A chain of Trust

- How to implement a secure supply chain approach

Ilkka Turunen – Solutions Architect - Sonatype
@IlkkaTurunen

The credibility slide

B.Sc (Eng) Software - 2011

LEAN / AGILE Research:

- **Product manager** FreeNest (ALM toolkit) 2009 - 2013
- **Project subject matter expert**: JAMK University 2009 - 2013

CI / CD Expertise:

- **Cloud Architect**: Cloudreach 2014-2015
  - LEAN / CD pipeline architecture
  - Analytics systems engineering
- **Founder** of OpenStack Finland User Group
- **Contributor** to Cloud Software Program Open Cloud Stack line now used as reference architecture for public DCs in Finland

Business:

- **Co-founder / CTO:** Nestronite 2009-2013

Current:

- **Solutions Architect** EMEA / APJ - Sonatype

A hands-on demo of CVE-2015-8103

- **This Article published:** November 6th 2015
- **Mitigation**: 6th of November
- **Fix committed to source;** Nov 7th 2015
- **Fixed version released:** Nov 11th 2015

- This much you probably know ☺

1/3 haven't been patched (End of May 16)

Search: "X-Jenkins" port "8080"

Export of N=10,000:

- 1.x series
  - **7795** (78% of all servers)
- All fixed versions (2.x, 1.x)
  - **6834** (68% of all servers)
- Servers with Fixed version (1-series):
  - **4629** (46% of all servers)
- Vulnerable Servers:
  - **3166** (32% of all servers)

ROMA MMXVI

EUROPE

Why is this happening?

- Why are there 1/3 unpatched instances left?
- What is the real cause?

3rd party components are behind this: commons-collections

November 6, 2015

What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common? This Vulnerability.

By @breenmachine

What?

The most underrated, underhyped vulnerability of 2015 has recently come to my attention, and I'm about to bring it to yours. No one gave it a fancy name, there were no press

Further analysis of 3rd party prevalence

- **Applications:**

- **Organizations downloads from Central Repo (2015):**

Orders	Quality Control		
Average downloads	# with known vulnerabilities	% with known vulnerabilities	% known vulnerabilities (2013 or older)
240,757	15,337	6.4%	66.3%

ROMA MMXVI | APPSEC EUROPE

Beyond Heartbleed: OpenSSL in 2014

(31 in NIST's NVD thru December)

- CVE-2014-3470 6/5/2014 CVSS Severity: 4.3 MEDIUM ← **SIEMENS** *
- CVE-2014-0224 6/5/2014 CVSS Severity: 6.8 MEDIUM ← **SIEMENS** *
- CVE-2014-0221 6/5/2014 CVSS Severity: 4.3 MEDIUM
- CVE-2014-0195 6/5/2014 CVSS Severity: 6.8 MEDIUM
- CVE-2014-0198 5/6/2014 CVSS Severity: 4.3 MEDIUM ← **SIEMENS** *
- CVE-2013-7373 4/29/2014 CVSS Severity: 7.5 HIGH
- CVE-2014-2734 4/24/2014 CVSS Severity: 5.8 MEDIUM ** DISPUTED **
- CVE-2014-0139 4/15/2014 CVSS Severity: 5.8 MEDIUM
- CVE-2010-5298 4/14/2014 CVSS Severity: 4.0 MEDIUM
- **CVE-2014-0160 4/7/2014 CVSS Severity: 5.0 MEDIUM ← HeartBleed**
- CVE-2014-0076 3/25/2014 CVSS Severity: 4.3 MEDIUM
- CVE-2014-0016 3/24/2014 CVSS Severity: 4.3 MEDIUM
- CVE-2014-0017 3/14/2014 CVSS Severity: 1.9 LOW
- CVE-2014-2234 3/5/2014 CVSS Severity: 6.4 MEDIUM
- CVE-2013-7295 1/17/2014 CVSS Severity: 4.0 MEDIUM
- CVE-2013-4353 1/8/2014 CVSS Severity: 4.3 MEDIUM
- CVE-2013-6450 1/1/2014 CVSS Severity: 5.8 MEDIUM
- …

As of 2014, internet scans by MassScan reveal 300,000 of original 600,000 remain unpatched or unpatchable

ROMA MMXVI

APPSEC EUROPE

Source: Security is Dead. Long Live Rugged DevOps: IT at Ludicrous Speed - Josh Corman, Gene Kim

Year in vulnerabilities 2015

- **CVE-2015-8103**
  - CVSS: **7.5 HIGH**
- Other vulnerabilities in NVD on Nov-Dec
  - 62 HIGH (CVSS 7-8.9)
  - 19 CRITICAL (CVSS 9-10)

10 CVEs == 97% of attacks in 2014

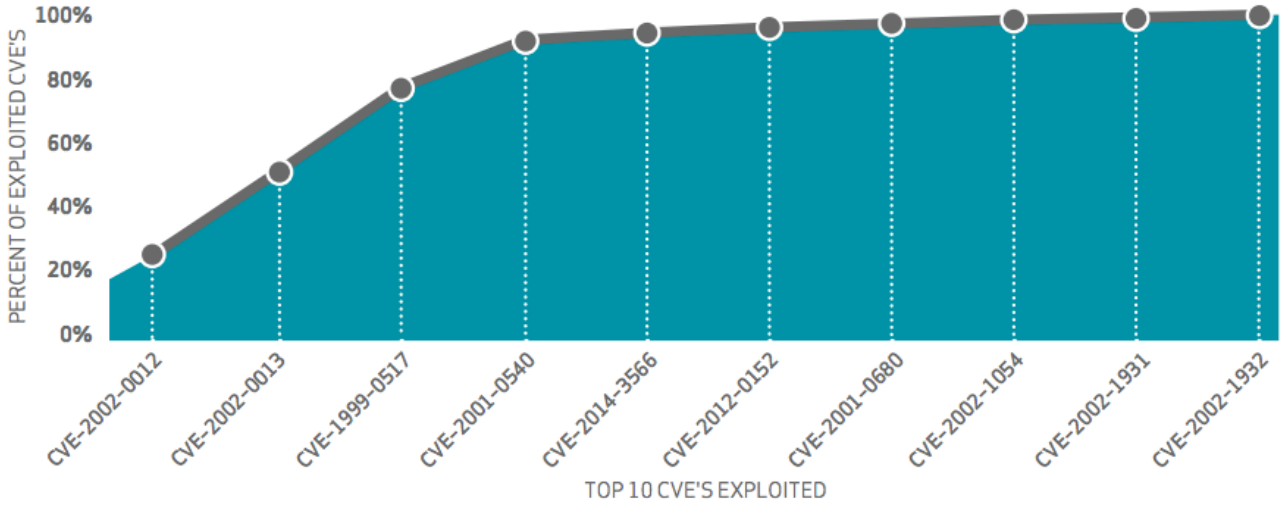
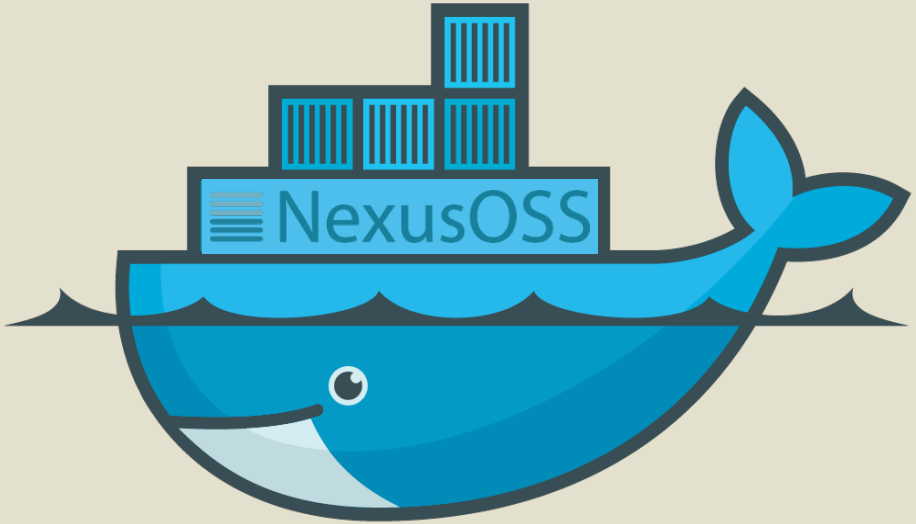


Figure 11.

Cumulative percentage of exploited vulnerabilities by top 10 CVEs

Source: Verizon Data Breach Report 2015



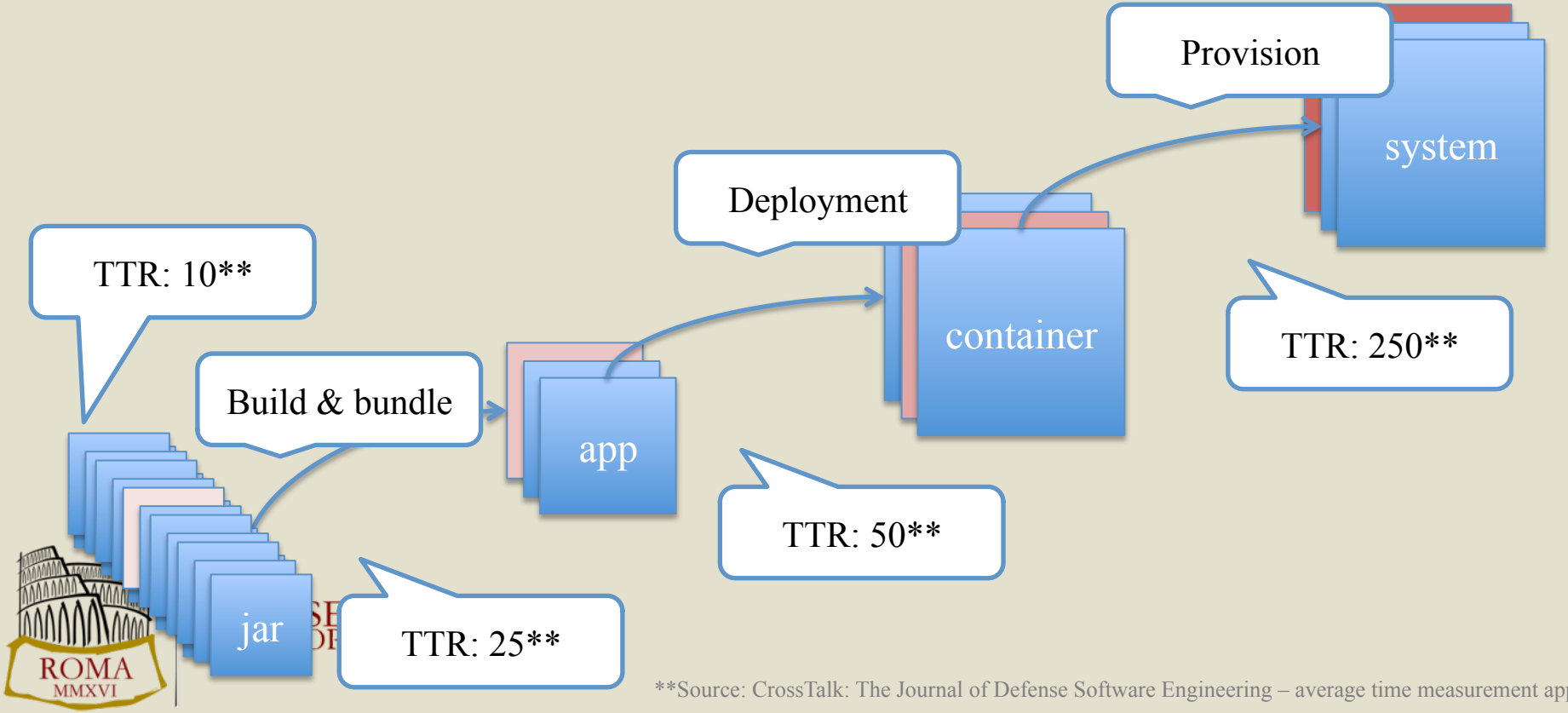
36%
High vulnerabilities

40%
High Vulnerabilities
In 2015 images

23%
High Vulnerabilities
In latest images

http://www.banyanops.com/blog/analyzing-docker-hub/

Compound risk: layered opacity



**Source: CrossTalk: The Journal of Defense Software Engineering – average time measurement applied

Section 2 – What does the law say?

SO HOW ARE WE TRYING TO PREVENT IT FROM HAPPENING?

A9 Using Components with Known Vulnerabilities

Application Specific	Exploitability AVERAGE	Prevalence WIDESPREAD	Detectability DIFFICULT	Impact MODERATE	Application / Business Specific
Some vulnerable components (e.g., framework libraries) can be identified and exploited with automated tools, expanding the threat agent pool beyond targeted attackers to include chaotic actors.	Attacker identifies a weak component through scanning or manual analysis. He customizes the exploit as needed and executes the attack. It gets more difficult if the used component is deep in the application.	Virtually every application has these issues because most development teams don't focus on ensuring their components/libraries are up to date. In many cases, the developers don't even know all the components they are using, never mind their versions. Component dependencies make things even worse.		The full range of weaknesses is possible, including injection, broken access control, XSS, etc. The impact could range from minimal to complete host takeover and data compromise.	Consider what each vulnerability might mean for the business controlled by the affected application. It could be trivial or it could mean complete compromise.

