There's Something Stuck In My Shoe!

Reflections on the adoption of fine and course grained authorization frameworks

Larry Koved

koved@us.ibm.com

IBM T.J. Watson Research Center
Yorktown Heights, New York 10598

14 July 2010

Outline

- **Technology Transfer**
  - Metrics of Success
- **Usable Security Touchpoints**
  - Outside & Inside
- **Your Customers' (Technical) Goals**
- **Examples:**
  - Java Standard Edition
  - Java Enterprise Edition
  - Web 2.0 Mashups
- **Final Thoughts**

Technology Transfer

- **Ideas – papers, patents, concepts, tutorials, standards, …**
- **Implementation – code, pseudo code, …**
- **Assists – tools, components, …**
- ***How do <u>you</u> define successful technology transfer???***

Example *Metrics of Success* – IBM Research Metrics

- **Technical accomplishments**
  - What did you do that was new and / or interesting / useful?
- **Contributions to the company's products & services**
- **External impact**
  - People buying your products and services
  - Professional activities, including
    - Publications, presentations, standards, patents, open source, etc.
- **Leadership and teamwork**

Example *Metrics of Success* – Technology Transfer

- **Technical accomplishments**
  - What did you do that was new and / or interesting / useful?
- **Contributions to the company's products & services**
- **External impact on "customers"**
  - People buying your products and services
  - Professional activities, including
    - Publications, presentations, standards, patents, open source, etc.
- **Leadership and teamwork**

Usable Security Touch Points – Outside & Inside

- **End-user experience**
  - Mobile device
  - Interactive Voice Response (IVR)
  - Web
  - Applications
  - …
- **Programming models**
  - Used to implement systems (e.g., see above)
  - Security models *bleeds through* to the end-user experience
    - *userid/password (basic auth), session tokens, OpenID, OAuth, PKI (e.g., PGP, SSL, HTTPS, WS-*), kerberos, LDAP, Active Directory,…*

What Are Your Customers' (Technical) Goals?

How much value do they assign to security?

What are they willing to spend?
One time?
Ongoing?

Java Standard Edition

14 July 2010

Java Standard Edition

- **Write Once, Run Anywhere *(WORA)***
- **Reference monitors protects sensitive resources access**
  - *Network, file system, Java runtime resources, …*
  - Principles: *CodeSource*: {URL, digital signature on the code}
  - Authorization: Stack-based “introspection”
- **Two contrasting models for authorization policy specification:**
  - Netscape’s browser required Applets to embed security policy calls / pop-ups
    - Based on Java 1.x security
    - Complex security manager logic
    - “Breaks” applications when the Java runtime patched
  - Sun required editing of a *textual* policy database
    - Elegant and far simpler security policy evaluation
- **Either way, end-users required to be security administrators**

Policies for the Sun reference implementation

- **Embed security policies in the application JAR file**
  - Eventually implemented by *OSGi*
  - Proposed: have Applet framework prompt whether to accept or modify the embedded policies
- **Begs the question:**
  - How to construct the policy file(s)?
  - *Very hard for for large ("real world") applications*

Policies for the Sun reference implementation

- **Dynamic option:**
  - Run the code and see what Permission(s) are required and build the database from this list
    - *Inspect the call stack when authorizations are required*
  - Cover only paths through the code that are covered by the test case(s)

Static Analysis: *Tools To The Rescue!*

- **Created security analysis algorithms & tools**
  - Java 2 Permission Analysis
    - Identify the Permission(s), including the *object* and *operation(s)*
    - Call path analysis (goal: sound/complete analysis, not too conservative)
    - Automaticlly identify *AccessController.doPrivileged()* call placement
  - Other security analyses
    - E.g., mutability / constants, scope reduction (public, protected, private)
  - Code signing
  - Etc.
- **Packaging**
  - Text / HTML
  - Eclipse IDE integration – SWORD4J
    - Permission Analysis++
- ***Substantially* reduced the "cost" of Java security analysis**
  - *Ongoing maintenance costs*
- **Successfully applied to several products**
  - Either
    - Required – compliance or needed for competitive reasons, or
    - Desire for tighter security – customer demand

