

# Java Security

David A. Wheeler

[dwheeler@dwheeler.com](mailto:dwheeler@dwheeler.com)

(703) 845-6662

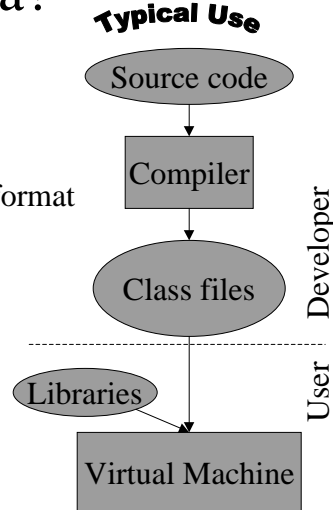
November 30, 1999

## Outline

- Java Basics
  - What's Java, Modes of Use, major components, implications, implementations
- Security-related capabilities (JDK 1.0, 1.1, "1.2")
- Selected upcoming developments
- Miscellaneous
  - Past breaches, malicious applets, advantages & disadvantages, key points

# What's Java?

- Java Technologies:
  - Java language
  - Virtual machine (VM)/class file format
  - Libraries
- Can use only VM or language
- Developed by Sun
- Not related to “Javascript”
- Cross-Platform (WORA)



December 1, 1999

(C) 1999 David A. Wheeler

3

# Java Modes of Use

- Applets: Auto-run when view web page
- Applications: Traditional program (performance?)
- Beans: Component (like OLE object)
- Servlets: Server-side applications
- Aglets: Intelligent Agents
- Doclets: Configurable doc generator
- Embedded Systems
- Smart Cards (“JavaCard”)

December 1, 1999

(C) 1999 David A. Wheeler

4

# Java Language

- Modern object-oriented (OO) language
  - OO with single inheritance + multiple “interfaces”
  - Classes grouped into hierarchical packages
  - Strong static typing (no arbitrary pointers)
  - Automatic garbage collection
  - Exceptions
  - Multithreaded
- Lacks enumerations and templates (generics)
- Syntax ~C++, semantics ~Ada95/Smalltalk

December 1, 1999

(C) 1999 David A. Wheeler

5

# Java Virtual Machine (VM) and Class File Format

- Class file defines names/types/values of class variables, constants, & methods
- Methods stored as instructions to stack-based VM
  - Very similar to UCSD p-code
- VM executes class files (inc. collections of them)
  - By interpretation, run-time compilation, or combination; performance is a significant issue
- Before execution, VM usually runs “bytecode verifier” to check legality of class file

December 1, 1999

(C) 1999 David A. Wheeler

6

# Java Libraries

- Set of built-in APIs, including:
  - GUIs
  - Networking
  - Computation
- Growth area
- Several classes are security-related
  - This presentation will skim ordinary crypto functions such as ones for encryption/decryption, certificate management, etc., since they are not essentially unique

December 1, 1999

(C) 1999 David A. Wheeler

7

## Class and Method Access Control Modifiers

Access Control Modifier	Class or Interface Accessibility	Member (Field or Method) Accessibility
Public	All	All if class or interface is accessible; interface members always public
Protected	N/A	Same package OR subclass
“default” (Package private)	Same package	Same package
Private	N/A	Only same class (not subclass)

December 1, 1999

(C) 1999 David A. Wheeler

8

## Implications of Java Basics

- No arbitrary pointers: references ~ capabilities
  - Only creator & createe have reference for new object
  - If objectset doesn't pass a reference, you can't manipulate that object
- Can only manipulate objects in limited ways
  - If data private, can only manipulate via methods
  - Methods can be used to protect data
  - Constructor method can limit who can create an object
- Software-enforced protection (small slips break it)

December 1, 1999

(C) 1999 David A. Wheeler

9

## Notes on Java Implementations

- “Java” is the general technology
- Multiple Java Implementations
  - Sun, Microsoft (derived), Kaffe, ...
  - This presentation emphasizes Sun's implementations
- Java caught in industry crossfire
  - Microsoft pollution, hostile approach, court cases (user best interests not in Microsoft's best interest)
  - Sun intransigence in relinquishing control to independent bodies (ISO, ECMA)

