Netscape Code/Object Signing

Jim Roskind
Java Security Architect
Netscape Communications Corp

Presented 6/25/97
to
The Open Group
in
London England

Road Map

- Motivation
- History of Java Security
- Ergonomics of Security
- Concrete Examples

Code and Data Signing: Why?

- **Actions beyond the Java/JavaScript Sandbox**
  - **File I/O**
  - **Network Connections**
  - **JavaScript context protection**
  - **Browser Control powers**
- **Signing provides source authenticity**
- **Tamper resistance**

Road Map

- Motivation
- History of Java Security
- Ergonomics of Security
- Concrete Examples

Fundamentals of Java Security

- **Interpreted language**
  - **Caller is definitively known**
  - **No direct memory manipulation**
- **No pointer arithmetic**
  - **GC rather than malloc/free**

More Fundamentals of Java Security

- **Code source location known**
  - **"codebase"; CLASSPATH; Signed classes**
- **Class based security**
  - **private, protected, package isolation**
- **Type Safety**
  - **No evil casting**

Java Attacks Methods

- **Luring Attacks**
  - **String was not final**
  - **System.out was not final**
- **Overly Large TCB Attacks**
  - **Human factors: font attacks**

Traditional Java SecurityManager Model

- Centralized SecurityManager
- Class granularity privileging
- Some Thread based security decisions

Centralized SecurityManager

- Security was separated from code
- CLASSPATH code is powerful
- Non-extensible base class

Class Granularity Security

- **Binary (two state) trust model**
- **Trust level based on Classpath vs URL**
- **Gigantic TCB**

Thread Based Security

- **Non-ergonomic**
  - **Programmers forget they have power**
- **Pure CLASSPATH thread got extra power**
  - **Luring attack problems**
  - **Data-driven AWT threads**

Implementation Flaws Due to Static Capabilities

- **We have met the enemy, they are *us***
- **Font.getFont()**
  - **Answer all System.property queries**
- **SecurityManager.checkPackageAccess()**
  - **No checking done on *first* class in applet**

Navigator 3.0x Security Evolution: SetScopePermission

- Binary Trust grew to 3 state model
- Programmers must say shazzam before using their powers
- The size of the TCB shrunk

Applying Cryptographic Technology: Signed Objects

- **Unforgeable source of Java and JavaScript**
- **Tamper proof seal**
- **How can we use them?**

Road Map

- Motivation
- History of Java Security
- Ergonomics of Security
- Concrete Examples

Ergonomics

- Designer can induce faults
- gets() vs fgets()
- malloc/free vs garbage collection

Ergonomics and Security

- To be secure, the system must be usable
- User security
- Programmer security
- Administrative support
- Extensible security infrastructure

User Security Ergonomics

- Don't provide too many dialogs
- Dialogs must be summarized
  - High, Medium, Low Risk
- Provide reasonably fine grain control
- Allow user to edit and review grants

Programmer Ergonomics

- **Support request for least privilege**
- **Encourage holding privilege for shortest duration**
- **Minimize significant API**
  - **enablePrivilege() induces user grant request as side effect**

Administrative Ergonomics

- Support "locking" of capabilities database
- Use scripting language
- Allow site customization
  - CA Database pre-load
  - System Principal selection

Advanced Programmer Features

- Targets are named
- Support macro targets
- Support run-time testing for target enablement

Advanced Programmer Features: Named Targets

- Each Principal has a separate name space
- Principals can define targets in their name space

Advanced Programmer Features: Macro Targets

- Restrict macros to within a principal name space
- Ensure "risk" factor is correctly calculated

Advanced Programmer Features: Security Infrastructure

- Enable/check of caller
- Get Principal of caller
- Match Principal of other class

Assurance Potential

- **Netscape's Java Security sub-structure has shrunk**
- **Developers get reduced windows of vulnerability**

JavaScript Security

- Traditionally very open for ease of writing
- Signed JavaScript establishes private haven for data
- You cannot mix signed and unsigned JavaScript
- JavaScript calls into Java to perform most powerful deeds

Road Map

- Motivation
- History of Java Security
- Ergonomics of Security
- Concrete Examples

Details: Making it all concrete

- Sample Targets
- Sample Macro Targets
- How to Experiment
  - Without signing
  - With signing
- Signing Java
- Signing JavaScript

Fine Granularity Capabilities

- **Full "standard" list in JavaDoc for netscape.security.Target**
- **Typical Items include:**
  - **UniversalFileRead**
  - **UniversalFileWrite**
  - **UniversalConnect**
  - **...**
  - **UniversalTopLevelWindow**

Macro Target Sample

- **TerminalEmulator includes:**
  - **UniversalFileRead**
  - **UniversalFileWrite**
  - **UniversalConnect**

Macro Target Sample: TerminalEmulator

- **Description to user is:**
  - **Access required by terminal emulators and other communications programs**

Macro Target Sample: TerminalEmulator

- **Detailed description:**
  - **Reading and writing files and establishing network connections. This form of access is required by terminal emulators such as the 3270 or VT100 emulator.**
- **List of Sub-Targets is also presented**

Experimenting Without Signing

- Critical developer trick
- Modify your prefs.js file, setting the following to true:
- "signed.applets.codebase_principal_support"

Really Signing

- Get a code signing cert
  - http://digitalid.verisign.com/nosintro.htm
- Get a code signing tool
  - http://developer.netscape.com/software/signedobj
- (Don't use JavaKey from JavaSoft... yet)

Java Signing

- Place classes into a subdirectory
- Create signed archive using tool
- Refer to signed archive via:
- <applet archive=your_arch.jar class=your_main.class>

JavaScript Signing

- **Reference JavaScript in an archive**
  - **<script archive=your_arch.jar src=your_script.js>**
- **Inline JavaScript (with sig in JAR file)**
  - **<script archive=your_arch.jar ID=your_label> ...**

Developer Tricks

- For portability to less secure platforms, include dummy netscape.* classes
- To get good stack traces, rename JIT DLL
- Modify your prefs.js file, setting the following to true:
- "signed.applets.verbose_security_exception"

Summary

- **Ergonomics: Human usability**
  - **It isn’t security, if it isn’t being used by humans**
- **Java and JavaScript provide portability**
- **Netscape is advancing security for user, programmer, and administrator**

Netscape Code/Object Signing

Jim Roskind
jar@netscape.com

Presented 6/25/97
to
The Open Group
in
London England