PCI – DSS
Req 6

6.1 Establish a process to identify security vulnerabilities, by using reputable outside sources for security vulnerability information, and assign a risk ranking (for example, as 'high,' 'medium,' or 'low') to newly discovered security vulnerabilities.

6.2 Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor-supplied security patches. Install critical security patches within one month of release.

IEC-62304

SOUP stands for
software of unknown (or uncertain) pedigree (or provenance),

Specific practices to take when using **SOUP** as part of a medical device may include **review of the vendor's software development process**, **use of static program analysis** by the vendor, design artifacts, and safety guidance

APPSEC EUROPE

Source: Wikipedia

PCI DSS Requirements v3.0	Milestone	Status *Please enter "yes" if fully compliant with the requirement*	If status is "N/A", please explain why requirement is Not Applicable	If status is "No", please complete the following		
				Stage of Implementation	Estimated Date for Completion of Milestone	Comments
Requirement 1: Install and maintain a firewall configuration to protect cardholder data						
1.1 Establish and implement firewall and router configuration standards that include the following:						
1.1.1 A formal process for approving and testing all network connections and changes to the firewall and router configurations	6					
1.1.2 Current network diagram that identifies all connections between the cardholder data environment and other networks, including any wireless networks	1					
1.1.3 Current diagram that shows all cardholder data flows across systems and networks.	1					
1.1.4 Requirements for a firewall at each Internet connection and between any demilitarized zone (DMZ) and the Internal network zone	2					
1.1.5 Description of groups, roles, and responsibilities for management of network components.	6					
1.1.6 Documentation and business justification for use of all services, protocols, and ports allowed, including documentation for security features implemented for those protocols considered to be insecure. Examples of insecure services, protocols, or ports include but are not limited to FTP, Telnet, POP3, IMAP, and SNMP v1 and v2	2					
1.1.7 Requirement to review firewall and router rule sets at least every six months.	6					
1.2 Build firewall and router configurations that restrict connections between untrusted networks and any system components in the cardholder data environment. **Note**: An "untrusted network" is any network that is external to the networks belonging to the entity under review, and/or which is out of the entity's ability to control or manage.						
1.2.1 Restrict inbound and outbound traffic to that which is necessary for the cardholder data environment, and specifically deny all other traffic.	2					
1.2.2 Secure and synchronize router configuration files.	2					
1.2.3 Install perimeter firewalls between any all wireless networks and the cardholder data environment, and configure these firewalls to deny or, control (if such traffic is necessary for business purposes), permit only authorized any traffic from between the wireless environment into and the cardholder data environment.	2					
1.3 Prohibit direct public access between the Internet and any system component in the cardholder data environment.						
1.3.1 Implement a DMZ to limit inbound traffic to only system components that provide authorized publicly accessible services, protocols, and ports.	2					
1.3.2 Limit inbound Internet traffic to IP addresses within the DMZ.	2					
1.3.3 Do not allow any direct connections inbound or outbound for traffic between the Internet and the cardholder data environment.	2					
1.3.4 Implement anti-spoofing measures to detect and block forged source IP addresses from entering the network.	2					
1.3.5 Do not allow unauthorized outbound traffic from the cardholder data environment to the Internet.	2					
1.3.6 Implement stateful inspection, also known as dynamic packet filtering. (That is, only "established" connections are allowed into the network.)	2					
1.3.7 Place system components that store cardholder data (such as a database) in an internal network zone						

The road is always paved with good intentions

Antipatterns

- **Security / CVE Checklists**
  - Human-led initiatives
  - Human-led considerations
- **Bulk approvals of components**
  - Again, smarter-than-thou
  - Doesn't scale
- **Deplugging completely**
  - The law may require it but it sure isn't nice

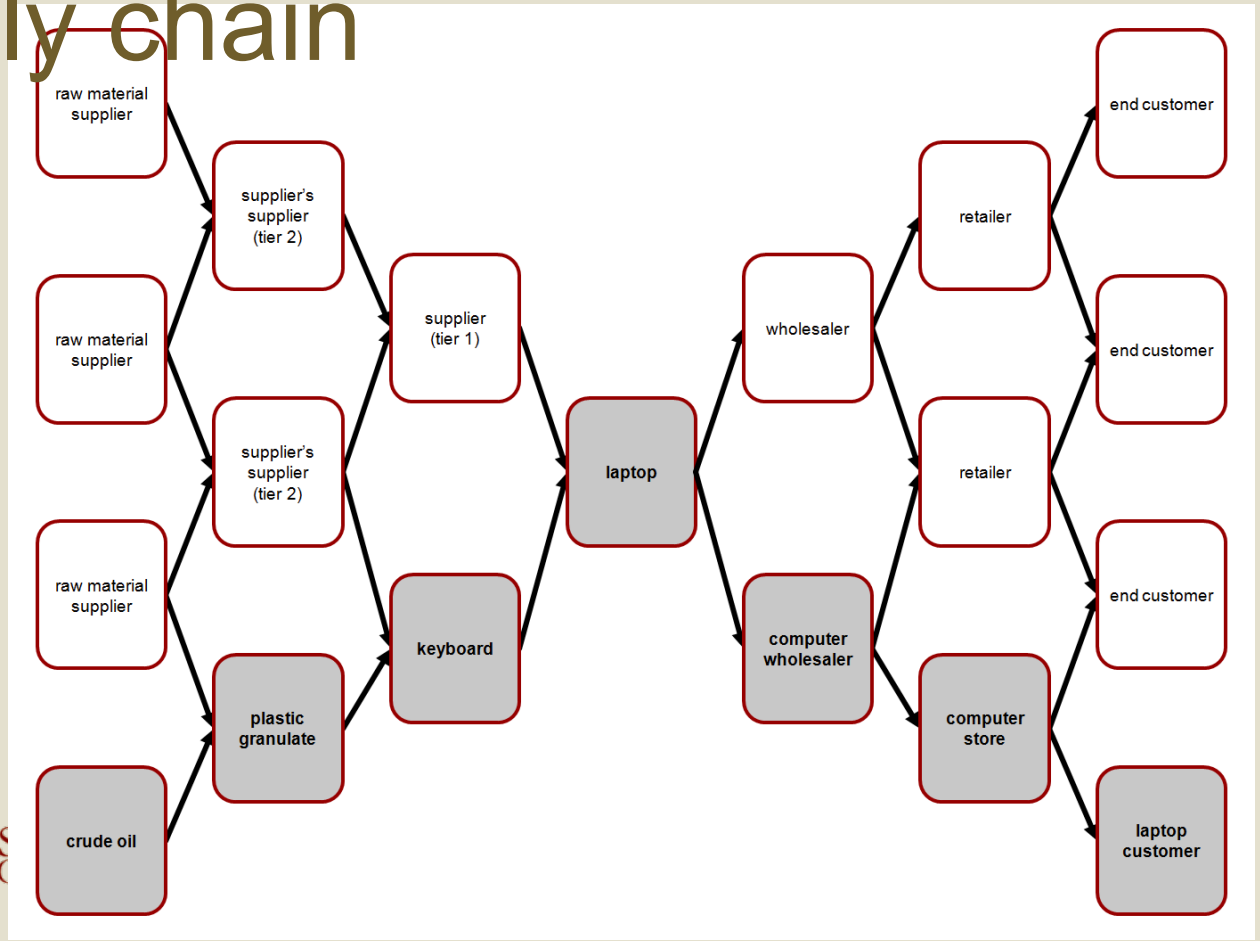
Outcomes

- Top 3 items are followed. The rest are fixed 'later'
- As organisation grows process grinds to a halt.
  - Shadow sourcing orgs emerge (hotspots)
- Not today

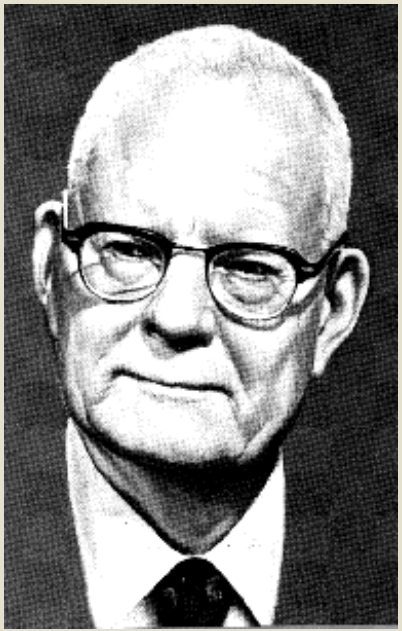
Introducing process where it matters

SOFTWARE SUPPLY CHAIN MANAGEMENT AND RUGGED DEVOPS

Supply chain



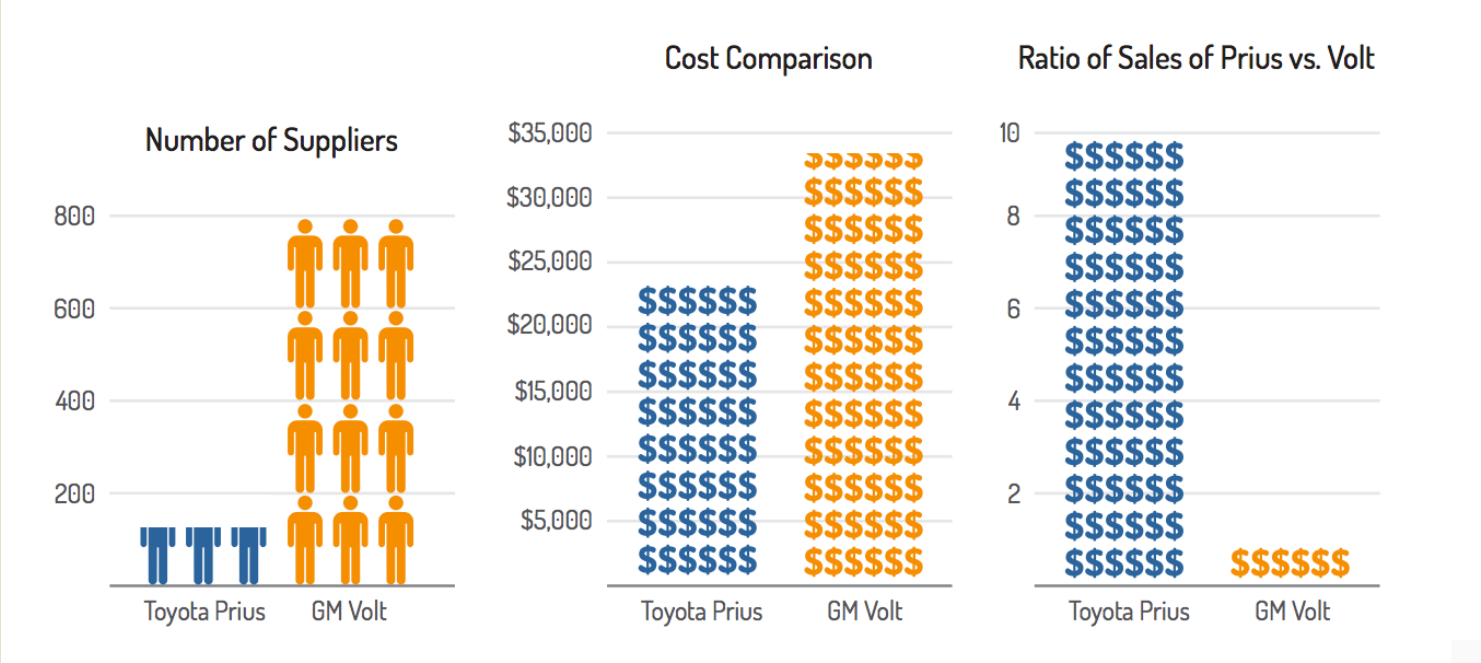
Sage advice from the man who helped bring Japan back from the brink



W. Edwards Deming's 14 Principles included:

- **Cease dependence on inspection to achieve quality.** Eliminate the need for inspection on a mass basis by building quality into the product in the first place.
- End the practice of awarding business on the basis of price tag. Instead, minimize total cost. **Move toward a single supplier** for any one item, on a long-term relationship of loyalty and trust.

