J2EE Security
JEE (gee?) – Security

A Presentation to Omaha's Cyber Security Forum
NEbraskaCERT.org/CSF/

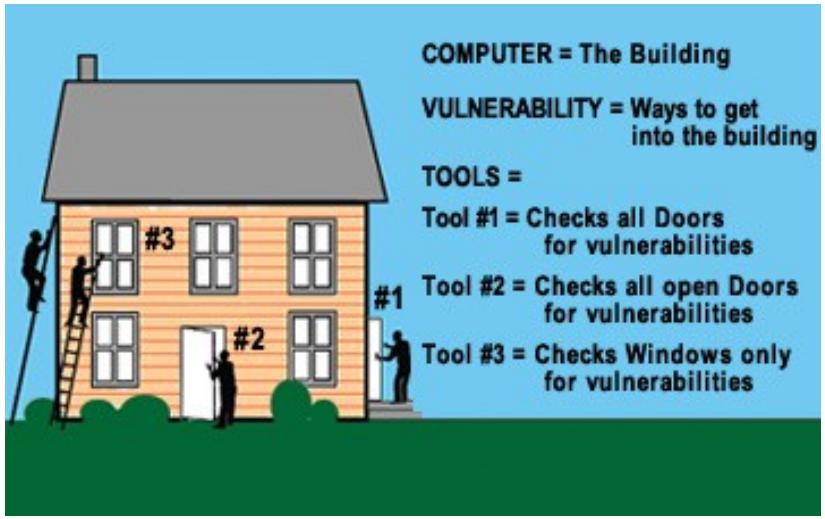
Slides Location:
www.MattPayne.org/talks

J2EE Security

- WHAT: Omaha's Cyber Security Forum
- TOPIC:J2EE Security
- BY: Matt Payne, CISSP Contact at www.MattPayne.org
- WHO: All Nebraska/Iowa Information Security Professionals
- WHEN: Wednesday - July 19, **11:30 am - 1:00 pm**
- WHERE: Johnny's Café 4702 South 27th Street, Omaha, NE
- WHY: To share information with like-minded professionals (and to share a FREE meal provided you RSVP!)
- HOW:YOU MUST RSVP to csfrsvp "at" NEbraskaCERT.org and provide name, company, phone and email address by Close Of Business Monday, 19 June.
- DESCRIPTION: Matt will be sharing some of his findings on the current J2EE security models. J2EE has become the standard infrastructure for many companies. With Several J2EE vendors in the current marketplace (Jonas/JBoss/Websphere/BEA and so) on the race for features has at times competed with the need for security.

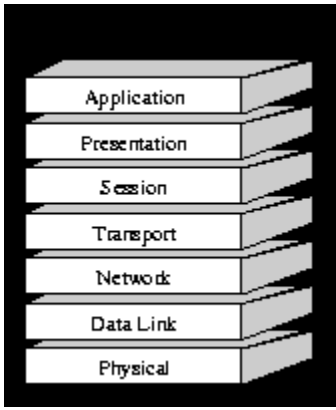
- Slides Location: www.MattPayne.org/talks
- Slides License: http://creativecommons.org/licenses/by-nc-sa/2.5/

Why?



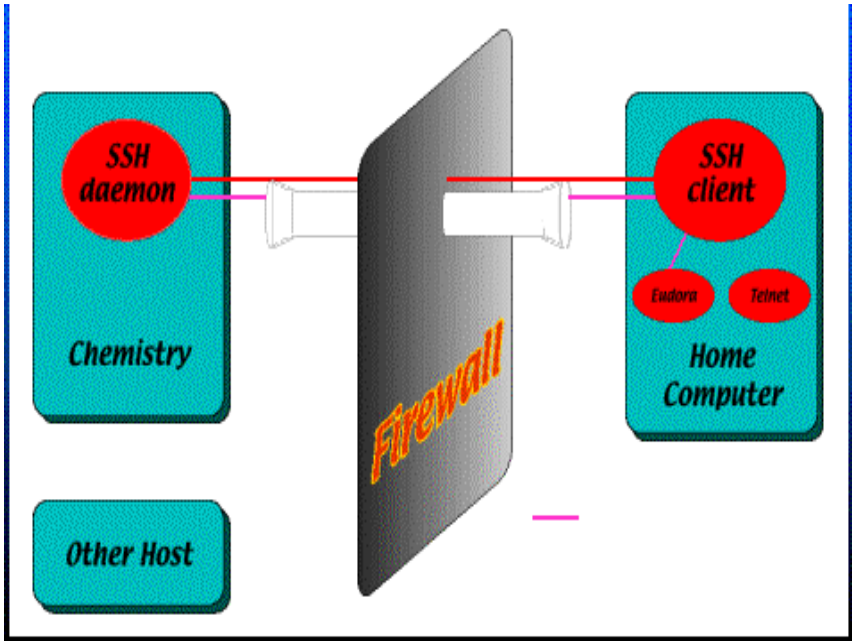
- It's important to secure all layers of the protocol stack!
- Too often the application layer is neglected.

- Remember! You can not test for the absence of flaw***s***
- e.g. using a chain saw to come in through the wall.



From COTSE.com
Check them out!

How do you know there's not an insider threat?



- There are several SSH libraries for Java
- Q: How do you know there aren't any tunnels?
- A: netstat –na
- Q: But what if the tunnels are not persistent?
- A: Verify the libraries (JARs) in the web application?

Competed with the need for security

- Innocent Code (**ISBN:** 0470857447) – users usually click OK.
  - http://innocentcode.thathost.com/
  - “webmitm works because users are used to clicking "OK" or "continue" buttons to make those warnings go away. How many users out there would understand the meaning of "The security certificate was issued by a company you have not chosen to trust" or "The name on the security certificate does not match the name of the site" anyway? HTTPS protects against MITM only if the users do not act irresponsibly.” – back cover of Innocent Code

Websites that email your old password.

