WhiteSource

How to Sleep Soundly at Night While Using Open Source

Guy Bar Gil, Product Manager

1

The Equifax Breach

2

SmallComp's M&A

3

Staying Secure

Introduction Slide

- Two dogs + one cat
- In my free time I enjoy:
  - Sports
  - Reading
  - Traveling

Guy Bar Gil, Product Manager

1

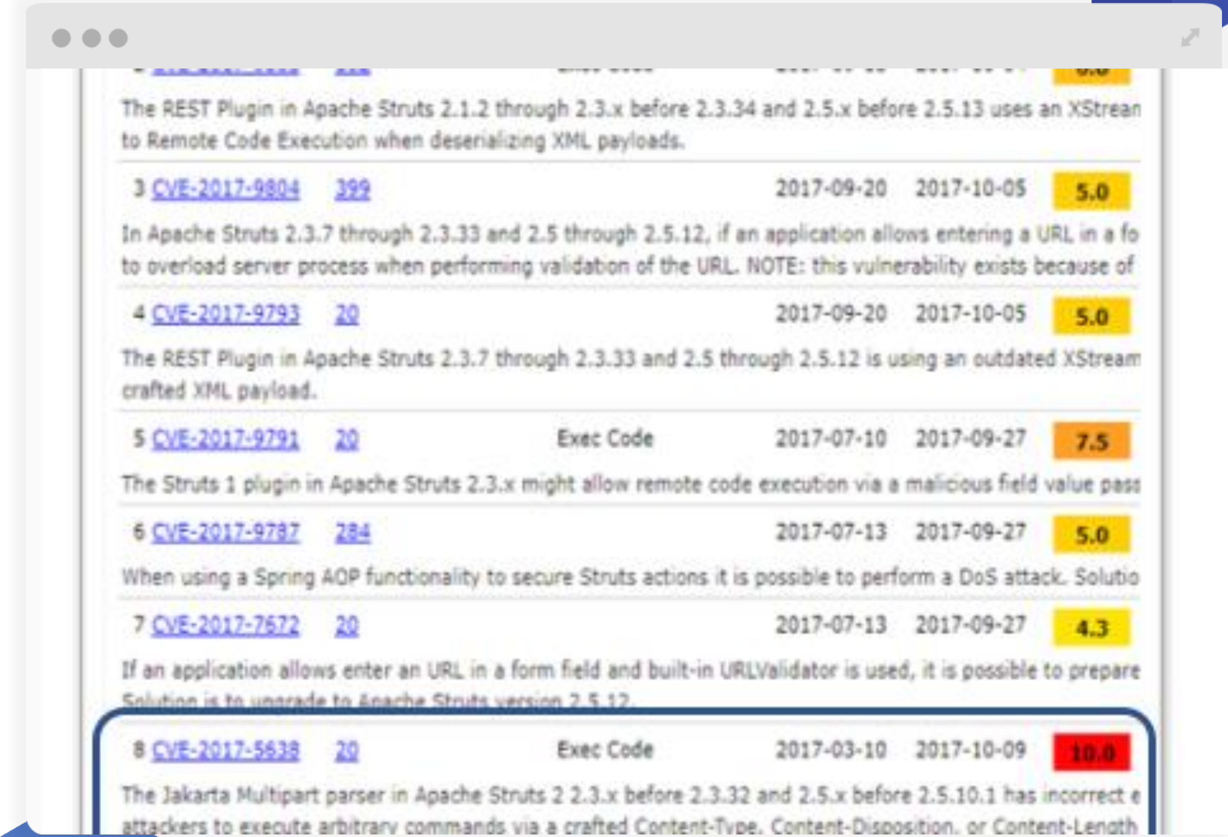
The Equifax Breach

EQUIFAX

- Consumer credit reporting agency
- Equifax collects and aggregates information
  - 800M+ individual consumers
  - 88M+ businesses
- Publicly traded (NYSE), 9.5K+ employees, $3.14B in revenues (2016)

Apache Struts

- **Open-source** framework, for creating Java web applications.
- CVE-2017-5638 allows remote code execution through the web applications.



The REST Plugin in Apache Struts 2.1.2 through 2.3.x before 2.3.34 and 2.5.x before 2.5.13 uses an XStrean
to Remote Code Execution when deserializing XML payloads.

3 CVE-2017-9804 399 2017-09-20 2017-10-05 5.0

In Apache Struts 2.3.7 through 2.3.33 and 2.5 through 2.5.12, if an application allows entering a URL in a fo
to overload server process when performing validation of the URL. NOTE: this vulnerability exists because of

4 CVE-2017-9793 20 2017-09-20 2017-10-05 5.0

The REST Plugin in Apache Struts 2.3.7 through 2.3.33 and 2.5 through 2.5.12 is using an outdated XStream
crafted XML payload.

5 CVE-2017-9791 20 Exec Code 2017-07-10 2017-09-27 7.5

The Struts 1 plugin in Apache Struts 2.3.x might allow remote code execution via a malicious field value pass

6 CVE-2017-9787 284 2017-07-13 2017-09-27 5.0

When using a Spring AOP functionality to secure Struts actions it is possible to perform a DoS attack. Solutio

7 CVE-2017-7672 20 2017-07-13 2017-09-27 4.3

If an application allows enter an URL in a form field and built-in URLValidator is used, it is possible to prepare
Solution is to upgrade to Apache Struts version 2.5.12.

8 CVE-2017-5638 20 Exec Code 2017-03-10 2017-10-09 10.0

The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1 has incorrect e
attackers to execute arbitrary commands via a crafted Content-Type, Content-Disposition, or Content-Length

The Equifax Breach – A Timeline

March 9th

- CVE-2017-5638 was published.
- The Apache Software Foundation released a patch for the vulnerability.
- Equifax administrators were told to apply the patch to any affected systems.

March 10th

Hackers gained access to Equifax's systems.

March 15th

Equifax's IT department ran a series of scans to identify unpatched systems.

May 13th

Hackers begin exfiltrating data.

July 29th

Equifax renews their expired public-key certificate

September 7th

Equifax publicized the breach

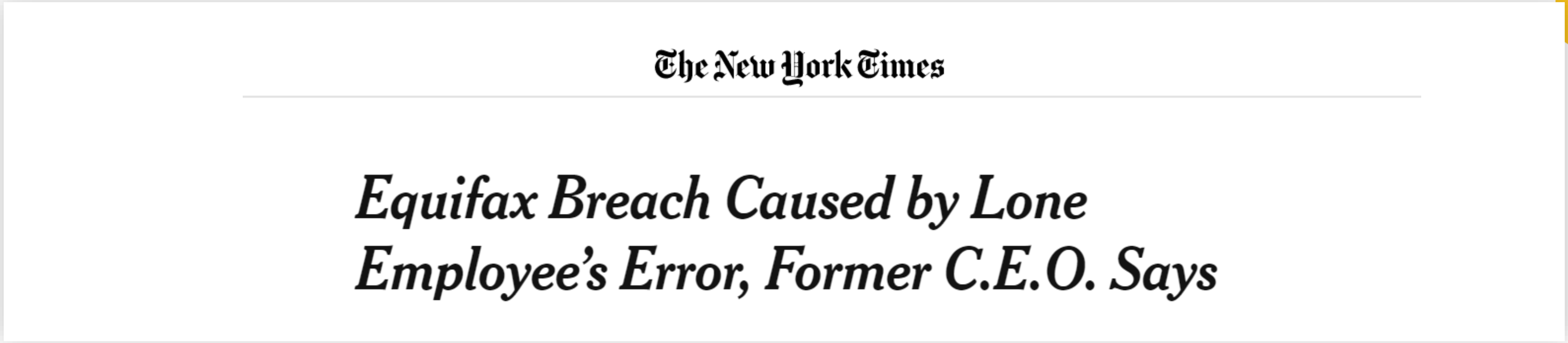
Incident Aftermath

145 million

People affected by the breach.

$3 billion

The amount Equifax spent upgrading its security and resolving consumer claims



The New York Times

Equifax Breach Caused by Lone Employee's Error, Former C.E.O. Says

What Can We Learn?

