# Twelve Java™ Technology Security Traps and How to Avoid Them

**Brian Chess, PhD**

Chief Scientist
Fortify Software
http://www.fortifysoftware.com

TS-1660

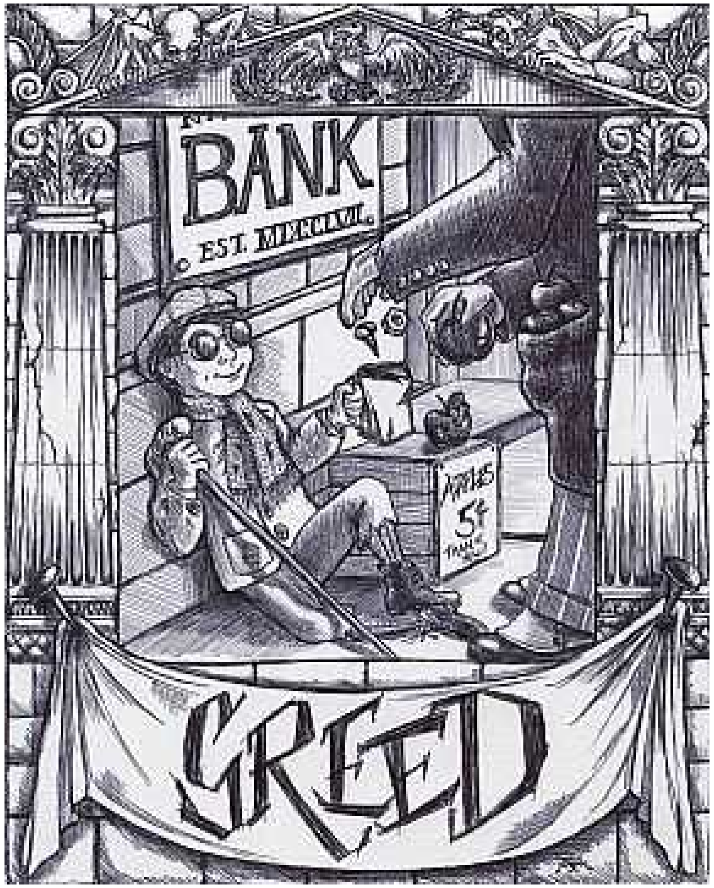
## Overview

- Just 12?
- The 12 traps
  - In: Mistakes, errors, boo-boos, oversights, and gotchas; Very network/web oriented
  - Out: Gnarly discussion of crypto APIs, single sign-on packages, other security features
- Getting security right: Policy, process, and tools

Cave cave deus videt
Abscondam faciem meam ab eis: et considerabo novissima eorum

## The Seven Deadly Sins

## The Seven Deadly Sins

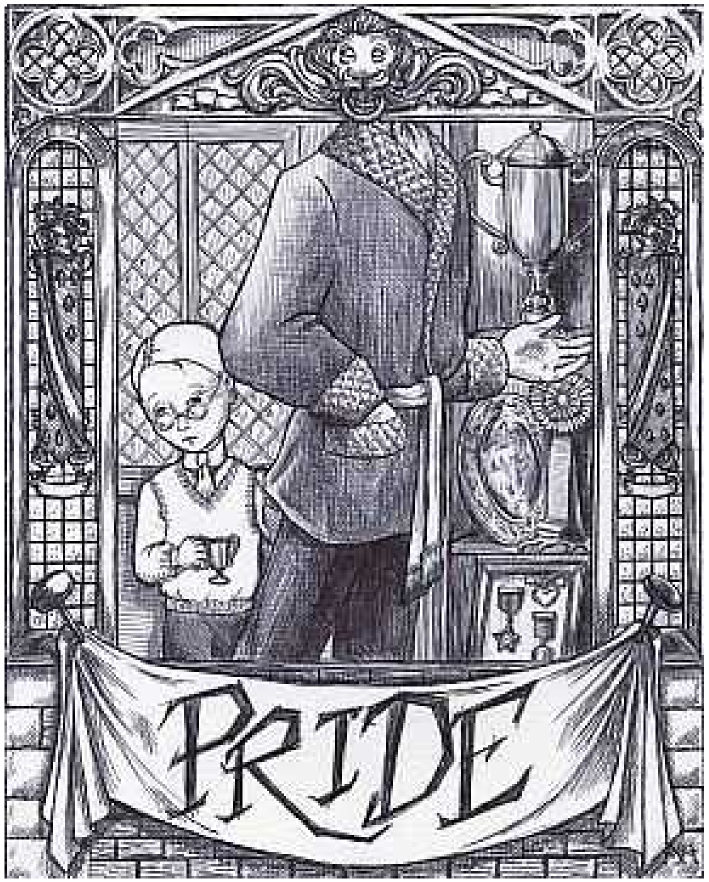


## The Seven Deadly Sins

## The Seven Deadly Sins

## The Seven Deadly Sins

# The Seven Deadly Sins



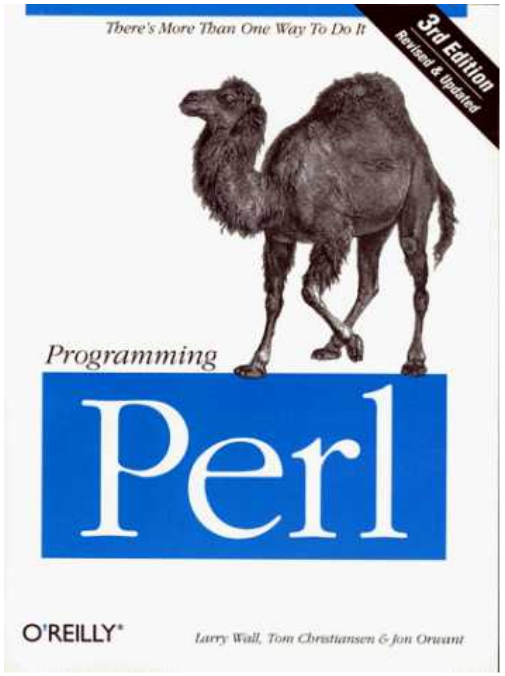
## The Seven Deadly Sins

## The Seven Deadly Sins

- Pope Gregory I
- 6th Century
- Why talk about sins?
  - Understand your actions
  - Know what to confess

## The Three Virtues of a Programmer

- Laziness
- Impatience
- Hubris


There's More Than One Way To Do It
3rd Edition
Revised & Updated
Programming
Perl
O'REILLY
Larry Wall, Tom Christiansen & Jon Orwant

Larry Wall is a subversive.

