DON'T DRINK SOUR MILK

and other avoidable risks in the new world of application security.

There has never
been a more interesting,
important or challenging
time to be a software
security professional than
RIGHT NOW.

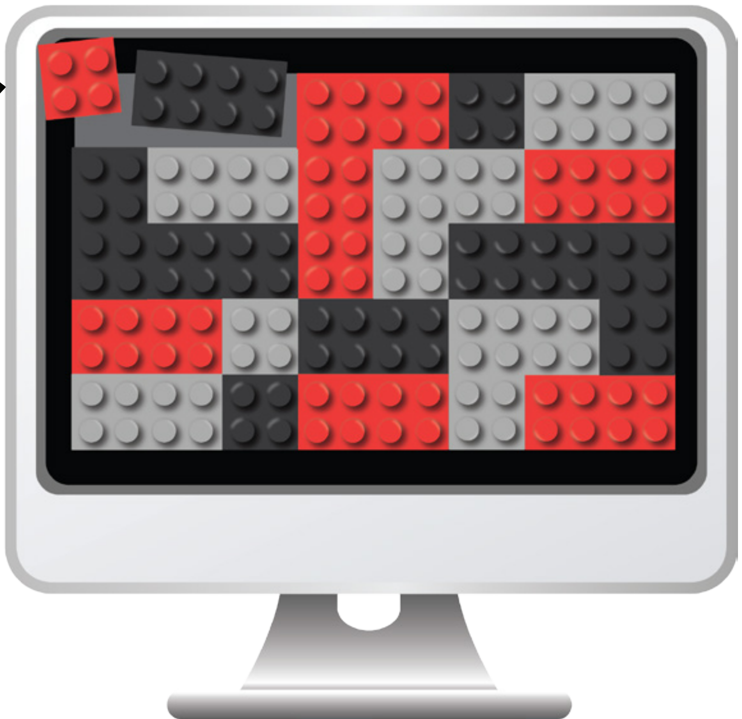
~~Your job description has been fundamentally re-written.~~

Applications are the new vector of attack.

Development is going faster than security can keep up.

Most source code has been replaced by open source components.

Did you know that 90% of a typical application is comprised of open source components which are assembled together like LEGO® building blocks?



Let’s start with a question: Is application security BROKEN?

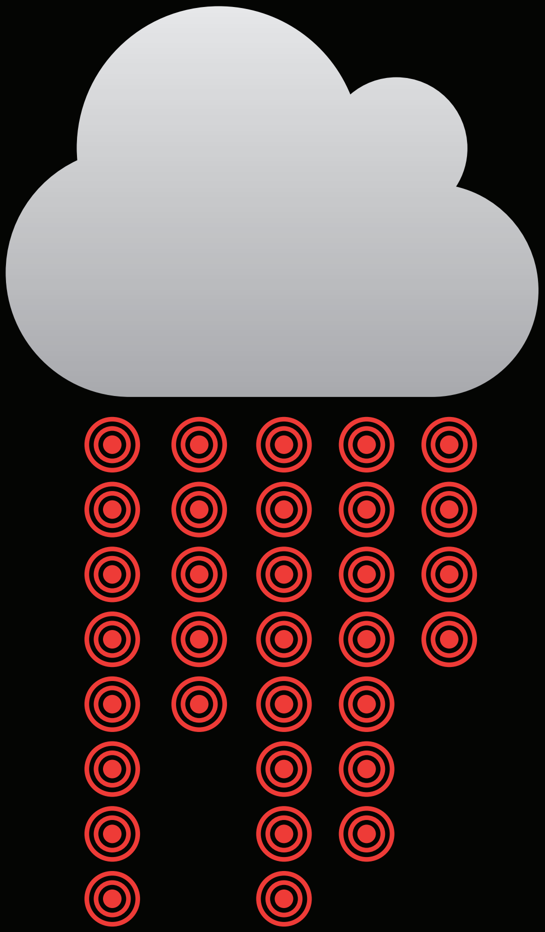
Security is bolted-on, not built-in.

Releases are monthly, weekly, or even daily. Security can’t keep up.

