



APPSEC EUROPE

# Using Third Party Components for Building an Application Might be More Dangerous Than You Think!

Achim D. Brucker

Fabio Massacci

Stanislav Dashevskiy



## Abstract

Today, nearly all developers rely on third party components for building an application. Thus, for most software vendors, third party components in general and Free/Libre and Open Source Software (FLOSS) in particular, are an integral part of their software supply chain.

As the security of a software offering, independently of the delivery model, depends on all components, a secure software supply chain is of utmost importance. While this is true for both proprietary and as well as FLOSS components that are consumed, FLOSS components impose particular challenges as well as provide unique opportunities. For example, on the one hand, FLOSS licenses contain usually a very strong "no warranty" clause and no service-level agreement. On the other hand, FLOSS licenses allow to modify the source code and, thus, to fix issues without depending on an (external) software vendor.

This talk is based on working on integrating securely third-party components in general, and FLOSS components in particular, into the SAP's Security Development Lifecycle (SSDL). Thus, our experience covers a wide range of products (e.g., from small mobile applications of a few thousands lines of code to large scale enterprise applications with more than a billion lines of code), a wide range of software development models (ranging from traditional waterfall to agile software engineering to DevOps), as well as a multiple deployment models (e.g., on premise products, custom hosting, or software-as-a-service).



APPSEC EUROPE

## About Us



### Achim D. Brucker

- Senior Lecturer (Software Security), University of Sheffield, UK
- Software Security Consultant
- Until 12/2015: Security Testing Strategist at SAP SE, Germany

### Stanislav Dashevskiy

- PhD Student at the University of Trento and SAP SE, France



APPSEC EUROPE

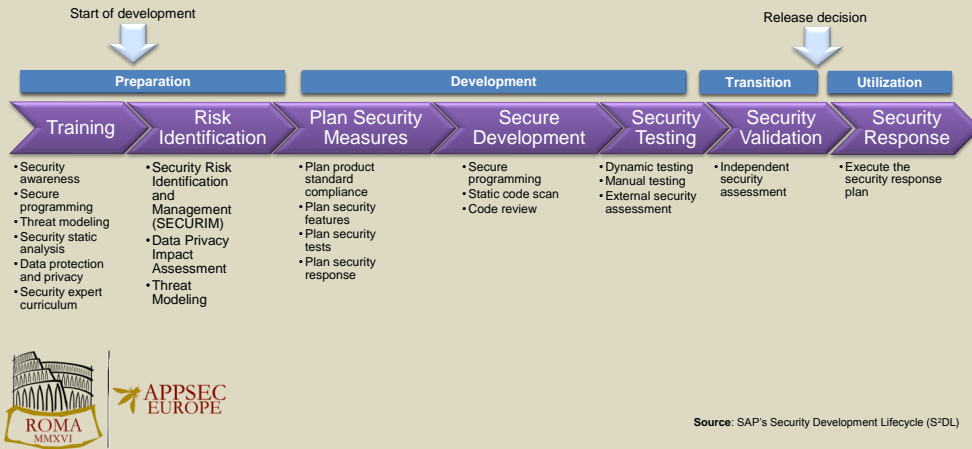
## Part I:

### Securing The Software Supply Chain or The Security Risk of Third Party Components

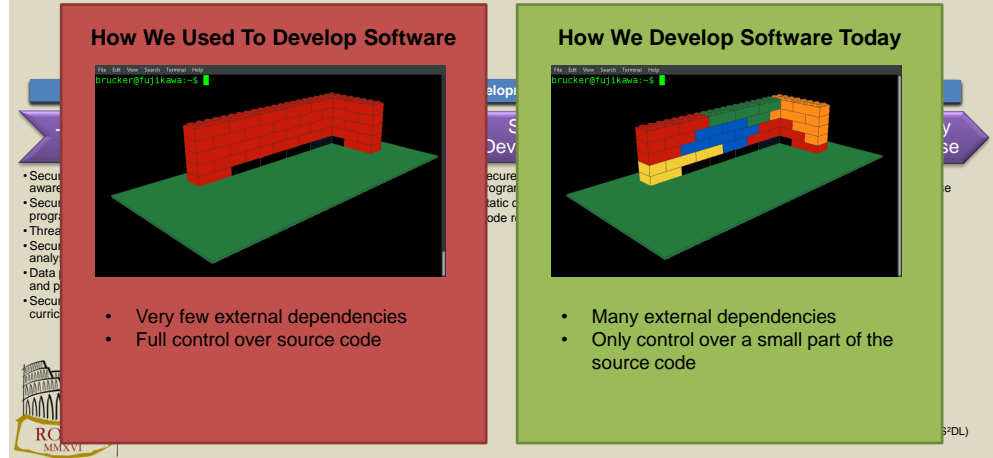


APPSEC EUROPE

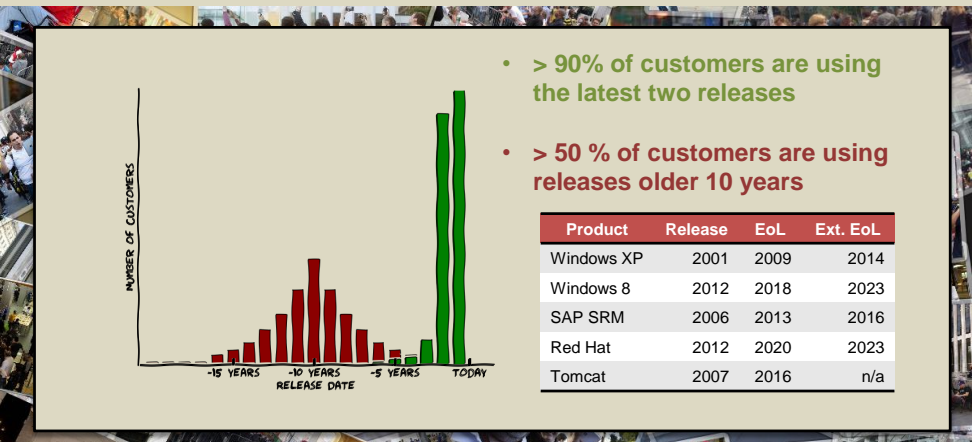
# Secure Software Development



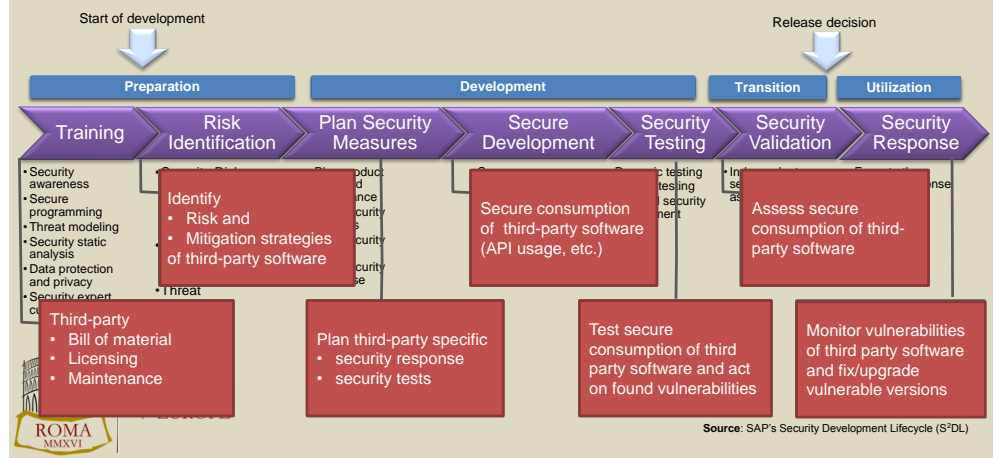
# Secure Software Development



# The Maintenance Challenge



# Secure Software Development



# Types of Third-Party Software

	Commercial Libraries Outsourcing Bespoke Software	Freeware	Free/Libre Open Source Software (FLOSS)
	<ul style="list-style-type: none"> <li>Outsourcing</li> <li>SAP HANA</li> </ul>	<ul style="list-style-type: none"> <li>Jabra Device Driver</li> <li>NVIDIA Device Driver</li> </ul>	<ul style="list-style-type: none"> <li>Apache Tomcat</li> <li>JQuery</li> </ul>
Upfront costs	High	Low	Low
Ease of access (for developers)	Hard	Medium	Easy
Modification of Source Code	Depends on contract	Impossible	Possible
Support contract	Easy	Hard	Medium



APPSEC EUROPE

# Types of Third-Party Software

	Commercial Libraries Outsourcing Bespoke Software	Freeware	Free/Libre Open Source Software (FLOSS)
	<ul style="list-style-type: none"> <li>Outsourcing</li> <li>SAP HANA</li> </ul>	<ul style="list-style-type: none"> <li>Jabra Device Driver</li> <li>NVIDIA Device Driver</li> </ul>	<ul style="list-style-type: none"> <li>Apache Tomcat</li> <li>JQuery</li> </ul>
Upfront costs	High	Low	Low
Ease of access (for developers)	Hard	Medium	Easy
Modification of Source Code	Depends on contract	Impossible	Possible
Support contract	Easy	Hard	Medium