## Values Programmers Are Taught

- Cleverness
- Simplicity
  - Encapsulation
  - Reuse
- Performance
  - Time
  - Space

## Values Programmers Are Not Taught

- Reliability
  - Conflicts with simplicity and performance
- Longevity
  - Maintainability

## Mistakes That Put Software in Jeopardy

7* Pernicious Kingdoms:

A taxonomy of programming errors that affect security

1. Input validation and representation
2. API abuse
3. Security features
4. Time and state

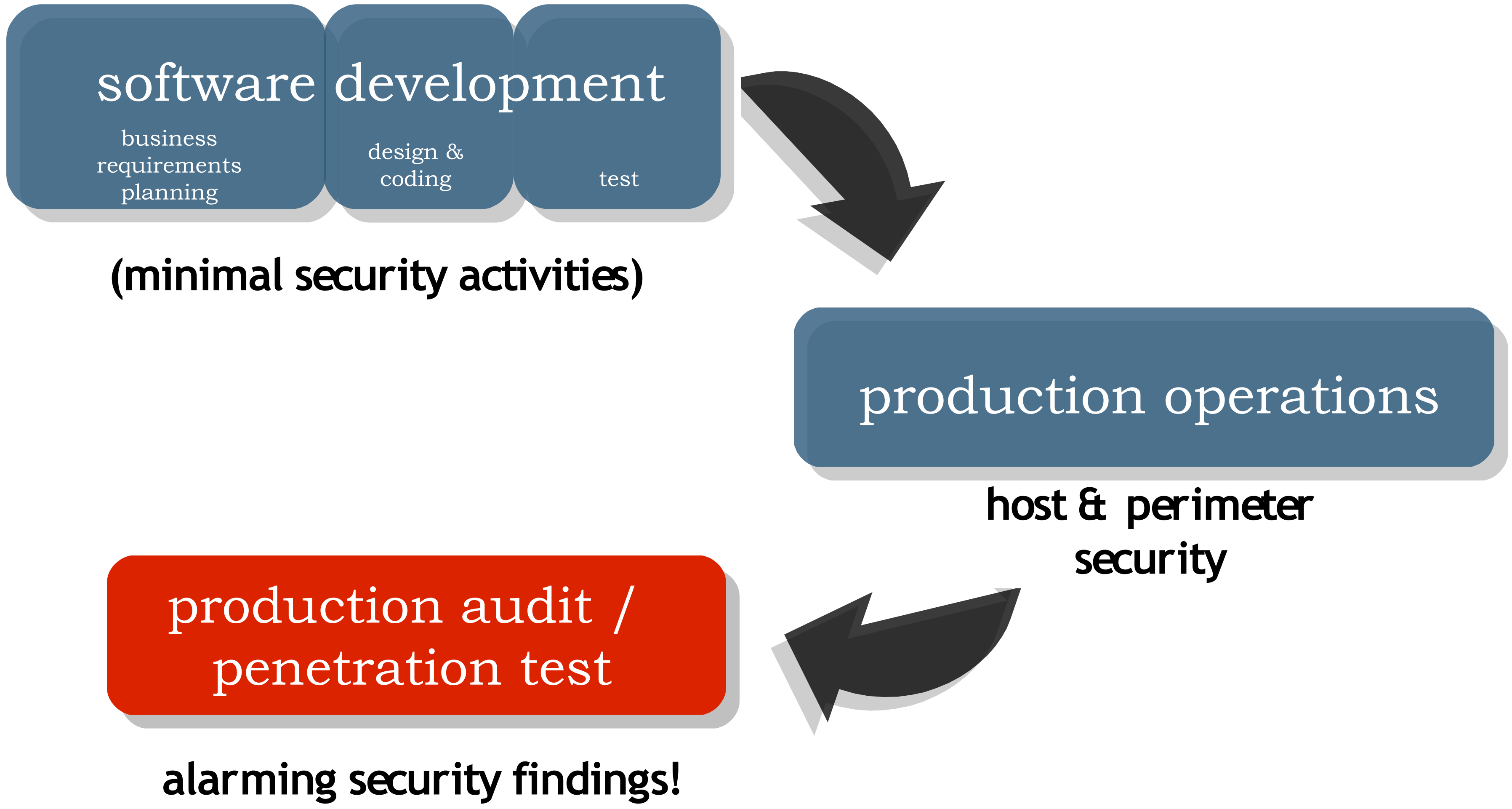
- Error Handling
- Code quality
- Encapsulation

* Environment

## Mistakes That Put Software in Jeopardy

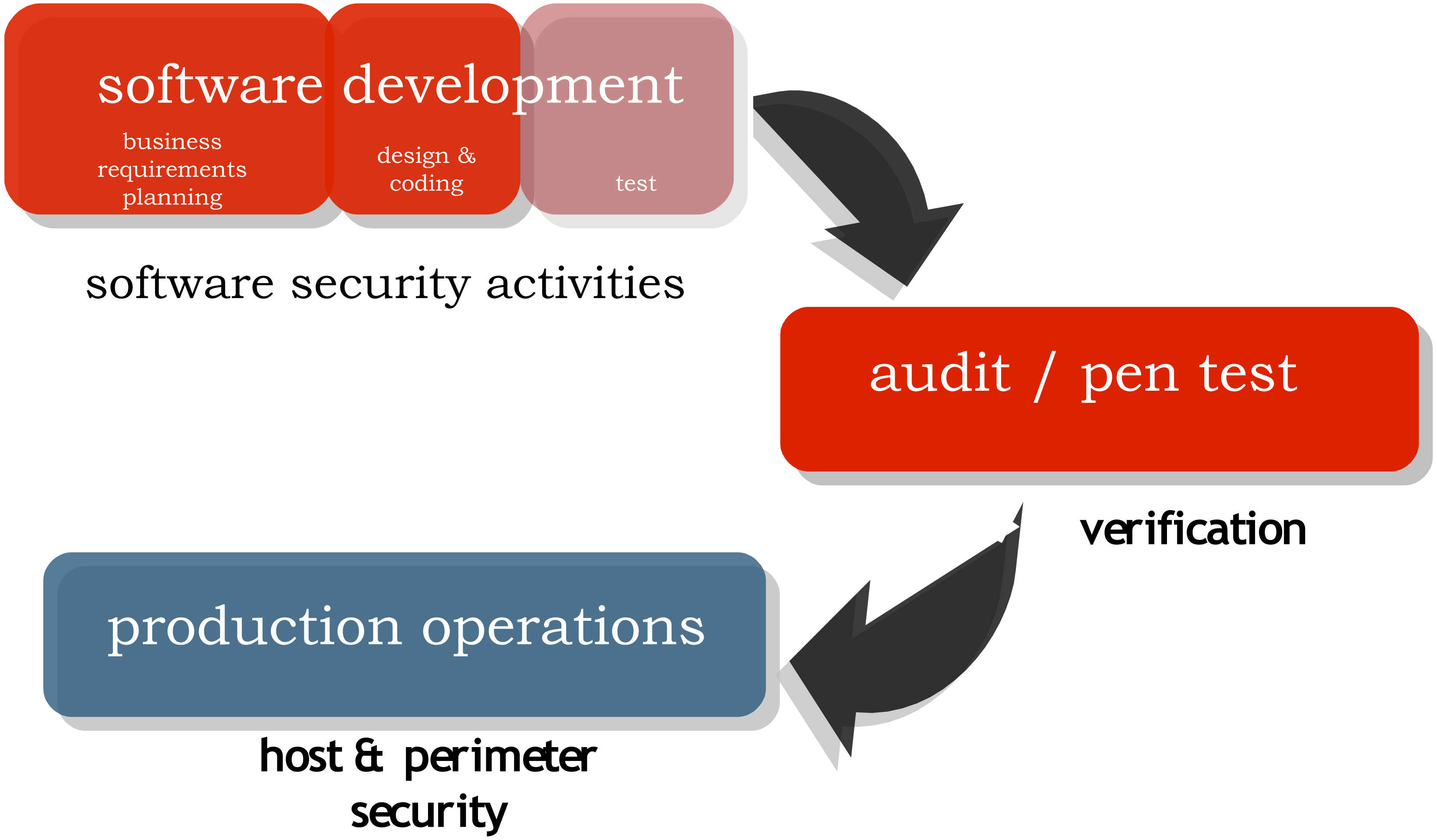
1. Software security can be handled as a sequence of bugs

## Current Solution: Build Then Measure



software development
business requirements planning
design & coding
test
(minimal security activities)
production operations
host & perimeter security
production audit / penetration test
alarming security findings!