#1

Act Fast

Exploits are public for everyone

#2

Continuously Monitor

300 new vulnerabilities published every month

#3

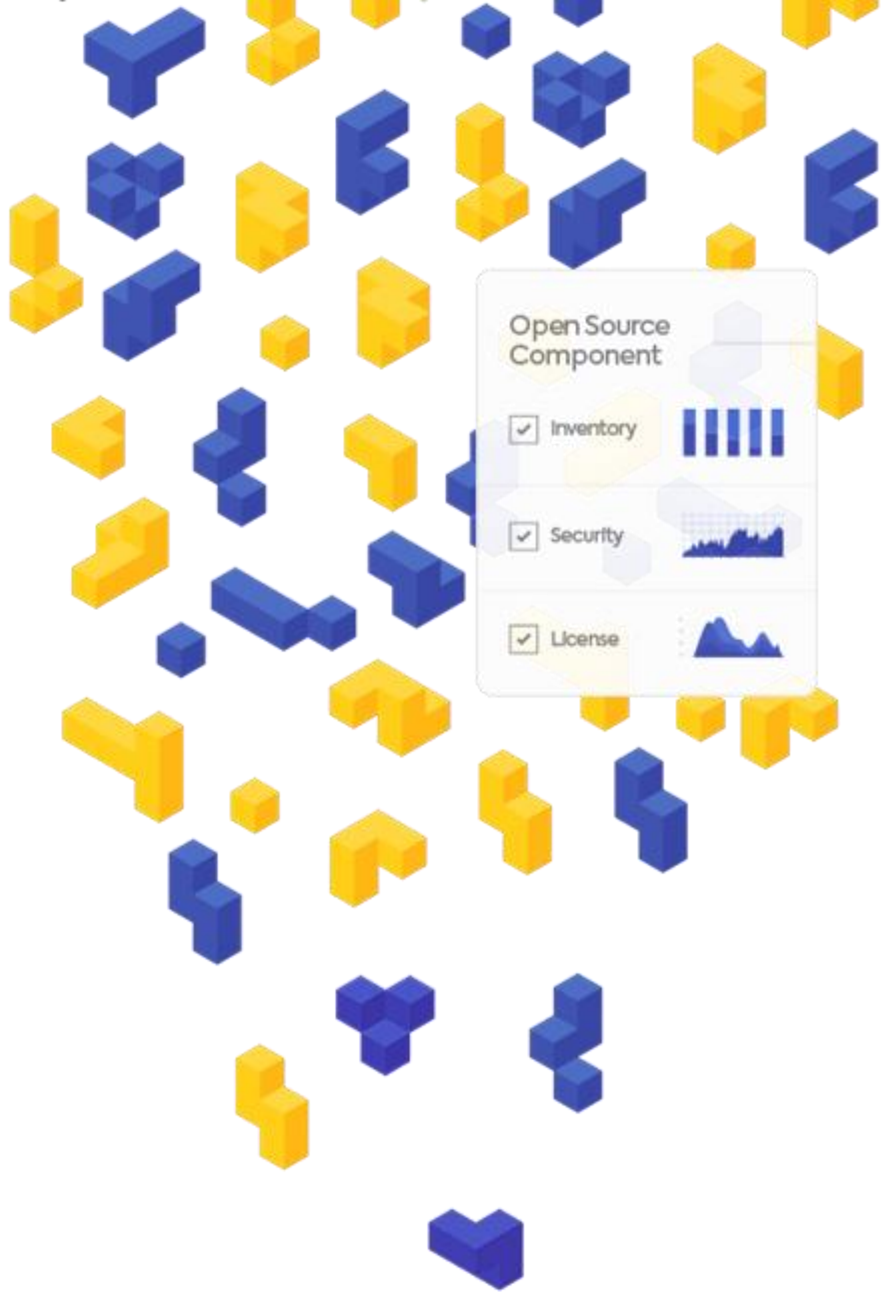
Get the Basics Right

Millions spent on security gear but it was poorly implemented

WhiteSource

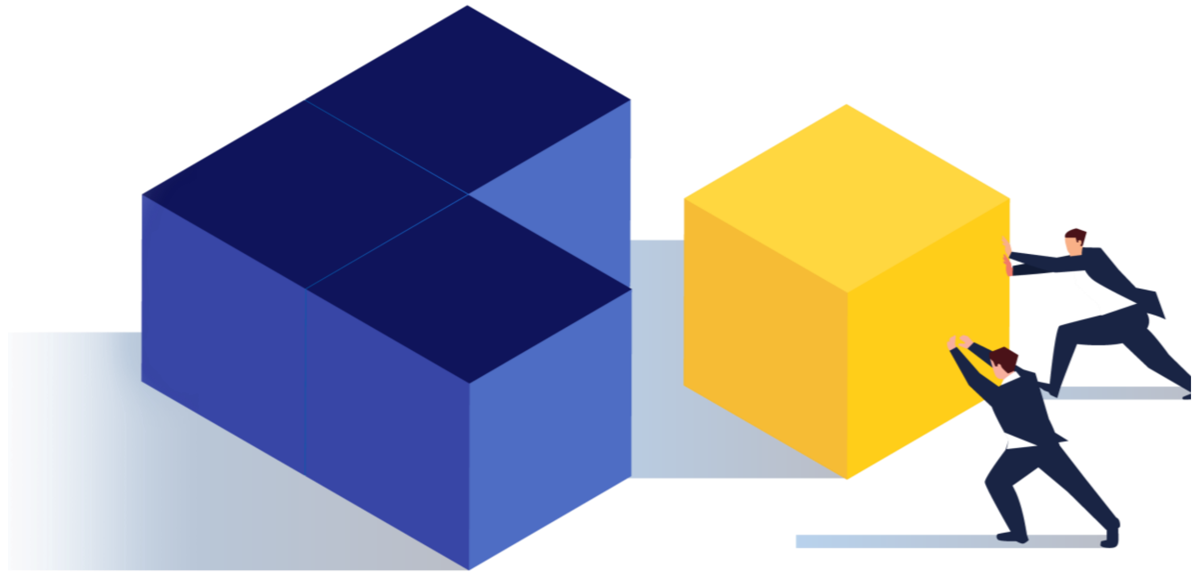
2

SmallComp's M&A



SmallComp’s M&A

- SmallComp required to do an open-source audit.
- Found a dependency licensed under AGPL.



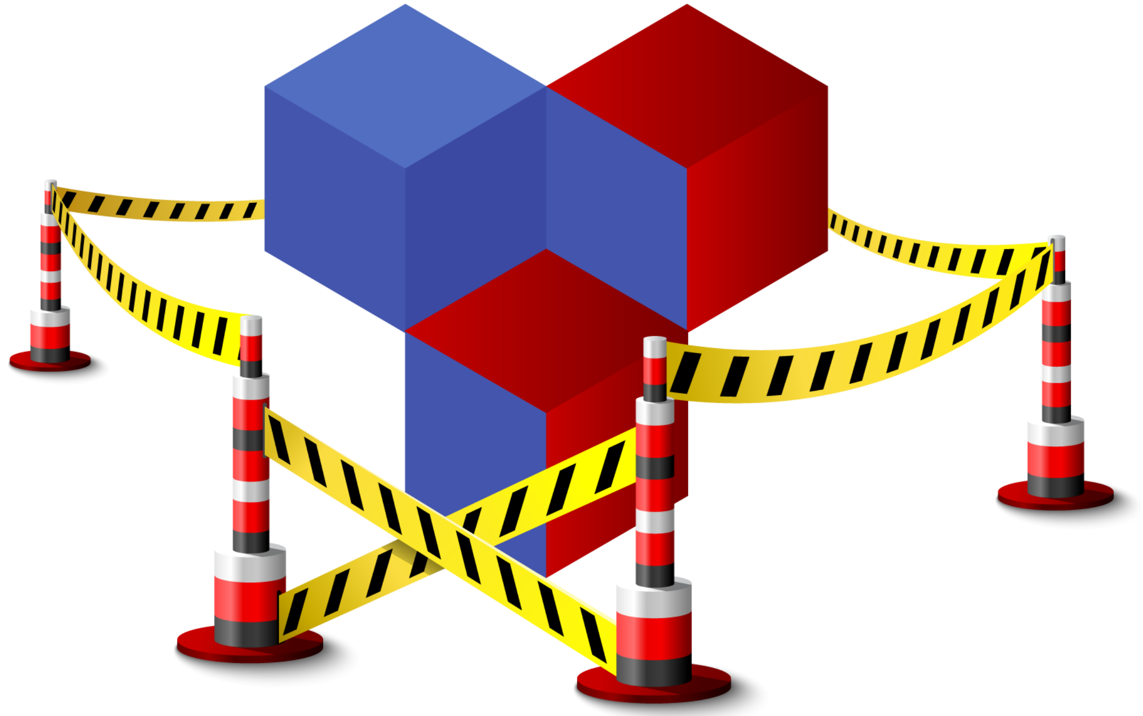
$4M Dollars in Escrow

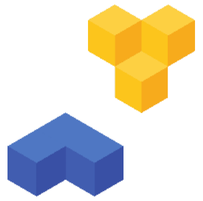
- Terms of the escrow:
  - Remove any trace of AGPL from the software.
  - 80% of customers must deploy the updated software to production.
  - Two year timeframe.
  - Development/deployment-related costs taken from the escrow.

