Making Security Accessible to Programmers

Lessons Learned

Larry Koved
IBM T. J. Watson Research Center
The Inspiration

• SABER
  - Bug finding in very large web apps.
    • A single line of code can bring down a server

• Based on our experience with securing Eclipse RCP runtime
  - Component-based standalone apps.
// The challenge:

// How to make security accessible to
// ordinary programmers?
// Without becoming a security expert
//Start:
// How to secure Java code?
// How do you keep it secured?
• Reflections secure web programming?
  -(as time allows)
  -Is it really different?
</development phase>

</little testing>
Here we go!

(Hold on to your seats!!)
// In the beginning...
// given the software design
{ write the code}
{ write test cases }
// all goes well!
// life is good!!
// “We going live on the web”
// “Make sure it is secure!!!”
PANIC!!!
// Find Java security books
// become a security expert
// Many Java security topics

// What are the key issues?
// For this discussion, the Java security essentials are the 3 legs:

1. Bytecode verifier
2. ClassLoader
3. SecurityManager
// Specifically, the Java authorization model

// Stack-based authorization
// CodeBase:
  // URL from which the code is loaded
  // Digital signature
Quick Review

Authorization sequence

AccessController.checkPermission(p)
SecurityManager.checkPermission(p)
SecurityManager.checkXXX(p)
org.foo.library.Cl2.Method()
org.foo.library.Cl1.Method()
com.bar.Cl0.Method()
Quick Review

Calling sequence

Authorization sequence

com.bar.Cl0.Method()
org.foo.library.Cl1.Method()
AccessController.doPrivileged()
org.foo.library.Cl1$.run()
org.foo.library.Cl2.Method()
SecurityManager.checkXXX(p)
SecurityManager.checkPermission(p)
AccessController.checkPermission(p)
// Permissions required?
{ install SecurityManager }

(magic incantation goes here)
How do we determine Permissions for the code?
// test the component for required Permissions
repeat {
    { write test cases }
    { run code }
    // review security exceptions
    // inspect stack trace
    // figure out the CodeBase missing Perms (!)
    { insert privileged code where appropriate }
    { grant permissions - use policy tool or file }
} until (code runs without SecurityExceptions)
// Is it secure yet?

(refrain)
Public void foo(String fn)
{
    
    . . .

    cmd = "/bin/cat /tmp/" + fn;
    System.exec(cmd);

    . . .

}
// injection attack

// e.g., command injection
Public void foo(String fn) {
    . . .
    cmd = "/bin/cat /tmp/" + fn;
    System.exec(cmd);
    . . .
}
Suppose the value of \texttt{fn} was:

```
"../../etc/passwd"
```
or

"<< /dev/null; rm -rf /*"
// tainted data detection
// manually inspect code for
// tainted data flows to
// privileged operations
{
    // “compute” program slices
    {sanitize inputs } +
    { removed privileged code }*
    { refine the Java Permission(s) }
} +
// test the component for required Permissions
repeat {
    { write test cases }
    { run code }
    // review security exceptions
    // inspect stack trace
    // figure out the CodeBase missing Perms (!)
    { insert privileged code where appropriate }
    { grant permissions - use policy tool or file}
} until (code runs)
// Is it secure yet?

(refrain)
// “Please make it reusable!”
{ turn it into a component }

// For now, use the OSGi framework.
{ create bundles, fragments, etc. }

// jar files in a specific format
// Same security semantics, // different policy syntax

{write AWK script to reformat}
{ write test cases }

// all goes well.
// life is good!!
{ write test cases }
// all goes well!
// life is good!!
// Is it secure yet?

(refrain)
// hostile shared memory
// environment
// any accessible
// mutable state?
// What are the Java accessibility rules
{ discover by writing test cases }
{set public / protected fields final}

{ make default fields final }

or

{ seal the package }
// Is it secure yet?

(refrain)
// final is not the same as
// C++ const
public static final HashTable ht = new foo();
ht.put(k, v); // mutable shared state
// find shared mutable state
// find returned mutable objects
{ return clone'd object?}

// find accessible mutable objects
{ return clone'd object?}

// don’t forget arrays!

// is the result of this testable?
// Is it secure yet?

(refrain)
// Other security best practices?
// Check for:

// SecurityManagers
// ClassLoaders
// Serialized sec. sens. fields
// other best practices
// Is it secure yet?

(refrain)
Component resources may need to be protected.