## Real Solution



## Mistakes That Put Software in Jeopardy

1. Failure to understand how the system works

## Mistakes That Put Software in Jeopardy

1. Failure to consider what could go wrong

# JAVA JEOPARDY!

Trust Me, I’m a Browser!

Configuration Consternation

You Can’t Trust Anybody These Days

It’s Not a Problem Because…I Used Java™ Technology

HTTP, How I Love Thee

## Trust Me, I’m a Browser!

```
rs = stmt.executeQuery(
  "select * from users "
  "where uname = '" + uName
  + "'");
```

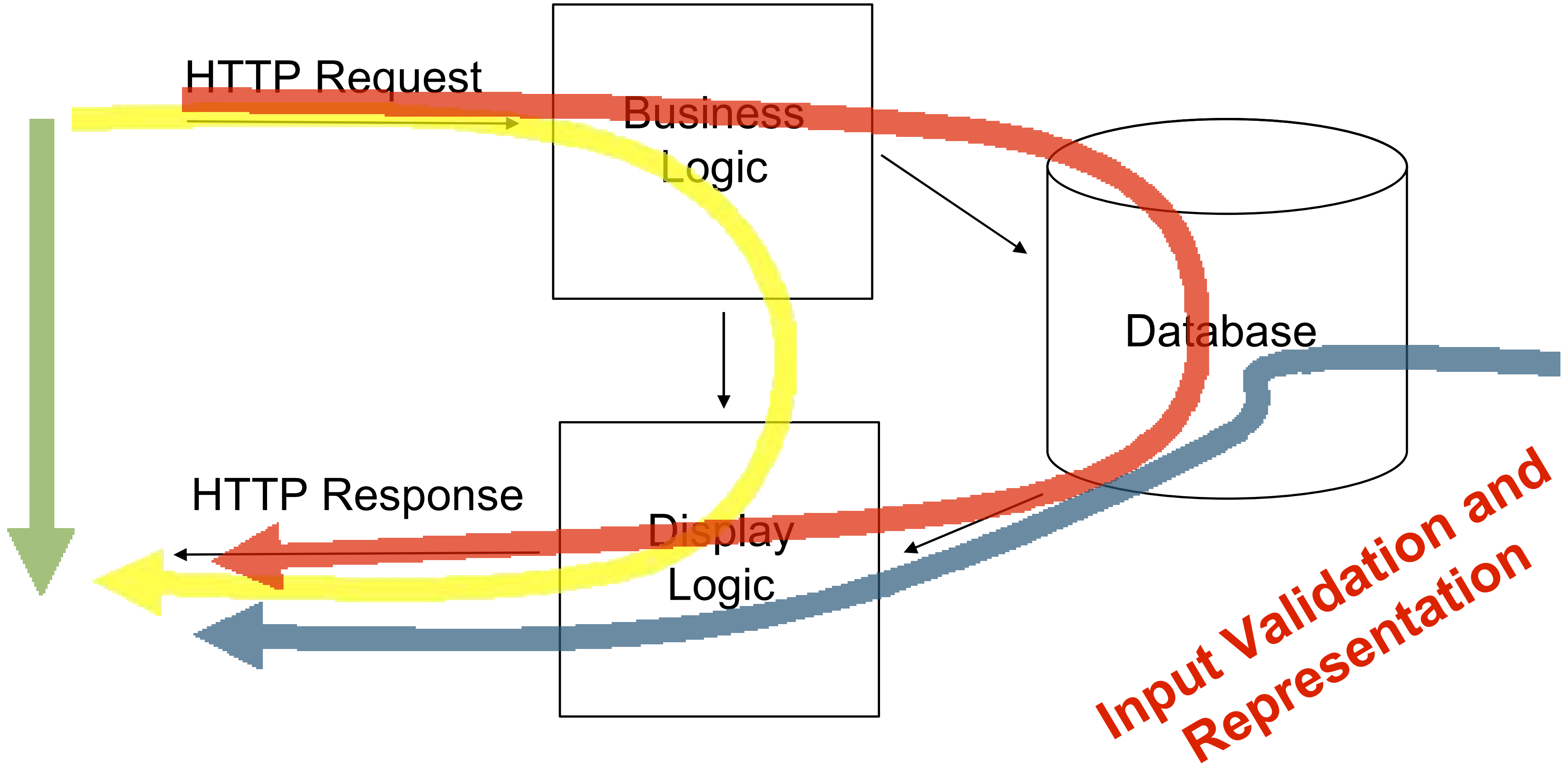
## #1 SQL Injection (And More)

- Injection attacks
  - SQL Injection
  - Command injection
  - File system traversal
  - XML injection
- Defense
  - Prepared statements (bind variables)
  - Whitelist
  - Blacklist

**Input Validation and Representation**

```
out.println(
“malformed input: “ + queryParameter);
```

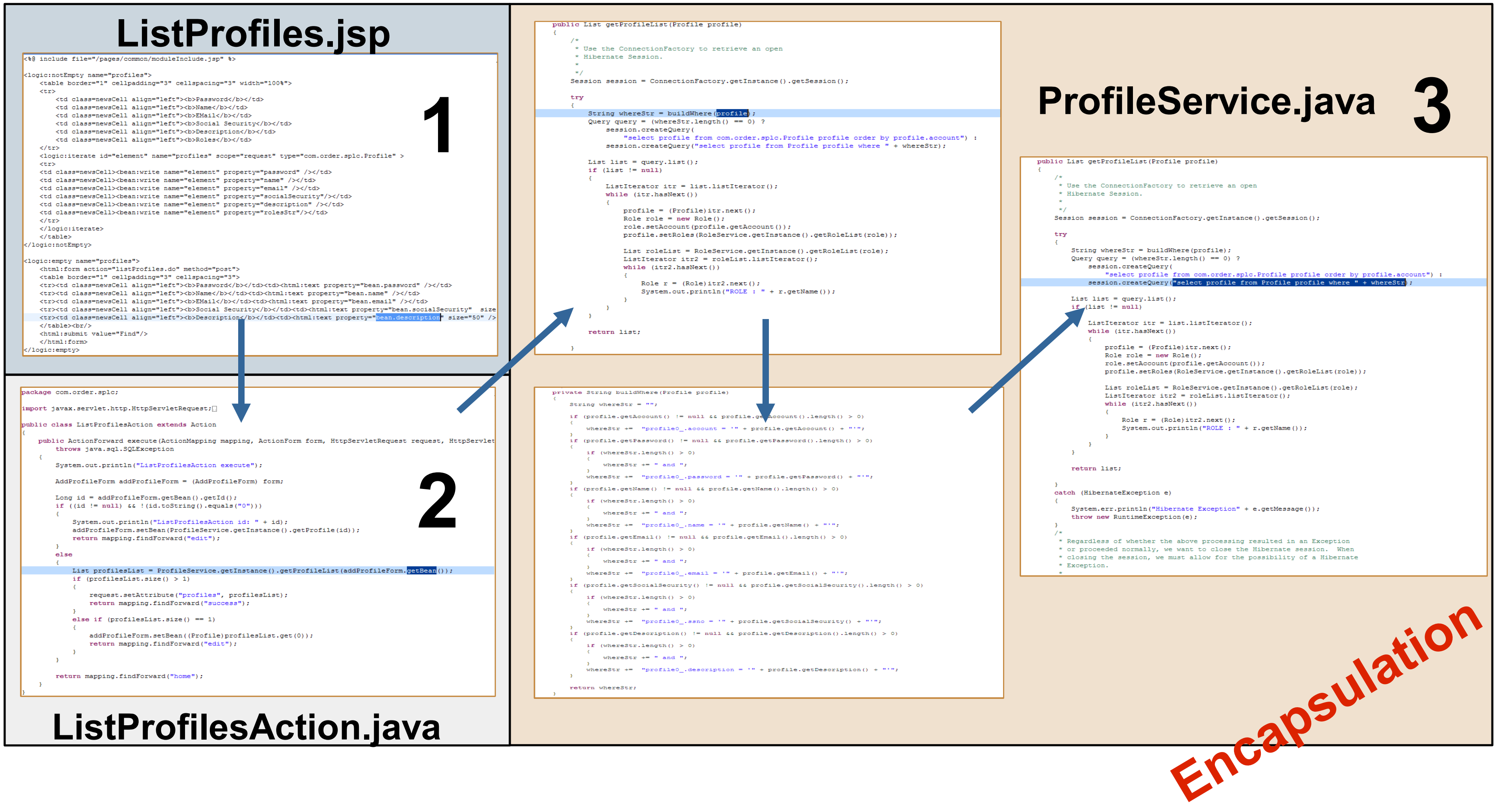
## #2 Cross-site Scripting (XSS)