Solving The Problem Isn't So Easy

Two main obstacles:

- Development + QA time is 1 year for 1 person.
- SmallComp's customers are hospitals, where solutions are often manually deployed and technicians are required to train staff.





They Did It!

SmallComp was able to fulfill the terms of the escrow

after 1 year and 8 months!

How Do We Avoid This Situation?

#1 **Set Clear Policies**

for the whole company in regards to licensing

#2 **Communicate**

the company’s policies to developers

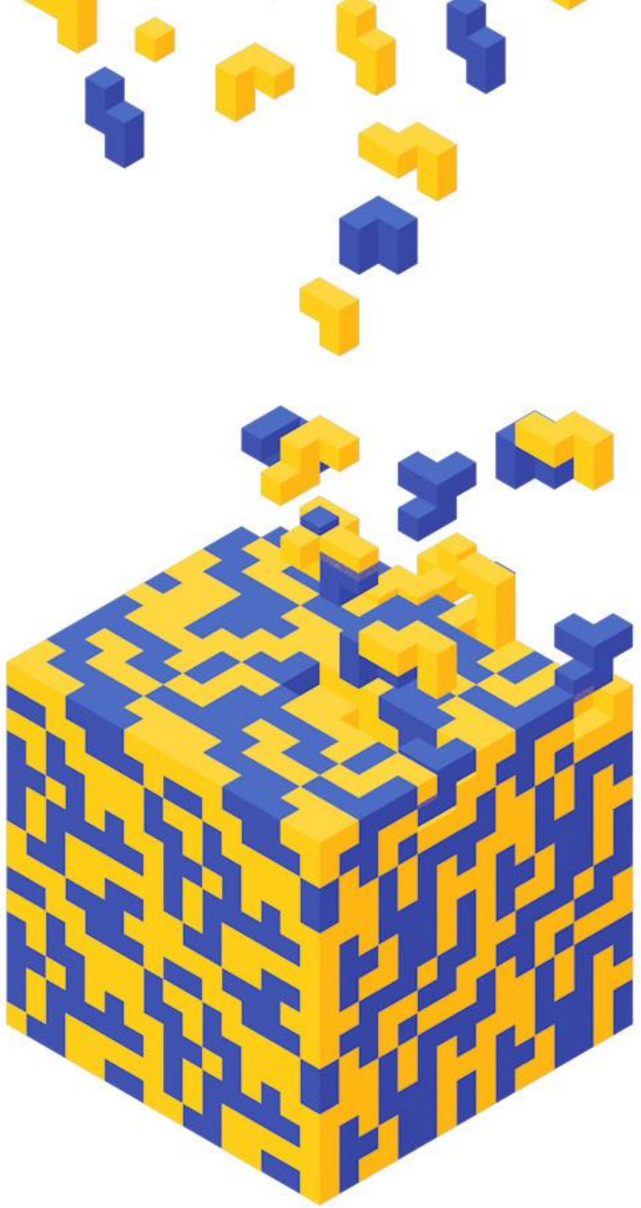
#3 **Enforce**

make sure your policies are being enforced

WhiteSource

3

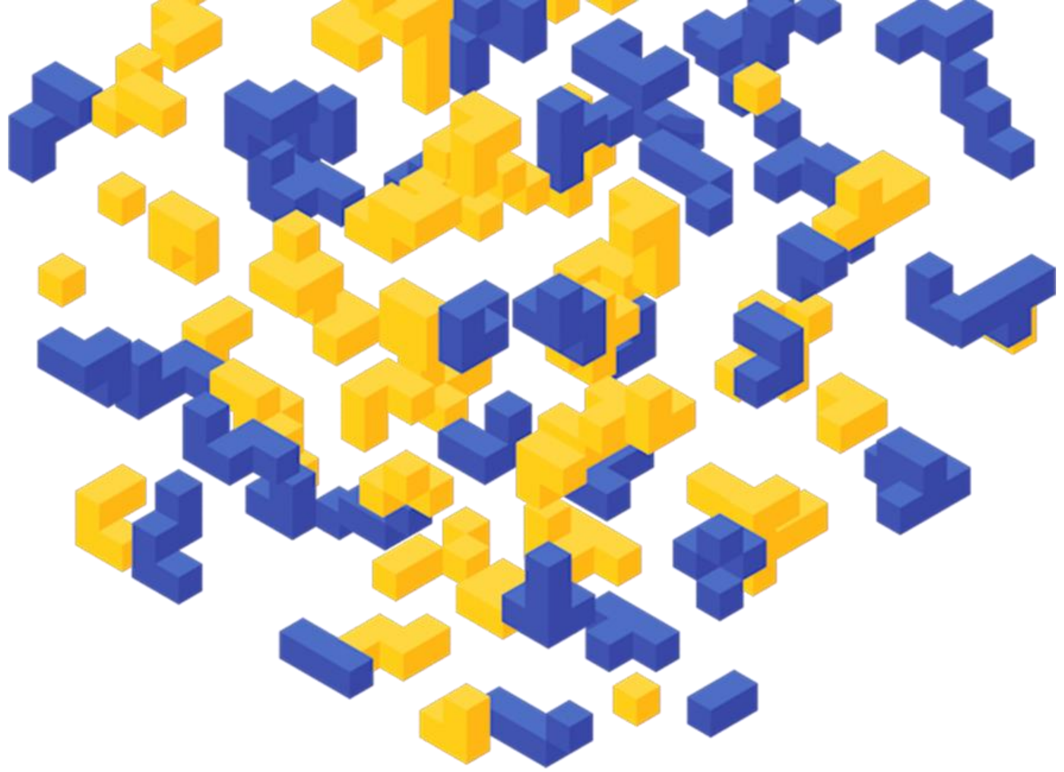
Staying Secure



Step 1:

Create Transparency

- Transparency is the baseline to everything
- Understand exactly what you’re using:
    - Direct Dependencies
    - Transitive Dependencies
    - Source files



WhiteSource

Lots of jars

?

Lots of jars, but lots more java beans

?

Step 2:
Detect Potential Issues

- Match your components to the most comprehensive DB possible:
  - Published CVEs
  - Vulnerabilities published in security advisories
  - Vulnerabilities detected by research teams
  - Thorough license detection

WhiteSource

Step 3: Prioritize

How would you prioritize your vulnerabilities?

WhiteSource

Step 3: Prioritize

- Prioritize by:
  - Business risk
  - Exploitability
  - Severity
  - Availability of fixes
  - Effectiveness

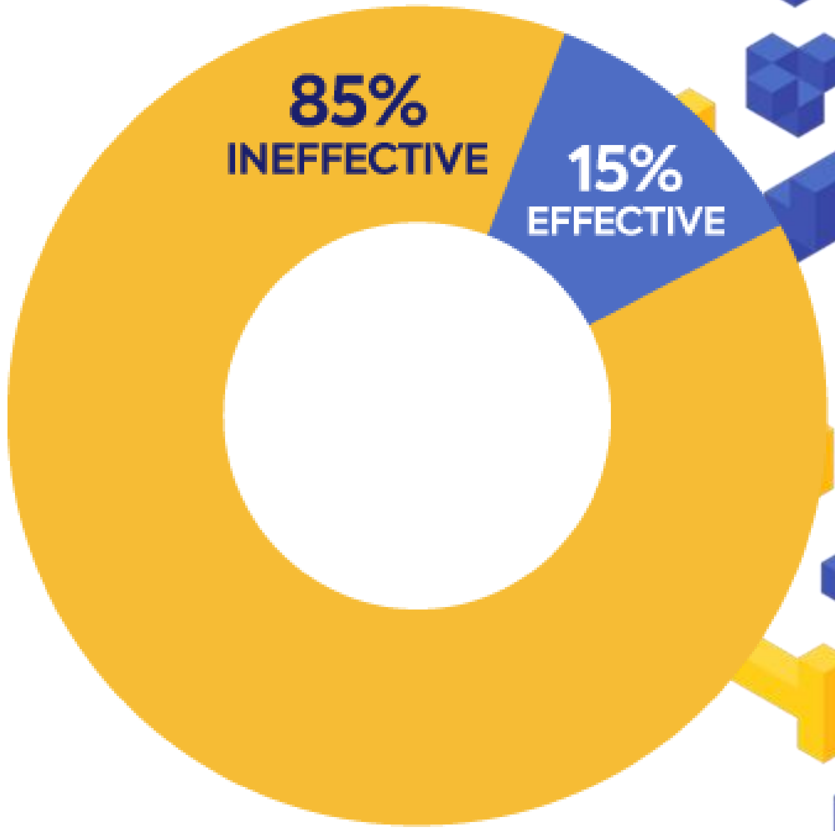
Vulnerabilities Prioritization

EFFECTIVE VS INEFFECTIVE
VULNERABILITIES IN A COMPONENT

EFFECTIVE VULNERABILITY
If the proprietary code is making calls to the vulnerable functionality