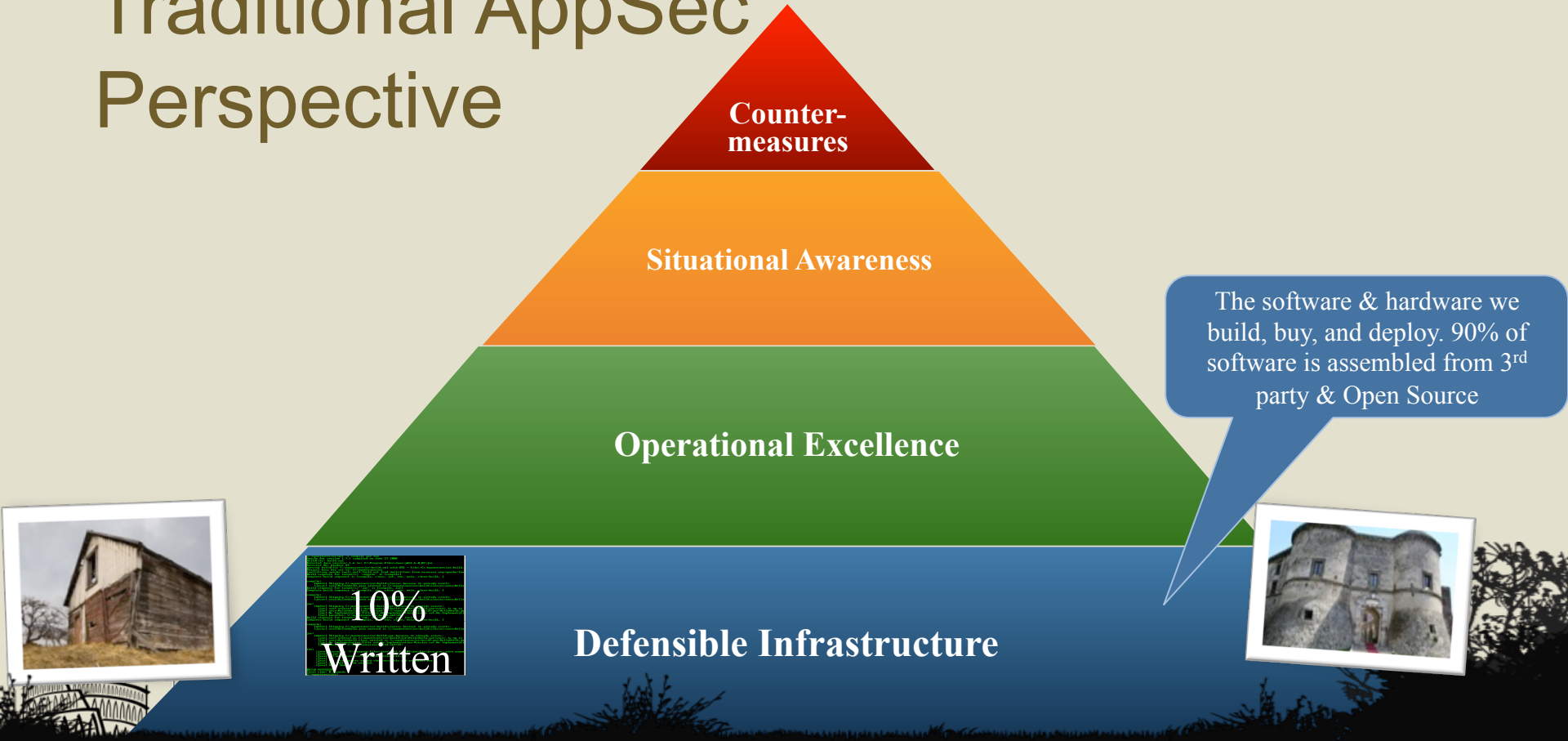
Benefits seen in other industries



ROMA MMXVI

APPSEC EUROPE

Traditional AppSec Perspective



Source: Security is Dead. Long Live Rugged DevOps: IT at Ludicrous Speed - Josh Corman, Gene Kim

Dependency managers == Software supply chain managers

Java / Maven2

```
<dependencies>
   <dependency>
  <groupId>javax.activation</groupId>
  <artifactId>activation</artifactId>
  <version>1.1</version>
</dependency>
```

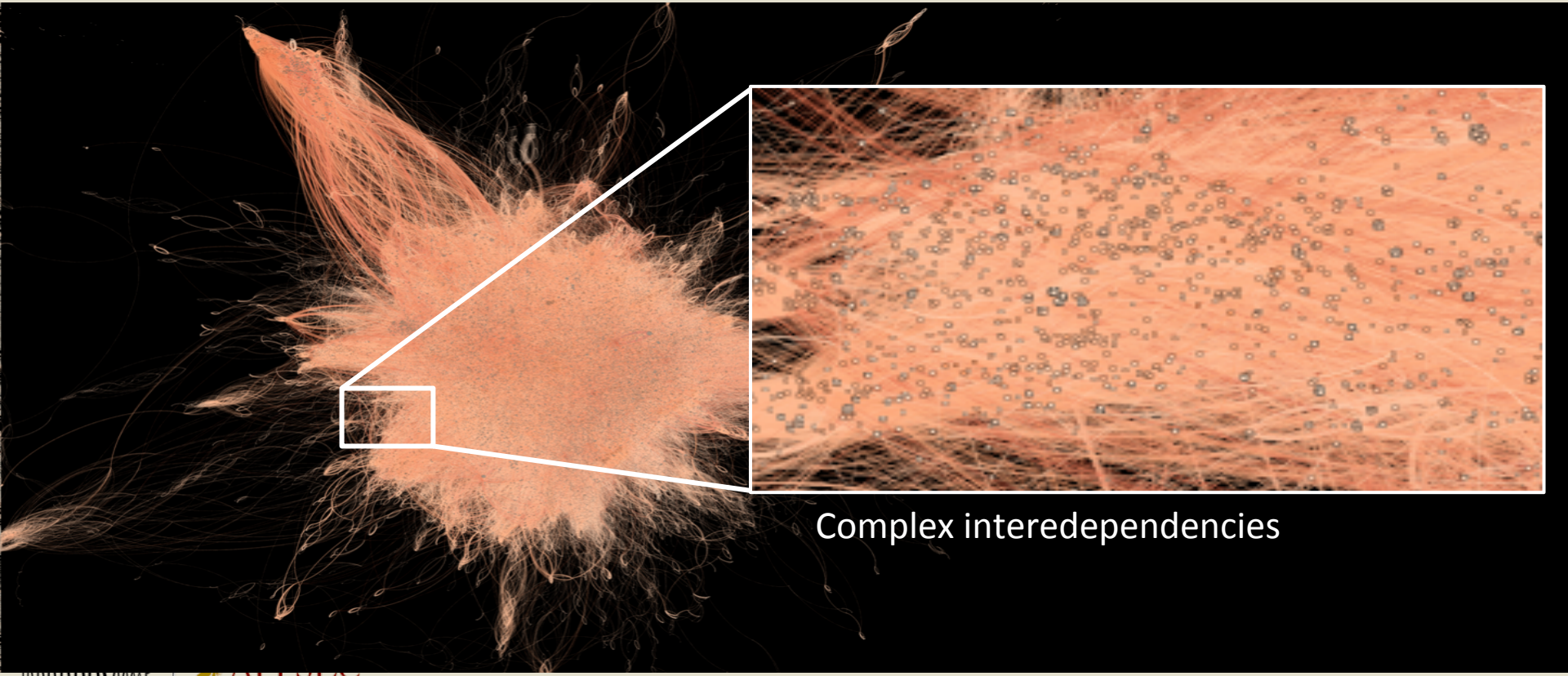
Ruby / Gem

```
source 'https://rubygems.org'

ruby '2.1.0'

gem 'rails', '4.1.0'
gem 'unicorn'
gem 'pg'
gem 'sass-rails', '~> 4.0.3'
gem 'uglifier', '>= 1.3.0'
gem 'coffee-rails', '~> 4.0.0'
```

Node.js / NPM

```
"dependencies": {
    "glob": "^5.0.3",
    "json-parse-helpfulerror": "^1.0.2",
    "normalize-package-data": "^2.0.0"
  },
  "devDependencies": {
    "standard": "^3.3.1",
    "tap": "^1.2.0"
  },
  "optionalDependencies": {
    "graceful-fs": "^4.1.2"
  },
  "license": "ISC"
}
```

Transitive dependencies (Maven central Aug 2015)



Complex interedependencies

Translated into a Software Context

1. **Control the amount** and quality of suppliers **or components** you use
2. **Standardise your component catalog** as opposed to allowing every team to reinvent their toolkit
3. **Leverage automated quality controls and governance guidelines** as early as possible in the software life cycle to eliminate easily avoidable risk.
4. Maintain **a bill of materials** of all software and their underlying components
5. **Institute leadership** that can help improve the overall state of the component supply chain

1 – CONTROL THE AMOUNT AND QUALITY OF SUPPLIERS

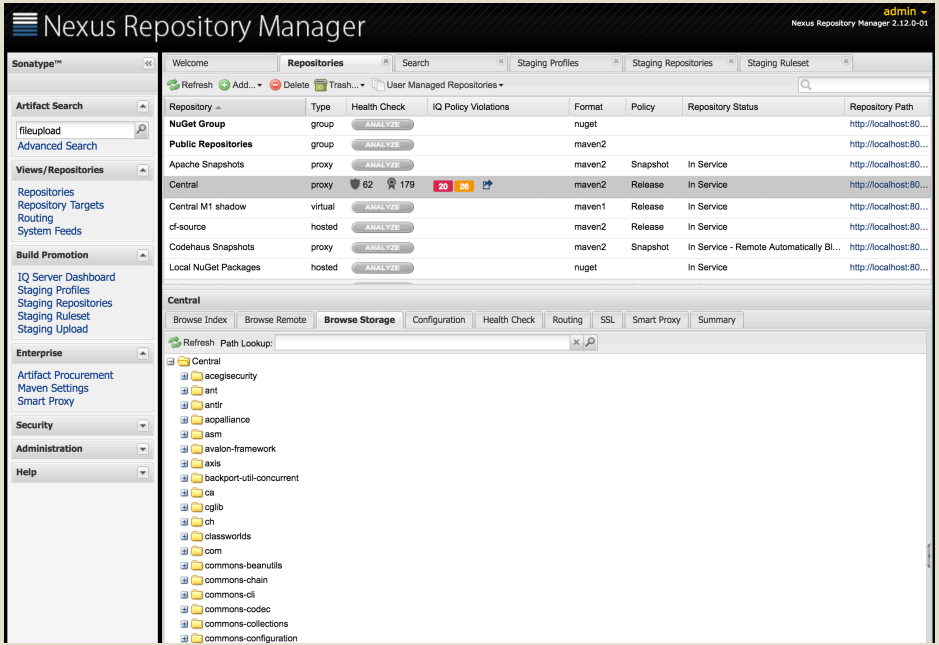
What to look out for in a good project to source

- Release Frequency (Latest / MTR)
- Popularity in Ecosystem (Dead project vs Stable)
- Internal popularity
- Number of vulnerabilities
- MTTR of said vulns
- Licenses
- Pull Requests Monthly Avg

APPSEC EUROPE

Artifact repositories are key to implementing this quality control

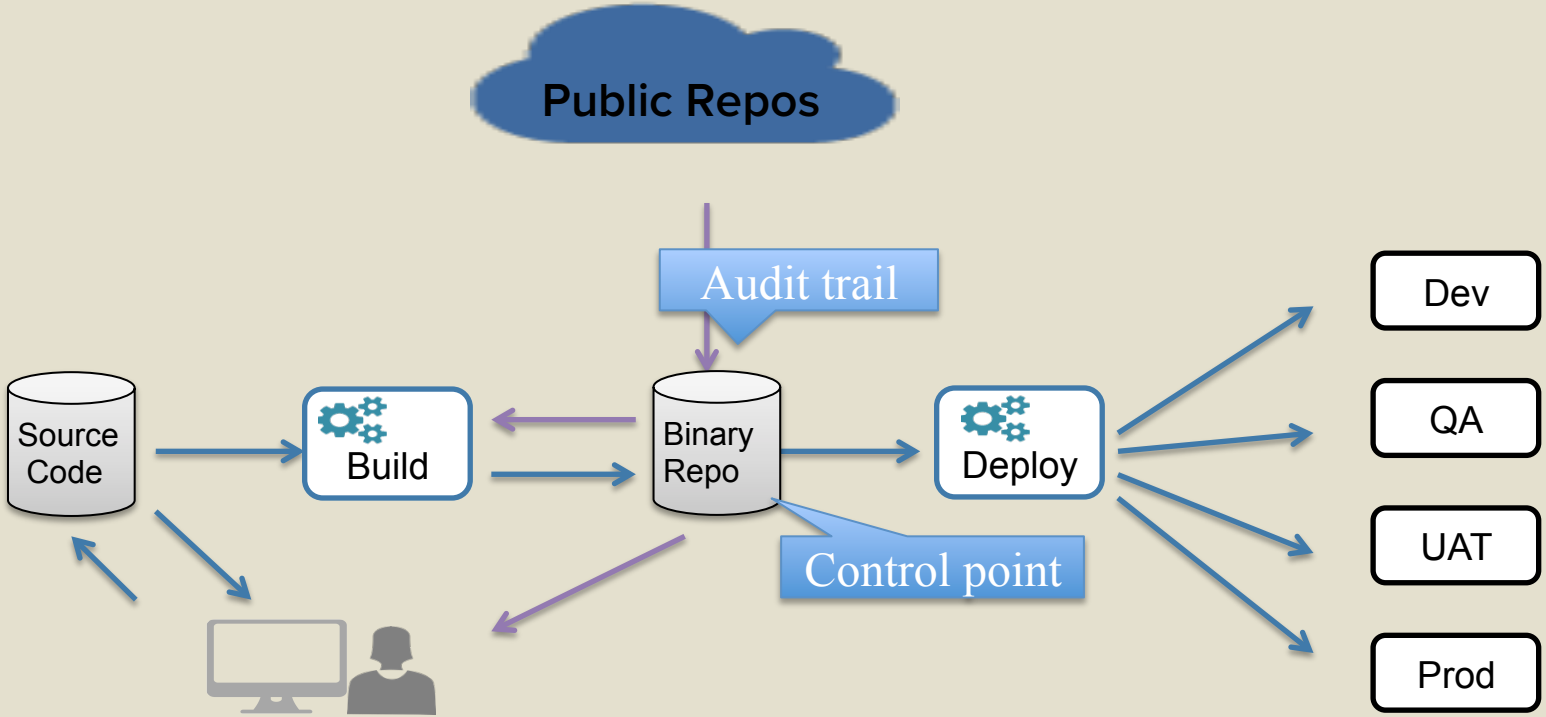
- Catalogs
- Keep track of
- Audit trails of all downloads
- Prevent shadow acquirements (well…. As best as you can ☺)



ROMA MMXVI

APPSEC EUROPE

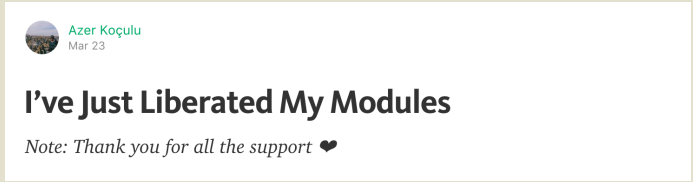
Software Factory & Component Based Development



#npmgate – March 22nd 2016

- “In this case, though, **without warning to developers** of dependent projects, Azer **unpublished his kik package and 272 other packages**.