## Bonus

```
n = req.getParameter(“name”);
session.setAttribute(“name”, n);
```

# Bonus: Trust Boundary Errors



```
<%@ include file="/pages/common/moduleInclude.jsp" %>

<logic:notEmpty name="profiles">
    <table border="1" cellpadding="3" cellspacing="3" width="100%">
    <tr>
        <td class=newsCell align="left"><b>Password</b></td>
        <td class=newsCell align="left"><b>Name</b></td>
        <td class=newsCell align="left"><b>EMail</b></td>
        <td class=newsCell align="left"><b>Social Security</b></td>
        <td class=newsCell align="left"><b>Description</b></td>
        <td class=newsCell align="left"><b>Roles</b></td>
    </tr>
    <logic:iterate id="element" name="profiles" scope="request" type="com.order.splc.Profile" >
    <tr>
    <td class=newsCell><bean:write name="element" property="password" /></td>
    <td class=newsCell><bean:write name="element" property="name" /></td>
    <td class=newsCell><bean:write name="element" property="email" /></td>
    <td class=newsCell><bean:write name="element" property="socialSecurity"/></td>
    <td class=newsCell><bean:write name="element" property="description" /></td>
    <td class=newsCell><bean:write name="element" property="rolesStr"/></td>
    </tr>
    </logic:iterate>
    </table>
</logic:notEmpty>

<logic:empty name="profiles">
    <html:form action="listProfiles.do" method="post">
    <table border="1" cellpadding="3" cellspacing="3">
    <tr><td class=newsCell align="left"><b>Password</b></td><td><html:text property="bean.password" /></td>
    <tr><td class=newsCell align="left"><b>Name</b></td><td><html:text property="bean.name" /></td>
    <tr><td class=newsCell align="left"><b>EMail</b></td><td><html:text property="bean.email" /></td>
    <tr><td class=newsCell align="left"><b>Social Security</b></td><td><html:text property="bean.socialSecurity"  size
    <tr><td class=newsCell align="left"><b>Description</b></td><td><html:text property="bean.description" size="50" />
    </table><br/>
    <html:submit value="Find"/>
    </html:form>
</logic:empty>
```

```
package com.order.splc;

import javax.servlet.http.HttpServletRequest;

public class ListProfilesAction extends Action
{
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServlet
        throws java.sql.SQLException
    {
        System.out.println("ListProfilesAction execute");

        AddProfileForm addProfileForm = (AddProfileForm) form;

        Long id = addProfileForm.getBean().getId();
        if ((id != null) && !(id.toString().equals("0")))
        {
            System.out.println("ListProfilesAction id: " + id);
            addProfileForm.setBean(ProfileService.getInstance().getProfile(id));
            return mapping.findForward("edit");
        }
        else
        {
            List profilesList = ProfileService.getInstance().getProfileList(addProfileForm.getBean());
            if (profilesList.size() > 1)
            {
                request.setAttribute("profiles", profilesList);
                return mapping.findForward("success");
            }
            else if (profilesList.size() == 1)
            {
                addProfileForm.setBean((Profile)profilesList.get(0));
                return mapping.findForward("edit");
            }
        }

        return mapping.findForward("home");
    }
}
```

```
    public List getProfileList(Profile profile)
    {
        /*
         * Use the ConnectionFactory to retrieve an open
         * Hibernate Session.
         *
         */
        Session session = ConnectionFactory.getInstance().getSession();

        try
        {
            String whereStr = buildWhere(profile);
            Query query = (whereStr.length() == 0) ?
                session.createQuery(
                    "select profile from com.order.splc.Profile profile order by profile.account") :
                session.createQuery("select profile from Profile profile where " + whereStr);

            List list = query.list();
            if (list != null)
            {
                ListIterator itr = list.listIterator();
                while (itr.hasNext())
                {
                    profile = (Profile)itr.next();
                    Role role = new Role();
                    role.setAccount(profile.getAccount());
                    profile.setRoles(RoleService.getInstance().getRoleList(role));

                    List roleList = RoleService.getInstance().getRoleList(role);
                    ListIterator itr2 = roleList.listIterator();
                    while (itr2.hasNext())
                    {
                        Role r = (Role)itr2.next();
                        System.out.println("ROLE : " + r.getName());
                    }
                }
            }

            return list;

        }
```

```
    private String buildWhere(Profile profile)
    {
        String whereStr = "";

        if (profile.getAccount() != null && profile.getAccount().length() > 0)
        {
            whereStr +=  "profile0_.account = '" + profile.getAccount() + "'";
        }
        if (profile.getPassword() != null && profile.getPassword().length() > 0)
        {
            if (whereStr.length() > 0)
            {
                whereStr += " and ";
            }
            whereStr +=  "profile0_.password = '" + profile.getPassword() + "'";
        }
        if (profile.getName() != null && profile.getName().length() > 0)
        {
            if (whereStr.length() > 0)
            {
                whereStr += " and ";
            }
            whereStr +=  "profile0_.name = '" + profile.getName() + "'";
        }
        if (profile.getEmail() != null && profile.getEmail().length() > 0)
        {
            if (whereStr.length() > 0)
            {
                whereStr += " and ";
            }
            whereStr +=  "profile0_.email = '" + profile.getEmail() + "'";
        }
        if (profile.getSocialSecurity() != null && profile.getSocialSecurity().length() > 0)
        {
            if (whereStr.length() > 0)
            {
                whereStr += " and ";
            }
            whereStr +=  "profile0_.ssno = '" + profile.getSocialSecurity() + "'";
        }
        if (profile.getDescription() != null && profile.getDescription().length() > 0)
        {
            if (whereStr.length() > 0)
            {
                whereStr += " and ";
            }
            whereStr +=  "profile0_.description = '" + profile.getDescription() + "'";
        }

        return whereStr;
    }
```

```
    public List getProfileList(Profile profile)
    {
        /*
         * Use the ConnectionFactory to retrieve an open
         * Hibernate Session.
         *
         */
        Session session = ConnectionFactory.getInstance().getSession();

        try
        {
            String whereStr = buildWhere(profile);
            Query query = (whereStr.length() == 0) ?
                session.createQuery(
                    "select profile from com.order.splc.Profile profile order by profile.account") :
                session.createQuery("select profile from Profile profile where " + whereStr);

            List list = query.list();
            if (list != null)

                ListIterator itr = list.listIterator();
                while (itr.hasNext())
                {
                    profile = (Profile)itr.next();
                    Role role = new Role();
                    role.setAccount(profile.getAccount());
                    profile.setRoles(RoleService.getInstance().getRoleList(role));

                    List roleList = RoleService.getInstance().getRoleList(role);
                    ListIterator itr2 = roleList.listIterator();
                    while (itr2.hasNext())
                    {
                        Role r = (Role)itr2.next();
                        System.out.println("ROLE : " + r.getName());
                    }
                }
            }

            return list;

        }
        catch (HibernateException e)
        {
            System.err.println("Hibernate Exception" + e.getMessage());
            throw new RuntimeException(e);
        }
        /*
         * Regardless of whether the above processing resulted in an Exception
         * or proceeded normally, we want to close the Hibernate session.  When
         * closing the session, we must allow for the possibility of a Hibernate
         * Exception.
         *
```