INEFFECTIVE VULNERABILITY
If the proprietary code is NOT making calls to the vulnerable functionality

After testing 2,000 Java applications, WhiteSource found **that 85% of all detected vulnerabilities were deemed ineffective.**

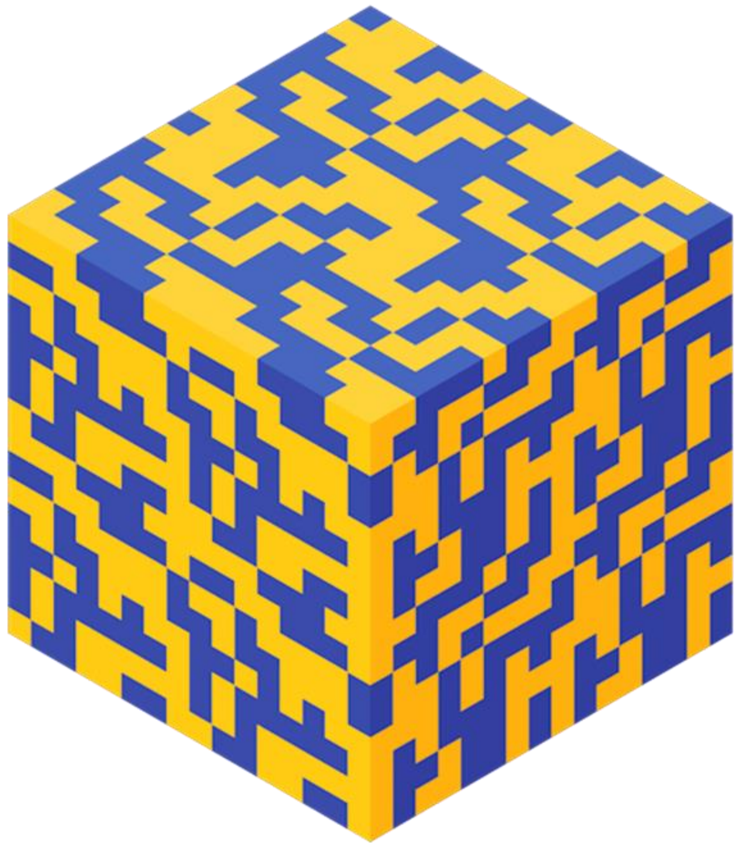


Step 4:

Execution

Understand the best path to remediation

- Upgrade the component’s version?
- Change the component?
- Set up an external defense?



WhiteSource

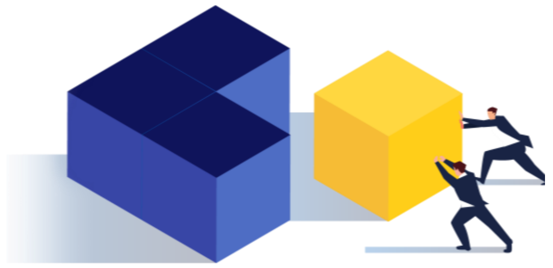
1
Create Transparency

2
Detect Issues

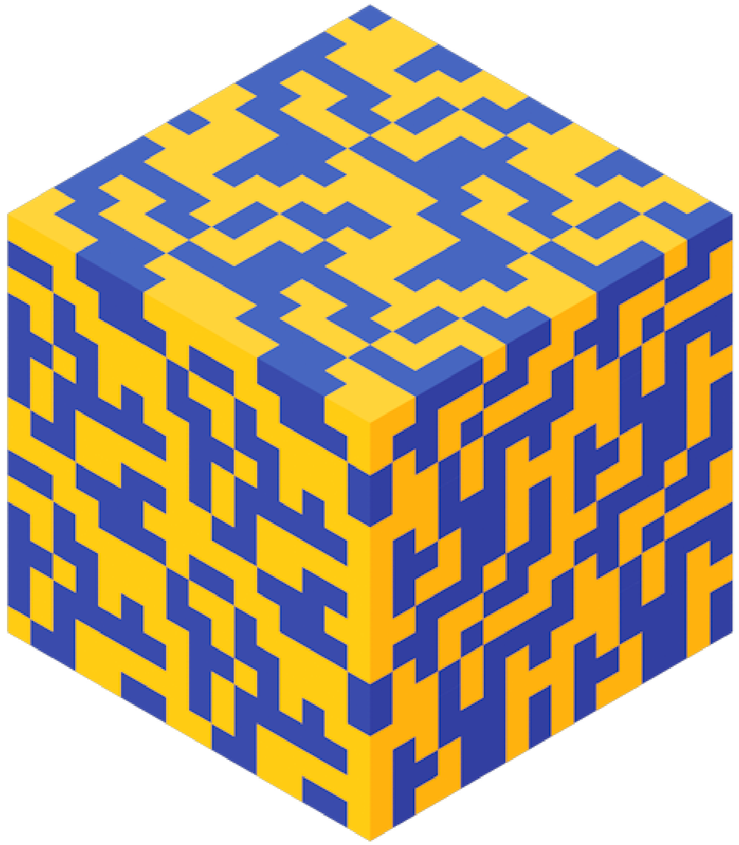
3
Prioritize

EFFECTIVE vs INEFFECTIVE
VULNERABILITIES IN A COMPONENT
EFFECTIVE VULNERABILITY
If the proprietary code is making calls to the vulnerable functionality
INEFFECTIVE VULNERABILITY
If the proprietary code is NOT making calls to the vulnerable functionality

4
Execute

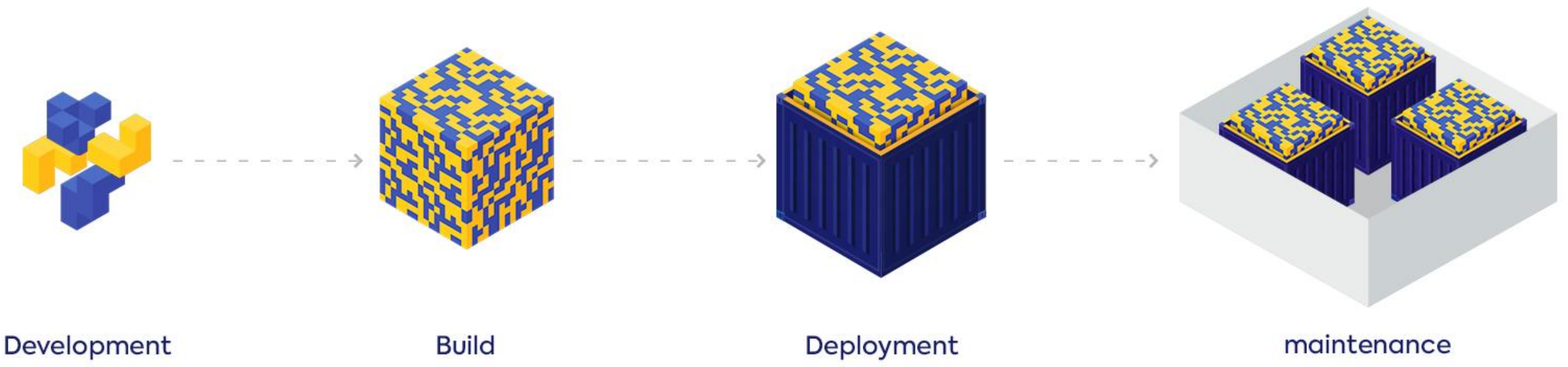


Let’s Talk About Implementation!