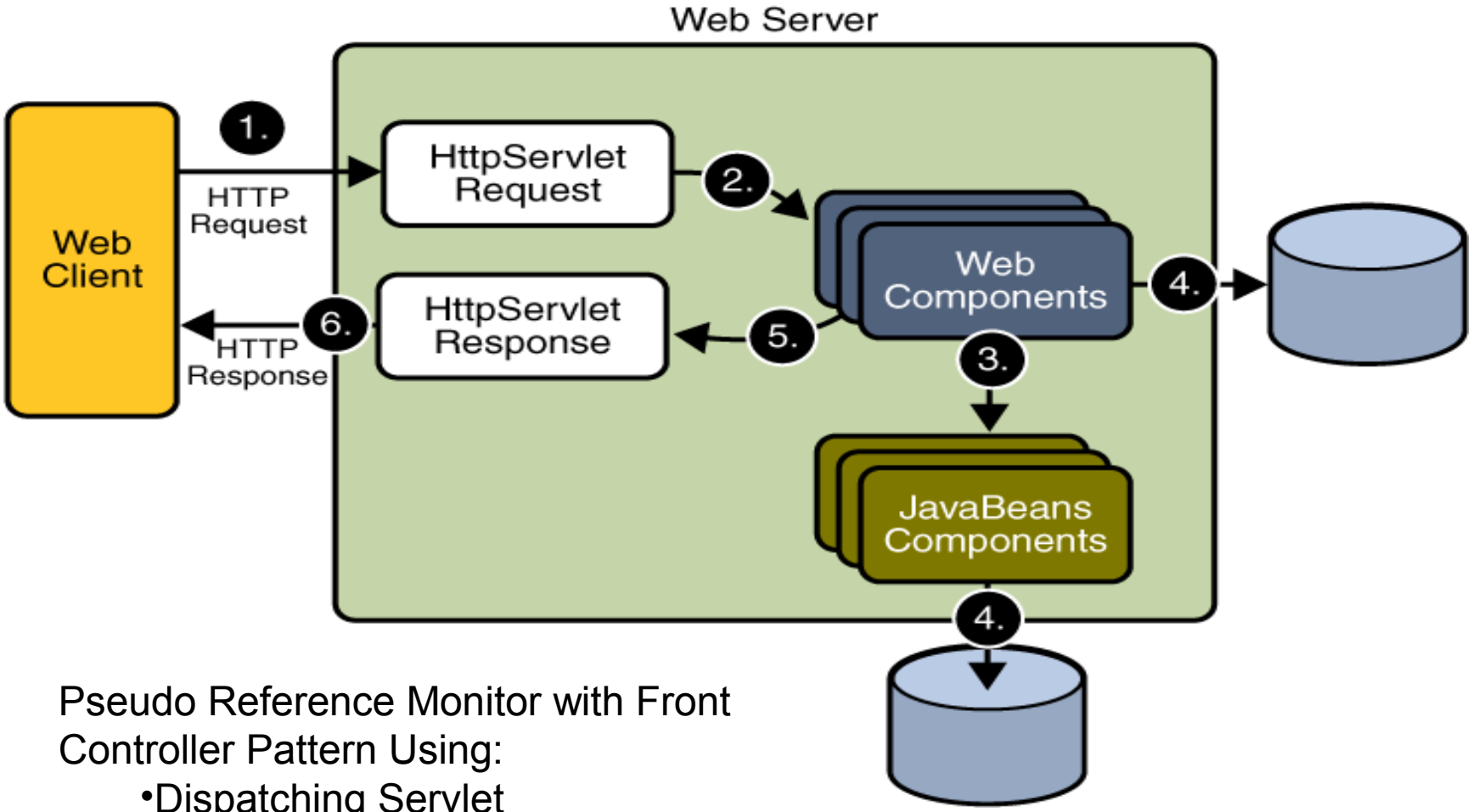
- This is bad no matter what toolset/language the site is built with!
- Example:
  - Welcome,

    You asked for your account information to be sent to you via the "Forgot Password" area of our site.

    Your login information:

    Email: payne@mattpayne.org
    Password: redacted

Overview: Standard JEE Web App

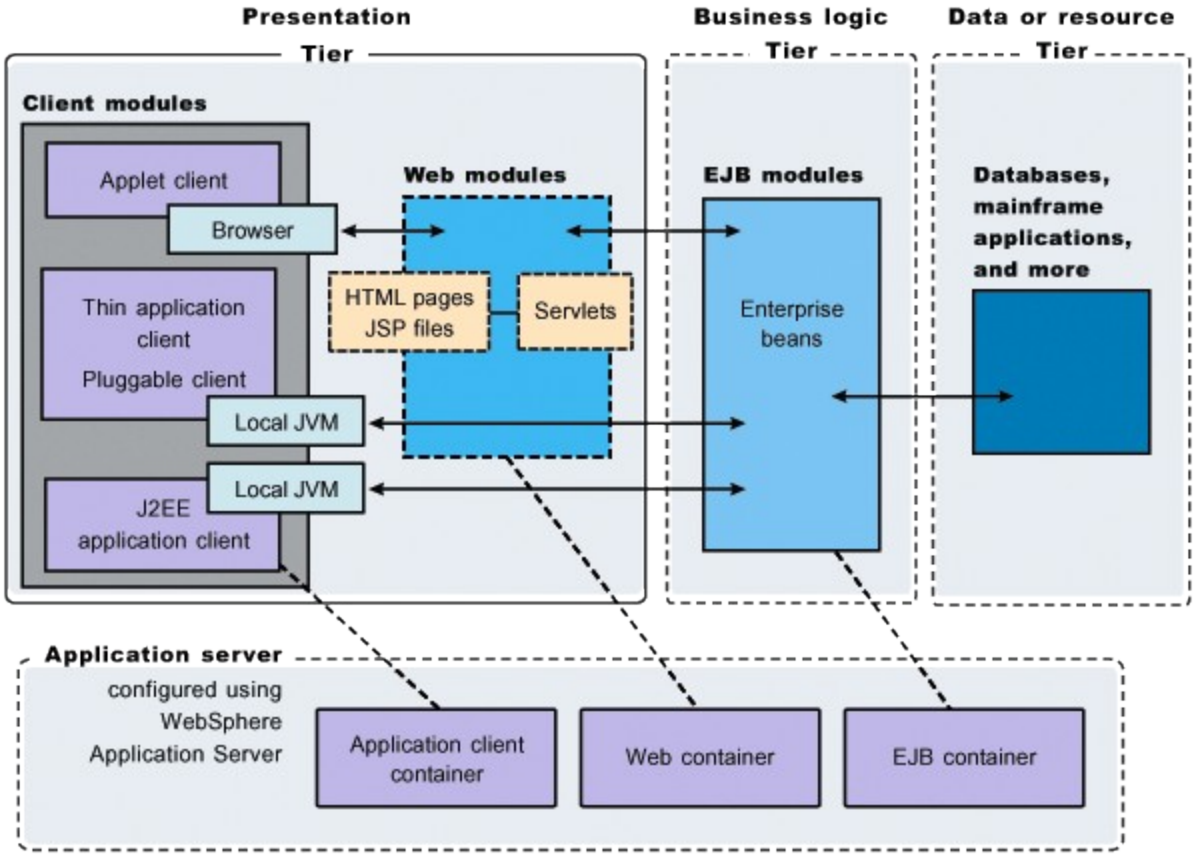


Pseudo Reference Monitor with Front Controller Pattern Using:

- Dispatching Servlet
- Servlet filters
- Application event listeners

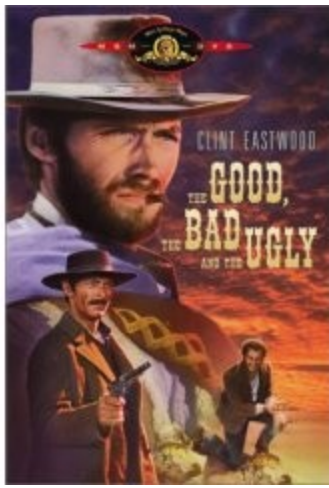
Source: http://java.sun.com/javaee/5/docs/tutorial/doc/Security-WebTier2.html

Example J2EE



From http://www-128.ibm.com/developerworks/websphere/zones/was/newto/

The good, the bad, and the ugly



- The Good
  - The standards support security and good interoperability
- The Bad
  - Most security is turned off by default!
- The Ugly
  - Getting from the default of no security to reasonable assurance….
  - Getting secure in J2EE can be ugly.
    - In the movie Ugly is Tuco, a bandit who's always only looking out for himself… not the good of the company
    - Why? Because J2EE is often UGLY.
      - Ugly XML
      - Ugly Complexity
      - Ugly Inertia – bad techniques that are still being practiced.

The Good: CIA

- Confidentiality
  - Basic Authentication
  - Basic Authentication with SSL!
  - Form based authentication
- Integrity
  - Signed server and **client side** SSL certificates
  - Signed library (java archive – JAR files)
  - From J2EE Blueprints talks about integrity of network messages:
    - “Message integrity is ensured by attaching a message signature to a message. “
- Availability
  - Java 2 Enterprise Edition (J2EE or just JEE) supports a wide variety of high availability solutions
    - e.g. clustering

The Good: AAA -- Authentication

- Authentication
  - JAAS, the Java Authentication and Authorization Service
  - End users of JEE applications may authenticate against LDAPs
    - Even Microsoft's Active Directory.
    - For a windows only Intranet the NTLM authentication technique is nice.
      - **tinyurl.com/zs5hy**
      - BUT! NTLM authentication is deprecated in the Windows world better to use:
    - **JAAS with Tagish SSPI JAAS provider:** JAAS with the Tagish SSPI-based login module is the way to go. The Tagish login module is based on the Windows SSPI API, which provides an authentication service for distributed environments using the best available protocol; i.e. it uses Kerberos when that is available and transparently falls back on NTLM when Kerbos is not available. In addition, SSPI returns the group membership information, which is necessary for servlet apps that use security roles and security constraints.
      - Source: **tinyurl.com/zs5hy** See also: http://free.tagish.net/jaas/

Client Certificate Authentication

- **GET /secure/showmyaccount.do**
  - **Mutual Authentication SSL**
    1. HTTP Server retrieved client' certificate from SSL
    2. HTTP Server transmits certificate information to JEE App Server
    3. JEE App Server map's client's certificate to a principal
    4. JEE App Server establishes identity of the principal
    5. If client is authorized, JEE App Server allows access to the resource
- **Stronger Authentication Mode**
- **Uses digital signature instead of password**

The Good: AAA -- Authorization

- JEE provides for programmaticly controlled authorization and more importantly declarative authorization!
- **Declarative Authorization**
- **Programmatic Authorization**

- Source: tinyurl.com/zkpy5

The Good: AAA -- Auditing Defined…

- java.sun.com writes: "*Auditing* is the practice of capturing a record of security-related events for the purpose of being able to hold users or systems accountable for their actions.
- A common misunderstanding of the value of auditing is evident when auditing is used solely to determine whether security mechanisms are serving to limit access to a system.
- When security is breached, it is usually much more important to know who has been allowed access than who has not.
- Only by knowing who has interacted with the system do we have a chance of determining who should be held accountable for a breach of security.
- Moreover, auditing can only be used to evaluate the effective security of a system when there is a clear understanding of what is audited and what is not."
    - Source: *J2EE BluePrints* **tinyurl.com/q3h4j**

The Good: AAA -- Who turns Auditing on?

- Deployers and System Administrators should not be the programmers
- "The Deployer is responsible for configuring the security mechanisms that will be applied by the enterprise containers. Each of the configured mechanisms may be thought of as a constraint that the containers will attempt to enforce on interactions between components. It should be possible for the Deployer or System Administrator to review the security constraints established for the platform, and to associate an audit behavior with each constraint so that the container will audit one of the following:
  - All evaluations where the constraint was satisfied
  - All evaluations where it was not satisfied
  - All evaluations independent of outcome
  - No evaluations"

  - Source: *J2EE BluePrints* **tinyurl.com/q3h4j**

The Good: AAA -- Auditing is not a free default

- "The J2EE programming model aims to shift the burden of auditing away from developers and integrators to those who are responsible for application deployment and management. Therefore, although not currently mandated by the J2EE specification, we recommend that J2EE containers provide auditing functionality that facilitates the evaluation of container-enforced security policy."
  - Source: *J2EE BluePrints* **tinyurl.com/q3h4j**
- **Semi-Automated Auditing with Aspect Oriented Programming shows promise**

The Good: Single Sign on solutions

- Java supports many single sign on solutions that are cross platform. E.g.
  - Central Authentication Service
    - Originally from Yale.edu
    - http://www.ja-sig.org/products/cas/
    - CAS offers:
      - An open and well-documented protocol
      - An open-source Java server component
      - A library of clients for Java, .Net, PHP, Perl, Apache, uPortal, and others
      - Integrates with uPortal, BlueSocket, TikiWiki, Mule, Liferay, Moodle and others

The Good: run-as

- Specifies the run-as identity to be used for the execution of the Web application
- A J2EE app can be configured to run code under the identity of the end user making the request.
  - This is a subject for another talk
    - Or better yet – a workshop
      - NEbraskaCERT.org and OJUG.org??

