New Protection Paradigms

- Sand-boxing
- System call trapping
- Java “introspection”

Traditional OS Protection

- Process based
- Each process is assigned a UID, access-control by UID
- Problems:
  - difficult to support tightly-collaborating modules in different protection domain
  - UID is too coarse-grained to be used as a “capability”

Solution 1: Sand-boxing

- Motivation: fault isolation by processes is expensive
- Fault isolation within a single address space
  - put code and data of a distrusted module into fault domain, a logically separate portion of the address space
  - modify the code to prevent the module from writing or jumping to an address outside its domain

Details of Sand-boxing

- Two segments: code and data
  - Addresses in a segment has the same upper address bits
- “sand-boxing” unsafe instructions
  - address checking and use of dedicated registers
  - or, simply set the upper address bits
- Special arbitration code for system calls

Alternative 2: Syscall Trapping

- Some OS provide capabilities for a parent process to intercept children’s system calls
  - ptrace
  - /proc
- Alternative security policies can be implemented through the mechanism

Java Security Mechanisms

- Strongly typed language
  - type safe (mostly statically verified, with some dynamic checking)
  - no pointers, every reference to an object can be controlled

Bugs in Early Java Security Mechanisms

- denial of service attacks
- covert channels
- unnecessarily exposed information
  - storage information used in hashcode
  - buffer overflow
- DNS weaknesses
- setting public proxy variable
- inter-applet security

Language and Byte-code differences

- example:
  - class CL extends ClassLoader {
    CL() { try {super();} catch(Exception e){}
    }
  - defeats type safety
- Illegal package names

New Java Security Architectures

- Digital signatures as principals
- Extended stack introspection:
  - a principal is attached with each class
  - registry of targets: (principal, string-name)
  - policy engine: access matrix
  - enablePriviledge(target) and checkPriviledge(target)
- Type hiding: class loader return subclasses