Java Security: a Ten-Year Retrospective

Li Gong
Mozilla Online Ltd.
lgong@mozilla.com
www.mozillaonline.com
December 10, 2009
300~ Pages of Meeting Notes

1000~ Meetings in 30 months
Why Security Technologies Seldom Make Into Actual Mainstream Products and Systems???

• Can count notable successes on one hand
  – Firewall
  – SSL/TLS
  – One-time password
  – Maybe anti-virus for Windows
The Answer Is:

- It is a social process, not just a technology issue
- The EKE story (Bellovin/Merritt, IEEE S&P, 1992)
- “Reducing Risks from Poorly Chosen Keys” (Lomas/Gong/Needham/Saltzer, ACM SOSP 1989)
- Plus luck – at the right place and the right time; be ready to take the single available shot
Major Distractions Circa 1996/7

• Export control of crypto packages
  – Key escrow/key recovery, RSA/Bsafe/Cylink/others,
    CDSA, MS CAPI
  – Church of Cryptology

• Constant onslaught of security bugs
  – The Friday fire drills
  – Microsoft is a Java licensee; but is it a good partner?

• Where is Java security headed
  – Is it just a component of the browser? More specifically the Netscape browser?
Minor Distractions

• Protect against decompilation of Java bytecode
  – Code obfuscation
  – Encrypted bytecode
• Control of resource consumption by applets
• Java on a smartcard
• Java as e-commerce platform (Java Wallet)
• JavaOS (Java Station)
  – Security needs for a standalone OS?
• Sun company wide security architecture and strategy?
Four Major Concerns for JDK 1.2

• Usability
  – Suitable for a wide variety of applications
• Simplicity
  – Easy to understand and analyze
• Adequacy
  – Enough features before the next release
• Adaptability
  – Do not over prescribe
  – Can evolve with ease
JDK 1.2 Security Feature List
(12/11/1996)

- Project code named Gibraltar
- Features
  - Authentication
  - Delegation
  - Fine-grained access control
  - Policy management
  - Audit
  - Secret sharing
  - Key generation
  - Storage of private keys (e.g., passwords)
- Alpha (05/1997), FCS (09/1997)
Another Java security workshop

• 6/17/1997
• MSFT, Netscape, IBM, Lotus, DEC, Marimba, W3C, AT&T, Cylink, HP, Intel
12-Month Battle with Netscape

• The three battles
  – JFC vs Netscape’s IFC (combined into Swing)
  – Hotspot vs Netscape’s proposed Java VM
  – Java security vs Netscape Java security extensions

• IBM as arbitrator
  – Arbitration resolution meeting 10/15/2007
  – Don Neal overall IBM taskforce lead (Bob Blakely took over the lead 3 months later)
More “Battles”

• Customers with special requests
  – Financial (Chase, Citicorp, Amex, etc.)
  – US government agencies
  – Big corps (IBM, Lotus, Novell, etc.)
  – Startups in new fields (@Home, etc.)
  – Sun internal (pJava, eJava, enterprise groups)

• Security audit of JDK 1.2
Java Security Advisory Council (12/1997)

• Java security vs underlying OS security
  – Dependence on, exposure of, API access to, interoperable with underlying OS security features

• Theory and Practice
  – How much can we apply existing theories and tools in semantics, analysis, certification, verification, assurance

• Secure distributed computing needs
  – Authentication, authorization, secure transaction, fault tolerance, agents and mobile computing

• Real-world impact
  – Users, developers, sys adms, educators, public opinion
Technical Example 1

• Implementation least privilege at the system level in JDK 1.2 turned out to be easier and more robust than a “bolted-on” binary sandbox model in JDK 1.0/1.1
Technical Example 2

• Public static native void beginPrivileged()
• Public static native void endPrivileged()
• Try {
    AccessController.beginPrivileged();
    System.loadLibrary("xyz");
} finally {
    AccessController.endPrivileged();
}
Example 2 (Cont.)

- Privileged System.loadLibrary("xyz");

- somemethod() {
    AccessController.doPrivileged(new PrivilegedAction() {
        public Object run() {
            System.loadLibrary("xyz");
            return null;
        }
    });
}
Technical Example 3

• GuardedObject
  – An object containing a resource (e.g., a file) and a specific guard (a permission)
  – The resource is accessible if the permission is allowed

• Access permission is checked at the point of resource consumption, ensuring the right check is done in the right context
  – Can pass objects around freely
  – Can prepare resources before actual requests
Observations – The Good

• Java security has matured
  – From “what it is” to “how to utilize the features”
  – Did too little, too much, or just right?

• Raised the bar for everyone else
  – Anyone designing a new language/platform must consider type safety, systems security, least privilege, etc.

• Impacted thousands of programmers on their security awareness
Observations – The Bad

• Those companies who can afford the time and effort to improve security do not feel incented to spend the resources

• Those who want to differentiate from the dominate players cannot afford the time and effort

• When rarely a good security platform emerges, industry competition would not allow it to be adopted across the board
Many/any extensible systems (e.g., browser add-ons, iPhone apps) need the same sort of protection/security infrastructure, but they tend to be built on different technology platforms, so reuse is difficult or impossible.
Observations – The Ugly

• A new thing (a toy widget, scripting language, etc.) starts nice and small, with limited usage scope and no security considerations
• It gains good traction
• The feature set keeps expanding
• Soon the “small toy” resembles a full system or programming platform, except without adequate security support
“Never Forget Class Struggle!”

• Email me at lgong@mozilla.com