Software is assembled with components, yet we can’t really see what we’re using.

We build known vulnerabilities into our software, then spend even more time and resources to get them back out.

THE EVIDENCE



58.1 million components with

KNOWN VULNERABILITIES WERE DOWNLOADED

from the (Maven) Central Repository last year.

You see... components age more like milk than fine wine.

Over time, they expire due to security or quality issues...but are still available for consumption.

Let’s get more specific.

Even after security alerts were issued and fixes provided...

there were
150,508
downloads of
BOUNCY CASTLE

with an exploitability score of 10.

there were
70,665
downloads of
APACHE STRUTS 2

with an exploitability score of 10.

there were
2,167,625
downloads of
HTTP CLIENT

with an exploitability score of 8.6.

Hmmm...that’s a lot of sour components flowing into your applications. And fresher versions have been available *for years*!

Source: Sonatype analysis of downloads from (Maven) Central Repository and NIST CVE database

THE BOTTOM LINE.

We are knowingly and consciously prolonging the life of component versions with KNOWN vulnerabilities.

And they are lurking in your applications.

Can we all agree? This is just not working!

We scan source code.
We manually enforce whitelists and blacklists.
We (think we) have golden repositories.

All tickets on the things-we-think-we-should-do-to-be-competent train.

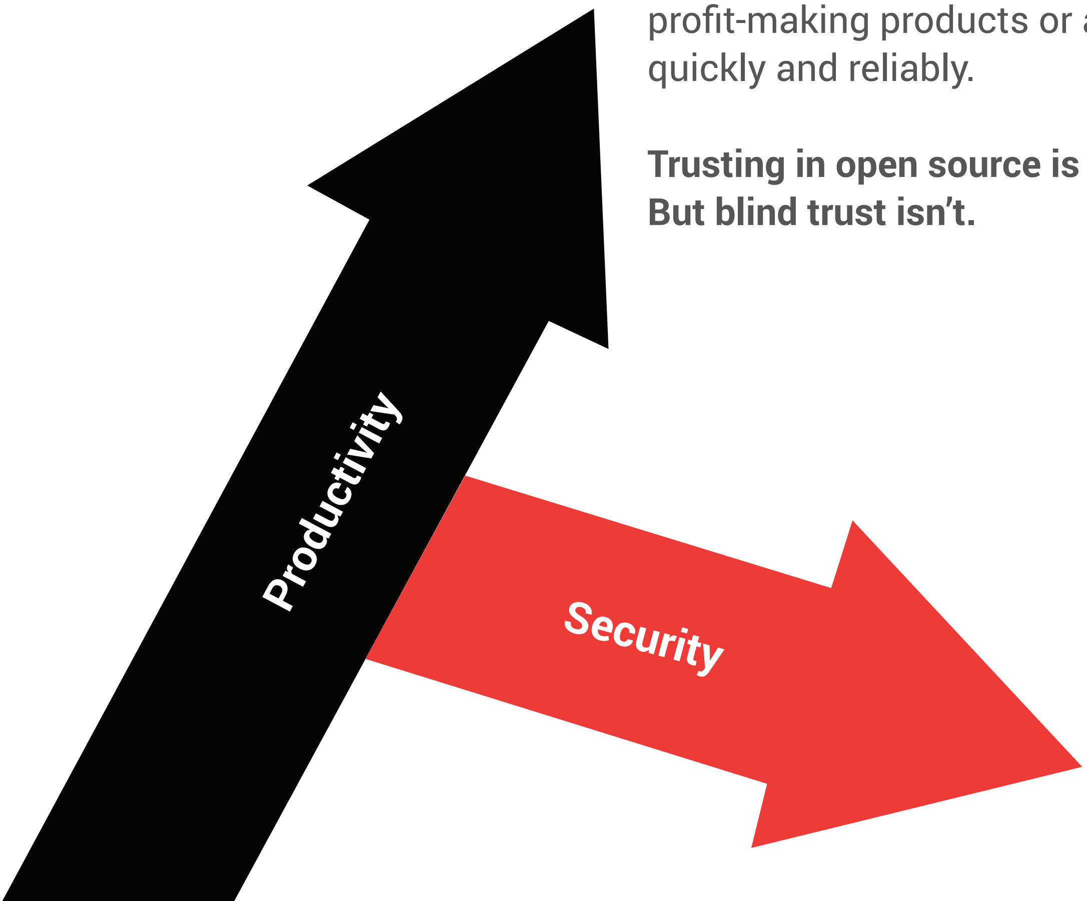
But your developers find work-arounds…
Cyber attacks are on the rise…
And software is still not secure…

The Facts: This is NOT an open source problem.