The Bad: None of this is turned on!

- The exiting the JVM “bug”
- Basic and form based authentication do not use SSL out of the box!
- Session snooping –
  - e.g. lambdaprobe.org
    - LambdaProbe is tomcat specific but the idea is generic.

The bad: XML Entity Reference Attack

- Passes malicious URIs as external entities
- Malicious Input data:

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE order [
3. <!ENTITY orderInfo SYSTEM “file:///etc/passwd"> ]>
4. <order>
5. <id>1</id>
6. <numberOfItems>1</numberOfItems>
7. <comments>&orderInfo;</comments>
8. </order>
```

- Implement a customized XML Entity Resolver
  - Source: **tinyurl.com/z2ynp**

The Ugly: A Basics' Eye Chart…

- Profiling
  - Carefully inspecting response from the server
    - HTML/JavaScript Code
  - Comments
  - Form fields, hidden fields
  - Links and URLS
  - HTTP Response such as Cookies
  - Exploit Improper Error Handling
    - Force application to crash
      - Pass invalid data
      - Access a non existent resource
      - Access unauthorized data
- With this information infer…
  - Application Architecture/Design details
  - Directory structure
  - Secure and insecure pages
  - Server Identification
- Code Injection
  - SQL Code Injection
  - Cross Site Scripting (XSS)
    - Cookie Theft
    - Session Hijacking
- MITM Attacks
- Source: **tinyurl.com/z2ynp**
- See also MIT Cookie Eaters papers

- Do Not Give Out Unnecessary Information
  - E.g.
    <%-- JSP style comment --%>
    NOT
    <!– HTML style comment -->
- Fail Safely
  - Handle all Checked and Unchecked exceptions
  - Handle HTTP errors and exceptions declaratively in web.xml

```
<error-page>
<error-code>HTTPError#</error-code>
<location>path/to/resource</location>
</error-page>
<error-page>
<exception-type>FQEN</exception-type>
<location>path/to/resource</location>
<error-page>
```

- Practice Input Validation
  - Semi Automate input Validation
  - Use Servlet/Filters to write general validation/filtering logic
  - Use frameworks such as Struts
  - Implement a customized XML Entity Resolver
- Practice Output Encoding
  - Escape HTML and JavaScript special characters to prevent XSS
  - Prepared statements or Hibernate.org to prevent SQL injection
- Use Cryptography to Secure Data
  - Implementation!
    - BouncyCastle.org
  - Key Management!

More Ugly Basics

- Poor Session Management
  - Replay Attack
  - Cookie Poisoning
- Broken Authentication
  - Man in the middle
  - Replay Attack

- Effective Session Management
  - Centralize the session creation process
  - Disable session creation from JSP
  - Invalidate Session when the user logs out
  - Reset Session Cookies when the users logs out
  - Define Session timeout in web.xml
  - Use techniques in the MIT Cookie Eater's Paper
  - Do not rely only on JSESSION cookie
  - Wrap cookie in message authentication code
    - Cookie holds random # among other things.
    - No sensitive info in cookie
  - Place timestamps in cookie
- Use Proper Authentication Model
  - HTTP BASIC Authentication
    - Password sent is BASE64Encoded; Not Encrypted
    - Use SSL to gain confidentiality and data integrity
    - No logout mechanism
      - Rig one up with cookies
      - session.invalidate()
      - Plus your own cookie
  - Form Based Authentication
    - Username/Password sent via POST; Use SSL

The Ugly: SSL

- The Ugly:
  - Declare in web.xml:
    1. <user-data-constraint>
    2. <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    3. </user-data-constraint>
  - Programmatic Check
    - “https”.equals(request.getScheme())
- The good:
  - The deployer may turn on SSL without any code changes!

General Availability Tips

1. Don't let your sessions grow too large…
2. For unauthenticated users, avoid any unnecessary access to expensive resources such as databases
3. Monitor and control long running queries – Logging & JMX
4. Consider handling only one request per user by synchronizing on the session object
5. Clean up resources, such as database connection, socket connections to other server
   - E.g. Blaine's Automated Standards Enforcement Project
     - Scan code for proper use of finally clauses
     - http://www.blainebuxton.com/ase/
   - Findbugs.sf.net is also worth a look.
6. Check error handling scheme to ensure that an error does not affect the overall operation of the Application
   - Write UML Use Cases and Abuse Cases
     - Gary McGraw Software Security: Building Security In
     - PodCast: http://www.itconversations.com/shows/detail966.html
7. "Fail Safely"

Ugly: Broken Access Control

- Enforce Proper Authorization
  - Front Controller Pattern
    - Define Entry points to Application Functionalities
    - Authorization Enforcement at the entry points
  - Authorization Models
    - Declarative
    - Programmatic
    - Combination of Declarative and Programmatic
      - (Preferred)

- Declarative

