Java Security

HotJava to Netscape and Beyond

Drew Dean   Ed Felten   Dan Wallach

Department of Computer Science

Princeton University

4/5/96

Introduction

- Remote Code is Everywhere
- Java - A new language for executable content on the World Wide Web
- HotJava - A Web browser written in Java
- Netscape 2.0 - A Web browser with Java support

Remote Code

- Allows interactive content
  - hot imagemaps
  - animations
  - front-ends to shared games, databases, etc.

☞ *Dynamic* Web pages

Java: Buzzword Compliance

- Portable byte code interpreter
  - Load-time compilation possible
- Abstract Window Toolkit (portable runtime)
- Reusable class modules (*applets*)
- Safe for untrusted code
  - Restricted file system / network access
  - Restricted access to browser internals
  - Load and run-time checks

Outline

- Introduction
- Java Semantics
- HotJava and Netscape Security Flaws
- Security Analysis
- Application Requirements
- User Interface
- Future work / Conclusions

Java vs. C++

- Name spaces and packages
- `public`, `protected`, `private`, or public-within-current-package members
- Language-level threads and synchronization
- No pointers
- Garbage collection
- Type safety

Java Type Safety

- Load-time code verifier
- Run-time exceptions
  - Array bounds
  - Type compatibility
  - Local vs. remote code security checks

Java Remote Bytecode

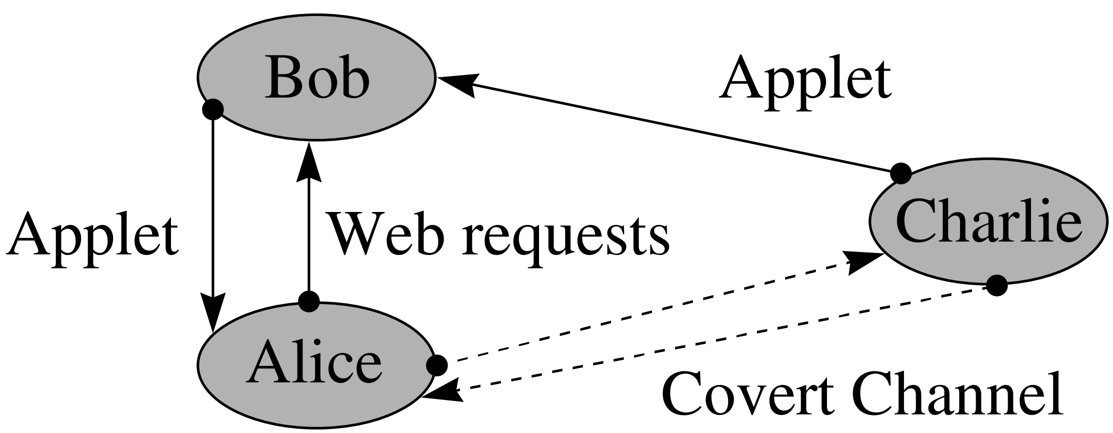
- ClassLoaders
  - ClassLoaders bind names to `Class` objects
  - Default ClassLoader can only load code from file system
  - Other ClassLoaders can access network, etc.
  - All classes tagged with their ClassLoader
- Applets may not create ClassLoaders

Java SecurityManager class

- ✓ New in Java beta versions
- Implements much of Java's security policy
- Runtime checks on dangerous methods (i.e. a *reference monitor* [Lampson])
  - Tamperproof, verifiable, always invoked
- Customizable (eventually)
- Can't be changed after browser initialization

HotJava 1.0α3 Security

- Covert Channels
  - URLs
  - DNS
  - Two- vs. three-party attacks



HotJava 1.0α3 Security

- Information available to leak
  - Mailcap files
  - **`System.getenv()`**
- Denial of service attacks
  - **`C:\TEMP`**
  - Acquire a system lock
- Man-in-the-middle attack
  - Set HTTP proxy server

Sun’s Response

- “Fixed in the next release”
  - **`getenv()`** is gone
  - **`accept()`** bug fixed
  - DNS/URL channels closed (not!)
  - ACLs removed (no file access at all in Netscape)
- Too bad
  - Denial of service attacks “lower in priority than system integrity”

Netscape 2.0 Security

- Separation of Netscape and Java code
  - can’t change HTTP proxy server
  - less chance for accidental or malicious bug introduction
- Fixed security policy
  - can’t trick users into lowering security
- Serious limits on functionality
  - no file system access

Netscape 2.0β Insecurity

- protected variables were effectively public
  - semantics of **`protected`** changed in JDK β2 (Netscape 2.0 β4)
  - could set **`SecurityManager.inCheck`**, opening DNS channel
- could learn the user's name [Burchard]
- could read the clipboard [Burchard]
  - fixed in JDK β2

JDK 1.0 Insecurity

- **`javap`**, the disassembler, calls **`sprintf()`** wrong
  - can overflow internal buffers
    - similar attack last year on **`syslog(3)`** [CERT 95:13]
  - examining a Java class can run arbitrary native code!
- similar bugs in Java 1.0α3 were fixed, but they forgot **`javap`**.