This is productivity exceeding security.

Open Source Software (OSS) is essential in our world today. Without it, we couldn't build our innovative, profit-making products or awesome new services quickly and reliably.

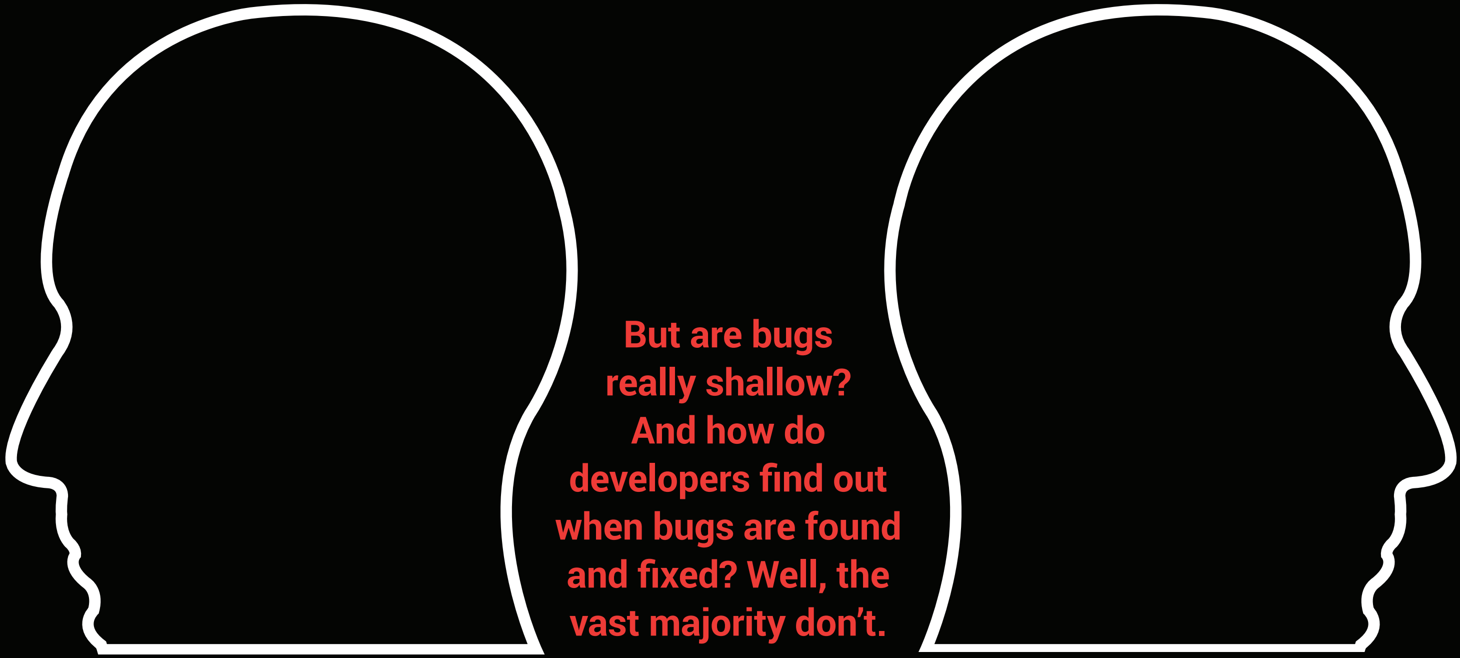
Trusting in open source is fine.
But blind trust isn't.



“What we’ve got here is a failure to communicate.”