```
1. <security-constraint>
2. <web-resource-collection>
3. …
4. <url-pattern>payment/*</url-pattern>
5. </web-resource-collection>
6. <auth-constraint>
7. <role-name>manager</role-name>
8. </auth-constraint>
9. </security-constraint>
```

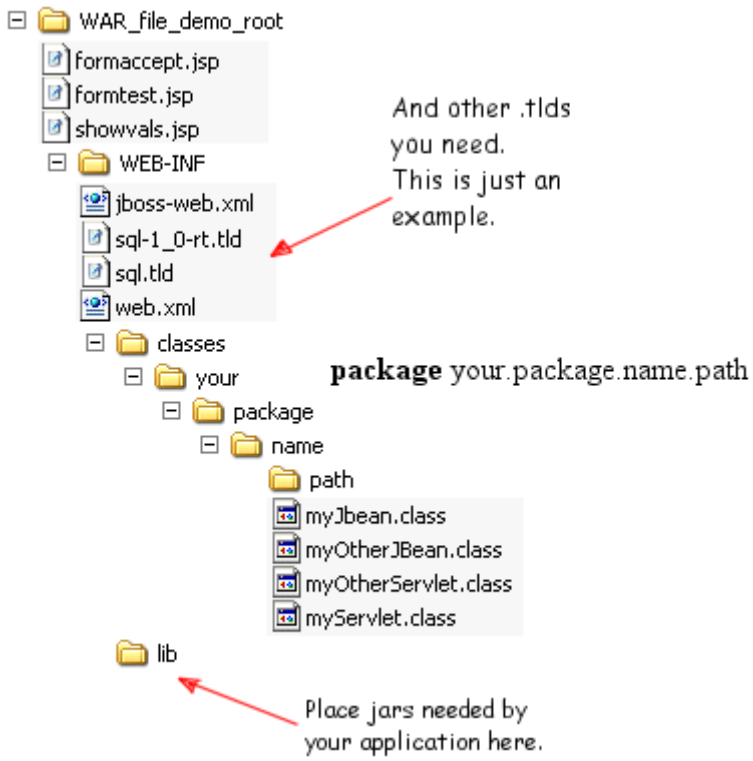
- **Programmatic**

```
1. If (!request.isUserInRole("manager")) {
2.     throw new UnAuthorizedException();
3. }
```

Advice: Hide things under WEB-INF

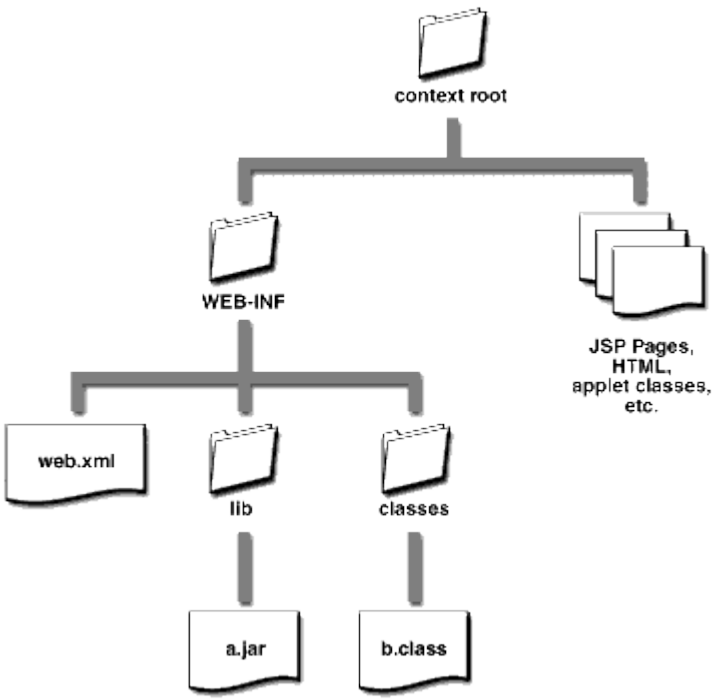
- Keep resources as non-url accessible
  - Keep configuration files under WEB-INF folder
  - Keep JSPs and JSP fragments under WEB-INF
- Disable Cache when generating sensitive data
  - Part of the front controller
  - **// HTTP 1.0 Browser**
  - **response.addHeader("Pragma", "no-cache") ;**
  - **// HTTP 1.1 Browser**
  - **response.addHeader("Cache-Control", "no-cache");**
  - **response.addHeader("Cache-Control", "no-store");**

WAR -- Web Application Archives



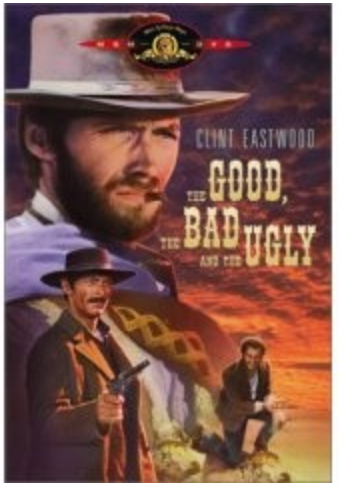
- WARs are a big part of the portability between different Servlet/JSP containers
- A WAR is a JAR with extra file structure
  - File Structure show on left
- A JAR is a ZIP with extra file structure (a META-INF folder)
  - **Wikipedia entry: tinyurl.com/k3tnq**

Example of Inertia



- From Designing Enterprise Applications with the J2EE Platform, Second Edition
  - © 2002
  - Online at **tinyurl.com/lqoka**
- Today this layout is considered BAD!
  - All* content should be under WEB-INF so a front controller can be used to control access.
    - **tinyurl.com/5vos6**

What can be done to reduce the UGLY?



- Ugly problem:
  - Lots of repetitive things to check!
  - People are best spent on less repetitive things – asking interrogative questions about vulnerabilities that are not yet well know
- Good Solution:
  - Semi-automate checking the **baseline** so we do not get had by what we already know is bad.

Why Semi-Automate?

- Semi-Automate -- Never Automate
  - Always have a doubting Thomas in the loop
  - Trust but verify – Ronald Regan **tinyurl.com/mfsdn**
- Hackers exploit the wide gap between *overworked/underpaid application developers* and overworked/underpaid administrators
- Confirm that the proper steps have been taken and are **still in place**
  - From the start – plan for this. Build security in from day one!
  - Define security policies
  - Run with least privilege – process and file system
  - Do not use self signed, or default SSL certificates
  - Watch out for SSL certificate expirations!
  - Logging and Alerts
    - Have scheduled everything's ok heartbeat alerts
    - As well as the Oh No Mr. Bill alerts!

How to Semi-Automate?

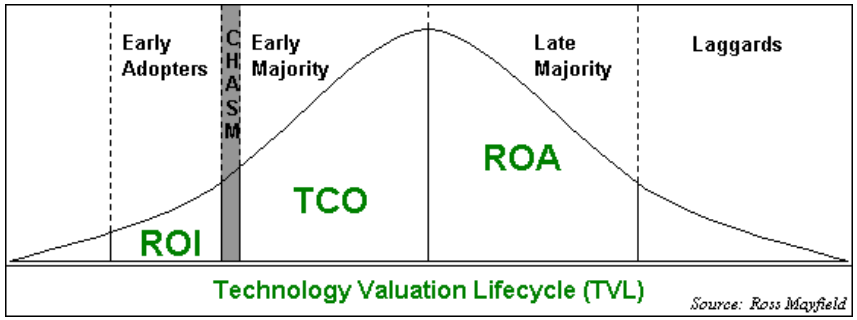
- Whenever possible Semi-Automation should be vendor neutral.
- After all Tomcat, Websphere, BEA, JBoss, Jetty, etc all follow J2EE specifications…
- Good thing Java is write once run anywhere (WORA)!
- Semi-Automated with tools that will check more than just the J2EE container…

OVAL: Open Vulnerability and Assessment Language

- A declarative language for describing vulnerabilities and semi-automating assessments
- The Language defn: “A collection of XML schema for representing system information, expressing specific machine states, and reporting the results of an assessment”
- The Repository defn: “The central meeting place for the OVAL Community to discuss, analyze, store, and disseminate OVAL Definitions”
  - Multiple repositories are now allowed!
  - Mitre hosts the oldest
    - ThreatGuard.com contributes OVAL definitions for Most Recent Microsoft Security Bulletins
  - Redhat.com now has a repository for their known vulnerabilities
    - **tinyurl.com/lmlar**
- OVAL.Mitre.org

OVAL 5 Just Came out

•The Early Adopters phase of OVAL is over.



Source: tinyurl.com/qg9wz

- OVAL Developer days
  - NSA and NIST moving to XCCDF and OVAL instead of English prose guidance
  - Q: XCCDF? A: eXtensible Configuration Checklist Description Format
    - wrap and customise multiple OVAL tests
- Early OVAL Adopters:
  - Citadel.com
    - Required by US GOVT
  - ThreatGuard.com
    - Java implementation of OVAL interpreter
      - OEM kit RSN!
- Mitre has free (BSD license) C++ based interpreter…

OVAL checks config files

- Even configuration files in “ugly” XML!

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Protected Area</web-resource-name>
        <url-pattern>/Edit.jsp</url-pattern>
        <http-method>DELETE</http-method>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>PUT</http-method>
    </web-resource-collection>

    <auth-constraint>
        <role-name>admin</role-name>
        <role-name>user</role-name>
    </auth-constraint>
</security-constraint>

<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>JSPWiki Editor</realm-name>
</login-config>
```