(Complete Mediation is out of scope for today's talk.)
// Now what?
// code signing!
{
    // is code signing necessary?
    // purchase certificate?
    // what are self signed certificate?
    // manage certificates?
    // learn to use the Sun signing tool?
    // where is the signing information?
}
// Ready to deploy

(sort of)
Let's review....
The steps (1 of 3)

• **Learn Java security**
  - CodeBase
  - ProtectionDomains
  - Code signing
  - Certificate management
  - SecurityManager
  - Stack-based access control
  - Permissions
  - Privileged code
  - Etc.
The steps (2 of 3)

- Install SecurityManager
- Figure out Permissions by CodeBase
  - Write & run test cases*
- Protect component-specific resources
- Injection attacks - tainted data
  - Write & run test cases*
- Turn into a component (OSGi)
  - Write & run test cases*
The steps (3 of 3)

• **Mutable shared state**
  - Discover accessibility rules
  - Accessible fields, objects, return objects
  - Write & run test cases?

• **Other security best practices**
  - Research & conduct code reviews
  - Write & run test cases?

• **Sign code; Manage Certificates**
  - **Repeat process as code is updated**
    - Integrate into the system build process
< securing code is too hard >
hmmm
< find an easier way >
While writing code, ...
Conform to security best practices

As integral part of the coding environment
Identify Required Permissions

• Show slice(s) - why Permissions are required.
  - Allow for source code navigation
• Add the required permissions to the appropriate configuration file(s)
  - As a “quick fix”
Add appropriate privileged code

• Show slice(s) - where privileged code may be inserted
  - Allow for source code navigation
• Add privileged code blocks “quick fixes”
Protect code-specific resources

(A topic for another time)
Check for tainted data in privileged operations

• Show slice(s) – c.f. & d.f.
  - Allow for source code navigation
• Add sanitizing code /
  modify privileged code “quick fixes”
Restrict access to mutable shared state

• Which fields / objects are problematic?
• Show slice(s) - c.f. & d.f.
  - Allow for source code navigation
• Add “quick fixes”? 
Write & run test cases

(What cases need to be created?)
Sign code (when appropriate)

(Not a separate tool)
Unit testing

• When are SecurityExceptions thrown?
• Which CodeBases need additional Permissions?
Repeat (as appropriate)
Add to system test
Repeat (as appropriate)
< "Maintenance" phase >
Lower cost?
Continuous?
SWORD4J

Security Workbench
Development Environment
for Java
// security rules as part of
// the Eclipse Java IDE

• Continuous feedback to the developer
• Supports system build
• SWORD4J is on IBM’s alphaWorks web site
```java
URLClassLoader loader;
loader = new URLClassLoader(urls, this.getClass().getClassLoader());
return loader;

public void foo(System.out)
{
    foo(null);
}
```

Converts the required permissions from analysis format into OSGi encoded form, and adds them to the OSGI-INF/permissions.perm file of your plug-in, OSGi-bundle, or Java project.
What about the Web?
// PHP
// JavaScript
And more!
(Oh my!!)
hmmm
// Lots of “best practices”
// Hard to secure
// Enable Apache security?
// Enable PHP security?
// Which config settings?
// How do they related to the Apache security settings?
// How to detect tainted data?
// What are the trusted sinks?
// What are the untrusted sources?
// How do I test all of this?
// What are the risks of reuse?
< RSS / ATOM / Mashups / ... >
// What about Web 2.0?!!
// Are my mashup secure?
(Sort of like a component model)
// client-side hostile shared memory environment

Indeed, a gadget with dancing pigs!
// SMash!!! OpenAjax Alliance

Inter Mashlet Communication

Vacation Planner Mashup

Activity Planner
Select the day: Sunday
Select the activity: Park
Add

Day | Activity
---|---
Saturday | Golf
Sunday | Park

Activity Locator

Mapping Table

<table>
<thead>
<tr>
<th>Mashlet Topic</th>
<th>Hub Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.vacation.planner.activity</td>
<td>hub.activity</td>
</tr>
<tr>
<td>com.vacation.locator.activity</td>
<td>hub.activity</td>
</tr>
</tbody>
</table>

Logical Hub 1.1

Hub 1.1 Instance

S. Bhola, M. Steiner, S. Chari, F. De Keuklaere
// How do I authenticate, from the browser thru a man-in-the-middle?
<You can't get there from here!>
(A story for another day)
< You get the general idea >
Secure software
Hopeless ?
// Are there success stories?
Secure Mashups (SMash)

See OpenAjax Alliance Hub 2.0
// Can we do more?
YES!!!
// Thank you.