Countless open source projects contribute millions of components to the open source community...

and 11 million developers download these components trusting that with more eyeballs, all bugs are shallow.



But are bugs really shallow? And how do developers find out when bugs are found and fixed? Well, the vast majority don’t.

Hello, Houston, we have a problem.

(and it is even worse than sour milk)

71% OF APPLICATIONS HAVE A **CRITICAL OR SEVERE VULNERABILITY** IN THEIR OPEN SOURCE COMPONENTS*

33% OF ORGANIZATIONS REPORT **BREACHES** RELATED TO A VULNERABLE OPEN SOURCE COMPONENT**

*Based on an open source risk analysis conducted on over 1,500 applications
**Based on the Sonatype 2014 Open Source Development Survey with more than 3500 participants

Worse yet, we can't even answer...

What open source components are being used, and where?

Which components have known security vulnerabilities?

Which production applications are at risk?

What are your license obligations?

Which open source vulnerabilities are most critical?

Do your programmers comply with your policies?

This is a software supply chain issue.

And supply chain issues are not new.

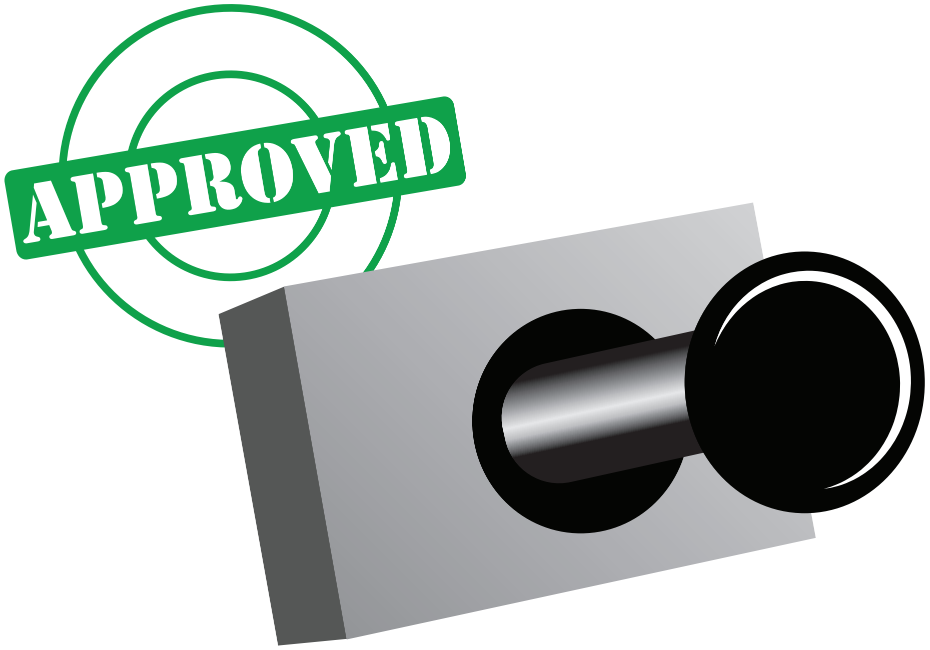
Think about supply chains for things like cars, planes and food. As manufacturing became more complex, it was mandatory to create supply chain processes and controls. You want to buy fresh milk, right? The same is true with vulnerable components that are part of your software supply chain. We can't just use expired and risky components in our applications without some very unpleasant side effects down the road. Heartbleed anyone? We need a complete inventory, like a "bill of materials" for each application. So when there is a "recall" you can act fast and efficiently.

Organizations can't recall a sour component version in their applications because they don't know if they're using it, and don't know which applications are affected.

We need to use the

SAFEST COMPONENTS

in our software supply chain.



Agree?

Here it comes...

The most important information about the easiest thing you can do to close that

application security gap.