- How? XPATH – A w3c.org standard! See tutorial w3schools.com/xpath
- Think directory paths plus extra syntax! E.g.:
  - “//auth-constraint/role-name” returns a list of all authorized roles. // is a wildcard tree search.

OVAL File Integrity Checks

- OVAL can check file integrity using cryptographic hash functions.
  - Md5 today
  - NUCIA is submitting a patch to add SHA-1 to the language.
    - Can also just pre-process SHA-1 hashes
      - Verify SHA-1 then produce MD5 to check with OVAL…

OVAL File System Checks

- These can be used to make sure that the WAR file's directory tree is laid out according to best practices.
  - Are the config files under WEB-INF?
  - Are the JSP files under WEB-INF?
  - Are opportunities for application profiling minimized?

OVAL in four Figures…

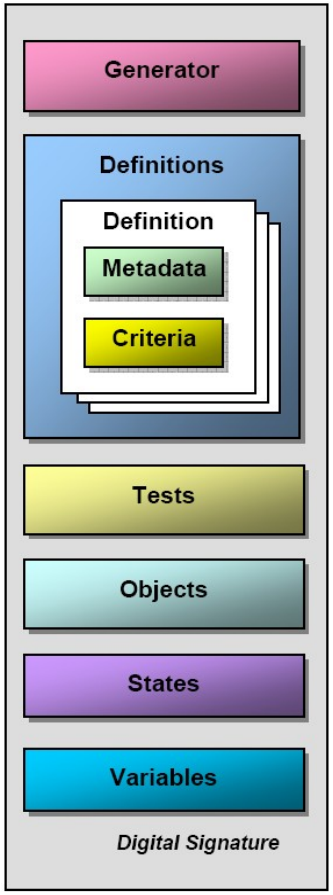


Figure 1 : OVAL Definition Document

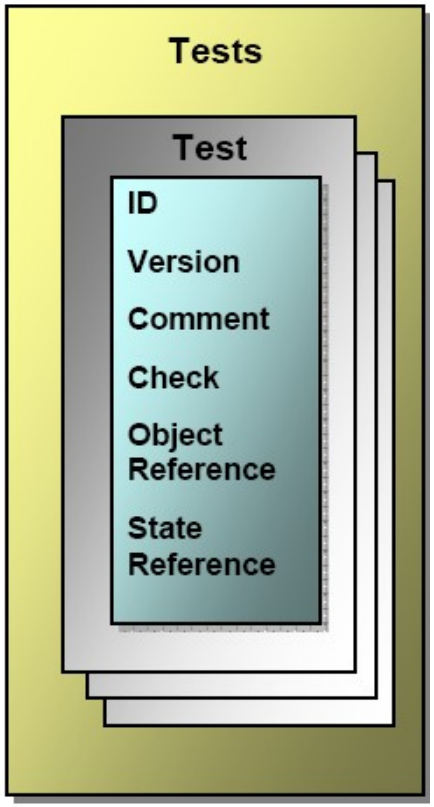


Figure 2 : OVAL Definition Test Structure

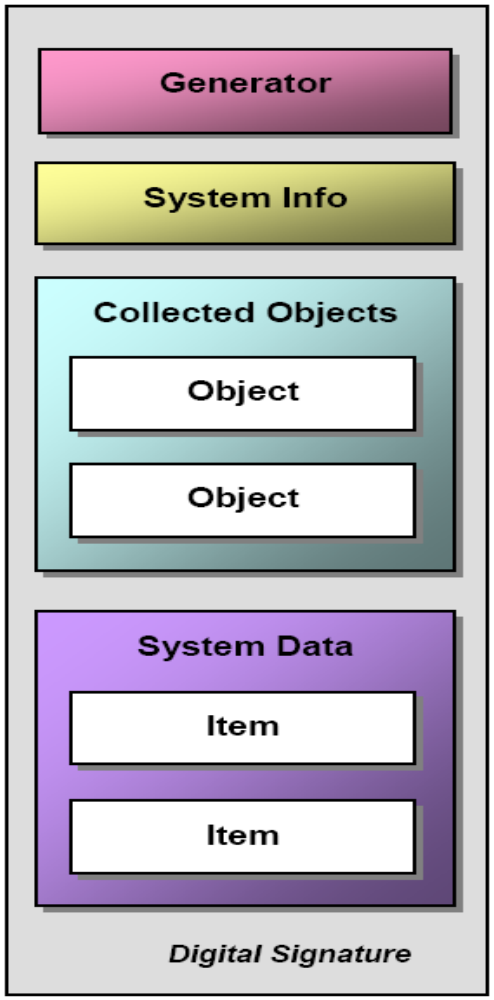


Figure 3 : OVAL System Characteristics Document

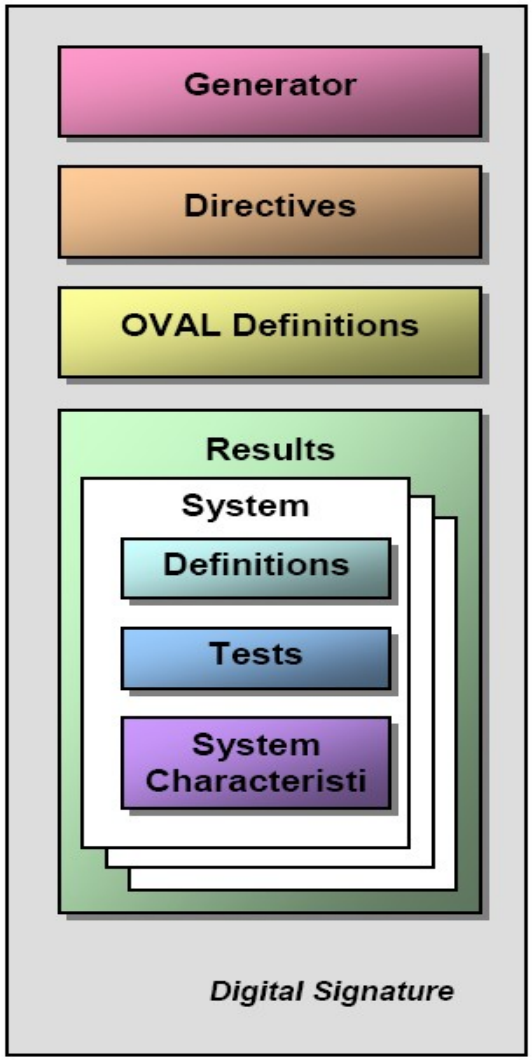


Figure 4 : OVAL Results Document

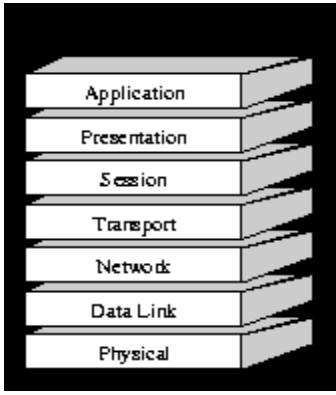
Src: OVAL Lang. Design. Doc.

And now a word from “our” sponsors…

- Come to
  - “***The bright future of the Extensible Configuration Checklist Description Format (XCCDF) and its friends***”
    - AND
  - “An OVAL & XCCDF tutorial”
    - Bring your laptop!
- Where? The NEbraskaCERT Conference: August 8-10, 2006

  http://www.certconf.org/

Don’t re-invent the Wheel!



- Stand on the **Shoulders of Giants**
  - http://www.cyber.com.au/users/conz/shoulders.html
- Remember -- It’s important to secure all layers of the protocol stack!
  - OVAL can help do this!
- Use existing tools
- Use OVAL to create DECLARATIVE definitions to semi-automate checking the results existing assessment tools