## Bonus: Trust Boundary Errors

- Moral to the story: Do not use the same container to store both trusted and untrusted data

# Configuration Consternation

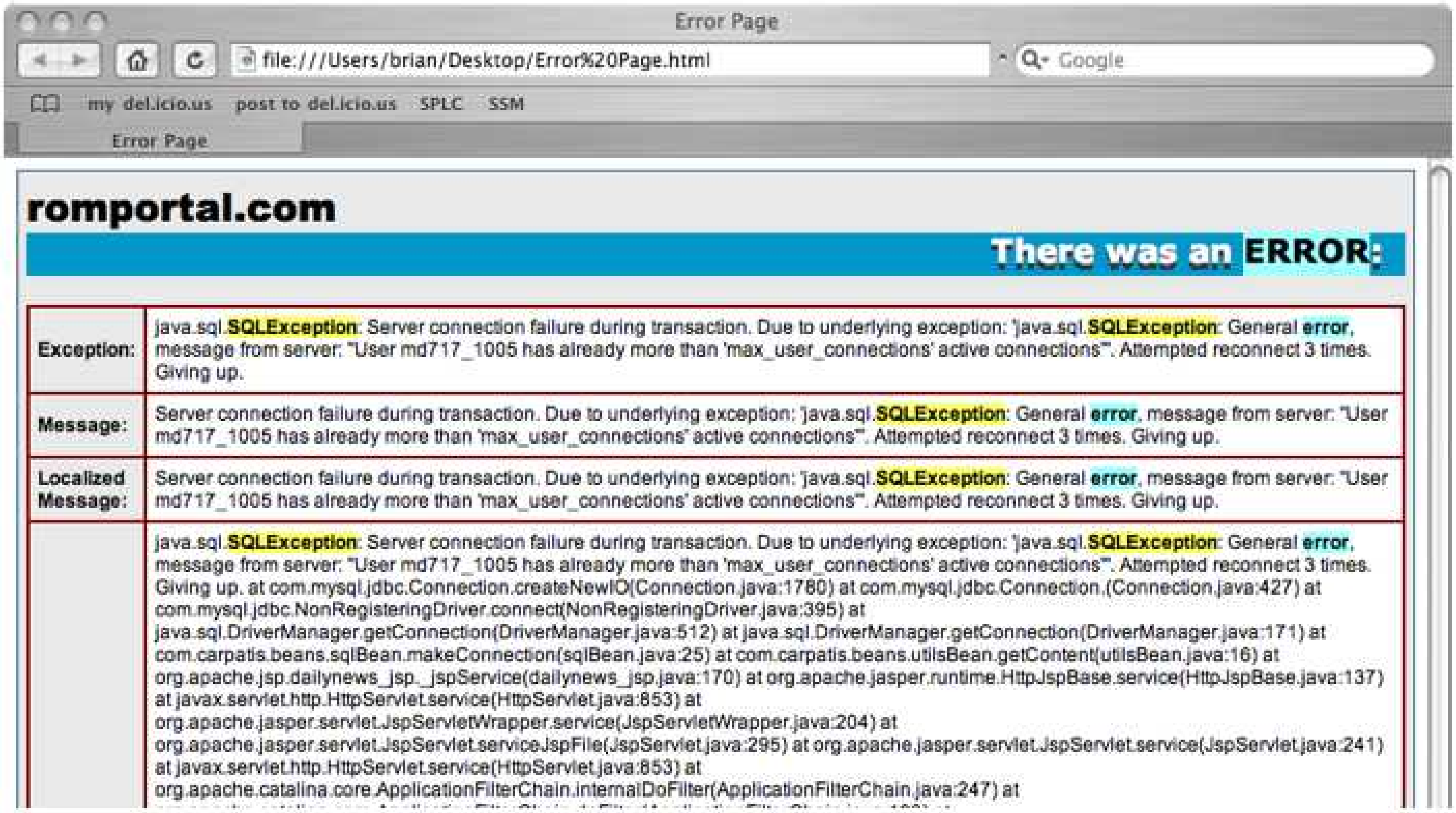
```
conn = DriverManager.getConnection
      (connStr, "scott", "tiger");
```

## #3: Bad Credential Management

Also popular

- Store cleartext username/password/key in configuration and/or properties file
- Protect with trivial encoding (base64, rot13)
- Ditto for encryption keys

Security Features



romportal.com

There was an ERROR:

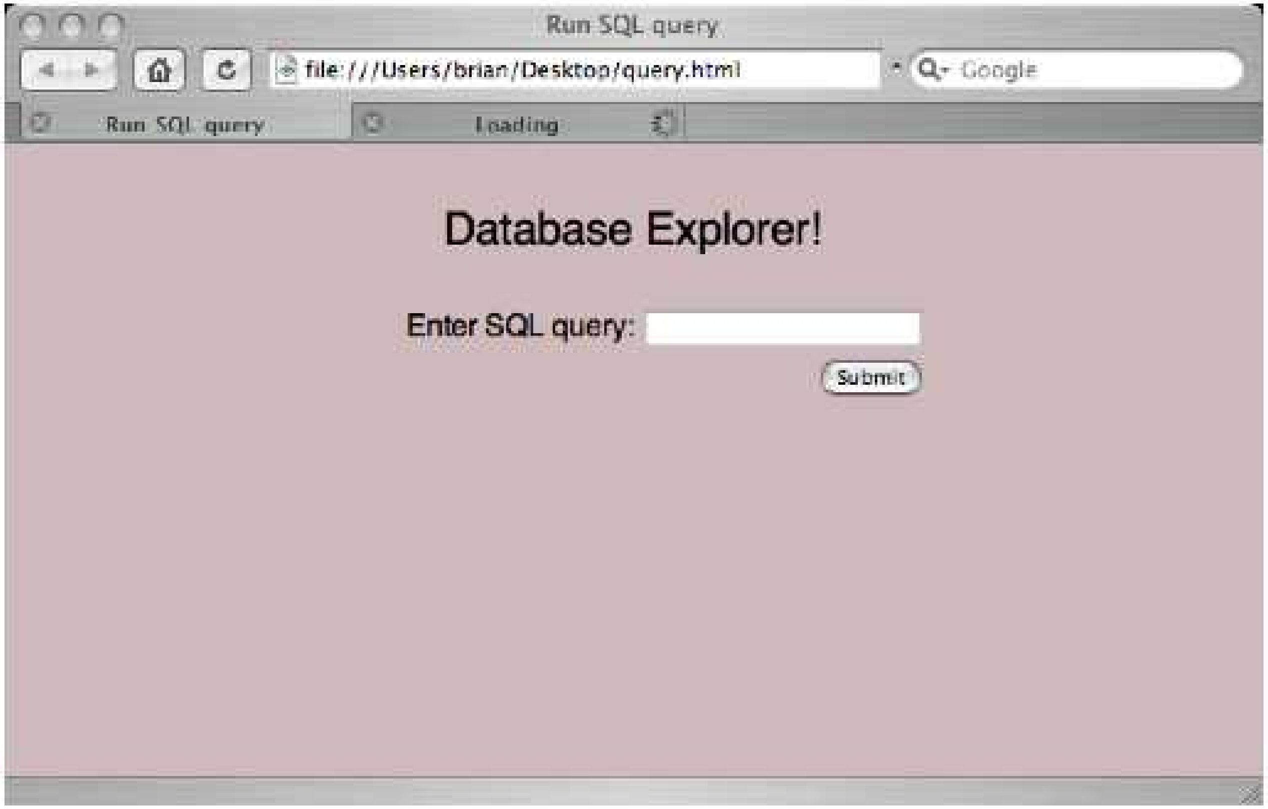
| | |
|---|---|
| **Exception:** | java.sql.SQLException: Server connection failure during transaction. Due to underlying exception: 'java.sql.SQLException: General error, message from server: "User md717_1005 has already more than 'max_user_connections' active connections"'. Attempted reconnect 3 times. Giving up. |
| **Message:** | Server connection failure during transaction. Due to underlying exception: 'java.sql.SQLException: General error, message from server: "User md717_1005 has already more than 'max_user_connections' active connections"'. Attempted reconnect 3 times. Giving up. |
| **Localized Message:** | Server connection failure during transaction. Due to underlying exception: 'java.sql.SQLException: General error, message from server: "User md717_1005 has already more than 'max_user_connections' active connections"'. Attempted reconnect 3 times. Giving up. |
| | java.sql.SQLException: Server connection failure during transaction. Due to underlying exception: 'java.sql.SQLException: General error, message from server: "User md717_1005 has already more than 'max_user_connections' active connections"'. Attempted reconnect 3 times. Giving up. at com.mysql.jdbc.Connection.createNewIO(Connection.java:1780) at com.mysql.jdbc.Connection.(Connection.java:427) at com.mysql.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:395) at java.sql.DriverManager.getConnection(DriverManager.java:512) at java.sql.DriverManager.getConnection(DriverManager.java:171) at com.carpatis.beans.sqlBean.makeConnection(sqlBean.java:25) at com.carpatis.beans.utilsBean.getContent(utilsBean.java:16) at org.apache.jsp.dailynews_jsp._jspService(dailynews_jsp.java:170) at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:137) at javax.servlet.http.HttpServlet.service(HttpServlet.java:853) at org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:204) at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:295) at org.apache.jasper.servlet.JspServlet.service(JspServlet.java:241) at javax.servlet.http.HttpServlet.service(HttpServlet.java:853) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:247) at |

## #4: Bad Error Handling

- Lack of top-level (global) error handling
- Lack of understanding about how valuable error messages are to an attacker

Errors

## http://www.mysite.com/admin/query.html



Database Explorer!

Enter SQL query:

## #5: Test Code Goes to Production

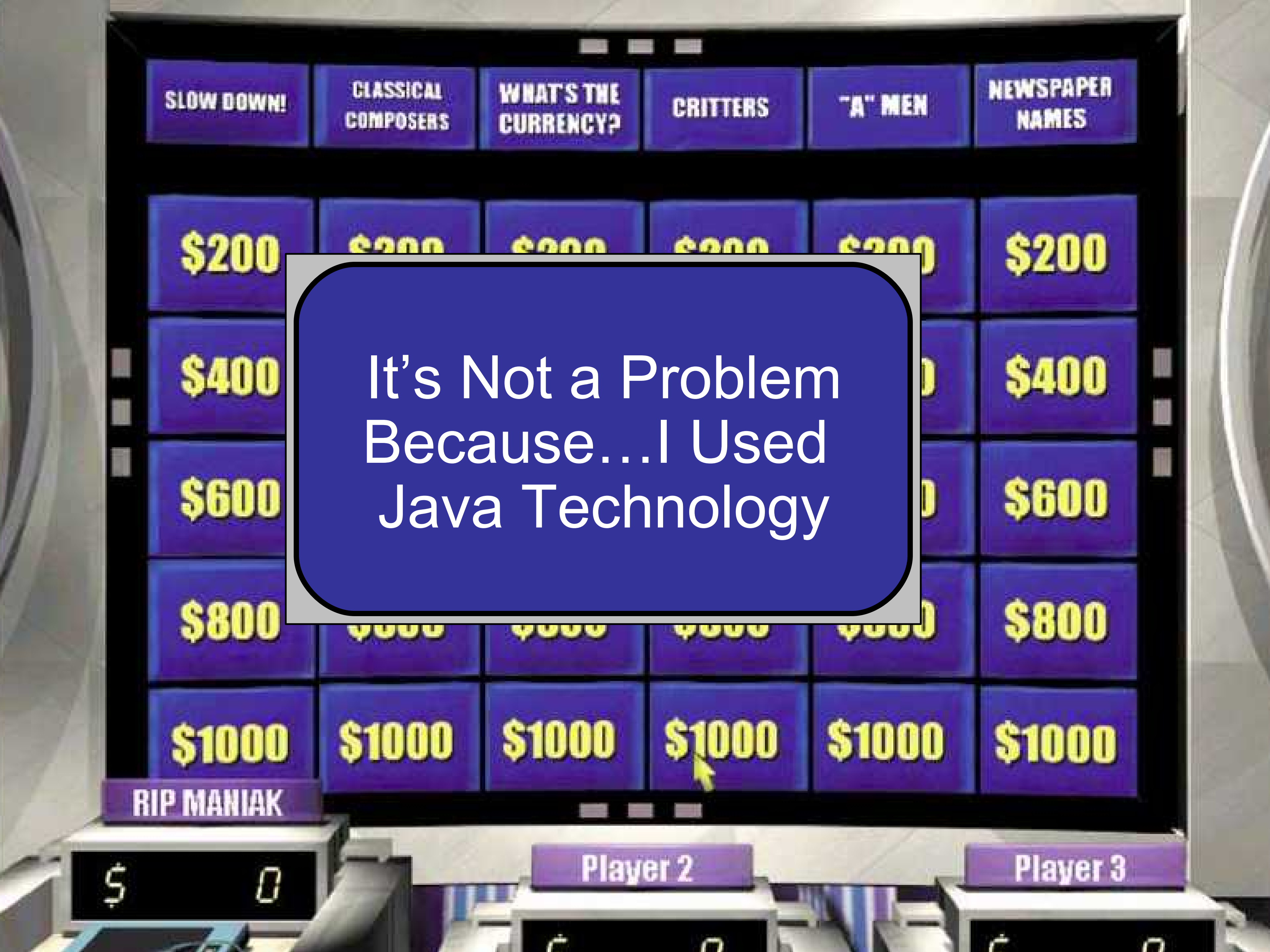
- “I’ll delete that before the release.”
- “I meant to delete that before the release.”