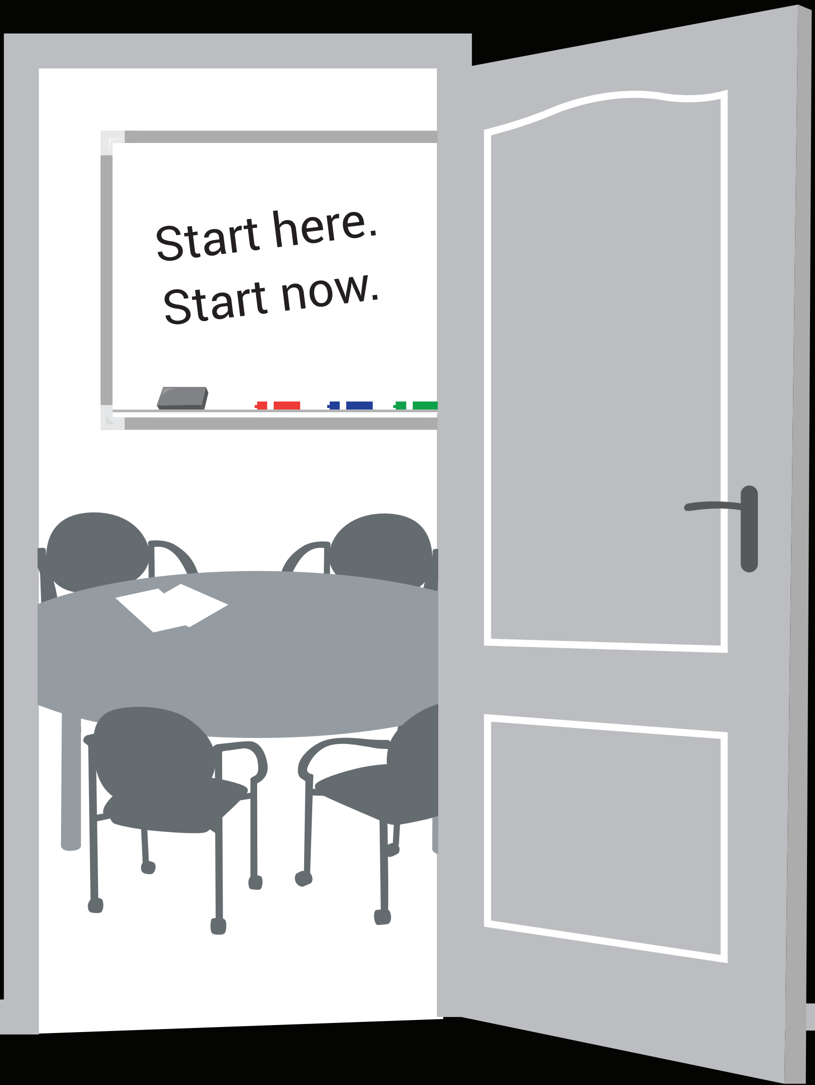
FAST.

Don’t drink
sour milk.

AND DON’T USE VULNERABLE COMPONENTS.

Both are undesirable and easy to avoid.

Here’s how.