Lessons Learned

- **Naïve assumptions**
  - Products would *want* or *need* Java security – willing to expend required resources
- **Some products adopted Java security**
  - Were motivated – standards/compliance, customer demand
  - Having prior working relationship with the development group was *very* helpful
- **Tooling made "Java security enablement" tractable (feasible, affordable)**

- **Target "product" must have *sufficient* interest**
  - Can be harder in the Open Source community
    - Even with "free" tools

- **For server-side, composite / dynamically loaded applications were not a concern**
  - Wrong security model for enterprise (web) applications

- **Too expensive to maintain secure Java code**
  - Even for the right target system, "costs" can be overwhelming if not sufficiently motivated
  - See "*Making Security Accessible to Programmers: Lessons Learned*"

Net Results: Mixed Success

- **Limited adoption of Java security**
  - Needed for compliance and/or meet customer demand
  - Adopters generally violate the Principle of Least Privilege
    - Large parts of the code base are assigned *AllPermission*

- **But *too expensive to maintain***
  - Model is too far complex (e.g., stack introspection, taint analysis, …)
  - Time consuming, even with tools
    - First time. Every time code changes.
  - Tools have limitations – soundness and completeness, measurable
  - Leaves some products with less security that is desirable

  - ***Web application security has comparable cost / complexity***

- **SWORD4J is available (free) via IBM's *alphaWorks* web site**

Java Enterprise Edition

14 July 2010

Java Enterprise Edition

- **Multi-company effort**
  - Excellent working relationship with “standards” group
  - Standards group motivate to have a “secure” standard
- **Access control on *function*, not *data* (!)**
  - Role-based
  - URI’s, EJB methods – course-grained authorization
  - Much, much simpler model
    - Even so, could get complex for “real world” web sites
- **Data protection – declaratively specified**
  - Declarative specification of authentication (none, basic auth, forms based auth)
  - Channel security – integrity, confidentiality
- **Specification only**
  - No reference implementation
    - Sun Microsystems produced one?
  - Implementation up to the compliant vendors
- **Usability is (largely) up to the implementing vendors**
  - E.g., based on use cases

Advice From a Wise Sage

- **A non-security mentor's advise:**
  - If it is in the standard, it must be implemented

- ***However,***
  - *Be careful what you wish for*

Lessons Learned

- **Success!**
  - Demand for security
  - Very good working relationship with the standards and product teams
  - Role-based access control in the specification
  - Implemented by all JEE compliant vendors

- **However,**
  - Function-centered authorization
    - Many (most?) authorization use cases are around data access
      - E.g., access to *your* bank account, not *any* bank account
      - *Authorization logic ends up in the application* ← *not declarative!*
        - > Against the intent of the specification,
          no reasonable alternatives afforded by specification
        - > Data-centric authorization proposals never became part of the specification
  - In practice, few roles are defined
    - Possible violation of the principle of least privileged
  - There are security vulnerabilities in the web programming model
    - E.g., injection attacks (not unique to Enterprise Java)
  - ***Design by committee has limitations***
    - *Usable (and complete) security may not be a priority*

Web 2.0 Mashup Security

14 July 2010

Web 2.0 Mashup Security

- **Web 2.0 and *mashups* were, in general, considered insecure**
  - "Best Practices" *encouraged* bypassing security
    - Bypass browser *Same Origin Policy* using a proxy server
    - Insecure handling of identity, credentials and delegation

- **Objectives:**
  - Secure cross-domain mashups, sharing of state, at "*the glass*"
    - *Secure by default*
    - Minimize programmer knowledge about (browser) security
  - Avoid fine-grained (Java Standard Edition) authorization
    - Too complicated, requiring fancy tools, high on-going maintenance costs
  - Coarse-grained authorization
    - Get as close to the data as is possible without burdening the programmer
  - Work in existing browsers
    - Critical mass (end-users) needed to be a commercially viable technology