Encapsulation

## Bonus: Struts Input Validation

```
 <form name="logonForm">
         <field property="username” depends="required">
             <arg0 key="logonForm.username"/>
         </field>
         <field property="password” depends="required,mask">
             <arg0 key="logonForm.password"/>
             <var>
                 <var-name>mask</var-name>
                 <var-value>^[0-9a-zA-Z]*$</var-value>
             </var>
         </field>
</form>
```

Configuration



It’s Not a Problem Because…I Used Java Technology

```
public class QuickSolver {
  private boolean minimize = false;
  private int timeLimit = 0;
  …
  private native int solveEq(String p);
}
```

## #6: Native Methods

- All the memory safety promises that Java makes?
  - Gone
- All of the type safety promises that Java makes?
  - Gone

Input validation and representation

```
public class SimpleServlet extends
  HttpServlet {
    public String acct;
    public Receipt rcpt;
    …
}
```

## #7: Concurrency/Synchronization

- Classic Concurrency errors
  - HttpServlet
  - Struts Action
  - Custom caches
- Also fun: deadlock

**Time and State**

## Bonus

```
xParam = req.getParameter(“x”);
x = Integer.parseInt(xParam);
if ((x > 0) && (x + HDRM < MAX)) {
  process(x); // x is legal
} else {
  giveError(x); // x is illegal
}
```

## Bonus: Integer Overflow

- Same problem as in C/C++, but without the appealing memory corruption target

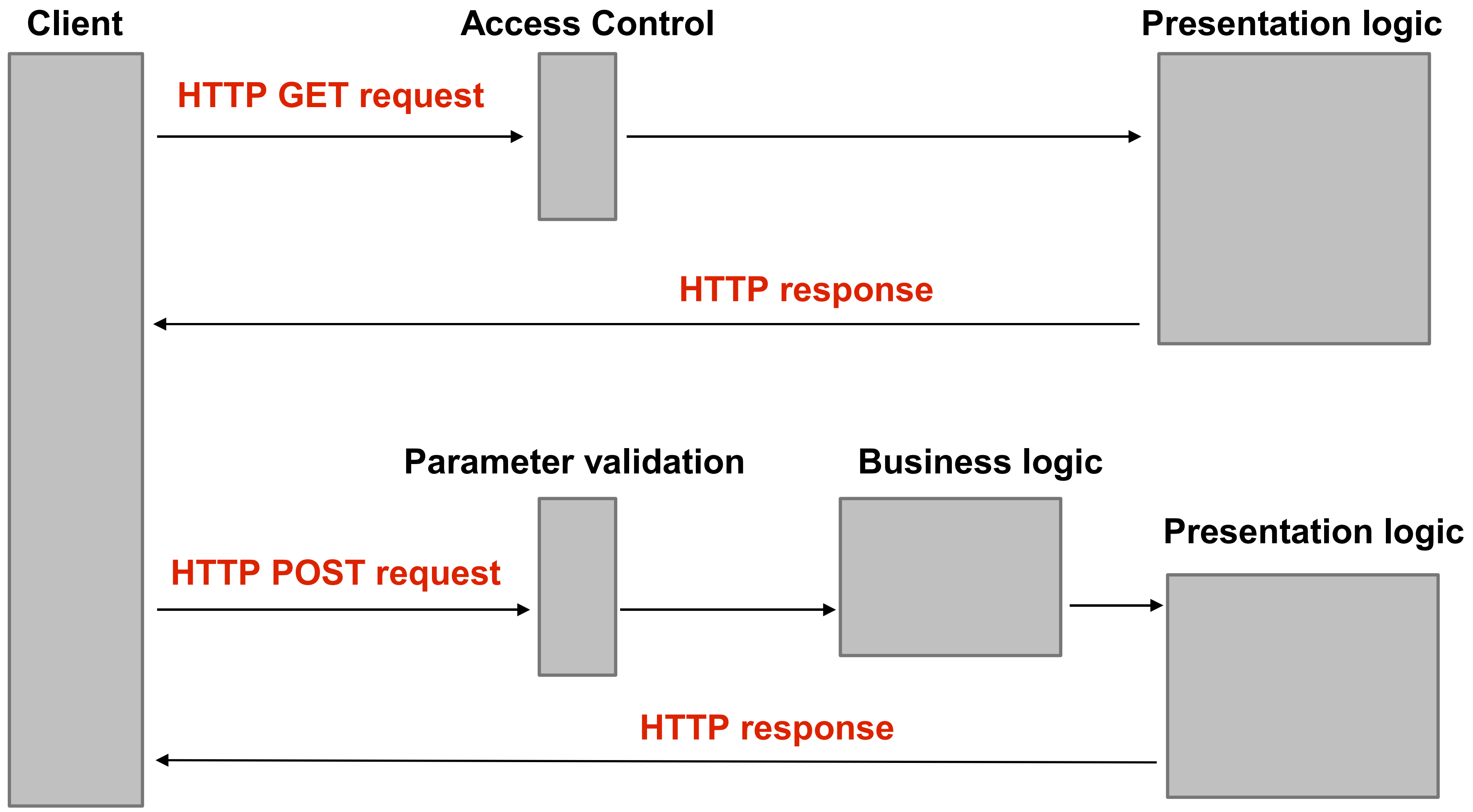
Input validation and representation

## Bonus: Bad Exception Handling

```
try {
  computeResult();
} catch (Throwable t) {
}
presentResult();
```

Errors

HTTP, How I Love Thee


Client
Access Control
Presentation logic
HTTP GET request
HTTP response
Parameter validation
Business logic
Presentation logic
HTTP POST request
HTTP response

## #8: Missing Access Control

- Missing back-end access control
- Access control errors are often the result of a distributed access control scheme. (Sprinkling a little access control in a lot of places is a recipe for mistakes.)

Security Features

```
private boolean doAuth(String usr,
                       String passwd) {
  if (!checkPasswd(usr, passwd)) {
    return false;
  }
  session = req.getSession();
  session.setAttribute(USER, usr);
  return true;
}
```

## #9: Bad Session Management

- Attackers have a leg up on session hijacking if
  - App transitions from unauthenticated to authenticated w/o issuing a new session ID
  - Failure to invalidate session at logout
  - Not enough randomness in session ID for period of use

Security Features

```
void printLocalizedGreeting() {
  lang = request.getHeader(“Accept-Language”);
  File f = new File(W_BASE + SEP + lang);
  if (!f.exists()) {
    throw new LangNotFoundError();
  } else {
    addToResponse(f);
  }
}
```

## #10 Cookies and Other Headers

- Cookies and HTTP headers cannot be trusted
- Often overlooked by validation logic
- 100% overlooked by Struts Validator
- Hidden fields often abused in the same way

Input validation
and representation

## Bonus: Unadvertised Parameters

```
// No one will ever think to try this on
// their own!
debug = request.getParameter(“dbg”);
```

Encapsulation

You Can't Trust Anybody These Days

```
try {
    sqlStr = assembleSql(BASE_QUERY,
                 user, passwd);
} catch (SQLException e) {
    logger.log(WARNING,
                “auth db exception “, e);
}
```

## #11: Logging Sensitive Data

- Harvesting log files for
  - E-mail addresses
  - Authentication data
  - Financial information
- Corp. privacy policy
- Regulatory compliance

**Encapsulation**

```
initCmd = System.getProperty(“init_cmd”);
runtime.exec(initCmd);
```

## #12: Trusting the Configuration

- Datacenter managers don’t give unfettered, unmonitored, unaudited control to system administrators, why should you?

Configuration

## Fundamental Tenants

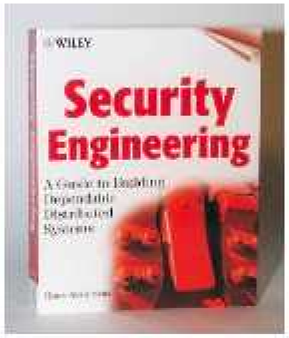
- Software security
  - Cannot be addressed as a series of bugs
  - Requires changing the way software is developed
- The solution
  - Policy
  - Process
  - Tools

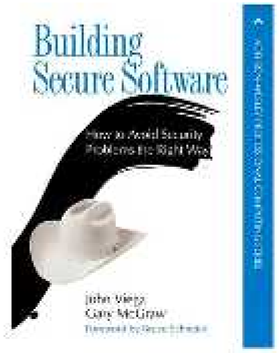
## Policy

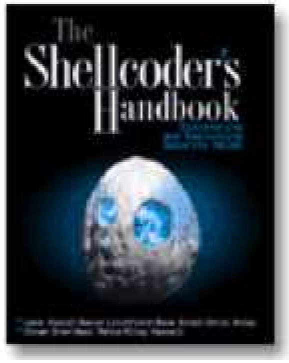
- Security is not optional
- The status quo is not acceptable
- Education is mandatory

## Education

- Study security
  - What do the attackers want?
  - How do attackers go about getting it?
  - How do software systems fail?