December 1, 1999

(C) 1999 David A. Wheeler

10

## Security-Related Capabilities (1 of 2)

- JDK 1.0 (Fall 1995)
  - Policy: “Sandbox” for applets; others unlimited
  - Mechanisms: SecurityManager, Bytecode verifier, Classloader
- JDK 1.1 (Spring 1997)
  - Policy: can also grant total trust to signed applets
  - Mechanisms: Java Archive (JAR), crypto-related APIs
- Inflexible: Too little or too much privilege

December 1, 1999

(C) 1999 David A. Wheeler

11

## Security-Related Capabilities (2 of 2)

- Netscape & Microsoft Extensions
  - Enabled more flexible approaches
  - Incompatible with each other and with Sun
- J2SE (Java 2 Platform Standard Edition) (Fall 1998)
  - Includes SDK 1.2 and runtime
  - Policy: can also grant fine-grained privileges to specific applets/classes based on source and/or signatures
  - Mechanisms: AccessController, ProtectionDomain, CodeSource, Permission, GuardedObject, ...
  - “Java Plug-in” supports both Microsoft & Netscape

December 1, 1999

(C) 1999 David A. Wheeler

12

# Java 1.0 Security Policy

- Sandbox Policy (for applets)
  - Cannot access local filesystem or devices
  - Network connections only to applet load source
  - Cannot invoke any local program or library
  - “Untrusted” indicator on top-level windows
  - Cannot manipulate basic classes or another ThreadGroup
  - Appletviewer CL can be initialized to vary these
- Applications unlimited in 1.0; can code a policy

December 1, 1999

(C) 1999 David A. Wheeler

13

# SecurityManager

- Class defines check methods called by system
  - E.G. “checkRead(String filename)”
  - Method throws exception if invalid
- To create a security policy from scratch:
  - Create a subclass (code) & instantiate
  - Install using System.setSecurityManager; this cannot be revoked or replaced
  - This is used to create the Sandbox
  - If no SecurityManager installed, all privileges granted

December 1, 1999

(C) 1999 David A. Wheeler

14

# Bytecode Verifier

- Checks a classfile for validity:
  - Code only has valid instructions & register use
  - Code does not overflow/underflow stack
  - Does not convert data types illegally or forge pointers
  - Accesses objects as correct type
  - Method calls use correct number & types of arguments
  - References to other classes use legal names
- Goal is to prevent access to underlying machine
  - via forged pointers, crashes, undefined states

December 1, 1999

(C) 1999 David A. Wheeler

15

# ClassLoader

- Responsible for loading classes
  - given classname, locates/generates its definition
  - always looks at “standard” classes first
  - every class has a reference to the classloader instance that defined it
  - keeps namespaces of different applets separate (different ClassLoader instances)
  - each ClassLoader instance ~ OS process
  - “CLASSPATH” classes trusted in JDK 1.0-1.1, system classes trusted, otherwise invokes bytecode verifier

December 1, 1999

(C) 1999 David A. Wheeler

16



## Java Archive (JAR) Format (1.1)

- Format for collecting & optionally signing sets of files
  - ZIP format + manifest + optional signatures
- Manifest
  - In file META-INF/MANIFEST.MF
  - Lists (some) JAR filenames, digests, digest algorithm(s) (MD5, SHA)
- Signatures
  - Separate manifest-like file, separate signature

December 1, 1999

(C) 1999 David A. Wheeler

17

## Java Cryptography Architecture (Added in 1.1)

- Java cryptography architecture (JCA)
  - Framework (API) for access to services implemented by pluggable “providers”
  - digital signature algorithms (DSA), message digest algorithms (MD5 & SHA-1), key-generation algorithms, simple certificate management (1.1 had no API for specific formats)
  - Simple key management tool (simple “database”)

December 1, 1999

(C) 1999 David A. Wheeler

18

## Problems with 1.0 through 1.1

- Sandbox too limiting
- “Trusted” programs given too much power
- Hard to define new security policy
  - Must write own SecurityManager
  - Must install it on its own JVM
- New privileges difficult to add
  - New method must be added to SecurityManager
  - Creates a backward incompatibility for each addition