- One of those was ***left-pad***. **This impacted many thousands of projects**. Shortly after 2:30 PM (Pacific Time) on Tuesday, March 22, we began observing **hundreds of failures per minute, as dependent projects — and their dependents, and their dependents… — all failed when requesting the now-unpublished package.”**



ROMA MMXVI

APPSEC EUROPE

http://blog.npmjs.org/post/141577284765/kik-left-pad-and-npm

```
module.exports = leftpad;

function leftpad (str, len, ch) {
  str = String(str);
  var i = -1;
  if (!ch && ch !== 0) ch = ' ';
  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }
  return str;
}
```

APPSEC EUROPE


ROMA
MMXVI
APPSEC
EUROPE
2. STANDARDISE YOUR CATALOG

Let’s refresh the stats

- **Average application:**

- **Assume an Organisation:**
- 30 Applications * 106 components * 5 versions * 60% unique components in app = **9540 Unique Components**

ROMA MMXVI

APPSEC EUROPE

Standardisation

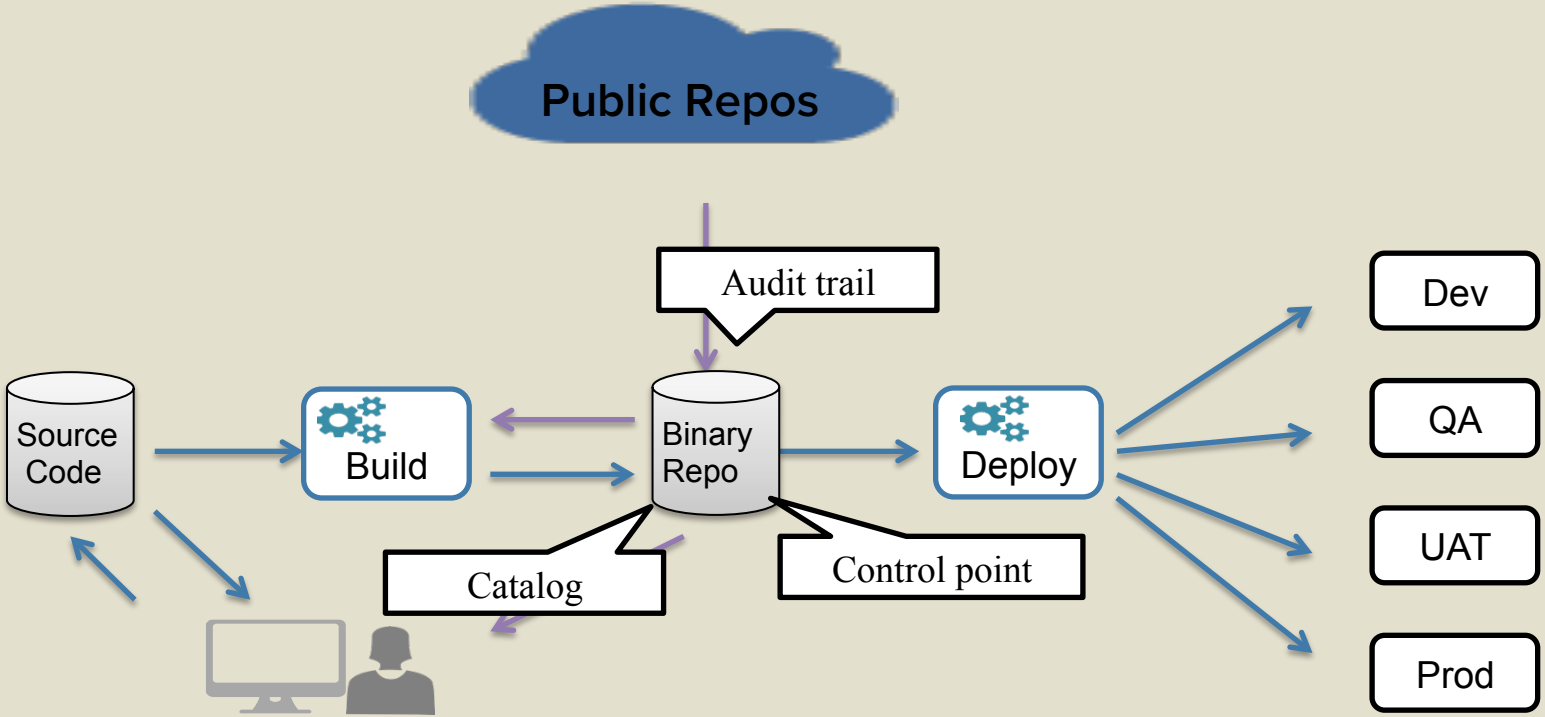
- Important to know the tools to build your defensible castle
- There is no one-size fits all solution to standardisation as teams and business lines differ

APPSEC EUROPE

Standardisation guidelines

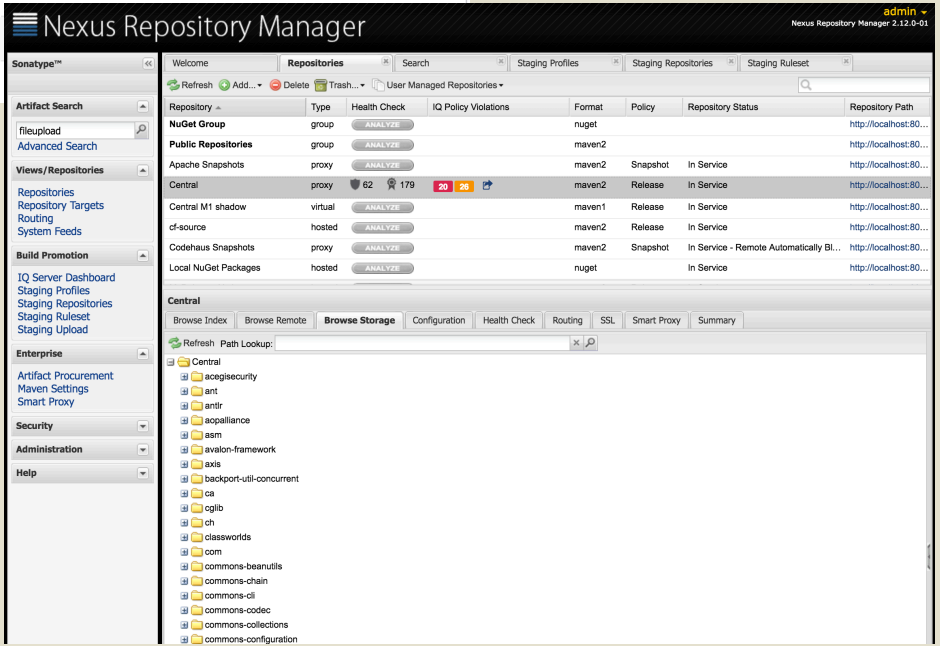
- Be picky about components.
  - Use cases?
  - Licensing?
  - Type? Should we be using 5 different auth libraries as a company or just one?
  - How many versions should we accept? N-1? N-2?

Software Factory & Component Based Development



How it could look like

Only one test fw	junit : junit : 3.8.1
	junit : junit : 3.8.2
	junit : junit : 4.11
	junit : junit : 4.4
	junit : junit : 4.8.2



ROMA MMXVI

APPSEC EUROPE

3. LEVERAGE AUTOMATION AND EXISTING WORKFLOWS

vBSIMM Framework model

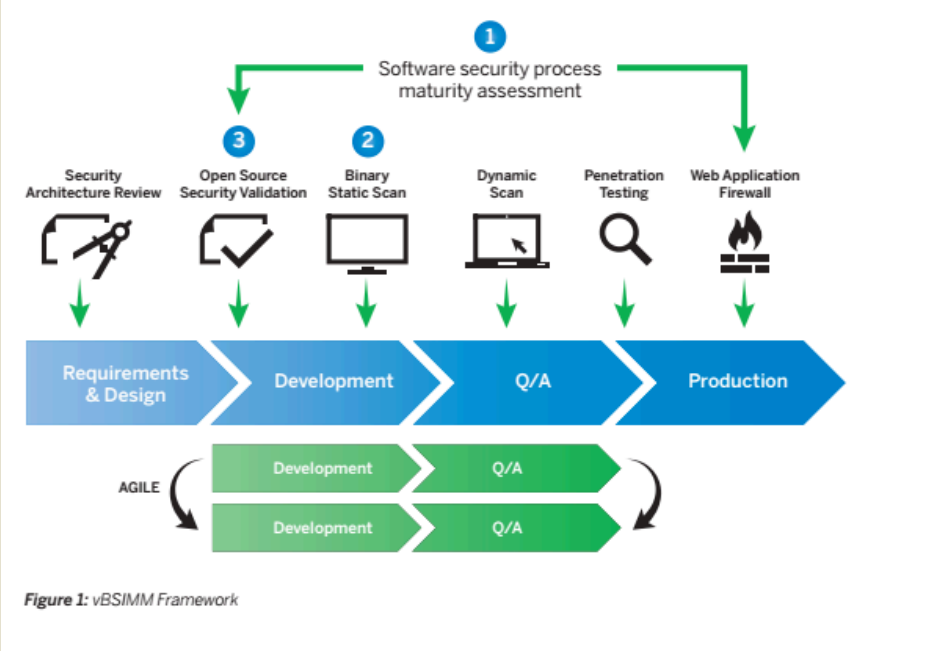
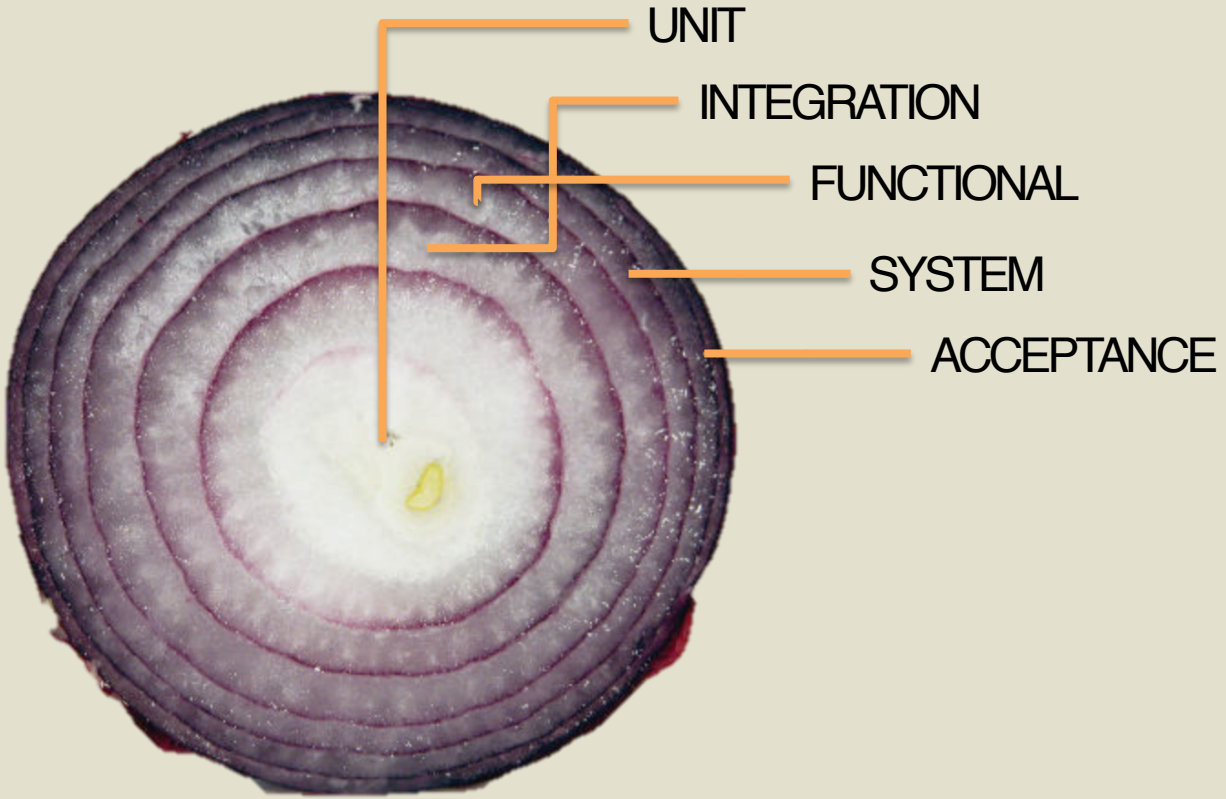


