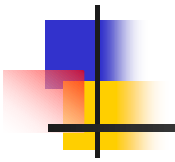
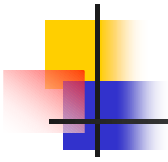
Extensible Security Architectures for Java



"... [With] software-based protection we can allow for more extensible security models ..."
Written by D Wallach etl-all
Presented by David Waters

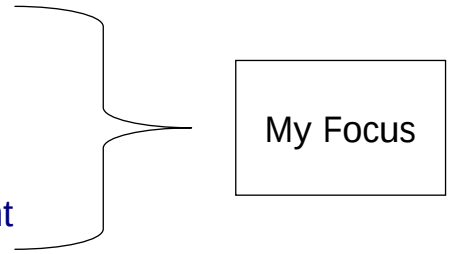


Goals

- To find extensible security systems
- That uses Software-based Methodologies
- And the Secure Services concept

Paper Overview

- Mobile code needs flexible security
  - Software based vs. hardware based methods
  - Memory protection vs. secure services
  - Possible solutions
    - Capabilities
    - Stack introspection
    - Namespace management
  - Evaluation
    - Criteria
    - Results



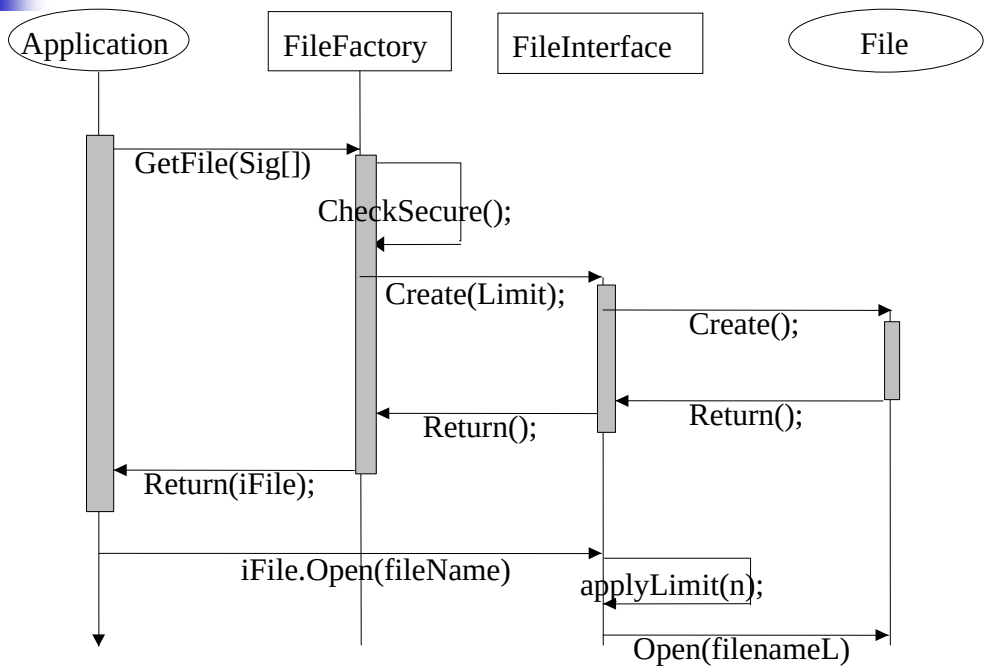
Current Environment(Java)

- Trusted/untrusted (local/remote) code can co-exist on the same JVM (and call each other).
- Java must be able to determine who initiated this call.
  - Reference to its ClassLoader.
  - Frame stack has reference to thread.
  - These combined mean that Java can search for remote code on the call stack.
- The security manager does just that.(Badly)

Capabilities

- Based on unforgeable references to a controlled resource.
  - "Any program which has a capability must have been permitted to use it."
- Programmes must explicitly request a capability to gain access.
  - (A good way of doing this in Java is through the factory pattern).
- Non-public Constructors.

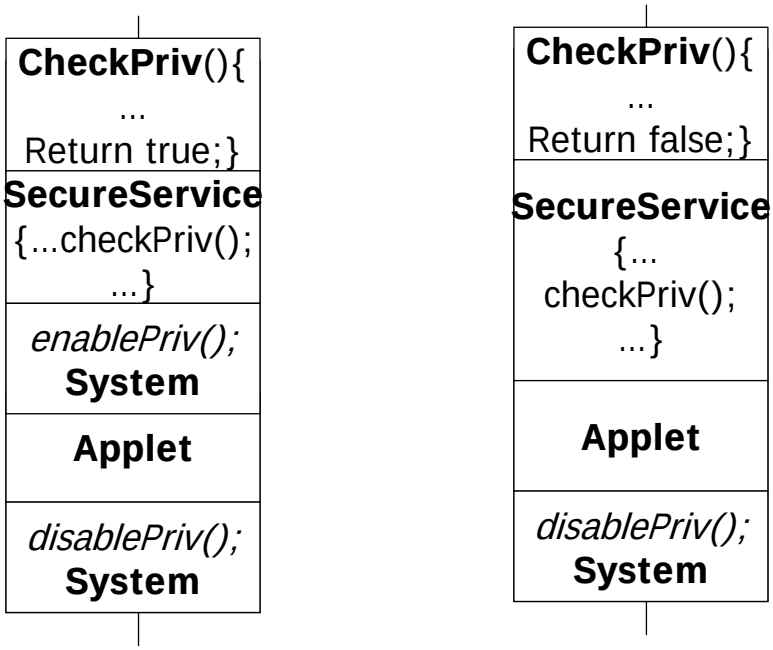
Capabilities



Extended Stack Introspection.

- (Used by both Netscape 4+ and IE 4+).
- Based on Simple Stack Introspection.
- Privileges created in the stack frame.
- Standard calls.
  - To get access.
    - Client call enablePrivilege.
    - Service calls checkPrivilege.
    - Client calls disablePrivilege when done.

Extended Stack Introspection.



Name Space Management

- Achieves security by showing/hiding/substituting all sensitive classes.
- This is done by replacing the class loader with one that maps (principles, class requested) —–> (class they are allowed to access).

Name Space Management

Class Requested/ Principle	*MS*	*IBM*	*David*
Java.net.socket	Nil	Security.io.Socket	Java.net.Socket
Java.io.file	Nil	Security.io.file	Java.io.File
Java.net.server Socket	Nil	Nil	Java.net.Server Socket

Comparison

	Extended Stack Inspection	*Capabilites*	*Name Space Management*
Exsisting Code Changes (User level)	Nil	Some	Nil
(System Level)	Some	Extensive	Some
(Kernel [JVM])	Nil	Nil	Some
Run time Panilties	Minimal	EvenLess	None(some load time)
New Code Changes (UserLevel)	Minimal	Some	Nil

Questions??????

- What is wrong with the SandBox?
- Allow multiple signatures?
- How to resolve permissions?
- Are our choices now going to limit what can be done in the future?
- Which of the possible solutions is best?