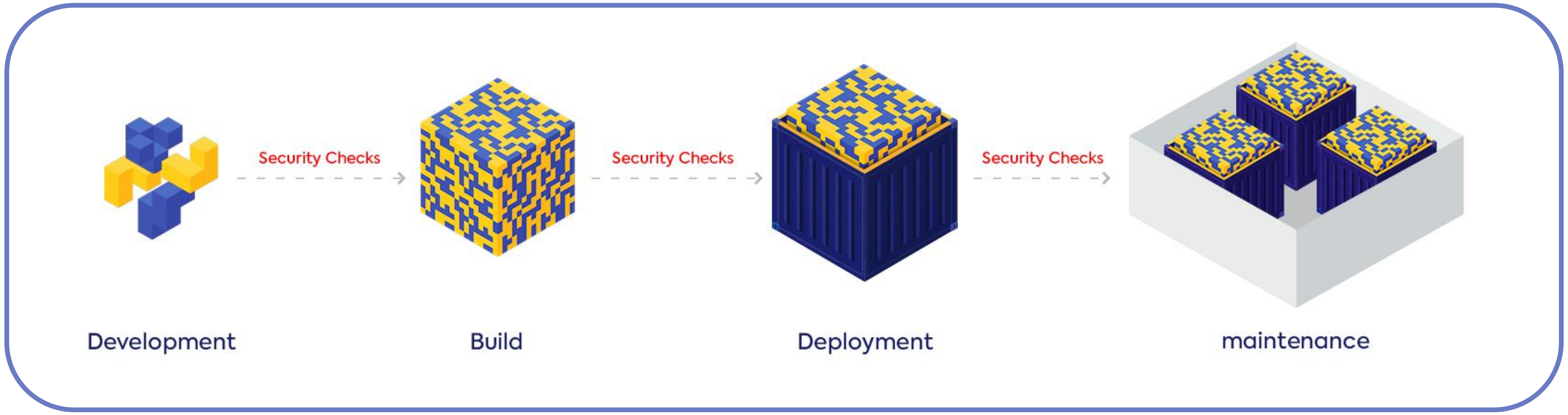
Step 1 : Creating Transparency

Identify the processes in your software development lifecycle (SDLC)

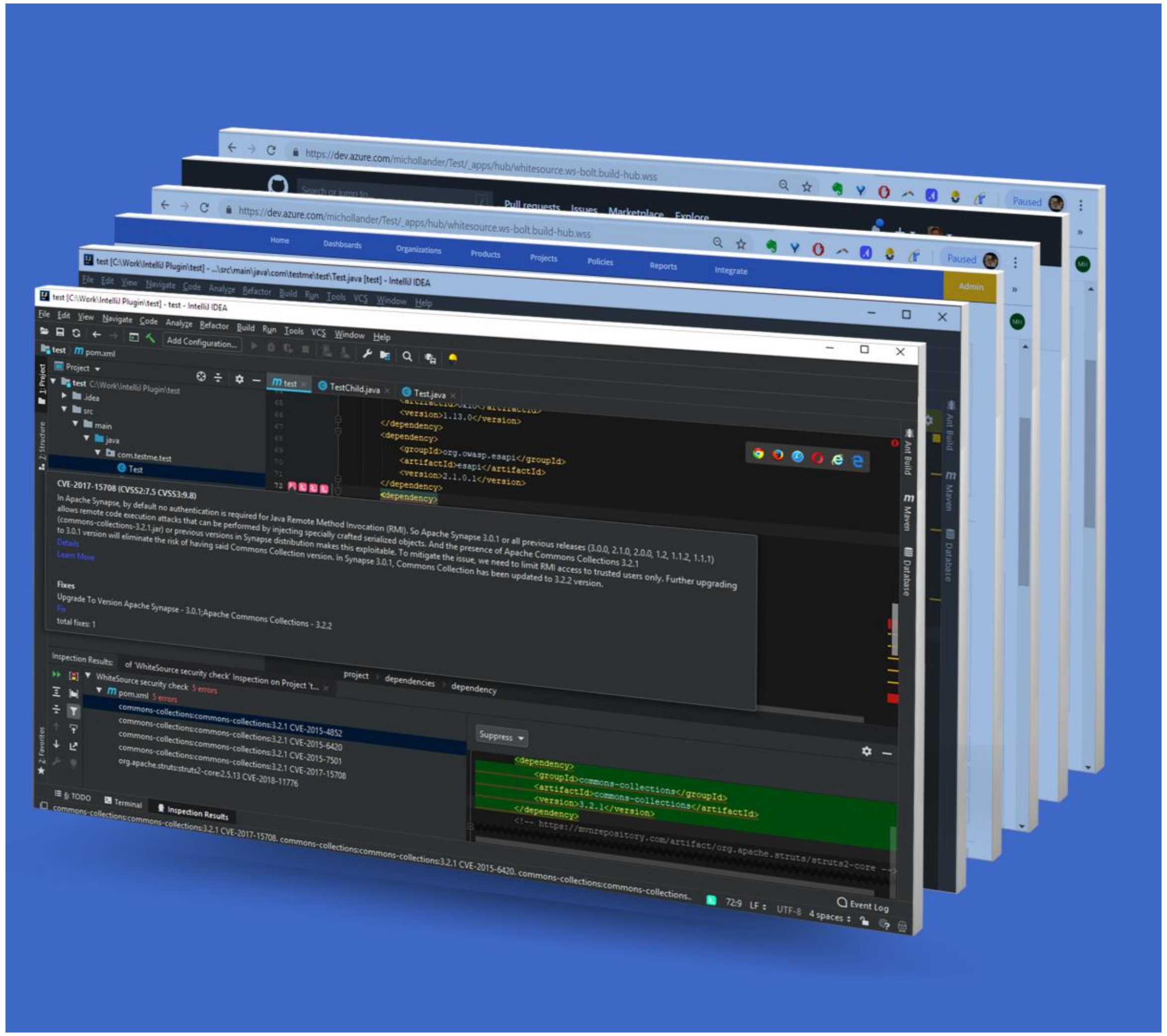


Step 2: Detect

- Where you want to implement security checks
- Where you can automate



Development



```
<version>1.13.0</version>
</dependency>
<dependency>
    <groupId>org.owasp.esapi</groupId>
    <artifactId>esapi</artifactId>
    <version>2.1.0.1</version>
</dependency>
<dependency>
```

CVE-2017-15708 (CVSS2:7.5 CVSS3:9.8)

In Apache Synapse, by default no authentication is required for Java Remote Method Invocation (RMI). So Apache Synapse 3.0.1 or all previous releases (3.0.0, 2.1.0, 2.0.0, 1.2, 1.1.2, 1.1.1) allows remote code execution attacks that can be performed by injecting specially crafted serialized objects. And the presence of Apache Commons Collections 3.2.1 (commons-collections-3.2.1.jar) or previous versions in Synapse distribution makes this exploitable. To mitigate the issue, we need to limit RMI access to trusted users only. Further upgrading to 3.0.1 version will eliminate the risk of having said Commons Collection version. In Synapse 3.0.1, Commons Collection has been updated to 3.2.2 version.