Figure 1: *vBSIMM Framework*

Source: FS-ISAC Appropriate Software Security Control Types for Third Party Service and Product Providers

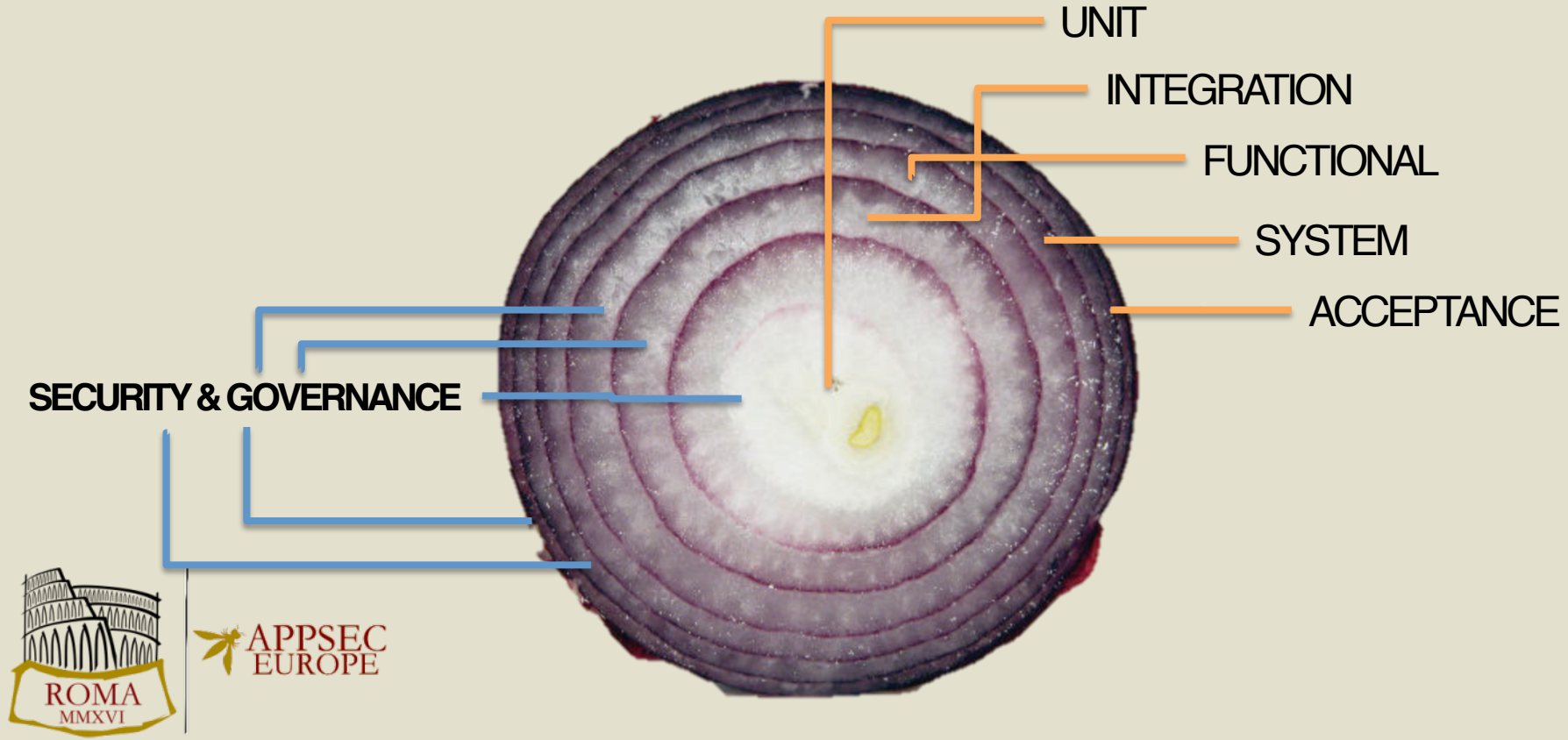
The onion model of software testing



ROMA MMXVI

APPSEC EUROPE

The onion model of * testing



ROMA MMXVI

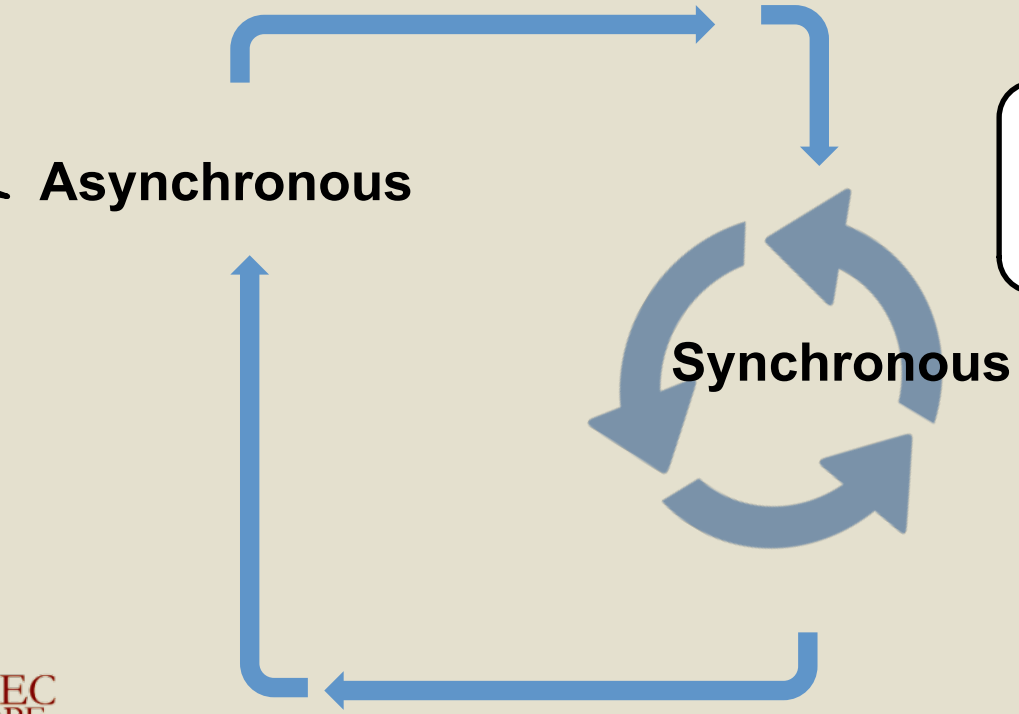
APPSEC EUROPE

```
[INFO] Evaluating policies... (ETA 30s)
[INFO]
------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO]
------------------------------------------------------------------------
[INFO] Total time: 37.210 s
[INFO] Finished at: 2015-10-21T18:38:53+01:00
[INFO] Final Memory: 17M/496M
[INFO]
------------------------------------------------------------------------
[ERROR] Failed to execute goal com.sonatype.clm:clm-maven-plugin:
2.1.1:evaluate (default-cli) on project WebGoat: Sonatype CLM reports
policy failing due to
[ERROR] Policy(No high sec vulnerabilities) [
[ERROR] Component(gav=commons-fileupload:commons-fileupload:1.2.1,
hash=384faa82e193d4e4b054) [
[ERROR] Constraint(No secs) [Security Vulnerability present because:
Found 4 Security Vulnerabilities, Security Vulnerability Severity >= 7
because: Found Security Vulnerability with Severity >= 7] ]]
```

Use the CI Pipeline to incrementally improve security practices

Deep dive checks

Human analysis and interpretation. Code reviews, audits, etc

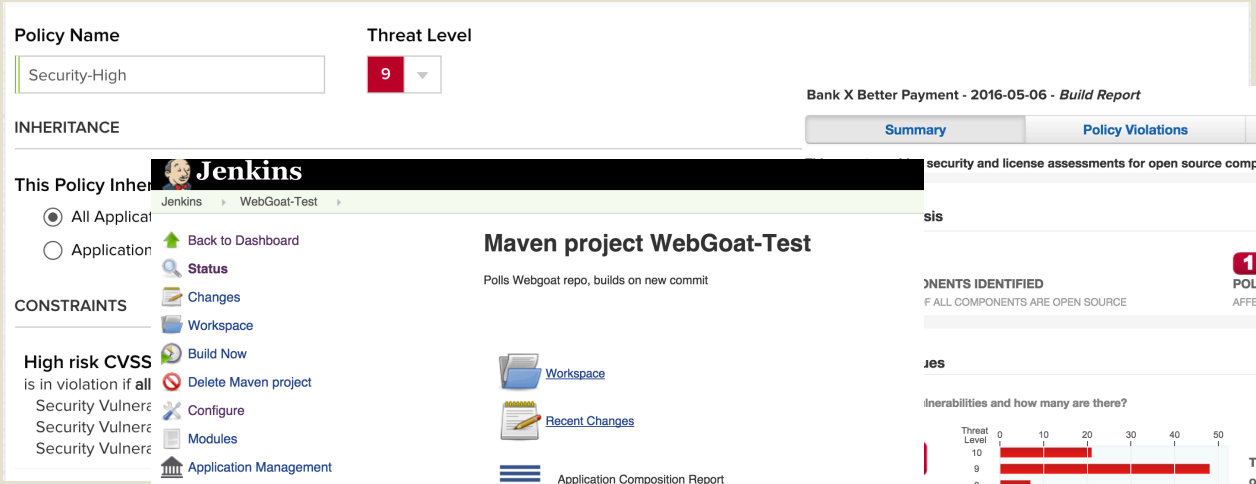


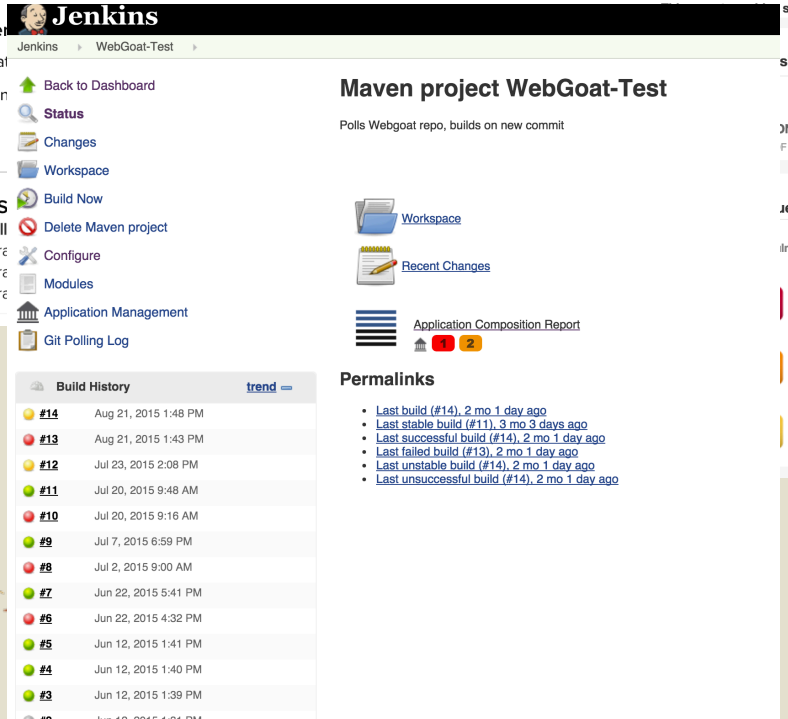
On-the-go analysis
as you build
Amplify signals
Fast feedback on a sufficient enough level of detail