December 1, 1999

(C) 1999 David A. Wheeler

19

## Netscape Extensions

- Navigator 4.0 added “Capabilities” API:
  - Call to request privilege enable (string)
  - If not been granted before, UI asks if ok
  - Privilege disabled when method returns, but can be re-enabled without UI
  - Can disable or revert, can select which certificate to use
- May grant privileges to certificates or codebase
- Problems: Incompatible (Netscape only)

December 1, 1999

(C) 1999 David A. Wheeler

20

## Microsoft Extensions

- Used CAB not JAR for signatures (incompatible)
- IE 3.0: Selected signed applets trusted
- IE 4.0: Fine-grained “Trust-Based Security”
  - User defines zones (std: Local, intranet, trusted sites, Internet, untrusted sites)
  - Each zone given privileges; standard privilege sets: High, Medium (UI file I/O), Low security
  - CAB file includes privilege request; query if beyond preapproved set (& okay with admin)
- Problem: Incompatible (IE on Win32 only)

December 1, 1999

(C) 1999 David A. Wheeler

21

## Security-Related Capabilities in Java 2 (SDK 1.2)

- Fine-grained configurable policies
  - Sample Security Policy
  - Runtime State: ProtectionDomain/CodeSource/Policy
  - Java 2 Runtime Security Check Algorithm
  - Permission & Its Subclasses
  - SecurityManager & AccessController
  - GuardedObject & Guard
- Java Cryptography Architecture (JCA) changes
- Java Cryptography Extension (JCE)

December 1, 1999

(C) 1999 David A. Wheeler

22

## Sample Fine-Grained Security Policy for One User

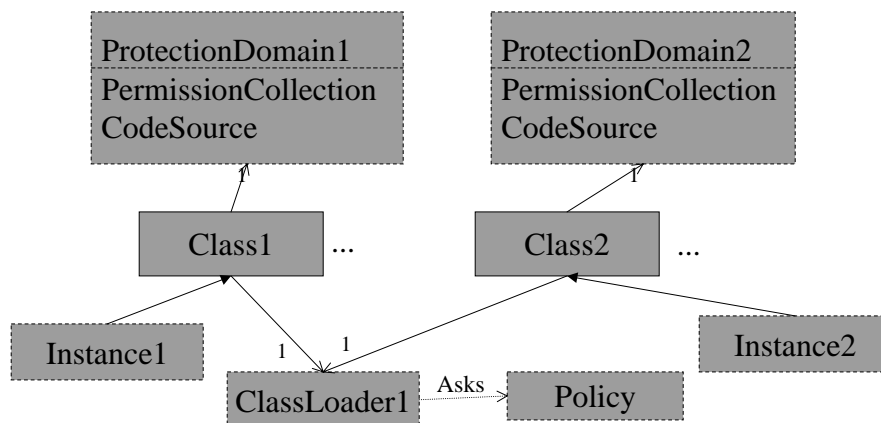
Source of Code (CodeSource)		Permissions
Base URL	Signature	
http://www.schwab.com/classes/stockeditor.jar	Schwab's signature	<ul style="list-style-type: none"> <li>Read/write file /home/daw/stocks</li> </ul>
http://*.schwab.com/	(not required)	<ul style="list-style-type: none"> <li>Connect/accept bankofamerica.com ports 1-1023</li> <li>Read file /home/daw/logo.png</li> </ul>

December 1, 1999

(C) 1999 David A. Wheeler

23

## Java 2: Each Class Has A ProtectionDomain



December 1, 1999

(C) 1999 David A. Wheeler

24

## ProtectionDomain Class

- ProtectionDomain class
  - Created from a CodeSource and a PermissionCollection
  - Defines the set of permissions granted to classes; change the PermissionCollection to change permissions
  - Each class belongs to ONE ProtectionDomain instance, set at class creation time (and never changed again)
  - Access to these objects restricted; getting its reference requires RuntimePermission getProtectionDomain
- One ClassLoader can have >1 protection domain

December 1, 1999

(C) 1999 David A. Wheeler

25

## CodeSource Class

- Created from:
  - a source (base) URL and
  - array of certificates
- Immutable
- “implies” method implements URL partial matches
  - Permits policies to use URL patterns