Details

Learn More

Fixes

Upgrade To Version Apache Synapse - 3.0.1;Apache Commons Collections - 3.2.2

Fix

total fixes: 1

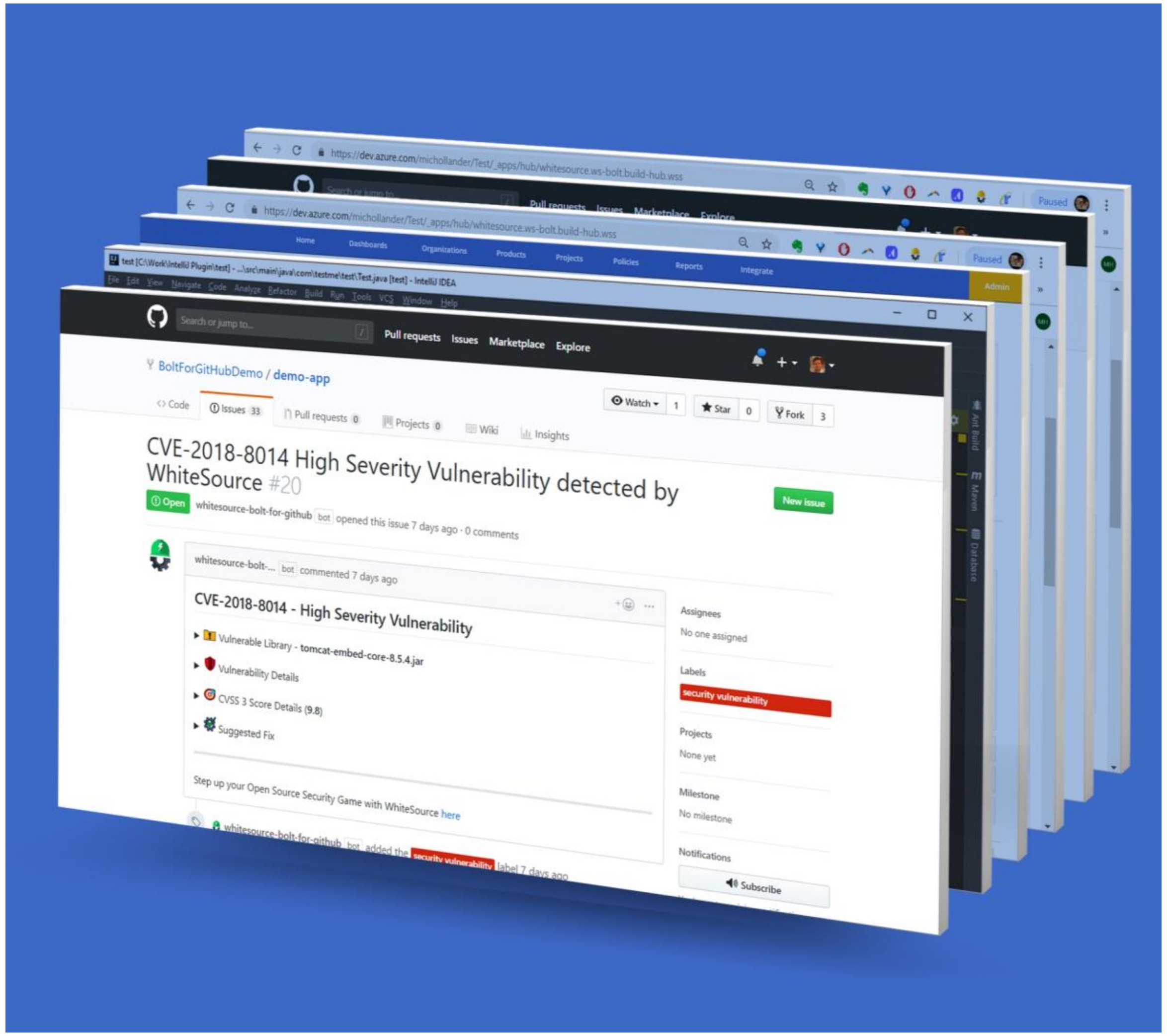
```
Inspection Results: of 'WhiteSource security check' Inspection on Project 't...
▼ WhiteSource security check 5 errors
  ▼ pom.xml 5 errors
      commons-collections:commons-collections:3.2.1 CVE-2015-4852
      commons-collections:commons-collections:3.2.1 CVE-2015-6420
      commons-collections:commons-collections:3.2.1 CVE-2015-7501
      commons-collections:commons-collections:3.2.1 CVE-2017-15708
      org.apache.struts:struts2-core:2.5.13 CVE-2018-11776
```

```
<dependency>
    <groupId>commons-collections</groupId>
    <artifactId>commons-collections</artifactId>
    <version>3.2.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.struts/struts2-core -->
```

```
commons-collections:commons-collections:3.2.1 CVE-2017-15708. commons-collections:commons-collections:3.2.1 CVE-2015-6420. commons-collections:commons-collections...
```

WhiteSource

Build



CVE-2018-8014 High Severity Vulnerability detected by WhiteSource #20

whitesource-bolt-for-github bot opened this issue 7 days ago · 0 comments

whitesource-bolt-... bot commented 7 days ago

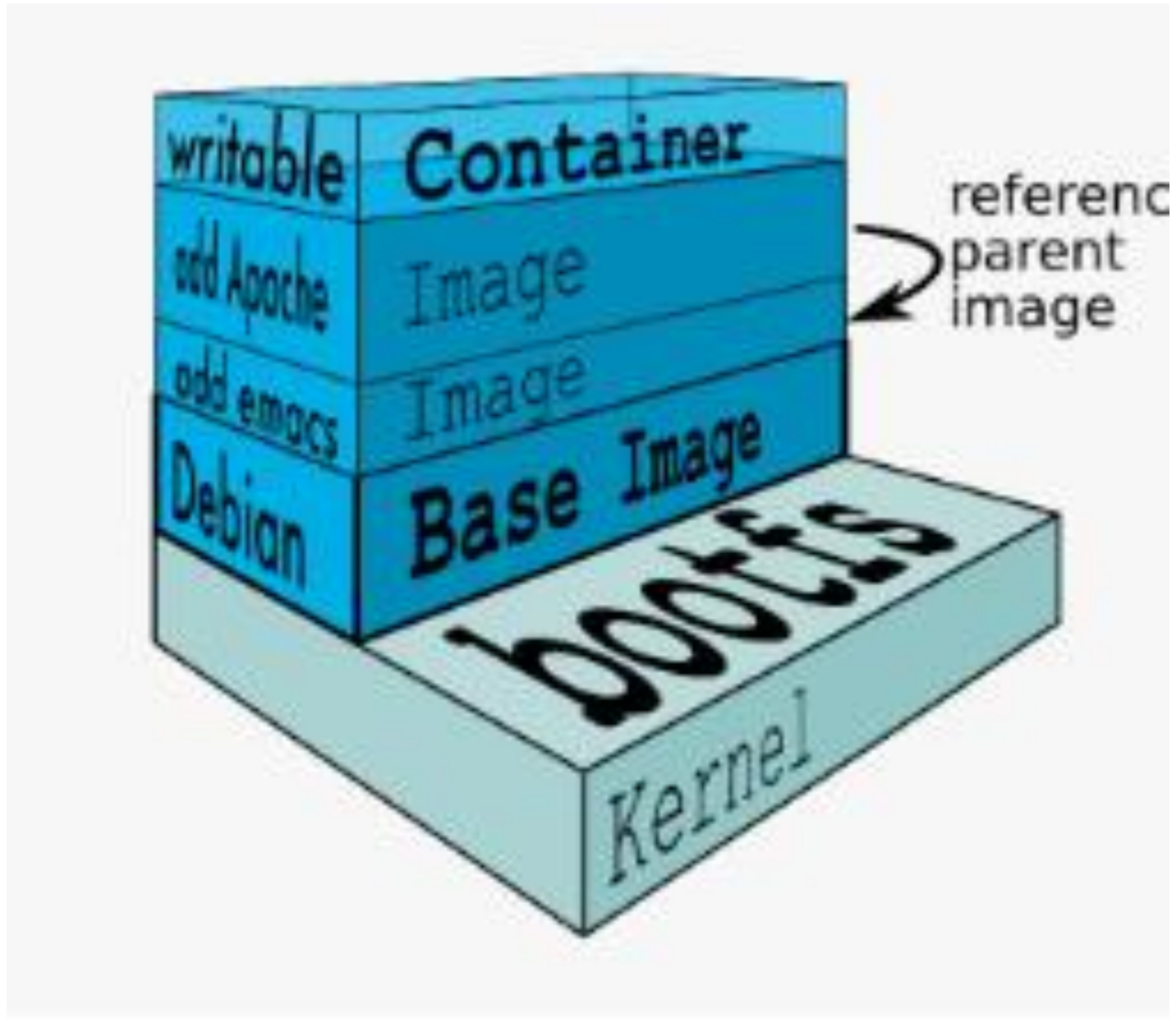
CVE-2018-8014 - High Severity Vulnerability

- Vulnerable Library - tomcat-embed-core-8.5.4.jar
- Vulnerability Details
- CVSS 3 Score Details (9.8)
- Suggested Fix

Step up your Open Source Security Game with WhiteSource here

Assignees: No one assigned
Labels: security vulnerability
Projects: None yet
Milestone: No milestone
Notifications: Subscribe