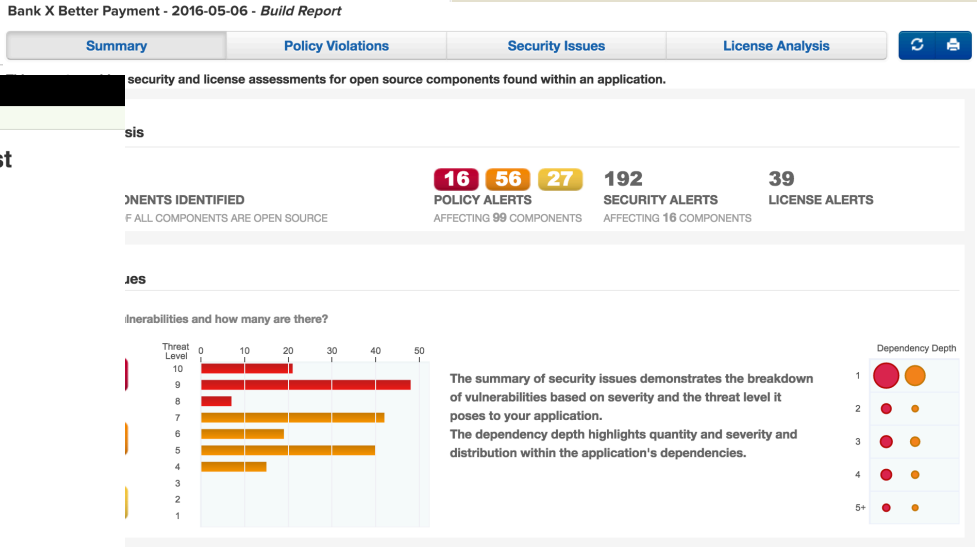
ROMA MMXVI

APPSEC EUROPE

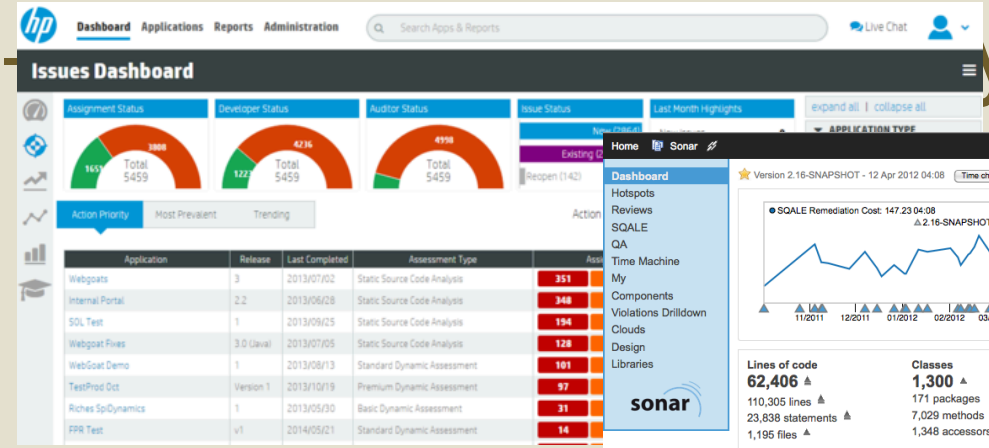
Synchronous testing

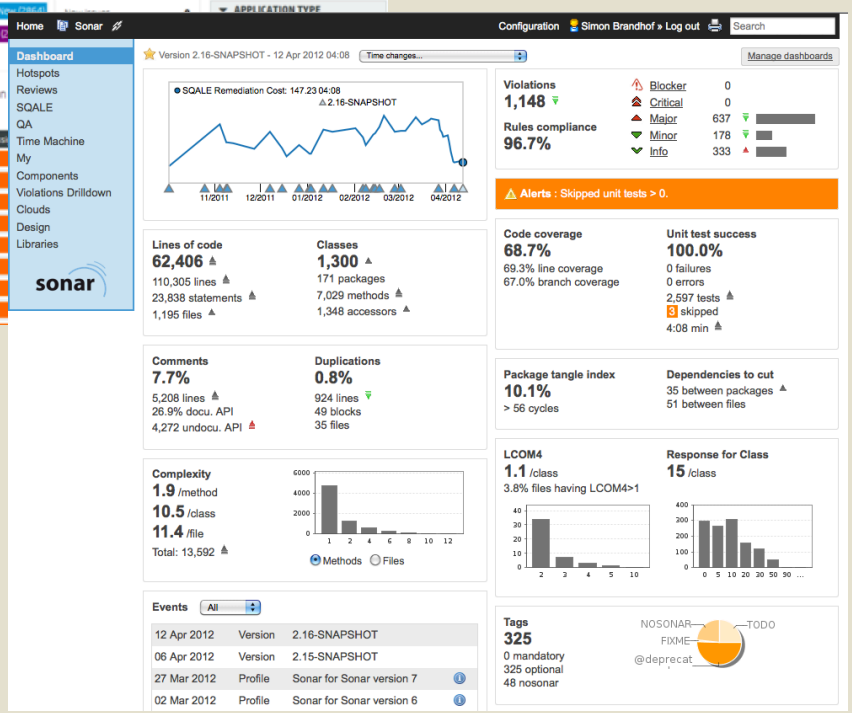






Asynchronous testing - ycle

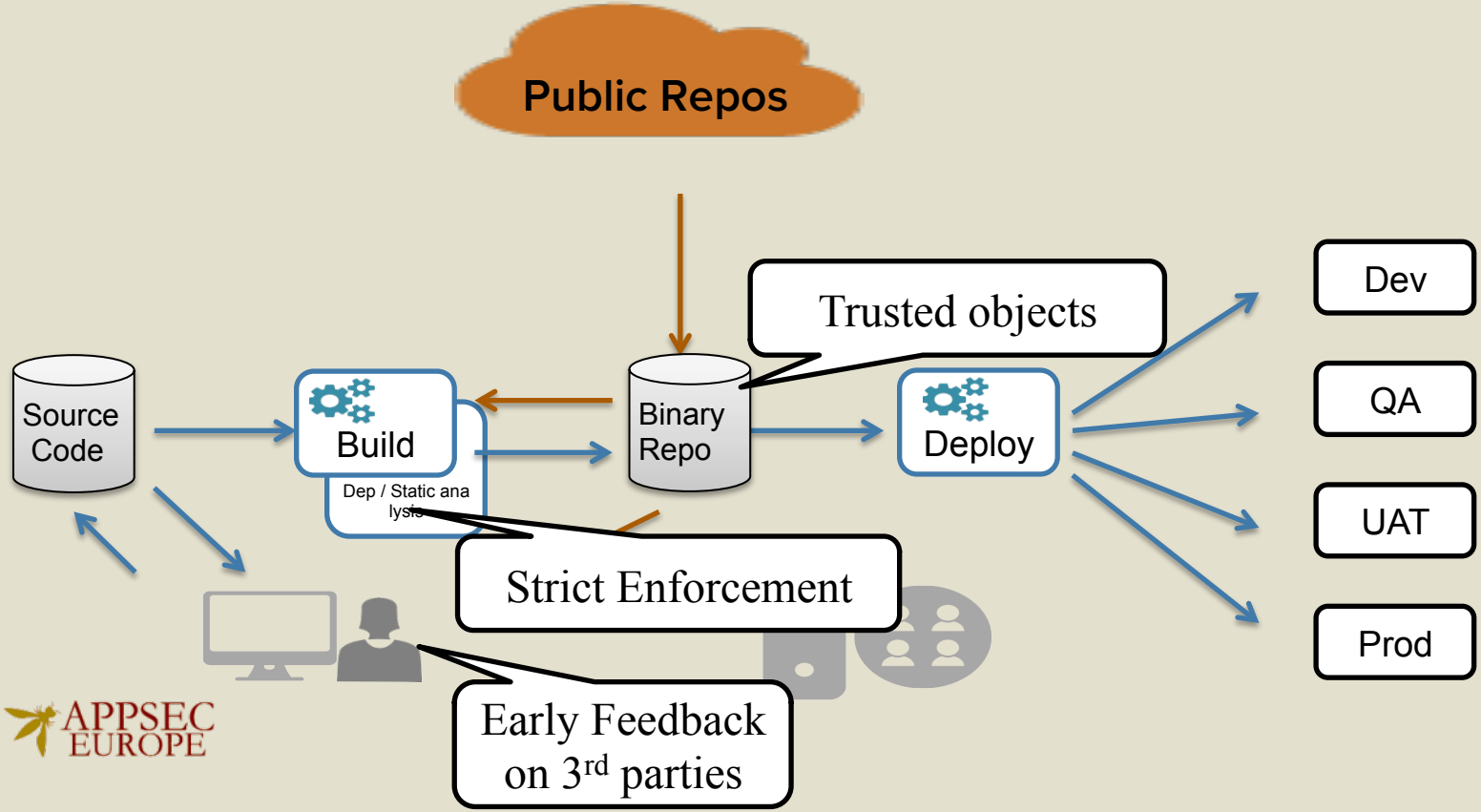


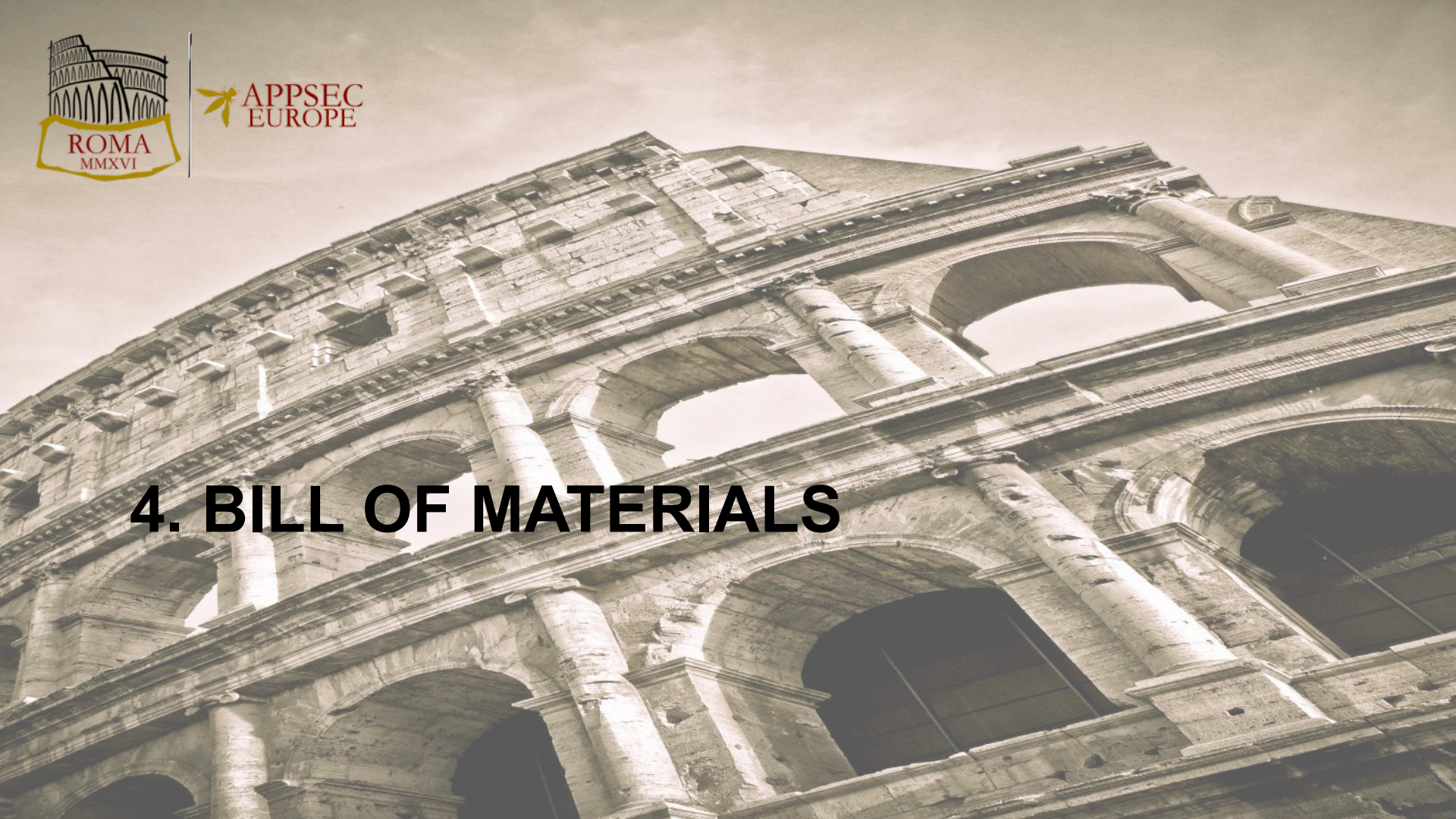


ROMA MMXVI

APPSEC EUROPE

Rugged Software Factory





4. BILL OF MATERIALS

Avoid reverse engineering

A **bill of materials** (**BoM**) is a list of the parts or components that are required to build a product. The **BoM** provides the manufacturer's part number (MPN) and the quantity needed for each component.

What is bill of materials (BoM)? - Definition from WhatIs.com
searchmanufacturingerp.techtarget.com/definition/**bill-of-materials-BoM**

More about Bill of materials

Feedback

Antipattern: Re: Re: Fwd: Re: Do we have this?

November 6, 2015

What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common? This Vulnerability.

By @breenmachine

What?