Netscape 2.0 Insecurity

- Denial of service attacks still available
- Applets can interfere with each other

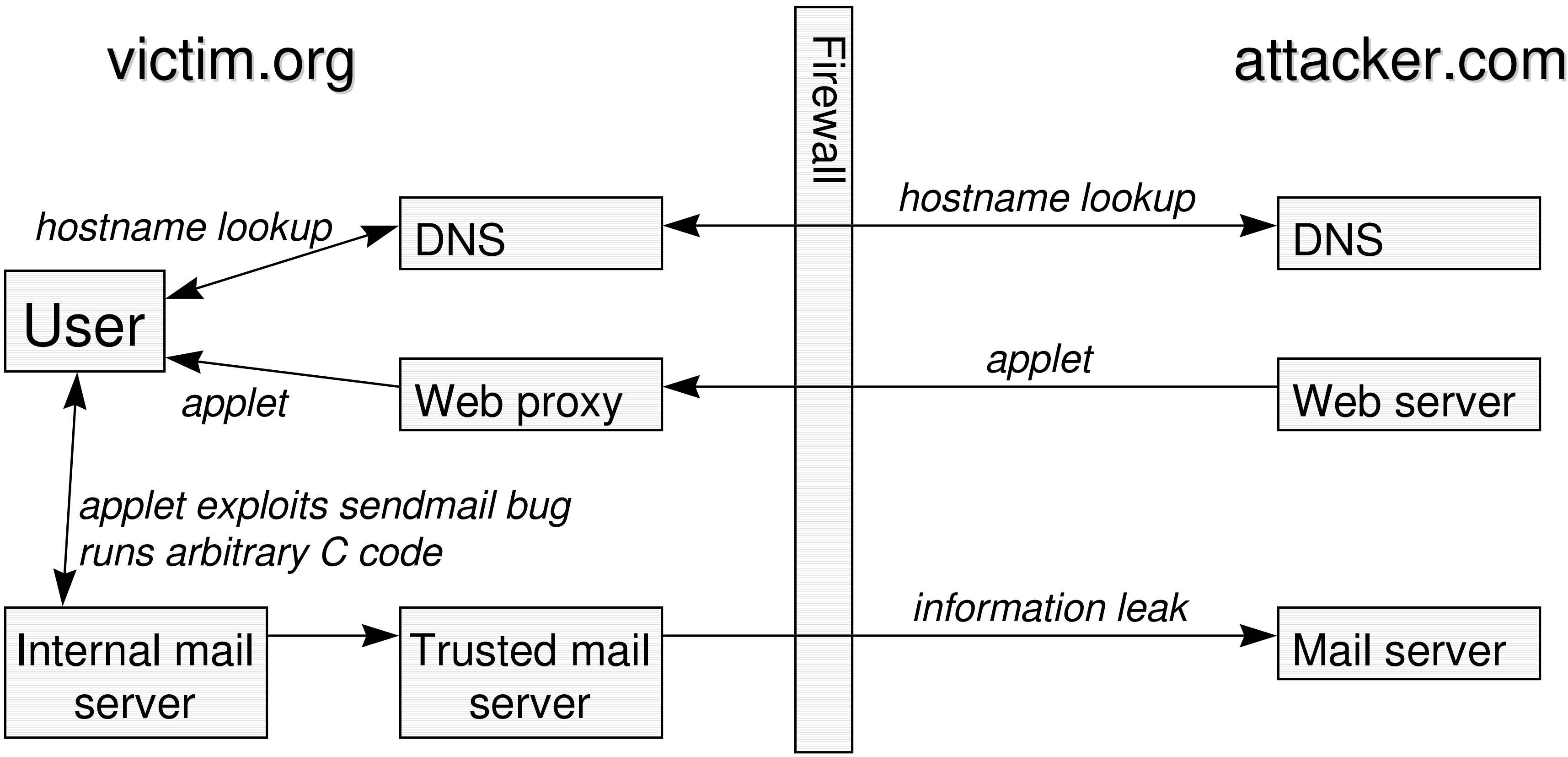
```
/*
 * @(#)check_code.c     1.51 95/12/02
 */

/*-
 * Verify that the code within a method block doesn't
 * exploit any security holes.
 *
 * This code is still a work in progress.  All currently
 * existing code passes the test, but so does a lot of bad code.
 */
```

Netscape 2.0 Insecurity

- Java trusts DNS
  - Internet hosts can have multiple IP addresses
  - Java host equality test is ***too lenient***
- With a hacked DNS server
  - Two-way channel to any machine on the Internet
  - Applets can connect to machines ***behind*** a firewall
    - Exploit numerous Unix and Windows bugs
    - Talk to internal Web and NetNews servers

Netscape DNS Attack



The DNS attack allows connections to *any* machine behind the firewall.