Deploy

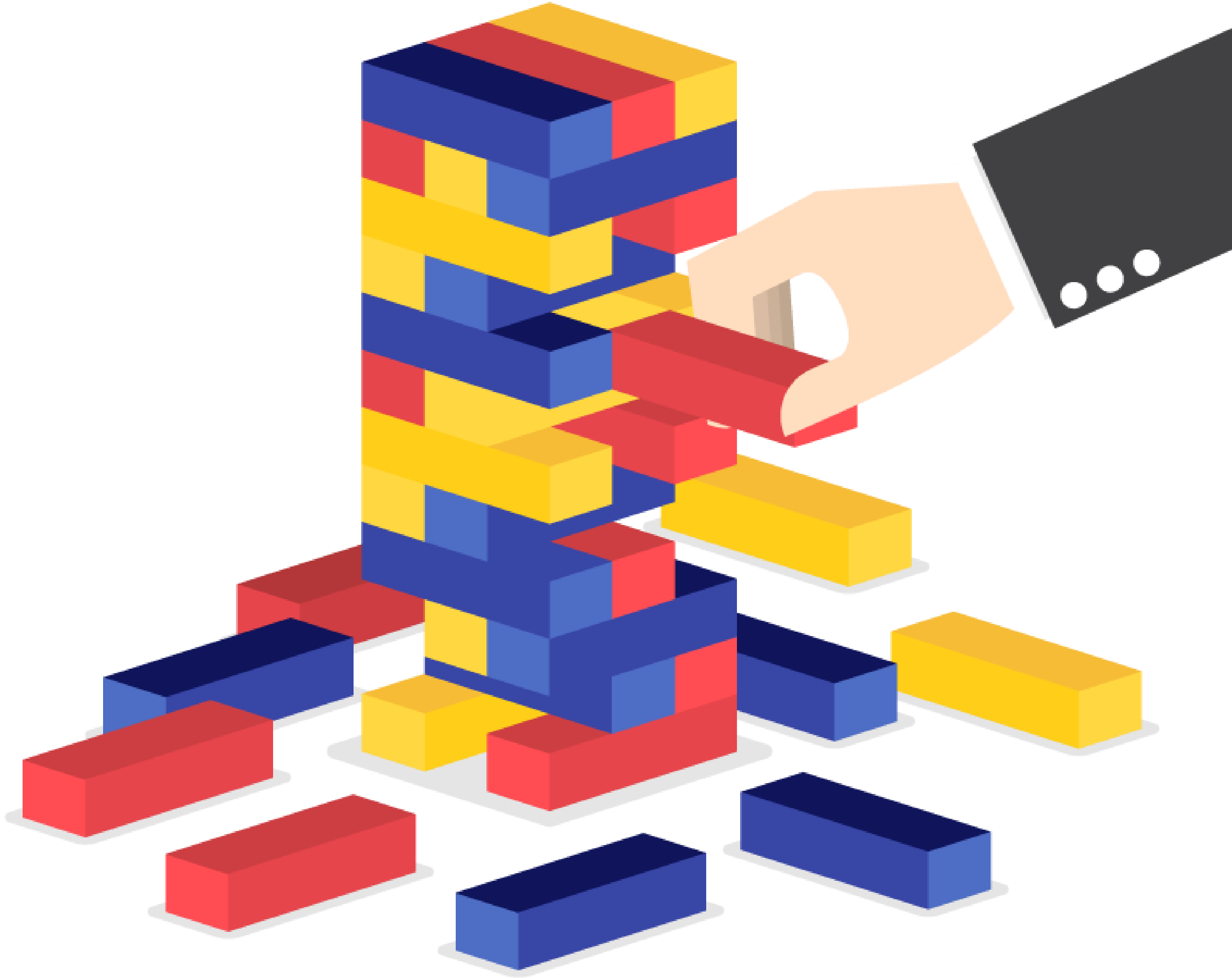

writable
Container
add Apache
Image
add emacs
Image
Debian
Base Image
bootfs
Kernel
referenc
parent
image

Maintain

New vulnerabilities are constantly being published

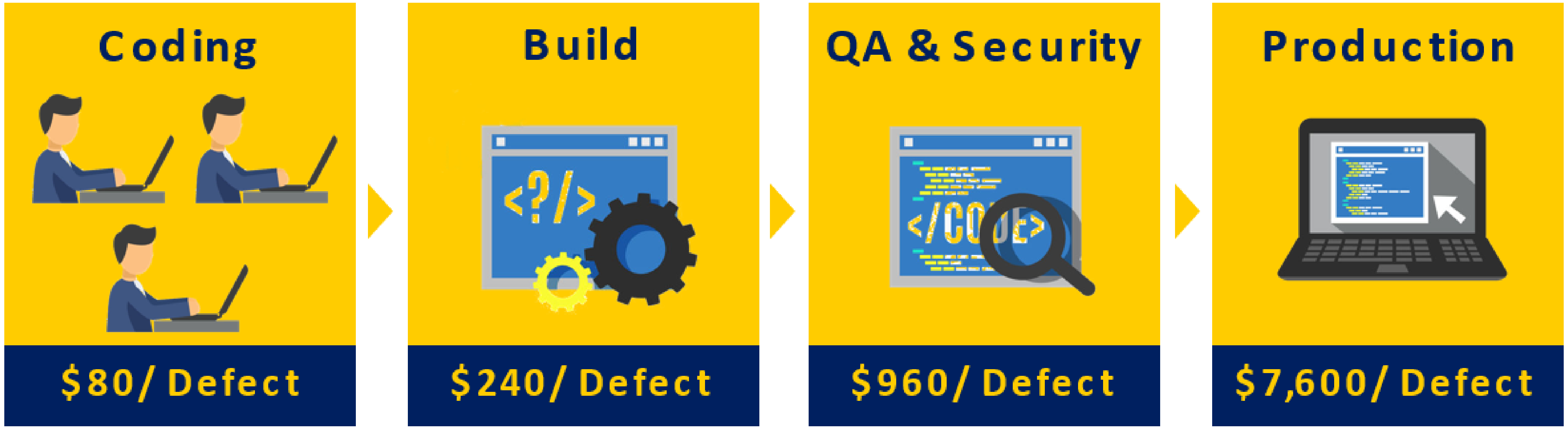
Step 3: Prioritize

- Act on early -> Shifting left
- Avoid allowing vulnerable components reach deployment