The most underrated, underhyped vulnerability of 2015 has recently come to my attention, and I'm about to bring it to yours. No one gave it a fancy name, there were no press

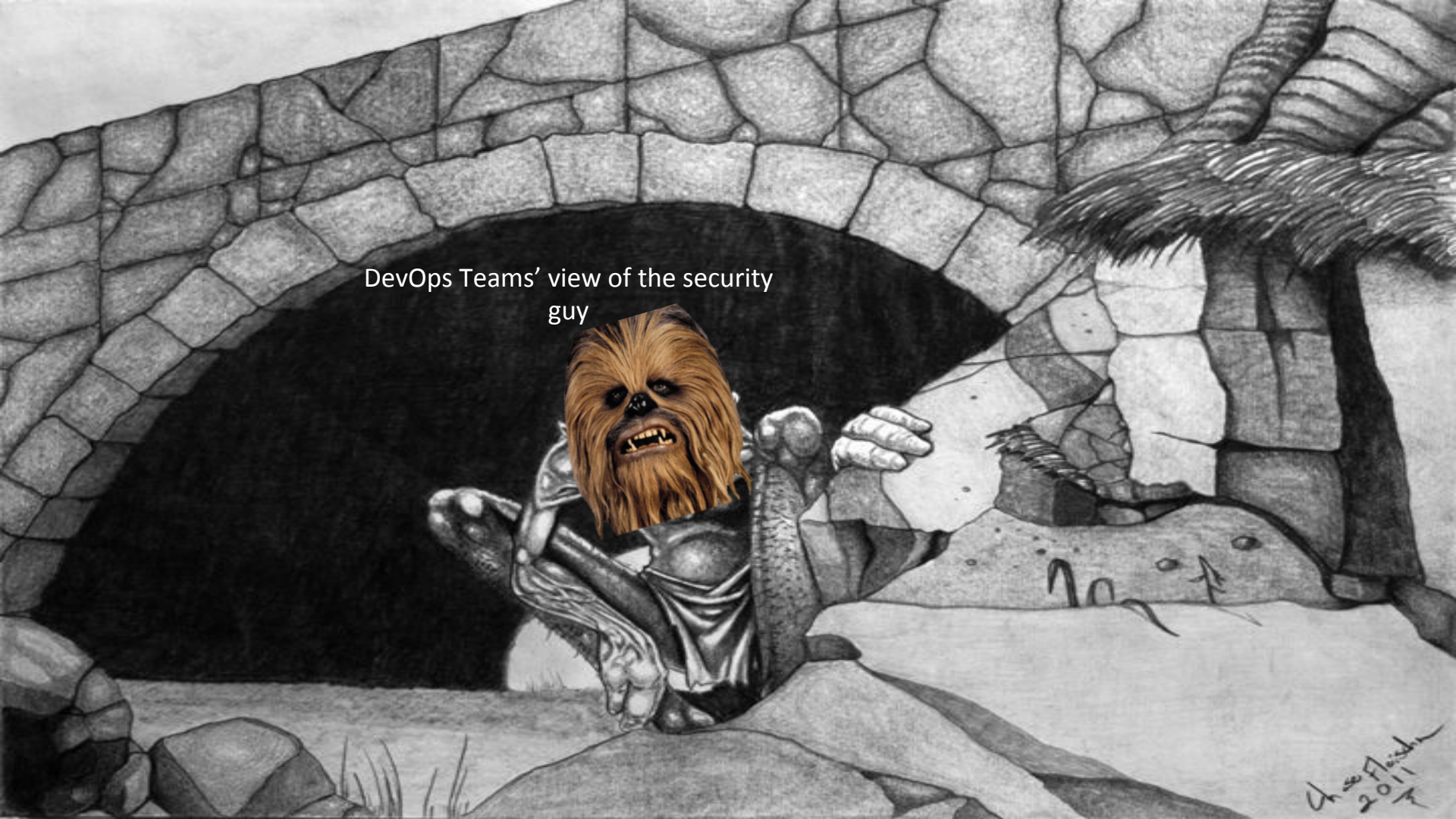
Policy Threat	Component	Popularity	Age	Release History
Search Name	Search Component			10 years
No Banned-deprecated	org.springframework : spring-context : 3.0.5.RELEASE		5.5 y	
	uk.ltd.getahead : dwr : 1.1.1		10.0 y	
Security-Critical	org.apache.struts : struts2-assembly : zip : all : 2.3.14		3.1 y	
	org.apache.struts : struts2-blank : war : 2.3.14		3.1 y	
	org.apache.struts : struts2-core : 2.3.14		3.1 y	
	org.apache.struts : struts2-mailreader : war : 2.3.14		3.1 y	
	org.apache.struts : struts2-portlet : war : 2.3.14		3.1 y	
	org.apache.struts : struts2-rest-plugin : 2.3.14		3.1 y	
	org.apache.struts : struts2-rest-showcase : war : 2.3.14		3.1 y	
	org.apache.struts : struts2-showcase : war : 2.3.14		3.1 y	
Security-High	commons-collections : commons-collections : 3.1		10.5 y	
	commons-fileupload : commons-fileupload : 1.2.2		5.8 y	
	org.apache.struts : struts-core : 1.3.10		7.4 y	
	org.apache.struts : struts2-assembly : zip : all : 2.3.14		3.1 y	
	org.apache.struts : struts2-blank : war : 2.3.14		3.1 y	
	org.apache.struts : struts2-core : 2.3.14		3.1 y	
	org.apache.struts : struts2-mailreader : war : 2.3.14		3.1 y	
	org.apache.struts : struts2-portlet : war : 2.3.14		3.1 y	
	org.apache.struts : struts2-rest-showcase : war : 2.3.14		3.1 y	
	org.apache.struts : struts2-showcase : war : 2.3.14		3.1 y	
	org.apache.struts.xwork : xwork-core : 2.3.14		3.1 y	
	org.springframework : spring-context : 3.0.5.RELEASE		5.5 y	

Manual searches or search API?

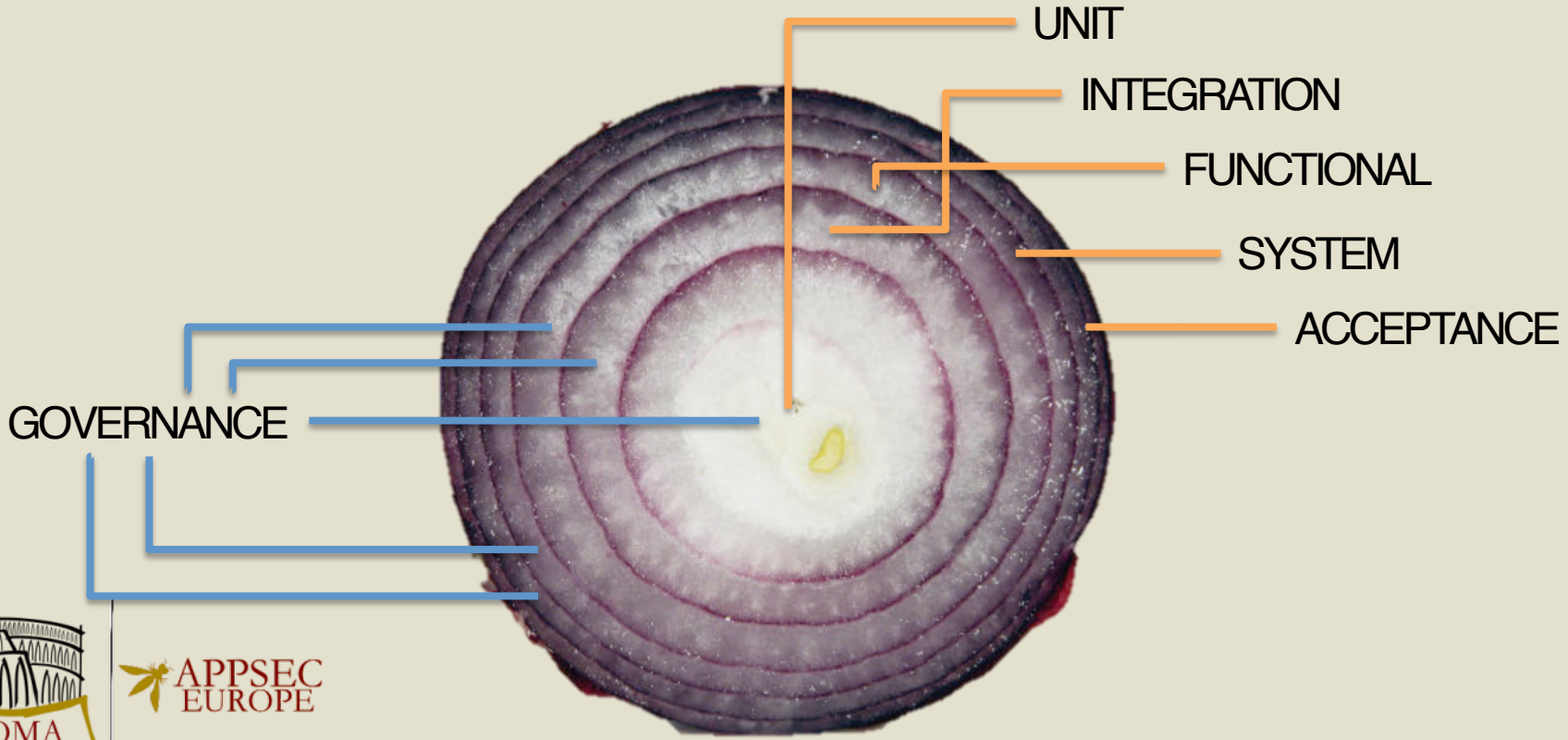

```
"results": [
   {
     "applicationId": "001",
     "applicationName": "Bank X Better Payment",
     "reportUrl": "http://localhost:8070/ui/links/application/001/report/cc8f94e42e3c4b6f93c86b35df9b648f",
     "hash": "40fb048097caeacdb11d",
     "componentIdentifier": {
       "format": "maven",
       "coordinates": {
         "artifactId": "commons-collections",
         "classifier": "",
         "extension": "jar",
         "groupId": "commons-collections",
         "version": "3.1"
       }
     },
     "threatLevel": 9
   },
   {
     "applicationId": "002",
     "applicationName": "Bank X build server",
     "reportUrl": "http://localhost:8070/ui/links/application/002/report/3c3b7ac5dc7344daa627248487a9662d",
   .....
•          },
```


APPSEC EUROPE

5. BUILD EXPERTISE AND INSTITUTE LEADERSHIP


DevOps Teams' view of the security guy

Build knowledge in teams in terms they understand



Security enabling, not blocking the process

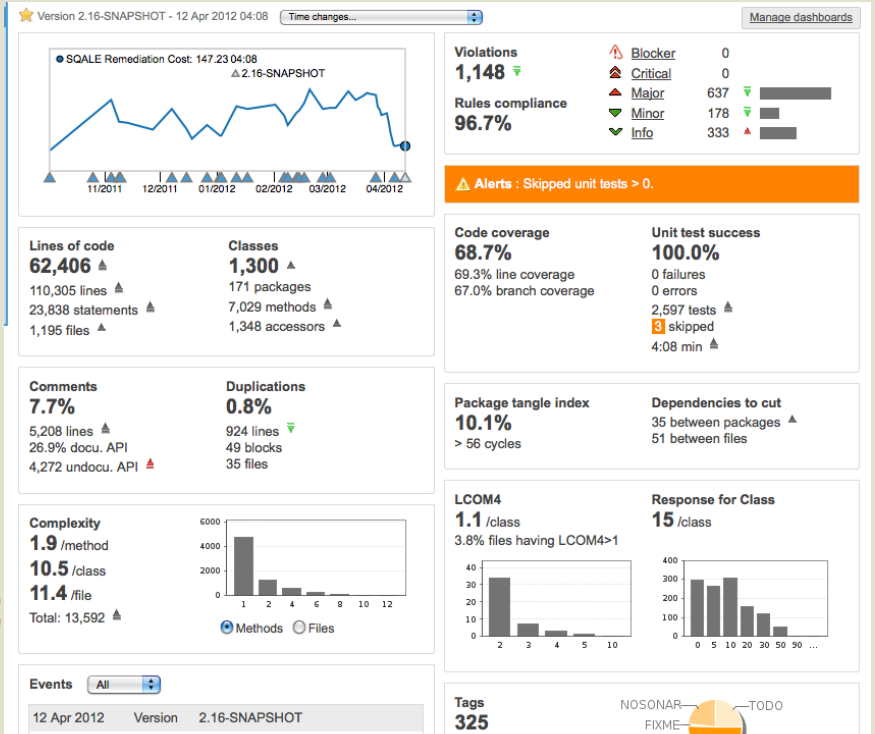
- Unit testing has become **TDD** (Test Driven Development)
- Usability testing has become **BDD** (Behaviour Driven Development)
- Integration testing has become **MDD** (Model Driven Development)

- Q.E.D Security testing needs to become **SDD** (Security Driven Development)