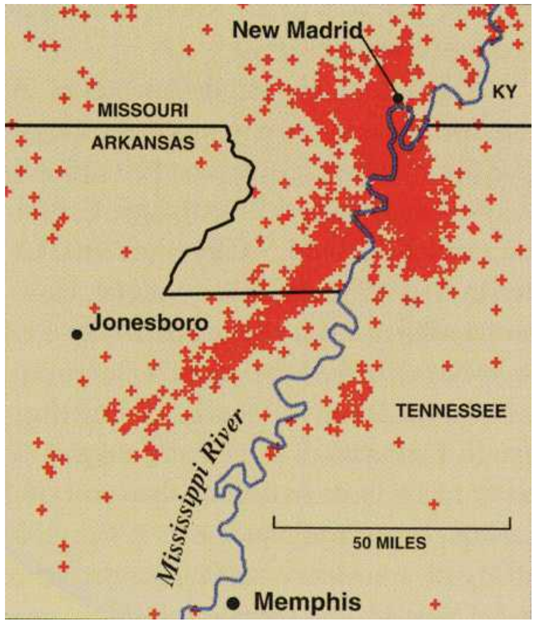






## Education

- Study your history!
  - What kind of security problems has your organization seen before?
- Collaborate with operations support

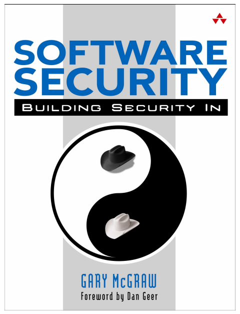


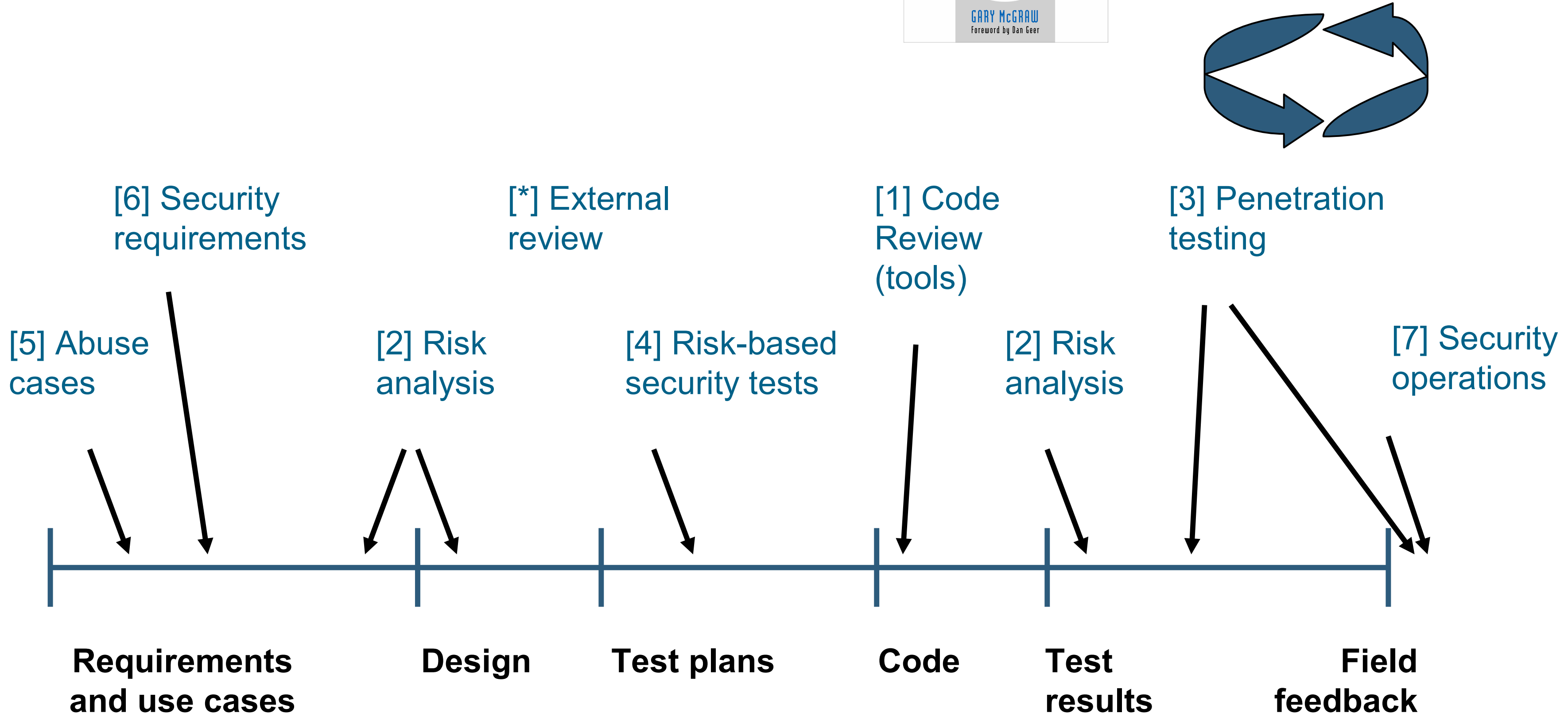
## Process

- Visualize a bad day
- Do code review
- Do security testing
- Sign off

# The Full Story

## Software Security Touchpoints





## Visualize a Bad Day

- Threat modeling
  - A threat is an entity
  - Who’s the bad guy?  What do they want?
- Risk assessment (with the design in mind)
  - Microsoft’s STRIDE, Sun’s ACSM/SAR
  - (Microsoft calls this “threat modeling”)
- Abuse cases

## Do Code Review

- Code review finds bugs
- Security errors are often easier to spot in code review than they are during testing
- Structured process for code review is essential
  - Assigned roles: author, coordinator, reader
  - Read the code ahead of time
  - 2 hour max meeting time
- You must know what to look for

## Do Security Testing

- A black box feel-good pen test is not enough
- A push-the-button webapp vulnerability scanner is not enough
- Base tests on abuse cases
- Hint: by the time you get to the testing phase, do you have a reason to believe you'll pass?

## Sign Off

- Software development is customer oriented: security needs a “customer”
- Create a gate

## Tool #1: Source Code Analyzer

- Use as part of code review (NOT in place of code review)
- Tools are good at hypothesizing bugs
- Roughly 50% of security defects are in play here

## Tool #2: Fuzz Tester

- Fuzzing is necessary but not sufficient. Look beyond port 80!
- Use existing regression framework
- Incorporate source code analysis findings
    - Attack surface (URLs, parameter names)

## Conclusions

- Software security cannot be addressed as a series of bugs
- Java technology is a world better than C, but there's plenty that still goes wrong
- Getting security right requires changing the way the software is developed
    - Policy
    - Process
    - Tools

# Twelve Java™ Technology Security Traps and How to Avoid Them

**Brian Chess, PhD**

Chief Scientist
Fortify Software
http://www.fortifysoftware.com

TS-1660