Web 2.0 Mashup Security

- **Selected a standards group: *OpenAjax Alliance***
  - Proposed a simple security model based on existing programming model – *pub-sub*
  - Confirmed that it was compatible with existing development model
    - Extended existing programming model with security “under the covers”
  - Worked in a multi-company task force to get buy-in for new security model
- **Provided a reference implementation**
  - Available via SourceForge
- **Identify product groups needing the technology**
  - Done in parallel with the standards activity
  - Grounded the work in customer security needs
  - Identify product-based advocates with influence
    - Senior management that recognizes the security need
    - Technical staff who can execute on the vision and integrate into product

Lessons Learned

- **Focused on the user community (developers) – the Standard**
  - Got active participation from the (developer) community
    - Attention paid to their tolerance for the hoops that must be jumped
  - Simple conceptual design
    - Alan Kay: S*imple things should be simple, complex things should be possible*
  - **Secure by default**

- **Maintained contact with the standards group and implementers to ensure forward progress**
  - Follow the community if it shifts direction
    - OpenAjax → OpenSocial → *???*

- **Released open source reference implementation on *SourceForge***

- **No strong (and secure) competition**
  - Repeatedly get out the message that there is a secure alternative for mashups
    - It is consistent with other strategic directions in the organization

- **Worked the corporate politics to gain a toehold and maintain forward progress**
  - Found champions in the product and development groups
    - Took advantage of “soap box” opportunities to advertise the work
  - Grounded in customer-drive use cases that mattered
  - Maintained regular contact with the internal development community

- **Low cost to implement in product *AND* maintain its security**

Final Thoughts

- **Security technology transfer is difficult**
  - *Seems similar experiences to HCI*
- **Who are your allies in support of the "business"?**
  - What motivates them to adopt your security? What is the ***value*** to your "customer"?
    - Customer demand?
    - Standards?
    - Reputation risk?
    - How do they assess the cost / benefit tradeoff?
  - Who are your strongest champions? Business? Technology?
    - Do you have scenarios that can be validated with customers?
    - Who is the right customer? What is their feedback? Have you talked to them?
      - *Really* listen to their feedback! Understand their viewpoint. *The real security issues may be elsewhere.*
      - How does your technology fit into their business model? A cost? Provide value-add? Risk mitigation?
  - What is the competition?
    - What are the natural affordances* of your technology?
      - How good a fit is the technology to the deployment environment?
      - Who is to use the technology? How well does it match their skills, job, business needs?
      - How do the costs / value of your technology compare to the competition? What are the alternatives?
  - Are there related standards or standards groups to support your effort?
    - Are their goals in alignment with your technology (technology and business)?
    - Who is driving the effort? What are their strategic and tactical goals?
    - How broad based is their support? Is it thriving?
      What are the tactical & strategic risks to adoption?
- **How do you line up your supporters?**

* "Affordances" as interpreted by Don Norman

Many Thanks!

- Sumeer Bhola
- Marina Biberstein
- David Boloker
- Ian Brackenbury
- Suresh Chari
- Hyen V. Chung
- Don Ferguson
- Matthew Flaherty
- Frederik De Keukelaere
- Jon Ferriaolo
- Don Ferguson
- Aaron Kershenbaum
- Eric Li
- Jeff McAffer
- Bilha Mendelson
- Tony Nadalin
- Nataraj Nagaratnam
- Javier Pedemonte
- Adam Peller
- Marco Pistoia
- Sara Porat
- Jay Rosenthal
- Michael Steiner
- Naohiko Uramoto
- Sachiko Yoshihama
- Mary Ellen Zurko

- Members of the OpenAjax Alliance and the Security Task Force
- Members of the Eclipse Equinox project
- Supportive management and sponsors
- ***Others whom I have inadvertently omitted (I apologize!).***