**FLOSS is**

- Widely used in industry
- Offers possibility for self-maintenance
- Vulnerability data is available



APPSEC EUROPE

# Data Sources

## Public

- FOSS information repositories**
  - Open Hub (formerly Ohloh)
  - Core Infrastructure Initiative (CII) Census project
- Public databases of vulnerabilities**
  - National Vulnerability Database (NVD)
  - Exploit Database website (ExploitDB)
  - Open Sourced Vulnerability Database (OSVDB)
- Project data**
  - Coverity FOSS scan service
  - Source code repositories

## Internal

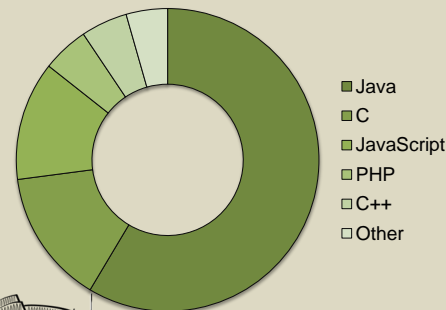
- Software inventory (e.g., Black Duck Code Center as used by SAP)



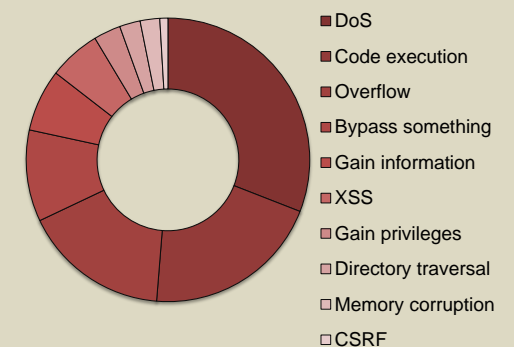
APPSEC EUROPE

# FLOSS Usage At SAP

## Programming Languages



## Vulnerabilities (CVEs)



Based on the 166 most used FOSS components (as of autumn 2015)



APPSEC EUROPE

## Part II:

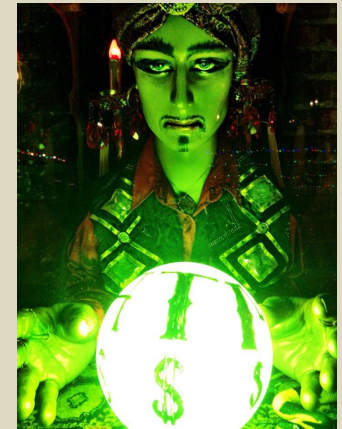
Security of Open Source Enterprise Frameworks  
or  
Assessing Risks and Planning Efforts of the Secure  
Consumption of FLOSS



APPSEC  
EUROPE

## What We Want

1. How many vulnerabilities will be published next year for component X?
2. How often do I need to ship a patch to fix a vulnerability caused by component X?

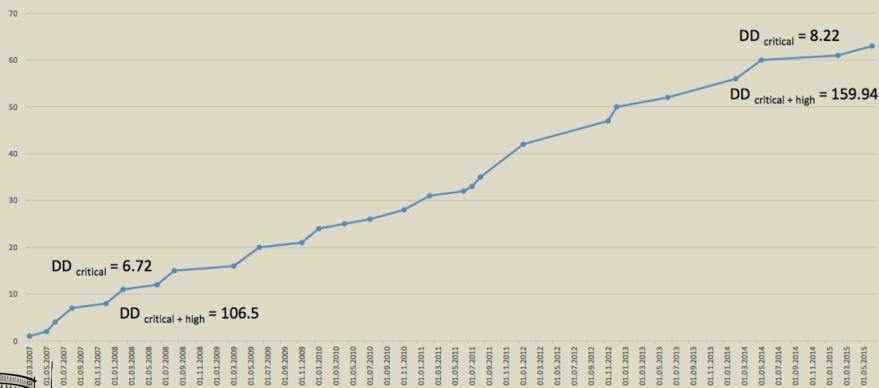


<https://www.flickr.com/photos/fimbretill/4507848067/>



APPSEC  
EUROPE

## Vulnerability Prediction?



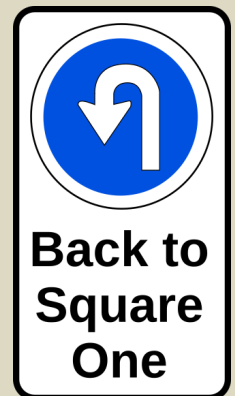
Tomcat 6.x publicly known vulnerabilities (CVEs)



APPSEC  
EUROPE

## Vulnerability Prediction: Problems

- There is not enough data
- Number of vulnerabilities depends on:
  - Age of the project
  - Number of users
- Sometimes you simply have no choice...



