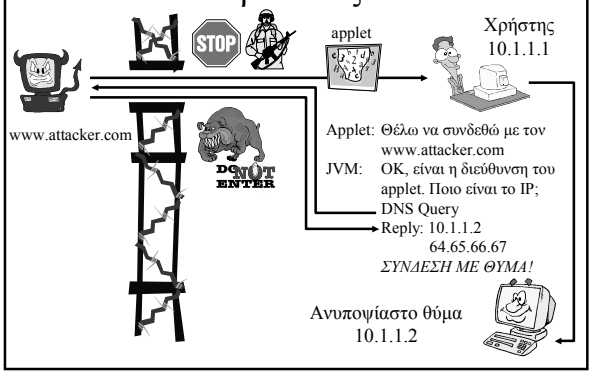
Java – Αξιοποιώντας τα κενά ασφάλειας

- Σφάλματα υλοποίησης ή προδιαγραφών;
  - Οι προδιαγραφές δεν είναι πάντα ακριβείς και σαφείς
  - ούτε πλήρεις
  - ούτε σωστές
- αλλά είναι μια καλή αφετηρία.
- Πολλά ζητήματα είναι απλώς σφάλματα υλοποίησης

Java – Ξεπερνώντας το firewall

- Μια εφαρμογή Java μπορεί να συνδεθεί μόνο προς τον εξυπηρέτη από τον οποίο προέρχεται
- Ο εξυπηρέτης αναγνωρίζεται από το όνομα DNS π.χ. www.attacker.com
- Ο «κακός» στέλνει πολλαπλές διευθύνσεις IP ως απάντηση στην ερώτηση «ποια είναι η διεύθυνση IP του www.attacker.com;»
- Στις οποίες περιλαμβάνεται η διεύθυνση του «θύματος»

Java – Ξεπερνώντας το firewall



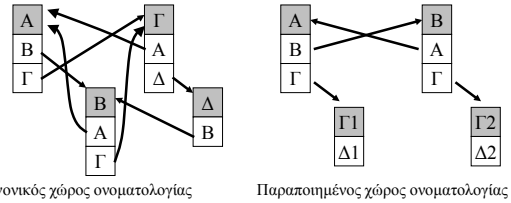
Java – Τελείες και κάθετοι

- Κλάσεις που φορτώνονται από τον δίσκο θεωρούνται «έμπιστες» (Java 1)
- Η Java χρησιμοποιεί κλάσης της μορφής java.lang.RuntimePermission
- Το αρχείο της κάθε κλάσης παράγεται από το όνομά της αντικαθιστώντας τις τελείες με καθέτους
- Το Netscape 2.01 επέτρεπε κλάσεις που ξεκινούσαν με τελεία
  - Κλάση: *.windows.temp.cache.file032*
  - Αρχείο: *\windows\temp\cache\file032*

  με πλήρη δικαιώματα!

Java – Οι φορτωτές κλάσεων

- Ένα πρόβλημα στον επαληθευτή επέτρεπε σε μία κλάση να μην καλεί τη συνάρτηση κατασκευής της υπερκλάσης της
- Τα applets κανονικά δεν μπορούν να φτιάξουν φορτωτές κλάσεων...
- αλλά ο έλεγχος γινόταν σε μία συνάρτηση κατασκευής υπερκλάσης
- Εχοντας φτιάξει έναν φορτωτή κλάσεων, το applet έφτιαχνε ένα παραποιημένο χώρο ονοματολογίας



Κανονικός χώρος ονοματολογίας

Παραποιημένος χώρος ονοματολογίας

Java – Βελτιστοποιήστε σωστά

```
interface Inter {void f();}

 class Secure implements Inter {
   private void f();
 }

 class Dummy implements Inter {
   public void f();
   static void attack() {
     Inter inter[2] = {new Dummy(), new Secure() };
     for(int j=0; j<2; ++j)     inter[j].f();
   }
 }
```