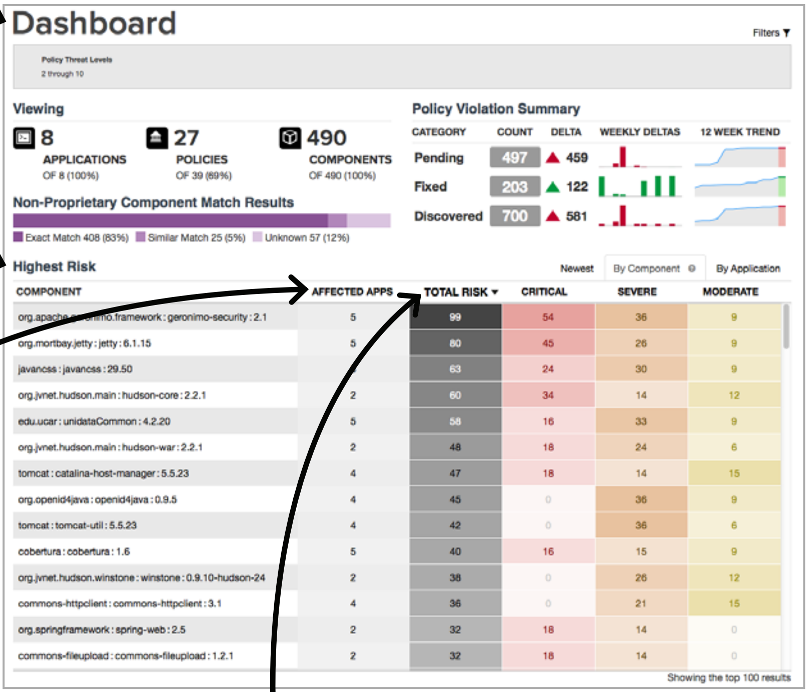


1 See what you're using.

Use a dashboard to visualize threats and prioritize action.

Manage risk by component, development stage and application.

Score risk with a customizable rating system.



Dashboard

Filters

Policy Threat Levels
2 through 10

Viewing

8	27	490
APPLICATIONS	POLICIES	COMPONENTS
OF 8 (100%)	OF 39 (69%)	OF 490 (100%)

Non-Proprietary Component Match Results

Exact Match 408 (83%) · Similar Match 25 (5%) · Unknown 57 (12%)

Policy Violation Summary

CATEGORY	COUNT	DELTA	WEEKLY DELTAS	12 WEEK TREND
Pending	497	▲ 459		
Fixed	203	▲ 122		
Discovered	700	▲ 581		

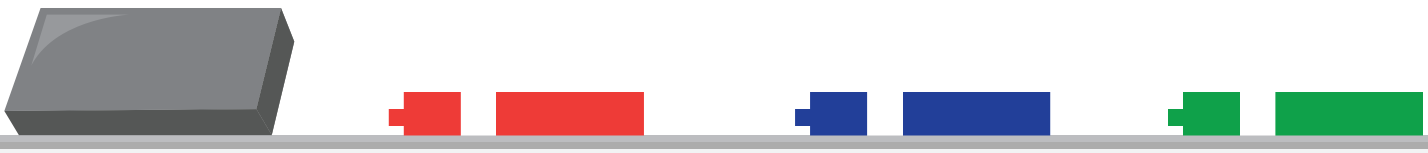
Highest Risk

Newest · By Component · By Application

COMPONENT	AFFECTED APPS	TOTAL RISK ▾	CRITICAL	SEVERE	MODERATE
org.apache.geronimo.framework : geronimo-security : 2.1	5	99	54	36	9
org.mortbay.jetty : jetty : 6.1.15	5	80	45	26	9
javancss : javancss : 29.50	[illegible]	63	24	30	9
org.jvnet.hudson.main : hudson-core : 2.2.1	2	60	34	14	12
edu.ucar : unidataCommon : 4.2.20	5	58	16	33	9
org.jvnet.hudson.main : hudson-war : 2.2.1	2	48	18	24	6
tomcat : catalina-host-manager : 5.5.23	4	47	18	14	15
org.openid4java : openid4java : 0.9.5	4	45	0	36	9
tomcat : tomcat-util : 5.5.23	4	42	0	36	6
cobertura : cobertura : 1.6	5	40	16	15	9
org.jvnet.hudson.winstone : winstone : 0.9.10-hudson-24	2	38	0	26	12
commons-httpclient : commons-httpclient : 3.1	4	36	0	21	15
org.springframework : spring-web : 2.5	2	32	18	14	0
commons-fileupload : commons-fileupload : 1.2.1	2	32	18	14	0