Back to  
Square  
One



APPSEC  
EUROPE

# Understanding Factors Is More Critical Than Predictions

- **When will a vulnerability appear in a FOSS component?**
  - We do not know
- **Can we distinguish features of projects causing "problems" for consuming software?**
  - We use maintenance effort of proprietary consumers to denote "problems"
  - Does the "security culture" of FOSS developers make a difference?
  - Does it make a difference which main language/technology is used?



APPSEC EUROPE

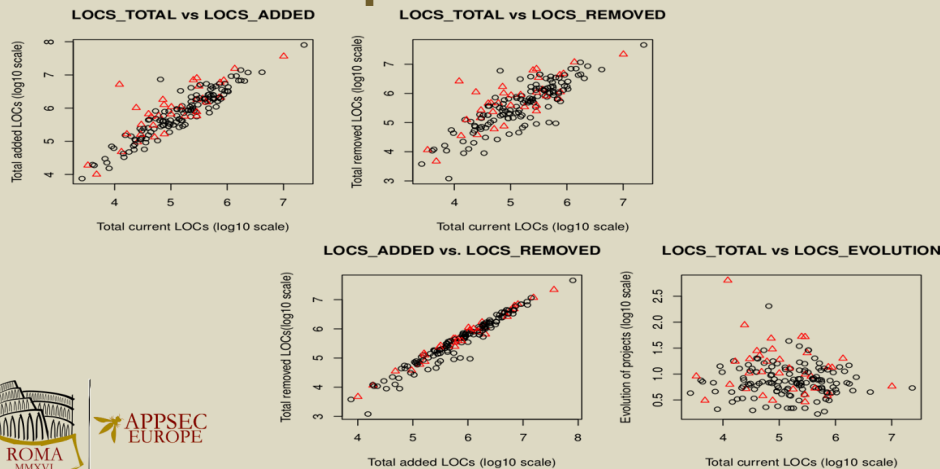
# Which Factors Are Interesting?

- **Collect all possible data, build a regression model to assess the impact of each factor**
- **Can we use all data that is available?**
  - Actual Total #LoCs of a component
  - Added Total #LoCs of a component
  - Removed Total #LoCs of a component
  - Changed Total #LoCs (added, removed, etc.)...



APPSEC EUROPE

# Relationships Between Factors



APPSEC EUROPE

# Different Maintenance Models

- **60 products are using Apache Tomcat**
  - Requires a lot of expertise to resolve security issues
  - It makes more sense to have a team of Apache Tomcat experts around
- **2 products are using a small JavaScript library**
  - This does not require any major expertise
  - However, if a company ends up using large number of products for which only the "local" expertise exists, it may be problematic



APPSEC EUROPE

## Centralized Security Maintenance

- Policy: dev. teams must select only components widely used and supported within a company
- A central team resolves vulnerabilities in all FOSS components and pushes changes to all consumers
- The security maintenance effort scales logarithmically with the number of products consuming a component

$$effort_i \propto \log(|vulns_i| * |products_i|)$$



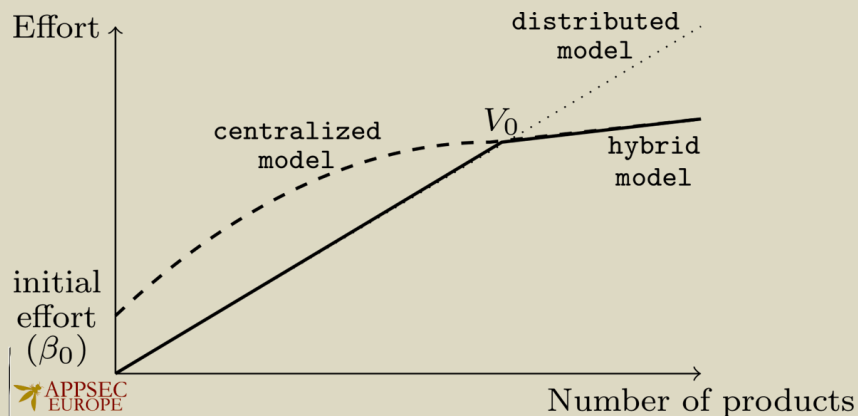
## Distributed Security Maintenance

- Policy: each dev. team is free of selecting appropriate components
- Each team has to take care of security issues individually
- While this model should decrease the effort for organizational aspects (not considered by us), it adds up for the technical part of the effort

$$effort_i \propto |vulns_i| * |products_i|$$



## Hybrid Security Maintenance



## Part III:

Practical Recommendations On  
Controlling Risk & Effort Of Using Third Party Components