JAVA – Το πακετάρισμα μετράει

- Πακέτο = περιοχή ονοματολογίας + προστασία
  - di.uoa.gr.aClass ≠ netscape.aClass
  - Εξ ορισμού μεταβλητές και μέθοδοι είναι προσπελάσιμα στις κλάσεις του πακέτου
- Κατά τη φόρτωση, ο διαχειριστής ασφάλειας ελέγχει αν μία κλάση επιτρέπεται να ενταχθεί στο πακέτο που ζητά
- Σφάλμα στον Explorer επέτρεπε σε οποιαδήποτε κλάση να ενταχθεί στο πακέτο ms.com

JAVA – Πείτε μου το IP σας

- Η προστασία της διεύθυνσης του χρήστη είναι σημαντική
- Προστατεύεται από εξυπηρέτες αντιπροσώπευσης
- Όχι όμως από μια άσχημα υλοποιημένη InetAddress.getLocalHost()
  - Υπάρχουν τουλάχιστον δύο διευθύνσεις σε κάθε υπολογιστή, η 127.0.0.1 και η «κανονική»
  - Επιστρέφοντας την κανονική, είναι πλέον δυνατόν να αποσταλεί στον «κακό»

JAVA - Ανίχνευση θυρών

- Φροντίζουμε να τοποθετηθεί ένα αρχείο Java στην cache, με λειτουργικότητα ανιχνευτή θυρών
- Με ένα προσεκτικά φτιαγμένο URL φορτώνουμε το αρχείο java, που έρχεται πια από το δικό μας μηχάνημα
- ... άρα μπορεί να ανιχνεύσει τις θύρες ...
- και σε συνεργασία με ένα πρόγραμμα που έρχεται από το δικτυακό τόπο του «κακού» τα αποτελέσματα επιστρέφουν σ' αυτόν.

Java – Προσοχή στις υπογραφές

- Class.getsigners() επιστρέφει έναν πίνακα με τις υπογραφές της κλάσης
- ... που είναι ο ίδιος με αυτόν που χρησιμοποιεί το σύστημα ασφάλειας
- ... και που το applet μπορεί να τροποποιήσει προσθέτοντας υπογραφές
- ... μέχρις ότου να πάρει όλα τα προνόμια που θέλει

JAVA – Οι ανακατευθύνσεις

```
myURL = GetCodeBase() + '/cgi-bin/redirect?target=file:///etc/passwd';
Image img = getImage(new URL(myURL));
```

- Πληροφορίες από το μηχάνημά μας αποστέλλονται σε άλλους
- Πληροφορίες από μηχανήματα που μας εμπιστεύονται αποστέλλονται σε άλλους
- Ο υπολογιστής μας δρα ως ενδιάμεσος σε επιθέσεις προς τρίτους
- Προσκόμιση κώδικα επίθεσης από άλλες δικτυακές περιοχές

Java – Ηθικό δίδαγμα

- Η Java έχει σχεδιασθεί με την ασφάλεια κατά νου
  - αλλά κάποια πράγματα έχουν ξεφύγει
  - ή υλοποιούνται λάθος
- Έχουν γίνει πολλά βήματα προόδου
- Η ίδια η ομάδα της Java μιλά πια για «διαχείριση κινδύνων» παρά για «ασφάλεια»

JAVA – Κατευθύνσεις για εξέταση

- Δυνατότητα για τήρηση ημερολογίων
  - Επιτρέπουν την αποτίμηση του μεγέθους της ζημιάς
  - Παρέχουν ενδείξεις για το το πρέπει να προσέχουμε στο μέλλον
  - Μπορούν να αποτελέσουν είσοδο για συστήματα ανίχνευσης εισβολών
  - Αποτελούν αποδεικτικό στοιχείο στα δικαστήρια
- Αφηρημένα συντακτικά δένδρα έναντι bytecode
- Αντίστροφη μεταγλώττιση και επαναμεταγλώττιση για προστασία από τεχνητά κατασκευασμένα προγράμματα