December 1, 1999

(C) 1999 David A. Wheeler

26

## Policy Class

- Provides interface to user policy
  - Given a `CodeSource`, returns a `PermissionCollection`
  - Used during setup of `ProtectionDomain` to set a class' permissions

December 1, 1999

(C) 1999 David A. Wheeler

27

## How a Class and ProtectionDomain Are Loaded

1. Loaded class C1 requests an unloaded class C2
2. C1's `ClassLoader` called, loads C2's class file, calls bytecode verifier
3. C2's `CodeSource` determined
4. Policy object given `CodeSource`, returns `Permissions`
5. If an existing `ProtectionDomain` has same `CodeSource` & `Permissions`, reused, else new `ProtectionDomain` created; C2 assigned to it

December 1, 1999

(C) 1999 David A. Wheeler

28

## Java 2 Runtime Security Check Algorithm

- If method M requires permission P
  - M's implementation calls current SecurityManager's checkPermission(P)
- By default this calls new "AccessController" class
  - For each call stack entry, unwind from caller:
  - if caller's ProtectionDomain lacks P, exception (fail)
  - if caller called "doPrivileged" without context, return
  - if caller called "doPrivileged" with context, check it: return if context permits P else exception (fail).

December 1, 1999

(C) 1999 David A. Wheeler

29

## Examples of Algorithm At Work

- Multiple ProtectionDomains:
  - Instance1 M1 calls Instance2 M2 calls System1 M3
  - System1 M3 (in System's ProtectionDomain) asks for a permission check
  - Permissions checked against the ProtectionDomains for System1, then Class2, then Class1
- doPrivileged call (without context):
  - Same example, but first System1 M3 calls doPrivileged
  - When permission check requested, ProtectionDomain for System1 checked and *no others* checked

December 1, 1999

(C) 1999 David A. Wheeler

30

## Context

- `getContext()` takes a snapshot of current execution context (“stack trace”)
  - snapshot includes ancestor threads
  - stored in type `AccessControlContext`
  - results can be stored & can used later to limit privileges (instead of enabling “all” privileges)
- Purpose: support actions “on behalf of another”
  - one thread posts event to another
  - delayed actions (“cron” job)

December 1, 1999

(C) 1999 David A. Wheeler

31

## Algorithm Implications

- Default privileges are the *intersection* (minimum) of all class’ permissions in call tree
  - Without `doPrivilege`, permissions only decrease
- “`doPrivilege`” enables “all” class’ privileges
  - Like Unix “`setuid`”; enables trusted classes to use their full set of privileges but only when requested
  - Without context enables all privileges
  - With context enables only those privileges *also* in given context; safe because resulting privileges always less than without context

December 1, 1999

(C) 1999 David A. Wheeler

32



## Warning: Don't Mix Protected Variables and Permission Checks

- If a method M1 is not overridden, the ProtectionDomain of its defining superclass used
- Methods running (even indirectly) with privilege shouldn't depend on protected variables
  - Attacker creates subclass with new method M2
  - M2 modifies protected variable used by M1
  - Cause M1 to be invoked; M1 influenced by M2!
- Identified by David A. Wheeler Oct 1999
  - Have not seen this in the literature

December 1, 1999

(C) 1999 David A. Wheeler

33

## Permission Class

- Permission class
  - Encapsulates a permission granted or requested
  - Can be set “readonly” (from then on immutable)
  - Can be grouped using classes PermissionCollection and Permissions
- This briefing's terminology:
  - permissions granted to a ProtectionDomain also called “privileges”
  - no separate “Privilege” class

December 1, 1999

(C) 1999 David A. Wheeler

34

## Permission Subclasses: FilePermission Class

- Gives rights to local files/directories
- Path name/pattern
  - Specific path: *file*, *directory*, *directory/file*
  - All files in directory: *directory/\**
  - All files recursively in directory: *directory/-*
  - For current directory, omit “*directory/*”
  - For all files (**dangerous**), “<<*ALL FILES*>>”
- Rights set (1+): read, write, execute, delete