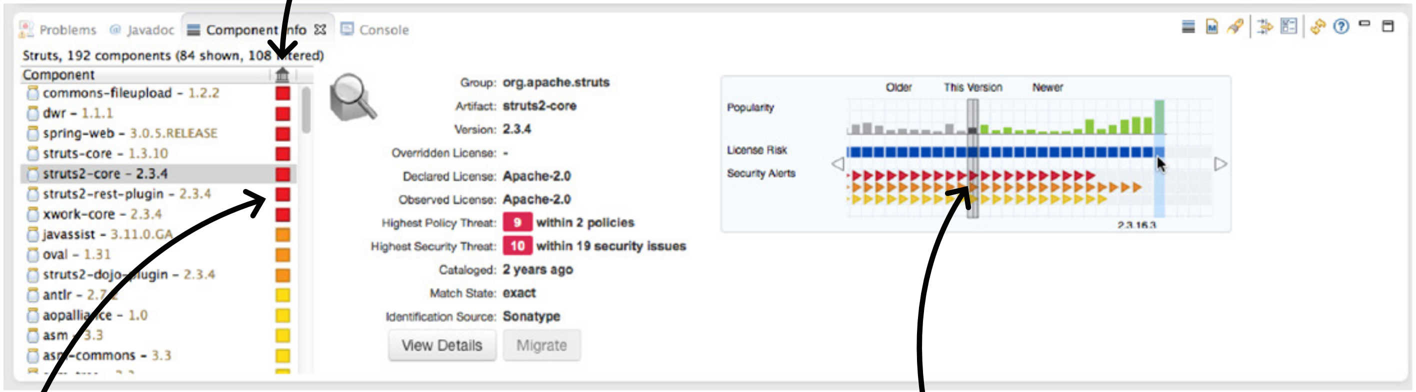
Showing the top 100 results

Easily identify which applications are affected when new vulnerabilities are found.



2 Help your developers.

Give developers color-coded information so they can choose better components from the start. That's as easy as using spellcheck!



Struts, 192 components (84 shown, 108 filtered)

Component	
commons-fileupload - 1.2.2	🟥
dwr - 1.1.1	🟥
spring-web - 3.0.5.RELEASE	🟥
struts-core - 1.3.10	🟥
struts2-core - 2.3.4	🟥
struts2-rest-plugin - 2.3.4	🟥
xwork-core - 2.3.4	🟥
javassist - 3.11.0.GA	🟧
oval - 1.31	🟧
struts2-dojo-plugin - 2.3.4	🟧
antlr - 2.7.2	🟨
aopalliance - 1.0	🟨
asm - 3.3	🟨
asm-commons - 3.3	🟨

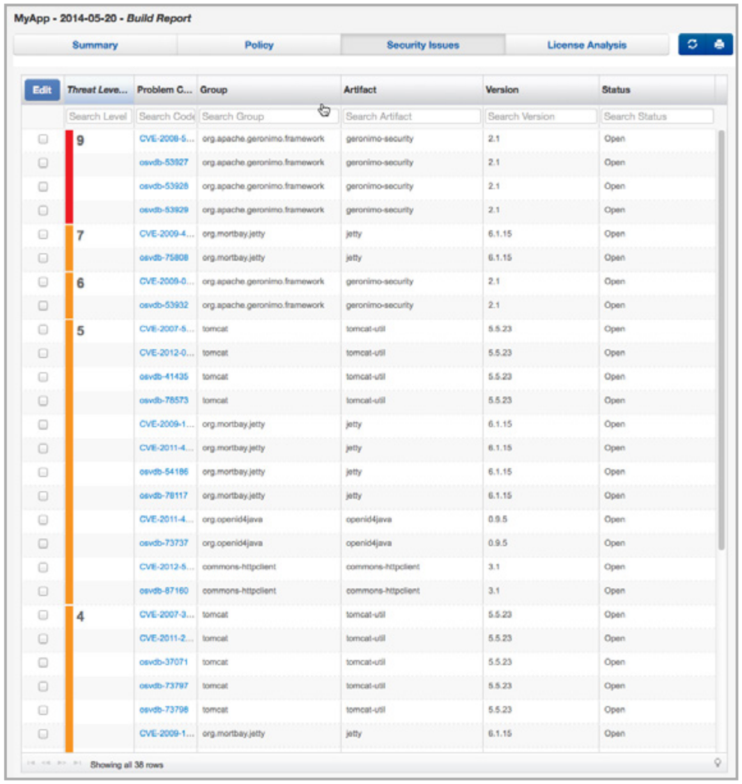
Group: org.apache.struts
Artifact: struts2-core
Version: 2.3.4
Overridden License: -
Declared License: Apache-2.0
Observed License: Apache-2.0
Highest Policy Threat: 9 within 2 policies
Highest Security Threat: 10 within 19 security issues
Cataloged: 2 years ago
Match State: exact
Identification Source: Sonatype

Red is bad. Orange is iffy. Yellow is not good, but not as bad as sour milk...