  - Java Application Verification Kit
  - Eclipse IDE project meta files
  - Code Analysis tools
  - Black box testing

AVK

- “
  Java Application Verification Kit (AVK) for t

  The AVK is available to help you test your application for correct use of J2EE APIs and to maintain portability across J2EE-compatible application servers.”
  - http://java.sun.com/j2ee/verified/
  - Checks hundreds of assertions!
    - Both security motivated and portability (availability) motivated

Eclipse Web Tools Platform (WTP)

- Version 1.5 released June 30, 2006
  - http://www.eclipse.org/webtools/
  - “The Eclipse Web Tools Platform (WTP) project extends the Eclipse platform with tools for developing J2EE Web applications. The WTP project includes the following tools: source editors for HTML, Javascript, CSS, JSP, SQL, XML, DTD, XSD, and WSDL; graphical editors for XSD and WSDL; J2EE project natures, builders, and models and a J2EE navigator; a Web service wizard and explorer, and WS-I Test Tools; and database access and query tools and models.”
- Some developers may not have heard the news – tell them and it will make their day…
- Good tools make for better – more secure – software

Code Analysis

- Using Findbugs, PMD, Metrics, NCSS, jLint to find flaws and bugs
  - FindBugs.sf.net (and others) have XML output options…
- Blaine Buxton's project & talk on
  - Automated Standards Enforcement Project
    - Scan code for proper use of finally clauses
    - http://www.blainebuxton.com/ase/

Black box testing..

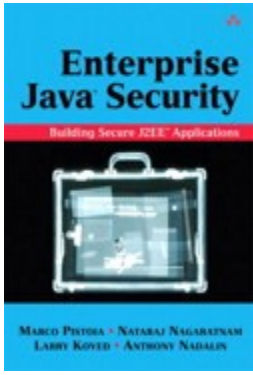
- Generic
  - Using WebScarab to find vulnerabilities in web applications – including J2EE applications
  - Also ParosProxy.org
- Custom
  - Unit tests written with httpunit.sf.net, webtest.canoo.com, and other tools

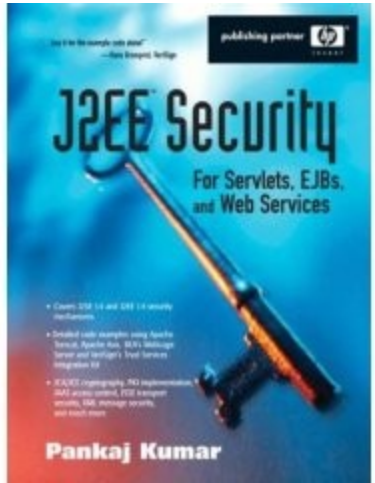
What to do next?