WhiteSource

Detect Issues As Early As Possible



WhiteSource

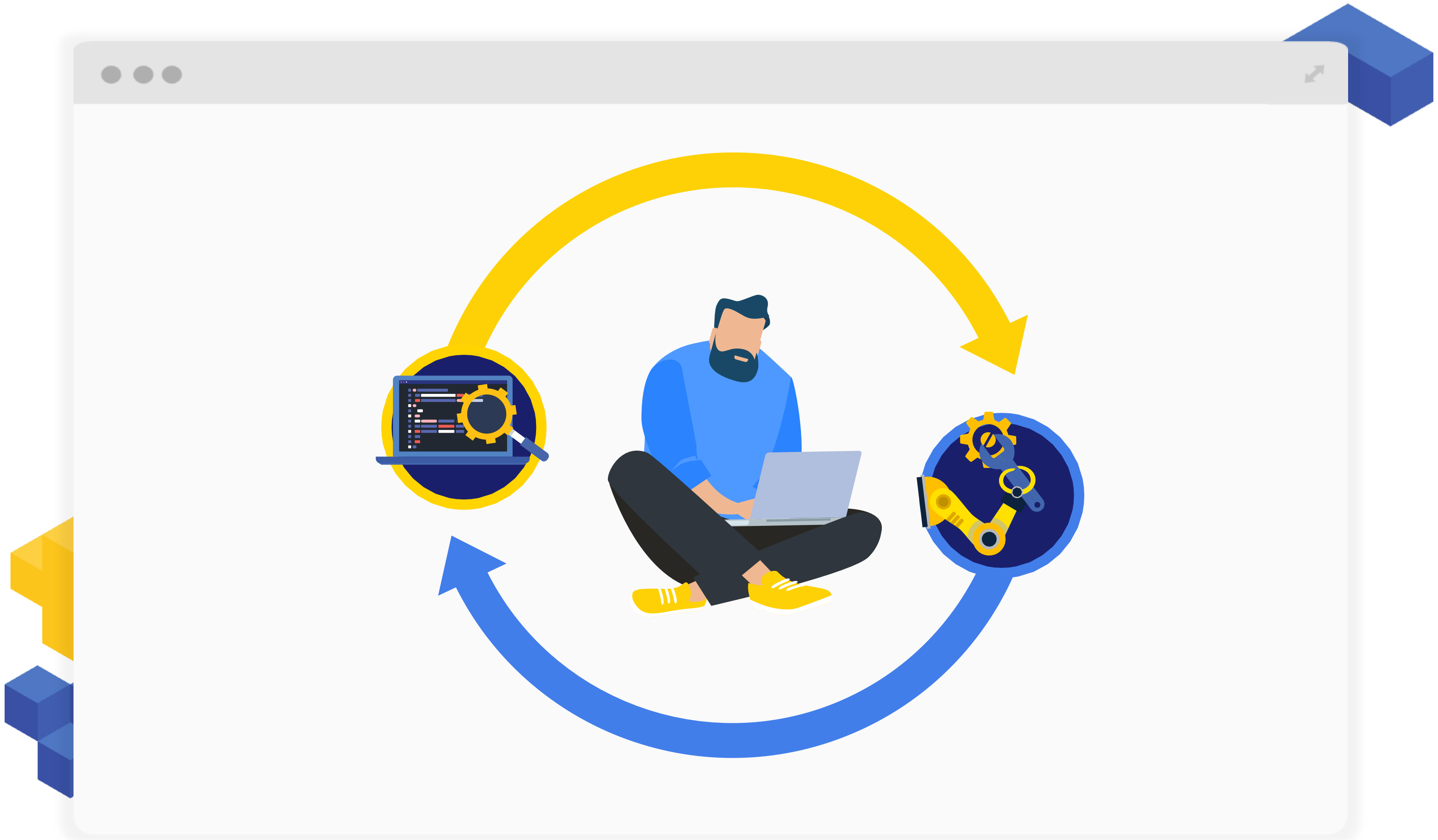
Step 4: Execution

Who's responsibility is it?

- Security team:
    - Setting policies
    - Educating developers
- Development team:
    - Execution

WhiteSource

Developers need robust tools, that fit into their workflows



WhiteSource

#1 **Educate**

On the basics of open-source security & compliance

#2 **Empower Teams**

By providing them the right tools

#3 **Enable Success**

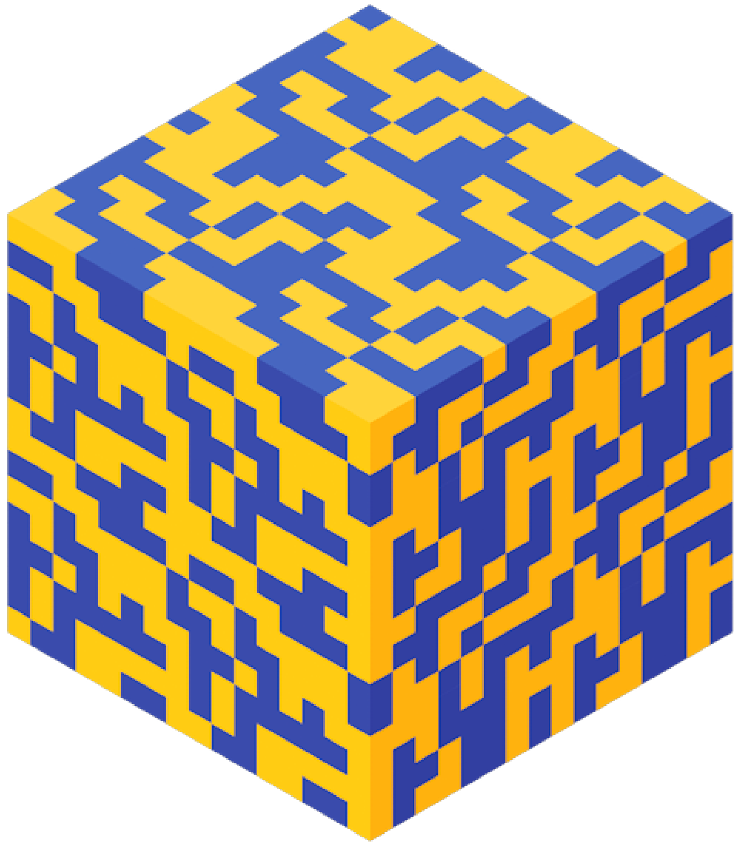
By creating a shared mission

The New York Times

Equifax Breach Caused by Lone Employee's Error, Former C.E.O. Says

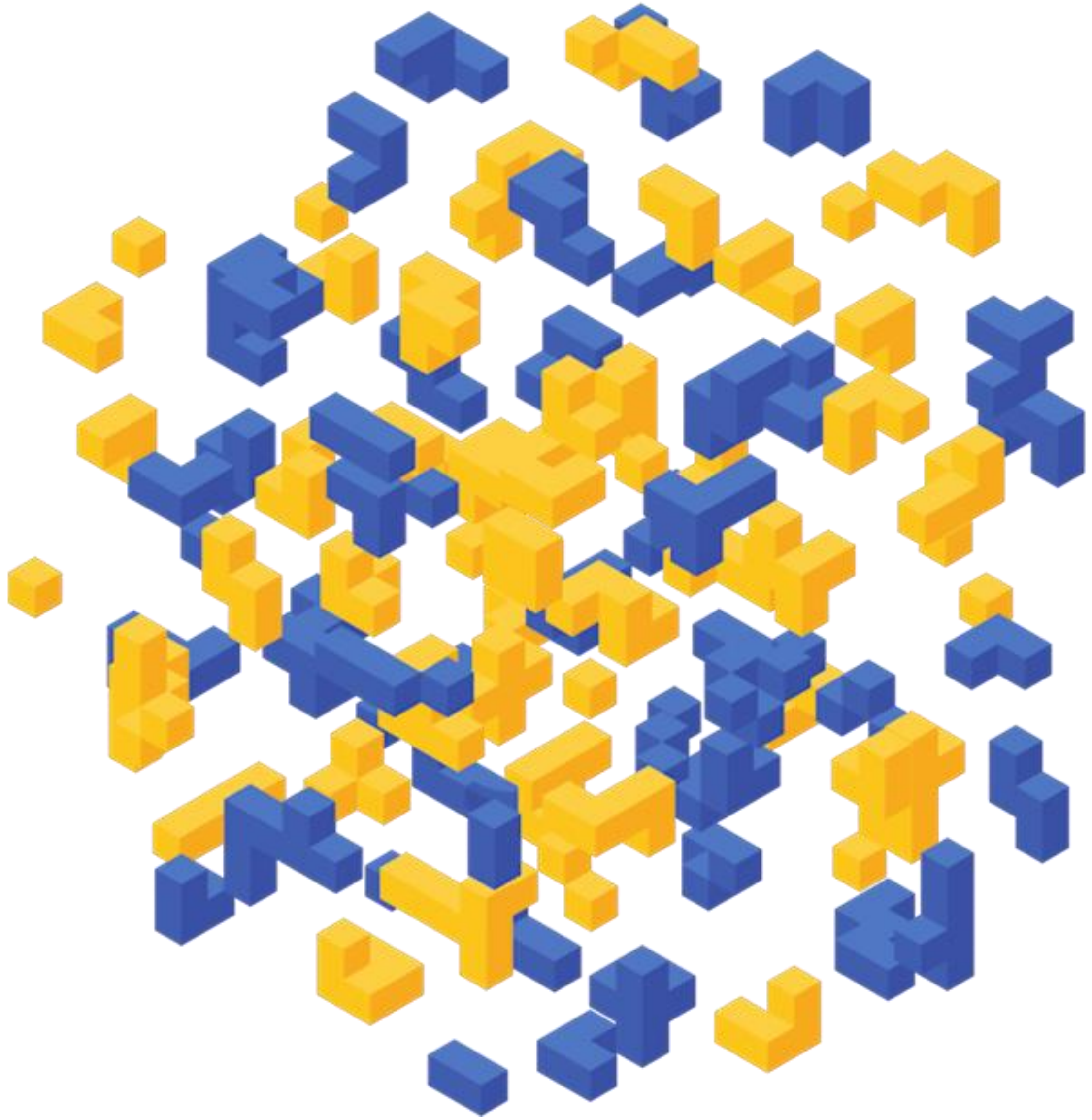
Don't Be That Guy

Q & A



Thank You!

For any questions, please contact me:
Guy.bar-gil@whitesourcesoftware.com



WhiteSource