## Strategies For Controlling Risks (1/2)

### Secure Software Development Life Cycle

- Maintain a detailed software inventory  
(Do not forget the dependencies)
- Actively monitor vulnerability databases
- Assess project specific risk of third-party components

### Obtaining components (or sources)

- Download from trustworthy sources  
(https, check signatures/checksums)



APPSEC  
EUROPE

## Strategies For Controlling Risks (2/2)

### Project Selection

- Prefer projects with private bug trackers
- Evidences of a healthy/working SDLC
  - Documented security fixes/patches  
(no “secret” security fixes)
  - Documented security guidelines
  - Use of security testing tools



<https://www.coreinfrastructure.org/programs>



APPSEC  
EUROPE

## Strategies For Controlling Effort

### Secure Software Development Life Cycle

- Update early and often
- Avoid own forks  
(collaborate with FLOSS community)

### Project selection

- Large user base
- Active development community
- Technologies **you** are familiar with
- Compatible maintenance strategy/life cycle
- Smaller (in terms of code size) and less complex might be better



APPSEC  
EUROPE

## Part IV:

### Conclusion



APPSEC  
EUROPE

## Conclusion

### Do not waste time with unimportant questions!

(Is FLOSS more/less secure as proprietary software)

#### Implement a secure consumption strategy:

- Risk assessment of third party consumption (at least security & licenses)
- Plan for the efforts of secure consumption
- Plan the efforts/costs for response and maintenance



## Conclusion

### Do not waste time with unimportant questions!

(Is FLOSS more/less secure as proprietary software)

#### Final advice:

- Accept that you can be hit by a “black swan” (e.g., heartbleed)
- If it happens:
  - Concentrate on understanding and fixing the issue
  - Understanding why you did not find the swan earlier should not be your first priority



## Thank you!

#### Contact:

**Achim D. Brucker**  
Department of Computer Science  
University of Sheffield  
Regent Court  
211 Portobello St.  
Sheffield S1 4DP, UK

<https://de.linkedin.com/in/adbrucker>  
<https://www.brucker.uk>  
<https://www.logicalhacking.com>  
[a.brucker@sheffield.ac.uk](mailto:a.brucker@sheffield.ac.uk)

**Stanislav Dashevskiy**  
University of Trento  
Scuola di dottorato in Informatica e Telecomunicazioni  
Via Sommarive, 14  
38123 Povo, Italy

<https://st.fbk.eu/people/profile/dashevskiy>  
[stanislav.dashevskiy@unitn.it](mailto:stanislav.dashevskiy@unitn.it)



## Bibliography

- Stanislav Dashevskiy, Achim D. Brucker, and Fabio Massacci. *On the Security Cost of Using a Free and Open Source Component in a Proprietary Product*. In International Symposium on Engineering Secure Software and Systems (ESSoS). Lecture Notes in Computer Science 9639, Springer-Verlag, 2016. <https://www.brucker.ch/bibliography/abstract/dashevskiy.ea-foss-costs-2016.en.html>
- Ruediger Bachmann and Achim D. Brucker. *Developing Secure Software: A Holistic Approach to Security Testing*. In Datenschutz und Datensicherheit (DuD), 38 (4), pages 257-261, 2014. <https://www.brucker.ch/bibliography/abstract/bachmann.ea-security-testing-2014.en.html>
- Achim D. Brucker and Uwe Sodan. *Deploying Static Application Security Testing on a Large Scale*. In GI Sicherheit 2014. Lecture Notes in Informatics, 228, pages 91-101, GI, 2014. <https://www.brucker.ch/bibliography/abstract/brucker.ea-sast-experiences-2014.en.html>
- Achim D. Brucker. *Bringing Security Testing To Development: How To Enable Developers To Act As Security Experts*, OWASP AppSecEU 2015. <https://youtu.be/LZoz4cv0MAG>  
<https://www.brucker.ch/bibliography/abstract/talk-brucker.ea-owasp-sectest-2015.en.html>