December 1, 1999

(C) 1999 David A. Wheeler

35

## Permission Subclasses: SocketPermission

- Host
  - Local machine: “”, “*localhost*”
  - Given machine: *IP address* or *hostname*
  - All hosts in a domain: *\*.domain*
  - All hosts: \*
- Portrange
  - Single port: *portnumber*
  - Port range: *port1-port2*, *port1-*, *-port2*
- Actions (1+): accept, connect, listen, resolve

December 1, 1999

(C) 1999 David A. Wheeler

36

## Permission Subclasses: PropertyPermission

- Gives rights to properties
  - Similar to OS environment variables
- Target
  - Specific property: os.name
  - Pattern: java.\*
- Actions (1+): read, write

December 1, 1999

(C) 1999 David A. Wheeler

37

## Permission Subclasses: Other Permission Subclasses

- RuntimePermission: string with permission name
  - createClassLoader
  - getClassLoader
  - setSecurityManager
  - exitVM
  - ...
- Many other specialized Permission subclasses
- AllPermission
  - special class meaning “all permissions”

December 1, 1999

(C) 1999 David A. Wheeler

38

# SecurityManager Changes

- New method `checkPermission(P)`
  - Throws exception if permission P not held, else returns
  - All previous “check” methods rewritten in terms of `checkPermission`
  - Permits creation of new Permissions without changing `SecurityManager`
- By default, calls on `AccessController` class
  - `AccessController` implements the new algorithm

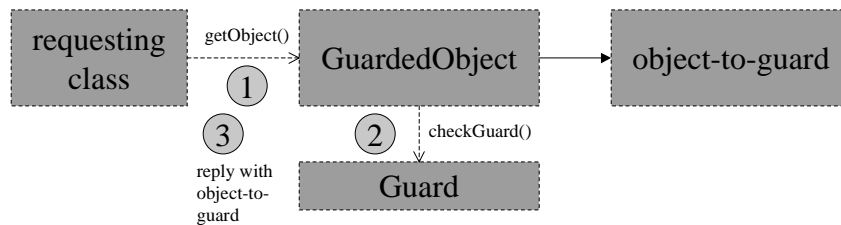
December 1, 1999

(C) 1999 David A. Wheeler

39

# GuardedObject (1 of 3)

- To protect *one* method in *all* instances, use `SecurityManager` directly as shown so far
- To protect a *reference* to an *individual* instance, consider using “`GuardedObject`”:



December 1, 1999

(C) 1999 David A. Wheeler

40

## GuardedObject (2 of 3)

- GuardedObject class encapsulates object-to-guard
  - asks “Guard” interface to determine if access ok
  - Permission implements Guard by calling SecurityManager. checkPermission(self)
  - PermissionCollection doesn’t implement (I’ve reported)
- Provider of object-to-guard does the following:
  - Instantiates new Guard (e.g., a Permission)
  - Instantiates GuardedObject, using object-to-guard and the guard
  - Gives GuardedObject’s reference to requestors

December 1, 1999

(C) 1999 David A. Wheeler

41

## GuardedObject (3 of 3)

- Clients who wish to use object-to-guard call GuardedObject’s getObject()
  - GuardedObject instance calls its Guard’s checkGuard()
  - if ok, object-to-guard’s reference returned
  - if not ok, security exception thrown

December 1, 1999

(C) 1999 David A. Wheeler

42

## Java Cryptography Architecture (JCA) Changes in 1.2

- Adds more APIs that providers can support
  - Keystore creation and management
  - Algorithm parameter management
  - Algorithm parameter generation
  - Conversions between different key representations
  - Certificate factory support to generate certificates and certificate revocation lists (CRLs) from their encodings (Sun implements X.509's)
  - Random-number generation (RNG) algorithm

December 1, 1999

(C) 1999 David A. Wheeler

43

## Java Cryptography Extension (JCE)

- Adds encryption, key exchange, key generation, message authentication code (MAC)
  - Multiple “providers” supported
  - Keys & certificates in “keystore” database
- Separate due to export control

December 1, 1999

(C) 1999 David A. Wheeler

44

## Other Areas In Development: JSSE and JAAS

- Java Secure Socket Extension
  - Implements SSL
- Java Authentication and Authorization Service
  - Based on PAM: pluggable authenticators for passwords, smart cards, biometric devices, etc.
  - Authenticators may be required, requisite (stop on failure), sufficient (but not required), or optional
  - Adds user-centric (vs. code-centric) control: permissions granted to Principal (not just CodeSource), implemented through a modified SecurityManager