APPSEC EUROPE

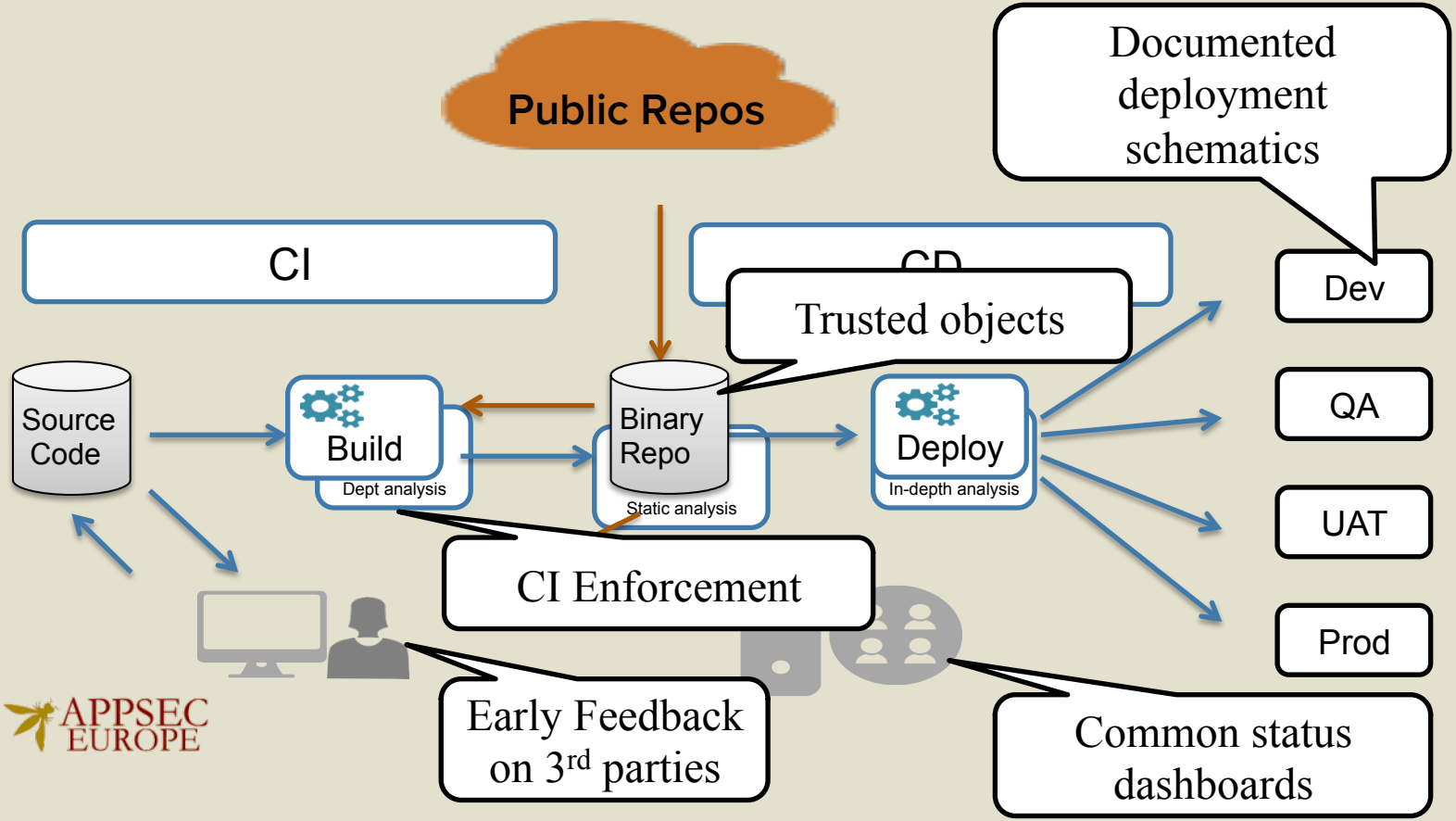
Be transparent with knowledge

- More eyes on data is better. Leverage Dashboards




ROMA
MMXVI
APPSEC
EUROPE
SO IN CONCLUSION

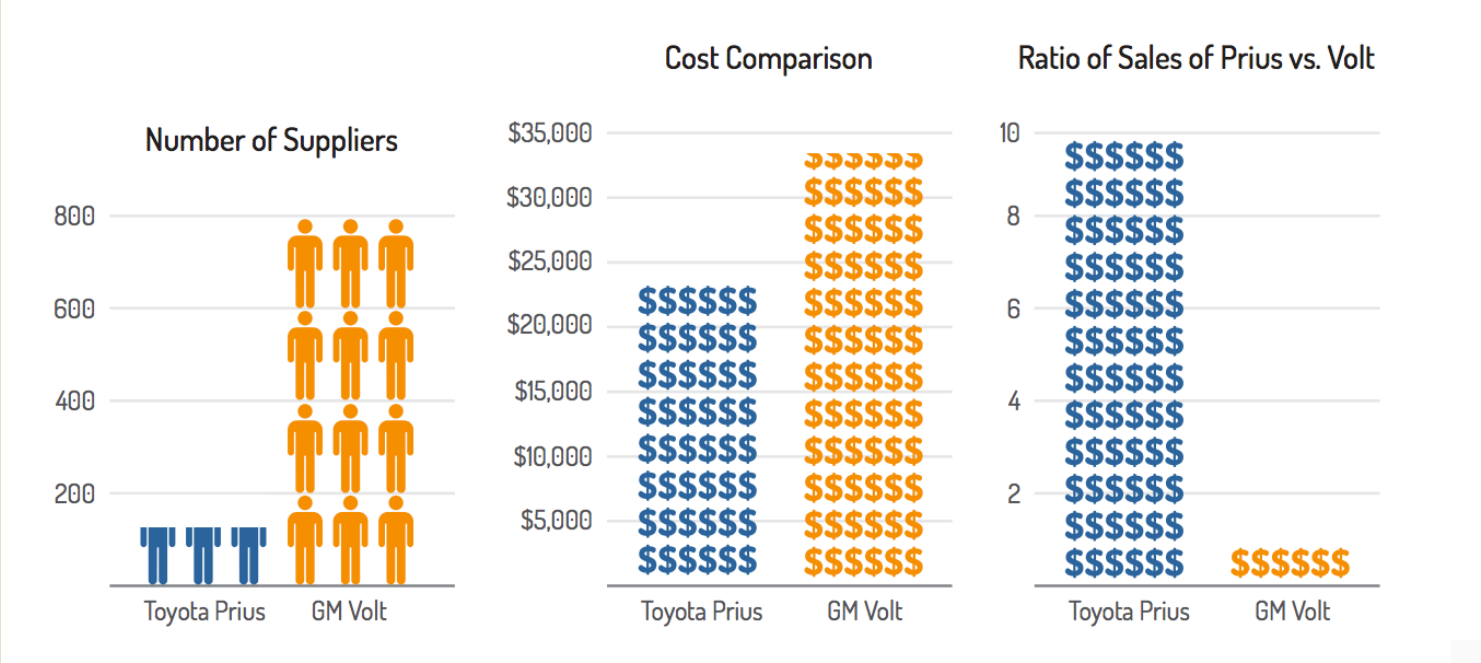
Rugged Software Factory



Translated into a Software Context

1. **Control the amount** and quality of suppliers **or components** you use
2. **Standardise your component catalog** as opposed to allowing every team to reinvent their toolkit
3. **Leverage automated quality controls and governance guidelines** as early as possible in the software life cycle to eliminate easily avoidable risk.
4. Maintain **a bill of materials** of all software and their underlying components
5. **Institute leadership** that can help improve the overall state of the component supply chain

Benefits seen in other industries



ROMA MMXVI

APPSEC EUROPE

Counter-measures

Situational Awareness

DevOps

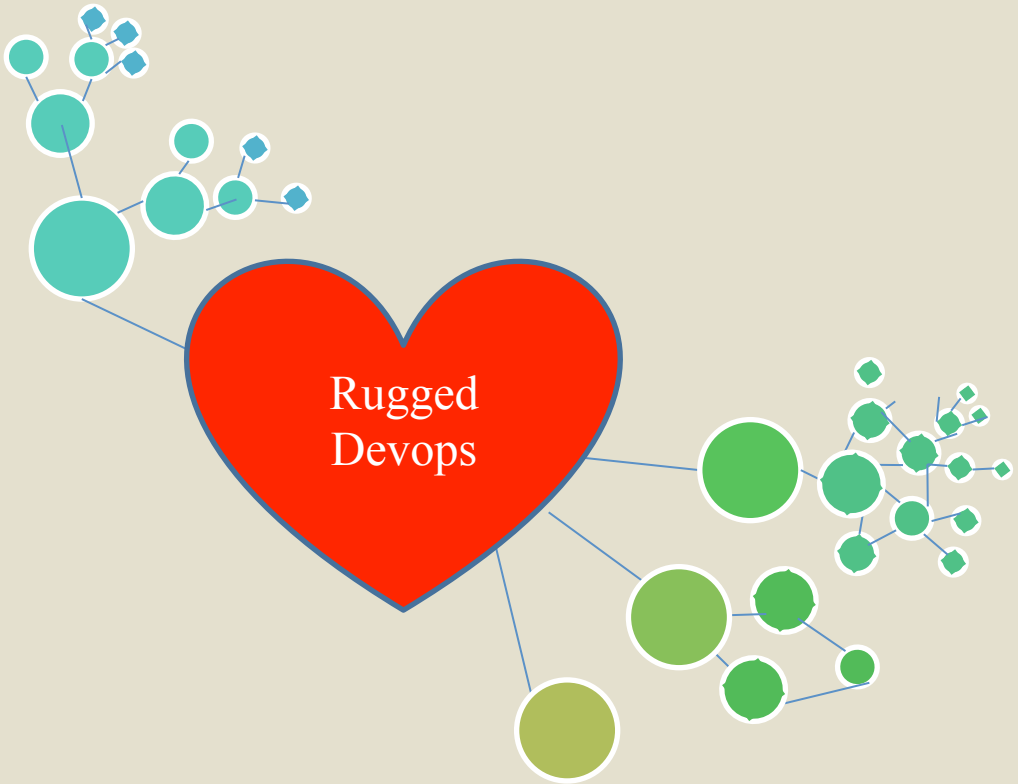
Operational Excellence

DevOps

Defensible Infrastructure

DevOps

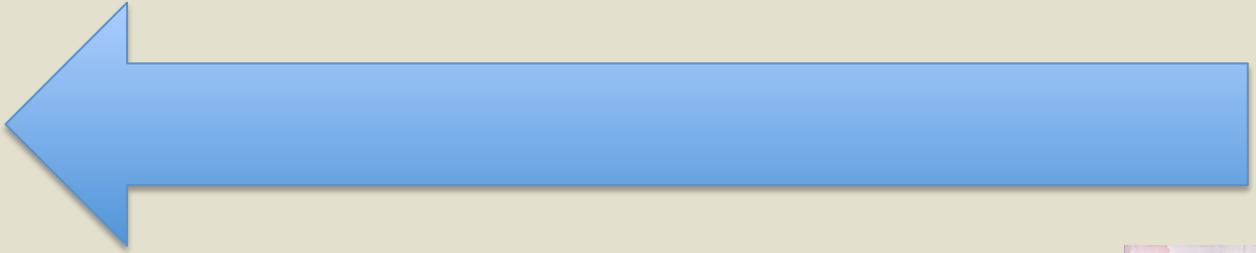
True collaboration via Transparency



ROMA MMXVI

APPSEC EUROPE

Shifting left



ROMA MMXVI

APPSEC EUROPE

Thanks - References

- **Wired Article – Hackers remotely kill Jeep on Highway:** https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/
- **State of Devops 2015:** https://puppetlabs.com/2015-devops-report
- **Rugged Devops Book:** http://devops.com/2015/04/20/the-rugged-devops-ebook/
- **Rugged Software:** http://www.ruggedsoftware.org/
- **DevSecOps:** http://devsecops.org
- **DevOpsSec – Securing Software Through Continuous Delivery by Jim Bird:** http://www.oreilly.com/webops-perf/free/devopssec.csp
- **"The Phoenix Project" by Gene Kim:** http://itrevolution.com/books/phoenix-project-devops-book/
- **State of Software Supply Chain 2015:** https://www.sonatype.com/state-of-the-software-supply-chain
- **7 Habits of Rugged Devops:** https://www.forrester.com/report/The+Seven+Habits+Of+Rugged+DevOps/-/E-RES126542
- **Verizon Data Breach Report:** http://www.verizonenterprise.com/verizon-insights-lab/dbir/2016/
- **CodeCentric CI Example:** https://blog.codecentric.de/en/2015/10/continuous-integration-platform-using-docker-container-jenkins-sonarqube-nexus-gitlab/
- **FS-ISAC:** https://www.sonatype.com/software-security-control-white-paper
- **IEC-62304:** http://www.iso.org/iso/catalogue_detail.htm?csnumber=38421
- **PCI-DSS:** https://www.pcisecuritystandards.org/document_library?category=pcidss&document=pci_dss
- **Reflections on NPMGate:** http://blog.npmjs.org/post/141577284765/kik-left-pad-and-npm
- **Lessons learnt again from NPMGate:** http://www.sonatype.org/nexus/2016/03/25/npm-gate-lessons-learned-again/

Tweet: **@ilkkaturunen**

ROMA MMXVI

APPSEC EUROPE