Java and Security
Evolving The Security Model

Jim Roskind
Java Security Architect
jar@netscape.com

Netscape Communications Corporation

Overview

- Java Security Features
- Java Attack Methods
- Problems in Traditional Java SecurityManager
- Netscape's 3.0x Security Evolution
- Class Signing
- Netscape's 4.0x Security Evolution

Java Security Features

- **Class based security**
  - **private; protected; package isolation**
- **No direct memory manipulation**
  - **No pointer arithmetic, GC rather than malloc/free**
- **Type Safety**
  - **No evil casting**
- **Code source location known**
  - **“codebase”; classpath; Signed classes**
- **Caller is known**
  - **Interpreted language; call stack is unforgeable**

Java Attack Methods

- **Luring Attacks**
  - **System.out was not final**
- **Overly Large TCB Attacks**
  - **Human factors: Font attack**
- **Type System Attacks**
  - **Type name confusion**
- **System Wide Attacks**
  - **DNS lookup variability**
- **Implementation Flaw Attacks**
  - **Partial construction is not prevented**

Traditional Java SecurityManager Model

- Centralized SecurityManger
  - non-extensible base class
  - security semantics separated physically from coding semantics
- Class Granularity Privileging
  - binary (two state) trust model
  - classpath vs URL-loading partitions level of trust
    - » gigantic TCB
- Some Thread based security decisions
  - non-ergonomic: Programmers forget to turn off powers
  - luring attack problems

Netscape's 3.0x Security Evolution

- **Binary Trust grew to 3 state model**
  - **Clark Kent vs Superman vs Untrusted**
  - **setScopePermission() reduced TCB**
  - **Performance validation for 4.0x planned features**
- **CallerDepth added to check*() calls**
  - **semantics of checking was exposed to callers**
- **Misc "last resort" defenses established**
  - **Method-name isolation across class loaders**
  - **Multiple class-load precluded at lowest levels**

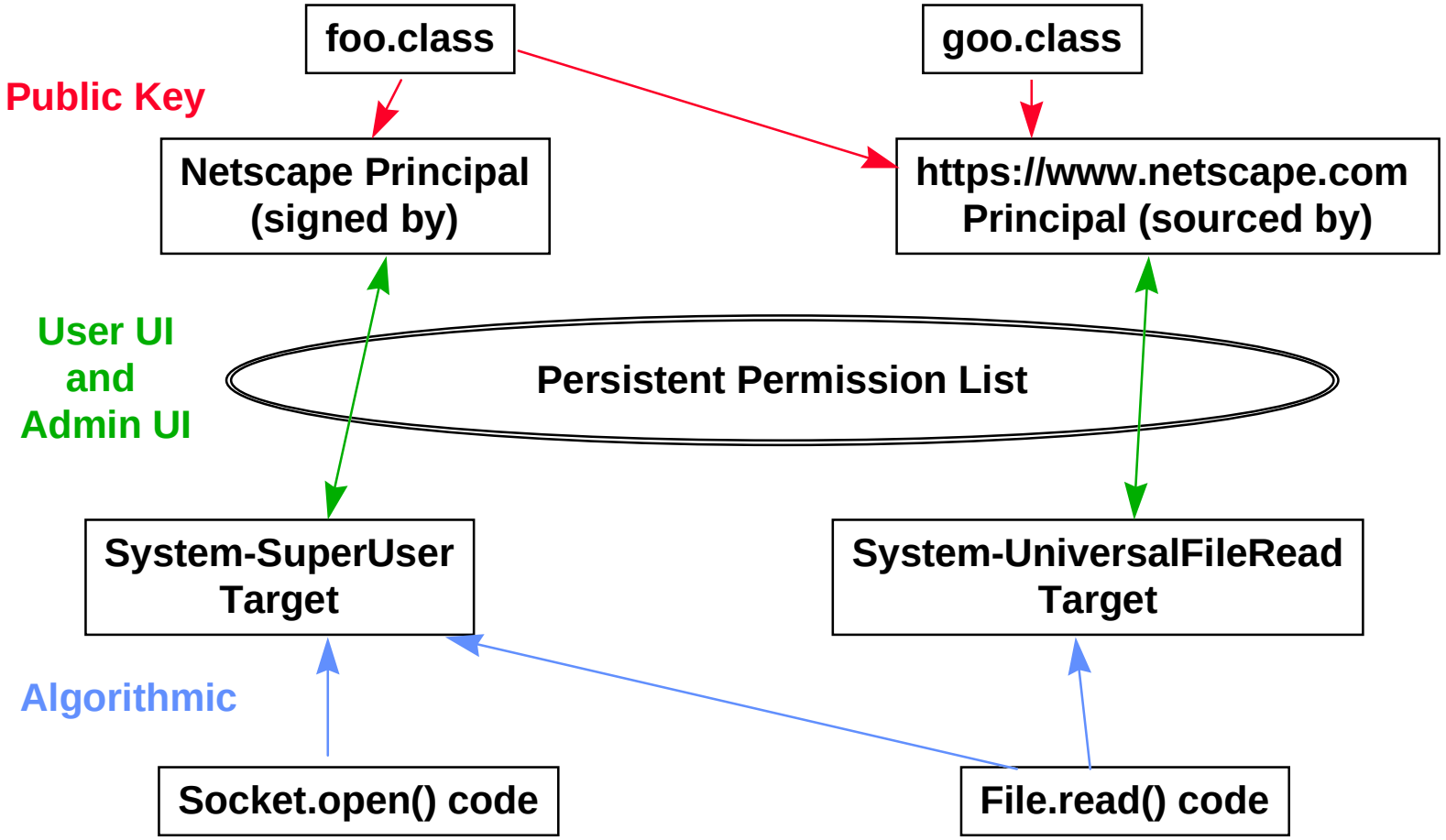
A Mediating code fragment: Demo of setScopePermission

```
// This class is loaded from classpath
// which makes it privileged (re: Clark Kent)

class Log ... {
    ...
    void write(String message) {
        ...
        SecurityManager.setScopePermission();
        stream.write(message); // TCB
        }

    ...
    }
```

Class Signing

- **Modeled after consumer trust in manufacturer**
  - **UI avoids user bombardment**
- **Signature establishes *only* Identity of author**
  - **User or Admin define access to targets for given principals**
- **Algorithms (few, and carefully reviewed) define relation between target and underlying resources**

Signed Class Trust Model



Netscape's 4.0x Signed Class Model

- **Centralized infrastructure**
  - Authoritatively maps classes to principals
  - Extensible list of targets
  - Safely records user's trust of principals
- **Statement Granularity Privileging**
  - Finer than "Class Granularity," requires "enable" of Priv
    - » Minimal TCB (grep for "enable" in code)
  - Multidimensional privileges (many targets)
    - » Principal of least privilege
- **Ergonomic stack empowerment**
  - Enablement of power is GC'ed with stack frame
  - Luring of powerful thread is neutralized

A Mediating code fragment: Demo of enablePrivilege

```
// This class is loaded from classpath
// which makes it privileged (re: Clark Kent)

class Log ... {
    ...
    void write(String message) {
        ...
        enablePrivilege("UniversalWrite");
        stream.write(message); // TCB
        }

    ...
    }
```

Summary

- Netscape is evolving the security model
- Ergonomics of security design are critical
- 99% of security comes from Java Sandbox
- 1% of security comes from highly visible TCB
- Signed Classes can safely extend the Sandbox
- Coming soon to a browser near you!