1. Remember the good, the bad, and the ugly!
   - The Good: J2EE enables secure systems
   - The Bad: J2EE servers are frequently not secure by default!
   - The Ugly: It's a lot of work to secure any server side system – regardless of the technology involved.
     - **Remember – you can not test for the absence of flaws!**
2. Contact Matt Payne at Payne@MattPayne.org if your interest in find out more about NUCIA's efforts with OVAL and XCCDF.
3. Don't throw the baby out with the bath water!
4. Review some of the sources for this talk…

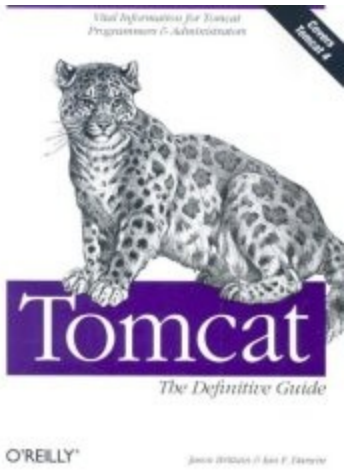
Books….

- *Mastering EJB 3.0*
  - "free" download at tinyurl.com/ogker
  - Chapter 11 discusses security
- Enterprise Java™ Security: Building Secure J2EE™ Applications
  - by Marco Pistoia; ISBN: 0321118898; Copyright 2004
  - Blah blah blah
  - Blah blah
- J2EE Security for Servlets, EJBs, and Web Services
  - by Pankaj Kumar; ISBN: 0131402641;
  - Copyright 2004
  - Blah blah blah
  - Blah blah blah

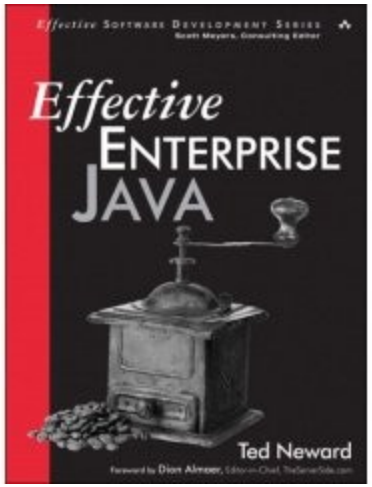




Books



- **Tomcat: The Definitive Guide**
  - by Jason Brittain, Ian F. Darwin;**ISBN:** 0596003188; Copyright 2003
    - Specific to tomcat4. Apache is now at tomcat 5.5.
      - But much of this info still applies.
    - The free sample chapter on security is GREAT!
      - http://www.oreilly.com/catalog/tomcat/chapter/index.html
      - Aka **http://tinyurl.com/o8eqt**
- **Effective Enterprise Java**
  - by Ted Neward; **ISBN:** 0321130006; Copyright 200
  - Blah blah blah
  - Only a few tips are directly related to security:



Books

- **Core Security Patterns: Best Practices and Strategies for J2EE(TM), Web Services, and Identity Management**
  - by Christopher Steel, Ramesh Nagappan, Ray Lai; **ISBN:** 0131463071; Copyright 2006
- **Inside Java 2 Platform Security: Architecture, API Design, and Implementation (2nd Edition)**
  - by Li Gong, Gary Ellison, Mary Dageforde ; **ISBN:** 0201787911; Copyright 2003
  - Covers the fundamentals of Java Security

Books

- **Head First Servlets and JSP: Passing the Sun Certified Web Component Developer Exam (SCWCD)**
  - by Bryan Basham, Kathy Sierra, Bert Bates; **ISBN:** 0596005407; Copyright 2004
  - Great book for the basics of server side java
  - Does a nice job of explaining the default security model

Some Websites

1. tinyurl.com/z2ynp
   - "You Are Hacked: Ten Secrets to Securing Your J2EE Web Applications by Shyamsunder, Neethiraj, and Nylund
2. http://java.sun.com/javaee/5/docs/tutorial/doc/
   - The Java EE 5 Tutorial
     - Chapters 28 thru 31
3. tinyurl.com/mv628
   - Struts best practices
4. http://www.onjava.com/lpt/a/2825
   - Servlet Best Practices
5. http://java.sun.com/blueprints/enterprise/
   - Enterprise BluePrints
6. http://www.oreillynet.com/pub/au/1370
   - Denis Piliptchouk 's five part Java vs. .NET Security series
7. tinyurl.com/h2dv6 – Brian Chess's Talk on
   - Twelve Java Technology Security Traps and How to Avoid Them
   - Java One 2006: http://developers.sun.com/learning/javaoneonline/
     - Service Component Architecture: Approach to Security, Transactions, and Policy

Some More Websites

1. **tinyurl.com/zs5hy**
   - How to Authenticate a Servlet App with Windows Passwords

Spring and the **Acegi Security System**

- Acegi is a another (future) talk entirely…
  - http://acegisecurity.org/
- **"Enterprise-wide single sign on:** Using JA-SIG's open source Central Authentication Service"
  - http://www.ja-sig.org/products/cas/
  - CAS offers:
    - An open and well-documented protocol
    - An open-source Java server component
    - A library of clients for Java, .Net, PHP, Perl, Apache, uPortal, and others
    - Integrates with uPortal, BlueSocket, TikiWiki, Mule, Liferay, Moodle and others
- Open source solutions frequently do not offer indemnification for there licenses. This is a huge problem for many businesses – especially when companies like IBM will indemnify you.

Thank you!

- Slides are at

  www.MattPayne.org/talks

- Comments? Contact me at

  www.MattPayne.org/contact

See you at

The NEbraskaCERT Conference:

August 8-10, 2006