Simply click on a safer version and update.

3 Create a "Bill of Materials."

Create and monitor a comprehensive bill of materials so you know what components are used, and where.



MyApp - 2014-05-20 - *Build Report*

Threat Leve...	Problem C...	Group	Artifact	Version	Status
9	CVE-2008-5...	org.apache.geronimo.framework	geronimo-security	2.1	Open
	osvdb-53927	org.apache.geronimo.framework	geronimo-security	2.1	Open
	osvdb-53928	org.apache.geronimo.framework	geronimo-security	2.1	Open
	osvdb-53929	org.apache.geronimo.framework	geronimo-security	2.1	Open
7	CVE-2009-4...	org.mortbay.jetty	jetty	6.1.15	Open
	osvdb-75808	org.mortbay.jetty	jetty	6.1.15	Open
6	CVE-2009-0...	org.apache.geronimo.framework	geronimo-security	2.1	Open
	osvdb-53932	org.apache.geronimo.framework	geronimo-security	2.1	Open
5	CVE-2007-5...	tomcat	tomcat-util	5.5.23	Open
	CVE-2012-0...	tomcat	tomcat-util	5.5.23	Open
	osvdb-41435	tomcat	tomcat-util	5.5.23	Open
	osvdb-78573	tomcat	tomcat-util	5.5.23	Open
	CVE-2009-1...	org.mortbay.jetty	jetty	6.1.15	Open
	CVE-2011-4...	org.mortbay.jetty	jetty	6.1.15	Open
	osvdb-54186	org.mortbay.jetty	jetty	6.1.15	Open
	osvdb-78117	org.mortbay.jetty	jetty	6.1.15	Open
	CVE-2011-4...	org.openid4java	openid4java	0.9.5	Open
	osvdb-73737	org.openid4java	openid4java	0.9.5	Open
	CVE-2012-5...	commons-httpclient	commons-httpclient	3.1	Open
	osvdb-87160	commons-httpclient	commons-httpclient	3.1	Open
4	CVE-2007-3...	tomcat	tomcat-util	5.5.23	Open
	CVE-2011-2...	tomcat	tomcat-util	5.5.23	Open
	osvdb-37071	tomcat	tomcat-util	5.5.23	Open
	osvdb-73797	tomcat	tomcat-util	5.5.23	Open
	osvdb-73798	tomcat	tomcat-util	5.5.23	Open
	CVE-2009-1...	org.mortbay.jetty	jetty	6.1.15	Open

Showing all 38 rows

How long will it take?

~~Five months~~

~~Five weeks~~

~~Five days~~

Five minutes

Actually, it takes less time, but we didn't think you'd believe us.

Never has anything with this much **IMPACT** been this

Do **YOU** want to make a difference?
Together we can fix this thing.

www.sonatype.com/easy

That is our **RANT.**

Thank you for listening.

Feel like a little RANT of your own? Share!

Share on Facebook   Tweet   Share   Share   Email

Watch for our next RANT called ***Raise the B.A.R.R. on Open Source Components: Ban Avoidable Risk & Rework.*** Don't want to miss it? Follow us on Facebook or Twitter to be the first to know.

Like   Follow

Brought to you by

Sonatype

We make it easy to build trusted software and keep it that way over time.

www.sonatype.com

Every day, developers rely on millions of third party and open source building blocks—known as components—to build the software that runs our world. Sonatype ensures that only the best components are used throughout the software development lifecycle so that organizations don't have to make the tradeoff between going fast and being secure. Policy automation, ongoing monitoring and pro-active alerts makes it easy to have full visibility and control of components throughout the software supply chain so that applications start secure and remain that way over time. Sonatype is privately held with investments from New Enterprise Associates (NEA), Accel Partners, Bay Partners, Hummer Winblad Venture Partners and Morgenthaler Ventures. Visit: www.sonatype.com