Netscape 2.0 Insecurity

- Java trusts bytecode to enforce language semantics
  - Superclass constructors may throw **`SecurityException`**
    - Prevents instantiation of **`ClassLoader`** and other classes
  - Exception can be ignored by custom bytecode
- A ClassLoader can break the type system

Running Machine Code

- Available tools
  - Read/modify any variable, and call any method
  - Use ints as object references, and vice versa
  - Double dereference any pointer
  - Access to C implementation of class `Class`
- Puzzle: Run arbitrary machine code?

Running Machine Code

Outline

- Introduction
- Java Semantics
- HotJava and Netscape Security Flaws
- **Security Analysis**
- Application Requirements
- User Interface
- Future work / Conclusions

Security Analysis

- Sun wants you to believe Java is secure
  - Applets don't have access to any information
  - There are no channels to leak information out
  - Safe language thwarts malicious applets
- We found
  - Interesting information available to applets
  - Channels exist to leak information out
  - Applets can execute arbitrary machine code

Security Policy

- No formal model
  - "A program that has not been specified cannot be incorrect; it can only be surprising." [YBK85]
- Why this is bad
  - We can't say what "secure" means
  - We can't verify an implementation

Accountability

- Java does **not** log applets or their actions
- Should log
  - File system and network access
  - Applet bytecode
- Evidence of an attack
  - Reconstruct what happened
  - Seek legal recourse

Integrity

- HotJava is harder to secure than Netscape
  - More state kept in Java
  - Lack of formal interface between browser and applets
  - Mistakes (public variables) become security problems in HotJava
- Browser in Java won't have C safety problems
- This issue will reappear in future HotJava releases (expected 1Q96)

Assurance

- Java and HotJava do not identify a TCB (trusted computing base)
- Security critical functionality spread throughout the code
  - Dynamic type checks
  - Not all native methods protected by **`SecurityManager`**
- Bugs in release edition - rushed shipment?

Anatomy of a File Open

```
public FileInputStream(String name)
throws FileNotFoundException {
    SecurityManager security =
        System.getSecurityManager();
    if (security != null) {
        security.checkRead(name);
    }
    try {
        open(name);
    } catch (IOException e) {
        throw new FileNotFoundException(name);
    }
}
```

Language Issues

- Public variables are **dangerous**
  - Why are they writable across name spaces?
- Java's **`package`** mechanism
  - Not as useful as parameterized module system (e.g. Standard ML's functors)
  - Hierarchical module system allows hierarchical protection

Intermediate Representation

- Abstract Syntax Trees vs. Bytecode
  - ASTs easier to type check
    - No need for global dataflow analysis
  - ASTs have same semantics as language
    - Bytecode has its own semantics
  - Comparable compilation speed

Application Requirements

- What do we want to write in Java?
  - Distributed applications
    - A/V conferencing, but not cross-network bugs
    - Loosely coupled computations (e.g. factoring), but neither stealing cycles nor denial-of-service attacks
    - Games, but not trojan-horse benchmarks
  - General Applications
    - Save/restore preferences, but not read **`/etc/passwd`**

User Interface

- Too easy for the user just to click OK
  - Goal: minimize user involvement in security
  - Trusted and unspoofable dialogs for file access
- Unforgable device access indicators
- Access to the clipboard
  - *Paste to applet* on *Edit* menu
    - Explicit user-initiated request, not applet-initiated

Digital signatures for applets

- Grant more trust to signed applets?
- Log user-approved capabilities per applet or per source
- User-specified or organization-specified policies?
  - Different degrees of trust between organizations
  - One rigid policy won't fit everybody
- Interaction with organization's firewall?

Future Work

- Design a set of security policies
- Implement policies in Netscape and/or HotJava
- Build a high-assurance Java runtime system

Conclusions

- Remote code is inevitable for the Web
- Java is promising, but has important bugs and design issues
- Stronger security measures can allow **more** functionality for untrusted applets without compromising privacy and integrity

http://www.cs.princeton.edu/~ddean/java/