December 1, 1999

(C) 1999 David A. Wheeler

45

## Past Java Security Breaches (1 of 2)

- 8 Serious Breaches listed in *Java Security* (1997)
  - “Jumping the Firewall” (DNS interaction)
  - “Slash and Burn” (slash starts classname)
  - “Applets running wild” (evil class loader installed and creates type confusion)
  - “Casting Caution” (failed to test if method private, type casting)
  - “Tag-Team Applets” (create type confusion)

December 1, 1999

(C) 1999 David A. Wheeler

46

## Past Java Security Breaches (2 of 2)

- “You’re not my type” (flaw in array implementation - type confusion)
- “Casting Caution #2” (as before, but in a loop test wasn’t repeated)
- “Big Attacks Come in Small Packages” (untrusted code could be loaded into sensitive packages, e.g. com.ms, and gain their privileges)
- Others have been announced since
  - See <http://java.sun.com/sfaq/chronology.html>
  - Many are problems in bytecode verifier or classloader

December 1, 1999

(C) 1999 David A. Wheeler

47

## Malicious Applets (Staying Within the Sandbox)

- Denial of Service
  - Deny platform use (busy threads, loop, exhaust GUI resources)
  - Kill other threads
- Invasion of Privacy
- Annoyance: constant sound
- Flashing display (causes seizures in some users)
- Steal CPU cycles (e.g. crack encryption)

December 1, 1999

(C) 1999 David A. Wheeler

48



## Java Advantages

- Permits controlled execution of less trusted code (vs. ActiveX)
- Permits fine-grained permission control
- Attention paid to security
- Portability
- “Instant installation”
- Sun’s source reviewable (*not* open source)

December 1, 1999

(C) 1999 David A. Wheeler

49

## Java Security Disadvantages (1 of 3)

- Hard to prove correct
  - complex from security point-of-view
  - rapidly expanding/changing
  - VM+libraries lacks formal security model
- Many internal interdependencies (vs. reference monitors); often breaks “all the way”
- Complex dependencies on other systems
  - OS, browsers, network (DNS), PKI

December 1, 1999

(C) 1999 David A. Wheeler

50

## Java Security Disadvantages (2 of 3)

- Applets evade many security measures (e.g. most firewalls)
- Breaches demonstrated
- Many areas immature
- No standardized auditing (MS extension)
- Simplifies reverse engineering of code (problem?)
- Poor performance may encourage security-weakening “shortcuts”

December 1, 1999

(C) 1999 David A. Wheeler

51

## Java Security Disadvantages (3 of 3)

- Weak against denial-of-service & nuisances
- Insecure implementation defaults (e.g. null ClassLoader or SecurityManager)
- Security policy management too complex for endusers and weak administrative support
- Flexible policies accepted by users may permit hidden breaching interactions

December 1, 1999

(C) 1999 David A. Wheeler

52

## Key Points

- Progression of Access Control Flexibility
  - JDK 1.0: Sandbox + total trust of local applications
  - JDK 1.1: Above + optional total trust with signature
  - SDK 1.2: Above + Fine-grained access control
- Java 2 ProtectionDomains
  - Checks call tree, by default intersection of permissions
  - doPrivilege permits permissions to be re-enabled
- GuardedObject to protect specific objects

December 1, 1999

(C) 1999 David A. Wheeler

53

## Useful References

- Li Gong, *Inside Java 2 Platform Security*, 1999, Palo Alto, CA: Addison-Wesley.
- G. McGraw & E. Felten, *Java Security: Hostile Applets, Holes, and Antidotes*, 1997, NY: John Wiley & Sons.
- Sun's Java website: <http://java.sun.com>
- Existing Java programs/info available at:
  - <http://www.gamelan.com>
  - <http://www.jars.com> (Java Applet Rating Service)

December 1, 1999

(C) 1999 David A. Wheeler

54

## About this Briefing

- This briefing is at: <http://www.dwheeler.com>
- This entire briefing is GPL'ed:
  - (C) 1999 David A. Wheeler.
  - This information is free information; you can redistribute it and/or modify it under the terms of the GNU General Public License (GPL) as published by the Free Software Foundation; either version 2 of the license, or (at your option) any later version. This information is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU GPL for more details. You should have received a copy of the GNU GPL along with this information; if not, see <http://www.fsf.org/copyleft/gpl.txt> or write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307

December 1, 1999

(C) 1999 David A. Wheeler

55

## Backup Slides

December 1, 1999

(C) 1999 David A. Wheeler

56

## Java Wallet

- Java Wallet = product suite for e-commerce
  - Java Commerce Client (JCC), Commerce JavaBeans, Gateway Security Model, Java Commerce Messages
- Java Commerce Client is client-side framework
  - Wallet-like UI, database, basic framework classes
  - Once configured, can download & install Commerce JavaBeans

December 1, 1999

(C) 1999 David A. Wheeler

57

## Java Wallet: Commerce JavaBeans

- Specialized JavaBeans (originally unrelated)
- Types of Commerce JavaBeans include:
  - User Interface (“Java wallet”)
  - Operation (e.g. Purchase operation)
  - Protocol (implements transfer, e.g. SET, post)
  - Instrument (form of payment, e.g. VISA)
  - Service (for local operations, e.g. address book)
- Electronic purchase requires: Purchase operation, 1+ protocol, 1+ instrument

December 1, 1999

(C) 1999 David A. Wheeler

58

## Java Wallet: Cassettes

- Commerce JavaBeans contained in “cassettes”
  - Cassette = signed JAR file with Commerce JavaBean(s)
  - Signed with one or more Roles
  - Cassette implements part of a transaction
  - Cassettes may be supplied by different vendors

December 1, 1999

(C) 1999 David A. Wheeler

59

## Java Wallet: Sample Purchase

- Buyer selects “PAY” on a web site
- Merchant sends Java Commerce Message:
  - operation (purchase), permitted instruments & protocols, cassette locations, info on goods, merchant, buyer
- JCC parses JCM, hands to operation bean
- Bean finds available instruments/protocols
- Buyer selects one & transaction completes

December 1, 1999

(C) 1999 David A. Wheeler

60

## Java Wallet: Gateway Security Model

- “Gateway Security Model”: must communicate to a cassette via its gate
  - Roles: all cassettes signed for 1+ roles
  - Ticket: Use-once token of the capabilities authorized by the role
  - Gates: Checks ticket (on cassette entry) - if ok, returns a permit
  - Permits: Capability returned by gate to cassette

December 1, 1999

(C) 1999 David A. Wheeler

61

## Java Wallet: Problems

- Poorly integrated with rest of Java
  - Security model (“Gateway security model”) completely different & not well integrated with protection domains
  - Originally “Commerce Beans” incompatible with JavaBeans and GUI classes duplicated
- Complex security model
  - Trustability difficult to ascertain
  - Complex infrastructure where simpler solutions exist
- Work appears to have halted mid-1998

December 1, 1999

(C) 1999 David A. Wheeler

62

## Some Potential Uses of Java in Secure Electronic Commerce

- Might purchase Java program/service implemented using Java
- Applets
  - GUIs for complex purchases
- Java Wallet: permits interacting mutually suspicious units of code
- Smart Cards: mass-produced, configurable
- Applet “instant installation” aids consumer

December 1, 1999

(C) 1999 David A. Wheeler

63

## Java Naming and Directory Interface (JNDI)

- Unified interface to multiple naming & directory services
  - E.G.: LDAP (v2 & v3), NIS(YP), NIS+, CORBA’s COS Naming, Novell NDS, DNS



December 1, 1999

(C) 1999 David A. Wheeler

64



## Java Card (Smart Cards)

- Limited space: 256 bytes RAM, 8K EEPROM, 16K ROM
- ISO 7816: command sent, card responds
- Multiple applets/card supported
- Subset JVM
  - Omits dynamic class loading, security manager, threads/synchronization, object cloning, finalization, large primitive data types (float, double, long, char)

December 1, 1999

(C) 1999 David A. Wheeler

65